

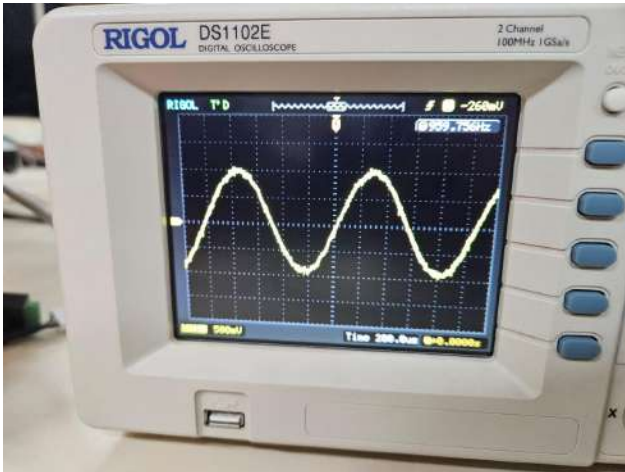
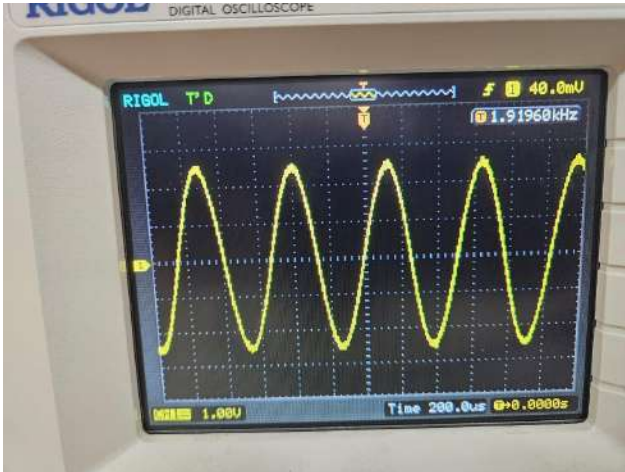
Lab-1

Raj-202201403, Bhoomish-202201414

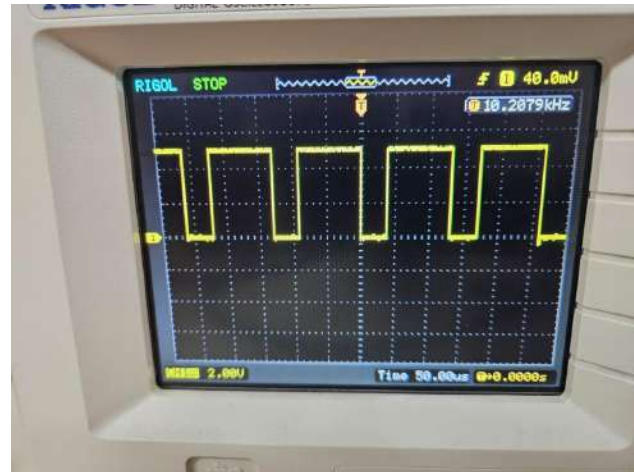
Manthan-202201416, Rakshit-202201426

Experiment 1:

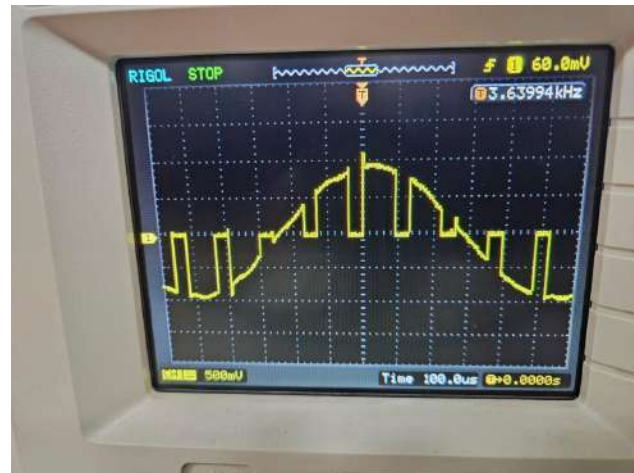
Experiment 1 (DCL 01)

Signal	Output
1kHz input	
2kHz input	

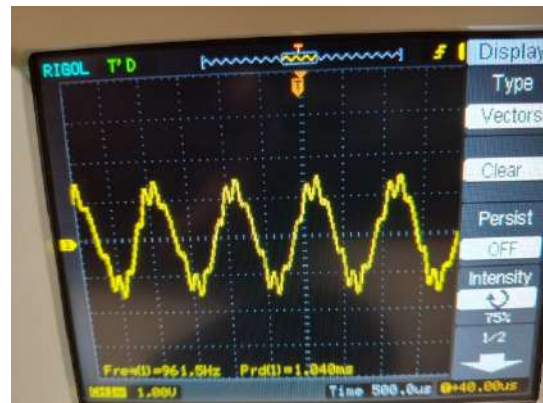
8kHz clock



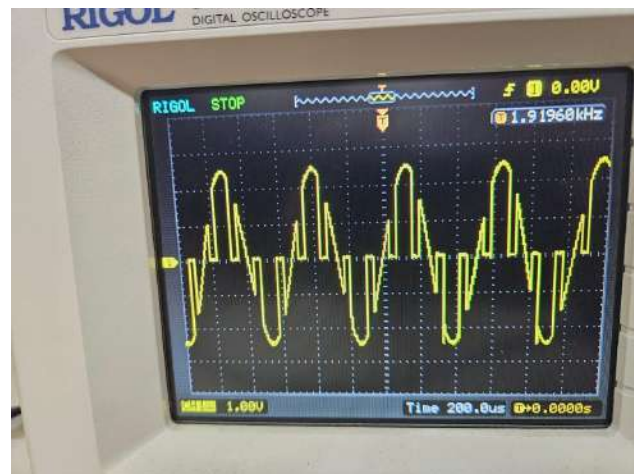
1kHz natural sampling



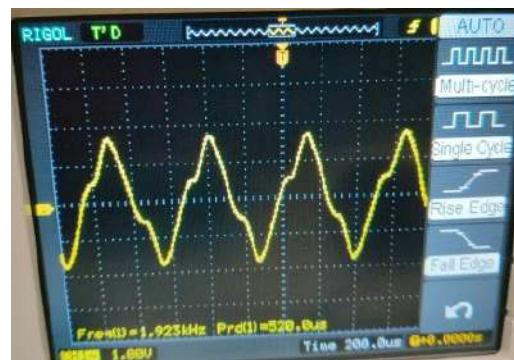
1kHz natural sampled output



2kHz natural sampling



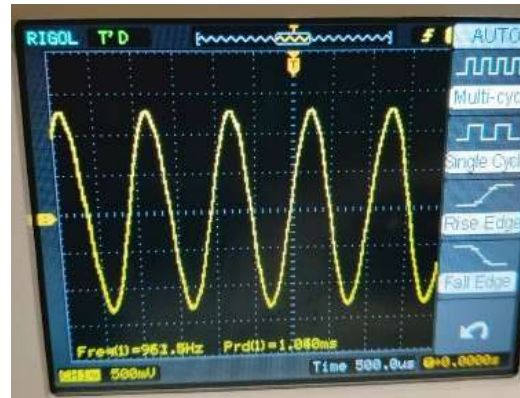
2kHz natural sampled output



1kHz sample and hold sampling



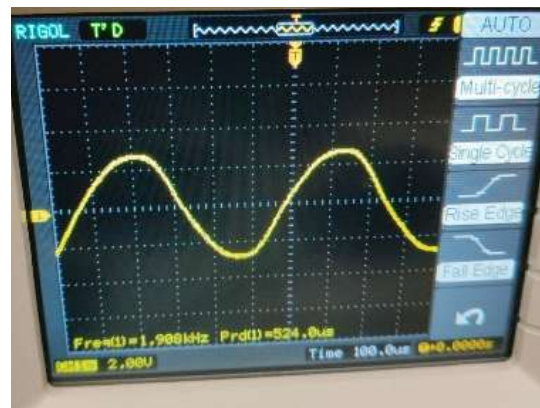
1kHz sample and hold sampled output



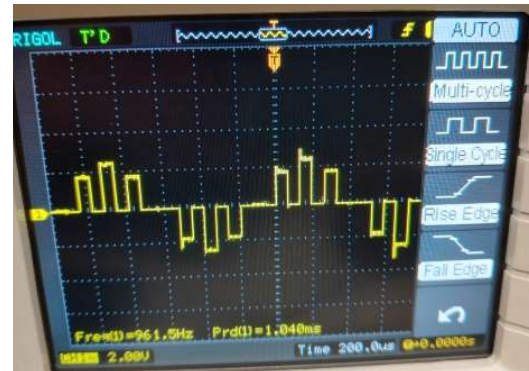
2kHz sample and hold sampling



2kHz sample and hold sampled output



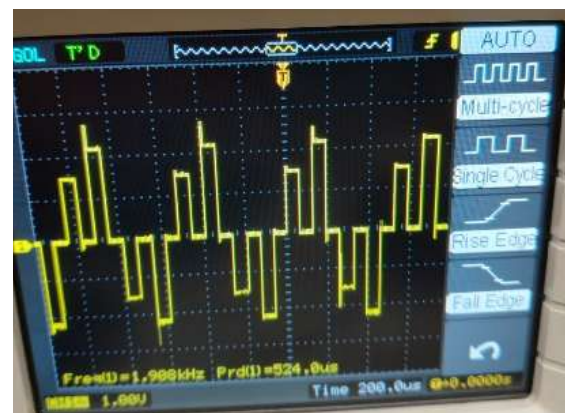
1kHz flat top sampling



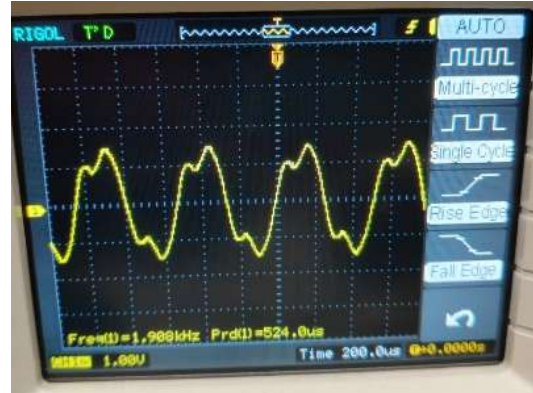
1kHz flat top sampled output



2kHz flat top sampling



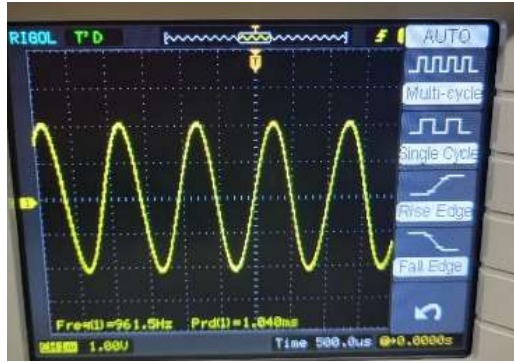
2kHz flat top sampled output



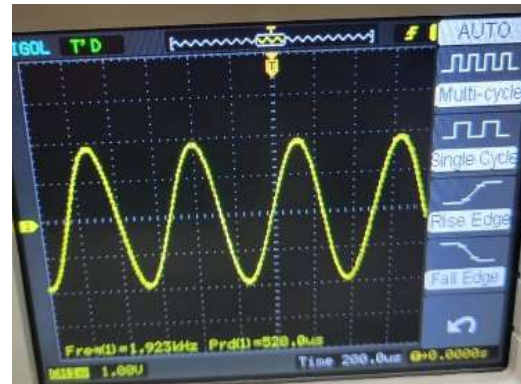
Conclusion:

- Through this experiment, we learned how different types of signal sampling works.
- We also observed how the different analog input signals are plotted in the CRO.
- We also observed the reconstructed signals which are created after different sampling methods.

