Music for Muscular Dystrophy

A Major Qualifying Project (MQP) Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degree of Bachelor of Science in

Electrical and Computer Engineering

Authors:

Joshua Hollyer
Dexuan Tang
Luke Harrington

Project Advisor:

Stephen Bitar

Date:
May 2024

Abstract

Muscular dystrophy is a condition that affects approximately 1 out of every three and a half to six thousand men [1]. This condition severely limits an individual's movement as they age, meaning tasks and hobbies that they once were capable of doing, become more difficult as time passes. As such, there is a need to provide musicians with muscular dystrophy a solution to restore their ability to interact with their instruments. Our group sought to provide one potential solution to this problem by working with an individual who has muscular dystrophy, and designing a way to allow them to play the drums given their limited mobility. To do this, we designed a EMF touch circuit which sends MIDI information to a computer to be able to play the drums or any other electronic instrument the user wants. This circuit was then printed onto a PCB and put in a portable enclosure so that the user could easily setup and use the final product. Upon delivering this project to the user, they shared it was fully functional and easy to use. They wanted there to be some more comfort, but overall they found it was able to do what they wanted of being able to play music.

Executive Summary

Muscular dystrophy is a genetic disease that leads to loss of muscle strength over the course of time. Due to this, those with the disease often have to give up activities that they once enjoyed as they lose the muscle control required to do them. Playing instruments is a way that millions of people across the globe express creativity, mitigate stress, and challenge themselves. Similar to other physical activities, musicians with muscular dystrophy often have to give up their passion as their condition worsens.

In order to give back this ability to some of these musicians, our group worked with a family friend of our project advisor who has muscular dystrophy. Given that building a mechanism to interact with a physical circuit would give limited access to the user and be difficult to troubleshoot if it stops working, so we decided for a more accessible approach: MIDI. MIDI is the universal communication standard for electronic music instruments to interact with music software, such as GarageBand, FL Studio, Abelton Live, and more. Thus, the goal of our project was to design a way for our target user to play the drums again through a MIDI interface. In order to accomplish this goal, our group devised a set of objectives:

1. Determine Project Goals with Target User
2. Determine the User Interface Design
3. Construct a Prototype
4. Bring the Prototype to the Target User and Determine its Ability to Accomplish Goal
5. Construct a Final Prototype

We accomplished these objectives through an extensive research process, iterative design, and direct communication with our target user. Prior to beginning this project, we explored ways that those with limited mobility are capable of playing instruments. This involved viewing previous projects that were similar in nature to ours, interviews with industry experts, and physical experimentation with different approaches to the problem. When physically testing ideas, we came up with three different solutions: sEMG, some form of analog gloves, and vision. sEMG would involve reading the electrical signals from the users muscles (which are still present in someone with muscular dystrophy) to create the MIDI notes. We determined this was not a great solution due to the

irregular nature of nerve signals in the body. Vision also proved too difficult due to the fact it requires large amounts of movement, such as in a VR setup, which would not be possible for our target user. The final solution, some form of analog inputs from gloves slowly turned into the idea of electromagnetic field (EMF) touch sensors.

Although our target user's movement was limited, they were capable of moving their thumb, index, and middle fingers with relative ease. To take advantage of this we designed a circuit that took inputs from three EMF touch sensors and used them as inputs into an Arduino. The signals were obtained using copper tape, which ran through coaxial cables into a circuit design roughly to filter, amplify, and rectify the signal so that it could be output as either a DC on or off that the Arduino could monitor as an input. The Arduino then converted the signals to MIDI inputs for a digital audio workstation (DAW). Once we determined that our solution was functional for our user, we designed a PCB for our circuit and a 3D printed hand rest/circuit mount for ease of use.

When we finally felt confident in our final product, we traveled to our target user to determine if it is intuitive and fun to use. When our user interacted with the pads, it was perfectly responsive, which was a great sign. Additionally, our target user noted that the delay compared to our original capacitive touch sensor approach was much better. However, the target user still noticed some delay when doing successive inputs which we determined was due to a 20ms delay that we implemented via code and forgot to change. Regarding the input interaction being comfortable for our target user, they noted that the 3D printed hand mount was much less comfortable that just the flat board. After playing for a while though, the target user noted that in the future they would prefer a more comfortable way to interact with the project that makes sure their fingers stay in the same spot. There are obviously things that we can and should improve over time, but fundamentally it worked for their use-case.

While we have completed our original sought after goal, there are still some aspects of the design we would have liked to improve on if we had more time. The first of which would be addressing the issues with the circuit's performance. We believe that adding some automatic gain control could help with this issue. Additionally, we could make a much larger PCB with surface mount parts, better tolerances, less sensitive component sizes, and cut up the board with identical placement so it is almost like multiple separate boards in one. In making this bigger board, we could also add 7 more sensors so there is one for every finger, and print another hand mount. Overall,

we accomplished the primary objective of enabling the target user to be able to play the drums through our EMF MIDI controller. Each iteration of the prototype improved upon the previous version and our final prototype accomplished good sensitivity, relative low latency, and easy to use.

Acknowledgements

1. First Acknowledgement

First and foremost, we would like to thank Stephen Bitar for all of the help he has provided throughout the year and the great relationship we have been able to build with him. He constantly challenged us to try new things and helped us when we struggled. He has taught us more than we could ever ask for and for that we are grateful.

1. Second Acknowledgement

We would also like to thank the target user for inviting us to their home on two different occasions to test and discuss how to improve the device and what they liked and didn't. They were hospitable and kind the whole time, and we are glad we got the chance to meet them.

1. Third Acknowledgement

Thank you Scott Barton for the time you took to help us integrate with your devices as well as the effort you spent trying to inspire us to come up with new ideas for how the user can interact with our device.

Contents

List of Tables

List of Figures

1 Introduction

Muscular dystrophy is caused by a mutated gene which slowly interrupts the production of proteins for muscular growth, which causes a progressive weakness and loss of muscle mass [3]. The mutation in the dystrophin gene is either a deletion, specific mutation, or duplication. It is 8,300 to 14,300 times more likely to be present in boys, with 1:3500-6000 men having the mutation [1]. However, women can still be carriers of the mutation while being asymptomatic, meaning they can give the gene to their children. Some common symptoms are falling frequently, walking on toes, and delayed growth in childhood. Around mid twenties is when symptoms can get more severe [3].

It starts when children are younger, often making it difficult to get up or sit down as well as them falling over frequently [3]. Soon they are restricted to a wheel chair and basic tasks like sitting up straight and sleeping become difficult. Even breathing becomes difficult and eventually their heart will give out. It is truly a terrible condition.

Music is an important feature in many people's lives but the loss of motor skills over time, as seen in people with muscular dystrophy, can prevent a person from creating their own music through playing instruments. The scope of our project is to create a MIDI interface that can be easily interacted with by people with muscular dystrophy. In this section, we look to research possible ways for people with muscular dystrophy to interact with a MIDI interface accurately to allow for a consistent and precise music playing experience. The three areas we chose to explore further are as follows: Radar/Vision, Gloves, and sEMG.

1.1 Background

1.1.1 Vision

Machine vision and radar could act as alternatives to traditional mouse and keyboard based computer inputs. Human movement can be detected and mapped in 3D space using Ultra-Wide Band (UWB)/mmWave radars or computer vision technologies. Then, these mappings can be processed and classified into computer key presses. For the scope of our project, these mappings can be processed into MIDI outputs.

Radars and machine vision sensors require zero force to actuate. Compared to traditional

buttons, this is beneficial when the user has limited mobility and strength. They can also act as analog inputs by measuring the direction and velocity of the user's movement.

We looked into two studies on the radar. In both studies, a transmitter and a receiver is needed. In one study, Yuzhang Zang used UWB radar to provide input for a video game. A handheld radar transmitter and a fixed radar receiver provided time of arrival, power, and RMS delay information. The UWB radar can detect hand movements in both the XYZ and rotation axes. They achieved a median spatial accuracy of 4cm [4]. The second study by Santhalingam et al. used mmWave radar to classify American Sign Language (ASL) for interacting with smart home devices. They used two fixed mmWave radars to record spectrograms from the signals reflected by the user's hand when the user performs ASL words. These spectrograms and words were then used to train a neural network. They achieved a maximum classification accuracy of 80% without additional signal processing and 95% with additional signal processing. The disadvantage of this method is the delay caused by the necessity to capture spectrograms for a fixed length of time (4 seconds) before feeding it into the neural network and the processing overhead caused by the neural network itself [5]. We tested the accuracy of this method by passing their data through our own neural network and achieved around 70% classification accuracy. However, we noticed that the data variation is large between individual users so the neural network needs to be trained with large amounts of individualized data to suit a specific user.

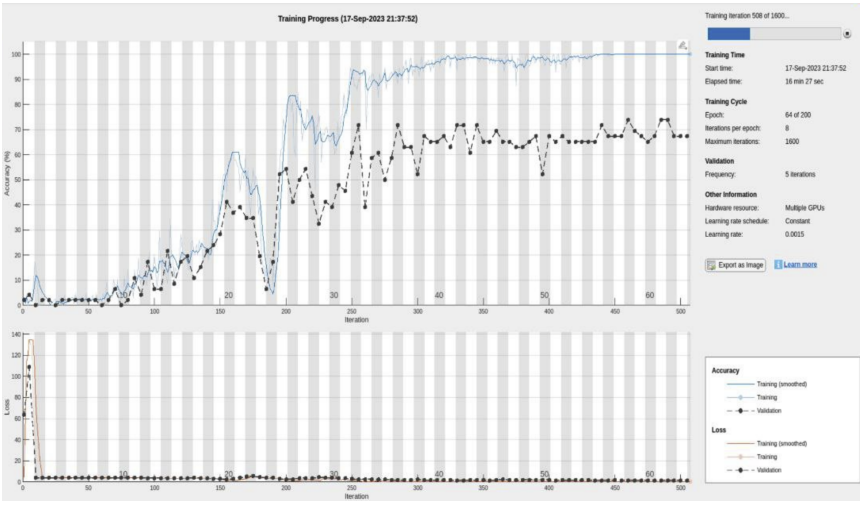

Figure 1: Training Result of Testing mmWave Data.

Compared to radar, motion tracking technologies using cameras are a more proven method

of human-computer interaction. Traditionally, motion tracking systems use markers that attach to the user's body. Software can analyze the spatial relationship between the markers in camera footage. However, many devices now use infrared dot projectors to project "virtual markers" onto the user. Devices such as Microsoft Kinect use this technology. In recent years, IR projectors are also used in smartphones and laptops to map the user's facial features for authentication purposes. We looked into Leap Motion Controller as a potential solution to track finger movements using machine vision [6]. The Leap Motion Controller projects a matrix of infrared dots onto the user's hand and then maps the dots in 3D using stereo cameras. It is designed to be used to track hand movements for Virtual Reality gaming. A study by Ganguly et al. have looked into the accuracy of the Leap Motion Controller in comparison to the marker based tracking system when measuring finger movements of a real human user. They concluded that while the Leap Motion Controller produced very good accuracy when measuring finger movements while the hand is fixed (0.5mm), its accuracy decreased by 5mm on average when the hand was moving constantly [7].

In conclusion, the Leap Motion Controller can potentially provide more consistent results due to its low latency and relatively high accuracy. It could also be less sensitive to interference since its sensing distance and Field of View is less than that of the radars. However, the Leap Motion Controller relies on proprietary hardware and software and requires line of sight to the objects being tracked. As an alternative, we could also explore a combination of the two radar methods by processing the mmWave radar data spatially to achieve lower latency compared to the machine learning method and higher accuracy than the UWB radar.

1.1.2 Gloves

While the previous examples relied on some form of vision to see the movement of a hand, this third solution involves using physical sensors to measure movement such as accelerometers, resistors, capacitors, and more. This solution has many different approaches and there is significant research that has already been done. We will discuss various solutions that have been made in the order of how close to a final product they are.

The first is a study that was published Jul 25 of 2021. This study looked to see the effectiveness of placing strain gauges in gloves to measure movement [8]. Strain gauges change their resistance depending on how much they have been either stretched or compressed, which can then

be used to create a corresponding voltage and as such a signal. There has been work into creating soft strain gauges that can be put into a glove in recent years vs. a stiff metal gauge. This study was able to find that after placing these gauges in a glove they were able to measure resistance changes, however with a latency of around 0.5s, which is significant [8]. Thus while this was a decent study, it needs much more work.

This leads to the second solution that two different types of gloves have used. They attached strings to the fingers of a glove and used a spring loaded potentiometer connected to it. There was one made by an independent designer named Hunter Scott and one on AliExpress. While the one on AliExpress looks promising, it lacks serious code backing it and details for its operation, so we decided not to pursue it as a viable option [9]. The design by Scott is much more rudimentary but clearer in its creation and operation. He chose to use joystick potentiometers due to their ability to be spring loaded and retract, allowing for measurements to be taken forwards and backwards, as well as the string being able to reset its position. He then used these to create a voltage divider and fed the input into an Arduino to read the value [10]. While it was rough in execution, the higher level operation of the gloves was effective at being able to read motion, and should be a consideration for how to create gloves.

