GESTURE-BASED/TACTILE MUSIC COMPOSITION

By

Adam Marcott

Tyler O'Neill

Final Report for ECE 445, Senior Design, Spring 2017

TA: John Capozzo

3 May 2017

Project No. 75

Abstract

We present a device which enables musicians to create and record music on the go, without the need to use proprietary software or music controller hardware. The device is a pair of gloves which detect users' taps on any surface, and it also recognizes their hand gestures to manipulate the sound produced. The sound is produced indirectly by the device by generating a MIDI signal. The signal is then wirelessly transmitted by a Bluetooth LE transmitter to any compatible MIDI synthesizer.

Contents

1. Introduction 1
   1.1 Objective 1
   1.2 Background 1
   1.3 High-Level Requirements 1
2 Design 3
   2.1 Power Subsystem 4
      2.1.1 LiPo Charger 4
      2.1.2 Li-ion Battery 4
      2.1.3 Voltage Regulator 4
   2.2 Sensor Unit 4
      2.2.1 Inertial Measurement Unit (IMU) 4
      2.2.2 Mechanical Switches 4
   2.3 Processing/Transmission 5
      2.3.1 Microcontroller 5
      2.3.2 Bluetooth LE Transmitter 5
      2.3.3 XBee 5
   2.4 Physical Design 5
   2.5 MIDI Standard Overview 6
   2.6 Software Overview 6
   2.7 Kalman Filter 7
3. Design Verification 9
   3.1 Power Subsystem 9
      3.1.1 LiPo Charger 9
      3.1.2 Li-ion Battery 9
      3.1.3 Voltage Regulator 9
   3.2 Sensor Unit 9
      3.2.1 IMU 9
      3.2.2 Mechanical Switches 9
   3.3 Processing/Transmission 9

3.3.1 Microcontroller 9
3.3.2 Bluetooth Low Energy Transmission Unit 10
We placed the two gloves at 0 m apart and at 5.5 m apart and observed no change in signal transmission. The simulated signal was a random sequence of taps. 10
4. Costs 11
4.1 Parts 11
4.2 Labor 11
4.3 Grand Total 11
5. Conclusion 12
5.1 Accomplishments 12
5.2 Uncertainties 12
5.3 Safety and Ethics 12
5.4 Future work 13
5.4.1 Hardware Development 13
5.4.2 Software Development 13
References 14
Appendix A Requirement and Verification Table 15
Appendix B Gesture Table 17

1. Introduction

1.1 Objective

Musicians face a unique problem as artists. When inspiration strikes, expensive recording equipment is not always readily available, especially on-the-go. It is possible, for example, to use a mobile drum sequencer application to "jot down" an idea for a drum pattern, but users are confined to a small space and an often-unintuitive interface. Instead of having a dedicated application for a step sequencer, there needs to be a cost-effective way to capture rhythmic taps and smooth gestures that can control any MIDI-based device wirelessly.

The aim of this project is to build a scalable, wireless glove that can capture motion and tactile data from a musician and generate Musical Instrument Digital Interface (MIDI) signals and sounds wirelessly on a mobile platform more intuitively than a traditional graphical user interface (GUI) and without the need for more expensive, specific hardware.

1.2 Background

Many current solutions exist, such as Air Beats[1], a glove that allows musicians to compose music digitally without interacting with a screen. There are simpler alternatives, such as the Xkey[2], but a MIDI keyboard requires a stable surface and ample space for movement in order to be an effective composition tool. Our device needs neither, as the gestures are compact and only need as much space as your hands occupy. There is yet to be a commercially successful gesture-based product as there is a high reliance on either expensive pre-existing technology with questionable reliability (digital paper technique), or a lack of throughput capability due to inexpressive language during pattern recognition. Our solution aims to solve both problems with current solutions by using a minimal set of viable hardware and an expressive language that can be used to quickly translate motion into synthesizable sound.

This expressive language uses simple movements of the hand (right and left, up and down, forward and backward) and rotations (rotate up and down, rotate left and right) in combination with finger placement to enable over 30 different control functions to be available at once. Both hands are able to generate MIDI notes and controls simultaneously to enable versatile composition. For example, when a user holds their left thumb to their index finger and moves their hand up or down, it will control volume, whereas if they were to make the same motion with their left thumb held to their ring finger it controls a modulation wheel. The most interesting part is that these MIDI control signals can be easily remapped by the user to control any MIDI-compatible device. Figure 1 shows the block diagram for the right hand, and Figure 2 shows the block diagram for the left hand.

1.3 High-Level Requirements

- Glove must be able to stay powered with a Li-ion battery for more than 2 hours.
- Glove must be able to process sensor data (switch presses/taps and gestures from a defined library) and map it to MIDI data.
- Glove must be able to wirelessly transmit MIDI data over Bluetooth LE.

