UPPSALA
UNIVERSITET

MAT-VET-F-23022
Degree project 15 credits
June 2023

Circuit design and hardware implementation of an analog synthesizer

Olle Murhed

Master's Programme in Engineering Physics

UPPSALA
UNIVERSITET

Circuit design and hardware implementation of an analog synthesizer

Olle Murhed

Abstract

Since the heyday of analogue synthesizers in the 70's, they have largely been replaced by digital hardware and software synthesizers. However, in recent years, there has been a revival in analogue designs, possibly due to its "warmer" sound. This projects aims to take part of this renewal by building a simple analogue synth design with the most basic modules (e.g. oscillators, filters, mixers, amplifier), accompanied by a step sequencer for programming melodies. This will be done by designing circuits and implementing them on breadboards.

The circuits were designed with inspiration from various online resources, along with theoretical analysis and simulation software for complex circuitry.

The result is a fully functional synthesizer with four sawtooth oscillators. The only modules missing from the initial design are battery support and a line out output for recording the output of the synthesizer. The pitch specification was met as the oscillator did not differ from the expected frequency by more than ±15 cents (hundredths of a semitone), for a range of five octaves.

Some possible improvements include better step sequencer user friendliness by installing a display to indicate the notes, more robustness by implementing the synth on a circuit board instead of breadboard.

Some improvements can be made for the synth. For example, a display for the step sequencer would facilitate melody programming. Moreover, implementing the synth on a circuit board instead of breadboards would greatly improve robustness and reduce the risk of sound disruptions.

Faculty of Science and Technology
Uppsala Universitet, Place of publication: Uppsala
Supervisor: Uwe Zimmermann
Subject reader: Teresa Zardán Gómez de la Torre
Examiner: Martin Sjödin

Contents

1 Introduction

When the first modern analogue synthesizers emerged in the 50's and 60's and later became popularized in the 70's, they had a significant cultural impact, paving the way for electronic music. Since then, the hardware digital synthesizers (hereafter "synth") of the 80's and software synths of subsequent decades has largely replaced the analogue synth due to reduced cost and increased sound design capabilities [1].

In recent years however, there has been a analogue synth revival. This could be a counter reaction to the perfected "sterile" sound of a digital synth; the transition to digital synths removed the inherent imperfections of purely analogue synths, including pitch drift, noise and distortion. While of course highly subjective, many agree that these flaws give character and warmth to the synths [2]. Another reason might be a general longing for more hardware, as an escape from an increasingly screen time intensive workflow for electronic music producers.

In other words, analogue synths has gained relevancy again. From a broad perspective, this project aims to contribute culturally as a part of the analog resurgence, and to create a tool for music performance. A more specified purpose and problem formulation is given below, following a brief introduction to synth basics.

1.1 Synth basics

Before formulating a specified purpose for this project, brief explanations of the fundamental modules of a synth are given. Many audio synthesis paradigms exist. In this project *subtractive synthesis*, which amounts to sculpting a harmonically rich waveform with filters, will be used. Below is a list of essential synth components along with short explanations.

1. An oscillator is a waveform generator. For synths the waveform should live in the audible frequency range, i.e. 20-20000Hz. Common types of waves are sine, triangle, square and sawtooth. The frequency of most audio-related oscillators is controlled with a voltage; the common abbreviation *VCO*, as in voltage controlled oscillator will henceforth be used.

2. Filters are an integral part of subtractive synthesizers. Particularly, low pass filters are commonly used as they naturally make high frequency-rich wave forms sound softer (i.e. less piercing high frequencies).

3. Mixers simply allow the users blend multiple VCO:s together (for synths with multiple VCO:s).

It should be mentioned that some important module types, present in almost all commercial synth designs, has been left out of the design, simply due to the scope of the project. These include envelope generators and low frequency oscillators, which are signal generators that control other parameters, such as amplitude or filter cutoff frequency.

For a modern touch, a step sequencer will be included in the synth design. This is a programmable tool for generating sequential melodies that loop, common in repetitive genres such as techno. In this project, the step sequencer will act as a substitute to a conventional claviature keyboard as seen on most synths.

1.2 Problem formulation

The purpose of the project is to design circuits for all modules as seen in figure 1 and implement them in hardware as a fully functional synth. For a successful project, all modules should be

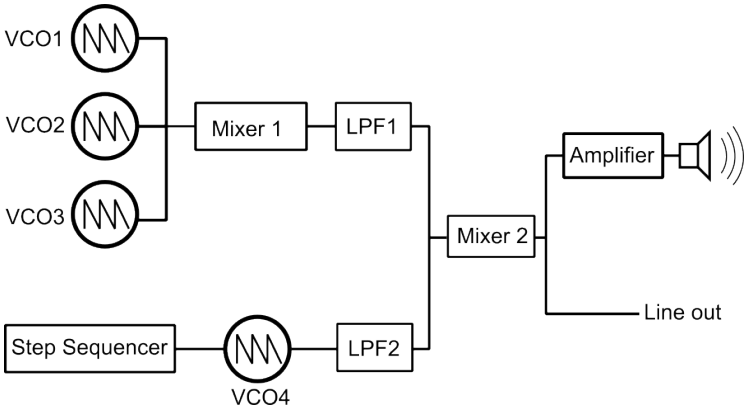


Figure 1: Block scheme for the synth design to be implemented in this project.

present and functional in the final build. In this design, VCO1-3 acts as drone oscillators (for example playing chords), while the pitch of VCO4 is programmable with the step sequencer, and will therefore act as a more flexible lead oscillator. VCO1-3 and VCO4 will have one low pass filter each, and the relative volume level between all VCO:s can be set with the two mixers. The synth design has two outputs: a built in speaker, preceded by an amplifier circuit, and a line out 3.5mm mono output.

1.2.1 Pitch specification

A pitch specification for the synth is needed to ensure musical usability - the synth has to be in tune to a certain degree. Before formulating a specification, a brief explanation of the mathematics behind basic western music theory is needed.

First and foremost, humans does not perceive pitch linearly: a 40Hz increase from 40Hz to 80Hz sounds like a bigger pitch jump than from 440Hz to 480Hz. Instead, a multiplication with a set factor always sounds like the same musical interval, independent of frequency range. With this in mind, the factor for any given interval can be found. An octave (doubling of frequency) is split up into twelve semitones. In the following way, the factor between two neighbouring semitones can be found, where f_2 is an octave above f_1:

$$f_2 = n_{semitone}^{12} f_1 = 2f_1 \tag{1}$$

$$n_{semitone} = 2^{\frac{1}{12}} = 1.05946... \tag{2}$$

Another less used interval is the *cent*, which is one hundredth of a semitone. Thus, its factor is:

$$n_{cent} = 2^{\frac{1}{1200}} = 1.0005778 \tag{3}$$

Humans can distinguish intervals down to 5-6 cents [3]. For this synth design, the maximum allowed detune is a slightly more generous 15 cents, referenced to an expected frequency. This, analogously to the previous equations, leads to the following allowed frequency range:

$$2^{\frac{-15}{1200}} f_{expected} < f_{measured} < 2^{\frac{15}{1200}} f_{expected} \tag{4}$$

Or, equivalently:

$$|\text{detune}| = \left| 1200 * \log_2 \left(\frac{f_{meas}}{f_{exp}} \right) \right| < 15 \text{ cents} \tag{5}$$

This should be fulfilled for all semitones in the playable range. A test procedure is explained in the method section. Note that this specification only applies to VCO4.

1.2.2 Step sequencer specification

The step sequencer should be programmable within a range of five octaves (including the limits), with semitone precision. Its output is control voltage (CV), which controls the pitch of VCO4. The CV should follow the common 1V/oct standard, which means that an increase of one volt in the CV will result in a doubled frequency, i.e. one octave jump upwards. To be consistent with the pitch specification, a maximum of 0.0125 volt deviation will be allowed for the CV (this corresponds to 15 cents). The control voltage should range from 0-5 volts to cover the specified range of 5 full octaves.

1.3 Delimitations

The final build will of this project will be a prototype built on breadboards. The main focus is on audio synthesis, not aesthetic design and user friendliness, although the latter will not be completely neglected.

2 Theory and circuit design

In this section, the theory of each module will be presented, along with circuit designs. As also explained further in the method section, the circuit have been designed with the help of existing literature, theory and simulation software LTspice.

