6-8-2015

Mobile Music: a musical therapy assistance device

Alex Hildebrand
Santa Clara University

Tanner Malkoff
Santa Clara University

SANTA CLARA UNIVERSITY

Department of Electrical Engineering
Department of Mechanical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Alex Hildebrand, and Tanner Malkoff

ENTITLED

MOBILE MUSIC: A MUSICAL THERAPY ASSISTANCE DEVICE

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
ELECTRICAL ENGINEERING

BACHELOR OF SCIENCE
IN
MECHANICAL ENGINEERING

Shoba Krishnan 6/8/2015

Thesis Advisor: Shoba Krishnan Date

6/8/2015

Department Chair, Mechanical Engineering: Drazen Fabris Date

Samiha Mourad 6/8/15

Department Chair, Electrical Engineering: Samiha Mourad Date

MOBILE MUSIC: A MUSICAL THERAPY ASSISTANCE DEVICE

By

Alex Hildebrand, Department of Mechanical Engineering

Tanner Malkoff, Department of Electrical Engineering

SENIOR DESIGN PROJECT REPORT

Submitted to
the Departments of Mechanical Engineering and Electrical Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements for the
Bachelor of Science Degree in
Mechanical Engineering and Electrical Engineering in the School of Engineering
Santa Clara University

Santa Clara, California

2015

Abstract

Children suffering from Cerebral Palsy often undergo gait therapy in order to strengthen and coordinate their trunks and legs. The patients often feel unmotivated to perform their gait training because the physical movements associated with it are difficult and there is no immediate reward. Physical therapists (PT) will often play music as an incentive to get the children focused on their physical therapy, but this is inefficient and impedes the PT from analyzing and tuning the patients' gait. Mobile Music is a small device designed for automating musical therapy during gait training. In this paper, we go through several design iterations in order to create a cheap, ergonomic device that will sense motion and use musical stimulation in order to encourage active participation in gait training. While we had some difficulties implementing the motion sensing algorithm in the final device, testing done in the prototype phase showed promising results in accurately detecting participation in physical therapy. We have determined that the problems associated with these are likely due to timing issues in the microcontroller unit due to multiple reasons and the lack of feedback to the device of the audio streaming state. With the addition of a mobile application and some slight changes to the code implementation of the motion sensing algorithm, we believe that these problems can be fixed. Because of the improvement in media player control that a mobile app enables, a newer, lower powered system configuration can be implemented that reduces audio output noise as well as reduce power consumption. These improvements combined with the low cost and simple interface could make Mobile Music into a very viable product with the potential to help children with movement disorders improve their gaits.

Acknowledgements

We would like to thank the following people and entities for their help during the process of this design project:

Santa Clara University, School of Engineering
The Avalon Academy
Dr. Shoba Krishnan
Dr. Timothy Hight
Dr. Radhika Grover
Anup Navare
Sam Pollock
Ethan Head
SCU Maker Lab

Table of Contents

Page

Abstract ..... **iii**

Acknowledgements ..... **iv**

1 Introduction **1**

1.1 Research 1

1.2 Customer Overview 4

1.3 Project Motivation 4

2 System Overview **6**

2.1 System Functions 6

2.2 Design Specifications 6

2.3 Similar Products 8

2.4 Initial Design Ideas 9

2.5 System Communications 12

2.6 Calibration Algorithm 13

2.7 Motion Sensing Algorithm 15

3 Prototype **18**

3.1 Hardware 18

3.2 Software 24

3.3 Testing and Feedback 26

4 Final Build **32**

4.1 Hardware 32

4.2 Software 40

4.3 Current Testing and Possible Changes 44

5 Engineering Standards **47**

5.1 Manufacturability Analysis 47

5.2 Economic Analysis 49

5.3 Intellectual Property Analysis 50

5.4 Ethics Analysis 54

5.5 Social Analysis 55

6 Final Thoughts and Future Developments **56**

6.1 New System Configuration 56

6.2 Mobile App Integration 57

6.3 Mechanical Design Updates 58

6.4 Conclusions 58

References **60**

Appendices **61**

Prototype BOM ..... A

Prototype Schematic ..... B

Prototype Pictures C

Prototype Code ..... D

Final Build BOM ..... E

Final Build Schematic ..... F

Final Build Layout ..... G

Final Build Drawings ..... H

Final Build Code I

Senior Design Conference Presentation ..... J

Chapter 1 – Introduction

1.1 – Research

For our research we wanted to look into Cerebral Palsy and the various therapy methods. We also wanted to do research about music therapy, and how music has been used to help people with neurological disorders. Lastly, our research needed to include a look into other similar products in order to see if there is a viable market for this product.

1.1.1 – Cerebral Palsy

Cerebral Palsy, or CP, is a general term under which numerous neurological disorders are placed. The commonality between these disorders is that they develop in infancy or early childhood, they do not worsen over time, and they leave their sufferers with limited mobility. CP is the most common motor disability in children, with estimates ranging from 1.5 to 4 cases per 1000 births (CDC, Data and Statistics for CP, 2008). The extent of a CP patient's disability can vary greatly, from people who are able walk with a slight limp to people who have no ability move their arms or legs. 30.6% of children with CP have very limited or no walking abilities (Autism and Developmental Disabilities Monitoring CP Network, 2008). Because of these mobility issues, people with CP often undergo rigorous physical therapy in order to gain mobility of their extremities. Therapy, however, can be tedious and boring, and many children with CP feel no motivation to perform their therapy. Our goal is to create a device that will utilize music as a reward system to help motivate these children suffering from CP.

Our device will be used to help adolescents with their gait therapy. Gait therapy is used to help the patients develop muscle and control in the legs. This type of therapy is performed in specialized walkers, many of which support the patient's body weight while still allowing free movement of the legs. It is to these walkers that the device will be mounted. One consideration that must be made is how to mount the device on the various styles of tubing used to build the structure of these walkers.

1.1.2 – Music Therapy

There have been multiple studies on the benefits of music during physical therapy. It has been seen that Rhythmic Auditory Stimulation has been effective in physical therapies for both gait and arm function. The article of the study published in the Journal of Music Therapy is titled "Effect of Rhythmic Auditory Stimulation on Gait Performance in Children with Spastic Cerebral Palsy," and was written by Dr. Eunmi Emily Kwak. Results from this study show significant improvements in stride length, velocity and symmetry (Kwak, 198). RAS works by attempting to synchronize the body's walking movements rhythmically with the music. Rather than random music and melody in traditional music therapy, RAS targets the rhythm of an individual's gait, which can be a complicated motion. The tempo of the music is then raised or reduced for each patient, in an attempt to control his or her gait. Over the three week experiment, the therapist-guided training (TGT) group increased 5% in cadence, 29.48% in stride length, 36.49% in velocity, and 16.97% in symmetry, beating the control group in every category (206-208). Results of this study can be found in Figure 1.1.

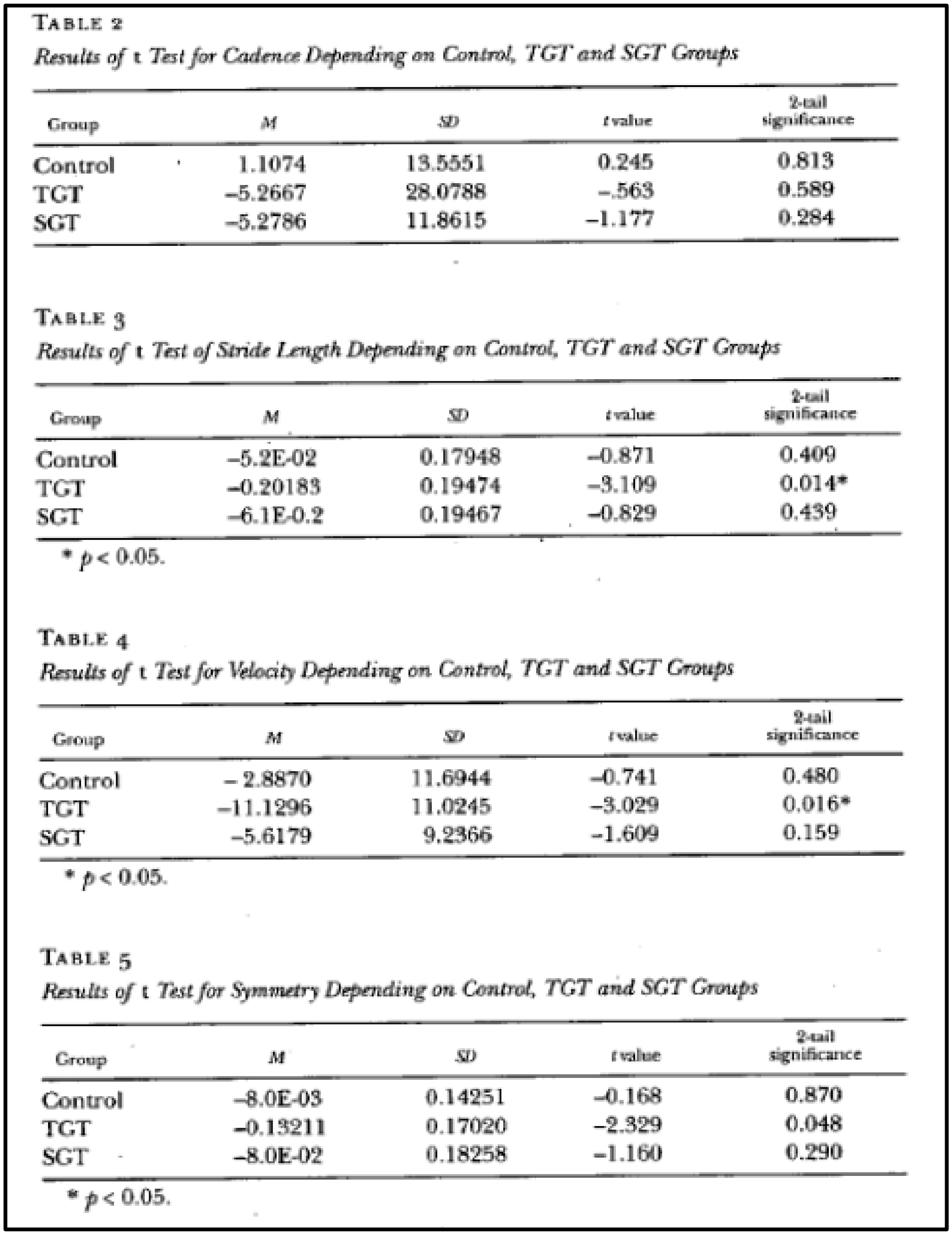


Table 2

Results of t *Test for Cadence Depending on Control, TGT and SGT Groups*

Group	*M*	*SD*	*t* value	2-tail significance
Control	1.1074	13.5551	0.245	0.813
TGT	–5.2667	28.0788	–.563	0.589
SGT	–5.2786	11.8615	–1.177	0.284

Table 3

Results of t *Test of Stride Length Depending on Control, TGT and SGT Groups*

Group	*M*	*SD*	*t* value	2-tail significance
Control	–5.2E-02	0.17948	–0.871	0.409
TGT	–0.20183	0.19474	–3.109	0.014*
SGT	–6.1E-0.2	0.19467	–0.829	0.439

* $p < 0.05$.

Table 4

Results of t *Test for Velocity Depending on Control, TGT and SGT Groups*

Group	*M*	*SD*	*t* value	2-tail significance
Control	– 2.8870	11.6944	–0.741	0.480
TGT	–11.1296	11.0245	–3.029	0.016*
SGT	–5.6179	9.2366	–1.609	0.159

* $p < 0.05$.

Table 5

Results of t *Test for Symmetry Depending on Control, TGT and SGT Groups*

Group	*M*	*SD*	*t* value	2-tail significance
Control	–8.0E-03	0.14251	–0.168	0.870
TGT	–0.13211	0.17020	–2.329	0.048
SGT	–8.0E-02	0.18258	–1.160	0.290

* $p < 0.05$.

Figure 1.1 - RAS Gait Therapy Results (reproduced without permission)

Another device which we found was D-Jogger. D-Jogger was designed as a running device, but using music tempo to help runners keep pace. The creators of this device have begun to look into its use on patients with Parkinson’s disease (smcnetwork.org).

1.2 – Customer Overview

Our customer for this project is The Avalon Academy in Burlingame, CA. They are a school which caters to the needs of children with CP and similar neurological diseases. To do this they use a teaching program called MISTS, which stands for Motion Integrated Special Teaching System, developed by their own Kinga Czegini. This system allows the school to provide their students with not only a standard education, but also time during the day to perform the movement therapy that they need. The system also takes into consideration students with speech or auditory difficulty as well

The Avalon Academy has been a wonderful resource for us about these patients and the disease. They also provide us a great test facility, because we are able to utilize the device on different patients and walkers to see what works and what does not work. Our device will be used to help the students with gait therapy, which in turn helps the patients develop muscle and control in the legs. This type of therapy is performed in specialized walkers which support the patient's body weight but still allows free movement of the legs. It is to these walkers that the device will be mounted.

The children at The Avalon Academy vary in the level of their disability. Some of the children are able to perform their gait therapy at a strong pace, while others have difficulty taking even the first steps. This was something that we needed to consider for the project, as we want our device to be universally effective for any patient who desires to use it.

We hope that upon completion of the project and the device, that The Avalon Academy will be able to implement it into their daily routines. If they are able to find success with the product, we hope that it could be taken further, to children and adults all over the world who need motivation to help them with their gait training.

1.3 – Project Motivation

The original motivation behind the project came from three things we wanted to accomplish with our senior design project. First, we wanted to create a physical device, which could easily be used by anyone. Second, we wanted to be able to help the community and those less fortunate. And lastly, we wanted to create a project that involved music. After finding The Avalon Academy, one initial idea was to create a musical instrument for these children, but upon seeing their movement abilities it was deemed something that they would probably be unable to use, unless the instrument was customized for each child.

From here the motivation for this project came from The Avalon Academy. The physical therapists at the school, showed us gait therapy, and had noticed that their students were more motivated to do their physical therapy (i.e. walking) when there was music playing in the room. They realized that they could use the music as a reward for actively performing physical therapy tasks, and asked us to develop a device that would automate when the music would play. Our motivation for this project is to help the community and these children in any way we can. We had a clear goal to work towards with the size and scope of the project already given.

Chapter 2 – System Overview

2.1 – System Functions

The main goal of this project is to design and build a device that will motivate a patient of CP to perform their walking physical therapy by rewarding them with music while they are actively walking. Figure 2.1 shows how the device will fit into its larger system. When the device detects activity, it will send a command to the media player telling it to begin streaming audio. The audio stream will then go from the media player, to the Mobile Music device, and out of a standard phone jack into a speaker or headphone. This music will encourage the participant to continue performing their physical therapy and therefore continue to hear the music. When the device detects inactivity, it will send a command to the media player to pause the audio stream. This will halt the music so that the participant will want to return to performing their physical therapy, so that the music will turn back on.

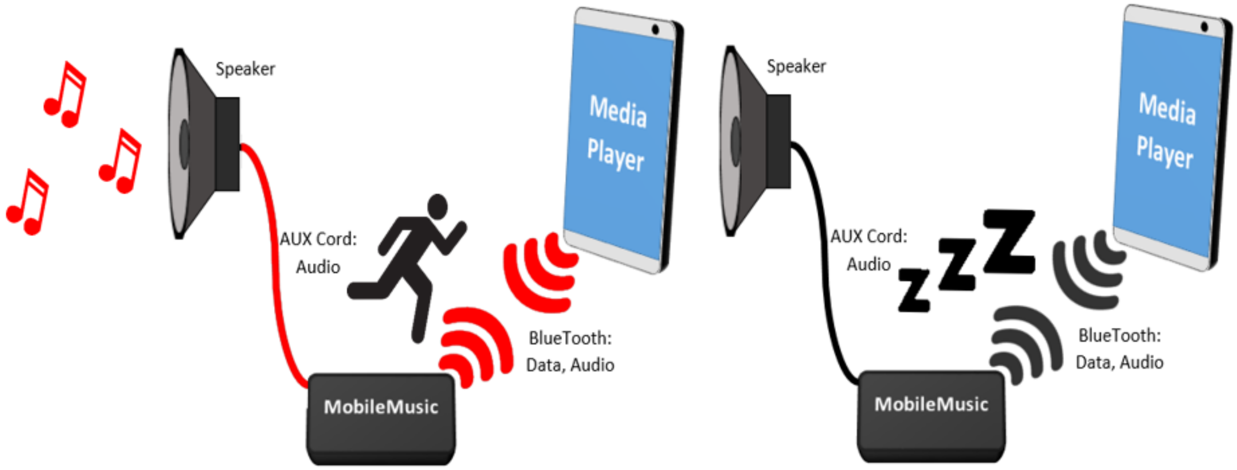


Figure 2.1 - System Block Diagram for Activity (left) and Inactivity (right)

2.2 – Design Specifications

Our device will be different from others we have researched in that it is the only thing on the market designed specifically for music therapy. The device will need to be small and it will need to easily mount on any patient's walker. Mobile Music will also need to be able to link with each patients preferred music player, regardless of manufacturer. The device settings, such as gait speed, should be able to be customized for each patient by the therapist to account for different preferences in music selection and different walking styles and levels of mobility.

Table 2.1 – Project Design Specification

ELEMENTS/		**PARAMETERS**	
REQUIREMENTS	UNITS	DATUM	**TARGET - RANGE**
PERFORMANCE			
Accuracy of motion detection	%	100	95-100
Speed of music play/pause	sec	1-2, longer if device goes to sleep mode	<1
Number of individual patients which are affected	number	1 therapist can control music for one patient	Each device controls music for each individual patient
Range of music travel	Ft	20-50	2-3 for each device
Performance tracking		No	Yes
Battery Life	Hours	7-10	3-5
Connectivity		N/A	Bluetooth, WiFi
Connectivity Range	ft	5-6	20-30
Data Record	kb	0	10
Product Cost	$	50	100
Manufacturing Processes	Hours	1	2
Size	Cu.in	50	<10
Weight	Oz.	16	<10
Testing			
Shock	G	5	20-30
Water- resistance		No	Yes
Installation Time	sec	60	30
Tubing Sizes Fit	In.	N/A	¾-1-1/2

2.3 – Similar Products

During our research we were able to find three products that functioned similarly to our idea. These were the Yamaha BODiBEAT and the Philips Activa. Both of these products are designed to help runners with their exercise.

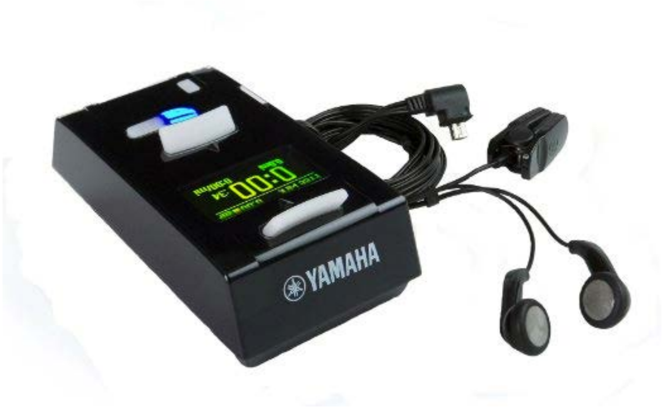


Figure 2.2 - Yamaha BODiBEAT (reproduced without permission)

The Yamaha BODiBEAT is designed for all types of work out, and the idea is that the device can select music for you to listen to, based on the intensity of your workout. To measure this intensity, the BODiBEAT uses an ear clip heart beat monitor and accelerometers within the device to decide what tempo of music should be played.

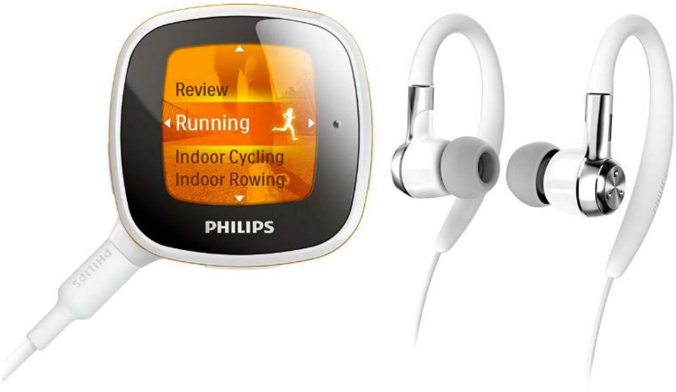


Figure 2.3 – Philips Activa (reproduced without permission)

The Philips Activa works similarly to the BODiBEAT in that its purpose is to select music base on the intensity of your workout. However, the Activa is meant specifically for running, and uses GPS to monitor the speed and intensity of the run.

During our research we were unable to find anything related to Cerebral Palsy or gait therapy the utilized music. Because of this we believe that there could be a market and a need for our type of device. People who need to perform this type of therapy, and feel unmotivated, all over the world could benefit from a simple device like this.

2.4 – Initial Design Ideas

As we began our initial design, we knew that we needed a relatively small device that could accurately detect the motion of the walker. This led to the testing of various methods available to accomplish these tasks.

2.4.1 – Gait Detection

For gait detection we considered and tested two methods. The first method was a Pololu wheel encoder. This component uses photo resistors to sense calibrated notches on the attached wheel. By counting the notches, the program is able to get a sense if the wheel is moving, and how far it has moved. The second component tested was an Accelerometer. This device measures accelerations in 3-axis and allows the programmer to measure various accelerations experienced by the device. Table 2.2 shows the various pros and cons we found with each gait detection method.

Table 2.2 – Detection Methods

Detection Method	Pros	Cons
Accelerometer	-Low Cost -Internal Device -Easily Configurable -Accurate	-Requires Calibration -Noisy -Measures Acceleration
Wheel Encoder	-Measures Distance -Reliable -Accurate	-Expensive -Externally Mounted - Presents Tripping Hazard

During initial testing, we discovered that both devices had a few negative traits. The biggest problem with the accelerometer was that it would require calibration once mounted on the device. It is also a very sensitive sensor, so the data received from it can be a bit noisy. The problems with the wheel encoder were that it was an expensive assembly of hardware pieces, and it needed to be mounted separately from the main Mobile Music device. This would present a tripping hazard to the patient and those around them.

It is for these reasons that we ended up going with the accelerometer. Because of this decision, the programming became very important, and mechanically we were able to build a smaller and simpler device.

2.4.2 – Mechanical Structure

Mechanically, the main goal of the structure is to hold and protect the necessary electrical components. Additionally, the structure acts as a user interface, holding the buttons, lights, and component jacks in place. Initial designs for the structure focused on protection and mounting of the device, as electrical components had not been chosen and the PCB had not been designed.

2.4.3 – Mounting Options

Because of the requirement for an adjustable mount, various designs were initially created. A spring loaded clip was considered and tested, but it was found that the spring fatigued far too quickly, and the clip was useless after a few days of use. Other design ideas included a "halo" style system with screws to grip the tube shown in the sketch of Table 2.3. These initial ideas eventually led us to the final sliding design that will be covered later.

Table 2.3 – Mounting Styles

Mount Type	Pros	Cons
Clip On	• Universally Mountable • Quick Installation • Off the Shelf	• Unstable • Requires Adhesive • Quickly Weakens
Adjustable	• Universally Mountable • Stable • Durable • Off the Shelf	• Slower Installation • Requires a Tool • Can Damage Paint

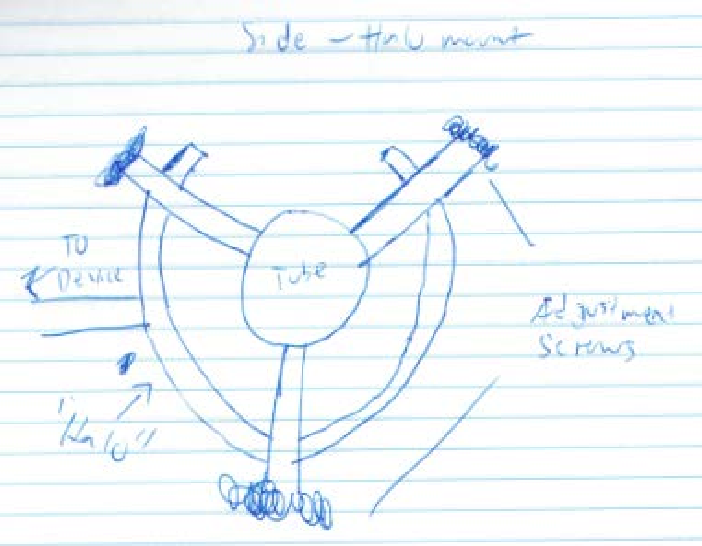


Figure 2.4 – Halo Style Mount Sketch

2.5 – System Communications

As an embedded electronic device, Mobile Music requires the transfer of data between multiple transmitters and receivers. In order to interface these different components, different types of communications are used, as shown in Figure 2.5 These specific communications were each chosen for different advantages with speed and number of pins required being the two main tradeoffs.

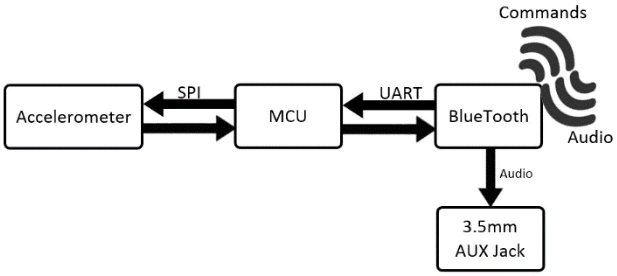


Figure 2.5 - Block Diagram Displaying Communications Used between Components

The digital communication protocol SPI is used between the MCU and the accelerometer in order to transfer acceleration data. SPI is a synchronous, master-slave protocol that requires 4 pins: a Master In Slave Out (MISO) pin, a Master Out Slave In (MOSI) pin, a clock (SCLK) pin, and a chip select (CS) pin. While we had the option to use the I2C protocol which only requires two pins, we decided to use SPI because we can operate at a higher clock frequency. SPI can operate at 5MHz as opposed to a maximum of 400kHz with I2C, so we can transmit more data, faster.

The digital communication protocol UART is used to send commands and configure the BlueTooth module. UART is a common and well supported asynchronous, serial communication that can operate with a baud rate of 115,200 symbols per second. Since the command frequency will be much slower than the collection of acceleration data, the speed of the communication protocol is sacrificed for the benefit of only requiring two pins: a Receive (RX) pin and a Transmit (TX) pin.

The wireless communication protocol BlueTooth is used to connect Mobile Music to the media player, stream audio, as well as to receive and interpret commands from the MCU. The specific profiles used in our device are the Advanced Audio Distribution Profile

(A2DP) for streaming digital audio signals, the Audio/Video Remote Control Profile (AVRCP) for basic media player control, and the Serial Port Profile (SPP) for sending and receiving data. These profiles can run simultaneously which means that sending data over SPP will not affect the A2DP music stream. BlueTooth also has a range of roughly 10m which is greater than our target specification of 20ft.

The final communication is the analog audio signal that transmits music from the BlueTooth module to the speaker or headphone. Because this is an analog signal, it is very susceptible to noise that can distort the intended sounds. In order to help eliminate noise coupled into the transmission lines, differential audio signals can be used for both the left and right channels. By ensuring that the differential audio signal pair is kept as close as possible in layout, we can assume that the noise coupled into both of them is equal. This means that when we subtract them to recover the full audio signal, we effectively eliminate any common noise leaving us with a much cleaner output signal. This method, however, requires and additional differential to single-ended output audio amplifier, since standard 3.5mm phone jacks use single-ended channel outputs.

2.6 – Calibration Algorithm

The calibration algorithm is used to ensure that no matter how the device is oriented, the device will only read the acceleration magnitude orthogonal to the acceleration vector of gravity. This also helps to eliminate any accelerations upwards and downwards due to bouncing. Before calibration is started, the device is on standby for one second so that any acceleration from the push of the button is not included in the calibration data. The device then proceeds to collect 200 samples for each axis and take the average of them to help filter out any high frequency noise, giving us x_avg, y_avg, and z_avg. Assuming that there was no or little external acceleration during calibration time, the resulting averages should represent the vector of gravity. From this reading, we can mathematically determine the rotational offset of our accelerometers recorded axes to the real world axes with the vector of gravity acting as the negative z axis, as shown in Figure 2.6.

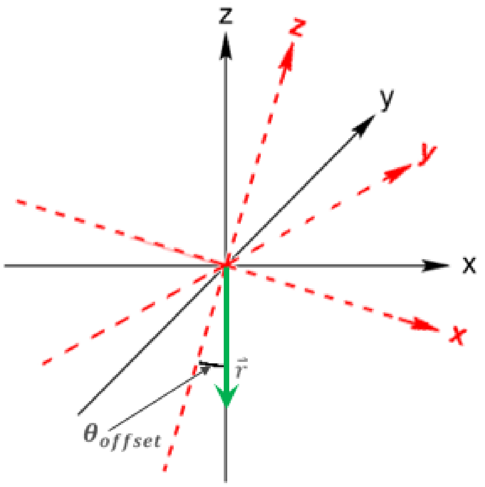


Figure 2.6 - Rotational Offset between Measured Axes (red) and Real Axes (black)

This angular offset can be determined by converting the measured gravity vector from Cartesian coordinates into spherical coordinates. We can then determine the parameter θ_{offset} between the negative z axis of the accelerometer and the measured vector using the following equations:

$$\bar{r} = \sqrt{\bar{x}^2 + \bar{y}^2 + \bar{z}^2} \qquad \text{(eq. 1)}$$

$$\theta_{offset} = \arccos\left(\frac{\bar{z}}{\bar{r}}\right) \qquad \text{(eq. 2)}$$

where $\bar{x}$, $\bar{y}$, and $\bar{z}$ represent the average acceleration values measured for each respective axis and the spherical coordinate parameter $\bar{r}$ represents the radius vector of these Cartesian coordinates.

Having gone through calibration and determined the rotational offset, we will be able to determine the calibrated acceleration in the XY-Plane. To do this, we must first know how to equate the magnitude in the XY-Plane in spherical coordinates. This can be equated by the equation $\left|\overrightarrow{XY}_{plane}\right| = r * \sin(\theta)$. All that we need to do now is replace θ with the calibrated angular offset subtracted from the measured angular offset, given by the equation $\left|\overrightarrow{XY}_{plane\_cal}\right| = r * \sin\left(\arccos\left(\frac{z}{r}\right) - \theta_{offset}\right)$. With this equation and a successful calibration, we are able to measure only acceleration values orthogonal to the vector of gravity, regardless of mounting orientation.

2.7 – Motion Sensing Algorithm

Once the data measurements have been adjusted for the calibration, we can begin the motion sensing algorithm. The main function of this algorithm is to interpret the calibrated acceleration data and control whether the device will play or pause the audio stream. Due to the cyclical accelerations of a walking motion, there is often a lot of jitter in the measurements where the values can fluctuate rapidly. This jitter is further enhanced by jerky motion patterns that are common to people suffering from Cerebral Palsy. In order to help filter out jitter, the motion sensing algorithm incorporates two methods of noise reduction: a hysteresis and a timing delay.

A hysteresis is the dependence of an output based on both the current and past inputs. This effect is utilized in our algorithm in the form of two thresholds, as shown in Figure 2.7. The upper and lower thresholds are represented in blue and yellow, respectively. The red and green areas represent inactivity and activity, respectively. Assuming that the previous acceleration value was below the lower threshold, an acceleration value must be greater than or equal to the upper threshold in order to trip the hysteresis and tell the device that activity is detected, represented by the transition from red to green in Figure 2.7. Once this threshold has been tripped, the acceleration value must go below the lower threshold to trip the hysteresis again and tell the device that inactivity is detected. The reason that the hysteresis does not trip when this occurs in Figure 2.7 is due to the inactivity timing delay.

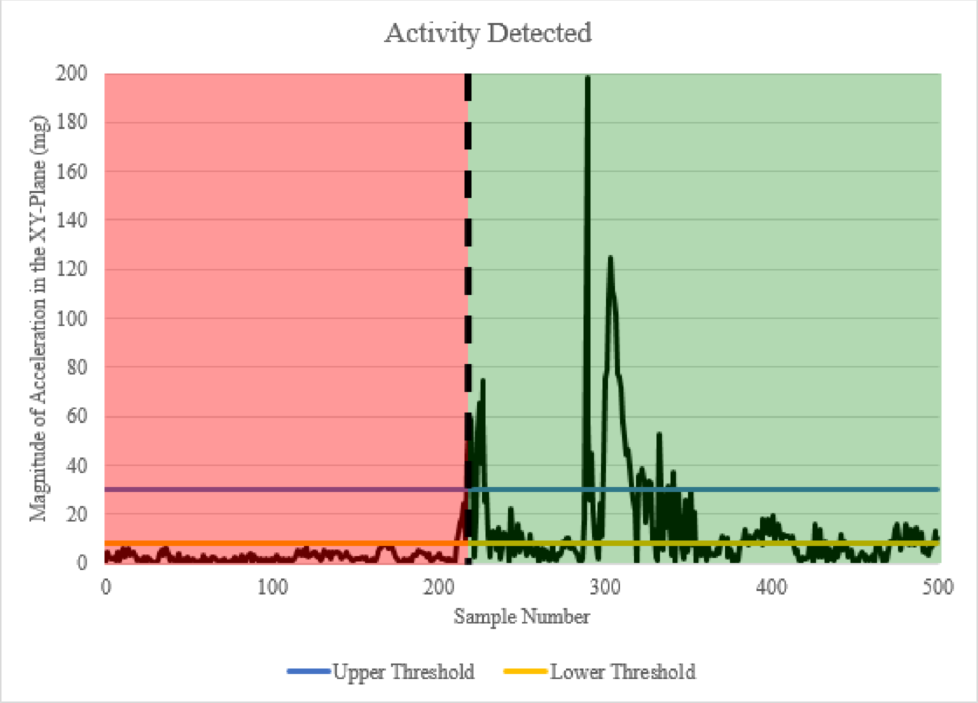


Figure 2.7 - Motion Sensing Algorithm Detecting Activity

The timing delay is used to further prevent jitter by filtering out the inherently high fluctuations in acceleration values associated with a walking movement. An example of this can easily be seen in the green area of Figure 2.8, which represents activity. The yellow area represents the inactivity delay during which time the device is still considered in the “active” state, but a watchdog timer is counting down. If an acceleration value is not greater than or equal to the lower threshold before the watchdog times out, the device will be considered in the “inactive” state, represented by the red region of Figure 2.8. If an acceleration value is greater than or equal to the lower threshold before the watchdog times out, the timer turns off and will not be reset until the acceleration value drops below the lower threshold again. It is important to note that during the green activity region the watchdog timer was being set and reset numerous times, but for the sake of best showing how the timing delay functions, only the last timing delay (during which the watchdog times out) is shown in Figure 2.8.