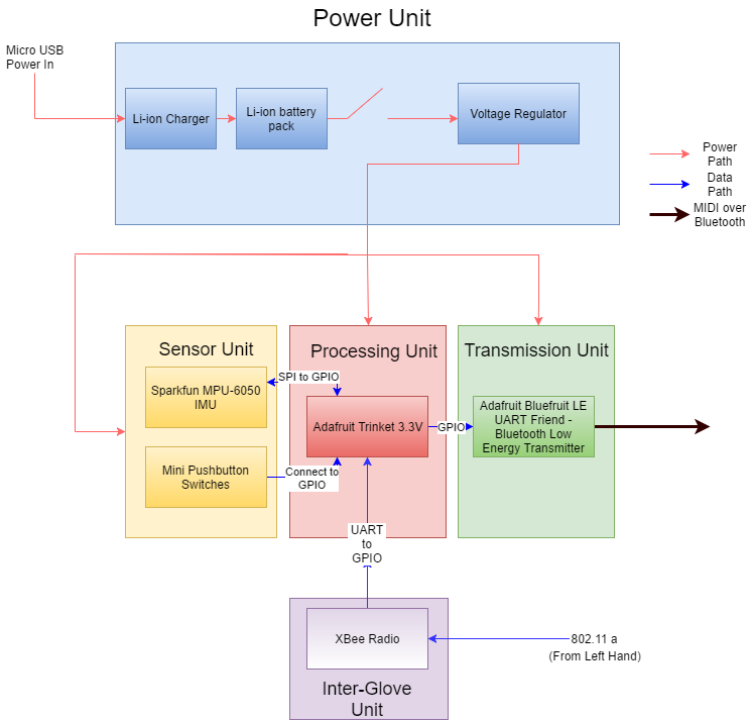

Figure 1 Right Glove Block Diagram

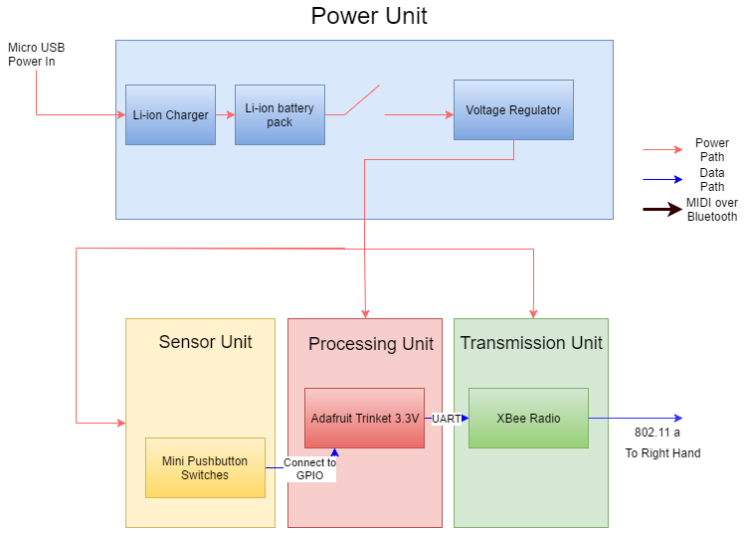

Figure 2 Left Glove Block Diagram

2 Design

Our design satisfies our project requirements for the following reasons: First, our Bluetooth transmitter is able to transmit MIDI data by leveraging the MIDI over Bluetooth protocol. We implement this protocol on the Adafruit Trinket microcontroller. Second, our sensors are able to recognize gestures with the IMUs providing orientation and acceleration data. The sensors also recognize taps, thanks to the mechanical switches located on the tip of the finger and the fingerprint of each finger. Third, our power subsystem includes a voltage regulator to provide a consistent 3.3V well over our 2-hour requirement (we use a Li-ion battery with 110mAh capacity). Figure 3 shows the circuit schematic for the right hand, and Figure 4 shows the circuit schematic for the left hand.

