Affordable Analog Synthesizer

By

Breanne Warner

Michael Jamrozy

Yash Bhushappaga

Final Report for ECE 445, Senior Design, Fall 2021

TA: Feiyu Zhang

8 December 2021

Project No. 18

Contents

1. Introduction

1.1 Problem and Solution

After analyzing the build costs and average market value, our team realized that music synthesizers are extremely expensive making them unreasonable to buy for a lot of individuals. There exist people who are interested in creating music using synths but may not be able to do so because of budget restrictions. The objective of the project is to create an affordable analog synthesizer. Also, According to Technavio, “the music synthesizers market is poised to grow by USD 62.90 million during 2021-2025, progressing at a CAGR of over 2% during the forecast period” [10]. Being able to create an affordable model holds values with the growth in market and demand for music synthesizers, as well as documentation for the homemade solution that we make.

Creating the synthesizer from scratch and utilizing cost analysis to obtain cheaper parts will help implement an effective and cost effective product which will meet the demands for an affordable music synthesizer.

1.2 High Level Requirements

1. To produce sounds using sawtooth wave and square wave with a controllable duty cycle. To be able to recreate the following kinds of effects: tremolo (variations in volume), vibrato (variations in pitch), sweeping cutoff filter, and resonance.
2. Produce the correct pitches for at least 24 consecutive keys, from the MIDI keyboard centered around A4 (440Hz) being able to produce sounds between 220 Hz and 880 Hz [9].
3. Have the ability to read key inputs from a file containing a sequence of key events on an SD card and play them back through the synthesizer as if they were notes being played on the keyboard.

1.3 Summary

We looked at many early analog synthesizers as inspiration for our design, especially the Minimoog. This creates sounds using a technique called subtractive synthesis, where oscillators produce a wave with frequency corresponding to the key being pressed. These waves are typically rich in harmonics, like square waves and sawtooth waves, and then they go through a low pass filter, hence the name ‘subtractive synthesis.’ The wave is also shaped by an envelope generator for amplitude and cutoff frequency. This makes it possible to produce a sound which, for example, starts off with higher harmonics that decay as the note is held, mimicking some acoustic instruments. Additionally, various control voltages can be modulated by a low frequency oscillator, such as the cutoff of the filter or the frequency of the oscillator. In our design we attempted to recreate the oscillator, low pass filter, amplifier, envelope generator and low pass filter, and use a microcontroller to receive input from an external MIDI keyboard or SD card.

