Microcontroller-based sound synthesis using affordable and off-the-shelf parts

Rodrigo Daltoé Madruga[1], **Marcelo de Oliveira Johann**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brazil

`{rodrigo.madruga,johann}@inf.ufrgs.br`

Abstract. *A basic software digital audio synthesis system is a system capable of generating arbitrary sound waves through the use of strict timed interrupts, look-up wavetables and a digital-to-analog converter. The advantage of the digital systems over the traditional analog ones is the use microcontrollers and memory devices that are easily configured, cheap, interchangeable, reprogrammable and make for a much smaller setup. As for 2021, a simple microcontroller-DAC system of high processing power can cost less than US$15.00 and be as small as a credit card.*

1. Introduction on sound synthesis

Analog sound synthesis is not a new concept. The first synthesizers were made of vacuum tubes and electro-mechanical technologies back in the beginning of the twentieth century. Sound synthesis got extremely popular in the 1960's with the popularization of discrete components and the rise of the usage of electronically generated sounds in music and movie scores.

The basic structure of an analog sound synthesizer consists of an input, a Voltage-Controlled Oscillator (VCO) in series with a Voltage-Controlled Filter (VCF) then in series with a Voltage-Controlled Amplifier (VCA) [Klein 1982]. The VCO is responsible for generating a single wave of multiple frequencies, with a main frequency that depends on the input of the user. Usually this is done via an electronic oscillator circuit, capable of generating basic function waves like sinusoids, triangular, sawtooth and square waves. The VCA is responsible for applying the ADSR envelope on the wave. The ADSR is the amplitude envelope used in synthesizers to mimic many characteristics of a naturally occurring sound. The ADSR envelope consists of four basic parts: Attack, Decay, Sustain and Release.

The Attack and Decay phases are responsible for emulating the impact of a player finger in acoustic instruments like a piano or guitar. The Sustain and Release phases are responsible for the natural fading of the sound. These four parameters are electronically controlled and can be manipulated by the user or another electronic system connected to the synthesizer. After the VCO generated a wave of multiple waves depending on the user's input, the VCA will control the gain of its internal amplifier to modulate the wave's amplitude.

The VFO is the most complex block of an analog sound synthesizer. After the wave had its frequency and amplitude controlled over time it can be filtered dynamically.

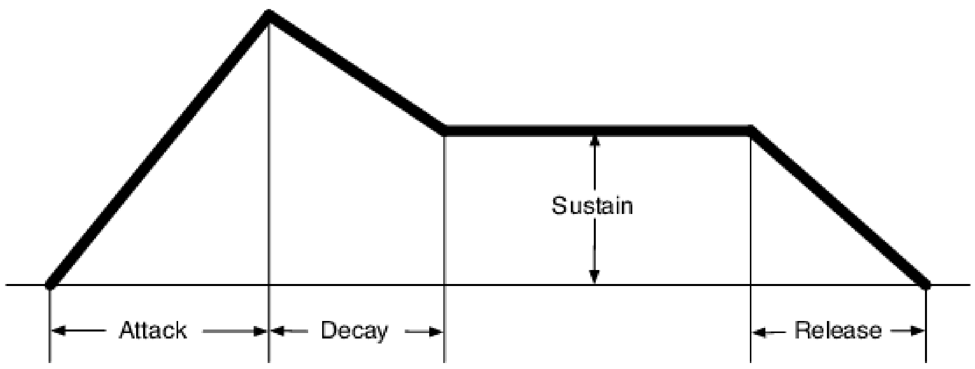

Figure 1. The ADSR amplitude envelope

The wave can have many harmonics and the filter can drastically change the sound characteristics of the wave. When the wave is in its final form, it can be mixed with other sounds, and the result connected to an output, to be listened or recorded.

The input can be of any kind, from a group of potentiometers to a weighted, piano like group of keys called keybed assembly. It is usual to make use of a variety of inputs in a synthesizer to control different parameters although as time passed MIDI keyboards started to become a standard. These keyboards consist of a keybed that has as an output a MIDI connection. MIDI is an audio control protocol that standardizes the many commands one can use in a synthesizer as what key was pressed or at what amplitude it is configured. With the use of MIDI keyboards many synthesizers consist of only potentiometers, referred sometimes as “knobs”, to control all the parameters of its internal blocks.

