S0undfingers Design Document

Team - Thomas Driscoll, August Gress, Kyle Patel
ECE 445 Design Document - Spring 2020
Group 8
TA: Dhruv Mathur

1 Introduction

1.1 Objective

Not everyone knows how to play an instrument. And if they do, they might not know how to do it well and be able to stay in key. But they would like to be able to make some music. People also might not want to disturb their neighbors or have the space in their house to store instruments, but still want to engage in making music. They also might have physical impairments that hinder them from being able to hold them.

Our solution is a Bluetooth-enabled glove that has five force sensitive resistors at the end of each of the five finger holes. When you would press the pad of your finger while within the glove against a hard surface, it would play a programmed tone through the speaker on your mobile phone. It would do this by using an internal app that outputs to audio device drivers provided by Android. The tone that would play would be programmable from the app, on a per-finger basis.

1.2 Background

Playing and interacting with music is a pastime that many enjoy, but due to financial, space, or other constraints, can be hard to realize. Coupled with the difficulty of learning a new instrument, there is just a lot of overhead. Many people, however, do make the tradeoffs to get to play and enjoy making music in their homes [1]. This usually leads to it negatively affecting the people around them however, as neighbors or others in the home have to hear the instrument regardless of whether they were interested in hearing it or not [2].

We saw through these issues that there should exist a way for someone to start playing along with their favorite songs, no matter which instrument in the piece they wanted to emulate. We also wanted to make sure that there was a way that the instrument noise could be controlled, so we designed our glove to play its tones using a phone's speaker or anything plugged into the phone's headphone jack.

The design that we propose is different from the original solution in two ways.

First, we do not use any kind of "mode." The original project used different modes to specify how different hand motions (like left to right or finger-bending movements) would affect the notes that would be played, such as in their "piano mode" where moving left to right would simulate playing down and up the piano respectively. We do not use movement to control what notes are played, instead it is based on which finger(s) is(are) currently pressing against a hard surface while within the glove. Further, the note that is played when this happens is completely programmable and not tied to a specific movement or finger.

In the original solution, there was also this concept of note "production", where the notes produced would always be in the same key so the music produced would be harmonic. We have nothing of the sort, and instead leave it to the discretion of the end user to decide what they would like to have play while they are using the glove.

There is one other competitor product on the market in the form of the MINI.MU Glove Kit, which is essentially a DIY motion driven glove for music production [3]. One of the biggest problems with

it however, is that it is hard to accurately track hand movements, as hands can have very fine motor control movements that are difficult for a motion sensor to detect. Our solution improves upon this by being completely force driven, meaning that it's very definite to know when you have made a sound (as you have physically pressed your finger against a sensor).

1.3 High Level Requirements

- Able to recognize finger taps within a pressure-sensitive bluetooth-enabled glove and turn those taps into signals based on which finger is being pressed.
- Able to send those signals from that glove via bluetooth to play a given sound from a mobile phone via an app.
- The latency between a finger tap and sound outputting through the phone is at most Bluetooth protocol latency (200ms) + 50ms for our processing $\pm$100ms (Total: 250ms $\pm$ 100ms).

