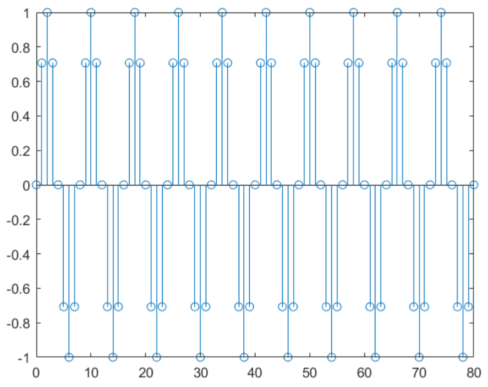
Mitchell Larson
EE 3221
12/7/17
Lab2

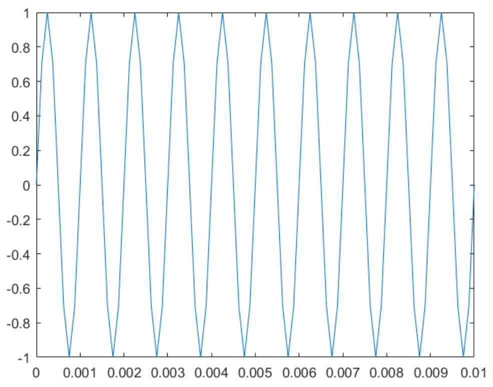
1) Sinusoidal MatLab Signal with Varying Frequencies

- fs = 8000 | f = 1000

Discrete



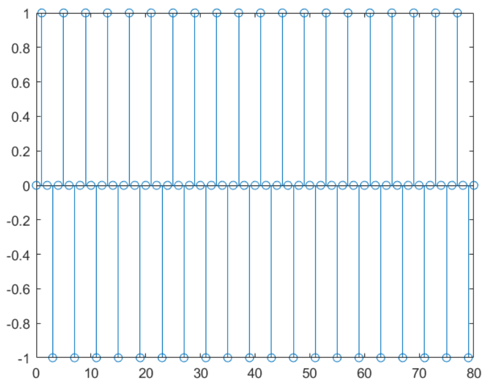
Continuous



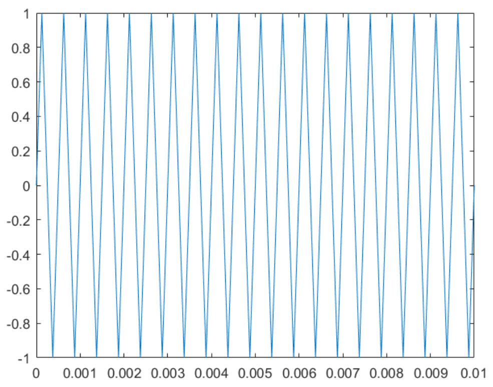
With this combination of sampling frequency and the frequency occurring, it is obvious that aliasing is not occurring. This is known because the sampling frequency obeys the Nyquist theorem. As expected, 8 samples are taken per period of the continuous signal, and the values appear to accurately represent the continuous signal. It would be possible to reconstruct the original signal from the samples taken.

- fs = 8000 | f = 2000

Discrete



Continuous

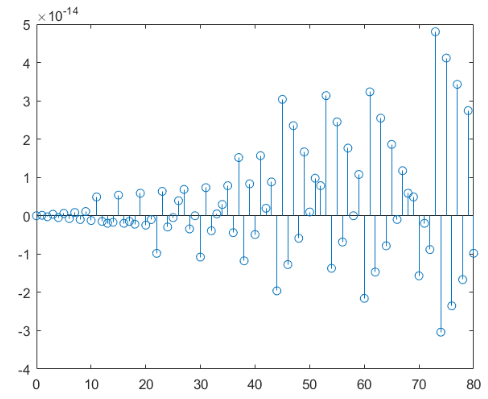


Similar to the previous example, aliasing does not occur because the sampling frequency satisfies the conditions of the Nyquist Theorem. While it would be possible to reconstruct the original signal from the samples, the signal is sampled as detailed as the previous example. The sampled signal now only stores the peaks/valleys and zeros of the continuous function.

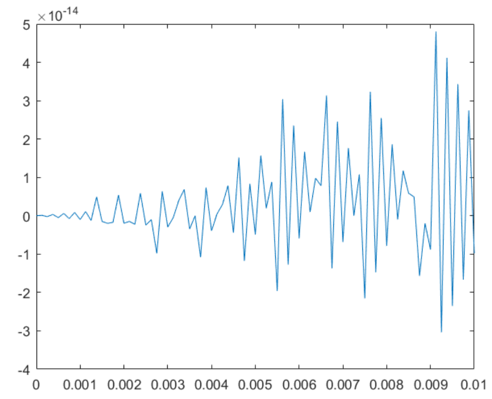
Mitchell Larson
EE 3221
12/7/17
Lab2

- fs = 8000 | f = 4000

Discrete



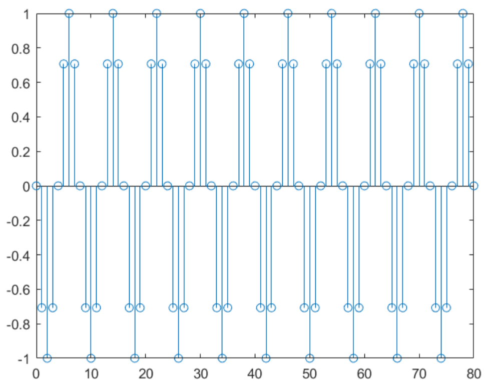
Continuous



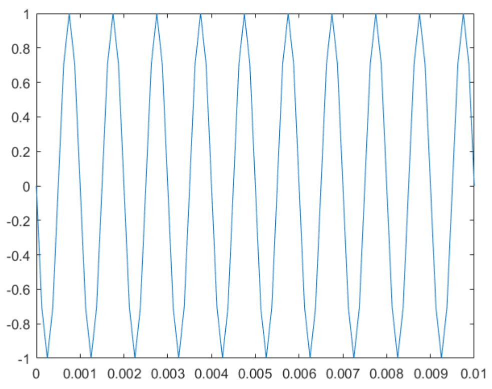
As the sampling frequency begins to approach the barrier defined by the Nyquist Theorem, the signal begins to deteriorate. As seen in the images above, the data become unrecognizable, and extremely small. This is because the signal is being sampled at every point where it becomes 0. The small variation from 0 seen here is most likely a result of floating point arithmetic done by the computer. Due to the conditions just discussed, it may be possible for aliasing to begin under these conditions.

- fs = 8000 | f = 7000

Discrete



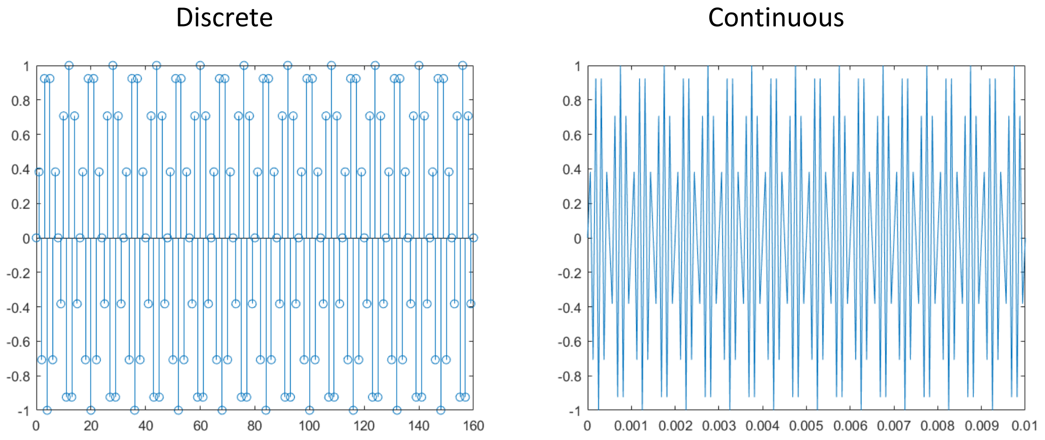
Continuous



While this signal appears to be a correct representation of the sin wave, it is important to note that it is not. The actual sin wave has a frequency of 7000, so each period of the wave should complete in 1/7000 units. This means that the 2nd period should finish at 2.86e-4 units. The period appears to complete at 2.5e-4 units. The reason that this is occurring is because the sampling frequency does not obey the Nyquist theorem. This means that aliasing is occurring, which changes the reconstructed signal.

