

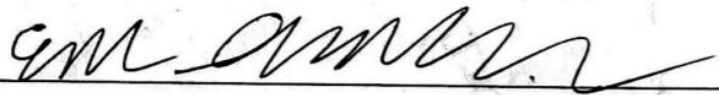
2 Birds 1 Capstone - Air Guitar


Karan Chawla, Erik Haukenes, Jacob Holton, Josie Li, Hua Uehara

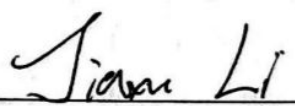
Date of Submission: 12/16/2019


Capstone Design ECE 4440 / ECE4991


Signatures











Statement of work:

Karan Chawla:

I developed the Android application that users will interact with to simulate playing a guitar - the application was developed in Android Studio using Java. The user interface of the application emulates that of a physical guitar fretboard (three distinct fret sections with six strings across them), also allowing simultaneous touch gesture recognition of up to ten points. For purposes to emulate how an actual guitar is played, it is assumed that there will be six input points maximum at any point. When handling the input, circles appear under the points of contact to help visually guide the user as to what strings are currently being strummed; as the strings are distinctively colored to serve as better differentiators, the circles are accordingly colored according to the closest string color. Apart from the user interface of the application, I developed the logic to determine which positions on the guitar are actively being played, for example if the same string is strummed on two different frets, then the lower fret will take precedence and the application will not transmit information relating to the upper fret/string position combination being played. I also developed the logic to transmit the fret/string combinations to the myRIO via serial bus communication using UART protocol.

Erik Haukenes:

I worked on the distance sensor multisim and ultiboard, assisted in routing the main PCB and MyRIO connections, and created the LabVIEW VIs that received information from the distance sensor and the Android application via the UART protocol. I also worked to determine how the data should be encoded and transferred when sent from the Android application to the MyRIO. In order for our virtual guitar application to work, the distance along the neck needed to be determined using the distance sensor. I used distance information and combined it with the data sent from the Android application with LabVIEW to feed into the Karplus-strong algorithm to determine the frequencies that needed to be played for any combination of distance and touch screen inputs. When we decided to pivot to a library of soundwaves, I created the guitar samples using audacity, and created the VI that would convert the samples from audio files to playable waveforms, and summed them together on the MyRIO.

Josie Li:

I wrote the Karplus Strong Algorithm both in real-time target and in FPGA. The initial Karplus Strong Algorithm was implemented in the real-time target whose limited timing mechanism counteracted the waveform performance. The alternative to implement the algorithm in FPGA chassis with a 40 MHz clock could output the correct pluck for each note. However, the limited resources in FPGA made me to explore the possibility in which sample arrays for all notes are generated in the real-time target and are then fed into a FIFO in FPGA. The conflict between reading from pins on FPGA and on real-time target made this alternative infeasible. In addition to implementing Karplus Strong Algorithms, I also worked with Hua to integrate different submodules with the final Karplus Strong Algorithm, as well as tested and modified the

whole LabVIEW system. For example, after testing the functionality of the accelerometer button, I integrated the LabVIEW code for button into the whole system so that the button can correctly turn on the waveform-generation subVI without delay.

Hua Uehara:

I worked primarily on LabVIEW code to integrate the different submodules. This involved making a frequency selector VI that takes input from the distance sensor and UART sub-VIs and uses them to index a table of frequencies, providing the frequency for each of the six strings. These frequency values could then be consumed by the Karplus Strong Algorithm to generate the sound output. After we switched to using pre-recorded soundwaves, I adapted the frequency selector to index the corresponding soundwave directly. I also did general LabVIEW related tasks including making test VIs for the accelerometer board, troubleshooting the Karplus Strong algorithm, converting test VIs for UART and distance sensors to subVIs, and outlining the main VI that utilizes subVIs for modularity.

Jacob Holton:

I worked primarily on the hardware and board design section. This includes the accelerometer and the main PCB. This task consisted of reading component data sheets and biasing the accelerometer and voltage regulators to our groups desired specifications. For the main PCB, I integrated UART and analog signals and routed them into myRIO. This board also regulated 5V output from the myRIO down to 3.3V to turn on the accelerometer board and push button signal. After the boards were designed, I ordered components and populated the boards. I then ran the boards through a number of functional tests verifying DC biasing and functionality of the components including the distance sensor and accelerometer.

Table of Contents

Contents

Capstone Design ECE 4440 / ECE4991	1
Signatures	1
Statement of work:	2
Table of Contents	4
Table of Figures	5
Abstract	6
Background	6
Design Constraints	7
External Standards	8
Tools Employed	9
Ethical, Social, and Economic Concerns	9
Intellectual Property Issues	12
Detailed Technical Description of Project	14
Project TimeLine	29
Test Plan	31
Final Results	32
Costs	33
Future Work	33
References	34
Appendix	36

Table of Figures

Figure 1. Android Application UI	15
Figure 2. Android Application UART LabVIEW Block Diagram	16
Figure 3. Karplus Strong Algorithm on myRIO real-time target	17
Figure 4. Karplus Strong Algorithm on myRIO FPGA target	17
Figure 5. LFSR random number generator	18
Figure 6. Random number generator scaling	18
Figure 7. FPGA target system integration	19
Figure 8: FPGA Utilization for single string Karplus Strong implementation	19
Figure 9. Karplus Strong Algorithm on myRIO FPGA target for two notes	20
Figure 10. Final Karplus Strong Algorithm with pre-recorded sound waves	21
Figure 11: Finger position to frequency table	22
Figure 12: Fret to frequency LabVIEW VI	22
Figure 13: Accelerometer Multisim Schematic	23
Figure 14: Pullup Resistor Accelerometer Modification	24
Figure 15: Accelerometer Ultiboard and Final PCB	24
Figure 16: Distance Sensor Multisim Schematic	25
Figure 17. Distance to fret LabVIEW subVI	25
Figure 18: Distance Sensor Ultiboard and Final PCB	26
Figure 19: Power and Routing Ultiboard and Final PCB	27
Figure 20. System integration LabVIEW VI block diagram	28
Figure 21. System integration LabVIEW VI front panel	28
Figure 22: Original Gantt Chart	29
Figure 23: Improved Gantt Chart	30

Abstract

S.H.R.E.D (Sensor Handheld Rock and Roll Electronic Device) is a musical instrument designed to give musicians the experience of playing an air guitar, while providing a realistic sound. The project will involve an Android phone application that takes in the finger positions of the user to determine the notes being played, as well as a distance sensor to determine which frets along the neck of the guitar are being played. An accelerometer will be used to determine when the guitar is being strummed, and all of the sensor and phone application data will be relayed to a National Instruments myRIO board via a printed circuit board with wired connection to the accelerometer and phone. The myRIO board then creates soundwaves for a variety of instrument types (for our purposes, a guitar) using signal processing techniques such as the Karplus-Strong string synthesis algorithm.

Background

The initial idea to create a set of musical ‘air’ gloves came from our group’s combined interests and experience with musical instruments, alongside previous exposure to the idea of using glove-based technologies and their sign-language applications. Our group chose to explore the musical applications of these gloves due to the overall intrigue and practical application of learning how to play a musical instrument without the burden of paying for expensive equipment. Throughout the initial design process of these gloves, our group realized the utilizing existing technology in the form of a smartphone for note capture would resemble a more sensible approach in the sense that mimicking a physical guitar would result in both improved usability and technical capacity of the project as we could utilize sensors internal to the phone - leading to the renewed proposal of a musical air guitar rather than air gloves.

Traditional guitar strings can also be painful to play when a musician has not developed callouses, making a touchscreen-based input appealing to beginners. Another use case of our device is in facilitating airplane travel for music hobbyists and others as booking an extra seat or risking damage to a musical instrument during air travel comes with a large financial risk not found in our product. Building off of this sentiment, our device is also suitable for a public environment in which a user can send the output of their signal to a set of headphones instead of through a speaker, useful for practice sessions in a quiet setting.

Projects that attempt to replace a physical musical instrument with either substitute physical items (smaller hand-held devices, programmable guitar necks, etc.) and/or wearable gloves have been constructed in the past, however our project differs in several key aspects from these companions. A product currently in the consumer market, Kurv Guitar [1], seeks to reinvent how specific hand-held devices control musical intonation and note-playing, however the project strays into the realm of being a ‘new’ musical device rather than a substitute for an existing one, the market our device aims to fill. Specifically, the Kurv Guitar functions via a hand-held button-operated device in the user’s left hand and a motion-controlled guitar ‘pick’ in

the controller's right hand. Through a combination of button presses and moving the pick in a specific orientation (upwards to play a note, downwards to play a chord, etc.) users are able to mimic playing a guitar. However, the system through which a guitarist would normally change octaves via moving their hand up/down the neck of the guitar has been fundamentally changed and is a definite distinction found in our project, alongside no physical button press for note selection.

Misa Digital is a company developing digital guitars with alternatives to standard strings [3]. The guitars are full sized, and rely on a capacitive fretboard that runs up and down the entire neck of the guitar. In order to play the notes, a touch screen is integrated into the body of the guitar where touching the displayed string produces a sound. The touch screen has multiple sections, such that touching different sections produces different synthesized instrumental sounds that can be played at the same time. The touch screen also has different modes, with one of them allowing for one string to be played at a time. This project is similar to ours, although it is housed in a full sized guitar, and their capacitive touch screen is used for playing the notes, instead of the fretboard note selection. Misa Digital has also not implemented accelerometer based strumming into any of their products.

Several pieces of our project rely on past experiences, one of the most prominent items being the accelerometer experience we gained through our Embedded Systems introductory course. The Android application that users will be interacting with will be coded in Java using Android Studio, with previous experience coming from prior Mobile Application Development courses. Control systems will be heavily utilized in our analysis of the musical notes to produce a guitar-like sound, as the Karplus-Strong algorithm relies on a cyclic averaging filter/feedback system - concepts taught in Embedded Systems Design, Digital Signal Processing, and ECE Fundamentals 2 and 3.

Constraints

Design Constraints

Several design decisions and constraints came into play when deciding how the prototype of the project would function and weighing this versus an actual deployment build of our project. Specifically, we utilized wired connections between components due to initial research that described the latency of bluetooth communication in smartphone communication being upwards of two seconds - a fact that would be detrimental to the real-time applications of playing an air guitar. As such, in addition to wanting to keep the overall project portable in all scenarios (therefore not looking into WiFi data transmission), the decision to keep wired connections was made. When looking into what type of phone would be used for UART communication, familiarity with a programming language (Java) and overall prevalence of Android devices in the current consumer market led us to choose an Android smartphone rather than an iPhone, however, full deployment of the project would incorporate devices of different manufacturers also (Apple, Windows, etc.). The major design limitation in our project that ultimately forced us

to switch from real-time string synthesis in the form of the Karplus-Strong algorithm to utilizing pitch shifted audio files was the amount of memory present in the FPGA. While testing the string synthesis portion of the algorithm, our team realized that there was not enough space to run six of these algorithms in parallel, in fact, only one string could synthesize a note at a time, leading us to utilize a different method of sound production.

Economic and Cost Constraints

The economic concerns with our project mainly centers around picking appropriately priced sensors that are effective enough to work within the scope of our project. With regards to the smartphone application and the myRIO we used to process the data on, we believe it is a fair assumption to assume that any user would have access to a smartphone to be able to run our application and if not, simple Android smartphones are available for relatively cheap (< \$25). Additionally, the myRIO was provided to us by National Instruments, and more economically available boards such as Arduinos (~ \$20) would be capable of running our software (the only constraint being available memory space) and the porting of LabVIEW code to work with a different board. With regards to the sensors used, our distance sensor and accelerometer were cost effective, totaling less than a combined \$35, however improvements can definitely be made in these areas as purchasing a more effective distance sensor would enable the user to consistently play frets outside of the top three and with more accuracy.

External Standards

As mentioned above, LKFS will be introduced and measured to ensure the safety of the musician and people around them. We may also consider workplace safe noise standards of 90dbA as defined by OSHA(Occupational Safety and Health Administration) [14].

