Oscilloscope Guitar Hero

Druck Green
Daniel Shaar

Table of Contents

1 Abstract (Daniel)

This project aims to recreate the classic video game *Guitar Hero* using only analog circuitry (the only exception being the use of an Arduino, whose usage is detailed later). The game is played through user interaction with moving notes, which are displayed graphically on an oscilloscope in XY mode. A user presses four buttons in coordination with the four notes on the screen. The notes form some rhythmic pattern, and a speaker generates audio feedback, which indicates correct and incorrect button presses made by the user. Additionally, the design contains a small variety of game modes that vary the difficulty of the gameplay.

The implementation of this design requires circuitry that generates waveforms for the oscilloscope to display, and circuitry that produces tones, which mirror the user's actions. The details of this will be discussed abstractly in section 2, and in full detail in section 3.

2 High-level Design (Daniel)

As alluded to in the abstract, the circuitry for the game is divided into two primary sections that deal with the visual and audio aspects of the game. The interaction between these sections is minimized in an attempt to make the circuit modular. However, due to the user interaction with the visuals, it is infeasible to have no such interaction between the visuals and the audio generated. In order to produce the visuals, waveforms are generated for one note and four positions, and are combined and sampled from in order to produce inputs into the XY mode of the oscilloscope. The movement of the notes is triggered by the Arduino input, and the XY output is probed by the oscilloscope.

The other input of the circuit is that of the user's button presses. Based on the timing of the user button presses and the positions of the notes, various tones are generated and mixed together to play a 4-bit song. The output of the audio mixer goes through an amplifier, which then plays on a speaker for the user to hear.

Due to the error found in component values, tuning capabilities are added through the use of potentiometers, which can be adjusted to make the gameplay more accurate and robust. The circuitry also aims to preserve modularity, so that additional extensions to

the game may be added. This modularity is primarily achieved through the use of buffering.

2.1 Block Diagram (Daniel)

The block diagram in Figure 1 attempts to show a high-level overview of the operation of the circuit. More information on the individual module functions can be found in Section 2.2. Precise circuitry and technical descriptions of modules can be found in Section 3.

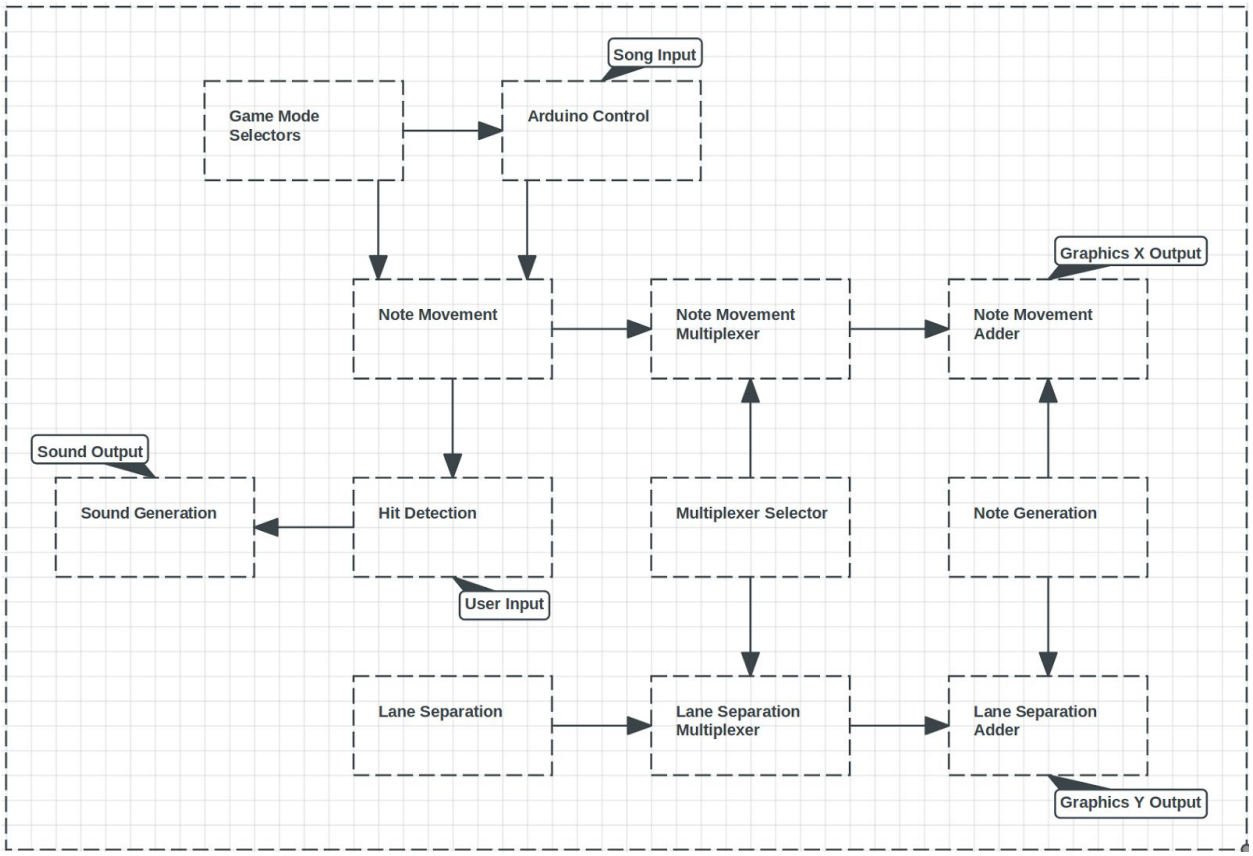

Figure 1: A modular block diagram of the circuit operation. The user input and song input (given by the Arduino) are labelled, as well as the graphics and audio outputs.

2.2 Module Descriptions (Daniel)

Before the detailing of the exact functionality and circuitry involved in producing the gameplay, a brief overview of each module's function and interactions with the other modules is given.

At the core of this game is the visual component for the user to interact with. In order to achieve the four notes necessary for gameplay, many waveforms are generated and used as input into the X and Y channels of the oscilloscope (note that this is intended for a scope tilted sideways, so that the aspect ratio of the screen is longer vertically than horizontally). Rapidly sampling the positions of the four notes, and adding the positions to the waveform that generates an individual note, allows all notes to appear on the screen at a reasonable frame rate. To make this as modular as possible, one circuit module is responsible for the creation of a single note on the screen. This module is the ***note generation*** module.

