TITLE PROJECT

Continued from page

FINAL Project 6.115

Goals: Create a harmonizer that takes audio input and replicates the same wave form characteristics at different frequencies to create a "choir" of the audio input. These complementary frequencies will be chosen via SPST switches. I am in part inspired by famous musicians who have version of this - Imogen Heap & Jacob Collier. I am interested in transforming soloists performances.

These goals will be accomplished after completing several tasks: wiring up the microphone and speaker circuit, wiring the PSoC to the R31JP for debugging purposes, implementing the FFT on the PSoC, accomodating an ADC & multiple DACs on the PSoC, and wiring the switch shematic.

Hardware Description: The audio signal enters the FET microphone which is amplified via an LM386 and then a series of scaling and filtering OP-AMPS. High frequencies are filtered via capacitors and the signal is maintained via a follower OP-AMP that goes to the PSoC. The PSoC reads this value via an ADC and processes the signal via an FFT.

Then there is a set of 8 SPST switches tied to ground that are connected to the PSoC via high impedance pins. If the switches are pressed the PSoC reads 5V and sends out corresponding signals via onboard 8-bit DACs.

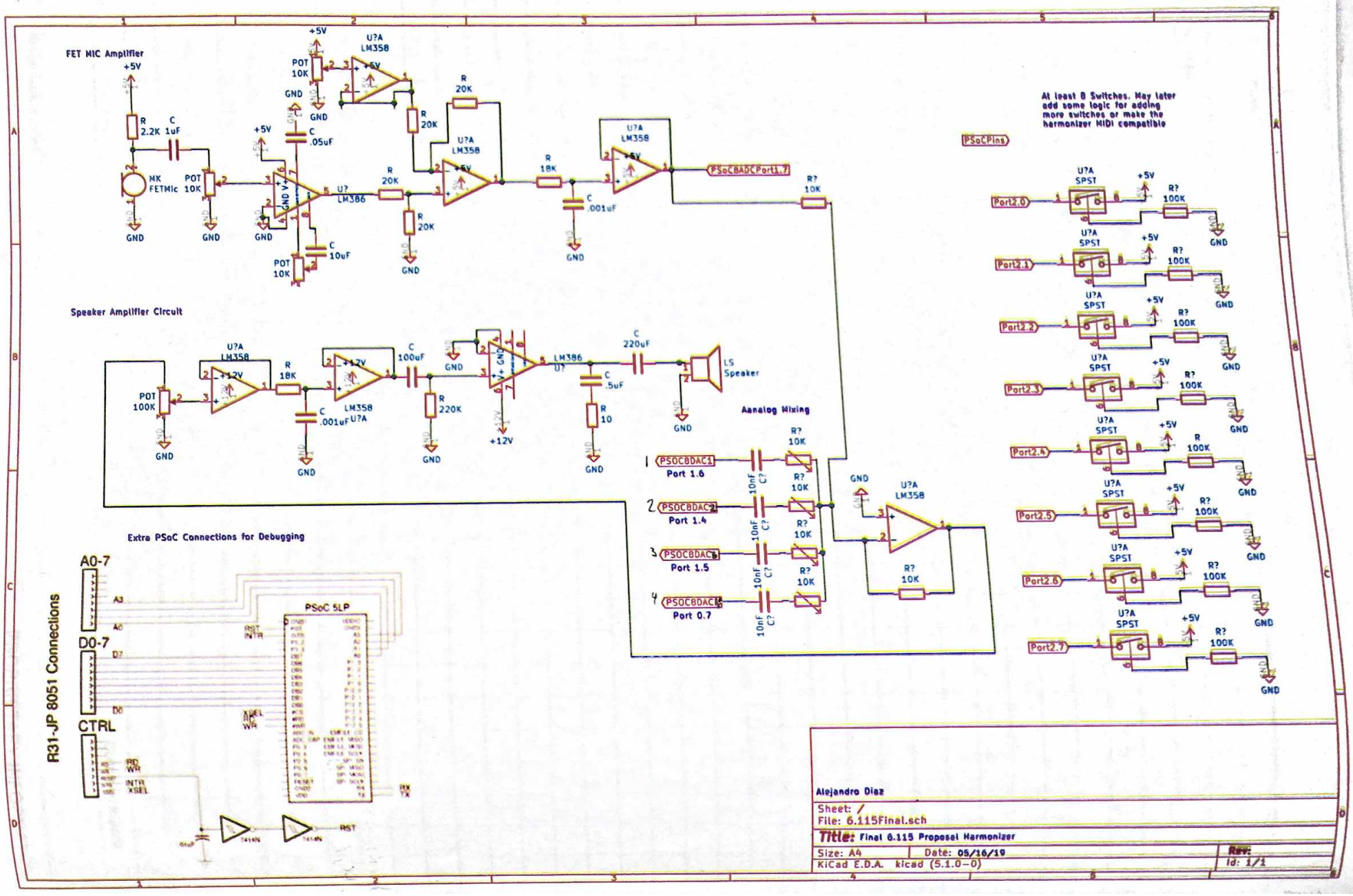
All outputted PSoC signals (up to 4) are then combined via a simple adder OP-AMP circuit and then fed into the speaker circuit.