2 Design

2.1 Block Diagram

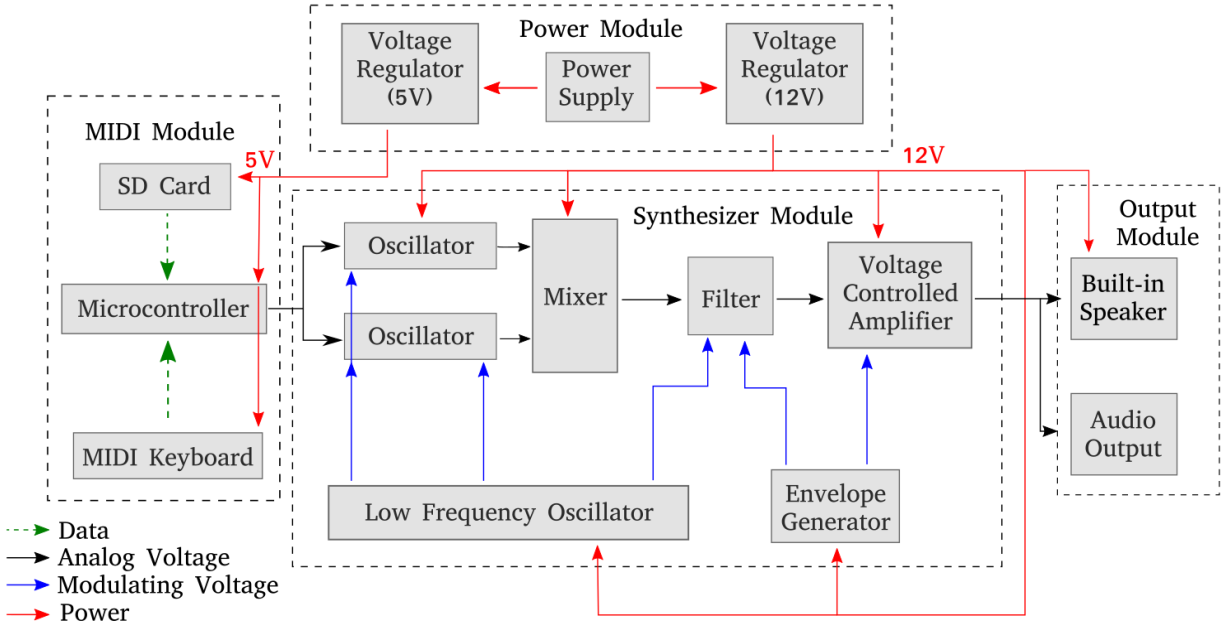


Figure 1: Block Diagram for All Modules Combined

2.2 Physical Design

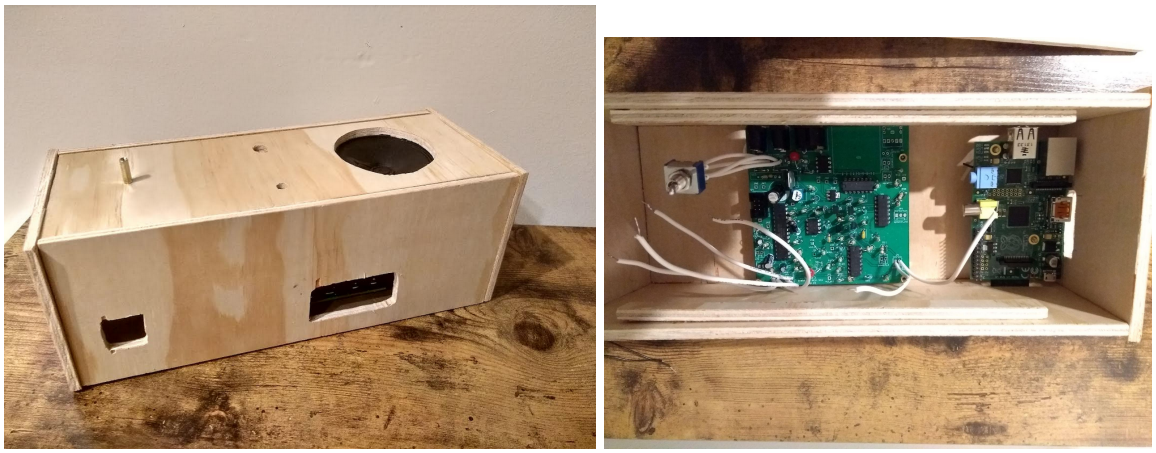


Figure 2: Box with PCB inside it

2.3 Module Descriptions

2.3.1 Microcontroller

The microcontroller is part of the MIDI module and is mainly used to interface with the external MIDI keyboard and SD card and DAC. It's purpose is to receive the MIDI messages and produce a voltage on the DAC, ranging from 0 to 5V, that corresponds with the frequency of the note. The DAC and SD card are connected through SPI pins, and the ISP programmer is also connected to the SPI bus. The MIDI connector connects to the ATMega through it's UART pin. An opto-isolator (in this case the 6N138) is required to get a UART signal from the MIDI connector [6].

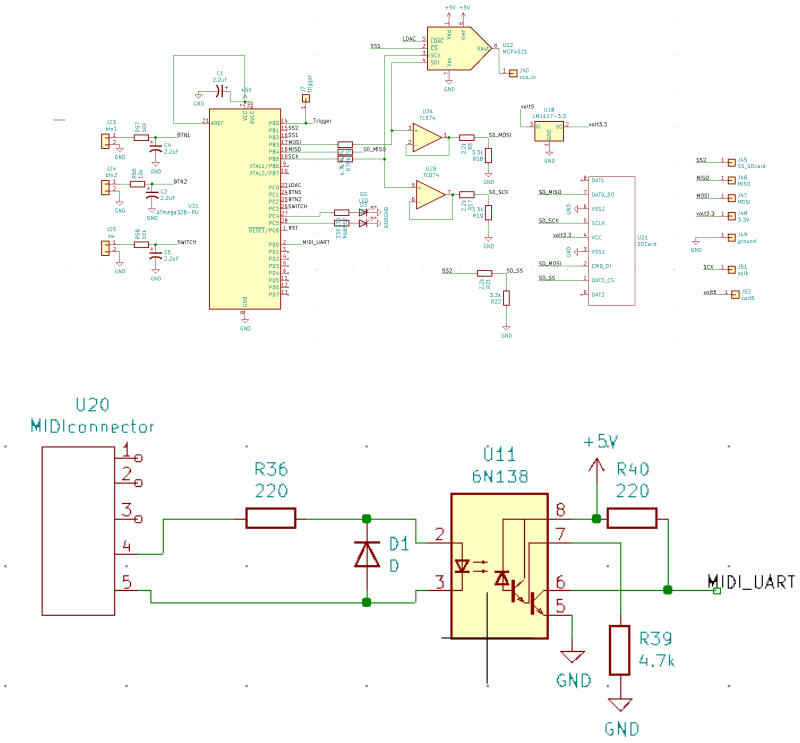


Figure 3: MIDI Input to UART

2.3.2 Voltage Controlled Oscillator

The voltage controlled oscillator is designed to create a complex waveform whose frequency follows an exponential curve with the input voltage. In this way, a control voltage of 2.5 would result in 440 Hz, the A4 note, and every 100 mV increase would increase the frequency by a half step (difference between two consecutive keys on a keyboard). The waveform is a mixture of a sawtooth and square wave with a ratio controllable by a potentiometer. The VCO design works by using an exponential voltage-to-current sink [4], which is then integrated by an op amp to produce a ramp up [5]. Once this reaches 5V, a transistor allows the capacitor to discharge, resetting the voltage back to zero and marking the end of one period. A square wave is generated from the sawtooth with an op amp used as a comparator. This was a trade-off between using two separate VCOs, like are used in many synthesizers we used as inspiration, and increasing the complexity of the design. Our design has the limitation that both

waveforms have the same frequency, while in real synthesizers it is often possible to change the relative pitch between the oscillators to produce various intervals.