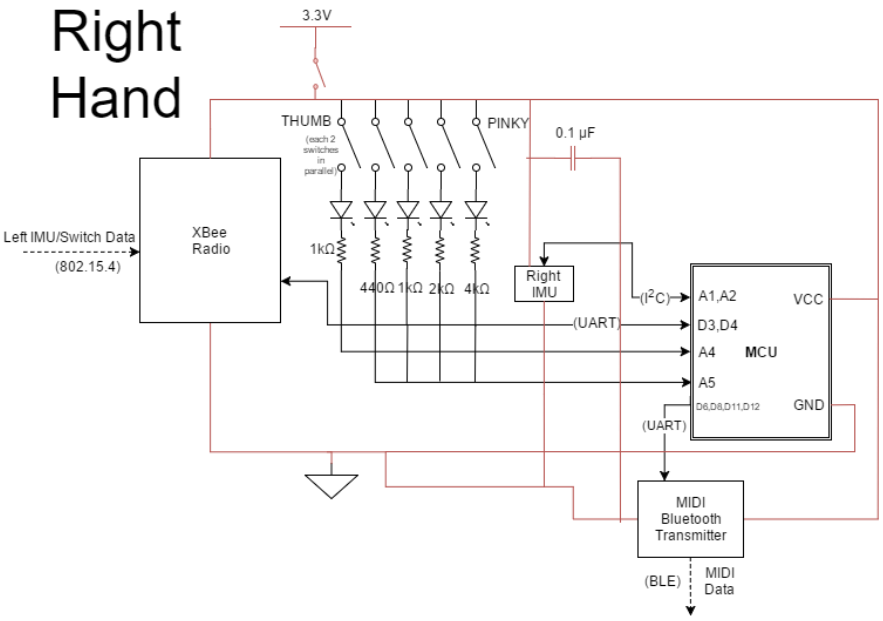

Figure 3 Right Hand Circuit Schematic

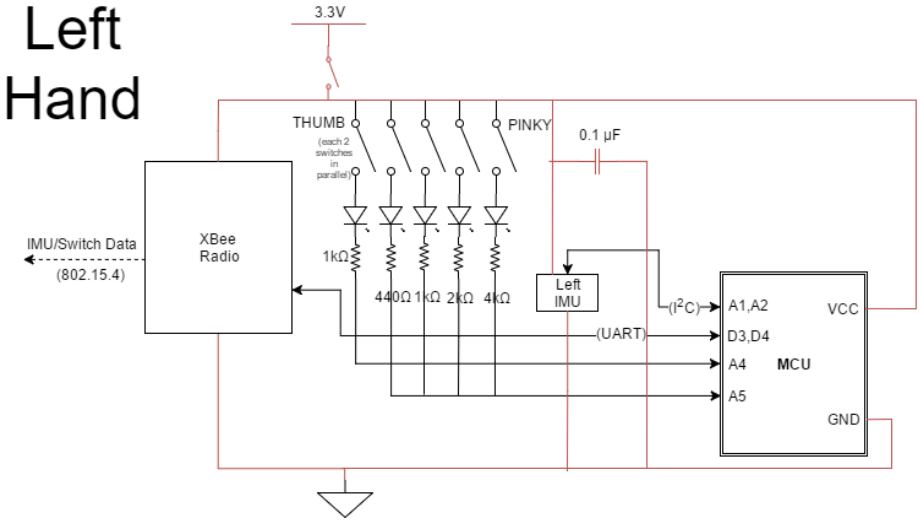

Figure 4 Left Hand Circuit Schematic

2.1 Power Subsystem

A power supply is required to safely power all components of the system.

2.1.1 LiPo Charger

The Lithium Polymer charger is an IC that allows us to safely charge the Li-ion battery pack. It will act in redundancy to the battery's built-in overvoltage and overcurrent protection to keep the battery in nominal ranges while going through a charge/discharge cycle.

2.1.2 Li-ion Battery

The battery can power all components of the system through the voltage regulator mechanism. It has a 110 mAh capacity, which is more than sufficient for this application, since the average current draw is about 10 mA.

2.1.3 Voltage Regulator

Transforms the battery's output voltage of 3.7 V to the voltage needed by the hardware components of the glove, which is 3.3 V. Also acts as a protection mechanism for the mechanical switches, which do not have built-in voltage regulation, and as a redundancy for all the other components which do have built-in protection circuits.

2.2 Sensor Unit

Sensors are required to enable the user to perform gestures and taps, which are recognized by the glove and converted into MIDI signals.

2.2.1 Inertial Measurement Unit (IMU)

An IMU is needed to detect and quantify hand movement and orientation for gesture recognition. We use one unit, placed near the middle knuckle on the right hand for maximum range of motion. The microcontroller's control function steps through and probes linear acceleration on all three axes and angular velocity on the x- and y-axis to make gesture decisions. The IMU uses a 400 kHz I^2C protocol and input to two digital GPIO pins on the microcontroller. This sensor data is then sent through a Kalman filter to reduce noise, as explained in Section 2.7.

2.2.2 Mechanical Switches

Mechanical switches are needed to detect taps on a surface. In this application, they will measure glove deformation, and we will determine a threshold of measurement that we will consider a "tap". These sensors will be placed in parallel on the fingertips and fingerprints of each finger on the glove. Staggered resistances will be used in a voltage divider to be able to enumerate any combination of pointer, middle, ring, or pinky finger in software by using only one analog input GPIO pin on the microcontroller. Resistances of 440 Ω, 1 kΩ, 2 kΩ, and 4 kΩ on the pointer, middle, ring, and pinky fingers, respectively. The thumb uses its own analog input because having five fingers in the voltage divider was accompanied with too narrow of a range to resolve certain combinations of finger presses. Table 1 shows example voltage ranges for various mechanical switch combinations.

Table 1 Sample Mechanical Switch Data

Fingers	Value (V)	Range (V)
Index	2.90 V	2.74 V - 3.06 V
Middle	2.67 V	2.61 V - 2.74 V
Ring	2.46 V	2.42 V - 2.50 V
Pinky	2.26 V	2.24 V - 2.27 V
Index, Middle	2.56 V	2.50 V - 2.61 V
Ring, Pinky	2.10 V	2.09 V - 2.11 V
Index, Middle, Ring, Pinky	2.03 V	0 V - 2.03 V

2.3 Processing/Transmission

2.3.1 Microcontroller

Sensor data from the IMUs and switches are manipulated in software to produce a MIDI data signal, which is sent through the Bluetooth LE unit at a rate of 9600 Baud. The Microcontroller used is an Adafruit Trinket 12 MHz unit.

