Wireless Midi Controller Glove

ECE 445 Design Document
Team 57 – Michael Brady, Sarah Palecki, and Allan Belfort
22 Feb 2018

Contents 2

1 Introduction

1.1 Objective

There are many musicians and DJs who create and perform music for a living or as a hobby all over the world, but many of these artists are restricted to creating and performing through the usual routine: pick up an instrument, play a tune, set down the instrument when finished playing, and then head over to the sound board to create effects. This is a process that takes time, and can seem boring to an audience who is watching a musician or DJ play music live. Nobody wants to watch a performer leaned over a soundboard for most of a performance! One of the most necessary uses for a soundboard arises in live mixing from DJs or artists, specifically in the electronic music category. Currently, the dance/electronic music genre is booming, with the genre rising to be the fifth most popular genre in the United States in 2016 [1]. According to the International Music Summit Business Report 2017, over "sixteen percent of people in the USA attended a club event with DJs in 2016" [1]. Clearly, the shows which DJs are playing are in high demand, but what if we can make these performances better?

Our goal is to mitigate the need for putting down an instrument, walking over to the soundboard, and hunching over it to create effects. Instead, we propose to streamline the process above by using a wearable, wireless MIDI Controller glove which could integrate effects such as volume control or panning position, all without pausing to put down an instrument and move to a soundboard. This way, with the flex of a finger or the flick of a wrist, effects can be added without the extra hassle of turning knobs and switches on a soundboard. Using flex sensors in the fingers and an accelerometer in the wrist, each gesture can be processed to create a different effect.

1.2 Background

There have been several attempts to make wireless MIDI controllers marketable in the past. Currently, they are priced at several hundred dollars, with one brand costing $300 [2], and another brand being launched on Kickstarter for a $199 pledge [3]. Before these two models, there was an unsuccessful Kickstarter campaign in 2014 called the "Mi.Mu Glove" which was backed by the artist Imogen Heap [4]. We envision that many artists who wish to use our gloves are performers who are starting out and will not be able to afford several hundred dollars worth of equipment.

Our glove is special for three reasons: it will be made cheaper than $199, it will achieve low latency while using Bluetooth, and it will be able to detect the flexing of the fingers. While the "Mi.Mu" glove was proposed to be made with flex sensors as well, the glove never reached the production phase. Thus, our glove will be the first of its kind to be produced with flex sensors and allow for finger movements to dictate desired effects.

1.3 High-Level Requirements

- Our gloves must be affordable compared to the gloves already on the market, ideally under $100.
- Gloves must be able encode gestures into MIDI signals, including the 30° flex angle of a finger (from the finger being held straight) or a 30° tilt angle of the hand (from the hand being held parallel to the floor).
- Gloves must be able to transmit MIDI signals via Bluetooth at least 1 meter away from the receiver which will be compatible with a computer Digital Audio Workstation (DAW) or a sequencer.

2 Design

Our design is composed of three separate blocks that will achieve successful functionality of the glove: the sensors on the glove, the control and communications block, and the power supply. The user will be able to send information to the DAW by performing different hand gestures. These will be recognized by the flex sensors on the glove's fingers or the accelerometer on the back of the hand which will be outputting readings into a microcontroller attached to the wrist of the glove. This module, in turn, will be processing these values and transmitting digital signals through the Bluetooth module attached to it. Other modules in this block include a power button, which will turn our system on or off, an LED that shows the power status of the system, and a button that starts a Bluetooth connection between the transmitter in the glove and the receiver connected to the DAW. Once these Bluetooth signals are received by the audio workstation, they are interpreted as modifications to a MIDI audio track, and will achieve variations in different audio parameters that can be controlled with this audio protocol. Finally, the power supply block will be in charge of delivering the necessary power to the modules in the processing and communications block of our glove, to ensure a proper and consistent functionality of our product.

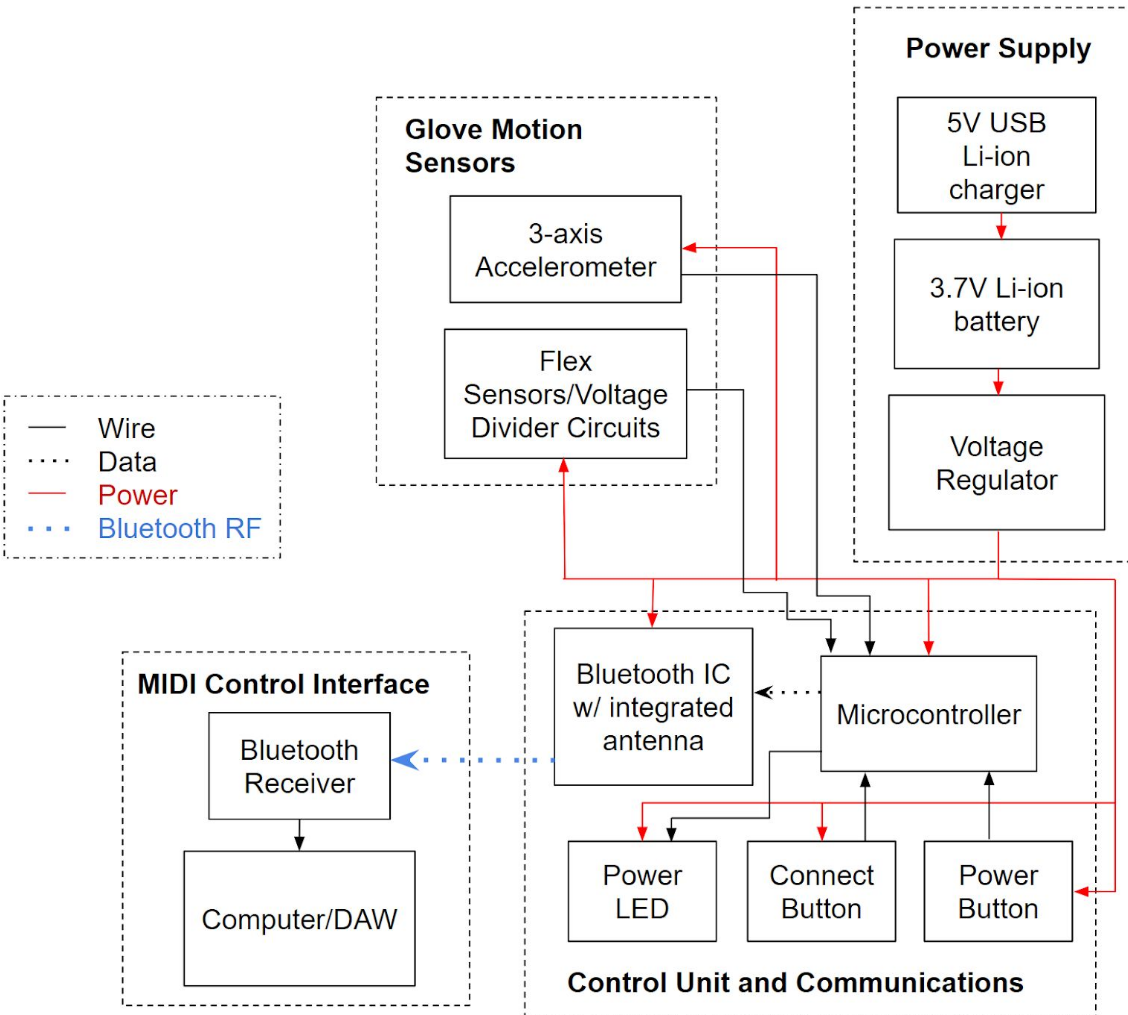

Figure 1: High Level Block Diagram

2.1 Glove Motion Sensors

This block modifies the signals sent to the control unit based on the gestures performed by the user wearing the glove. These gestures are recognized both by three individual flex sensors placed on the knuckles of the glove in all fingers excluding the thumb and little finger, as well as a three-axis accelerometer located on the back of the hand of the glove. These sensors communicate with the control unit and communications block, and are powered by the power supply block.