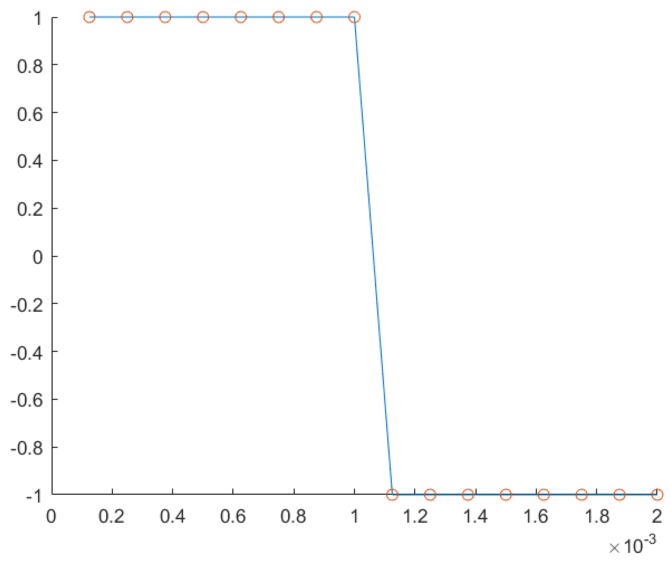
Mitchell Larson
EE 3221
12/7/17
Lab2

- fs = 16000 | f = 7000



The results of this simulation may appear to not accurately represent the signal, but the sampling frequency is high enough to avoid aliasing and capture the original signal. Since the sampling is not occurring at the peaks of the original signal, the sampled signal does not appear to be correct, but the Nyquist Theorem assures that the signal can be reconstructed from the sampled data. Aliasing will not occur.

2) Frequency of square wave

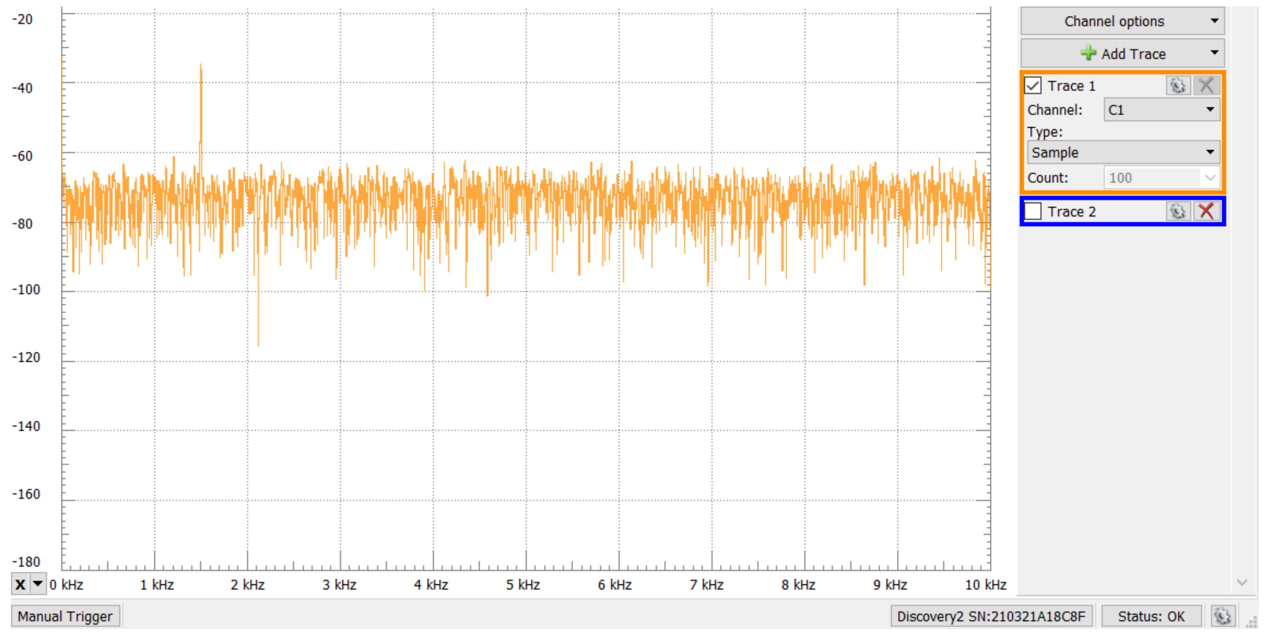


The frequency can be derived by analyzing the continuous representation of the signal. If the signal shown above describes a single period of the original signal, then it is known that a period of the original signal completes in 2e-3 time units. To find the frequency, the inverse of that number should be taken. This results in a frequency of 500 units per second.

Mitchell Larson
EE 3221
12/7/17
Lab2

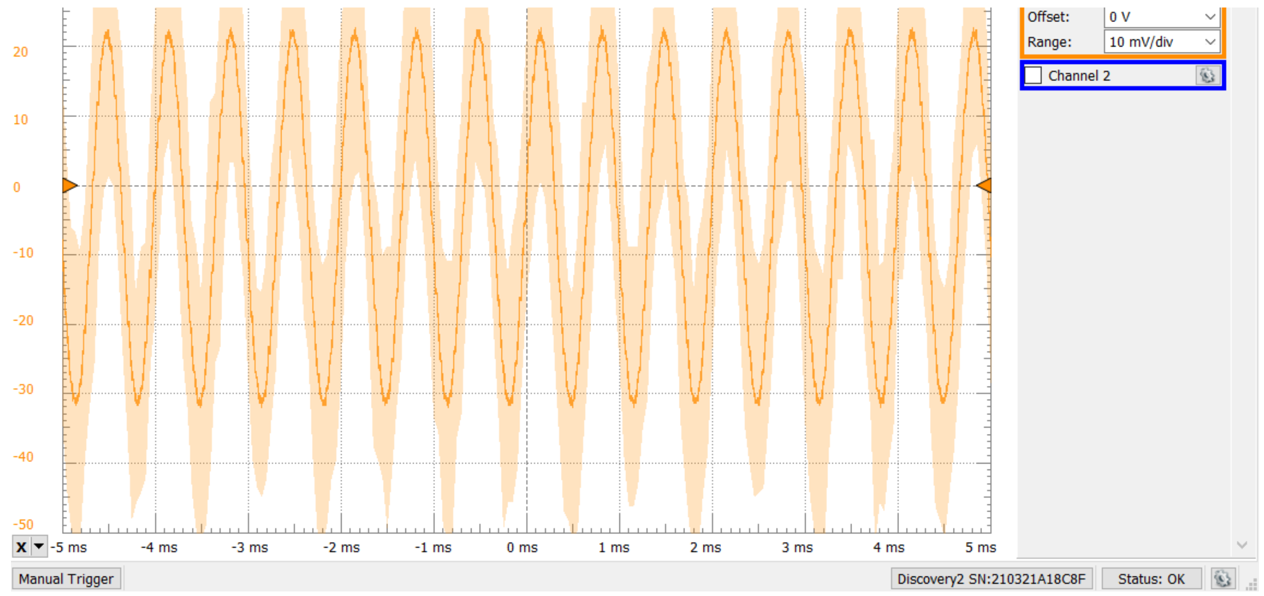
3) Signal Generation Using the Math.h Sine Function

- f = 1500 Hz



Trace 1
Channel: C1
Type: Sample
Count: 100
Trace 2
Discovery2 SN:210321A18C8F

Figure 1 - Frequency Spectrum Captured for 1.5 kHz



Offset: 0 V
Range: 10 mV/div
Channel 2
Discovery2 SN:210321A18C8F

Figure 2 - Continuous time signal captured for 1.5 kHz

As expected, a sinusoidal signal was generated that appeared to have a frequency of 1.5 kHz. This was then verified by observing the frequency spectrum. A spike occurred near 1.5 kHz, which suggests that the signal did indeed have a frequency of 1.5 kHz.