1.4 Visual Aid

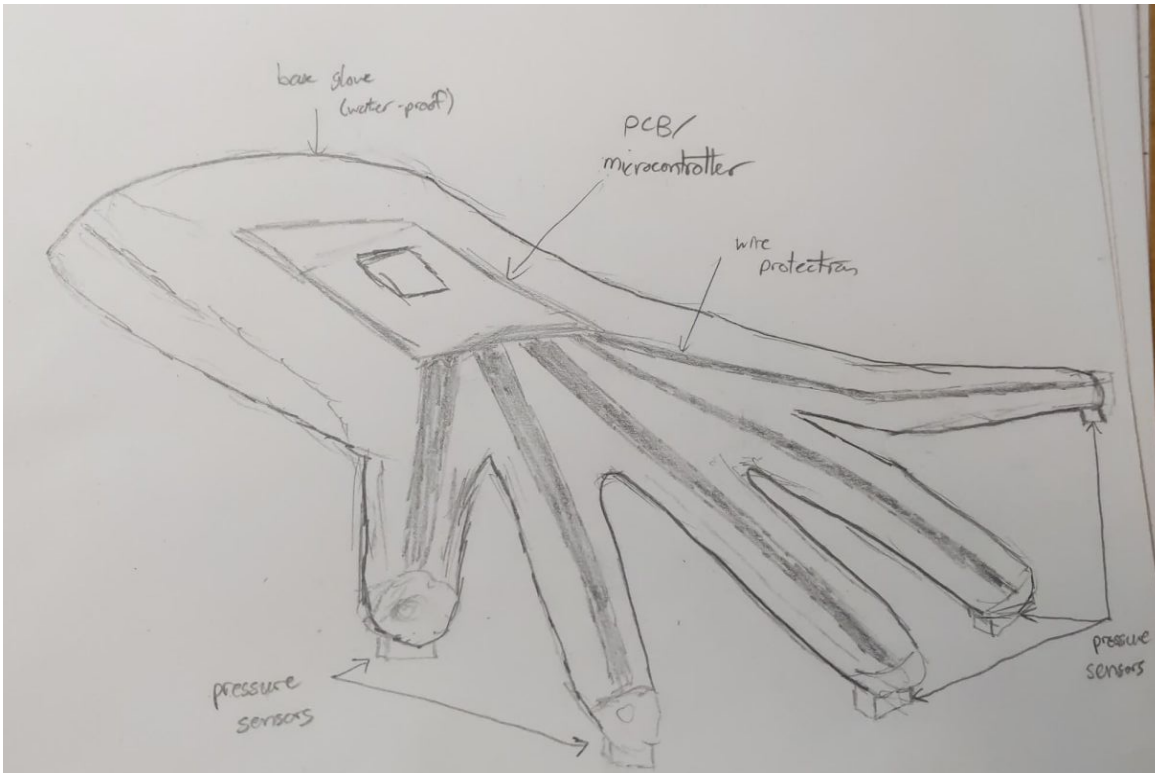

Fig. 1 Visual Aid

The above visual aid (Fig. 1) is an attempt by the designers to show what a probable prototype of our glove would look like. The microcontroller/PCB sits on the back of the palm and any/all wirings are covered with waterproof casings. Pressure sensitive plates are attached to the ends of each finger, to give the user as much freedom as possible in expressing themselves musically.

2 Design

2.1 Block Diagram

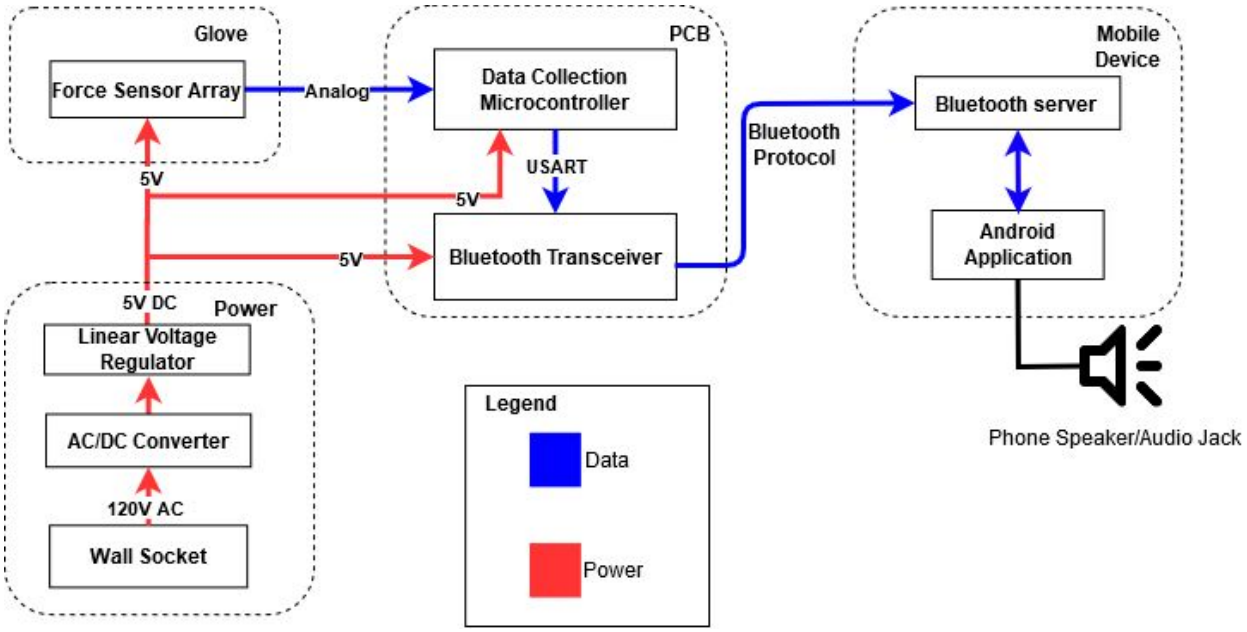

Fig. 2: Block Diagram

2.1.1 Glove

The glove would have a sensor array consisting of 5 FSR402 force-sensitive variable resistors that connect directly to the analog I/O ports on the data collection microcontroller. These ports connect to the data collection microcontroller's 8-channel, 10-bit ADC. The applied force range of these sensors is about 0.1-10N (10g-1kg) [4]. For reference, the touchweight of a piano key is made to be about 50g (or 0.5N) [5]. We would need to find appropriate pressure thresholds for finger presses based on the actual voltage readings that are outputted from the ADC in order to register finger presses like piano keys. Like any variable resistor, a measurement resistor will need to be a part of its circuit to create a voltage divider circuit. We would also have a soft membrane between the finger and the sensor for comfort, and to transfer the pressure from the user's finger to the pressure sensor. Each sensor would have 5V supply voltages.

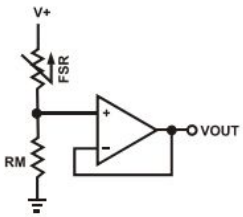

Fig. 3: Example configuration of a force-sensitive resistor (op-amp optional)

2.1.2 PCB

We would use an ATMega328P microcontroller to control the glove and collect sensor data, and an HC-05 Bluetooth module to communicate between the glove and transmit data. We would power the glove using a 5V wall adapter. The code running on the microcontroller would be written using the Arduino IDE, and the PCB substrate would be FR-4 using 2 layers.