For communication protocols, we will be utilizing a UART between the Android phone and the myRIO, as well as I2C to receive data from our distance sensor. The Android application can configure the USB-C port to act as a UART Interface through software [5]. Both UART and I2C options are built into the myRIO and can be utilized using LabView [15]. With regards to the specific parameters of transmission, UART standard protocol was adhered to with a specified baud rate (9600 bits/sec.), 1 stop bit, 0 parity bits, and 8 data bits being transmitted held constant through both the receiver (myRIO) and transmitter (Android smartphone).

IPC Standards guided the PCB design process. IPC-2221 draws out the standards for PCB trace widths which limits the spacing between traces to be greater than 0.05mm. This standard was held throughout the board design process. The myRIO limits the current through the 5V pin to be 100mA. With that in mind, IPC-2221 limits trace widths based on the max current through the trace and the temperature changes above ambient temp. Based on these metrics the minimum trace width is 0.1mm. The thinnest trace width used in the design was 0.254mm which is well within the minimum value. IPC-7721 maps the correct use of jumper wires in a product. This design fills the requirement with covered jumper wires that end in soldered through hole connectors. This complies with the IPC standard.

Tools Employed

With regards to the note selection portion of the project, the Android application was written using Java and XML in Android Studio. While Android Studio provides some flexibility in the form of dragging-and-dropping components to aid in the design of the user interface, the majority of this was done using XML due to specific positioning requirements. Technical development with regard to writing Java code followed conventional project structure as we promoted high code cohesion and low coupling practices, clearly separating code responsible for tasks such as obtaining user permission to transmit data, touch gesture recognition, and the transmission of data. The technical implementation of handling continuous gesture recognition and transmission of data involved learning more about concurrency, specifically how to effectively use locks and thread-safe data structures so that the two processes running in parallel would not disrupt each other.

In order to prototype and design our PCB layouts the National Instruments circuit design suite were used, specifically Multisim and Ultiboard. Both tools played a major role in routing the myRIO device to the accelerometer, distance sensor, and UART cable, as well as in creating the individual PCBs for those devices. Each of us had experience with these tools through prior classes, however we had to learn how to create footprints for new parts not in the database for this project. The FreeDFM website was also a tool employed to ensure the PCBs we created were manufacturable.

The myRIO programming was done in LabVIEW using both the real-time CPU and FPGA targets. The real-time CPU is the default target environment with lots of memory, while the FPGA target is used to perform high-throughput operations but with limited resources. As none of us had prior experience with LabVIEW before, we had to learn from scratch. Some notable challenges include reconfiguring the project structure and changing blocks when switching to FPGA target programming. With the version of the project utilizing the Karplus Strong algorithm, the UART communication with the phone and distance sensor measurements were done in the real-time target and the Karplus Strong and accelerometer board detection was done in the FPGA target. In our final code using the prerecorded soundwave library we ran the entire project in the real-time target.

Ethical, Social, and Economic Concerns

Human-facing concerns regarding our project were considered briefly with the initial design and layout of our proposal, specifically regarding noise output and data security. Foremost, with regards to the social aspect of noise output, a social norm against randomly playing music is very evident in public, however, this is somewhat mitigated and countered by street performers who have crowds gather to listen to their creativity. With our design of the project, the audio of the myRIO can be routed to an assortment of output devices, for demonstration purposes we utilize speakers, but in a public social context it would be more appropriate to use headphones. The overall size and maneuverability of our product is not a concern with regards to taking up excessive amounts of space in a social setting; the overall

product was designed with compactness and ease of transport in mind - the application and guitar pick PCB are hand-held devices and the rest of the equipment is stored in a wearable fanny pack.

On a human-to-human basis, we believe our project fits into the same niche that public performers currently occupy as we are simply changing the means of musical instrument sound production, while not taking away from the end product. Furthermore, our project may target audiences not currently hit by traditional music instruments, as a production-scale deployment of our device would involve an application-specific processor that would drive the economic cost to be on levels comparable to physical instruments. Namely, while our prototype uses an expensive myRIO, our processor simply needs to be able to perform a small quantity of the operations a myRIO is capable of and with the ability for the Karplus-Strong algorithm to manipulate sound output to sound like a multitude of instruments, we are capitalizing on in a broader market than a single specified instrument may.

Environmental Impact

The use of a reusable battery pack makes our project more sustainable, however this benefit must be weighed with the environmental impact of the production of that battery pack. The production of lithium ion battery packs impacts the environment in many different ways. The mining of lithium uses fresh water in regions where water is scarce, and chemical leakages at processing plants have a history of contaminating drinking water and harming the ecosystems of marine wildlife [8]. Additional environmental impact concerns the resources involved with production of the electronic devices used in our project - however we believe the longevity of these devices (smartphones, processors such as myRIOs, and off-the-shelf sensors) offer a compelling argument against physical instruments. Namely, physical instruments contribute to overharvesting efforts in forests with regards to the quality and quantity of different timbers that compose instruments such as guitars. Additionally, the adaptability of a future iteration of our project to allow a user to set what stringed instrument output they prefer (e.g. guitar, ukulele, etc.) will further aid in counter-deforestation efforts.

Sustainability

Alongside the use of consumer electronics such as smartphones, our project was designed with portability in mind, and as such we feel that the overall sustainability measures we have taken are reasonable. Specifically, the software we are using, as discussed prior, is able to be modified to output various instrument sounds, extending longevity efforts when compared to needing to continuously tune and buy instruments as older ones decay. Furthermore, in travel scenarios our project is easily disassembled and able to be put back together, leading to fewer risk associated with damaging or losing larger instruments such as cellos through air travel. Additionally, the battery we are using is rechargeable and the sensors we are using are both relatively cheap, durable, and have lifespans of several years.

Health and Safety

There are three major safety concerns associated with our project. The first is ensuring that the output of our speakers is not too loud, as to not damage the ears of the musician and people around them. According to LKFS (Loudness, K-weighted, relative to Full Scale), a loudness standard for audio levels normalization, because our output music is broadcast through a speaker, we will limit its loudness to -24 LKFS max integrated [9]. To test the final output loudness, we can use Loudness Meters like Youlean Loudness Meters to test and compress the notes [10].

Another danger associated with our project lies in the infrared distance sensor. Although the planned laser is only class 1 which are supposed to be harmless, too much exposure to infrared waves can damage the lens and cornea of the eye, and as such appropriate precautions should be taken with distance sensing components [11].

Lastly, when using a lithium ion battery, thermal runaway is a major concern. Thermal runaway occurs when lithium ion batteries are damaged or shorted. In the case of thermal runaway, the battery will begin to overheat and at the worst case the battery can be explosive[12]. Considering the hazards and conditions of thermal runaway in lithium ion batteries they should be treated with caution and handled with care. That being said, with the 6800mAh batteries we are using, thermal runaway will not pose significant danger on those using the mechanism. Furthermore, the battery includes its own overcharge and discharge protection[13].

Manufacturability

The current state of the wiring is neither neat, nor stable. A few design changes could be made to create better wiring and prevent loss of contact. This includes soldering connector wires directly into the main PCB and using sleeving for aesthetics and organization. Wires would not be inserted into header boards once manufacturable. Furthermore, the processor chip will replace the myRIO and be integrated on the main PCB. Lastly, the device would be housed in a waist pouch with the laser sensor integrated more stable than it was when it was sewn onto the side of the fanny pack. The PCB can be easily manufactured in bulk as most of the components are surface mount and through hole parts. The components are spaced well and all parts are manageable sizes to solder. The accelerometer may be a little small, but is easily integrated with flux and solder paste. As of the design now, there are a few design flaws that would need to be fixed like removing the RC filter at the output of the pushbutton and adding the correct biasing resistor for push button switch active high functionality.

One alternative we considered is to use 18 capacitive sensors instead of the mobile App to capture the frets. This design would require additional PCB boards for the capacitive sensors and their auxiliary filter systems which will cause the board layout to become overly complicated. Moreover, transmitting data to myRIO would require either 18+ parallel wires, or a

dedicated processor to consolidate the state of the capacitive touch sensors. These additional systems would increase the cost and introduce possible manufacturing uncertainties.

On the other hand, using cell phones to capture frets does not require as much additional hardware. As most people have their own cell phones, the whole process can be reduced to simply downloading the App and connecting to the myRIO via a USB-C cable and a USB-to-TTL converter cable [7]. This process is more straight-forward and cost efficient. Moreover, the LCD screen on the phone is more scalable than capacitive sensors and allows us to introduce new features like lighting up the screen on user presses in our future improvements. To allow multiple simultaneous notes, the design with 18 capacitive sensors requires filtering and transmitting mechanism for each sensor. It also does not provide a direct way for the user to visualize the notes when playing, unless placing LEDs on top of each capacitive sensor. On the other hand, the LCD screen on the mobile phone can allow up to 10 multi-touches at the same time and lighting up the notes when playing requires only an additional function.

Ethical Issues

The ethical issues surrounding our project are two-fold, namely the level of audio output and the security of the overall project. With regards to the latter, security is a concern as this device may be used in both public and private settings; ensuring that proper measures are in place such that data communication maintains integrity and only authorized users can access the data being communicated are relevant concerns. With regards to the Android application, when a user connects a USB-to-TTL cable in order to facilitate data transfer between the application and myRIO, the application prompts the user to explicitly express permission for USB serial communication. Additionally, the application only allows only one connection to be open at a time, with the user needing to establish permission if the cable is disconnected - in the event that permission is not granted, the application will not allow any data to be communicated. Additionally, all communication from the sensors (distance & accelerometer), as well as the application is routed strictly through wired connections, removing the possibility for interference through the air such as in Bluetooth or other wireless communication techniques. With regards to the audio output and improper decibel levels that may cause either distress and/or actual harm, the output of the myRIO itself may be directed towards a set of headphones when in a public setting, versus playing through speaker output when in a private or performance setting.

Intellectual Property Issues

US Patent #9666173 [17] discloses a method for playing a virtual musical instrument to determine an input tool when a gesture (touch input, motion recognition, etc.) is generated and provide an adaptive sound output change based on the input tool, with the use of an electronic device to support this objective. The patent claims several methods for this communication - all are dependent claims that cite a Korean patent (Serial number 10-2015-0113725), and we will discuss them and their potential overlap with our project as follows.

The first method that is claimed specifically involves displaying musical instrument shapes on the display of the electronic device, and when a touch input is received through the

touch screen display, the corresponding sound data is loaded and the output is processed through a speaker or other sound interface. The electronic device that is provided is also disclosed under the patent and is claimed to comprise: a touch screen display, at least one speaker or sound interface, at least one processor to connect the touch screen display, and memory to store instructions. We believe our project is different enough than this method although the overall application is the same (utilizing an electronic device to produce a sound akin to creating a virtual musical instrument). Foremost, the electronic device as discussed in the claim is composed of separate touch display and processing units along with containing all the memory to produce the sound output, while our Android application is a central unit that contains a touch display and only carries out a subset of the processing and memory functions. Our application's display is also completely different than the patented claim, as the claim contains visuals of different musical instruments, while our application simply displays a portion of a guitar's neck to register different notes. Additionally, the phone is connected to another processing unit, the myRIO which utilizes the Karplus-Strong algorithm, whereas the claim states that the memory stores sound data. Furthermore, the incorporation of additional elements such as an accelerometer and distance sensor serve to further differentiate our products.

Another method of sound production that is claimed in the same patent involves using gesture recognition technology via a sensor found in the electronic device to output sound; if the gesture made corresponds to a gesturing relating to a musical instrument then that sound is played, otherwise no sound output occurs. We feel that there is a clear distinction between this method and our product, as our primary & sole mean of gesture control is through touch recognition.

