WIRELESS MIDI CONTROLLER

By

Allan Belfort

Michael Brady

Sarah Palecki

Final Report for ECE 445, Senior Design, Spring 2018

TA: Anthony Caton

02 May 2018

Project No. 57

Abstract

We have created a Wireless MIDI controller glove that is capable of communicating by MIDI protocol as an electronic music instrument. The glove is capable of manipulating MIDI output in two different ways: sending MIDI “Note On” messages and MIDI “Control Change” messages. By bending different fingers on the glove, effects such as pan, vibrato, or pitch bend are controlled via a MIDI “Control Change”. Through a tilt of the wrist, a volume change is initiated via a MIDI “Note ON” message. By utilizing these two MIDI Controller encodings, we allow an electronic music artist to comfortably change the volume or sound of the output from their electronic MIDI capable instrument.

Contents

1 Introduction

1. Block Diagram

In order to achieve our goal of creating a Wireless MIDI Controller, we have created a block diagram to visually show communication between all necessary design functions. Our design is composed of four separate blocks that will achieve successful functionality of the glove: the glove motion sensors, the control and communications, the MIDI control interface, and the power supply. Our end goal is to manipulate user generated MIDI code. This is done by sending information to the Digital Audio Workstation (DAW) when the user performs different hand gestures. Thus, a performance requirement is that the user will need to have a DAW installed on their computer (such as FL Studio). These hand gestures are registered by the 3-axis accelerometer and the 3 flex sensors, located in the Glove Motion Sensors block of Fig 1. These sensors output analog voltages to the microcontroller located in the control unit and communications block. A power button is attached to the microcontroller that, when pushed, turns on the system and lights up a red LED (referred to as "Power Led" in Fig. 1). Upon receiving values from the glove motion sensors block, the microcontroller will output values via the Bluetooth IC with integrated antenna. The Bluetooth IC must be able to perform at a baud rate of 115200 bits per second. These values are received by a Bluetooth receiver, run through a serial to MIDI Converter (Hairless MIDI), output to a virtual MIDI port (LOOP MIDI), and then processed by the DAW in the MIDI Control Interface block, according to Fig. 1. Finally, the power supply block consists of a 5V USB battery charger, which charges a 3.7V Li-ion battery. The output of this battery is sent through a voltage regulator, ensuring the output voltage is within +/- 5% of the 3.3V, which in turn powers the system.

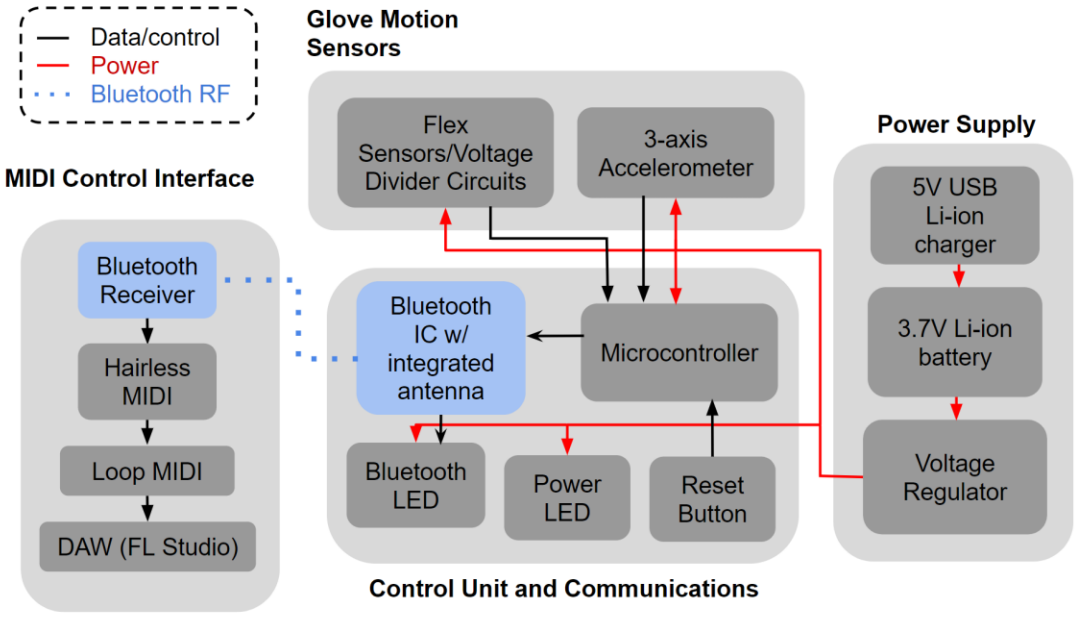


Figure 1. Block Diagram

2 Designs and Verifications

2.1 Design Procedure

We had many different design decisions to make during our design process. These included weighing different options for the glove motion sensors, the power supply, the control unit and communication, and the MIDI control interface.