The third most complete implementation is the one by CyberGlove. The intent of this glove is to be used with motion capture to operate things such as robot hands. They have their own patented technology as well as wireless functionality [11]. Unlike the previous examples, this was a finished commercial product. Their newest model, the CyberGlove III, has up to 22 sensors to measure the palm arch, finger movement, and an external camera for motion tracking [11]. The problem is that their product does not have extensive details for its operation. Our team reached out to them for further details but we have yet to hear back.

The most complete solution that we have found are the Mi.Mu gloves. These gloves have a fully integrated hardware and software that allows a user to play music by just moving their hands. The gloves have buttons for changing modes, an IMU to measure the motion and rotation of a hand, stretch sensors in the fingers to be able to get button pressing. The gloves have a microcontroller that communicates over wifi with their software (or plugin) to make music [12]. Thus these gloves are fully capable of doing what is desired for our project, but we are not sure of their sensitivity and if they are able to be operated using the small amounts of motion that are available to someone with muscular dystrophy. Additionally, they are not yet a released product and are on preorder,

so the real world effectiveness is to be seen. We reached out to their company and are currently waiting for a reply.

Thus for analog gloves there have been four different attempted solutions: metal based strain gauges, fabric based strain gauges, spring loaded potentiometers and strings, and IMUs/accelerometers. Each of these has their own pros and cons but learning or using the Mi.Mu glove seems to be the most viable option for the analog glove path.

1.1.3 sEMG

Electromyography (EMG) is a medical procedure used most commonly to determine the condition of muscles in the human body and the response those muscles have to impulses from nerve cells or motor neurons [13]. Nerve cells cause muscles to contract via small electrical signals (myoelectric signals) which allow the human body to move as expected. In order to perform an EMG, small needles called electrodes are inserted into the muscles through the skin and then these myoelectric signals are displayed onto an oscilloscope [14]. The signals that are gathered from this procedure can help medical professionals diagnose conditions such as: amyotrophic lateral sclerosis (ALS), carpal tunnel syndrome, cervical spondylosis, and muscular dystrophy [15]. Although EMG is an accurate way to diagnose these conditions, it is extremely invasive due to the fact that electrodes are placed inside of the body. A similar method of extracting myoelectric data is surface electromyography or sEMG which uses an area electrode that is placed on top of the skin. Surface electromyography is used most commonly in areas such as prosthesis (artificial body part) control, "...ergonomics, movement and gait analysis, and sports medicine." [16].

For the scope of our project, we determined one of the areas of our research should be directed towards the use of sEMG to control a MIDI interface. We were interested in this area of research due to the fact that in people with muscular dystrophy, nerve impulses still reach the muscles but don't cause the same contraction within the muscle when compared to someone without the condition [17]. This fact inspired us to research the use of sEMG further as we expect a normal electric output on an oscilloscope via a sEMG from a person with muscular dystrophy.

In our initial research, we found projects online that carried similar goals to that of our project. For example, an electrical and computer engineering project at Cornell University saw a student make a simple game using a microcontroller and a TFT display [18]. The inputs are

two signals (on the left and right arm) from an sEMG that control a sprite's movement in a one-dimension plane. From the project website, this process was accomplished by filtering the sEMG from each arm through a bandpass filter with a low pass cutoff of 500Hz and a high pass cutoff of 10Hz. Then, the signal is amplified with a gain of 100 before being sent into a microcontroller and then the game data being outputted onto the TFT display . This project shows off the potential use of sEMG signals for game/entertainment uses but does raise some concerns. The project page provides a few videos of gameplay and signal outputs. When analyzing the gameplay video, there are a few moments when the user provides a nerve impulse by contracting the muscle in one of the arms and the game either doesn't register the movement or double counts the movement. It is difficult to determine if these discrepancies are due to poor signal quality or issues with the game code, but it provides insight into issues that may arise when using a sEMG for MIDI applications.

The next project that we researched related directly to the scope of our project. A project posted by user johnwmillr to the DIY website intructables.com saw a sEMG signal be used to control a MIDI interface. The project overall consists of two halves. The first half involves the construction of a nerve impulse amplifier circuit proposed by Linear Technology on the datasheet for their LT1167 instrumentation amplifier chip. The second half uses an Arduino to convert the signals from the first circuit to MIDI notes and send them to an interface. The final project results in musical note volume and pitch output corresponding to nerve impulse caused by the contraction of either the right or left arm respectively [19]. Our group found a very similar project to this on the website backyardbrains.com which uses a custom made PCB called the Muscle SpikerShield Pro to measure up to six sEMG signals and send them directly to an Arduino to be converted to a MIDI output [20]. These two projects are very similar and thus yielded similar results. Both project pages provided videos of their final results and it can be seen in both videos that although an output sound is recorded corresponding to muscle contraction, accuracy seems to be an issue. For example, in the instructables project, johnwmillr states in his video that although he can somewhat control the volume with his right arm to a degree of accuracy, the pitch doesn't stay constant because the voltage output of the sEMG is constantly changing. The output of this project does provide sound but this sound is not directly musical and is rather just a collection of random noise that begins and ends corresponding to the output of a nerve impulse. Similarly, in the video provided by backyardbrains, they choose to represent their project in the form of a MIDI drum kit. Although some beats do line up with the nerve impulse/muscle contraction, there are audible

"phantom beats" that are played even though no visible contraction was provided. Similar to the Cornell project, these two projects provide insight into potential accuracy issues when using sEMG. This can be a cause for concern regarding our project as our goal is to provide an accurate musical interface for people with muscular dystrophy and if there are accuracy issues in this process, the user could lose some of the authenticity and precision of playing a real instrument.

The projects we chose to research inspired us to do some physical research of our own. For our research, we chose to build the nerve impulse amplifier circuit provided by Linear Technology and utilized by johnwmillr. The goal of building this circuit was to determine the quality of the output signal to see if we could improve its accuracy for potential use in our project. For this circuit, we used the list of materials seen in 1.

Component	**Quantity**
LM358	1
LT1167	1
15nF Capacitor	1
0.01uF cap	1
0.47uF cap	1
100 resistor	1
6k resistor	1
10k resistor	1
12k resistor	1
30k resistor	2
1M resistor	2
Audio Jack	1
sEMG Nodes	1

Table 1: Components used for first sEMG experiment

After constructing this circuit seen in Figure 2, we hooked it up to an oscilloscope and viewed the output seen in Figure 3 after continuous muscle contraction in Luke's right arm.

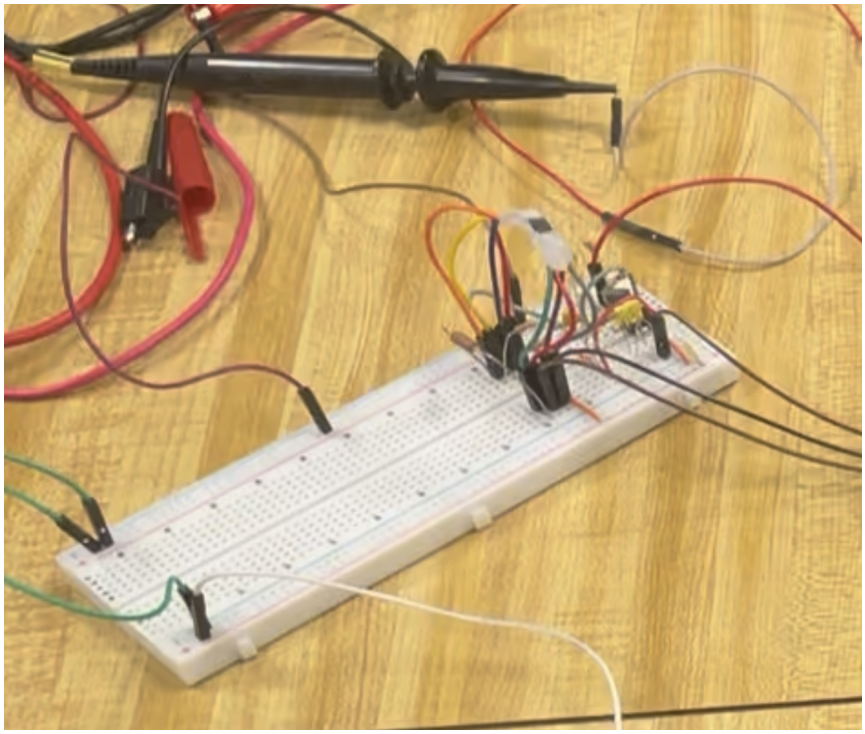

Figure 2: The sEMG circuit.

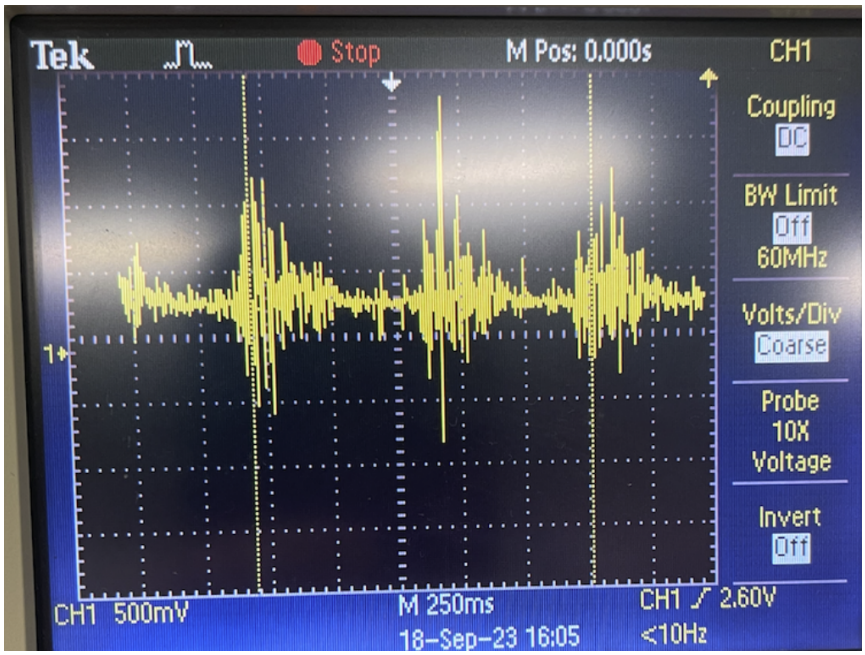

Figure 3: Output of the sEMG Visualized on an Oscilloscope.

The output of this circuit provides a clear indication when a muscle is being contracted/nerve

impulse is being sent, but that signal is by no means "clean". It could be possible that the messiness of this signal has to do with some of the inaccuracy seen in the above projects but it is hard to tell. If we were to use the sEMG for our project, we would want to "clean up" the signal by possibly using a schmitt trigger or digital signal processing so that we see either a low or high value which we can then use as a definitive input to our system.

Overall, the use of sEMG as an input to our project is an exemplary candidate. As seen in the example projects, its use for a MIDI interface is possible and has been done before. Although it can be done, before we decide to use it in our project, we need to determine if the inaccuracies can be eradicated so as to provide our users a precise and enjoyable music making experience.

1.1.4 Takeaways

After researching these three potential solutions for designing a MIDI input device, we gained some useful insight. First is that the machine vision method is more repeatable and produces less latency than the mmWave radar approach. Analog gloves possess less complexity and less latency than the machine vision approach; however they are application specific. We could adapt the machine vision software to different instruments, but the analog glove is limited to a single type of instrument. sEMG is invasive due to the amount of sensors needed to be attached to the user's body. It is also difficult to acquire accurate and consistent signals using sEMG. In the next section, we conducted preliminary tests on the three approaches to determine a single or a combination of possible solutions.

2 Preliminary Test Results

2.1 Vision

As found in the background research section, we determined that the Leap Motion controller was the best option due to its complete hardware and software implementation. To use it, a few pieces of software and drivers were needed. First was installing Ultraleap Tracking, which is both a tool and the driver used for controlling the Leap Motion when it is connected to a computer over USB. It is able to take the readings from the Leap Motion and feed it into other pieces of software and is able to provide accurate motion tracking. It also has some customization such as choosing how the Leap Motion is being mounted (tabletop vs. head mounted vs. above display). Now that the computer can grab readings from the Leap Motion, it needs in some way to be able to convert this into MIDI information. Two pieces of software have been developed for this reason: MIDI Paw and Glover. We first tried using MIDI Paw and while it functionally can do many different things, it is not as refined and polished as Glover, and as such we decided to use Glover instead. In this software you can select various inputs, which include other devices like their own Mi.Mu gloves, select your movement type from the device, then select a MIDI control to go with that movement. You can then use its output to go into any DAW (digital audio workspace) that you desire.