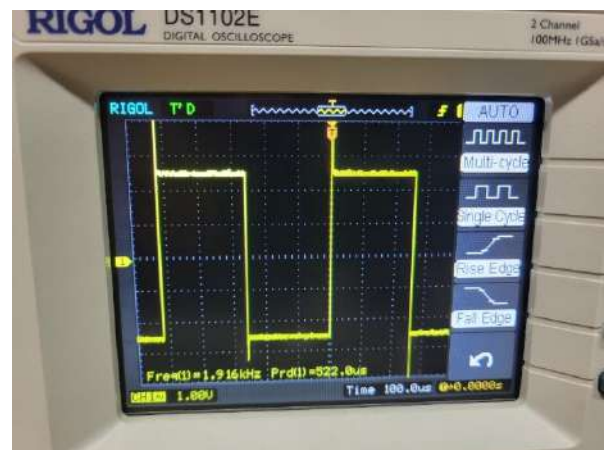
Experiment 2 (DCL 01)

Signal	Output
1kHz input signal	

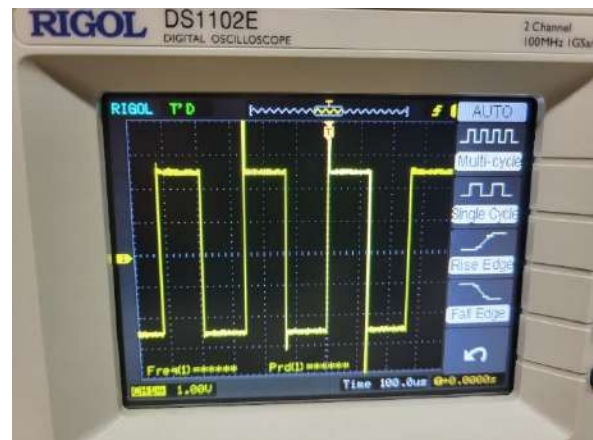
2kHz input signal



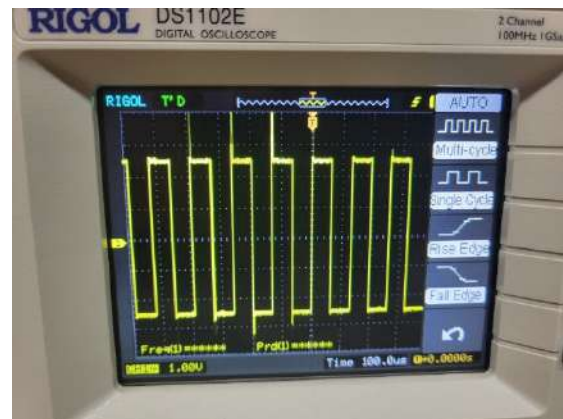
2kHz sampling clock



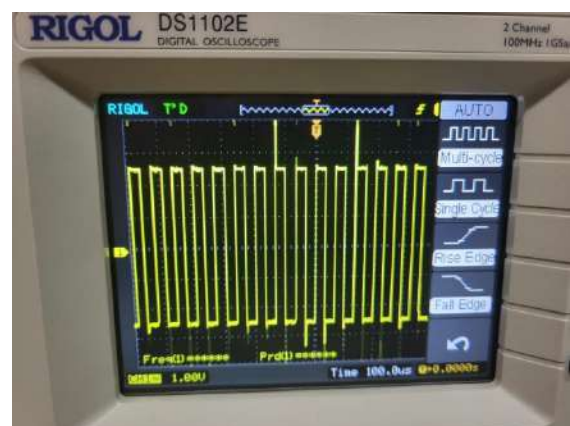
4kHz sampling clock



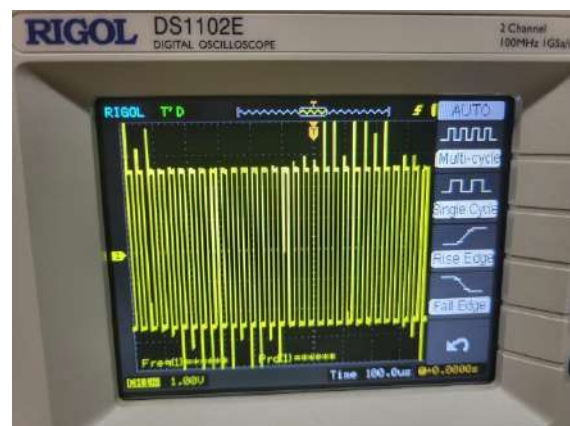
8kHz sampling clock



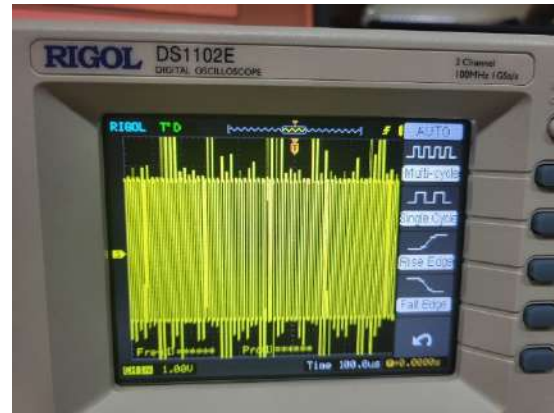
16kHz sampling clock



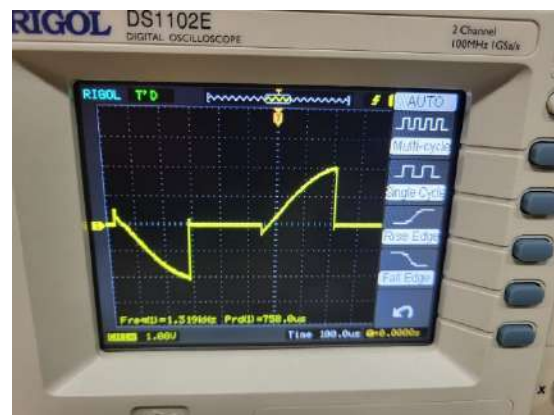
32kHz sampling clock



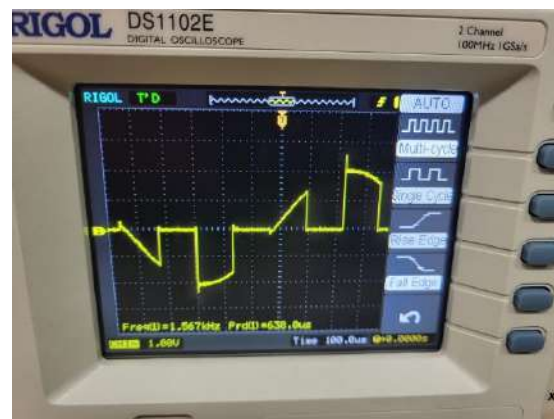
64kHz sampling clock



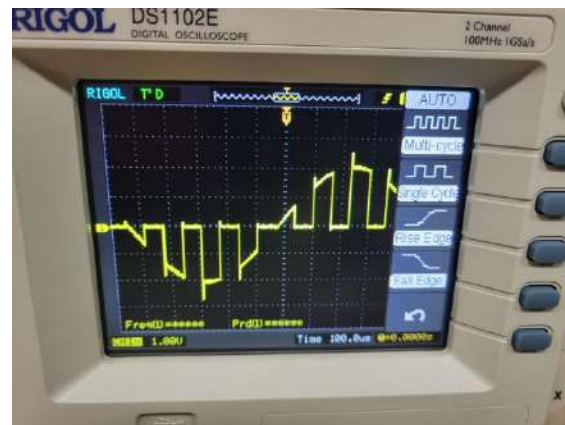
1kHz Natural Sampling (2kHz)



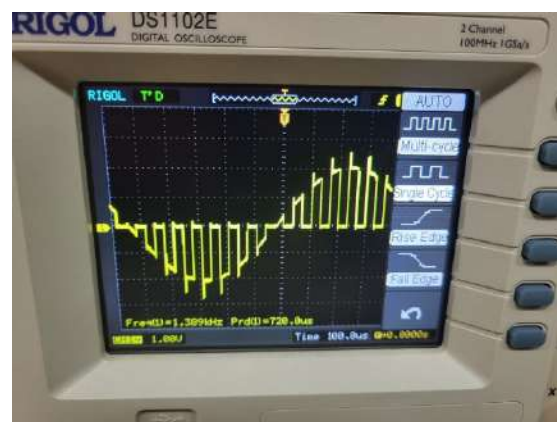
1kHz Natural Sampling (4kHz)



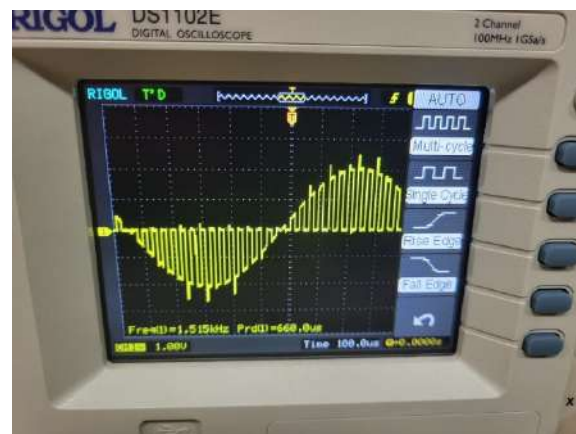
1kHz Natural Sampling (8kHz)



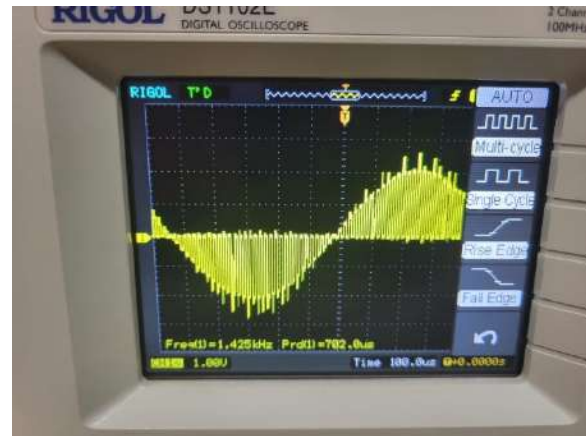
1kHz Natural Sampling (16kHz)