Observe that only the circuit *structures* are presented in the theory section; the actual component values chosen for the circuits are presented in the material section (3.2).

2.1 Oscillators

As mentioned before, this synth will use a sawtooth wave for its four oscillators. A simple sawtooth wave can be built from a repetitive charging and discharging of a capacitor. The charging phase is almost instantaneous and is realised with an inverting Schmitt trigger. This oscillator principle belongs to the *relaxation oscillator* realm [4]. The capacitor is then discharged via the variable current source. How quickly the capacitor is discharged depends on the capacitance and current strength as will be explained soon. But first, the Schmitt trigger is described.

2.1.1 Inverting Schmitt trigger

A schmitt trigger is a type of comparator circuit with two thresholds. When the input voltage sinks below the lower threshold, the output is activated (meaning that the voltage is set to the inputted V_{cc}). When the input voltage rises above the higher threshold, the output is instead deactivated. This double threshold gap is called hysteresis, and is useful for feedback systems with noisy signals.

Figure 2 shows the Schmitt trigger in the context of a sawtooth generator. The Schmitt trigger output is activated whenever the voltage over the capacitor (which is at the same node as

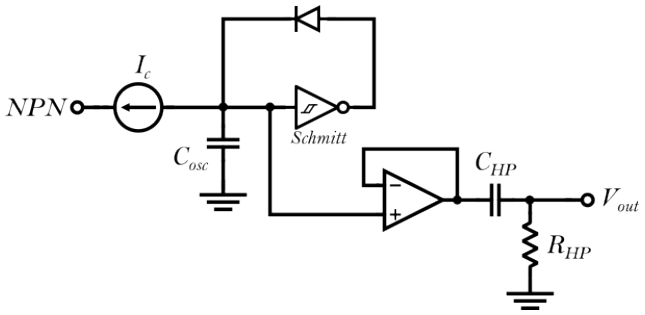


Figure 2: Basic relaxation oscillator as designed by Moritz Klein [5]. The capacitor C_{osc} is repeatedly instantaneously charged by the Schmitt trigger, then slowly discharged by the current source. The buffer and high pass filter to the left removes any DC offset to increase headroom for op-amps further down the signal chain.

the Schmitt trigger input) falls under the lower threshold. This results in an almost instantaneous voltage rise at the input node, and the Schmitt output is again deactivated. In this way, the Schmitt trigger effectively deactivates itself directly, and therefore acts like a pulse generator.

2.1.2 Oscillator frequency

As aforementioned, the capacitor discharging speed (which is directly related to oscillator frequency) is dependent on the current source strength I_c, capacitance C_{osc}, but also on the Schmitt trigger hysteresis, i.e. the voltage difference V_H between the lower and upper thresholds (since the Schmitt trigger inherently prevents any other voltages). Note that before any attenuation or amplification further down in the signal chain, the hysteresis V_H directly decides the amplitude of the waveform. What the current source actually is is described in the exponential converter section.

To find a theoretical expression for the oscillator frequency f, one can start with the basic expression of a capacitor charge, and take its derivative with respect to time:

$$Q = CV \tag{6}$$

$$I_c = \frac{\mathrm{d}Q}{\mathrm{d}t} = C_{osc}\frac{\mathrm{d}V}{\mathrm{d}t} = C_{osc}\frac{V_H}{T} = C_{osc} f V_H \tag{7}$$

$$f = \frac{I_c}{C_{osc} V_H} \tag{8}$$

T is the period. Equation (8) shows that I_c, C_{osc} and V_H should be found such that the frequency can be set between the audible frequency range of 20 to 20 000 Hz. The Schmitt trigger hysteresis V_H is dependent on the component's supplied voltage. The current source I_c is set via the exponential converter, but its useful range is fairly limited. The capacitor C_{osc} is then chosen to scale the discharge speed to the audible region.

2.1.3 Offset removal

At the end of the oscillator signal chain, as seen in the lower right section of figure 2, an op-amp buffer and a simple high pass filter is implemented. This is needed since the Schmitt trigger hysteresis is centered around half of the supplied voltage, i.e. $V_{cc}/2$. The chosen C_{HP} and R_{HP}

should result in a low cut-off frequency f_c that removes off-set (zero-frequency signals) but does not affect the audible range. In other words, a cut-off frequency close to 0Hz is preferable.

$$f_c = \frac{1}{2\pi R_{HP} C_{HP}} \tag{9}$$

Now, the output voltage V_{out} of the complete oscillator should be centered around zero. This creates more headroom for op-amps further down the synth signal chain.

2.2 CV

According to the synthesizer specification, the control voltage from the step sequencer should range from 0 to 5 volts. This is accomplished with from the Arduino via a digital-analog converter (DAC), further explained when discussing the step sequencer circuit in section 2.9.

2.3 Conversion from linear CV to exponential current

Until now, the current source in the oscillator has remained unexplained. As mentioned, the main input to the oscillator, to steer its frequency, is the linear CV from the step sequencer. Moreover, the 1V/oct standard should be used, meaning that a CV increase of 1V should result in a pitch octave, i.e. a doubling of frequency. From the oscillator description, in particular equation 8, the oscillator frequency is proportional to the current source I_c. In summary, a circuit that converts a linear voltage to a exponential current is needed. The resulting circuit is heavily inspired by Klein [5], and is shown in figure 3. In the following sections, its functionality is described from right to left, in reverse signal chain order. Note that only VCO4 need to convert CV coming from a step sequencer. For VCO1-3, the voltage (and pitch) will instead be set using potentiometers. Figure 4 shows the corresponding circuit for VCO1-3.

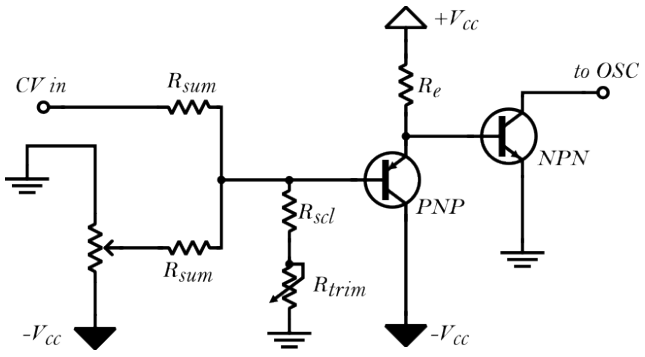


Figure 3: Circuit for converting control voltage to an exponential current source for VCO4.

2.3.1 NPN transistor: exponential converter

The actual exponential conversion happens in the last step of the circuit, at the NPN transistor. It is often taught that the base-emitter voltage is a constant drop of 0.7V, due to the properties of silicon. This holds for many applications, but in reality, the base-emitter voltage is a logarithmic function of the current through the transistor; each time the collector current is doubled, the base-emitter voltage increases by 18 mV (whether we look at collector, emitter or base current

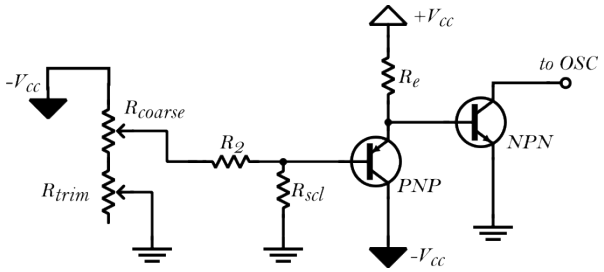


Figure 4: Circuit for converting control voltage to an exponential current source for VCO1-3. Note that there is no CV input from any step sequencer here; the voltage that decides the oscillator pitch is set by the R_{coarse} and R_{trim} potentiometers.

here does not matter since the base- and emitter current are roughly proportional, and $I_c = I_e - I_b$) [6]. By looking at this function reversely, the collector current can roughly be seen as exponential function of the base-emitter voltage V_{BE}:

$$I_c = e^{aV_{BE}+b} \tag{10}$$

Where a and b are some variables. Note that the domain of this function is quite limited - a high base-emitter voltage would lead to unreasonably large currents.

