ASEN 3300, Fall 2022: Lab 7 Report Submission

Names:	Jared Steffen
	Brady Sivey
	Joshua Geeting
Section:	012

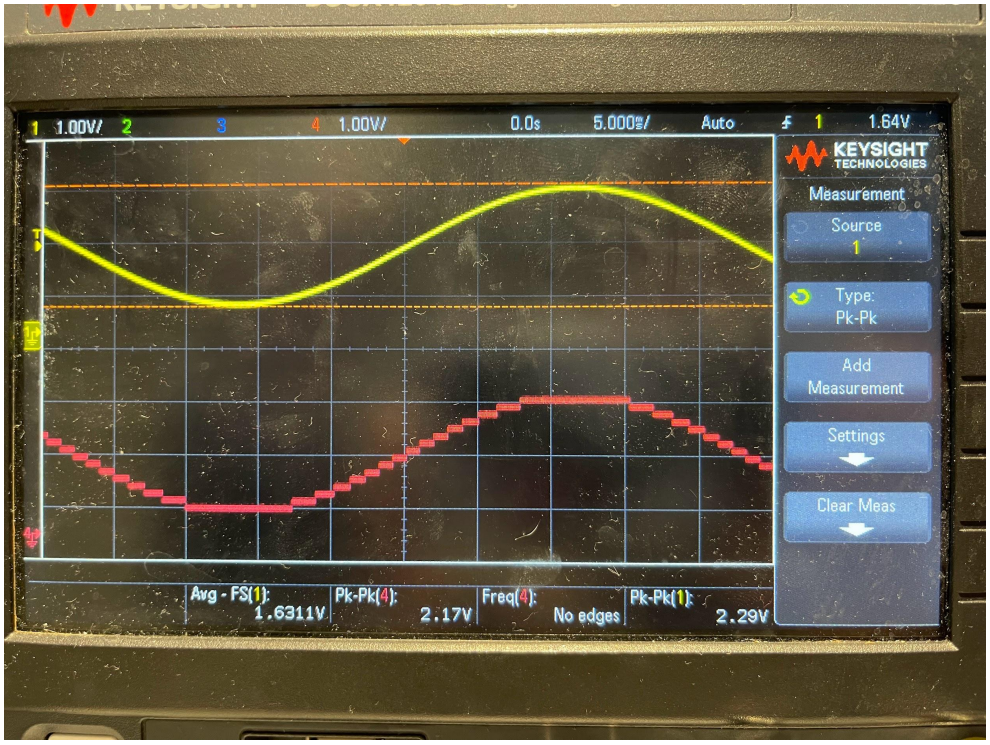
Please have the LAB DOCUMENT OPEN while you use this template. It is stripped and provided simply as a place to record your results and submit them. All instructions are in the lab document.

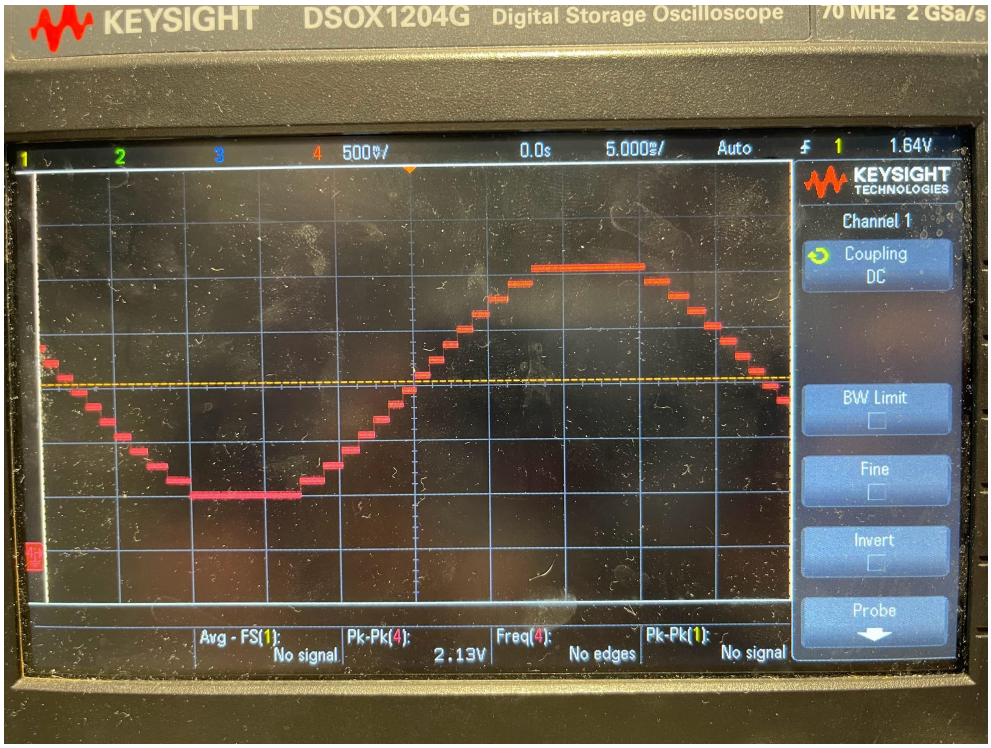
4. Experiment (25 points)