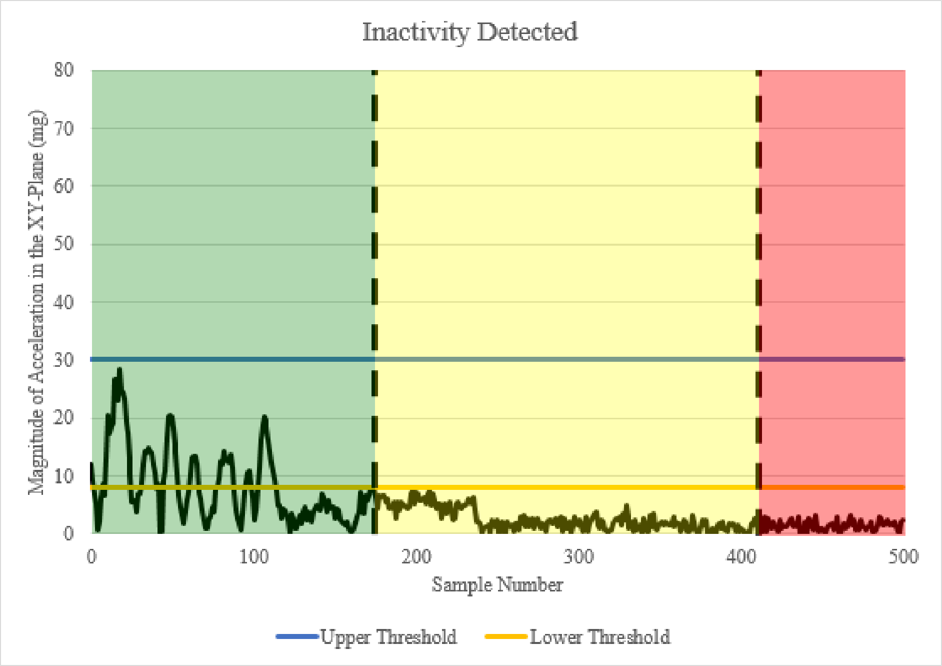


Figure 2.8 - Motion Sensing Algorithm Detecting Inactivity

Chapter 3 – Prototype

3.1 – Hardware

For the prototype of our design, we tried to use as many “off the shelf” parts as possible in order to cut down on processing and fabrication time and to allow for easy interchangeability between device components. Using readily available parts was also advantageous in that it allowed for easy software debugging and code updates, unlike a truly embedded device.

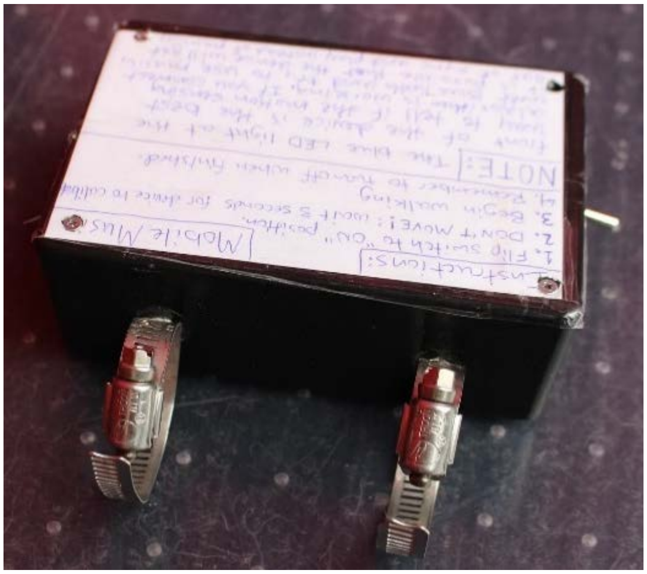

Mobile Mus
Instructions:
1. Flip switch to "ON" position.
2. DON'T MOVE!: wait 3 seconds for device to calib
3. Begin walking
4. Remember to turn off when finished.
NOTE: The blue LED light at the
front of the device is the best
way to tell if the motion sensing

Figure 3.1 – Outside View of the Prototype

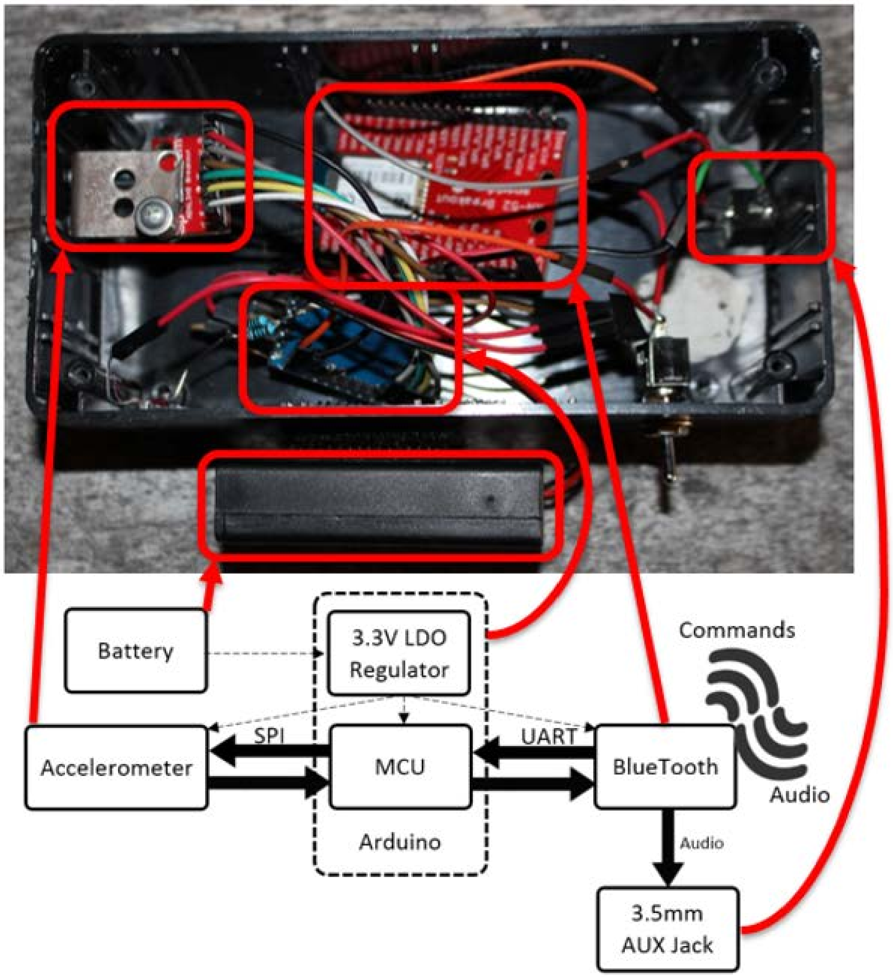


Figure 3.2 – Inside View of the Prototype with Corresponding Block Diagram

3.1.1 – MCU

The Microcontroller Unit (MCU) is the central processor for the embedded device. It is responsible for all communications, all state machines, and the execution of the calibration and motion sensing algorithms. In order to focus on determining the viability of the algorithms and state machines, we chose to use the Arduino Pro Mini for the prototype, as shown in Figure 3.3. This brand of MCU "break out" board is very well documented and has a large online community support, so many of the functions to interface with the other components have already been written. This allows for much easier debugging of the communications with peripheral components and much quicker validation of the internal data processing. The Arduino Pro Mini uses an ATmega328P MCU that can use an internal clock frequency of 8MHz and can operate off of a voltage supply of 3.3V with a current draw of roughly 5mA. It can also be programmed through an FTDI USB to UART chip and has dedicated pins for SPI communication as well as interrupt capabilities for interfacing with other components.

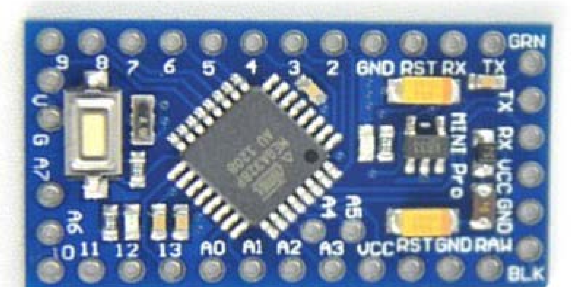


Figure 3.3 - Arduino Pro Mini 3.3V Microcontroller (reproduced without permission)

3.1.2 – Accelerometer

The accelerometer is responsible for collecting acceleration data for each of the three Cartesian axes and relaying that information to the MCU for processing. We chose to use the ADXL345 from Analog Devices, because it can be bought as a ready to use "break out" board and has a high acceleration sensing resolution of 4 milligravity/LSB for a range of ±16 gravity. It is also capable of running at a sampling rate of 200Hz with a very low current draw of 140μA at a supply voltage of 3.3V. The ADXL345 communicates through the SPI digital protocol as well as two programmable interrupt pins. These pins can be set to trigger for many things, but we only use one of the pins to trigger when there is a new set of acceleration data available. The interrupt will only reset after the MCU requests data through the SPI interface. This allows some flexibility with possible timing conflicts due to extra processing because the most recent data will be sent when requested, even if the previously held data was never read.

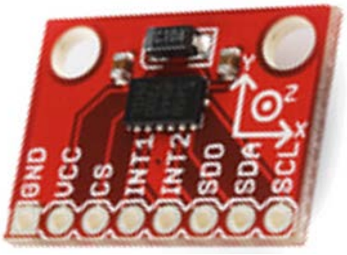


Figure 3.4 - Analog Devices ADXL345 (reproduced without permission)

3.1.3 – BlueTooth Module

The BlueTooth module is responsible for streaming audio from and transmitting audio control commands to the media player over a wireless connection. The BlueTooth module that we chose for this specific application was the RN-52 by Microchip Technology. We chose to use this component because it is capable of using the A2DP, AVRCP, and SPP BlueTooth profiles mentioned earlier so that it can stream audio, send audio controls, and transmit and receive data, respectively. The BlueTooth module also has a built in DAC and DSP, so that it can directly output an analog music signal, instead of the digital audio signal that is sent over BlueTooth. The RN-52 outputs the analog audio signal as differential pairs for both the left and right channels. While this is great for noise reduction, standard 3.5mm phone jacks are single ended outputs. This coupled with the lack of easy access to a differential to single ended amplifier means that only one of the differential outputs can be connected to the audio jack output, resulting in half the intended audio signal (-6dB) and a much worse signal to noise ratio. This device also has a GPIO pin that can be toggled in order to send the play/pause command through the AVRCP profile. This pin is connected to a GPIO pin on the Arduino board that is used to control the audio stream through the BlueTooth module. The RN-52 also has a current draw of roughly 30mA at a supply voltage of 3.3V while streaming audio through the A2DP profile, which makes it the largest power consumer of the three main components.

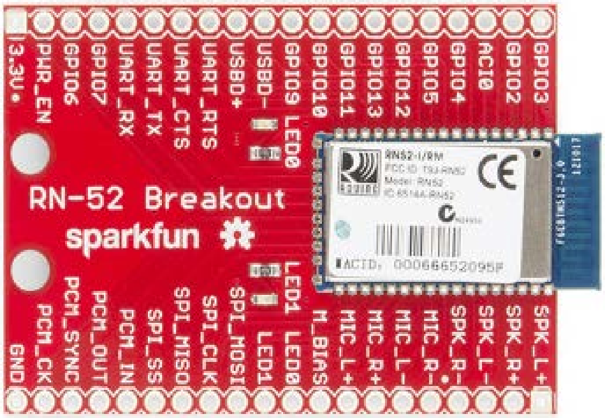


Figure 3.5 - Microchip Technology RN-52 (reproduced without permission)

3.1.4 – Power Management

For the prototype, the power management was limited to a battery pack and a 3.3V regulator located on the Arduino board, as shown in Figure 3.6. The battery pack holds three AAA batteries in series, providing roughly 750mAh at 4.5V. For the current draw of the prototype, this equates to roughly 15 hours of battery life. The raw battery voltage of 4.5V is greater than some of the components maximum supply voltages, and therefore must be regulated down to 3.3V by a fixed output LDO linear regulator. The output of this regulator provides power to all of the components in the prototype.

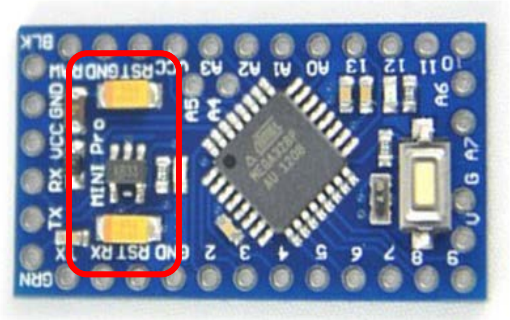


Figure 3.6 - 3.3V LDO Regulator (reproduced without permission)

3.1.5 – Peripherals

The 3.5mm phone jack is a standard part that is used to connect the audio from our device to the speaker or headphone. We chose to use a stereo, panel mounted jack from RadioShack because it fit our needs for a right and a left channel and was readily available. We also incorporated an LED with a current limiting resistor into the prototype. It is controlled by a GPIO pin of the Arduino so that it can act as a visual confirmation that the algorithm is working. The LED is also used to indicate that the device is calibrating and can be very helpful in quickly trying to test new code. The SPST power switch is connected between the positive output from the battery pack and the unregulated supply voltage pin on the Arduino board. It enables the device to be turned off when not in use, which saves power, and reset the device in case the software stops working or the device gets out of sync.

3.1.6 – Structural Components

The main structure of our prototype is a medium project enclosure purchased from RadioShack. The reason that this part was chosen was because it provided plenty of room for all of the necessary electrical components, and it is easy to modify the enclosure with holes for switches and mounts. For this prototype we tested both the clip style mount and an adjustable pipe clamp. The clip, shown in Figure 3.7, needed to be attached to the enclosure using a hardening putty adhesive. This was a small problem because it meant that once attached, the mount could not be adjusted further.

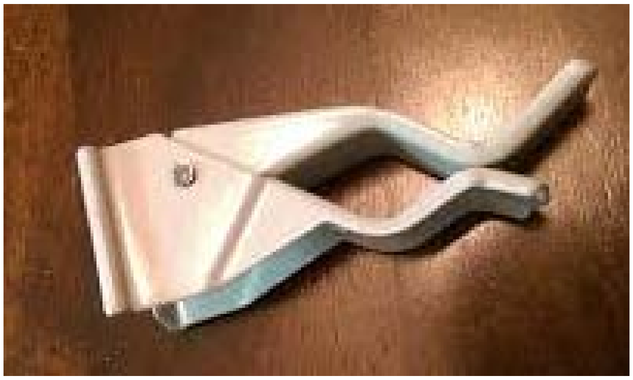


Figure 3.7 – Clip Style Mount

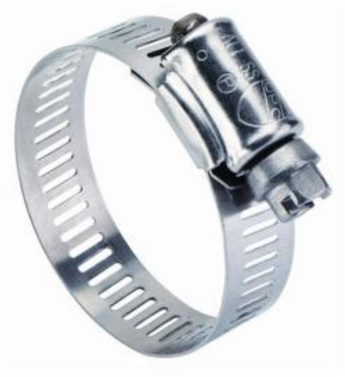


Figure 3.8 – Adjustable Pipe Clamp

The pipe clamps proved to be much more useful, as they were mounted with machine screws, and did not have any spring components to fail. The metal clamp did however scratch the paint on one of the walkers. To prevent this from happening again, we purchased some rubberized tape to protect the tubing. The clamps were able to provide a tight fit on the tube, and allowed us to test the various electrical components necessary for the system.

3.2 – Software

The software of the prototype runs on the MCU of the Arduino Pro Mini and creates an autonomous machine that requires little human interaction to operate. The software for the prototype consists mainly of two parts: the algorithm implementation and the communications with external components.

3.2.1 – Algorithm Implementation

In order to implement the calibration and motion sensing algorithms effectively, three state machines are used. The SYSstate machine is used to switch between the calibration state and the motion sensing state of the device. For the prototype, the SYSstate variable is initially set to the calibration state. Once the calibration algorithm is successfully executed, the SYSstate variable is set to the motion sensing state where it will remain unless the device is reset by toggling the power switch, as shown in Figure 3.9. If the device is in the motion sensing state, then it will execute the motionSense function. This function reads the acceleration values from the ADXL345, computes the calibrated acceleration value, and implements the hysteretic threshold of the motion sensing algorithm through controlling the MVMNTstate machine, as shown in Figure 3.9. The MVMNTstate machine has no direct outputs, but is used in the control of the PPstate machine which switches between play and pause states for the audio stream. If the MVMNTstate is in the motion state, then the PP state is in the play state and the watchdog timer resets. If the MVMNTstate is in the no motion state and the polled watchdog timer expires, then the PPstate switches into the pause state. This functionality can be seen in Figure 3.9. Because the AVRCP profile only allows us to toggle the play or pause of the audio stream and not directly control its state, the toggle signal is only sent when the PPstate switches states. The PPstate also controls whether the indication LED is on or off, depending on whether it is in the play or pause state, respectively.

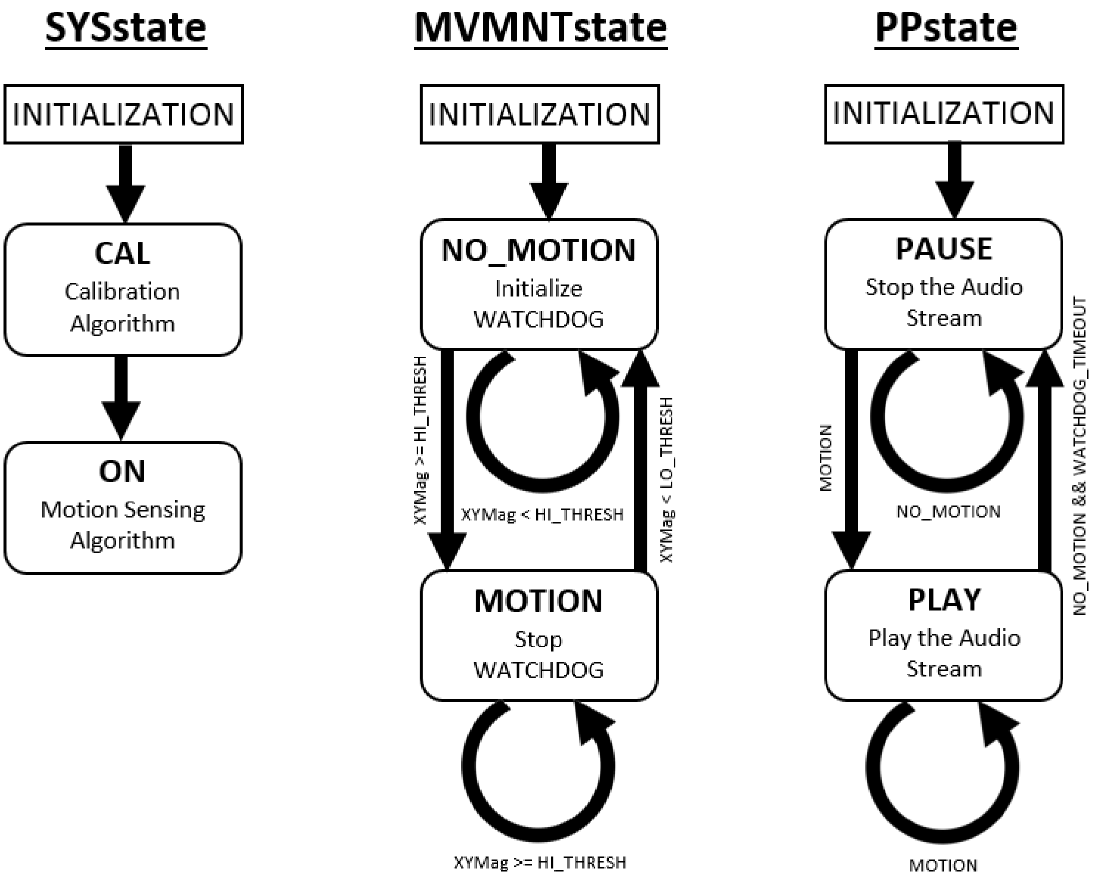


Figure 3.9 - State Diagrams for Prototype Device

3.2.2 – Communications

In the prototype, the only external communications that the MCU is responsible for are toggling a GPIO pin to control the play/pause command of the RN-52 and communicating with the ADXL345 using SPI and an interrupt. For SPI communication some setup was required, such as declaring a Chip Select pin and choosing a clock phase and polarity, but besides that only a read register and a write register function are needed to properly communicate with the accelerometer. Since Arduino already has predefined SPI byte transfer functions in its IDE, these two functions mostly consist of manipulating the Chip Select pin and adjusting bits specific to performing certain tasks only on ADXL345 registers such as read and multibyte read. A DATA_READY interrupt is used to inform the MCU that the accelerometer has new data waiting to be read. This process will happen 200 times per second, the sampling rate of the accelerometer, so that there are plenty of clock cycles in between to perform the required mathematical and logical analyses.

3.3 – Testing and Feedback

For testing our prototype, we visited our customer, Avalon Academy, in order to validate the calibration and motion sensing algorithm by testing the device on some of their students with varying ranges of limited mobility. We also wanted to get any quantitative feedback regarding the functionality of the algorithms as well as the rest of the device.

For mechanical testing we wanted to ensure that our mounting methods were both functional with different tubing sizes, and strong enough to keep the device from moving on the walker. The feedback from these tests would provide us with the necessary information to choose a mount for the final design.

3.3.1 – Test Procedure

The prototype was dropped off to Avalon Academy for two weeks so that they could demo it. During this time, the faculty was instructed to note if, when, and how the device stopped working ideally. While this worked for collecting quantitative data, we needed to construct a different method of collecting acceleration data as well as when the PPstate switches from PLAY to PAUSE and vice versa. In order to accomplish this, a simpler set up using only an ADXL345 accelerometer, an Arduino Pro Mini board, and an FTDI USB to UART board for connecting the Arduino to a laptop. The code was kept exactly the same as the prototype, but some Serial.print() statements were added in order to acquire data for analysis. First, the calibrated angular offset is printed in degrees. This is followed by a stream of x, y, and z acceleration values that have not been adjusted for calibration. Finally, if the PPstate ever changes from PLAY to PAUSE or PAUSE to PLAY, an ‘A’ or a ‘B’ are printed to the laptop, respectively.

Once we have the raw acceleration values and the calibration offset, a program like Microsoft Excel can be used to do the same processing that occurs in the MCU to determine the PPstate and effectively control the audio stream. We then graph this data with the printed PPstate state change markers mentioned earlier to obtain the graphs in Figure

3.10. By visually inspecting the graphs, we can determine the effectiveness of the motion sensing and calibration algorithms.

While the device was on location, we had the users try out both mounting styles on the same box. We asked them to note how long the various methods took to install or uninstall, and any issues they ran into with the stability, or functionality of the mount. There were two different individuals at The Avalon Academy who tried to do installs, and we performed them ourselves during our multiple visits to the school.

3.3.2 – Test Results and Feedback

The results of our quantitative testing can be seen in Figure 3.10. Each of these graphs represents the calculated acceleration values for four different children with Cerebral Palsy over a ten second period, measured at 200 samples per second. It is important to note that much more data was taken – an average of 40,000 sample points per patient, and all of this data was analyzed, but for the sake of simplicity and to have the total sample numbers normalized, we look at only a selection of this data to demonstrate the testing process and analysis. The x and y axes represent the sample number and the magnitude of acceleration in the XY-Plane in scaled by milligravity, respectively. The upper and lower thresholds of the hysteretic threshold are represented by the horizontal blue and yellow lines, respectively. A switch in PPstate from PAUSE to PLAY or PLAY to PAUSE is represented by a vertical orange or grey line. The remaining light blue lines represent the accelerations of the students as they were working on their gait training.

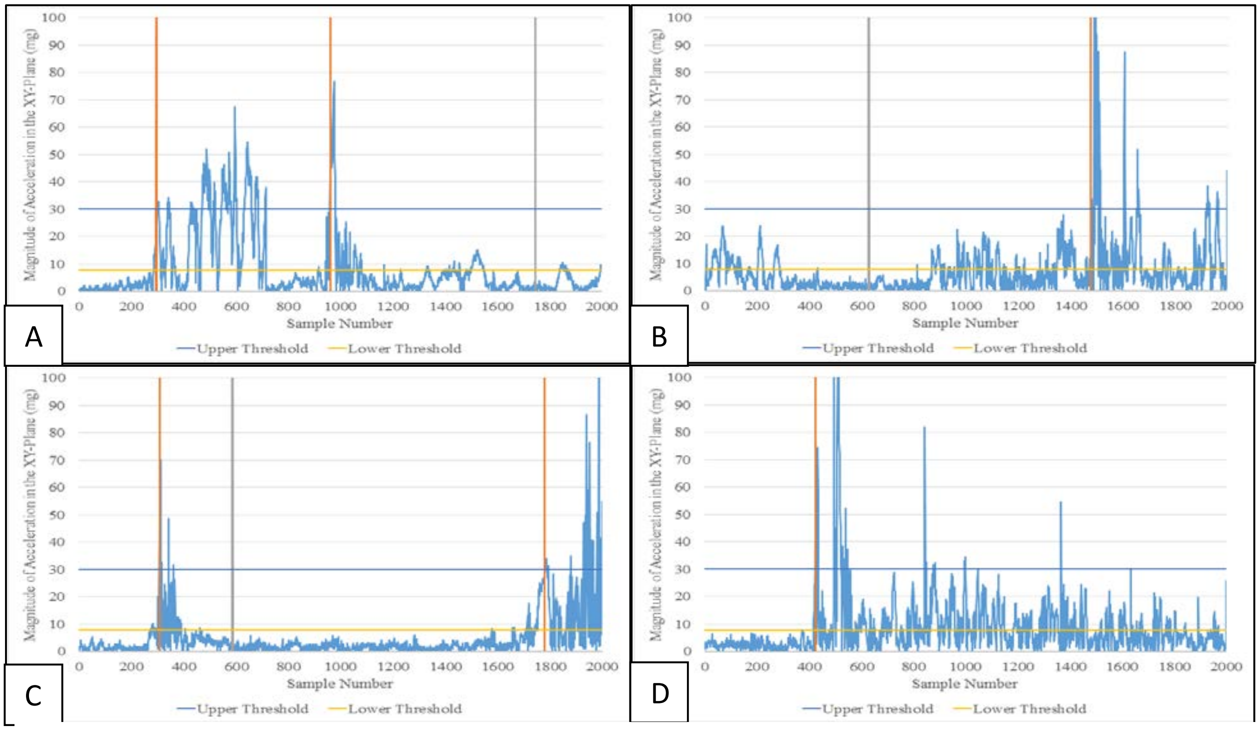


Figure 3.10 - Results from Prototype Testing

These graphs give us valuable insight into the effectiveness of both the calibration and motion sensing algorithms. If the calibration function determines the correct angular offset and it gets successfully integrated into the motion sensing algorithm, then we would expect the acceleration values to tend towards zero as motion stopped. If any of this process was interfered with, for example a lot of movement occurs during the calibration data collection, then we would expect to see some kind of static offset to the acceleration offset. This offset is due to the constant vector of gravity having an impact on the final acceleration readings; the measurement plane is no longer orthogonal to the gravity vector. The magnitude of this offset id dependent on how off the calibrated offset is from the actual offset. As we can see from the graphs in Figure 3.10, especially graph C from sample 600 to 1600, this was not the case. Our calibration algorithm and the implementation of its calculated offset into the motion sensing algorithm appears to work successfully.

With the ideal functionality of the motion sensing algorithm as a reference, we analyzed the accuracy of the motion sensing algorithm outputs (play and pause) with respect to the collected data. In Figure 3.10, graphs B, C, and D all have no errors in them, performing identically with our ideal motion sensing algorithm, but graph A presents a problem. There are two PLAY commands back to back when there should be a PAUSE command somewhere in between them. A likely cause of this issue is that the device tried to quickly change states from PLAY to PAUSE and back to PLAY again, due to a watchdog timeout shortly followed by an acceleration value greater than the higher threshold. This guess is reasonable since the number of samples between the last acceleration value greater than the lower threshold and the second "accidental" play signal appears to be equal to the timing delay. However, it could also be possible that a timing error was encountered and the MCU simply didn't have enough processing time to send the PLAY marker. If were able to directly control the playing and pausing of the audio stream from the media player, this problem wouldn't be a big deal as the device would just continue to stream the audio and ignore the second play signal since it is already playing. Unfortunately this is not the case, and our only control over the media player is a play/pause toggle command. This means that the only way of knowing the state of the audio stream is to make an assumption that the device is paused during initialization and then to keep track of PPstate changes. With the kind of error that occurred in graph A, the device would send the second play/pause toggle command with the intent of streaming the audio, but the media player only sees it as a command to toggle from its current state, play, to its opposite state, pause. This effectively makes audio command communications between the MCU and the media player out of sync so that now when the MCU tries to play music, it will actually pause it and vice versa.

For the qualitative feedback on the prototype after two weeks of use, the faculty at Avalon Academy tended to agree with our findings on synchronicity issues with the motion sensing algorithm. They didn't explicitly note that the device and the media player were getting out of sync, but they did note that sometimes the device would play music instead of pausing it or pause music instead of playing it. They also had some feedback on the quality of the audio signal. Their feedback suggests that the audio gain is too low out of the BlueTooth module and that the signal is very noisy, making it difficult to hear the music playing during the gait training sessions.

As for mechanical feedback, we learned that the spring inside of the clip on mount degraded very quickly. After only a couple of days of use, the clip mount was essentially useless, as there was not enough force applied to the tube to stop the device from rotating. It was this result that pushed us towards an adjustable style mount. We decided that by avoiding springs entirely, we do not have to worry about the fatigue of the part and its possible lifespan.

The major complaint about the pipe clamp used as an adjustable mount was that it was time consuming to install. To install the pipe clamp, either a hex head or a flat head screw driver is needed, and it takes a while just to get to a point where you are putting tension on the clamp. It was for this reason that we knew a mount would need to be designed and built by us.

3.3.3 – Corrections

In order to try to ameliorate these problems in the final design of the board, some hardware and software corrections needed to be made. First was the issue of synchronicity. As mentioned earlier, this problem is inherent to the toggle interface that the AVRCP profile provides between the device and the media player. The best solution to this problem would be to use a mobile app to create an audio streaming state in the media player and control that with predefined symbols sent over the standard SPP profile. Unfortunately, we do not have access to an app so we cannot eliminate the problem, but we can help make it less often. In the prototype, the AVRCP command was went by toggling a GPIO pin on the BlueTooth module. This process requires various delays to ensure proper operation, which can increase that risk of running into timing problems. By sending the command via UART to the device, we can greatly reduce this delay time and therefore reduce the likelihood of the PLAY/PAUSE states getting out of sync between the device and the media player.

From the feedback about the quality of the audio output, it is obvious that an audio amplifier is needed for this device. As mentioned in Chapter 2, the use of an audio amplifier would help both eliminate noise and increase signal gain. By taking advantage of the differential outputs that the BlueTooth module outputs, we can subtract the differential lines from each other, effectively eliminating any noise common to both transmissions as well as doubling the amplitude of the signal. On top of this, the audio amplifier also uses filtering and applies its own additional gain, eliminating even more noise and delivering even more gain to the desired signal.

To correct the experiences of the user in regards to the mounting system, we knew that we would have to design something that did not need tools, was sturdy, and was quick to install. A few different ideas were looked into, including the previously mentioned "halo" mount. The simplest design idea involves two rounded pieces which clamp around the walker tubing, and could be hand tightened. Because it needed to be expandable, we chose to use a slider design which will be shown in the next chapter.

Chapter 4 – Final Build

4.1 – Hardware

Hardware for the final design was made using almost entirely custom made parts. The main components of the final build are the printed circuit board (PCB) and the shell and mounting assemblies.