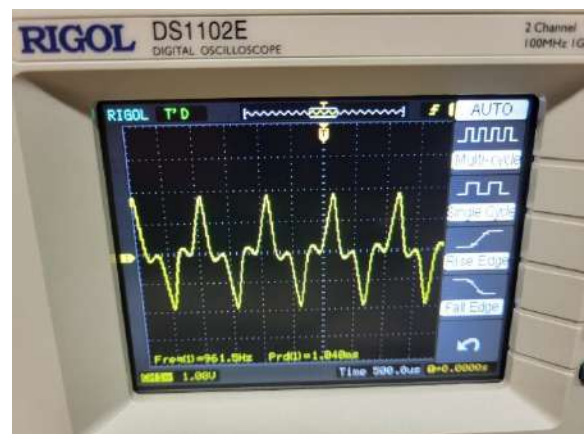
1kHz Natural Sampling (32kHz)



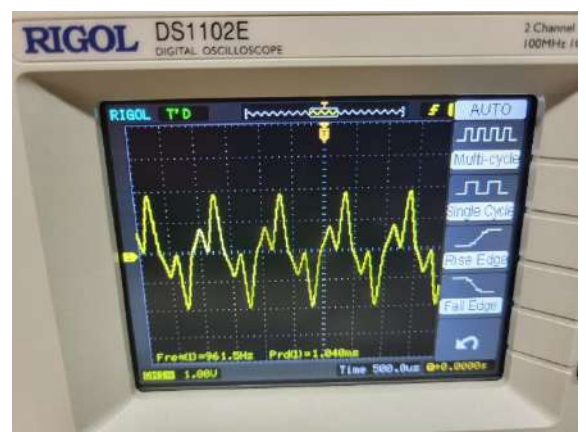
1kHz Natural Sampling (64kHz)



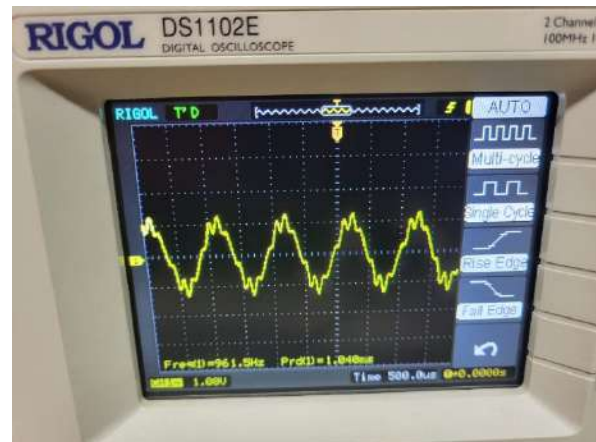
1kHz Natural Sampling Output (2kHz)



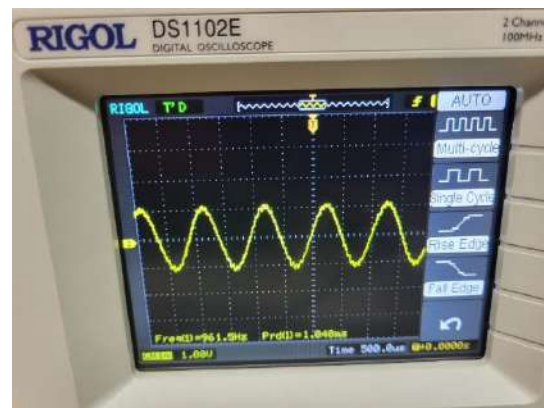
1kHz Natural Sampling Output (4kHz)



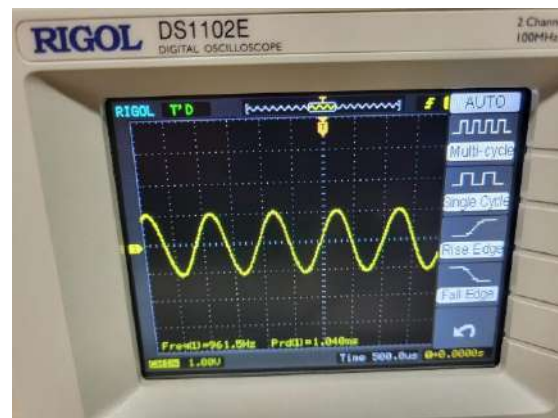
1kHz Natural Sampling Output (8kHz)



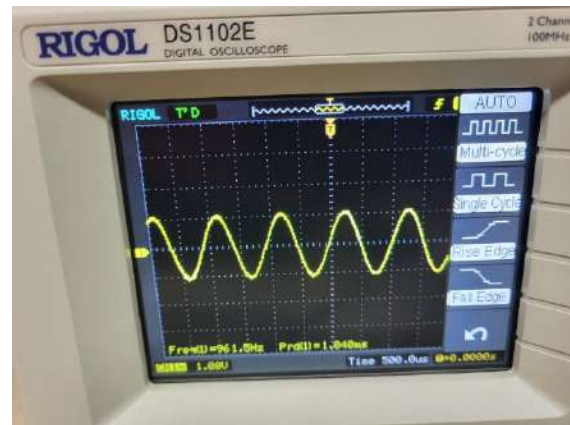
1kHz Natural Sampling Output (16kHz)



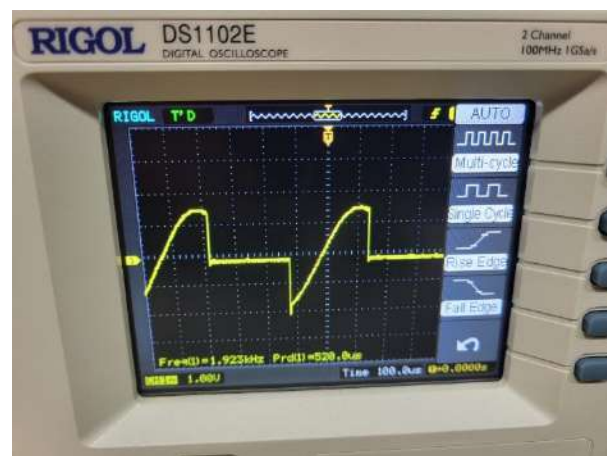
1kHz Natural Sampling Output (32kHz)



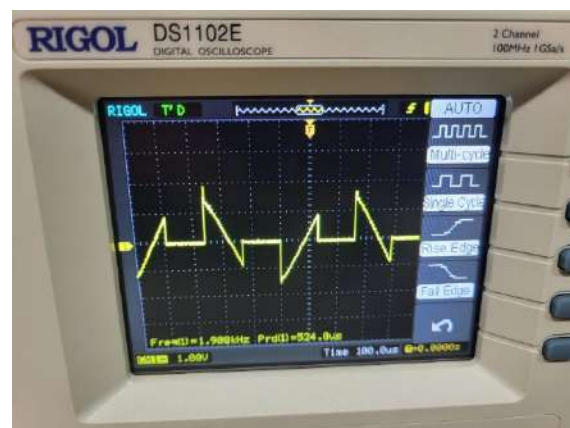
1kHz Natural Sampling Output
(64kHz)



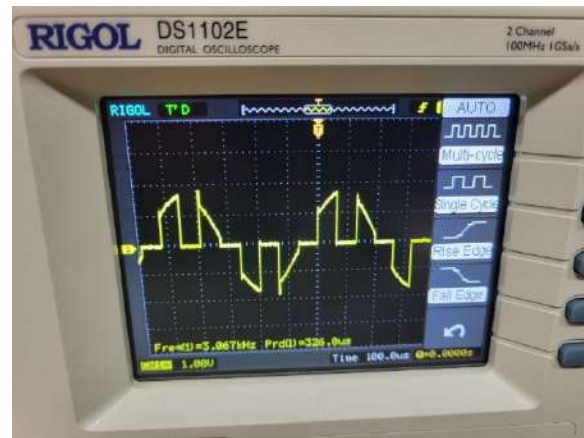
2kHz Natural Sampling (2kHz)



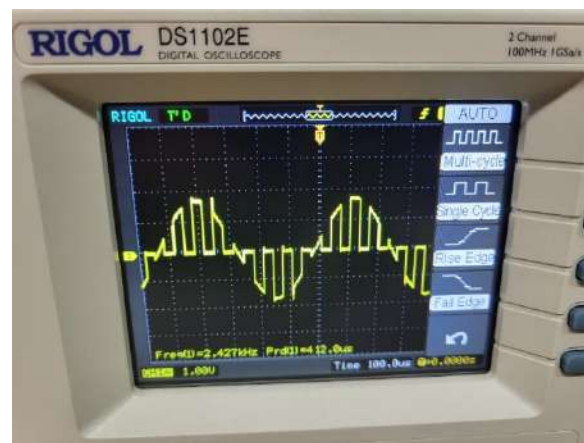
2kHz Natural Sampling (4kHz)



2kHz Natural Sampling (8kHz)



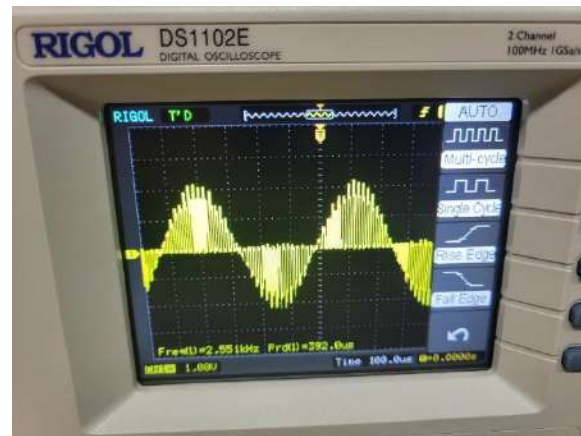
2kHz Natural Sampling (16kHz)



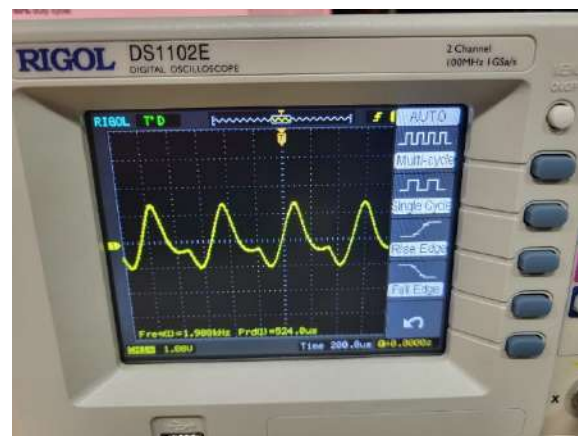
2kHz Natural Sampling (32kHz)