4.1 Quantization

J. Record your responses to 4.1.j below:

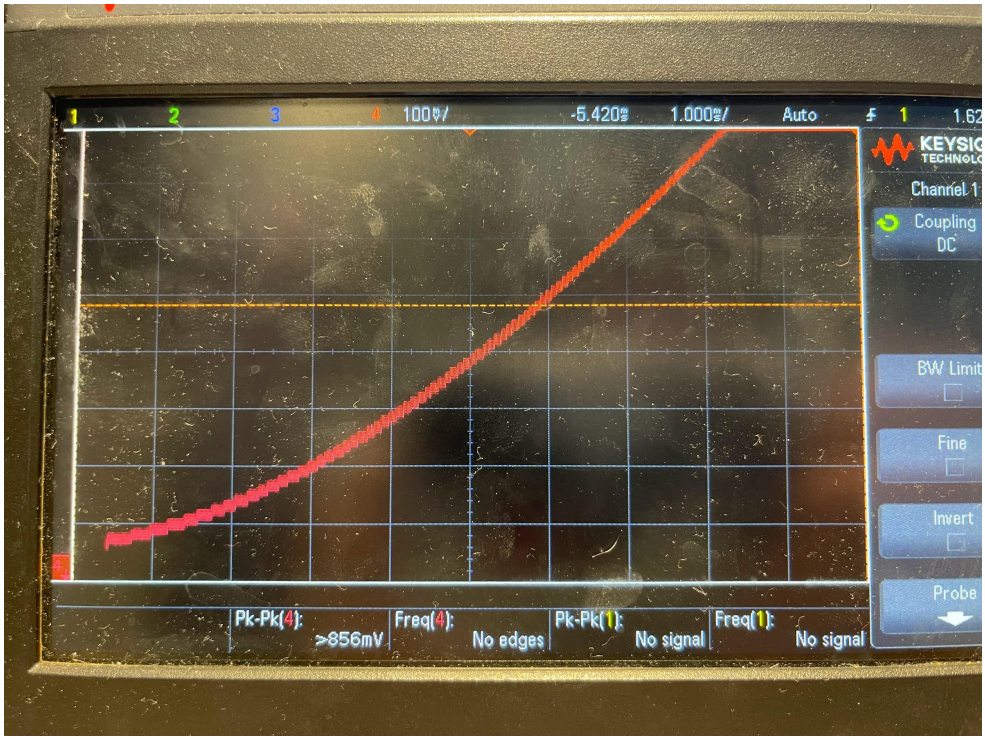
V_{PP} 4 bit:	2.17 V
Frequency 4 bit:	21.7 Hz
V_{PP} 12 bit:	2.29 V
Frequency 12 bit:	21.8 Hz
4 bit quantization level (V)	16


1 1.00V/ 2 3 4 1.00V/ 0.0s 5.000s/ Auto 1 1.64V
KEYSIGHT TECHNOLOGIES
Measurement
Source 1
Type: Pk-Pk
Add Measurement
Settings
Clear Meas
Avg - FS(1): 1.6311V
Pk-Pk(4): 2.17V
Freq(4): No edges
Pk-Pk(1): 2.29V