The ***multiplexer selector*** module creates the rapid switching mechanism that allows multiple notes to appear on the screen. It is essentially used as an oscillating selector input into the multiplexer modules mentioned later. To position the notes horizontally, we make use of a ***lane separation*** module in conjunction with the ***lane separation multiplexer*** and ***lane separation adder*** modules. The responsibility of these module is to display the vertical state of each note, which is a fixed value. The lane separation module generates these fixed values. The lane separation multiplexer then selects which vertical position to display. Finally, the lane separation adder adds this value to the appropriate waveform generated by the note generation module.

However, *Guitar Hero* would not be much of a game if the notes never moved. To accomplish this task, the notes must progress vertically from an initial position to a final position, and reset appropriately once they hit the end. This is done in the ***note movement*** module, which is combined with the ***note movement multiplexer*** and ***note movement adder*** modules in the same manner as before.

User interaction capabilities are handled in the ***hit detection*** module. This module consists of 4 buttons and circuitry to detect when notes have been hit and missed. This module utilizes the output of the note movement module to determine if the user's timing is accurate.

Finally, once the results of the user's button presses have been processed, ultimately some noise should be produced! This final functionality is handled in the ***sound generation*** module. Using just the output from the previous module, it generates and amplifies different tones depending on whether the user successfully timed his or her button presses.

3 Modules

3.1 Graphics (Druck)

The following modules together make up the visual component of this project. They include the circuitry necessary to parametrically generate one circle, as well as the circuitry to bias and sample that circle to simultaneously display four circles. Each circle has its own y-offset, and is able to move across the screen along the x-axis independently of the other circles.

3.1.1 Note Generation (Druck)

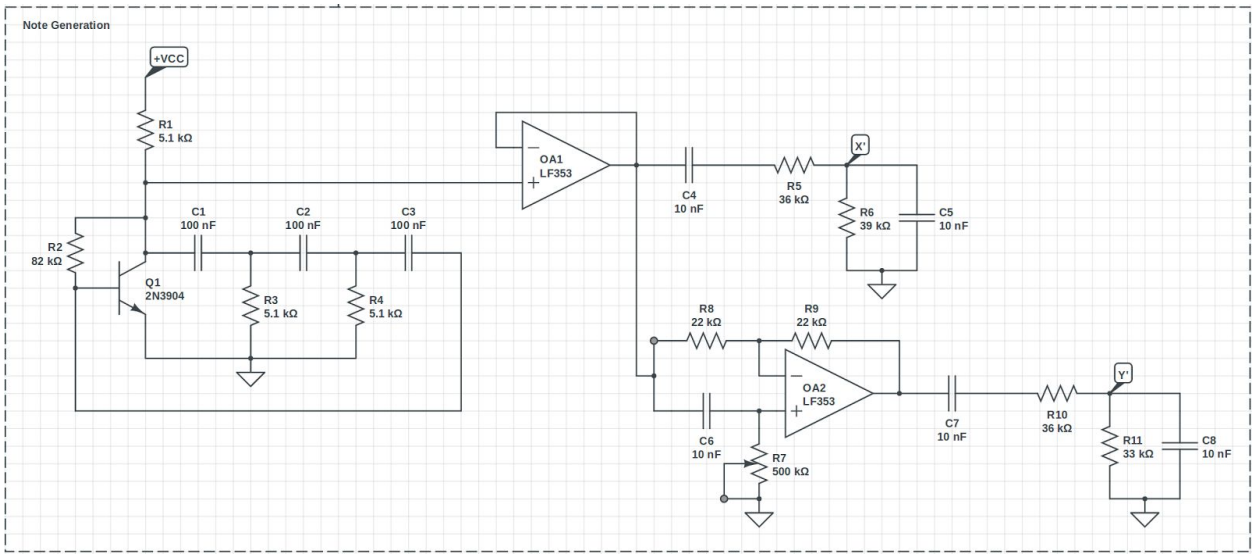

Figure 2: A Phase-shift oscillator leading into an all-pass filter. The outputs of the oscillator and the filter are combined with offsets, and later fed into the X and Y channels of the oscilloscope respectively.

The circuitry depicted in the Figure 2 is used to trace a single circle on the oscilloscope. In order to parametrically trace out a circle using the X and Y channels on a ‘scope, two sinusoids which are of equal magnitude, but 90° out of phase must be given as input. Small deviations in these relative magnitudes or phases can quickly lead to a noticeably deformed output. To achieve the accuracy needed for a clean output, this circuit operates at a very low frequency of 175 Hz. This lower frequency means less distortion from cheap op-amps and a closer approximation to a sine wave from the initial oscillator stage. The first stage of this circuit, a phase shift oscillator [1], generates one of the

sinusoids needed for the circle. The output of this stage isn't an exact sinusoid, but a close approximation.
An op-amp buffer is used to reduce the high output impedance of the oscillator stage, about 5kΩ due to the collector resistor. The second op-amp in the circuit is used as an active all-pass filter [4]. The transfer function of this circuit has unity gain for all frequencies and a phase of 90° at 1/RC, where R is R7 and C is C6. The output from the buffer is passed into the all-pass filter where a 500 kΩ POT has been used in place of R, so that the output sine wave can be adjusted to give an exact 90 degrees phase shift. A POT is used here in place of a resistor because, as mentioned earlier, the output of the oscillator stage is only an approximation to a sine wave. Therefore the input to the all-pass filter contains multiple frequencies, and setting 1/RC equal to the frequency of the input will not produce the desired exact 90 degree phase shift.

Furthermore, the shifting of different frequencies by different amounts leads to distortion in the shape and amplitude of the output. To get the desired equal amplitudes of the shifted sine waves, the output of the buffer and all-pass filter are AC coupled through C4 and C7 and fed through voltage dividers, which adjust their relative amplitudes. The voltage dividers serve the additional function of sizing the circles on the 'scope.

3.1.2 Lane Separation (Druck)

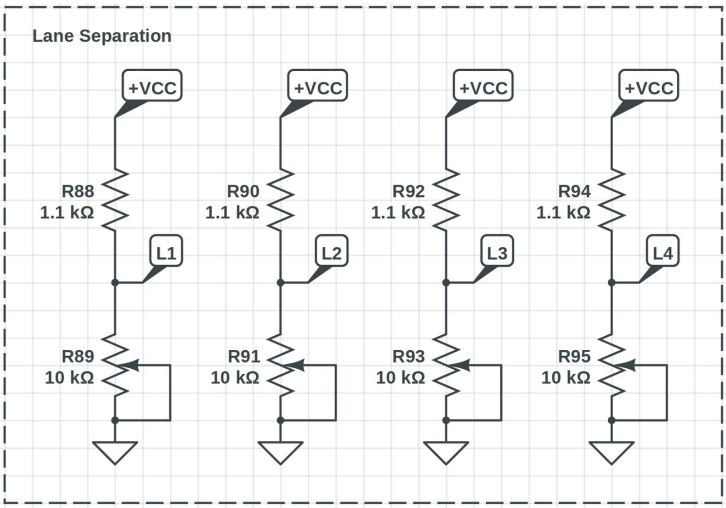

Figure 3: Four voltage dividers (one per note) with potentiometers to tune the horizontal position of the notes on the tilted oscilloscope screen.