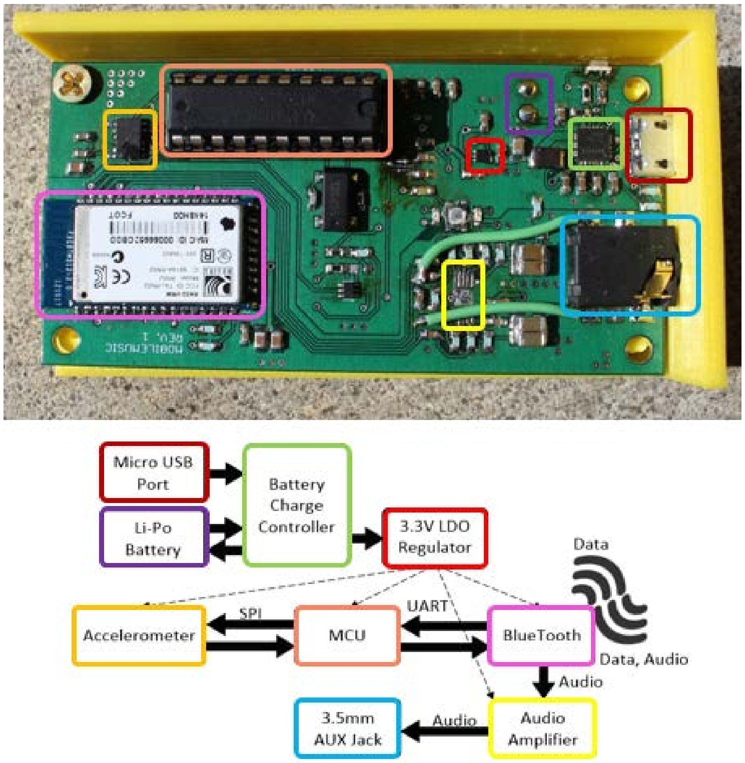


Figure 4.1 - Inside View of the Final Build with Corresponding Block Diagram

4.1.1 – Unchanged Components

From the prototype to the final build, only two of the components remained the same: the ADXL345 and the RN-52, as shown in Figure 4.1. The ADXL345 was kept because it provided adequate acceleration measurement resolution while requiring very little power. The RN-52 was kept because it was capable of using the A2DP and AVRCP profiles necessary for the BlueTooth communication to work. It is also important that it has a built in DSP and DAC so that it outputs analog audio signals. A common benefit of keeping these components in the final build is that the configuration and communication function can be kept relatively the same. The only difference the final build and the prototype is that they are now being used as discrete components wired appropriately via a PCB instead of breakout boards wired together, as shown in Figure 4.1.

4.1.2 – MCU

For the final build, we decided to use the MSP430G2553 MCU from Texas Instruments, as shown in Figure 4.1. This MCU has a 16 bit, RISC architecture and can run at up to 16MHz. We chose to use this MCU instead of the ATmega328P from the prototype because of its low power capabilities, lower cost, and additional external interrupt capabilities. The MSP430G2553 is capable of running off of slightly more than 4mA of current, roughly 20% less power than the ATmega328P. It also only costs $2.80 while the other MCU cost $3.70, giving a roughly 25% price reduction. On top of all of this, every GPIO pin of the MSP430G2553 can be used as an interrupt while the Arduino board was only capable of using two pins as external interrupts. With the added features of the final build, it is necessary to have more than two interrupt capable pins.

4.1.3 – Audio Amplifier

As mentioned in the feedback from the prototype, there was a need in the final design for an audio amplifier in order to have loud, noiseless music. For this purpose, we chose to use the TPA6112A2 150mW Audio Amplifier from Texas Instruments. We chose to use this part because it was the most compatible with the device requirements. First, it has two channels for stereo audio. It also takes a differential signal input and has a single ended output, which helps prevent noise and acts as a conversion from the differential output of the RN-52 to the single ended outputs of the 3.5mm phone jack. As an added bonus, it can run off of supply voltage of 3.3V, so there are no special power requirements. Unfortunately, as seen in Figure 4.1, there were difficulties in assembling this component so it was left off in order to test the rest of the device. This will be further addressed in the "Current Testing and Possible Changes" section to follow.

4.1.4 – Power Management

The power management portion of the design consisted of three components: a battery, a charge controller, and a voltage regulator. The battery used is a rechargeable Lithium Ion Polymer (LiPo) Single Cell from Adafruit Industries, as seen in Figure 4.1. We chose this battery because of its high energy capacity of 1200mAh as well as its compact size of 1.3” x 2.4”. It is also important that it has a nominal voltage of 3.7V, since this can easily be regulated down to the device supply voltage of 3.3V by a linear regulator without thermal considerations given our current draw is never more than 100mA. For reference, the calculated current draw should be around 80mA RMS when all components are operating at maximum current draw. This current draw also implies that the device should have at least a 15 hour battery life, which greatly surpasses our target of 3-5 hours. This component also includes built in voltage protection circuity to ensure that the battery can be overcharged and possibly have a catastrophic failure.

The charge controller acts as the voltage/current regulator for charging the LiPo battery. For the final design, we chose to use the BQ24253 charger from Texas Instruments, as shown in Figure 4.1. One advantage of this part is that it is a switch mode power supply (SMPS), so it has a high efficiency of 95% and there are little thermal concerns. It is also important to note that the BQ24253 has a switching frequency of 3MHz, meaning that smaller value LC filter components can be used in the design. Because these components are smaller, they generally cost less and require less space on the PCB, which is also valuable. The BQ24253 is capable of charging a single cell LiPo battery as well as powering the rest of the device from the standard 5V USB power. An additional feature is that it has outputs that can control LEDs that will give an indication of the charging status, which is useful information to the customer.

The final component of the power management is the voltage regulator that supplies power to every other component on the device. For this responsibility, we chose to use the TLV113333D LDO linear regulator from Texas Instruments, as shown in Figure 4.1. This component consists of two channels that each regulate to 3.3V, each with its own enable pin. This can be used to turn off the power supply of the BlueTooth module and audio amplifier when they are not in use, while the rest of the device can still function. This component is also thermally rated for two times our estimated current draw and has a nominal efficiency of 89%, so heat should not be a problem.

4.1.6 – Peripherals

The peripherals of the final design are the components that interface between the PCB and the enclosure. There are several peripherals that exist on the device, such as the micro USB jack, the 3.5mm phone jack, and the button, as well as indicator LEDs. The micro USB jack is solely used as the charger input for the BQ24253, effectively charging the battery and simultaneously powering the device. This plug does not need to be used in order for the device to properly operate, but the device should be charging whenever possible to ensure that it will not run out of charge while in use. The 3.5mm phone jack is used as an audio output for the device that can connect with most speakers or headphones. The specific audio jack that we chose contains a switch pin, which allows the device to detect whether an auxiliary cord is plugged in. While designing the PCB, we had to consider how the user was going to interact with these items. The audio jack and micro USB were placed on the thinner side of the PCB in order to ensure easy access for the user, as shown in Figure 4.1. This positioning also allows the cords to be more readily wrapped around the tubing that the device is mounted on, creating a much lesser potential for tripping due to dangling wires. Near the micro USB jack, there is an RGB LED with a small window in the case that indicates the charging status of the battery. This LED operates off of the voltage provided by the micro USB jack, so it will only work when the battery is being charged. If the battery is not fully charged, both the red and green LEDs will be on making an orange-yellow light. When the battery is done charging, the red LED will turn off, leaving only the green LED on to shine through the window in the enclosure.

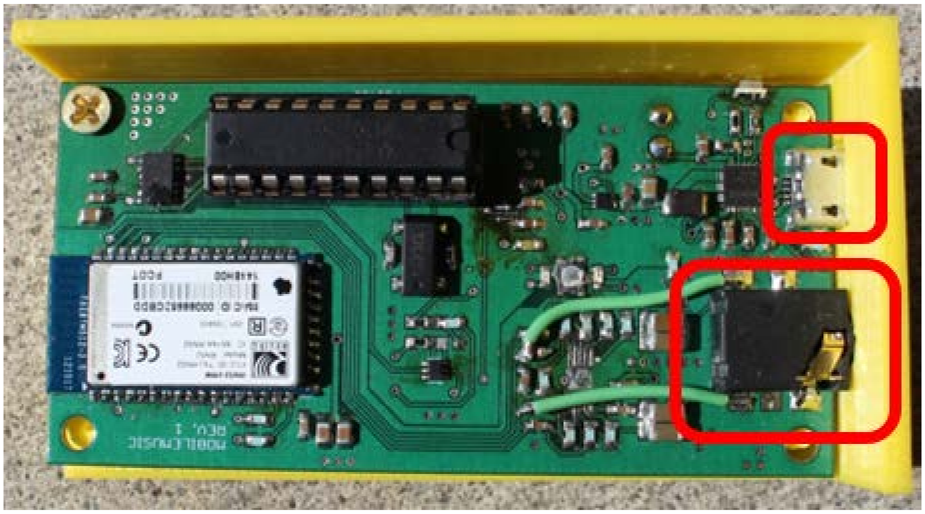


Figure 4.2 – Micro USB and Audio Jack Locations

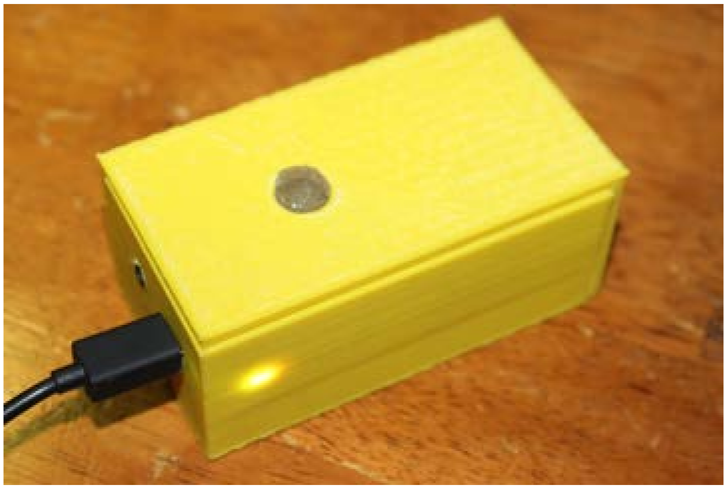


Figure 4.3 - Charging LED

The last peripheral, the button, is actually three in one. When used, the button presses down on a momentary switch which sends a signal to the MCU for it to decode, based on time held and current states. Beside the micro switch, there are two LEDs on the PCB. A red LED informs the user whether the device is calibrating, on, or asleep, while a blue LED shows the BlueTooth connectivity of the device. In order for the LEDs to be seen by the user and add some aesthetics to the device, we created a translucent button from an epoxy resin mold. This allows the light from the indication LEDs to shine through the button, so the user can get feedback as to what the devices current operating state and BlueTooth connectivity are.

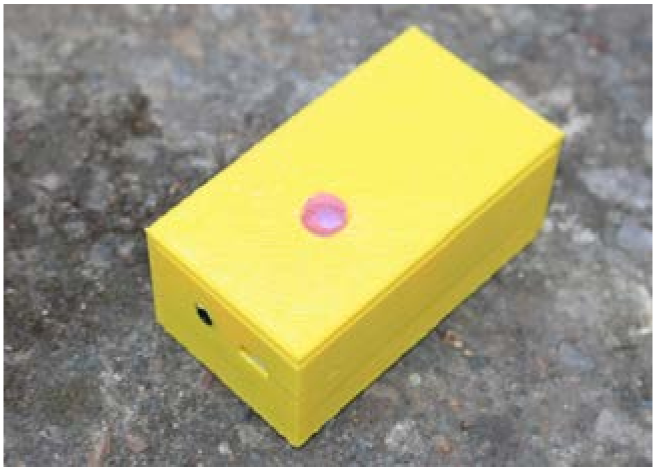


Figure 4.4 – Button LEDs, Red and Blue

The button fits into the shell based on an interference between itself, the top of the case, and the micro switch. This allows the button to stay in place and remain functional after multiple presses, despite having no permanent connections with any part of the assembly or PCB.

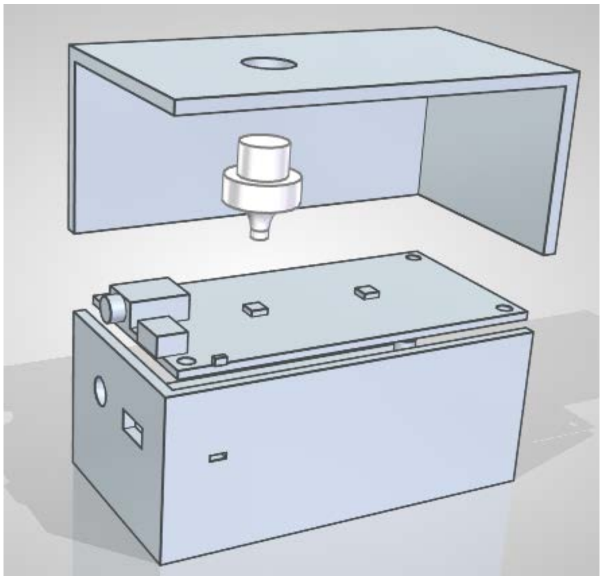


Figure 4.5 – Button Fit Solid Model

4.1.7 – External Shell Designs

During final design stages, two shell styles were considered to house the electronic components. Both of these designs had similar shape and size, and both are two piece designs. The first style was very simple in design, just a top and a bottom piece that slid together.

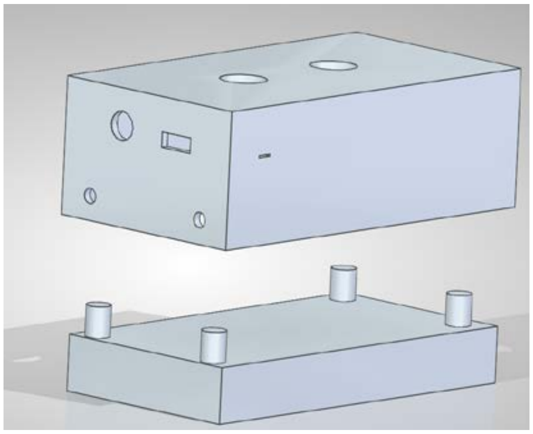


Figure 4.6 – Shell Design Style 1

It can be seen that this design style was created while the device was still going to have two buttons. The second button was deemed unnecessary once we decided that sleep mode would be a sufficient way to maintain the battery on the device. This design served as a base for Style 2, which was what we ended up putting into production.

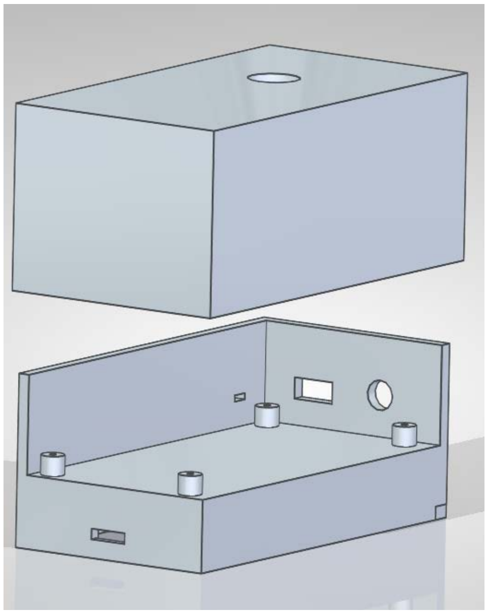


Figure 4.7 – Shell Style 2

Shell Style 2 uses a similar two piece design, but here, two of the walls are attached to the bottom and two are attached to the top. This helps with assembly, as there is no need to worry about being able to slide the USB and audio jacks into their proper holes. This design also gave us much more open access to the PCB while it was installed. This was very helpful for any debugging or hardware repair that needed to occur.

4.1.8 – Mount Design

From the prototype we decided that an adjustable style mount was the best choice. However, we did not want to continue using the hose clamps because of how long the installation took, and it required tools to mount to the walker. To combat these problems, we decided to design our own mount, which would be easily hand tightened to a secure fit on the walker tubing.

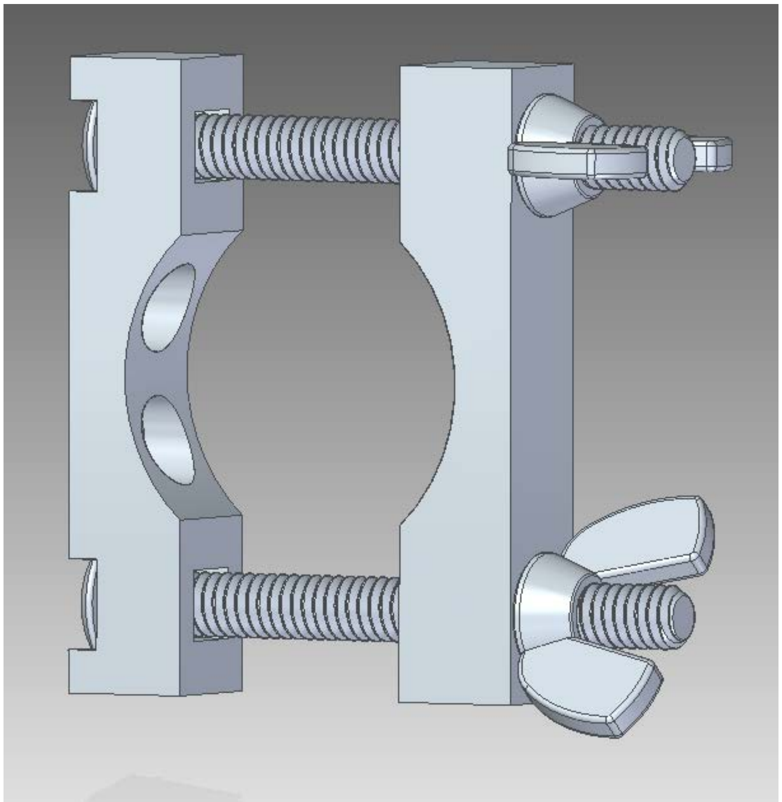


Figure 4.8 – Final Mounting Design Assembly

The final mount design comes in two pieces, a device side and a clamping side. As can be seen in Figure 4.8 the mount device side hold two carriage bolts in place. These bolts are 1.75" apart, and provide the channel that holds the walker tubing.

Once the tubing is within the channel, the outside half of the mount can be slid along the bolts until the outer edge of the tubing is reached. From here the user simply spins the wing nuts on, and then tightens until they feel that the device is securely attached to the walker. With this mount design we believe that we have solved most of the issues that plagued the other style mounts. This mount provides a very secure connection with the device as well through the use of two machine screws.

4.2 – Software

The software for the final design was very different than that of the first design. The main reason for this is the switch from the Atmel MCU to the Texas Instruments MCU. These two components use completely different IDEs and code styles, so most of the code either had to be modified or rewritten. Only the more logic and math oriented functions such as motion sensing and calibration algorithms remained relatively unchanged.

4.2.1 – Algorithm Implementation

Many parts of the algorithm implementation for the prototype were simply copied over from the Arduino code. The state flows of the MVMNTstate and PPstate in the final build are equivalent to those of the prototype. Despite this, we are still experiencing difficulties in successfully implementing the motion sensing algorithm. These bugs can be attributed to problems with the inactivity delay portion of the algorithm. Instead of polling the elapsed time for the inactivity delay like the prototype code, the final build code uses a timer interrupt that acts much more similarly to a true watchdog timer.

The only other big change from the prototype was the SYSstate variable which includes a SLEEP state and a new state flow. In the SLEEP state, the MCU disables power to the audio amplifier and the BlueTooth module and then enters a low power mode. The two audio components require the most current out of any device at roughly 60mA combined. In the low power mode the MCU consumes roughly 0.6μA of current, making the total system current usage in the SLEEP state roughly 1mA when accounting for the accelerometer, and quiescent currents. With a battery capacity of 1200mAh, the device should theoretically be able to remain in sleep mode for fifty days from a full charge until shutdown.

Whereas in the prototype the user had no control over the flow of SYSstate, they are now capable of controlling it via the button. The system initializes the SYSstate variable to the CAL state, where the device will execute the calibration algorithm. After calibration is done, the device will enter the ON state and begin executing the motion sensing algorithm. The device will remain in this state until user input is received. If the user quickly presses the button, the state will change to CAL until calibration is finished and then back again to the ON state. If the button is pushed and held, the device will enter the SLEEP mode where it will stay until given further input. In order to exit the SLEEP state, the user must press and hold the button until the device enters the CAL state again, followed by the ON state. This entire process is illustrated in Figure 4.9. The code to implement this multifunctional button took advantage of the MSP430s interrupt capabilities. It only requires one external interrupt and one timer interrupt, which saves one pin on the MCU if the device were to use two buttons and two external interrupts. This may not seem like it is worth the effort, but even with one button in the final design the MCU had no available GPIO pins. This means that the final design would not have been possible without this dual purpose button.

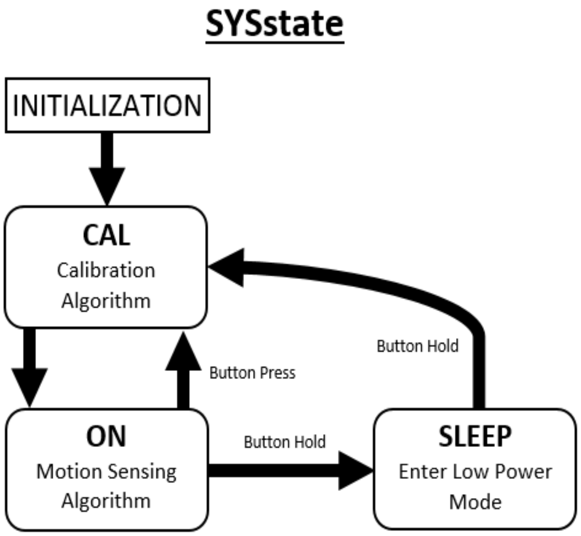


Figure 4.9: SYSstate State Diagram for the Final Build

4.2.2 – Communications

The final design requires many more communications between the MCU and other components than the prototype did. On top of this, the libraries that allowed for simple communications on the Arduino platform are not available on the MSP430 platform. This means that each communication function was made from scratch at the register level. These communications include interactions with the indication LEDs, SPI with the ADXL345, UART with the RN-52, multiple interrupts and multiple GPIO pins.

In order to lower power consumption, the indicator LEDs are never fully on, but are instead flashed in different patterns to signify different states to the user. These patterns are created by utilizing the MSP430's timer interrupts. The red LED is used to indicate the state of the SYSstate variable. When the state is SLEEP, the red LED is completely off. When the device enters the CAL state, the light will initially be on and dimmed by PWM to represent the initial timing delay followed by rapid flashing with a 50% duty cycle. When in the ON state, the LED will flash at a slower rate and at a much smaller duty cycle, in order to further help conserve power. Ideally the blue LED would be controlled by the MCU as well, but as of now it is controlled by the BlueTooth module itself. This light is meant to show whether the device is connected, disconnected, or in the process of connecting with similar light patterns to the red LED.

In order to implement the SPI communication with the accelerometer, all of the read and write functions had to be written from scratch. This was accomplished by manipulating the RX and TX buffers as well as polling the flag registers. These inform the MCU of the status of the communication to ensure that information is not lost in between. Once these functions were written to initialize the digital communications protocol, the actual communication with the ADXL345 was virtually equivalent to the prototype.

All of the read and write functions for the UART communication with the BlueTooth module had to be written from scratch as well. Through a similar process of manipulating RX and TX buffers and polling flag registers, these basic communication functions were created. Even though the software currently only requires the MCU to send the play/pause command, we tried to generalize the communication functions in order to allow for flexibility in the future. Unfortunately, this implementation was running into timing problems sometimes. If the device was trying to send two commands to the RN-52 within a short time span, only the first command is registered. This is causing the device to easily get out of sync with the media player, as mentioned with the prototype. In an attempt to prevent this from happening, a delay was added to the end of the read and write functions but it was unsuccessful in solving the problem

For the final build code, there are three new interrupts in addition to the DATA_READY interrupt already used in the prototype code. The BUTTON interrupt is used to help control the functionality of the button, as mentioned in the previous section. This interrupt works in conjunction with a timing interrupt to allow for the button to be used for two functions instead of one. Another interrupt is the JACK interrupt, which informs the MCU whether the 3.5mm phone jack is being used. This interrupt also makes it so that the audio amplifier will automatically shut down when there is no audio output present, so the interrupt is meant mainly to update the MCU instead of affecting any of the state flows. Lastly there is the GPIO2 interrupt, which is connected to the GPIO2 pin of the RN-52. This interrupt lets the MCU know that there has been a change in the BlueTooth connectivity state. We did not have time to work on the code to query the connection status, but the hardware is still implemented.

Lastly, there are two GPIO pins that the MCU uses. The first is connected to the GPIO9 pin of the BlueTooth module. The output state of this pin determines the communication mode of the RN-52. When the pin is high, the UART sent to the module is relayed over the SPP profile to the media player. When the pin is low, AVRCP commands and configuration data can be sent and received from the RN-52. This functionality is integrated into the UART communication functions. The other GPIO pin is connected to the enable pin of the LDO channel that supplies power to the audio amplifier and the BlueTooth module. As mentioned earlier, this pin is only pulled low during the SLEEP state otherwise it is left high.

4.3 – Current Testing and Possible Changes

Unfortunately, we were unable to completely finish debugging and testing the new device, meaning that many problems were left unresolved. Although we were unable to solve them completely, we did have time to hypothesize about what was causing them and how they can be resolved. This section will consist of the issues that we have or have not entirely figured out, as well as their solutions or possible solutions, respectively.

4.3.1 – Electrical Testing and Changes

For the electrical hardware, we were almost able to get the device completely functional with the exception of the audio amplifier. It is still unclear whether this is due to a design error or an assembly error. It is unlikely that it is due to a design error, as it is based off of a reference design. Since the audio amplifier has an exposed pad on the bottom of it and a very small pitch, it is very difficult to tell if the connections are soldered properly since we had to hand assemble the device. In order to check the design of the audio amplifier, we would likely have to get the PCB assembled professionally in order to ensure it is properly connected. From here, we can determine the validity of the amplifier design.

Through testing the board we have also come across many smaller changes that could be made to the PCB and design. First, there is a pull up resistor required between the VCC and RST pins of the MSP430G255, in order for the MCU to function properly as an embedded device. Also, the shutdown pin for the audio amplifier was accidentally connected to the wrong output of the Schmitt trigger, so that it was controlled by the button instead of the audio jack. There is also an extra trace coming off of one of the Schmitt trigger pins. While this has no effect electrically, it is still good practice to avoid unconnected traces as they can act like antennae. By shorting out the Schmitt trigger and looking at the interrupt signals on an oscilloscope, it was obvious that the RC constant of the switch circuit was capable of filtering out all of the jitter. This means that the Schmitt trigger can be completely eliminated from the design. Lastly, the external 32.678kHz crystal connected to the MCU never ended up being used in the code. This component can be eliminated in order to free up two GPIO pins that can instead be connected to the

UART_RTS and UART_CTS pins of the RN-52. This pins will allow us to have more control over the flow of the UART communication.

4.3.2 – Software Testing and Changes

The software provided the most difficulty in terms of debugging. Overall, the biggest problem that remains with the software involves timing issues. For this reason, all of the possible suggestion in this section are related to eliminating or ameliorating this problems. The first issue was the problem regarding the functionality of the timing delay. A possible change to the code would be to base the inactivity delay on the number of acceleration samples that have passed since the last MOTION state instead of a time value. This would isolate this function from any sort of timing and therefore eliminate any sort of timing complications associated with it.

Another issue that arose during the initial testing of the software was UART timing conflict. A possible solution to this would be to utilize the UART_CTS and UART_RTS pins of the BlueTooth module. These pins allow the devices to know when each other are busy, so that data transmission can be delayed until both are ready to communicate. I do not know exactly how we would go about implementing the code associated with this, but it has the potential to eliminate miscommunications due to a component being busy with a different process. Another method of reducing timing conflicts is to make the processing time shorter. This could be accomplished by using more efficient floating point operations during the motion sensing algorithm. This would likely involve writing code at the assembly code level in order to fully optimize the calculations performed. This combined with the first solution would hopefully lead to a functional device, but it is likely that more debugging would still be required.

4.3.3 – Mechanical Testing and Changes

The original plan for mechanical testing is shown in the figure below, titled Experimental Protocol. We initially wanted to test 11 different aspect of the project, but were unable to complete all of them.

Table 4.1 – Experimental Protocol

Evaluation	Equipment	Accuracy	Trials	Expected Outcome	TESTED?
Drop Testing	Mobile Music, Measuring Tape, Speaker	1ft	10	Survive drop from 5ft w/o cracks or loss of function	Yes
Water Resistance	Water, Spray bottle, Mobil Music, Stopwatch, Batteries	N/A	5	Water Resistance from spray and splash	No
Battery Life	Stopwatch, Mobile Music, Speaker, Batteries	10 min	3	3-5 hours	Yes
Installation Time	Stopwatch, Screwdriver, Mobile Music, Walker	1s	5	30s	Yes
Weight	Scale, Mobile Music	0.1 lb	4	<4 oz	Yes
Size	Measuring tape, Mobile Music	1 cu.in.	4	< 30 cu. In.	Yes
Manufacturing Time	3D printer, PCB Components	10s	4	2 hours	No
Motion Detection Accuracy	2 Stopwatch, walker, Mobile Music, Speaker	0.5s	5	95%	No
Tubing Size Fit	Screwdriver, Mobile Music	0.1in	5	3/4" to 1-1/2"	Yes
Assembly Time	Screwdriver, Mobile Music components, hardware, stopwatch	10s	5	5 min.	No
Range	Measuring Tape, Mobile Music, Speaker	1ft	5	15-20 ft	No

By the end of our time with this project, we have been able to complete six of our desired tests. These were the drop test, battery life, installation time, tubing size fit, size, and weight. We were unable to complete some of the originally planned tests because of problems with the software and the audio amplifier.

Chapter 5 – Engineering Standards

5.1 – Manufacturability Analysis

The shell that has currently been built was rapid prototyped using a MakerBot 2.0, located in the Santa Clara University Maker Lab. The various pieces of the shell and mount were printed using the standard PLA filament, designed for use with the MakerBot. This method of manufacturing was chosen because we knew that only one or two devices would need to be built over the course of the project. Because the MakerBot was readily available to us, and provided very quick production time, it was the perfect choice for the project.

However, this manufacturing method would not be ideal for mass production of this product. The printed pieces have a relatively high cost of production when compared to other methods such as injection molding. The speed of these processes is also much quicker than the MakerBot. The main reason injection molding was an unrealistic goal for us is the initial cost to design and build molds. Despite the device being a simple shape, there are small details which would need to be machined in after the mold was created. For the span of our project, it would have been very difficult to design mass production methods for this product.

Having to hand solder all of the components to the PCB was extremely time consuming and was the cause of several malfunctions during the initial testing. If this board were to be reproduced again, it is highly recommended that the device be assembled professionally with the use of reflow ovens and solder paste. While this will add some costs to the final product, the amount of time and stress that it saves is worth it. If the device is going to be mass produced, then it is recommended that the dimensions of the PCB be optimized for panelization. This will help to ensure the lowest cost per board. Is it is also important to note that the more devices that are created and assembled, the more money is saved on the per unit price. These savings can then be passed on to customers.

Another part of our manufacturing process that was challenging was creating the clear button. Initially we thought that this would be something easily made on a lathe out of acrylic. However upon further research we learned that acrylic would be too brittle for the dimensions we needed on the button. From here we looked into rapid prototyping the button along with the shell. The issue here was that only colored translucent filament is available, not clear as we would have liked.

The final method we found was to cast the piece from a mold. In order to create the mold, we first printed a version of the button on the MakerBot, with regular filament, shown below.

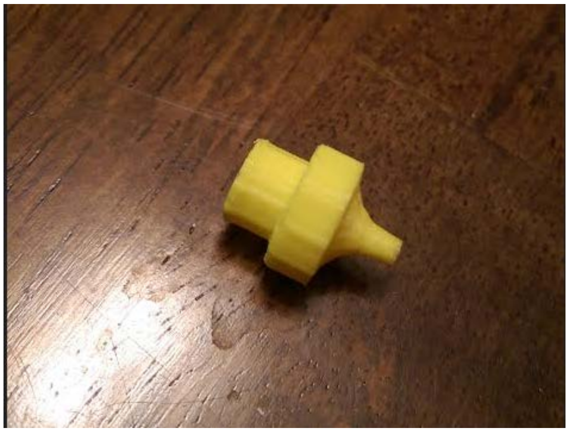


Figure 5.1 – Printed PLA Button Model

With this, we used EasyMold Silicone Putty, which is a two part mold creating putty, to build our mold for the resin. We were able to get a pretty good mold with this product, but would probably use something different for final production of this piece.

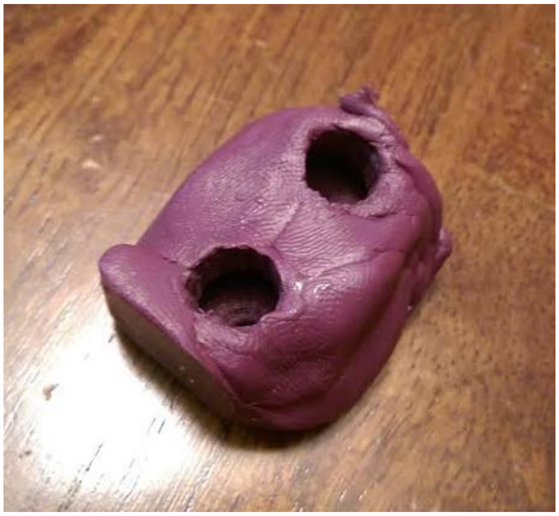


Figure 5.2 – EasyMold Silicone Mold

The epoxy resin we used was two part EasyCast resin. This is a clear resin, which is supposed to degas as it hardens, and become clear as glass. Our molded piece did not come out completely degassed, which led to some problems. We believe that this is because of the size of the mold, and how little area there was for the gas to dissipate.