The speaker circuit consists of an adjustable pot (for volume) & 2 LM3585 that filter high & low frequencies. The LM386 drives the speaker w/ a DC block in place.

Continued to page

SIGNATURE DATE

DISCLOSED TO AND UNDERSTOOD BY DATE


FET MIC Amplifier
Speaker Amplifier Circuit
Extra PSoC Connections for Debugging
Aanalog Mixing
At least 8 Switches. May later add some logic for adding more switches or make the harmonizer MIDI compatible
PSoCPins
PSoCBADCPort1.7
PSOC8DAC1
PSOC8DAC2
PSOC8DAC3
PSOC8DAC4
Port 1.6
Port 1.4
Port 1.5
Port 0.7
LS Speaker
PSoC 5LP
A0-7
D0-7
CTRL
R31-JP 8051 Connections
Alejandro Diaz
Sheet: /
File: 6.115Final.sch
Title: Final 6.115 Proposal Harmonizer
Size: A4
Date: 05/16/19
Rev:
KiCad E.D.A. kicad (5.1.0-0)
Id: 1/1

TITLE **PROJECT**

Continued from page

233Hz Bb on trumpet (start 4G) - [illegible] this Bb should be 233.08Hz

Software Description: The audio signal is read via an 8-bit ADC on the PSoC. ADC values are read @ a 32kHz sample speed via an interrupt that is triggered when the ADC is ready to be read. These values are stored in a 1024 uint8 array. In this same interrupt DAC values are sent out by using the calculated frequency to determine a step at which to run through the stored incoming signal.

In the main loop the data is copied into a new array and an FFT is taken using the function defined in FFT.c. Once the FFT is taken the frequency of the signal is calculated by finding the maximum magnitude of the real & imaginary #s returned from the FFT.

Then all the switches are polled and values of 1 correspond to being pressed. Based on the switches that are pressed different steps are generated to create different audio signals that are outputted in the interrupt for the ADC. DACs must be enabled and disabled in this step depending on the amount of switches pressed

EXERCISES: (1) FET MIC (2) Speaker Circuit
(3) PSoC to R31JP Serial printing (4) FFT on PSoC (5) generating new signals on PSoC (6) EXTRA changes & bugs