The note generated from the previous module will be used to trace four notes with different X and Y DC voltage offsets. The circuitry that converts one circle into four will be discussed later. These next two modules focus on the DC offsets for the X and Y

channels of the ‘scope. This module, shown in Figure 3, provides the DC offsets along the Y-axis of the ‘scope, which separates the notes into lanes. In this implementation, the notes move along the X-axis of the ‘scope in lanes which are generated by this module. Each lane corresponds to a button that the user should press to hit the note in that lane.

Because the notes cannot move along the Y-axis, the voltage offsets produced by this module are static and the circuitry is simple. Each offset is taken from a different resistor voltage from the positive supply rail to ground. POTs have been used for the bottom resistor in each divider for easy tuning of offsets. As with the note generation, small deviations from the desired output can lead to noticeable distortions in the visual component of the project, which is why the offsets are tunable.

3.1.3 Note Movement (Druck)

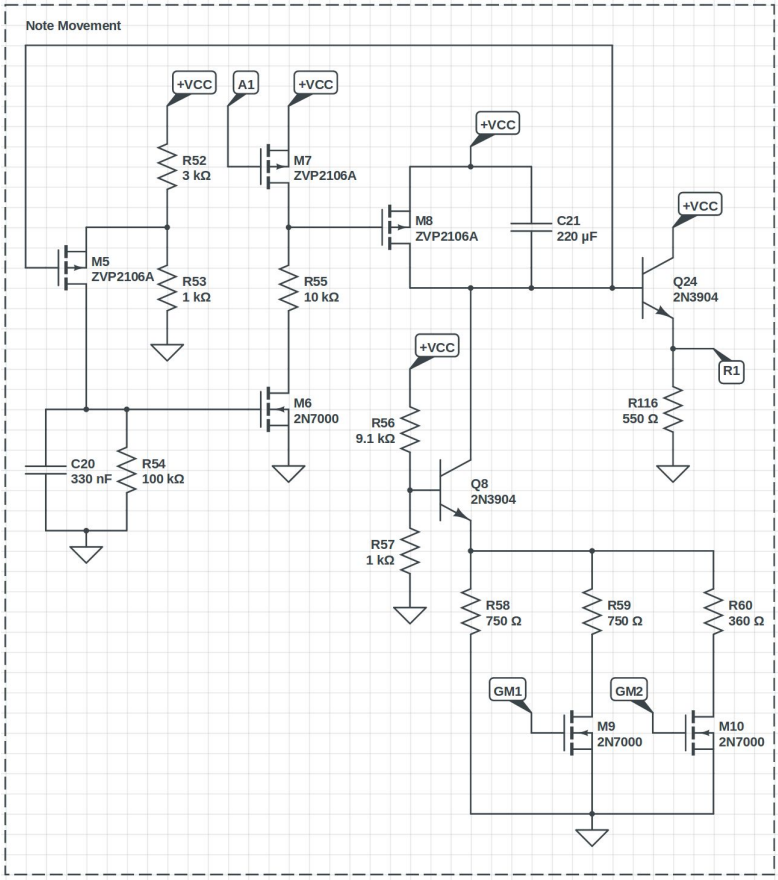

Figure 4: One of the four ramp generating circuits used to move the notes across the screen. The ramp is triggered upon receiving a signal from the Arduino (A1), and has a different shape depending on the game modes (GM1 and GM2)

Because the notes are moving across the ‘scope screen along the X-axis at a constant rate, the biasing circuitry for the X ‘scope channel is more involved. Figure 4 above depicts the biasing circuitry for a single note. The circuit acts as a monostable ramp generator, where A1 is the trigger. The output is normally idle at +15VDC. When triggered, the ramp decreases linearly to approximately +2VDC, and then jumps back up to +15VDC.

The trigger is an active low pulse. An Arduino handles the timing of the triggers for the four copies of this circuit and constitutes its own module to be discussed later. Upon triggering, M7 turns on and M8 turns off, which allows C21 to begin discharging. When the input, A1, returns to its high state, the gate of M8 is left floating because, at this point in time, M6 is also off. This floating node will keep M8 on through the discharging of C21 due to the very small amount of gate to source leakage current in MOSFETS.

In order to produce a linear discharge, Q8 acts as a current source in series with C21. The period of the discharge can then be found using:

$$i = C\frac{dV}{dt}$$

Where dV is +15V minus the gate voltage needed to turn on M5. The current source is made using the emitter follower configuration for Q8. The emitter resistor of Q8 determines the current of this current source, which controls the speed at which C21 discharges, which is the speed at which the notes will move across the ‘scope screen. In order to implement multiple difficulty modes, our project includes a Game Modes module which has two outputs, GM1 and GM2. Here, GM1 and GM2 are used to control the speed of notes by changing the resistance at the emitter of Q8. GM1 and GM2 are turned on sequentially, in that order, and each one halves the previous emitter resistance thereby doubling the note speed.

The node connecting Q8 to C21 is fed back to the gate of M5. This feedback ensures that the output jumps back up to +15VDC after it has reached its minimum value. The source of M5 is biased using R52 and R53 such that it turns on at a gate voltage of about 1V. When the gate hits 1V, M5 turns on, which turns on M6 and then M8, which finally re-charges C21. Once M6 and M8 are turned on, M5 will quickly turn off before C21 is fully charged. C20 and R54 are added to the gate of M6 so that C21 can continue to charge while the gate of M6 discharges. Finally, some time after C21 has been fully charged, M6 shuts off and the gate of M8 is left floating at ground. So as not to interfere with the linear discharge, the feedback node output is buffered using Q24 as an emitter follower.