Another patent that is of interest is WO2009111815A9 [18], which describes a stringless digital guitar composed of an array of touch sensitive sensors along its neck, as well as a touch and display screen to receive user inputs via a stylus or pick on the body. Additionally, the display may be used to set device settings, with additional features such as a MIDI output port, USB port, and Ethernet port, the latter two being able to be used to power the device as well as communicate with other USB devices or personal computers. The patent itself appears to be an independent claim as it does not rely on previously established guitars that act as MIDI controllers as prior designs utilize strings and vibrations therein to determine pitch amongst other variables relating to sound output. This stringless patent claims a novel design in which the input method (user's fingers or stylus) generates electrical signals that are processed to produce sound output, along with the ability to receive user input to modify the settings of the instrument on the display. It is also specified that the touch switches may report information in the x, y, and z dimensions such that positional and depth information may be used in determining the output sound, along with each individual button being able to be lit up upon touch. Our project design, although similar in that a touch display will be used and absent of any physical strings, differs significantly in overall implementation. Foremost, our apparatus is not consistent with a typical physical guitar in which a physical neck/body exists as with this patent, in addition to the primary usage of our touch display being for note capture vs. aiding with modifying input/output

settings. Additionally, the note capture method for the patent involves an array of touch sensors/buttons that is not found in our project, in addition to the sound output being produced via MIDI files in the patent, whereas our project utilizes the Karplus-Strong algorithm to produce sounds in real-time.

A third patent relevant to our project is WO2009108029A2 [19], which discloses a method and device that embodies a virtual guitar onto a mobile device. The purpose of the application is to display the whole guitar fingerboard within the small display size without any difficulties in playing, while requiring the same technique as playing a real guitar. Using a motion detection sensor (acceleration sensor) the collinear movement of a user's fingers is able to be detected, resulting in the movement of the fingerboard in the same direction and at the same speed as the movement of the user's fingers. Additionally, the application supports multi-touch gesture recognition and appears to offer claims independent of previous patents in the same vein. While our application also mimics a guitar fingerboard, a distinction appears in that our application does not directly take in acceleration data in order to move the fretboard, instead we serve a static fingerboard image and our project instead utilizes an external distance sensor attached to our user in order to determine pitch - this data is routed to the myRIO directly from the sensor. Furthermore, the mechanism by which sound is outputted is not disclosed nor patented, leading us to believe that the distinctions in the function, appearance, and utility of the application and overall project would allow us to reasonably file a claim for a patent without infringing on the property of the prior mentioned patents.

Detailed Technical Description of Project

The overall project works through the combination of an Android smartphone, distance sensor and accelerometer PCBs, a central power PCB, and a myRIO for system integration and sound output. At a high-level, user input in the form of frets played on the phone is continuously sent via UART communication directly to the myRIO, and in conjunction with the fret offset obtained from the distance sensor as well as the accelerometer and push-button input found on the accelerometer PCB, we produce an audio output akin to a guitar chord.

Phone Application

The phone application was developed in Android Studio using Java for business logic and XML to enhance the user interface and appearance of the application. The overall development was split into three portions: the user interface, touch gesture recognition, and piping of the data out of the smartphone and through the USB port connection to the myRIO. With regards to the user interface, the application was developed to mimic an actual guitar fretboard, with three distinct fret sections (the top three frets on a guitar) visible to the user, as seen in Figure 1. Within the application logic, the three distinct frets were given unique id values - fret '1', '2', '3', in addition to each of the six horizontal strings being given unique identification names - string '1', '2', '3', '4', '5', '6'. The six strings were given distinct colors in order to enable visual feedback to the user with regards to which set of strings were being

strummed at a given time. The decision to have a static set of three frets on the application instead of creating a 'scrollable' interface which would display more frets was made to simplify the learning curve of the application, as well as properly emulate an air guitar with the distance sensor serving as a differentiator for different frets.

Additionally, when a user interacts with the application, a circle is registered in the location of the touch gesture press (to provide more visual feedback) and the corresponding x and y locations are saved in a thread-safe data structure - a concurrent hashmap. Touch gesture recognition was implemented using Android's 'MotionEvent' class [4]. Threads are independent sections of the program that may run in parallel, and two threads were utilized extensively in this application - one for continuous touch gesture recognition and another for sending data via the USB connection. Whenever the content of the hashmap changed (a touch gesture was either added or removed to the map based on finger press/release), a lock was placed before any modifications occurred. This lock signifies to the thread sending data out of the application that the hashmap is currently being modified and as such, the next batch of data being streamed out will be composed of the new data found in the modified hashmap. However, the current data being piped out of the application will be safely composed of the old map's contents without any worry of compromised data integrity. The lock is then removed from the hashmap once mutations to its entries have finished and this interaction between the two threads runs continuously while the application is open.

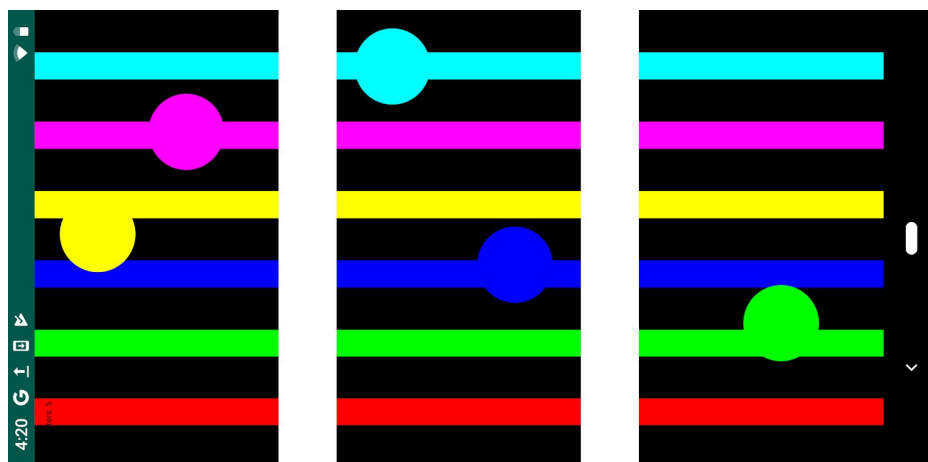


Figure 1: Android Application UI

With regards to the contents of the hashmap itself, some post-processing must be done on the raw data captured by the application in order to determine the frets being played. Raw data captured consists of the x and y coordinates of a finger press and this data is used to determine the closest string and what fret section the touch corresponds to. Furthermore, as with a real guitar, if the same string is touched within two or more different fret sections, the lower fret and string combination is the only data sent from this string as pressing a lower fret effectively 'closes' an upper fret. The data actually being transmitted from the application was packaged in a

UART data frame with the following parameters: baud rate of 9600 bits/second, 8 data bits transmitted at a time, 1 stop bit is transmitted, and 0 parity bits are used [5]. As there are six strings, six transmissions are necessary to communicate information from each string and fret combination, with a leading 'x' byte serving as a divider in the LabVIEW code to differentiate between transmissions. The first actual bit of transmission always corresponds to the first string, with the second bit corresponding to the second string, etc. up to the sixth string. The content of the transmission, as described prior, relates to the fret being pressed on the particular string - if no fret is being pressed then a '0' is transmitted, otherwise a '1', '2', or '3' is transmitted relating to the lowest pressed fret for that string. Figure 2 shows the LabVIEW code for reading inputs from the UART, in which the data is received on the UART pin on the myRIO as a string that is split into its individual characters for each string of the guitar.

In addition to the concurrent thread operations of the application, explicit permission is required to send data via the USB port of an Android smartphone. As such, upon application launch a dialog box appears to notify the user of additional permissions the program needs (*Intent_Action_Grant_USB*), and if the user approves of these elevated permissions then the app functions as described above; however, if a user declines to grant these permissions then the app's interface and touch gesture recognition system will continue to function, however no data will be transmitted.

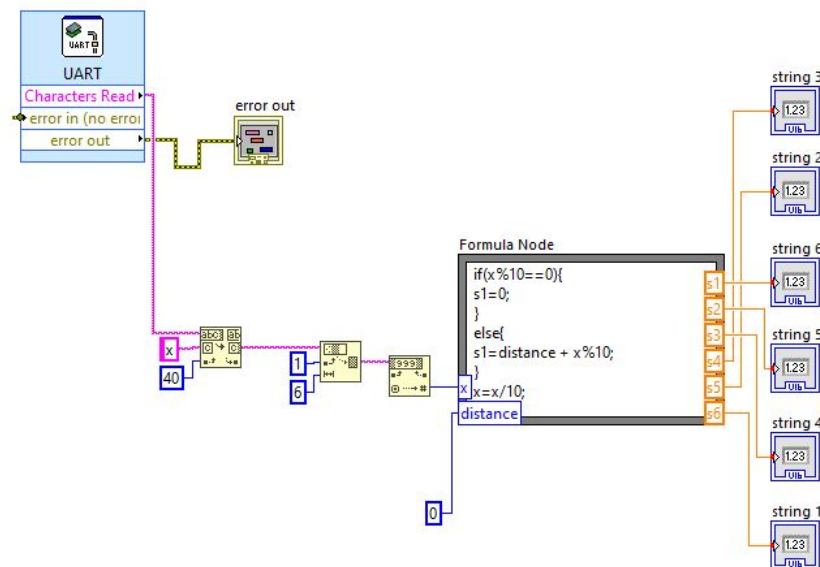


Figure 2. Android Application UART LabVIEW Block Diagram
Waveform Synthesis

In order to simulate the actual guitar sound, a waveform-synthesis technique, Karplus Strong Algorithm, is developed in LabVIEW 2019 with myRIO template and high-throughput

toolkit. Karplus Strong Algorithm modifies the wavetable during sampling by averaging values that are generated randomly in the buffer. In our design, we average the last two values in the wavetable using the formula $Y_t = \frac{1}{2}(Y_{t-p} + Y_{t-p-1})$, where t is the current index and p is the offset. The frequency of the note is determined by the length of the wavetable. For example, to generate a 1-second note at 55 Hz given the sampling rate of 8000 Hz, the size of the wavetable is $\frac{\text{sampling rate}}{\text{frequency}} = 146$. The wavetable is initially filled up with randomly generated values ranging from $(- \text{output voltage})$ to $(+ \text{output voltage})$. By looping through the averaging algorithm 8000 times, the sampling voltage gradually decreases and thus simulate the guitar pluck.

The initial implementation of the Karplus Strong Algorithm ran on the myRIO real-time (default) target as shown in Figure 3. Although the design could produce the expected waveform visually in LabVIEW Chart, the sound was muffled and unpleasant to the ear. This was due to issues in timing, as the real-time target was either not able to perform calculations fast enough or not able to change the output voltage at precise time increments. In order to meet timing constraints, the process was moved to the FPGA target, utilizing the high-throughout module and the on-board 40MHz clock, allowing for very fast operations.

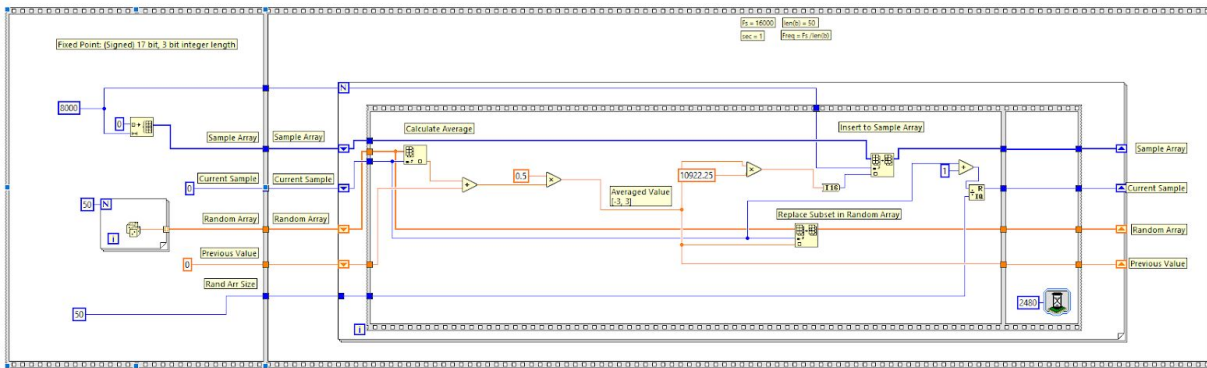


Figure 3. Karplus Strong Algorithm on myRIO real-time target

Figure 4 displays the Karplus Strong Algorithm implemented on the myRIO FPGA target in LabVIEW. In order to create a 1-second waveform with decent sound quality, the sampling rate should be greater than 5000Hz. By experimenting several sampling rates, we decided that 8000Hz is a desired sampling rate as it saves resources without damping the sound quality too much. Because the system utilizes the 40MHz clock on the FPGA target, a sampling rate of 8000Hz leads to a clock waiting time of 5000 clock ticks (1 tick = 25 ns). This produced the correct waveform and sound, a pleasant sound resembling the pluck of a guitar string.