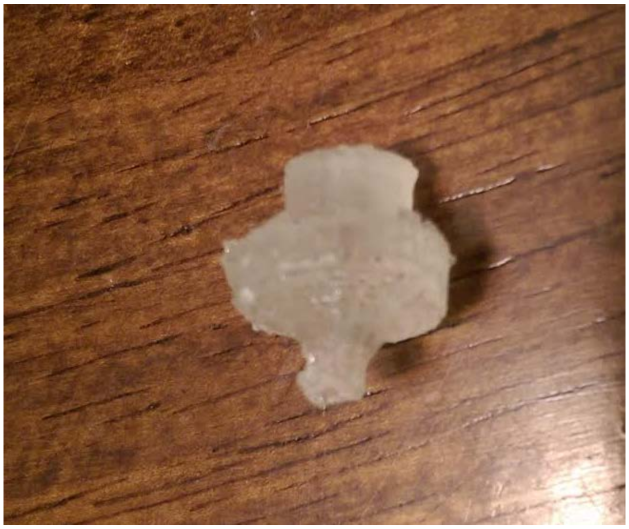


Figure 5.3 – EasyCast Resin Button

For future manufacturing, we would move to a machined mold for this part, and a different mold design which would allow the resin to reach its proper end form. The main reason for the poor quality of the button is the rushed timeline over which it was redesigned and created. After we realized that machining a button would not work, we only had about two weeks to redesign for it to be molded.

5.2 – Economic Analysis

One of the original goals of the project was to make sure that the device could be manufactured at a cost of $100 or less. The reasoning for this is to keep price relatively low, especially if the device was being made in low production quantities to ensure that it will still be readily available to those who need it. We know that the market for this device is not huge, as there are only certain facilities that would have a need for this product, but even for low production this device can be economically viable. As shown in Figure XX, at a production quantity of ten, we were able to produce the device for less than our target price of $100.

Table 4.1 – Simplified BOM for Different Order Quantities2

Component	Part Number	Unit Price: QTY 1	Unit Price: QTY 10
Micro Controller	MSP430G2553	$2.80	$2.25
Accelerometer	ADXL345	$7.32	$6.58
BlueTooth Module	RN-52	$23.30	$19.42
Audio Amplifier	TPA6112A2	$2.40	$1.63
Power Management	1200mAh LiPo BQ24253 TLV7113333D	$14.01	$12.58
Misc. Components		$17.05	$13.18
PCB Fabrication		$33.00	$26.56
Shell	3D Printed	$2.75	$2.75
Mount	3D Printed	$1.00	$1.00
Misc. Hardware		$6.70	$5.36
	TOTAL	$110.33	$91.31

Also, by making the device universal, users will not need to purchase more Mobile Music devices if multiple students do not perform gait training at the same time. We saw at The Avalon Academy that they usually only conduct therapy sessions with three or four students at a time. Therefore, the school could use four or five devices to cover the needs of more than twenty students.

5.3 – Intellectual Property Analysis

The first part which this patent analysis focuses on is the adjustable mounting system that was designed for use with this part. The mount is a two piece bracket, which uses off the shelf carriage bolts and wing nuts for its adjustability. The brackets are adjustable through their ability to slide along the bolts. This can also be helpful for oval or rectangular tubing, as the mount does not need to be excessively sized in the direction of the minor diameter/side.

The title of this invention would be "Sliding Adjustable Tubing Clamp." The purpose of the invention is to allow a single structure or device to mount to multiple tubing sizes or shapes. The invention was developed for a musical therapy device, which needed to be attached to walkers of different shapes and sizes. The reason for creating a sliding style clamp was to allow for complete opening of the mount to ease installation. The mount was also designed to be low cost, using off the shelf parts along with two simple rapid prototyped parts. These parts could be easily adapted to an injection molding process to lower costs at large volumes. The clamp was designed on February 20, 2015 by Alex Hildebrand, and built and tested during the months of March, April and June of 2015.

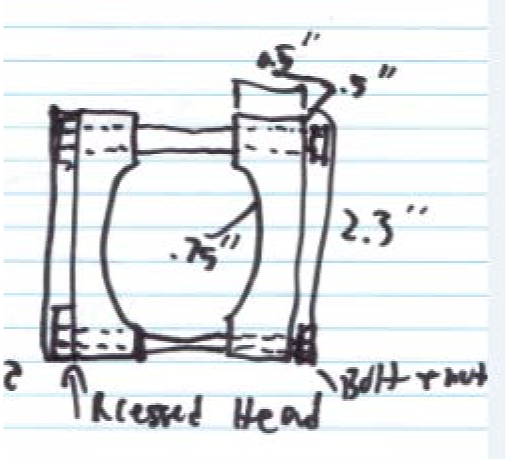


Figure 5.4- Initial Sketch (Drawn By Alex Hildebrand)

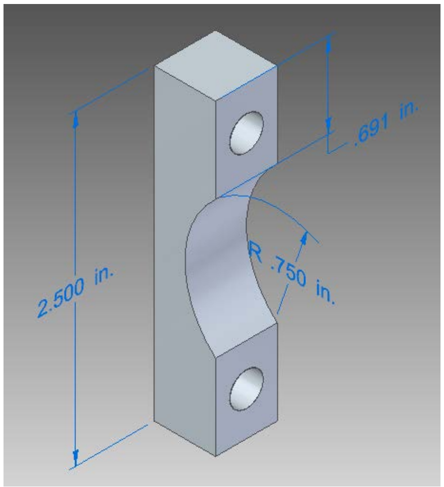


Figure 5.5 – Outside Clamp CAD Model

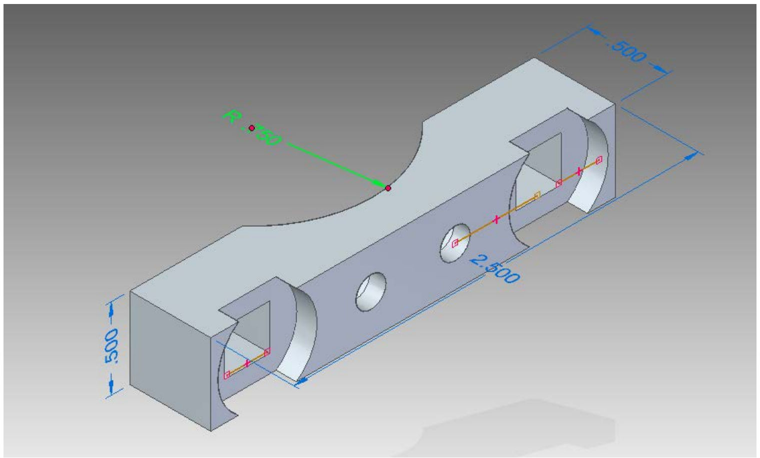


Figure 5.6 – Inside Clamp Model

The two most relevant U.S. classifications found during the patent search were 403 and 248. Class 403 is defined as "the generic class of connections between two or more rigid or semi rigid members at substantially a single locus." Sub-classes that apply are 398, 399, and 290. These subclasses represent the adjustability, yoke-style connection and open end of the design. Class 248 "provides for devices which carry the weight of an article or articles or otherwise hold or steady it or them against the pull of gravity." This is the primary purpose of any mounting device, and the subclass for tubing is 74, specifically 74.4, which is about separable clamps.

Similar patents found include, US 7179010 B2 – Adjustable pipe clamp assembly.

This patent is the most similar in style to our clamp. The primary difference is that only one end is adjustable, where ours uses two adjustment points. This mount is also used U-shaped support channel, commonly called Unistrut. This is another limiting factor for the adjustability of the design. Also found was WO 20060077326A3 – Adjustable Mounting bracket. This patent application is quite vague, but it describes two L-shaped bracket members which slide along mounting plates. Also required are mounting Jaws, placed on the brackets to hold the pipe, or member in place. This design is much more complicated than ours, and likely designed for a different purpose than ours.

Overall, it seems like this mounting style could be patented. All sorts of DIY robotics or computer systems builders could use a simple and cheap mount for almost anything in their projects. This could help to bring costs down on these one off machines, as they would not need to spend money on expensive prototyping materials, like 80/20 and similar. Claims that could be made about the clamp are that it uses very inexpensive parts, it could be built by anyone with access to a 3D printer and a hardware store if the files were available online, and how it could easily be scaled to almost any size necessary.

In addition to this, the calibration could be eligible for a patent because it meets the Patent Acts qualifications of a novel, useful, and nonobvious process. After a brief patent search, there have been no previous patents for processes that calibrate the rotational offset of three dimensional accelerometer using gravity as a reference for the sake of measuring accelerations orthogonal to gravity. While a more thorough search would need to be conducted before filing this patent, it initially appears as though the algorithm qualifies as novel. The calibration algorithm is clearly useful as seen by its role in Mobile Music and is not obvious due to its lack of prior implementation. All of these reasons qualify the calibration algorithm for a patent, contingent on a more thorough patent search.

Finally, the entirety of the code for the final build itself is actually protected as intellectual property. Because all of the code used for programming the MCU was either obvious or original, it is protected under the Copyright Act. The code is protected by copyright because it is considered a tangible form of expression. Copyright is not dependent on any paperwork being filed with the U.S. Copyright Office, but is assumed as soon as it is created. However, if the code is registered with the Copyright Office, then there would be legal advantages should infringement ever be filed.

5.4 – Ethics Analysis

As a whole, Mobile Music has very little potential to be used unethically. Any tool has the potential to be indirectly used for immoral purposes. For example, it would technically be possible for someone to hack this device and use it to administer shocks to someone else when they stop performing a repetitive movement. In reality, the amount of effort required to do this would be impractical. The potential benefits of this device far out way the potential harm.

That being said, we still have ethical obligations to Avalon Academy as well as other potential customers. We have an ethical obligation to create a product that can be useful to as many children with cerebral palsy as possible. This responsibility holds precedence over any legal rights we may have that would sacrifice the universality of my product. For example, we have an obligation to design a low cost system, as per the request of Avalon Academy, so that the devices can be readily available to all children with Cerebral Palsy regardless of their family's economic situation. This obligation outweighs our legal freedom to mark up the price of the product so that we could make a larger profit.

We also have an ethical obligation to create a useable product. This means that the final product needs to be functional, cheap, reliable, and user friendly. The most important of these is that the device is functional. We define that the device is functional if it causes a noticeable increase in physical therapy participation for the students at Avalon Academy. This is the essence of the project; the other three categories are secondary, but still of great importance to the success of the project.

Finally, we have an ethical duty to ensure that my device doesn't cause any harm to its users. Since this device is primarily being used by people with special needs, we have a duty to make the device as hardy and as fail-proof as possible. Most of this responsibility is taken care of in the development of the case and connector for the device to ensure that the device will not shatter if dropped or cause a tripping hazard. There were also extra precautions taken to reduce the thermal emissions of the device to prevent overheating of the LiPo battery which could potentially burn users.

5.5 – Social Analysis

The need for our project is rooted in social equity in that it provides a cost effective benefit to a disadvantaged group of people. This type of technology is even more important in the niche market of assistive technologies. Because it is such a small market, most of the technology used is either out of date or are generally more expensive than general consumer technologies. Our project provides an economically viable product, so that hopefully more people suffering from limited mobility will have access to our assistive technology. As a social benefit, our project helps to ameliorate the gap in physical ability between those afflicted with movement disorders, such as Cerebral Palsy, and those with typical mobility.

Our device makes physical therapy sessions more efficient by providing musical encouragement to the participants, creating motivation to complete sometimes difficult physical activities while simultaneously allowing the physical therapist to spend more time focusing on the patient's progress instead of convincing them to participate. These factors are even more exemplified when the patient is a child, where physical therapy can be more successful in that the muscle memory for gait is still forming, but patients are more unmotivated to provide continual participation due to loss of focus. Our device will provide a beneficial stimulus that will help maintain the patient's focus on their necessary gait training exercises. By aiding in physical therapy and effectively increasing the mobility of the patients, our device helps enable independence for a group of people that are generally dependent on someone else for basic human needs.

Chapter 6 – Future Developments and Final Thoughts

6.1 – New System Configuration

One possible development for a future group would be to implement a new, lower power system configuration. The difference between this configuration and the original configuration is that the device is now wireless and the media player is now connected to a headphone or speaker, as seen in Figure 6.1.

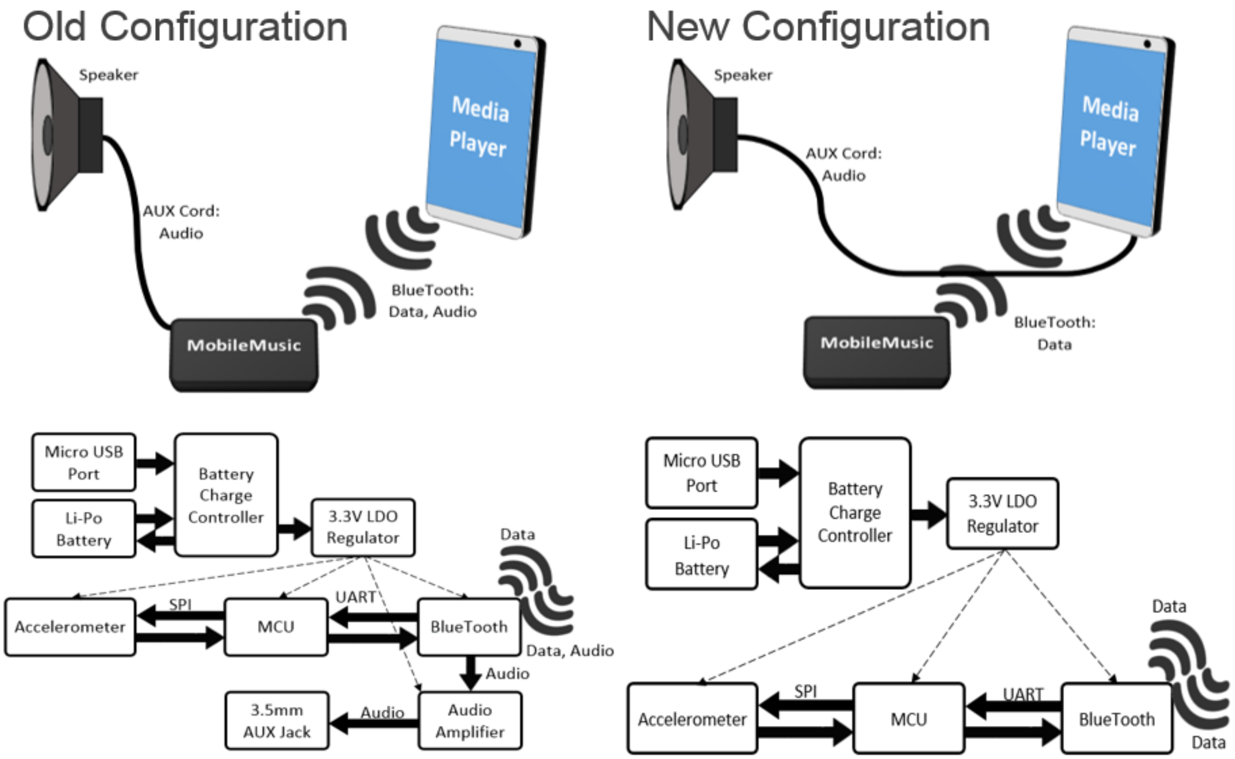


Figure 6.1 - Comparison between the New Configuration and the Old Configuration

This new configuration has multiple benefits for device operation. Since the audio amplifier is not being used in this configuration, it can be kept in shut down mode for the duration of operation. Since the audio amplifier sinks an estimated 30mA, this will greatly increase the battery life of the device from 15 hours to 24 hours. Also, since the BlueTooth player will no longer be streaming audio, it will require slightly less power, further increasing the battery life. Another benefit is that there would not need to be any cords attached to the device during operation. This would greatly reduce the risk of tripping due to a hanging wire. The only downside to this configuration is that the media player will have to be physically connected to a headphone or speaker.

Because of the jack detect, the current hardware is capable of both system configurations depending on the presence of an audio output in the 3.5mm phone jack. The code to do this, however, has not been developed. It would essentially be another state machine that would use the interrupt from the switch of the 3.5mm phone jack as an input. It would then control how the play or pause command is communicated to the media player. In the old configuration, this was done by sending a UART command to the RN-52 to send a play/pause toggle command to the media player over the AVRCP profile. This method would not work with the new configuration because the AVRCP profile is only active when the A2DP profile is active and the device is no longer streaming music. Instead, the device would send a symbol to the player through the SPP profile that it would decode as either a play command or a pause command. In order to receive and decipher the symbol, a mobile app would be needed for the media player.

6.2 – Mobile App Integration

Another future development is to incorporate a mobile app the device. This would greatly help to improve the accuracy of the motion sensing algorithm. Because we would no longer be confined to preset functions of the AVRCP profile, we could send commands to play and pause, not just a play/pause toggle. Since the device would be able to have exact control of the play/pause state of the media player, it would help correct, if not eliminate some of the errors that arise from never exactly knowing what state the media player is in. The app could also include a way for users to download custom threshold and delay values to the code. This would help physical therapists fine tune the motion sensing algorithm and increase it efficacy.

A mobile app would also enable data collection from the device. It could possibly use a graphical user interface to display acceleration data for the physical therapists to analyze and adjust the patient's gait training accordingly. The device could also record activity and inactivity times for a given therapy session. This could give the physical therapist feedback as to how efficient the training session was as well as a quantitative way of looking at patient progress over time.

Ideally, the app would be multiplatform meaning that it would work with both Android and iOS devices. However, due to licensing required for Apple apps, most prototyping and testing would be done on the android platform. Unfortunately, most of the media players used by Avalon Academy run iOS so they would be incompatible with the first revisions of the app. Towards the end of the project, we began working with a Computer Engineering Masters student, Anup Navare, on developing this application. So far, the app is only capable of sending and receiving UART communications sent through the BlueTooth SPP profile.

6.3 – Mechanical Design Updates

Future updates on the mechanical design are not really necessary, but could be a nice way to add an ergonomic touch. If we were to redesign the device, it would be nice to round the corners and edges with a small filet. This would make the device more pleasant to hold, and would prevent the cords from catching on the sharp edges. In the future, it would also be beneficial to finish the product in some sort of rubberized coating. The purpose of this coating would be to add to the shock protection of the device as well as to make the enclosure more water resistant by sealing up the spaces between the shell halves and the button. This coating would also make Mobile Music easier to grip, and less likely to slip from the user's hand and get broken.

From a manufacturing standpoint, this product would be better of being molded and machine finished. This would likely lead to a much more consistent product, which could more easily modified in terms of material choice and quantity produced. The parts are simple enough that molds could be fairly inexpensive to machine.

6.4 – Conclusions

Overall, Mobile Music has turned out to be a partially successful Senior Design Project. We were not able to complete all of our initial goals, but we were able to successfully implement and test many different aspects of our design. The parts which we were unable to complete were in the finer details with the exception of the bugs with the motion sensing algorithm and AVRCP commands.

That being said, a lot was accomplished over the course of the project. We designed and built a case to house the electrical components, and provide the user with an easy interface. We were not able to get to a point where the shell was water resistant, as the joints were not sealed, and the tolerance on the button was slightly off. In regards to the mounting system, we were essentially completely successful. We were able to design a mount that can work on almost any product, is totally scalable, and works quickly and effectively. The use of 3D printed parts also makes the design accessible to anyone who needs a solid pipe mounting system. Being only a two person team, we are happy with the progress that we made during the year.

References

“Data and Statistic.” *Centers for Disease Control and Prevention*. N.p., 2008. Web. 22 Oct. 2014. <http://www.cdc.gov/ncbddd/cp/data.html>

Moens, Bart, Leon Van Noorden, and Marc Leman. *D-Jogger: Syncing Music With Walking*. smcnetwork.org. N.p., 2010. Web. 15 Feb. 2015. <http://smcnetwork.org/files/proceedings/2010/66.pdf>.

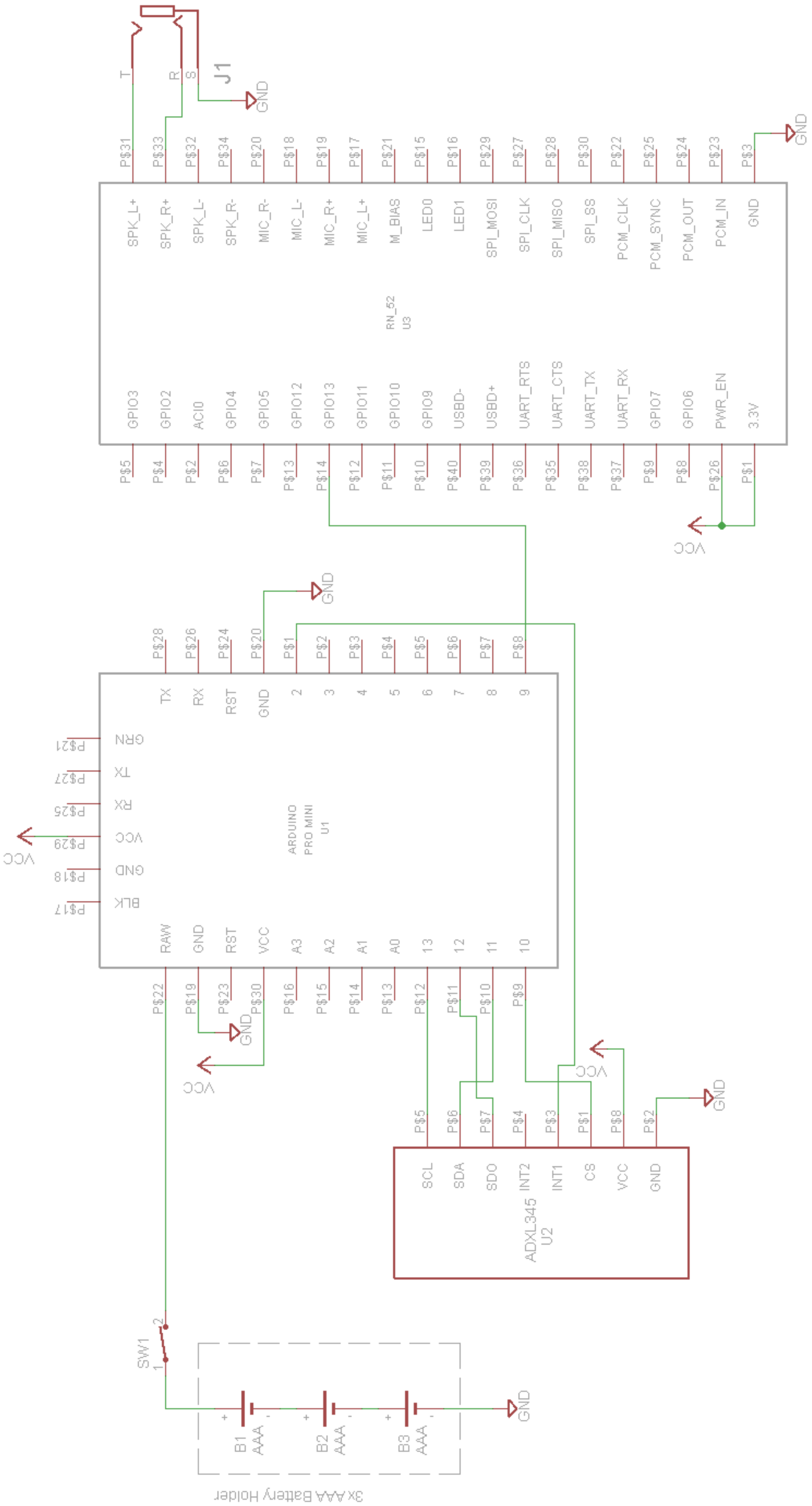
Kwak, Eunmi Emily. "Effect of Rhythmic Auditory Stimulation on Gait Performance in Children with Spastic Cerebral Palsy." *Journal of Music Therapy* 44.3 (2007): 198-216. *Oxford Journals*. Web. 28 Oct. 2014.

Appendices

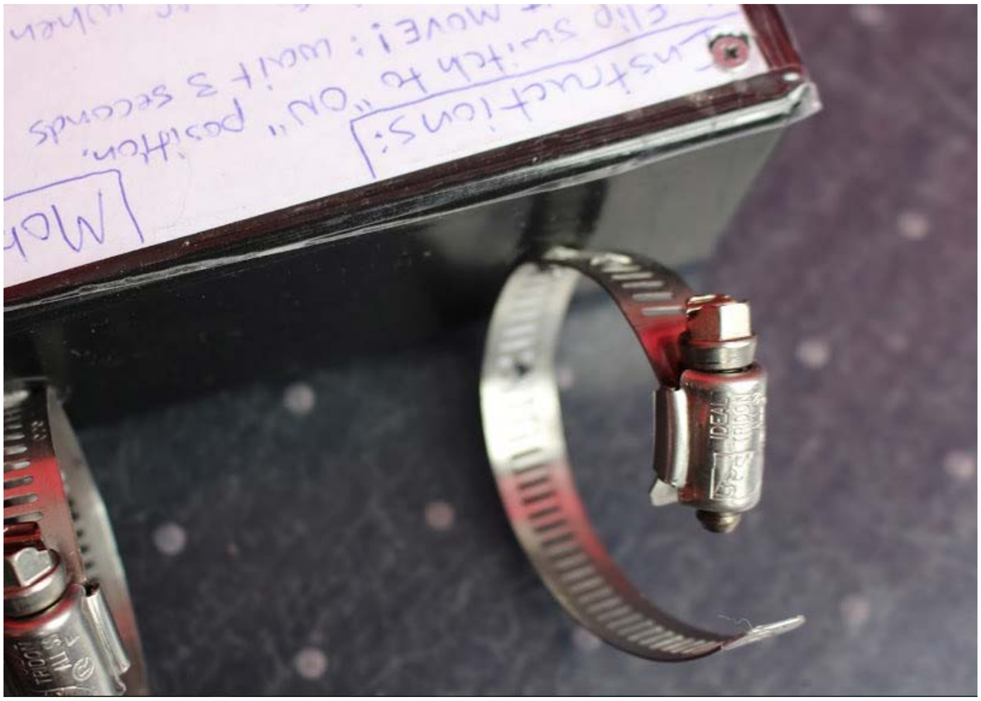
Appendix A – Prototype BOM

DESCRIPTION	**QUANTITY**	**PRICE/UNIT**	**SUPPLIER**
AAA Battery	3	$ 1.00	Amazon
3x AAA Battery Holder	1	$ 2.00	RadioShack
SPST Power Switch	1	$ 3.50	RadioShack
Arduino Pro Mini 3.3V	1	$ 10.00	Sparkfun Electronics
FTDI Breakout Board	1	$ 15.00	Sparkfun Electronics
ADXL345 Breakout Board	1	$ 18.00	Sparkfun Electronics
RN-52 Breakout Board	1	$ 45.00	Sparkfun Electronics
3.5mm phone Jack	1	$ 1.75	RadioShack
LED and Resistor	1	$ 1.50	Amazon
Wires	1	$ 5.00	Amazon
Project Enclosure	1	$ 4.99	RadioShack
2" Hose Clamps	2	$ 1.83	Home Depot
#6 1/2" Machine Screw	2	$ 1.67	Home Depot
#6 Lock Nut	2	$ 1.70	Home Depot
	TOTAL	**$ 120.14**	