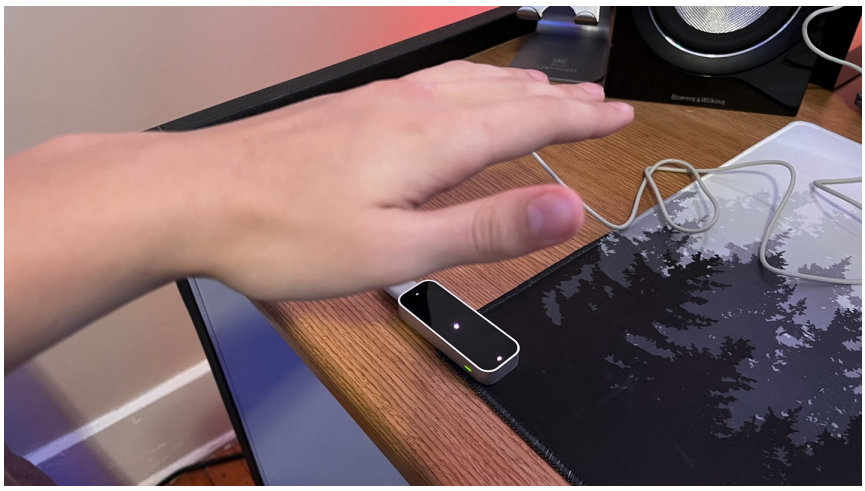

Figure 4: The Leap Motion on a Desk Being Used.

The Leap Motion was placed on a desktop and selected in this software. We began by doing simple tests, such as moving the hands up and down to adjust volume, and side to side for

a filter. All of these things worked which was great. Proceeding this, we wanted to move to more nuanced hand movements, which are more realistic for someone with muscular dystrophy, to control singular note actions. This unfortunately did not work out as well as expected. First, it is unable to do sustain and release, and second it is extremely hard to tune. You can have to manually set values for where your fingers need to be to trigger a note on and this changes based on the height of your hand above the controller. Additionally its reading accuracy is not as good as it needs to be, and leads to a lot of mishit notes because of both the controller and the fact that when trying to move individual fingers you often move other fingers as well, which adds more to the error. There is another mode in the Glover software that allows for the controller to play set chords or scales and assign these to different movements. This, while usable, leads to all the middle notes being played, sort of like a slide whistle.

The main results from this part of the research is that the Leap Motion is good for doing analog movements for filters, volume, and more, and may work in combination with the hall effect sensor solution. However, as a standalone musical instrument, it is not adequate for what this project demands.

2.2 Analog Gloves

The analog gloves we researched all depend on some form of mechanical sensors that measure joint movements of the user's hands. The disadvantage of mechanical sensors is that they require some actuation force in order to produce an output. Because the analog gloves mentioned in the previous section are difficult to obtain, we could not verify the feasibility of using mechanical sensors with muscular dystrophy. Therefore, we decided to test a hybrid approach of the radar and the analog gloves. We constructed a prototype that uses magnets that are attached to gloves and a keyboard shaped hall effect sensor array. When using the device, the user wears the glove and places each finger above one hall effect sensor. And an input is registered when the magnet is at a certain proximity to the hall effect sensor. The input is then converted to MIDI notes using an Arduino microcontroller and sent to a computer running a Digital Audio Workstation (DAW).

Figure 5: Hall effect sensor keyboard prototype.

Compared to radars, the hall effect sensor is more accurate and consistent because the distance detection doesn't depend on complex algorithms. Rather, hall effect sensors produce an analog voltage output directly correlated to the distance of the magnet from the sensor. Compared to mechanical sensors, the hall effect sensor does not require any actuation force and its sensing distance can be adjusted in firmware.

During testing, we found out that the hall effect sensor keyboard requires very little finger movement and the note it produced is repeatable. However, because the short actuation distance is required, it is difficult to produce a velocity output using the method commonly found in traditional MIDI keyboards, where the velocity is determined by the time difference of the keypress passes through two thresholds. We also found out that the hall effect sensor approach is not customizable and is best used for designing instruments that mimics a traditional and signal purpose instrument.

2.3 sEMG

Although sEMG was an exciting idea that has been used to interact with MIDI in the past, our group decided to not pursue this solution any further. Initially, the group had hoped that although the sEMG signal was inconsistent and complex in both amplitude and frequency when visualized on an oscilloscope, it could be cleaned up. The plan was to use a half wave rectifying circuit to

get rid of the negative component of the signal, then use an envelope detector to focus on the amplitude of the peak of the signal, and then finally to use a schmitt trigger to produce a clean square wave. Unfortunately, before constructing this the group realized that even with all of this signal processing, the signal would not be of use in a MIDI interface. When interfacing into MIDI, notes are produced with two different variables —- a pitch and a velocity. When processing the signal as planned, a schmitt trigger would allow for an analog on or off at a predetermined voltage level which isn't complex enough to provide either pitch or velocity data. The group realized that an sEMG would only be able to be used as an on or off input to the system, which when trying to play an instrument besides a drum is not complete enough. For this reason, the group determined that this method of interfacing with MIDI should not be pursued any further as there are better solutions.

3 Methodology

The goal of our project is to successfully design and produce a device which will allow musicians with muscular dystrophy to practice their music. The device will interface into a MIDI environment which will allow for various instruments to be played. Additionally, this device will need to be accurate and comfortable enough to adequately provide an enjoyable experience for the user. In order to achieve our project goal, our group developed five objectives. Figure 6 is a graphical representation of our objectives.

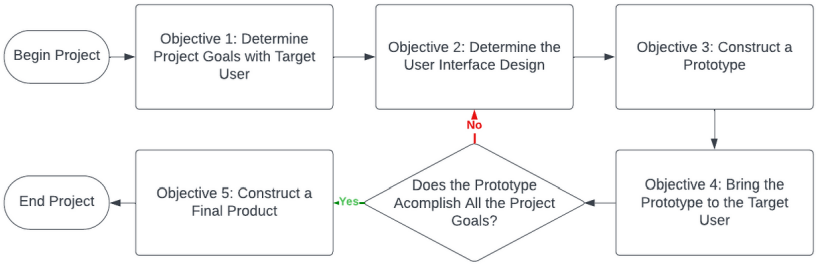

Figure 6: Objective flow chart.

3.1 Objective 1: Determine Project Goals with Target User

In order to successfully complete this project, we first need to identify a specific target user. Although our project is targeted towards all musicians with muscular dystrophy, we need to identify a specific target person to design our device around. Through our research process, our group met with Tim Yates, the Head of Research and Innovation at Drake Music which is an organization that pioneers the use of accessible music technology for people with disabilities [21]. Tim, who has worked on numerous design projects similar to ours, informed our group that it is pivotal to work closely with a specific target user from the start of the design process. He mentioned that it is extremely easy to design something that is technologically advanced but has no benefit to the target user and that our group should avoid this by practicing co-design. From this advice, our group plans on centering our project design around a specific individual with muscular dystrophy in order to work with them throughout the entire process. Our project advisor, Professor Steven Bitar, invited us to participate in this MQP project as he has a family member with muscular dystrophy who happens to enjoy making music. Because of this, we plan on designing our project

around the music making preferences of this individual. Before we begin the design process of our device, our group plans on meeting with this individual to determine their physical limitations as well as their musical preferences. Their musical preferences are important in the design process as our group needs to determine what instruments the user wants to play, the precision at which the user wants to play them, as well as how they want to interact with the instrument (ie. pressing keys on a keyboard, strumming a guitar, etc). In this meeting, our group will bring a very basic MIDI interface designed from the preliminary test results of our research. The purpose of this is to provide the individual with a sense of direction when understanding what a MIDI interface is and modified to fit the constraints of their condition. By meeting with this individual at the beginning of our project, our group will be able to design a project that more directly serves the needs of a specific user which could be applied to other users in the future.

3.2 Objective 2: Determine the User Interface Design

Once the group has determined the specific goals of our project by communicating with the target individual, we can begin designing. In this stage of the project, the group will use the information gathered in both our initial research and objective 1 and use it to determine a potential solution. The group will perform both literature and physical research in order to determine the best input into the MIDI interface that will accomplish the project goals set with the target individual.

3.3 Objective 3: Construct a Prototype

Once the group has determined the user interface for the design, the fabrication process can begin. The goal of this objective is for the group to build a functioning prototype that incorporates the design features set by the individual in objective 1 through the interface design determined in objective 2.

3.4 Objective 4: Bring the Prototype to the Target User

Upon the completion of the prototype, the group needs to test its actual function compared to the desired function set by the target individual. For this part of the project, the group will meet in person with the target individual and have them use the prototype. From this, the target

individual will be able to inform the group on how the prototype works for them and how it compares to the project goals set in objective 1. If the target individual determines that the prototype accomplishes all of the project goals and desired aspects, the group can move on to building a final product. However, if the target individual doesn't feel that the prototype successfully accomplishes the project goals, then the group will begin an iterative process. In this iterative process, the group will go back through objectives 2, 3, and 4 in order to determine a new user interface, construct a prototype, and have the target individual test the prototype until the proposed solution accomplishes all of the project goals.

3.5 Objective 5: Construct a Final Product

Finally, once the target individual approves the prototype, the group can begin constructing a final product. In this part of the project, the group will use the same project characteristics that were approved by the user, but this time will construct something more presentable for the target individual. This will be a completed product that is user friendly, compact, and durable. Through the completion of these objectives, the group hopes to accomplish the project goals set by Bitar and our target individual.

4 Initial Project Preparation

When beginning this project, we determined a set of objectives that we wanted to complete in order to successfully accomplish our project goal. We initially planned on meeting with our target user straight off the bat in order to determine the goals they wanted us to accomplish in order to accommodate their needs. Upon discussing this plan further with Professor Bitar, he agreed that this meeting was important but proposed to us a different way of approaching it. Unfortunately, there was going to be some delay in when we would be able to meet with our target user and additionally, the wide variety of methods of approaching this project might be overwhelming to learn about all at once. Because of this, Professor Bitar tasked us to spend a few weeks constructing the best prototypes of the best methods of interfacing into MIDI that we could in order to arrive at our meeting with our target user with some ideas that they could base their project goals around something they have already seen and interacted with before. We agreed with this idea and for the next few weeks, began developing three interfacing methods. We did so first by speaking to people who have done research and work in similar topics and then after using the principles we learned to create our first prototypes.

4.1 Interviews

4.1.1 Meeting with Drake Music

Our first interview was with Tim Yates, a member of Drake Music. Drake music works to help people with disabilities play music using technology [21]. During our meeting with Yates he gave lots of insight into the actual process of designing one of these instruments. He believes that to properly make the instrument, the target user should essentially be a co-designer. With them making suggestions and constant feedback as you are creating the product. It is often easy to design something technically impressive and cool to use, but when it is actually in the hands of the target user it can be completely unusable. He believes that creating an instrument that is not intrusive and has feedback from someone with muscular dystrophy would be a great way to continue to create our project. As such, we made sure that we could show our design concepts to someone with muscular dystrophy (see later section on the target user's interview).

4.1.2 Meeting with Professor Barton

With our further insight into how to work with a target user, we also wanted to speak with a music professor at the school who deals with music technology, which is how we came in contact with Scott Barton. When we spoke with him, he spoke of the importance of how you interact with an instrument. When someone traditionally plays a piano or a guitar, for example, hitting one string creates a single note. But in the modern era, technology allows for a more abstract and generative approach to music, where say you may only want control over a melody and let a computer create a chord progression. He also showed us some of his work he has done, and one of which was a self playing drum set that can be controlled with MIDI.

4.1.3 Meeting with Mimu

One possible solution in our initial research is the Mimu gloves. Mimu gloves use resistive sensors made of conductive fibers for measuring bends in fingers and IMUs to measure the movements of the hands and wrists. It also provides haptic and visual feedback. The sensor data is sent over wifi to a computer and processed into MIDI inputs using the Glover software.

In order to determine the feasibility and design of the Mimu gloves, we met with Adam Stark who works at Mimu's software team. We explained to Adam the scope of our project and our progress so far. Then we asked him about the capabilities of the Mimu gloves.

From the interview, we learned that the Mimu gloves are sensitive to small finger movements for all fingers except the thumbs, and the IMUs are sensitive for determining the hand orientations. However The Mimu gloves are not suitable for our target user because wrist movements will be very difficult to measure if the hands are placed on a desk and the Mimu gloves are currently not in production due to supply chain issues.

Adam also suggested we look further into the Leap motion controller for detecting finger movements. To solve the low sensitivity when using the leap motion from above, he proposed either to place the Leap motion controller under an acrylic surface or building a sling system to support the hands. He also mentioned TouchKeys and Genki wave rings [22]. TouchKeys are capacitive sensors that can be placed on top of piano keys which measure the player's hand location on the keys. Genki wave rings are bluetooth MIDI controller rings containing IMUs that can be worn on

fingers.

4.2 Initial Designs

4.2.1 Hall Effect/Arduino Capacitive Touch

The Hall effect sensor keyboard prototype shown in Figure 5 was attached to a computer mouse shaped palm rest for ergonomics. Hall effect sensors were placed under where the user's fingers would naturally rest. The nitrile glove was replaced with a fabric glove with circular magnets attached on each finger tips for re-usability.