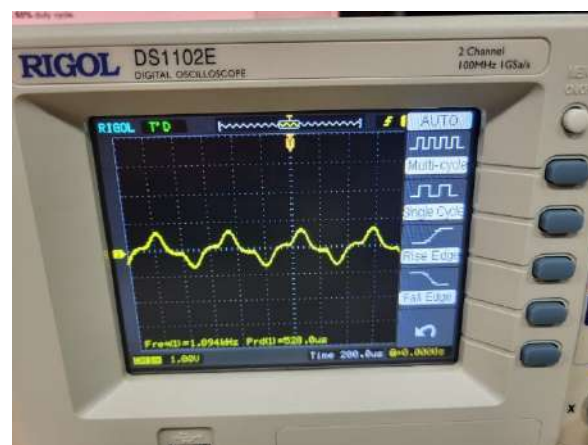
2kHz Natural Sampling (64kHz)



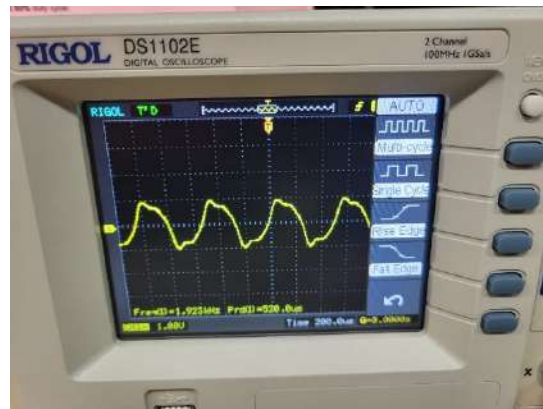
2kHz Natural Sampling Output (2kHz)



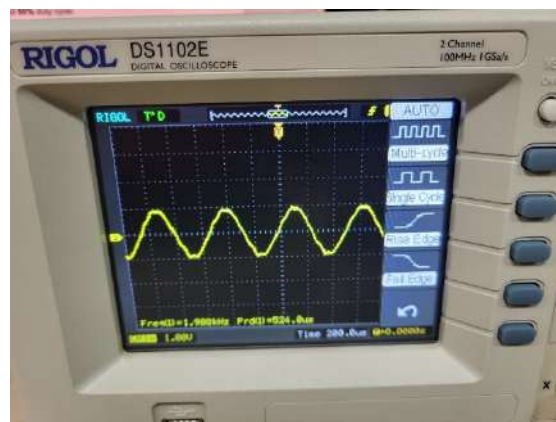
2kHz Natural Sampling Output (4kHz)



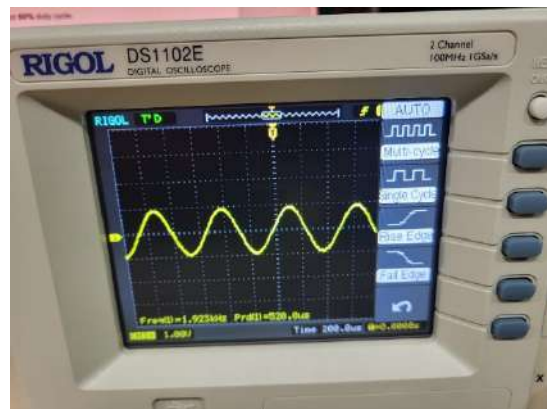
2kHz Natural Sampling Output (8kHz)



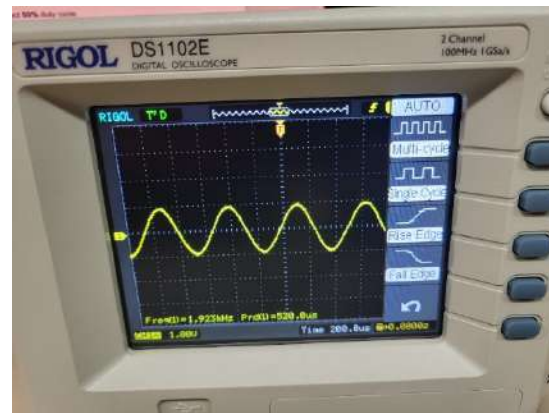
2kHz Natural Sampling Output (16kHz)



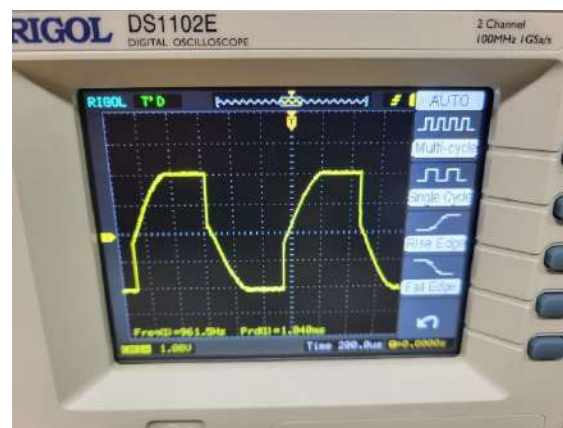
2kHz Natural Sampling Output (32kHz)



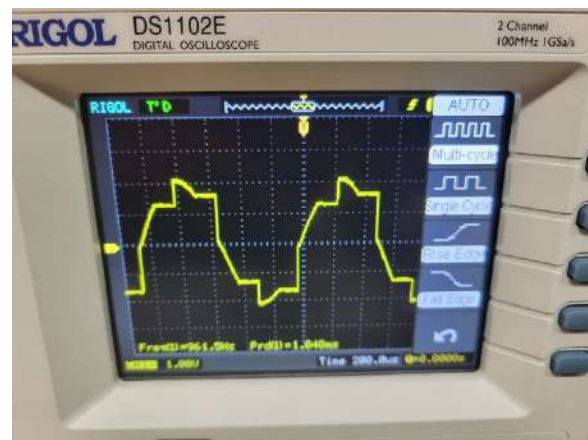
2kHz Natural Sampling Output
(64kHz)



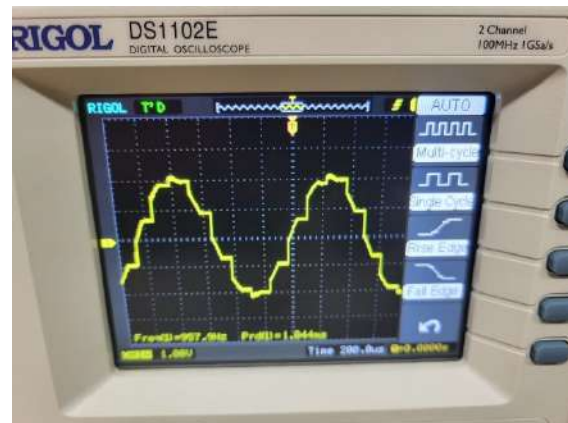
1kHz Sample and Hold Sampling
(2kHz)



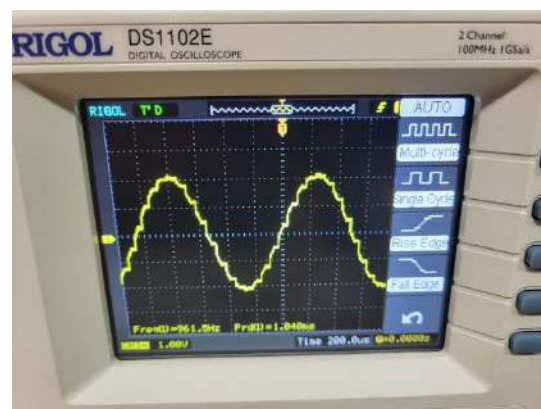
1kHz Sample and Hold Sampling
(4kHz)



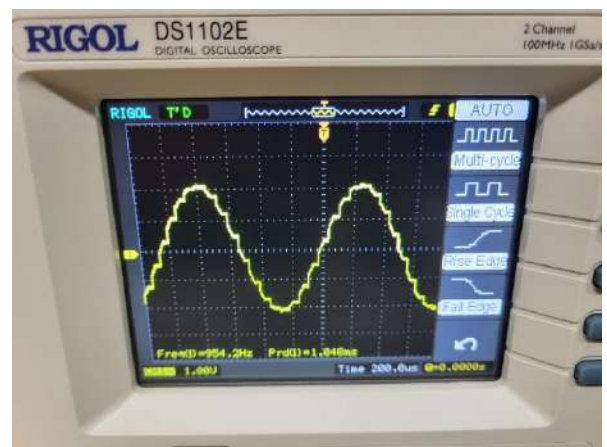
1kHz Sample and Hold Sampling
(8kHz)



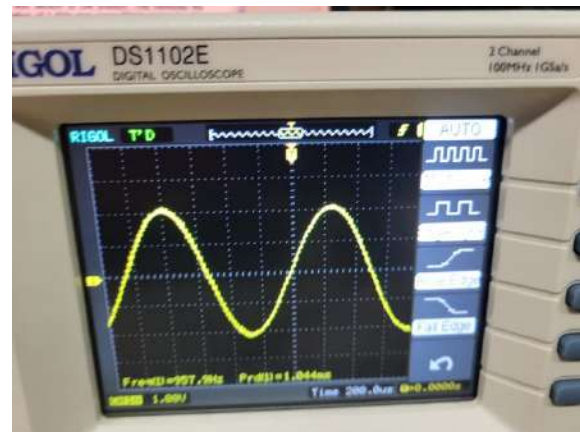
1kHz Sample and Hold Sampling
(16kHz)



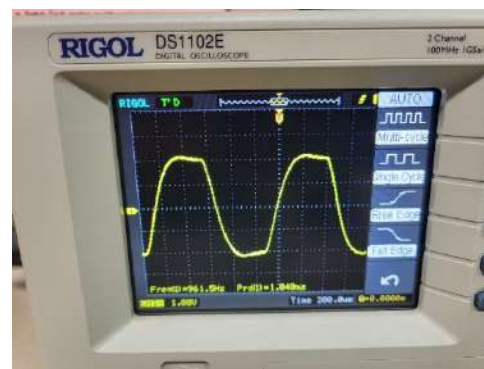
1kHz Sample and Hold Sampling
(32kHz)



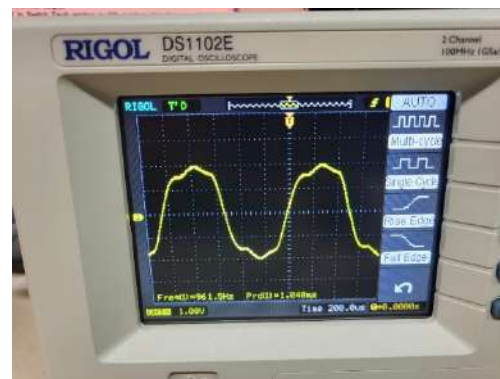
1kHz Sample and Hold Sampling
(64kHz)



