The Emperor's New Instrument

Oscar Chen, Yuqi Gong, Karen Song

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*—A system that synthesizes music in real time based on hand movement and gesture. The system uses computer vision to map gestures and hand positions to predefined oscillators and frequencies, wireless gloves to map hand rotations to volume and pitch bending, and a software synthesizer to play music on the laptop speaker. This project allows people to play music with great freedom in musical expressions and at an affordable price.**

***Index Terms*— Computer vision, music synthesis, real-time playing, sensor, virtual instrument, wearable circuits**

I. Introduction

The virtual music system that we are building intends to make playing music more affordable and accessible to the general public. As long as the user has a compatible laptop and our music system, they can experience playing different instruments at a low cost anywhere and at any time they want.

Our team is planning to build a virtual music instrument controlled solely by hand movements and gestures. The left and right hands will each take on their own functions and collectively create sounds of different volume and pitch. There will also be a screen showing where the user's hand is, and which corresponding note is being played at the moment. Users will also be able to apply music effects like bending and execute simple commands such as pause/resume using gestures.

On the hardware and circuit side, there will be a pair of wireless wearable circuit gloves on the users' hands to detect tilt, and a central connection module that integrates the sensor's data and sends it to the users' laptops via a receiving module. On the laptop, there will be a software synthesizer that generates sounds in real-time based on the collected data, allowing users to set and use different gestures for commands and music effects.

In addition, our team has evaluated environmental and economic factors in designing our product. We aim to keep production and maintenance costs low, minimize energy consumption, and limit possible waste production from the system. Currently, music systems with similar functions (such as theremins) cost over $600. Our team intends to reduce the system development cost to under $400 to make it even more affordable.

II. Use-Case Requirements

Before designing the system, our team came up with a list of user requirements and their corresponding metrics (see section IV, design specifications)

The targeted user group of our product are people who are interested in playing music, own a windows/macOS laptop, and are comfortable and capable with setting up basic software.

In terms of human interaction, users will need our system to be fast and accurate. In terms of the speed of sound production, the delay between users changing their hands and the speaker outputting a different sound should not be humanly perceivable.

Also users will need the volume of the output sound to match their expectation. An instrument will be not useful if its sound production can not be reliably controlled. The volume control needs to be both accurate and intuitive. If the user halves the left hand's pitch angle, the perceived volume must also be halved.

Besides the performance of the instrument, users are also concerned with other factors such as portability, general usability and difficulty of setup. The users need the system to be a reasonable size to carry around, therefore the system should not weigh more than a typical laptop.

Users might prefer to use the musical systems for long periods of time without recharging, especially in occasions like parties. The batteries on the gloves need to last long enough for the users to enjoy a music session without being interrupted in the middle. Therefore our gloves should be able to function at least 5 hours before they need to be recharged again.

Before enjoying all the music, the users must be able to set up the system on their computers first. Our group ended up deciding that our product will be supported both on Windows and macOS. In order to make the system accessible to the users, the success rate of setting up on either system should be over 90%.

III. Architecture and/or Principle of Operation

A simple block diagram that lists the three subsystems of our project is shown in Fig. 1. For a more detailed block diagram that lists the individual components within a subsystem, please refer to Fig. 5 on page 3. Our system is composed of three major subsystems: wearable gloves, receiving module, and software running on a laptop. The first subsystem, the wearable gloves, consists of two identical circuits mounted on a pair of gloves. Each wearable circuit contains a 3-axis gyroscope IC, an Arduino Nano, a radio transmitter, and a LiPO battery. The wearable gloves will detect the rotation angles of both hands and send them to the receiving module via 2.4GHz radio. The second subsystem, the receiving module, contains a radio receiver and an Arduino Uno. It collects the angle information via radio and forwards it to the laptop via USB. The third subsystem, the software running on a laptop, synthesizes sounds based on the video feed of one's right hand and the angle information sent

by the receiving module. The software system contains four parts, a demux, a gesture recognition program, a hand tracking program, and a synthesizer. It also collects the video feed from an external webcam and writes audio streams to the laptop's speaker.

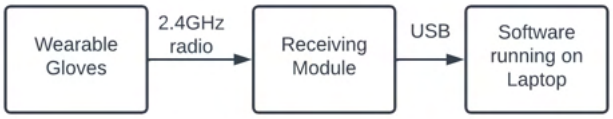

Fig. 1. Subsystem block diagram.

A. *Wearable Gloves*

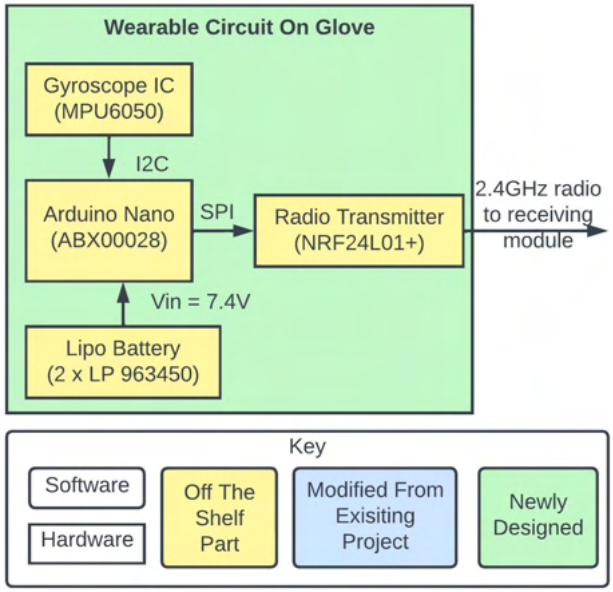

Fig. 2. Block diagram of one glove circuit.