Mitchell Larson
EE 3221
12/7/17
Lab2

- f = 2641 Hz

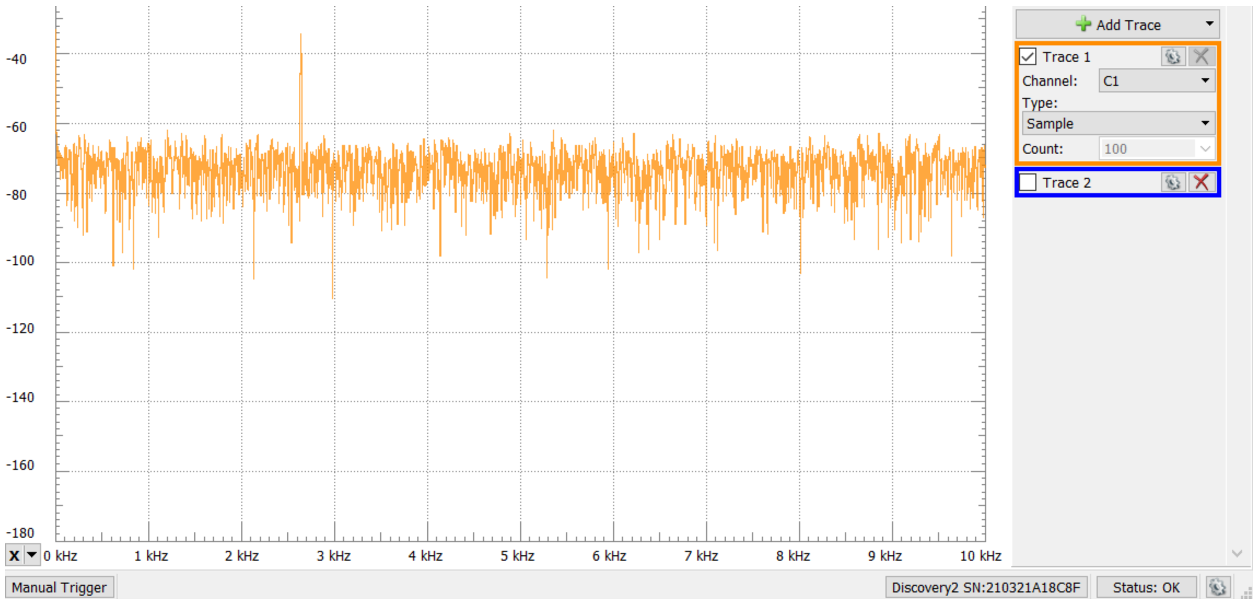


Figure 3 - Frequency Spectrum Captured for 2.641 kHz

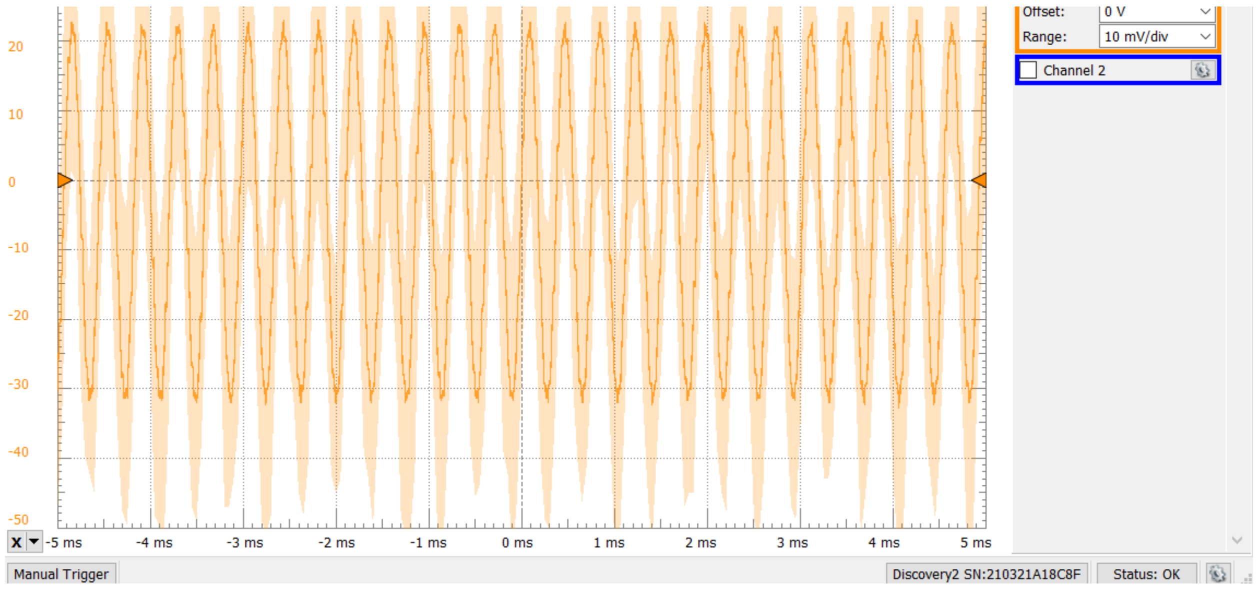
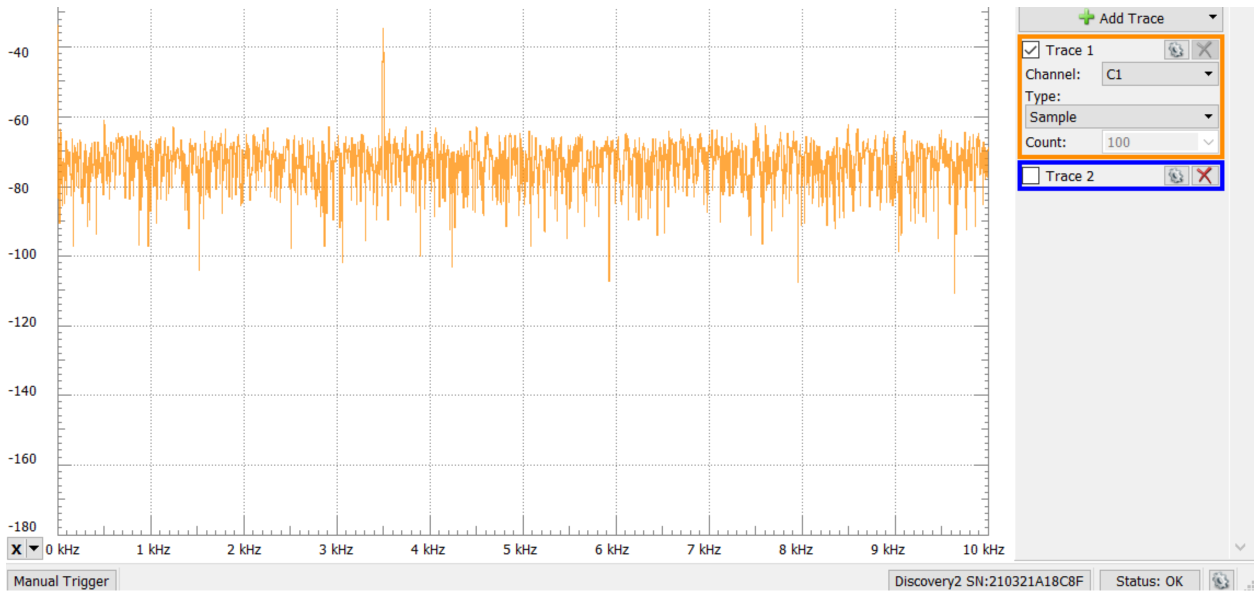