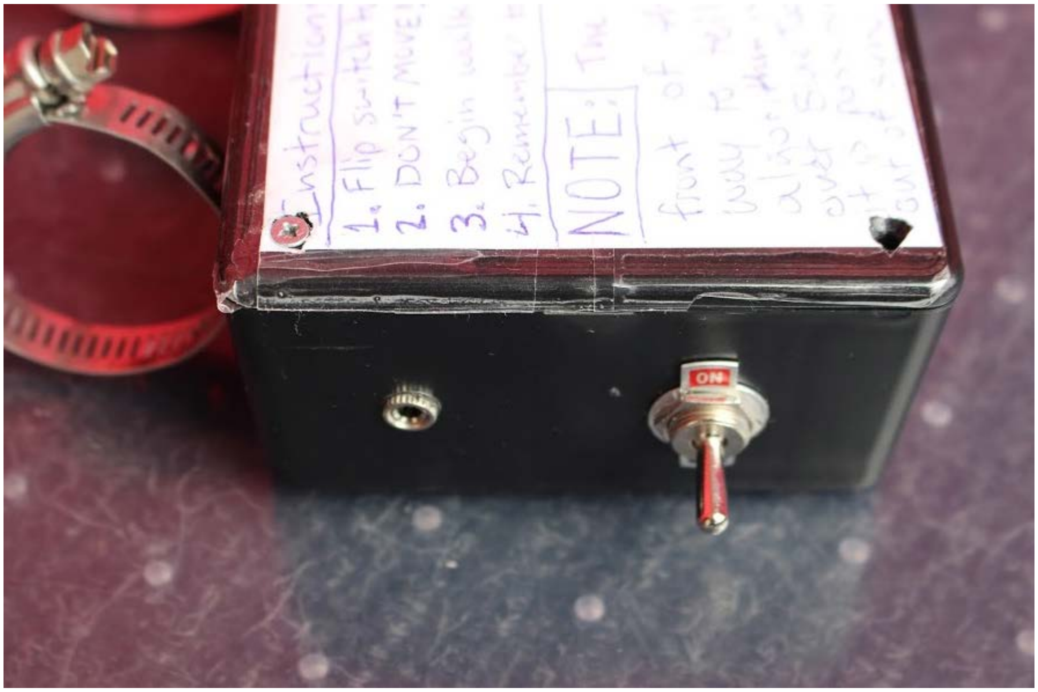
Appendix B – Prototype Schematic



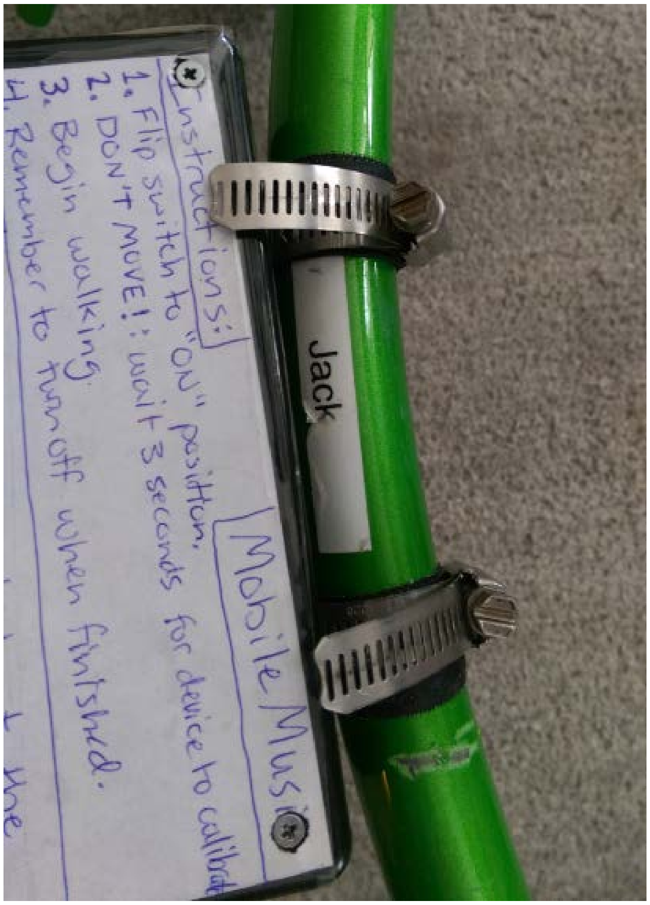
Appendix C – Prototype Pictures

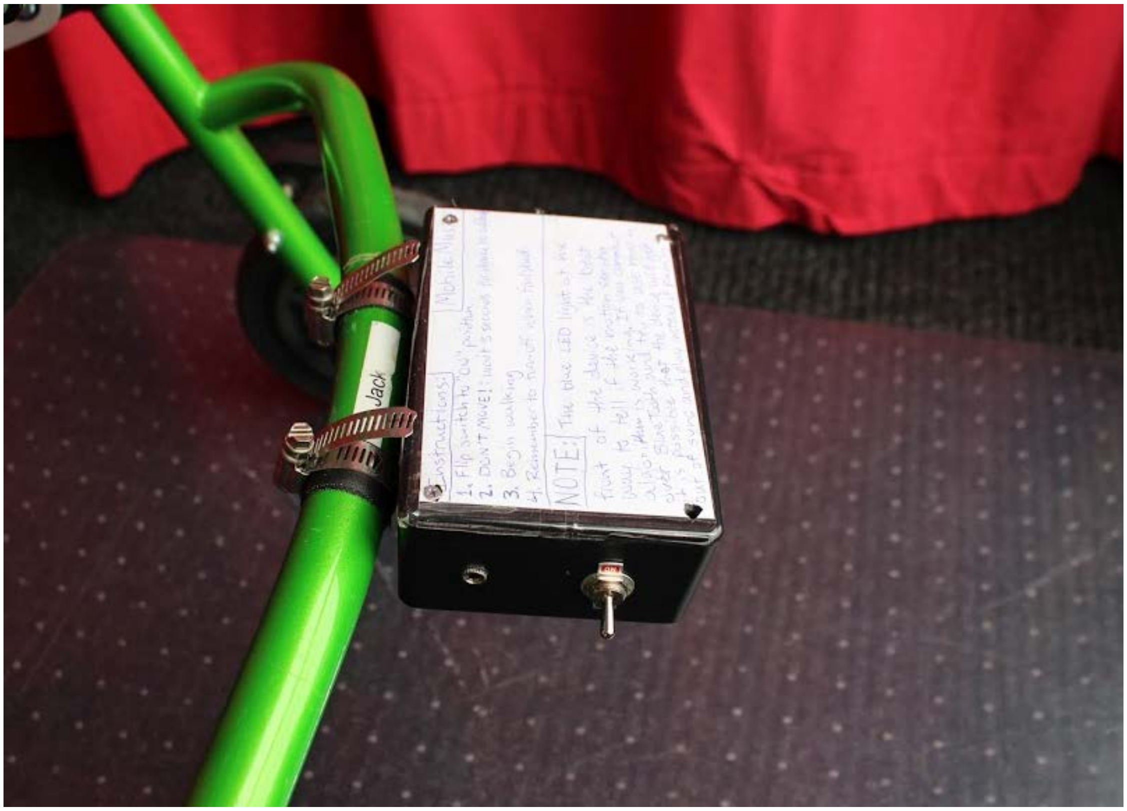


Prototype with Pipe clamps

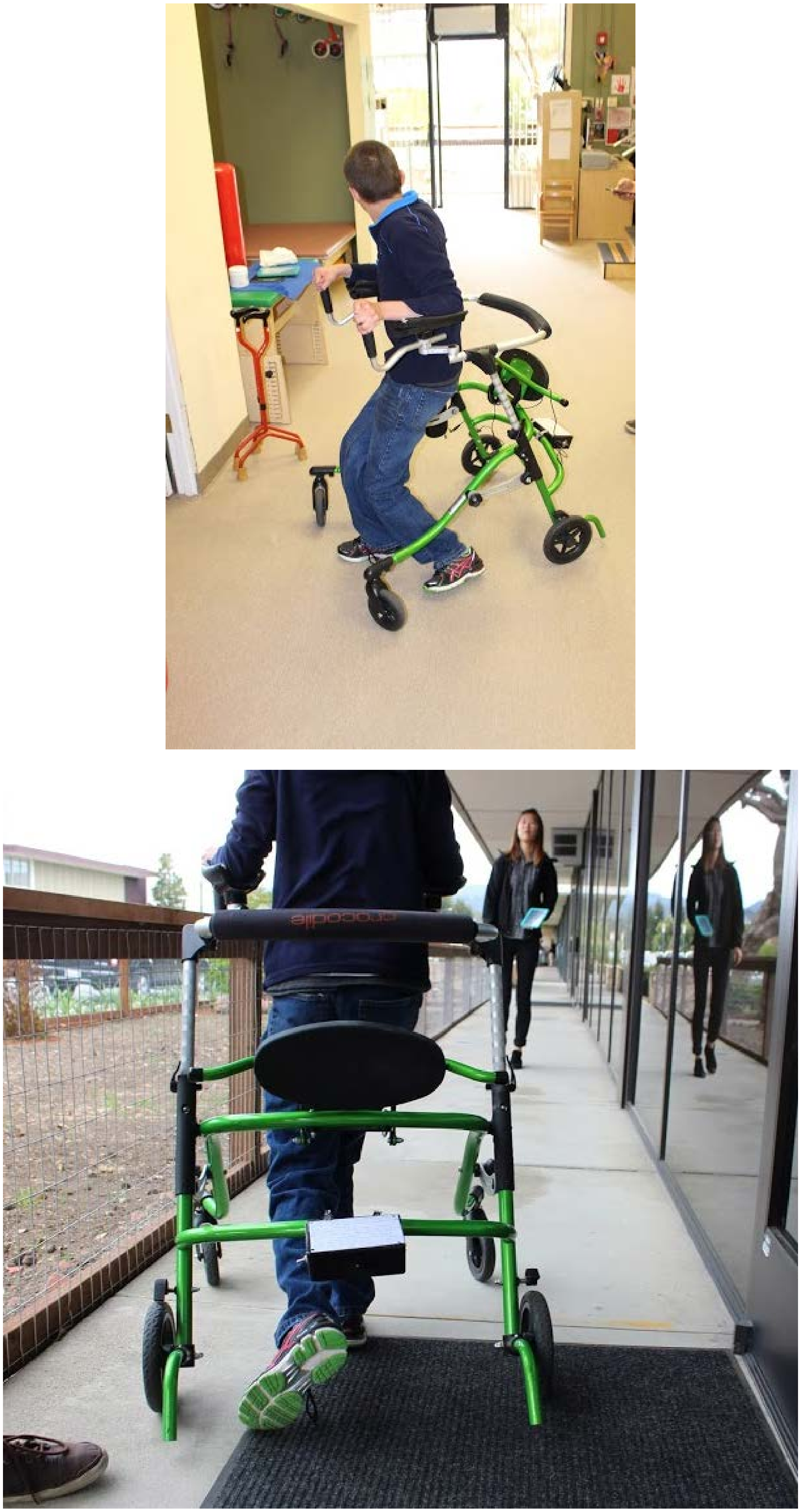


Peripherals

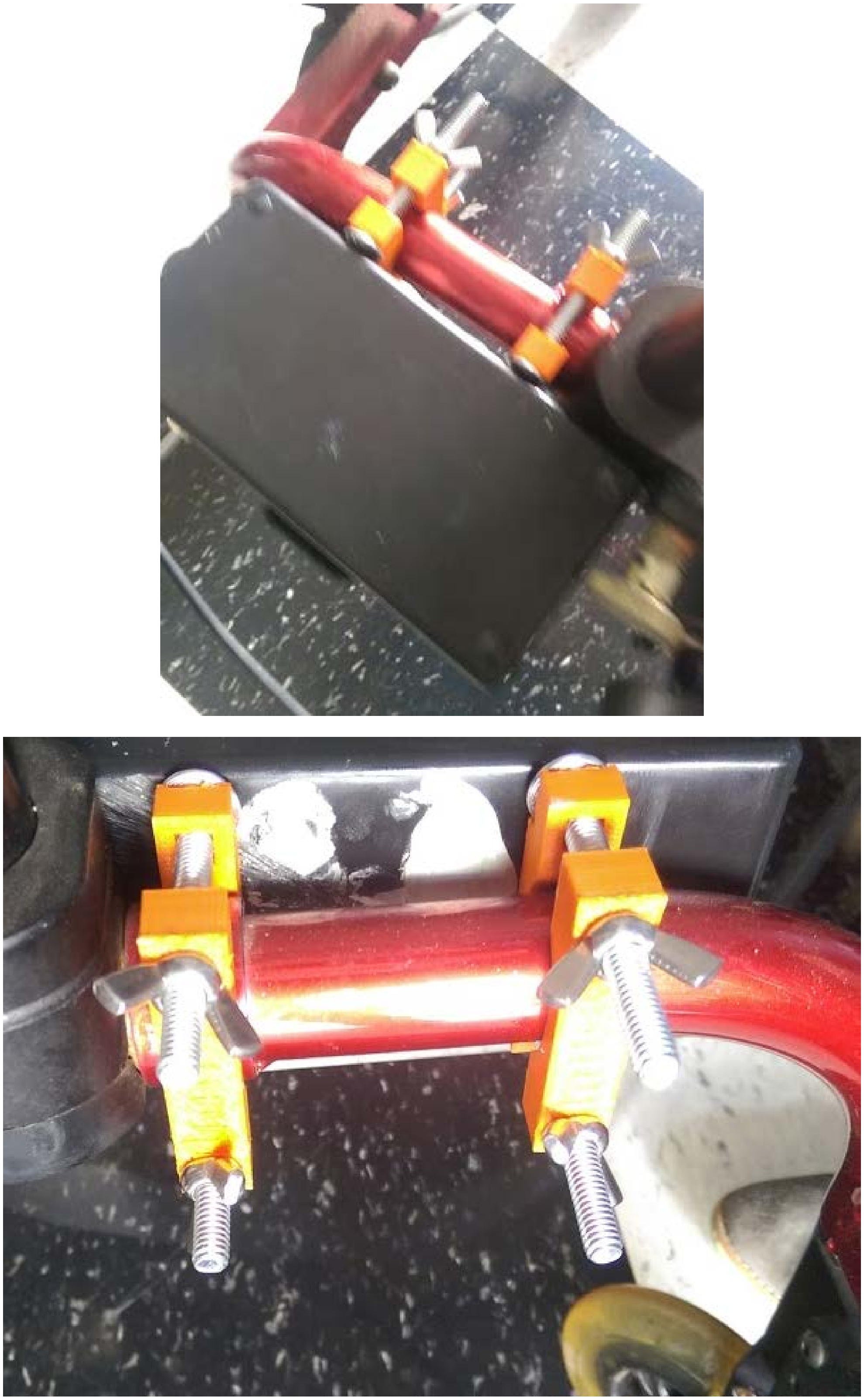

Instructions:
1. Flip switch to "ON" position.
2. DON'T MOVE!: wait 3 seconds for device to calibrate
3. Begin walking
4. Remember to turn off when finished.
Mobile Musi
Jack



Mounted Position



Prototype Testing



Final Mount Prototype

Appendix D – Prototype Code

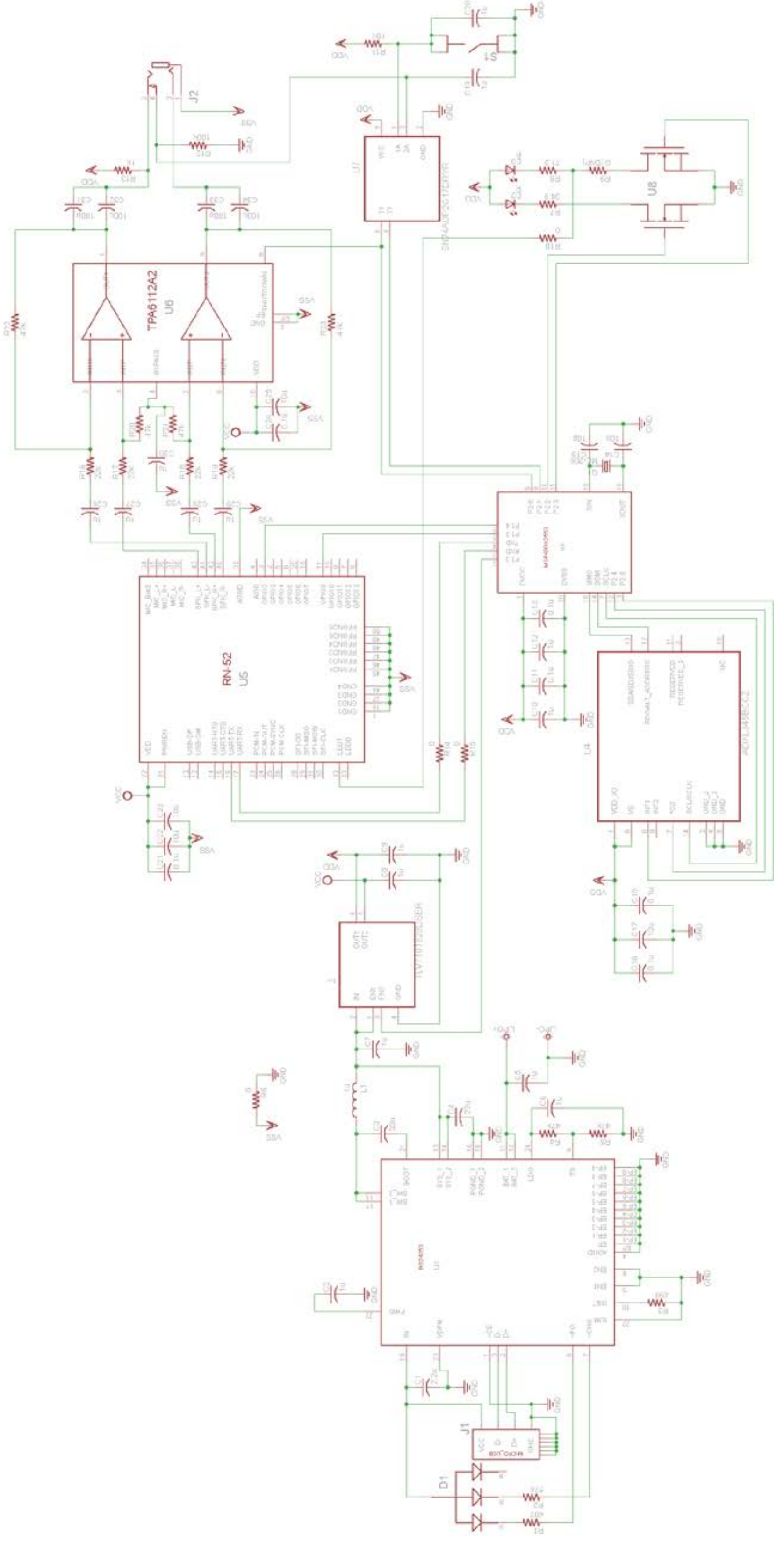
```
#include <SPI.h>
#include <math.h>
#include <Timer.h>

// TESTING
#define BUTTON_DELAY 50

// GENERAL
#define PLAYPAUSE_DELAY 100
#define CAL_DELAY       500
#define CAL_SAMPLES     200
#define HI_THRESH       30    // WILL NEED TO BE VARIABLE
#define LO_THRESH       8     // WILL NEED TO BE VARIABLE
#define INACT_DELAY     1000  // WILL NEED TO BE VARIABLE

// ADXL345 REGISTERS
#define BW_RATE         0x2C      //Data Rate Register
#define POWER_CTL       0x2D      //Power Control Register
#define INT_ENABLE      0x2E      //Interrupt Enable Register
#define INT_MAP         0x2F      //Interrupt Mapping Register
#define INT_SOURCE      0x30      //Interrupt Source Register
#define DATA_FORMAT     0x31       //Data Formatting Register
#define DATAX0          0x32      //X-Axis Data 0
#define DATAX1          0x33      //X-Axis Data 1
#define DATAY0          0x34      //Y-Axis Data 0
#define DATAY1          0x35      //Y-Axis Data 1
#define DATAZ0          0x36      //Z-Axis Data 0
#define DATAZ1          0x37      //Z-Axis Data 1

// PINS
#define CS  10  //Chip Select pin
#define GPIO13  9
#define LED 8

// INTERRUPTS
volatile int DATA_INT = 1;
volatile int FLAG = 0;

// FLAGS & QUEUES // NEED TO BE IMPLEMENTED TO PREVENT PLAY/PAUSE TOGGLE
MISHAPS
int playPauseFlag = 0;
int playPauseQueue = 0;

// TIMERS // NEED TO BE IMPLEMENTED (SEND INFO OVER SPP)
unsigned long activeTime = 0;
unsigned long inactiveTime = 0;
unsigned long startTime = 0;

// STATES
enum MVMNTstate_t {MOTION, NO_MOTION} MVMNTstate = NO_MOTION;
enum PPstate_t {PLAY, PAUSE} PPstate = PAUSE;
enum SYSstate_t {ON, CAL, SLEEP} SYSstate = CAL;
enum BTstate_t {OFF, DISCONNECTED, /*PAIRING,*/ CONNECTED} BTstate =
DISCONNECTED;
```

```
// OTHER
double deltaTheta;
Timer t;

void setup(){
  // initiate SPI.
  SPI.begin();
  SPI.setDataMode(SPI_MODE3);

  Serial.begin(115200);

  pinMode(CS, OUTPUT);
  digitalWrite(CS, HIGH);
  pinMode(GPIO13, OUTPUT);
  digitalWrite(GPIO13, HIGH);
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);

  attachInterrupt(0, dataAvail, RISING);

  writeRegister(DATA_FORMAT, 0x0B);  // +/- 16G range w/ full res
  writeRegister(BW_RATE, 0x0B);      // RATE:  200 Hz
  writeRegister(INT_ENABLE, 0x80);   // enable DATA_READY INT
  writeRegister(INT_MAP, 0x00);      // map DATA_READY to INT1
  writeRegister(POWER_CTL, 0x08);    // measurement mode
}

void loop()
{
  if(SYSstate == CAL) // switch statement instead of if statements
  {
    deltaTheta = calibration();
    SYSstate = ON;

    //Serial.print("Delta Theta = ");       // for testing/debug
    //Serial.println(deltaTheta);           // for testing/debug
  }
  if(SYSstate == ON)
  {
    // Serial.println("DATA");               // for testing/debug
    // Serial.println(getXYMag(deltaTheta)); // for testing/debug
    motionSense(deltaTheta);

    switch(MVMNTstate) // could get rid of global enum; make local to loop(),
motionSense() output 1 for motion 0 for no motion
    {
      case NO_MOTION:
        if((PPstate == PLAY) && (millis() - startTime > INACT_DELAY))
        {
          PPstate = PAUSE;
          //playPause();
          t.pulse(GPIO13, PLAYPAUSE_DELAY, HIGH);
          digitalWrite(LED, LOW);
          //Serial.println("PAUSE");
        }
        break;
      case MOTION:
        if(PPstate == PAUSE)
        {
          //playPause();
          digitalWrite(LED, HIGH);
          t.pulse(GPIO13, PLAYPAUSE_DELAY, HIGH);
          //Serial.println("PLAY");
        }
        PPstate = PLAY;
        startTime = millis();
        break;
      default:
        MVMNTstate = NO_MOTION;
        break;
    }
  }

  t.update();
}

void writeRegister(char registerAddress, char value){
  // set CS pin low to start a SPI packet
  digitalWrite(CS, LOW);
  // transfer the register address over SPI
  SPI.transfer(registerAddress);
  // transfer the desired register value over SPI
  SPI.transfer(value);
  // set the CS pin high to end the SPI packet
  digitalWrite(CS, HIGH);
}

void readRegister(char registerAddress, int numBytes, unsigned char *
values){
  // set bit 7 to read
  char address = 0x80 | registerAddress;
  // if multibyte read, set bit six
  if(numBytes > 1)
    address = address | 0x40;
  // set the CS pin low to start a SPI packet
  digitalWrite(CS, LOW);
  // transfer starting address
  SPI.transfer(address);
  // read specified registers
  for(int i=0; i<numBytes; i++)
    values[i] = SPI.transfer(0x00);
  // set the CS pin high to end the SPI packet
  digitalWrite(CS, HIGH);
}

void dataAvail(void)
{
  DATA_INT = 1;
}

double calibration(void)
{
  double calX = 0.0;
  double calY = 0.0;
  double calZ = 0.0;
  double calR = 0.0;
  int count = 0;
  unsigned char values[6];
  unsigned long startTime = millis();

  while((millis() - startTime) < CAL_DELAY); // possible problem when
millis() resets...

  while(count < CAL_SAMPLES)
  {
    if(DATA_INT)
    {
      DATA_INT = 0;
      count++;
      readRegister(DATAX0, 6, values);
      calX += ((int)values[1]<<8)|(int)values[0];
      calY += ((int)values[3]<<8)|(int)values[2];
      calZ += ((int)values[5]<<8)|(int)values[4];
      //Serial.println(count);
    }
  }

  calX = (double)calX/CAL_SAMPLES;
  calY = (double)calY/CAL_SAMPLES;
  calZ = (double)calZ/CAL_SAMPLES;
  calR = sqrt((calZ*calZ) + (calY*calY) + (calX*calX));
  return acos(calZ/calR);
}

double getXYMag(double deltaTheta)
{
  double XYMag = 0.0;
  double x = 0.0;
  double y = 0.0;
  double z = 0.0;
  double r = 0.0;
  unsigned char values[6];

  while(!DATA_INT);

  DATA_INT = 0;
  readRegister(DATAX0, 6, values);
  x = (double)(((int)values[1]<<8)|(int)values[0]);
  y = (double)(((int)values[3]<<8)|(int)values[2]);
  z = (double)(((int)values[5]<<8)|(int)values[4]);
  r = sqrt((x*x) + (y*y) + (z*z));
  XYMag = r*sin(acos(z/r) - deltaTheta);
  return abs(XYMag);
}

double motionSense(double deltaTheta)
{
  double XYMag;

  XYMag = getXYMag(deltaTheta);
  if(((XYMag >= HI_THRESH) && (PPstate == PAUSE)) || ((XYMag >= LO_THRESH) &&
(PPstate == PLAY)))
    MVMNTstate = MOTION;
  else
    MVMNTstate = NO_MOTION;
  return XYMag;
}

void playPause(void)
{
  digitalWrite(GPIO13, LOW);
  delay(BUTTON_DELAY);
  digitalWrite(GPIO13, HIGH);
  delay(BUTTON_DELAY);
}
```

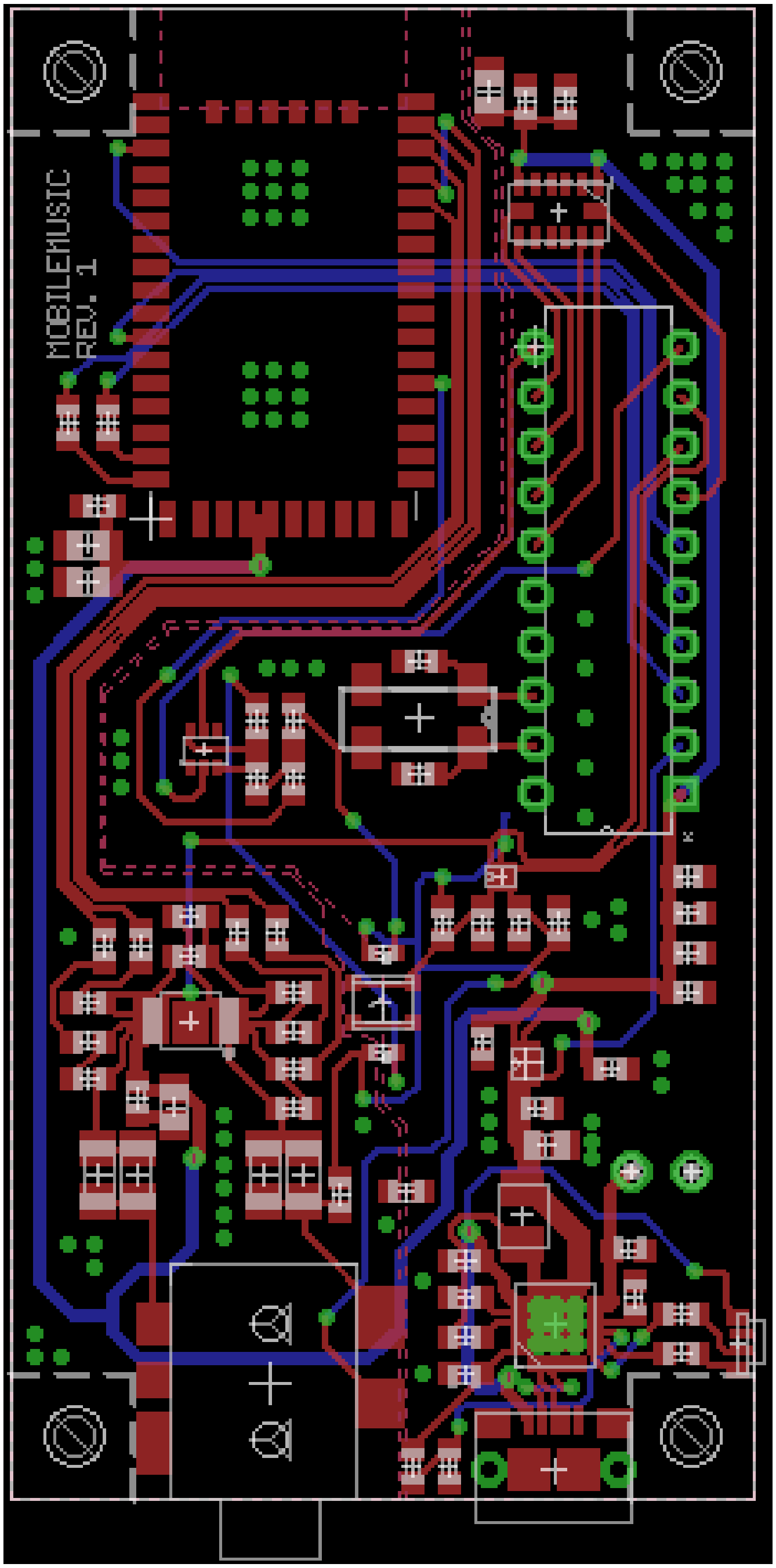
Appendix E – Final Build BOM

PART NUMBER	QUANTITY	PRICE/UNIT	REFERENCE	DESCRIPTION
258	1	$ 9.95	B1	LiPo Battery, 3.7V 1200mAh
587-1253-1-ND	1	$ 0.13	C1	2.2uF 10V X5R 0603
490-1532-1-ND	6	$ 0.10	C11, C13, C16, C18, C21, C24	0.1uF 16V X7R 0603
712-1307-1-ND	2	$ 0.50	C14, C15	10pF 250V NPO 0603
490-7207-1-ND	4	$ 0.26	C17, C22 , C23, C25	10uF 6.3V X5R 0805
1276-1946-1-ND	15	$ 0.04	C2, C5, C6, C7, C8, C9, C10, C12, C19, C20, C26, C27, C28, C29, C30	1uF 10V X7R 0603
1276-2042-1-ND	1	$ 0.10	C3	33nF 50V X7R 0603
490-4539-1-ND	4	$ 0.81	C31, C32, C33, C34	100uF 6.3V X5R 1206
490-1719-1-ND	1	$ 0.27	C4	22uF 6.3V X5R 0805
MSL0201RGBW1CT-ND	1	$ 1.80	D1	RGB LED
160-1830-1-ND	1	$ 0.43	D2	RED LED
511-1589-1-ND	1	$ 0.50	D3	BLUE LED
609-4616-1-ND	1	$ 0.46	J1	USB power jack
CP1-3514SJCT-ND	1	$ 1.70	J2	Headphone jack w/ switch
AE9998-ND	1	$ 0.29	J3	20 pin DIP socket
587-2164-1-ND	1	$ 0.28	L1	1uH 2.2A 51.6mOhm
300-8742-1-ND	1	$ 0.96	Q1	32.768kHz Crystal
RMCF0603FT402RCT-ND	1	$ 0.10	R1	402 1% 1/10W 0603
MCT0603-10.0K-CFCT-ND	1	$ 0.08	R11	10K 1% 1/8W 0603
MCT0603-100K-CFCT-ND	1	$ 0.08	R12	100k 1% 1/8W 0603
MCT0603-1.00K-CFCT-ND	1	$ 0.08	R13	1k 1% 1/8W 0603
MCT0603-22.0K-CFCT-ND	4	$ 0.08	R16, R17, R18, R19	22k 1% 1/8W 0603
P576HCT-ND	1	$ 0.10	R2	576 1% 1/10W 0603
P499HCT-ND	1	$ 0.10	R3	499 1% 1/10W 0603
MCT0603-47.0K-CFCT-ND	6	$ 0.08	R4, R5, R20, R21, R22, R23	47k 1% 1/8W 0603
MCT0603-0.0-ZZCT-ND	5	$ 0.13	R6, R9, R10, R14, R15	0.0 Jumper 1/8W 0603
P24.9HCT-ND	1	$ 0.10	R7	24.9 1% 1/10W 0603
P71.5HCT-ND	1	$ 0.10	R8	71.5 1% 1/10W 0603
CKN10290CT-ND	1	$ 0.33	S1	Momentary switch
296-36446-1-ND	1	$ 3.32	U1	LiPo charge regulator
296-28766-1-ND	1	$ 0.74	U2	2 CH LDO
296-28429-5-ND	1	$ 2.80	U3	MSP430 MCU
ADXL345BCCZ-ND	1	$ 7.32	U4	ADXL345
RN52-I/RM-ND	1	$ 23.30	U5	RN-52
296-10848-1-ND	1	$ 2.40	U6	Audio Amplifier
296-27365-1-ND	1	$ 0.55	U7	Schmitt trigger, 2CH
UM6K1NTNCT-ND	1	$ 0.56	U8	2 CH. NFET
MOBILE_MUSIC_PCB_REV1	1	$ 33.00	N/A	PCB
B20001	1	$ 2.60	B1	Shell Base Style 2
T20001	1	$ 2.24	T1	Shell Top Style 2
C20001	1	$ 0.85	N/A	Battery Cover Style 2
M20001	1	$ 1.15	N/A	Sliding Mount, Inside
M20002	1	$ 1.15	N/A	Sliding Mount, Outside
M20003	4	$ 0.20	N/A	Sliding Mount, Carriage Bolt
M20004	4	$ 1.18	N/A	Sliding Mount, Wing Nut
B20002	1	$ 0.25	N/A	Button, Style 2
S20001	5	$ 0.29	N/A	#4 Self Tapping Screws
	TOTAL	**$ 115.07**		