2.1.3 Mobile/Real-Time App

Our app will be completely stored and run on a user's phone (a Google Pixel 3a for our demo), serving three purposes.
The first would be to connect via Bluetooth to the gloves themselves and receive digital input from them. The second purpose would be to process in real-time the input received from the gloves and output that to an audio jack. Finally, the app would allow the selection of customizable sounds. The user would select the instrument desired and a corresponding sound from that instrument for each finger. The overarching framework for our project will run on Corona SDK. Due to the need to access the underlying operating system and device drivers, we plan to use the Android MediaPlayer API to guarantee native Android development (our app will not be accessible using Apple systems for our demo). To initialize and interact with the Bluetooth transceiver, the app, on startup, will also utilize the Android native Bluetooth API, which will configure a majority of our settings and allow us to customize the Bluetooth connection for our needs. In order to test our design, we will also require a mock Bluetooth device. To accomplish this, as Bluetooth is just a protocol, we will be mocking our Bluetooth device using Wireshark [6], an open-source library used to test Bluetooth protocols. Further, we will be using the general purpose testing suite JUnit, as Android is Java-native.

2.1.4 Power

This glove will be powered by a barrel jack connection to a wall socket, using a Sparkfun TOL-15312 wall adapter device, which acts as an AC/DC converter and outputs 5V DC. On our PCB will be a barrel jack connector that outputs to a linear regulator on the PCB attached to the glove. The linear regulator would be a Texas Instruments LP2985-N device, which has an input voltage range of 2.5-16V [7], within our required 5V.