2.3.2 PNP transistor: emitter follower

In theory, the NPN looks like a ready-to-use exponential converter. However, there is a problem: The current through a transistor is temperature dependent - the variables a and b depends on temperature. In terms of the synthesizer output, this means that the frequency will shift whenever the synth is heated or cooled. To be specific, the current through the NPN will increase with increased temperatures, leading to a raised pitch. This could endanger the specification goal of staying in tune within ±15 cents.

Conveniently, a PNP transistor has the opposite response to temperature shifts. In figure 3, it is inserted before the NPN, with the ambition of cancelling any temperature-induced collector current shifts in the NPN transistor [6].

The PNP is inserted in the form of a so called emitter follower, which basically copies the base voltage to the emitter, along with a voltage off-set. In this way, it does not affect the circuit much, other than balancing out the temperature dependence.

2.3.3 Voltage formatting

It is important that the base voltage at the PNP transistor is within a range that ultimately leads to audible frequencies. This is handled with voltage summing and dividing with resistors, potentiometers and trim resistors as seen in the left side of figure 3. Recall that the CV is specified to 0-5V.

2.4 Filter

A simple low pass filter can be constructed by connecting a resistor and a capacitor in series to ground. An expression for the output voltage (over the capacitor) can be derived by looking at the circuit as a voltage divider. Finding the cut-off frequency f_c(defined by yielding an amplitude decrease of 3dB) is then easy:

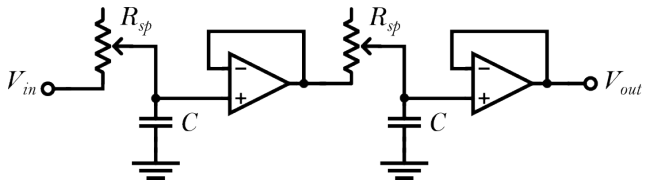


Figure 5: Filter 1 and filter 2 circuit. The filter design is a second order variable low pass filter, using simple buffers that prevent the two filters from interfering with one another. The R_{sp} index stands for stereo potentiometer - both resistance values are simultaneously set by the same knob.

$$f_c = \frac{1}{2\pi R_{sp} C} \tag{11}$$

The simple filter circuit is expanded in figure 5. Here, two first order filters are put in series, effectively creating a second order filter. This will affect the sound more drastically with a filter slope of 12dB/octave instead of 6dB/octave for the first order case. The potentiometer allows for variable cut-off frequency. Note that both resistance values will be set simultaneously by a stereo potentiometer (hence the *sp* subscript). The buffers hinder the two subfilters from affecting each other.

For audio purposes, the filter cut-off should be variable within 20 to 20000 Hz. A 100nF capacitor and a 100kΩ potentiometer achieves this, with f_c approaching infinity as R_{sp} tends to zero, and $f_c = 15.9Hz$ with the potentiometer set to max. By setting the potentiometer to 0 Ohms, the cut-off frequency approaches infinity.

2.5 Mixers

The mixer in this synth design are inverting adder circuits, preceded by a potentiometer for each input signals for setting levels.

As seen in the synth block diagram (figure 1), there are two mixers in this design. Mixer 1 mixes signals from VCO1-3, while mixer 2 mixes the combined filtered signals from VCO1-3 and the filtered VCO4 signal. Mixer 1 is shown in figure 6, and mixer 2 is identical but with only two inputs, and is therefore left out.

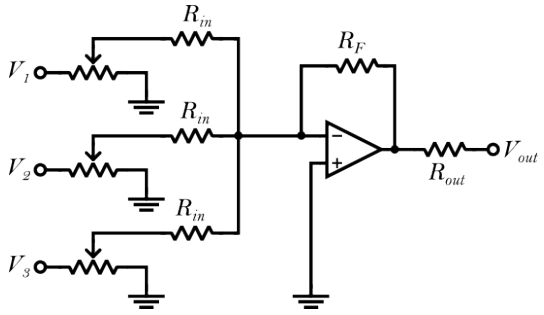


Figure 6: Circuit for mixer 1. The inputs V_1, V_2 and V_3 come from VCO1-3. Mixer 2 is identical, with the exception of only having two inputs, coming from filter 1 and filter 2.

2.6 Amplifier circuit and speaker

The speaker is one of the synth outputs (the other being line out). To drive the speaker the signal coming from mixer 2 must be amplified; the LM386 is a common amplifier in audio circuits and is therefore used here. With its data sheet, a proposed speaker circuit is given, which is redrawn in figure 7. [7]:

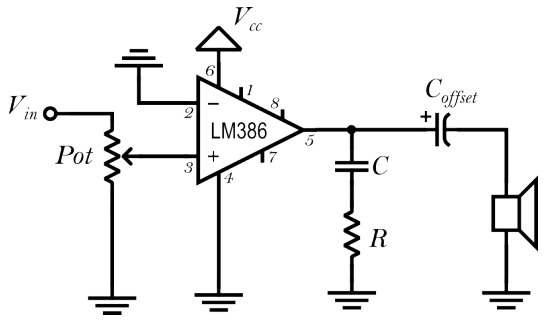


Figure 7: Amplifier circuit with speaker. The design is taken from the LM386 data sheet, redrawn here.

2.7 Line out

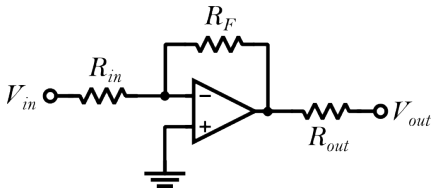


Figure 8: Simple line out circuit. R_F and R_{in} should be chosen such that the V_{out} peak-to-peak is around 3 volts. To further follow the line out standard, R_{out} should be chosen in the range of 100-500Ω.

Line out is a standardized output for audio. Typical levels are 3 volts peak-to-peak and low output impedance, 100-500Ω. This can simply be achieved by using an inverting amplifier, followed by an output resistor within the range given above, as seen in figure 8. The amplification gain is given by:

$$\text{Gain} = -\frac{R_F}{R_{in}} \tag{12}$$

As also mentioned later in the method section, the resistance values R_F and R_{in} can be decided after measuring the input voltage, such that the output voltage peak-to-peak value resides around 3 volts. The output signal V_{out} can then be connected to an 3.5mm aux jack port.

2.8 Power supply

During the construction and testing phase, the circuits will be powered by a power supply. For better portability however, the synth should also be able to receive power via battery (9V). Since the design needs both positive and negative V_{cc} (for example, all op-amps are powered with $\pm V_{cc}$) along with ground, a voltage inverter will be used. A voltage converter IC (such as the TL7660) can, along with two additional capacitors as described in its data sheet [8], invert a DC voltage.

Furthermore, some of the components (the Arduino, DAC and Schmitt triggers) need a power supply of 5 volts. For this, a voltage regulator, which can downscale an input voltage, will be used.

2.9 Step sequencer hardware

The Arduino NANO will comprise the basis for the sequencer. Figure 9 shows a minimised step sequencer circuit (multiple buttons and LEDs will be used), with signal flow generally going from left to right. With simple button and potentiometer circuits (to the left in figure 9), the Arduino will register digital and analog values respectively. The button will make use of the Arduino's internal pull-up resistor, as set in the code (see Appendix A), hence the subscript for D_{pu}.

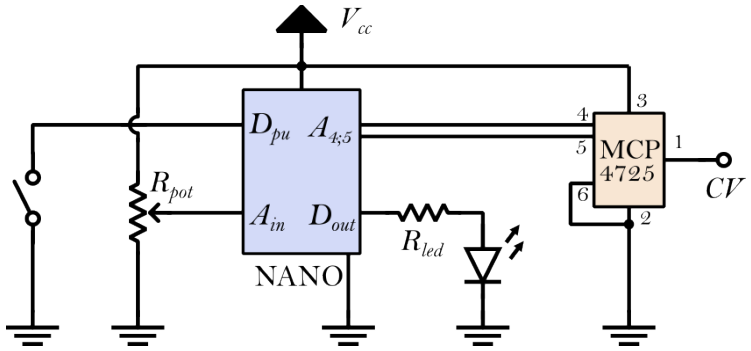


Figure 9: Minimised circuit for step sequencer. For the actual implementation, the button and LED circuits will be multiplied. Digital inputs (with internal pull-up resistor activated) and analog inputs will be used for the buttons and potentiometer, respectively. Analog and digital outputs will be used for DAC communication and LEDs, respectively.