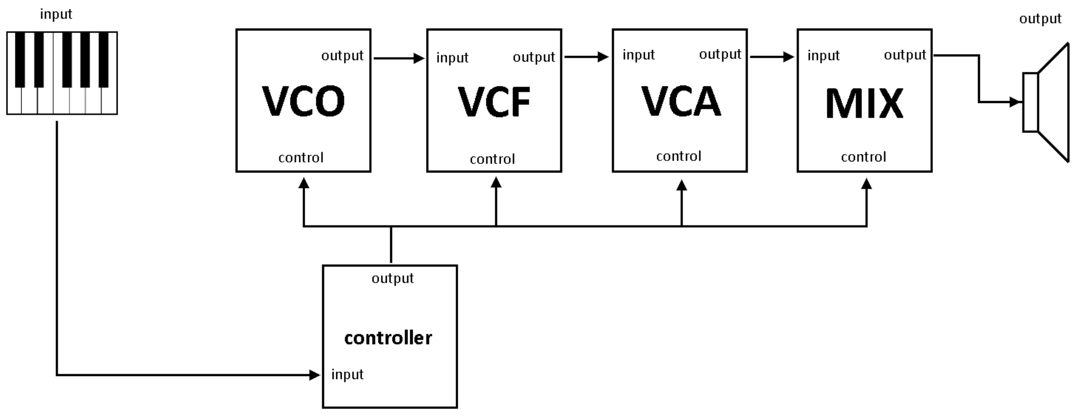

Figure 2. Analog synth basic structure

Finally yet importantly, we have a controller. The controller is the block that control other block’s parameters. The controller can define what ADSR profile a VCA will perform or what wave the VCO will generate. The controller can be the user, controlling those parameters directly or an electronic device inside the synthesizer. The controller itself can have many VCO’s internally in order to control the main VCO, VCA and VCF.

They usually oscillate in slow frequencies and are called Low-Frequency Oscillators.

The basic waveforms generated by the VCO's are generally described by basic mathematical functions such as a sinusoid, a triangular wave, a sawtooth wave and a square wave [Puckette 2007]. Many synthesizers use a random wave as well by generating white noise as a possible output. The output signal of a single VCO is called a voice, be it a sum of many waves or only one.