Continued to page

SIGNATURE DATE

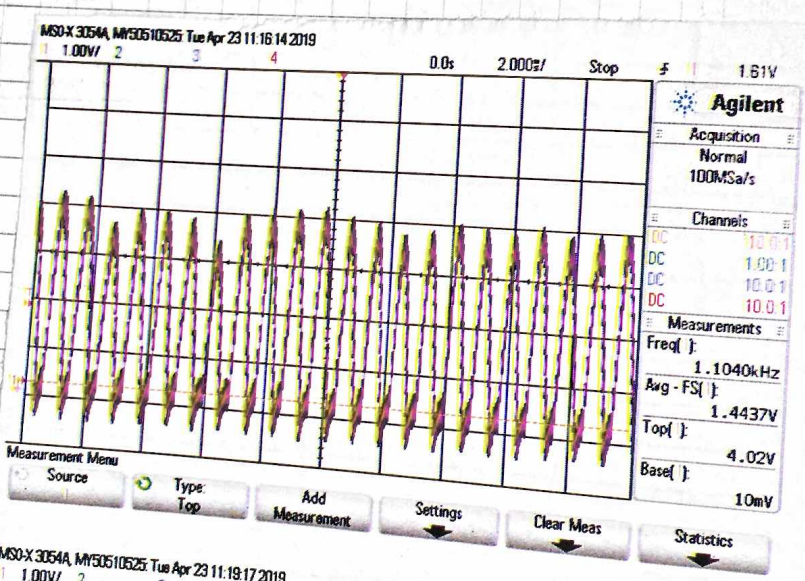
DISCLOSED TO AND UNDERSTOOD BY DATE

Continued from page

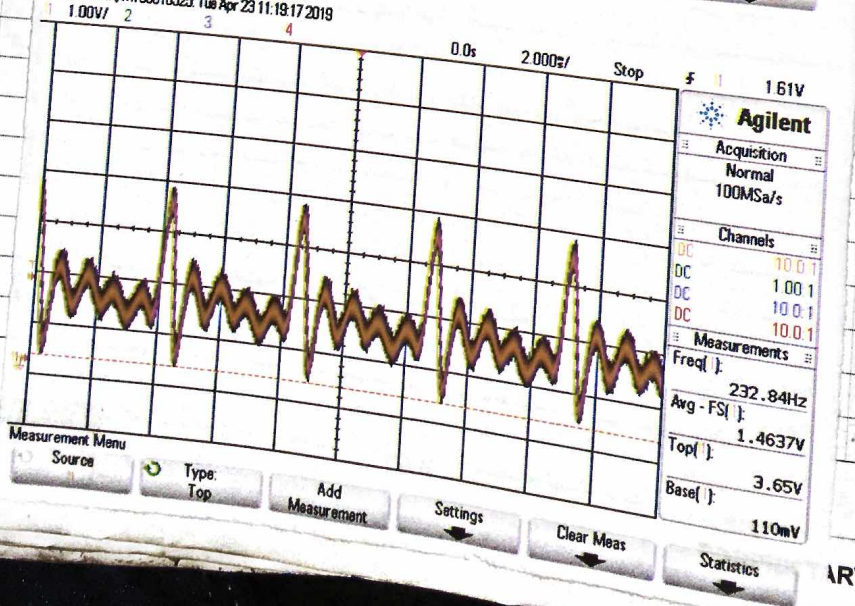
EXERCISE 1: FET Microphone Circuit

- wired the FET microphone circuit w/ LM386 & 3 LM358's, accidentally mixed a wire connection & spent an hour debugging
- scoped @ different places, but am unable to get a clean signal
- discovered a faulty connection to a 10k pot & was able to get a fairly clean output

11 kHz whistle measured using FET circuit (scope 45)