As for the CV output, a digital-analog converter (DAC) will be used to set the voltages. A 12-bit DAC has a resolution of 4096, which should be sufficient to satisfy the synthesizer specification of ±15 cents. For this project, the MCP4725 will be used. The pins are connected according to its data sheet [9]. Pins 4 and 5, which handle communication, are connected to the Arduino's A4 and A5 pins respectively. This last connection choice is to simplify programming, as explained in section 3.3.2.

2.10 Step sequencer software

Some simple conversions, for example the potentiometer output to tempo, are needed in the code. These are conversions are documented with comments in the code, see appendix A.

3 Method

3.1 Theoretical preparation

Most overall circuit structures for all synth modules were decided through existing circuit designs from literature and the online "do it yourself" synth community.

The preliminary component values were chosen mainly in two ways: through theory according to the equations presented in the theory sections and, for complex circuits, simulations in the LTspice software.

The circuit diagrams were drawn with the Inkscape software.

3.2 Instruments and materials

Table 1: Material for VCO1-4

Component	Value	Qty	Name
Diode	0.7V	4	
C_{osc}	1nF	4	
Schmitt trigger	1.03V (hyst.)	2	SN74HC14N / CD40106BCN
op-amp		4	LM324N / OP77
R_{HP}	100kΩ	4	
C_{HP}	$1\mu F$	4	

When all modules had preliminary design with set component values, a complete component list was compiled. All electrical components and their values can be found in tables 1-6b. All tables are complete with component type (same name as in circuit figures in the theory section), value, quantity and IC name where applicable.

Table 2: Material for exponential converters.

(a) Material for exponential converter VCO4.

Component	Value	Qty	Name
NPN		1	BC547
PNP		1	2N3906
R_e	1MΩ	1	
Pot	100kΩ	1	
R_{avg}	100kΩ	2	
R_{scale}	1kΩ	1	
R_{trim}	1kΩ	1	

(b) Material for exponential converter for VCO1-3.

Component	Value	Qty	Name
NPN		3	BC547
PNP		3	C32716
R_e	1MΩ	3	
R_{coarse}	100kΩ	3	
R_{trim}	10kΩ	3	
R_{scl}	3.9kΩ	3	
R_2	100kΩ	3	

Table 1 shows the material for all oscillators VCO1-4. The collector current from the NPN transistor in the exponential converter was found (through LTspice simulations) to exhibit the wanted exponential behaviour in the range of 66nA to 1.92 μA. The remaining values were decided through equations 8 and 9, with the exception of the diode and Schmitt trigger hysteresis, which are inherent values to the component type (the latter being found in the IC data sheet).

Table 3: Material for the two variable LP filters.

Component	Value	Qty	Name
op-amp		4	LM324N
Stereo pot	100kΩ	2	
C_{LP}	100nF	4	

Table 2a and 2b show the chosen values for the voltage formatting and exponential converters, found mostly via LTspice simulations.

The material list for the two second order variable low pass filters are shown in table 3. The capacitor C_{LP} was chosen such that the lowest possible cut-off frequency would be sufficiently low to effectively filter out the waveform completely, according to equation 11.

Tables 4a and 4b contain the mixer components. Note again that they are very similar in design; mixer 2 is a copy of mixer 1 (as shown in figure 6) but with one less input.

Table 4: Material list for mixer 1 and mixer 2.

(a) Material for mixer 1.

Component	Value	Qty	Name
op-amp		1	OP200
R	100kΩ	4	
Pot	100kΩ	3	
R_{out}	560Ω	1	

(b) Material for mixer 2.

Component	Value	Qty	Name
op-amp		1	LM324N
R	100kΩ	3	
Pot	100kΩ	2	
R_{out}	560Ω	1	

Table 5a shows the component values of the line out circuit. The resistors values of the inverting amplifier in the circuit, seen in figure 8, was chosen after first measuring the maximum input peak to peak voltage on the oscilloscope, in order to reach the desired output voltage amplitude according to the line out standard.

Table 5: Material list for the line out circuit (a) and the speaker amplifier circuit (b).

(a) Material for line out circuit.

Component	Value	Qty	Name
op-amp		1	LM324N
R_{in}	68kΩ	1	
R_F	10kΩ	1	
R_{out}	560Ω	1	
3.5mm out		1	??

(b) Material for amplifier circuit.

Component	Value	Qty	Name
Amp IC		1	LM386
Pot	10kΩ	1	
R	10Ω	1	
C_{offset}	250μF	1	
C	50nF	1	

The speaker amplifier circuit was, as mentioned, largely inspired by the LM386 data sheet, which included a proposed amplifier circuit [7]. It also provided component values, which were directly copied to this project and presented in table 5b.

Table 6: Material lists for the step sequencer and power supply.

(a) Material for the step sequencer hardware.

Component	Value	Qty	Name
Microcontroller		1	Arduino NANO
DAC	12-bit	1	MCP4725
Button		6	
R_{led}	560Ω	4	
Led		4	

(b) Material for portable power supply.

Component	Value	Qty	Name
Battery	9V	1	
Battery connector		1	
Voltage converter	12-bit	1	ICL7660
C	10μF	2	

Table 6a shows the components needed for the step sequencer.

Table 6b shows the components values for the power supply. As explained in the theory section, the circuit needed for converting the voltage was provided by the ICL7660 data sheet. Capacitance values were also recommended.

When the list of components was completed, an inventory check was made. Missing components were ordered. The ordering throughout the execution phase was done iteratively after some design revisions were made, which caused some construction delay.

3.3 Execution

3.3.1 Hardware construction

Before construction began, some essential and beneficial tools were acquired. These are presented in table 7.

Table 7: Material and equipment for construction and testing.

Equipment	Model/type
Breadboard	Pro'sKit BX-4112N (Qty: 5)
Jumper cables	
DC power supply	Teknikum ±12V & 5V supply
Oscilloscope	Tektronix TBS 1152B-EDU
Multimeter	AMPROBE AM-510-EUR
Connecting wire	
Wire stripping tool	
Component tester	

In accordance with the delimitations stated in the introduction, the synthesizer was built as a prototype on breadboards. Generally, each individual synth module was first built provisionally with jumper cables, to ensure that circuit design was functional (using the oscilloscope) in practice and to test breadboard layouts. The general method was as follows:

1. Fetch all needed components for the circuit according to the relevant material list. Use the "component checker" to ensure that the component values are good.
2. Construct the circuit according to the circuit design figures, using jumper cables.
3. Test whether the circuit is functional using the oscilloscope, i.e. check if the output is expected.
4. Investigate any unexpected flaws:
   (a) Double check circuit connections
   (b) Check whether the involved components are functional
   (c) Make design revisions
5. Iterate until the circuit is functional
6. Replace the jumper cables with manually cut connection wires using the wire stripping tool.

This method allowed for quick revisions, while not spending time and effort cutting wires for a possibly faulty circuit.

The above method explains the tactics for a single synth module. As for the whole synth, the construction order was chosen such that any needed re-designs could be detected early in the project phase (allowing for increased time margin for new component orders). This was done by first building the lower signal chain in figure 1 (starting with VCO4), ensuring that all unique circuit types were implemented and tried before duplicating them. Then, when assured that all circuits were functional (or after needed revisions), the circuits could simply be duplicated to the other signal chain, without the provisional jumper cable step. For example VCO4, when complete, could basically be copied to VCO1-3. In a similar fashion, mixer 2 and filter 2 could be copied to implement mixer 1 and filter 1 without much thought.

3.3.2 Software (coding the step sequencer)

The Arduino IDE was used for programming the Arduino NANO. For simplified communication between the DAC and the Arduino, the Adafruit_MCP4725 library was used. This library assumes that pins A4 and A5 of the Arduino are connected to pins 4 and 5 of the DAC, respectively, which explains the wiring in figure 9. The full code can be seen in appendix A.

The buttons needed some debouncing software to hinder functional glitches when pressing them. This was done by inserting a time buffer of 50 milliseconds from the last button state change before acting on a button press, a technique provided by the Arduino documentation [10].

