DIGITAL FM MUSIC SYNTHESIZER

Omar Ahmed | Uvaldo Resendez | Jessica Stensby

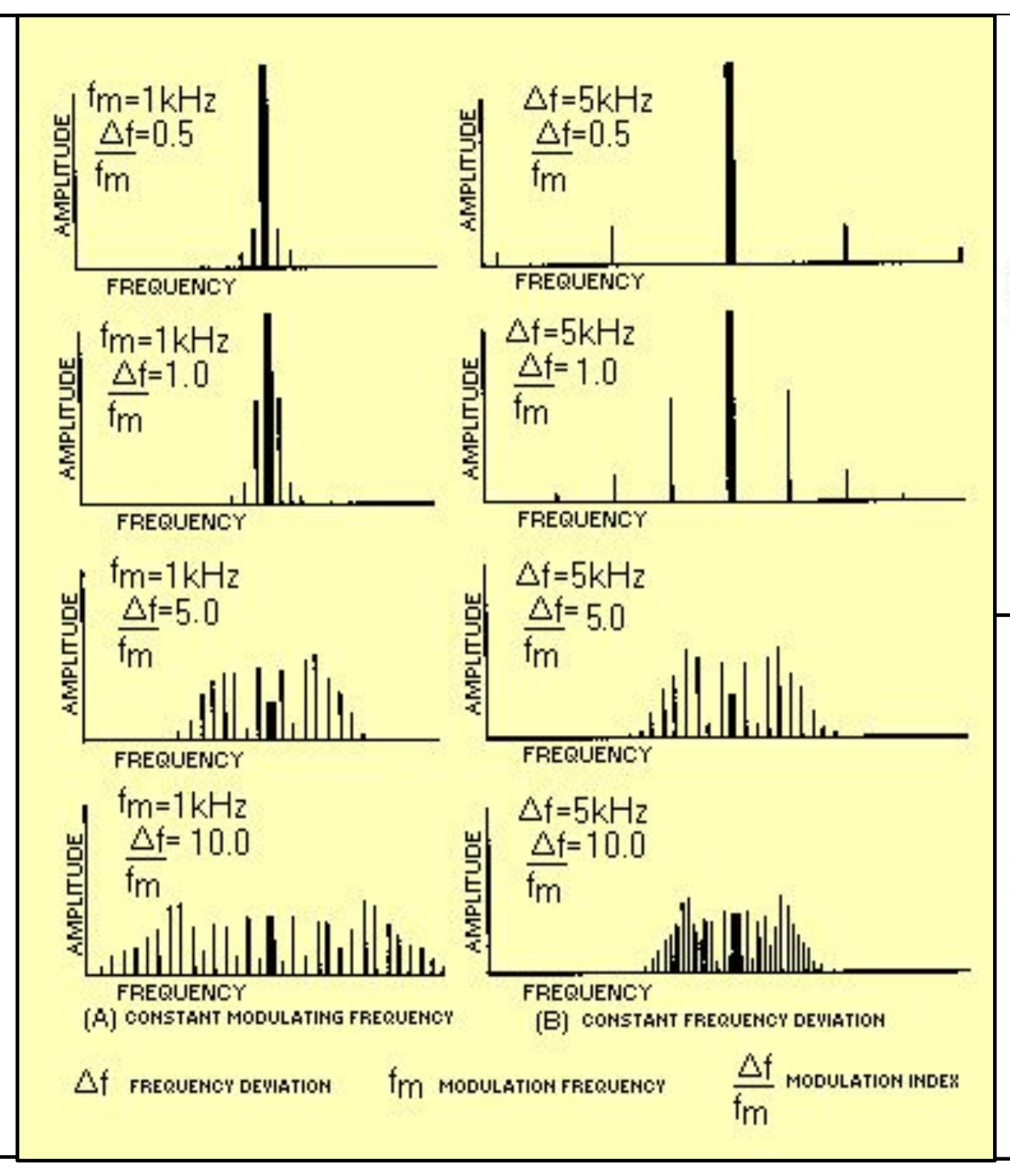
SAN DIEGO STATE UNIVERSITY

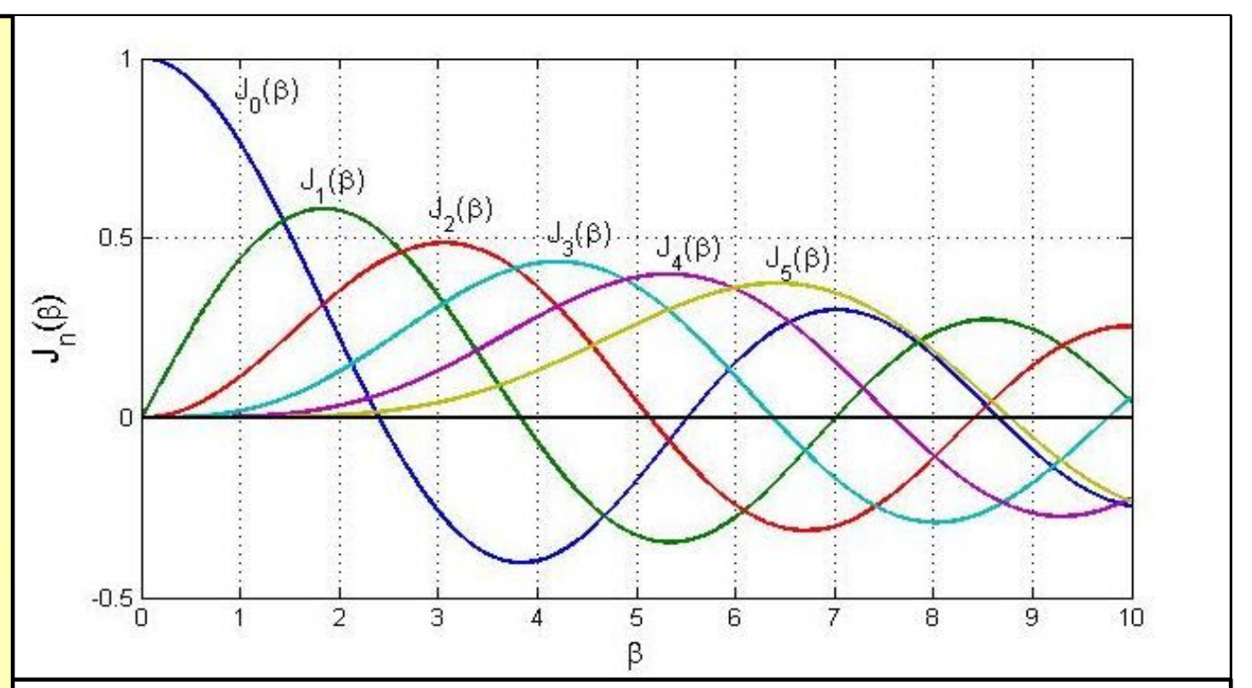
Abstract :

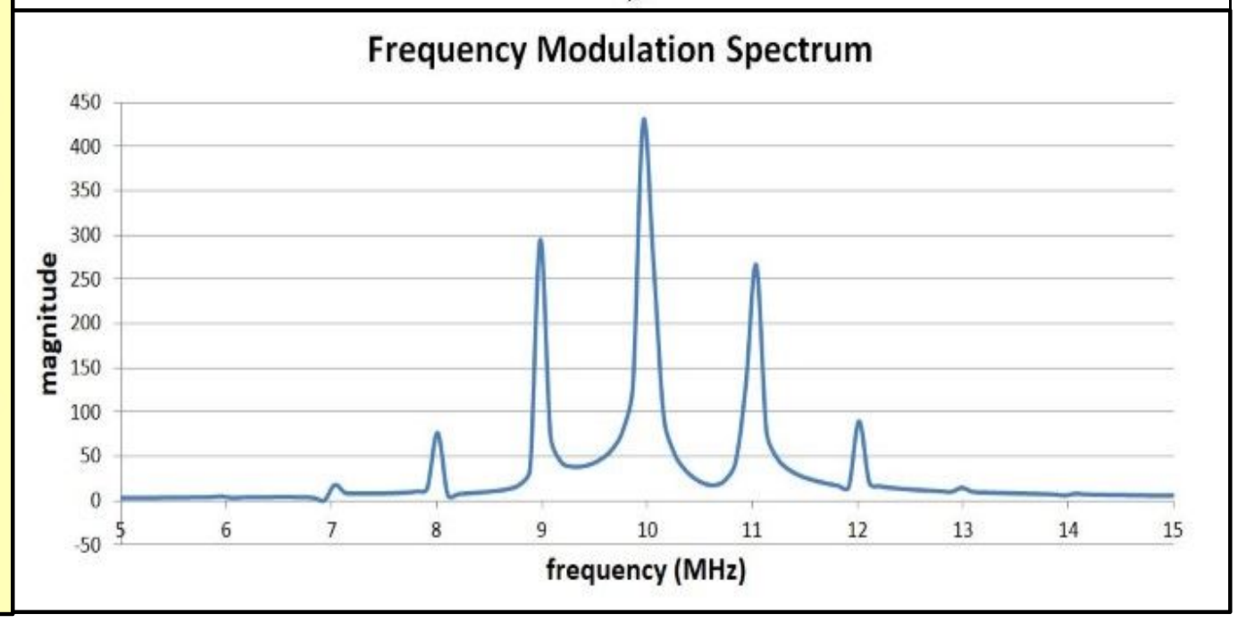
The shape of a sine wave is a familiar and iconic shape in mathematics. This project's goal is to manipulate the traditional sine wave by use of frequency modulation to craft unique sounds. By using frequency modulation and MIDI data, which represents only qualities of sound waves and doesn't carry a sound quality aspect, we can use simple math and functions to change pure tones into fun, distorted sounding music.