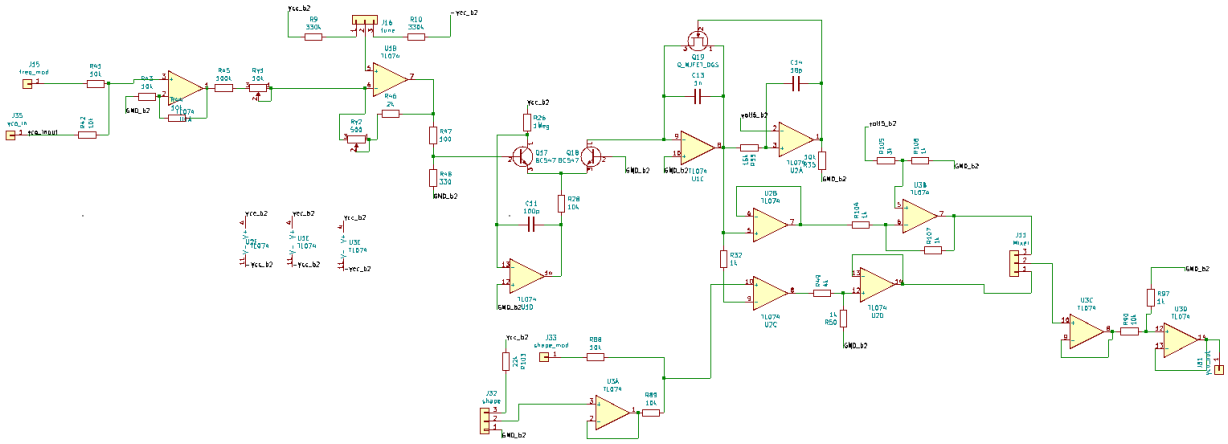


Figure 4: VCO Schematic

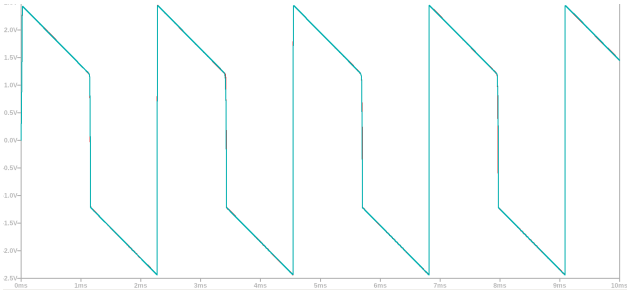


Figure 5: VCO Simulation

2.3.3 Voltage Controlled Filter

The VCO's output waveform then gets filtered by a low pass filter with resonance. Our design is based on the Moog Ladder Filter patent [3], where the control voltage changes the bias current to some transistors which affect the filter's cutoff frequency. Resonance is achieved by using the output as negative feedback, and the amount of resonance is controlled by a potentiometer. To generate the bias current, the same exponential voltage-to-current sink is used as in the VCO.

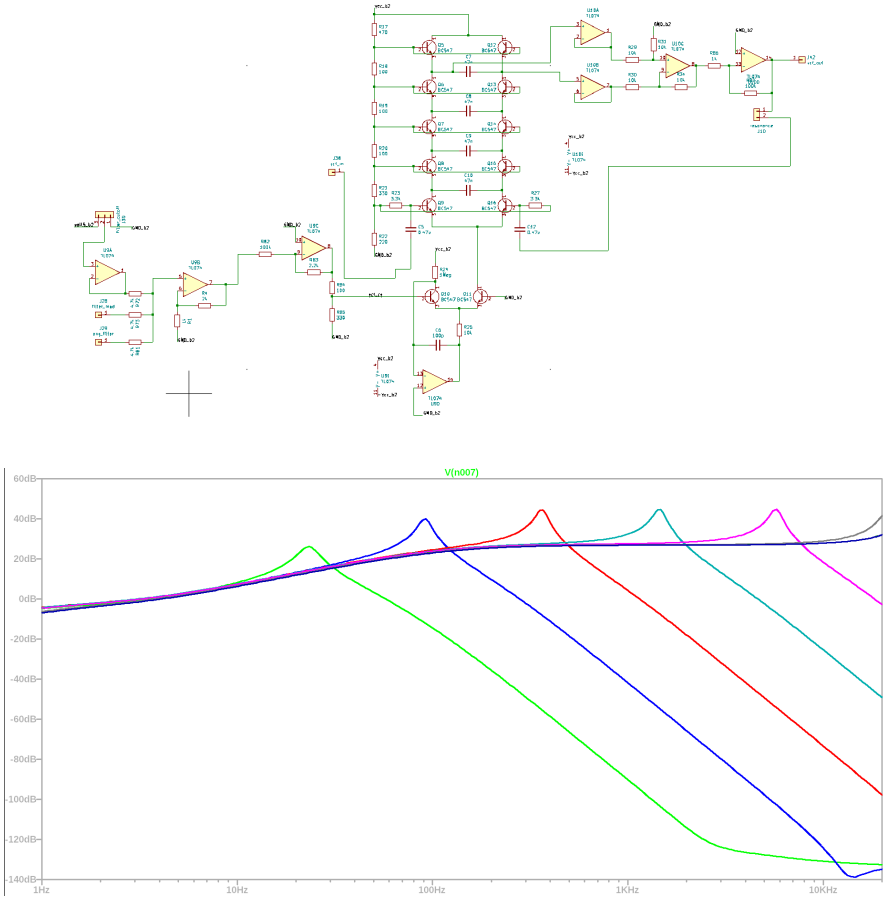
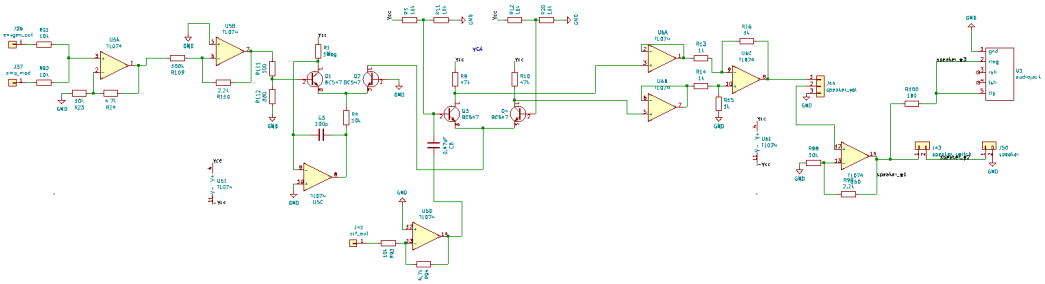