Figure 4 - Continuous time signal captured for 2.641 kHz

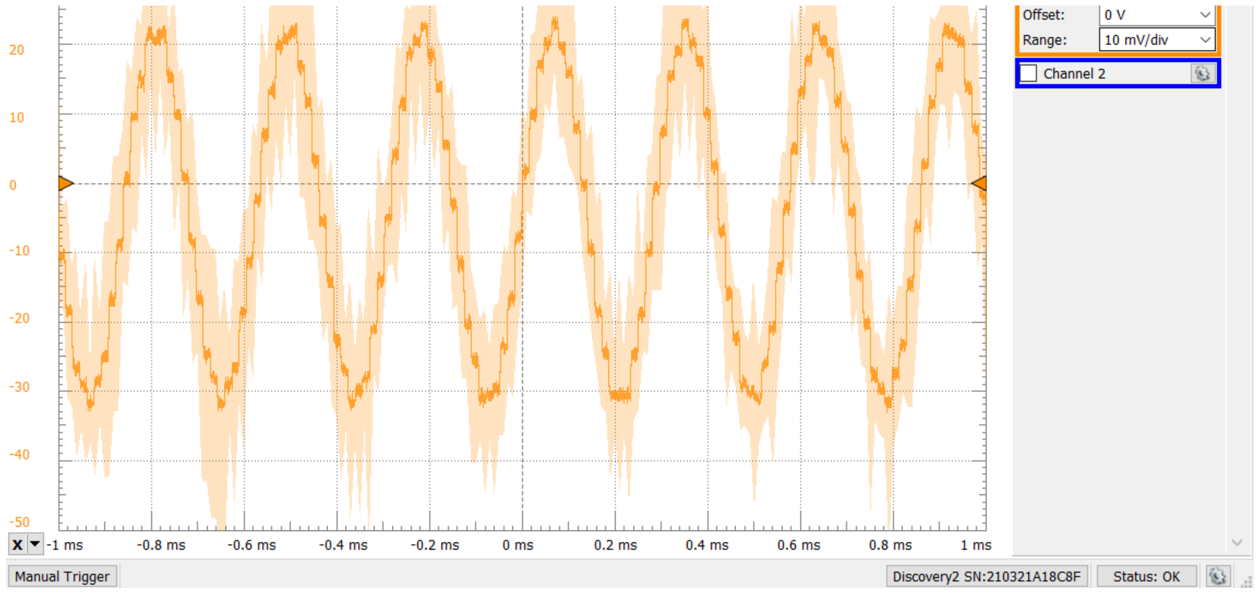
The signal still appeared to generate as expected. The time continuous view showed a sinusoidal signal, and the frequency spectrum helped to verify correct frequency (appears to be very close to the expected value, with a spike at 2.641 kHz). Aliasing was not expected to occur at this sampling frequency for a 2.641 kHz signal. The screenshots help verify correct behavior.

- f = 3500 Hz



Channel: C1
Type: Sample
Count: 100
Discovery2 SN:210321A18C8F
Status: OK

Figure 5 - Frequency Spectrum Captured for 3.5 kHz



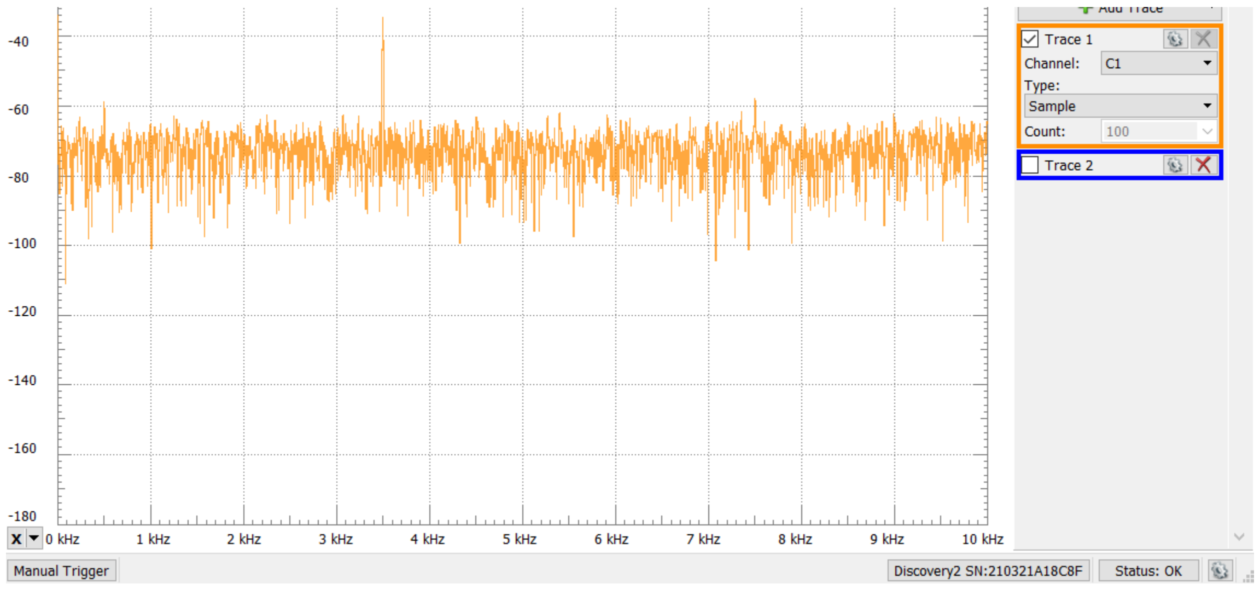
Offset: 0 V
Range: 10 mV/div
Discovery2 SN:210321A18C8F
Status: OK

Figure 6 - Continuous time signal captured for 3.5 kHz

With a sampling frequency of 8 kHz and a signal frequency of 3.5 kHz, the Nyquist Theorem states that aliasing will not occur. This is verified in the screenshots above (sinusoidal signal with a frequency spectrum that spikes at 3.5 kHz). A time continuous signal with an amplitude close to 20 mV and a frequency of 3.5 kHz is captured. This verifies that correct signal generation was able to occur without any effects of aliasing.