2.3.2 Bluetooth LE Transmitter

We need a Bluetooth LE unit because it will enable us to use our glove with any MIDI controller supporting MIDI over Bluetooth. The unit will receive a MIDI signal from the microcontroller and transmit it wirelessly to any (arbitrary/outside project scope) receiver which we assume will be a MIDI controller that will enable playback.

The Bluetooth unit we have chosen has 32 kB RAM, 256k Flash Memory and a 9600 Baud transmission rate.

2.3.3 XBee

We switched from Bluetooth to XBee for our inter-glove communication in our final implementation. This unit uses the 802.15.4 protocol to allow the gloves to work up to 200 feet away from each other indoors (well over the absolute minimum of the average wingspan). The slave (left) glove sends switch data to the master (right) hand. The XBee connects to the Trinket using UART protocol.

2.4 Physical Design

Our physical design diagram shows the key placements of the devices we use in our project. Notably, we have consolidated most of the crucial elements -- the microcontroller, transmitter, IMU, and power unit

-- together on the back of the glove. This placement vastly simplifies connecting components and keeps the design concise. Figure 5 shows the more complicated physical design of the right-hand glove.

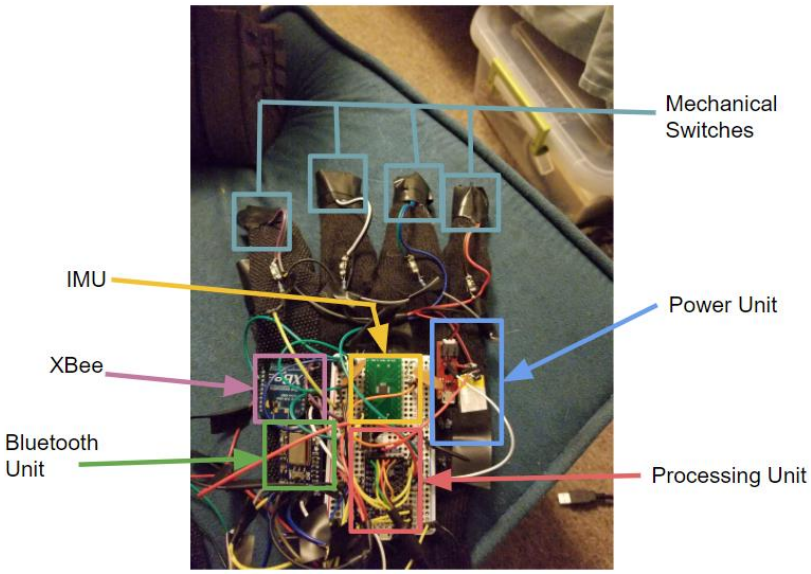

Figure 5 Right Glove Physical Layout

2.5 MIDI Standard Overview

MIDI has been a long-established standard in the world of electronically generated music, owing to its simplicity and versatility. MIDI works by sending messages about musical notes, pitch, velocity, and a wide variety of other parameters that are encoded as control signals. These messages are encoded as packets in a simple format. Notes are sent as three bytes -- the first byte determines which MIDI channel to use (we use exclusively Ch. 1), the second byte determines which note was generated, and the third byte encodes velocity (how hard the note was pressed). The other type of messages are called control bytes and they follow a similar format. Again the first byte determines channel, the second byte, called the control byte, determines which parameter is being modified, and the third byte specifies the value for the parameter[12].

2.6 Software Overview

In the design of the software, each hardware module is given an associated software module, and an API for each module was created as necessary. The modularity of the software is such that a simple compiler flag is all that needs to be changed to build the left or right hand package.

In the main loop, first all sensor data is taken in, and for the left hand, sensor data is transmitted unchanged to the right hand for processing. This is all the left hand does in its loop. For the right hand, all sensor data from its own sensors AND those polled from the left hand is sent through subroutines that generate MIDI packets. MIDI notes are generated by analyzing what switches are pressed based on ranges of analog values. And when switches are released a MIDI NoteOff packet is sent. MIDI control packets are generated by gesture processing algorithm. This is the Kalman filter with appropriate scaling and keeping track of change in angle each time the main loop runs. Finally, all packets generated are sent through BT unit.

Pressing the thumb and one more finger causes the device to enter Gesture mode. In this mode, IMU data (preprocessed by Kalman filter), is used to estimate rotational deltas. These deltas are scaled to [0-127] to affect a database of MIDI control numbers stored in memory. When an element of the database is changed by a delta, a MIDI control signal is generated so that the MIDI endpoint and the microprocessor have synchronized data.

2.7 Kalman Filter

We make a prediction of the state at time *k* using all previous states up to time *k-1*, called the *a priori* estimate: $\hat{x}_{k|k-1}$ using a state transition model *F* and a control model *B* which gives us an angle from the angular rate $\dot{\theta}_k$, as shown in Equation (2.1).

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + B\dot{\theta}_k \tag{2.1}$$

We calculate *P*, the error covariance matrix at time *k*, using all previous matrices up to time *k-1* by the following equation. Here Q_k is the process noise covariance matrix, as shown in Equation (2.2).