FM Modulation :

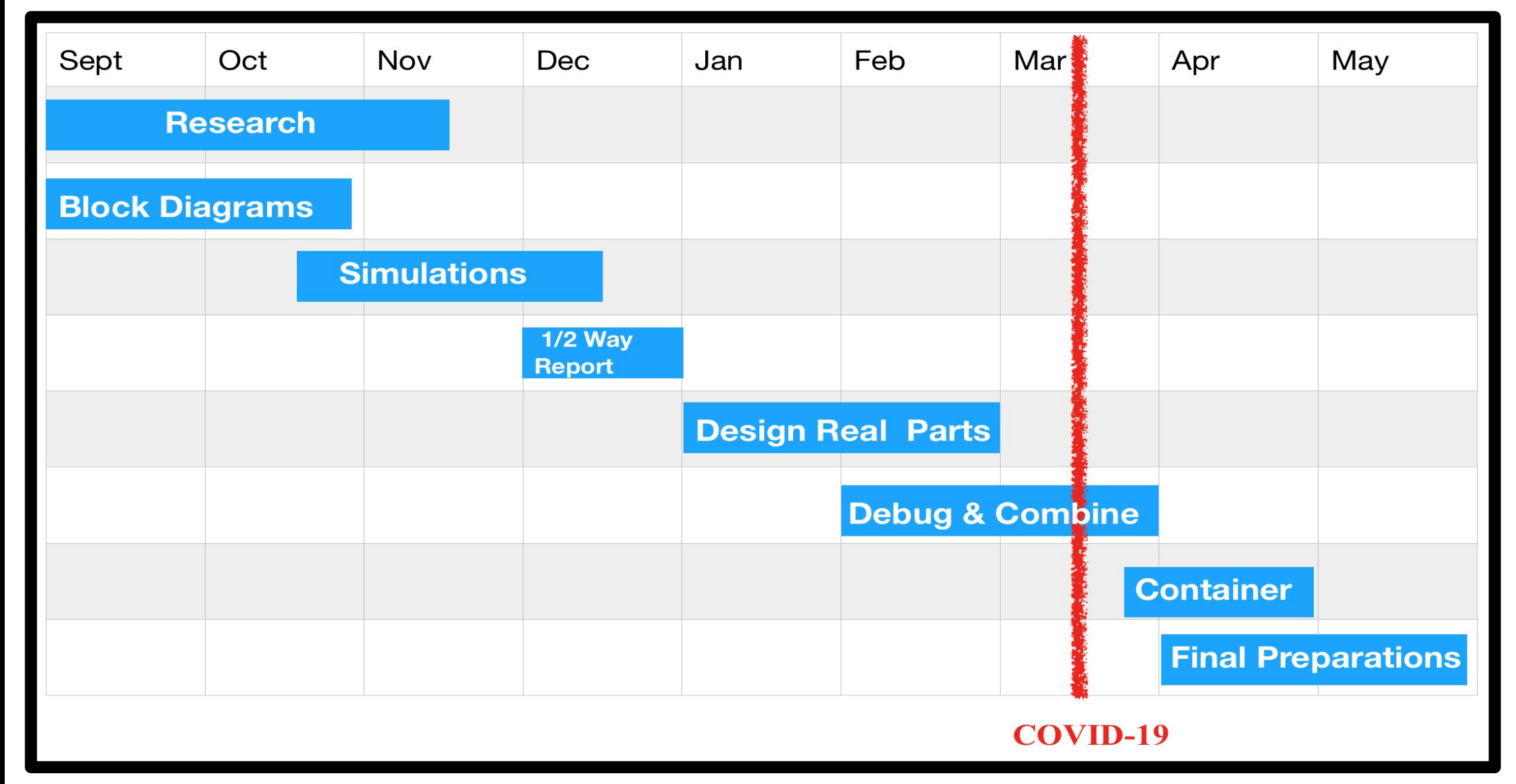
Fm modulation is the process of changing the frequency of periodic (carrier) waves through time in a way governed by an external signal. When both the carrier wave and the external signal are sinusoidal they display interesting spectral characteristics. Sidebands are created in accordance to the external signals frequency and amplitude. These sidebands could be characterized by inputting the ratio of amplitude to frequency (modulation index) into the Bessel equation of the first kind.





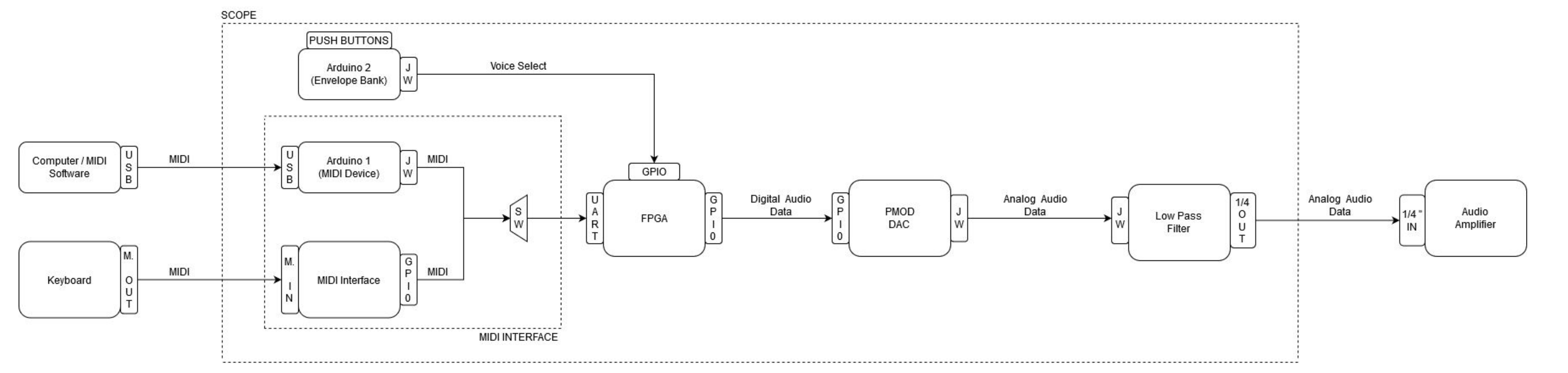


TIMELINE & CUTOFF :



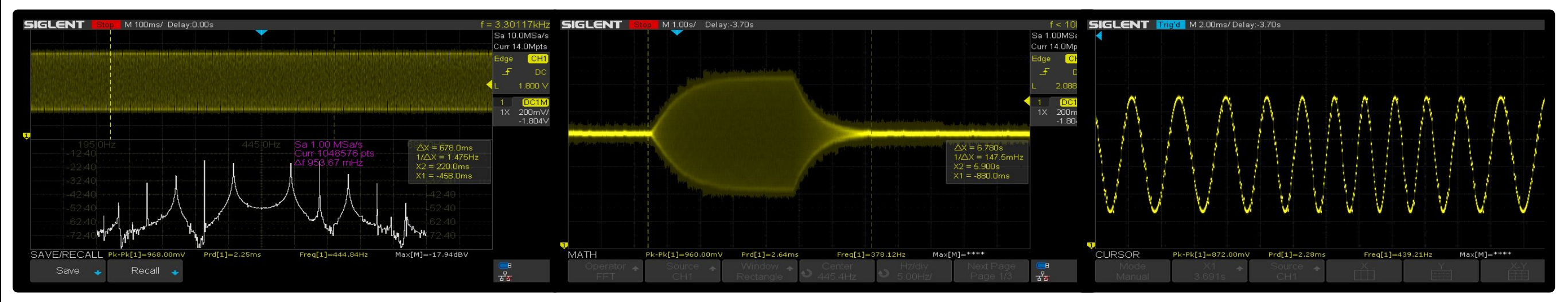
HOW MIDI WORKS

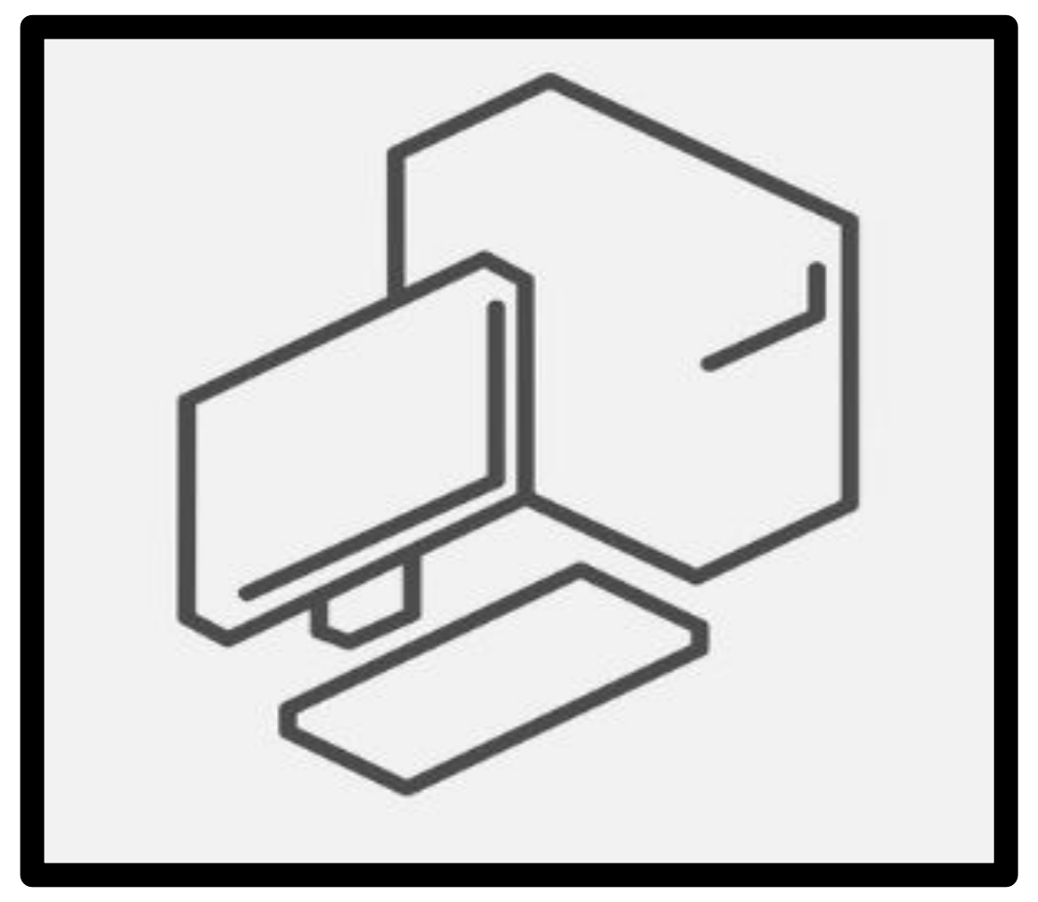
MIDI is a set of numerical values that provide the information of a note or series of notes. There is no recording so MIDI does not have a quality association with it. It stores note velocity, note pitch (the key), attack and delay, and release. All of these are qualities of the note played, represented by numbers, and therefore can be mathematically manipulated and converted into analog signals, which is the technique used for this synthesizer.