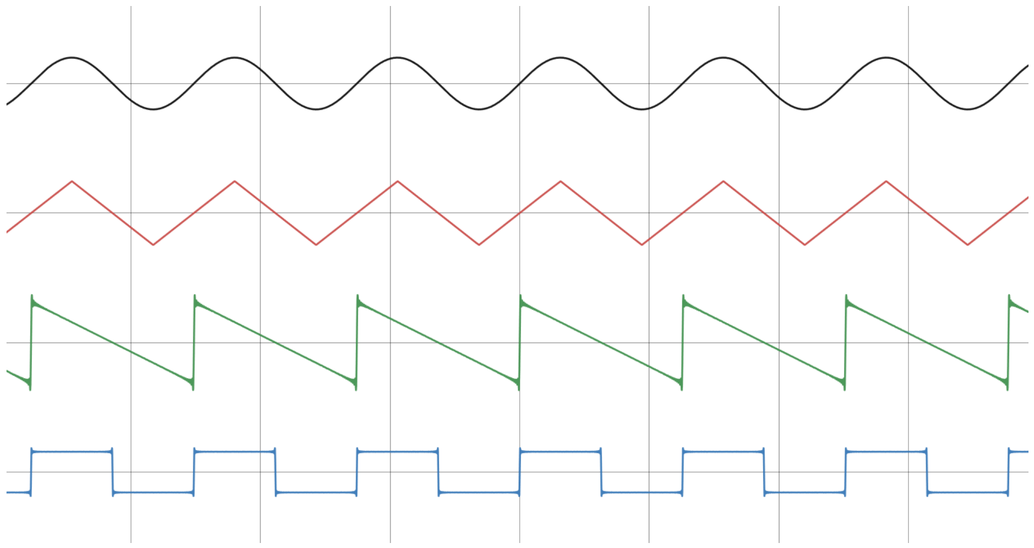

Figure 3. Sinusoidal, triangular, sawtoothed and square waves

By controlling the many basic blocks of an analog synthesizer, we can use many synthesis techniques in order to create very different and complex sounds. The most popular analog sound generating techniques are Additive Synthesis, Subtractive Synthesis, Frequency Modulation Synthesis and Sample-Based Synthesis. Additive synthesis consists of using one or multiples VCOs generating basic waveforms and summing them to result in a more complex waveform. Subtractive synthesis is a synthesis method that involves generating complex waveforms and then filtering the resulting wave in order to remove of boost specific frequencies. Frequency modulation synthesis consists in varying the frequency of the wave generated in the VCO using a signal created by another VCO, therefore modulating the frequency of the main signal. This control frequency is usually in the same order of magnitude of the main wave, sometimes even higher (usually the control frequency ranges from the same frequency to double the original frequency). Sample-based synthesis consists in recording sound previously and then replaying them controlling the speed, the position and the direction of the replay.

2. Digital software sound synthesis

The structure of a hardware digital synthesizer is not much different from an analog one. The main way to generate sound signals digitally is with a Direct Digital Synthesizer (DDS). Instead of a VCO there will be a Frequency Control Register (FCR) connected to a Numerically-Controlled Oscillator (NCO). You can connect digital filters and multipliers to act as the digital versions of the VCA and VCF [Vankka et al. 2001]. Before the output, you need to pass the digital signal through a Digital-to-Analog Converter (DAC) so your data bytes are converted to an analog sound and through a Reconstruction Low-pass Filter (RLF) in order to remove the jagged high frequencies, product of the DAC conversion.

This project aims to design a software version of a hardware DDS system. Instead of a FCR we have a Phase Accumulator (PA), a variable that is used to read the Look-Up Table (LUT) used in the synthesis in the place of the NCO. The LUT is nothing more than a numeric vector that stores the many values of a sampled wave with a fixed sample rate.

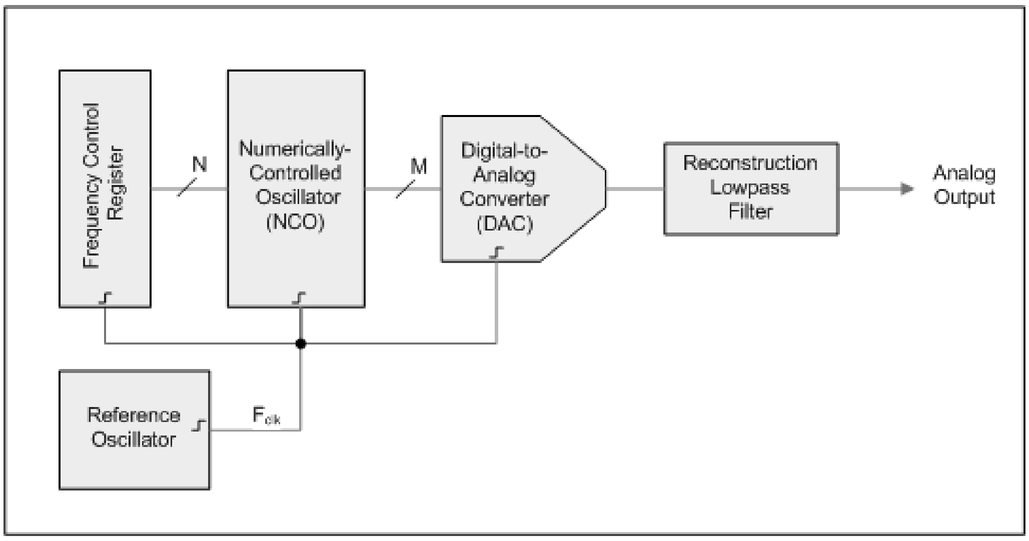
Frequency Control Register
N
Numerically-Controlled Oscillator (NCO)
M
Digital-to-Analog Converter (DAC)
Reconstruction Lowpass Filter
Analog Output
Reference Oscillator
F_{clk}

Figure 4. Hardware DDS system

That way the software structure emulates the hardware structure. A variable counter will be added an increment every iteration. These iterations will be constructed in a loop. Then the counter's value will be used as index to read the numeric vector in order to retrieve the current value of the wave we sampled. This numeric value can be filtered (together with a sample of its past values), multiplied or divided in order to apply a software gain and then sent to the output to be converted.