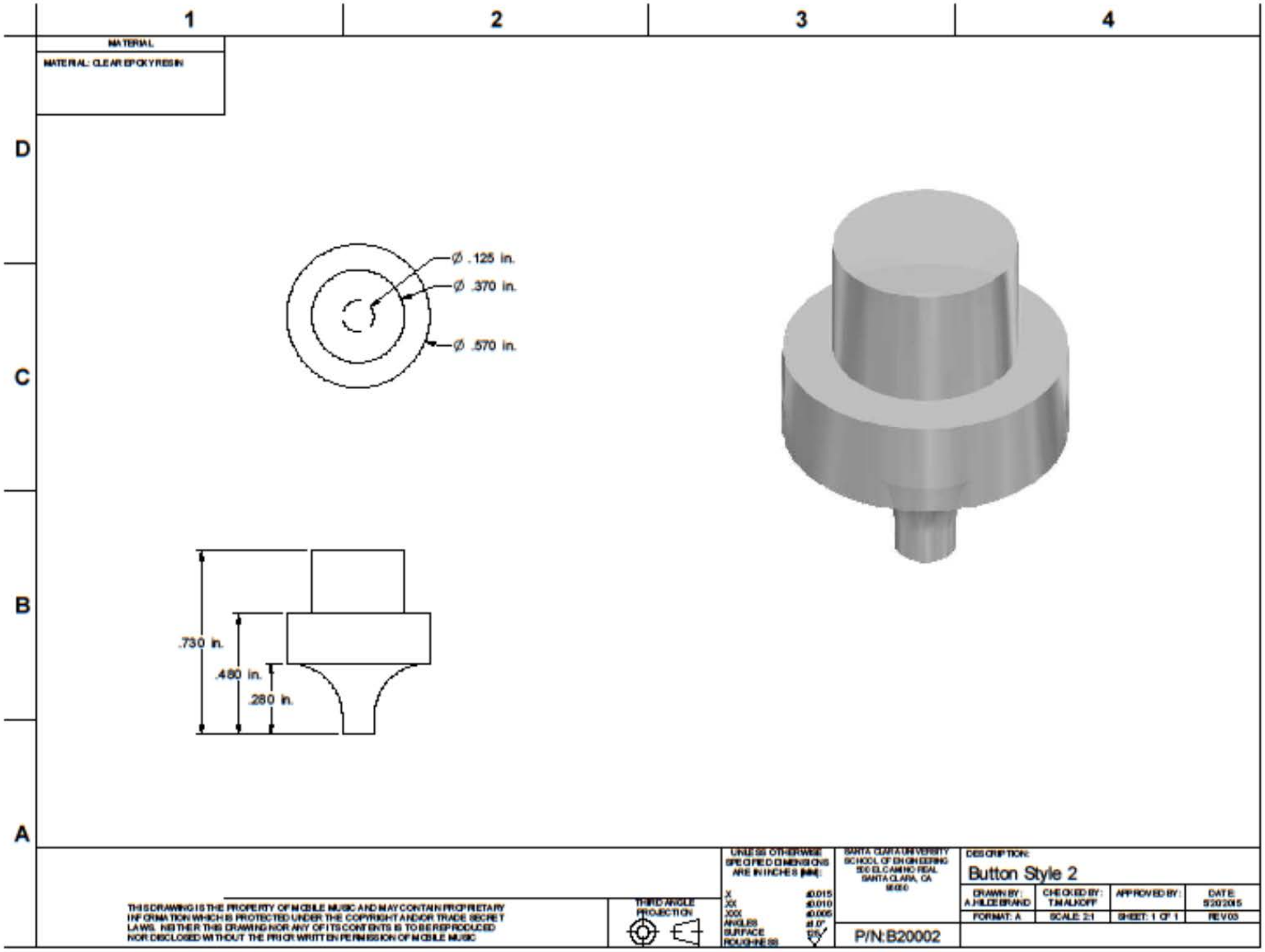
Appendix F – Final Build Schematic

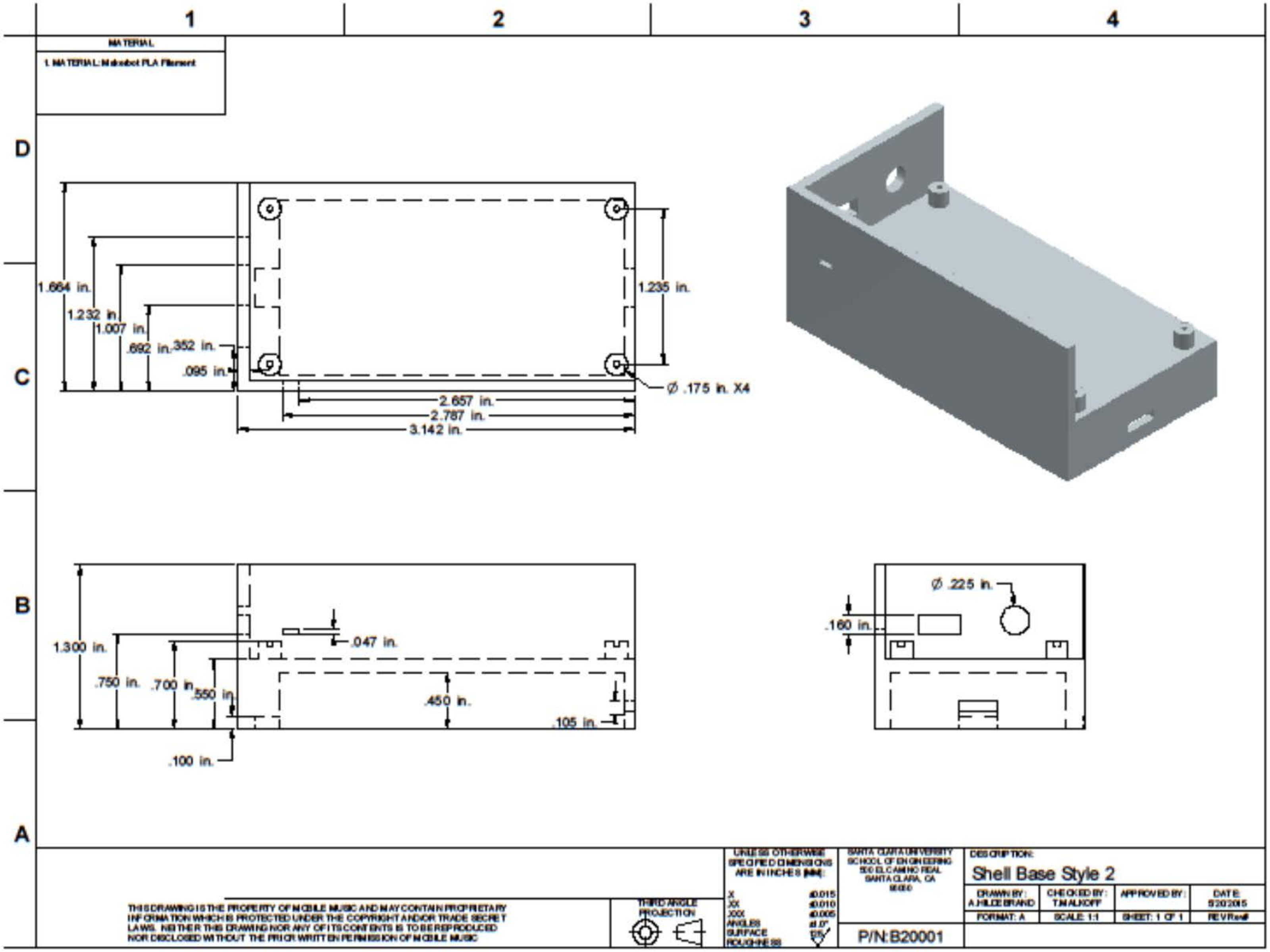


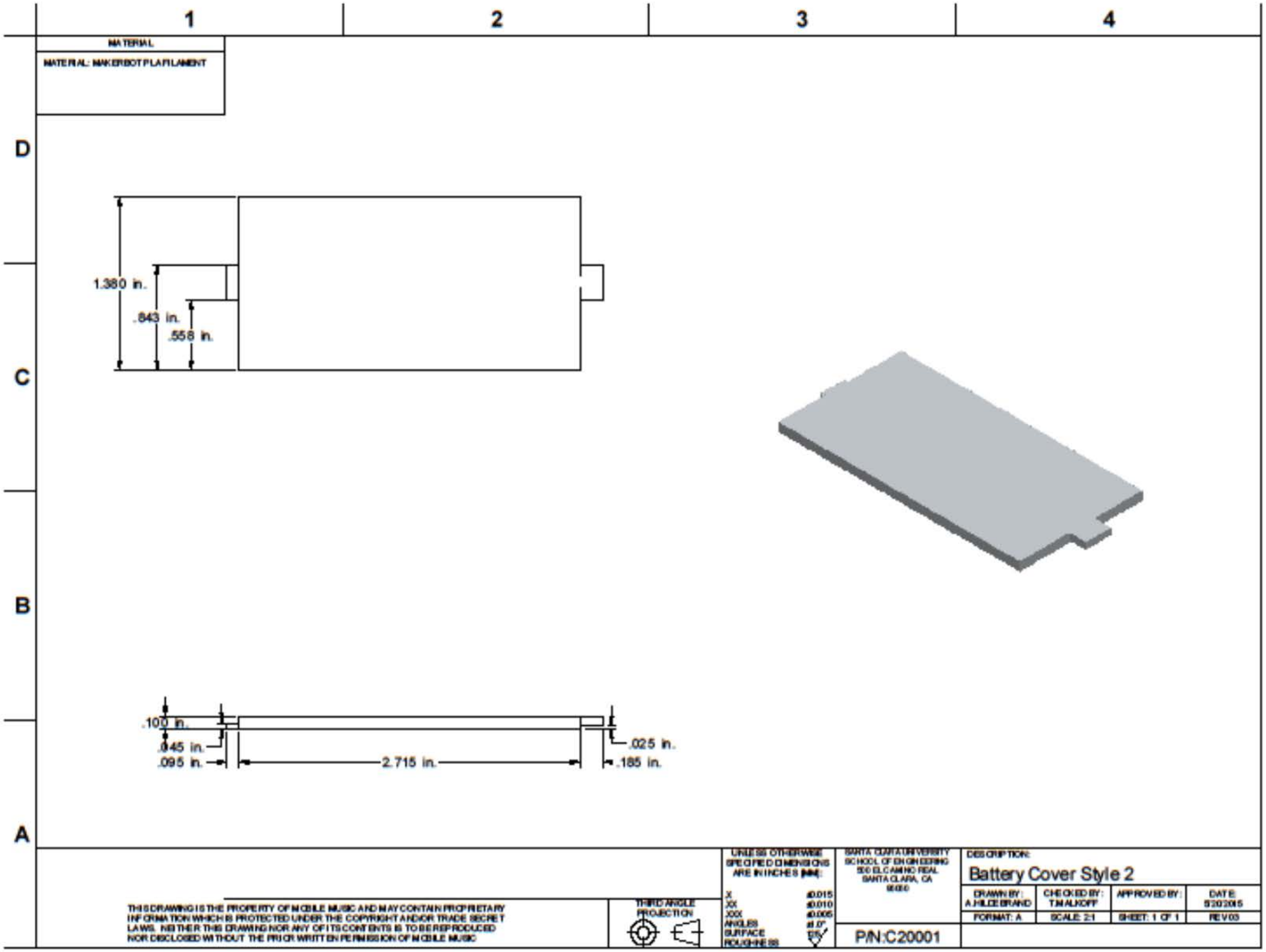
Appendix G – Final Build Layout

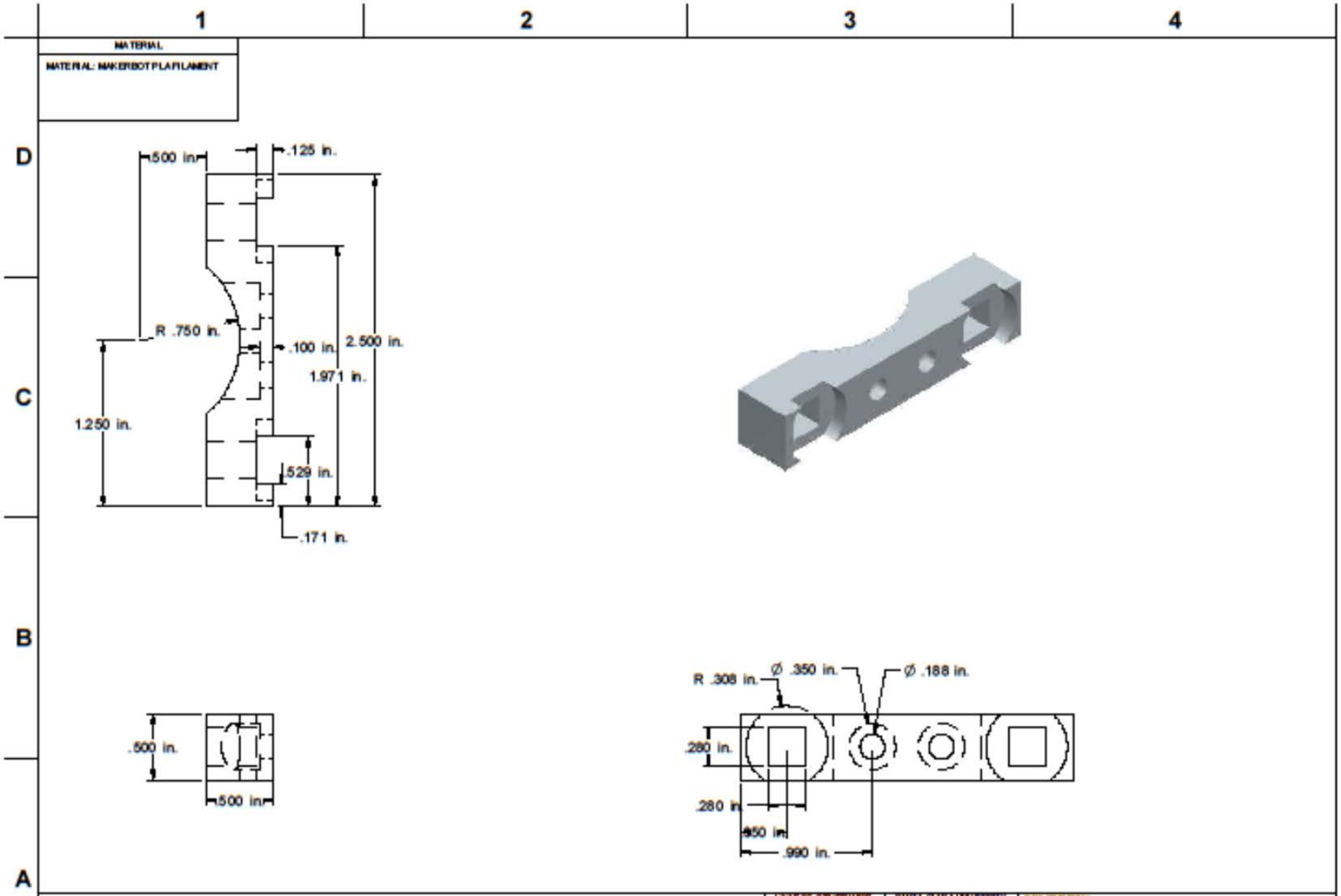


Appendix H – Final Mechanical Drawings




1
2
3
4
D
C
B
A
MATERIAL
1. MATERIAL: Makerbot PLA Filament
1.664 in.
1.232 in.
1.007 in.
.692 in.
.352 in.
.095 in.
1.235 in.
Ø .175 in. X4
2.657 in.
2.787 in.
3.142 in.
1.300 in.
.750 in.
.700 in.
.550 in.
.100 in.
.047 in.
.450 in.
.105 in.
Ø .225 in.
.160 in.
THIS DRAWING IS THE PROPERTY OF MOBILE MUSIC AND MAY CONTAIN PROPRIETARY INFORMATION WHICH IS PROTECTED UNDER THE COPYRIGHT AND/OR TRADE SECRET LAWS. NEITHER THIS DRAWING NOR ANY OF ITS CONTENTS IS TO BE REPRODUCED NOR DISCLOSED WITHOUT THE PRIOR WRITTEN PERMISSION OF MOBILE MUSIC
THIRD ANGLE PROJECTION
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES [MM]:
.X ±0.015
.XX ±0.010
.XXX ±0.005
ANGLES ±1.0°
SURFACE ROUGHNESS 125
SANTA CLARA UNIVERSITY
SCHOOL OF ENGINEERING
500 EL CAMINO REAL
SANTA CLARA, CA
95053
P/N:B20001
DESCRIPTION:
Shell Base Style 2
DRAWN BY: A.HILDEBRAND
CHECKED BY: T.MALKOFF
APPROVED BY:
DATE: 5/20/2015
FORMAT: A
SCALE: 1:1
SHEET: 1 OF 1
REV:Rev#


1
2
3
4
D
C
B
A
MATERIAL
MATERIAL: MAKERBOT PLA FILAMENT
1.380 in.
.843 in.
.558 in.
.100 in.
.045 in.
.095 in.
2.715 in.
.025 in.
.185 in.
THIS DRAWING IS THE PROPERTY OF MOBILE MUSIC AND MAY CONTAIN PROPRIETARY INFORMATION WHICH IS PROTECTED UNDER THE COPYRIGHT AND/OR TRADE SECRET LAWS. NEITHER THIS DRAWING NOR ANY OF ITS CONTENTS IS TO BE REPRODUCED NOR DISCLOSED WITHOUT THE PRIOR WRITTEN PERMISSION OF MOBILE MUSIC
THIRD ANGLE PROJECTION
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES [MM]:
X ±0.015
.XX ±0.010
.XXX ±0.005
ANGLES ±1.0°
SURFACE ROUGHNESS 125
SANTA CLARA UNIVERSITY
SCHOOL OF ENGINEERING
500 EL CAMINO REAL
SANTA CLARA, CA
95053
P/N:C20001
DESCRIPTION:
Battery Cover Style 2
DRAWN BY: A.HILDEBRAND
CHECKED BY: T.MALKOFF
APPROVED BY:
DATE 5/20/2015
FORMAT: A
SCALE: 2:1
SHEET: 1 OF 1
REV 03



MATERIAL

MATERIAL: MAKERBOT PLA FILAMENT

THIS DRAWING IS THE PROPERTY OF MOBILE MUSIC AND MAY CONTAIN PROPRIETARY INFORMATION WHICH IS PROTECTED UNDER THE COPYRIGHT AND/OR TRADE SECRET LAWS. NEITHER THIS DRAWING NOR ANY OF ITS CONTENTS IS TO BE REPRODUCED NOR DISCLOSED WITHOUT THE PRIOR WRITTEN PERMISSION OF MOBILE MUSIC

THIRD ANGLE PROJECTION

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES [MM]:

X	±0.015
.XX	±0.010
.XXX	±0.005
ANGLES	±1.0°
SURFACE ROUGHNESS	125√

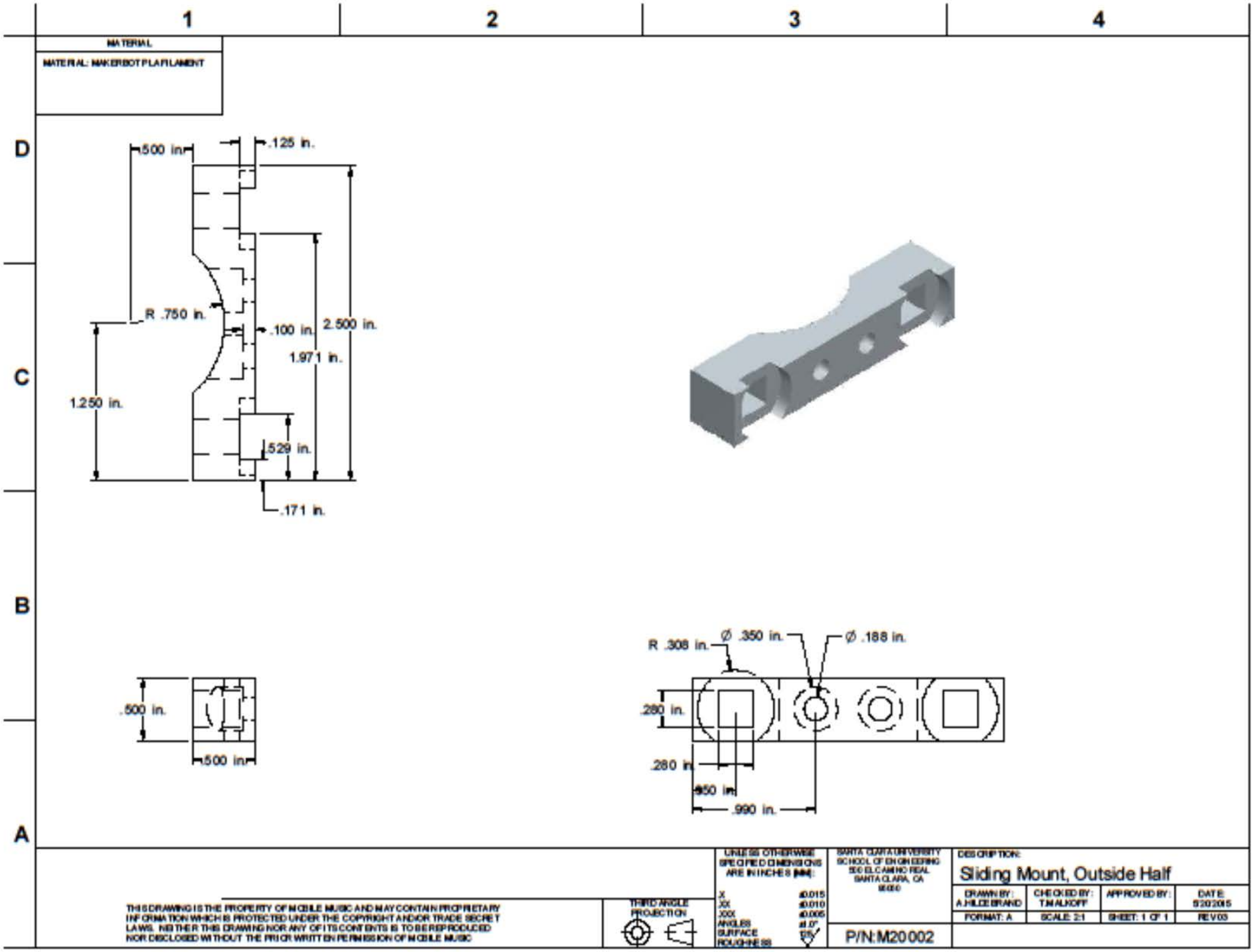
SANTA CLARA UNIVERSITY
SCHOOL OF ENGINEERING
500 EL CAMINO REAL
SANTA CLARA, CA
95050

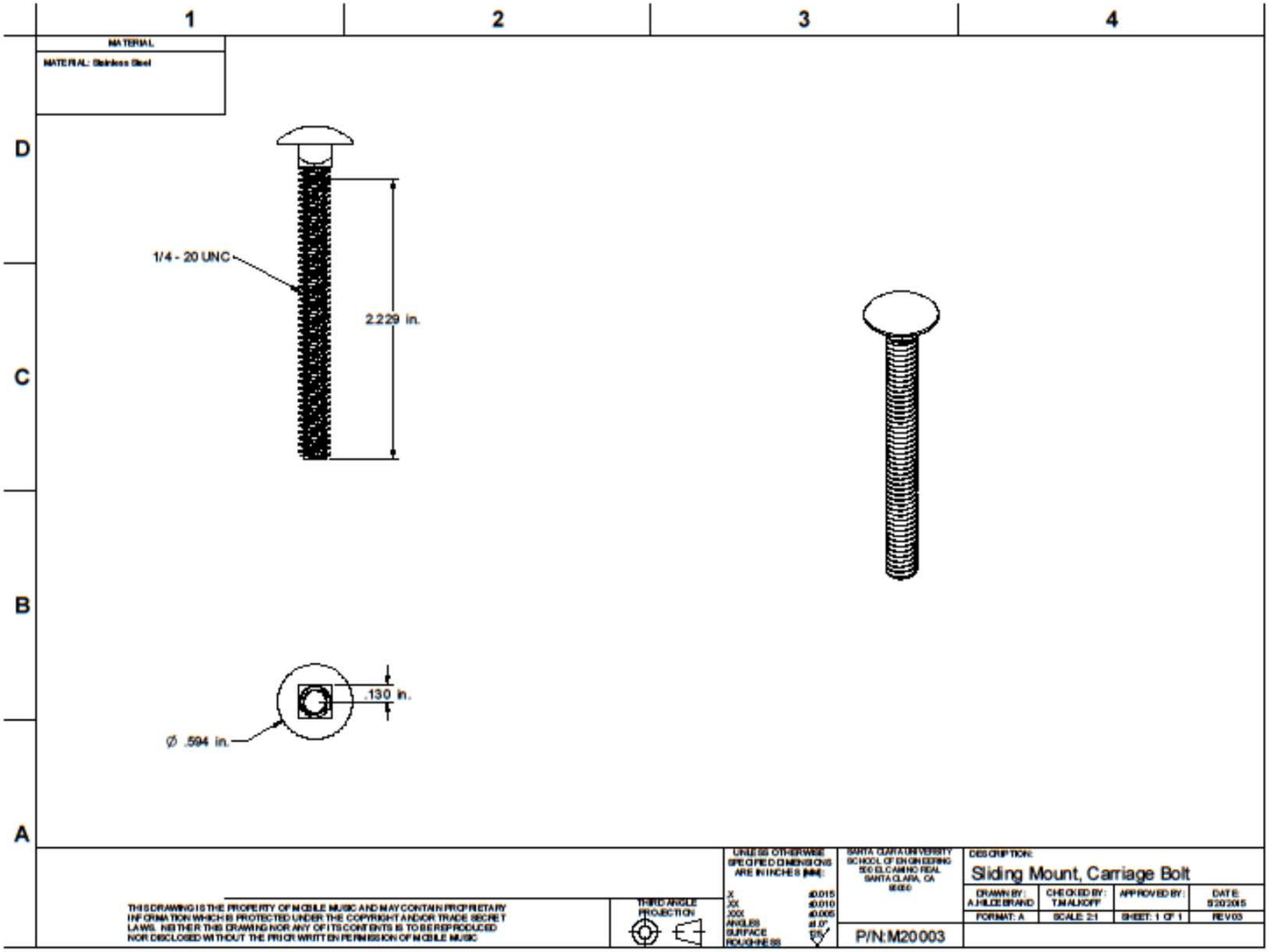
P/N:M20001

DESCRIPTION:

Sliding Mount, Inside Half

DRAWN BY:	CHECKED BY:	APPROVED BY:	DATE:
A.HILDEBRAND	T.MALKOFF		5/20/2015
FORMAT: A	SCALE: 2:1	SHEET: 1 OF 1	REV 03


1
2
3
4
D
C
B
A
MATERIAL
MATERIAL: MAKERBOT PLA FILAMENT
.500 in.
.125 in.
R .750 in.
.100 in.
2.500 in.
1.971 in.
1.250 in.
.529 in.
.171 in.
.500 in.
.500 in.
R .308 in.
Ø .350 in.
Ø .188 in.
.280 in.
.280 in.
.650 in.
.990 in.
THIS DRAWING IS THE PROPERTY OF MOBILE MUSIC AND MAY CONTAIN PROPRIETARY INFORMATION WHICH IS PROTECTED UNDER THE COPYRIGHT AND/OR TRADE SECRET LAWS. NEITHER THIS DRAWING NOR ANY OF ITS CONTENTS IS TO BE REPRODUCED NOR DISCLOSED WITHOUT THE PRIOR WRITTEN PERMISSION OF MOBILE MUSIC
THIRD ANGLE PROJECTION
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES [MM]:
.X ±0.015
.XX ±0.010
.XXX ±0.005
ANGLES ±1.0°
SURFACE ROUGHNESS 125
SANTA CLARA UNIVERSITY
SCHOOL OF ENGINEERING
500 EL CAMINO REAL
SANTA CLARA, CA
95053
P/N:M20002
DESCRIPTION:
Sliding Mount, Outside Half
DRAWN BY: A.HILDEBRAND
CHECKED BY: T.MALKOFF
APPROVED BY:
DATE: 5/20/2015
FORMAT: A
SCALE: 2:1
SHEET: 1 OF 1
REV 03



MATERIAL

MATERIAL: Stainless Steel

THIS DRAWING IS THE PROPERTY OF MOBILE MUSIC AND MAY CONTAIN PROPRIETARY INFORMATION WHICH IS PROTECTED UNDER THE COPYRIGHT AND/OR TRADE SECRET LAWS. NEITHER THIS DRAWING NOR ANY OF ITS CONTENTS IS TO BE REPRODUCED NOR DISCLOSED WITHOUT THE PRIOR WRITTEN PERMISSION OF MOBILE MUSIC

THIRD ANGLE PROJECTION

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES [MM]:

.X	±0.015
.XX	±0.010
.XXX	±0.005
ANGLES	±1.0°
SURFACE ROUGHNESS	125√

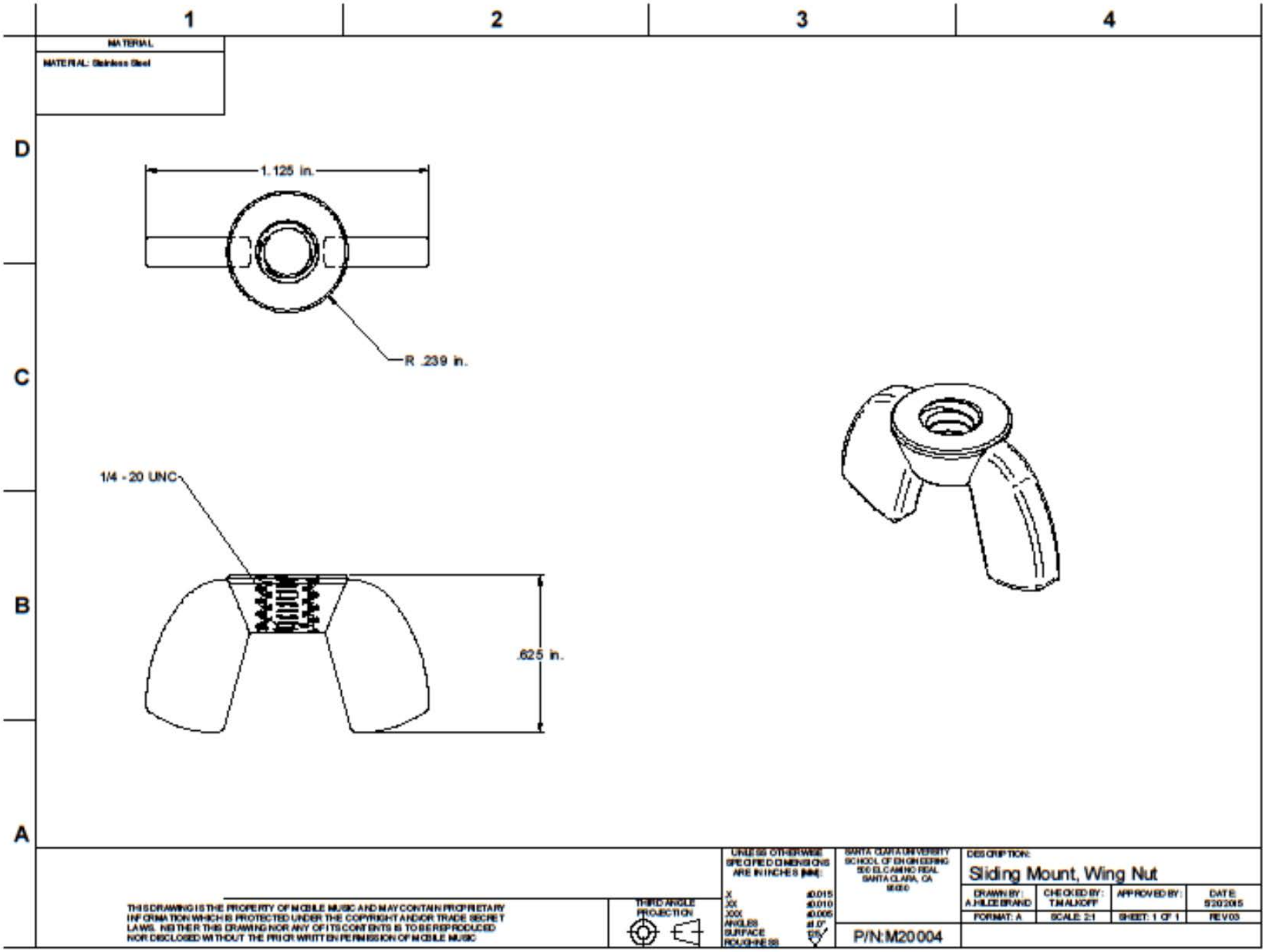
SANTA CLARA UNIVERSITY
SCHOOL OF ENGINEERING
500 EL CAMINO REAL
SANTA CLARA, CA
95050

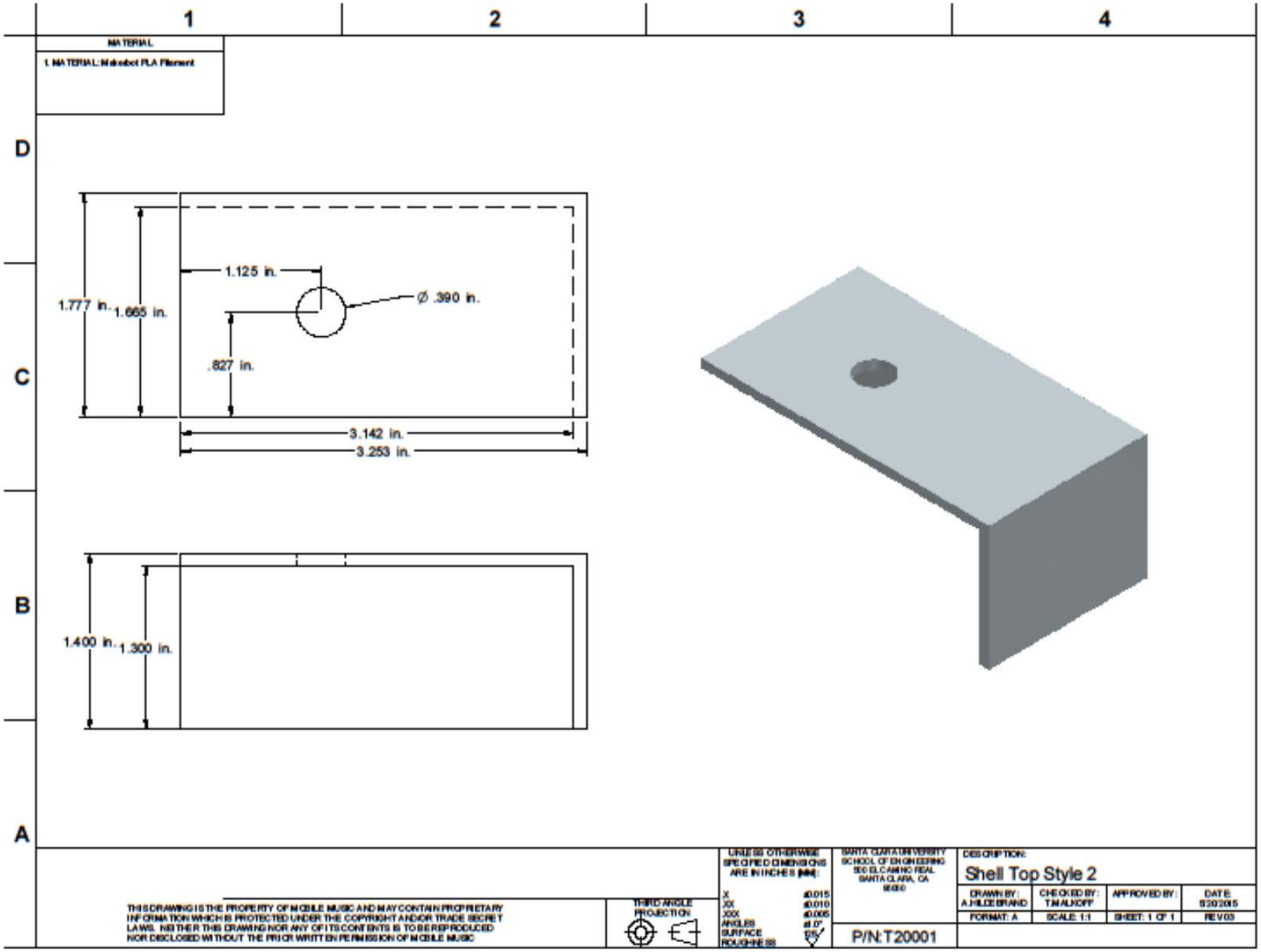
P/N: M20003

DESCRIPTION:

Sliding Mount, Carriage Bolt

DRAWN BY: A.HILDEBRAND	CHECKED BY: T.MALKOFF	APPROVED BY:	DATE: 5/20/2015
FORMAT: A	SCALE: 2:1	SHEET: 1 OF 1	REV 03


1
2
3
4
D
C
B
A
MATERIAL
MATERIAL: Stainless Steel
1.125 in.
R .239 in.
1/4 - 20 UNC
.625 in.
THIS DRAWING IS THE PROPERTY OF MOBILE MUSIC AND MAY CONTAIN PROPRIETARY INFORMATION WHICH IS PROTECTED UNDER THE COPYRIGHT AND/OR TRADE SECRET LAWS. NEITHER THIS DRAWING NOR ANY OF ITS CONTENTS IS TO BE REPRODUCED NOR DISCLOSED WITHOUT THE PRIOR WRITTEN PERMISSION OF MOBILE MUSIC
THIRD ANGLE PROJECTION
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES [MM]:
X ±0.015
.XX ±0.010
.XXX ±0.005
ANGLES ±1.0°
SURFACE ROUGHNESS 125
SANTA CLARA UNIVERSITY SCHOOL OF ENGINEERING 500 EL CAMINO REAL SANTA CLARA, CA 95053
P/N:M20004
DESCRIPTION:
Sliding Mount, Wing Nut
DRAWN BY: A.HILDEBRAND
CHECKED BY: T.MALKOFF
APPROVED BY:
DATE: 5/20/2015
FORMAT: A
SCALE: 2:1
SHEET: 1 OF 1
REV03



THIS DRAWING IS THE PROPERTY OF MOBILE MUSIC AND MAY CONTAIN PROPRIETARY INFORMATION WHICH IS PROTECTED UNDER THE COPYRIGHT AND/OR TRADE SECRET LAWS. NEITHER THIS DRAWING NOR ANY OF ITS CONTENTS IS TO BE REPRODUCED NOR DISCLOSED WITHOUT THE PRIOR WRITTEN PERMISSION OF MOBILE MUSIC