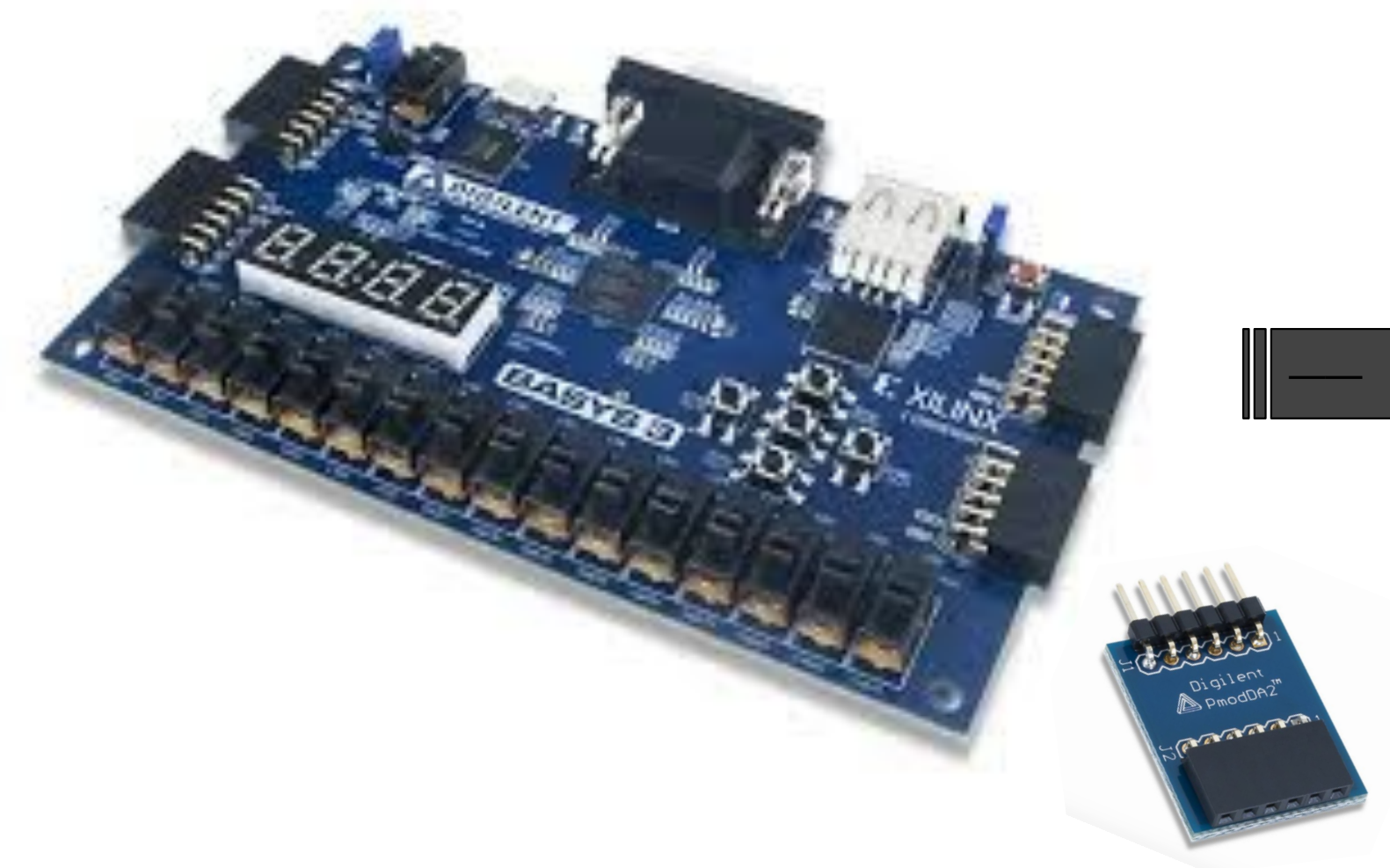


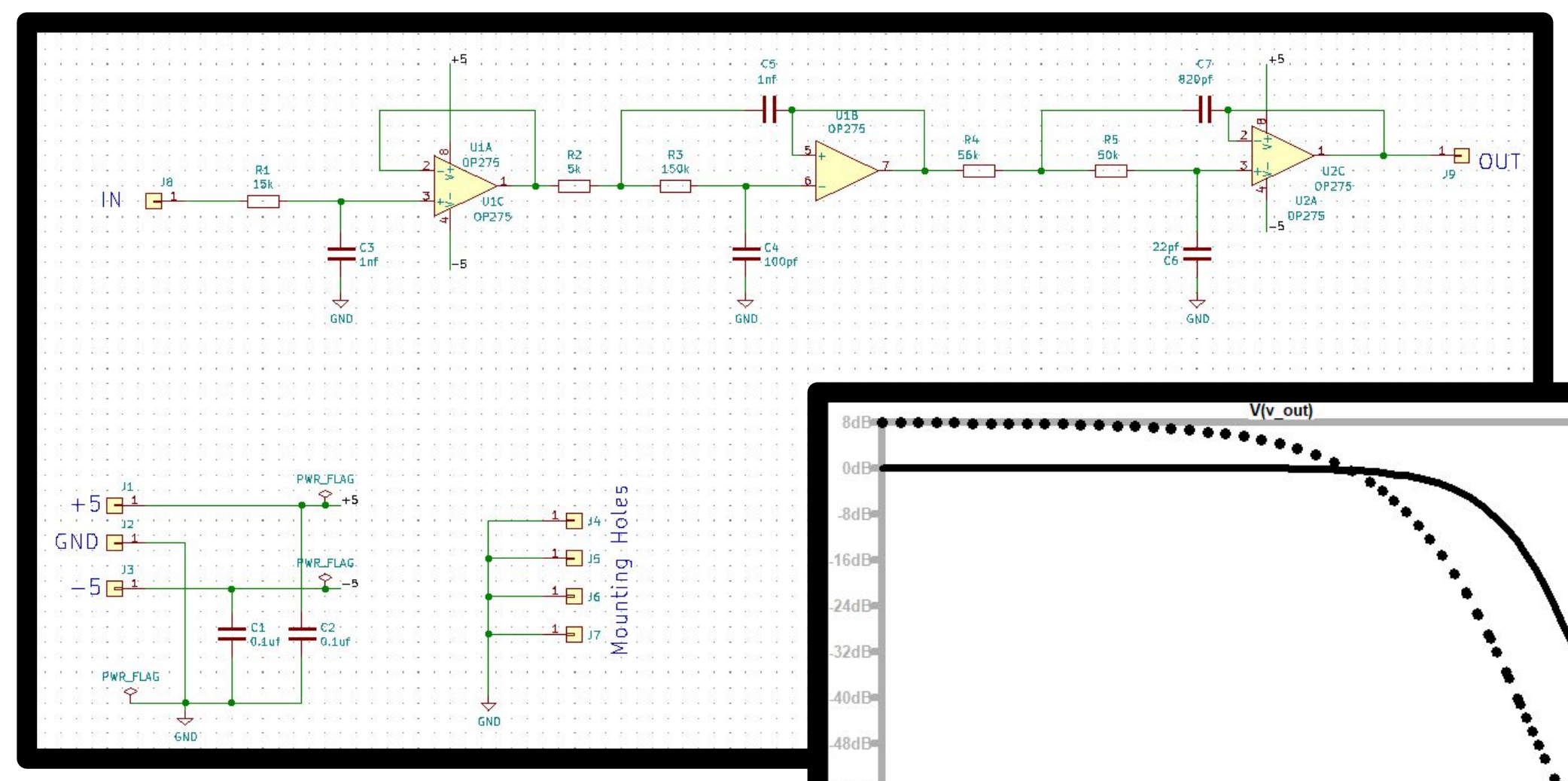
Results :