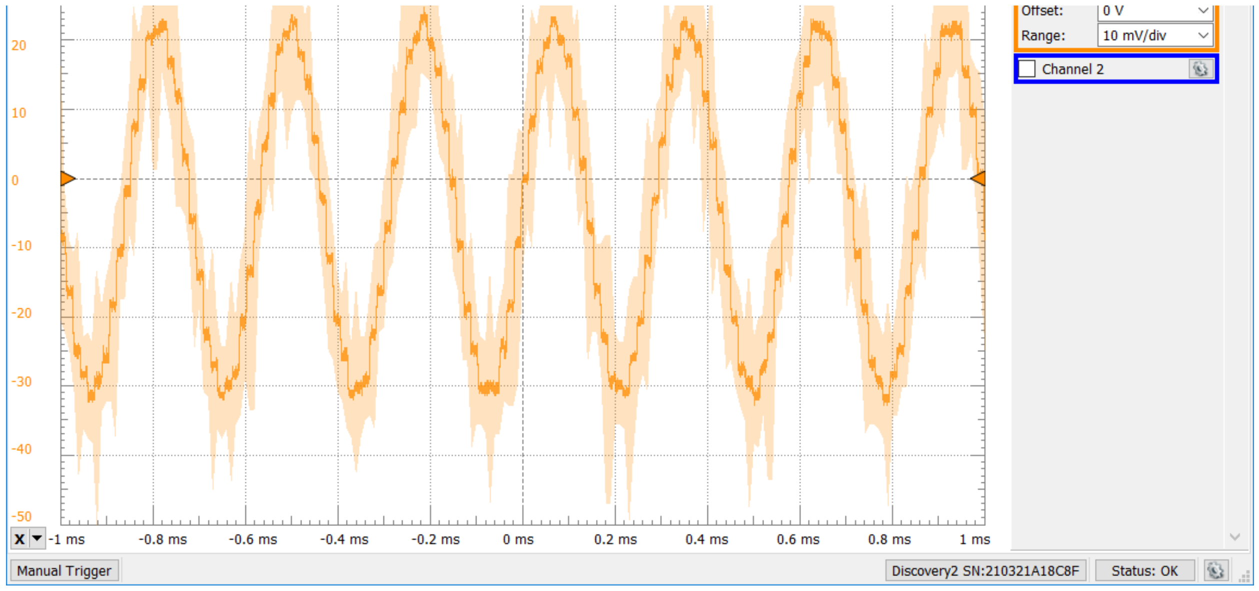
Mitchell Larson
EE 3221
12/7/17
Lab2

- f = 4500 Hz



Trace 1
Channel: C1
Type: Sample
Count: 100
Trace 2
Discovery2 SN:210321A18C8F

Figure 7 - Frequency Spectrum Captured for 4.5 kHz. Note that aliasing is occuring



Offset: 0 V
Range: 10 mV/div
Channel 2
Discovery2 SN:210321A18C8F

Figure 8 - Continuous time signal captured for 4.5 kHz. Note that the actual signal generated is not 4.5 kHz

With a sampling frequency of 8 kHz and a signal frequency of 4.5 kHz, the Nyquist Theorem states that aliasing will occur. While at first glance, the time continuous signal that is generated may look correct, it does not have the correct frequency. The signal being generated should have a frequency of 4.5 kHz, but by using the frequency spectrum view, it can be observed that the signal generated has a frequency near 3.4 kHz. This incorrect behavior is caused by the effects of aliasing since the sampling frequency of 8 kHz is not greater than twice the maximum frequency of the signal (4.5 kHz). The effect of aliasing could be avoided, and the proper signal could be generated if a sampling frequency greater than 9 kHz was used.

- f = 7000 Hz

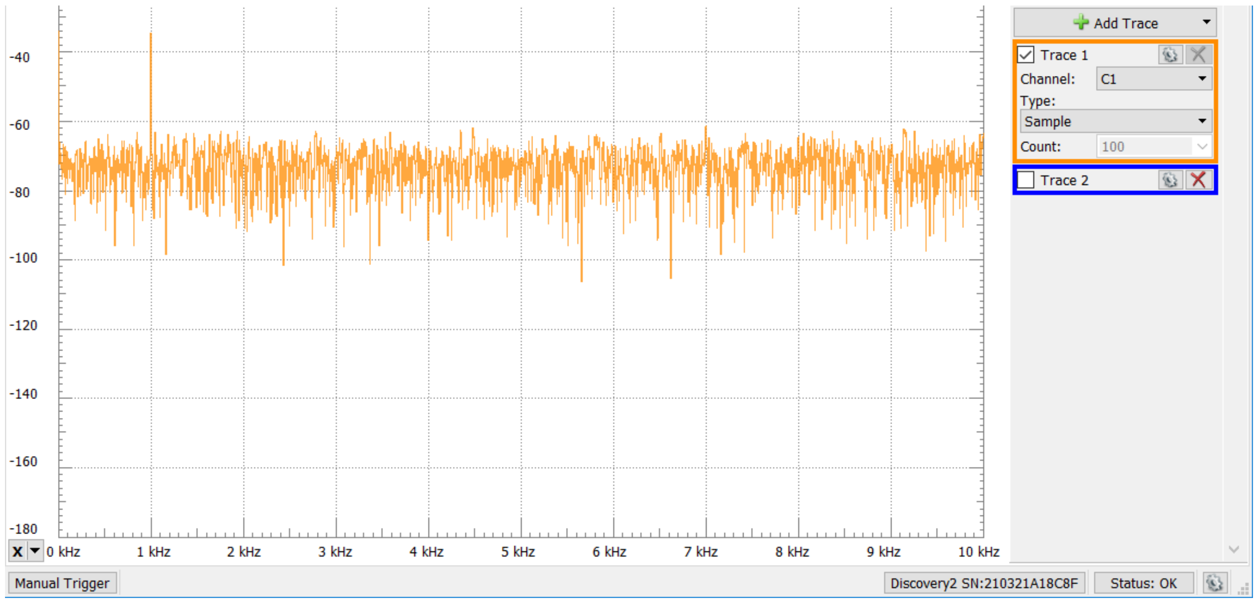


Figure 9 - Frequency Spectrum Captured for 7 kHz. Note that aliasing is occurring

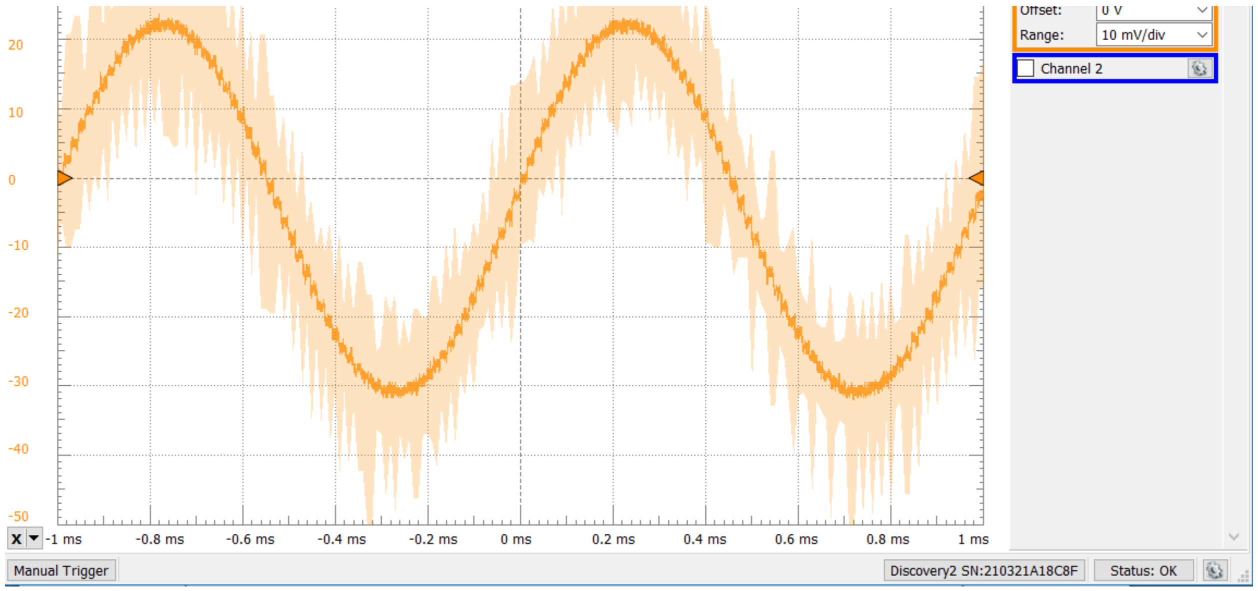
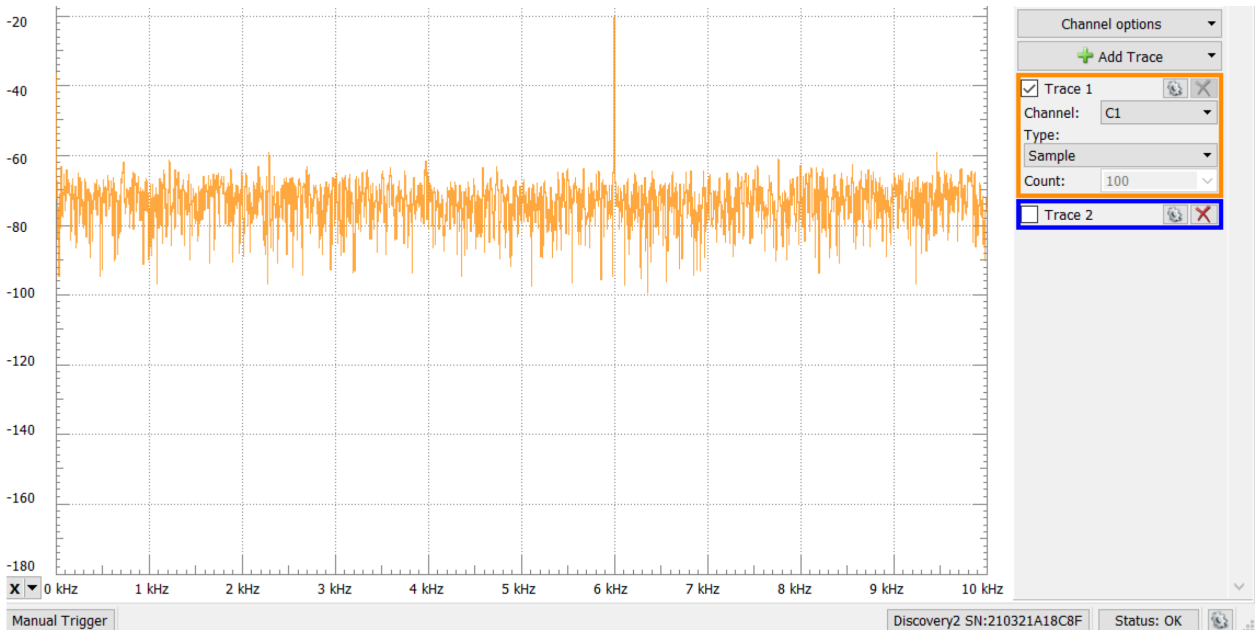