The wearable gloves (see Fig. 2 above) are two identical wearable circuits that collect the 3-axis angular positions of both hands and send it to the receiving module. Since it is a fully isolated module, the circuit on each glove is powered by two 3.7V rechargeable LiPO batteries connected in series. A gyroscope IC is used to detect the 3-axis angular positions, a radio IC is used for sending the position data to the receiving module, and an Arduino Nano is used to integrate everything together.

B. *Receiving Module*

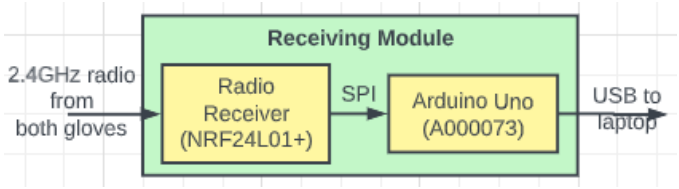

Fig. 3. Block diagram of the receiving module.

The receiving module (see Fig. 3 above) uses a radio receiver to collect the data sent by both gloves, and an Arduino Uno to forward it to the laptop via an USB cable. Each radio transmitter will send its address and one angle(pitch angle for the left hand, roll angle for the right) to the receiver. The receiver uses the addresses to differentiate between the angles, writes them into the corresponding index of a buffer array, and sends the buffer to the laptop via serial communication.

C. *Software running on laptop*

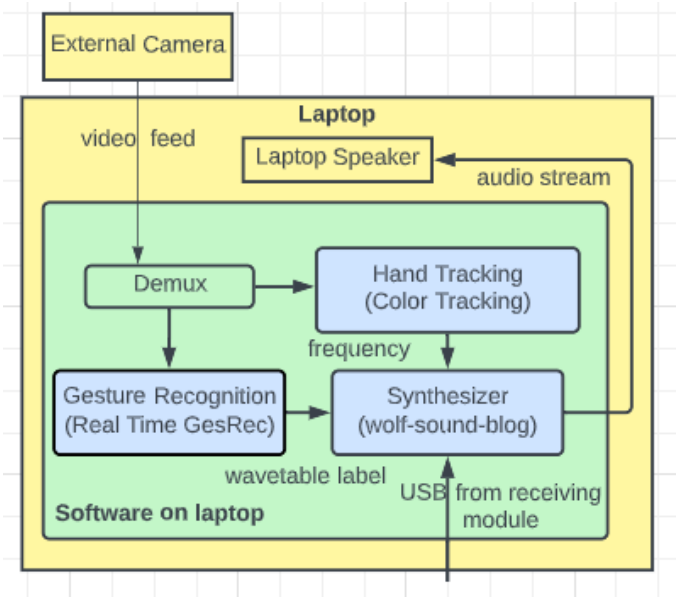

Fig. 4. Block diagram of the software running on laptop.

The software running on the laptop (see Fig. 4 above) is designed to synthesize sound in real time based on video feed from the external webcam and angular position information from the receiving module. It has four major components, a demux, a gesture recognition program, a hand tracking program, and a synthesizer. The demux allows the hand tracking program and the gesture recognition program to run in a time-multiplexed manner. The gesture recognition allows the user to select one of the provided wavetables using gestures. The hand tracking program allows the user to control pitch through hand movement. The synthesizer will first map USB data to volume and pitch bending. Then, with the wavetable and the bended frequency, it will sample the loaded wavetable, interpolate the samples, and write the result to the laptop's speaker.