Appendix I – Final Build Code

```
/**********************************************
*            MOBILE MUSIC - main.h            *
**********************************************/

#include <msp430.h>
#include "config.h"
#include "RN52_UART.h"
#include "ADXL345_SPI.h"
#include "BlueTooth_Commands.h"
#include "motion_sense.h"
#include "LEDs.h"

#define BUTTON_TIME           12000

volatile float deltaTheta = 0.0;
volatile float mag_float = 0.0;
volatile unsigned int mag_int = 0;
volatile unsigned int redLEDCounter = 0;
volatile unsigned int blueLEDCounter = 0;

const unsigned int threshUpper = 125;//30;
const unsigned int threshLower = 30;//8;
unsigned long inactivityDelay = 30000;

int main(void) {
      WDTCTL   =   WDTPW + WDTHOLD;    // Stop WDT

      // Initialize
      clockConfig();
      timerConfig();
      IOConfig();
      SPIConfig();
      UARTConfig();
      initMotionSense();

      __enable_interrupt();               // Enable all interrupts

      beginCMDMode();

      while(1)
      {
            switch(SYSstate)
            {
                  case SLEEP:
                        if(prevSYSstate != SLEEP)
                        {
                              P1OUT &= ~LDO_EN;
                              P2OUT &= ~(RED_LED | BLUE_LED);
                              blueLEDstate = LED_OFF;
                              prevSYSstate = SLEEP;
                        }
                        _BIS_SR(LPM0_bits + GIE);          // Enter LPM0 w/
interrupt
                        break;
                  case CAL:
                        prevSYSstate = CAL;
                        redLEDstate = LED_ON;
                        TA1CCTL1 &= ~CCIFG;
                        TA1CCR1 = TA1R + 50;
                        TA1CCTL1 |= CCIE;
                        calibrationDelay();
                        redLEDstate = RAPID;
                        TA1CCTL1 &= ~CCIFG;
                        TA1CCR1 = TA1R + 50;
                        TA1CCTL1 |= CCIE;                         // CCR1
interrupt enabled
                        deltaTheta = calibration();
                        SYSstate = ON;
                        break;
                  case ON:
                        if(prevSYSstate != ON)
                        {
                              prevSYSstate = ON;
                              redLEDstate = BLIP;
                              P1OUT |= LDO_EN;
                              TA1CCTL1 &= ~CCIFG;
                              TA1CCR1 = TA1R + 50;
                              TA1CCTL1 |= CCIE;                         //
CCR1 interrupt enabled
                        }

                        motionSense(deltaTheta, threshUpper, threshLower);

                        switch(MVMNTstate)
                        {
                              case NO_MOTION:
                                    if(prevMVMNTstate == MOTION)
                                    {
                                          prevMVMNTstate = NO_MOTION;
                                          playPause();
                                          //TA1CCTL0 &= ~CCIFG;
                                          //TA1CCR0 = TA1R + inactivityDelay;
                                          //TA1CCTL0 |= CCIE;
                                          //
                                          //for(k = 0; k < 100; k++)
                                          //    __delay_cycles(1000);
                                    }
                                    break;
                              case MOTION:
                                    if(prevMVMNTstate == NO_MOTION)
                                    {
                                          prevMVMNTstate = MOTION;
                                          playPause();
                                          TA1CCTL0 &= ~CCIFG;
                                          TA1CCTL0 &= ~CCIE;
                                    }
                                    //if(PPstate == PAUSE)
                                    //{
                                    //    playPause();
                                    //    PPstate = PLAY;
                                    //}
                                    break;
                              default:
                                    MVMNTstate = NO_MOTION;
                                    break;
                        }
                        break;
                  default:
                        SYSstate = SLEEP;
                        break;
            }
      }
}

// Port 2 interrupt service routine
#pragma vector=PORT2_VECTOR
__interrupt void PORT2_ISR(void)
{
      if(P2IFG & BUTTON)            //  P2.0: CAL/PWR pushbutton
      {
            if(P2IES & BUTTON)
            {
                  TA0CCTL0 &= ~CCIFG;
                  TA0CCR0 = TA0R + BUTTON_TIME;       // Reset CCR0 to
current time,;
                  TA0CCTL0 |= CCIE;                         // CCR0 interrupt
enable,
                                                                        //
add button hold delay
            }
            else
            {
                  TA0CCTL0 &= ~CCIE;            // CCR0 interrupt disabled
                  if(SYSstate != SLEEP)
                        SYSstate = CAL;
                  else
                        redLEDstate = LED_OFF;
            }
            P2IFG &= ~BUTTON;             // P1.3 IFG cleared
            P2IES ^= BUTTON;        // Toggle the interrupt edge,
                                                //   the interrupt vector
will be called
                                                //   when P1.3 goes from
HitoLow as well as
                                                //   LowtoHigh
      }
      if(P2IFG & JACK)        //  P2.1: Headphone jack detect
      {
            P2IFG &= ~JACK;
      }
      if(P2IFG & DATA_INT)    //  P2.5: Accelerometer data ready
      {
            dataFlag = 1;
            P2IFG &= ~DATA_INT;
      }
}

// Timer0_A0 interrupt service routine
```

```
#pragma vector = TIMER0_A0_VECTOR
__interrupt void TIMER0_A0_ISR (void)
{
      TA0CCTL0 &= ~CCIE;            // CCR0 interrupt disabled
      if(SYSstate == SLEEP)
      {
            _BIC_SR_IRQ(LPM0_bits);          // Exit LPM0
            SYSstate = CAL;
            P2IES |= BUTTON;
      }
      else
      {
            redLEDstate = LED_ON;
            SYSstate = SLEEP;
            P2IES &= ~BUTTON;
      }
      P2IFG &= ~BUTTON;             // P1.3 IFG cleared
}

// Timer1_A0 interrupt service routine
#pragma vector = TIMER1_A0_VECTOR
__interrupt void TIMER1_A0_ISR (void)
{
      TA1CCTL0 &= ~CCIE;            // CCR0 interrupt disabled
      if(SYSstate == CAL)
            stillCALDelay = 0;                 // Calibration delay is over
      if(SYSstate == ON)
      {
            playPause();
            PPstate = PAUSE;
      }
}

// Timer1_A1 interrupt service routine
#pragma vector = TIMER1_A1_VECTOR
__interrupt void TIMER1_A1_ISR (void)
{
      switch(__even_in_range(TA1IV, TA1IV_TAIFG))
      {
            case TA1IV_TACCR1:        // CCR1
                  switch(redLEDstate)
                  {
                        case LED_ON:
                              redLEDOn();
                              break;
                        case LED_OFF:
                              redLEDOff();
                              break;
                        case BLIP:
                              redLEDBlip();
                              break;
                        case RAPID:
                              redLEDRapid();
                              break;
                        default:
                              TA1CCTL1 &= ~CCIE;
      // CCR0 interrupt enabled
```

```
                              break;
                  }
              break;
            case TA1IV_TACCR2:        // CCR2
                  switch(blueLEDstate)
                  {
                        case LED_ON:
                              blueLEDOn();
                              break;
                        case LED_OFF:
                              blueLEDOff();
                              break;
                        case BLIP:
                              blueLEDBlip();
                              break;
                        case SLOW:
                              blueLEDSlow();
                              break;
                        default:
                              TA1CCTL2 &= ~CCIE;
      // CCR0 interrupt enabled
                              break;
                  }
                break;
      }
}
```

```
/**********************************************
*           MOBILE MUSIC - config.h           *
**********************************************/

// Port 1
#define LDO_EN          BIT0                                      // P1.0
(GENERAL OUTPUT)    BlueTooth and audio amp LDO enable
#define RXD             BIT1                                      // P1.1
(PERIPHERAL INPUT)  BlueTooth UART RX (CMD and SPP)
#define TXD             BIT2                                      // P1.2
(PERIPHERAL OUTPUT) BlueTooth UART TX (CMD and SPP)
#define GPIO9           BIT3                                      // P1.3
(GENERAL OUTPUT)    BlueTooth CMD/SPP switch (LO/HI, respectively)
#define GPIO2           BIT4                                      // P1.4
(INTERRUPT INPUT)   BlueTooth status change interrupt
#define SPI_CLK         BIT5                                      // P1.5
(PERIPHERAL OUTPUT) Accelerometer SPI clock
#define MISO            BIT6                                      // P1.6
(PERIPHERAL INPUT)  Accelerometer SPI output
#define MOSI            BIT7                                      // P1.7
(PERIPHERAL OUTPUT) Accelerometer SPI input

// Port 2
#define BUTTON          BIT0                                      // P2.0
(INTERRUPT INPUT)   CAL/PWR pushbutton
#define JACK            BIT1                                      // P2.1
(INTERRUPT INPUT)   Headphone jack detect
#define RED_LED         BIT2                                      // P2.2
(GENERAL OUTPUT)    Drives red LED
#define BLUE_LED  BIT3                                      // P2.3 (GENERAL
OUTPUT)    Drives blue LED
#define CS              BIT4                                      // P2.4
(GENERAL OUTPUT)    Accelerometer SPI chip select
#define DATA_INT  BIT5                                      // P2.5
(INTERRUPT INPUT)   Accelerometer data ready interrupt

// Combinations
//#define BT_ON         LDO_EN + GPIO9

// STATES
enum SYSstate_t {ON, CAL, SLEEP};
enum BTstate_t {BT_OFF, DISCONNECTED, CONNECTED};
enum LEDstate_t {LED_OFF, LED_ON, BLIP, SLOW, RAPID};
// enum CONFIGstate_t {WRLSS_PLYR, WRLSS_DVC} CONFIGstate = WRLSS_DVC;

// STATE VARIABLES
enum SYSstate_t SYSstate = CAL;
enum SYSstate_t prevSYSstate = ON;
enum BTstate_t BTstate = BT_OFF;
enum BTstate_t prevBTstate = DISCONNECTED;
enum LEDstate_t redLEDstate = LED_OFF;
enum LEDstate_t blueLEDstate = LED_OFF;

void clockConfig(void)
{
      BCSCTL1 = CALBC1_16MHZ;                                     // Set DCO
to 16MHz
```

```
      DCOCTL = CALDCO_16MHZ;                                      // Set DCO
to 16MHz
      BCSCTL2 = SELM_0 + DIVM_0 + DIVS_1;
}

void timerConfig(void)
{
      TA0CTL = TASSEL_1 + MC_2 + ID_2;    // ACLK, continuous mode
      TA1CTL = TASSEL_1 + MC_2 + ID_2;    // ACLK, continuous mode
      TA0CCR0 = 0;
      TA0CCR1 = 0;
      TA0CCR2 = 0;
      TA1CCR0 = 0;
      TA1CCR1 = 0;
      TA1CCR2 = 0;
}

void IOConfig(void)
{
      P1OUT |= LDO_EN;
      P1DIR |= LDO_EN;

      P2IES &= ~BUTTON;                   // Set trigger LO-HI
      P2IFG &= ~BUTTON;                         // P1.3 IFG cleared
      P2IE |= BUTTON;                     // P1.3 interrupt enabled

      P2DIR &= ~JACK;
      P2IES |= JACK;                            // Set trigger LO-HI
      P2IFG &= ~JACK;                     // P1.3 IFG cleared
      P2IE |= JACK;                             // P1.3 interrupt enabled

      P2OUT &= ~RED_LED;
      P2DIR |= RED_LED;

      P2OUT &= ~BLUE_LED;
      P2DIR |= BLUE_LED;
}
```

```
/***********************************************
*          MOBILE MUSIC - RN52_UART.h         *
***********************************************/

void UARTConfig(void)
{
      // Pin Config
      P1SEL        |=  RXD   |   TXD;                                  //
USCI_A0 config for UART
      P1SEL2   |=  RXD   |   TXD;                                // USCI_A0
config for UART
      P1OUT    &=  ~GPIO9;                                             //
GPIO9 HIGH
      P1DIR    |=  GPIO9;                                              //
GPIO9 HIGH

      // Peripheral Config
      UCA0CTL1   |=   UCSWRST;                                   // Disable
USCI_A0
      UCA0CTL1  |=   UCSSEL_2;                                   // Select
UART clock as SMCLK (8MHz)
      UCA0BR0    =   69;                                         // Set UART
baud rate to 115200 (SMCLK = 8MHz)
      UCA0BR1    =   0;                                                //
Set UART baud rate to 115200 (SMCLK = 8MHz)
      UCA0MCTL   =   UCBRS_4;                                    // Set UART
baud rate to 115200 (SMCLK = 8MHz)
      UCA0CTL1   &=  ~UCSWRST;                                   // Enable
USCI_A0
}

unsigned char UARTRead(void)
{
      while (!(IFG2 & UCA0RXIFG));                         // USCI_A0 TX
buffer ready?
      return UCA0RXBUF;
}

void UARTWrite(unsigned char tx)
{
      while (!(IFG2 & UCA0TXIFG));                         // USCI_A0 TX
buffer ready?
      UCA0TXBUF = tx;
      while (UCB0STAT & UCBUSY);                                 //
Transmission completed?
      __delay_cycles(1000);
      return;
}

void sendCMD(unsigned char* tx, unsigned int txSize)
{
      unsigned int i;
      for(i = txSize; i > 0; i--)                          // Loop for
number of reads required
      {
            UARTWrite(tx[txSize - i]);
      }
}

void sendSPP(unsigned char* tx, unsigned int txSize)
{
      unsigned int i;
      for(i = txSize; i > 0; i--)                           // Loop for
number of reads required
      {
            UARTWrite(tx[txSize - i]);
      }
}

void receiveCMD(unsigned char* rx, unsigned int rxSize)
{
      unsigned int i = 0;
      for(i = rxSize; i > 0; i--)                           // Loop for
number of reads required
      {
            rx[rxSize - i] = UARTRead();
      }
}

void receiveSPP(unsigned char* rx, unsigned int rxSize)
{
      unsigned int i = 0;
      for(i = rxSize; i > 0; i--)                           // Loop for
number of reads required
      {
            rx[rxSize - i] = UARTRead();
      }
}
```

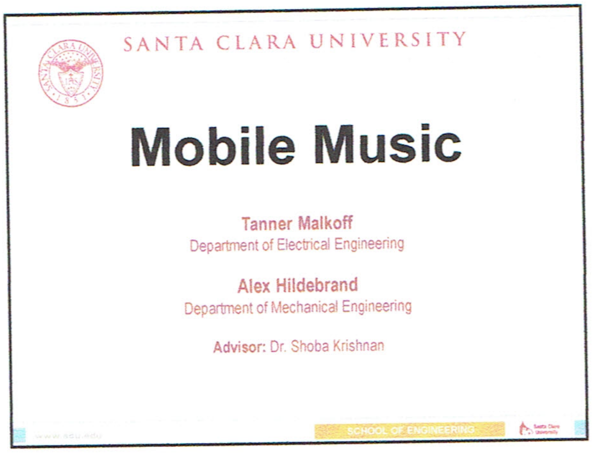
```
/**********************************************
*     MOBILE MUSIC - BlueTooth_Commands.h     *
**********************************************/

#include <stdlib.h>
//#include "RN52_UART.h"

#define STATUS_CMD            "Q\r\n"
#define PLAY_PAUSE_CMD  "AP\r\n"
#define AOK_CMD               "AOK\r\n"

unsigned char BTStatusRX[4];

unsigned int getBTStatus(unsigned char* rxString, unsigned int rxSize)
{
      if(rxSize != 4)
            return 0;
      sendSPP(STATUS_CMD, sizeof(STATUS_CMD));
      receiveCMD(rxString, rxSize);
      return 1;
}

unsigned int beginCMDMode(void)
{
      unsigned char rx0[] = "CMD";
      unsigned int rxSize = sizeof(rx0) - 1;
      unsigned char* rx1 = (unsigned char*)malloc(rxSize * sizeof(unsigned
char));
      unsigned int i;
      P1OUT &= ~GPIO9;
      receiveCMD(rx1, rxSize);
      for(i = rxSize; i > 0; i--)                           // Loop for
number of reads required
      {
            if(rx1[rxSize - i] != rx0[rxSize - i])
                  return 0;
      }
      return 1;
}

unsigned int playPause(void)
{

      sendCMD(PLAY_PAUSE_CMD, sizeof(PLAY_PAUSE_CMD));
//    receiveCMD(AOK_CMD, sizeof(AOK_CMD));
      return 1;

}
```

```
/**********************************************
*        MOBILE MUSIC - motion_sense.h       *
**********************************************/

#include <math.h>

#define CAL_DELAY    20000
#define CAL_SAMPLES  300

enum MVMNTstate_t {MOTION, NO_MOTION};
enum PPstate_t {PLAY, PAUSE};
enum MVMNTstate_t MVMNTstate = NO_MOTION;
enum MVMNTstate_t prevMVMNTstate = NO_MOTION;
enum PPstate_t PPstate = PAUSE;

volatile unsigned int dataFlag = 0;
volatile unsigned int stillINACTDelay = 0;
volatile unsigned int stillCALDelay = 0;

void initMotionSense(void)
{
      SPIWrite(POWER_CTL, 0x00);                // standby mode
      SPIWrite(INT_ENABLE, 0x00);               // disable DATA_READY
interrupt
      SPIWrite(DATA_FORMAT, 0x0B);              // +/- 16G range w/ full res
      SPIWrite(BW_RATE, 0x0B);                  // BW:  200 Hz
      SPIWrite(INT_MAP, 0x00);                  // set DATA_READY to INT1
      SPIWrite(POWER_CTL, 0x08);                // measurement mode
      SPIWrite(INT_ENABLE, 0x80);               // enable DATA_READY
interrupt
}

void calibrationDelay(void)
{
      stillCALDelay = 1;
      TA1CCTL0 &= ~CCIFG;
      TA1CCR0 = TA1R + CAL_DELAY;
      TA1CCTL0 |= CCIE;                         // CCR0 interrupt enabled
      while(stillCALDelay);
}

float calibration(void)
{
      float calX = 0;
      float calY = 0;
      float calZ = 0;
      float calR = 0;
      unsigned int count = CAL_SAMPLES;
      unsigned char values[6];

      while(count > 0)
      {
            if(dataFlag)
            {
                  dataFlag = 0;
                  count--;
                  SPIMBRead(DATAX0, values, 6);
                  calX += (float)(((int)values[1]<<8)|(int)values[0]);
                  calY += (float)(((int)values[3]<<8)|(int)values[2]);
                  calZ += (float)(((int)values[5]<<8)|(int)values[4]);
            }
      }

      calX = calX/CAL_SAMPLES;
      calY = calY/CAL_SAMPLES;
      calZ = calZ/CAL_SAMPLES;
      calR = sqrt((calZ*calZ) + (calY*calY) + (calX*calX));
      return acos(calZ/calR);
}

float getXYMag(float offset)
{
      float XYMag = 0.0;
      float x = 0.0;
      float y = 0.0;
      float z = 0.0;
      float r = 0.0;
      unsigned char values[6];

      while(!dataFlag);
      SPIMBRead(DATAX0, values, 6);

      x = (float)(((int)values[1]<<8)|(int)values[0]);
      y = (float)(((int)values[3]<<8)|(int)values[2]);
      z = (float)(((int)values[5]<<8)|(int)values[4]);
      r = sqrt((z*z) + (y*y) + (x*x));
      XYMag = r*sin(acos(z/r) - offset);
      if(XYMag > 0)
            return XYMag;
      else
            return -1.0*XYMag;
}

void motionSense(float deltaTheta, unsigned int threshHi, unsigned int
threshLo)
{
      float XYMag = getXYMag(deltaTheta);
      if(((XYMag >= threshHi) && (PPstate == PAUSE)) || ((XYMag >= threshLo)
&& (PPstate == PLAY)))
            MVMNTstate = MOTION;
      else
            MVMNTstate = NO_MOTION;
}
```

```
/**********************************************
*            MOBILE MUSIC - LEDs.h            *
**********************************************/

#define LED_TIME_RAPID        500
#define LED_TIME_SLOW         2000
#define LED_TIME_BLIP_ON      600
#define LED_TIME_BLIP_OFF     20000//1500
#define LED_TIME_ON     100

void redLEDBlip(void)
{
      if(P2OUT & RED_LED)
      {
            P2OUT &= ~RED_LED;
            TA1CCR1 = TA1R + LED_TIME_BLIP_OFF;
      }
      else
      {
            P2OUT |= RED_LED;
            TA1CCR1 = TA1R + LED_TIME_BLIP_ON;
      }
}

void redLEDRapid(void)
{
      P2OUT ^= RED_LED;
      TA1CCR1 = TA1R + LED_TIME_RAPID;
}

void redLEDOn(void)
{
      P2OUT ^= RED_LED;
      TA1CCR1 = TA1R + LED_TIME_ON;
}

void redLEDOff(void)
{
      P2OUT &= ~RED_LED;
      TA1CCR1 = TA1R + LED_TIME_SLOW;
}

void blueLEDBlip(void)
{
      if(P2OUT & BLUE_LED)
      {
            P2OUT &= ~BLUE_LED;
            TA1CCR2 = TA1R + LED_TIME_BLIP_OFF;
      }
      else
      {
            P2OUT |= BLUE_LED;
            TA1CCR2 = TA1R + LED_TIME_BLIP_ON;
      }
}

void blueLEDSlow(void)
```

```
{
      P2OUT ^= BLUE_LED;
      TA1CCR2 = TA1R + LED_TIME_SLOW;
}

void blueLEDOn(void)
{
      P2OUT ^= BLUE_LED;
      TA1CCR2 = TA1R + LED_TIME_ON;
}

void blueLEDOff(void)
{
      P2OUT &= ~BLUE_LED;
      TA1CCR2 = TA1R + LED_TIME_SLOW;
}
```

Appendix J – Senior Design Conference Presentation



SANTA CLARA UNIVERSITY

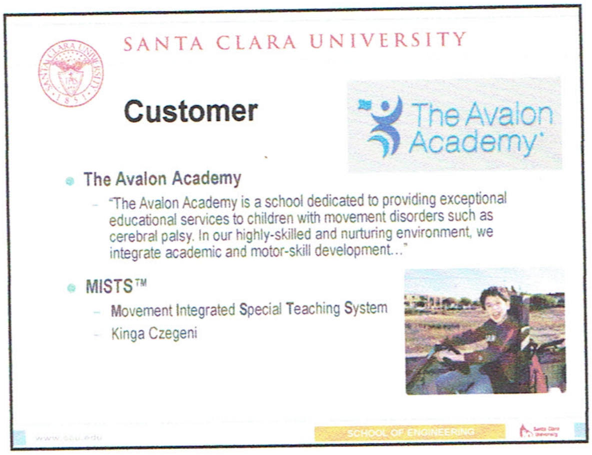
Mobile Music

Tanner Malkoff
Department of Electrical Engineering

Alex Hildebrand
Department of Mechanical Engineering

Advisor: Dr. Shoba Krishnan

www.scu.edu SCHOOL OF ENGINEERING



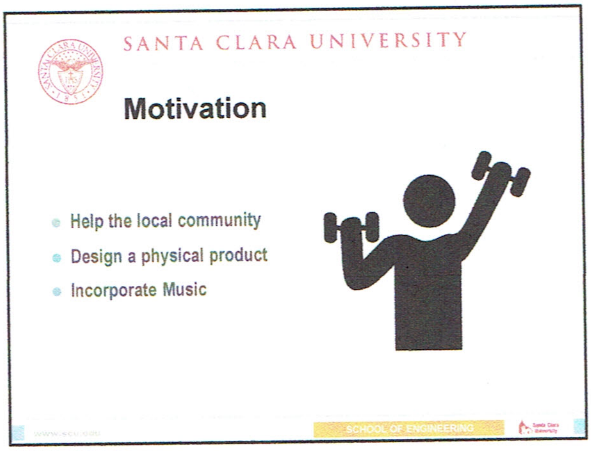
SANTA CLARA UNIVERSITY

Customer

The Avalon Academy

- The Avalon Academy
  - "The Avalon Academy is a school dedicated to providing exceptional educational services to children with movement disorders such as cerebral palsy. In our highly-skilled and nurturing environment, we integrate academic and motor-skill development…"
- MISTS™
  - Movement Integrated Special Teaching System
  - Kinga Czegeni

www.scu.edu SCHOOL OF ENGINEERING

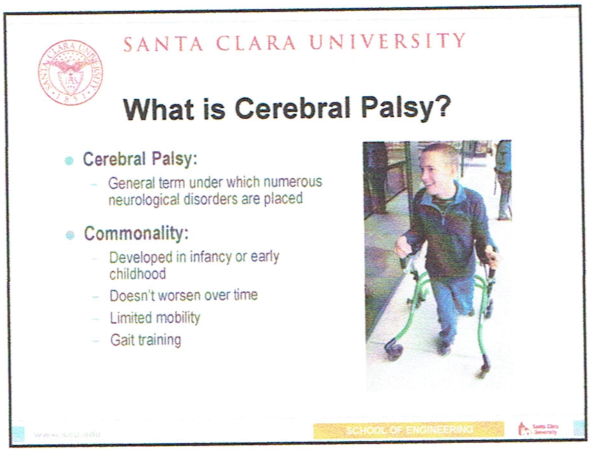


SANTA CLARA UNIVERSITY

Motivation

- Help the local community
- Design a physical product
- Incorporate Music

www.scu.edu SCHOOL OF ENGINEERING

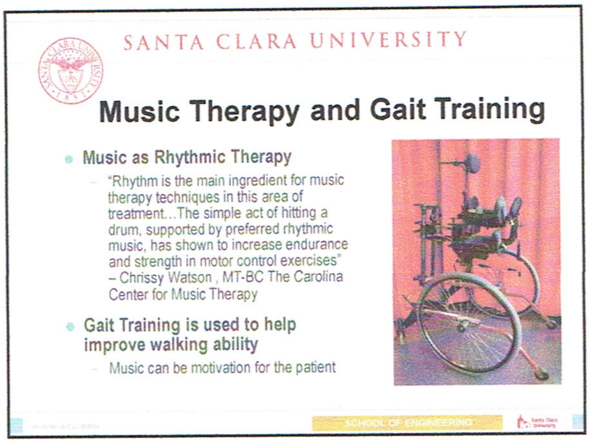


SANTA CLARA UNIVERSITY

What is Cerebral Palsy?

- Cerebral Palsy:
  - General term under which numerous neurological disorders are placed
- Commonality:
  - Developed in infancy or early childhood
  - Doesn't worsen over time
  - Limited mobility
  - Gait training

www.scu.edu SCHOOL OF ENGINEERING

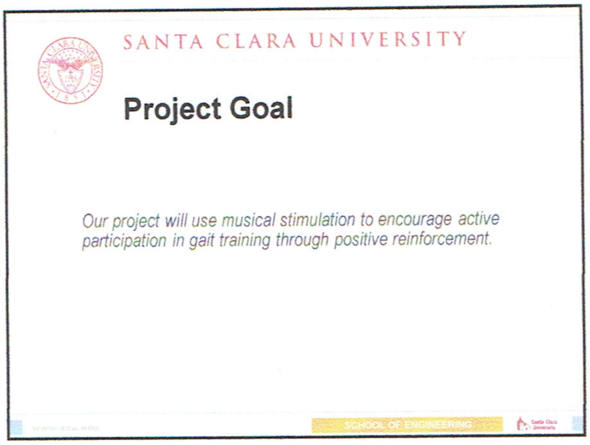


SANTA CLARA UNIVERSITY

Music Therapy and Gait Training

- Music as Rhythmic Therapy
  - "Rhythm is the main ingredient for music therapy techniques in this area of treatment…The simple act of hitting a drum, supported by preferred rhythmic music, has shown to increase endurance and strength in motor control exercises" – Chrissy Watson , MT-BC The Carolina Center for Music Therapy
- Gait Training is used to help improve walking ability
  - Music can be motivation for the patient

www.scu.edu SCHOOL OF ENGINEERING

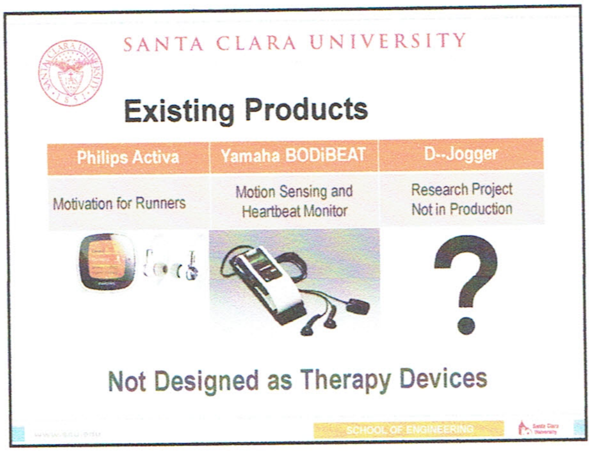


SANTA CLARA UNIVERSITY

Project Goal

Our project will use musical stimulation to encourage active participation in gait training through positive reinforcement.

www.scu.edu SCHOOL OF ENGINEERING



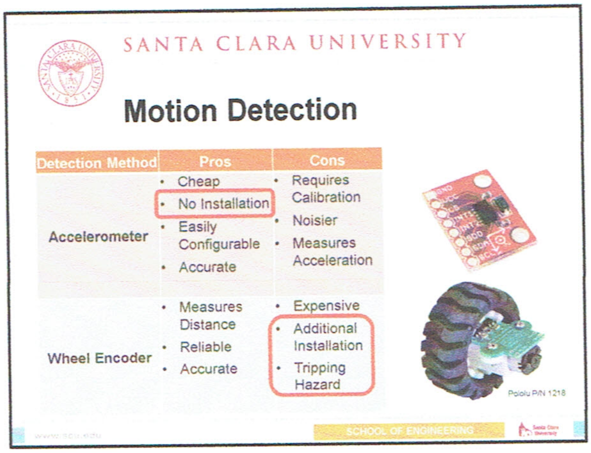
SANTA CLARA UNIVERSITY

Existing Products

Philips Activa	Yamaha BODiBEAT	D--Jogger
Motivation for Runners	Motion Sensing and Heartbeat Monitor	Research Project Not in Production

Not Designed as Therapy Devices

www.scu.edu SCHOOL OF ENGINEERING



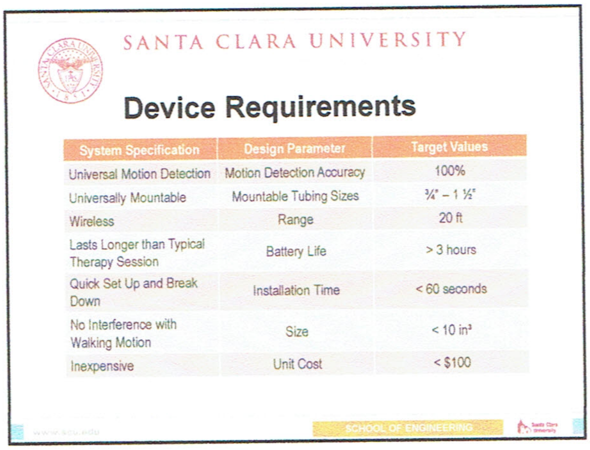
SANTA CLARA UNIVERSITY

Motion Detection

Detection Method	Pros	Cons
Accelerometer	• Cheap • No Installation • Easily Configurable • Accurate	• Requires Calibration • Noisier • Measures Acceleration
Wheel Encoder	• Measures Distance • Reliable • Accurate	• Expensive • Additional Installation • Tripping Hazard

Pololu P/N 1218

www.scu.edu SCHOOL OF ENGINEERING

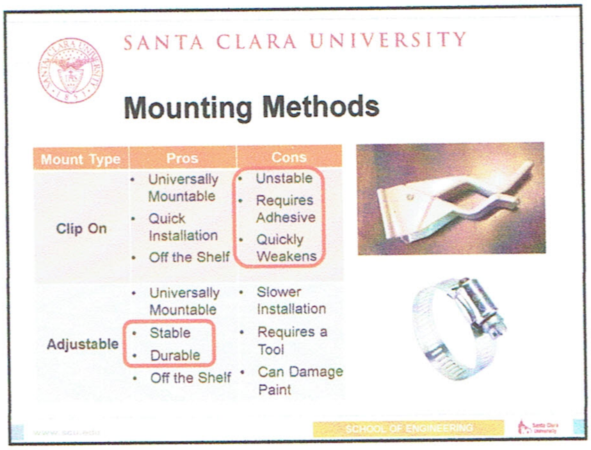


SANTA CLARA UNIVERSITY

Device Requirements

System Specification	Design Parameter	Target Values
Universal Motion Detection	Motion Detection Accuracy	100%
Universally Mountable	Mountable Tubing Sizes	¾" – 1 ½"
Wireless	Range	20 ft
Lasts Longer than Typical Therapy Session	Battery Life	> 3 hours
Quick Set Up and Break Down	Installation Time	< 60 seconds
No Interference with Walking Motion	Size	< 10 in^3
Inexpensive	Unit Cost	< $100

www.scu.edu SCHOOL OF ENGINEERING

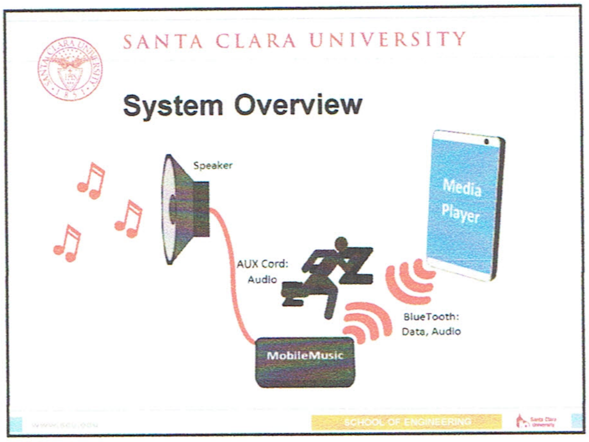


SANTA CLARA UNIVERSITY

Mounting Methods

Mount Type	Pros	Cons
Clip On	• Universally Mountable • Quick Installation • Off the Shelf	• Unstable • Requires Adhesive • Quickly Weakens
Adjustable	• Universally Mountable • Stable • Durable • Off the Shelf	• Slower Installation • Requires a Tool • Can Damage Paint