Figure 10 - Continuous time signal captured for 7 kHz. Note that the actual signal generated is not 7 kHz

With a sampling frequency of 8 kHz and a signal frequency of 7 kHz, it was expected that aliasing would occur again. This is exactly what was observed. Based on the previous signal generation and this signal generation, it seemed that the resulting frequency of the generated signal was the difference between the sampling frequency and the actual frequency of the signal. So, in this case, 8 kHz – 7 kHz resulted in a generated signal with a frequency of 1 kHz. To avoid the effects of aliasing, a sampling frequency greater than 14 kHz would need to be used.

4) 48 kHz Sampling Frequency

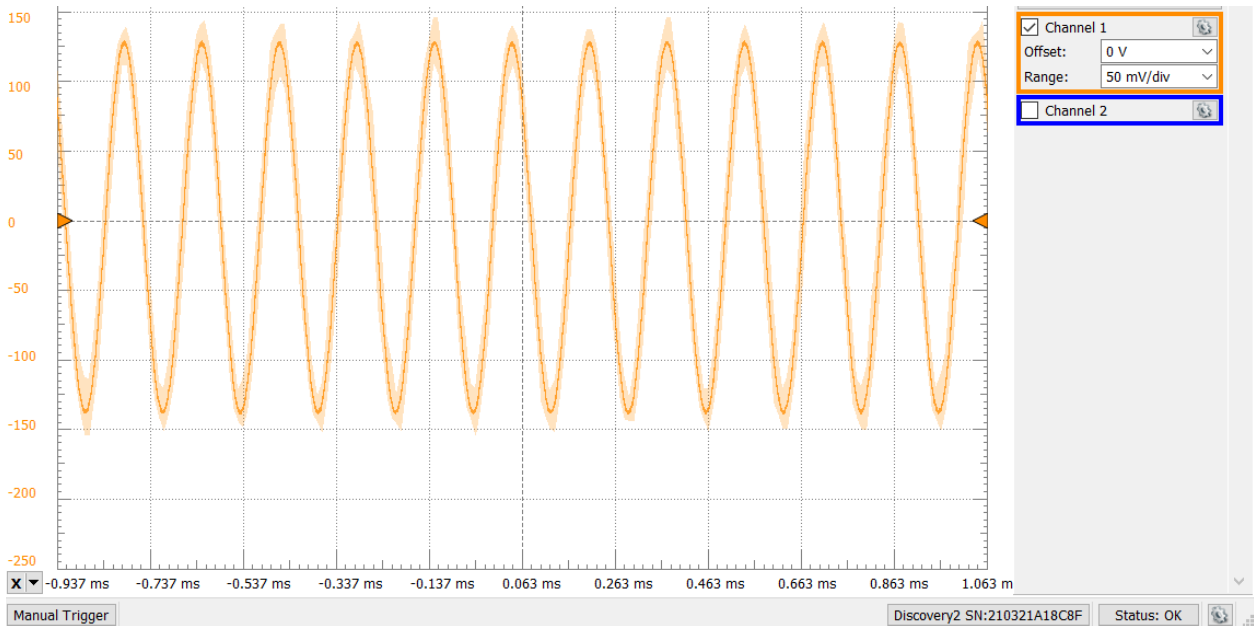
With a sampling frequency of 48 kHz and 8 values in the look up table for the sine function, it was predicted that the generated signal would have a frequency of 6 kHz (fs / table size). After the sampling frequency was changed to 48 kHz and the code was compiled and downloaded to the board, the following oscilloscope and frequency spectrum images were captured.



Trace 1
Channel: C1
Type: Sample
Count: 100
Trace 2
Discovery2 SN:210321A18C8F
Status: OK

Figure 11 - Frequency Spectrum Generated for a signal sampled at 48 kHz and 8 discrete values.

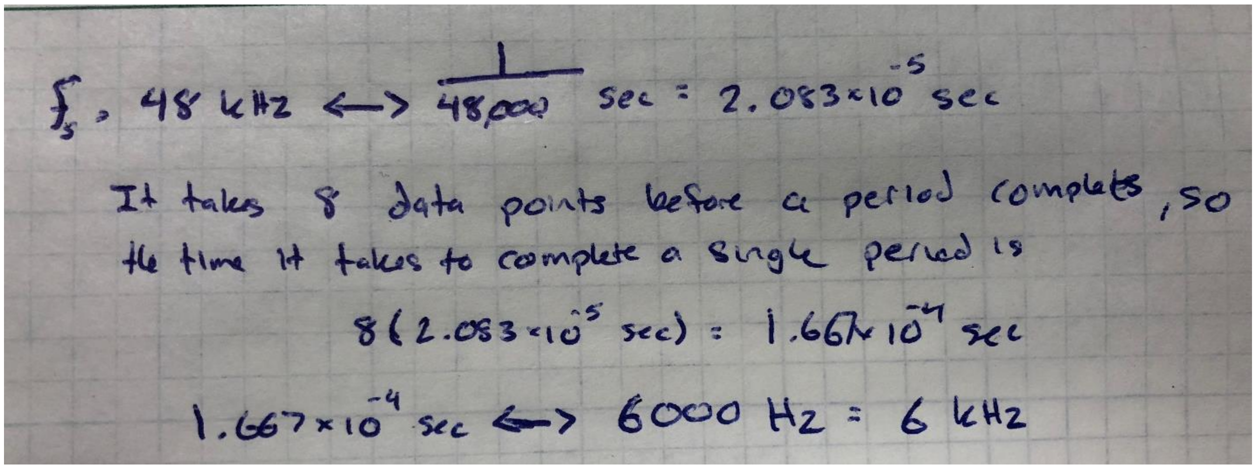
Mitchell Larson
EE 3221
12/7/17
Lab2



Channel 1
Offset: 0 V
Range: 50 mV/div
Channel 2
Manual Trigger
Discovery2 SN:210321A18C8F
Status: OK

Figure 12 - Time Continuous Signal Generated for a signal sampled at 48 kHz and 8 discrete values.