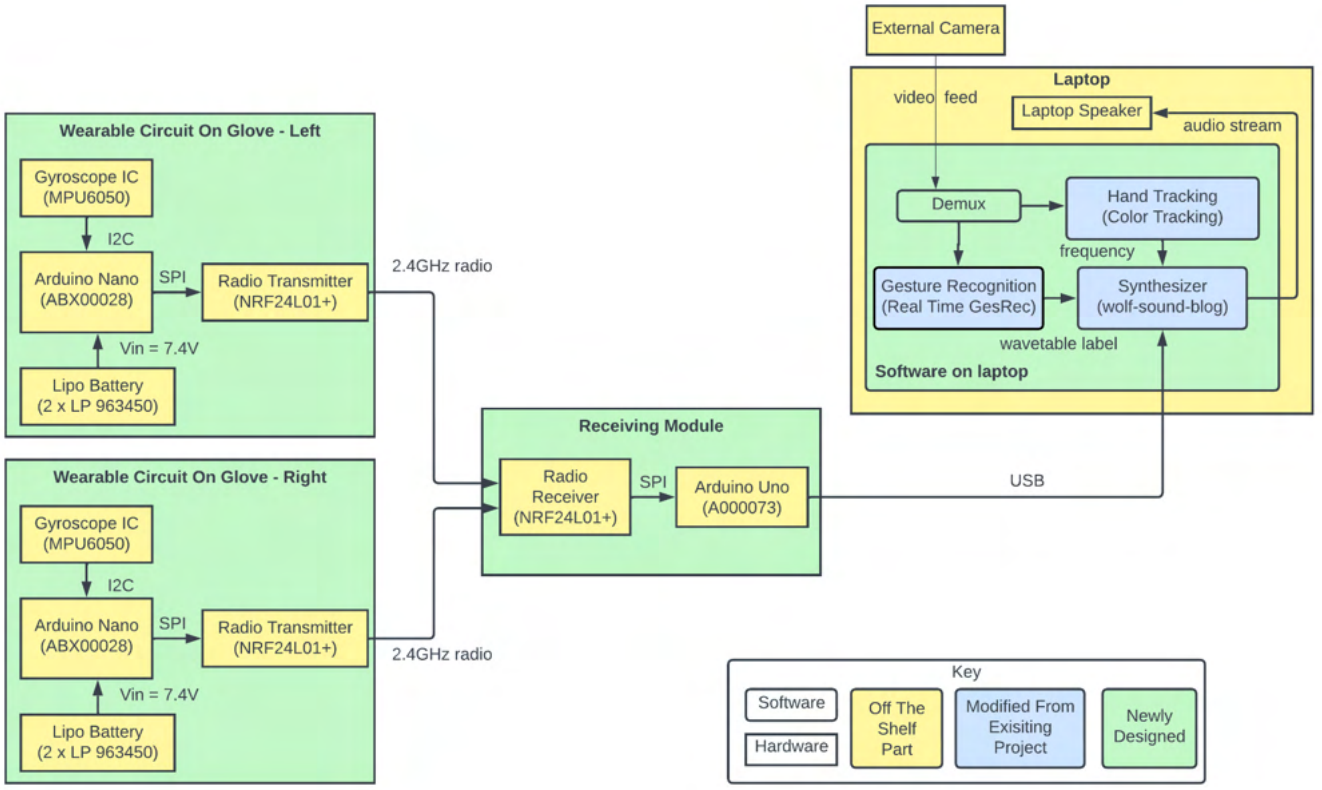

Fig. 5. Block diagram of the entire system.

IV. Design Requirements

To ensure there is no perceivable sound production delay, the latency between users moving their hands and a change in the sound output should be less than 10ms. We chose this number based on a past study on hearing aids' latency. The researchers reported that 10ms is the threshold where sound latency becomes noticeable and disturbing.

We also hope that our gesture recognition system will be 80% accurate. This includes all the gesture detection used to select an oscillator before playing as well as to pause and resume playing. Although we want to maximize the accuracy, after learning that a finely tuned CNN usually has a gesture recognition accuracy of 87% and a wearable gyroscope based system has an accuracy of 90%, we decided to scale down our accuracy metric to 80% due to software and hardware limitations.

Since the user is controlling the volume of the instrument using the pitch angle of his or her left hand, we also want the volume control to be intuitive and accurate. Specifically, halving the pitch angle should halve the volume (or in the decibel world, a decrease of 10 dB). We want our instrument to decrease the sound output by 10dB $\mp$ 0.5dB whenever the pitch angle detected by the gyroscope IC is halved. Because of the difference in laptop speakers, we cannot determine the maximum volume supported by our instrument. However, we need to ensure that no sound is being produced when the pitch angle is 0° (hand is parallel to the ground) .

In terms of portability, since we want our users to be able to carry our instrument around, we expect the total weight of our instrument (excluding the laptop) to be less than 4.8 pounds (the weight of a 16-inch Macbook Pro).

In terms of battery life, we want our gloves to function for at least 5 hours after the LiPO batteries are fully charged. We chose this number because it is recommended to eat one's meal between 4-5 hours apart. Realistically, we expect our user to play for only 1.5-2 hours in one sitting, as this is the typical length of a music rehearsal.

Finally, to ensure our instrument's accessibility, we expect it to be both Windows and macOS compatible. We will conduct user experiments on both OSs and expect the success rate of initial setups to be over 90%.

V. Design Trade Studies

Since we are transmitting angle information wirelessly from the wearable gloves to the receiving module, we mainly considered the latency, power consumption, and cost of three different protocols: WiFi, Bluetooth Low Energy, and Radio. For each protocol, we found the most popular chip that is supported by the Arduino library and compared their specifications (see Table 1 below).

Table I. Comparison between three wireless protocols

Protocol	Chip	Operation current (mA)	Latency(ms)	Unit Cost
WiFi	ESP8266	70	2-100	$7.50
BLE	HC-10	30	6	$10
Radio	NRF24L01+	12	5	$5

In terms of latency, theoretically ESP8266 has the lowest among the three. However, people have reported that the latency is usually not steady and depends on network traffic. In the worst case, the latency can be up to 100ms, which would violate our design requirement. Moreover, since we want to meet the 5-hour battery life requirement, the wireless transmitter needs to draw as little current as possible. ESP8266, being the most power-hungry of the three, is not the optimal choice.

In terms of latency, HC-10 and NRF24L01+ are about the same. However, HC-10's draw about twice as much current during operation. Most importantly, a HC-10 receiver can not listen to multiple HC-10 transmitters at the same time. This means that we need 4 HC-10 in total to collect angle information from both gloves. Not only does this drive up the total cost, it introduces complexity to the receiver module's circuit. As a result, HC-10 is not the optimal choice either.

NRF24L01+ ranks best in all specifications. Most importantly, NRF24L01+ supports building a radio network, where each NRF24L01+ can receive signals from 5 NRF24L01+ transmitters. As a result, we only need three NRF24L01+ (two transmitters, one receiver) to collect data from both hands. In summary, choosing it as our wireless transmitter will help us the most in meeting the 5-hr battery life requirement, the 10ms latency requirement, and keeping the cost of our instrument low.

A. Battery choice for wearable gloves

Since our wearable gloves are not connected to the laptop, all of its components need to be powered by batteries. Since both the radio and gyroscope chip are drawing power from the Arduino Nano, we need to find a suitable battery to power the Arduino. The most obvious choices are dry-cell battery, lithium-ion battery, and LiPO battery. Since we do not want our users to constantly change the batteries on their gloves and create unnecessary chemical waste, we quickly ruled out dry-cell batteries. Dry-cell batteries usually do not support recharging and have a lower capacity compared to LiPO batteries. For example, a typical 9V battery has a capacity of 500 mAh, where a 3.7V LiPo battery has a capacity of 1800mAh.

