IMUsic2

By

Colin Flavin

Nick Russo

Helen Swearingen

Final Report for ECE 445, Senior Design, Spring 2020

TA: Eric Liang

08 May 2020

Project No. 33

Abstract

In this era of technology, music should be easily accessible for all to play. The previous solution to this problem, in fall of 2019, was to create a dance-oriented instrument using wearables on the wrists and ankles to create sounds. Our solution works at a smaller scope and looks to create a software instrument which tracks the gestures and movements of a user's hands with an off-the-shelf dual IR sensor set.

Table of Contents

1. Second Project Motivation

1.1 Problem Statement

Emotional expression has been portrayed in a variety of mediums, including writing, painting, sculpture, or theater. However, perhaps the two most ancient and universal are those of music and movement. Traditionally, music is dictated by trained musicians, whose interaction with instruments requires dexterity and coordination, as well as the interaction with a physical instrument. This forces composition and choreography to be separated. The goal of this project is to bridge this gap between composition and choreography via a wireless instrument using IR sensors to track a person's motion.

1.2 Solution

We propose an instrument that combines music and movement into one intuitive instrument that allows for expression both through the musical and the choreographic medium. Our instrument will heavily utilize the LeapMotion Controller as a way to track hand positions in three dimensions to make an instrument controlled by hand gestures and positions. The gestures and hand positions will be sent via a USB connection to a laptop, where a note-picking algorithm will take in the data and wirelessly transmit the sound to be played to a mixing board, where we will have master volume controls and routing to a speaker system. The instrument will function similarly to a simplified theremin, with the user's left hand controlling the note's pitch and volume with the x and y position relative to the motion controller. The right hand will be used to control the timbre of the sound by performing different hand gestures.

1.3 High Level Requirements

1. Must be able to process gestures, mix sounds accordingly, and play them back with a delay of 100ms or less between performed gesture and the played sound.
2. The output frequency range should be able to play at least two octaves of notes with a dynamic range of 40-70dB.
3. Must recognize 13 separate hand gestures corresponding to each semitone and a rest, as well as measure within 5% tolerance the x-y position of the user's hand from the motion controller.

1.4 Visual Aids

1.4.1 Physical System

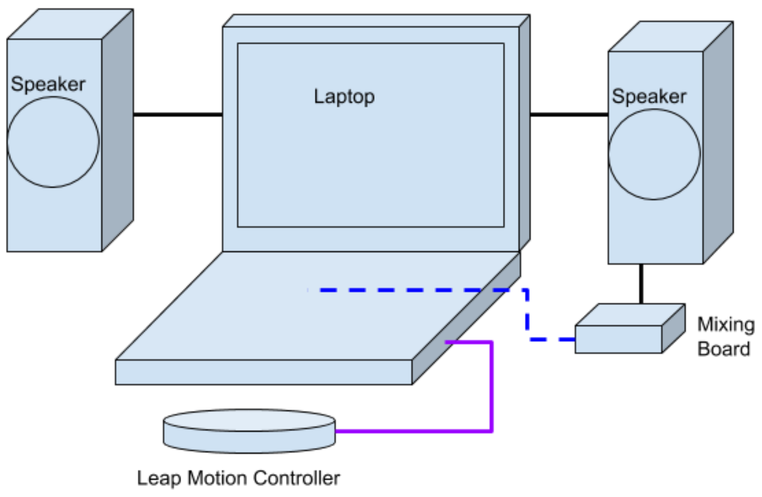


Figure 1 Physical Diagram

Our physical system consists of four distinct parts: a laptop, which hosts the gesture and sound processing software, the mixing board, where the sound is generated, speakers to play the sound, and the Leap Motion controller to sense the positions of the player's hands.