As was predicted, a signal with a frequency of 6 kHz was generated. The reason that this occurred can be explained with the following calculations:



$f_s = 48 \text{ kHz} \longleftrightarrow \frac{1}{48{,}000} \text{ sec} = 2.083 \times 10^{-5} \text{ sec}$

It takes 8 data points before a period completes, so the time it takes to complete a single period is

$8(2.083 \times 10^{-5} \text{ sec}) = 1.667 \times 10^{-4} \text{ sec}$

$1.667 \times 10^{-4} \text{ sec} \longleftrightarrow 6000 \text{ Hz} = 6 \text{ kHz}$

Figure 13 - Calculations for deriving a 6-kHz signal

Mitchell Larson
EE 3221
12/7/17
Lab2

5) Generating a 2 kHz Signal

Using the same logic demonstrated in the previous calculation, a 2-kHz signal could be generated if a sampling frequency of 8 kHz and 4 data points were used. To do this, lines 6 and 7 in the code needed to be changed. The following screen captures show the oscilloscope and frequency spectrum readings, along with the code used.

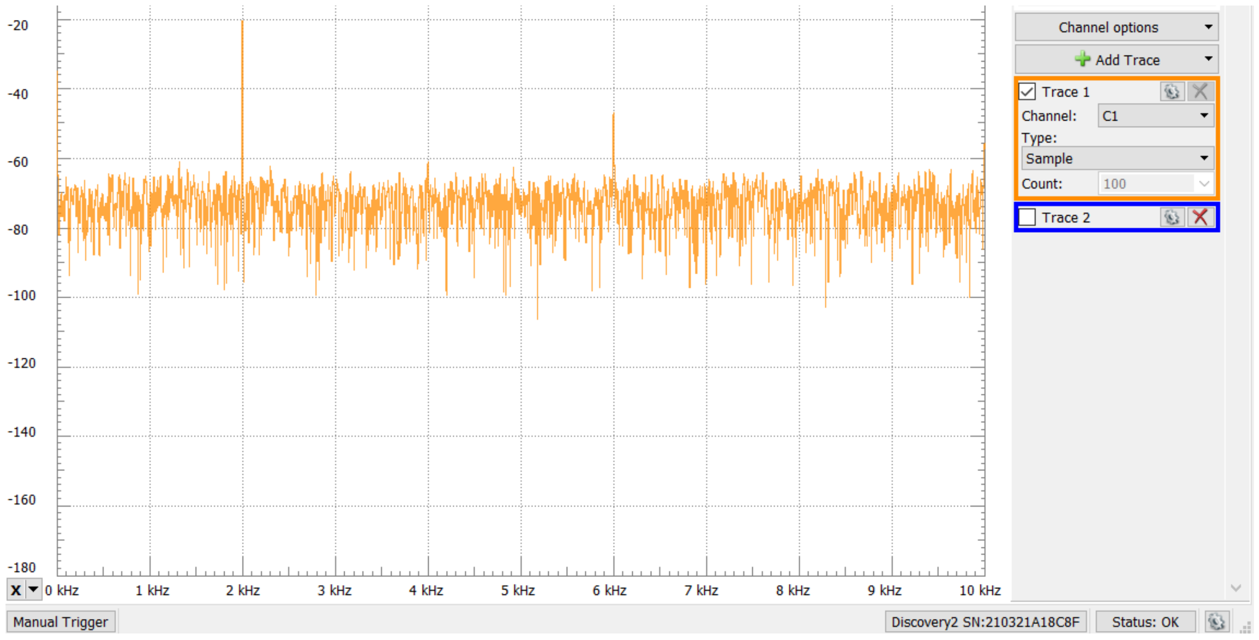


Figure 14 - Frequency Spectrum for the signal generated using fs = 8 kHz and 4 data points.

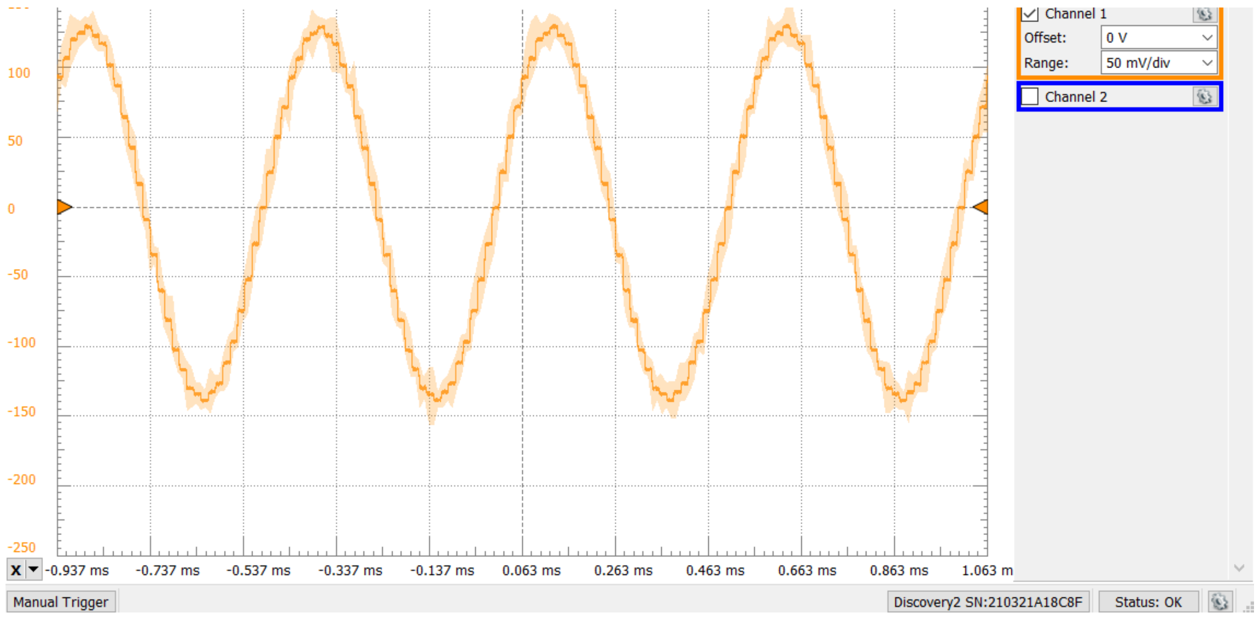
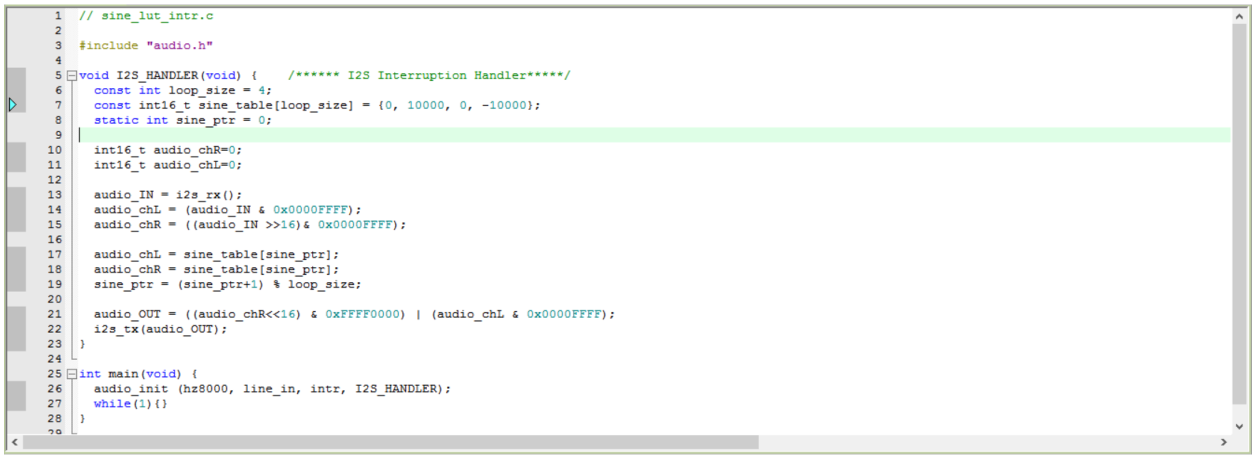


Figure 15 -Continuous Time signal generated using fs = 8 kHz and 4 data points.