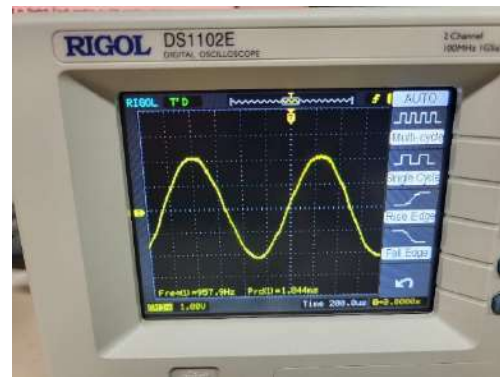
1kHz Sample and Hold Sampling
Output (2kHz)



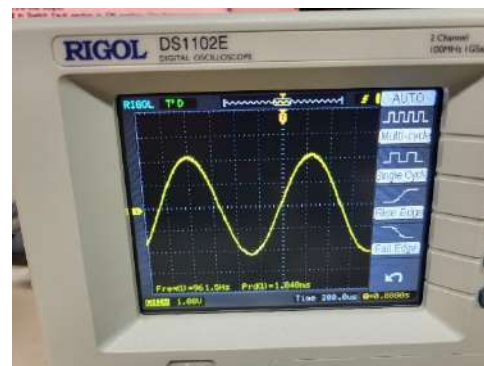
1kHz Sample and Hold Sampling
Output (4kHz)



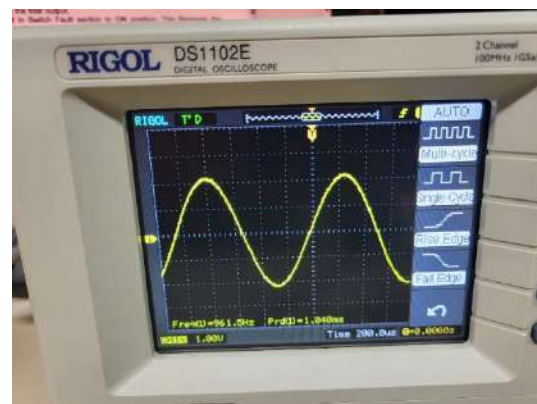
1kHz Sample and Hold Sampling
Output (8kHz)



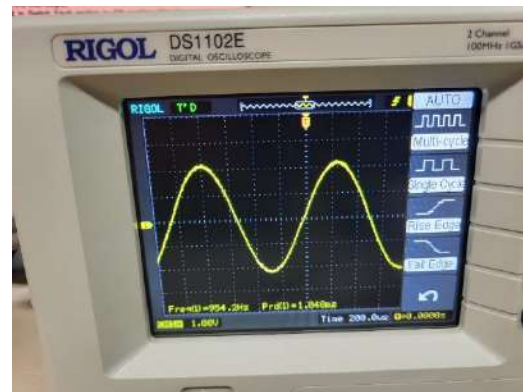
1kHz Sample and Hold Sampling
Output (16kHz)



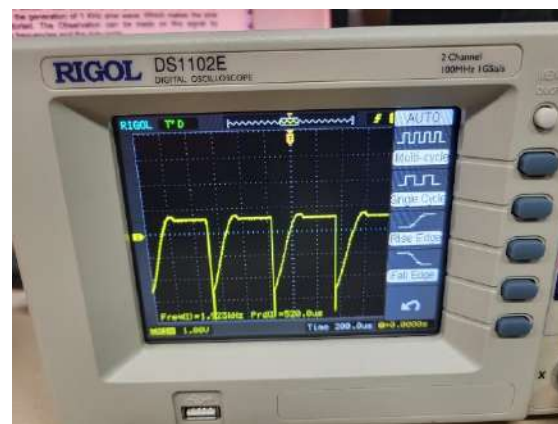
1kHz Sample and Hold Sampling
Output (32kHz)



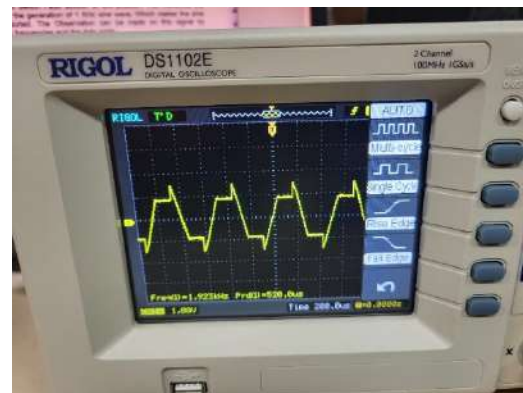
1kHz Sample and Hold Sampling Output (64kHz)



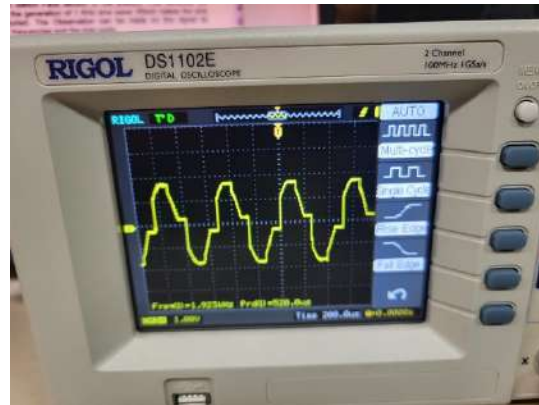
2kHz Sample and Hold Sampling (2kHz)



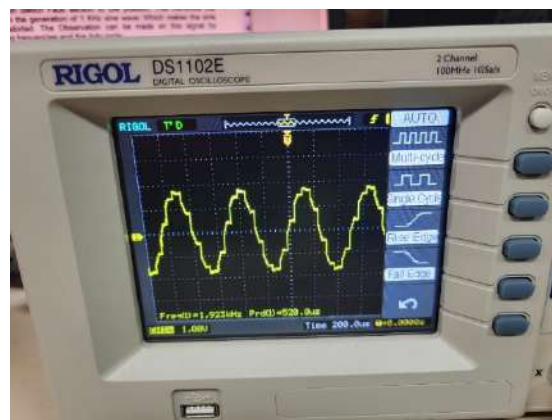
2kHz Sample and Hold Sampling (4kHz)



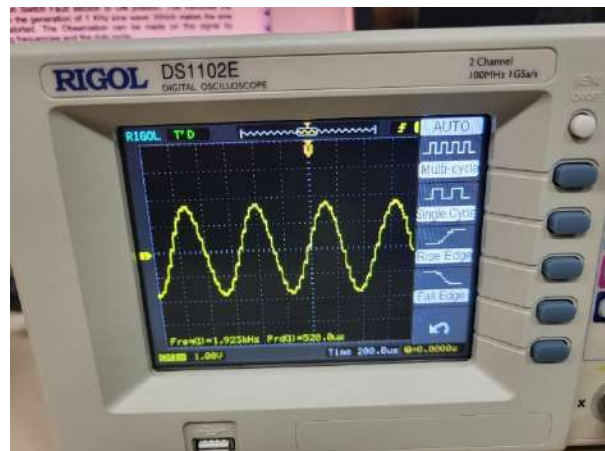
2kHz Sample and Hold Sampling
(8kHz)



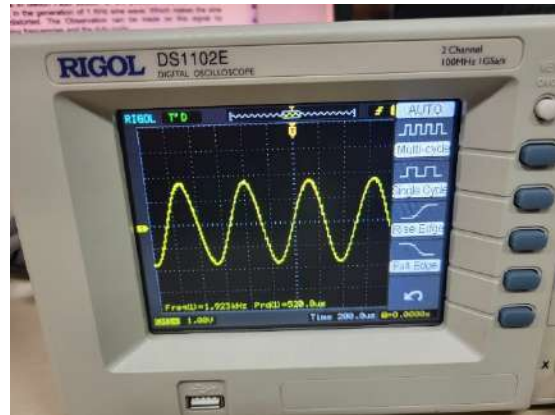
2kHz Sample and Hold Sampling
(16kHz)



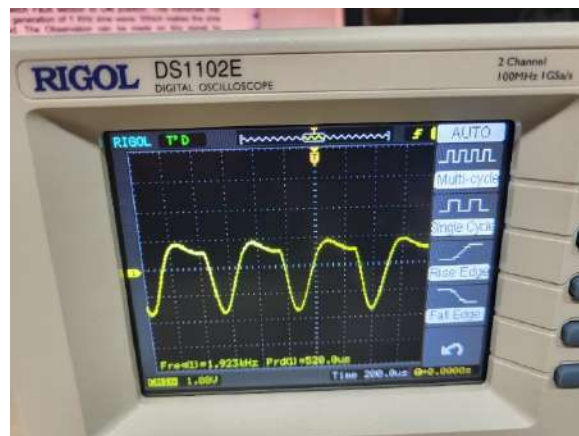
2kHz Sample and Hold Sampling
(32kHz)



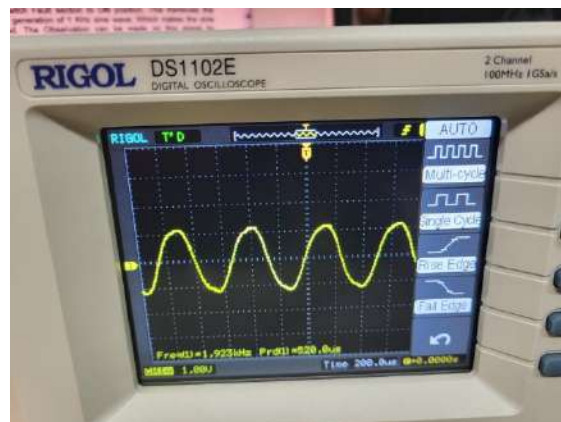
2kHz Sample and Hold Sampling
(64kHz)



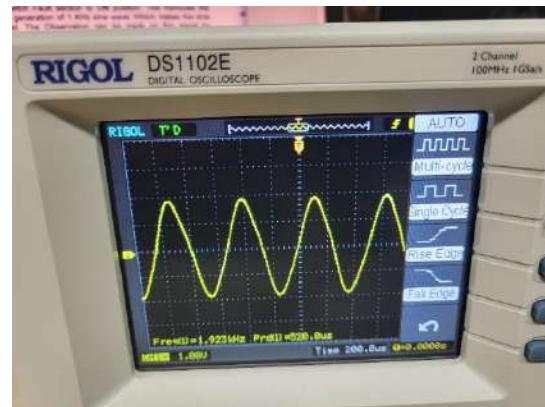
2kHz Sample and Hold Sampling
Output (2kHz)



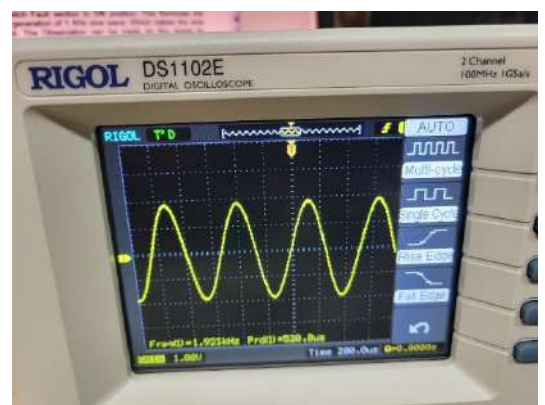
2kHz Sample and Hold Sampling
Output (4kHz)