1.4.2 Block Diagram

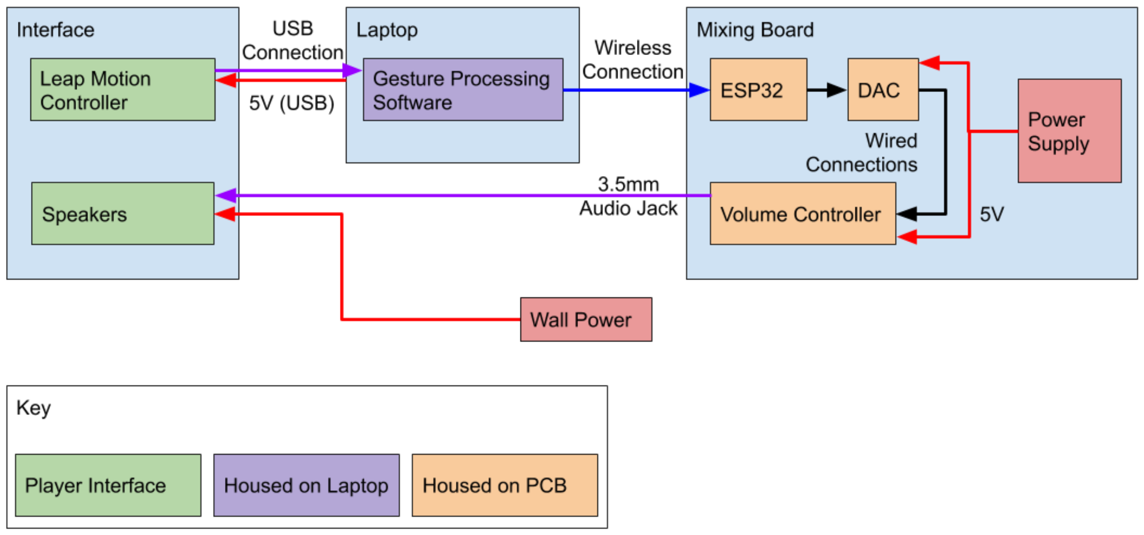


Figure 2 Block Diagram

The block diagram and physical design reflect each other. The Leap Motion controller communicates with and is powered by a USB connection to the laptop. The mixing board is powered by a commercial power supply and contains the DAC and master volume controller, and connects to the laptop via WiFi and the speakers via 3.5mm audio jack. The speakers are powered from a standard wall outlet.

2. Implementation

2.1 Implementation Details + Analysis

Given the time frame and social distancing conditions, we have a fair amount of the project largely untouched. That being said, we have more complete schematics for the project, as well as some software implementation and flow diagrams. In addition, we will explore the latency of the system by dissecting each part, and the delay each causes.

In this system, there are several causes of delay: gesture sensing by the Leap Motion controller, gesture mapping software, audio signal creation by SuperCollider, WiFi communication between the laptop and the ESP32, and the D/A conversion. Both the gesture mapping system and the SuperCollider code will be dictated by the software we implement, so no absolute latency can be determined. However, both would be utilizing table lookups, which can have a runtime of O(1) if a map container is utilized [1], so we will assume their latency to be minimal.

The Leap Motion Controller has one of the highest latencies of our system components. Although no absolute latency can be found in their datasheets, they do claim their system latency is always >30ms [2]. This also takes into account the graphics they create, and they claim this to be the bulk of their latency. As a baseline, we will use this 30ms value, but testing is needed to find the average latency without displaying hand motions on the laptop.

The WiFi communication system has the next highest latency of our project components. This is bounded by the processing time on the router, which for WiFi has an average of around ~2-4ms for a typical environment without very many users [3]. Again, we will take the maximum value as a worst case scenario, so this adds 4ms to our total system latency.

Finally, the D/A converter also has some latency. Its datasheet advertises it as a 384kHz D/A converter [4], so it should have a latency of around 2.6µs. Putting this all together, we can see that the total latency of our system has a maximum of around 34ms. This is much less than our system requirement of 100ms, so our project would definitely be successful on that front.

30ms (Leap Motion Controller latency) + 4ms (WiFi latency) + 0.0026ms (DAC latency) = 34.0026ms

2.2 Schematics

Although due to social distancing constraints, we cannot meet or use the lab to create any hardware devices, we can create schematics for our circuit used in our music generation module. The following section delves into the separate subcircuits used, as well as their purpose in the module.