233 Hz Bb on trumpet (scope 46)



- For reference a Bb on trumpet should be 233.08Hz

Continued to page

SIGNATURE

DISCLOSED TO AND U

ARY INFORMATION

TITLE

Continued from

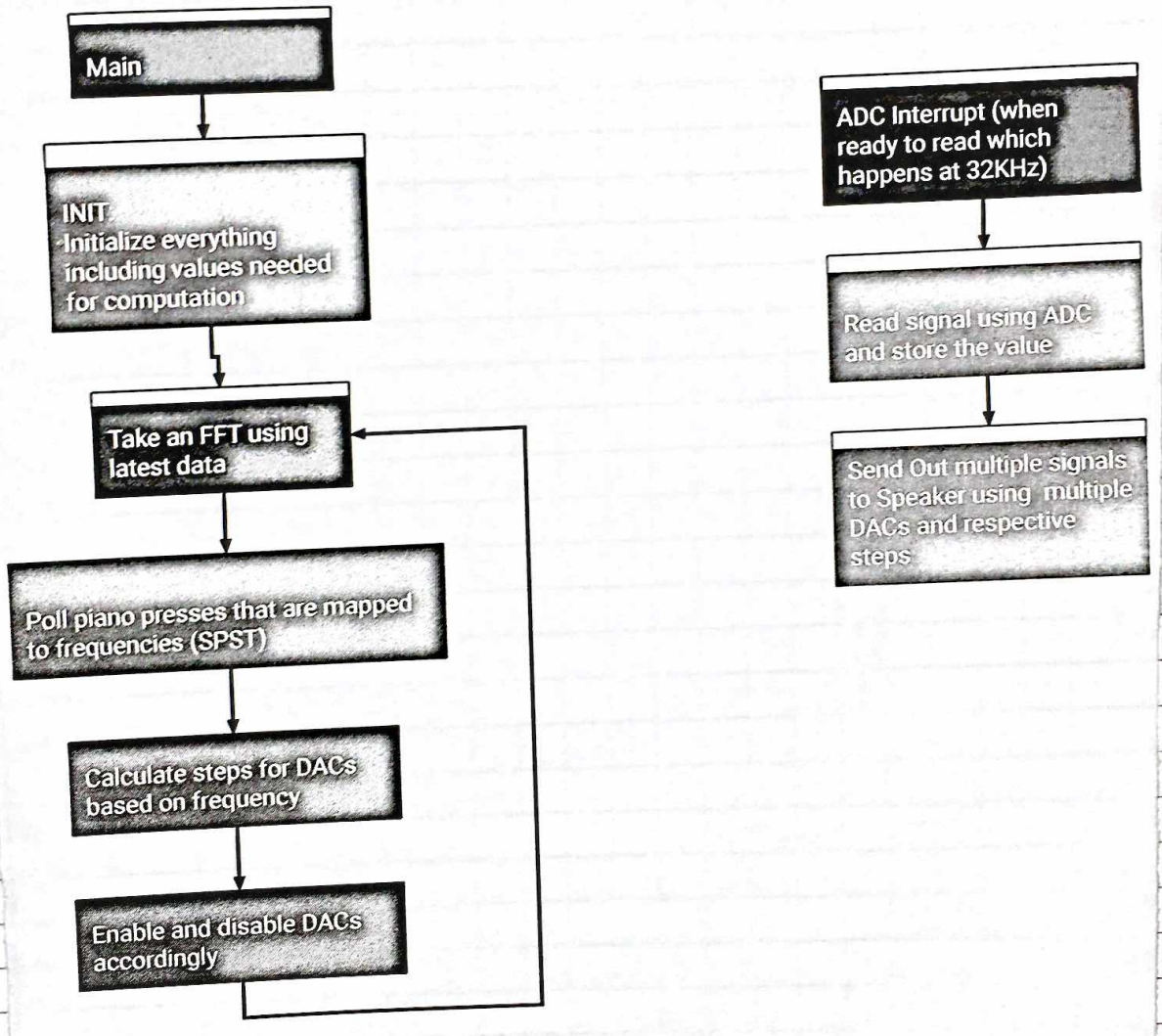
SIGNATURE

DISCLOSED TO AND

TITLE **PROJECT**

Continued from page

Soft Ware Flowchart



Continued to page

PROJECT

TITLE

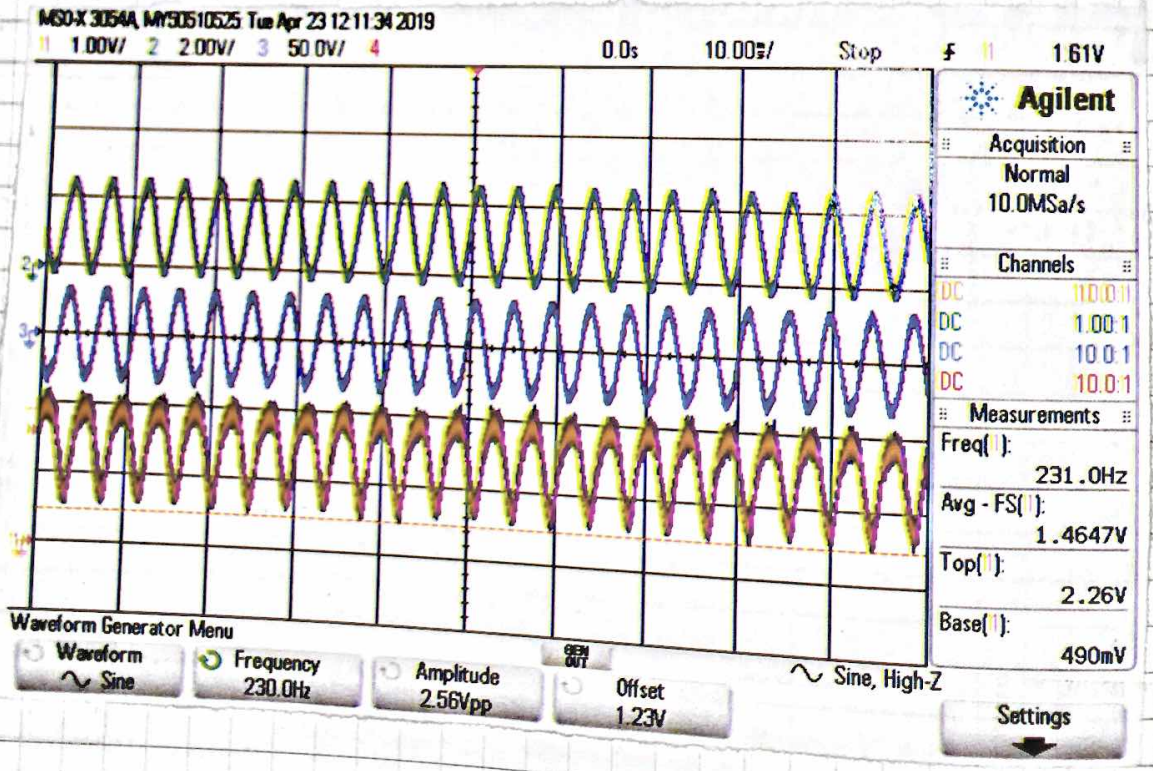
Continued from page

EXERCISE 2: Speaker Circuit

- as described earlier a series of amplifying and filtering steps using LM358 and LM386

- straightforward implementation using a pot to control volume

Ch 2 230Hz 2.56 Vpp Amplitude 1.23V offset (scope 47)
Ch 3 speaker output
Ch 1 mic input



SIGNATURE

Continued to page

DISCLOSED TO AND UNDERSTOOD BY

DATE

DATE

TITLE PROJECT

Continued from page

EXERCISE 3: Wiring the PSoC to R31JP for serial transmission (Debugging)