Figure 7: Hall effect sensor and glove prototype.

The same Arduino code from the first prototype was used. The code first checks the voltage output from each Hall effect sensor and converts them to an 8 bit value. It then sends the corresponding MIDI note for the sensors with readings above a set threshold through USB to a computer running a DAW. The pentatonic scale was used since there are only five hall effect sensors on this prototype.

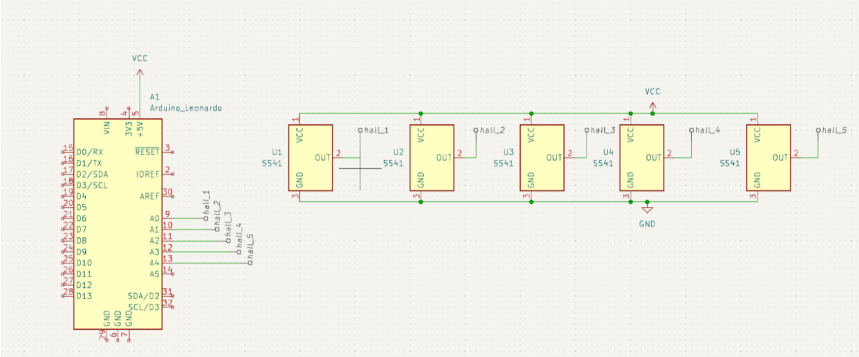

Figure 8: Hall effect sensor and arduino schematics.

We tested the device by playing a melody. Although the Hall effect sensors performed well in terms of latency. We found it to be difficult to place fingers above the sensors precisely due to the narrow field of view of the sensors. This may be especially difficult for our target users. The magnets also caused the fingers on the glove to attract to each other which causes difficulties when storing and putting on the glove.

To improve on these drawbacks, we tested capacitive touch sensors. Compared to Hall effect sensors and gloves, capacitive touch sensors provide direct tactile feedback to the user, thus making positioning the fingers easier. Without the bulky glove, it is also easier for the target user to see the placement of their finger in relation to the capacitive touch pads. It also retains the user interface advantages of the Hall effect sensors since it also requires no actuation force and very little movements.

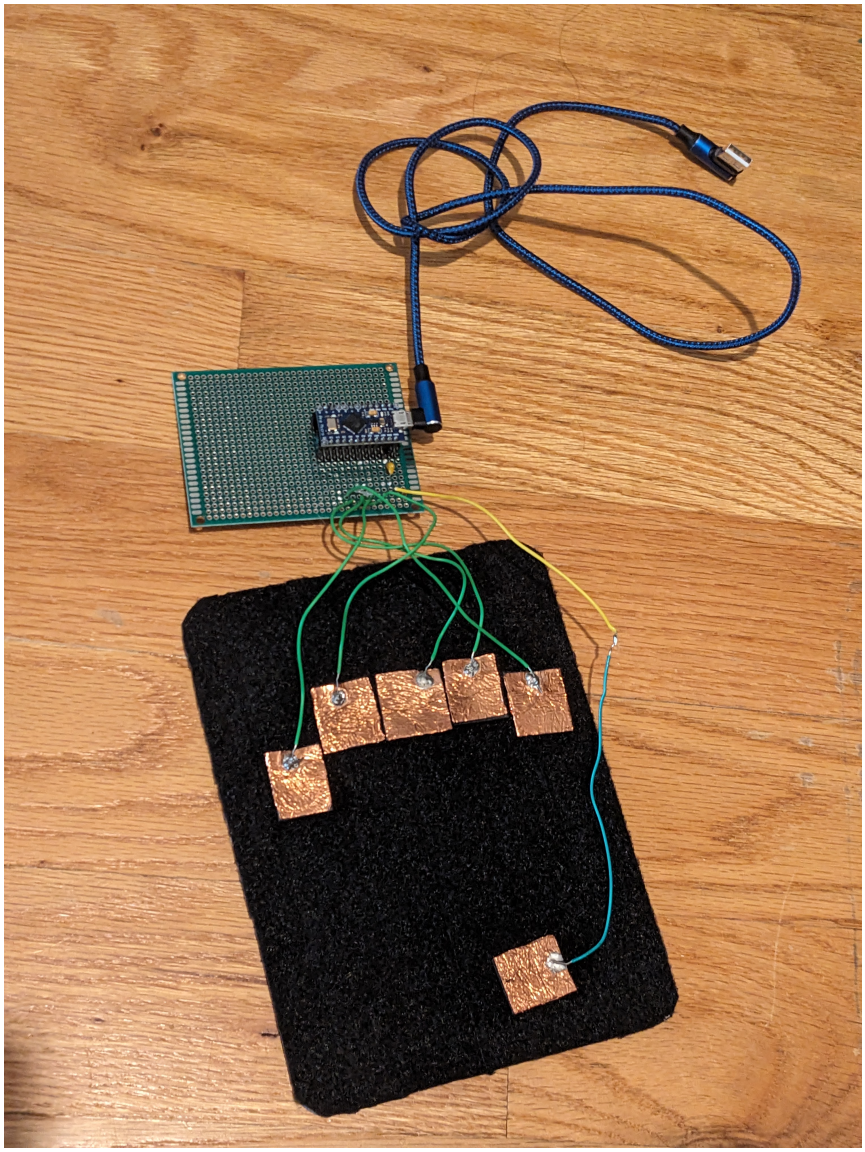

Figure 9: Capacitive touch sensor pads and velcro base plate.

The touch sensors were constructed using copper foil tapes on top of velcros as seen in Figure 9 . The sensors were then placed based on the target user's hand size on a 3D printed plate covered with velcro. An additional pad for grounding the user's palm was added because through testing we noticed the capacitive touch sensor produced less latency when the hand was capacitively coupled to the Arduino ground.

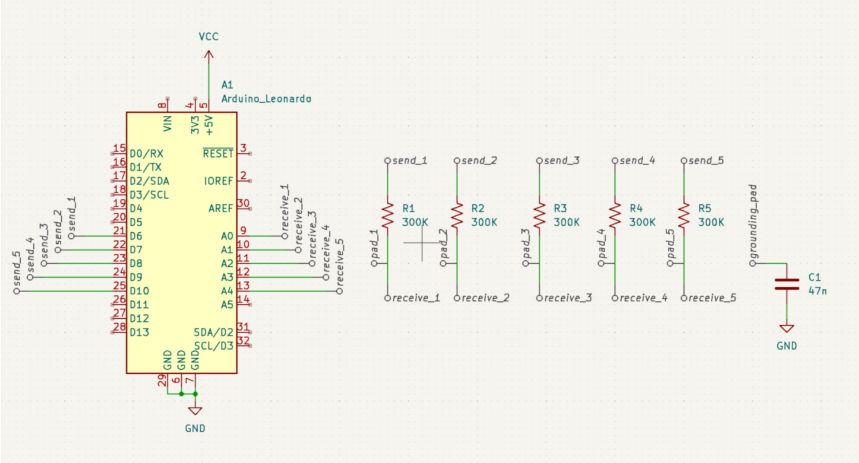

Figure 10: Capacitive touch sensor and arduino schematics.

Figure 10 shows the connection between the Arduino and the capacitive touch sensor pads. Each capacitive sensor and series resistor forms an RC lowpass filter with the input voltage being the send pins and the output being the receive pins. Each pad pin represents one of the two plates of a capacitor. The user's hand forms the other plate and is capacitively coupled to ground through C1.

We modified the Arduino code with the "CapacitiveSensor" library which replaces the hall effect sensor detection section [23]. The library detects touch by sending a 5V step from the send pins and measures the rise time on the receive pins. The two pins are always at the same voltage when the pads are not touched, therefore the rise time is zero. When the user touches the pad, the rise time can be calculated with tr=2.2RC. The library takes multiple rise time samples for accuracy and converts the rise time measurements to a 8 bit integer. This enabled us to use the same threshold detection and MIDI code.

Depending on what instrument is chosen within the DAW, the capacitive touch MIDI controller can be operated in two ways. If an instrument with sustain, such as a pipe organ, is chosen, the user can hover their finger above the touch pads and play the notes by touching the pads. If an instrument without sustain, such as drums, is chosen, the user can either play using the same method or the user can place their fingers on the pads first and play by lifting fingers and hitting the touch sensors again.

Through testing, we found capacitive touch sensors to be more user friendly than Hall effect sensors. It was very easy to play a melody or a drum beat with the first method. However,

we noticed a significant delay when testing the second method of playing. This is caused by the Arduino having to measure the rise time of all sensors when all fingers were placed down. In order to decrease rise time, we initially tried to decrease the resistance from the initial 680K to 300K. This resulted in less delay but also reduced sensitivity. We noticed that the sensitivity improved when the other hand was placed on the laptop, thus we added a pad to ground the palm of the hand through a 47nF capacitor to mimic the capacitance between the Arduino's ground and the laptop chassis.

Since an Arduino can only run programs sequentially, another way to solve the latency problem is to use less sensors per Arduino. This would enable multiple rise time measurements being taken simultaneously. Through testing we concluded that three or less sensors per Arduino produced acceptable delay.

4.2.2 Analog EMF Touch Sensor

Due to the issues we were having regarding latency when using the Arduino directly for the capacitive touch sensors, we met with Professor Bitar who proposed to us that we use an analog circuit that outputs into the input of an Arduino. In order to complete this task, we wanted to re-use our copper tape strips that we had setup initially for the Arduino touch sensors. In order to do this, we first put a probe on the end of a wire attached to one of these copper strips and viewed an oscilloscope when a finger touched down on the pad. When contact with the pad was made, a 60Hz 1V peak sine wave was seen on the oscilloscope. From this, we knew we wanted an approximately 3.3V or 5V DC output as these are nominal voltage values used at inputs into an Arduino. In order to do this, we first would send our input signal from the copper strip into a unity gain buffer in order to isolate the signal into a new stage of the circuit with a low impedance, then we would use an envelope detector to half wave rectify the signal and then charge/discharge a capacitor in order to receive a closer to DC threshold that the signal would fall between. Then, using a comparator circuit, we would set a threshold below the lowest value our capacitor discharged to and then have that comparator output a DC voltage. In order to test if this circuit worked, we first constructed it in Multisim as seen in Figures 11 and 12.

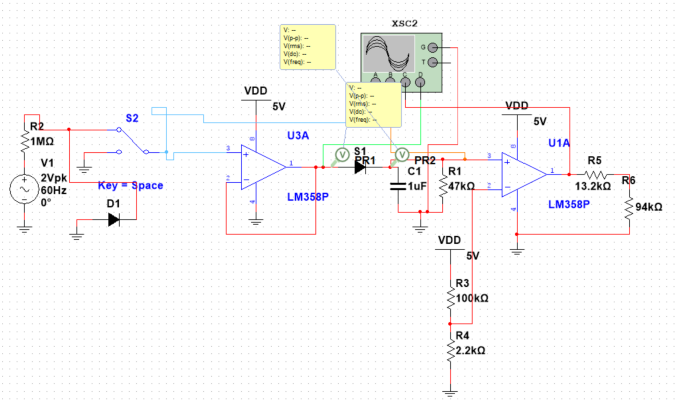

Figure 11: The schematic of our circuit seen in Multisim.

Figure 12: Output of our simulated circuit as seen on an oscilloscope.

When simulating this circuit, it performed how we expected it to. Looking at Figures 11 and 12, the blue signal is the input into the whole circuit, seen at the output of S2. We simulated this with a 60Hz 2V peak source that would pass through a 1M ohm resistor, simulating the human body, and then traveling in parallel to either the input of an LM358 Op-Amp or a diode with its anode attached to ground in order to somewhat half wave rectify the signal even before the unity gain buffer. Then, after passing through the unity gain buffer (seen as the green signal on the oscilloscope), the signal would pass through the envelope detector (seen as the orange signal on the oscilloscope). The output of the envelope detector was then sent into another LM358 setup as a comparator with a threshold set (using a voltage divider) on the inverting input to be a little less than the minimum value of the envelope detector. Then, using another voltage divider, the output voltage of approximately 3.3V was achieved. Although this circuit simulated perfectly with our

chosen component values and the LM358 chips, when we went to build this circuit and simulate it in real life, we ran into an issue. When using this circuit, our envelope detector would not charge an discharge the capacitor around the peaks of the input signal, instead it was sitting around 0V and then charging to a few millivolts before discharging negatively. We were extremely confused as to how this was happening and reached out to Professor Bitar for some insight. Professor Bitar suggested that the LM358 may not have a high enough input impedance (around 10M ohms) for our circuit to work as desired so in order to mitigate this, he proposed that we use a MCP6002 chip which has an input impedance of $10\hat{1}3$ ohms. When we substituted this op-amp chip into our initial circuit, it began to function as desired and we decided to then hook it up to the Arduino to test it further. Upon doing this we found that when touching the copper pads to input into the circuit, it functioned well, however, if the user were to touch areas like ground or the laptop that the Arduino was plugged in to, the circuit would trigger. This was extremely problematic as not only was the circuit not functioning as we desired, but this would not provide a satisfying music making experience for the user. We were once again in a situation where we were unsure with how to solve this issue but luckily, in our next scheduled meeting with Professor Bitar, he proposed a new circuit to us. Professor Bitar had modified our initial circuit to produce the schematic in Figure 13.