The software DDS still needs a filtered DAC to generated the signals that are used in the output of the synthesizer. As for a reference oscillator, we need a hard realtime interrupt to keep the sample rate stable and not introduce sampling frequency jitter. This is important for jitter noise can degrade the quality of the generated signal.

Inputs will be read by the microcontroller and then applied accordingly in the software DDS to control the frequency of the wave generated, its amplitude and many other properties.

As the core of the software DDS is a look-up table, any waveform can be generated. Digital synthesizers can emulate analog synthesis methods as additive synthesis, subtractive synthesis, frequency modulation synthesis and sample synthesis, all of them with obvious limitations due to the digital nature of the system. Not only a digital synthesizer can emulate the basic analog synthesis methods but also can perform other methods, reserved to digital only synthesizers, as for example Wavetable Synthesis. Wavetable synthesis is the result of the basic structure of a DDS. Having a look-up table and applying its values to an output is the definition of wavetable synthesis. We can dynamically change those values, scan at different rates, using different increments, after that filter, and amplify those values before applying them to the DAC in the output.

Many auxiliary modules used in modular analog synthesizers can still be emulated in software DDS as stutterers, arpeggiators, delay and reverb modules, etc. With a software DDS, the limit of voices and effects is determined by the performance of the microcontroller used. A FPGA could be used but it would defeat both the lowcost and availability aspects of the project. Fortunately, as for 2021, cheap and powerful microcontrollers are the norm, with the performance envelope being pushed every year. As every single module of the DDS is run in software, the processor in the microcontroller needs to be powerful enough to make all the calculations and accesses before the sample time ends and another cycle begins. This project aims to not only develop a basic microcontroller-based synthesizer but also to test the performance limit the chosen microcontroller can reach.

3. Digital vs. Analog

A digital system has some limitations compared to the analog counterpart. The main one is the discretization of all the signals. A digital signal is a discrete version of the continuous wave of the analog signal used as reference. Discrete signals have a finite number of possible values instead of infinite possible values as continuous signals [Hartmann 2013]. These discrete values are generated by reading the continuous values of the analog signal and converting them to a digital quantity through the process of discretization. The interval of time these conversions occur is called the Sample Rate (SR). The sample rate amount of times the analog signal is converted to a digital value in one second, being measured in Hertz [Hz]. The amount of possible digital values that can be obtained each conversion is a product of how many bits are used in the conversion itself. The more bits, the more possible values can be measured. The amount of bits of the digital value is called Bit Depth (BD). The amount of possible values in the amplitude axis of the signal is calculated by elevating 2 to the power of BD. As a general rule, the more bits and the biggest the sample rate the better but there will be a point where the cost of the improvement will be way to high compared to the negligible improvement of the signal's quality. The industry standard for professional studio quality sound can go as far as a 192 kHz SR and 24 bits for the digital values [Gentile and Cushing 1999]. This project aims to achieve a result close the CD standards (16 bits/44.1kHz), with initial sampling values of 16 bits BD, 40 kHz SR.