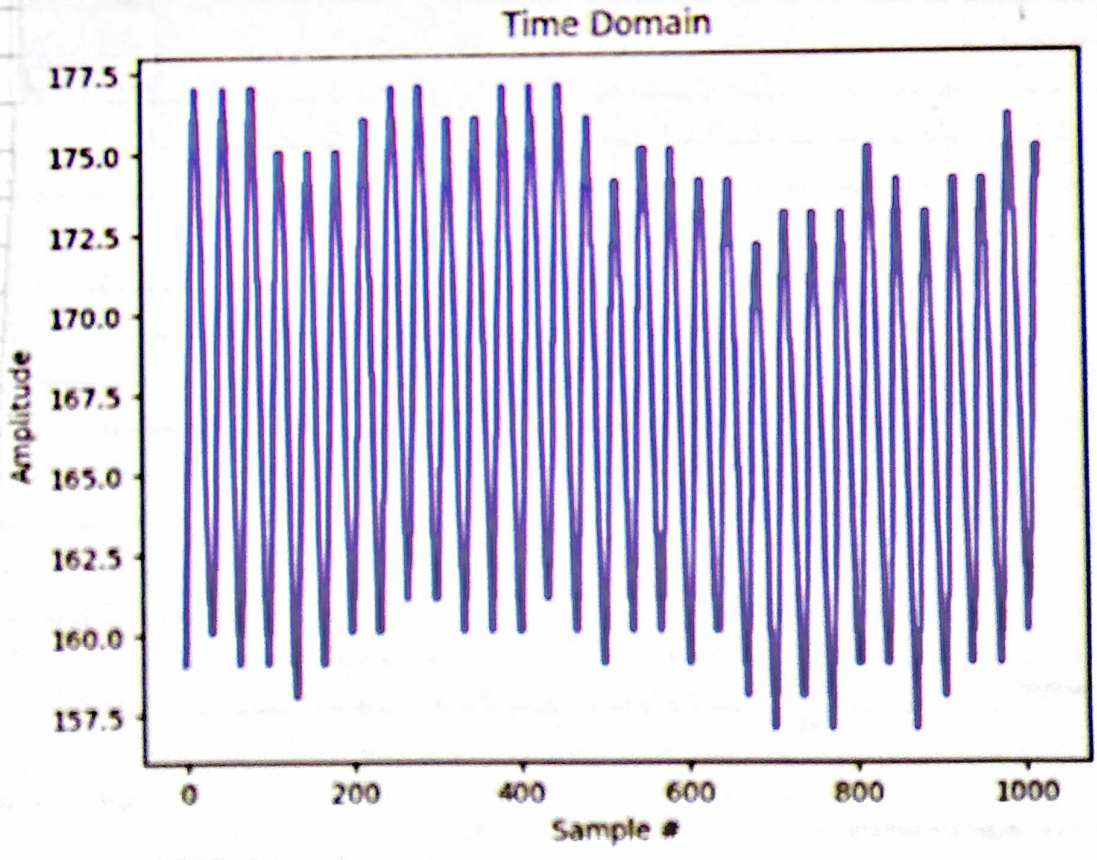
- Initially tried to use UART micro USB port, but even after following multiple tutorials & videos.

- Instead I decided to connect to the R31JP over serial using interrupts on the R31JP to read from the PSoC, code in attached Appendix

- had to add some delays to printing code on the PSoC in order to get proper transmission, I also wrote specific functions for the PSoC to print #'s and variables

- got the PSoC to print the values read in from the ADC

- took the values and graphed them in python to verify the signal



page

SIGNATURE

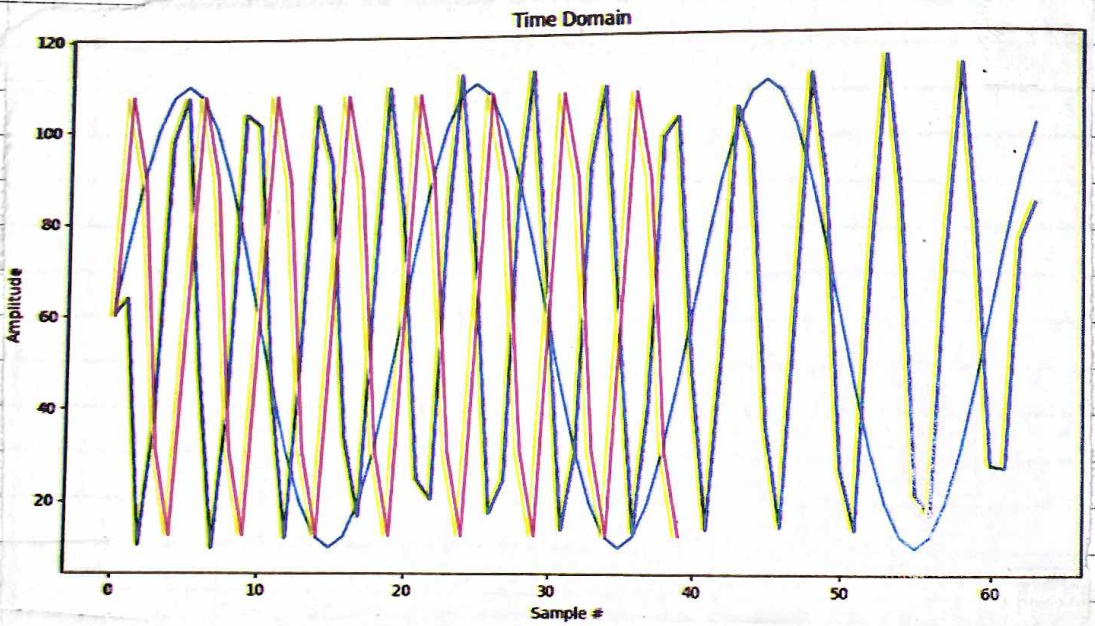
DISCLOSED TO AN

MATION

TITLE PROJECT

Continued from page

- began to see how to use FFT to shift my frequencies
- compared a shifted FFT to IFFT vs. a sped up original signal