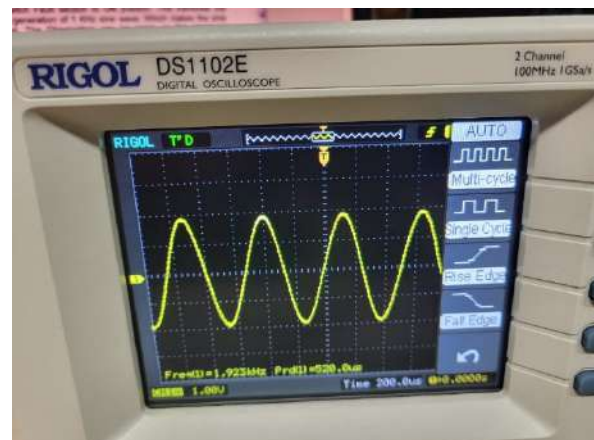
2kHz Sample and Hold Sampling
Output (8kHz)



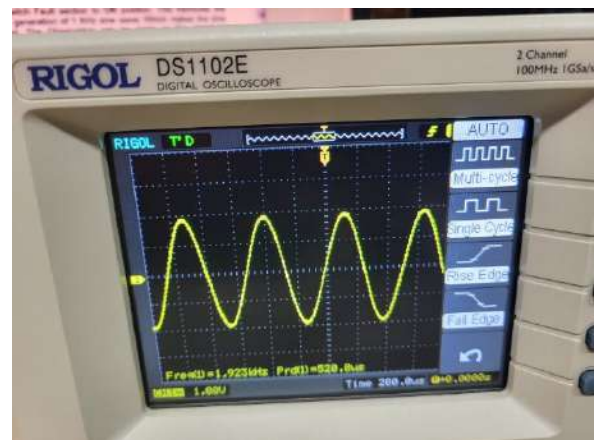
2kHz Sample and Hold Sampling
Output (16kHz)



2kHz Sample and Hold Sampling
Output (32kHz)



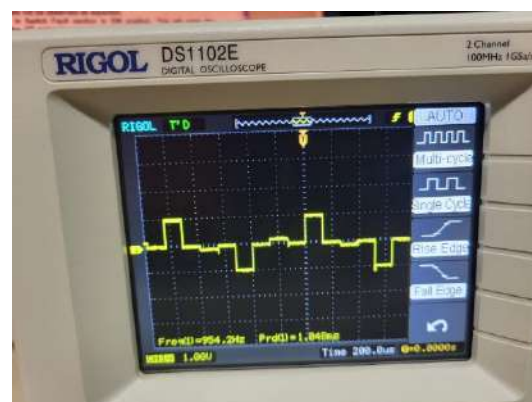
2kHz Sample and Hold Sampling Output (64kHz)



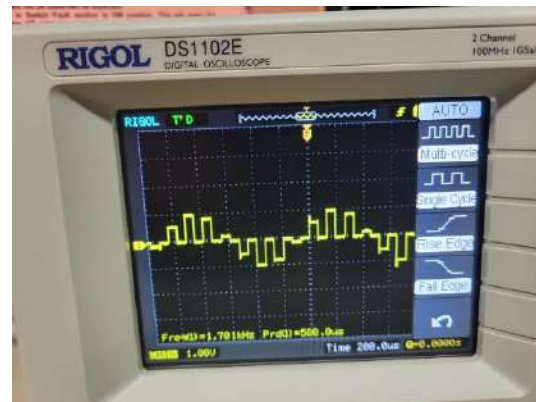
1kHz Flat Top Sampling (2kHz)



1kHz Flat Top Sampling (4kHz)



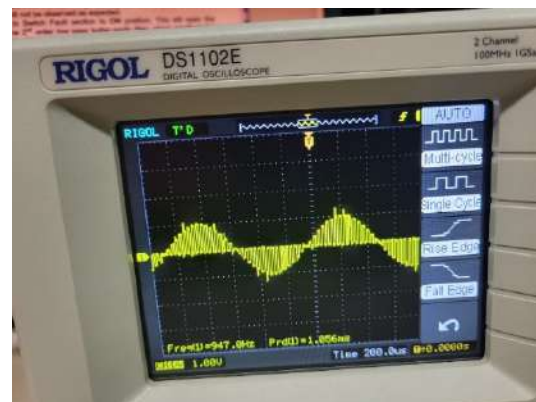
1kHz Flat Top Sampling (8kHz)



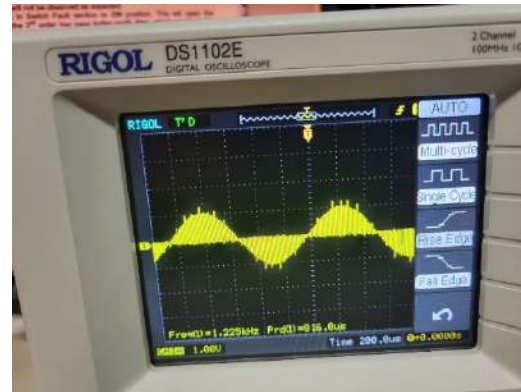
1kHz Flat Top Sampling (16kHz)



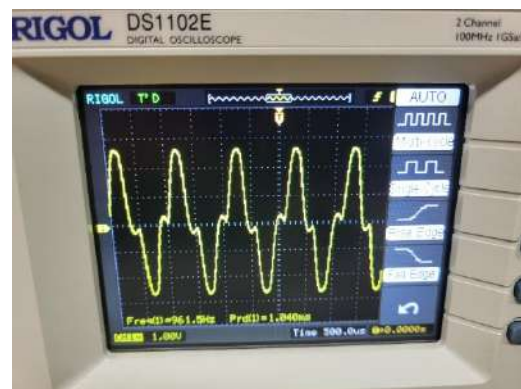
1kHz Flat Top Sampling (32kHz)



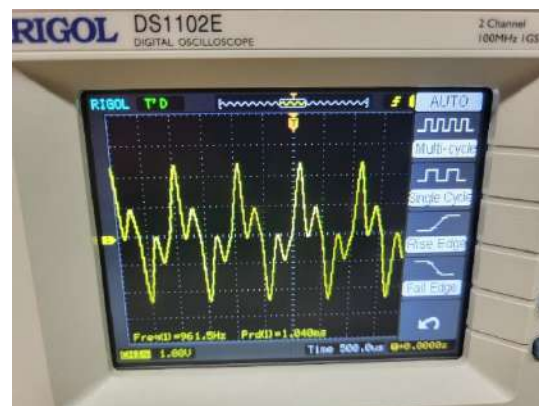
1kHz Flat Top Sampling (64kHz)



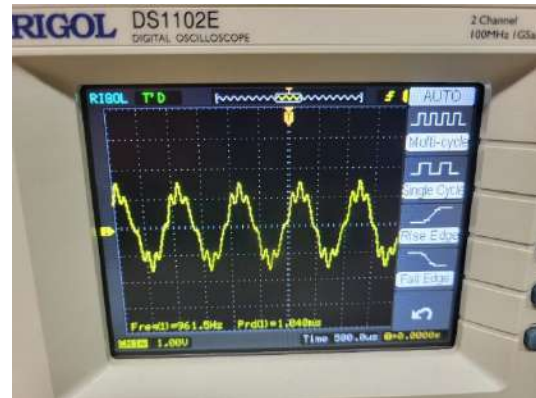
1kHz Flat Top Sampling Output (2kHz)



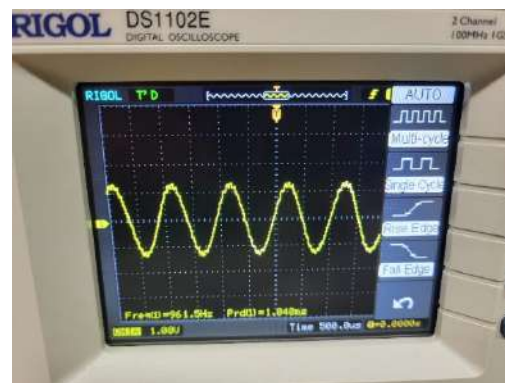
1kHz Flat Top Sampling Output (4kHz)



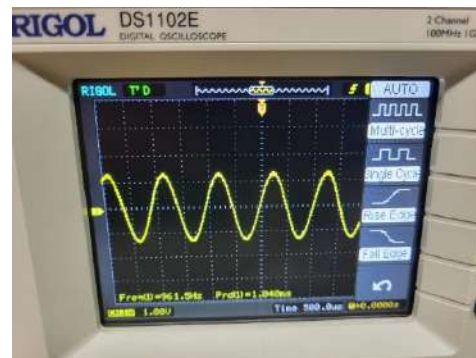
1kHz Flat Top Sampling Output (8kHz)



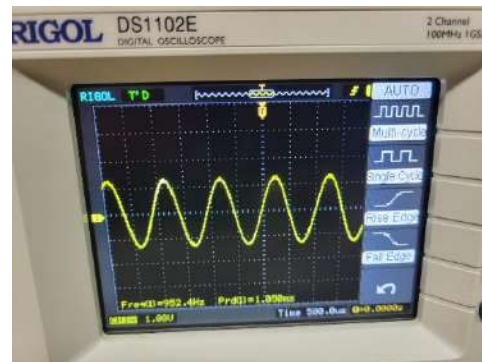
1kHz Flat Top Sampling Output (16kHz)



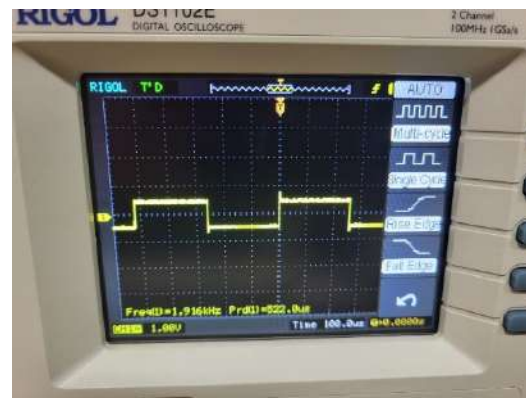
1kHz Flat Top Sampling Output (32kHz)



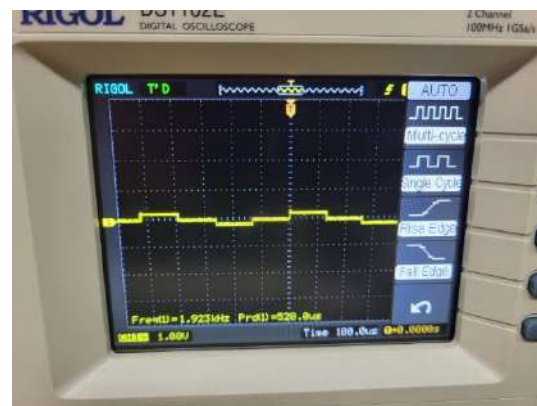
1kHz Flat Top Sampling Output
(64kHz)