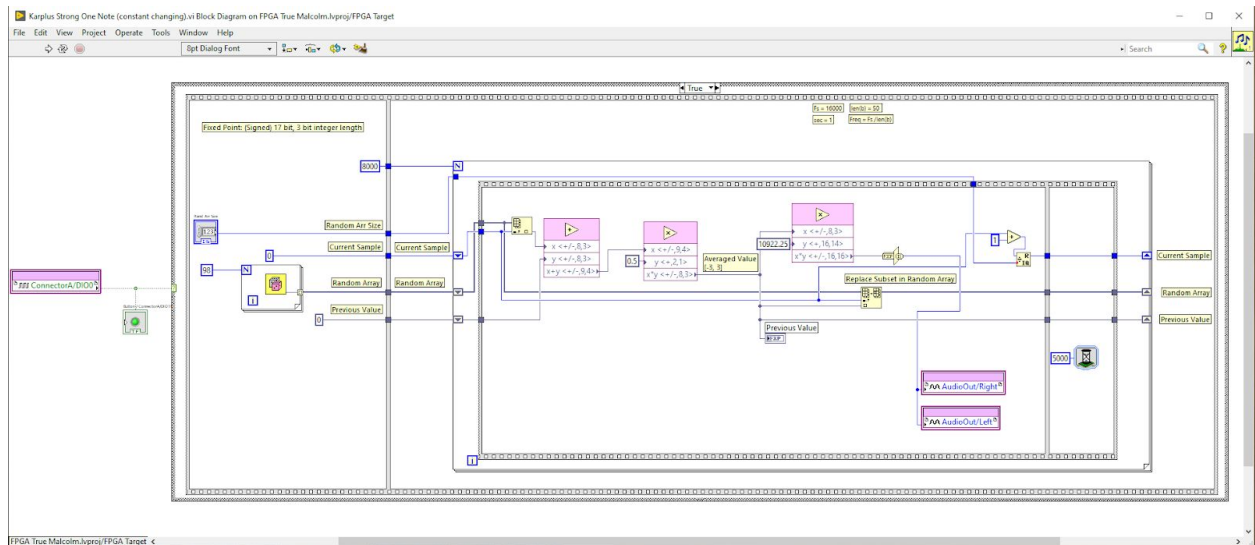


Figure 4. Karplus Strong Algorithm on myRIO FPGA target

Because the FPGA target does not support floating-point numbers calculation, the built-in subVI for random number generation that outputs a floating-point number between 0 and 1 cannot be utilized in the FPGA design. Hence, in order to fill the wavetable with random numbers, a linear feedback shift register (LFSR) algorithm is constructed as a subVI. This online-obtained LFSP algorithm outputs a random integer from 0 to 65535 as shown in Figure 5. The output integer is scaled down to a fixed-point number (FPN) between 0 and 3 as shown in Figure 6.

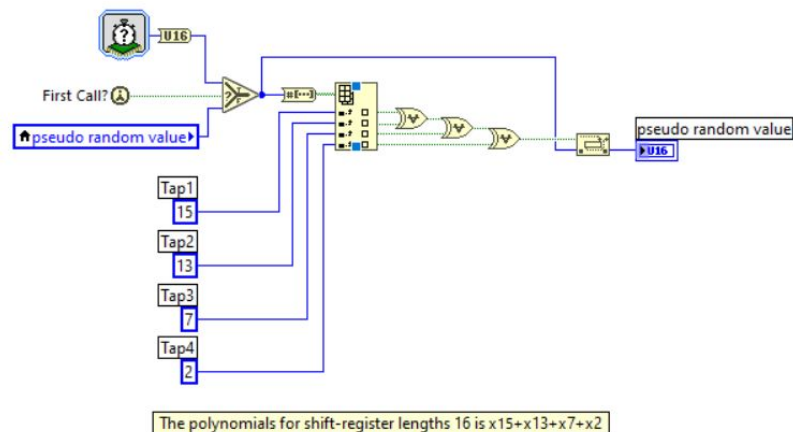


Figure 5. LFSR random number generator

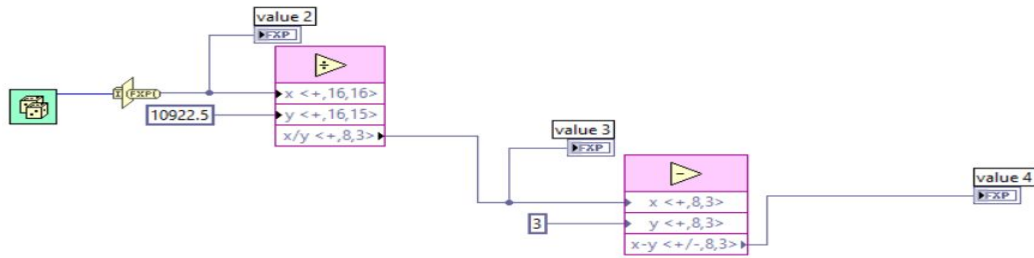


Figure 6. Random number generator scaling

Figure 7 shows the integration of one-note Karplus Strong Algorithm with the whole system (UART, distance sensor, and frequency mapping). The frequency mapping subVI takes inputs from UART and distance sensor subVIs and outputs wavetable sizes respectively. Only one non-zero wavetable size is passed to the Karplus Strong Algorithm subVI.

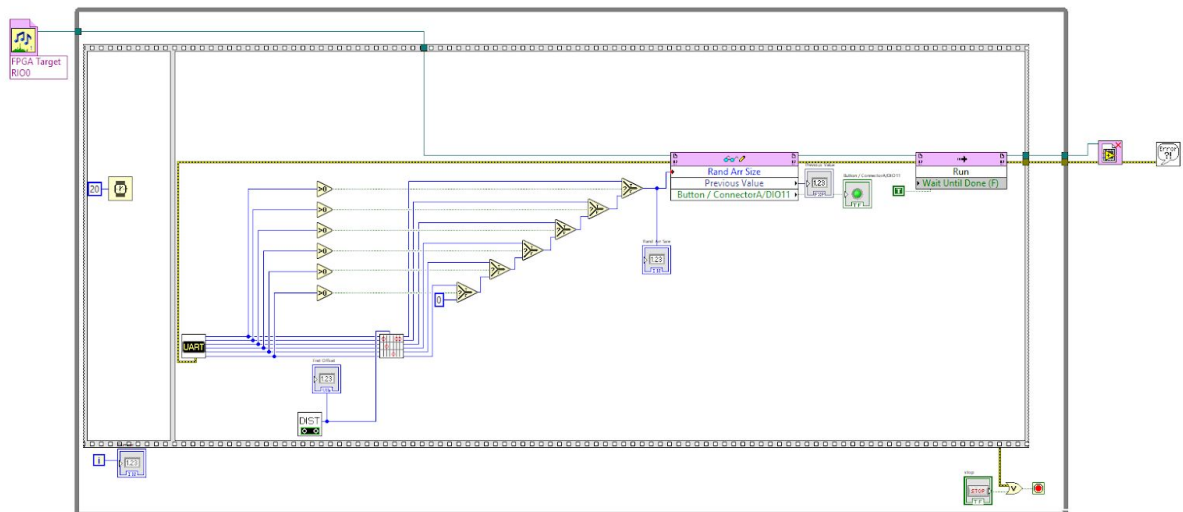


Figure 7. FPGA target system integration

By modifying the wavetable size, the Karplus Strong Algorithm implemented on the FPGA target could output the 1-second waveform at different frequencies. However, this implementation consumed too much of the FPGA resources with just one string. As shown in Figure 8, a single string implementation of the Karplus Strong algorithm uses 88% of total slices, which measure the reconfigurable logic blocks inside the FPGA. Attempting to run just two out of the six strings needed caused the FPGA to run out of resources (The block diagram for running two notes are shown in Figure 9).

Device Utilization	Used	Total	Percent
Total Slices	3884	4400	88.3
Slice Registers	11496	35200	32.7
Slice LUTs	10929	17600	62.1
Block RAMs	5	60	8.3
DSP48s	1	80	1.2

Figure 8: FPGA Utilization for single string Karplus Strong implementation

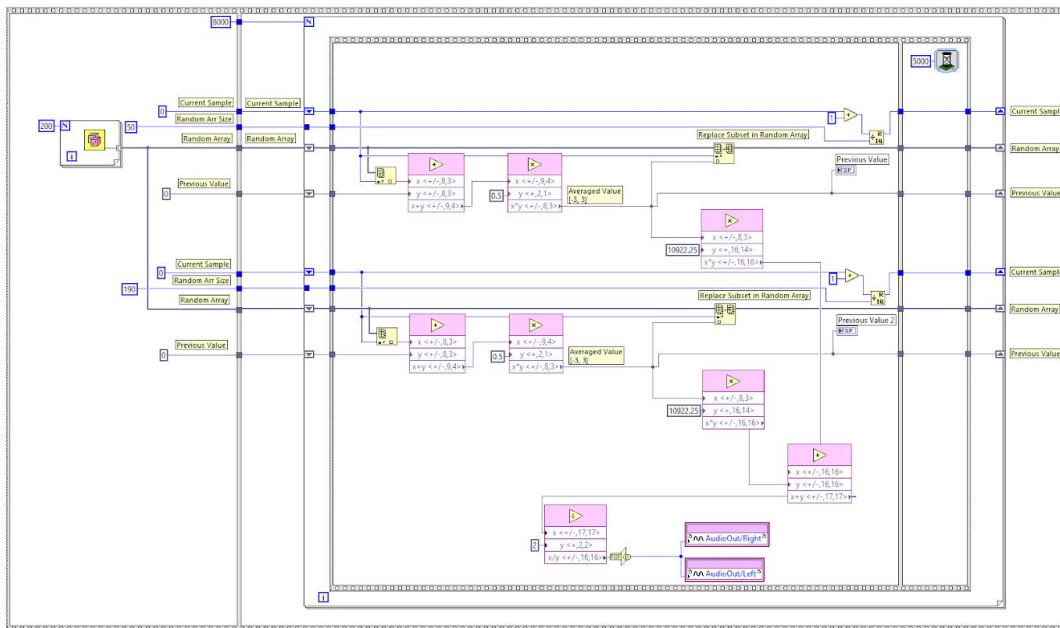


Figure 9. Karplus Strong Algorithm on myRIO FPGA target for two notes

To reduce the resources used on the FPGA target, we tried to generate sample arrays of all notes when starting the system on the myRIO real-time target. The myRIO real-time target will then determine what arrays to pass on to the FPGA target based on the inputs from UART and the distance sensor. The output arrays are fed into FPGA host-to-target FIFOs without overusing the FPGA resources. However, the FPGA target conflicts with the input pins on the real-time target and causes the analog and digital input pins for the distance sensor and push button unable be recognized by the real-time myRIO target.

After struggling to sufficiently optimize the implementation, we decided to switch from the Karplus Strong Algorithm to using pre-recorded sound waves. The sound waves could be found online and then clipped/adjusted in Audacity to fit our use case. In order to save resources and speed up the program, we use pre-recorded sound waves of 1 second and 16000Hz sampling frequency. Figure 10 shows the final Karplus Strong Algorithm with pre-recorded sound waves. This was able to replace the Karplus Strong algorithm without any major change to other subVIs.

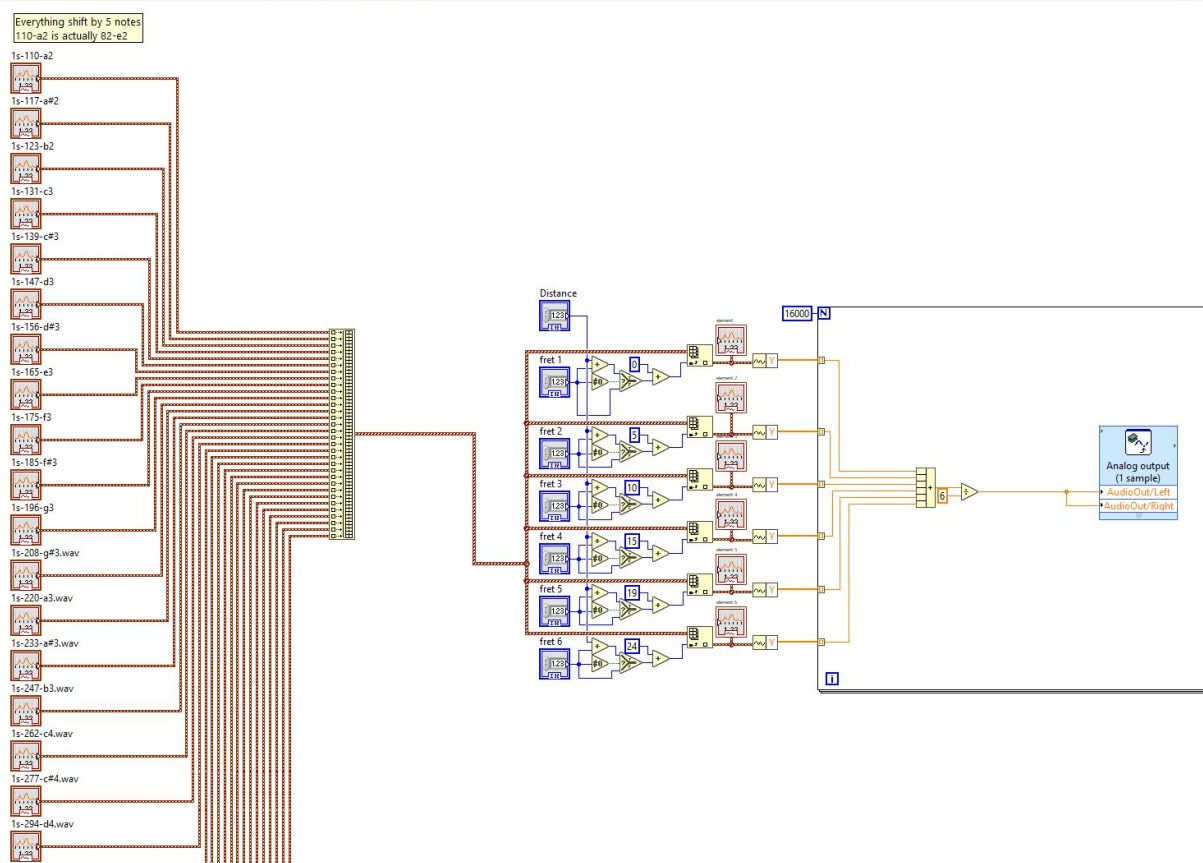


Figure 10. Final Karplus Strong Algorithm with pre-recorded sound waves
Frequency Mapping

The frequency lookup table is used to map the finger positions from the phone and fret number from the distance sensor to the frequency of each string. A mapping of finger position to frequency is seen in Figure 11. Though this shows 120 finger positions, many of the frequencies are repeated. For example, string 1 fret 6 through 20 is the same as string 2 fret 1 through 15. In total there are just 45 frequencies, so each string can index the same table with an offset depending on the string. The LabVIEW VI for this frequency lookup table is seen in Figure 12. For each string, the frequency table index is the fret position from the phone, fret offset from the distance sensor, and the constant index offset depending on the string. When the phone indicates that a string is not pressed, the index is just the offset, and corresponds to the open note for that string. When switching to pre-recorded soundwaves, the frequency values are replaced with the full soundwave of that note.

Fret	String Number (Note)					
	1 (E)	2 (A)	3 (D)	4 (G)	5 (B)	6 (E)
1	82.41	110.00	146.83	196.00	246.94	329.63
2	87.31	116.54	155.56	207.65	261.63	349.23
3	92.50	123.47	164.81	220.00	277.18	369.99
4	98.00	130.81	174.61	233.08	293.66	392.00
5	103.83	138.59	185.00	246.94	311.13	415.30
6	110.00	146.83	196.00	261.63	329.63	440.00
7	116.54	155.56	207.65	277.18	349.23	466.16
8	123.47	164.81	220.00	293.66	369.99	493.88

Figure 11: Finger position to frequency table

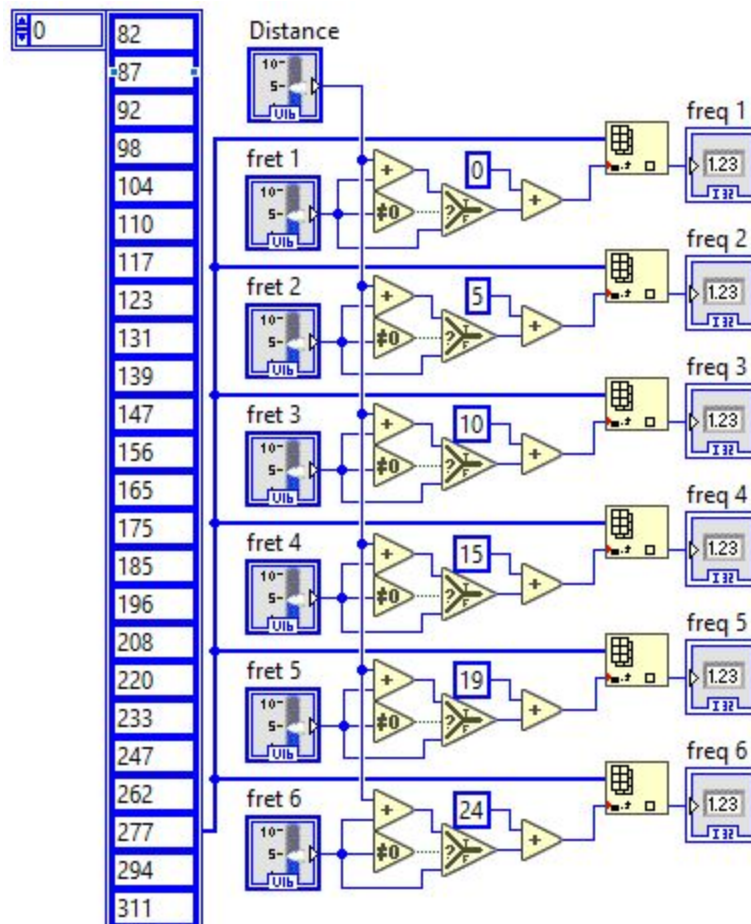


Figure 12: Fret to frequency LabVIEW VI

Accelerometer Board

The accelerometer board is the second large-scale component in our project, shown in Figure 13, consists of two main components: the accelerometer and pushbutton. The accelerometer chosen is an ADXL335BCPZ-RL7, which input 3.3V and outputs an analog voltage signal which sensitive to 300mV per 1g of force. The 0g operating point is 1.5V with a

Figure 13: Accelerometer Multisim Schematic

The pushbutton will send a 3.3V signal to the myRio enabling the strumming function. If the button is pressed, then the guitar is being played, otherwise, the guitar will not be played. The pushbutton was added to the accelerometer board because guitarist will often use strumming patterns of up and down strums. If the button were not there the device would play every time the accelerometer board moves. We chose a button that is fairly flat with a height of 2mm so the air guitarist can hold the accelerometer board and press the button with little difficulty, as it has an operating force of 200gf. When operating a push button, it is important to understand the state of the voltage that is switched on and off. The myRIO DIO ports are set to be active high at 3.3V with a 40kOhm pullup resistor. In order for the button to function the way it was designed to be was to create a voltage divider at that port to lower the 3.3V active high down to the order of 100mV level. a 3.3kOhm resistor was added to create the voltage divider. In theory this drop the active high voltage at DIO down by an order of $3.3 \cdot (3.3/43.3) = 251\text{mV}$ when the button is not pressed. With the button pressed DIO is set to 3.3V. Thus giving the user full capability to toggle the strumming board on and off. The new configuration is not the traditional way of using a switch but the options were limited once testing began. Another aspect of the push button design that was not thought out well enough was the RC filter added to the output of the switch. This was added in theory to avoid debounce. However, all this did was slow the switching time by charging the large 470uF capacitor. In the final design, this capacitor and 100Ohm resistor were not soldered onto the board to avoid the RC time constant, 0.047sec, as shown in Figure 14.

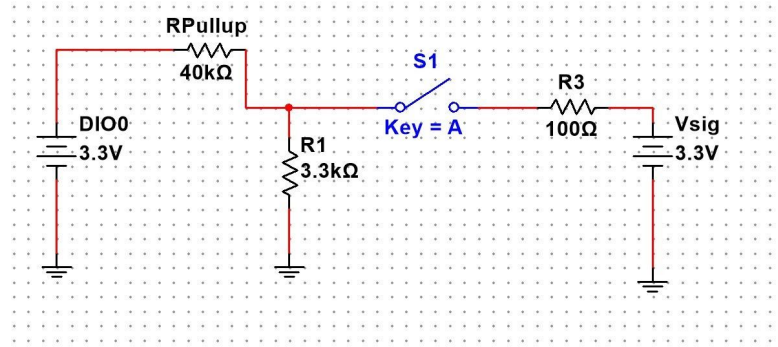


Figure 14: Pullup Resistor Accelerometer Modification

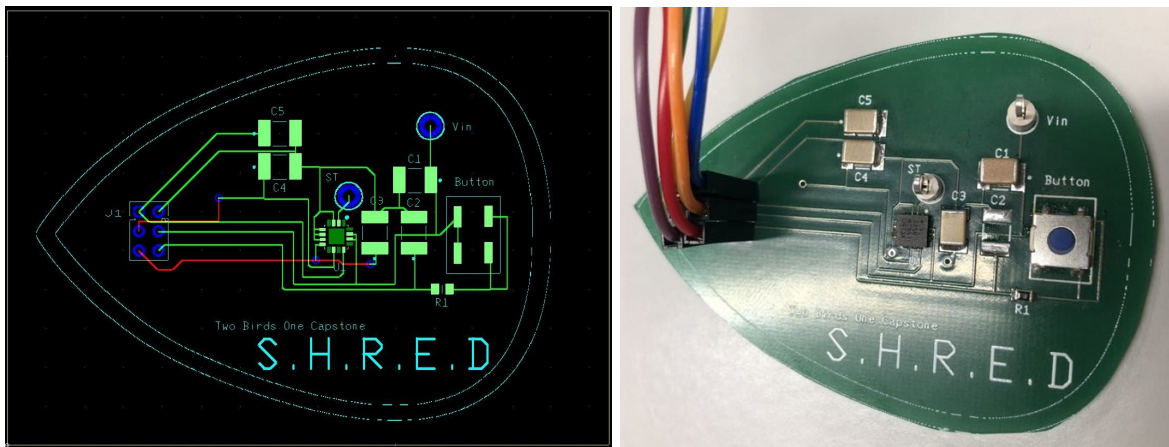


Figure 15: Accelerometer Ultiboard and Final PCB

Distance Sensor

The distance sensing board is a small external PCB shown in Figure 16 specifically designed for use of the Sharp GP2Y0A60SZLF sensor that connects to the main PCB through four connections: ground, 5V supply, enable, and an output voltage. The board was designed with multiple bypass capacitors of values 0.1uF, 4.7uF, and two 10uF, for the voltage supply as recommended in a data sheet for the product given by Pololu Electronics, and the enable was wired with a 10kOhm pullup resistor to VCC. As the device always needed to be enabled, the enable pin was open to ground when connected to the main PCB. The selection pin was wired directly to ground through a 0Ohm resistor such that the device operated off of a 5V power supply rather than a 3.3V power supply, as the range of output voltage values given by the sensor would be wider and allow for easier differentiation between different defined fret distances. The fret differentiation was determined by the voltage output, with different voltage spacings defined for each fret as the relationship between the voltage and distance was nonlinear.

The distance sensor utilizes an infrared laser, it operates best when the object is directly aligned with it. That being said, the fanny pack should be worn at the middle right of the person's waist. When the user moves the phone closer to the fanny pack, the shorter distance indicates a higher notes, and vice versa. The laser and sensor uses time of flight to measure the

distance. Time of flight measures phase shift changes between incident and reflected infrared signals [6]. The board layout was created in ultiboard to take up as little space as possible, as the distance sensor was to be mounted on the fanny pack.

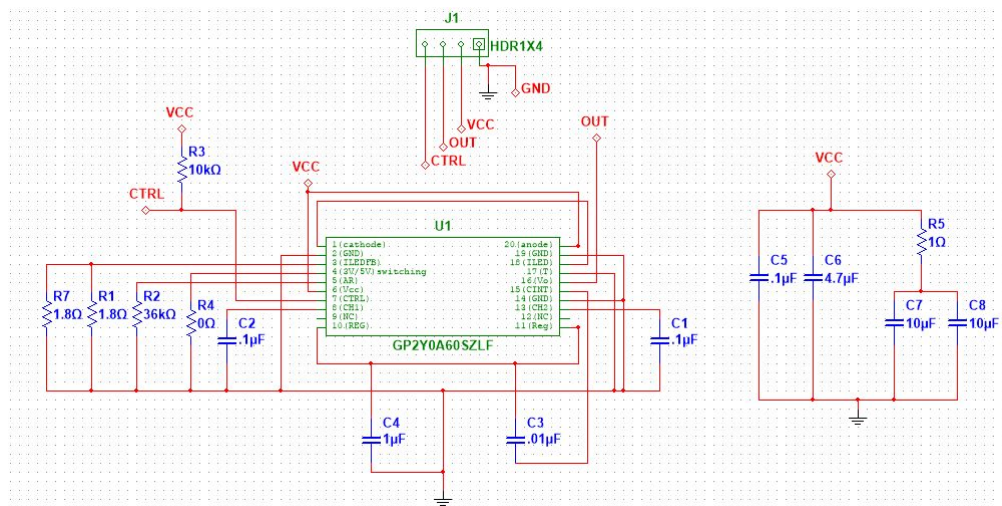


Figure 16: Distance Sensor Multisim Schematic

Figure 17 shows the LabVIEW subVI for mapping the distance reading to fret offset. The output voltage from the distance sensing board is connected to analog pin A/A10 which is pin 3 section A in myRIO. The floating-point number read from pin A/A10 is passed on to a built-in subVI called AC & DC Estimator PtByPt which averages values and eliminates noises from analog readings. The modified value is then scaled up by 100 and is fed into a case structure which maps the distance range to a fret offset number. The default fret offset is 0 which means there is no offset as the value is between 0 and 155. If the value is larger than 155, the offset will be 1. Further cases could be added, however differentiating between smaller distances was somewhat unreliable with the sensor used.

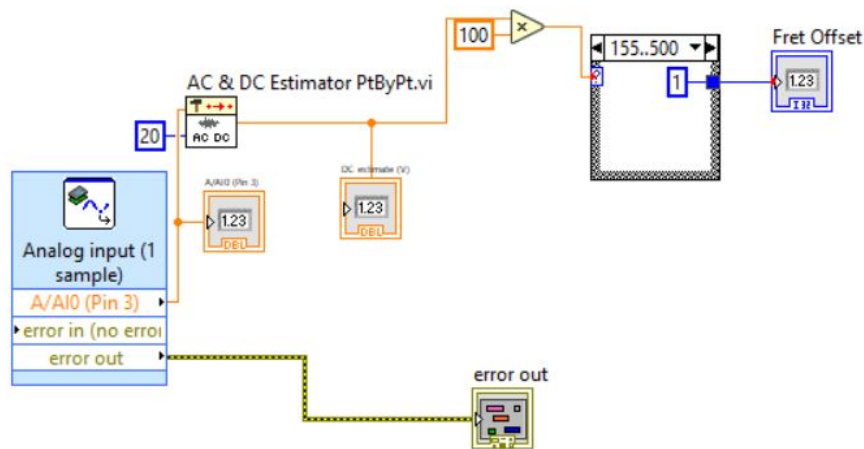


Figure 17. Distance to fret LabVIEW subVI

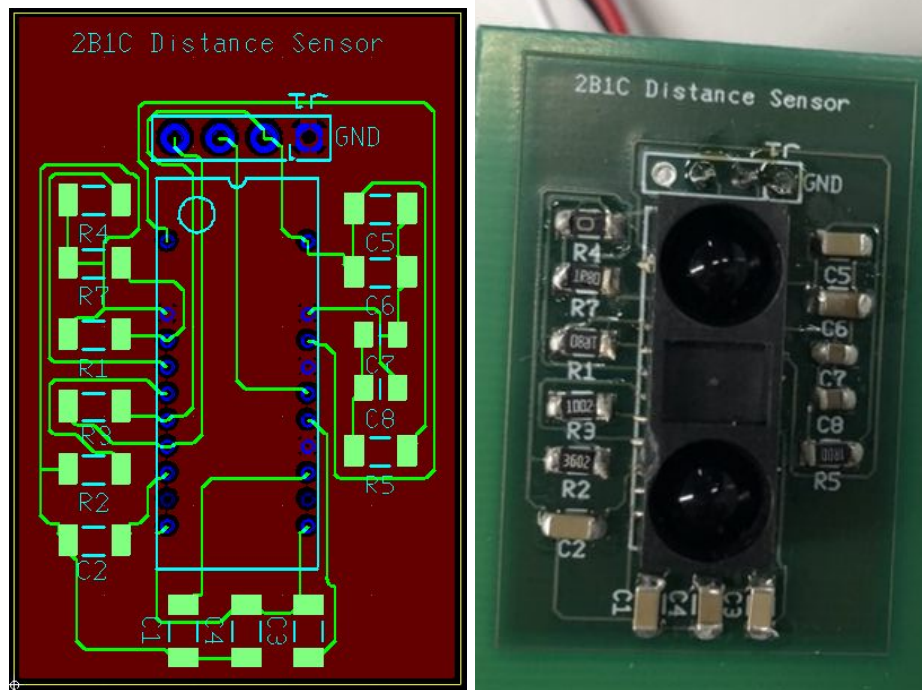


Figure 18: Distance Sensor Ultiboard and Final PCB

Power Source

The main source of power for this system is derived from the MyRIO wall charger port. However, the device is capable of running off of a 12V battery. With typical idle power draw of 2.6W and max at 14W, a battery with 6Ah capacity is able to drive the device for over 4hrs assuming we are driving the myRIO at its maximum power ($10 \times 6\text{Ah} / 14\text{W} = 4.28\text{hrs}$). With the correct power source in place, the myRIO outputs 5V to the main PCB. The main PCB routes 5V to the distance sensor and regulates the 5V down to 3.3V for the accelerometer board. The voltage regulator chosen for this design is a MAX1818EUT33+T. This regulator is commonly used for regulating to 3.3V as it is the default output voltage. That being said, the biasing for this regulator is standard. Following the data sheet, a 1uF bypass capacitor is added to the input and a 3.3uF to the output. Last, a 100kOhm pullup resistor is situated between the input and the output nodes. The remaining space on the main PCB goes towards routing signals between the myRIO and the various data capture peripherals of the Air Guitar.

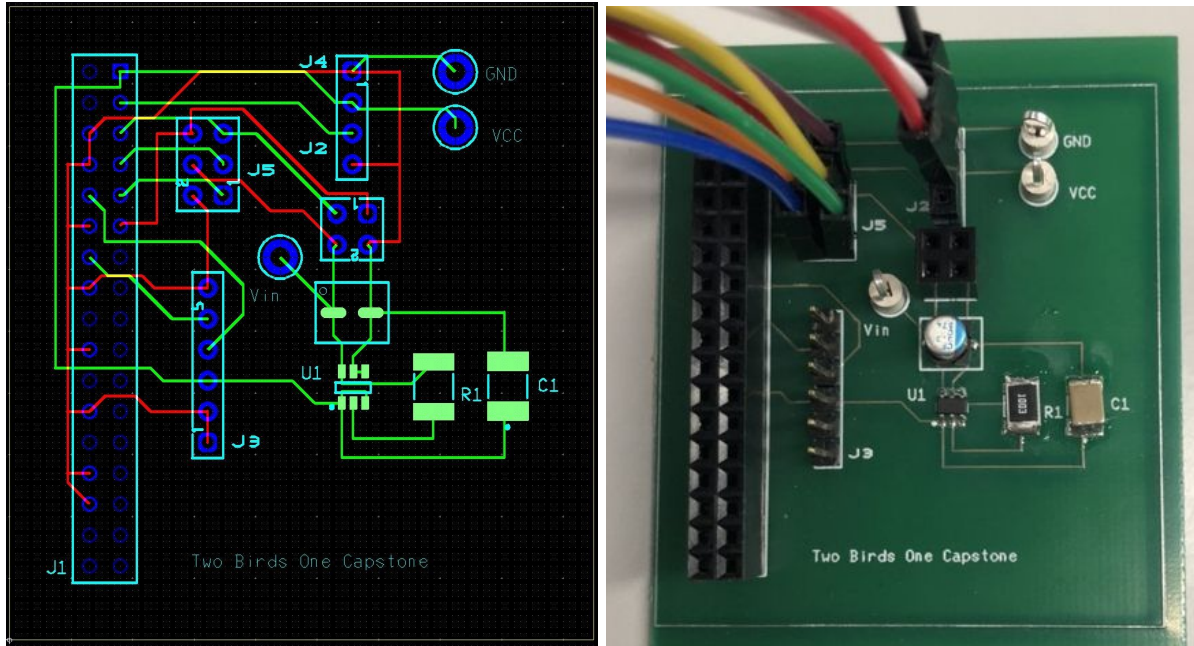


Figure 19: Power and Routing Ultiboard and Final PCB

System Integration

The final LabVIEW integration code is shown in Figure 20. The whole system is implemented within a while loop that will only stop if an error message is thrown or the stop button is triggered in the front panel. The UART subVI reads values from the phone application and outputs six integer values indicating which string notes are pressed. At the same time, the distance sensor subVI observes the distance and maps it to a corresponding fret. The push button on the accelerometer board is connected to the digital input pin A/DIO0 (pin 11 section A) of the myRIO. As '1' indicates the button is pressed and the user is strumming and '0' means resting, a button reading of '1' will trigger the frequency mapping subVI which outputs a chord sound.