Between lithium-ion batteries and LiPO batteries, although lithium-ion batteries are cheaper and have a higher power density than LiPO batteries, they are less safe and age faster than LiPO batteries. Specifically, lithium-ion batteries can catch on fire easily if they are punctured. Since the batteries are attached to the gloves, this is an immense safety risk we are not willing to take in exchange for a lower price and a higher power density. However, there is one complication to using LiPO batteries: their output voltage is usually around 3.7V, where the Vin for Arduino Nano needs 7-12V. This means that each glove needs to be powered by two LiPO batteries put in series. Although this seems like it will make the gloves heavier, LiPO batteries usually come in different sizes and weights. As a result, we can easily create a 7.4V Vin with two small LiPO batteries.

B. Synthesizer choice

The major part of sound production lies within the synthesizer, and we need to decide between using a pre-existing synthesizer or building one from scratch. Although using a pre-existing synthesizer will enable users to create more musical effects, it introduces many problems. First, it is hard to find an open-source synthesizer that is both Windows and macOS compatible. Even after finding one, we still need to read its source code and figure out how to pass in our inputs to its interface. Not only is this a time consuming task, it makes fixing latency problems virtually impossible. As a result, developing our own synthesizer is more preferable. Since both our gesture recognition program and the hand tracking program are written in Python, it is very easy to write the output of these modules to the synthesizer with low overhead. Also, fixing Python library compatibility issues is also easier than fixing compatibility issues of an open-source software written by someone else.

VI. System Implementation

A. Wearable Gloves

Each hand glove consists of two LiPO batteries, an arduino nano, a gyroscope (MPU6050) and a radio transmitter (NRF24L01+). The two LiPO batteries will be in series and connected directly to the arduino and provide a voltage at about 7.4 Volts. The arduino will be the power supply for MPU6050 and NRF24L01+. MPU6050 is used to measure the angular position and NRF24L01+ is used to send the angle to the computer.

When the user raises his/her left hand, the gyroscope will measure the pitch angle and the transmitter will send the data to the computer. The volume will be increased from 0 to 100 percent as the pitch angle increases from 0 degrees to 90 degrees.

When the users' right hand is raised, the angle will also be transmitted to the computer by NRF24L01+ and it will act as a pitch bend. The pitch will be raised or lowered smoothly based on the angle. If the angle is negative, the pitch will be lowered according to the absolute value of the angle. Otherwise, the pitch will be increased. The angle of positive 90 degrees will cause the output to bend 2 semitones up, and the angle of negative 90 degrees will make the produced note bend 2 semitones down.

The component layout of the wearable circuit glove is shown in figure 6.

Fig. 6. Two gloves are identical. Right hand (right) will include a color coded middle finger for position tracking color detection when the product is in use. The color of the middle finger will be determined based on the color of the user's shirt.

B. Receiving Module

The receiver component consists of an NRF24L01+ radio receiver, which can listen to up to 5 different NRF24L01+ radio transmitters. It captures the information from the wearable circuit gloves and sends the information to the Arduino Uno, which will later transmit the data through a USB port to the laptop and into the synthesizer.

Our team decided to remove the autonomous lighting system, because it is counter intuitive for the user to carry around a large number of LEDS while the system is supposed to be portable. In addition, the external camera our team introduced for better video quality already has such functionality.

C. Software running on laptop

The software component of the system consists of mainly three parts: the gesture recognition model (based on *Real-time GesRec*), the color tracking model (based on *Multiple Color Detection in Real-Time using Python-OpenCV)* and the synthesizer (based on *Wolf-sound-blog*). All of these components are developed based on altering preexisting code bases to meet our design requirements. These models are carefully selected after our team browsed Github and examined dozens of similar models.

For the gesture recognition model, we will only be using a subsection of the gestures provided in the dataset since there are not as many commands and music effects that our music system needs. This subset of gestures will contain gestures that are relatively unique and easy to be detected, such as holding a fist or pointing a thumb up. Figure 7 below shows how *Real-time GesRec* generates a classification output for each individual gesture inside its database. The hand tracking models will mainly be used to identify simple gesture commands.

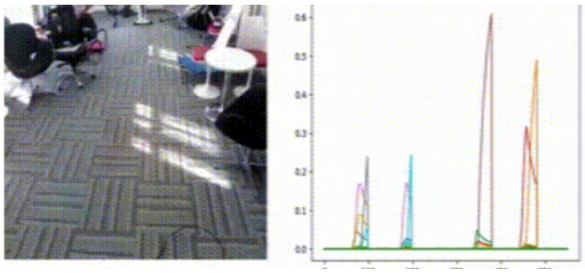

Fig. 7. Real-time GesRec shadowing classification scores (right) of each gesture presented on the video (left)

Initially our team attempted to use a hand tracking module to keep track of hand locations (shown in figure 8). Then, we realized that mediapipe, the library used to identify and mark out the hand, takes a long process time, more than 8ms without the rendering process. This visible lag interferes with our user requirement of fast sound production. Instead, we decided to use a color detection system (shown in figure 9). This detection method takes less than 2.5ms to both track and render a frame, which is much faster.

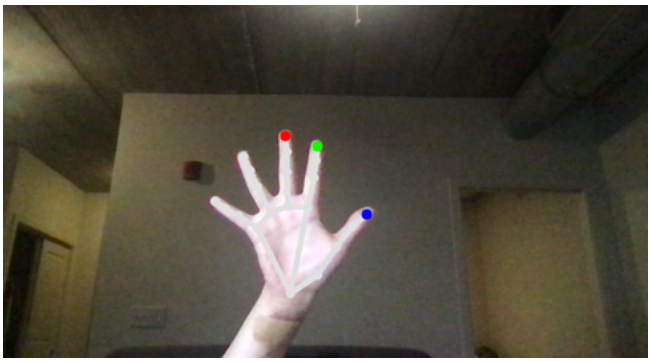

Fig. 8. Hand tracking module using mediapip, tracks the location of the user's thumb, index finger and middle finger tip. Code based on Yoha.