3.1.4 Multiplexer Selector (Druck)

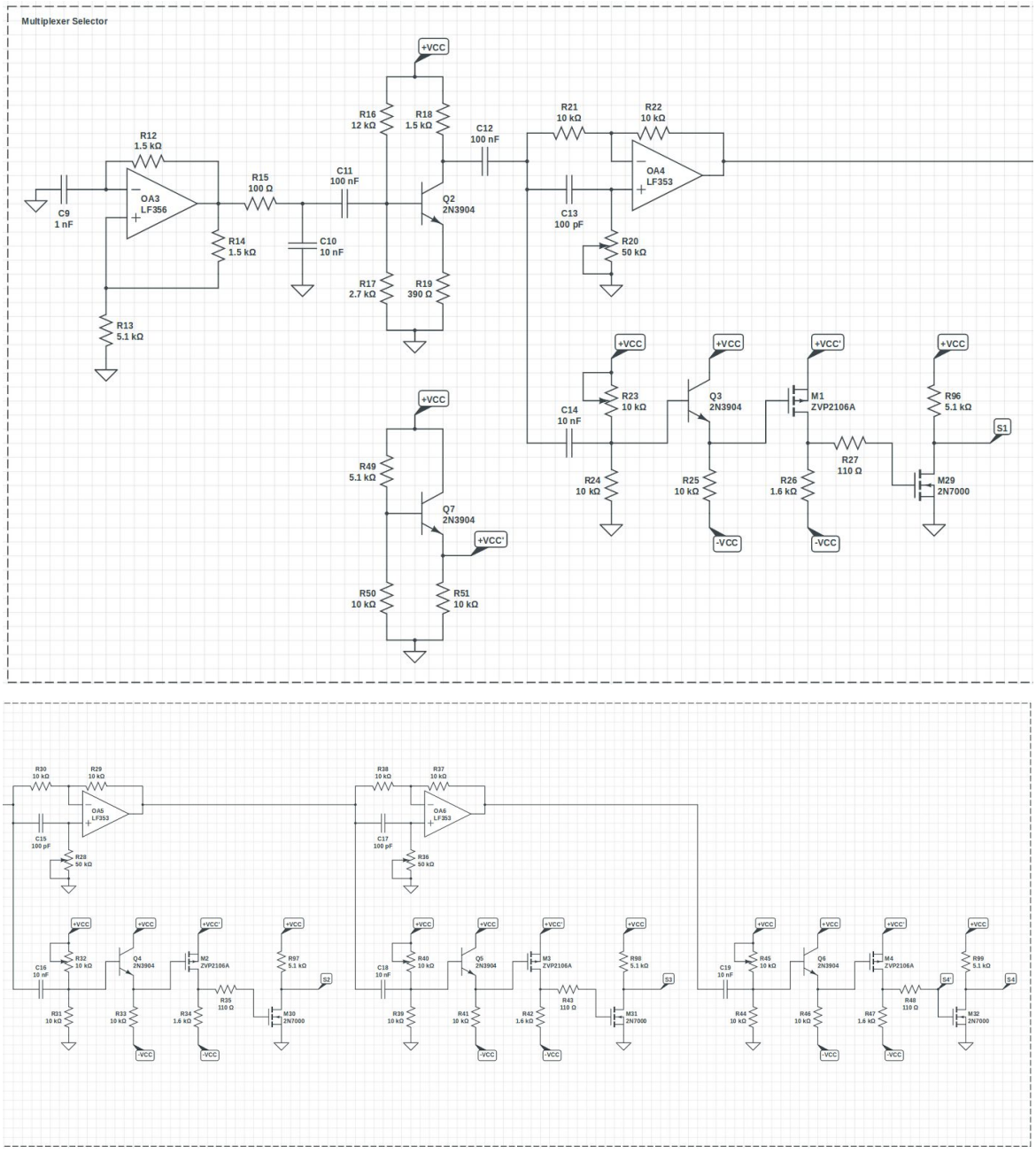

Figure 5: The multiplexer selector creates an amplified Schmitt Trigger Oscillator with a duty cycle of 25%. This first selector wave is then fed through 3 all-pass filters, which give the desired phase shifts for the remaining selecting waveforms.

The previous modules make up the basic building blocks for the visual component of this project. This module, and those that follow, assemble those blocks. The general

strategy for tracing out multiple notes is to draw a single note, while rapidly switching between the four different pairs of X and Y DC biases. This strategy is the analog equivalent to using a digital multiplexer with a rapidly switching select input. Take, as an example, just the Y channel biasing from the lane separation module. This module output four DC biasing signals. Those four signals will be the inputs to the multiplexer, and a rapidly switching select signal will quickly cycle the output through each of the inputs. Finally this output is added to the Y channel of the note generation, so that the overall effect is to draw a circle on the 'scope in four different Y positions at once.

This module, shown in figure 5, constitutes the select input signal in the process just described. It has four roughly square wave outputs. Each output is associated to the biasing for a single note. When that output is high, the DC biases for that note are fed through to the outputs of their respective multiplexing stages. Because the multiplexers are single output circuits, the outputs of this module cannot overlap when they are high. These specs confine the square waves output by this modules to have duty cycles of no more that 25%, and to be 90° out of phase from each other so as to never be high at the same time as any other output.

The circuit begins with an op-amp, OA3, Schmitt trigger with RC feedback to produce the initial oscillations needed to generate sinusoids [3]. A normal square wave output for this kind of oscillator has fourier series coefficients which are proportional in magnitude to $\frac{1}{n}$ at odd Harmonics, and are zero at even harmonics. However, due to the large voltage swing and frequency of this particular application, around 30 Vpp at 50 kHz, the output is skewed to become trapezoidal. The fourier coefficients of this wave are spaced even further apart than a normal square wave, which is ideal for this application. To take advantage of this fourier series to produce a sine wave, the oscillator output is fed through a passive low pass filter in order to attenuate all harmonics but the first. The resulting sinusoid is heavily attenuated. A common emitter amplifier, Q2, is used to undo the attenuation.

Because this module has four phase shifted outputs, the previous sine wave is passed through three all-pass filters. The resistors at the non-inverting inputs of these op-amp filters have been replaced with POTs for similar reasons to those in the Note Generation module. These sine waves will serve as inputs to makeshift comparators to produce the final square waves. Before they can be compared, however, they are AC coupled, biased, then buffered to ensure a clean output. The biasing, in the form of the voltage dividers before the emitter followers, determines the duty cycle of the resulting square wave. This is because a higher DC bias means more of the sinusoid will be above the DC voltage that it is being compared to. The reason for having these voltage divider biasing

networks with POTs for the pull-up resistor is so that the duty cycles of the outputs can be tuned to be as close to 25% as possible without causing overlap.

Finally, the buffered sinusoids are fed into the gates of p-channel MOSFETS, whose sources have been biased around +10V. The drains of the PFETs act as the outputs of our comparators whose inputs are the sinusoids and +10V minus the threshold voltage of the PFETs. Finally, these outputs are inverted using NFETs in preparation for the multiplexing stage.