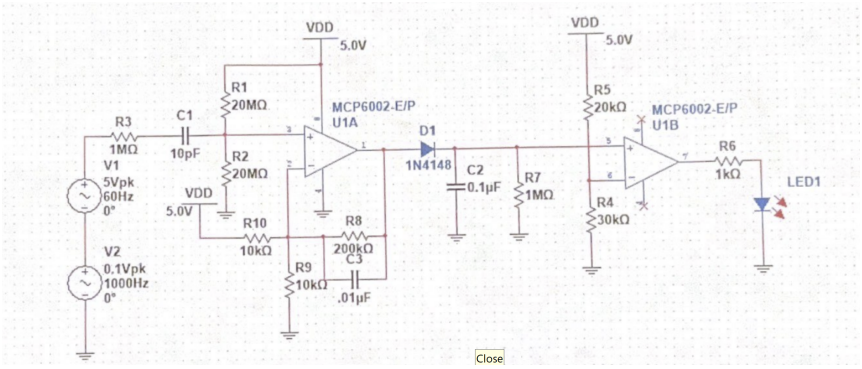

Figure 13: Schematic of the modified circuit.

When testing this circuit with the Arduino, it performed how we wanted and most importantly, didn't have the issue of triggering when ground was touched. Although this circuit accomplished what we wanted, we didn't have time to replicate and build more for other fingers in order to use it for testing with our target user, but we were confident that testing with just the Arduino touch sensors would provide us with enough feed back before we decided on continuing

down this path.

4.2.3 Leap Motion

Another path our team explored was the use of the leap motion controller. The previous section in Vision delves into our initial experimentation with it, but we wanted to further explore its capabilities with it. Our first step was to purchase the more advanced leap motion 2, which offers better sensing, a wider viewing angle, and a smaller size (and from now on I will just say leap motion but am referring to the 2nd version).

Figure 14: Comparison of the leap motion (rear) and the leap motion 2 (front) [2].

To further improve the testing, we wanted to make sure that the leap motion was positioned in a way for optimal sensing. Our two design ideas for this were to place the leap motion on a tripod facing the user's hands so that it gets a stable, wide angle aerial view and to create a head mount to allow for a VR style view for the camera. The first option had difficulty recognizing the hands due to the distance of the camera to the hands, so that idea was scrapped fairly quickly. The second option was more promising, since it was close to the user's hands and has a symmetrical view of the hands. The initial testing for this was promising, but it seemed to be having accuracy issues. What we discovered was that it did not like having hands flat on a surface, it in fact preferred facing inwards (as if you were about to clap). The more serious issue was that the readings from the headset are with reference to the headset and not the hands relative to their environment. What this means is that if the user turns their head slightly or forwards/backwards, it sees the hands as

"moving." What this translates too is getting changing inputs from just simple head movements.

Figure 15: Leap Motion 2 mounted to a helmet for a VR view of hand movement.

Additionally, the sensitivity of the leap motion is variable and it is not consistent, which is the opposite of what you would want out of a controller. What all of these observations had led us to believe was that the leap motion has potential, but would not be a right fit for this application. This was confirmed with our meeting with the target user.

5 Determine Project Goals with Target User – Objective 1

In order to successfully complete our first objective, we needed to meet with our target user to determine the goals of our project. In preparation for this meeting, we made sure that the leap motion and the Arduino capacitive touch sensors were presentable and ready to be used with our MIDI interface. Upon meeting with our target user, we first introduced ourselves and our overall goal for the project. We then went through a series of questions to determine what they wanted from this project. We first determined that their most accessible form of interfacing into MIDI was through Apple's Garageband as they own a Macbook. From this, we also determined they were comfortable with interacting with their laptop on their own as they are able to use the trackpad easily. Next, we asked about the type of instruments they would want to play. We determined that playing the drums was a priority, but they also would be interested in playing the piano and other instruments as well. We then determined with the user that if using their fingers, the thumbs, pointer, and middle fingers would be the easiest for them to use consistently. Using this information, we set up the capacitive touch sensors set up on our 3D printed pad with the Arduino and had them try playing the drums. From this set up, we determined that our current approach had a few issues. First, the fact that the touch sensors were laid out on a flat surface was uncomfortable for our user. Because of this, our user suggested a sort of support like a ball or mouse shaped object could be used for them to rest their hand on for more comfort. Second, when playing the drums, our user was attempting to maintain a fast tempo and because of the latency issues we knew about on the Arduino only capacitive touch sensor circuit, some beats would be missed which was problematic for our user's music making experience. From this, we determined with our user that using our analog EMF touch sensors would be a better approach as they would not have the latency issue. Next, we introduced our user to the leap motion and our desired use for it being filter selection and user when playing instruments. We then set up the device and asked our user to attempt using it but we found that because of how their hands naturally sit, the leap motion was unable to identify their hands and use them properly. From this information, we made the decision with our user to abandon the Leap Motion totally. At the end of our meeting with our user, we determined the following project specifications. Our target user wanted us to design something with touch sensors that allowed their hand to sit comfortably over the sensors and then be able to set up and play their desired instrument with ease and no latency. Upon the completion of this interview, we had

accomplished our first objective and were ready to move onto the design phase of our project.

6 Determine the User Interface Design – Objective 2 & 3

This section documents how we took the knowledge obtained from our meeting with the target user to create a usable prototype to fit his needs. This was the beginning of an iterative process that began after we accomplished objective 1.

6.1 Revision 1

After speaking with the target user, we knew that we needed to make the device into a form factor that could fit under the palm of his hand and give him access to the EMF touch sensors. We decided on a small ball design which had a large area to place the sensors. We decided to do three because he had explained that his mobility of his thumb, pointer finger, and middle finger were the best. Inside of this enclosure we placed the Arduino Pro Micro and our circuitry for the EMF touch sensors, which we built into a PCB. The schematic for this PCB can be seen in Figure 16.

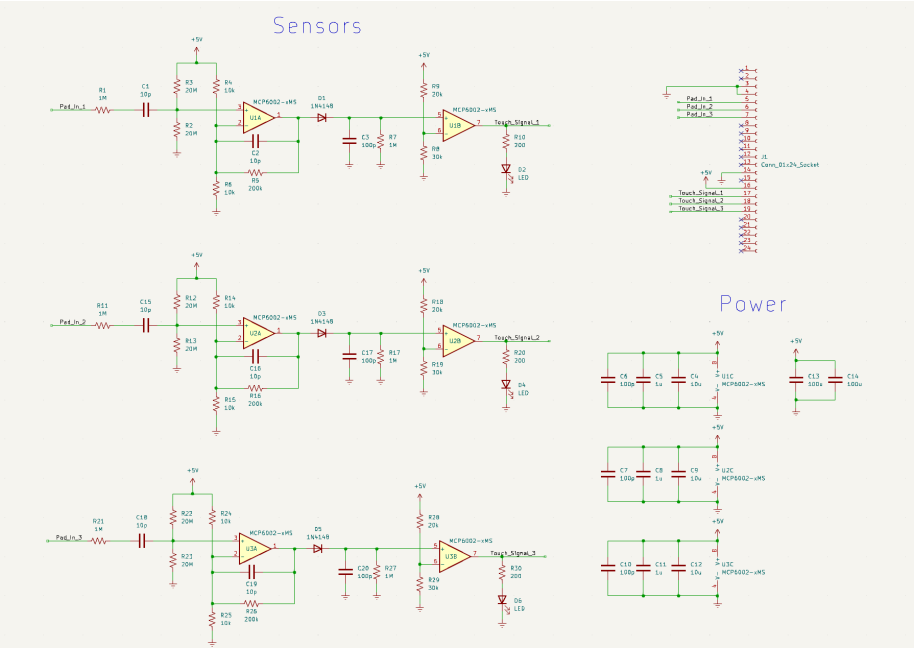

Figure 16: Schematic for the first revision of the PCB.

Here we used the circuit seen in Figure 13 and created 3 copies, one for each sensor. We

also created a connector which can plug into the Arduino's header pins. Finally, we added some various decoupling capacitors for the ICs and the board as a whole. This schematic was then converted into two different PCBs. One, a miniaturized version, and one a hand soldered version. This was done because the lead time of the smaller board was around a month and we wanted a PCB to perfect the code and tuning with software. The miniaturized PCB can be seen in Figure 17 and Figure 18.

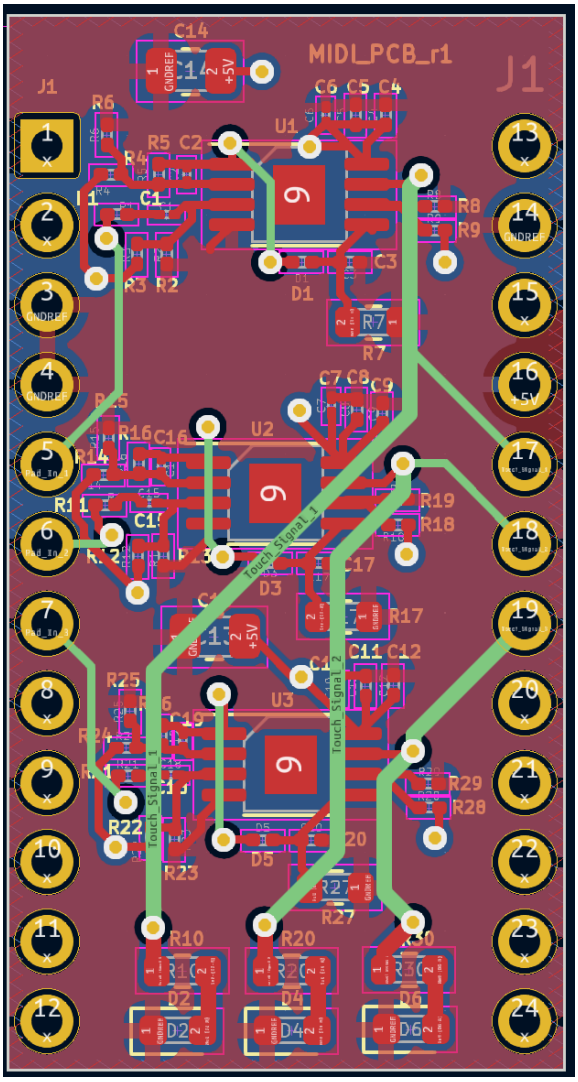

Figure 17: Miniaturized PCB for the first prototype design.

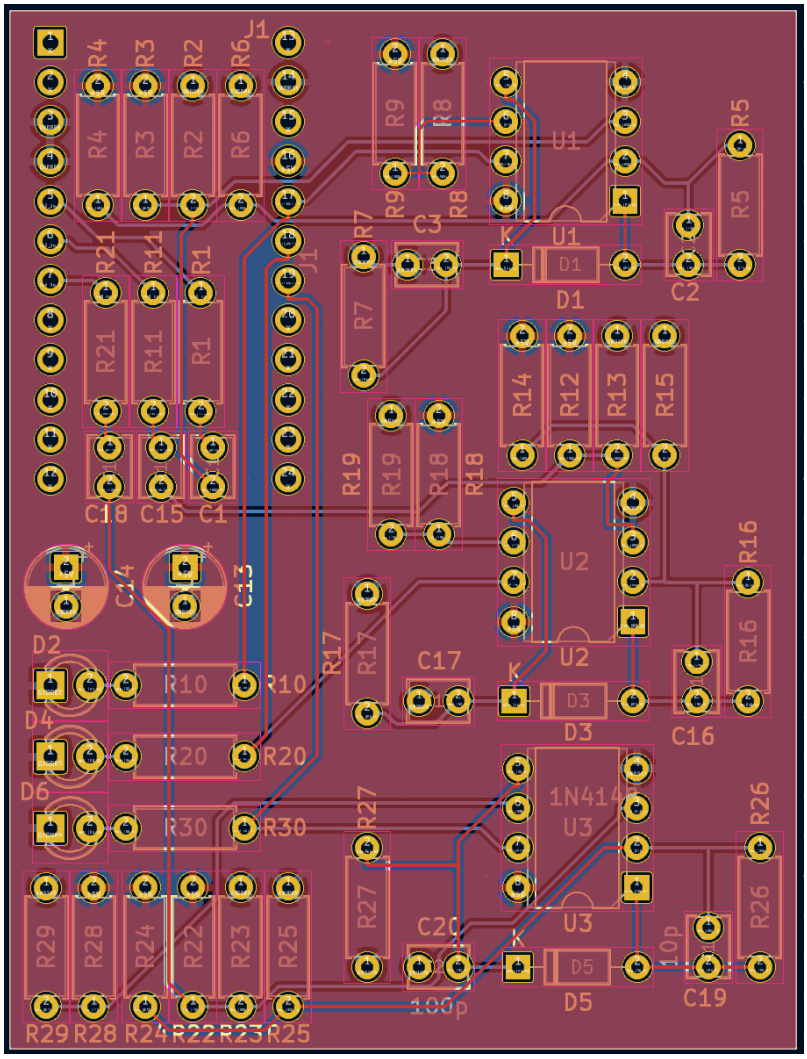

Figure 18: Large PCB for the first prototype design.