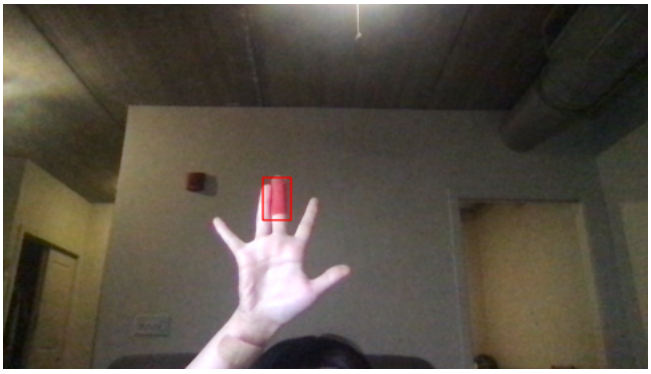

Fig. 9. Color tracking module that detects the largest area of a specific color under different lighting conditions

The color tracking model is used to identify which of the thirty-two note areas the user's pitch controlling hand (right hand) is likely lying in (depicted in Figure 8). First of all it will identify where in the video feed is the user's right hand middle finger, which is covered, then classify which area on the screen does the hand fall into. This screen division

information will also be shown on the laptop during the user's session on the video feed.

Fig. 10. Screen area splitting for different pitches on a typical 16:9 laptop screen (16 * 2 sections)

The synthesizer will first load a pre-defined oscillator class based on the gesture detected by the gesture recognition program. When the user is playing, the hand tracking program will pass in the note frequency that corresponds to the user's hand position. The USB cable from the receiving module will pass in the volume and pitch bend information. Using the note frequency and the pitch bend information, the synthesizer will generate a numpy array that represents the oscillator oscillating at the corresponding frequency, multiply the array by the volume, and write to the audio stream buffer by buffer. When the user changes to a different note, the synthesizer will gradually "glide" to the new note by gradually changing the frequency. The glide duration can be changed before playing for a different sound effect.

D. *External Components*

In order to ensure the quality of video feed and color detection, our team decided to add a mounted camera on top of the laptop as an external and more unified video source. This particular camera also has a lighting function.

VII. Test, Verification and Validation

Below is a summary of the test and validation results.

Criteria	Requirement	Results	Requirement met?
Latency	<10ms	8.33ms	Yes
Portability	<4.8lbs	0.97lbs	Yes
Volume accuracy	± 0.5dB	0.1dB	Yes
Battery Life	>5hrs	21.6hrs	Yes
Plug-and-playability	90%	100%	Yes

Table II. Summary of the test and validation results

A. *Use-Case Specification-Latency*

In our system, latency consists of four parts: radio transmission, serial communication, sound processing, and displaying the video feed. Since the latter two can happen in parallel, and sound processing takes longer than showing a video frame on screen (verified using performance analysis library *cProfile*), the critical path of latency only consists of the first three parts.

To measure the latency of the overall system, we waved our left hand between the max volume position and the min volume position. We also recorded the test with a 60FPS camera and used iMovies to analyze the latency. In iMovies, we saw that the volume changes half a frame later than the video frame(see fig 11 below). Since one frame in a 60FPS video corresponds to 16.66ms, half a frame corresponds to 8.33ms. We also verified that we could not perceive this level of latency.

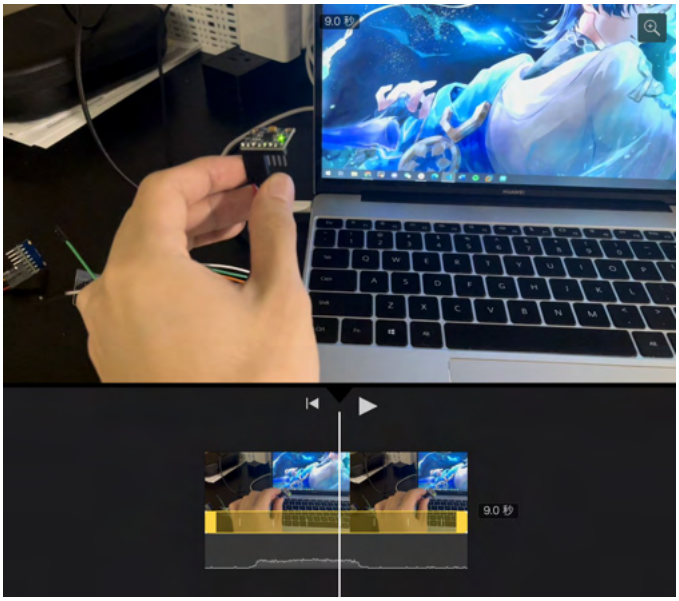

Fig. 11. Latency measured in iMovies.

B. *Use-Case Specification-Portability*

For the portability test, we measured our system (excluding the laptop) on a scale. We aimed to bring our system's overall weight below the typical weight of a laptop(4.8lbs) so that it is easier for the user to carry our system around. The overall weight of our system was 0.97lbs, and half of the weight (0.48lbs) comes from the webcam we purchased. In reality, the user can save more weight by using the laptop's built-in camera, given that it has a relatively good resolution and color accuracy.