2.2.1 Power Schematic

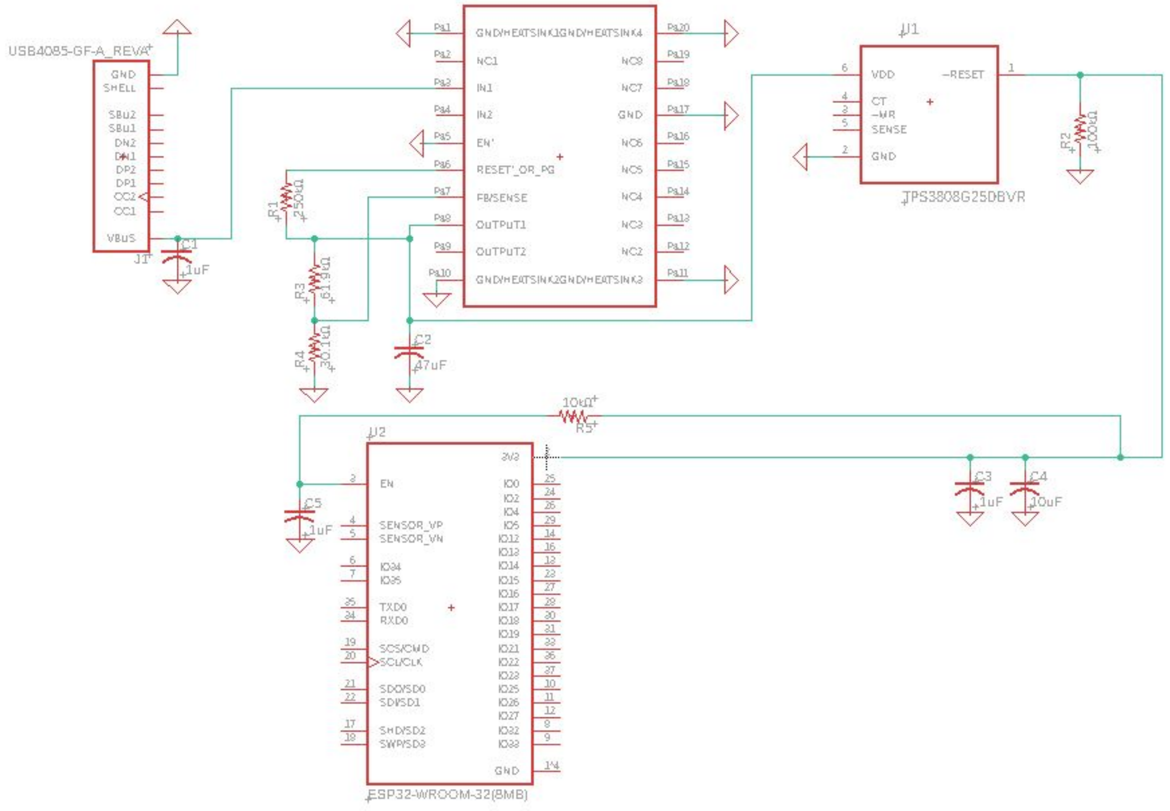


Figure 3 Power Schematic

The Power module consists of a USB-C port that the battery pack plugs into. The power is sent to a voltage regulator/voltage supervisor combination to ensure that a steady supply of power is sent to the ESP32. The regulator has a voltage divider step-down as well that takes in the 5V from the battery and passes 3.3V for the ESP32.

2.2.2 ESP Schematic

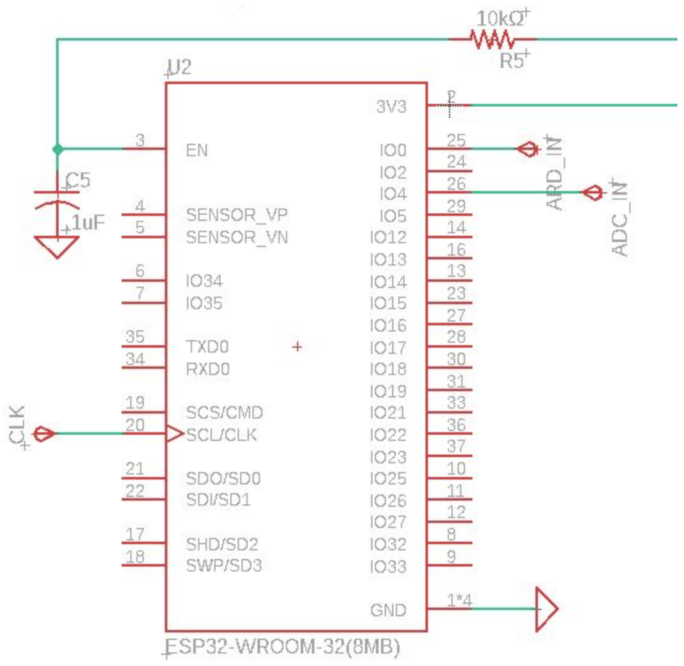


Figure 4 ESP Schematic

The ESP takes in power to the EN pin to power it and takes a CLK signal at pin 20. IO0/2 (pins 24/25) are outputs to the ADC.

2.2.3 DAC schematic

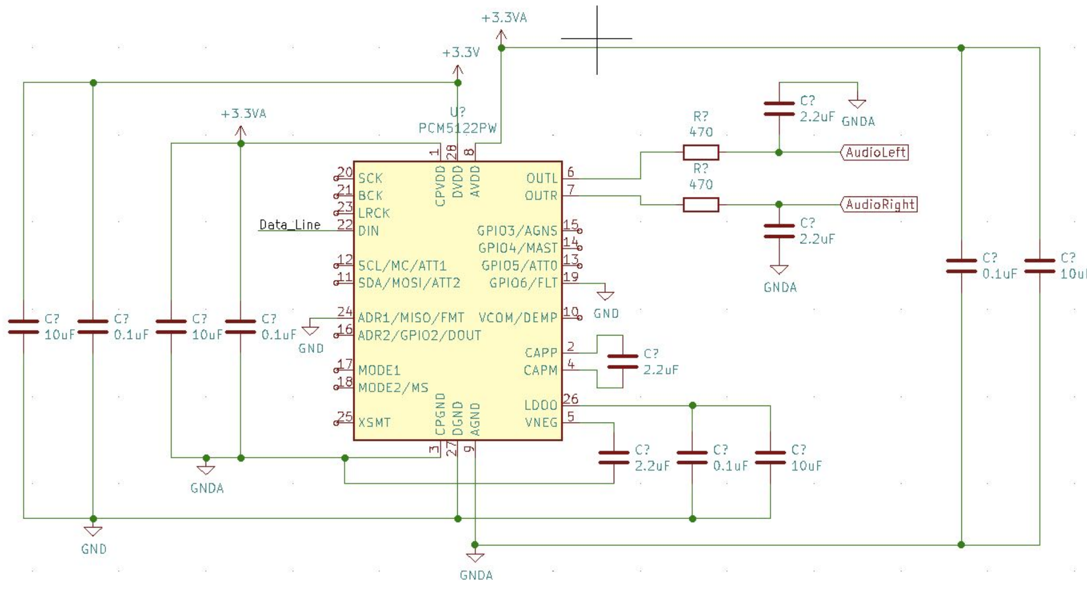


Figure 5 DAC Schematic

For our D/A converter, we chose an audio grade D/A converter, with 106dB capabilities for a 16 bit digital signal, and a rate of 384 kHz. With these specifications, we are able to achieve latency of <100ms, which would satisfy our high level requirements. It uses I^{2}C specifications for audio communication, so we would need to be sure that is the data stream sent to the ESP from SuperCollider.