We created enclosures with two different palm rest sizes for the next meeting with our target user. From the target user's feedback we learned that he perfer a raised surface for his palm and sensors placed on a flat surface in front. The CAD models of the enclosures were created in Fusion 360 as shown in Figure 19. The palm rest is a hemisphere with a diameter of 7 cm for the small size and 9 cm for the large size. Velcro was applied to the front square area of the palm rest for placing sensor pads. The bottom of the palm rest houses the PCB. The enclosures were 3D printed in PLA plastic seen in Figure 20 and Figure 21.

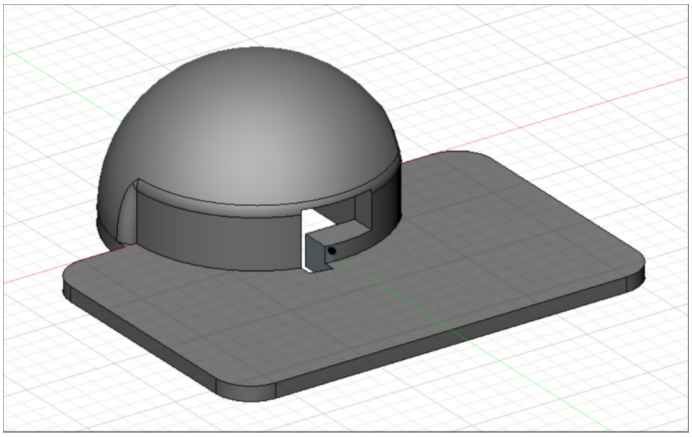

Figure 19: A 3D model of the small size palm rest for the miniaturized PCB

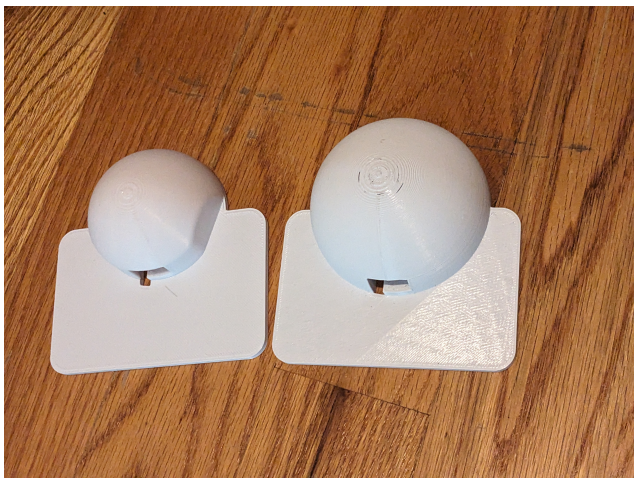

Figure 20: Palm rests for the miniaturized PCB view from the top.

Figure 21: Palm rests for the miniaturized PCB view from the bottom.

After some waiting, we were able to receive the first variation of the PCB. When we initially hooked it up to our computer we noticed that the LEDs (which indicate MIDI clicks) were flashing when pads they weren't being connected too were being touched. After looking through the schematic, we realized we used capacitors that were too small by 3 orders of magnitude: C2, C3, C16, C17, C19, C20 in Figure 16. Additionally, we realized that we did not need the series connected R1, R11, R21, C1, C15, and C18. We made these modifications to the PCB and noticed better results, but we still noticed that the sensors are extremely sensitive. Just holding your hand near the pads or the cables going to the pads would trigger them. This also mean that if the wires were near each other they would trigger each other (for example if you touched pad 1 it would trigger pad 2). To fix the sensitivity issue we could that placing a ground plane below the pads (just a piece of metal we tie to the ground of the PCB) we have much more consistent sensitivity so that it only triggers when we touch the pads. However, we knew that if we wanted to make sure the wires for each pad don't trigger each other we need zero coupling between the two different wires. This requires essentially shielding the pad inputs all the way from the pad to the input of the op-amps. As such, we were forced into redesigning the PCB and making some small changes to the schematic.

6.2 Revision 2

6.2.1 Circuit

Since the first design needed some improvements, we made a new schematic and PCB with the changes discussed in the previous section. The new schematic can be seen in Figure 22.

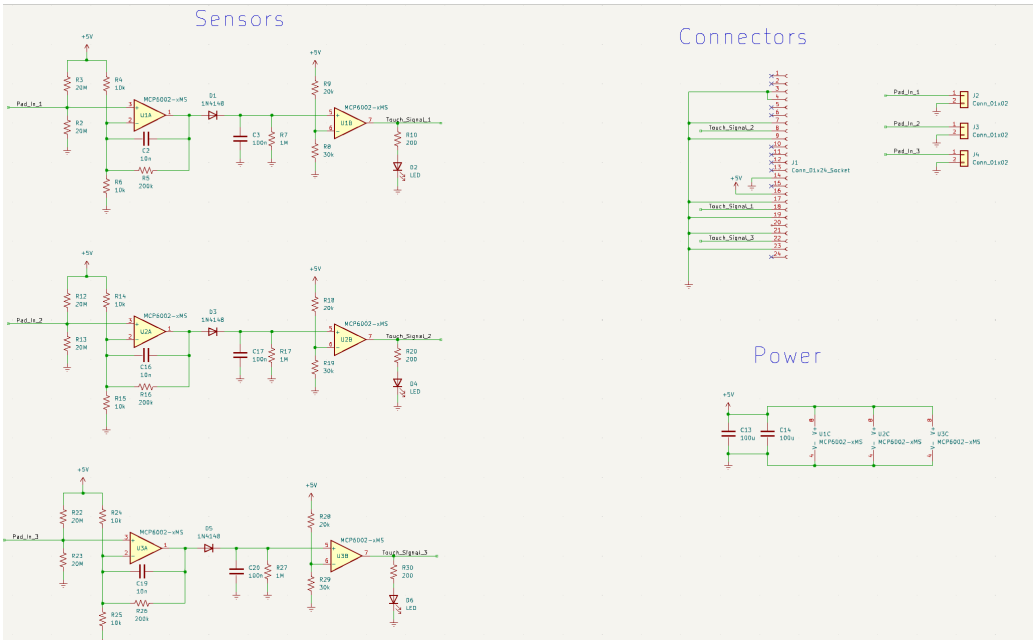

Figure 22: Modified schematic for the second revision of the PCB.

Looking at Figure 22 we can see the changes mentioned above as well as now connectors which are SMA to allow for coaxial connections to the pads. This will allow for better isolation.

Including these schematic changes, we needed to make changes to the PCB to allow for better signal isolation. The second revision of the PCB can be seen in Figure 23.

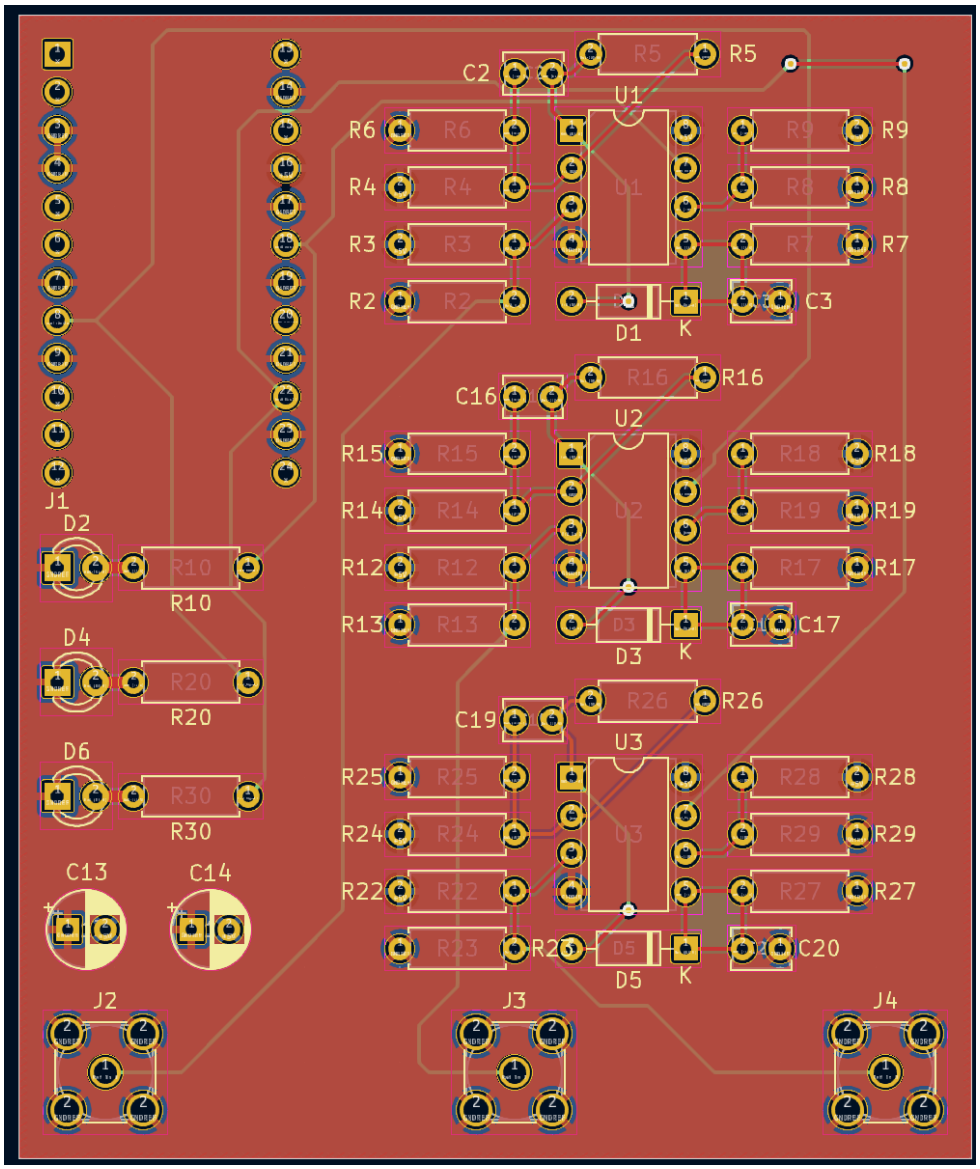

Figure 23: Second revision of the PCB.

The first main change is that we opted for a 4 layer design that is GND — Signal — GND — Power. This allows for us to have essentially a ground should around our signals. The additional layers also helped with some of the routing and to minimize traces on the top layer. Also, the board was separated circuit by circuit this time and the inputs from the pads were separated to allow for better isolation. Everything else is pretty similar to r1.

Once the parts arrived, we constructed the circuit seen in Figure 24. When we went to test this circuit, the issues caused by the miss-sizing of the capacitors had been resolved. However, we

still noticed significant issues when attempted to operated the circuit without a properly established ground plane. Additionally, we noticed that since our EMF touch sensors operate based on AC noise present in the environment, if the environment wasn't inherently "noisy" then the input to the circuit wouldn't have a high enough voltage level to trigger the output signal. We were sure of this because when the user would put their hand close to a noisy AC source, the circuit would work perfectly. The next challenge we had to face in this design process, is how to make the user/environment consistently "noisy".

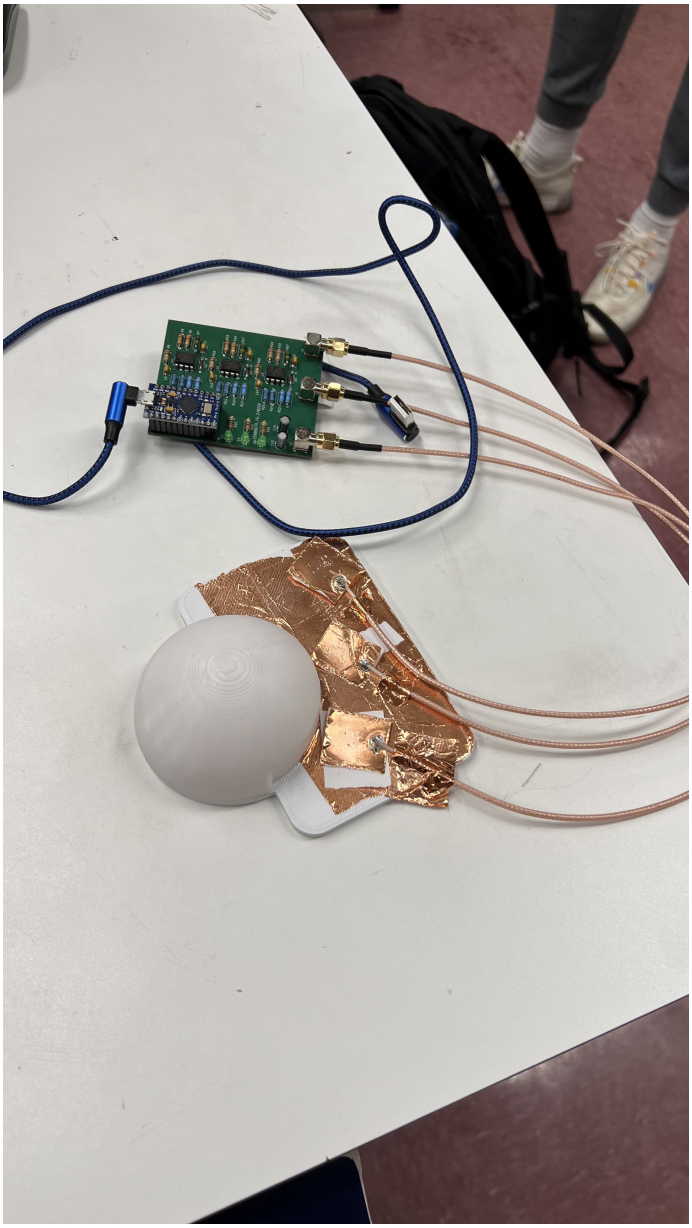

Figure 24: Constructed revision 2 circuit with touch sensors hooked up.