C. *Design Specification-Volume Accuracy*

Due to the different models of laptop speakers and the background noise level, we could not test our volume accuracy with absolute metrics. Instead, we decided to use a relative metric. After some research online, we found that humans perceive a +10dB in volume as “twice as loud”, so in our synthesizer, we used a non-linear volume scaling to accommodate our non-linear volume perception. To test the volume accuracy, we tilted the gyroscope to different degrees and measured if halving the pitch angle would result in a -10dB in sound. We used *Decibel X* (a decibel meter app) to measure the volume. In fig. 12 below, you can see a -9.9dB change in volume when we tilted the gyroscope from 90 degrees to 45 degrees.

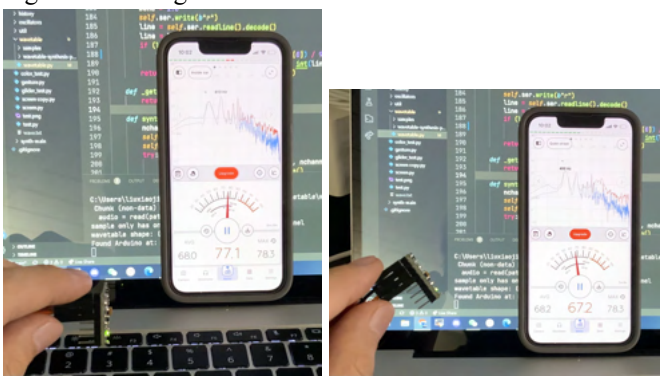

Fig. 12. Volume measured with *Decibel X.*

D. Design Specification-Battery Life

In order to measure the battery life of our system, we used the formula *Battery Capacity / Current * (1 - discharge safety)*. Since our LiPO battery has a capacity of 1800mAh, and a discharge safety of 40%, we only need to measure the operating current. Using a multimeter (see fig. 13 below), we measured the current draw of our system, and found that our system has a battery life of about 21.6 hours. This was the result of two optimizations. First, LiPO batteries have a much higher power density than traditional alkaline batteries. Secondly, because the user will be very close to the laptop while using our system, the radio transmission distance is relatively small. As a result, we set the power level of both the transmitter and receiver to minimum power level, which in turns reduces the current draw and increases the battery life.

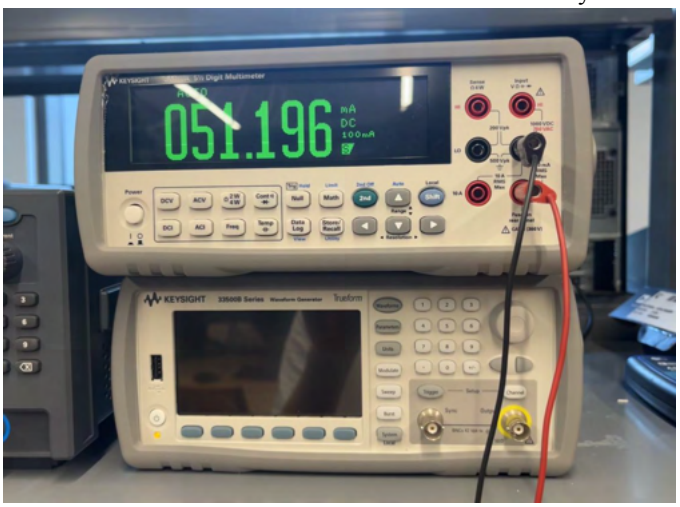

Fig. 13. Operating current measured with a multimeter.

E. Design Specification-Plug-and-Playability

To ensure plug-and-playability, we developed our system concurrently on both Windows and macOS. When using python libraries, we mostly used libraries supported by both OSes. As a result, in the final user testing, where we found 2 participants with a Windows laptop and 3 participants with a MacBook, all of them were able to set up our system on their first try.

VIII. Project Management

A. Schedule

The entire project spans a little more than 2 months. Our team divided the project into three smaller sub-parts: design, development and testing, and final integration.

Specific schedule is shown in fig. 14.

B. Team Member Responsibilities

Based on the different areas of concentration in their studies, team members are assigned different tasks. There are several overlapping tasks in order to speed up the process and make integration later on easier. The basic task assignment is listed below:

- Oscar is working on the synthesizer and arduino design and implementation,
- Yuqi is working on the wearable circuit and arduino design and implementation
- Karen is working on designing and developing the hand monitor system.

Team member responsibilities / task breakdown could also be found in fig. 14.

C. Bill of Materials and Budget

We bought a light sensor but it was not used because we decided to use an external camera with the lighting system. The external camera can provide enough light for our product. For most components, we bought more than we needed in case we broke some of them.

A more detailed list of the purchased items along with a description (including the model number, manufacturer, and cost) can be found in table III on the last page.

D. Risk Management (used to be Risk Mitigation Plans in Design Document)

During this project, there are a series of risks that we encountered. Fortunately we were able to come up with either solutions or alternatives to these issues.

Since we aim to produce a system that is low in cost, and our circuit does not contain any extremely overpriced items, we did not encounter any problems in our budget planning. In addition, although several things got pushed around due to sudden events, our original schedule contained enough extra time for these delays.

There are several possible risks that could show up during our process. During the design process, our team is relatively

worried about the integration of the hand gesture and movement detection system. As a result, we decided to use a demux structure so that our program will either perform gesture recognition or hand tracking. The video feed will be fed into two time-multiplexed models, one for gesture recognition and one for hand tracking. Splitting the functionality allows us to access more pretrained models and hand tracking programs, which greatly reduces the development time.

Our team decided that the majority of the models that we are using will be coming from preexisting code. We included multiple model source codes that we researched to ensure at least one of them is compatible with our product. Our team also included different approaches to the same function, for example using both position and color for hand location tracking, in case one approach does not work or does not meet our requirements. At one point of the project, our integrated system speed did not meet our time metric requirements. The latency between gesture input and sound output is relatively high, and there is a clear lag that can be felt by users. Fortunately our backup plan of using color detection instead of mediapipe hand tracking is present, and worked ideally.