3.4 Pitch and CV test

To test whether the pitch and CV specifications were fulfilled, the VCO4 frequency was first scaled using two different input voltages (1st and 3rd octaves). More precisely, the following algorithm was used:

1. Make sure all semitones are set to 0.
2. Voltage offset: Set the step sequencer to play the first octave. Use R_{offset} (leftmost potentiometer in figure 3) to tune VCO4 to play a frequency f_{ref} of roughly 200Hz (this will theoretically set the step sequencer range of 100-3200Hz).
3. Voltage scaling: Set the step sequencer to play the third octave. Use R_{trim} to tune VCO4 to play two octaves higher, i.e. 800Hz. Steps 2 and 3 might need several iterations as both R_{offset} and R_{trim} affect pitch.
4. Pitch test: Set the step sequencer to oct = 0. Let n = -1. Repeat until n = 4:
   (a) Calculate the expected frequency $f_{exp} = 2^n f_{ref}$
   (b) Measure the frequency of VCO4
   (c) Calculate the detune according to equation 5, and check whether it is within the allowed range of ±15 cent.
   (d) Also measure the step sequencer CV output voltage (for controlling the step sequencer specification).
   (e) Let n = n + 1

The semitones in between the octaves can be assumed to be well tuned if the octaves are.

4 Results and discussion

Firstly in the result and discussion section, the hardware implementation is presented, and it is discussed whether the initial design idea is fulfilled. The outputted waveforms are then presented, after which the results from the pitch and CV tests are shown.

4.1 Hardware implementation and layout

The hardware implementation of the synthesizer is shown in figure 10. All modules from the general design specification scheme in figure 1 are present in the hardware expect for the line out output and battery power support. All existing modules are fully functional (but have room for improvement as will be discussed below). Generally, the signal chains flows from the sides to the middle horizontally, with the step sequencer and VCO4 chain from the left and VCO1-3 from the right. Further details are described in the caption of figure 10.

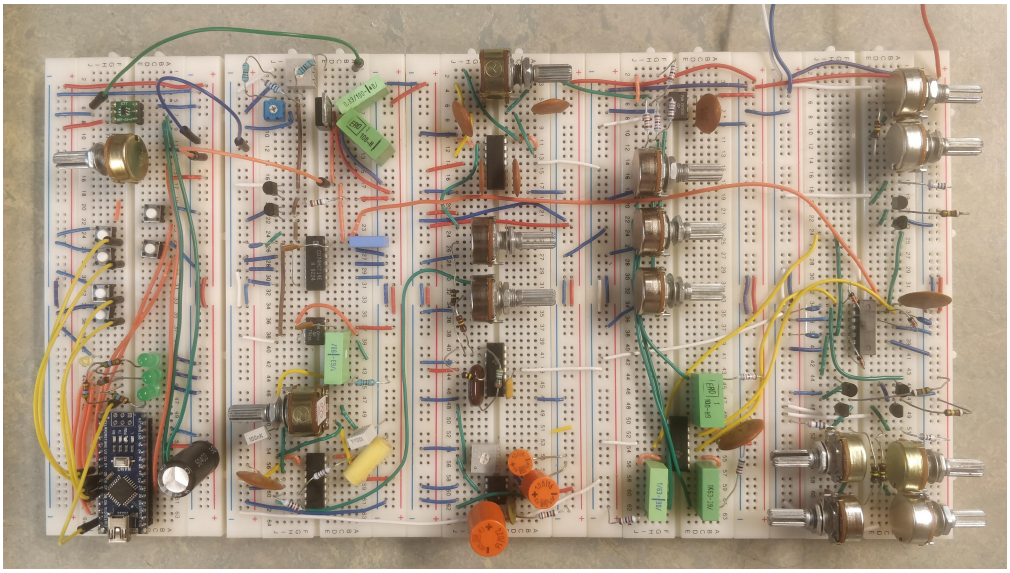


Figure 10: Final hardware implementation of the synthesizer. One can discern five breadboards; the leftmost comprise the step sequencer. The second one hosts the CV exponential conversion, VCO4, and filter 2. The third (middle) hosts filter 1, mixer 2, line out (not complete yet) and speaker amplifier circuit. The fourth breadboard hosts mixer 1 and the high-pass filters of VCO1-3, while the right-most breadboard comprise the rest of VCO1-3 with their corresponding exponential converters.

4.2 Sawtooth waveform qualities

Overall, the sawtooth waveforms generated from the oscillator circuits looks as expected. Figure 11 and 12 shows the waveform generated by VCO1 at the speaker output (with filter 1 completely open) for a low (279.5Hz) and high (1.640kHz) frequency respectively. Since VCO2 and VCO3 use the same Schmitt trigger IC, their result is here assumed to be similar enough to be left out here.

For the high frequency, one deviation from a pure sawtooth can be seen: there are ripples occurring after each quick rise. For the lower frequency in figure 11, the ripples are also present. Furthermore, one can see that the downwards ramp of the sawtooth is slightly convex for lower frequencies. This curved ramp is not present earlier in the signal chain (as can be seen from the filter screenshots in figure 13), and could therefore possibly be explained by a non-linearity at the LM386 amplifier IC.

The ripples are not present earlier in the signal chain either, as is also evident from figure 13b. The filter input signal, which is the same as the VCO4 output and is indicated by the yellow line, does not have ripples. Figure 13a, where the filter is completely open (highest possible cut-off

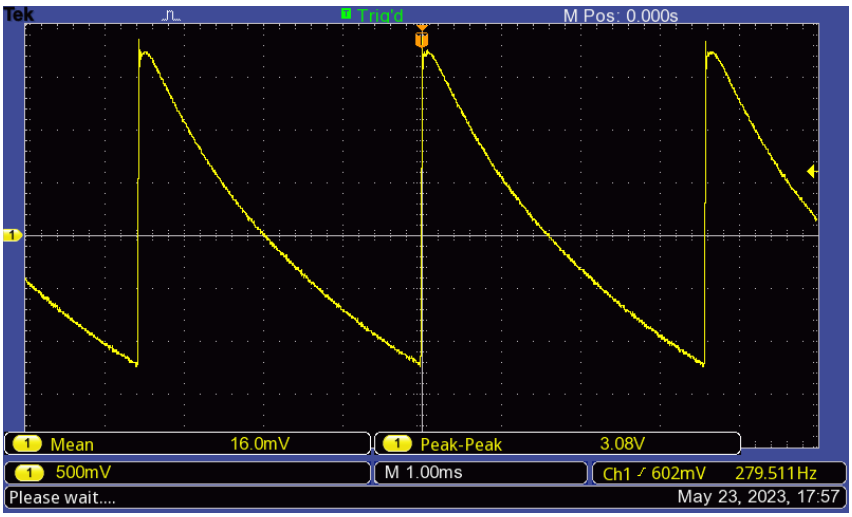


Figure 11: VCO1 279.5Hz waveform at speaker output. Compared to a pure sawtooth, this one has some ripples after each high peak and slightly convex downward ramp.