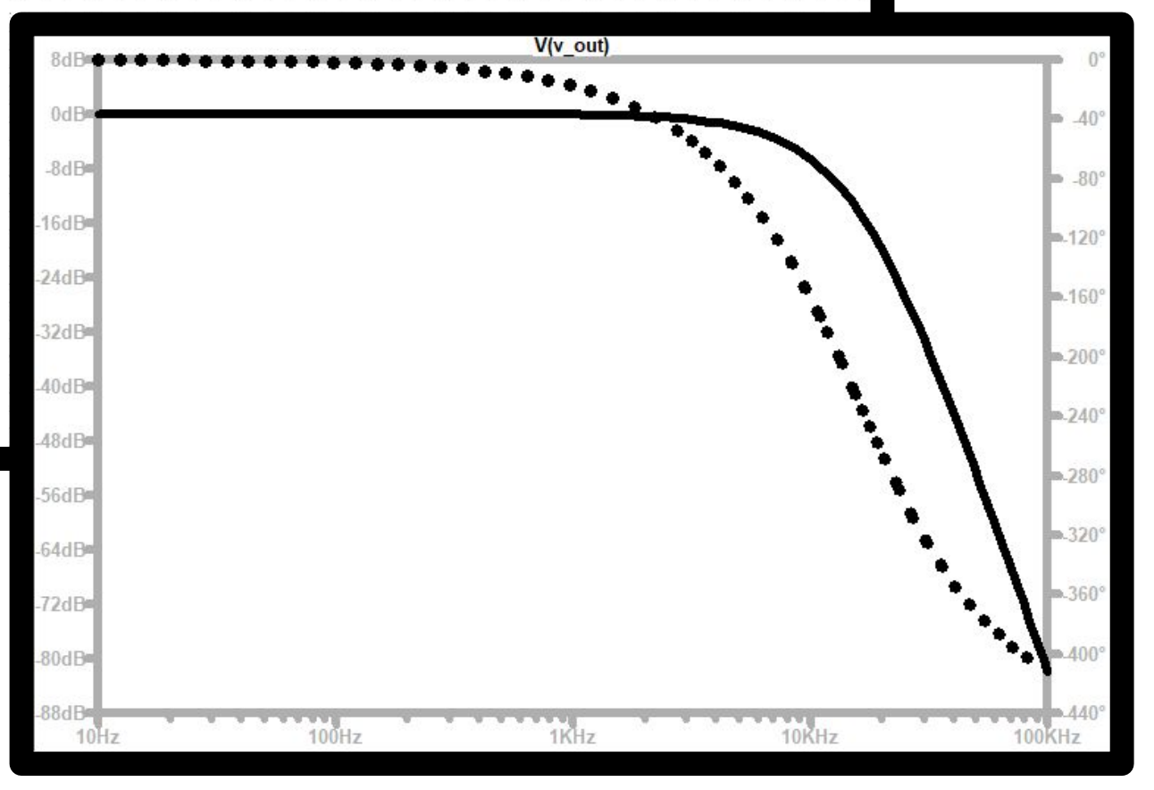
We were able to create the FM modulator and verify our result using the Bessel Function characteristic equation. The first image below shows the spectrum of one of our test were we set the the modulation index to 2.41 and as predicted by the characteristic equation we were able to NULL the carrier. We were also able to apply both an amplitude and a modulation envelope to our signal. The second picture illustrates the effect of our amplitude envelope. Due to the lack of an assembled project, these signals were all created by sending MIDI commands to the FPGA Via MATLABs serial commands.











Computer & Interface :

The computer based interface was part of the preliminary design goals and schematics. The concept was that by using MIDI as our way of transmitting sound information, we would be able to use software and composing programs to send MIDI data to the Arduino. However, it was discovered that Arduinos do not take non-binary data well. After hours of research and code, it was concluded that it would take outside devices such as a shell program on a computer, or an Arduino shield to send forth the MIDI data.
This was determined to be out of the scope of this project, and the full brunt of the interface was transferred to the Arduino.

Arduino Implementation :

The Arduino is the control center for the digital synthesizer. The role of the Arduino is to accept MIDI data from a compatible instrument. The Arduino would be in charge of not only converting and transmitting this data to the FPGA so it could be read properly, but also store all of the preset variables the FPGA would use to manipulate the MIDI files.

The Arduino also became a bigger hardware component, as physical buttons representing different sound profiles, such as "Guitar" or "Electric organ", would be hooked up and set to trigger the Arduino to send the accompanying variables to the FPGA. This would be the interface users could interact with, and allow them to play around with the synthesizer.

Lastly, the Arduino would be in charge of making sure the BAUD rate was synchronized amongst the devices so that information could be transmitted properly in the synthesizer.

FPGA Implementation :

The FPGA consisted of four main system level blocks. These include three finite state machines (the UART module FSM, the MIDI interface FSM, and the output DAC FSM) and a sound core. The UART and MIDI interface FSMs fetches and decodes the input MIDI commands and sends the appropriate information to the sound core. The sound core does the computation and outputs the digital sound bits which is then made ready for the DAC by the output DAC FSM. On the sound core, oscillations were performed using numerically controlled oscillators (NCOs), structures that oscillates at a frequency dictated by the number given at its input, and modulation were performed on the frequency of the oscillating wave by changing the input of the NCO in the desires way. Modulation and amplitude envelopes were also applied on the output waves to further increase the complexity of the sound waves (making them sound cooler).

Filter Implementation :

At the output of the DAC it is necessary to have a filter to to ensure that there are not any replicas of the signal present. The filter was designed using Analog Devices' Filter Wizard Tool with a 3 dB cutoff frequency of 20 Khz, and designed to have an 18 dB per octave response. From this original circuit, component values were changed to match what was readily available. The final circuit and frequency response are shown above in the flow chart.

Special thanks to Professor Dorr