FIgure 6: Frequency Response of VCF with Increasing Control Voltage, Maximum Resonance

2.3.4 Voltage Controlled Amplifier

The filtered waveform then is amplified by the voltage controlled amplifier. The control voltage comes mainly from the envelope generator, which triggers whenever a key is pressed. This voltage creates a bias current to a differential pair of transistors, which controls their gain. Then op amps are used to convert the differential signal to a single-ended signal. Our VCA design isn't perfect, and the control voltage has a limited range where it is good, so we planned to keep our inputs within this range. This is because the collectors of the transistors are very close to 12V when the bias current is small, and the op amps don't work very well so close to their supply voltage.



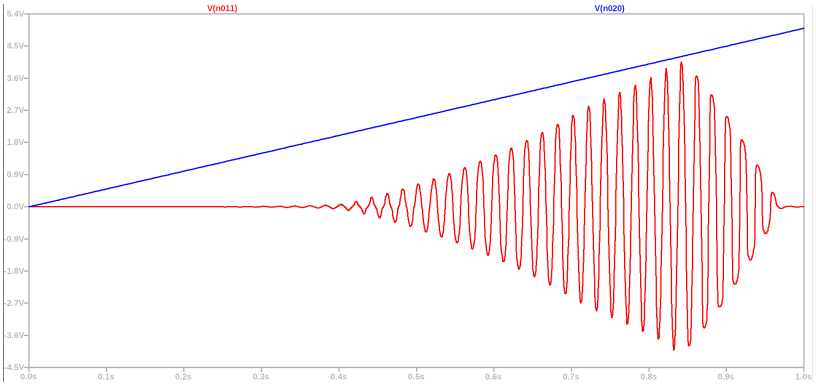
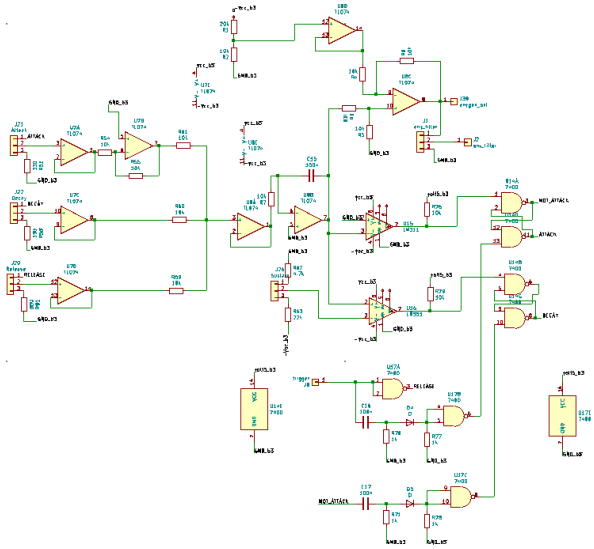


Figure 7: VCA. Blue: Control Voltage, Red: Output

2.3.5 Envelope Generator

We implemented an ADSR envelope generator, which stands for attack, decay, sustain and release. Attack is how long it takes the envelope to reach its maximum value once the key is pressed, decay is how long it takes for it to drop down to the sustain level, and release is how long it takes to drop to zero once the key is released. We implemented the envelope generator using comparators and some logic gates to determine which of the ADSR stages it is currently in, and this signal gets integrated to produce a linear ramp. Potentiometers for the various stages allow for changing the integrated voltage, making it possible to control the timing of the various stages.



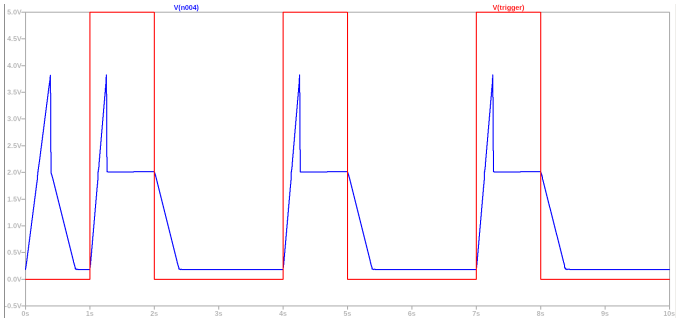
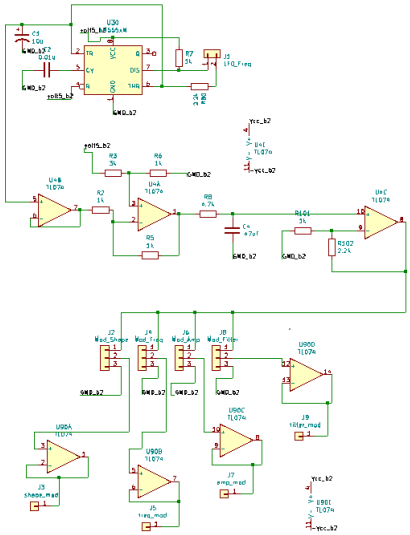


Figure 8: Envelope Generator. Red: Trigger, Blue: Output

2.3.6 Low Frequency Oscillator

The low frequency oscillator is intended to be used as a modulating input to the various control voltages in the other subcircuits. We implemented it with a simple 555 timer circuit, taking the output from the charging and discharging capacitor to give something close to a triangle wave. A potentiometer controls this charging speed, thus changing the frequency. The output is then shifted to be centered around 0V and then the signal goes to the four modulating inputs we included in our design.