2.1.1 Three-Axis Accelerometer

This device will measure the 3-D tilt of the glove and the acceleration with which the glove is moved in any direction, and will be located on the back of the hand. It will send data to the microcontroller to be interpreted as information on the movements performed by the user.

Requirement	**Verification**
1. Must be able to detect acceleration up to ±1.0g on each of the three axes.	A.) The accelerometer will be powered by a 3.3V supply and each of its outputs (x,y,z) will individually be connected to an oscilloscope. It will then be slowly rotated along each of the three axes. B.) As long as the output for each axis varies throughout the entire rotation, this proves the device is able to detect acceleration up to and including 1.0g (due to gravity).
2. Accelerometer must be able to reliably detect orientation when relatively stable.	A.) The glove will be powered on and the microcontroller will be connected to a computer through USB to read serial output. B.) The specific program to take in accelerometer data and encode it into 0-127 digital values based on orientation will be run. The output must show a linear increase in the output value according to the degree of glove rotation for both left/right and up/down orientations.

2.1.2 Flex Sensors / Voltage Divider Circuits

Shown in the Circuit Schematic section (Section 2.6) is the voltage divider circuit that is employed by the flex sensors. This block interprets the bend of the finger via a flex sensor over the knuckle and, using the mentioned voltage divider circuit, changes the voltages being output to the microcontroller.

Requirement	**Verification**
1. Resistance level to be found upon testing, but must be able to change resistance by 2x original flat resistance upon bending it 90 degrees.	The flex sensor terminals will be connected to a multimeter and the resistance will be measured with the sensor flat, and then fully bent at ninety degrees. The resistance when bent must be at least twice the resistance from when it’s flat.

2. The voltage divider circuit must have a ΔVout of at least 0.5V when the flex sensor is fully bent	The output of the voltage divider circuit will be tested with a voltmeter. The output voltage will be recorded when the flex sensor is unbent, and recorded once again when the flex sensor is fully bent at ninety degrees. The difference should be at least 0.5V.

2.2 Control Unit and Communications

This block interprets the readings given by the sensors in the previous block and maps them to predetermined numerical values that will be digitally transmitted to the DAW through the Bluetooth interface. The modules in this block are powered by the power supply block, and information is transmitted wirelessly to the MIDI control interface block. The power LED and control buttons are also included in this block, which show the status of the system and control its operation.

2.2.1 Microcontroller

The microcontroller interprets the readings given by the motion sensors, and represents them as values that will serve as inputs to the MIDI interface. The information on these values and the functions they are meant to modify are then passed to the Bluetooth module.

Requirement	**Verification**
1. The microcontroller must be able to map the sensor readings into integer values to be processed digitally.	A.) Program a simple program into the microcontroller that is able to read the voltage level being fed into its analog pins, and outputs its value as digital values to the console.
2. Must be able to communicate with UART at rates greater than 50 kbps.	A.) Connect the microcontroller to a USB UART bridge and a terminal emulator (PuTTY) B.) Set the baud rate to 50 kBd (instructions shown by heading number 3.8.3.22 here: https://www.ssh.com/ssh/putty/putty-manuals/0.68/Chapter3.html [7]) C.) Send and receive back at least 50 characters D.) Check to verify all characters are correct

2.2.2 Bluetooth IC with Integrated Antenna

This chip must be both a bluetooth transmitter and have an integrated antenna. It will read the information supplied by the microcontroller and generate a bluetooth signal that will be interpreted by the bluetooth receiver connected to the DAW.

Requirement	Verification
Must provide connectivity over a 1 meter range	A.) The microcontroller will be connected to a computer through its micro usb connector in the FTDI USB UART circuit. The chip will be programmed to send MIDI packets at 100ms intervals. The receiver circuit will be connected via its USB UART converter with a micro usb cable to the computer and messages will be monitored using Putty on the console. B.) The glove and receiver will be placed 1 meter apart to demonstrate ability to transmit at range.
Must be able to receive information for transmission via a UART connection	
Must have an operating frequency range of 2402-2480 MHz, at rates greater than 50 kBd	A.) Using the program from the first verification, set the baud rate to 50 kBd. B.) Transmit 50 MIDI signals and check to ensure that all signals are received correctly via the console output.

2.2.3 Power Button and Latching On/OFF Function

This button will power the entire system on or off, taking the battery as its input. The output feeds into the voltage regulator. The circuit allows for a momentary push switch to turn on the circuit and allows the microcontroller to turn the device off on command. The circuit can also be turned off by holding the power button down over 3 seconds.

Requirement	Verification
Button must be easily pressible, and must not present mechanical malfunctions.	Ensure that button has no mechanical issues by pressing down upon it with little force.
Power circuit must turn on (output >3V) with one button press and turn off (output 0 ±.05V) after holding the button down over 5 seconds	A.) Hook the circuit up to a 4.2V power supply at the input terminals. Measure the output at the output terminals with a digital multimeter on voltmeter setting. B.) Press the push button once. The output must be >3V C.) Hold the push button down for 5 seconds. The output must be within .05V of 0.

2.2.4 Power LED

The power LED will inform the user on whether the system is on or off.

Requirement	Verification
The LED should be clearly visible to the user from at least 1 meter away, a distance longer than an extended arm.	A.) Using the simple LED circuit in shown in Section 2.6, check to see whether the LED turns on with the minimum drive current. B.) Measure 1 meter away from the LED. C.) Check to see if LED state is visible.

2.2.5 Connect Button

This button on the glove will make the bluetooth module enter pairing mode, where it will look for a bluetooth receiver to transmit its data to.

Requirement	Verification
Button must be easily pressible, and must not present mechanical malfunctions	Ensure that button has no mechanical issues by pressing down upon it with little force.

2.3 Power Supply

A power supply is necessary for our sensors to be giving relevant values, and for both our microcontroller and bluetooth IC to be performing their required functions. A battery will be the source of power, and its output voltage will be regulated to ensure consistency in the functioning of our system.