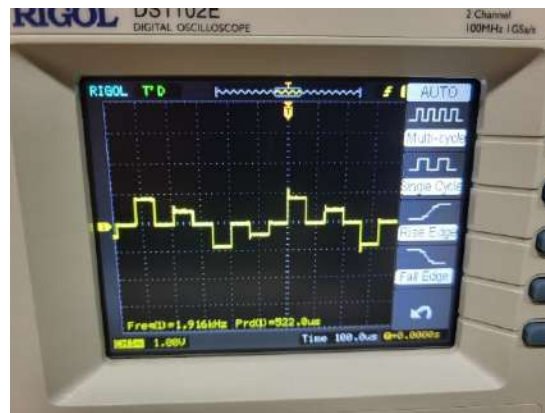
2kHz Flat Top Sampling (2kHz)



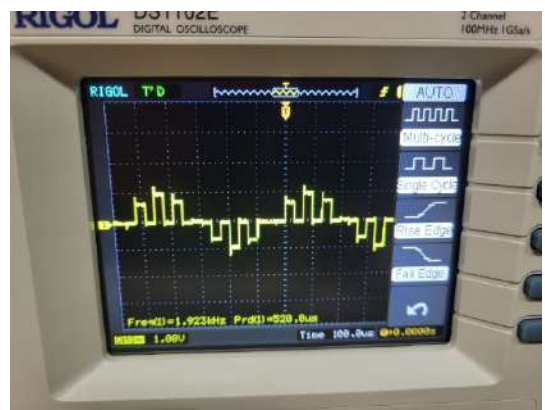
2kHz Flat Top Sampling (4kHz)



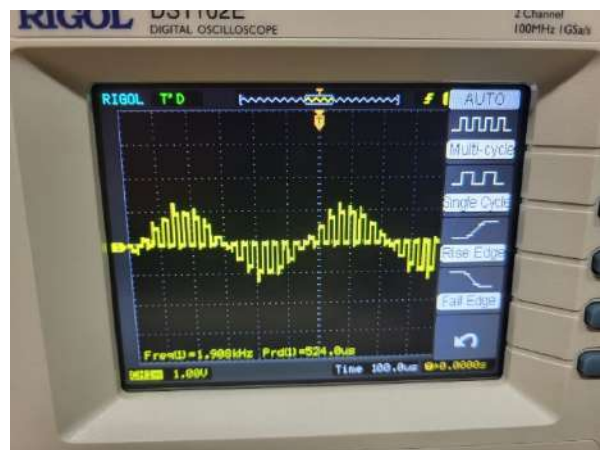
2kHz Flat Top Sampling (8kHz)



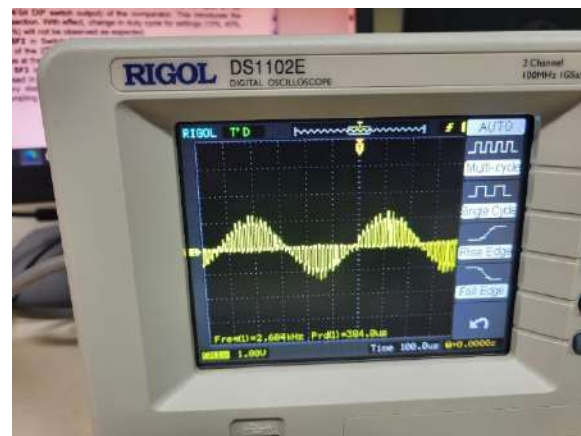
2kHz Flat Top Sampling (16kHz)



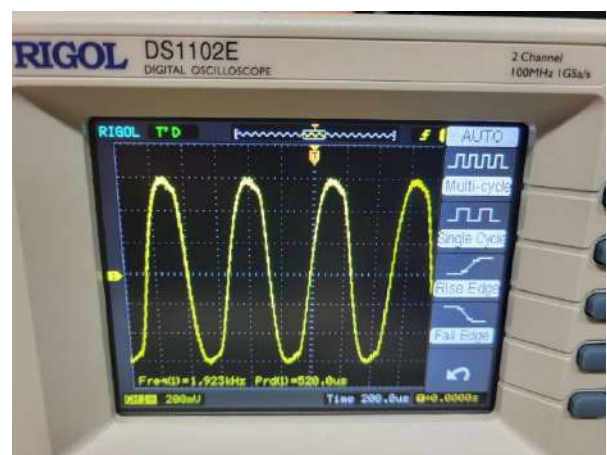
2kHz Flat Top Sampling (32kHz)



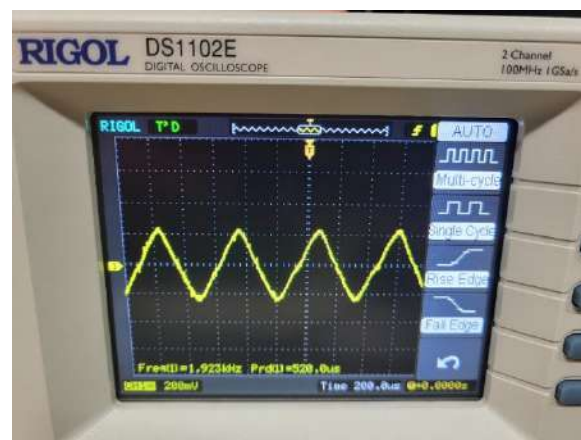
2kHz Flat Top Sampling (64kHz)



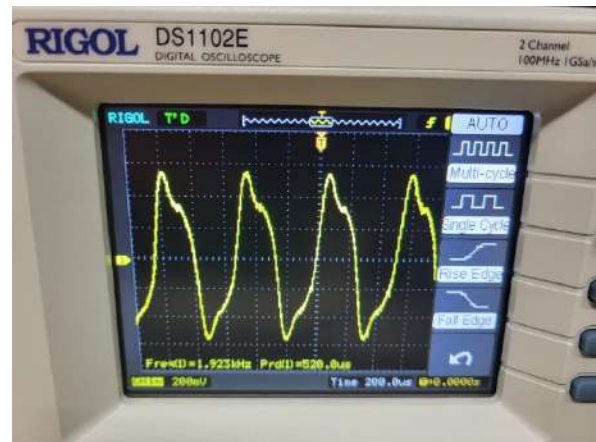
2kHz Flat Top Sampling Output (2kHz)



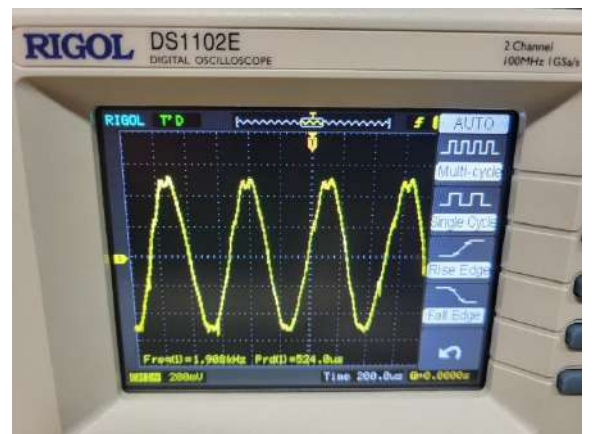
2kHz Flat Top Sampling Output (4kHz)



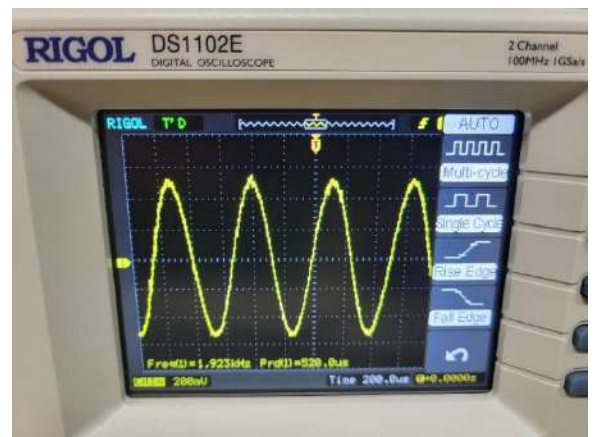
2kHz Flat Top Sampling Output (8kHz)



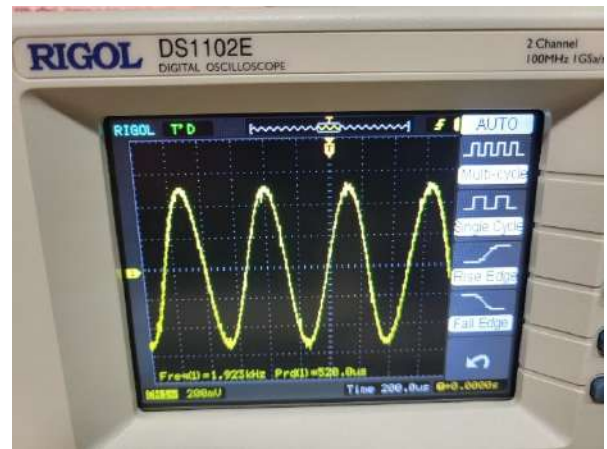
2kHz Flat Top Sampling Output (16kHz)



2kHz Flat Top Sampling Output (32kHz)



2kHz Flat Top Sampling Output
(64kHz)



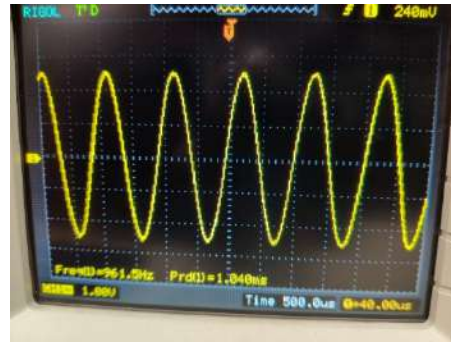
Conclusion:

- Through this experiment, we learned how by varying the sampling frequency, we see the impact of it on the amplitude of the reconstructed signals.
- We see the output of Natural sampled, flat top sampled and sample and hold sampled signal outputs and their corresponding reconstructed output of 2nd order low pass butterworth filter.