For the glove motion sensors, it was important that the user could move their hand in such a way that it would be easy to facilitate, trigger a sensor response, and cause no strain. For this reason, we chose to use 3 flex sensors (located on the ring finger, middle finger, and pointer finger of the hand) that the user could operate to do a MIDI "Control Change". Initially, we had decided upon using 4 flex sensors, with an additional flex sensor in the little finger. However, upon testing, we realized it was hard to flex the little finger without also flexing the ring finger, and feared this would mess with the integrity of sensor outputs. Attached to the output of each of these 3 flex sensors is a voltage divider circuit. The general circuit design is shown in Fig. 2. The voltage output of this circuit is determined by Eq.1 shown below:

$$V_{out} = V_{in}\frac{(R_{flex})}{(R_{flex} + R_2)} \tag{1}$$

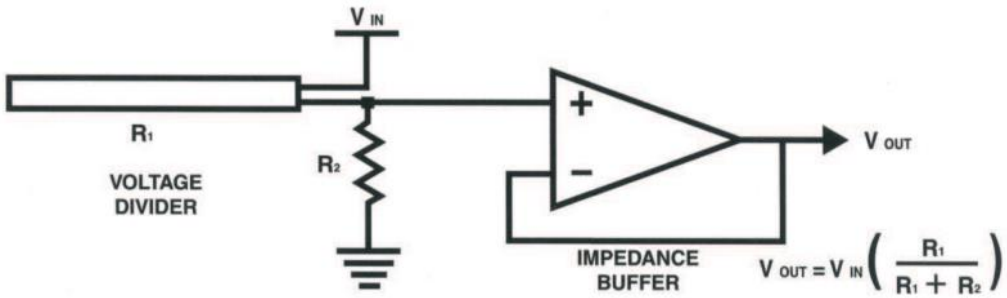


Figure 2. Voltage Divider Circuit

We also made several decisions regarding the power supply of our Wireless MIDI Controller. Due to the wireless nature of the product, it was important to choose a power supply that would enable the user to be away from their MIDI equipment as well as from a wall outlet. For this reason, we chose to use a rechargeable battery to power our system. We found a 300mAh Li-Ion battery that will operate our glove for a minimum time of two hours. In order to ensure easy operation, we chose to use a 5V USB charging device that the user could easily plug in to a computer to charge the glove. We pursued this option rather than using a charger cable that would connect to a wall outlet as we truly wanted this device to be as compact as possible and eliminate the need for a charging cable. The power supply circuits are shown in the "Design Details" section under Fig.6, Fig.7, Fig. 8, and Fig. 9.

There were also decisions to be made for the control unit and communication. As the wireless feature of the MIDI controller is essential, we chose to use Bluetooth to transmit our data wirelessly to the

computer as most modern computers come with built in Bluetooth functionality. Additionally, we have chosen to use a power button that the user may press in order to turn the glove on or off. This was the simplest and cheapest option. We initially had proposed to use an additional button, the "connect button"; however, upon designing our PCB board, we evaluated it would not be of much use to our system and removed it from our circuit. As a result, the glove design is even simpler as the user only needs to press the power button to turn the glove on or off. There is a red LED installed that lights up when the glove is on so the user can know the power status of the glove. Finally, we decided on using the ATMega328 microchip as the microcontroller unit where the inputs and outputs of our system would be handled. This chip was chosen because it has 23 I/O pins, 23KB ISP flash memory with read while write capabilities, and a SPI serial port. With this functionality, we could connect the 3 flex sensors, 3 accelerometer inputs, and any additional sensors we might have chosen to add to the glove if we pleased. This was the cheapest, most reliable, and smallest microchip we could find with this functionality. Size was an important consideration as we wanted our PCB board to be as small as possible in order to fit comfortably on the back of the hand. The control unit and communication circuits are shown in the "Design Details" section under Figure 7.

Finally, it was important to consider different options for the MIDI control interface. We chose to use a DAW as this is what most musicians in electronic music use to edit output from their respective MIDI instruments. The DAW we chose for our project was FL Studio. In order to communicate with the DAW, we first needed the computer to recognize our glove as a MIDI-enabled device, which required software to first change serial output into MIDI-recognizable output. We also needed software to create a virtual MIDI port that our code could be put into and forwarded to be recognized by the DAW. The simplest and most straightforward options for these two functions were to use Hairless MIDI and LOOP MIDI, respectively, both of which will be described in greater detail later in this report.

2.2 Design Details

All of our requirements were met via the verification strategies listed in each device's respective Requirement and Verification Table in Appendix A. The exception to this is the Power Button, which we have gone into more detail about in Section 4.2 "Uncertainties".

2.2.1 Three-Axis Accelerometer

This device measures the 3-D tilt of the glove and the acceleration with which the glove is moved in any direction. It is located on the back of the hand. It can sense acceleration up to +/-3G on the X, Y, and Z axes. It was important to choose an accelerometer that can reliably detect orientation when relatively stable because it continuously sends data to the microcontroller upon overcoming a "movement" threshold. This threshold was chosen to be a change greater than or equal to .07V. Upon testing our accelerometer, we obtained the data reproduced in Tables 1, 2, and 3:

Table 1. Accelerometer Z axis data

	Tilting X Axis (Volts)	Tilting Y Axis (volts)
-90°	1.64	1.62
-45°	1.83	1.84
0°	1.95	1.95
45°	1.87	1.82
90°	1.64	1.65

Table 2. Accelerometer Y axis data

	Tilting X Axis (Volts)	Tilting Z Axis (Volts)
-90°	1.29	1.28
-45°	1.32	1.34
0°	1.60	1.60
45°	1.86	1.84
90°	1.93	1.93

Table 3. Accelerometer X axis data

	Tilting Y Axis (Volts)	Tilting Z Axis (Volts)
-90°	1.92	1.91
-45°	1.83	1.80
0°	1.60	1.61
45°	1.36	1.38
90°	1.29	1.28

2.2.2 Flex Sensors / Voltage Divider Circuits

The specific circuit design for the voltage divider circuits affected by the flex sensors is shown below in Fig. 3:

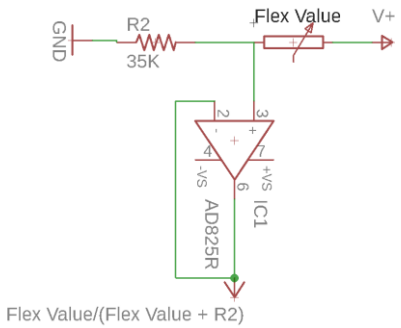


Figure 3. Flex Sensor Voltage Divider Circuit

The value of the fixed resistance of the voltage divider circuit in Fig. 2 was chosen with careful consideration. In order to maximize the output voltage, either the input voltage has to be increased or the fixed resistance, denoted as R_2 in Fig. 3 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°. This leads to the following equation:

$$V_{out} + B = V_{in}\left(\frac{AR_f}{AR_f + R_2}\right) \tag{2}$$

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 equation, a conclusion on the sensitivity of the circuit can be achieved, which can be modelled as

$$B = \frac{V_{in}R_2R_f(A-1)}{R_2^2 + R_2R_f(A+1) + AR_f^2} \tag{3}$$

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

$$\frac{dB}{dR_2} = \frac{V_{in}R_f(A-1)\left(\left(R_2^2 + R_2R_f(A+1) + AR_f^2\right) - (2R_2^2R_2R_f(A+1)\right)}{(R_2^2+R_2R_f(A+1) + AR_f^2)^2} \tag{4}$$

Setting the above relation equal to zero, and solving for R2, we conclude that

$$R_{2,opt} = \sqrt{A}R_f \tag{5}$$

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.

Table 4: Flexed and Unflexed Resistance Measurements of Sensor 1

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 5: Flexed and Unflexed Resistance Measurements of Sensor 2

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

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 3.3V 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 4: Graph of Fixed Resistance vs. Sensitivity over range of 0-100kΩ

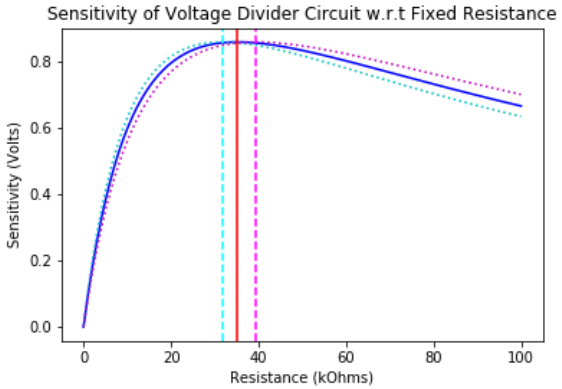
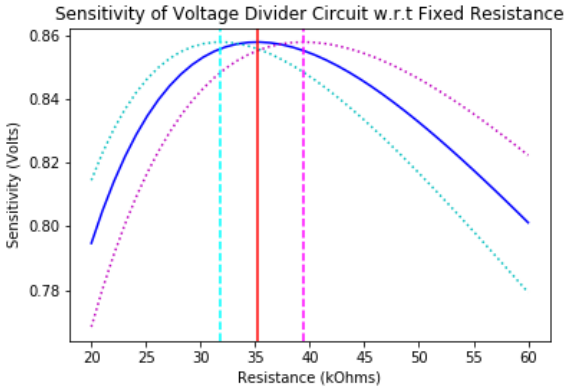