2.3.1 USB Li-ion Charger

This charger must be able to charge the Li-ion battery via USB port. The maximum output voltage of the charger must be less than the supplied 5V by the USB. We have chosen to use a 4.2V battery charger.

Requirement	Verification
Must have maximum output voltage less than or equal to maximum battery output.	A.) Discharge a Li-ion battery to 3.6V B.) Charge the battery at the output of the charger, which is supplied via 5V USB port. C.) When the “DONE” LED turns green and signifies termination of the charge cycle, check that the battery is charged to at most 4.2V with a voltmeter.

2.3.2 Li-ion Battery

The Lithium Ion battery must be able to provide power to all of the components so that the glove can operate wirelessly, without needing a power cable. It must also be small enough to not be an obstruction to the user wearing the glove.

Requirement	Verification
The battery must operate in the range of 3.7-4.2V and last over 2 hours.	A.) Connect a fully charged (as verified by a voltmeter) battery to the test circuit shown in section 2.6. The positive end should connected to the +V port and the negative end should be connected to the -V port. B.) Battery will begin to discharge at 200 mA. C.) Use a voltmeter to ensure that the battery voltage remains above 3.7V after 2 hours have passed.

2.3.3 Voltage Regulator

This linear voltage regulator chip supplies the required voltage to the blocks shown in the block diagram. It must be capable of having the capacity to output at least the upper bound of the li-ion battery (4.2V) and regulate the chosen battery output voltage +/- 5% from the source voltage.

Requirement	Verification
The voltage regulator must provide 3.3V and 5.0V output +/- 5% from a 4.2V source to simulate a fully charged Li-Po battery	A.) Connect the input and ground to a power supply on the voltage setting. Set the supply to output 4.2V. B.) Measure the output of the circuit using an oscilloscope, ensuring that the output stays within 5% of the required voltages.

2.4 MIDI Control Interface

This block represents the receiving end of the data that will be sent from the glove. After data from the sensors has been collected, processed and transmitted, the MIDI control interface will receive the bluetooth signals and input them to the DAW.

2.4.1 Bluetooth Receiver

This module will be a bluetooth receiver with an antenna to receive signals coming from the glove's bluetooth transmitter. It will output the signal via USB to the DAW.

Requirement	**Verification**
Must be able to operate within the range of the transmitter (at least 1 meter) and receive signals from it without loss of information.	A.) The microcontroller will be plugged into a computer and a program (such as PuTTY, mentioned above) will be run to send multiple test packets similar to the MIDI signals sent during normal operation. The receiver circuit will be connected via USB to the computer and messages will be printed via a serial output on the console. B.) The glove and receiver will be placed 1 meter apart to demonstrate ability to transmit at range.
Must be able to send out information to the DAW via a USB connection	A.) The same procedure as above will be used, but the receiver circuit will send messages to the DAW instead of printing them to the console. B.) The DAW should show the signals coming in and modifying the selected parameter.

2.4.2 Computer/DAW

At this stage, the signals received from the glove will be interpreted as modifiers of sound parameters of a MIDI signal. This will be done in a Digital Audio Workstation (DAW) software running on a commercial computer equipped with the required input ports.

Requirement	**Verification**
Must be equipped with 1GB of RAM and MIDI interface	Computer specifications will show RAM and MIDI interface software must be downloaded.
Must be able to run Symbolic Sound's Kyma DAW software.	A) Computer specifications must match those required to run Kyma, as per http://kyma.symbolicsound.com/installing-kyma-7/ B) Software is correctly installed and runs on the desired computer.

2.5 MIDI Encoding

The signals which will be communicated over Bluetooth to the DAW are those of the MIDI protocol. We chose to use MIDI encoding as it is a standard communication type used by many of today's musicians: "MIDI is an industry standard music technology protocol that connects products from many different companies including digital musical instruments, computers, tablets, and smartphones" [10]. Midi encoding works by sending "messages", which are composed in binary from several different bytes of information. On average, each "message" ranges from containing three to four bytes.

There are two types of bytes that MIDI uses to form its messages: the "Status Byte" and the "Data Byte", and each has a separate functionality. The MIDI processor can look at the most significant bit of each byte that is being processed to tell if the byte is classified as a "Status Byte", which has an MSB of 1, or a "Data Byte", which has an MSB of 0. An example of a Status Byte is shown below [11]:

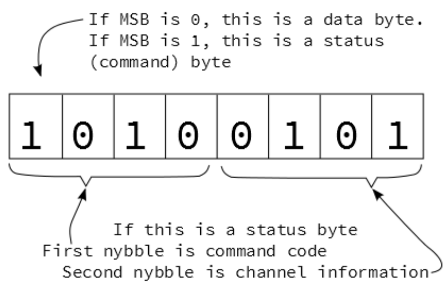

Figure 2: Description of MIDI byte

Status Bytes specify the command type that the MIDI processor will be performing and which channel it will be performed on. The first 4 bits of the byte encode the command type and the last four bits of the byte encode which channel will be used.

Data Bytes are bytes that are sent to the processor after a Status Byte has been sent, and describe the specific data that should be encoded for the specific command type that was listed in the first 4 bits of the Status Byte. Data Bytes must have an MSB of 0, as mentioned above, so that leaves 7 bits left of the byte that can be used to store data- or 127 separate values.

In summary, a Status Byte tells the processor which command will be given and to which channel. A Data Byte tells the processor more information about the command sent by the Status Byte by telling the processor the exact value that we would like to change. Per the design of the gloves, we will be encoding changes in two different Midi messages: "Note On" and "Program Change". The tilt of the wrist will change the "Note On" message while the flex of a finger will change the "Program Change".

The “Note On” message works as shown below:

MIDI Status Messages					
Message Type	MS Nybble*	LS Nybble*	Number of Data Bytes	Data Byte 1	Data Byte 2
Note On	0x9	Channel	2	Note Number	Velocity

* A Nybble is equivalent to 4 bits

Figure 3: ‘Note On’ MIDI message description

In “Data Byte 2” of the “Note On” message, it can be seen that there is a value for “Velocity”. This 0 to 127 value encodes the volume of the note. This “Velocity” value in “Data Byte 2” of the “Note On” message is the value we will be changing when the wrist is tilted. A value of 0 is equivalent to no volume, or a note not playing, while a value of 127 is the loudest a note can get. When the wrist is held flat, there will be a “Velocity” of 0 being sent to the MIDI Processor. However, as the wrist is tilted and the angle that the hand makes with the floor is increased, the “Velocity” value being sent to the MIDI Processor will increase.

Next, the “Program Change” message is shown below:

MIDI Status Messages				
Message Type	MS Nybble*	LS Nybble*	Number of Data Bytes	Data Byte 1
Program Change	0xC	Channel	1	Program Number

* A Nybble is equivalent to 4 bits

Figure 4: ‘Program Change’ MIDI message description

In “Data Byte 1” of the “Program Change” message, it can be seen that there is a value for “Program Number”. The 0 to 127 value encodes the sound (patch) of the note. For example, a Program Number of 0 could be a standard keyboard sound, while a Program Number of 32 could be accordion sounds. The specific patch values are pre-set by the user using the DAW. For our glove purposes, we will be able to switch between 3 different patches at a time which are pre-set by the user before powering on the glove. If one of 3 fingers attached to a flex sensor is bent, the Program Number will change.

In conclusion for added clarity, an example of a “Note On” and “Program Change” message are shown and described below:

Note On: 10010000 00111100 01110010

Hex: 9 0 3 C 7 2

Explanation of each byte broken down:
- 9 : Status Byte encoder for "Note On"
- 0: Encoded to Channel 0
- 3C: Note that is being played on a range from 0 to 127. Here, the value is 60 (3C in hex = 60) and corresponds to middle C
- 72: The velocity of the note on a range from 0 to 127. Here, it is 114 (72 in hex = 114) This is the value we will be changing.

Program Change: 11001111 00010001

Hex: C F 1 1

Explanation of each byte broken down:
- C: Status Byte for "Program Change"
- F: Channel 15
- 11: Patch Number. Here, it is 17 (11 in hex = 1). This is the value we will be changing.

2.6 Schematics

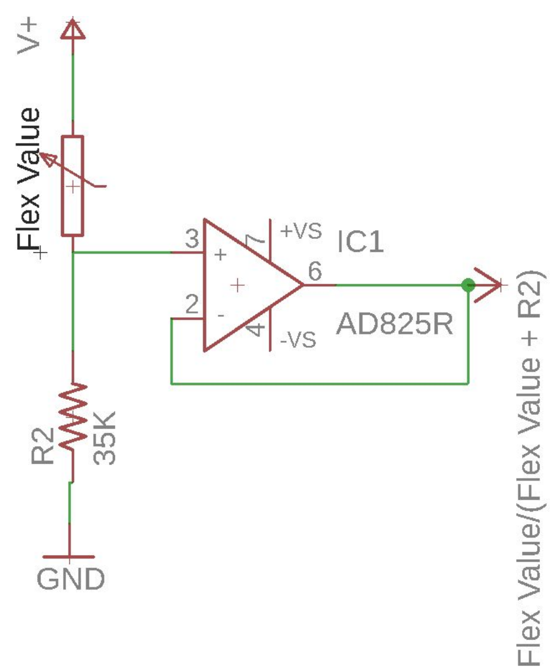

Figure 5: Flex Sensor Voltage Divider Circuit

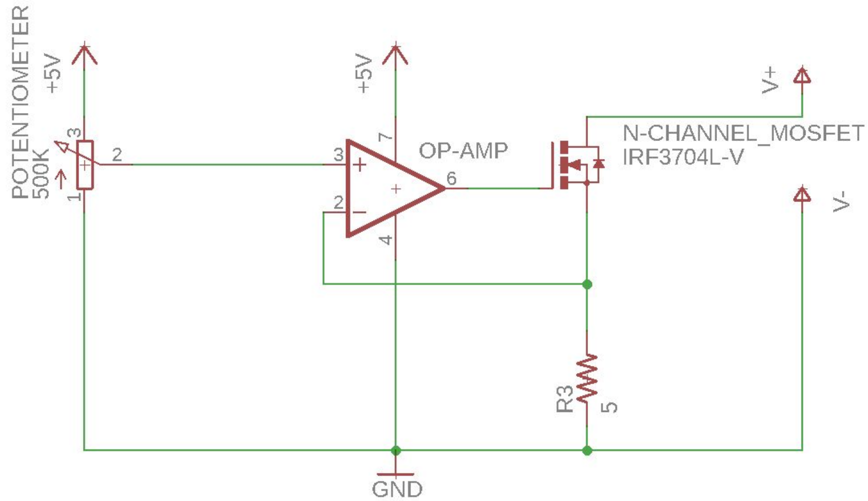

Figure 6: Battery Discharge Circuit

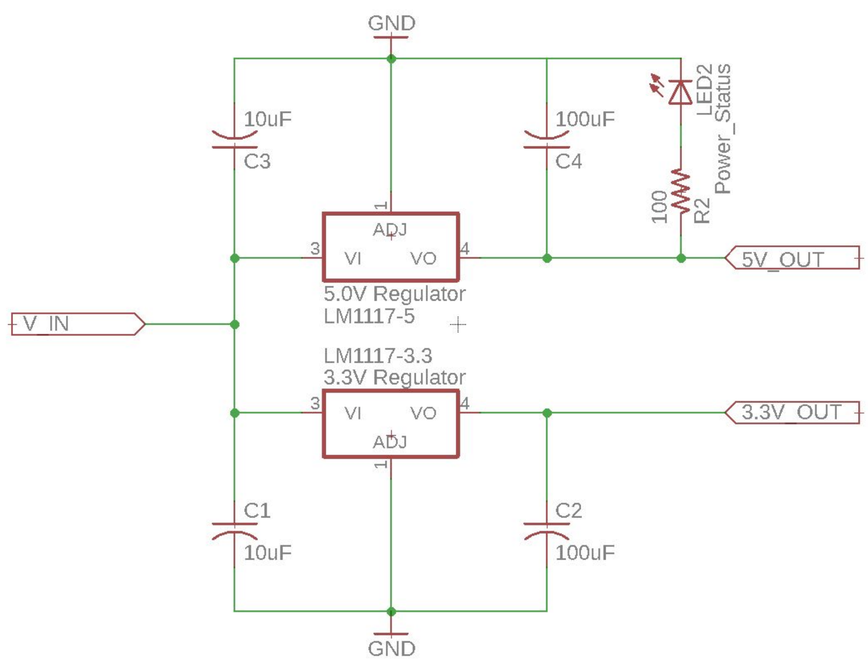

Figure 7: Linear Voltage regulator circuit schematic with power button and indicator

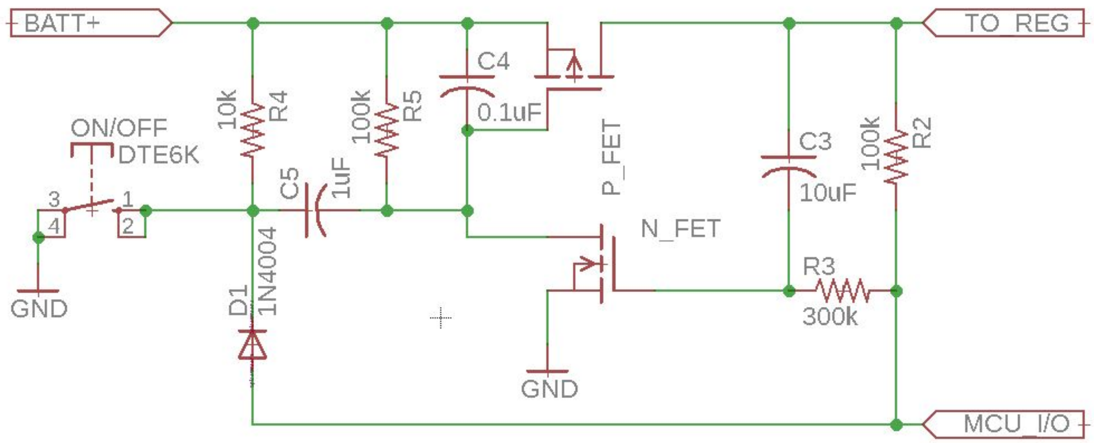

Figure 8: Latching push-button power controller schematic [8]

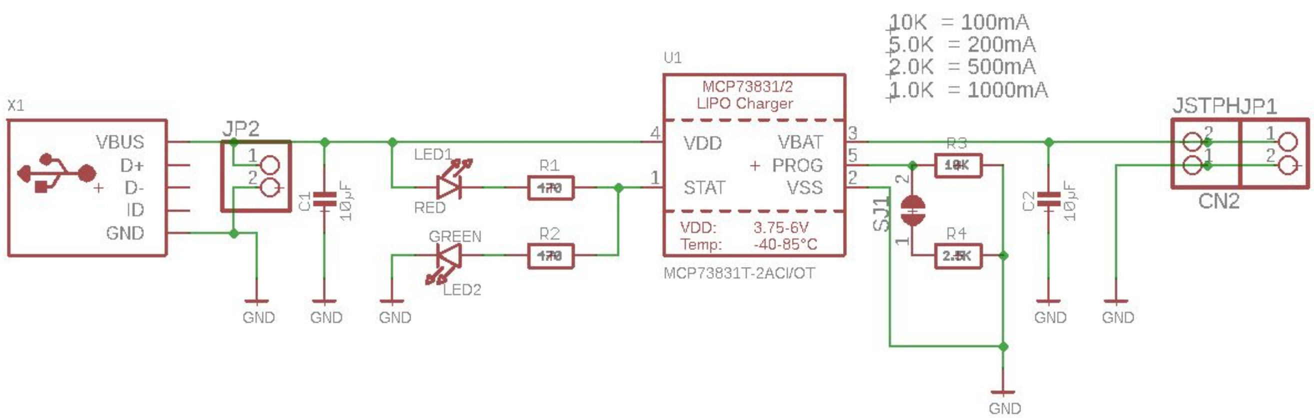

Figure 9: 4.2V battery charging circuit schematic [9]

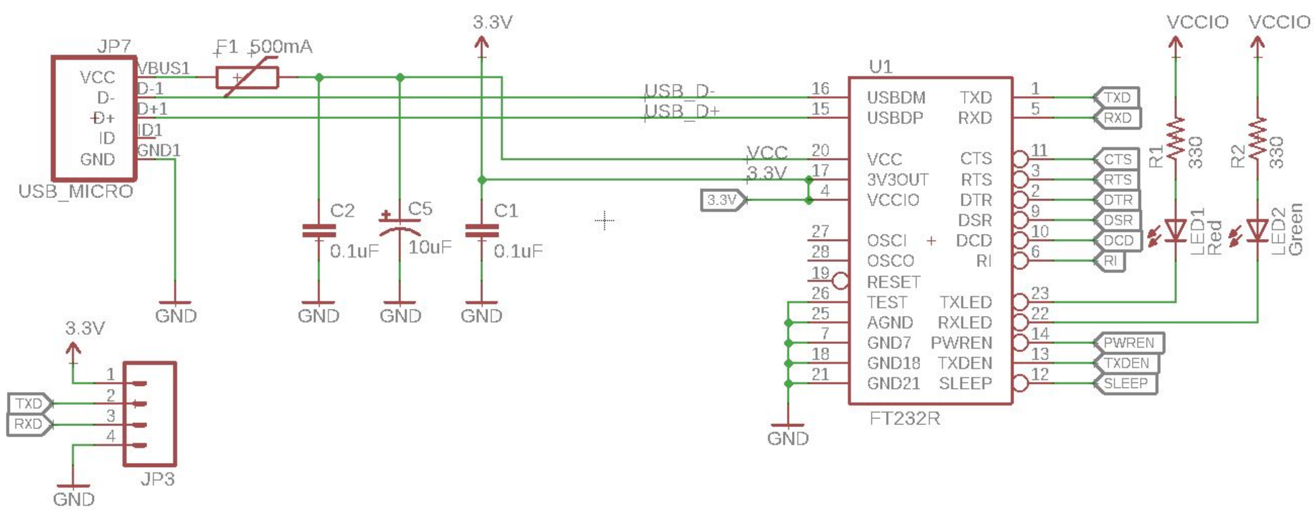

Figure 10. FTDI UART USB Serial converter circuit schematic

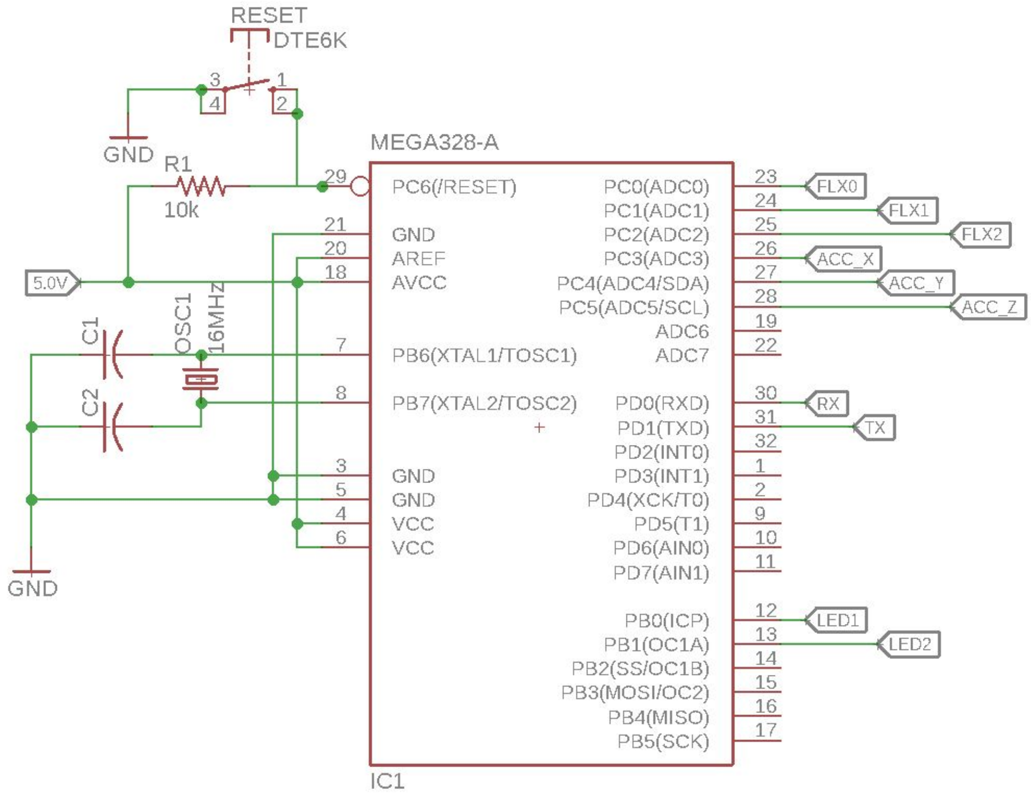

Figure 11: Microcontroller circuit schematic

2.7 Risk Analysis

The biggest threat to a consistent functionality of our system is the accuracy of the three-axis accelerometer that will be attached to the glove. This component is meant to output a reading proportional to the acceleration of the hand, and the direction in which it is being moved. The greatest challenge in implementing it into our design is in getting the correct readings for how the hand is being moved, as such a small component is hard to calibrate and consequently hard to ensure that the values it outputs are consistent. This is a critical item in our design, as it will be the measurement device for multiple of the gestures we want to detect with the glove. The flex sensors are only able to detect the curl of the fingers, while the three-axis accelerometer will be able to measure the direction of the hand in two different axis and how much it moves in each direction. It is therefore important to focus on getting this component to work properly, but its functionality goes beyond our implementation of it in our system and has us relying on the chip manufacturer's design of the piece.

Much of the task of interpreting the motion of the hand also falls on the microcontroller block, which maps the accelerometer readings into data that can be processed by the DAW. The microcontroller can be made to counter some inconsistencies on behalf of the accelerometer, and cut off some of the error from the gesture recognition portion before it is sent to the DAW through a Bluetooth signal. In other words, there is *some* ameliorating that can be done in the microcontroller if the accelerometer does not meet the requirements detailed previously in this section. This would mean, however, that the mapping performed by the microcontroller would not be able to be as sensitive with respect to the degree at which the hand is moved, and might mean that some sensitivity in gesture reading will be lost in this regard.

If the accelerometer gives inaccurate or imprecise readings, we can resort to calibrating other parts of our system to counter any errors from the component, but will come at a loss to the sensitivity of the gesture recognition and its effects on how much the MIDI parameters are modified.

2.8 Physical Design

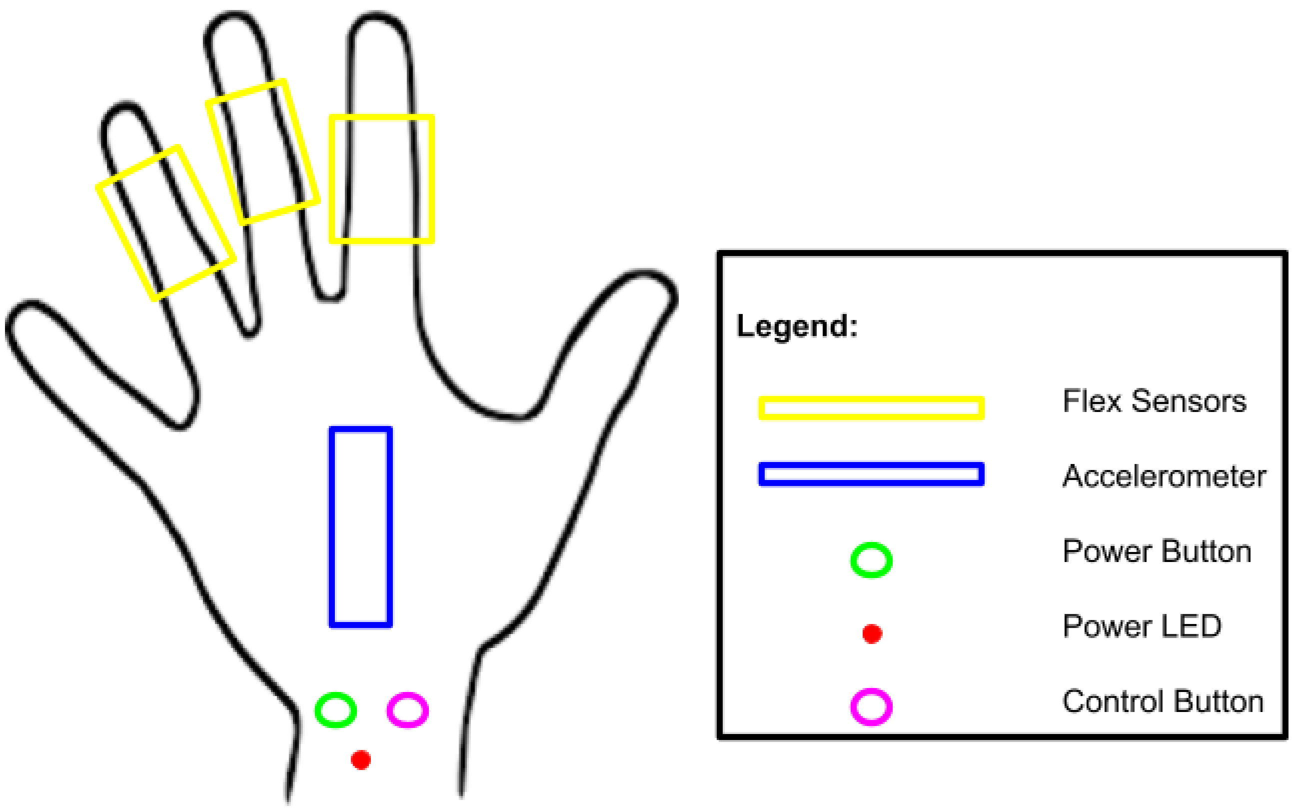

Figure 12: Physical device layout

The diagram above shows the layout of our components on a glove that covers the user's hand. Our diagram shows the glove as it would appear when on a hand that is pressed flat. As can be seen from the yellow rectangles in the picture, we propose to implement three separate flex sensors on each of the fingers of the hand (not including the thumb). These flex sensors will be put over the knuckles of each finger on the back of the hand. Next, it can be seen from the diagram that we propose to implement an accelerometer on the back of the hand, as indicated by the blue rectangle. Additionally, there will be two separate buttons near the wrist of the glove: one power button as well as one control button. These are represented by the green and pink circles on the diagram, respectively. Finally, we would like to include one LED so that the user can tell if the glove has been powered on or off. We will be using a flexible yet sturdy fabric that allows the user's hand to breathe and comfortably make several different gestures without creating moisture inside the glove. We are proposing to use biking gloves, or another glove in which the fabric does not cover the fingertips. This is so that the user can still use touchscreens and easily grip items without the concern of slip due to fabric.

2.9 Tolerance Analysis

Some of the most critical components of our project are the flex sensors used to measure the amount of bend in the fingers of our glove. The bend of the fingers accounts for 3 of the 5 different control effects that the glove is able to modulate (2 for hand tilt and 3 for finger bend). It is important that these sensors give consistent and accurate readings so that user inputs provide predictable control. This directly affects the sensitivity of the voltage divider circuit that each sensor controls, as the voltage levels they output have to vary by at least 0.5 volts, per our requirements. This means that the output voltage should be at least half a volt higher when the sensor is flexed versus when it is unflexed. In the voltage divider circuit we'll use (shown in section 2.6), the formula that describes the output voltage is given by

$$V_{out} = V_{in}\left(\frac{R_f}{R_2+R_f}\right).$$

In order to maximize the output voltage, either the input voltage has to be increased or the fixed resistance, denoted as R_2 in the equation above has to be decreased. The resistance of the flex sensor when it is unflexed, R_f, is an independent variable which we cannot modify. To calculate the sensitivity of the voltage divider circuit it is necessary to characterize the behavior of the voltage divider circuit with a fully flexed sensor at 90^o. This leads to the following equation:

$$V_{out} + B = V_{in}\left(\frac{AR_f}{R_2+AR_f}\right).$$

where B denotes the sensitivity of the voltage divider circuit, and A denotes the constant by which the resistance of the flex sensor increases with respect to its unflexed resistance. The requirements of our system are for B to be greater than 0.5V, and A to be greater than 2. In other words, the output voltage of the circuit should increase by at least 0.5V if the resistance of the flex sensor increases by a factor of 2 after being flexed. After manipulating the above equations, a conclusion on the sensitivity of the circuit can be achieved, which can be modelled as

$$B = \frac{V_{in}R_2R_f(A-1)}{R_2^2+R_2R_f(A+1)+AR_f^2}.$$

In order to maximize the sensitivity of the circuit, the derivative of B with respect to R_2 has to be found, which leads to

$$\frac{dB}{dR_2} = \frac{V_{in}R_f(A-1)((R_2^2+R_2R_f(A+1)+AR_f^2)-(2R_2^2+R_2R_f(A+1))}{(R_2^2+R_2R_f(A+1)+AR_f^2)^2}.$$

Setting the above relation equal to zero, and solving for R_2, we conclude that

$$R_{2,opt} = \sqrt{A}R_f$$

This suggests that the optimal value of the fixed resistance to be used in each voltage divider circuit is proportional to the unflexed resistance of the flex sensor, by a constant equal to the square root of its flex ratio.

We tested two flex sensors to obtain practical values of R_f, and determine if a flex ratio of 2 could be achieved. This was done by testing the unflexed and 90-degree flex resistance of each sensor, in order to test repeatability and consistency in our readings. It is common for these components to be inconsistent in their readings, so we wanted to have an idea of the range of resistances we would be dealing with and model them accordingly. Each sensor had its two terminals connected directly to a multimeter, from which we read the resistance values. The degree of bend was measured with a protractor and is said to be the angle between the base and the tip of the sensor when it is flexed. We also ensured the readings were consistent by testing each sensor multiple times consecutively, and not letting much time pass between tests to simulate consistent and unpaused flexing and use. The data obtained is reproduced below.

Sensor 1 Resistances (kOhms)		
	Bend Degree	
Trial #	0	90
1	24.7	42.8
2	22.9	43.5
3	23.1	44.7
4	23.2	43.3
5	23.2	44.5
6	22.7	41
7	22.7	45.7
8	23	44.5
9	23.1	44.8
10	22.5	45.5

Table 1: Flexed and Unflexed Resistance Measurements of Sensor 1

Sensor 2 Resistances (kOhms)		
	Bend Degree	
Trial #	0	90
1	27.9	59.9
2	27.4	60
3	26.3	60
4	26.3	59.1
5	26.8	58.6
6	26.8	58
7	26.9	58.7
8	26.2	56.2
9	25.8	58
10	26.1	56.7

Table 2: Flexed and Unflexed Resistance Measurements of Sensor 2

From the collected data we were able to determine that the flex ratio is of 2.03 in sensor 1, and 2.32 in sensor 2 when using the minimum and maximum values of resistances at each bend level. The average unflexed resistance using data from both sensors is 24.88kΩ. The minimum unflexed resistance is 22.5kΩ from both sensors, and the maximum unflexed resistance is 27.9kΩ from both sensors. From this data we can analyze the range of fixed resistance values needed to maximize the sensitivity of our circuit, given that the input voltage is 5V and A meets our minimum requirement of 2. Graphing sensitivity versus fixed resistance using the data obtained above, we are able to determine a range of fixed resistance values.

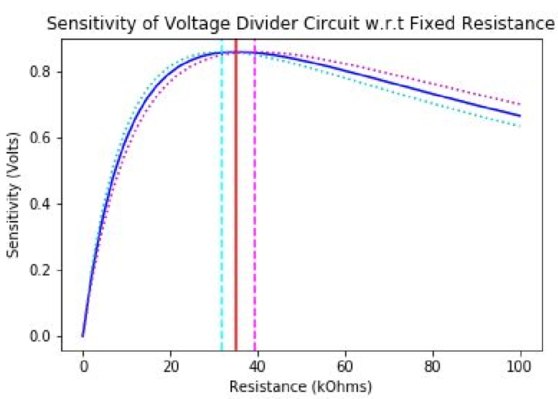

Figure 13: Graph of Fixed Resistance vs. Sensitivity over range of 0-100kΩ

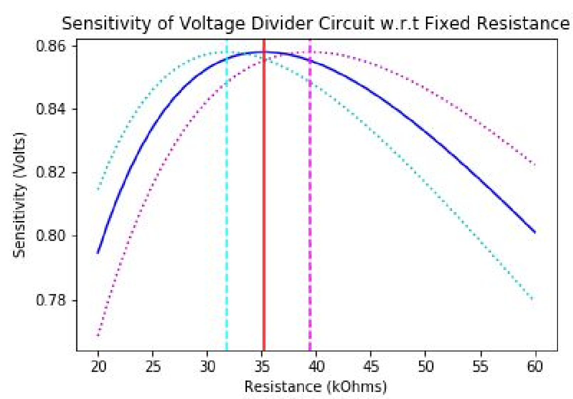

Figure 14: Graph of Fixed Resistance vs. Sensitivity over range of 20-60kΩ

Both of the above graphs show the same relationships, but have different ranges in the fixed resistance axis in order to improve data visualization. The three curves shown above are the sensitivity relations when using the minimum (blue dotted), maximum (pink dotted) and average (blue solid) R_f found when characterizing the sensors at hand. The vertical lines indicate the optimal fixed resistance value that maximizes the sensitivity for each of the three curves, with the solid red vertical line being the fixed resistance used if the unflexed resistance of the sensor was the average R_f observed. Said line is traced at a value of 35.19kΩ, which causes the sensitivity to be of 0.858V, thus surpassing our requirement. If this value were used for a fixed resistance in our circuit, but R_f was inconsistent in the resistance that the circuit observed, we can estimate the variation in sensitivity given the curves plotted using the maximum and minimum values obtained in our data. This requires us to find what value for sensitivity would be given if $R_2 = 35.19k\Omega$ and R_f is in the range of values obtained. When carrying out these calculations, we find that the maximum deviation from the 0.858V point is of 0.0027V. In other words, if an unflexed resistance lies within the range of values observed when characterizing the flex sensors, the maximum deviation in the sensitivity of the circuit would be of 0.0027V below the maximum sensitivity, or a decrease of 0.3%. Similarly, a range of fixed resistances between 31.82kΩ and 39.45kΩ can be chosen in order to affect the sensitivity in a similar way, as denoted by the resistances that maximize the curves created with the maximum and minimum unflexed resistances in the graphs above. Such a small variation is negligible for the functionality of our circuit, as our microcontroller will not be as sensitive to such a voltage difference in order to avoid noise or unwanted movements to affect the MIDI values being transmitted to the DAW.