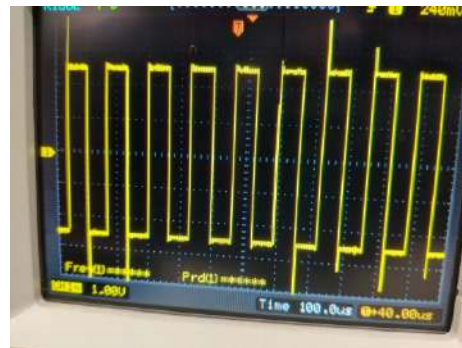
Experiment 4 (DCL 01)

Signal	Output
--------	--------

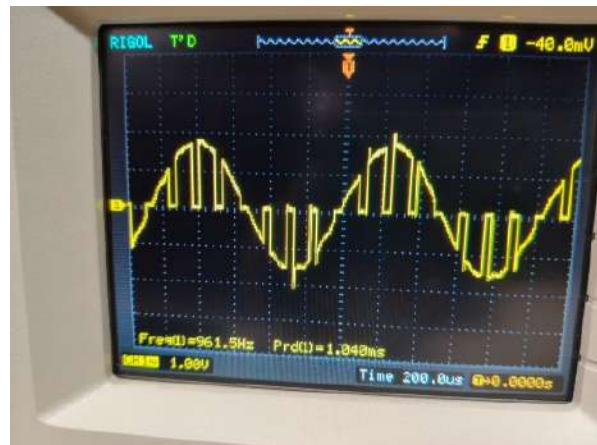
1kHz input signal



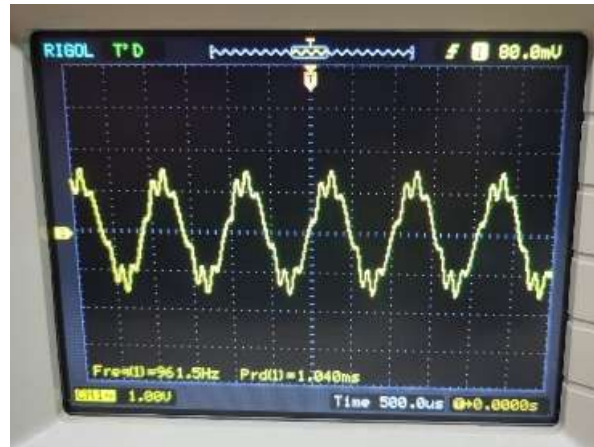
8kHz clock



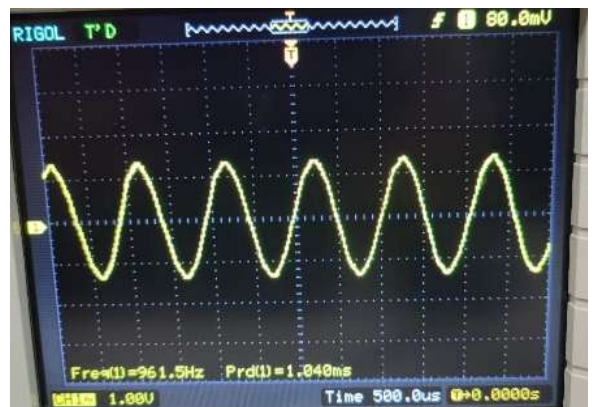
1kHz naturally sampled signal



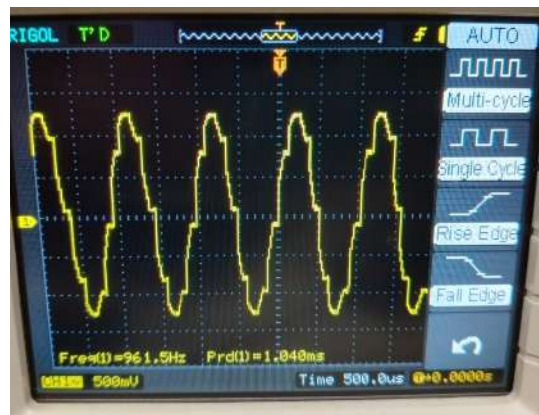
1kHz naturally sampled signal after
2nd order LPF



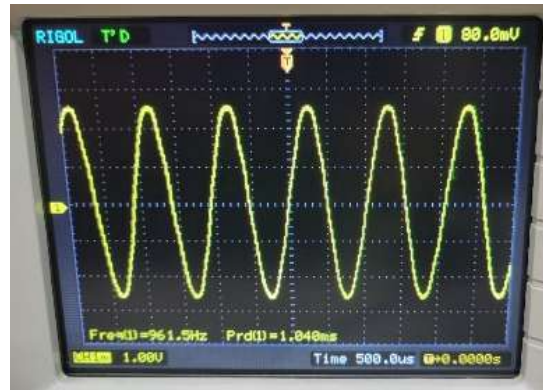
1kHz naturally sampled signal after
4th order LPF



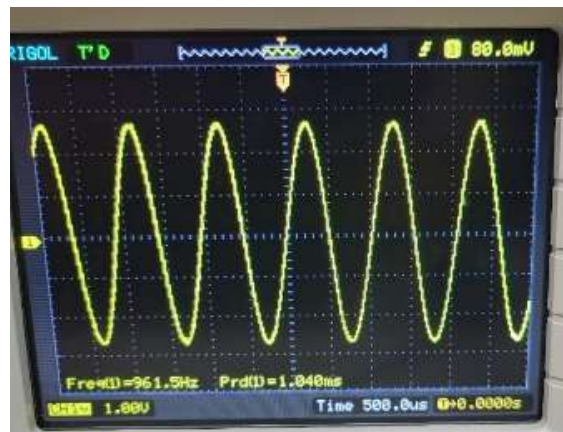
1kHz sample and hold signal



1kHz sample and hold signal after 2nd order LPF



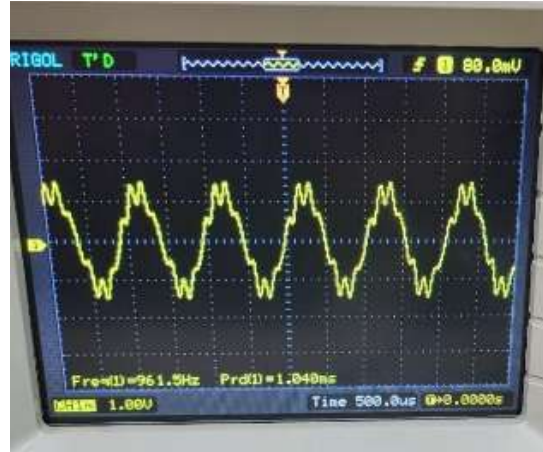
1kHz sample and hold signal after 4th order LPF



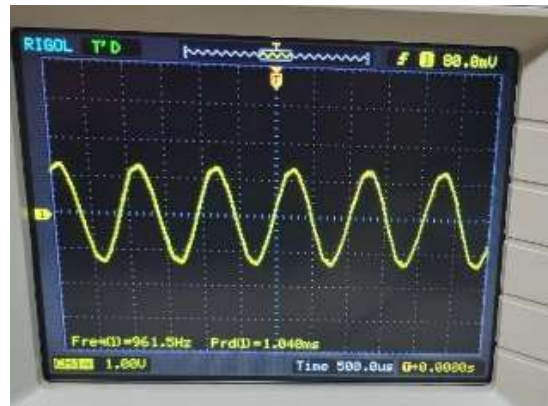
1kHz flat top sampled signal



1kHz flat top sampled signal after 2nd order LPF



1kHz flat top sampled signal after 4th order LPF



Conclusion:

- Through this experiment, we learned how by varying the final reconstruction filters namely the 2nd order and 4th order low pass butterworth filters, we are able to see the effect of it on the reconstruction of the signal.
- We observed the differences in the reconstructed signal obtained from 2nd order and 4th order low pass butterworth filters.

Experiment 2:

Matlab code:

```

ts=1/16; %sampling interval
time_interval = 0:ts:1; %sampling time instants
%%time domain signal evaluated at sampling instants
signal_timedomain = sin(pi*time_interval); %sinusoidal pulse in our example
fs_desired = 1/160; %desired frequency granularity
Nmin = ceil(1/(fs_desired*ts)); %minimum length DFT for desired frequency granularity
%for efficient computation, choose FFT size to be power of 2
Nfft = 2^(nextpow2(Nmin)) %FFT size = the next power of 2 at least as big as Nmin
%Alternatively, one could also use DFT size equal to the minimum length
%Nfft=Nmin;
%note: fft function in Matlab is just the DFT when Nfft is not a power of 2
%freq domain signal computed using DFT
%fft function of size Nfft automatically zeropads as needed
signal_freqdomain = ts*fft(signal_timedomain,Nfft);
%fftshift function shifts DC to center of spectrum
signal_freqdomain_centered = fftshift(signal_freqdomain);
fs=1/(Nfft*ts); %actual frequency resolution attained
%set of frequencies for which Fourier transform has been computed using DFT
freqs = ((1:Nfft)-1-Nfft/2)*fs;
%plot the magnitude spectrum
plot(freqs,abs(signal_freqdomain_centered));
xlabel('Frequency');
ylabel('Magnitude Spectrum')

```

```

% Define the signal parameters
fs = 16e6; % Sampling rate in Hz (16 MHz)
ts = 1/fs; % Sampling interval in seconds
tstart = -8; % Start time in microseconds
tend = 8; % End time in microseconds
time_interval = tstart:ts*1e6:tend; % Time range in microseconds

```

```

% Define the signal
signal_timedomain = 3 * sinc(2 * time_interval - 3);

```

```

% Desired frequency resolution
df_desired = 1; % 1 KHz

```

```

% Compute the Fourier Transform using contFT function
[X, f, df] = contFT(signal_timedomain, tstart, ts*1e6, df_desired);

```

```

% Plot the magnitude spectrum
figure;
plot(f, abs(X));

```

```

xlabel('Frequency (kHz)');
ylabel('Magnitude Spectrum');
title('Magnitude Spectrum of s(t)');

```

```

% Plot the phase spectrum
figure;
plot(f, angle(X));
xlabel('Frequency (kHz)');
ylabel('Phase Spectrum');
title('Phase Spectrum of s(t)');

```

```

function [X,f,df] = contFT(x,tstart,dt,df_desired)
%Use Matlab DFT for approximate computation of continuous time Fourier
%transform
%INPUTS
%x = vector of time domain samples, assumed uniformly spaced
%tstart= time at which first sample is taken
%dt = spacing between samples
%df_desired = desired frequency resolution
%OUTPUTS
%X=vector of samples of Fourier transform
%f=corresponding vector of frequencies at which samples are obtained
%df=freq resolution attained (redundant--already available from
%difference of consecutive entries of f)
%%%%%%%%%%
%minimum FFT size determined by desired freq res or length of x
Nmin=max(ceil(1/(df_desired*dt)),length(x));
%choose FFT size to be the next power of 2
Nfft = 2^(nextpow2(Nmin))
%compute Fourier transform, centering around DC
X=dt*fftshift(fft(x,Nfft));
%achieved frequency resolution
df=1/(Nfft*dt)
%range of frequencies covered
f = ((0:Nfft-1)-Nfft/2)*df; %same as f=-1/(2*dt):df:1/(2*dt) - df
%phase shift associated with start time
X=X.*exp(-j*2*pi*f*tstart);
end

```

Explanation:

This MATLAB code computes the Fourier Transform of a signal. It samples a sinusoidal signal, determines the appropriate FFT size for a desired frequency resolution, computes the FFT, and

plots the magnitude spectrum. The second part does a similar process for a sinc signal, using a custom function `contFT` to calculate and plot both magnitude and phase spectra.

Output:

