Air Guitar Gloves

By

Akshay Ivatury

Brian Tsai

Bobby Zhang

ECE 445 Final Report, Senior Design, Spring 2017

TA: Jose Sanchez Vicarte

December 2017

Project No. 70

Abstract

The Air Guitar Gloves are a pair of gloves that emulate playing a real guitar though a limited range of chords. The main components of this project are the gloves which provide chord selection and string information, the microcontroller which maps the inputs to the corresponding notes, and the audio codec which converts the microcontroller output to an analog signal. We will be utilizing MIDI protocol in this project to determine which note or set of note we will be playing, with the audio codec chip being responsible for decoding the MIDI output and translating it to analog audio output that will go to a standard 3.5mm headphone jack.

Table of Contents

1 Introduction **3**

1.1 Objective 3

1.2 Background 3

1.3 High-Level Requirements 4

1.4 General Overview of Modules 4

1.4.1 Power Module 4

1.4.2 Sensor Module 4

1.4.3 Data Module 4

2 Design **5**

2.1 Design Procedure 5

2.2 Sensor Module 7

2.2.1 Left Hand Conversion Circuit 7

2.2.2 Right Hand Conversion Circuit 9

2.3 Data Module 11

2.3.1 Microcontroller 11

2.3.2 Audio Codec 14

2.3.3 Instrument Select 15

2.3.4 Audio Output 15

2.4 Power Module 15

2.4.1 Lithium-Ion Battery 15

2.4.2 Voltage Filter and Regulator 16

3 Requirements & Verifications **16**

3.1 Data Module 16

3.1.1 High-Level Algorithm 16

3.2 Sensor Module 17

3.2.1 Flex Sensors 17

3.2.2 Accelerometer 17

3.3 Power Module 17

3.3.1 Power Supply Draw 17

4 Cost Analysis **18**

5 Conclusion **19**

5.1 Accomplishments & Future Work 19

5.2 Ethics & Safety 19

References **20**

Appendix A Verification Table **21**

1 Introduction

1.1 Objective

People all over the world at some point in their lives have wished they could play guitar and have probably expressed this by playing an ‘air guitar’ along with one of their favorite songs. Despite its popularity, many people still miss out on the experience of playing guitar. Learning how to play guitar primarily involves two resources that govern the majority of people’s lives: time and money. Monitoring finger fret placement, memorizing chord shapes, and keeping track of which strings one’s right hand is playing all take valuable time to build muscle memory for. If the learning curve does not deter an individual, then the prices of a decent guitar and the equipment needed will hold back most others. Additionally, if a person decides that they can spare both of those, the sheer size and unwieldiness of the average guitar can cause even a professional guitarist transportation nightmares.

We proposed to create “Air Guitar Gloves” that allow the user to play guitar through gesture control. The left hand controls what notes and chords are playable by linking a chord to each gesture. The right hand dictates when and how these chords are played by deciding if the user’s hand is strumming or if they are playing individual strings. The sensor data is then converted into a series of bits that are sent to the microcontroller to convert into MIDI data. From there, the MIDI data is put through an audio codec chip to create audio output that is sent to a 3.5mm audio jack.

1.2 Background

Similar products all require hardware or software that hooks up to the gloves in order to function. This limits the user’s ability to move around comfortably while playing. Whether it is a method that replaces gesture control with a camera that detects the user’s hand positions or the need for an external sound system, none of these solutions allow the user to move freely about an area with ease.

Our project differs by providing gloves that do not require the user to hook their gloves up to computers or other bulky hardware aside from earphones. Thus, users will have a portable and affordable alternative for playing guitar.

1.3 High-Level Requirements

1. The gloves can play any combination of strings.
2. The gloves can operate for 2 hours without needing to be recharged.
3. Gloves include major and minor variations of the C, A, G, E, and D chords.

1.4 General Overview of Modules

1.4.1 Power Module

The power module provides 5V, 2.5V, 2.8V. And 3.3V to the system. It does this by taking in unregulated 5V from a battery pack via USB power, and filtering that power through a low pass filter to eliminate spikes. The 5V is used to power the microcontroller and both left and right hand signal conversion circuits. 2.5V and 2.8V are used to power various parts of the audio codec chip. The 3.3V is used to power the oscillator that is used with the audio codec chip.

1.4.2 Sensor Module

This module is responsible for determining which chord and strings the user would like to play as well as converting the analog data into digital data for the microcontroller to read. The left hand sensor module has flex sensors that are bent depending on which chord the user would like to play. These flex sensors are used in comparative circuits which then output the appropriate high or low voltage. The right hand circuit contains seven switches, six of which are used for individual strings, and the seventh one is used for strumming. When the strum button is held, an accelerometer output is also read to determine if the user is strumming upwards or downwards.

1.4.3 Data Module

The data module contains the microcontroller, instrument select, and audio codec components. The microcontroller we chose was an ATmega328 for its large number of I/O lines and 32kB of programmable flash memory. This microcontroller takes in the digital outputs of the left and right signal conversion circuits and outputs the corresponding MIDI values to the audio codec chip. The audio codec chip then decodes the MIDI values and converts them to an analog signal which will be output through a standard 3.5mm headphone jack.

2 Design

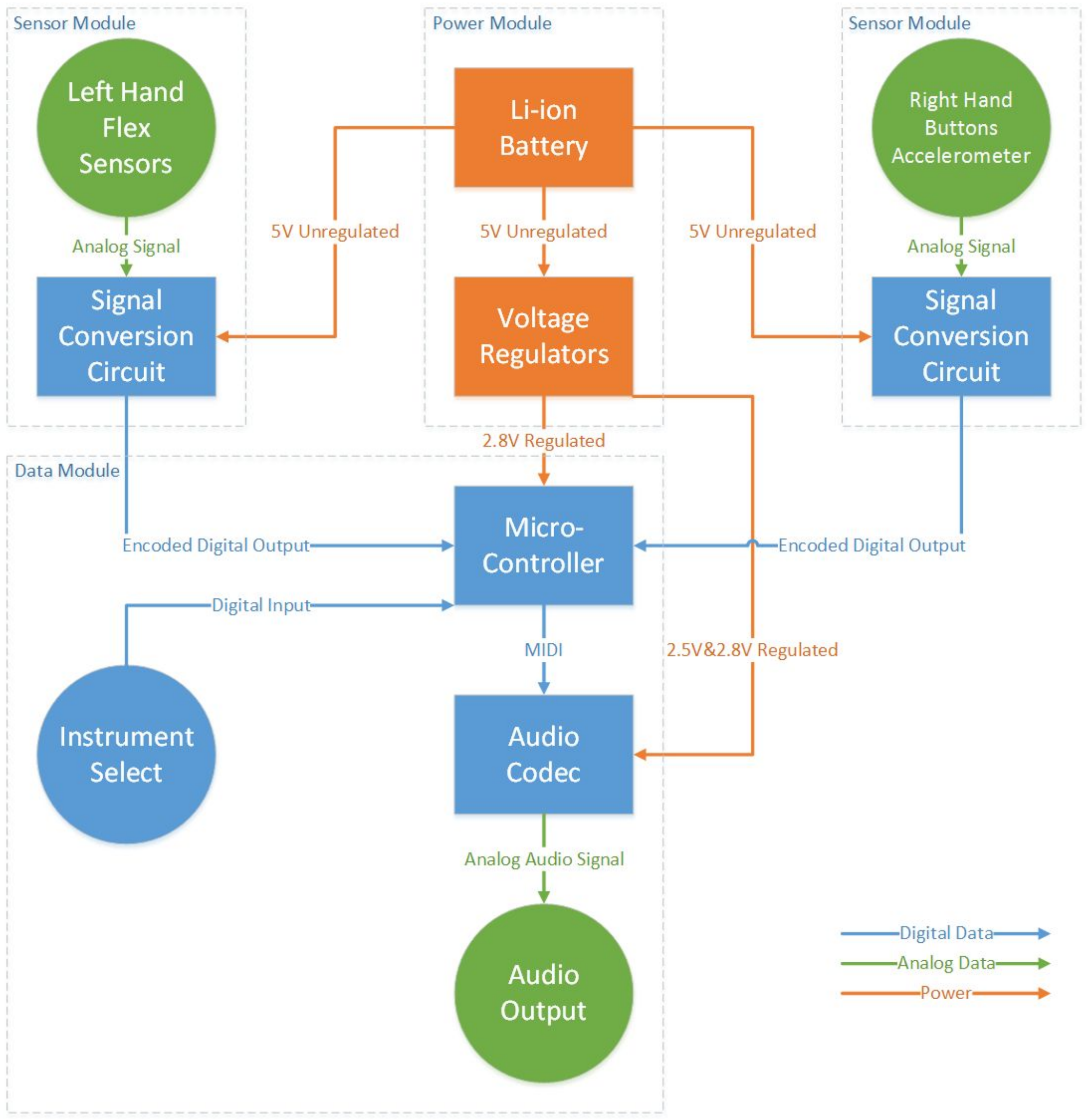

Figure 2.1: Block Diagram

2.1 Design Procedure

For the power module, we chose 5V as the input to the left and right hand signal conversion circuits because it would be the easiest way to power them. We chose the ATmega328 as the microcontroller for this project because we believed that it had enough flash memory to hold our final algorithm. Some other design options that were considered were running the entire system off 3.3V and powering the microcontroller with 3.3V also, however we did not choose this option because it resulted in unnecessary complexity by forcing us to create more voltage regulators than we needed, and it would make registering a high input in the microcontroller difficult because of the microcontrollers relative thresholds, and because the operational amplifiers consistently output a voltage lower than expected. We also chose to use the VS1103 chip for decoding the MIDI outputs from the microcontroller. We could have used the VS1053 chip or the VS1003 to decode the MIDI; however, those chips would have been excessive because they have extra features that we don't use. We chose the VS1103 for the more effective use of its resources.

The right hand sensor module allows us to play any combination of strings by using a button to alternate between strumming and non-strumming modes. In the strumming mode, the accelerometer outputs high when all strings should be played. If the user is in non-strumming mode, there are six buttons that each correspond to a respective string on the guitar. Pressing any of them will play their respective string, allowing the user to choose any combination.

The left hand sensor module uses various gestures to incorporate three variations of the parent chords C, A, G, E, and D. Two of the three variations for the parent chords are their major and minor forms. The last one is a miscellaneous variation that was added for more playable chords. This adds up to a total of 15 playable chords.

Based on our estimates, our design draws about 1.2Ah total. As such, the portable battery pack should be able to supply a minimum of 4200 mAh at 1.2A. This gives us a minimum lifespan of two hours before needing to be recharged.

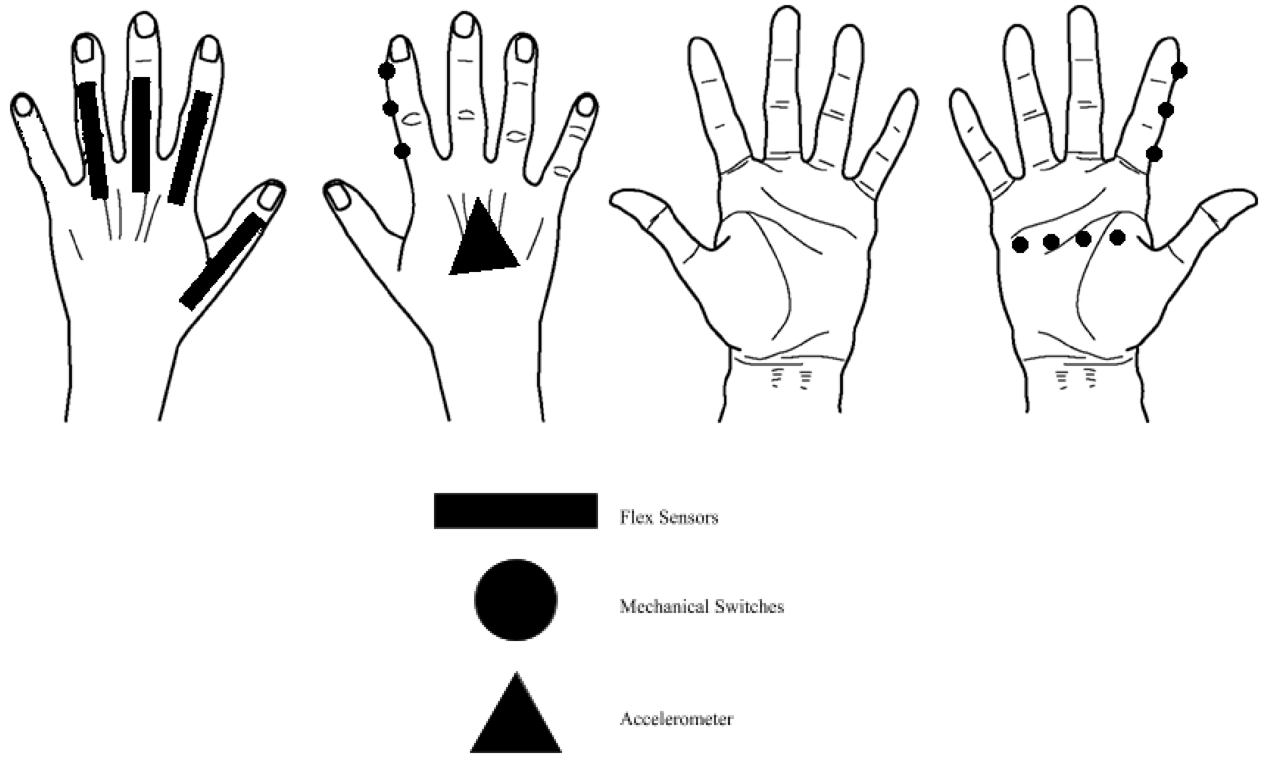

Figure 2.2: Sensor/Switch Layout

The two gloves are comprised of the sensors and switches shown above. The left hand has four flex sensors on the back of the glove for gesture input corresponding to various chords. The right hand has seven switches, four on the palm and three along the side of the index finger. The switch closest to the top of the index finger allows the user to alternate between strumming and non-strumming modes while the other six correlate to plucking one of the respective strings on a guitar. The right hand also uses an accelerometer on the back to measure acceleration for strumming.

2.2 Sensor Module

Overview: The sensor module is responsible for translating physical user input to their corresponding digital values that are then processed by the data module.

2.2.1 Left Hand Conversion Circuit

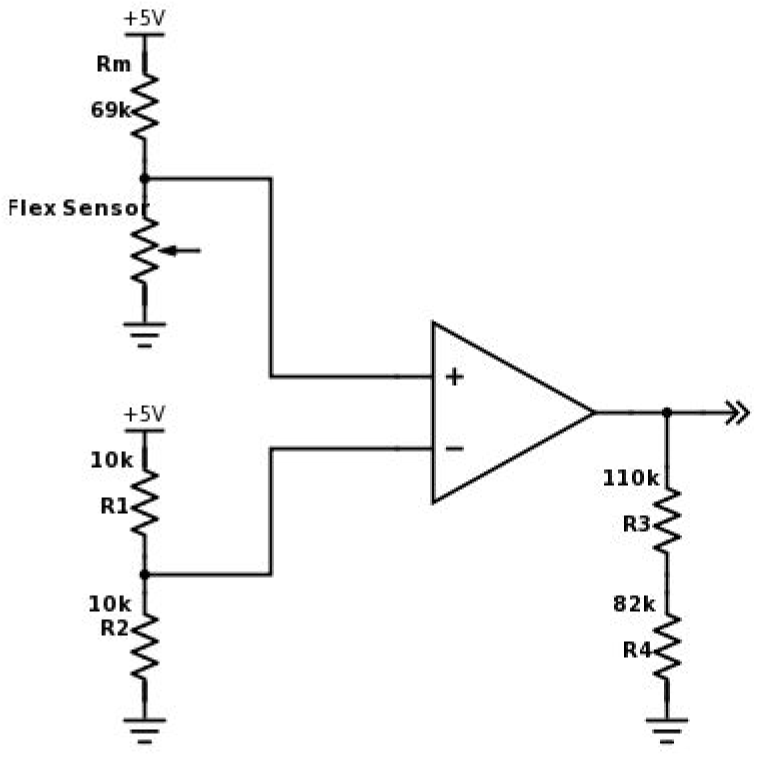

Figure 2.3: Variable Deflection Threshold Switch

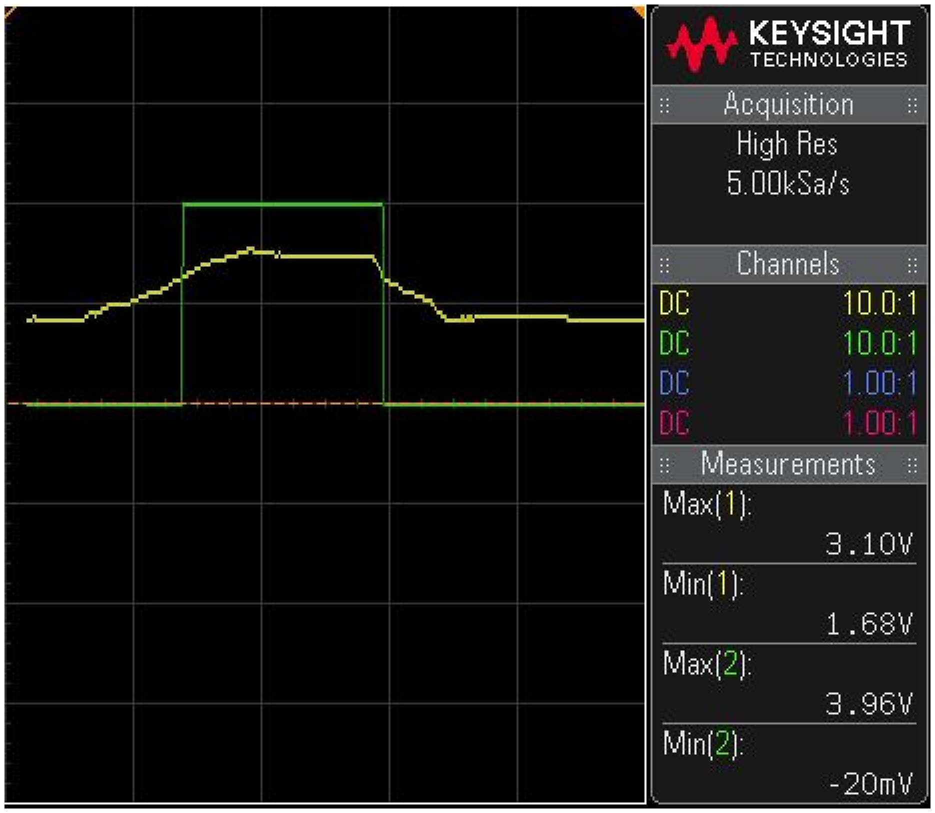

Figure 2.4: Flex Sensor Bending Waveform

The left hand circuit contains four flex sensors that are placed on the outside of the glove behind each finger. These flex sensors act as variable resistors in a comparative circuit. The circuit includes a hysteresis resistor, R_3, which "acts as a 'debouncer', eliminating any multiple triggering of the output that might occur [2]." It was recommended that the parallel combination resistance of R_3 and R_4 be about 47kΩ. Thus, we chose resistances of 110kΩ and 82kΩ. The value for R_m was determined by how far we wanted the user to bend their fingers in order to flip the digital output. We decided the user should only have to bend their fingers about 30° to keep the gestures for chord selection as comfortable as possible. A resistor value of 69kΩ was what we found to be the best fit as anything above was uncomfortable while maintaining gestures and anything below was too easily triggered.

Next, we will discuss how the flex sensors function. As the flex sensors are bent, their resistances increase from roughly 30kΩ to almost 70kΩ. This changes the voltage entering the positive terminal of the operational amplifier used in the circuit shown by Figure 2.3. The effect of the changing flex sensor resistances on this voltage is depicted in Figure 2.4 where the non-rectangular waveform is the positive terminal input voltage and the rectangular waveform is the operational amplifier output. The positive terminal input voltage increases from roughly 1.5V to 3V as the flex sensor is bent. Because we used a negative terminal input voltage of 2.5V to compare against, the operational amplifier outputs 4V when the positive terminal passes that threshold. The output is then sent through an inverter to pull signals up to 5V or down to 0V to complete the conversion of the analog data to digital logic values in the left hand.

We chose to use four flex sensors because we have a total of 15 playable chords, so with 16 possible permutations with the flex sensors in on and off states allows us to meet this requirement nicely with minimal excess. Additionally, we specifically omitted the pinky finger from having a flex sensor as most people have some sort of coupling between it and the ring finger. This provides severe difficulty in moving them independently of one another. Thus, four flex sensors was an efficient and comfortable design decision. The left hand uses these four flex sensors to produce four output bits that are then sent to the microcontroller.

2.2.2 Right Hand Conversion Circuit

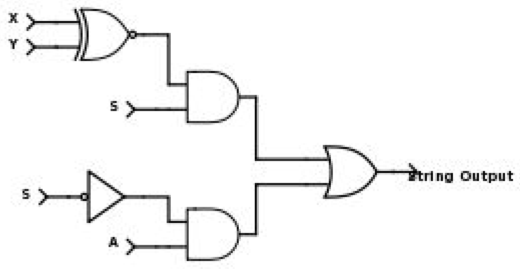

Figure 2.5: Boolean for Each String

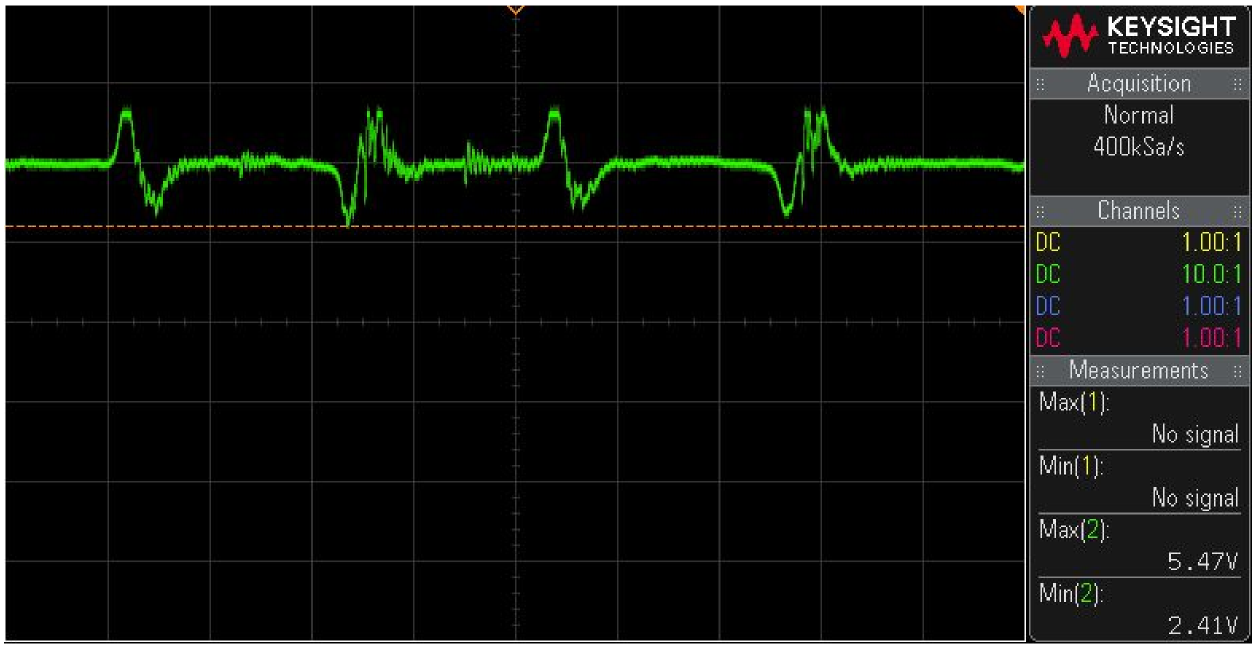

Figure 2.6: Accelerometer Output Waveform

The right hand circuit will contain seven switches and one accelerometer. The switches are for choosing to either strum or pluck an individual string. Four switches will be placed on the palm of the right hand horizontally to represent the strings D, G, B, and high E. Three more switches will be placed on the side of the index finger. Two of the three switches correlate to the strings low E and A, thus giving us all six strings on a guitar. When one of these switches is pressed it plays the note of that string with the chord picked by the left hand, or multiple notes if the user presses more than one at a time. The last switch closest to the tip of the index finger will be the button for strumming. When this switch is held down, the user will be able to mimic the motion of strumming to produce output from all strings via the accelerometer.

We used the SCA610-E23H1A accelerometer because we only needed a single axis and it outputs voltages at a high enough sensitivity. At zero acceleration, it outputs…

$$V_{out}(0) = \frac{V_{supply}}{2} \text{ Volts}$$

$$= 2.5 \text{ Volts as our supply is 5V}$$

If we factor in the sensitivity of this accelerometer at 1.333V/g. The voltage output becomes…

$$V_{out} = (Acceleration \times Sensitivity) + V_{out}(0) \text{ Volts}$$

While the accelerometer is mounted on the right hand, gravity causes the resting voltage output to be approximately 3.6V. With this, we set upper and lower thresholds to be 4.3V and 2.7V respectively via voltage dividers. Using comparative circuits similar to the one depicted in Figure 2.3, we were able to detect if the user was strumming up or down. To ensure that the accelerometer was able to surpass these thresholds, we measured its output in Figure 2.6 when being moved in both directions. As shown, the minimum and maximum voltages pass our thresholds successfully. Because the threshold comparators only output high values if the accelerometer is above their respective voltages, they needed to be sent through a single XNOR gate to output high if both thresholds were passed or if neither were passed. This completes the conversion of the accelerometer analog output to a digital logic input.

The switches were connected to a voltage supply of 5V on one end and the output to the microcontroller on the other. When the switch is pressed, it sends a high logic value and we connected the output to ground through a resistor to ensure that it sends a low logic value when not pressed. This was a necessary precaution because the output would be connected to a floating pin otherwise and would be guaranteed to register as a low logic value.

The right hand circuit outputs a total of six bits, one for each string on the guitar. Each string only outputs high if in strumming mode and the converted accelerometer digital input is high or if in non-strumming mode and the respective string switch is pressed. This logic is illustrated in Figure 2.7 where S is the input for strumming mode, X and Y are the comparative logic values, and A is the string's respective switch. These six outputs are then sent to the microcontroller to decode.

2.3 Data Module

Overview: The data module will contain the microcontroller, audio codec chip, and audio output. These are connected using wires, and are all contained in a box that can clip unto a person's waist.

2.3.1 Microcontroller

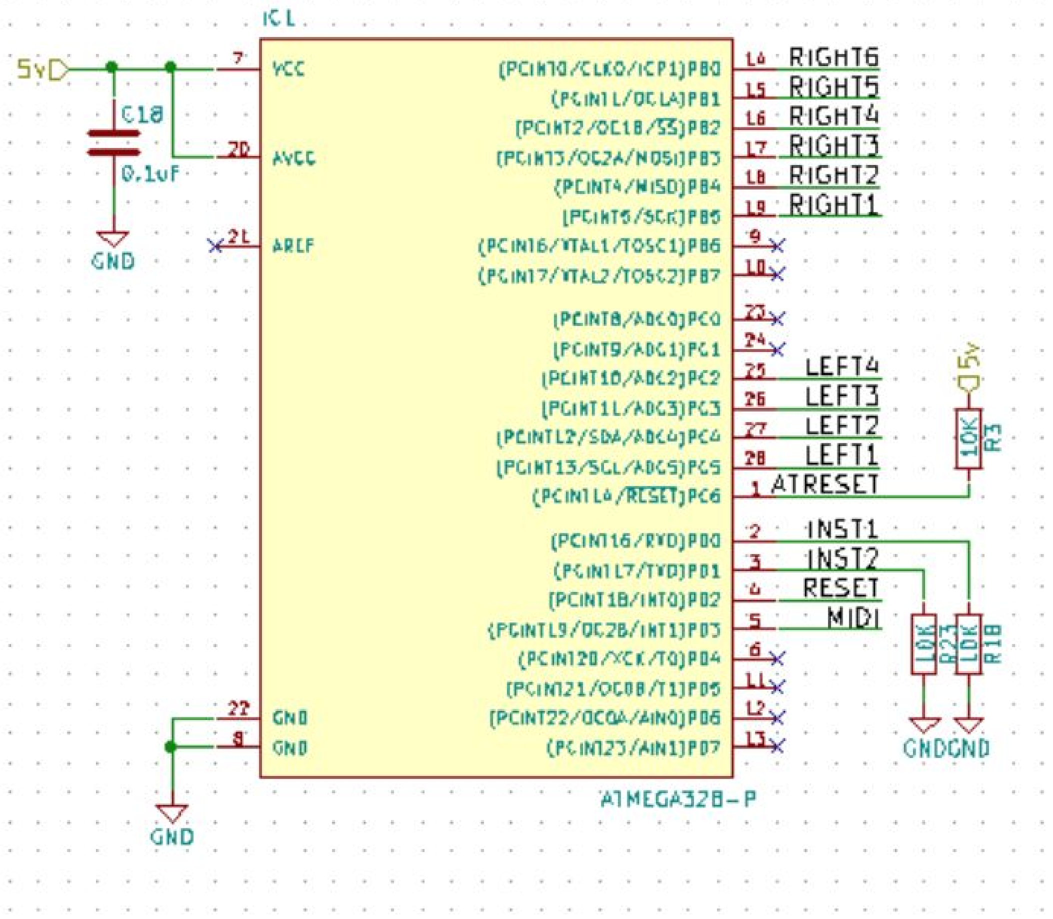

Figure 2.7: Schematic for ATmega328

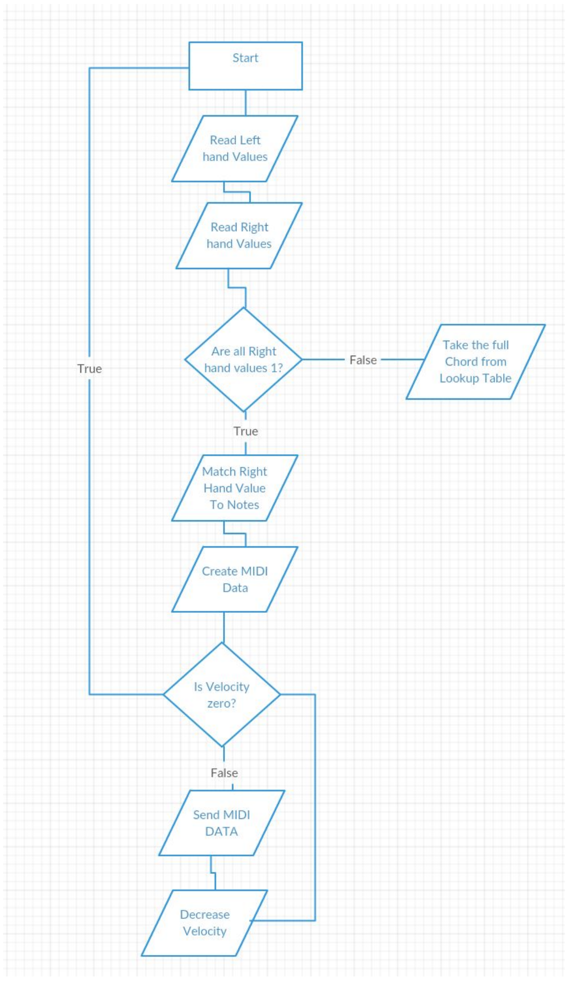

Figure 2.8: High Level Algorithm

The microcontroller, an ATmega328, is responsible for converting the digital output values from the left and right hand sensor modules into their corresponding MIDI data for the VS1103 audio codec chip. This microcontroller was chosen for its affordability, internal clock speed, and 32kB programmable memory. This chip contains 3 bays of I/O ports, each of which contain 7 IO lines. The first bay reads the output of the left hand sensor module, the second bay reads the output of the right hand sensor module and the last bay contains the buttons connected to the instrument select as well as the output to the audio codec chip.

The microcontrollers data conversion algorithm can be seen in Figure 2.8 above. This algorithm first reads in the output values from the left and right hand signal conversion circuits. Then we determine which combination of strings we need to play by checking the values of the right hand. Next, we figure out which notes are needed by inputting the left hand values into a lookup table. We then set the appropriate velocity (volume), and output the MIDI data to the audio codec chip. We then check the velocity and decrement it appropriately until it becomes zero. If another note is played between them we turn off the current note and restart in the new note until it ends or it is interrupted.

The process for converting a chord to its particular MIDI data value is as follows:

Chord → String Notes → Piano Note Number → MIDI Number → MIDI Status & Data Value

Consider the C major chord. This chord consists of the strings E A D G B E being played at the respective frets of 0-3-2-0-1-0. We then take the fret numbers and map them to their corresponding note numbers which are E_2, C_3, E_3, G_3, C_4, and E_4. These values will then correspond to the MIDI note values of 40, 48, 52, 55, 60, and 64. Next, these values are mapped to the command, channel, and velocity we want the audio codec chip to decode. The command and channel are each 4 bits of an 8 bit value that denote whether the note is on or off for a particular channel. The velocity is an 8 bit value that denotes the volume of the particular note we are trying to play. The last value is the note which is also an 8 bit value. This total 3 byte value is then read in by the desired channel of the audio codec chip and the chip plays the corresponding instruments note accordingly. The table below is an example of playing each note as loud as possible for a particular chord given the appropriate left and right hand inputs. The MIDI commands would be note_on, velocity = 127, channel 1.

Chord/Note	**MIDI Data**	**Left Hand Input**	**Right Hand Input**
C_{major}/E_2	0x09287F	0000	111111
C_{major}/C_3	0x09307F	0000	111111
C_{major}/E_3	0x09347F	0000	111111
C_{major}/G_3	0x09377F	0000	111111
C_{major}/C_4	0x093C7F	0000	111111
C_{major}/E_4	0x09407F	0000	111111

Table 2.9 Example MIDI output

2.3.2 Audio Codec

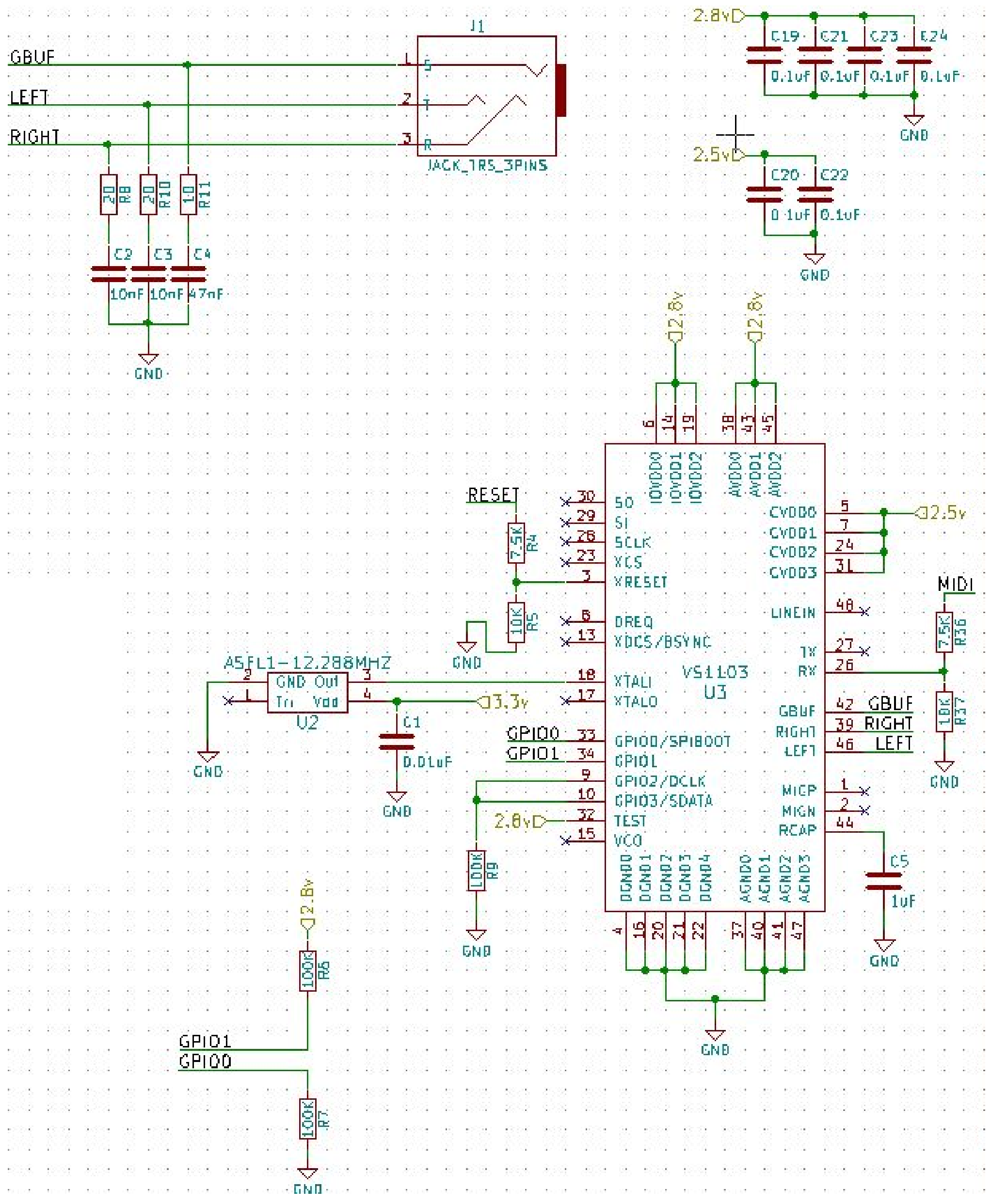

Figure 2.10: Schematic for VS1103 and Audio Jack

The VS1103 chip is responsible for reading in the MIDI values and converting them to the corresponding analog output for the chosen instrument. We chose this chip because it has the ability to decode real time MIDI data and run on fairly low power when compared to attempting to load MIDI libraries and decode using another microcontroller. This allows us to feed the real time MIDI data into the VS1103 chip and output the corresponding instrument sound and volume level associated with that MIDI data. This chip contains a digital to analog converter, and we will use this converter to output the analog sound to the audio output. This chip runs at 2.8V analog and IO power input and 2.5V digital power input with a maximum current draw of 60mAh for analog power and 15mAh for digital power.

2.3.3 Instrument Select

The instrument select will be two buttons that connect to the microcontroller that will be on the data module housing. These two buttons will be responsible for switching between an acoustic guitar and electric guitar. When the button is pressed, it will trigger the microcontroller to stop all notes. Then we will write either 0x21 or 0x22 to channel one depending on which instrument we pick. This will cause the audio codec chip to change its instrumental output.

2.3.4 Audio Output

The audio output is a standard 3.5mm headphone jack that is connected to the left and right pins on the VS1103. They read in the analog signal and output it to whatever is connected to the standard 3.5mm TRS jack component.

2.4 Power Module

Overview: The power module is responsible for supplying power to all the other modules.

2.4.1 Lithium-Ion Battery

The portable battery pack supplies power to both signal conversion circuits and regulated power to the audio codec chip, microcontroller, and audio jack. The battery will have a capacity of 5000mAh which suffices for more than our needs.

2.4.2 Voltage Filter and Regulator

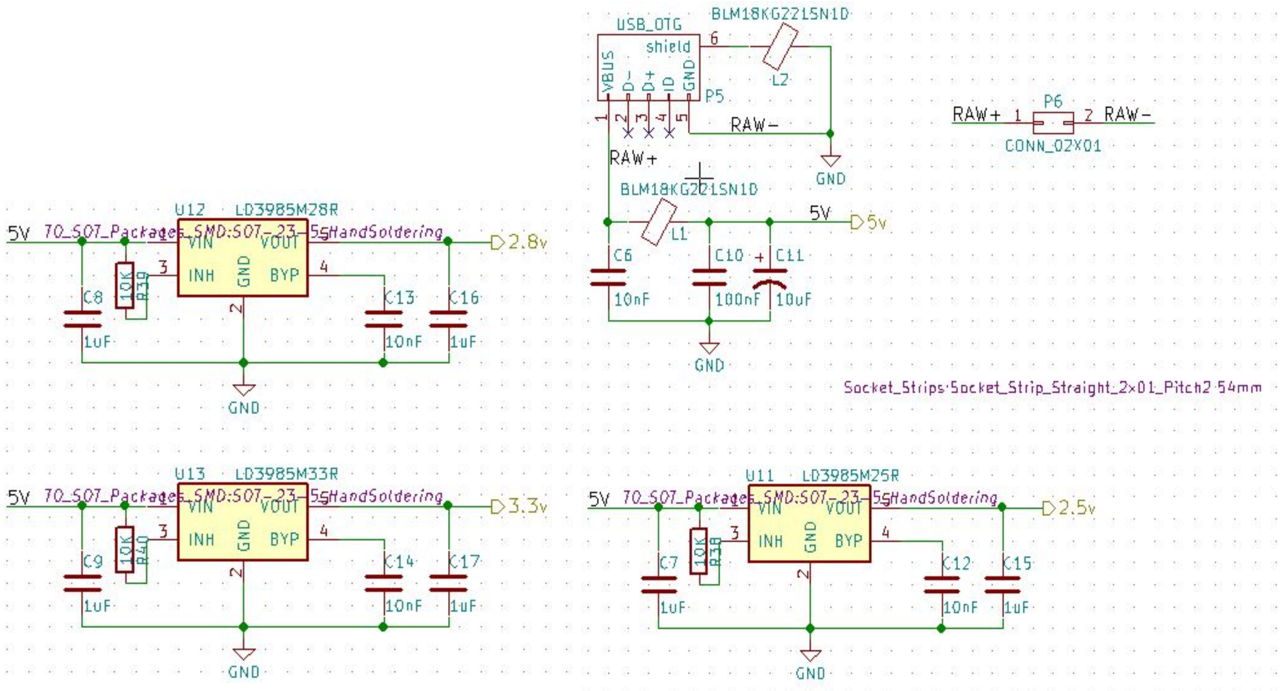

Figure 2.11: Voltage Filter and Regulator Schematic

The voltage filtering circuit consist of three different size capacitor and a ferrite bead which forms a low pass filter to eliminate spikes in USB power. The voltage regulators are used to provide regulated voltages to the microcontroller and the audio codec chip. The microcontroller requires a power supply of 1.8V-5.5V while the audio codec requires 2 different input voltages of 2.8V and 2.5V. Additionally the oscillator for audio codec requires 3.3V. Thus, we use three voltage regulators for 2.5V, 2.8V, and 3.3V to supply all of those components.

3 Requirements & Verifications

3.1 Data Module

3.1.1 High-Level Algorithm

In order to make the high-level algorithm in Figure 2.8 to work a number of requirements need to be met. These include being able to read the left and right hand signal conversion outputs and make sure that the microcontroller can read them. In order to verify this, we used the Arduino serial monitor to confirm when a string or chord was changed. Once this was verified, we made sure that the VS1103 chip was receiving the correct input by sending a known note and measuring the output of the VS1103 with a tuner to make sure that the note was the same note we sent in. Next, we made sure that the entire high level algorithm functioned within a short enough amount of time (roughly 20ms-30ms) by utilizing the stopwatch functions available on the Arduino. The 16MHz internal clock was plenty fast enough, and there was almost no lag between a user changing inputs and the corresponding output change.

3.2 Sensor Module

3.2.1 Flex Sensors

Figure 2.4 illustrates the verification of the flex sensors in the left hand sensor module by showing the effect of the changing flex sensor resistances on the comparative circuit outputs. The non-rectangular waveform is the voltage that we compared to our set threshold value, while the rectangular waveform is the comparative circuit output for the left hand. The flex sensor starts out flat and is slowly bent to a 30° angle before being slowly flattened. As the flex sensor increases our positive comparative voltage input, it surpasses the threshold and causes the output to become high. Once it begins flattening again, the voltage lowers and the output follows suit.

3.2.2 Accelerometer

Figure 2.6 depicts the verification of the accelerometer that was used in the right hand sensor module. The waveform shown is the analog accelerometer output as it is moved upwards and downwards. When moving upwards, we achieve a maximum voltage that surpasses our upper threshold voltage. Similarly, we measure a minimum voltage that is below our lower threshold voltage. With these two thresholds being properly set, we were able to verify that the accelerometer was interacting appropriately with the logic that was in place for the right hand.

3.3 Power Module

3.3.1 Power Supply Draw

We estimated that the total current draw of our circuit would be roughly 1Ah, and by having a battery pack that can provide at least 2Ah we would be able to meet our power requirements for portability. This 1Ah estimated current draw came from the fact that the datasheet for the flex sensors said that each flex sensor would draw roughly 100mAh - 150mAh. However, after actually finishing the project we found out total current draw to be roughly 350mAh. After some probing we found out that the flex sensors didn't draw nearly as much power as the datasheet said it would, with the average being less than 10mAh per flex sensor. We believe this discrepancy is due to the fact that the flex sensor datasheet may have been referring to the total current draw including a comparator circuit they included, whereas we only needed to know the draw of the actual flex sensors.

4 Cost Analysis

Our fixed salaries will be $40/hr for 14hrs/wk per person. The design has taken roughly this entire semester of 14 weeks if we ignore the weeks of demonstrations and final papers. Our labor cost is calculated as…

$$3 \times \frac{\$40}{hr} \times \frac{14hr}{wk} \times 14wks \times 2.5 \ = \ \$58,800$$

Category	Component	Cost
Miscellaneous	Pair of Neoprene Gloves	$25
Left & Right Sensor Modules	5 2.2” Flex Sensors (Spectra Symbol)	5($7.95) = $39.75
	7 Switches (Sparkfun)	$4.95
	1 SCA610-E23H1A Single Axis Accelerometer (Murata)	$35.53
	4 LM358N Dual Op-Amp (Texas Instruments)	4($4.36) = $17.44
	Assorted Logic Chips and Passive Components (Digikey)	$20
	2 PCBs (PCBWay)	$10
Data Module	ATmega328 Microcontroller (Digikey)	$1.96
	VS1103 Audio Codec (VLSI Solutions)	$9.95
	3.5mm Audio Jack (Sparkfun)	$1.50
Power Module	Lithium-Ion Battery Pack (Mouser)	$20
	Voltage Regulators (Mouser)	$1.27

Our total cost will then add up to $58,987.35.

5 Conclusion

5.1 Accomplishments & Future Work

We accomplished most of the goals we set in the beginning of the project. We were able to meet all of our high level requirements and the current requirements for portability. The left and right hand circuits properly converted the analog outputs to digital inputs for the microcontroller. Our algorithm worked successfully and the microcontroller was fast enough to mask any lag between the user input and the sound output. We were not able to implement the accelerometer in the final demonstration due to connectivity issues with the ribbon cable. We were also not able to implement the instrument select or port the code to the ATmega328 due to time constraints and improper planning. In the future we would like to see the instrument select and accelerometer implemented, and would also like to see the system run off the ATmega328 instead of an Arduino. This project has the potential to grow due to the fact that the VS1103 has the ability to synthesize over 100 instruments. By adding more gestures in the left and right hand circuits we could potentially emulate other instruments as well. Another future implementation we would like to see is a 5-pin MIDI output. This way the user would have the ability to plug in their own MIDI synthesizer instead of using the VS1103 chip.

5.2 Ethics & Safety

The primary safety concern of our project is the lithium-ion battery. At high temperatures, the battery may undergo heat failure and combust. We used a battery back to monitor and ensure the user's safety.

Another point of safety concern is the user's perspiration that may accumulate while using the gloves. We did not include any circuitry inside the gloves that could potentially shock the user or be corroded by the user. Moreover, the gloves are not completely closed off and will allow ambient airflow to ventilate the user's hands.

While working in the electronics lab, followed the lab safety training that we have all undergone in order to avoid any risk of being shocked, burned, etc.

We have ensured that every design aspect of our project is original or properly cites respective references so as to not infringe upon any copyrighted designs of existing products.

While using wearable electronics, all users should avoid water, avoid handling the hardware in a manner that it was not intended for, and ensure that there is enough room around the user to prevent any accidental injuries. If the power supply becomes excessively hot, the user should remove the gloves and stop use.

References

[1] Spectrasymbol, “Flex Sensor,” 2009 [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf [Accessed: February 8, 2017].

[2] Flexpoint Sensor Systems, “Bend Sensor Technology Electronic Interface Design Guide,” 1997 [Online]. Available: http://www.flexpoint.com/technicalDataSheets/electronicDesignGuide.pdf [Accessed: February 8, 2017].

[3] Analog Devices, “Single-Axis, High-g, iMEMS Accelerometers,” 2005 [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL193.pdf [Accessed: February 8, 2017].

[4] VLSI Solution, “VS1103b Datasheet,” 2014 [Online]. Available: http://www.vlsi.fi/fileadmin/datasheets/vs1103.pdf [Accessed: February 23, 2017].

[5] Murata, “SCA610-E23H1A Single Axis Accelerometer with Analog Interface,” 2015 [Online]. Available: http://www.murata.com/~/media/webrenewal/products/sensor/accel/sca6x0/sca610-e23h1a%20datasheet%20revb.ashx?la=ja-jp [Accessed: February 23, 2017].

[6] Ieee.org, "IEEE IEEE Code of Ethics", 2017. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: February 8, 2017]

[7] ECIA Authorized, “LD3985 Datasheet, Inventory, & Price | ECIA”, 2017. [Online]. Available: http://www.eciaauthorized.com/en/search/LD3985 [Accessed: February 8, 2017]

Appendix A Verification Table

1. Sensor Module

1.1 Left Hand

Requirements	Verification	Status
1. The total current draw of the flex sensors should not exceed 600mAh ± 100mAh (2 pts) 2. The flex sensors must change output when bent past or straightened below a 70° angle (5 pts)	1. Probing with an ammeter will allow us to see the current draw of all the flex sensors. With the removal of one flex sensor from the new gesture mapping, the total current draw is unlikely to exceed the decided limit. 2. Flex sensors were carefully bent at one point at an angle of about 70° while changing resistor values. Chords should change appropriately.	

1.2 Right Hand R&V

Requirements	Verification	Status
1. The accelerometer should be able to output sound when moved up or down and the strum switch is held (2 pts) 2. The accelerometer must only affect the output when the strum switch is held (5 pts) 3. All string switches must correspond to their respective strings when the strum switch is not being held. (5 pts)	1. While holding the strum button, chords should be played while moving the right hand up or down. This confirms the accelerometer's ability to output sound. 2. The accelerometer will be shook while the strum switch is not being held and while it is being held. Sound should only play while the strum switch is held. 3. The string switches will be pressed while the strum switch is held and while it is not held. The string switches will only output sound if the strum switch is not held.	

2. Data Module

Requirements	Verifications	Status
1. The microcontroller must be able to operate at 2.8V - 5V and between 50mAh - 100mAh in order to meet power requirements for portability. (0 pts) 2. The microcontroller must be able to read the output of the left hand hand signal conversion circuit. (3pts) 3. The microcontroller must be able to get the corresponding MIDI values from the lookup table with the left hand inputs. (4 pts)	1. We will supply 2.8v to the power pin of the ATmega328 and measure the input current to make sure that this does not exceed 100mA. 2. We will connect LEDs to the output of the left hand conversion circuit and also connect LEDs to the output of the microcontroller, which will be mapped to the input of the left hand conversion circuit. Then, if the LEDs match up, we know that the microcontroller is able to read the left hand conversion circuit properly 3. We will have the micro controller output its MIDI values to an Arduino Uno. We will then check the serial log and confirm that the MIDI values are the correct values for that particular chord.	No (Incorrect Microcontroller used for final demo) Yes Yes

Requirements	Verifications	Status
4. The micro-controller must be able to read the output of the right hand hand signal conversion circuit. (3 pts) 5. The micro-controller must be able to parse	4. We will connect LEDs to the output of the right hand conversion circuit and also connect LEDs to the output of the micro-controller, which will be mapped to the input of the right hand conversion circuit. We will then test to make sure that each button for individual strings works by making sure that both the input and output LEDs match. We will then	Yes

the MIDI data using the right hand values and only send the values corresponding to the strings picked, or all strings if the strum function is engaged (4 pts) 6. It must be able to properly convert the digital sensor data to MIDI data in under 20ms-30ms so there is no lag between playing a note and hearing it. (7 pts)	make sure that the strum works by pressing down the strum button and making the strum motion with the accelerometer. This should cause all the LEDs to light up on both the input and output LEDs 5. We will have the micro controller output its MIDI values to an Arduino Uno. We will then check the serial log and confirm that the MIDI values are the correct values for the that particular set of strings or chord. 6. We will create a test harness in another micro-controller, such as an Arduino Uno, and feed in digital values to the micro-controller and measure the response time when the MIDI data is read back into the Arduino Uno, and will confirm that the value is correct.	Yes Yes

3. Audio Codec

Requirements	Verifications	Status
1. This chip must be able to operate between 4.5V - 5V and must draw less than 100mAh-150mAh in order to meet power requirements for portability. (0 pts) 2. This chip must be able to decode the MIDI data values that are output from the microcontroller to analog audio output (2 pts)	1. When running the VS1103 we will probe the voltage and current draw to make sure it does not exceed 5v and 150 mAh. 2. We will send known MIDI values into the VS1103 chip and measure the analog output with an oscilloscope and make sure that the corresponding output matches the MIDI value's note.	Yes Yes

3.1 Instrument Select

Requirements	Verifications	Status
1. When either button is pressed, the corresponding instrument will be heard in the audio output (2 pts)	1. We will test this by playing notes and comparing their sound output against the sound of acoustic vs electric	No

3.2 Audio Output

Requirements	Verifications	Status
1. This 3.5mm audio jack properly outputs the desired sound (0 pts)	1. We will play a known sound through the VS1103 chip and measure its sound output from the 3.5mm audio jack with a tuner to confirm that they are the same note.	Yes

4. Power Module

Requirements	Verification	Status
1. Battery should be able to supply power for a minimum of 2 hours before needing to be recharged. (5 pts) 2. Supply 5V ± 5% to the left and right sensor modules. (0 pts)	1. Based on our estimates, our design will draw about 1.5A, so the battery will have to be able to supply a minimum of 4Ah to be safe. We use a capacity of 5Ah to guarantee 2hrs within our tolerance range. 2. We will use a voltmeter or oscilloscope to monitor the battery voltage and ensure that it is within our voltage range.	

4.1 Voltage Regulation

Requirements	Verification	Status

1. Must be able to clean up the noisy 5V ± 50mV signal from the USB power input (1 pts) 2. Must be able to provide 3.3V to the clock of the VS1103 (0 pts) 3. Must be able to provide 2.8V and 2.5 to multiple ports on the VS1103 (0 pts)	1. Will observe the USB power before and after the filtering to make sure that the signal has fluctuations that are less than 60mV 2. This voltage range will be monitored with a voltmeter or oscilloscope 3. This voltage range will be monitored with a voltmeter or oscilloscope	