3.1.5 Multiplexing (Druck)

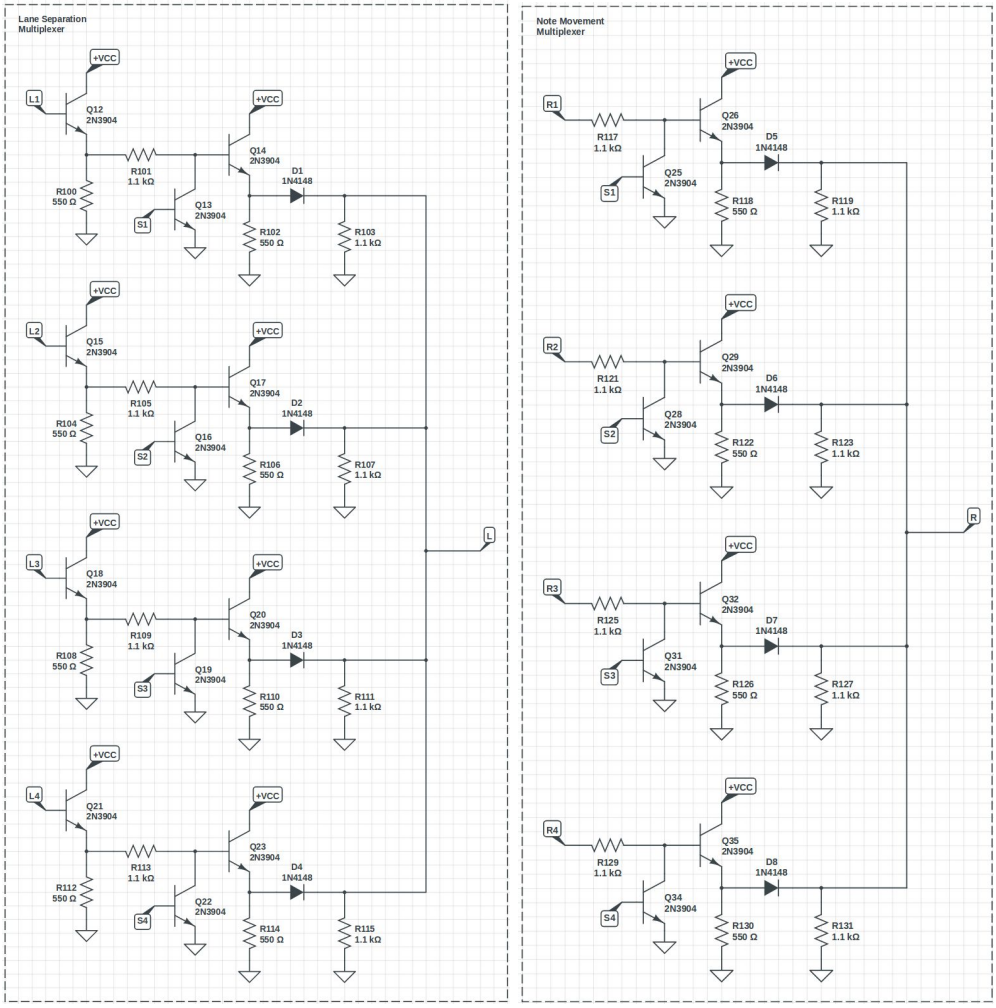

Figure 6: Left side: multiplexer for the lane separation module's biasing. Right side: multiplexer for the note movement module's biasing. Each MUX consists of a shunting transistor followed by a buffer then diode OR-er to sum up the four input channels.

The lane separation and note movement modules each have their own multiplexer depicted in Figure 6. Each multiplexer has four channels, where the input to each channel is one of the outputs of that multiplexer's respective DC biasing circuit. The inputs are fed through a resistor to a shunting transistor and a buffer. The base of the shunting transistor is tied to the selector output for that DC signal. Because the selector has a 75% duty cycle, the voltage at the base of the buffer transistor is equal to the DC biasing voltage 25% of the time, and equal to ground otherwise.

These four buffer outputs are then tied together using a diode "OR" to form the output of the multiplexer. In order for the diode "OR" to work, a small load is needed at the cathode end of the diodes. This is due to the capacitance within the pn junction of the diodes, which slows down the output waveform if the discharge path impedance is too high. This small load resistance is the reason for the buffering on the anode ends of the diodes. The low output impedance from the buffering is required to drive the load resistor.

3.1.6 Adding (Druck)

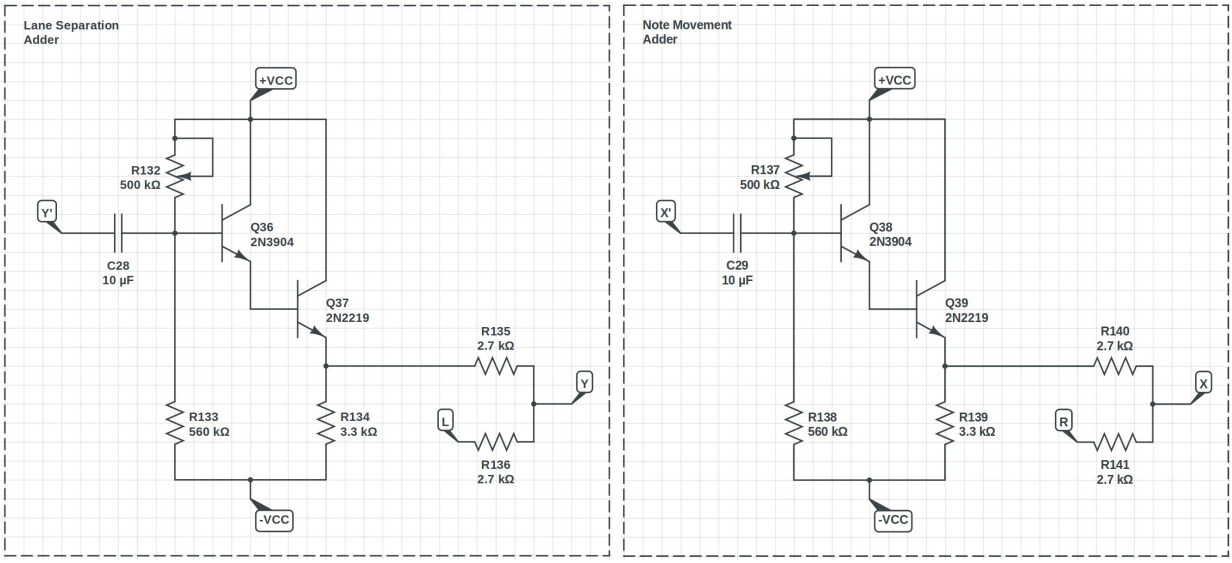

Figure 7: Each channel of the note generation module is buffered and then averaged with the multiplexer output. Left: Y channel, Right: X channel.