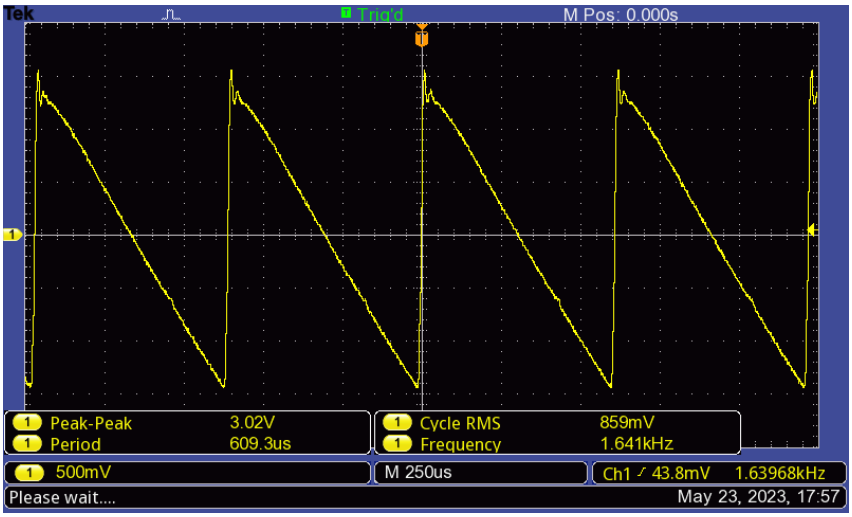
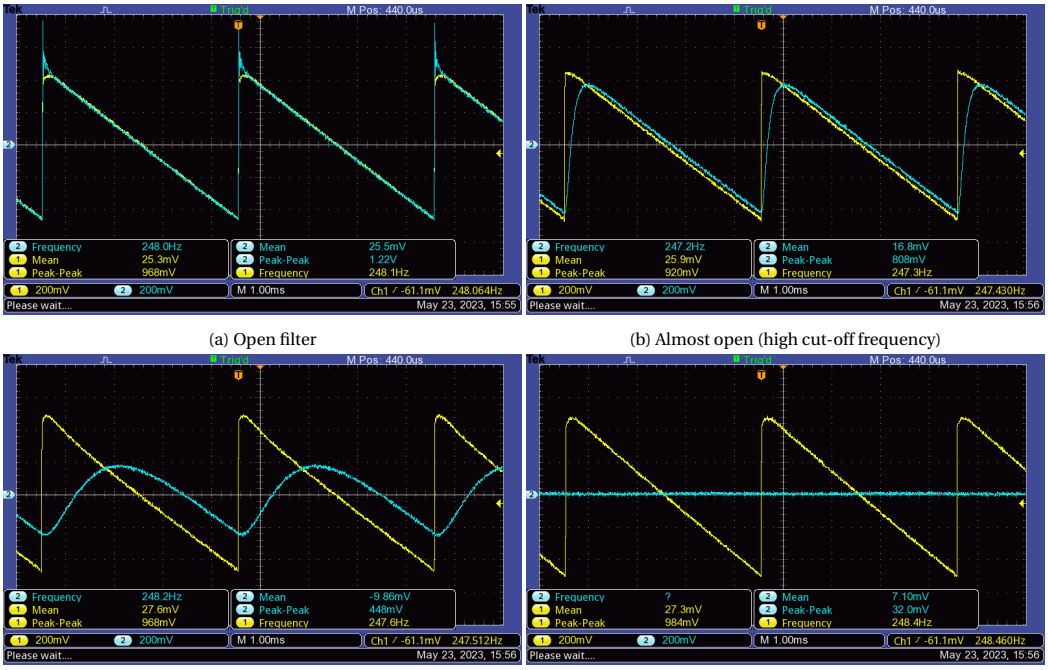


Figure 12: VCO1 1.640kHz waveform at speaker output. It looks like a pure sawtooth, except for the ripples at the peaks.

frequency), indicates that the ripples are created at least partly in the filter circuit (possibly by the multiple op-amp buffers).

4.3 Filter behaviour

The filter behaves mostly as expected. Figure 13 shows the successive closure of filter 2 from completely open to closed, e.g. with cut-off frequencies ranging from infinitely high (as the low-pass filter potentiometers approaches zero Ohms) to 16.0 Hz.



(a) Open filter

(b) Almost open (high cut-off frequency)

(c) Almost closed (low cut-off frequency)

(d) Closed filter (lowest possible cut-off frequency; approaching 0Hz according to equation 11

Figure 13: Successive closure of filter 2, with VCO4 as source. The yellow line shows the input, and the blue line shows the filtered output.

When sweeping through the filter, the sound changes relatively smoothly. However, the "sweet spot", i.e. where the sound is perceived to change the most, of the filter is narrow. This is because of the linearity of the potentiometer; the resistance increases linearly with the knob position. The following reasoning highlights why this is a problem: With the filter completely closed (e.g. $R_{pot} = 100k\Omega$), the cut-off frequency is 16.0 Hz from equation 11. When opening the filter halfway, (e.g. $R_{pot} = 50k\Omega$), the cut-off frequency has only doubled to 31.9 Hz. For sawtooth waveform with fundamentals in the normal range of 300Hz, the output is practically silent for this wide range of knob positions.

The problem is still present for high cut-off frequencies. With the knob position at 90 , the cut-off frequency is 159Hz. At 97.5 ($R_{pot} = 2.5k\Omega$), the cut-off frequency is 638Hz. There, the knob position is not far from completely open, which results in theoretically infinitely high cut-off frequency.

An exponential behaviour, where for instance a knob increase of 10 percentage points would

consistently result in a doubled cut-off frequency, would broaden the sweet spot. This is also common in commercial synths, and makes up a large potential improvement on this synth design. Replacing the linear stereo potentiometer in the design with a exponential one would achieve this.

4.4 VCO4 pitch specification result

Here, the result for the pitch test, as described in the method section, is presented. Table 8a shows the expected frequency for a range of five octaves, calculated from a reference, along with measured frequency and detune. The detune is calculated using equation 5. The octave refers to the octave set by the step sequencer. For this test, octave 1 was chosen as a reference at 211.67Hz. As can be seen from the detune row, all other octaves are inside the allowed range of ±15 cents, so the pitch specification is fulfilled.

Octaves 1 trough 4 are precisely tuned, well below the 5-6 cent threshold perceptible by humans. The detune of the lowest and highest octaves are above that range however.

Table 8: Results from pitch test (a) and CV test (b).

(a) Result for the pitch test. The first octave (oct=1) from the step sequencer acts as reference frequency at 211.67Hz (bold), from which the other f_{exp} are derived. From the measured frequencies f_{meas} for each octave, the detune is calculated from equation 5. All octaves lie within the allowed range of ±15 cents, so the pitch specification is met.

Oct	f_{exp} [Hz]	f_{meas} [Hz]	Detune [cent]
0	105.8	107.2	+11.1
1	N/A	**211.67**	N/A
2	423.34	423.61	+0.554
3	846.68	846.8	+0.123
4	1693.36	1699	+2.88
5	3386.72	3415	+7.22

(b) Result for CV specification test. V_{exp} shows the expected voltage output from the step sequencer output, which coincides with the octave according to the 1V/oct standard. V_{meas} shows the measured voltage for each octave. Compared to specified maximum deviation of 0.0125V, the outputted voltage is slightly too high for octave 0, and too low for octaves 3 through 5.

Oct	V_{exp}	V_{meas}	Deviation
0	0 V	0.0161 V	0.0161 V
1	1 V	1.00 V	0.00 V
2	2 V	2.01 V	0.01 V
3	3 V	2.98 V	-0.02 V
4	4 V	3.95 V	-0.05 V
5	5 V	4.91 V	-0.09 V

4.5 Step sequencer results

In this section, the hardware implementation of the step sequencer and its specification results are presented. Figure 14 shows the step sequencer hardware (here temporarily separated from the rest of the synthesizer). One of the step sequencer specifications was for it to be programmable within a range of 5 octaves. This is realised with the four bottom-most buttons; from left to right, their functionality is semitone down/up and octave down/up. Both button pairs have a modulo behaviour, meaning for example that the octave jumps back to 0 if incremented from 5. The upper right button increments the current step of the sequence to be set.

The buttons mentioned above are enough to cover the specified functions. In addition however, some further functionality was implemented. The three green LEDs indicate in binary which step is currently programmed (i.e. which note of the melody is being changed). The sequence is eight notes long; for longer melodies, more LEDs need to be installed for correct step indication.

The upper left button toggles between two modes: *set* and *play*. During *play* mode, the sequence loops continuously, and the melody can be programmed while listening to the whole

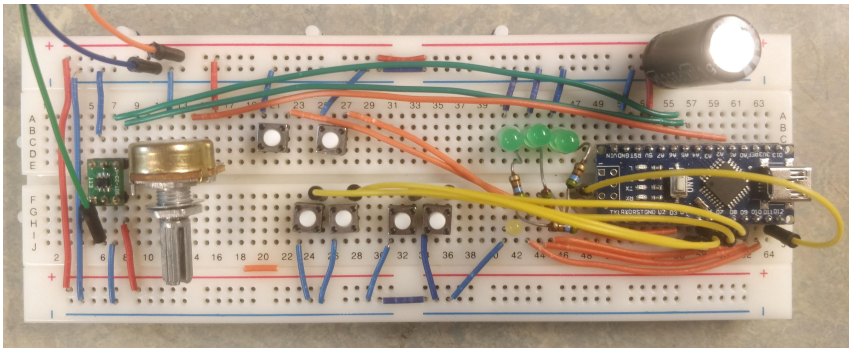


Figure 14: Caption

melody loop. During *set* mode, the current step is outputted instead of the loop, which facilitate finding the wanted note. There is a small yellow LED between the right-most button and the Arduino; during set mode, this LED is on constantly. During play mode, it flashes during the first note of the sequence, effectively also indicating the tempo.

Lastly, the potentiometer to the left in figure 14 adjusts the tempo.

Now, the CV specification is presented and discussed. The outputted control voltages V_{meas} from the DAC are presented and compared to the expected voltages in table 8b. The measured voltages for octaves 0, 3, 4 and 5 deviate more than the specified maximum of 0.0125V, and does therefore not meet the requirements.

In this synth, where the pitch scaling is manually tunable, this is not a problem since a trim resistor adjustment can account for the low voltages. However, if the step sequencer is used with another synthesizer with an 1V/oct input, the frequency of the oscillator will be too high for octaves 0 and 2, and too low for octaves 3-5.

4.6 General discussion and possible improvements

As mentioned earlier, all modules except the line out output and battery support are implemented in hardware. All modules implemented so far are functional, but there are however room for improvements; already mentioned is the narrow filter sweet spot. Further possible improvements are discussed below.

A display screen would be a significant upgrade to the step sequencer as it would greatly enhance user friendliness. With the current implementation, the user is rather blind when choosing semitone and octaves for each step - a screen indicating which note is selected would facilitate melody programming immensely.

A second improvement would be to implement the whole synth on a circuit board (although a breadboard prototype implementation was the intended goal for this project). The breadboard construction is rather fragile, which is not optimal for such a hands-on tool as a synth. For instance, some of the potentiometers tend to loosen from the breadboard when twisted, causing sound disruptions and sudden pitch changes.

5 Conclusions

In general, the results of this project has been satisfactory. Electrical circuits has been designed for all modules present in the initial design idea (figure 1), and most of them (excluding line out and battery support) have been implemented in hardware with functional results.

The pitch specification has been met - the measured frequency does not differ from the expected frequency more than ± 15 cents for a range of 5 octaves including the high octave limit. Moreover, for octaves 1 through 4, the pitch detune does not exceed even 3 cents, well below the specified allowed threshold.

However, the step sequencer output does not meet the specified requirements. As was shown in table 8b, the measured output control voltages differed from the expected voltages, which would lead to out-of-tune oscillators when connected to any other synth using the 1V/oct standard.

There are room for improvements in many different aspects. For enhanced user friendliness, a display showing which note is selected would greatly improve the melody generation workflow. For improved packaging and robustness, the synth could have been implemented on a circuit board with soldered components - this would furthermore improve the aesthetics and portability of the synth. For improved sound generation, exponential stereo pots could have been used in the second order low pass filters. This would have broadened the filter cut-off sweet spot, and thus facilitate pleasing sound settings.

References

[1] Wikipedia. Analog synthesizer. https://en.wikipedia.org/wiki/Analog_synthesizer, 2023. Accessed: 2023-05-08.

[2] Mark Doty. Resurgence of analog synthesizers. https://performermag.com/best-instruments/best-music-keyboards-synth/resurgence-of-analog-synthesizers/, 2016. Accessed: 2023-05-08.

[3] Dominik B Loeffler. Instrument timbres and pitch estimation in polyphonic music. Master's thesis, Georgia Institute of Technology, apr 2006.

[4] Wikipedia. Relaxation oscillator. https://en.wikipedia.org/wiki/Relaxation_oscillator, 2023. Accessed: 2023-05-24.

[5] Moritz Klein. Diy vco part 3: Tuning your vco (and making sure it stays in tune). https://www.youtube.com/watch?v=dd1dws6pSNo&list=PLHeL0JWdJLvTuGCyC3qvx0RM39YvopVQN&index=3&ab_channel=MoritzKlein, 2018. Accessed: 2023-05-24.

[6] North Coast Synthesis Ltd. Exponential converters and how they work. https://northcoastsynthesis.com/news/exponential-converters-and-how-they-work/, 2018. Accessed: 2023-04-17.

[7] Texas Instruments. Lm386 low voltage audio power amplifier datasheet. https://www.ti.com/lit/ds/symlink/lm386.pdf, 2017. Accessed: 2023-05-24.

[8] Renesas. Icl7660 datasheet. https://www.renesas.com/eu/en/document/dst/icl7660-datasheet, 2010. Accessed: 2023-06-04.

[9] Microchip. Mcp4725 datasheet. https://www.electrokit.com/uploads/productfile/41013/MCP4725_2009.pdf, 2009. Accessed: 2023-06-03.

[10] Arduino. Debounce. https://docs.arduino.cc/built-in-examples/digital/Debounce, 2015. Accessed: 2023-05-24.

[11] Tony Miln. Regeneration: Synths beyond sustainability. https://soundgas.com/blog/regeneration-synths-beyond-sustainability-by-tony-miln/, 2018. Accessed: 2023-05-24.

[12] Rare earth magnet. https://www.sweetwater.com/insync/rare-earth-magnet/, 2009. Accessed: 2023-05-24.

6 Populärvetenskaplig sammanfattning

De analoga syntharna hade sin storhetstid på 70-talet. Som namnet antyder använde sig dessa av analoga kretsar för att generera ljud - med denna analoga lösning (med kontinuerliga signaler till skillnad från den digitala världens binära system) tillkom imperfektioner såsom brus och tonhöjdsavvikelser, som enligt många bidrog till ett varmt ljud. På senare år har de analoga syntharna gjort något av en comeback, efter att under decennier i allt större utsträckning ha ersatts av digitala synthar i både hård- och mjukvaruformat.

Målet med detta projekt är ta del av denna analoga pånyttfödelse i musikproduktionsvärlden, genom att designa en enkel analog synth utifrån tidigare projekt, elektronikteori och simuleringsprogram, samt implementera denna i hårdvara. Resultatet blev en fullt funktionell synth med bland annat fyra oscillatorer (ljudkällor), två lågpassfilter (som skulpterar ljudkällorna och kan göra ljudet mindre diskant), två mixers för att ställa in ljudnivåer oscillatorerna emellan, samt en förstärkarkrets med tillhörande högtalare. I tillägg har en digital step sequencer (verktyg för programmerbar melodigenerering) lagts till.

I stort fungerar synthen som väntat. I skrivande stund saknas dock stöd för batteridrift samt möjlighet att spela in synthens output (via en line out-utgång). Oscillatorn kan spela rent; dess tonhöjdsavvikelse understiger den övre gränsen (enligt specifikationen) på 15 hundradelars halvton över ett spann av fem oktaver.

Det finns flertalet möjliga förbättringar av synthen. Användarvänligheten för step sequencern kan förbättras med en skärm för att bland annat indikera vilken ton som valts - i nuläget sker denna inställnignen i blindo. Därtill kunde själva hårdvaruimplementeringen med fördel kunnat ske på ett kretskort istället för den nuvarande breadboardlösningen som är instabil och ger ett väldigt provisoriskt intryck.

7 Reflektion kring arbetets hållbarhetsaspekt

I detta avsnitt diskuteras projektet utifrån ett hållbarhetsperspektiv, med avstamp från tre av FN:s hållbarhetsmål (*Sustainable Development Goals* ; SDG).

- **Mål 12: hållbar konsumtion och produktion** Att synthen byggdes på breadboards i detta projekt kan anses som direkt positivt för detta mål - de komponenter som användes i kretsarna kan enkelt plockas isär igen och återanvändas. Dessutom har många av synthens komponenter använts i andra projekt tidigare. Denna cirkulära produktion är dock svårare att genomföra ifall synthen i förlängingen skulle produceras kommerciellt; många moderna hårdvarusynthar produceras med ytmontering med så pass små kretsar att de blir svåra att reparera vid eventuella skador [11]. Detta påverkar mål 12 negativt, då mindre elektronik repareras och återvinns, och istället ersätts av nyproducerat.

- **Mål 15: ekosystem och biologisk mångfald**. (Denna diskussionspunkt överlappar även med mål 12, se ovan). För förenkling av diskussionen av denna punkt dras detta projekt hållbarhetsaspekter i förlängingen till synthbranchen i stort. Som del av elektronikbranchen finns en stor risk att synthbranchen bidrar till miljöförsämringar kring gruvor. Exempelvis används neodymium i många synthar, framförallt i drivarkretsar för hörlurar och högtalare [12].