2.1.5 Glove Schematic

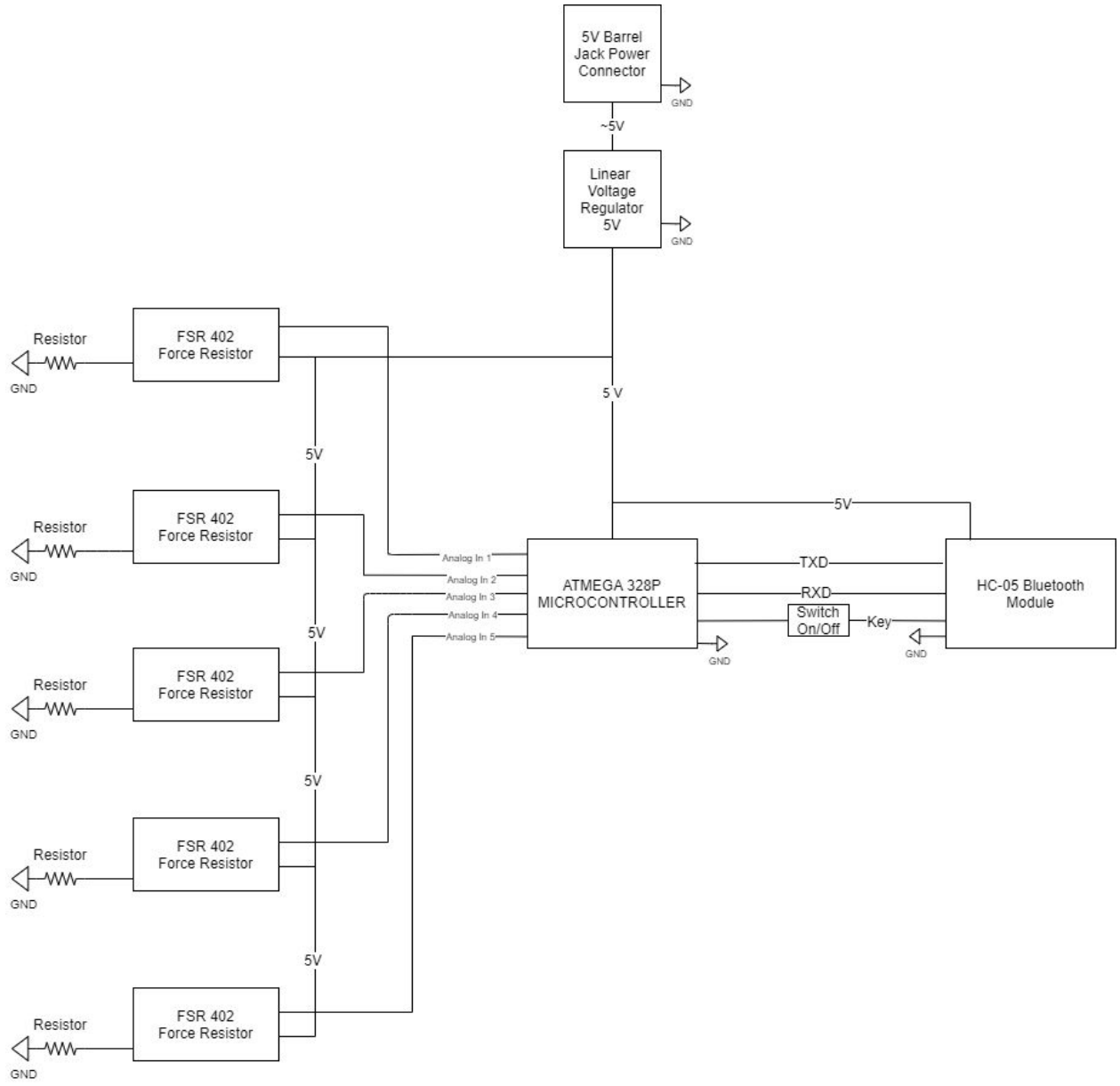

Fig. 4 Glove Schematic

The side of the schematic (Fig. 4) to the left of the ATmega328P MICROCONTROLLER is the array of FSR 402 force resistors that will be placed at the tip of each finger in the glove. The change in resistance when you press your finger against the sensor will then be reported to the microcontroller in the middle, which processes the input to be sent to your mobile phone via the HC-05 Bluetooth Module on the right. The barrel jack connector on top will be how all of these devices receive the 5V power they require.

2.1.5 Glove

Module	*Requirements*	*Verification*
Glove: FSR 402 Force Sensitive Resistors	1. Each force sensitive resistor can register a keypress 95 ± 5% of the time.	1.) a.) Supply 5V to microcontroller and glove. b.) Press each force sensitive resistor with a force of ~0.5N± 2% c.) Repeat Part B 20 times for each resistor. d.) Confirm in the Arduino IDE serial terminal that the press is registered 95% of the time, on average.

2.1.6 PCB

Module	*Requirements*	*Verification*
PCB: Microprocessor (ATMega328P)	1. Must be able to facilitate sequential collection of quantized FSR 402 data. 2. Must be able to send digital sensor readings to Bluetooth Microchip via USART.	1.) a.) Open a serial monitor that can output whether a sensor was pressed or not. b.) Press down on each FSR 402 resistor in a sequence of 5 presses without repeating fingers. c.) Confirm that the sequence of presses in the terminal matches the real sequence of presses. 2.) a.) Select the COM port that corresponds to the output of the Bluetooth module. b.) Confirm that sensor readings (can be raw ADC values) can be read in the Arduino's serial monitor.
PCB: Bluetooth Microchip (HC-05)	1. Has a discoverable Bluetooth profile on the mobile device. 2. Must be able to maintain connection with a bluetooth-connected phone with over a period of 5 minutes while in motion.	1.) a.) On a mobile device, open Bluetooth settings and select a Bluetooth device to pair with. b.) Confirm that the address of the Bluetooth microchip appears in the pairing list. 2. a.) Pair the Bluetooth module with the mobile device. b.) Confirm that randomly moving the device within 2 feet of the device does not compromise the Bluetooth connection between the phone and device.

2.1.7 User Phone

Module	*Requirements*	*Verification*
Android application: Corona SDK	1. Users should be able to view supported instrument sounds. 2. Users should be able to pair a particular sound to a particular finger. 3. Front-end should have an icon showing the current status of Bluetooth pairing with gloves. 4. >80% unit test coverage for every React component.	1.a) Unit test dropdown menu so that each string has an associated value (Test is boolean) 1.b) Unit test each option becomes a global variable for use upon selection. (Test is boolean) 2.a) Software integration test where each finger and pairing combo is tested. (Test is boolean) 2.b) At demo, show that test works in reality 3) Simulate a Bluetooth device (see below) on a laptop or PC as part of the testing package. Test that this mock device works with the system.
Bluetooth Server[6]	1. Service discovery process successfully caches the security key for bluetooth operation. 2. Process only accepts recognized Bluetooth profiles. 3. When paired, connection lasts until it is outside of range (30m ± 5m). 4. When paired, <1% data loss. 5. Data stream is received server-side and manipulatable.	ALL REQUIREMENTS: Create a simulated Bluetooth device with PC (i.e. mock protocol) for testing purpose 1.a) Cache security key from mock Bluetooth (unit test cache space) 1.b) Verify security key matches mock device (unit tests for successful verification, unit tests for unsuccessful verification) 1.c) Test connection (unit tests for successful pairing, unit tests for unsuccessful pairing) 2.a) Add mock Bluetooth device to whitelist and verify connection 2.b) Change device security key and attempt to connection; verify connection is impossible (test is boolean) 3.a) With the phone, generate a bluetooth signal. 3.b) Signal must be received from 30 ± 5m 4.a) Using mock Bluetooth device, use ping functionality to monitor data loss 4.b) Verify data loss <1% 5) Simple unit test that data stream is received from Bluetooth module

Module	*Requirements*	*Verification*
Power: 5V Texas Instruments LP2985-N Linear Regulator	1. Provides 5 ± 0.05V to other project components (regulator provides 1% accuracy in output voltage). [7]	1. Using a DMM, measure the voltage difference across all hardware devices to ensure that the supplied voltage is within appropriate range for 30 minutes.

2.1.8 Other Design Considerations

Something else that we had considered was doing some sort of synthesis when a finger was pressed within the glove instead of playing a sample. While possible, we felt that giving the user to play any sample of any instrument instead of just the ones we have programmed synthesis algorithms for was a better solution to the problem.

We also considered playing sounds directly from the glove as an option instead of having them play through the phone, but we thought that adding a speaker array to the glove would significantly add to the bulk of it. We also thought that having it play through the phone would allow for greater flexibility on the user's behalf.