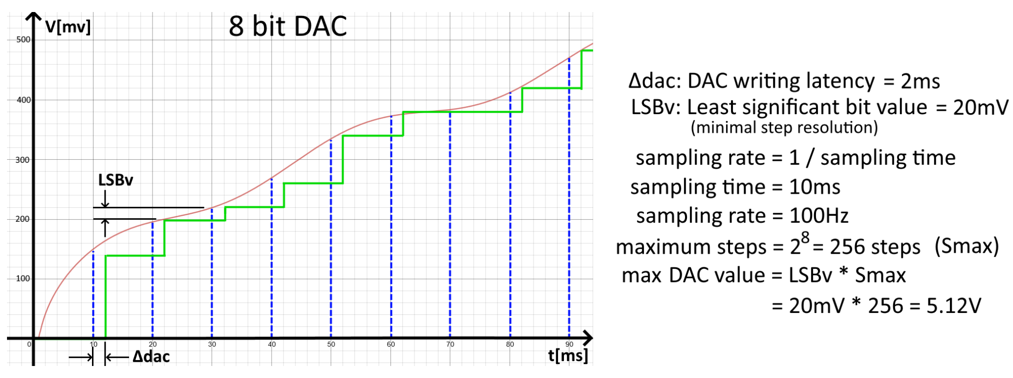

Figure 5. Analog signal discretization example

4. Project parts and modules

4.1. MCU

The main goals are to develop a software DDS system that is not only cheap and accessible but also simple to design and replicate. The first and most important choice to be made is which microcontroller (MCU) module would be used. In the early days of consumer electronic components market Arduino was always the answer but in the last decade ARM and Espressif architectures have conquered their space with better performance and affordable modules. As discussed before, it is paramount to choose a powerful microcontroller processing wise. The most popular budget, 32 bits, modern microcontroller in the market is the ESP32. Not only is less than half the price of any ARM module it still have a competitive performance. Easily available, cheap and powerful, this is the MCU of choice for the project. The ESP32 is a dual-core, 240 MHz microcontroller capable of handling up to 16 MB of flash memory (4 MB standard), up to 8 MB of PSRAM in some modules and the I²C, I²S, SPI, UART, SDIO protocols. The performance limit of the MCU will be tested based on preliminary tests on previous prototypes. The somewhat limited number of GPIOs can be extended with the use of expansion modules based on the aforementioned serial protocols. Basic ESP32 boards can be found for less than US$5.00.

4.2. DAC

The next most important device of the project is the DAC. After a lot of research on what module would be the best choice, it was decided to go for the PCM5102A. It is an I²S DAC with a built-in reconstruction filter. I²S is a very flexible protocol that allows the programmer to set a custom SR and a value of BD that goes from 16 up to 32 bits. One important detail about the chosen DAC is the built-in filter, for on its absence there would be the need to make one via software, consuming precious microseconds of processing time. Just as the MCU, this DAC can be found on every parts dealer website and it is very affordable, being sold for less than US$7.00.

4.3. Inputs

As this projects aims to develop a self-contained design, a five octaves keybed assembly will be used as main input for the musical notes. Many knobs, buttons, LEDs and switches could be implemented using GPIO extenders. The main input interrupt routine should be able to read not only what key was pressed in the key matrix but also read the speed with which the key was pressed. That is possible with key matrixes that implement double switching per key. The key has two switches, and a rubber pad that makes the connection between the two poles of each key. The rubber pad is not symmetrical, connecting the first switch before the second. The faster the user presses the key, the shorter the interval between the two switch presses will be, allowing the microcontroller to measure the speed of the key press. The key matrix will be multiplexed in both dimensions to reduce the number of pins used to map it. As the interrupt routine used to read the key matrix has to complete all the readings in a very short time period the multiplexing topology was chosen one. For other less time-critical inputs, the design will use or shift registers or serial protocol expanders.

5. Deep Note as benchmark sound

To test the performance of the software DDS sound engine the Deep Note was chosen to be the first voice to be implemented. The Deep Note was a sound created by James Moorer in 1982 as a way to demonstrate the power of his company's sound engine that would become THX in the near future. The deep note makes use of 30 sawtooth voices that change simultaneously in frequency and amplitude. There is also a good use of reverb and randomness to some properties of the voices used.

To generate a sound that resembles the Deep Note in any degree the software DDS should be able to change all the properties of all voices in any given instant. For that there will be two interrupt routines running in the ESP32. One at 40 kHz that updates the DAC's output value. This one is the sampling interrupt. The second interrupt will happen every 1 ms and will be used to read the inputs and change the properties of each voice over time. Using this second interrupt is possible to create LFO's of different frequencies and apply their values to a certain voice's frequency or amplitude control. This way we can use the synthesis methods described in Additive, FM and wavetable synthesis techniques.

To simulate the best way to generate specific sounds and what wavetables could be used as LUTs, an application will be developed to generate and play this wavetables. After the best LUT is chosen by the programmer, it can be exported as a vector of values to be used in the microcontroller's code.

6. The sound engine

Making use of both cores of the ESP32 the sound engine will be divided into two hard real-time interrupt routines. One is going to track all the envelopes and ADSR, FM and LFO's outputs that control the many voices being played. The second interrupt is going to apply all controls to the voices and calculate the output, apply filters and effects and then send the resulting value to the DAC.

After the sound engine is completed and tested, it will not only be able to run many voices simultaneously but also many layers of voices. A user can press a sequence of keys and then this sequence can be replayed, with period configurable by the user. This allows the user to play many layers of a certain music track, using the many voices in the microcontroller's memory. An UI will be developed to make the configuration of the many synthesizers' properties easy.

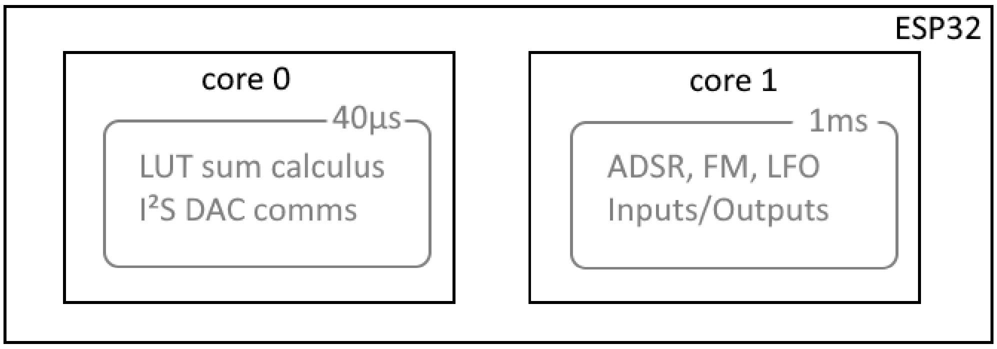

Figure 6. Internal interrupt architecture

7. Proposal Roadmap

This section presents the timeline to be followed on the process of designing, building and testing of the final prototype. The main goal now is to develop a fully functional software digital DDS system with fully functional keybed and UI, capable of emulating the basic voices and effects of any budget commercial keyboard. The following proposal shows the next steps of development:

Proposed Steps	2022				
	01	02	03	04	05
Keybed timing tests	X	X	X		
Dual-core timing tests	X	X	X		
basic sound engine tests	X	X	X		
Sound effects vector tests			X	X	
Key-press layer automation			X	X	
Writing	X	X	X	X	X
Presentation					X

Table 1. Project implementation timeline.

In Table 1, the estimated timeline of all the tests and new features that are planned to be implemented and tested. Since the keybed, dual-core and basic sound engine timings and tests have already begun in the prototype, the first three months will be dedicated to polishing up the code and finishing the many test variables. March and April will be dedicated to sound tests, in order to hear the final product of the many algorithms and evaluate the quality and similarity with the commercial counterparts. Lastly, the project will be presented to an examination committee.

8. Final Considerations

As referenced in 4.1 MCU, the performance limits of the ESP32 with the given interrupt timings are hard to define by simulations or estimates. The fastest and most reliable way to test the limits of the design was to develop some prototype codes and prototype circuit boards and to run it with a few controlled variables: The sample-rate, the amount of voices being processed simultaneously, the amount of bits used for the wave vectors and the latency for the slow interrupt routine that will work on the input/output and ADSR/LFO stages.

Two prototypes were developed: Prototype 1 was built to test the feasibility of a basic synthesizer design using an ESP32 and a common cheap DAC (the MCP4725 12 bits DAC was used for the built-in DAC of the ESP32 is rather non-linear and bad performing). The prototype had only one button that when pressed activated a sequence of notes and control changes. It was able to run 11 voices at a SR of 100 kHz. This test made clear that the slow interrupt of 1 ms could easily handle all input/output and ADSR/LFO operations with a lot of spare room for extra functions.