2.2.4 Volume Controller Schematic

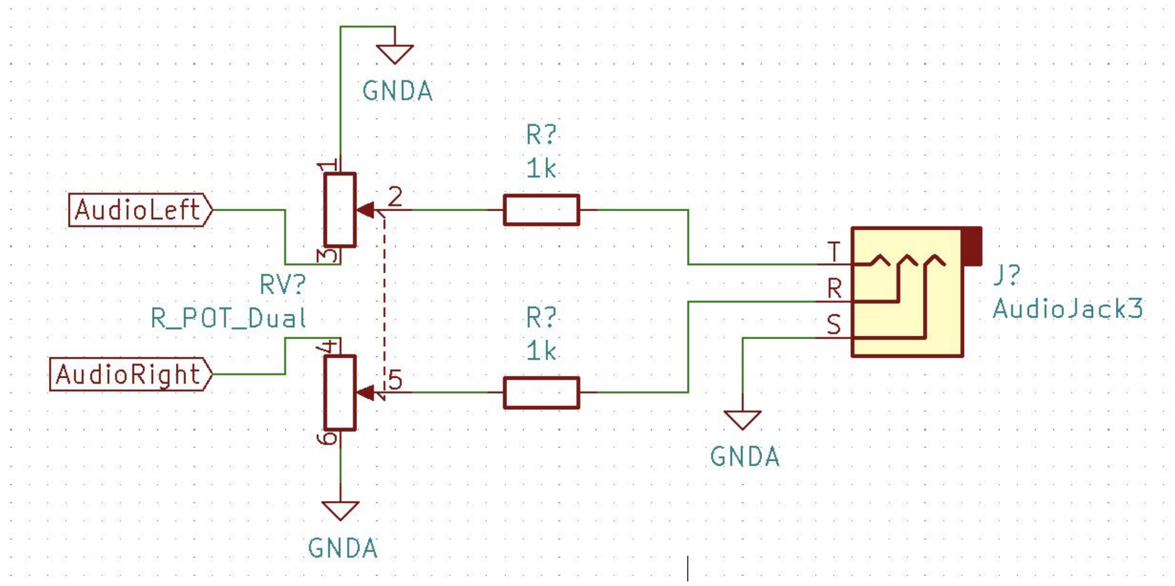


Figure 6 Volume Controller Schematic

The volume controller is somewhat crude, it only contains a slide dual potentiometer to control the resistance of the left and right channels of the analog signal for the right and left channels. The variable resistance will be able to control the master volume of the total signal. The load resistors would be used to tune the system in accordance of a 70dB maximum volume.

2.3 Software

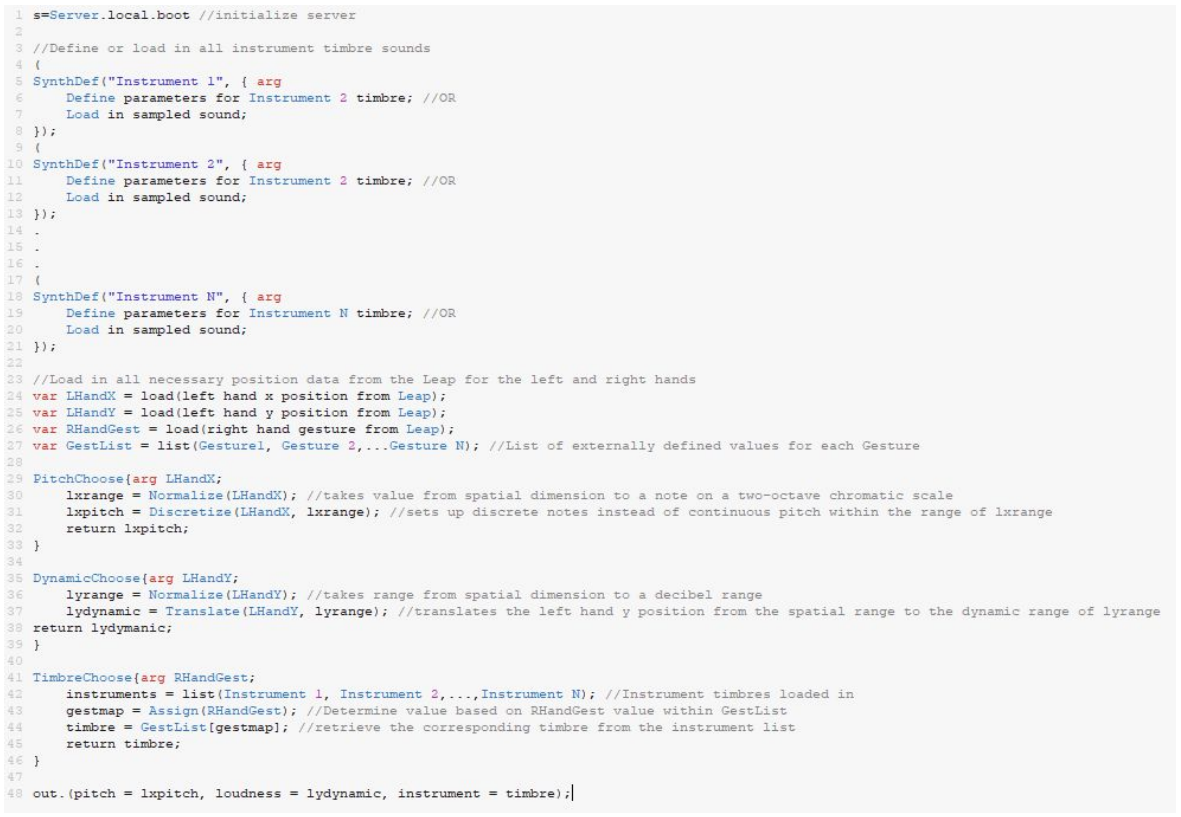
Software is a major portion of this project. Gestures must be identified by a Leap motion controller API, then sent to SuperCollider to be converted to a note with pitch, timbre, and dynamics, and then this information must be sent via WiFi to an ESP32 to be converted into an analog signal for our speaker to use.