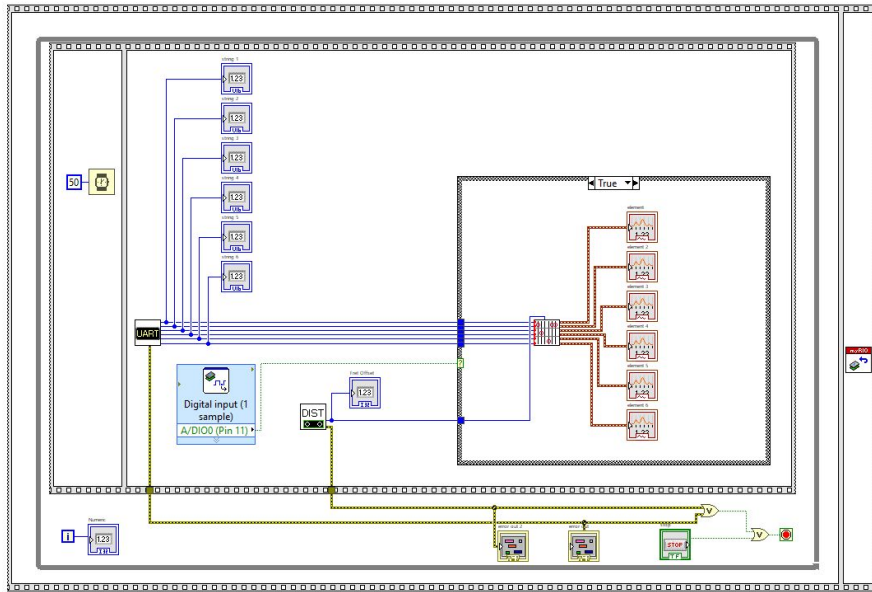


Figure 20. System integration LabVIEW VI block diagram

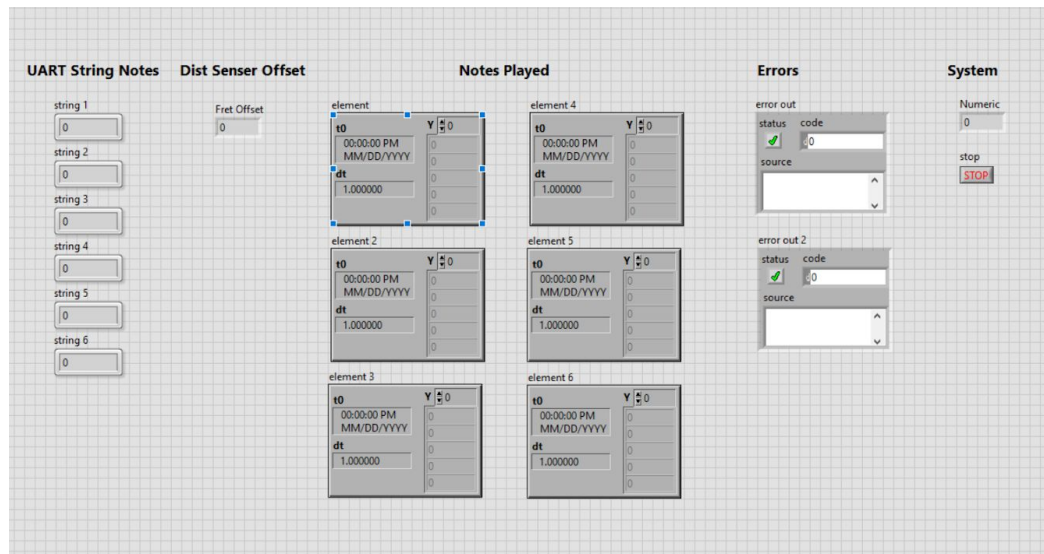


Figure 21. System integration LabVIEW VI front panel

Project Time Line

Overall, the changes between our proposed and final Gantt chart were centered around better parallelization of tasks and extending time out to complete certain tasks such as testing and integration. With these changes, we were able to stay on track and obtain a clearer scope of the project; for example, with the the phone application we extended the total development time and started testing the application sooner than laid out in our first iteration chart. Additionally, items such as communication between the phone and myRIO and integration of the separate LabVIEW programs extended out further than initially thought, due to the initial hurdle of understanding how to effectively program in LabVIEW and the technical development of the Karplus-Strong algorithm. The most important change to our design occurred roughly a week before Capstone Demos, as we had realized there would be no effective way to get multiple strings to produce audio output using the Karplus-Strong algorithm, so we focused our efforts on pitch shifting frequencies and utilizing compressed audio files for sound output - a process which took roughly two days to complete.

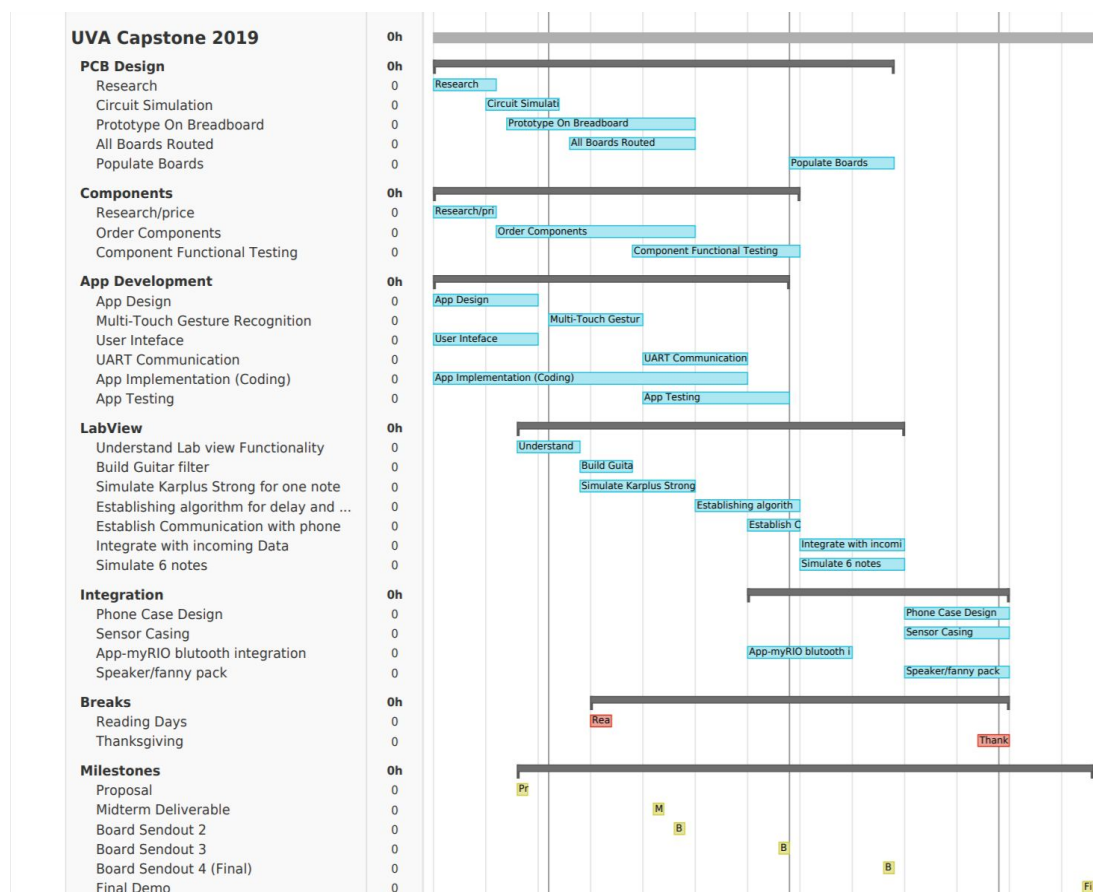


Figure 22: Original Gantt Chart

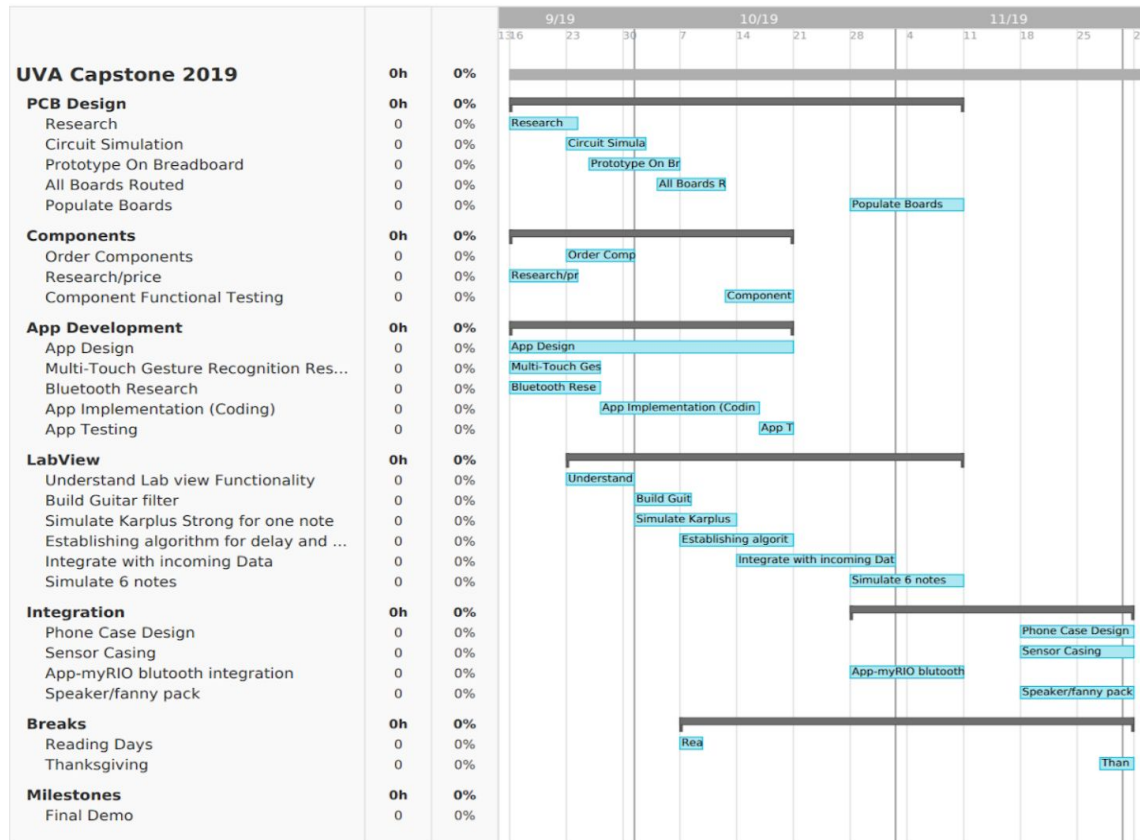


Figure 23: Improved Gantt Chart

On the hardware and component side, we extended component testing, ordering, and board integration deadlines, namely due to the inability to run end-to-end tests until the other portions of the project were completed.

Throughout the project, we effectively parallelized our overarching tasks of smartphone application development, LabVIEW programming, and hardware design. Karan focused primarily on the Android application, taking over the day-to-day design, implementation, and development of the overall application due to prior familiarity with the coding language and development environment. Erik focused on the board design and testing of the distance sensor, alongside LabVIEW programming which transformed the output of the Android application into specific frets and frequencies - integrating information from the distance sensor as well. Jacob worked on the board designs of the accelerometer and the central PCB which links with the myRIO, Android smartphone, distance sensor, and accelerometer. Josie and Hua both worked together to implement the Karplus-Strong algorithm on the myRIO FPGA, as well as integrate the sensor data into the overall LabVIEW project. Serial execution of tasks came towards the end

of the project as the parallized portions wrapped up, making sure step-by-step integration of the different parts functioned as intended, coming together as a coherent final product.

Test Plan

As the project was divided into several subcomponents to be worked on in parallel and then integrated towards the end of individual development lifecycles, we will discuss the testing procedure for each subcomponent separately, with full integration testing coming at the end.

Throughout the development of the Android application, a continuous testing plan was put into place in order to ensure progress in the form of overarching categories: user interface, touch gesture recognition, and data streaming through the USB port. The initial test plan considered the development of the user interface and touch gesture recognition in parallel, as the interface was a static representation of a subsection of a physical guitar fingerboard, while the touch gesture recognition dynamically interacted with the interface. Specific testing into the user interface simply regarded the similarity in replicating a physical fretboard on a smaller device screen, with redesigns due to usability concerning the number of fret sections displayed and color distinctions between strings. The touch gesture recognition portion was initially tested prior to implementation of the data streaming, with initial focus on a single detection point which would then feed into multiple data point detection. Testing of touch gesture detection was fairly intuitive, with albeit significant redesign occuring to visually aid the user when using the application, as colored circles would appear under detection points relating to the closest string. In the same line, multiple point detection followed the initial plan of ensuring that the correct number and color of points under the detection points showed up, in addition to movement across strings dynamically changing the color of the associated circle. Where our initial test plan deviated from the final one was in the development of the data streaming portion of the application, as although the overall testing was the same end-to-end, the technical implementation of the data streaming differed from initial thoughts. Specifically, user permission had to be obtained explicitly to access and communicate via the USB port, in addition to the implementation of separate threads - one thread handled continuous user input while a distinct thread was responsible for data communication. Testing throughout this process had not initially accounted for simultaneous call-and-response actions in which the user input was being modified while data was continuously output, leading to a redesign of the data structures used to store state and memory to be thread-safe and lockable. While the technical implementation of the data streaming and touch gesture recognition was changed throughout the process, the end-to-end testing remained the same in the connection of the Android smartphone to a myRIO using a USB to TTL serial cable, and ensuring that the proper bytes were being read into the LabVIEW program.

For each peripheral component (phone UART, distance sensor, accelerometer board) an isolated test VI was made to sample values. Once those were verified to work as expected and provide expected values, they were made into reusable subVIs, and used in integration tests at each advancing stage of the program workflow. That is, testing integration with the distance

sensor, UART and frequency selection table, then with the Karplus Strong algorithm (running every second), and lastly with accelerometer input acting as a trigger for the Karplus Strong algorithm.