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q_k \tag{2.2}$$

The following measures the error between the observation and the *a priori* estimate and is called the innovation, as shown in Equation (2.3).

$$\tilde{y}_k = z_k - H\hat{x}_{k|k-1} \tag{2.3}$$

The following computes the innovation covariance. It calculates how much we should trust the current measurement. *R* is the measurement noise covariance matrix, as shown in Equation (2.4).

$$S_k = HP_{k|k-1}H^T + R \tag{2.4}$$

We then compute the Kalman gain *K* at time *k*. This measures how much we should trust the innovation at the current time step, as shown in Equation (2.5).

$$K_k = P_{k|k-1}H^T S_k^{-1} \tag{2.5}$$

We are now ready to calculate the *a posteriori* estimate for the current state based on the *a priori* estimate and our innovation, as shown in Equation (2.6).

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_k \tag{2.6}$$

And finally we decrease the error covariance matrix as our new estimate has decreased the error in observing the true state, as shown in Equation (2.7).

$$P_{k|k} = (I - K_k H)P_{k|k-1} \tag{2.7}$$

Figure 6 shows the Raw IMU data, and Figure 6 shows the output of the Kalman filter.

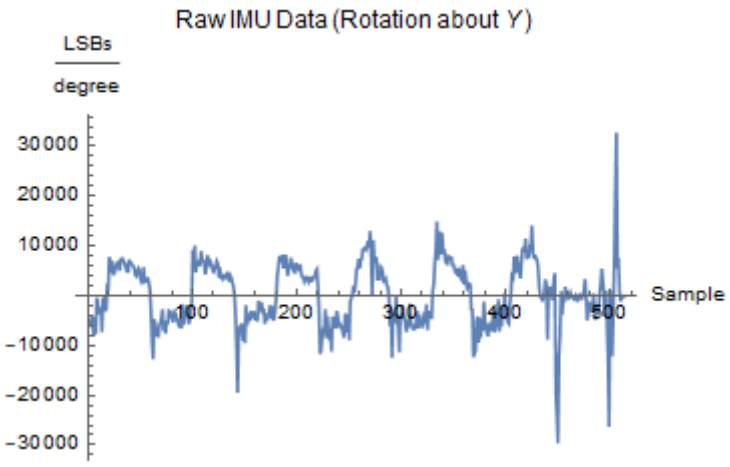

Figure 6 Raw IMU Data for Rotation about Y-axis

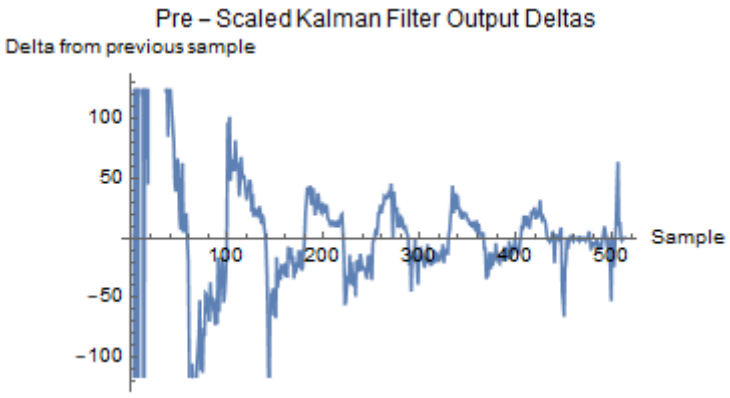

Figure 7 Kalman Filter Output

3. Design Verification

3.1 Power Subsystem

3.1.1 LiPo Charger

We verified operation of the Li-ion charger by starting with the battery fully-discharged. From there, a Micro-USB cable was plugged into the charger and monitored the cell temperature with an IR thermometer. Within one hour, well within the requisite four hours, the cell voltage had reached 3.7 V all while staying under 45° C.

3.1.2 Li-ion Battery

We simulated continuous operation of the device by forcing the glove to run in full-power mode. All components with low power mode -- the XBee Radio, IMU, and Bluetooth transmitter -- had those power saving features disabled. After two hours, the device was still powered up, as required.

3.1.3 Voltage Regulator

After a period of one minute of probing the voltage regulator output with a voltmeter, the voltage was maintained between 3.2 V and 3.4 V, as required.

3.2 Sensor Unit

3.2.1 IMU

We verified the gyroscope of the IMU by starting with the device unpowered. We also changed the scaling of the IMU to give us output in radians. Right after powering up the glove, gesture mode was enabled by pressing together the thumb and index finger. After rotating about the X axis by +12.2 rad, we observed that the MIDI control value for control number #70 (associated with index finger X rotation) was 127, as required.

We failed to verify the accelerometer of the IMU. This is because we were unable to filter out the noise of the accelerometer output using techniques we had learned. The Kalman filter was unusable as we'd need a second order approximation of linear acceleration instead of a first order approximation of angular rate. We attempted to use a simple low pass filter, but the noise environment proved to be dynamic, and so while the filter would work in one position, it would fail in another.

3.2.2 Mechanical Switches

After placing a weight of 4.5 pounds on a switch, we observed the switch's associated LED light up.