A second prototype was then developed to apply the learned concepts and results of the previous iteration towards a more advanced design in order to test if all of the desired features could be achieved with the proposed cost and availability requirements. In order to isolate the simulation variables, the prototype was divided in two blocks, both using ESP32s. One board would only read the inputs and operate an UI output and the other board would receive the commands from the first board and operate the DAC. This prototype already had the chosen DAC (the PCM5102A), a more modest SR of 40 kHz in order to increase the sampling time (from 10 µs to 25 µs) and function to operate the module that reads the 5 octaves keybed in the 1 ms interrupt routine. By separating the two main parts of the project, it was possible to measure the timings and performance of all codes in a separate manner, although the final design will make use of only one MCU. The results were very promising. The reading of inputs and setting of outputs takes much less than 1 ms to be completed, which leaves the core 1 with a lot of unused processing time that could be used by asynchronous functions. The performance of core 0 was put to the test, now operating with a much better DAC using I²S. It was able to run an impressive number of 61 voices simultaneously, passing the resulting sound wave through a reverb filter and then outputting the resulting wave to the I²S pins.

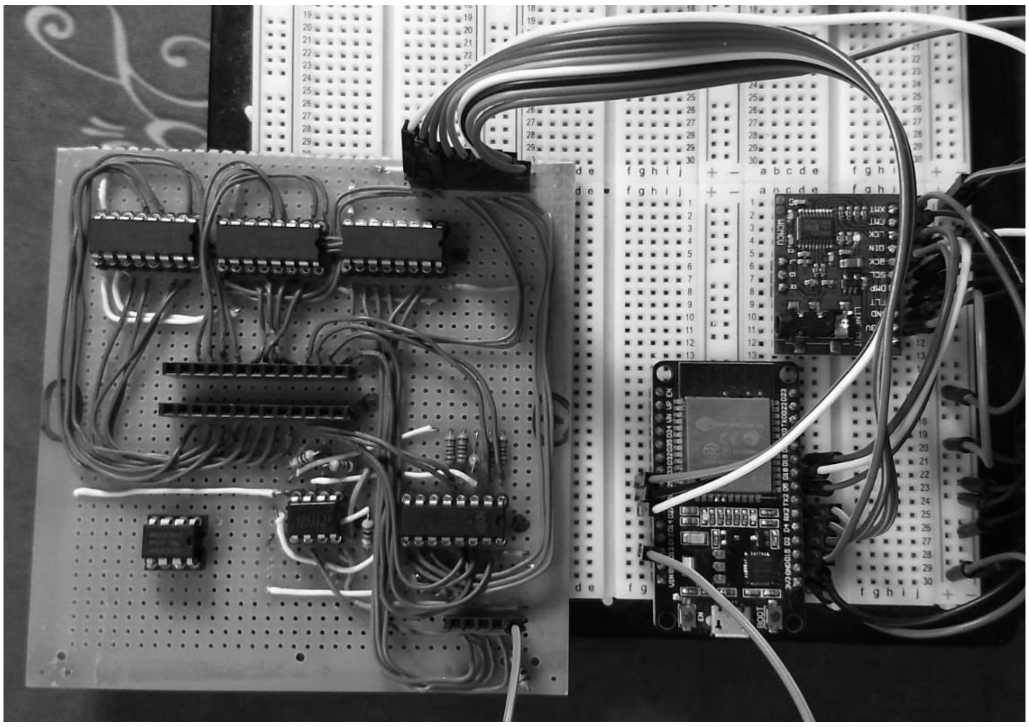

Figure 7. Picture of second prototype: Multiplexing board, ESP32 and DAC

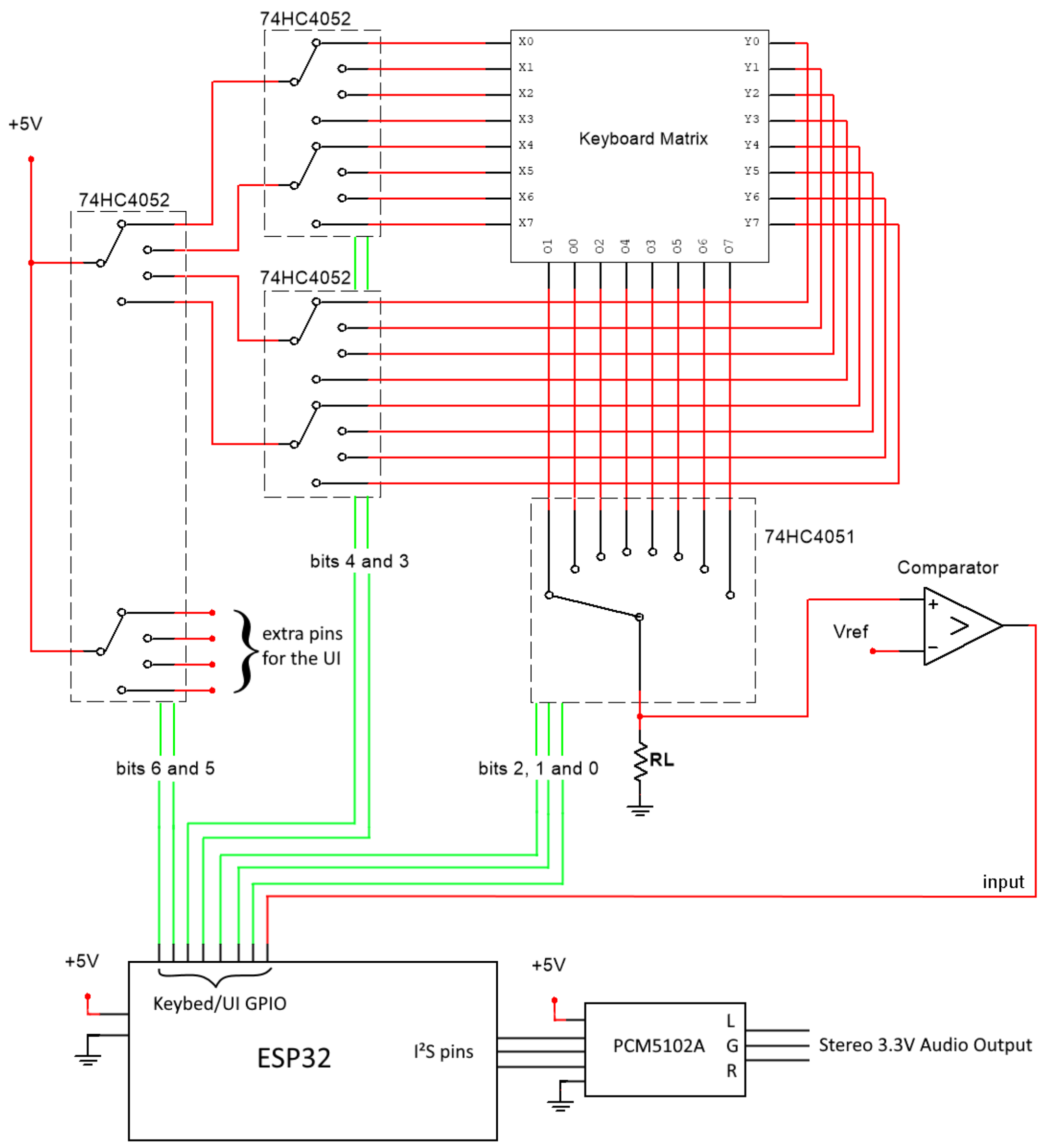

Figure 8. Second prototype circuit design

References

Gentile, K. and Cushing, R. (1999). *A Technical Tutorial on Digital Signal Synthesis*. Analog Devices.

Hartmann, W. M. (2013). *Principles of musical acoustics*. Springer.

Klein, B. (1982). *Electronic Music Circuits*, volume 21833. Sams Technical Publishing.

Puckette, M. (2007). *The Theory and Techniques of Electronic Music*. World Scientific Publishing Company.

Vankka, J., Halonen, K. A., and Halonen, K. (2001). *Direct digital synthesizers: Theory, design and applications*, volume 614. Springer Science & Business Media.