The test plan for the routing PCB was as followed. First, each through hole component was soldered onto the board and inspected visually. If the solder was applied uniformly, the next step was to solder on the surface mount components. These were inspected for good contacts respectively. Once all components were integrated on the board, the DC biasing test began. Applying 5V to the input of the voltage regulator, the board was tested for 3.3V at the output of the regulator as well as on each node 3.3V was associated with. If parameters were within the reasonable range of $\pm 100\text{mV}$ the board was assembled correctly. Moving on to the accelerometer board, the surface mount and through hole components were individually soldered and inspected for good connection. Since this board is connected to the routing PCB, it would be tested both isolated and integrated. Isolated, the board was supplied 3.3V to drive the accelerometer. The accelerometer was measured with an oscilloscope to verify proper biasing and operation. Measurements were made in the X, Y, and Z directions. The switching capabilities of the pushbutton were tested in transient looking for any high frequency oscillations at the switch on and off. With a fully functional push button and accelerometer, the accelerometer board is ready to be integrated with the routing PCB. Once integrated, the same DC transient measurements were made with the components being driven by the regulated 3.3V from the routing PC. Now connected to the myRIO, the metrics for the accelerometer and pushbutton were verified in a LabVIEW file. The distance sensor was tested by first ensuring the 5V power supply was stable with an oscilloscope. The sensor itself was then tested at different distances with a standalone LabVIEW VI, which closely matched the output distance characteristics given on the data sheet for different voltages. For integration, the distance was tested in the overall labview program to ensure that the code was functional in shifting the note down by one fret.

Final Results

Overall, the functionality of our device was in-line with our overall goal at the beginning of the semester, but at various points we fell short of a fully fledged air guitar, whereas in other aspects we exceeded our initial prospects. Specifically, our project does function in that a user can effectively utilize an Android smartphone to mimic a physical guitar and play chords through the phone's interface. This information is continuously streamed out to the myRIO through a UART cable that is connected to the phone's USB port and successful reading & integration of the data in terms of computation occurs on the myRIO. Additionally, we do receive information from the distance sensor corresponding to the fret number (at a specified distance) that the phone is held away from the central fanny pack, however we ran into some implementation issues based on the quality of the sensor chosen. Namely, rather than allowing for full emulation of a guitar's fretboard based on a scale of 0-5 (scaled based on voltage readings from the distance sensor), we found it more effective to use 0 or 1 reading from the sensor in order to make pitch

determinations more consistent given the sensor quality. This signifies that we could simply shift the guitar down a single set of frets, rather than the proposed full fret-board of a guitar. This however ended up not being a huge issue, as a majority of basic chords are played on the first four frets of the guitar. With regards to the accelerometer and push-button PCB, we were able to successfully get data from both devices onto the myRIO, however due to time constraints with proper development of the PCB and the integration of components within LabVIEW, we did not effectively utilize the accelerometer data. As originally planned we hoped to use the accelerometer data to determine the sound level output of the audio, however, we instead simply left this portion of data alone and used the push button for its intended purpose - audio would only sound if the push button was triggered. With further regard to the accelerometer time constraint, our involvement with the Karplus-Strong algorithm resulted in a heavy time commitment and if we had not spent as much time trying to reduce the resources involved in the algorithm we fully believe we could accurately incorporate the accelerometer data. When looking at the LabVIEW integration and sound synthesis code itself, we achieved all our targets as all components of the project were successfully able to be integrated and run on the myRIO. Additionally, with the overall time investment in the Karplus algorithm, we were able to quickly swap to utilizing stored audio files rather than creating real-time synthesis while still maintaining full integration and a realistic guitar sound, the ultimate end-goal of our project.

Costs

The total cost of developing our device was \$431.08. \$221.57 of the total budget went into paying for connectors, and board components. This includes replacement and spare parts for the 6 iterations of the board design. The remaining \$210.00 was the cost of our six board iterations. Furthermore, the myRIO cost was not included. The myRIO costs around \$500, but thanks to the University of Virginia's partnership with National Instruments we were able to loan the microcontroller free of cost. In further applications we would have used an MSP microcontroller which would add around \$3 per unit. If mass produced at 1000 units the total unit cost would be \$44,750.05 and the price per unit would drop to \$44.75. This price was estimated assuming myRIO would be replaced by a MSP430 microprocessor and memory. The cost estimate can be found in the Appendix.

Future Work

Future work regarding our project should consider several aspects of design, namely the smartphone application, distance sensor and accelerometer implementations, and the technical use of the myRIO. With regards to the overall application, a possible extension would be in changing the framework used in the development process; Java is unique to Android development whereas React Native may be used to build both Android and iOS applications, extending the reach of our product if it were to go into development. Furthermore, research into differing modes of communication between the Android and myRIO may improve usability - USB port communication was chosen due to the reduction in latency compared to Bluetooth communication, however other methods such as LoRaWAN may provide the same benefits

while removing the physical connectivity of the two systems, thereby increasing overall usability.

Because the current LabVIEW implementation on the FPGA target fails due to memory constraints, one future task is to investigate possible ways to reduce the memory usage for Karplus Strong Algorithm. One possible solution is to generate and store sample arrays of all notes on the real-time target prior to passing them to the FPGA target. This approach requires us to resolve the conflict of recognizing digital and analog pins on both the real-time target and the FPGA target. Moreover, both the VI for the Karplus Strong Algorithm and the VI with pre-recorded sound file can only generate and play waveforms of 1 second long. One possible extension is to enable interrupt so that two continuous strums within 1 second can be played. This would allow for a more realistic guitar playing experience for the user, and allow for different strumming rhythms to be produced.

With respect to sensing a strum of the air guitar, the current design could use a velocity sensor instead of an accelerometer in order to get more clear data for when the guitar is being played. A capacitive touch button could also be implemented instead of the physical push button switch. This would allow the user to perform strumming patterns with more ease while also adding a modern aesthetic.

References

- [1] “Kurv.” [Online]. Available: <http://kurv.io/>. [Accessed: 15-Sep-2019].
- [2] A. Ivatury, B. Tsai, and B. Zhang, “Air Guitar Gloves,” Dec. 2017.
- [3] “Kitara gets axed, replaced by the Misa tri-bass.” [Online]. Available: <https://newatlas.com/misa-digital-tri-bass/28662/>. [Accessed: 26-Sep-2019].
- [4] “MotionEvent,” *Android Developers*. [Online]. Available: <https://developer.android.com/reference/android/view/MotionEvent>. [Accessed: 26-Sep-2019].
- [5] “UART | Android Things,” *Android Developers*. [Online]. Available: <https://developer.android.com/things/sdk/pio/uart>. [Accessed: 26-Sep-2019].
- [6] “Principles Of Measurement Used By Laser Sensors And Scanners,” *Acuity Laser*. [Online]. Available: <http://www.acuitylaser.com/support/measurement-principles/>. [Accessed: 26-Sep-2019].
- [7] “USER GUIDE AND SPECIFICATIONS NI myRIO-1900.” National Instruments, 01-May-2016.
- [8] A. Katwala, “The spiralling environmental cost of our lithium battery addiction,” *Wired UK*, 05-Aug-2018.

- [9] “ATSC Recommended Practice: Techniques for Establishing and Maintaining Audio Loudness for Digital Television.” Advanced Television Systems Committee, 12-Mar-2013.
- [10] “Youlean Loudness Meter - Free VST, AU and AAX plugin,” *Youlean*.
- [11] “Laser Safety: Class 1, 1C, 1M, 2, 2M, 3R, 3B, and 4 - ANSI Blog,” *The ANSI Blog*, 14-Sep-2018.
- [12] M. Hartmann and J. Kelly, “Thermal Runaway Prevention of Li-ion Batteries by Novel Thermal Management System,” in 2018 IEEE Transportation Electrification Conference and Expo (ITEC), 2018, pp. 477–481.
- [13] “Amazon.com: High Capacity Rechargeable Battery, Elevin(TM) DC 12V 6800-18000mAh Super Rechargeable Li-ion Lithium Battery Pack +US Plug (18000mAh): 29: Toys & Games.” [Online]. Available: https://www.amazon.com/Capacity-Rechargeable-Elevin-TM-6800-18000mAh/dp/B07HFPBQML/ref=sr_1_1?keywords=Elevin%28TM%29+_+Electronic+Product&qid=1569547841&sr=8-1. [Accessed: 26-Sep-2019].
- [14] “Understanding the I2C Two Wire Bus Interface with NI LabVIEW - National Instruments.” [Online]. Available: <https://www.ni.com/en-us/innovations/white-papers/13/understanding-the-i2c-two-wire-bus-interface-with-ni-labview.html>. [Accessed: 26-Sep-2019].
- [15] J. A. Morente, A. Salinas, S. Toledo-Redondo, J. Fornieles-Callejon, A. Mendez, and J. Porti, “A New Experiment-Based Way to Introduce Fourier Transform and Time Domain #x2013;Frequency Domain Duality,” *IEEE Trans. Educ.*, vol. 56, no. 4, pp. 400–406, Nov. 2013.
- [16] “MSP430G2553 | MSP430G2x/i2x | MSP430 ultra-low-power MCUs | Description & parametrics.” [Online]. Available: <http://www.ti.com/product/MSP430G2553>. [Accessed: 06-Dec-2016].
- [17] Samsung Electronics, “Method for playing virtual musical instrument and electronic device for supporting the same,” *Justia*. [Online]. Available: <https://patents.justia.com/patent/9666173>. [Accessed: 02-Dec-2019].
- [18] “WO2009111815A9 - A digital instrument,” *Google Patents*. [Online]. Available: <https://patents.google.com/patent/WO2009111815A9/en>. [Accessed: 02-Dec-2019].
- [19] “WO2009108029A2 - Device and method to display fingerboard of mobile virtual guitar,” *Google Patents*. [Online]. Available: <https://patents.google.com/patent/WO2009108029A2/en>. [Accessed: 02-Dec-2019].

Appendix

Part Total	Price Per Unit	Quantity Ordered	Bulk Price	Description
0.1	0.1	1	6.28	36k
0.93	0.31	3	73.65	0.1uf
0.25	0.25	1	60.61	10kpf
0.42	0.42	1	111.06	1uf
0.1	0.1	1	13.63	10k
0.1	0.1	1	5.45	0ohm
0.1	0.1	1	18.98	1ohm
0.36	0.18	2	88	4pin Block
0.18	0.18	1	38.13	4700pf
0.24	0.12	2	42.72	10uf
0.2	0.1	2	18.98	1.8Ohm
12.77	12.77	1	6,274.77	Laser Sense
9.98	4.99	2		Myrio Cable
3.12	0.78	4	411.19	Button
0.3	0.1	3		0.1uf
1.47	0.49	3		100Ohm
42.25	8.45	5		470uF
19.86	6.62	3	4,313.66	Accel
6.24	0.78	8	1313.36	1uf
2.45	0.35	7	717	test point
1.86	0.62	3		6pin Conn

3.88	1.94	2		32pin Conn
1.59	0.53	3		6pin Straight
8.22	2.74	3	1248.36	Regulator
3.36	1.12	3	441.62	3.3uf
1.2	0.4	3	132.02	100k
2.9	0.58	5		4pin Conn square
1.41	0.47	3		4pin Conn row
7.9	3.95	2	1975	Jumper Wires
0.92	0.46	2		6pin Block
2.21	2.21	1		32pin small
0.2	0.1	2	13.86	10k
6.43	6.43	1	3650	proximity sensor
0.22	0.11	2	16.81	24k
0.23	0.23	1	54.1	0.1uf
0.52	0.52	1	153.09	4.7uf
0.11	0.11	1	88.2	2pin block
21.25	21.25	1	21250	UART USB
19.68	19.68	1		Break Out Board Laser
3.9	1.95	2		Wires
6.62	6.62	1		Accelerometer
0.1	0.1	1	4.13	0.1uf
24.95	24.95	1		Break Out Board Accel

Parts Cost(Prototype):	221.08	Parts Subtotal (Bulk):	42534.66	
Boards Designed:	6	Part Price Per Unit:	42.53466	
Cost Per Board:	35	Bulk Board Cost:	2215.84	
Board Cost:	210	Subtotal:	44750.5	
Total Cost:	431.08	Price per Unit:	44.7505	