- **Mål 3: god hälsa och välbefinnande**. Projektet kan anses bidra indirekt positivt till detta mål. Ett av projektets övergripande syften var att i förlängingen bidra kulturellt, genom att skapa ett verktyg för musikalisk gestaltning. Att ha möjlighet att utöva sin hobby bidrar säkerligen till välbefinnande, och detta projekt bidrar till ytterligare redskap för detta. Det är emellertid diskutabelt huruvida den faktiska hårdvaran bidrar till detta - en betydligt mer resurssnål mjukvaruemulering av en analog synth bör även den bidra till välbefinnande (även om just önskad minskad skärmtid är en förekommande orsak till hårdvaruinköp inom musikproduktion).

8 Appendix A: Arduino code

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
Adafruit_MCP4725 dac;
#define DAC_RESOLUTION    (9)



// debounce (inspired by https://docs.arduino.cc/built-in-examples/digital/Debounce)
unsigned long prevDebounceTime = 0;  // the last time the output pin was toggled
unsigned long debounceBuffer = 50;



// sequencer variables & consts
const int MAX_STEPS = 8;
byte step_played = 0;  // step currently played by sequencer
byte step_set = 0;     // step currently set by user
bool PLAY_MODE = true; // True = Play mode, False = Set mode
int period = 500;
unsigned long current_time;


// Musical constants
const int MAX_SEMITONES = 13; //octave included
const int MAX_OCTAVE = 5;
const int tempo_lower = 10;
const int tempo_upper = 200;

int tempo_val = 0; //value from analog input [0, 1023], set by pot.
float tempo;       //tempo converted to [tempo_lower, tempo_upper]

int MAX_NOTES = 12*MAX_OCTAVE + 1; //include uppermost octave

// note type, including octave and semitone
typedef struct {
  int oct;
  int semi;
} note_type;

// sequence of notes to be played
note_type sequence[MAX_STEPS];


const byte pin_octUp = 8; //D8
const byte pin_octDwn = 7; //D7

const byte pin_semiUp = 6; //D6
const byte pin_semiDwn = 5; //D5

const byte pin_step = 4; //D4

const byte pin_tempo = A1;



//button state variables
```

```
byte octUp_state, octUp_prev;
byte octDwn_state, octDwn_prev;

byte semiUp_state, semiUp_prev;
byte semiDwn_state, semiDwn_prev;

byte step_state, step_prev;


const int scale = 4095;

void setup(void) {
  Serial.begin(9600);
  Serial.println("MCP4725A1 Test");
  dac.begin(0x60);

  pinMode(pin_octUp, INPUT_PULLUP);
  pinMode(pin_octDwn, INPUT_PULLUP);
  pinMode(pin_semiUp, INPUT_PULLUP);
  pinMode(pin_semiDwn, INPUT_PULLUP);
  pinMode(pin_step, INPUT_PULLUP);


  // initialise sequence to octave 1, semitone 0.
  for(int i = 0; i < MAX_STEPS; i++) {
  sequence[i].oct = 1;
  sequence[i].semi = 0;
  };


  delay(1000);


  current_time = millis();
}

void loop(void) {



  ///////////////// OCT UP ////////////////////////////////

  int reading_octUp = digitalRead(pin_octUp);

  if (reading_octUp != octUp_prev) {
    prevDebounceTime = millis();
  }

  if ((millis() - prevDebounceTime) > debounceBuffer) {

    if (reading_octUp != octUp_state) {
      octUp_state = reading_octUp;

      if (octUp_state == HIGH) {
        sequence[step_set].oct = mod(sequence[step_set].oct + 1, MAX_OCTAVE +
     1);
        Serial.println(sequence[step_set].oct);
      }
    }
  }
```

```
117   octUp_prev = reading_octUp;
119
121
      /////////////// OCT DOWN ////////////////////////////////
123
      int reading_octDwn = digitalRead(pin_octDwn);
125
127   if (reading_octDwn != octDwn_prev) {
        prevDebounceTime = millis();
129   }
131   if ((millis() - prevDebounceTime) > debounceBuffer) {
133     if (reading_octDwn != octDwn_state) {
          octDwn_state = reading_octDwn;
135
          if (octDwn_state == HIGH) {
137         sequence[step_set].oct = mod(sequence[step_set].oct - 1, MAX_OCTAVE +
       1);
            Serial.println(sequence[step_set].oct);
139       }
        }
141   }
143   octDwn_prev = reading_octDwn;
145
147
      ///////////////// SEMI UP ////////////////////////////////
149
      int reading_semiUp = digitalRead(pin_semiUp);
151
      if (reading_semiUp != semiUp_prev) {
153     prevDebounceTime = millis();
      }
155
      if ((millis() - prevDebounceTime) > debounceBuffer) {
157
        if (reading_semiUp != semiUp_state) {
159       semiUp_state = reading_semiUp;

161       if (semiUp_state == HIGH) {
            sequence[step_set].semi = mod(sequence[step_set].semi + 1,
      MAX_SEMITONES);
163         Serial.println(sequence[step_set].semi);
          }
165     }
      }
167
      semiUp_prev = reading_semiUp;
169
171
173   /////////////// SEMI DOWN ////////////////////////////////
```

```
175   int reading_semiDwn = digitalRead(pin_semiDwn);
177
      if (reading_semiDwn != semiDwn_prev) {
179     prevDebounceTime = millis();
      }
181
      if ((millis() - prevDebounceTime) > debounceBuffer) {
183
        if (reading_semiDwn != semiDwn_state) {
185       semiDwn_state = reading_semiDwn;

187       if (semiDwn_state == HIGH) {
            sequence[step_set].semi = mod(sequence[step_set].semi - 1,
        MAX_SEMITONES);
189         Serial.println(sequence[step_set].semi);
          }
191     }
      }
193
      semiDwn_prev = reading_semiDwn;
195

197   ////////////////////// STEP ////////////////////////////////

199   int reading_step = digitalRead(pin_step);

201
      if (reading_step != step_prev) {
203     prevDebounceTime = millis();
      }
205
      if ((millis() - prevDebounceTime) > debounceBuffer) {
207
        if (reading_step != step_state) {
209       step_state = reading_step;

211       if (step_state == HIGH) {
            step_set = mod(step_set + 1, MAX_STEPS);
213         Serial.println(step_set);
          }
215     }
      }
217
      step_prev = reading_step;
219

221   ////////////// TEMPO /////////////////
      tempo_val = analogRead(pin_tempo);
223
      // Map 0-1023 to BPM range
225   tempo = tempo_val*(float(tempo_upper-tempo_lower)/1023.0)+float(tempo_lower
        );

227   //Map BPM to sixteenth note duration
      //BPM bpm = BPM/60 bps <==> 60/BPM sec/beat <===> ...
229   // ...15/BPM sec/16th ===> 15000/BPM millisec/16th
      period = round(15000.0/tempo); //milliseconds
231
```

```
233
235   ////////////////////////// PLAY ///////////////////////////////////
      if (millis() > current_time + period) {
237     step_played = (step_played + 1) % MAX_STEPS;

239     //prints current step, octave and semitone
        Serial.print("step: ");
241     Serial.print(step_played);
        Serial.print(" oct:");
243     Serial.print(sequence[step_played].oct);
        Serial.print(" semi:");
245     Serial.print(sequence[step_played].semi);
        Serial.print('\n');
247     toDAC(sequence[step_played].oct, sequence[step_played].semi);
        current_time = millis();
249   }

251 }

253 //toDAC: converts the chosen octave and semitone to a value between 0 and
        4095
    //and tells the DAC to output the corresponding voltage
255 void toDAC(int oct, int semitone) {
      int fullnote = 12*oct + semitone;
257   float CV = round(fullnote*(4095/MAX_NOTES)); //49 semitones possible
      dac.setVoltage(CV, false);
259 }

261 //a mod function that handles negative inputs (as opposed to the % operator)
    // inspired by cwalger's answer: https://forum.arduino.cc/t/modulo-with-
        negative-int/158317/6
263 int mod(int a, int b) {
      return a < 0 ? ((a+1) % b) + b-1 : a%b;
265 }
```

DAC_test.ino