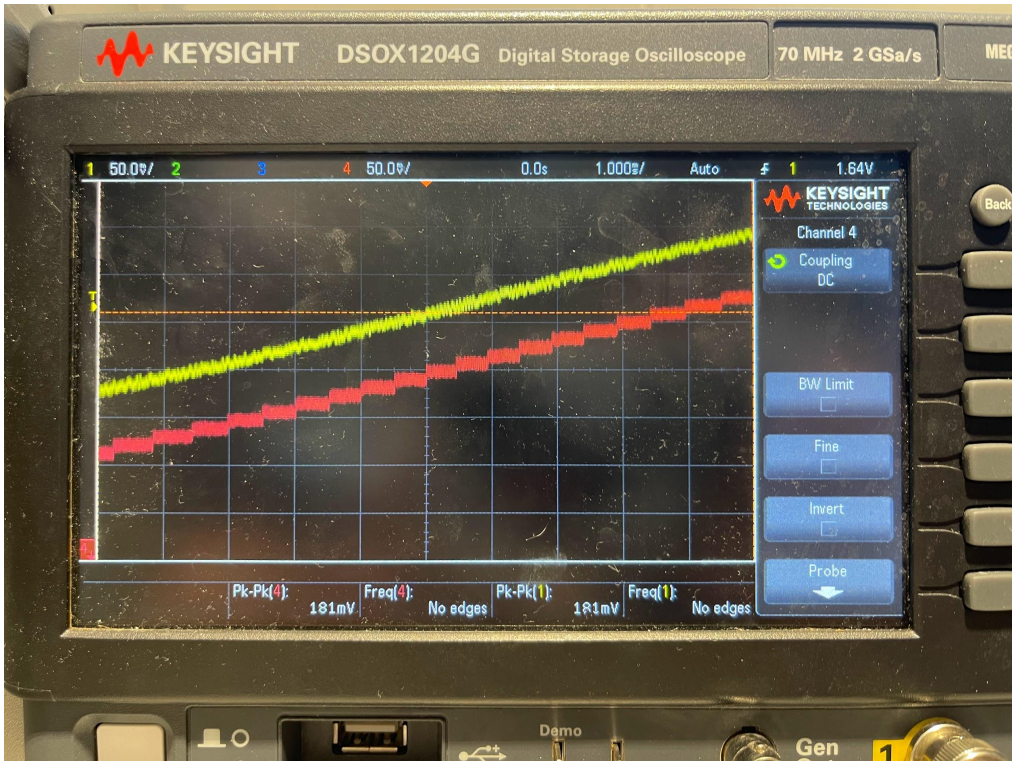

KEYSIGHT DSOX1204G Digital Storage Oscilloscope 70 MHz 2 GSa/s
1 2 3 4 500mV/ 0.0s 5.000s/ Auto 1 1.64V
KEYSIGHT TECHNOLOGIES
Channel 1
Coupling DC
BW Limit
Fine
Invert
Probe
Avg - FS(1): No signal
Pk-Pk(4): 2.13V
Freq(4): No edges
Pk-Pk(1): No signal

N. Record your responses to 4.1.n below:

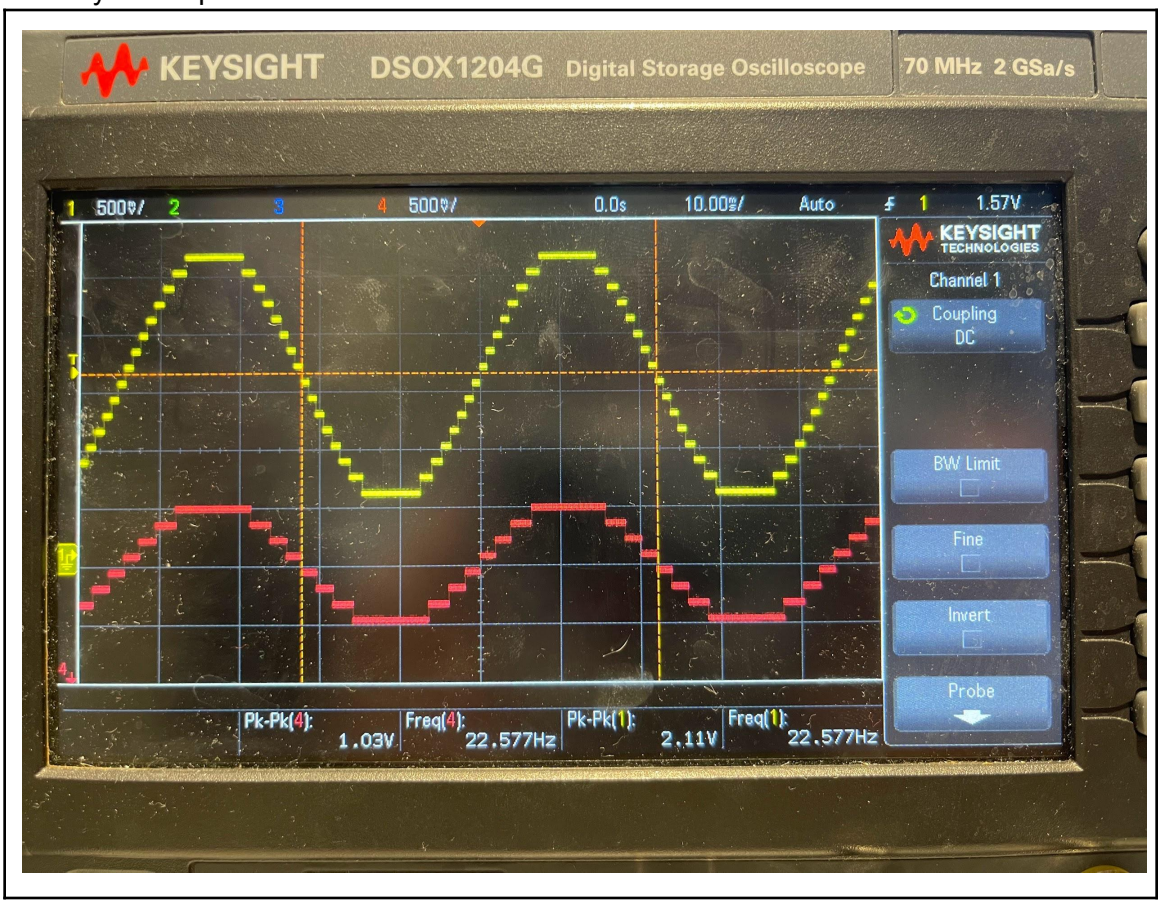
V_{PP} 8 bit:	2.29 V
Frequency 8 bit:	21.76 Hz
V_{PP} 12 bit:	2.29 V
Frequency 12 bit:	21.8 Hz
8 bit quantization level (V)	256
Can 12 bit quantization levels be seen?	4096