2.3.1 Gesture Mapping

We take data corresponding to hand positions from Leap Motion's proprietary software. Information for each hand includes values for the location in space of the palm and its normal vector, the average angle of the fingers, and the position of each fingertip [5]. The aggregation of these values will be compared to the ranges of values set for each distinct gesture, then labelled as one of those gestures. The gesture number or hand position and relevant hand designation is then sent to the SuperCollider portion of the software.

2.3.2 SuperCollider

SuperCollider is a musical programming language developed for working with sound. It allows for real-time synthesis of sound and integration of external variables to be used to control various parameters of the sounds you're working with. For our purposes, we will be using SuperCollider to synthesize our output sound timbre, control dynamics, and discretize and pick the proper pitch.



```
s=Server.local.boot //initialize server

//Define or load in all instrument timbre sounds
(
SynthDef("Instrument 1", { arg
    Define parameters for Instrument 2 timbre; //OR
    Load in sampled sound;
});
(
SynthDef("Instrument 2", { arg
    Define parameters for Instrument 2 timbre; //OR
    Load in sampled sound;
});
.
.
.
(
SynthDef("Instrument N", { arg
    Define parameters for Instrument N timbre; //OR
    Load in sampled sound;
});

//Load in all necessary position data from the Leap for the left and right hands
var LHandX = load(left hand x position from Leap);
var LHandY = load(left hand y position from Leap);
var RHandGest = load(right hand gesture from Leap);
var GestList = list(Gesture1, Gesture 2,...Gesture N); //List of externally defined values for each Gesture

PitchChoose{arg LHandX;
    lxrange = Normalize(LHandX); //takes value from spatial dimension to a note on a two-octave chromatic scale
    lxpitch = Discretize(LHandX, lxrange); //sets up discrete notes instead of continuous pitch within the range of lxrange
    return lxpitch;
}

DynamicChoose{arg LHandY;
    lyrange = Normalize(LHandY); //takes range from spatial dimension to a decibel range
    lydynamic = Translate(LHandY, lyrange); //translates the left hand y position from the spatial range to the dynamic range of lyrange
return lydymanic;
}

TimbreChoose{arg RHandGest;
    instruments = list(Instrument 1, Instrument 2,...,Instrument N); //Instrument timbres loaded in
    gestmap = Assign(RHandGest); //Determine value based on RHandGest value within GestList
    timbre = GestList[gestmap]; //retrieve the corresponding timbre from the instrument list
    return timbre;
}

out.(pitch = lxpitch, loudness = lydynamic, instrument = timbre);
```

Figure 7 SuperCollider Pseudocode

The SuperCollider code takes in left hand X and Y positions (*LHandX* and *LHandY*) and the right hand Gesture (*RHandGest*, value obtained from processing in another program). It initializes a list of Synths (various sounds either constructed in SuperCollider or sampled sound files loaded in from outside, titled *instruments*) and maps each possible gesture (corresponding to a certain Synth) to a list (*GestList*). **PitchChoose** then takes *LHandX*, does a transformation on the spatial data obtained from the Leap, and maps it to a discretized list from 0 to 24. These numbers correspond to the 25 pitches of a two-octave chromatic scale, and correspond to ~10mm distances, or about 1cm per note. **PitchChoose** discretizes the value from the change of range to one of the 25 chromatic pitches and returns the value as *lxpitch.* **DynamicChoose** does the exact same thing for *LHandY*, translating the spatial data to a value between 40 and 70, corresponding to the decibel intensity of our desired output sound and returning it as *lydymamic.* However, *lydynamic* is a continuous value and can take on any value between 40 and 70. Lastly, **TimbreChoose** takes in *RHandGest,* mapping it to one of the values in *GestList.* The value is then used as an index for *Instruments,* choosing and returning the Synth to be used as *timbre.* Finally, the output sound is compiled using the returned using the obtained *lxpitch, lydynamic,* and *timbre* specifications to be output to the sound mixing board.