The last step to producing the visuals is to add the X and Y biases to the parametrically generated notes. For this, the simplest solution is to use a resistor averager. However, the output of the note generation has high impedance and cannot source much current. To fix this, a biased darlington emitter follower is used to buffer the note generation outputs as shown in Figure 7. The biasing ensures the output of the buffer is centered

at zero, which pre-compensates for the 1.4V drop across the two base-emitter junctions.

3.2 Audio (Daniel)

The following modules together make up the audio component of this project. They include the circuitry necessary to detect the validity user input, as well as the circuitry to create and amplify tones. The pitches of the four notes are the first four notes of the e-minor pentatonic scale with an error tone of d#. These notes are mixed together, and then amplified using a class AB amplifier to yield music.

3.2.1 Hit Detection (Druck)

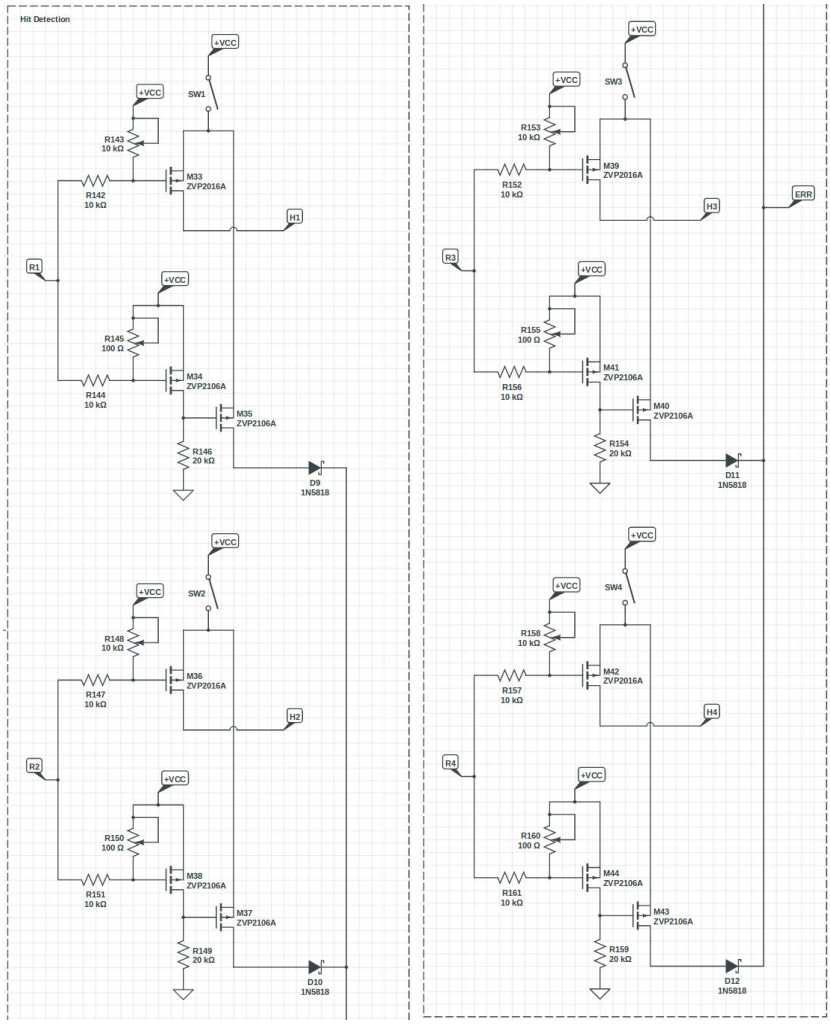

Figure 8: Four AND gates using a physical switch in series with a FET to produce the successful hit logic for each note. Similar logic for detecting a missed note, but the input to the FET is inverted.

Figure 8 shows the circuitry used to determine when the user has hit a button correctly or incorrectly. The outputs of this module will be used to directly power the oscillators in the sound generation module. The outputs H1-H4 are high when the user has hit a button at the right time, and are zero otherwise. The last output, ERR, goes high when the user presses a button at the wrong time, and is zero otherwise.

The outputs H1-H4 are the product of the series combination of the button corresponding to that output, a PFET whose gate is related to the X channel biasing of the note corresponding to that output, and the supply rail. In order for the output to go high both switches, the button and the FET, must be on. The PFET's gate has been biased using a pullup POT such that the FET turns on when the X channel DC bias reaches +3V or below. The result is that H1 goes high when SW1 is pressed and note 1 is near the bottom of the screen and so on for H2-H4.

The circuitry for generating ERR is very similar except that the PFET drain is connected to the gate of another PFET in order to invert the X channel biasing input. The final ERR signal the OR of four copies of this circuit, where the OR-ing is carried out using schottky diodes to reduce voltage loss.