Since our system has a physical component, it is possible that items do not arrive on time or the quality of ICs (especially the tilt sensor and the radio chip) is not guaranteed. There is not much we can do about items not arriving except order early. However we did ensure that whatever did arrive functioned properly and as expected by starting component testing immediately after each batch of chip arrives. Luckily we did not receive a malfunctioned chip, therefore we did not need to order the same components again.

Another possible risk comes from our system setup. There could be potential issues with python libraries, and these issues could be specific to either Windows or MacOS. Our team planned to downscale this section of the project to only supporting one operating system in case the other one really does not work.

One risk that we failed to acknowledge was the possibility of not being able to attach the component on the gloves properly. During the middle of the project, we realized that the best way to attach our circuit on our glove is through sewing, however none of us in the group knew how to perform this task. Though fortunately we were able to overcome this problem and complete our product.

Most problems that we faced in our project were correctly predicted and either mitigated or fixed.

IX. Ethical Issues

In every product created, there will be ethical issues regarding the things that a user with bad intentions could do. Although the area of sound production generally does not face many complicated ethical issues, there are still things that our team and our users need to think about when using the product.

First of all, since our product involves producing sound on electronics, it is subjective to privacy invasion issues. It is possible that a malicious user could purposely make the sound very loud and therefore cause disturbance to other users, and there is no procedure on the developer's end that could mitigate such a problem, because sound production naturally varies on the type of laptop used and detection is highly dependent of the location the sound is measured.

Another possibly ethical issue is that our product has not yet come up with a way to cover and protect its circuit, batteries and other hardware components. Under correct use, our product should not harm any user. However, it is still possible that malicious people trick children and other people who are weak in judgment into doing things like touching the battery when it is charging or pulling on the wires forcefully. Although this is not a problem in our design, it still could cause people to be injured, and it will be really hard to tell who is ultimately responsible for this. In future products our team will probably add a protection layer over the gloves in order to keep everything in place and prevent people from touching things that they should not have access to.

In summary, our design alone is not prone to being ethically exploited unintentionally. However due to limited time of development we do not yet have enough measures to stop this system being used by malicious users to create safety hazards and invade other people's privacy.

X. Related Work

The source of inspiration for the designing of this project is from a youtube video posted by youtuber musician Grégoire Blanc. In this video he plays the song Clair de Lune on a Theremin – a very special musical instrument invented by a Russian scientist Leon Theremin in the 1920s. In this video, Blanc moves his hands in front of a device, then music of different volume and pitch is created.

The Theremin system uses two antennas to generate an electro-magnetic field that can create a tone. It's the only musical instrument that has no strings, keys or pipes, and is played without touching the actual object. There are multiple videos explaining how the theremin works and how to play it. In one of these videos, the channel owner interviewed one of the best Theremin musicians, Carolina Eyck, who also provides a demo.

The Theremin gave our team the inspiration to create a touchless musical system that can be played by simply moving your hands in the air. Instead of magnetic fields, our team decided to go along with a different approach, a more modern approach to sound production.

Our final product is close to a real Theremin in terms of its functionalities. It could produce music through motion, but at the same time has advanced options such as changing the output sound type. It is probably cheaper in terms of price, however it requires more effort to set up, and relies more on the environment.

XI. Summary

In the end, our system was able to meet the design specifications and include all the functions that we would like.

It is overall a very successful attempt to create a new instrument. Our system has the ability to produce sound quickly and accurately, but due to time constraints we were unable to come up with a written documentation that instructs the users on the more advanced functions such as the approximate degree of sound bend. If we had more time on this project, our system should be able to support more instruments, which should be our area of advantage in comparing our project with traditional musical instruments. In addition we will probably focus more on the aesthetics of our system, and cover up the circuit for better safety precautions.

A. Future work

At this moment, our team is unsure whether or not we will be continuously working on this project, because we are not certain about our availability in future semesters. If we were to work on this project further, we will consider adding in the functionality using gestures to swap between music quadrants, and introduce even more different types of instruments in the system. If we have access to a larger screen, it is even possible to have multiple users play on the same screen and create harmony.

B. Lessons Learned

From this project, the most important design lesson that our group learned is that nothing really works according to our planning and there should always be a series of risk management/mitigation plans in place to detect the issue earlier and provide a probable solution to minimize the damage.

Time management is also an important thing that our team acquired during the process. Although our initial schedule looks ideal and reasonable in the beginning of the semester, there are always unexpected events, both related (for example something just does not work) and not related to this class (there is suddenly a lot of work coming from another class) that happens. Therefore it is important and useful to always re-examine our schedule, and make sure that there is sufficient extra time left in each section for flexibility.

Material wise our team figured out that our gyroscope chip, MPU6050, is really nice, and we could use it on future projects that involves motion sensing.