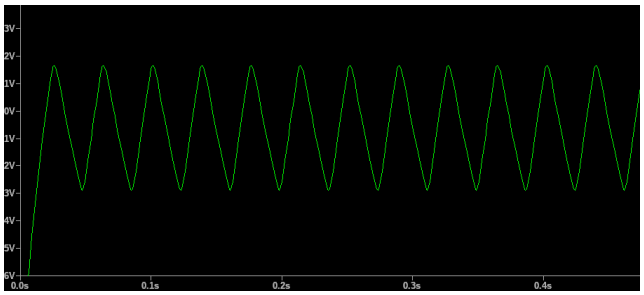


Figure 9: LFO Expected Output

3. Design Verification

Unfortunately we did not get most of our original design working as specified by the verification in Appendix A. These verifications for the synthesizer subsystem mainly involve viewing the signal through an oscilloscope and checking that it matches with the simulated plots seen above. We did however get a few parts to work, which will be explored next.

3.1 Synthesizer Module

Refer to the Requirements and Verification table in Appendix A.1 for the Synthesizer module. The verification that was able to be demoed is the output of the Low Frequency Oscillator (LFO). The voltage controlled oscillator, VCO, input for the LFO was simulated using the oscilloscope. As referenced in the design, the LFO takes in the VCO output. During the build the VCO output was not transmitting which led the team to building a VCO output simulation on the breadboard to then create the expected output from the LFO as seen below.

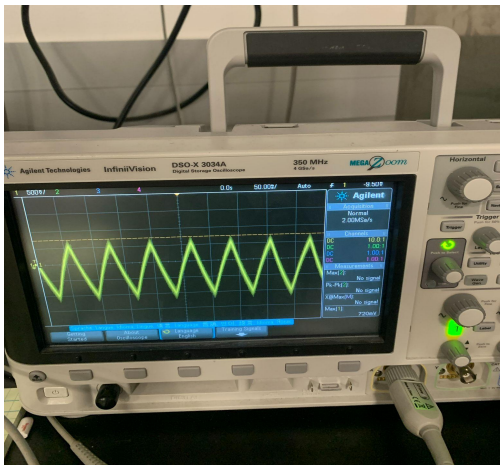


Figure : Oscilloscope Output from LFO

3.2 Microcontroller Module

Refer to the Requirements and Verification table in Appendix A.2 for the microcontroller module. In order to test the microcontroller, we probed the output from the DAC to see if there was an output. The DAC was able to output 1V and 3V. Also, to signify that the microcontroller was programmable we wrote a program that would have a blinking light display to show that the program was loaded to the microcontroller, refer to Appendix B.

3.3 Power Module

Refer to the Requirements and Verification table in Appendix A.3 for the microcontroller module. To test the power module we were able to use the power supply with +12V and -12V as well as +5V and -5V. The power supply for the synthesizer module was verified by probing the power supply and the readings were accurate.

4. Costs

4.1 Parts

Table 1. Parts List

Item	Unit Cost	Quantity	Total Cost	Manufacturer
100k Potentiometer	$0.86	20	$17.20	Adafruit
10k Potentiometer	$0.95	2	$1.90	Adafruit
Knobs	$0.45	20	$9.00	Adafruit
Power Switch (DPDT)	$1.25	1	$1.25	Adafruit
Switch (speaker, SD/Keyboard)	$0.64	2	$1.28	Mouser
Button	$0.95	2	$1.90	Adafruit
Speaker	$1.95	1	$1.95	Adafruit
Power Supply	$14.95	1	$14.95	Adafruit
DC Barrel Jack	$0.95	2	$1.90	Adafruit
Audio Jack	$0.68	1	$0.68	Mouser
MIDI Connector	$1.75	1	$1.75	Mouser
SD Connector	$1.95	1	$1.95	Sparkfun
TMA 5V Switching Regulator	$4.50	1	$4.50	Mouser
ATMega328	$2.58	1	$2.58	Digikey
DAC (MCP4921)	$2.58	1	$2.58	Mouser
Quad Op Amp (TL074)	$0.75	18	$13.48	Mouser
Optoisolator (6N138)	$0.80	1	$0.80	Digikey
Diodes (1N914)	$0.10	10	$0.97	Mouser
NPN BJT (BC547)	$0.18	20	$3.50	Mouser
Quad NAND (74HC00)	$0.55	2	$1.10	Mouser
Voltage Comparator (LM311)	$0.53	3	$1.59	Mouser
3.3V Regulator (LM1117)	$0.60	1	$0.60	Digikey
N-channel JFET (J111)	$0.41	3	$1.23	Mouser
ISP Programmer Headers	$0.46	1	$0.46	Digikey

MTA 100 2-pin connector	$0.14	15	$2.09	Digikey
MTA 100 3-pin connector	$0.18	15	$2.72	Digikey
WM4200 2-pin header	$0.16	15	$2.39	Digikey
WM4201 3-pin header	$0.22	15	$3.35	Digikey