3.2.2 Sound Generation (Daniel)

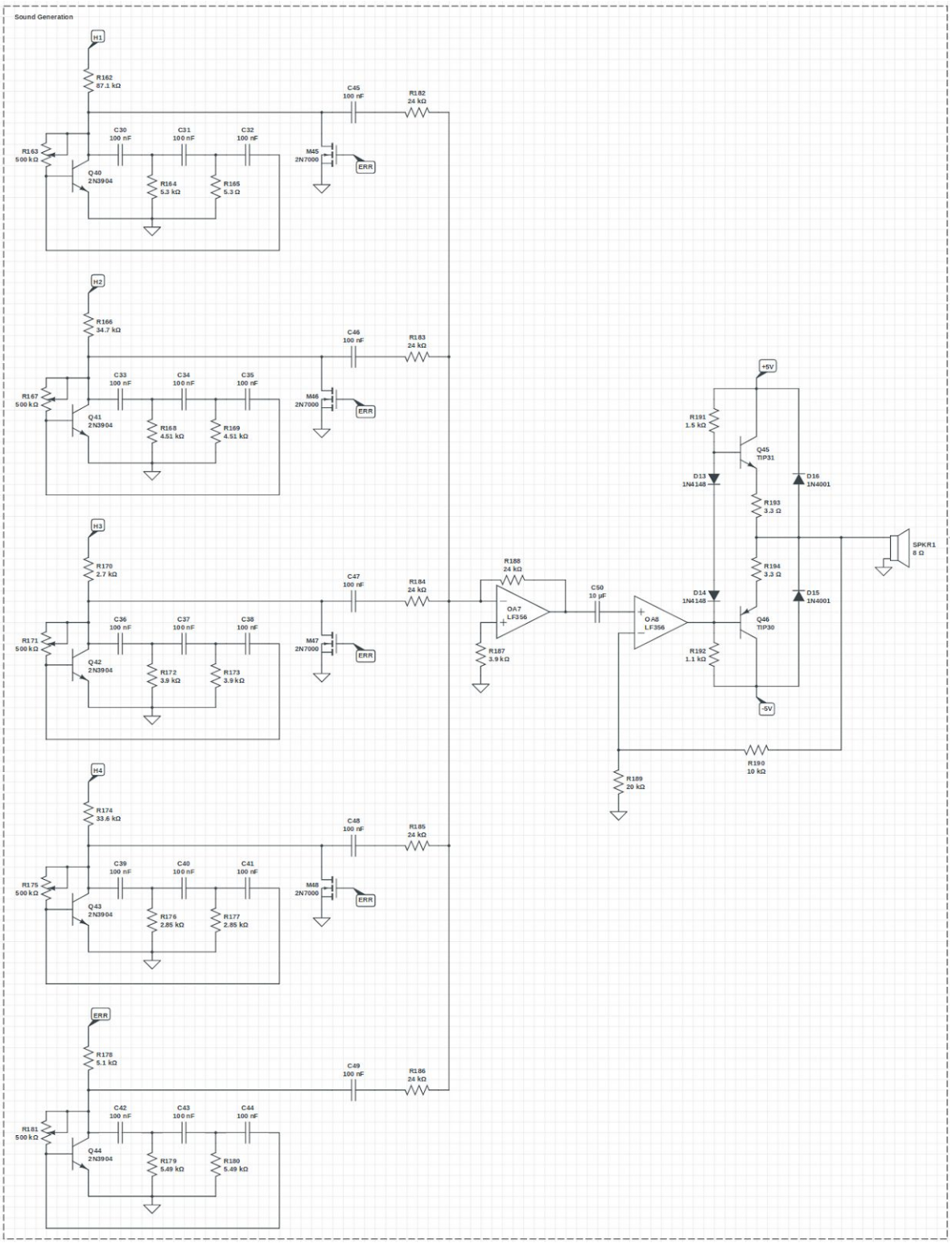

Figure 9: Five tone generation circuits that use the hit detection input to determine which tones will be mixed together. The output of the op amp adder is then amplified using a class AB amplifier and played.

This module makes use of the same style of phase-shift oscillator as in the note generation module. When the hit detection detects valid hits, it powers the respective oscillators, which are AC coupled through capacitors (C45 for instance), and mixed together using the op amp adder topology. Each individual sine wave has amplitude around 3VPP. We do not care about the exact magnitude of these pitches, as long as they are approximately equivalent. If the amplitudes were significantly different, certain notes would sound much quieter than other, which would be undesirable.

The error tone is a special case. In the event that the user presses a button during an inappropriate time, the remaining tones are shunted to ground, and only the error tone will play. The error output from the hit detection module powers both the error tone, and the transistors that shunt the remaining tones to ground.

The output of OA7 is then passed into the class AB amplifier. This class AB amplifier uses, in the push-pull section of the amplifier, TIP30 and TIP31 power transistors, which can handle the power requirements of the circuit. The rails of the amplifier are also different from the rails in the rest of the circuit - they are lowered to ±5V. Using the larger rail values will lead to severe overheating of the transistors, as well as large amounts of current through the speaker.

3.3 Game Modes (Daniel)

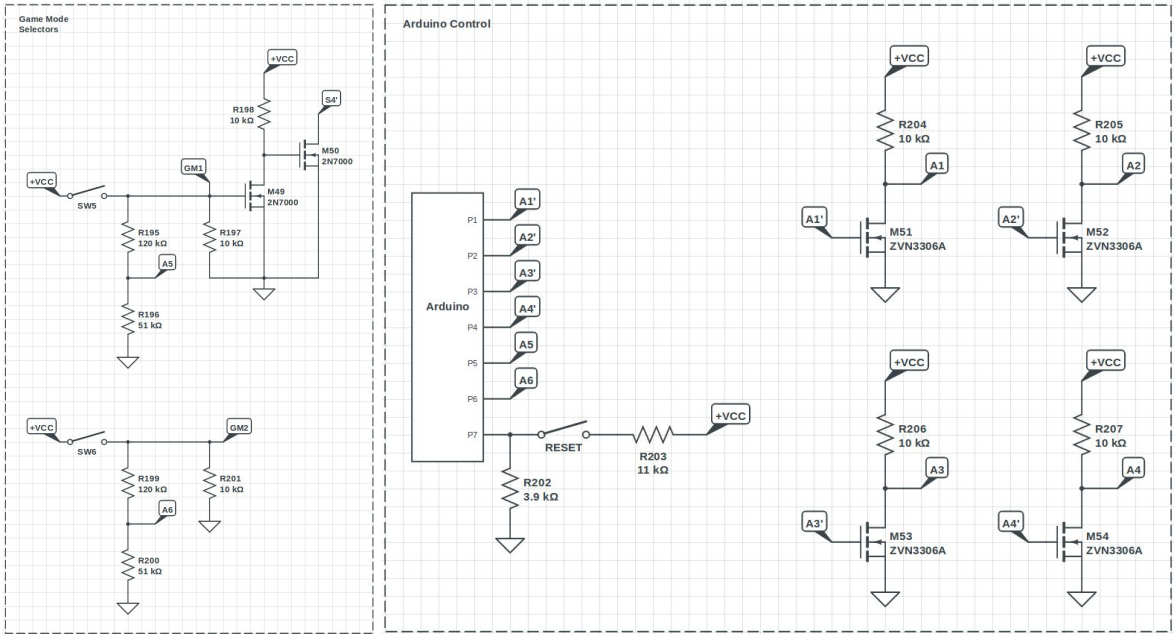

Figure 10: The two diagrams details the implementation of the game modes and Arduino operation.

The first four Arduino pins are used to control the triggering of the vertical movement of the four notes. They are passed through an inverter to meet the requirements of the ramps. When the Arduino pulses A1’ on for example, A1 sends a ground signal to the note movement module, which triggers the first note to move across the screen.