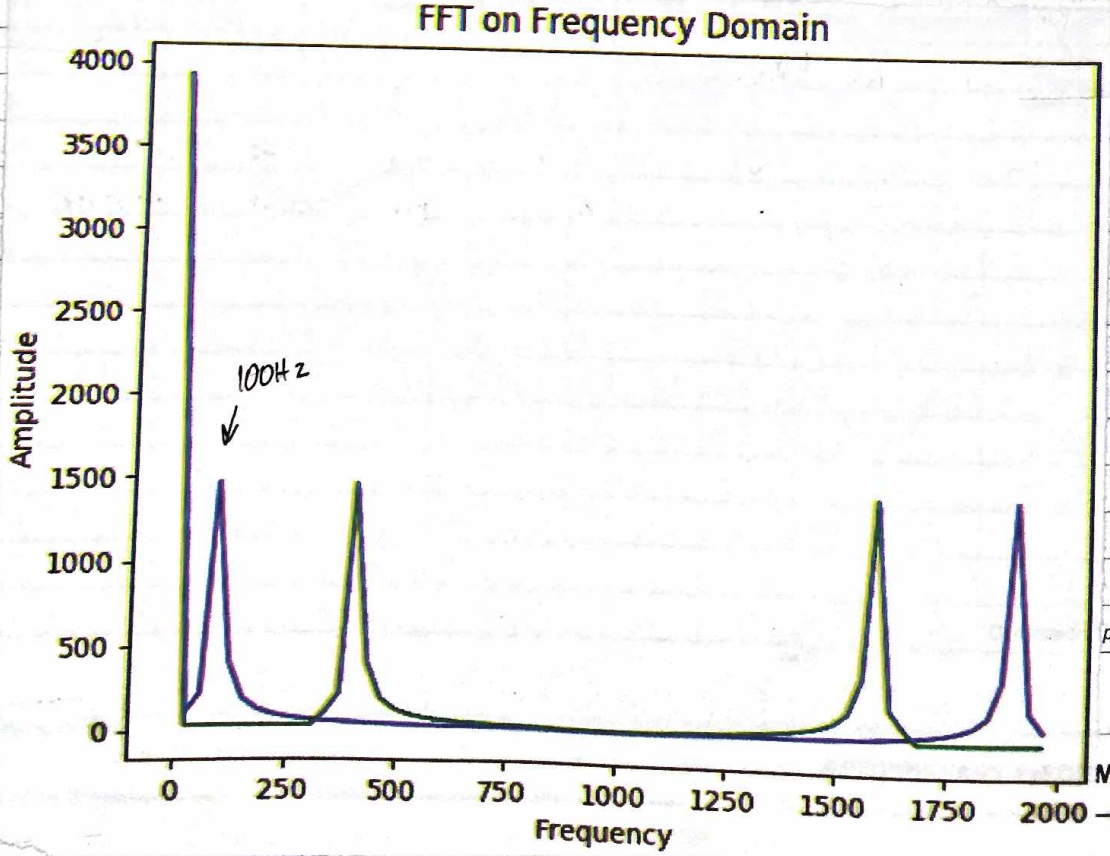
blue original generated 100Hz Sine wave

RED Original Signal sped up

Green iff of shifted FFT of orginal signal

BLUE FFT of Original signal

Green Purposefully shifted Blue FFT



SIGNATURE

DISCLOSED TO AND U

page

MATION

TITLE PROJECT

Continued from page

EXERCISE (4): Fast Fourier Transform on PSoC

- used an FFT from MIT
- first tested in python & then in C on my computer
- results are on the previous page
- initially my idea was to take the FFT and then shift the peaks in the frequency domain and go back to time domain using an IFFT, however limited computational power caused me to decide to simply calculate the frequency and then speed up the original signal accordingly

based on a 32kHz sample rate 1 sample takes .0000 3125 sec

want to capture frequencies between 50Hz to 4KHz
T=20ms T=.25ms

$\sim 30\text{ms frame} \rightarrow \frac{I}{1\text{ sample}} \rightarrow \frac{.030\text{sec}}{.0000125\text{ sec}} = 960 \text{ Samples} \approx 1024 \text{ samples}$

-decided to do a 1024 point FFT to capture frequencies between 50Hz to 4kHz

-had problems with type conversions and header files
-still had errors did not a problem with storage of sine & cosine tables and dynamic allocation using malloc
-circumvented this by calculating sine and cos values each time
-eventually got the FFT to function and print values to screen using print function from last exercise