In regards to software, we considered several frameworks for the project, including Unity, Koitlin, and Sencha Touch [8]. While each of these frameworks have their merits, we chose to use Corona SDK as it listed as the best cross-platform solution (in case we wanted to scale our product later) to stream audio. Our specific solution also took advantage of the underlying Android MediaPlayer API but there were other audio handlers available. Our choice of the MediaPlayer API was focused on ease of integration, as the support was native and our team had experience with the API on previous projects.

2.2 Physical Design

Fig. 5 is what we plan for our physical representation of the glove to look like from a top down view. Essentially, we want to have pressure-sensitive resistors in the fingers to capture the finger press input from the user, and a box on the top of the hand that will contain all the necessary wiring and controllers.

We modelled the size of the hand and fingers based on historic averages [9]. We also took care to ensure that the size of our pressure-sensitive resistors would be small enough to fit within those averages.

Fig. 6 is another view from the side, where you can see how high the box on the top of the hand would be. We wanted to strike a balance between having it be large enough that we could fit all the electronics we needed but not so large that it would impede a user's ability to tap effectively.

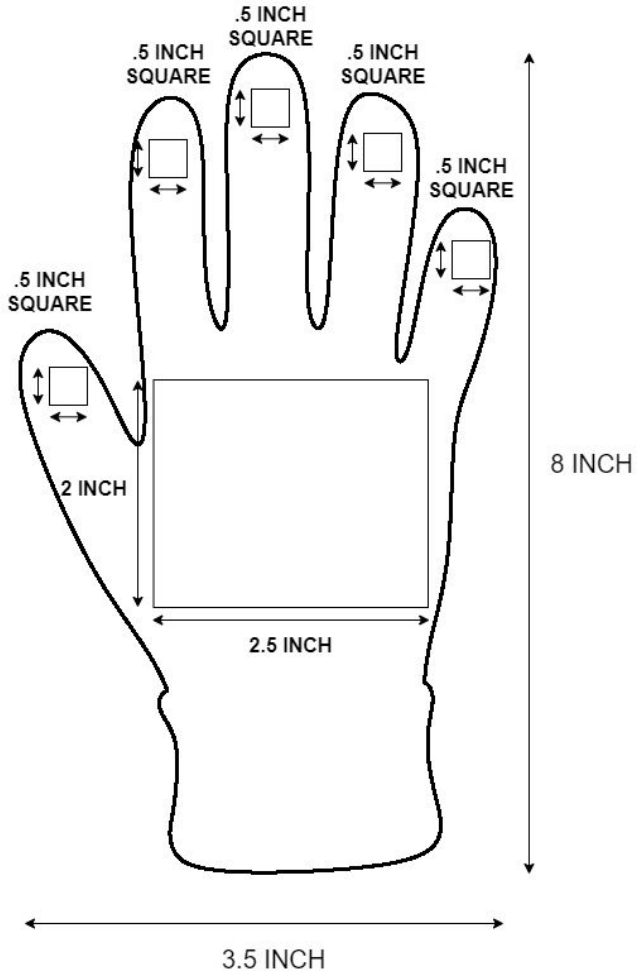

Fig. 5 Physical Diagram (Top Down View)

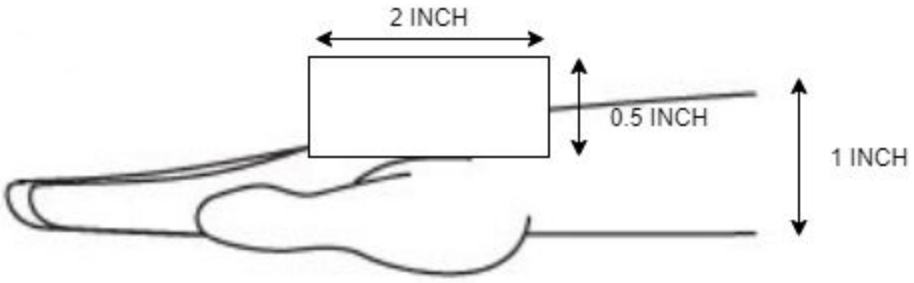

Fig. 6 Physical Diagram (Side View)

2.3 Tolerance Analysis

The primary tolerances to be considered are the latency of data transfer and the repeatability of force readings for the FSR402. We also want the sensors not to be overloaded with current in the event that they are pressed down with too much force, which in our case would be 100N or 10kg of force.

According to the FSR Integration Guide from Interlink [10], the current limit for the FSR402 is 1mA/cm^2 or 1.267mA for the surface area of the FSR402. If the input voltage to an FSR is 5V, then the total resistance of measuring voltage divider circuit must be greater than 3.946kΩ at any given point. The resistance of the FSR is approximately 250Ω at 100N, so the measurement resistor needs to be at least 3.7kΩ to prevent accidental overload, most likely greater than 4kΩ in reality.