The seventh pin of the Arduino is used to reset the song. Lastly, the fifth and sixth pins receive inputs from the game mode selector module to indicate to the Arduino that it should make the notes trigger at a different rate to match the varying speed of movement of the notes.

The first game mode switch controls not only the speed of the notes using the same circuitry as the second game mode switch, but also controls the number of notes. When SW5 is open, GM1 is 0, and S4’ is tied to ground, which makes only three notes appear on the screen. However, when SW5 is closed, GM1 is +VCC, and S4’ is no longer tied to ground, which makes the fourth note appear.

4 Process

4.1 Challenging Modules (Druck)

By far the most challenging module in this project is the multiplexer selector (3.1.4). This module, which cycles the output of the multiplexer through its inputs by shunting the appropriate signals to ground, unexpectedly became the most time consuming module of the project to design and implement. To reiterate from section 3.1.4, the output for this module consists of four separate channels, each of which outputs a square wave with a duty cycle of 25% or less and none of which should be high at the same time as another channel. This means each channel is 90 degrees out of phase from its neighbor. The design process for this module included many iterations which were fully built and tested but did not meet the spec.

The initial design simply used a five stage inverter ring oscillator, and took the outputs from four consecutive stages of the oscillator. Unfortunately, this design was fueled by the misconception that such a ring oscillator with N stages would have an output with 1/N duty cycle. In reality, adding more stages to an inverter ring oscillator puts the duty cycle of the output closer to 50%. Furthermore, the smallest duty cycle obtainable with this scheme, using only a three stage oscillator, is roughly 33%. Therefore it is

impossible for this design to meet the spec of this module, regardless of stage count, due to the hard 25% duty cycle cap.

The next design involved generating a single square wave with the desired duty cycle, and passing it through 3 consecutive all-pass filters in order to shift the wave and generate the other three channels. The general scheme was inspired from the note generation module, which uses an all-pass filter to generate two sine waves that have a phase shift. For similar reasons as the distortion introduced by the all-pass filters discussed in 3.1.1, this design did not work. all-pass filters only produce the desired phase shift for a single frequency, not for a square wave which has many. The result of passing a square wave through an all-pass filter was very heavy distortion, which made the output unusable.

However, this design inspired the final design for this circuit. The idea to use three all-pass filters to generate the appropriate phase shifts remained, but the inputs were changes to sinusoids instead of square waves. Square waves with the appropriate duty cycles were then constructed from these sinusoids as detailed in 3.1.4.

4.2 Future Improvements (Daniel)

The primary improvement that could be implemented in the future would be to clean up the visuals displayed on the ‘scope. In the current design, any error in the multiplexer selector causes ghosting to appear on the scope, which is shown in Figure 11 below.

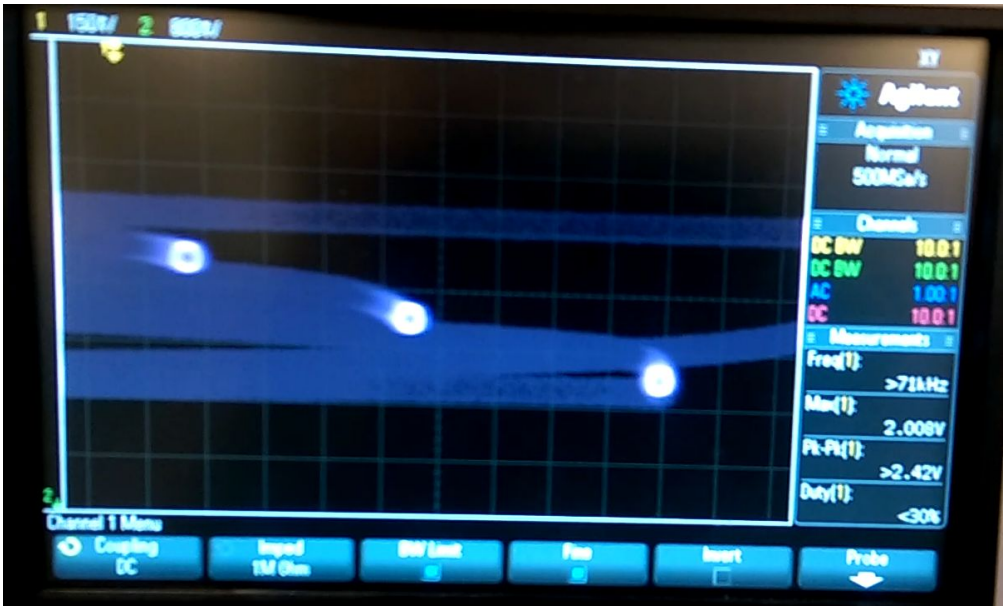

Figure 11: The oscilloscope showing ghosting when drawing the notes on its screen.

One way this issue was mitigated after the completion of this project was utilizing the Z mode on the oscilloscope. This allowed us to only send samples to the ‘scope when the multiplexer selector was selecting a single input.

However, due to noise in other parts of the circuit and the reliance on manual tuning, the ghosting was not eliminated entirely. Using more precise component values to remove manual tuning across the entire circuit would be a potential future improvement. This would make the circuit more robust to error.

5 Conclusion (Daniel)

Ultimately, circuitry is created in order to solve a problem. In the case of this design, the aim is to create a circuit that could appeal to a wide audience and provide entertainment. Additionally, in the case of games, it is always desirable to create a satisfying user experience with both visual and audio effects. Faithfully reproducing the fun of *Guitar Hero* involved over 1500 circuit components and many challenges concerning the stability of operation in many modules. However, despite these concerns and potential future improvements, the circuit described in this document creates a fun and interactive game with interesting design components.

For our team, what makes a circuit interesting is not only that it is applicable to a wider audience and interactive, but also that it is practically usable and novel. An interesting circuit is also challenging to design. Many parts of our circuit require us to think creatively to implement digital functionalities in an analog setting. This is something that challenges us by forcing us to find solutions to real-world non-idealities that don’t exist in digital electronics.

6 References

Coates, Eric. "RC Phase Shift Oscillators." *Learn About Electronics*. Eric Coates, 17 Feb. 2017. Web. 15 May 2017.

Entner, Robert. "Ring Oscillator." *Ring Oscillator*. Institute for Microelectronics, 28 Mar. 2014. Web. 15 May 2017.

Hom, Gim. "Op Amps." 6.101 Lecture. MIT Room 34-304, Cambridge, MA. 7 Mar. 2017. Lecture.

Keim, Robert. "Focusing on Phase: The All-Pass Filter." *All About Circuits*. All About Circuits, 08 Nov. 2016. Web. 15 May 2017.