R. Record your responses to 4.1.r below:



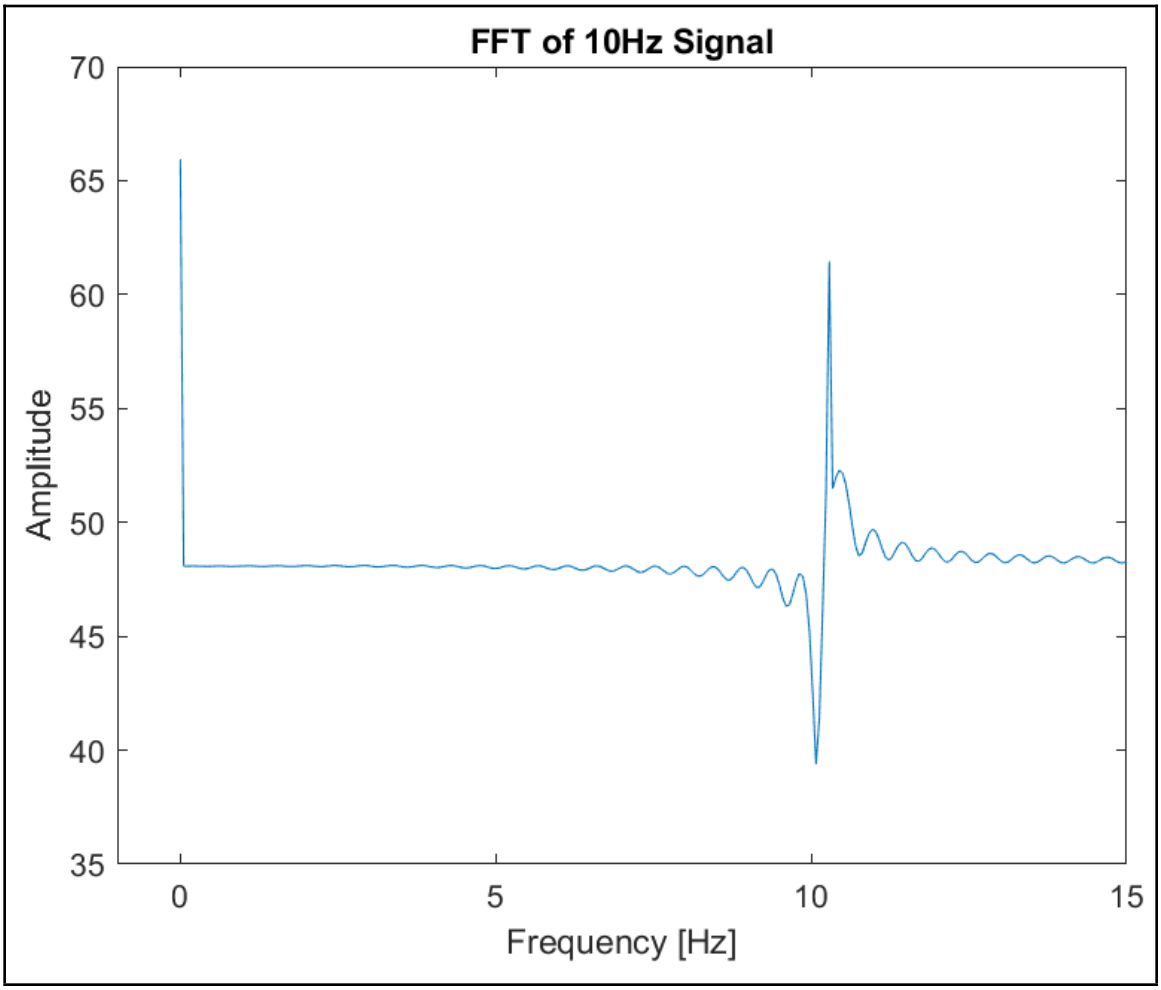
V. Record your response to 4.1.v below:



4.2 Sampling Frequency and Aliasing

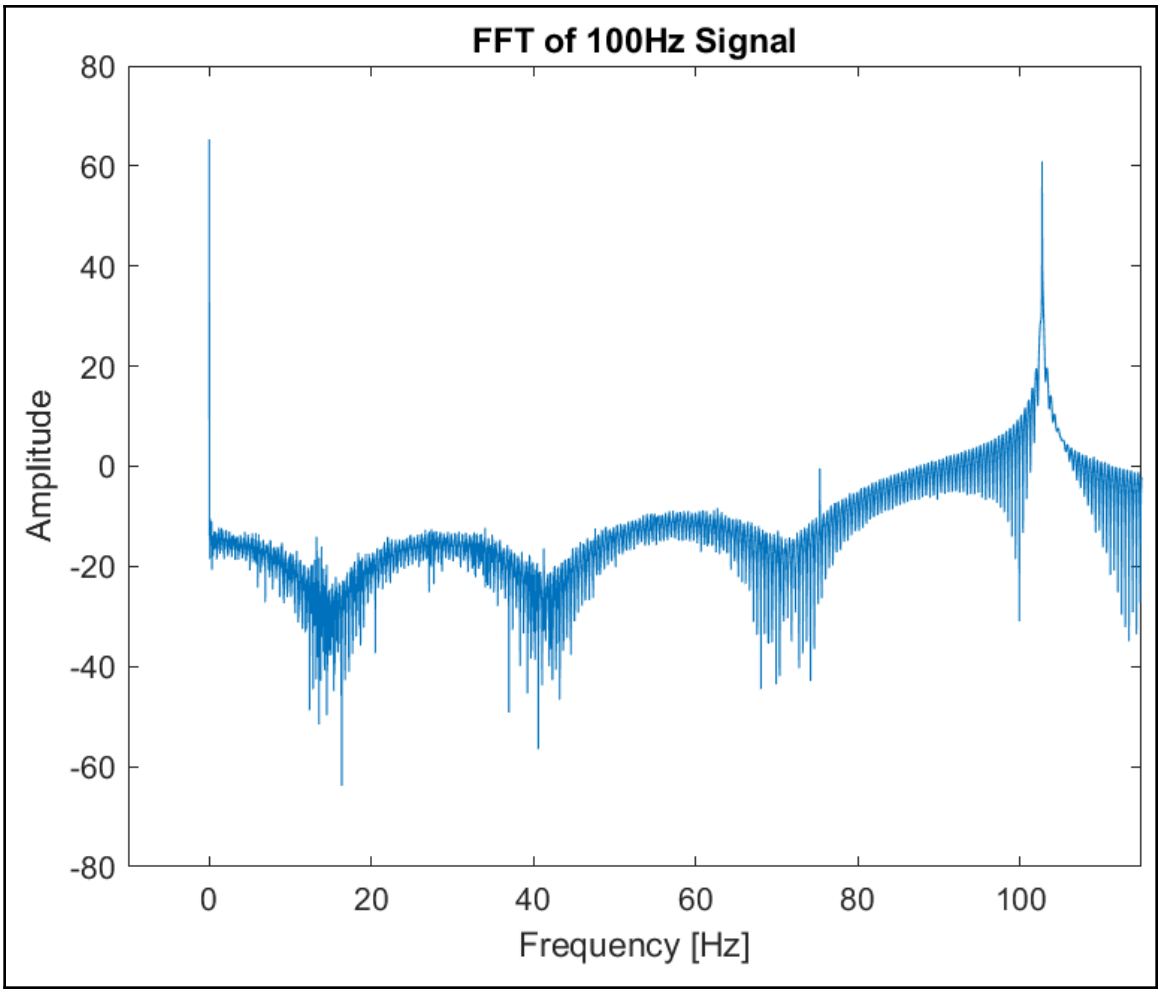
F. Record your responses to 4.2.f in the spaces below:

Dominant f in FFT (Hz):	10 Hz
Is aliasing present?	No



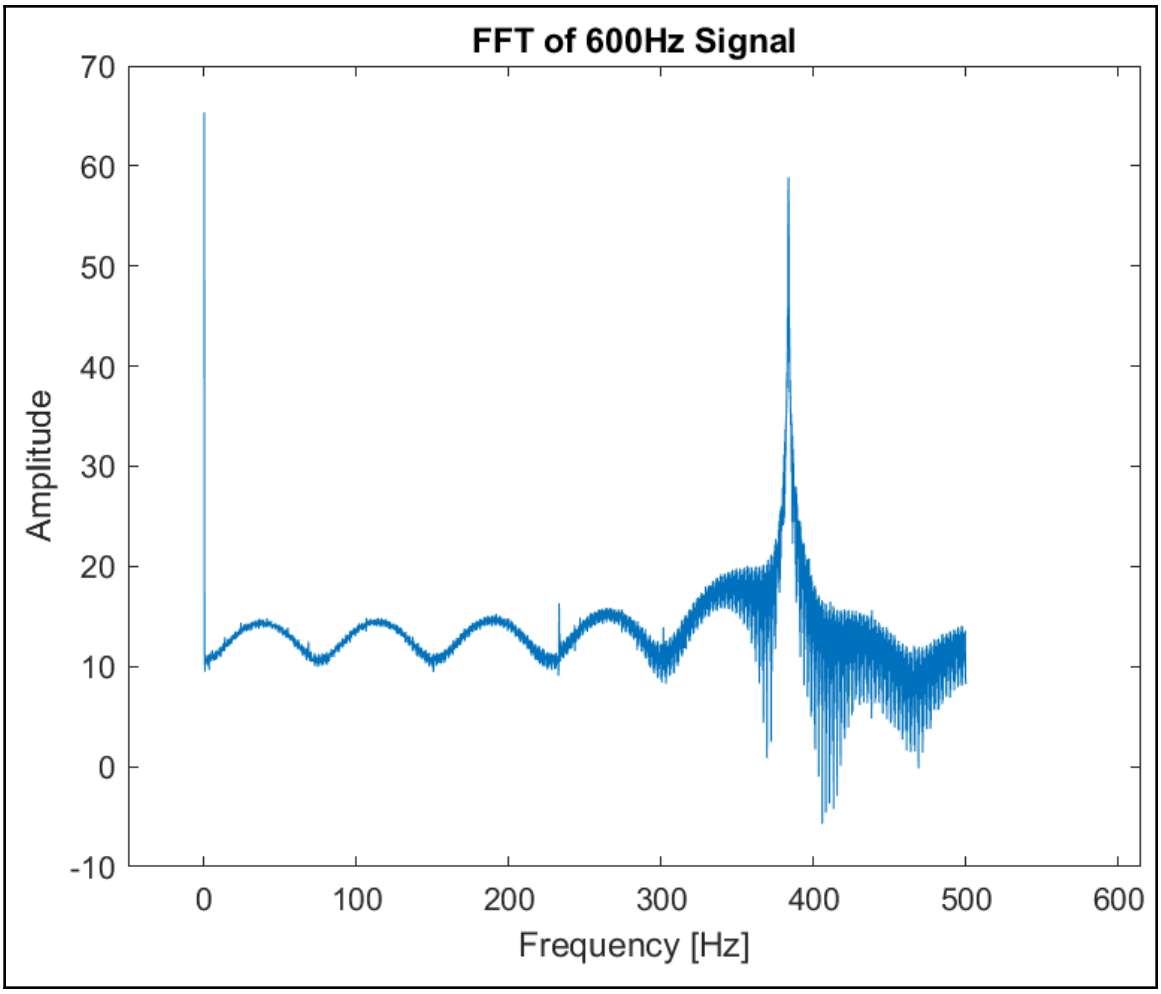
G. Record your responses to 4.2.g in the spaces below:

Dominant f in FFT (Hz):	100 Hz
Is aliasing present?	No



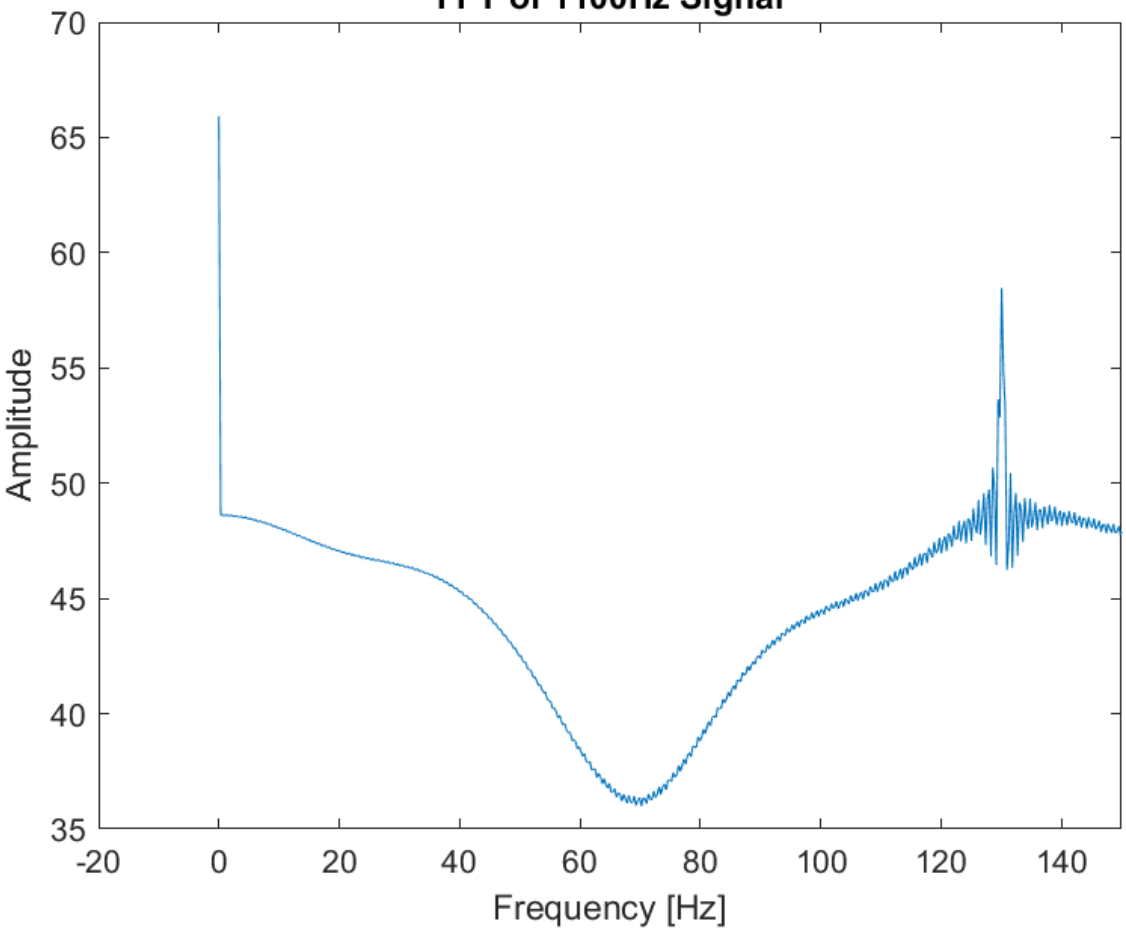
H. Record your responses to 4.2.h in the spaces below:

Dominant f in FFT (Hz):	400 Hz
Is aliasing present?	Yes



I. Record your responses to 4.2.i in the spaces below:

Dominant f in FFT (Hz):	100 Hz
Is aliasing present.	Yes



Note: We know that we are supposed to see 100 Hz, but we had a TA verify that everything we did was correct.

4.3 Computational Speed

C. Record your response to 4.2.c below

Arduino Computation time: 18.73 seconds

D. Record your response to 4.2.d below:

MATLAB computation time: 2.655 seconds

E. Record your response to 4.2.e below:

The MATLAB computation time is much faster than that of the Arduino Due

---

Analysis (25 points)

5.1 Quantization (12 pts)

A. Record your response to 5.1.a below:

The sinusoidal frequency observed from quantization part i was approximately 130 Hz. Based on a sampling frequency of 1000 Hz and an input frequency of 1100 Hz, we should see a frequency of around 100 Hz due to aliasing. We could remove this aliasing by sampling the wave at 2200 Hz.

B. Record your response to 5.1.b below:

The more bits that you have, the smaller that the resolution signal gets assuming we maintain a consistent full scale resolution sine wave for the different bits. When observing the wave for 4 bits, we observed around 16 levels, which is consistent with what it should have been. For 8 bits, it is a little tougher to observe all 256 levels, but we could approximately tell that it did approximately match what we expected theoretically. We would need to observe the input sine wave at a scale of 1/4096 to observe the resolution for the sine wave at 12 bits.

C. Record your response to 5.1.c below:

Range = 2.75 - 0.55 = 2.2 V
Signal Range = 0.25 V

For 12 bits, (0.25 V/2.2 V) * (2^12) = 465.45 bins which rounds down to 465 bins.

For 8 bits, (0.25 / 2.2) * (2^8) = 29.09 bins which rounds down to 29 bins.

D. Record your response to 5.1.d below:

For 4 bits
Number of bins used = 2^(bits) = 16 bins
LSB = 1Vpp/2^(bits), so LSB equals 0.0625 V

E. Record your response to 5.1.e below:

One way to use a small number of bits to represent a large amplitude signal that changes slowly would be to use a quantization technique that adjusts quantization levels based on the signal's characteristics. Increasing the sample rate to oversample and averaging over a period of time could be a method for reducing the effective noise.

5.2 Sampling Frequency and Aliasing (10 pts)

A. Record your response to 5.2.a below:

If the Arduino Due was able to instantly complete the read command, the Nyquist frequency would be 500 Hz.

B. Record your response to 5.2.b below:

The actual Nyquist frequency of the Due sampling sketch including computation time is ~425 Hz.

C. Record your response to 5.2.c below:

Aliasing can be used for signal processing and you can capture signals with a specific structure at a lower example sample rate than Nyquist without significant information loss. An example is when we input a 1100 Hz sine wave into the Arduino Due with a sampling frequency of 1000 Hz and Nyquist frequency of 500 Hz, a signal of 100 Hz is what our input signal frequency appears to be.

D. Record your response to 5.2.d below:

The use of a low-pass filter prior to A/D conversion is desirable and sometimes necessary when there are frequency components close to the Nyquist frequency because those higher frequency components can fold back to appear as low frequency components, which can give incorrect data. The use of the low-pass filter eliminates those higher frequency components so that the represented signal is accurate and not corrupted by the folding back of higher frequency components.

5.3 Computational Speed (3pts)

A. Record your response to 5.3.a below:

The processor is an Atmel ATSAM3X8E ARM cortex-M3 CPU and has a speed of 18.73 seconds. Matlab had a processing speed of 2.65 seconds.Intel(R) Core(TM) i7-8559U CPU @ 2.70GHz, 2712 Mhz, 4 Core(s), 8 Logical Processor(s). The matlab had a much quicker processing time than the arduino as matlab was using a microprocessor from the computer. Micro-processor are able to focus on 1 thing and do that task much faster than a regular processor which has to focus on multiple tasks.

B. Record your response to 5.3.b below:

We cannot make a fair comparison between MATLAB and Arduino C++ programs. MATLAB code is distinct in its own right, whereas Arduino relies on C++. Arduino programming is notably more complex. In MATLAB, only 7 lines of code were needed, while Arduino required 23 lines. Additionally, Arduino code is compiled, whereas MATLAB code is interpreted. With Arduino, you need to compile the code and then transfer it directly to the device, while MATLAB can read it directly. These represent two different approaches. MATLAB's random number generator is considerably more secure, whereas Arduino relies on hardware and C++ libraries for its random numbers, which, in reality, are only somewhat random. Lastly, MATLAB primarily uses floating-point algorithms, whereas Arduino is recommended to use fixed-point algorithms. Both can use either one, but each excels with only one. Floating-point allows for high precision and can handle a broader range of values, whereas fixed-point has more limited precision and is better suited for Arduino due to its restricted processing power and memory. Here, we can observe the substantial differences between MATLAB and Arduino C++ when comparing the programs, and this is why there is a significant difference in computing time.