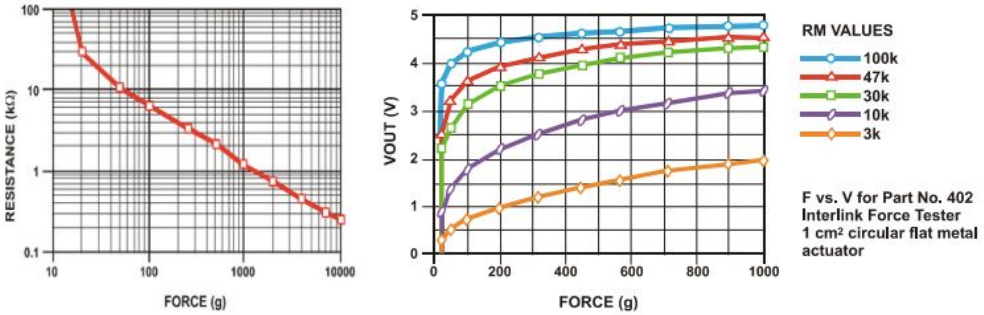

Fig. 7: FSR 402 V_{out} vs. Force for Various R_m Values & Force vs. Resistance Curves [10]

When designing our solution to optimize for latency, we chose the size of the data stream to be 1KB no matter the number of fingers that were being pressed. This was so that when we were sending information about which fingers were being pressed, there would be less variables that could affect the transfer latency of $250 \pm 100ms$.

3 Comparison to Previous Design

3.1 Differences

The original solution was an IMU-based glove that would detect hand movement and turn that into input to their software. They used this input to create different modes like “piano mode,” in which they could detect horizontal left and right movement to simulate hands moving up and down keys. Their design also had 5 bending resistors along the inside of each finger that could further be used as input. In their case, they used this to create a “Guitar Mode,” where they could detect this bending and simulate the bending of strings on a guitar. They also could detect combinations of different finger bends and use that to create specific guitar chords. All of the audio synthesis would be accomplished with a separate audio processing subsystem with the sound directly outputting to a speaker from that subsystem.

In our solution we also have a glove, but we have 5 pressure-sensitive resistors at the tip of each finger in the glove. When pressed, a sound would be played on a mobile phone (or through any audio device connected through the phone’s audio jack) connected to the glove via Bluetooth. The sound that

plays is pre-sampled and fully programmable via a mobile app on the phone, allowing the user to mimic any instrument's sounds.

There are 2 key differences between our two solutions. First, our design does not require any expensive processing or synthesis. This allows us to send output to the phone potentially much quicker than the original design, as we don't have as much processing overhead. Second, we have a drastically different set of input we accept from the user. Where the original design would take into account motion and bending movements using synthesis techniques to transform that into predefined useful output, we streamline this by only having pressure sensors at the fingertips and allowing the user to manually define what pre-sampled sounds they would like to play.

A trade-off between our solutions is that our user can use 5 tones at a time, even if those tones can be in any order and can be any instrument playing any note or sound. A user of our solution doesn't have the entire range of tones immediately at the disposal of the user of the original group's solution.

3.2 Analysis

3.2.1 Software

A major component of our solution will be running on a mobile device (assume a Google Pixel 3a) and we were planning on using the Android MediaPlayer API to deal with device drivers. The original group's corresponding component had a program called SuperCollider running on a PC to handle audio input streams. Our solution has several obvious improvements. Using a mobile device enhances usability for a variety of environments and we use a native Android API. This has direct access to the device drivers, minimizing our audio latency. In general, comparisons between user space (SuperCollider) and kernel space (the device drivers accessed by MediaPlayer API) are very situation dependent and labeling a quantitative value to them is not something most people do. However, it is generally accepted knowledge that the code executed in kernel space is several times faster than its user-space counterpart.

3.2.2 Bluetooth vs. WiFi

In the original solution, the previous group proposed using a Wifi module that ran at a minimum of 115 kbps. Our solution using Bluetooth protocols instead, enabling much faster data transfer. While the exact speed varies based on the distance and exact nature of the server and client, Bluetooth gives a maximum speed of 3 Mbps [11]. Further, we are sending much less data than the original group. The only information we care to send are boolean values of "Pressed" or "Not Pressed" via our bluetooth connection for each finger. The original group necessitated tracking the spatial position of each finger and the glove as a whole, a much more complex task . We estimate that our data packet content would never exceed 10 bytes:

(size(boolean) * 10) = (1 byte * 10) = 10 bytes

Based on storing several dozens of floating point values, 10 flex sensors and probably additional information, we estimate that their data packets would be between around 600 bytes at minimum:

(size(float) * (3D coordinates) *(estimated unique coordinates) = (4 * 3 * 50) = 600 bytes

Therefore, with faster speed and less data to be transfered, the risk of latency is much lower in our design and the overall performance for the user is improved.

3.2.3 Flex Resistor vs. Force Resistor

Something that we considered in our design also was the accuracy of the main interface unit, the force-sensitive resistor at the tips of the fingers. The flex resistor that was used in the previous design to control some potential inputs, the TSP-L-0012-103-3%-RH, has worse accuracy in terms of expected measured resistance at specific bending angles when compared to our FSR 402 force resistor. Their flex resistor had an accuracy of around $\pm 20\%$ per part [12] where our force resistor has a part to part accuracy of $\pm 6\%$ or $\pm 2\%$ per part [10]. This means that the pressing force requirement for the resistor will be much more consistent across all fingers vs. the flex resistor's bending requirement. Our solution will improve the user experience as it means that the user will feel more reassured that when they press their finger inside the glove the correct sound will play.