3 Costs

An average recent graduate from the ECE department will make $68,000, which is about $33.00/hour. We believe that the education level of recent graduate is sufficient enough be able to work on the production of these gloves. Thus, we thereby believe that our fixed development costs are about $13,440:

$$3 \,*\, \$33/hr. \,*\, 7hr./week \,*\, 16\ weeks\ = \$13,440$$

Our prototype costs are estimated to $103.65, as shown below:

Part	Cost (Prototype)	Cost (Bulk)
Sparkfun 2.2” Flex Sensors (x3) (Sparkfun; SEN-10264)	$23.85	$21.48
Triple Axis Accelerometer (Analog Devices; ADXL35)	$14.95	$11.96
Class 1 Bluetooth Module x 2 (Roving Networks; RN-41)	$46.00	$42.06
USB LiIon/LiPoly charger (Adafruit; MCP73831)	$6.95	$5.56
Lithium Ion Polymer Battery - 3.7V 500mAh (Adafruit; 503035)	$7.95	$7.16
Voltage Regulator -1.7V to -37V (ECE SHOP; LM1117)	$0.85	$0.85
PCBs (PCBWay)	$3.10	$0.10
Total	**$103.65**	**$89.17**

Thus, overall, we are estimating that the total cost of this product is $13,543.65.

4 Schedule

Week	Sarah	Allan	Michael
2/26	Order major parts and prepare for design review		
3/5	Write test program for MIDI data transfer	Work on communication with Kyma	Get arduino working with computer
3/12	Voltage Dividers x3 PCB design finished Order minor parts	Battery discharge circuit finished Order minor circuit parts	Power Supply/MCU PCB design finished Order minor circuit parts
3/19	Spring Break	Spring Break	Spring Break
3/26	Get respective circuits soldered and verify functions. Do progress reports.		
4/2	Write program to interface devices		Configure MCU
4/9	Test Bluetooth Bit-rate	Test bluetooth range	Test bluetooth dropout
4/16	Build device and test entire system		
4/23	Troubleshooting/practice using for demonstration/work on final paper		
4/30	FINAL DEMO	FINAL DEMO	FINAL DEMO

5 Ethics and Safety

Within our proposed design, there are several safety concerns. One of the necessary implementations of the project is to have the glove be rechargeable when being used wirelessly. Thus, it will require rechargeable batteries. We propose to use Li-Ion rechargeable batteries for this glove, which can create certain concerns as they may explode or burn if misused or mishandled. It is important to ensure that there is no internal short circuit, as this leads to increased heat within the battery and an exothermic reaction, increasing the risk of additional combustion [5]. To ensure this is not the case, we will test the batteries thoroughly before using in the glove. Another concern with these batteries possible explosion upon overcharging. In order to ensure this will not be the case, we will test the batteries thoroughly by charging them ourselves before implementing them within the glove. It is also important to stress that high quality and reliable charging systems must be used to charge the batteries. A link to a lab safety guide, detailing to the user how to handle the battery, is provided to the user here: https://cdn-shop.adafruit.com/product-files/1578/1578+msds.pdf

Another safety concern is the glove coming into contact with moisture. We propose to use a moisture resistant fabric as well as use rubber coatings on any wires or electrical contact that is being made in order to prevent the user from experiencing any shock. Along these same lines, it is important to store the glove in a cool, covered area so that the glove does not overheat.

One ethical concern is the use of the gloves to create and copyright music which has already been made. This follows the IEE Code of Ethics 7.8.2 [6]. It is important to ensure that artists are not copying previously made works and the violation of interests do not occur. This follows along the line of modifying music and sounds that are under copyright. It is not our intention to make a product which enables the infringement of musical property. Thus, it is up to the user to make these types of ethical decisions.

We believe that this product falls within the lines of IEEE ethics 7.8.1 [6], and it is our goal to comply with ethical design and sustainable development practices for this project. Thus, we will disclose any and all possible safety and ethical issues to the user, so that we may fully disclose any limitations our product has in order to adhere to IEEE ethics 7.8.6 [6].

6 References and Citations

[1]Watson, 'IMS Business Report 2017', 2017. [Online]. Available: http://www.internationalmusicsummit.com/wp-content/uploads/2017/09/IMS-Business-Report-2017-vFinal3.pdf [Accessed: 2- Feb- 2018].

[2] Global DJ, 'Tornado A1 | MIDI Controller Wireless 3D-Gloves', 2018. [Online]. Available: http://global-dj.com/product/tornado-a1-midi-controller-wireless-3d-gloves/ [Accessed: 2- Feb - 2018].

[3]B. Rogerson, 'The gloves are in on the wireless MIDI control battle', 2016. [Online]. Available: http://www.musicradar.com/news/tech/the-gloves-are-on-in-the-wireless-midi-controller-battle-635024 [Accessed: 2- Feb- 2018].

[4] M. Mu, "Mi. Mu Gloves for music", 2014. [Online]. Available: https://www.kickstarter.com/projects/mimu/mimu-glove-for-music [Accessed: 2- Feb- 2018].

[5]L. Florence, "Safety Issues for Lithium Ion Batteries", 2012. [Online]. Available: https://www.ul.com/global/documents/newscience/whitepapers/firesafety/FS_Safety%20Issues%20for%20Lithium-Ion%20Batteries_10-12.pdf [Accessed: 3- Feb- 2018].

[6] Ieee.org, "IEEE IEEE Code of Ethics", 2018. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: 3- Feb- 2018].

[7] SSH Communications Security, Inc., "Chapter 3: Using PuTTY", 2018. [Online]. Available: https://www.ssh.com/ssh/putty/putty-manuals/0.68/Chapter3.html [Accessed: 20- Feb- 2018].

[8] Mosaic Documentation Web, "Latch and Toggle Power Circuits". [Online]. Available: http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/electronic-circuits/push-button-switch-turn-on/latching-toggle-power-switch [Accessed: 21- Feb- 2018].

[9] Adafruit, "Adafruit Micro Lipo w/MicroUSB Jack - USB LiIon/LiPoly charger - v1". [Online]. Available: https://www.adafruit.com/product/1904 [Accessed: 21- Feb- 2018].

[10] Midi.org, "What's Midi?". [Online]. Available: https://www.midi.org/ [Accessed 22- Feb- 2018].

[11] J. Byron, "MIDI Tutorial". [Online]. Available: https://learn.sparkfun.com/tutorials/midi-tutorial/messages [Accessed 22- Feb- 2018].