6.2.2 Arduino Code

The output signals are sent to three of the digital pins on the Arduino Pro Micro. We chose this Arduino because of its small form factor and its native USB support enabled by the Atmel Atmega32u4 microcontroller. We need this capability for MIDI communication between the Arduino and the computer.

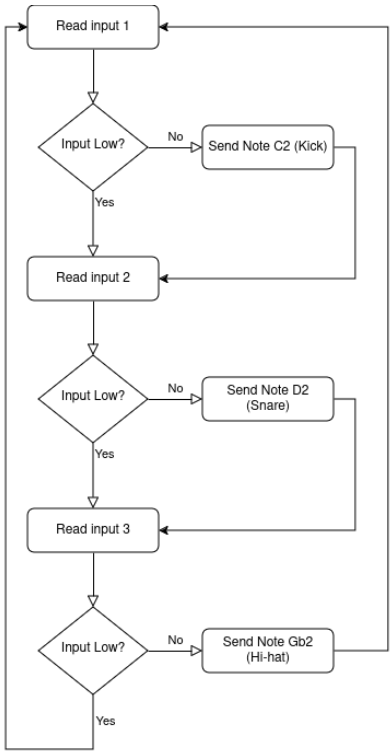

Figure 25: Arduino code flowchart

The flowchart of our Arduino code listed in Appendix A is shown in Figure 25. The Arduino code is depended on the MIDIUSB library and PitchToNote.h lookup table for sending MIDI packets. Main.ino (Appendix 2.A) first sets up all input pins and dependencies. Then it reads each with a for loop using the "readButtons" function. When a input is digital high, the code invokes the "playNotes" function which looks up the corresponding note for the input and sends the pre-defined MIDI packets through USB using the "noteOn" and "noteOff" helper functions.

7 Results and User Experience – Objective 4 & 5

7.1 Prototype Hardware Testing Performance

The sensitivity of the EMF touch sensors is influenced by the intensity of 60Hz AC noise in the surroundings. To evaluate this, we conducted tests across three distinct environments: Professor Barton's lab, which had a low level of AC noise; a residential building with a moderate AC noise level; and an ECE lab with high AC noise levels. In the residential building, the final version of our prototype showcased good performance. However, in areas with low AC noise levels, such as Professor Barton's lab, the circuit struggled to achieve sufficient sensitivity unless the user was in contact with a substantial metal object or an AC source such as the insulation of an extension cord. On the other hand, in the high-noise ECE lab environment, Channel 1 of the circuit became overly sensitive. Despite these variations in sensitivity, under optimal AC noise conditions, the circuit operated smoothly, showing no significant latency.

The palmrest feels comfortable to use and the positions of the pads can be adjusted to fit the users hand size and mobility levels. Additionally, for those who prefer a different setup, a flat velcro pad can be utilized as an alternative to the palm rest.

The complexity of the circuit is low for our prototype, which includes three EMF sensor pads. Each circuit utilizes only one dual-opamp IC and passive components. The final prototype PCB comprises 52 components, and the surface mount version of the circuit fits onto a PCB with the same dimensions as an Arduino Pro Micro. However, the complexity would increase if we were to add more sensors for future versions.

7.2 Meeting with Professor Barton

After we felt satisfied with the function of our circuit, we reached back out to Professor Barton in order to test our circuit using his self play drum set. Professor Barton's drum set is played through MIDI notes in Ableton Live being sent to servos that control drum sticks on a real drum kit. From the beginning of this project, our group had hoped to be able to allow Sam to fully enjoy the experience of playing the drums again by collaborating with Professor Barton and using our project as an input into his MIDI interface. In order to allow Sam to do this, we first needed to test

if our circuit could successfully control the drum kit. To do this, Professor Barton used program called Max8 to extract the MIDI information from our project and convert it to information that Ableton was able to use. Once this was completed, we were able to confirm that our project was able to fully control three elements of the drum kit as our project only has 3 inputs. We saw very little latency between the touch from the user and the response of the servos on the drum kit. As well as playing the drums with single inputs (ie. touching a sensor once equates to hitting the drum element once), Professor Barton showed us the potential of using the arpeggiator feature in Ableton. This feature allows for one input to be set equal to a design sequence of notes. For example, if our target user wanted to play something faster than they could tap the EMF touch sensors then they could implement the arpeggiator to play a pre-built fast sequence. Separately, we also discussed the potential to isolate songs so that the percussion portion of the songs were muted in order to allow our target user to play along with songs they like. Professor Barton said that implementing both the arpeggiator and percussion isolation was available to our group in his lab, if our target user was able to visit WPI's campus.

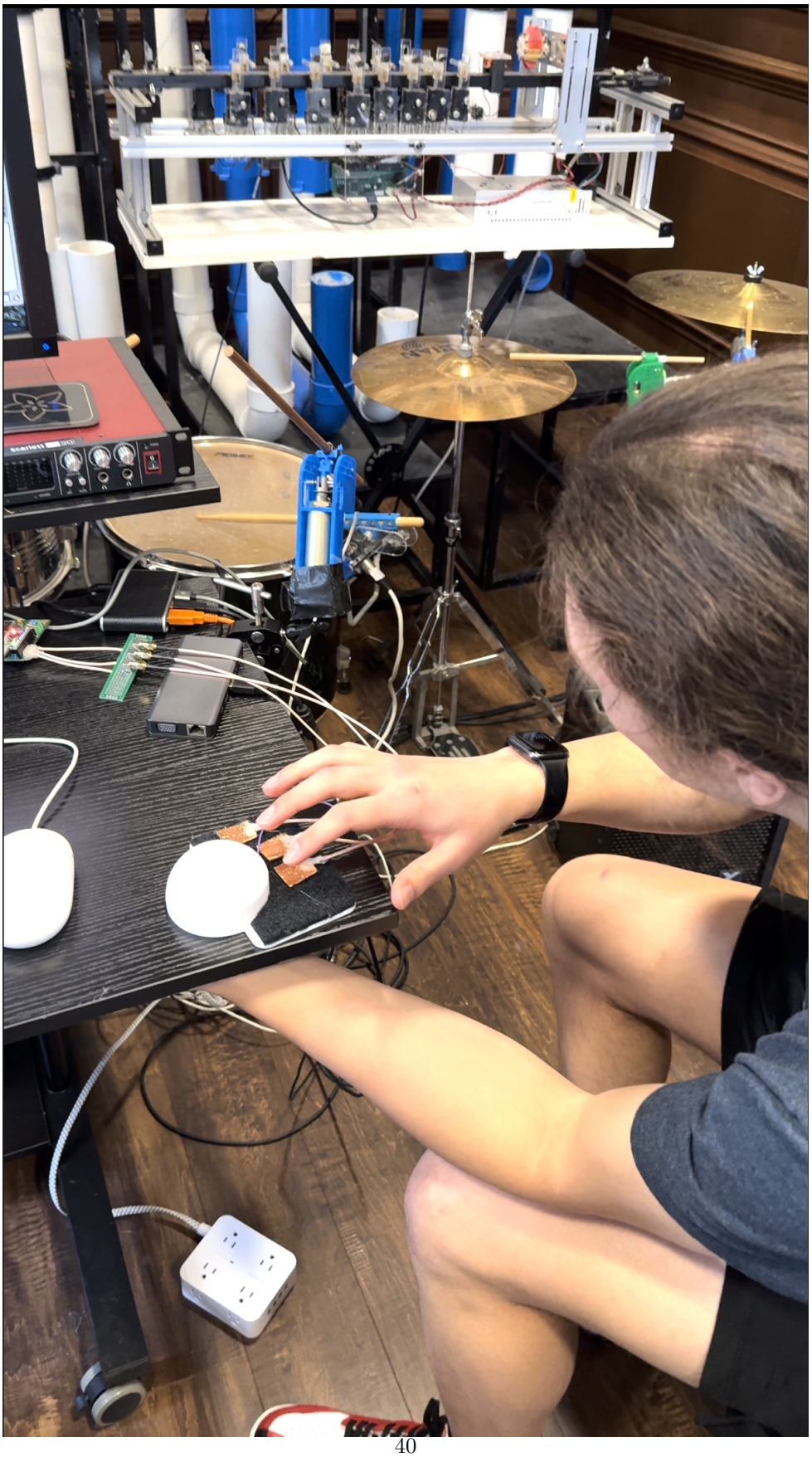

Figure 26: Josh interacting with Professor Barton's drum kit using our project as an input.

7.3 Meeting with Target User

When we finally felt confident in our final product, we traveled to our target user to determine how well our project accomplished the project goal we had established. Our target user's home was luckily perfectly noisy in terms of AC. Because of this, when our user interacted with our project, it was perfectly responsive from the perspective of the PCB circuit. Additionally, our target user noted that the delay compared to our original capacitive touch sensor approach was much better. However, the target user still noticed some delay when doing successive inputs which we determined was due to a 20ms delay that we implemented via code and forgot to change. Regarding the input interaction being comfortable for our target user, they noted that the 3D printed hand mount was much less comfortable that just the flat board. After playing for a while though, the target user noted that in the future they would prefer a more comfortable way to interact with the project that makes sure their fingers stay in the same spot. One solution to this is that we noticed our target user was able to interact with single hand gaming controllers. We could possibly 3D print a similar shape so that the touch sensors resided in the spots where his fingers would normally press the buttons on the actual controller. Overall, our target user said they thoroughly enjoyed interacting with our project and said that our project was a "game changer" for people with disabilities similar to them. There are obviously things that we can and should improve over time, but for the scope of this project, we believe we did a great job at accomplishing the overall goal. A image of our target user interacting with our project can be seen in Figure 27.

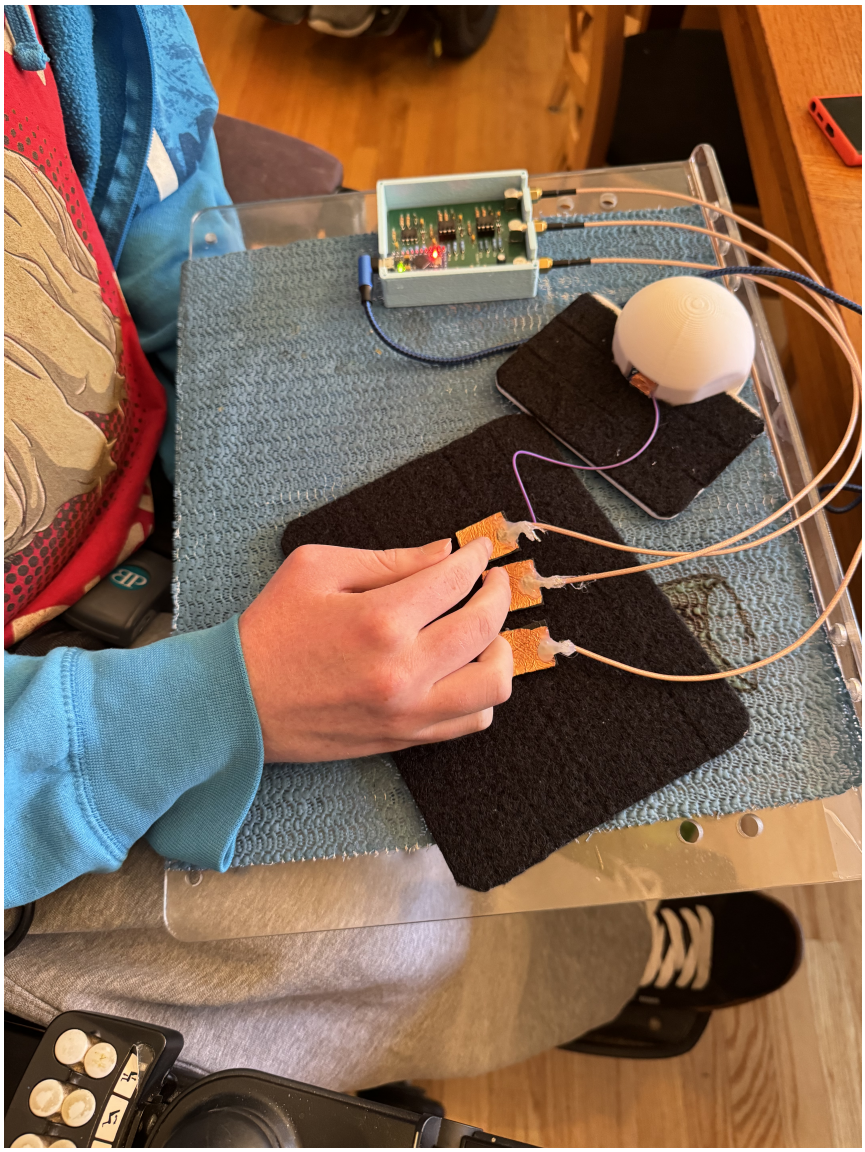

Figure 27: The target user interacting with the final prototype.