3.3 Processing/Transmission

3.3.1 Microcontroller

After powering up the device, we performed a sequence of 100 random taps. Although there were a few extraneous notes generated, and a few notes that were missed, the correct note was generated 93 times out of 100. This satisfies our 90% accuracy requirement.

3.3.2 Bluetooth Low Energy Transmission Unit

We placed the two gloves at 0 m apart and at 5.5 m apart and observed no change in signal transmission. The simulated signal was a random sequence of taps.

4. Costs

4.1 Parts

Table 2 Parts Costs

Part	Manufacturer	Retail Cost	Quantity	Actual Cost
Trinket 3.3V Microcontroller	Adafruit	$6.95	2	$13.90
XBee 802.15.4	Digikey	$17.50	2	$35.00
MPU 6000 IMU[11]	SparkFun	$29.95	1	$29.95
Mini Pushbutton Switch	SparkFun	$0.35	20	$7.00
Li-ion Battery 110 mAh[9]	SparkFun	$6.49	2	$12.98
LiPo Charger Micro-USB[8]	SparkFun	$7.95	2	$15.90
Voltage Regulator 3.3V[10]	SparkFun	$1.95	2	$3.90
Red LED Sequin	SparkFun	$0.29	10	$2.90
Total				**$121.53**

4.2 Labor

We assume that based on the average UIUC Computer Engineering graduate salary for the 2015 graduating class of $84,250[7] and for an average workload of 10 hours per week for each partner that we will be looking at $421.25/week for a total of $3791.25 for the remaining 9 work weeks.

4.3 Grand Total

Summing the values calculated in the labor and parts section, we reach a grand total of $3912.78 (not including taxes).

5. Conclusion

5.1 Accomplishments

The physical construction of the gloves was satisfactory. We could mount the components onto the gloves and power them up using the attached battery packs. The right-hand microcontroller was successful in processing switch presses and sending corresponding MIDI notes. Rotational gestures were also able to be recognized to an extent, and MIDI control signals were correctly sent.

The addition of LEDs and use of the analog switch circuit were a modest boon in the aesthetics and efficiency, respectively, of the device. We also managed to produce a software package that is configurable and extensible by means of modular development.

5.2 Uncertainties

We were disappointed by the hardware failure of the IMUs. We identified the issue as since the chip package was so tiny, soldering it onto a breakout board most likely caused physical damage.

Similarly for the XBee radios, they were mounted precariously onto the gloves, and soldering may have caused damage to the units, or it is possible some pins got shorted together.

The durability of the device, owing to the judicious use of soldering, is not great. In the future, we'd like to minify the space needed for all the components and perhaps use a custom-built PCB.

5.3 Safety and Ethics

Our design and product complies with the IEEE Code of Ethics. The following two points from the Code are especially relevant for our design:

1. "*to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment*" [3]
   We considered the safety and health risks of our design and have provided a list of precautions to take when using our hardware. We also designed our hardware to reduce any potential risks to the users.
   2. "*seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others*;" [3]
   We will properly cite the contributions from previous research and similar products.

The hardware operates at a low voltage, so there is no prominent risk of electric shock from normal use. However, the sensor unit contains fragile components that can be broken when the hardware is not used properly. For example, the mechanical switch can be damaged when too much force is applied, and the IMU can be damaged if the hardware is dropped to the ground from high places. Also, since we do not need any validation data to generate a library of calibration data, human subjects will not be needed, thus data integrity will not be an issue.

If the Lithium-ion battery is not charged properly, it can explode. During the build process, we used a thermometer to monitor the temperature of the power subsystem to make sure the cell temperature does not reach unsafe levels. To make sure the battery does not get overcharged, all of the charging circuitry was tested thoroughly so the maximum threshold was not exceeded.

We have signed the following battery safety sheet:
https://courses.engr.illinois.edu/ece445/documents/GeneralBatterySafety.pdf

List of precautions to take when using the wearable hardware:

- Avoid using the hardware with wet hands or near water.
- Avoid dropping or squeezing the hardware.
- Use the hardware within the safe temperature range: 0°C to 45° C (32° F to 113° F).
- Be cautious for the potential electrostatic discharge.
- Before using, check that there is enough space around you. Also, make sure the hardware cannot slip out of your hand. If the hardware hits a person or object, this may cause accidental injury or damage.
- Be cautious of the hardware overheating. Remove the hardware immediately if it feels too hot.

5.4 Future work

5.4.1 Hardware Development

An additional IMU on the left hand would allow for a more comprehensive library of gestures to be mapped to MIDI control functions. It would be useful to consider alternatives to the mechanical switches used to generate MIDI notes. For example, a piezoelectric sensor could not only generate notes, but it could also add a note velocity to the signal. Another consideration is to try using light sensors on the fingertips to control note on/off signals. Finally, ultrasonic range sensors could be used to add a relative glove displacement component to the gestures.

5.4.2 Software Development

The biggest goal for future software development is building a MIDI synthesizer App for Android 6.0 that support MIDI over BLE. We were surprised in the building of our project that no such App exists as of present. Creating such an App would be a huge plus if we can show that our hardware is one of very few that can make use of the nascent standard.