Figure 5: Graph of Fixed Resistance vs. Sensitivity over range of 20-60kΩ



Both of the above graphs show the same relationships, but have different ranges in the fixed resistance axis in order to improve 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=35.19kΩ 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.

2.2.3 Microcontroller

The specific circuit for the microcontroller is shown below in Fig. 4. The microcontroller simply takes the analog inputs of the sensors, and uses the downloaded program to convert this to MIDI code as described later on in Section 2.2.14 “MIDI Encoding”. It also enables the Bluetooth baud rate to be set at 115200 bps.

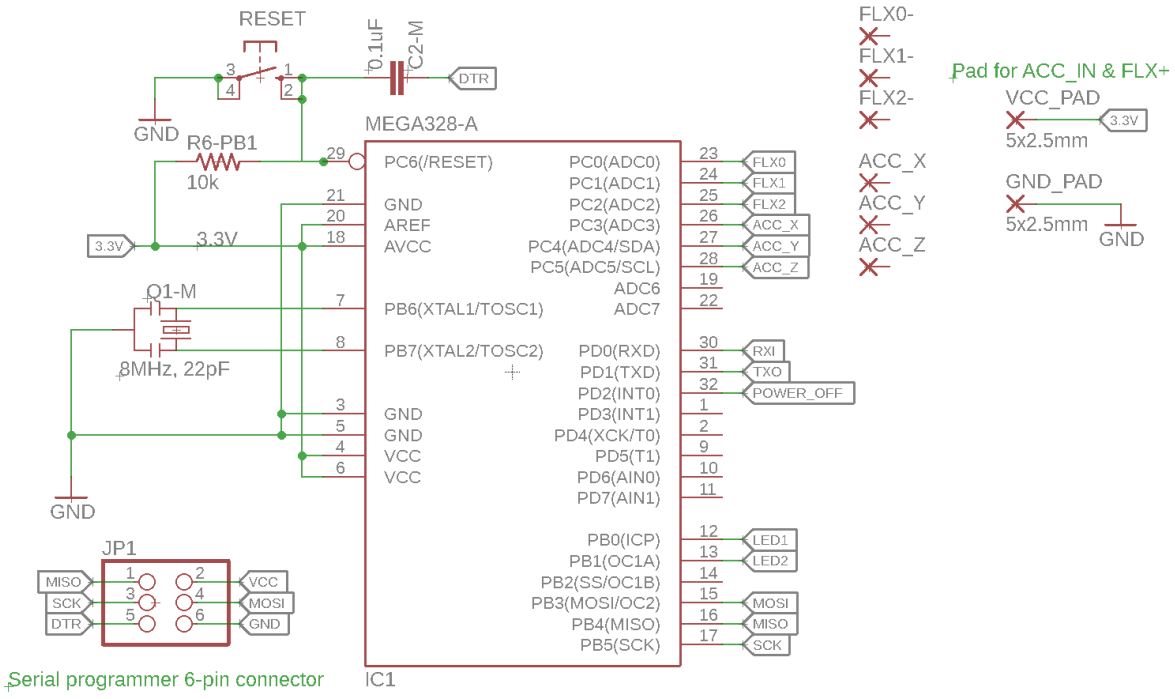


Figure 6. Microcontroller Circuit

2.2.4 Bluetooth IC with Integrated Antenna

The Bluetooth functionality is integrated into the TXD and RXD pins above in Fig. 4 and below in Fig. 5:

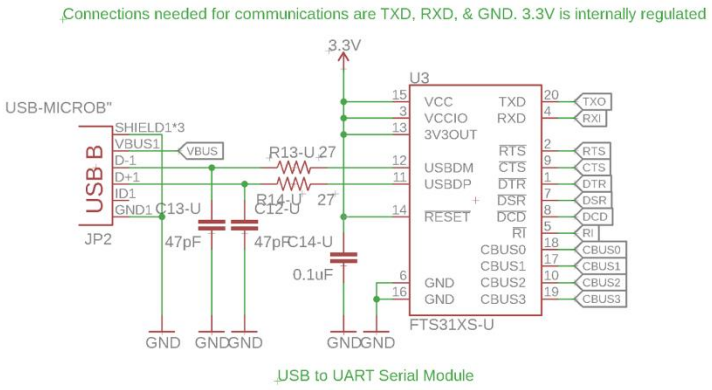


Figure 7. Communication Circuit

The TXD pin is used for transmission of data and the RXD pin is used to receive the data. The Bluetooth transmitter is connected to the TXD and RXD pins on the microcontroller as shown in Fig. 6 and the Bluetooth receiver is connected to the TXD and RXD pins in Fig. 7. Upon powering the Bluetooth transmitter, we were able to pair it with a laptop's built-in Bluetooth receiver. We then changed the name of the Bluetooth transmitter to appear as "MIDI Glove" and set the default baud rate to 115200 bps.

2.2.5 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. The specific circuit for the Power Button and Latching On/Off Function is shown below in Fig. 8:

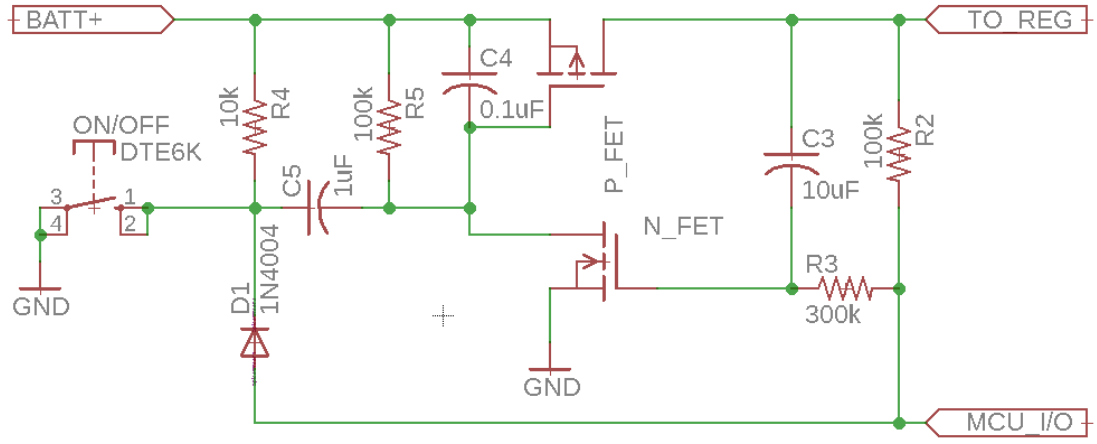


Figure 8. Power Button and Latching On/Off Circuit

2.2.6 Power LED

The power LED will inform the user on whether the system is on or off. The input of this LED is from the output of the voltage regulator. The specific circuit for the Power LED is shown below in Fig. 9:

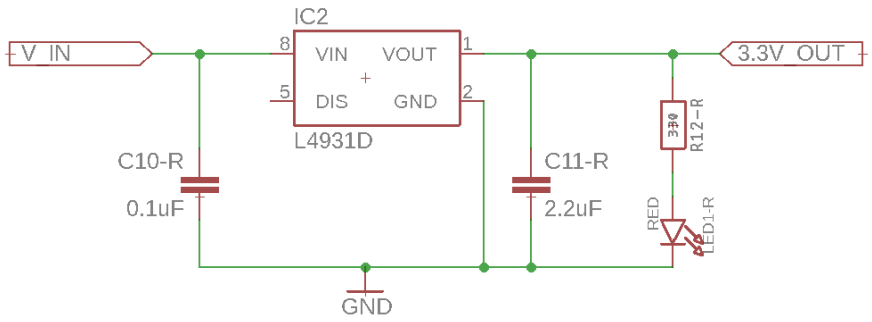


Figure 9. Power LED Circuit

2.2.7 USB Li-ion Charger

This charger must be able to charge the Li-ion battery via a 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 3.7V battery charger. The specific circuit for the USB Li-Ion charger is shown below in Fig. 10:

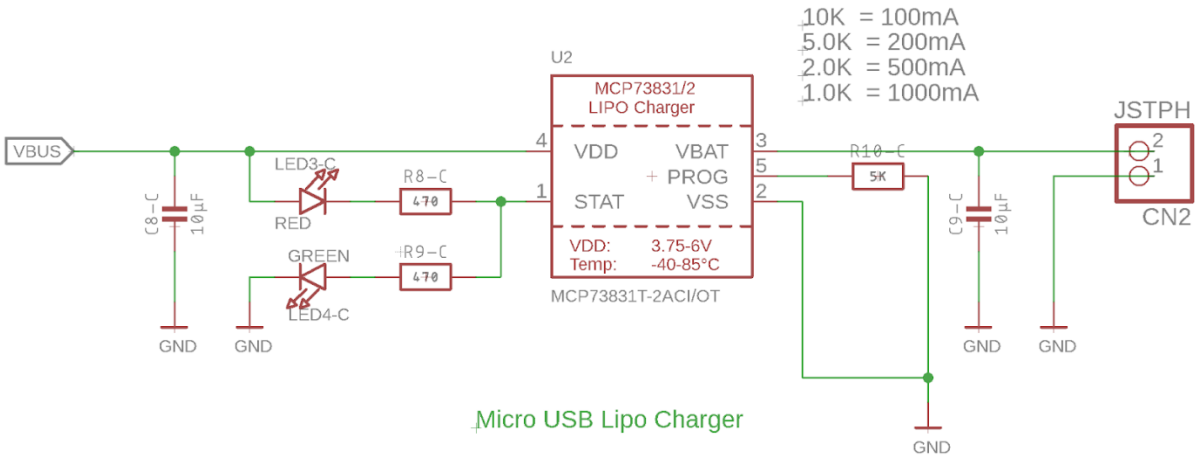


Figure 10. USB Li-Ion Charger Circuit

There are 2 additional LEDs on the charger: 1 red LED and 1 green LED. The red LED turns on when the battery is charging and the green LED turns on when the battery has been charged.

2.2.8 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. In order to test the battery, the battery discharge circuit is shown below in Fig. 11:

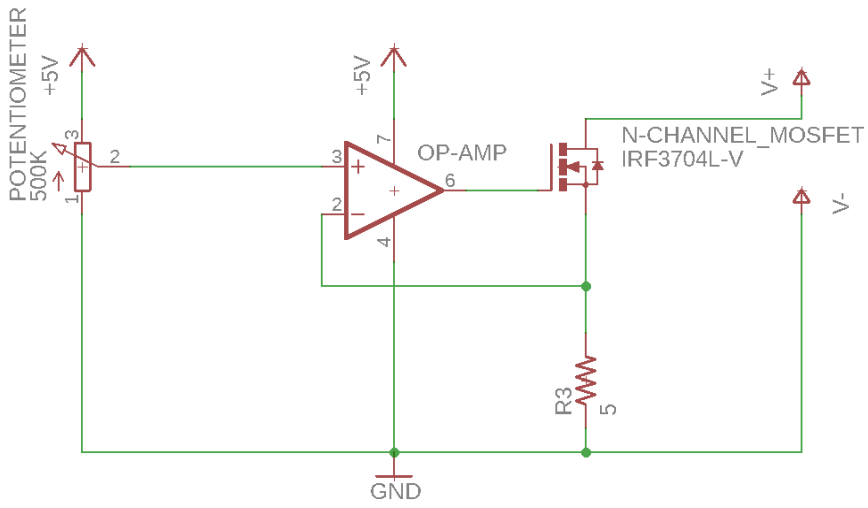


Figure 11. Battery Discharge Circuit

2.2.9 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 (3.7V) and regulate the chosen battery output voltage +/- 5% from 3.3V. The specific circuit for the voltage regulator is shown above in Fig. 9 and is labeled as “IC2”.

2.2.10 Bluetooth Receiver

We assume that the user has a computer/laptop with Bluetooth capabilities. Upon powering the glove, the Bluetooth transmitter inside the glove automatically starts looking for devices to be paired with. The user must go into the Bluetooth settings on their computer and pair it with the glove, which will show up as “MIDI Glove” upon searching for compatible devices. The Bluetooth receiver’s default baud rate is 115200 bps. Upon integrating the Bluetooth into the glove, we set the Bluetooth Transmitter baud rate to 115200 bps, so that there are no settings the user must manipulate upon connecting the glove.

2.2.11 Hairless MIDI

This program is used to convert the serial information from the glove that is sent over Bluetooth into a MIDI-recognizable format. This is free, downloadable software from:

http://projectgus.github.io/hairless-midiserial/ [12]

Upon installing and launching the software, the user must select the input port as the Bluetooth port on their computer, and the output port as “LOOP MIDI”. By going to the preferences tab, the user should check that the baud rate is set to “115200 bps”. Then, the information sent over the Bluetooth port will show up in the window as MIDI encoding and the data transmission will be successful.