2.3.3 ESP32 WiFi Communication

For our final project, we chose to utilize the ESP32's WiFi capabilities. Our original choice was to use Bluetooth, but after some considerations, the WiFi has lower latency (>2ms) than Bluetooth (with a latency of ~200ms) which allows our system to maintain a latency >100ms, to eliminate lags in the sound for the user, as well as satisfy our high level requirements. Currently, to connect to the WiFi, each time the user reconnects, they must reconnect the ESP32 to WiFi in order to establish a common wireless connection.

Up to this point, we only have a block diagram for implementing this software portion. If given more time, we would implement this completely, as well as write an interface for connecting the ESP32 more seamlessly.

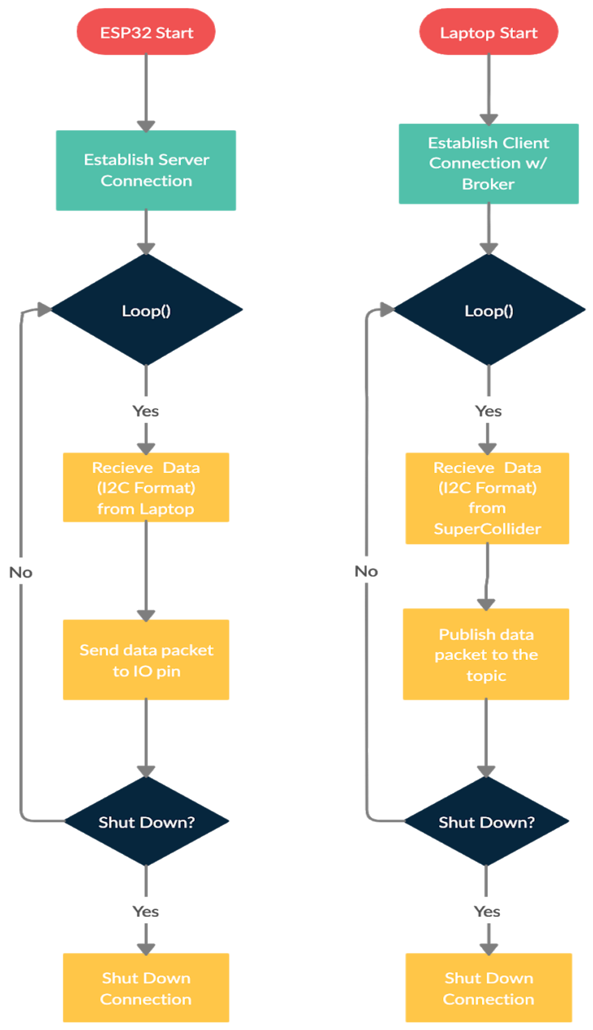

ESP32 Start
Establish Server Connection
Loop()
Yes
Recieve Data (I2C Format) from Laptop
Send data packet to IO pin
Shut Down?
No
Yes
Shut Down Connection
Laptop Start
Establish Client Connection w/ Broker
Loop()
Yes
Recieve Data (I2C Format) from SuperCollider
Publish data packet to the topic
Shut Down?
No
Yes
Shut Down Connection

Figure 8   WiFi Communication Flowchart

2.4 Costs

Table 1: Parts Costs

Part	**Manufacturer**	**Source**	**Cost ($)**
LeapMotion Controller	UltraLeap	UltraLeap	89.95
ESP32 Wifi Module	Espressif	DigKey	3.75
Anker PowerCore Battery	Anker	Amazon	25.99
PCM5100APWR Audio DAC	TI	DigiKey	2.47
PS45G Dual Slide Potentiometer	TT Electronics	DigiKey	4.45
Logitech Z150 Multimedia Speakers	Logitech	Amazon	24.99
		Total	**151.60**

Table 2: Labor Costs

Team Member	**Hourly Wage**	**Weekly Hours**	**Number of Weeks**	**Cost Per Member**
Colin Flavin	$40/Hour	10	10	$4,000
Nick Russo	$40/Hour	10	10	$4,000
Helen Swearingen	$40/Hour	10	10	$4,000
Total				$12,000

In total, our parts and labor costs come to $12,150.60.

3. Second Project Conclusion

With the given circumstances, we were only able to complete a small fraction of this project. Distance between group members made meeting difficult, and components that required a higher level of teamwork could not be completed. Lack of parts, equipment, and lab access prevented any completion of hardware components, as well as their testing. Software could not be completed and debugged without the hardware or correct environments to do so, as well as the time constraints of this project. However, we were still able to complete some major aspects of this project. In this section, we will discuss what we implemented, what is still uncertain and needs to be tested, the ethics and safety of our project, and ways to improve our project given the time and space to do so.

3.1 Implementation Summary

In this project, many things needed to be completed. As a team, we got through much of it. The hardware design was largely completed. In addition, software for the SuperCollider portion was created, flow diagrams were created for the WiFi communication system, and software implementation was started with this portion, although no code was presentable enough to be placed in this document. Mathematical analysis was also done to demonstrate the latency of our system would be within our tolerance.

Colin Flavin handled the design for the schematics for the ESP32 and the Power modules as well as the code for note picking in SuperCollider.
Nick Russo was responsible for the mathematical analysis, and the schematics for the master volume control and DAC.
Helen Swearingen handled gesture mapping and interface with the Leap Motion controller, and communication system flow.

3.2 Unknowns, Uncertainties, Testing Needed

Some portions of the project were not able to be completed. The hardware setup could not be completed without the soldering equipment, so that remained unfinished. The PCB was not completed because of time constraints. Because we do not have access to any hardware components, we are also not able to test many components of our design. Without this hardware, we cannot test the D/A conversion, the wireless communication system, or the latency of our system.

We can test the D/A conversion after the hardware setup by sending a digital signal block into our D/A converter, and then with an oscilloscope testing the analog signal. We would go through and test the frequency of the device, the amount of bits it can handle at a time, and check that the signal is never outside of 5% of the expected signal, as created by a simulation. We would test the wireless communication system by sending a dummy data block from the laptop to the ESP32, and checking that the output of the specified IO pin matches the dummy data block. Finally, we can test the latency of our sound generation module by timing how long different pitches from SuperCollider take to reach the ESP32 using a ping, and then timing how long the D/A converter takes to complete a data packet with an oscilloscope.

In addition, some items could not be completed due to cost restraints. The Leap motion controller costs around $90, which is out of our personal budget. Thus, we cannot test any code for gesture mapping. If we could test this, we would test that the correct pitch is selected when a given gesture from the right hand is performed. We would also test a variety of timbres and dynamic combinations with the left hand as well. Since these would be discretized table lookups, we would hope that it could correctly identify a gesture 95% of the time. Since the Leap motion controller is an optical sensor, it can lose track of fingers if they are obscured, such as when the palm is pointed up. We would have tested our initial list of gestures to make sure they were clear and distinct, and modified them as necessary to make sure they were detectable and intuitive.

With our current project design, each discrete pitch will only have ~1cm of space in the interaction box. This seems somewhat small, but with how much motion could be done within a performed piece, it might not matter. However, without being able to physically test the qualitative “feel” of the instrument, it is unknown whether or not this is not enough space.

3.3 Ethics and Safety

IEEE code of ethics states that our project should “improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies” [6]. Our project satisfies this as it promotes the bilinearity of the formation of music and dance. In addition, the IEEE code of ethics also states we should seek, offer and accept criticism for our technical work [6]. By participating in this course, going to weekly meetings with course staff, and presenting our ideas to our peers, we are actively following this code, while seeking to improve the project idea with the intent of creating the best possible end result. In addition to ethical concerns, our project also has some safety concerns.

When dealing with audio, one main safety concern is dealing with the listener's hearing. If unbounded, high volumes of noise can cause prolonged hearing loss. According to OSHA, sounds over 120dB can cause the listener immediate pain, while 85dB of sound over prolonged periods of time can sustain permanent hearing damage [7]. Our mixer will have a max volume of 70dB allowed to account for this.

3.4 Improvements on Previous Project

Ambulatory disabilities are the most common type of disability in the US, with 6.6% of people having this kind of impairment [8]. The previous iteration of this project, while allowing for more expression through motion, also restricts the use of the device to those with full motor capabilities to use it to the fullest extent. Our version of this project opens access to those who are unable to stand or move freely for extended periods of time to make just as much use of the instrument as someone without any disability.

Additionally, our instrument can be used in areas with more restricted space, allowing it to be used in a wider variety of locations. The sensing is all contained within the LeapMotion controller, and the processing is done on a laptop with a small plugin board. All you need is a flat surface and the ability to move your hands, and the instrument can be fully used, as opposed to needing adequate space to dance and walk about.

3.5 Potential Current Project Improvements

One of the major improvements that could be made on our current design would be to implement two LeapMotion controllers working together as the sensors for our instrument. An additional Leap would mean that our project takes up more space and would cost more, but would be a great quality of life improvement for the user. There would be less incidental overlap of the right and left hands with more provided horizontal space. Also, the x position space for the left hand to move would double, allowing either for an additional two octaves of range, or for the space per note to be doubled, allowing for less required precision for each note. This way, trembling hands or small twitches or jerks would have less of an impact on the sound. This would require additional work within the software, necessitating a program to select which Leap to take data from at a certain time and careful boundary conditions for when the hand hovers close to the overlap between the two interaction boxes.