8 Conclusion and Future Work

8.1 Issues with the Design

Upon concluding our project, our group identified a central issue with the overall performance of our circuit for our target user. The main issue stems from the inputs to our circuit being dependent on AC noise in the environment. Because of this, the AC noise can't be too high or low, it must be just right for optimal user experience. For example, when testing in our campus lab which is inherently flooded with AC noise, two of the three outputs were constantly high making the circuit unusable. Similarly, when testing in a different room on campus that has little AC noise, it was extremely difficult to get the outputs to switch high. The only way to get the circuit to perform in low AC noise environment was to find something that was plugged into a wall outlet, and touch it with one hand while interacting with the circuit using the other hand. This is obviously problematic for our target used as we want our project to be usable to them in all environments. As mentioned earlier, our target user also noticed some latency when interacting with our final prototype. This is an issue but thankfully this is due to an error in our code in which we manually set a 20ms delay between each input. This can easily be changed or deleted. The final issue we noticed was that the means of interacting with the EMF touch sensors could be improved. From our target user's experience, it was clear the current means of interacting wasn't comfortable enough to use for long periods of time.

8.2 Potential Future Solutions and Improvements

While we have completed our original sought after goal, there are still some aspects of the design we would have liked to improve on if we had more time. The first of which would be addressing the issues with the circuit's performance. We believe that adding some automatic gain control could help with this issue either through potentiometers or via a micro-controller. Additionally, we could make a much larger PCB with surface mount parts, better tolerances, and less sensitive component sizes. Additionally, we could split up the board into three electrically and electromagnetically isolated sections with identical placement so it is almost like multiple separate boards in one. In making this bigger board, we could also add 7 more sensors so there is one for every finger. The user can choose which ones they do and don't want to use depending on their capabilities. Our target

user complained about issues regarding the lack of comfort when interacting with the sensors. If we were to continue the project, this would be the main focus as we want to make our project as easy to use for our user as possible. Additionally, for ease of use we could also add a track pad or similar interface to the hand mounts so they can control the DAW more easily.

8.3 Final Discussion

Overall, we accomplished the primary objective of mimicking the experience of traditional drumming while ensuring it is accessible for the target user with muscular dystrophy. We accomplished this goal through a custom designed MIDI controller with EMF sensors that detect the AC noises in the environment upon user contact. We improved the prototype in aspects such as latency, sensitivity and user interface design through a iterative design process and meetings with the target user. We concluded our project by testing our final prototype and meeting with the target user. Looking forward to the future, we have outlined aspects of the designed that could be improved. These improvements include better comfort, lower latency, more inputs, and more robust operations regardless of the amount AC noise in the room.

Appendices

A Code

A.1 Main.ino

```
// This program reads digital inputs from the front
// end circuits and converts them to note using a
// look up table "pitchToNote.h", and sends the nodes through MIDI
// Modified from examples for the MIDIUSB library:
// https://github.com/arduino-libraries/MIDIUSB

// Arduino code for A Major Qualifying Project (MQP)
// Music for Muscular Dystrophy
// Submitted to the Faculty of
// WORCESTER POLYTECHNIC INSTITUTE
// in partial fulfillment of the requirements
// for the Degree of Bachelor of Science in
// Electrical and Computer Engineering

// Authors: Joshua Hollyer, Dexuan Tang, Luke Harrington
// Project Advisor: Stephen Bitar

// Requires MIDIUSB lib
#include "MIDIUSB.h"
#include "pitchToNote.h"
#define NUM_BUTTONS  3

// Sensor pin defination
const uint8_t pad1 = A2;
```

```
const uint8_t pad2 = 5;
const uint8_t pad3 = 14;

// Notes are standard MIDI drum notes
const uint8_t pads[NUM_BUTTONS] = {pad1, pad2, pad3};
const byte notePitches[NUM_BUTTONS] = {pitchC2, pitchD2, pitchG2b};

uint8_t notesTime[NUM_BUTTONS];
uint8_t pressedButtons = 0x00;
uint8_t previousButtons = 0x00;
uint8_t intensity;

void setup() {
  for (int i = 0; i < NUM_BUTTONS; i++)
    pinMode(pads[i], INPUT);
}

void loop() {
  readButtons();
  playNotes();
  feedback();
}

// Go through buttons every 20msec and read their digital levels
void readButtons()
{
  for (int i = 0; i < NUM_BUTTONS; i++)
  {
    int VelTh1 = 0;
    int VelTh2 = 0;
    int timediff = 0;
```

```
      if ((digitalRead(pads[i]) == HIGH))
      {
        bitWrite(pressedButtons, i, 1);
        intensity = 127;
        delay(20);
      }
    // }
    else{
      bitWrite(pressedButtons, i, 0);
  }
}}

// determine which note should be played based on button readings
void playNotes()
{
  for (int i = 0; i < NUM_BUTTONS; i++)
  {
    if (bitRead(pressedButtons, i) != bitRead(previousButtons, i))
    {
      if (bitRead(pressedButtons, i))
      {
        bitWrite(previousButtons, i , 1);
        noteOn(0, notePitches[i], intensity);
        MIDIUSB.flush();
      }
      else
      {
        bitWrite(previousButtons, i , 0);
        noteOff(0, notePitches[i], 0);
        MIDIUSB.flush();
      }
```

```
    }
  }
}

// functions that sends MIDI packets
void noteOn(byte channel, byte pitch, byte velocity) {
  MIDIEventPacket_t noteOn = {0x09, 0x90 | channel, pitch, velocity};
  MIDIUSB.sendMIDI(noteOn);
}

void noteOff(byte channel, byte pitch, byte velocity) {
  MIDIEventPacket_t noteOff = {0x08, 0x80 | channel, pitch, velocity};
  MIDIUSB.sendMIDI(noteOff);
}
```

A.2 pitchToNote.h

```
#define pitchC8  108

#define pitchB7  107
#define pitchB7b 106
#define pitchA7  105
#define pitchA7b 104
#define pitchG7  103
#define pitchG7b 102
#define pitchF7  101
#define pitchE7  100
#define pitchE7b 99
#define pitchD7  98
#define pitchD7b 97
#define pitchC7  96
```

```
#define pitchB6  95
#define pitchB6b 94
#define pitchA6  93
#define pitchA6b 92
#define pitchG6  91
#define pitchG6b 90
#define pitchF6  89
#define pitchE6  88
#define pitchE6b 87
#define pitchD6  86
#define pitchD6b 85
#define pitchC6  84

#define pitchB5  83
#define pitchB5b 82
#define pitchA5  81
#define pitchA5b 80
#define pitchG5  79
#define pitchG5b 78
#define pitchF5  77
#define pitchE5  76
#define pitchE5b 75
#define pitchD5  74
#define pitchD5b 73
#define pitchC5  72

#define pitchB4  71
#define pitchB4b 70
#define pitchA4  69
#define pitchA4b 68
```

```
#define pitchG4  67
#define pitchG4b 66
#define pitchF4  65
#define pitchE4  64
#define pitchE4b 63
#define pitchD4  62
#define pitchD4b 61
#define pitchC4  60

#define pitchB3  59
#define pitchB3b 58
#define pitchA3  57
#define pitchA3b 56
#define pitchG3  55
#define pitchG3b 54
#define pitchF3  53
#define pitchE3  52
#define pitchE3b 51
#define pitchD3  50
#define pitchD3b 49
#define pitchC3  48

#define pitchB2  47
#define pitchB2b 46
#define pitchA2  45
#define pitchA2b 44
#define pitchG2  43
#define pitchG2b 42
#define pitchF2  41
#define pitchE2  40
#define pitchE2b 39
```

```
#define pitchD2  38
#define pitchD2b 37
#define pitchC2  36

#define pitchB1  35
#define pitchB1b 34
#define pitchA1  33
#define pitchA1b 32
#define pitchG1  31
#define pitchG1b 30
#define pitchF1  29
#define pitchE1  28
#define pitchE1b 27
#define pitchD1  26
#define pitchD1b 25
#define pitchC1  24

#define pitchB0  23
#define pitchB0b 22
#define pitchA0  21
/// Full 88 key range note pitch high to low MIDI byte value mapping array
const int notePitch[] = {pitchC8,
                              pitchB7, pitchB7b, pitchA7, pitchA7b, pitchG7, pitchG7b,
                              pitchF7, pitchE7, pitchE7b, pitchD7, pitchD7b, pitchC7,
                              pitchB6, pitchB6b, pitchA6, pitchA6b, pitchG6, pitchG6b,
                              pitchF6, pitchE6, pitchE6b, pitchD6, pitchD6b, pitchC6,
                              pitchB5, pitchB5b, pitchA5, pitchA5b, pitchG5, pitchG5b,
                              pitchF5, pitchE5, pitchE5b, pitchD5, pitchD5b, pitchC5,
                              pitchB4, pitchB4b, pitchA4, pitchA4b, pitchG4, pitchG4b,
                              pitchF4, pitchE4, pitchE4b, pitchD4, pitchD4b, pitchC4,
                              pitchB3, pitchB3b, pitchA3, pitchA3b, pitchG3, pitchG3b,
```

```
pitchF3, pitchE3, pitchE3b, pitchD3, pitchD3b, pitchC3,
pitchB2, pitchB2b, pitchA2, pitchA2b, pitchG2, pitchG2b,
pitchF2, pitchE2, pitchE2b, pitchD2, pitchD2b, pitchC2,
pitchB1, pitchB1b, pitchA1, pitchA1b, pitchG1, pitchG1b,
pitchF1, pitchE1, pitchE1b, pitchD1, pitchD1b, pitchC1,
pitchB0, pitchB0b, pitchA0};
```

Input materials for Appendix B

References

[1] K. T. Nozoe, R. T. Akamine, D. R. Mazzotti, D. N. Polesel, L. F. Grossklauss, S. Tufik, M. L. Andersen, and G. A. Moreira, "Phenotypic contrasts of duchenne muscular dystrophy in women: Two case reports," *Sleep Science*, vol. 9, pp. 129–133, 07 2016.

[2] "Ultraleap launches second generation of iconic hand tracking camera – leap motion controller 2 — ultraleap."

[3] M. Clinic, "Muscular dystrophy - symptoms and causes," 02 2022.

[4] Y. Zang, "Uwb motion and micro-gesture detection-applications to interactive electronic gaming and remote sensing," 05 2016.

[5] P. S. Santhalingam, A. A. Hosain, D. Zhang, P. Pathak, H. Rangwala, and R. Kushalnagar, "mmasl," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, pp. 1–30, 03 2020.

[6] "Leap motion controller 2 – ultraleap."

[7] A. Ganguly, G. Rashidi, and K. Mombaur, "Comparison of the performance of the leap motion controllertm with a standard marker-based motion capture system," *Sensors*, vol. 21, p. 1750, 03 2021.

[8] S. S. Mechael, Y. Wu, Y.-T. Chen, and T. B. Carmichael, "Ready-to-wear strain sensing gloves for human motion sensing," vol. 24, pp. 102525–102525, 05 2021.

[9] "88.99usd 30 percent off, open source somatosensory gloves, wearable mechanical gloves, exoskeleton somatosensory control, robot control - aliexpress."

[10] "How to build a pair of $15,000 sensor gloves for 40$ — hunter scott," 01 2015.

[11] "Cyberglove iii."

[12] "Mimu gloves."

[13] M. Clinic, "Electromyography (emg) - mayo clinic," 2018.

[14] J. H. Medicine, "Electromyography (emg)," 2023.

[15] "Emg detecting neuromuscular abnormalities - brigham and women's hospital."

[16] "Myoelectricity - an overview — sciencedirect topics."

[17] "Uptodate."

[18] "Ece4760 final project: Emg signal controlled game."

[19] j. o. m. website!, "Make muscle midi music!."

[20] "The midi muscle machine."

[21] "About us — drake music."

[22] "Touchkeys – andrew mcpherson."

[23] "Capacitivesensor - arduino reference."