2.2.12 LOOP MIDI

This piece of software allows us to create a virtual MIDI port that allows our device to be recognized as a sound peripheral by the computer, and is able to communicate with the DAW. Once Hairless MIDI converts the serial code received through Bluetooth from the glove into MIDI code, it redirects the messages to this virtual sound port. This port is then selected from within the DAW as an input instrument from which it is to receive MIDI messages.

2.2.13 DAW (FL Studio)

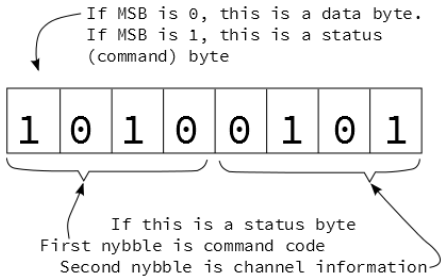
At this stage, the signals received from the glove will be interpreted as modifiers of sound parameters of a MIDI signal. This will be done in a Digital Audio Workstation (DAW) software running on a commercial computer equipped with the required input ports. The DAW we used for purposes of this project was FL Studio. We found this software to contain the flexibility in creating and editing sounds that we looking for, which made it simple to control what sound parameters were modified by the different gestures from the glove.

2.2.14 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 a most significant bit (MSB) of 1, or a "Data Byte", which has an MSB of 0. An example of a Status Byte is shown below in Fig. 12 [11]:

Fig. 12 MIDI Code Structure



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 "Control Change". The tilt of the wrist will change the "Note On" message while the flex of a finger will change the "Control Change".

The “Note On” message works as shown below:

Fig. 13 MIDI “Note On”

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

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 “Control Change” message is shown below:

Fig. 14 MIDI “Control Change”

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

* A Nybble is equivalent to 4 bits

In “Data Byte 1” of the “Control Change” message, it can be seen that there is a value for “Control Number”. The 0 to 127 value encodes the sound (patch) of the note. For example, a Control Number of 0 could be a manipulating the pan, while a Control Number of 32 could be manipulating the vibrato. The specific Control Numbers 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 Control Number will change.

In conclusion for added clarity, an example of a “Note On” and “Control 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.

Control Change: 11001111 00010001

Hex: B F 1 1

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

2.3 Physical Design

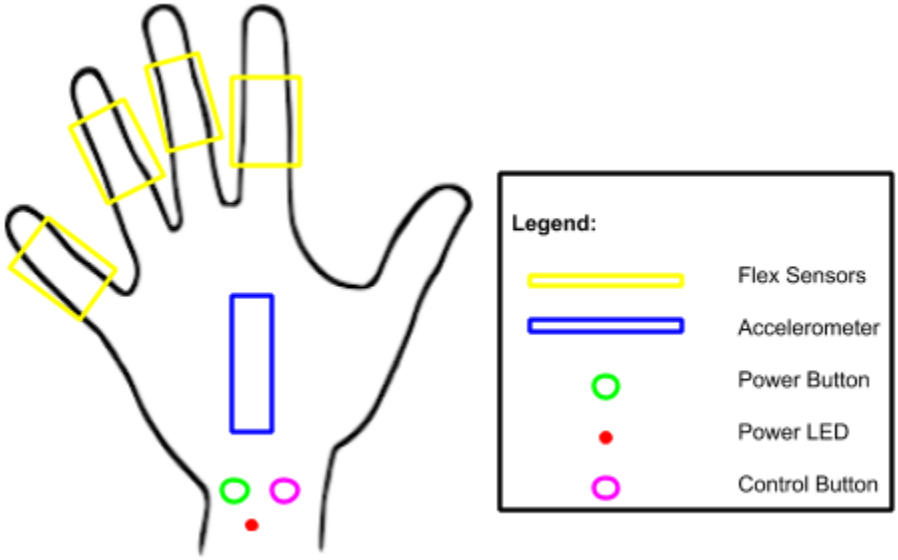


Fig. 15 Physical Design

The diagram above in Fig. 15 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.

3. Costs

3.1 Parts

Table 6: Production Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
Flex Sensors 2.2'' (x3) (SEN-10264)	Sparkfun	23.85	21.84	21.84
Triple Axis Accelerometer (ADXL35)	Analog Devices	14.95	11.96	14.95
Class 1 Bluetooth Module (RN-41)	Roving Networks	23.00	21.03	23.00
USB Li-Ion/LiPoly Charger (MCP73831)	Adafruit	6.95	5.56	6.95
Lithium Ion Polymer Battery (3.7, 500mAh) (503035)	Adafruit	7.95	7.16	7.16
PCBs	PCBWay	5.00	5.00	5.00
Total ($)		**81.70**	**72.55**	**73.9**

3.2 Labor

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:

$$3 * \frac{\$33}{hr.} * \frac{7\ hr.}{week} * 16\ weeks = \$13{,}440 \tag{6}$$

Thus, our total cost, including production costs and fixed development costs, is $13,513.90 dollars.