Other more modest work that needs to be done on the software side is improving switch detection to improve note accuracy. As it stands right now, sometimes extra notes are generated, especially when a user tries to enter gesture mode. We foresee adding some kind of delay or low pass filter to improve accuracy.

Another possible development is abstracting away the use of MIDI over BLE altogether. Since the microcontroller already knows how to generate a MIDI signal, it can conceivable work with any MIDI synthesizer using more traditional transport systems such as the MIDI jack or MIDI over USB.

References

[1] Shah, Utsav, Abhishek Kannekanti, Pinaj Basutkar, Yebai Zhao, and Paula Garcia. "Air Beats: A System for Eyes Free Music Composition." Rochester Institute of Technology, 15 June 2016. Web. 13 Oct. 2016.

[2] Xkey Technical Specifications, 2016. Web. 27 Nov. 2016.

[3] "Code of Ethics IEEE", Institute of Electrical and Electronics Engineers, Inc. (2006). http://www.ieee.org/. Retrieved Web. 20 Oc. 2016 from the website temoa : Open Educational Resources (OER) Portal at http://www.temoa.info/node/23284.

[4] "What is a strain gage?," 2003. [Online]. Available: http://www.omega.com/prodinfo/straingages.html. Accessed: Feb. 9, 2017.

[5] I. Buchmann, "Battery safeguards; protection circuits – battery university," 2016. [Online]. Available: http://batteryuniversity.com/learn/article/safety_circuits_for_modern_batteries. Accessed: Feb. 20, 2017.

[6] "Salary averages: ECE ILLINOIS,". [Online]. Available: https://www.ece.illinois.edu/admissions/why-ece/salary-averages.asp. Accessed: Feb. 24, 2017.

[7] "Bluetooth BC118 Datasheet", *SparkFun*, 2017. [Online]. Available: https://cdn.sparkfun.com/datasheets/Wireless/Bluetooth/BC118_DS.pdf. [Accessed: 27- Mar- 2017].

[8] 2017. [Online]. "Lipo Charger", Available: http://cdn.sparkfun.com/datasheets/Components/General%20IC/33244_SPCN.pdf. [Accessed: 28- Mar- 2017].

[9] 2017. [Online]. "Lithium Ion Battery", Available: http://cdn.sparkfun.com/datasheets/Prototyping/spe-00-DTP401525-110mah-en-1.0ver.pdf [Accessed: 28- Mar- 2017].

[10] 2017. [Online]. "3.3V Voltage Regulator", Available: https://www.sparkfun.com/datasheets/Components/LD1117V33.pdf [Accessed: 28- Mar- 2017].

[11] 2017. [Online]. "MPU-6000 IMU", Available: http://cdn.sparkfun.com/datasheets/Components/General%20IC/PS-MPU-6000A.pdf [Accessed: 28- Mar- 2017].

[12] "Bluetooth LE MIDI Specification", *Midi.org*, 2017. [Online]. Available: https://www.midi.org/specifications/item/bluetooth-le-midi. [Accessed: 28- Mar- 2017].

Appendix A Requirement and Verification Table

Table 3 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
The charger will allow the battery pack to be charged to 3.6 - 3.8V within 4 hours while staying below the maximum charging temperature of 45°C[4].	From the fully discharged state, we will monitor battery cell voltage over the time period to ensure that the required voltage is reached. We will also use an IR thermometer to ensure the battery does not exceed the maximum temperature during this charging cycle. (10 points)	Y
Must power all components stably for at least 2 hours from a full charge (which we will define as within 10mAh of its 110mAh listed capacity).	From the fully charged state we will measure over a period of 10 seconds the current draw from the battery and see if the average current draw from that period corresponds to an average power draw that would exceed the listed capacity → FAIL, or be within it → SUCCESS (10 points)	Y
Must provide 3.2-3.4V from the 3.6-3.8V source over a measurement period of 60 seconds while glove is on.	Using a voltmeter we will measure the output voltage for 1 minute to ensure it is within the nominal range for that period. (10 points)	Y
Gyroscopic functionality can distinguish an angle on all three axes to an accuracy within +/- 0.1 rad with 5% error.	1. Start out with any arbitrary MIDI parameter set at 0. (as is the default for device startup) 2. Make an adjustment of 0.1 * 122 = 12.2 rad in the positive direction 3. Verify that the MIDI parameter is then set at a value of 122 +/- 6 (corresponding to 5% error) (10 points)	Y
Accelerometer functionality reports 9.81 m/s^s in the z-axis with a maximum allowable error of 5%, and 0 m/s^2 in the y-axis and x-axis with a maximum allowable error of 5%.	Testing Accelerometer: 1. Test the accuracy of the gravitational acceleration by placing the IMU on a flat surface. 2. Examine microcontroller output to verify that the acceleration reported in the z-axis corresponds to 9.81m/s^2 +/- 5%, and 0 m/s^2 +/- 5% in the x-axis and y-axis. (10 points)	N
Mechanical switch can detect a force of up to 20N without malfunctioning.	Pressing down with a force of up to 20N on any switch should send a high signal (LED	Y

	will light up). We will test this by placing a weight of 4.5lbs on a switch. (10 points)	
Translates gestures and taps into a MIDI signal with greater than 90% accuracy.	From the defined set of gestures and taps we will perform a series of 100 randomly chosen actions and expect the microcontroller to generate the correct MIDI signal for at least 90 of those actions. We will test this by examining the voltage of the BT transmitter's TX pin against expected MIDI packets. (25 points)	Y
The Bluetooth LE connection allows communication between the two gloves and a receiver within 0 to 5.5m.	We will place a receiver at the minimum and maximum distance and ensure that signals are still received. (15 points)	Y