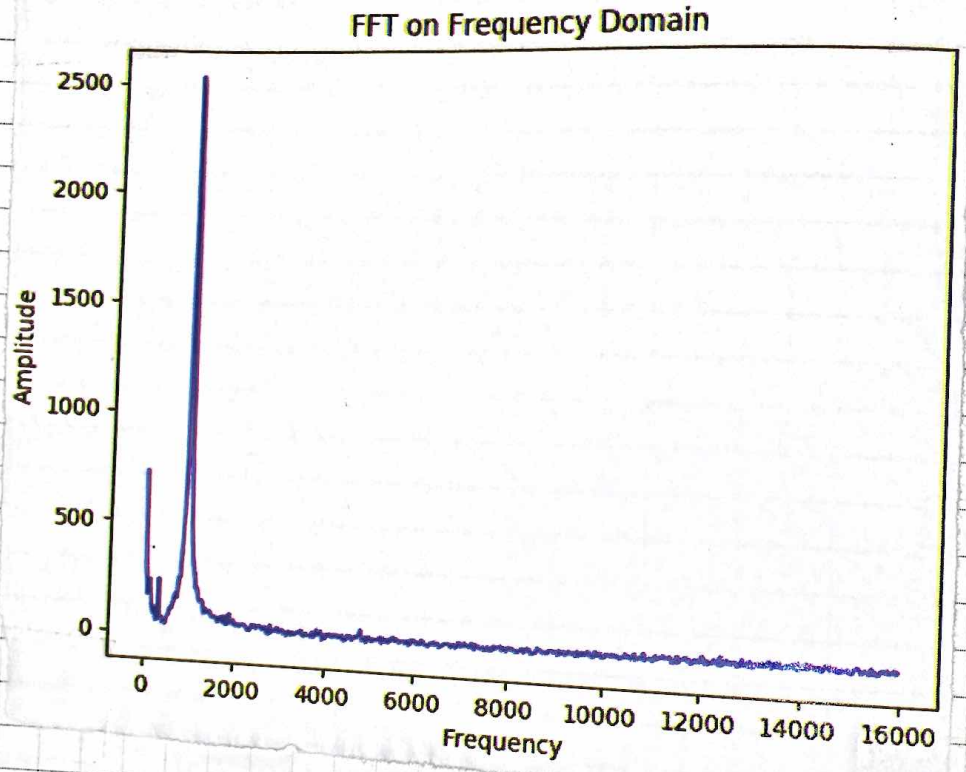
Continued to page

SIGNATURE DATE

DISCLOSED TO AND UNDERSTOOD BY DATE

TITLE PROJECT

Continued from page



- FFT of actual 937 Hz signal recorded by my mic circuit (FFT of signal on page 59)
- correctly identifies the frequency

SIGNATURE

DISCLOSED TO AND UNDERSTOOD BY DATE

DATE

Continued to page

TITLE PROJECT

Continued from page

EXERCISE (5): Generating new signals using PSoC and frequency & incoming audio

- added in the switch circuit for 8 SPST switches pulled down to ground that read high when pressed (attached to PSoC pins

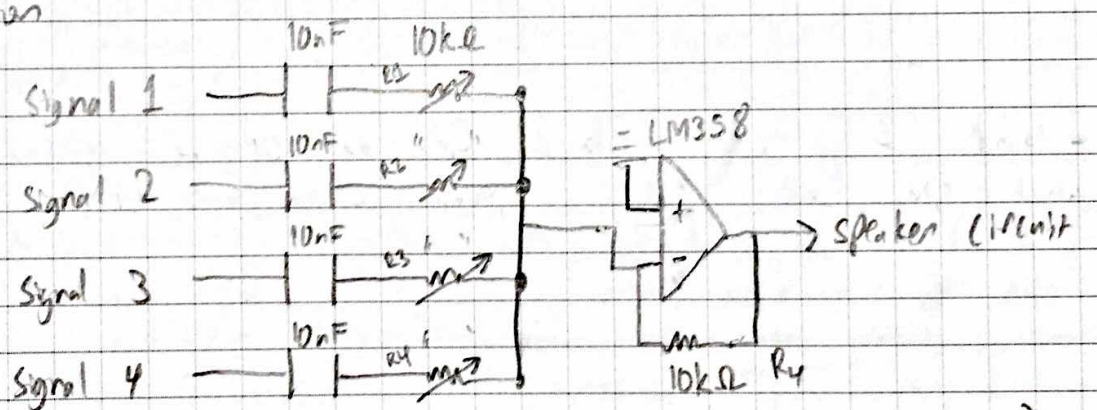
- based on the calculated frequency & the button pressed a step is calculated like so:

Step = desired frequency / input frequency

- up to 4 keys can be pressed and the steps are assigned to up to 4 8-bit DACs

~~Had~~ had trouble here because I wired 3 of the DACs to ports 0.2, 0.3, 0.4 which all have capacitors and were filtering out the AC signal (took hours to debug)

- these signals are then sent out & mixed via RC series connections that feed into an adder OP-AMP configuration



ignoring capacitance $V_{OUT} = -R_4\left(\frac{V_1}{R_1} + \frac{V_2}{R_2} + \frac{V_3}{R_3} + \frac{V_4}{R_4}\right)$

Continued to page

SIGNATURE DATE

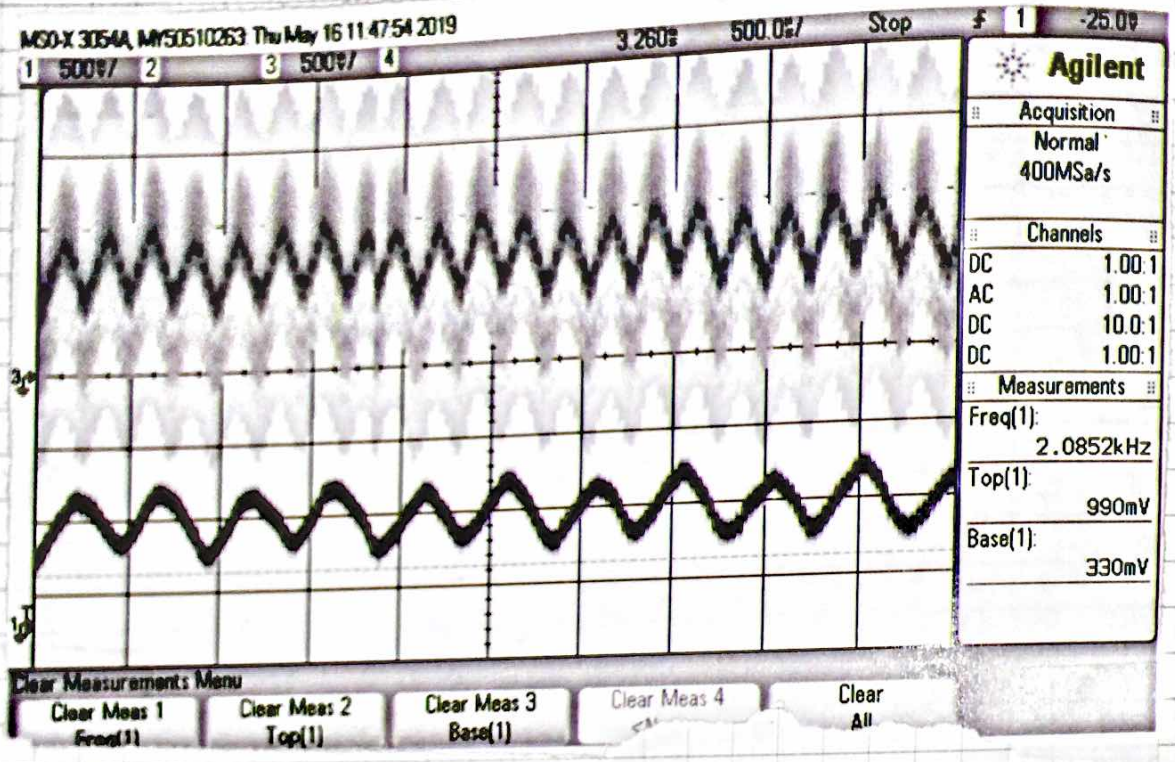
DISCLOSED TO AND UNDERSTOOD BY DATE

TITLE

PROJECT

Continued from page

2 wires on different channels an octave apart



- played 2 notes C6 & C7 an octave apart which gave me the appropriate frequencies of 2.0852kHz & double the frequency on channel 3

Continued to page

SIGNATURE DATE

DISCLOSED TO AND UNDERSTOOD BY DATE

TITLE

PROJECT

Continued from page,

EXERCISE 6: Even more changes & bugs

- the 1024 point fourier transform was really slow and I realized I only needed to capture frequencies between 50Hz & 2kHz so I downsampled the 1024 array to 128 point array for an 128 point FFT that is significantly faster.

- I would have liked better frequency granularity than 31.25Hz (32,000/1,024), but I would have to expand my window and therefore the FFT which seemed a suboptimal idea

- I would have liked to sample @ a higher sample rate like 64,000, but this would have been too computationally expensive due to the extra interrupts and larger FFT

- added functionality for automatic harmony by simply scaling the incoming frequency up by half steps. Added a switch to toggle this mode

Conclusion:

- Was able to take in signals in real time and output them @ different frequencies according to the key presses. However due to the limitations of the PSoC computation power/speed I was unable to get the clarity of of audio waves that I would have liked.

- Likely next steps would be utilizing a more powerful computer or 2 @ the same time. Also it would be worthwhile to upgrade the mic, speaker, DACs, and ADC

Continued to page

SIGNATURE DATE

DISCLOSED TO AND UNDERSTOOD BY DATE