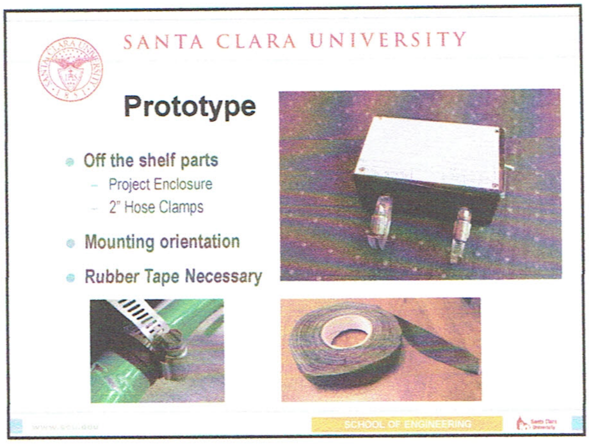
www.scu.edu SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

System Overview

www.scu.edu SCHOOL OF ENGINEERING

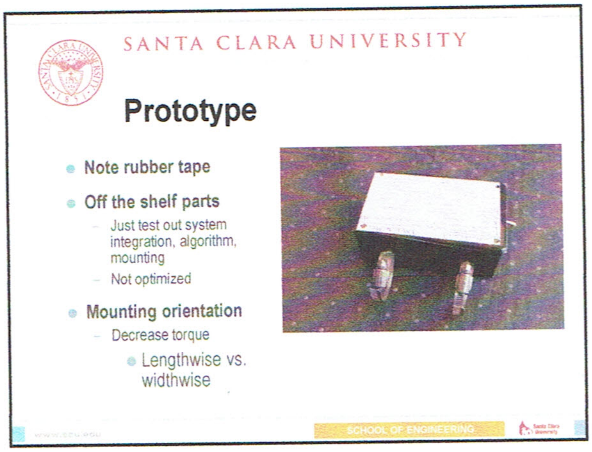


SANTA CLARA UNIVERSITY

Prototype

- Off the shelf parts
  - Project Enclosure
  - 2" Hose Clamps
- Mounting orientation
- Rubber Tape Necessary

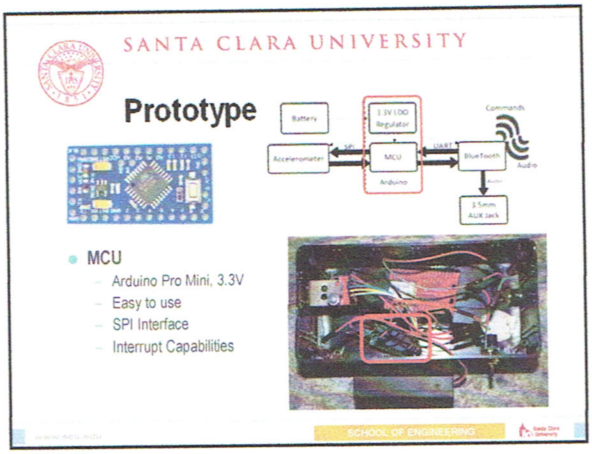
www.scu.edu SCHOOL OF ENGINEERING


SANTA CLARA UNIVERSITY

Prototype

- Note rubber tape
- Off the shelf parts
  - Just test out system integration, algorithm, mounting
  - Not optimized
- Mounting orientation
  - Decrease torque
    - Lengthwise vs. widthwise

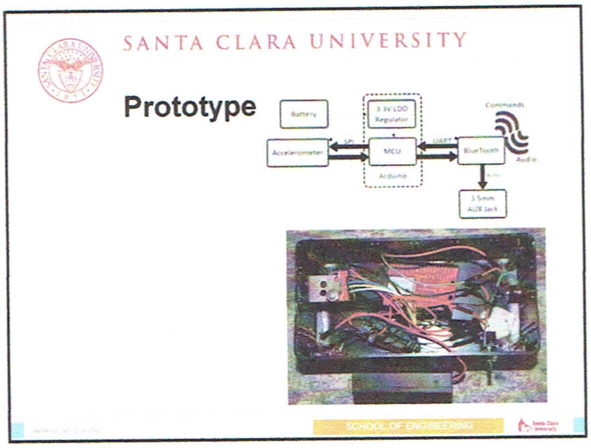
www.scu.edu SCHOOL OF ENGINEERING Santa Clara University


SANTA CLARA UNIVERSITY

Prototype

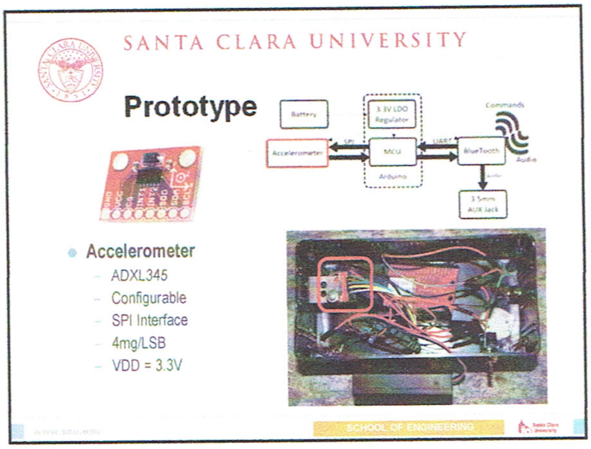
- MCU
  - Arduino Pro Mini, 3.3V
  - Easy to use
  - SPI Interface
  - Interrupt Capabilities

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University


SANTA CLARA UNIVERSITY

Prototype

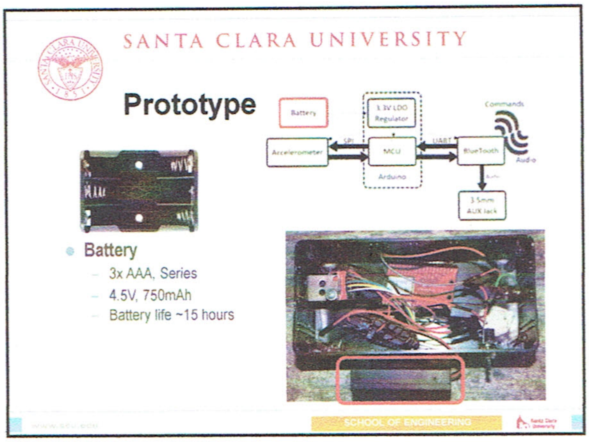
www.scu.edu SCHOOL OF ENGINEERING Santa Clara University


SANTA CLARA UNIVERSITY

Prototype

- Accelerometer
  - ADXL345
  - Configurable
  - SPI Interface
  - 4mg/LSB
  - VDD = 3.3V

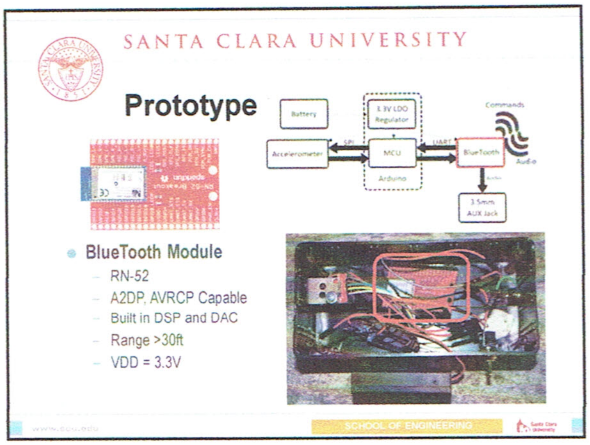
www.scu.edu SCHOOL OF ENGINEERING Santa Clara University


SANTA CLARA UNIVERSITY

Prototype

- Battery
  - 3x AAA, Series
  - 4.5V, 750mAh
  - Battery life ~15 hours

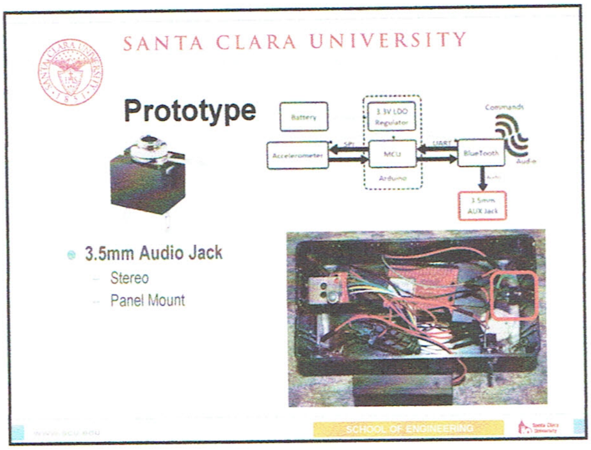
www.scu.edu SCHOOL OF ENGINEERING Santa Clara University


SANTA CLARA UNIVERSITY

Prototype

- BlueTooth Module
  - RN-52
  - A2DP, AVRCP Capable
  - Built in DSP and DAC
  - Range >30ft
  - VDD = 3.3V

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University



SANTA CLARA UNIVERSITY

Prototype

Battery

3.3V LDO Regulator

Accelerometer

SPI

MCU

Arduino

UART

BlueTooth

Commands

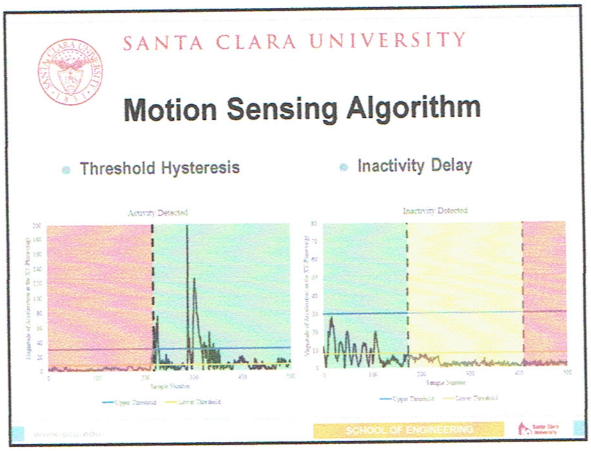
Audio

3.5mm AUX Jack

- 3.5mm Audio Jack
  - Stereo
  - Panel Mount

www.scu.edu

SCHOOL OF ENGINEERING



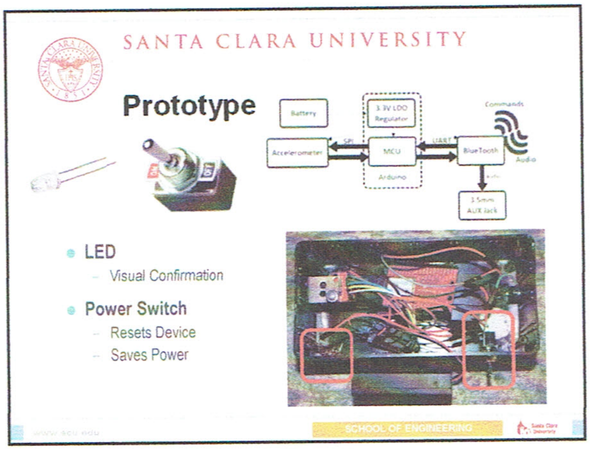
SANTA CLARA UNIVERSITY

Motion Sensing Algorithm

- Threshold Hysteresis
- Inactivity Delay

www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

Prototype

Battery

3.3V LDO Regulator

Accelerometer

SPI

MCU

Arduino

UART

BlueTooth

Commands

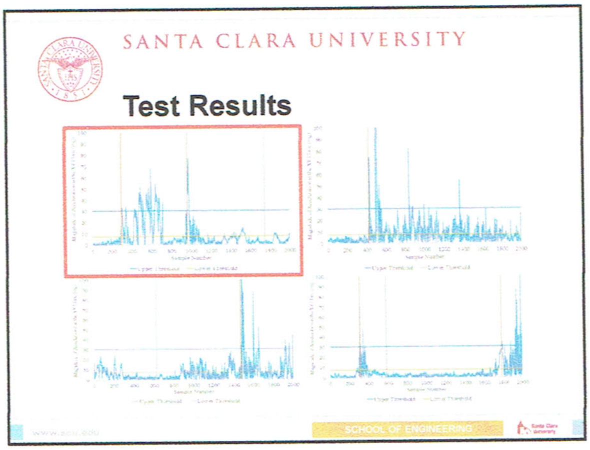
Audio

3.5mm AUX Jack

- LED
  - Visual Confirmation
- Power Switch
  - Resets Device
  - Saves Power

www.scu.edu

SCHOOL OF ENGINEERING

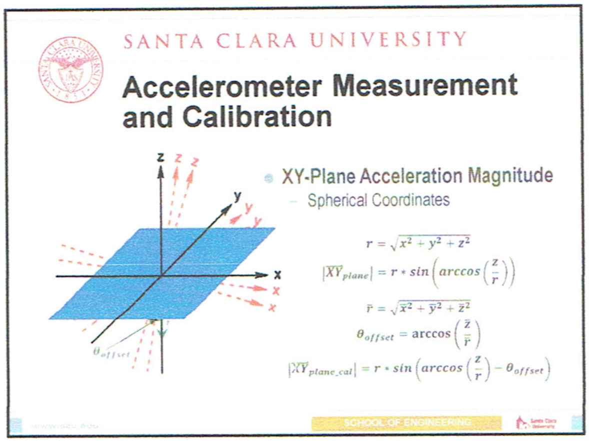


SANTA CLARA UNIVERSITY

Test Results

www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

Accelerometer Measurement and Calibration

- XY-Plane Acceleration Magnitude
  - Spherical Coordinates

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$|\overline{XY}_{plane}| = r * sin\left(arccos\left(\frac{z}{r}\right)\right)$$

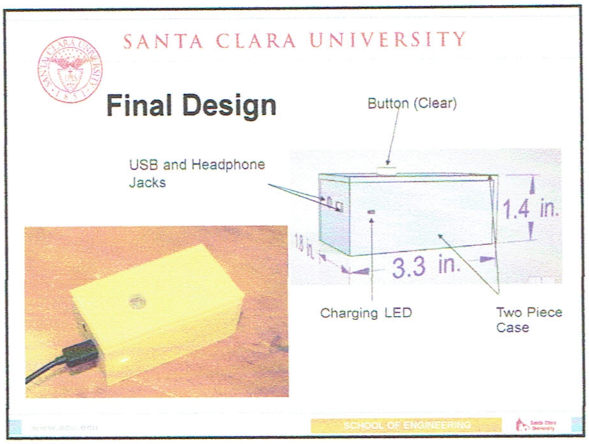
$$\bar{r} = \sqrt{\bar{x}^2 + \bar{y}^2 + \bar{z}^2}$$

$$\theta_{offset} = \arccos\left(\frac{\bar{z}}{\bar{r}}\right)$$

$$|\overline{XY}_{plane\_cal}| = r * sin\left(arccos\left(\frac{z}{r}\right) - \theta_{offset}\right)$$

www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

Final Design

Button (Clear)

USB and Headphone Jacks

1.4 in.

1.8 in.

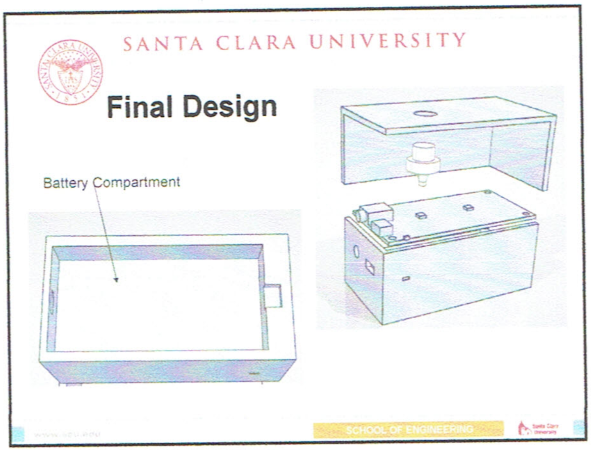
3.3 in.

Charging LED

Two Piece Case

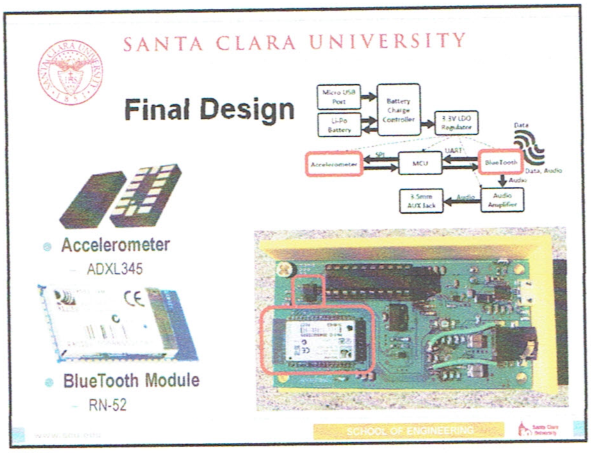
www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

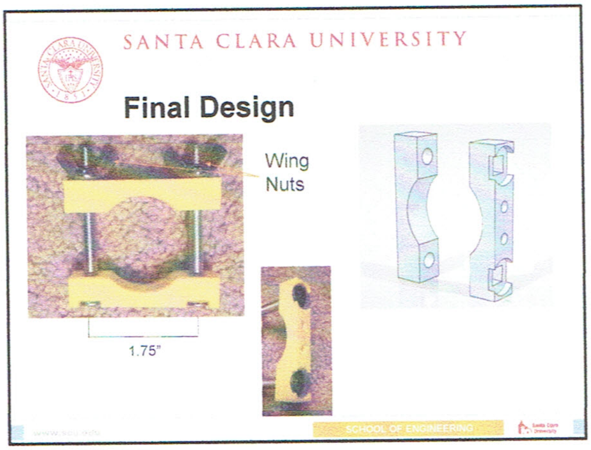
Final Design



SANTA CLARA UNIVERSITY

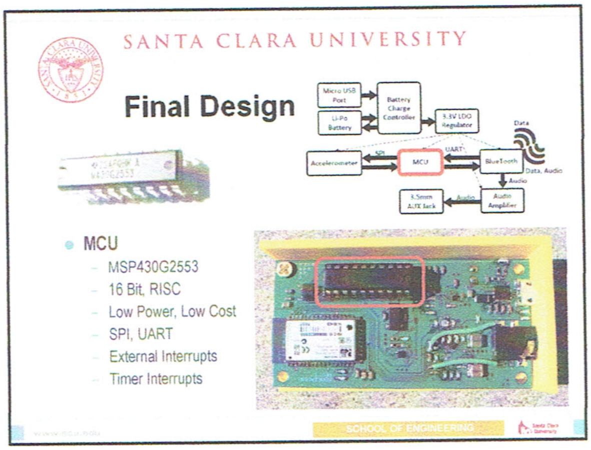
Final Design

- Accelerometer
  - ADXL345
- BlueTooth Module
  - RN-52



SANTA CLARA UNIVERSITY

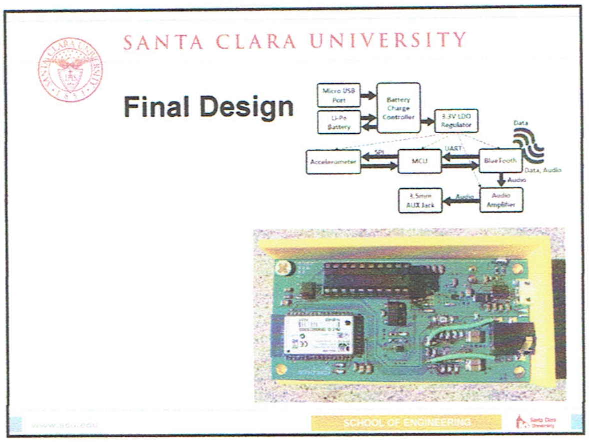
Final Design



SANTA CLARA UNIVERSITY

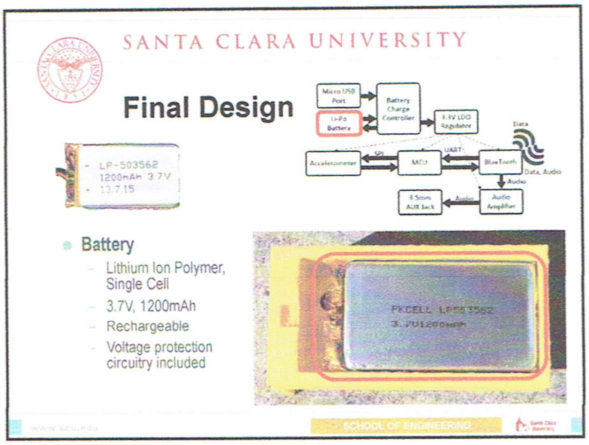
Final Design

- MCU
  - MSP430G2553
  - 16 Bit, RISC
  - Low Power, Low Cost
  - SPI, UART
  - External Interrupts
  - Timer Interrupts



SANTA CLARA UNIVERSITY

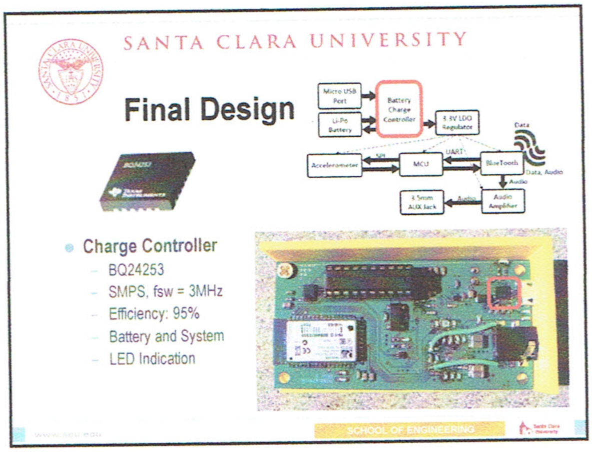
Final Design



SANTA CLARA UNIVERSITY

Final Design

- Battery
  - Lithium Ion Polymer, Single Cell
  - 3.7V, 1200mAh
  - Rechargeable
  - Voltage protection circuitry included

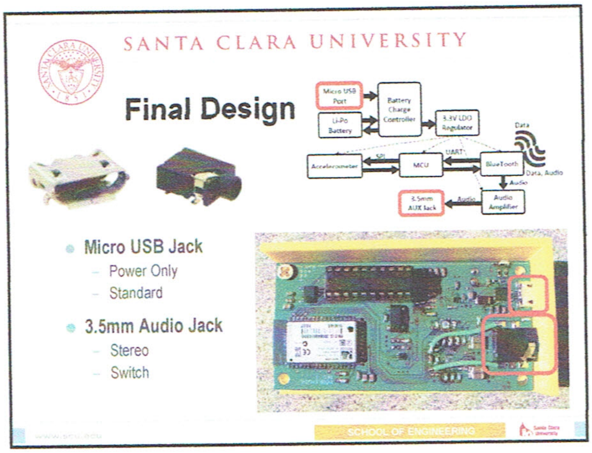


SANTA CLARA UNIVERSITY

Final Design

- Charge Controller
  - BQ24253
  - SMPS, fsw = 3MHz
  - Efficiency: 95%
  - Battery and System
  - LED Indication

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

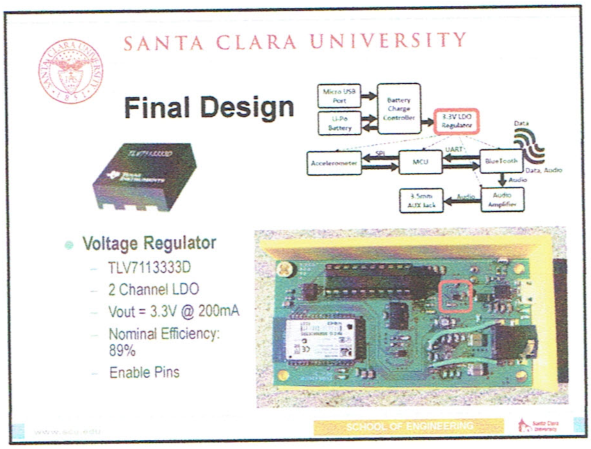


SANTA CLARA UNIVERSITY

Final Design

- Micro USB Jack
  - Power Only
  - Standard
- 3.5mm Audio Jack
  - Stereo
  - Switch

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

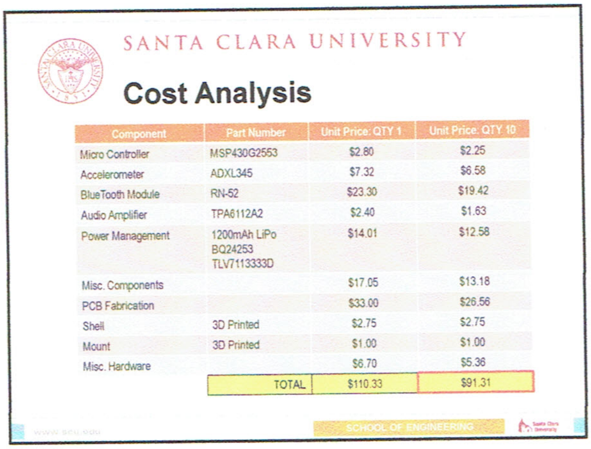


SANTA CLARA UNIVERSITY

Final Design

- Voltage Regulator
  - TLV7113333D
  - 2 Channel LDO
  - Vout = 3.3V @ 200mA
  - Nominal Efficiency: 89%
  - Enable Pins

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

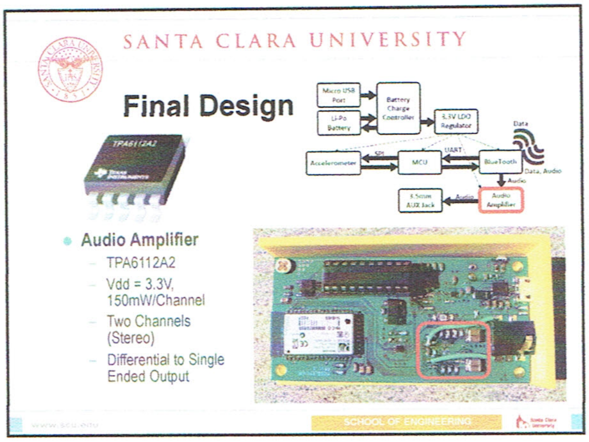


SANTA CLARA UNIVERSITY

Cost Analysis

Component	Part Number	Unit Price: QTY 1	Unit Price: QTY 10
Micro Controller	MSP430G2553	$2.80	$2.25
Accelerometer	ADXL345	$7.32	$6.58
BlueTooth Module	RN-52	$23.30	$19.42
Audio Amplifier	TPA6112A2	$2.40	$1.63
Power Management	1200mAh LiPo BQ24253 TLV7113333D	$14.01	$12.58
Misc. Components		$17.05	$13.18
PCB Fabrication		$33.00	$26.56
Shell	3D Printed	$2.75	$2.75
Mount	3D Printed	$1.00	$1.00
Misc. Hardware		$6.70	$5.36
	TOTAL	$110.33	$91.31

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

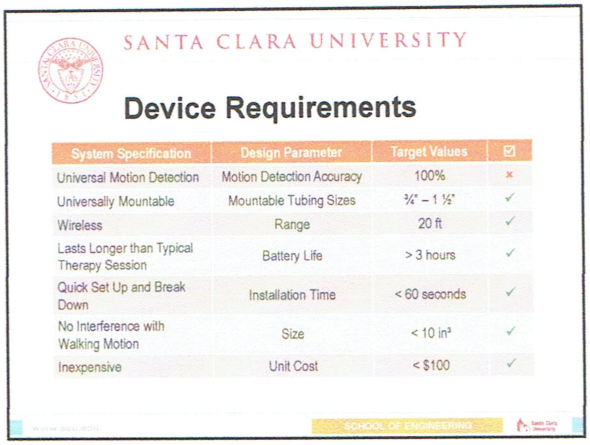


SANTA CLARA UNIVERSITY

Final Design

- Audio Amplifier
  - TPA6112A2
  - Vdd = 3.3V, 150mW/Channel
  - Two Channels (Stereo)
  - Differential to Single Ended Output

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University



SANTA CLARA UNIVERSITY

Device Requirements

System Specification	Design Parameter	Target Values	☑
Universal Motion Detection	Motion Detection Accuracy	100%	✗
Universally Mountable	Mountable Tubing Sizes	¾" – 1 ½"	✓
Wireless	Range	20 ft	✓
Lasts Longer than Typical Therapy Session	Battery Life	> 3 hours	✓
Quick Set Up and Break Down	Installation Time	< 60 seconds	✓
No Interference with Walking Motion	Size	< 10 in³	✓
Inexpensive	Unit Cost	< $100	✓

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

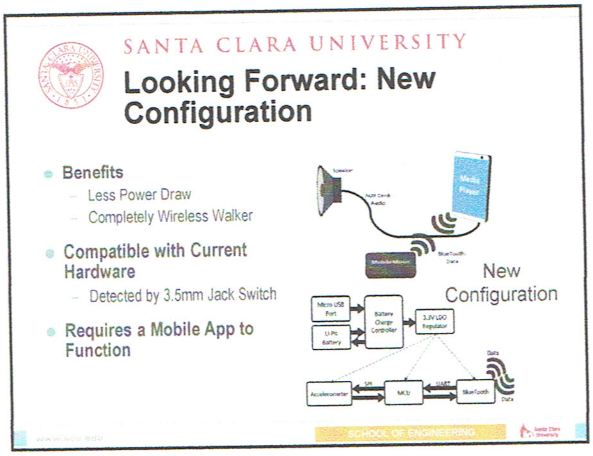
SANTA CLARA UNIVERSITY

Looking Forward: Manufacturing

- Looking to move towards injection molding
  - Cheaper to produce parts at quantity
  - More material choices
- Rubberized coating
  - Dip or spray coating to be applied
  - Will aide with shock absorption

www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

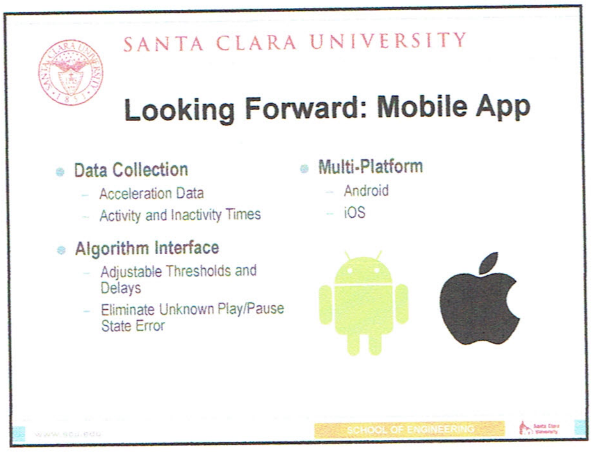
Looking Forward: New Configuration

- Benefits
  - Less Power Draw
  - Completely Wireless Walker
- Compatible with Current Hardware
  - Detected by 3.5mm Jack Switch
- Requires a Mobile App to Function

New Configuration

www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

Looking Forward: Mobile App

- Data Collection
  - Acceleration Data
  - Activity and Inactivity Times
- Algorithm Interface
  - Adjustable Thresholds and Delays
  - Eliminate Unknown Play/Pause State Error
- Multi-Platform
  - Android
  - iOS

www.scu.edu

SCHOOL OF ENGINEERING

SANTA CLARA UNIVERSITY

Acknowledgements

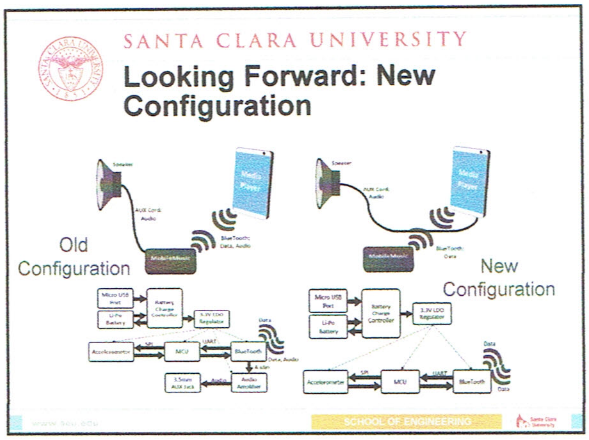
Santa Clara University, School of Engineering
The Avalon Academy
Dr. Shoba Krishnan
Dr. Timothy Hight
Dr. Radhika Grover
Anup Navare
Sam Pollock
Ethan Head
SCU Maker Lab

Santa Clara University
School of Engineering

The Avalon Academy®

www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

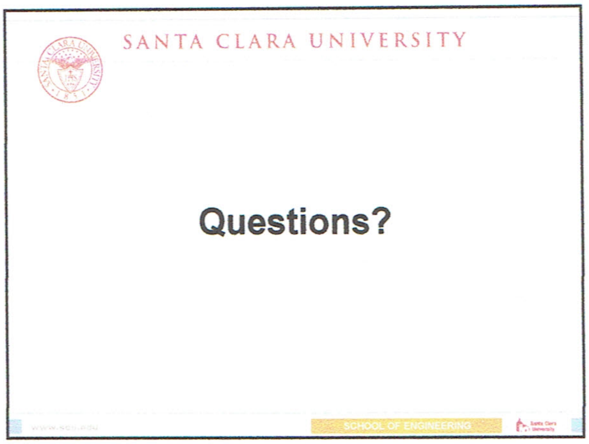
Looking Forward: New Configuration

Old Configuration

New Configuration

www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

Questions?

www.scu.edu

SCHOOL OF ENGINEERING