4.2 Labor

In order to analyze the fixed labor costs for having a three person team, it was estimated based off of an average UIUC ECE graduate makes per year in 2021. According to The Grainger College for Engineering ECE graduates in 2018-2019 starting salaries average $91,781/year [2] based on 40 hour work weeks on average. Breaking this down into hourly wage of $44.16/ hour on average, and the expectation for the team will be that each person commits at least 10 hours a week to this course.

$$3\ team\ members\ \times (\$44.16/hour) \times\ 10\ hour/week\ x\ 10\ weeks\ x\ 2.5\ =\ \$33,120$$

The total cost is calculated by combining the labor cost as well as how much the total is for the parts being used. The total amount for all the parts is $99.63 thus the total amount is:

$$\$33,120\ +\ \$99.63\ =\ \$33,219.60$$

5. Conclusion

5.1 Accomplishments

Upon testing the design utilizing the original PCB and realizing that the PCB orders would not suffice for the expected output a new version of the project was created in order to meet our high-level requirements. We utilized a Raspberry Pi which generated audio samples in real time and played them through an external speaker. We did not succeed in connecting it to a MIDI keyboard, so we instead used a standard keyboard where one row of keys simulated one octave of a piano keyboard, and the bottom row of keys controlled some of the effects that were originally intended to be set with potentiometers. When connected to a monitor, these controls are displayed as well as debug information we used while programming. The Raspberry Pi boots from an SD card, so this gave us an easy way of meeting our original requirement to play songs from the SD card, which we implemented. They are stored as text files with a sequence of notes specified by duration, note and octave. For example “8F4” represents an eighth note F sharp (capitals indicate sharp) note in the fourth octave, and a file contains many such words separated by whitespace. This version of the project utilizing the Raspberry Pi is very compromised; it is not analog like we wanted, there is some delay between pressing a key and hearing it, the use of a computer keyboard is not ideal, it takes a long time to start working because of the boot time, it only works with an external speaker, and the kinds of sounds we could get from it were fairly limited. It

sounded more like an old video game than what we had anticipated, though this likely could have been fixed with a better program. However, it can produce basic waveforms at the pitch corresponding to the key being pressed, shape the waveform with an envelope, filter them with a low pass filter, and play back notes from an SD card, so it mostly met our high level requirements.

We also managed to get some parts of our original design working, mainly the low-frequency oscillator, the power subsystem and parts of the microcontroller. We could successfully program the microcontroller and output voltages between 0-5V through the DAC. We also built a wooden case to contain our PCBs.

5.2 Uncertainties

Looking into the future of the project, there are many things that could be improved upon and additional work added on. First, the overall PCB design would need to be improved. The reason for splitting up our original PCB design was to accommodate for the size constraints when ordering PCBs through the school. There was not enough room on one 10cm by 10cm PCB to fit all the necessary components and connections on one board. Even after splitting it between three boards the components were very close together, which made troubleshooting difficult, and since we needed power and signal wires running between the PCBs they became very difficult to manage. We had intended to use four types of MTA connectors (3 pin male/female and 2 pin male/female) for connecting potentiometers to the board and connecting wires between boards, but we either forgot to order two of these kinds or misplaced them, forcing us to solder wires directly to the board. This also made the PCBs very difficult to manage because now connections couldn't be easily detached. This leads into another uncertainty about if this had an affect on the performance of the PCB. For example, each component was very close together which may have created issues with connectivity between parts as well as it being difficult to debug. The future project work would be to redesign and simplify the synthesizer module, exploring different ways to create square, triangle and saw-tooth waves. This could include cutting down on the size of the design, only using one PCB, or even using a DSP chip to assist in the production of sound.

This was our first time using KiCAD and designing a PCB, and the PCB turned out to be complicated and divided among three boards. It is not that surprising then that we found errors in our PCB layout. For example, when troubleshooting our voltage-controlled oscillator we discovered that two pins which should have been connected, the output of an op amp and a resistor, were not. In our schematic they were connected, but somehow we missed connecting them in the PCB editor and didn't notice any issues. We soldered a wire between these pins which fixed this one specific issue, but still the VCO as a whole didn't work and there were likely other issues in the PCB. We did not have the time to go through every part of the design and find all possible errors.

One uncertainty of the project is the power supply. The power supply we chose is very noisy which causes a peak to peak voltage difference of 300mV. While there were many expensive power supplies with +12V and -12V, we needed to buy a cheap one to stay on budget, and we forgot to consider noise when purchasing it. A 100mV change to the VCO input causes a different note to be played, so 300mV peak-to-peak noise would have been unacceptable, and we observed noise in the power supply causing

noise on this input. It might be necessary to pay the high price for a good power supply in order to have a usable analog synthesizer.

Additionally, there was uncertainty in some of the components we used. We planned to use a switching 5V regulator, but it wasn't working in our PCB and when we tested it on its own we couldn't seem to get it working. We probably were connecting it wrong, but still it wasn't clear from the datasheet how it was supposed to be connected. This wasn't a major issue because we had a linear voltage regulator as a backup, but we had a similar issue with the 6N138 optoisolator, used for MIDI input. There is an official reference schematic using an opto-isolator for getting UART from MIDI [6], but the particular component used there is obsolete and we could not buy it. Instead we used a modern optoisolator, but we had difficulty figuring out how to use it. In the end our MIDI input wasn't working and although we didn't find out the specific issue, it was likely a problem with how we connected the 6N138 in our schematic, since we ended up making an educated guess on how to use it and that probably was wrong.