Appendix B Gesture Table

Table 4 Gesture Table

<table>
<tr><th>Symbol</th><th>Function</th><th>Range</th><th>Requirement</th></tr>
<tr><td>Z Translation (move up/down)</td><td>
<table>
<tr><th>(L/R) Thumb +</th><th>Function/Control Byte Value</th></tr>
<tr><td>L Index</td><td>Volume control/#7</td></tr>
<tr><td>L Middle</td><td>Modulation Wheel/#1</td></tr>
<tr><td>L Ring</td><td>Breath Controller/#2</td></tr>
<tr><td>L Pinky</td><td>Foot Controller/#4</td></tr>
<tr><td>R Index</td><td>Expression Controller/#11</td></tr>
<tr><td>R Middle</td><td>Effect Control #1/#12</td></tr>
<tr><td>R Ring</td><td>Effect Control #2/#13</td></tr>
<tr><td>R Pinky</td><td>General Purpose Controller 1/#16</td></tr>
</table>
</td><td>[0.5g, 8g]</td><td>• Discretizes movement into 128 bins.
• From any given position a positive delta followed by the exact same delta opposite in sign should map to the same bin.</td></tr>
<tr><td>Y Translation (move left/right)</td><td>
<table>
<tr><th>(L/R) Thumb +</th><th>Function/Control Byte Value</th></tr>
<tr><td>L Index</td><td>Balance/#8</td></tr>
<tr><td>L Middle</td><td>Pan/#10</td></tr>
<tr><td>L Ring</td><td>General Purpose Controller 2/#17</td></tr>
</table>
</td><td>[0.5g, 8g]</td><td>• Same requirements as Z Translation</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>L Pinky</td><td>General Purpose Controller 3/#18</td></tr>
<tr><td>R Index</td><td>General Purpose Controller 4/#19</td></tr>
<tr><td>R Middle</td><td>General Purpose Controller 5/#80</td></tr>
<tr><td>R Ring</td><td>General Purpose Controller 6/#81</td></tr>
<tr><td>R Pinky</td><td>General Purpose Controller 7/#82</td></tr>
</table>
</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X Translation (move in/out)</td>
<td>
<table>
<tr><th>(L/R) Thumb +</th><th>Function/Control Byte Value</th></tr>
<tr><td>L Index</td><td>Granular volume control/#39</td></tr>
<tr><td>L Middle</td><td>Granular modulation wheel/#33</td></tr>
<tr><td>L Ring</td><td>Granular breath controller/#34</td></tr>
<tr><td>L Pinky</td><td>Granular foot controller/#36</td></tr>
<tr><td>R Index</td><td>Granular expression controller/#43</td></tr>
<tr><td>R Middle</td><td>Granular effect control #1/#44</td></tr>
<tr><td>R Ring</td><td>Granular effect control #2/#45</td></tr>
<tr><td>R Pinky</td><td>Granular general purpose controller 1/#48</td></tr>
</table>
</td>
<td>[0.5g, 8g]</td>
<td>

Same requirements as Z Translation

</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>X Rotation (roll left/right)</td><td><table>
<tr><th>(L/R) Thumb +</th><th>Function/Control Byte Value</th></tr>
<tr><td>L Index</td><td>Sound Controller 1/#70</td></tr>
<tr><td>L Middle</td><td>Sound Controller 2/#71</td></tr>
<tr><td>L Ring</td><td>Sound Controller 3/#72</td></tr>
<tr><td>L Pinky</td><td>Sound Controller 4/#73</td></tr>
<tr><td>R Index</td><td>Sound Controller 5/#74</td></tr>
<tr><td>R Middle</td><td>Sound Controller 6/#75</td></tr>
<tr><td>R Ring</td><td>Sound Controller 7/#76</td></tr>
<tr><td>R Pinky</td><td>Sound Controller 8/#77</td></tr>
</table></td><td>[10º/s, 90º/s]</td><td>Same requirements as Z Translation</td></tr>
<tr><td>Y Rotation (pitch up/down)</td><td><table>
<tr><th>(L/R) Thumb +</th><th>Function/Control Byte Value</th></tr>
<tr><td>L Index</td><td>Sound Controller 9/#78</td></tr>
<tr><td>L Middle</td><td>Sound Controller 10/#79</td></tr>
<tr><td>L Ring</td><td>Effects 1 Depth/#91</td></tr>
</table></td><td>[10º/s, 90º/s]</td><td>Same requirements as Z Translation</td></tr>
</table>

L Pinky	Effects 2 Depth/#92
R Index	Effects 3 Depth/#93
R Middle	Effects 4 Depth/#94
R Ring	Effects 5 Depth/#95
R Pinky	Portamento Control/#84