Glossary of Acronyms

CV - Computer Vision
LED - Light-emitting Diode
LiPO battery - Lithium-polymer Battery
USB - Universal Serial Bus

References

[1] Gunduz, A et al.(2019) Real-time-GesRec[Source code] https://github.com/ahmetgunduz/Real-time-GesRec
[2] Takahashi, K (2020) hand-gesture-recognition-using-mediapipe[Source code]https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe
[3] Bahadur, A (2017) HandMovementTracking[Source code]https://github.com/akshaybahadur21/HandMovementTracking
[4] Handtracking.io(2021)Yoha[Source code] https://github.com/handtracking-io/yoha
[5] Surge Synth Team(2018) Surge[Source code] https://surge-synthesizer.github.io/
[6] Alan, Ruiz, T (2021) Synthe[Source code] https://github.com/18alantom/synth
[7] "Debussy - « Clair De Lune » on the Theremin." Youtube, uploaded by GréGoire Blanc, 22 Dec. 2013, www.youtube.com/watch?v=PjnaciNT-wQ&ab_channel=Gr%C3%A9goireBlanc.
[8] "Theremin - The World's Strangest Instrument Explained By @Carolinaeyckvideos." Youtube, uploaded by DW Euromaxx, 3 Feb. 2021, www.youtube.com/watch?v=SsQloi46AQU&ab_channel=DWEuromaxx.
[9] "Steve Vai Guitar Lesson - Bending Notes - Alien Guitar Secrets: Passion & Warfare." Youtube, uploaded by TrueFire, 14 Jul. 2016, www.youtube.com/watch?v=uBINGNIFnGI&ab_channel=TrueFire.
[10] goodday451999. Multiple Color Detection in Real-Time using Python-OpenCV[Source Code] https://www.geeksforgeeks.org/multiple-color-detection-in-real-time-using-python-opencv/#article-meta-div
[11] Wilczek, J (2019-2023) Wolf Sound. https://thewolfsound.com/

Item	Model Number	Manufacturer	Source	Quantity	Per Cost	Total Cost	Notes
Lipo Battery	LP963450	EEMB Battery	Amazon	5	$16.99	$84.95	---
Connector Cable for Battery	JST-PHR-02	Elechawk	Amazon	1*10pcs	$6.99	$6.99	Did not plan for buying this in the design report.
MPU6050 Gyroscope	3-01-0122	HiLetgo	Amazon	3*3pcs	$3.33	$9.99	Purchased 9, but only used 2
Arduino Nano	ABX00028	Arduino	Digikey	3	$12.75	$38.25	Purchased 3, but only used 2
Micro-USB Cable for Arduino Nano	7P6EV4	Amazon Basics	Amazon	1*3pcs	$13.85	$13.85	---
Arduino Uno and Cable	A000073	Arduino	Digikey	1	---	---	We used our own Arduino Uno.
NRF24L01+ Radio Transmitter	BC22523	Makerfire	Amazon	10	$1.20	$11.99	Purchased 10, but only used 3
Jumper wire	DBX-FB07-24AWG-10CM-TC-120	Chanzon	Amazon	1*120pcs	$9.49	$9.49	Did not plan for buying this in the design report.
Half Finger Gloves		Monochef	Amazon	1*2pcs	$7.99	$7.99	This was included in the design report.
Light Sensor	2748	Adafruit Industries LLC	Digikey	1	$2.50	$2.50	Purchased but not used.
Total Cost				**---**	**---**	**$186.00**	---

TABLE III. BILL OF MATERIALS

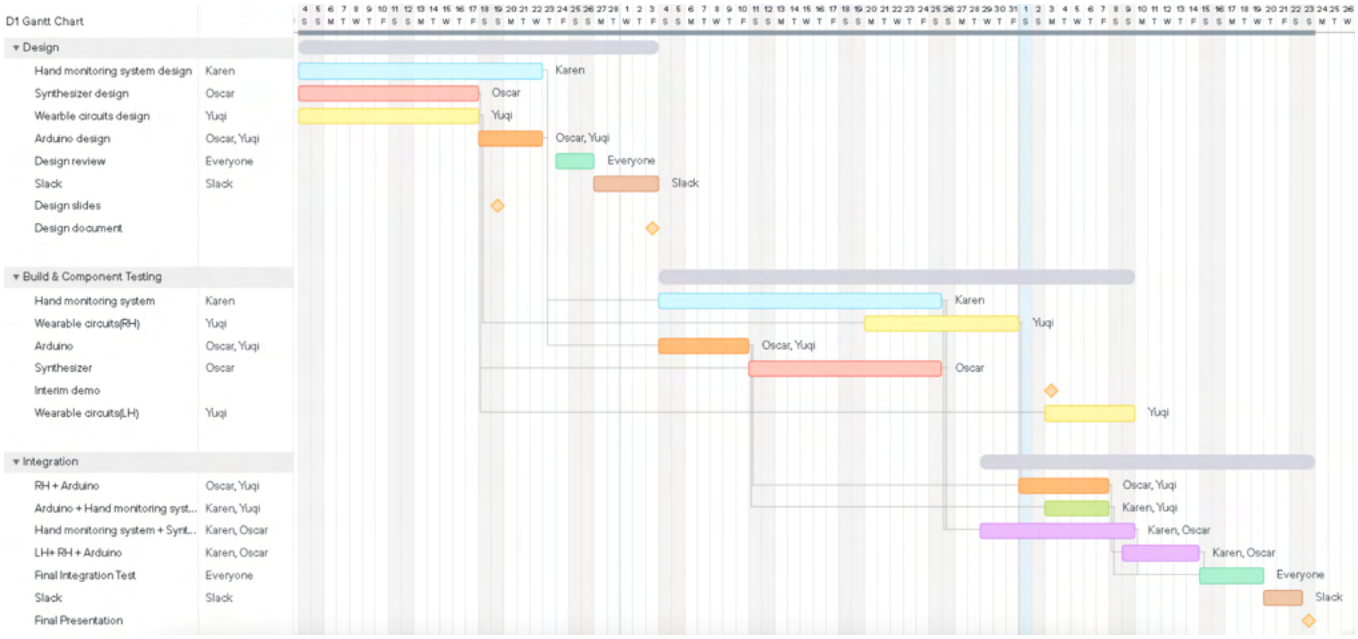

Fig. 14. Schedule