5.3 Future work

If we were to work on this in the future, we would focus on getting the PCB right. We would use only one large PCB, and we would place the components in a more ordered and logical way and add more space between the components to make troubleshooting easier. This would also make placing the traces much easier, and it would probably help us catch issues like missing traces. Additionally, we would build out our circuit designs on a breadboard first, before ordering the PCB, to make sure our designs for the MIDI input and synthesizer subsystem are working. We tried doing this, but it was taking up too much time and the PCBs were already ordered so there wasn't much we could do if we found out the design didn't work at this stage. We would also make sure to buy a better, less noisy power supply.

5.2 Ethical considerations

In the planning stages of the project, and after reviewing IEEE Code of Ethics document, section 7, we did not perceive there to be a privacy risk of data or information that is stated in the first statement of the IEEE Code of Ethics. We did not use any user specific data in the design and therefore no user security was at risk or needed to be protected. In section 1.5 the ethics document specifies that it is a professional's opportunity to acknowledge and correct errors [1]. Throughout the process of designing and building, the team was able to uphold this and work towards effectively debugging to correct errors that we came across. For example, we ensured that we went through verifying why something would not work if we could not fix it in time for the final demo. Also, in the design stage, we planned to uphold the code detailed in the ACM Code of Ethics and Professional Conduct [7]. Specifically, ACM states in Section 2, Professional Responsibilities, professionals working should ensure that they are creating high quality work and communicating with either stakeholders or team for transparency. We were able to uphold and maintain this while developing the project as the team communicated with each other when there were issues or when a new direction for the project was proposed and made. It was important to do so so that the team was always aware of any design changes and challenges.

Also, in regard to ethics at a design level, many books and other resources exist for our project to explain the circuits in synthesizers, and our project has been implemented before. Schematics of many old synthesizers can be found easily online. Some particular circuits have also been patented, though most of these patents are old and have expired, such as the patent for the Moog ladder filter [3]. If we use any designs from some reference material, patent or schematic, we will need to first make sure that we can legally use it and then reference where it came from.

References

[1] "IEEE Code of Ethics", ieee.org, 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 15- Sep- 2021].

[2] Grainger College of Engineering, “2018-2019 Illini Success Survey Outcomes* UNDERGRADUATE STUDENTS,” https://ecs.engineering.illinois.edu/files/2020/04/UG-ECE-2018-2019.pdf. [Online]. Available: https://ecs.engineering.illinois.edu/files/2020/04/UG-ECE-2018-2019.pdf. [Accessed: 22-Sep-2021].

[3] R. A. Moog, “ Electronic high-pass and low-pass filters employing the base to emitter diode resistance of bipolar transistors,” United States Patent 3475623A, Oct. 28, 1969.

[4] A. Lanterman, “Exponential Voltage-to-Current Conversion & Tempco Resistors,” in *ECE4450 (Analog Circuits for Music)*, 5-Apr-2021.

[5] H. Chamberlin and H. Chamberlin, “Basic Analog Modules,” in *Musical applications of microprocessors*, Indianapolis, Indiana: Hayden, 1987, pp. 177–220.

[6] B. J, “ MIDI Tutorial ,” MIDI tutorial. [Online]. Available: https://learn.sparkfun.com/tutorials/midi-tutorial/hardware--electronic-implementation. [Accessed: 27-Sep-2021].

[7] “The code affirms an obligation of computing professionals to use their skills for the benefit of society.,” Code of Ethics. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 15-Sep-2021].

[8] “AnalogWrite,” analogWrite() - Arduino Reference. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/. [Accessed: 28-Sep-2021].

[9] Note names, MIDI numbers and frequencies. [Online]. Available: https://newt.phys.unsw.edu.au/jw/notes.html. [Accessed: 30-Sep-2021].

[10] Technavio, “Global music SYNTHESIZERS MARKET: 48% growth to emerge from North America DURING 2021-2025: Technavio,” Global Music Synthesizers Market | 48% Growth to emerge from North America During 2021-2025 | Technavio, 21-Apr-2021. [Online]. Available: https://www.prnewswire.com/news-releases/global-music-synthesizers-market--48-growth-to-em erge-from-north-america-during-2021-2025--technavio-301273258.html. [Accessed: 15-Sep-2021].

Appendix A Requirement and Verification Table

Table 1 Synthesizer System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Voltage-controlled oscillator, with a voltage input from 0-5V, produces square and saw waves with frequency dependent on input voltage..	a. Probe sawtooth and square wave wires, check that they look good on the oscilloscope. Increase control voltage from 0 to 5V and verify that frequency increases.	N
1. A mixer combines the outputs of the two oscillators into any ratio.	a. View the oscilloscope as the potentiometer is turned, verify that square and sawtooth are mixed.	N
1. Low-pass filter with controllable cutoff and resonance. The cutoff is voltage-controlled within a range of 0 to 5 volts. Resonance is controlled simply by a variable resistor which adjusts the feedback into the filter.	a. Verify that the voltage controlled cutoff works: Connect the input of the filter to the VCO's square wave. View the output of the filter on an oscilloscope. Sweep the cutoff control voltage, and make sure the resulting square wave becomes smoother as the cutoff decreases. b. Now keep the voltage fixed and vary the resonance to the filter. Check to see that the resonance appears on the oscilloscope. On the oscilloscope, resonance appears as ripples after steep transitions.	N
1. When the trigger line goes high, the envelope starts increasing (attack phase), then decreases during the decay, then stays at a constant sustain level, and then drops to around 0V during the release phase after the trigger goes low.	a. Connect the output of the envelope generator to the oscilloscope and program the microcontroller to set the trigger signal every few seconds. Verify that oscilloscope output is as expected from the simulation, and see that turning the potentiometers affects attack, decay, sustain and release time.	N
1. The voltage controlled amplifier has two inputs, the audio signal coming from the filter as well as the amplitude envelope from the envelope generator. The envelope controls the gain of the amplifier.	a. Get the input signal from the VCO's square wave, and vary the control voltage. Check that as it increases the amplitude on the oscilloscope increases and as voltage decreases the amplitude decreases.	N