4 Cost and Schedule

4.1 Manpower Cost

The average salary of a 2017-2018 ECE Illinois Computer Engineering Grad (as our group is composed of) was $92,430 [13]. Working 52, 40 hour weeks (for a total of 2,080 hours a year), this breaks down to $44.43 per hour. This is what we will assume for our fixed hourly working rate. We also are going to assume we would have the normal amount of time usually available at this point of the semester available to us again, so a total of 10 weeks to work at ~10 hours per week. Once again, this equation will neglect any time we would be spending with ad agencies marketing our product, or any other outside work to get it to market. Therefore, our cost for manpower will be:

$$2.5 \times Number\ of\ Group\ Members \times Fixed\ Hourly\ Cost \times \#\ of\ Hours\ per\ Week \times \#\ of\ Weeks = Cost$$
$$2.5 \times 3 \times \$44.43 \times 10 \times 10 = \$33,322.50$$

As shown above, we calculate the manpower cost to be $33,322.50 for this prototype.

4.2 Part Cost

Below we have a list of all of the parts required to make our board, broken down into bulk and prototype pricing.

Part	***Cost (bulk)***	***Cost (prototype)***
FSR 402	$4.0299 * 5=$20.15	$7.576 * 5=$37.88
ATMega328P	$3.16 * 1 = $3.16	$4.30 * 1 = $4.30
HC-05	$6.15 * 1 = $6.15	$10.52 * 1 = $10.52
Sparkfun TOL-15312	$5.95 * 1 = $5.95	$5.95 * 1 = $5.95
LP2985-50DBVR	$0.50 * 1 = $0.50	$0.50
Corona SDK	$0.00	$0.00
Android MediaPlayer API	$0.00	$0.00
Wireshark	$0.00	$0.00
JUnit	$0.00	$0.00
Final Part Cost	*$35.91*	*$58.65*

4.3 Shop Cost

We estimate the shop hourly rate to be $30 per hour. We have a somewhat complex design, as we must design a safe layer between the user's finger and the force resistor to prevent any harm (while still being pliable enough that the force from finger presses will still be read). We also must affix a box to the top of the glove to house the electronics and make it rigid enough to withstand repeated vibrations and movement. Due to all this, we estimate our total shop hours to be 50-60 hours. We will take the high side of this estimate for our shop cost.

$$Total\ Hours \times Fixed\ Hourly\ Cost = 60\ x\ \$30 = \$1800$$

4.4 Total Cost

We calculate the total cost including manpower, shop labor, and parts to be $35,151.96 accounting for bulk parts and $35,187.87 for the prototype.

4.5 Schedule of Work

Week...	***Kyle Patel***	***August Gress***	***Thomas Driscoll***
Week 1	Finish Design Doc, prepare slides for design review specifically for block diagram	Finish design doc, Prepare slides for design review (specifically the schematic and descriptions of parts)	Finish Design Doc
Week 2	Do design and participate in peer reviews	Do design review and design peer review	Set up initial framework for user-facing portion of app
Week 3	Implement/debug microcontroller code for collection of sensor data.	Begin doing research on what we refinements we need to make to our design doc	Set up mock Bluetooth device. Test physical connection using Arduino Uno
Week 4	Complete additional debugging of sensor data collection by this time	Make those changes to the design doc to make the final report easier	Write unit tests for API, finish mock Bluetooth testing
Week 5	Research any other parts we would need to design a PCB	Research any other parts we would need to design a PCB	Continue front-end dev, finish unit tests
Week 6	Assist with any issues in mock PCB design, ensure design completion	Make a mock PCB for the project to make sure we had all parts we needed	Work with August on board/mobile app integration
Week 7	Start thinking about formatting for the final report, reach out to ECE Editorial Services	Start thinking about formatting for the final report, reach out to ECE Editorial Services	Write unit tests for front-end dev
Week 8	Test speed of connection and debug any latency issues	Ask any followup questions with course staff before beginning final design paper	Test speed of connection and debug any latency issues
Week 9	Work on preliminary tasks for final design	Work on final design paperwork	Final tweaks for server/API/front-end

	paperwork		
Week 10	Work on final design paperwork	Clean up any remaining tasks on the final paper, check formatting	Work on final design paperwork
Week 11	Turn in final design paperwork	Turn in final design paperwork	Turn in final design paperwork and do final presentation

5 Ethics & Safety

The ethical or safety issues with our project pertain to the physical gloves themselves, the microcontroller and Wi-Fi chips.

Citing the IEEE Code of Ethics #1 [14], we will work to ensure that the construction of our gloves is structurally sound such that a user will not be concerned with electrical hazards such as exposed wires or static shock, or any harm from burning ICs or plastic. Further, a likely source of potential harm would be liquids spilling on the glove, ruining the circuity and causing an electrical hazard to form near the user's hands. To prevent this, all circuits in our glove will have a protective layer on the top of them that prevents any spillage into the sensitive electronics underneath.