4. Progress Made on First Project

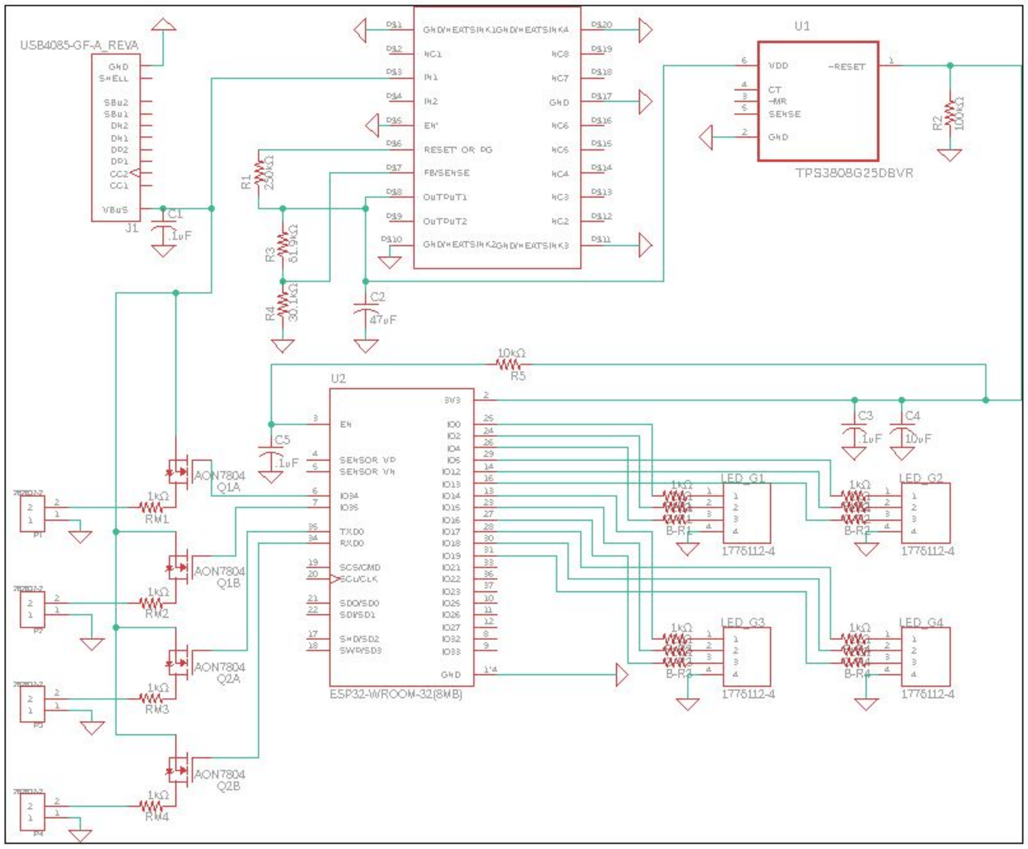


Figure 9 First Project PCB Schematic

We made progress on our first project, primarily on the PCB and our coordination with Champaign Urbana Adventures in Time and Space (CU Adventures).

We had finished a first draft schematic and routing for the PCB and discussed potential problems and ways to improve it, and looked into local options for manufacturing it. We had also met with CU Adventures, presented our Design Document, and gotten approval and feedback from them. With course and sponsor feedback secured, we had a parts list ready to order.

References

[1] "unordered_map::at - C++ Reference", *Cplusplus.com*, 2020. [Online]. Available: http://www.cplusplus.com/reference/unordered_map/unordered_map/at/. [Accessed: 08- May- 2020].

[2] R. Bedikian, "Understanding Latency: Part 1 - Leap Motion Blog", *Leap Motion Blog*, 2013. [Online]. Available: http://blog.leapmotion.com/understanding-latency-part-1/. [Accessed: 08- May- 2020].

[3] C. Anders, "Average Latency of a WiFi Network", *Quora*, 2020. [Online]. Available: https://www.quora.com/What-is-the-average-latency-of-a-WiFi-network. [Accessed: 08- May- 2020].

[4] "PCM512x 2-VRMS DirectPath™ , 112-dB and 106-dB Audio Stereo DACs With 32-Bit, 384-kHz PCM Interface", 2018. [Online]. Available: http://www.ti.com/lit/ds/symlink/pcm5121.pdf. [Accessed: 09- May- 2020].

[5] "API Overview", *Leap Motion Developer*, 2018. [Online]. Available: https://developer-archive.leapmotion.com/documentation/cpp/devguide/Leap_Overview.html. [Accessed: 06- May- 2020].

[6] "IEEE Code of Ethics", Ieee.org, 2016. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 03- Apr- 2020].

[7] "How Loud is Too Loud?". [Online]. Available: https://www.osha.gov/SLTC/noisehearingconservation/loud.html. [Accessed: 03- Apr- 2020].

[8] "2017 Disability Statistics Annual Report", *Rehabilitation Research and Training Center on Disability Statistics and Demographics*, 2017. [Online]. Available: https://disabilitycompendium.org/sites/default/files/user-uploads/2017_AnnualReport_2017_FINAL.pdf