1. The low frequency oscillator will generate a triangle wave from about 1 Hz to 20 Hz. It's output will be in the range from -2V to 2V. It can be used to modulate other parameters of the synthesizer in varying amounts.	a. Verify that the waveform is a triangle wave with the oscilloscope and that its frequency changes as the potentiometer is changed. b. Connect the synthesizer output to a speaker. Verify that for each knob (volume, pitch, filter cutoff, square wave duty cycle), turning it increases the modulation of that particular sound.	Y

Table 2 MIDI System Requirements and Verifications Table

Requirement	Verification	Verification status (Y or N)
1. Microcontroller can be programmed from ISP headers	a. Program some test code to blink the LEDs, upload it and check that it works.	Y
1. Microcontroller can receive inputs from the external MIDI connector.	a. Probe the UART input pin on the ATMega. Verify that when a key on the keyboard is pressed, there is some digital data being sent to the microcontroller. This verifies that the MIDI to UART circuit works. b. Write code using the Arduino MIDI library to light and LED whenever a key is pressed, and turn it off when no key is pressed.	N
1. Microcontroller can output a specific voltage onto the DAC (MCP4921) over the SPI lines. This voltage	a. First, verify that communicating with the DAC works. Write a program that toggles the voltage between two arbitrary values (e.g. 1V and 3V) every second. Verify this is working by probing the DAC output with a multimeter. b. Combine this with the MIDI library so that output voltage corresponds with the key pressed. A4 should result in 2.5V and every key up or down from there increases or decreases voltage by 100 mV. Check with a	Y

	multimeter while holding down various keys.	
1. The microcontroller can read data from an SD Card.	a. Upload a test program using the SD library to blink LEDs. It should read a text file on the card, and every '1' character turns the LED on while '0' turns it off. Put some test code onto b. Update the code so that it reads key inputs from the file and outputs correct voltages to the DAC. Verify by listening to the output.	N

Table 3 Power and Output Requirements and Verifications Table

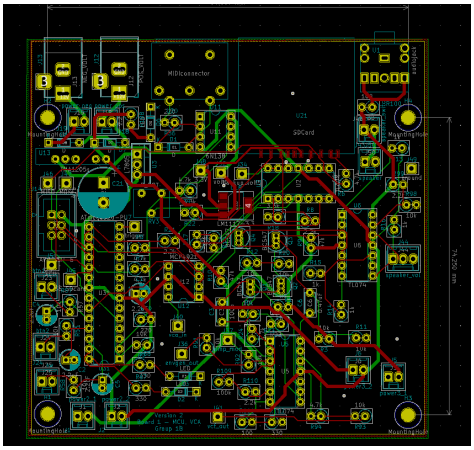
Requirement	Verification	Verification status (Y or N)
1. Power system provides +12, -12 and Ground to the circuit within 0.1V.	a. Check power levels with the multimeter. For noise, view the signal with an oscilloscope and measure peak-to-peak noise variations.	Y
1. Produce 5V from the 12V input for use by the microcontroller.	a. Verify with a multimeter. Measure between ground and the 5V output, and check that it is indeed 5V.	Y
1. Produce 3.3V from 5V for use by the SD card.	a. Verify with a multimeter like above but with 3.3V.	N
1. Output is audible with an external speaker.	a. Plug in an external speaker to the line out and verify it sounds like the audio signal.	Y
1. Output is audible with the built-in speaker.	a. Listen and make sure it sounds like the audio signal.	N

Appendix B Arduino Code (microcontroller without MIDI)

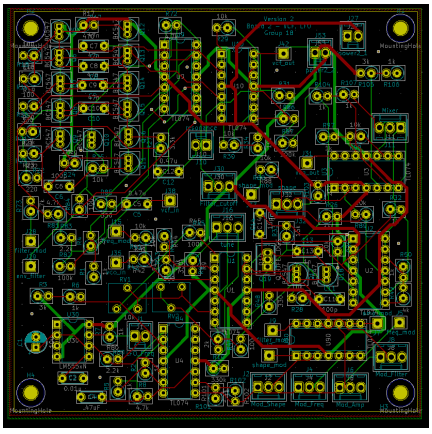
```
float vref = 5;

// see https://cyberblogspot.com/how-to-use-mcp4921-dac-with-arduino/
void setVoltage(uint16_t value)
{
  uint16_t data = 0x3000 | value;
  digitalWrite(SS_DAC, LOW);
  SPI.beginTransaction(SPISettings(16000000, MSBFIRST, SPI_MODE0));
  SPI.transfer((uint8_t)(data >> 8));
  SPI.transfer((uint8_t)(data & 0xFF));
  SPI.endTransaction();
  digitalWrite(SS_DAC, HIGH);
}

void loop() {

  float vout = 1;
  digitalWrite(LED1_PIN, HIGH);
  setVoltage( round(vout/vref * 4096) );

  delay(100);
  digitalWrite(LED1_PIN, LOW);
  vout = 3;
  setVoltage( round(vout/vref * 4096) );

  delay(100);


  // /dac output = vref*val/4096 --> val = round(out/vref * 4096)
}
```

Appendix C PCB Design

PCB Board 1



PCB Board 2



PCB Board 3