4. Conclusion

4.1 Accomplishments

At the demonstration stage of this project, the glove is working as proposed. The glove motion sensors are set up accurately. The user is able to tilt their wrist and the velocity of the note being played will change accordingly. Upon the bend of a finger, the user is able to utilize a control change that they pre-set. For our demonstration, our control changes were pre-set to change the “vibrato”, “pan”, and “pitch bend” of the note being played.

The power supply has been integrated successfully, as the glove is able to stay on for over 5 hours without needing to be re-charged. Upon the battery running out of power, all that is needed is to plug the glove into a USB port via the USB charger integrated on the board in the wrist of the glove. The measured battery and voltage regulator output remain in the desired range of 3.3V+/- 5% for over 5 hours.

The control and communications unit is functional. The microcontroller sends receives values from the flex sensors and accurately sends out MIDI signals via the Bluetooth transmitter on the board. The Bluetooth light remains stable upon successful transmission and reception of the MIDI code between the glove and the computer. The power LED lights up and turns red upon powering up the glove. However, the power button integrated into the glove is not working. We believe this is possibly due to an unsuccessful PCB board design or a faulty soldering job. Because of this, in order for the user to turn on the glove, they must power up the glove by physically connecting and disconnecting the battery from the glove.

Finally, the MIDI Control Interface is working as proposed. By using a Bluetooth receiver that is integrated into most laptops/computers and setting up the Bluetooth transmitter located on the PCB board to pair and communicate, MIDI code is successfully transmitted to the computer. Upon successfully setting up Hairless MIDI and LOOP MIDI, FL Studio is able to read the data as MIDI encoding and set up the appropriate outputs.

4.2 Uncertainties

The only uncertainty that occurred was due to the power LED. As mentioned above, there are two hypotheses as to what went wrong: the soldering job or the PCB board design. The problem could be determined by either de-soldering and re-soldering the button or re-designing the PCB board. Once the problem is discovered, the appropriate solution could be tested: either ordering a new button or improving the PCB board design to improve on this mistake.

4.3 Ethical considerations

Within our proposed design, there are several safety concerns. One of the necessary implementations of the project was to have the glove be rechargeable when being used wirelessly. This meant our design included the use of rechargeable batteries. We used Li-Ion rechargeable batteries for this glove, which could create certain concerns as they may explode or burn if misused or mishandled. It was important to ensure that there was no internal short circuit, as this could lead to increased heat within the battery and an exothermic reaction, increasing the risk of additional combustion [5]. To ensure this was not the case, we tested the batteries thoroughly before using in the glove. Another concern with these batteries was the risk of possible explosion upon overcharging. In order to avoid this scenario, we tested 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 were 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 proposed to use a moisture resistant fabric as well as use rubber coatings on any wires or electrical contact made in order to prevent the user from experiencing any shock. Along these same lines, we recommend 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 IEEE 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 was 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].

4.4 Future work

In the future, we would like to add several hardware and software functionalities. On the hardware end, we would like to integrate a successfully working power button that could easily turn the glove on and off. Pressure sensors could be added in the tips of each of the fingers with a flex sensor. This could facilitate additional recognizable gestures, and thus additional MIDI encodings. We would also like to add in LEDs in each of the fingers with the flex sensors that could change their brightness based upon the level of bend of the finger. On the software end, we would like to enable a “tapping” feature on the accelerometer that could distinguish between a single “tap” and a “double-tap” by using the z-axis output of the accelerometer, which could also lead to additional MIDI encodings. Finally, we would like to change the sensor mapping algorithm from a simple one fit linear algorithm to a specifically designed algorithm for each individual sensor.

References

[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].

[12] A. Gratton, "The Hairless MIDI to Serial Bridge". [Online]. Available: http://projectgus.github.io/hairless-midiserial/ [Accessed 2-Apr-2018].

Appendix A Requirement and Verification Table

Table 7. Three Axis Accelerometer

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.

Table 8. Flex Sensors/Voltage Dividers

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.

Table 9. Microcontroller

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

Table 10. Bluetooth Transceiver

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.

Table 11. Power Button, Latching On/Off function

Requirement	Verification
Button must be easy to press, 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.

Table12. Power LED

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.

Table13. USB Li-Ion 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.

Table14. Li-Ion Battery

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.

Table15. Voltage Regulator

Requirement	Verification
The voltage regulator must provide 3.3V 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.

Table16. Bluetooth Receiver

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.