An additional source of safety concern is the user-facing application, specifically in regards to the ACM Code of Ethics 2.9 [15]. While we expect the user of our prototype to load the application from source code provided by the designers, bad actors could potentially hijack the Bluetooth connection in the app itself to download malware onto a user's phone [16]. These concerns, while valid, are an extremely low risk as our application will not be downloaded outside of the authors knowledge for the duration of the project. Further, we will be whitelisting the gloves such that the app will reject any interaction that is not associated with that Bluetooth identifier (BD_ADDR).

Finally, regarding regulatory standards, since we are creating a receive-only device, we are exempt from type approval [17]. If we were to take the product to market, we would need to test at an accredited testing house, followed by an application for Part 15 certification [17]. However, since we are not, we do not need to address those issues at this time. Our understanding is that this would be a relatively simple process that would require time and money to pay for accreditation, neither of which are available to us.

References

[1] Rock, L. and Rock, L., 2020. *Music Lessons Doncaster, Melbourne | Guitar, Singing & Piano - Invest In Learning Guitar: How Much Does It All Cost?*. [online] Music Lessons Doncaster, Melbourne | Guitar, Singing & Piano. Available at: <https://www.leadersofrock.com/posts/2015/7/27/invest-in-learning-guitar-how-much-does-it-all-cost> [Accessed 3 April 2020].

[2] Kay, L., 2020. *Bombed By Bass: The Neighbours' Music Turned My Flat Into A Hellhole | Laura Kay*. [online] the Guardian. Available at: <https://www.theguardian.com/commentisfree/2014/dec/05/noisy-neighbours-drug-taking-my-flat-hell hole> [Accessed 3 April 2020].

[3] Shop.pimoroni.com. 2020. *MINI.MU Glove Kit – Pimoroni Store*. [online] Available at: <https://shop.pimoroni.com/products/mini-mu-glove-kit?variant=21240542560339> [Accessed 16 April 2020].

[4] Interlink Electronics, "FSR 400 Series Round Force Sensing Resistor," FSR 402 Data Sheet, Oct. 26, 2010. Available: https://cdn.sparkfun.com/assets/8/a/1/2/0/2010-10-26-DataSheet-FSR402-Layout2.pdf

[5] K. E. Lile, "Piano Finders," *Piano Finders: Touchweight*. [Online]. Available: https://www.pianofinders.com/educational/touchweight.htm. [Accessed: 16-Apr-2020].

[6] "Bluetooth," *Bluetooth - The Wireshark Wiki*. [Online]. Available: https://wiki.wireshark.org/Bluetooth. [Accessed: 16-Apr-2020].

[7] "LP2985-N Micropower 150-mA Low-Noise Ultra-Low-Dropout Regulator in a SOT-23 Package Designed for Use With Very Low ESR Output Capacitors," http://www.ti.com/lit/ds/symlink/lp2985-n.pdf, Dec-2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/lp2985-n.pdf. [Accessed: 23-Apr-2020].

[8] "14 Best Android Frameworks for App Development in 2019," *Intellectsoft Blog*, 01-Apr-2020. [Online]. Available: https://www.intellectsoft.net/blog/best-android-frameworks/. [Accessed: 16-Apr-2020].

[9] Healthline. (2020). *Average Hand Size: For Adults, Children, Athletes, And More*. [online] Available at: <https://www.healthline.com/health/average-hand-size#adults> [Accessed 3 April 2020].

[10] "Interlink FSR Integration Guide, PDF." Interlink Electronics, Westlake Village, CA. Available at: https://cdn.sparkfun.com/assets/4/d/0/f/7/DS-9375-Force_Sensitive_Resistor_0.5in.pdf [Accessed 16-Apr-2020]

[11] A. Iwaya, “Is Bluetooth Faster than Wi-Fi?,” *How-To Geek*, 21-Jun-2014. [Online]. Available: https://www.howtogeek.com/191546/is-bluetooth-faster-than-wi-fi/. [Accessed: 16-Apr-2020].

[12] “TSP Series ThinPot Datasheet,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/datasheets/spectra-symbol/spectra-symbol-tsp-series-thinpot. [Accessed: 17-Apr-2020].

[13] “All Campus Undergrad 2017-2018 Report,” *Engineering Career Services*, 31-Dec-2018. [Online]. Available: https://ecs.engineering.illinois.edu/files/2019/03/IlliniSuccess_AnnualReport_2017-2018_FINAL.pdf. [Accessed: 25-Feb-2020].

[14] ieee.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 3 April 2020].

[15] Acm.org. (2020). *The Code affirms an obligation of computing professionals to use their skills for the benefit of society.*. [online] Available at: https://www.acm.org/code-of-ethics [Accessed 3 April 2020].

[16] Chicago Tribune. (2020). *Chicago Tribune - Widespread Android Virus Could Hide in Popular App Updates*. [online] Available at: https://www.chicagotribune.com/news/ct-xpm-2012-04-16-sns-201204161154usnewsusnwr20120413041 3androidapr16-story.html [Accessed 3 April 2020].

[17] Blueradios.com. (2020). [online] Available at: <https://www.blueradios.com/Bluetooth_Global_Certification_Requirements.pdf> [Accessed 3 April 2020].