Mitchell Larson
EE 3221
12/7/17
Lab2

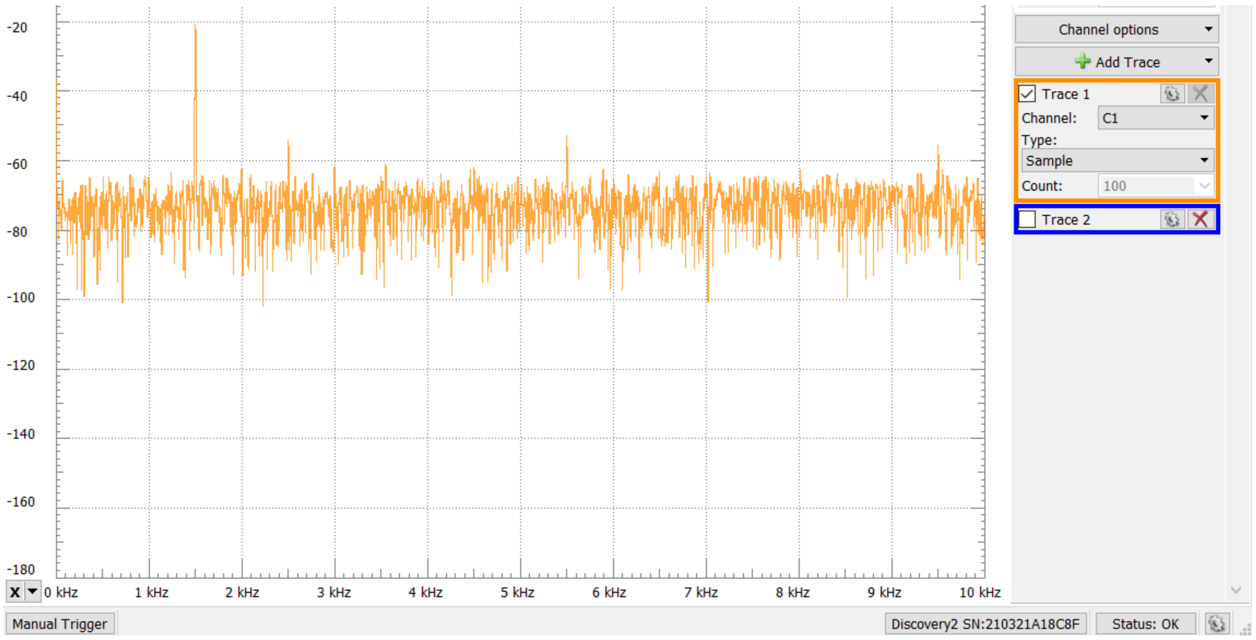


```
// sine_lut_intr.c

#include "audio.h"

void I2S_HANDLER(void) {     /****** I2S Interruption Handler*****/
  const int loop_size = 4;
  const int16_t sine_table[loop_size] = {0, 10000, 0, -10000};
  static int sine_ptr = 0;

  int16_t audio_chR=0;
  int16_t audio_chL=0;

  audio_IN = i2s_rx();
  audio_chL = (audio_IN & 0x0000FFFF);
  audio_chR = ((audio_IN >>16)& 0x0000FFFF);

  audio_chL = sine_table[sine_ptr];
  audio_chR = sine_table[sine_ptr];
  sine_ptr = (sine_ptr+1) % loop_size;

  audio_OUT = ((audio_chR<<16) & 0xFFFF0000) | (audio_chL & 0x0000FFFF);
  i2s_tx(audio_OUT);
}

int main(void) {
  audio_init (hz8000, line_in, intr, I2S_HANDLER);
  while(1){}
}
```

Figure 16 - Code used to generate a signal with a frequency of 2 kHz

6) Generating a 1.5 kHz Signal

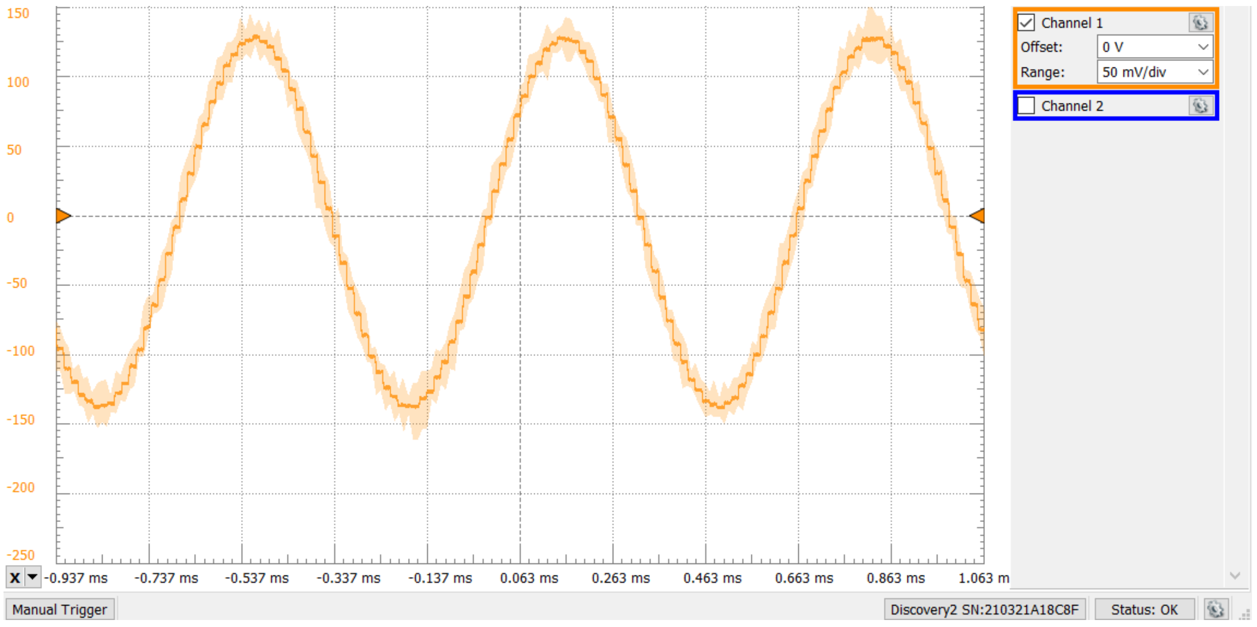
To generate a signal with a frequency of 1.5 kHz, 16 data points were used with a sampling frequency of 8 kHz. The screen captures of the oscilloscope and frequency spectrum are shown below, as well as the code used to generate them.



Channel options
Add Trace
Trace 1
Channel: C1
Type: Sample
Count: 100
Trace 2
Manual Trigger
Discovery2 SN:210321A18C8F
Status: OK

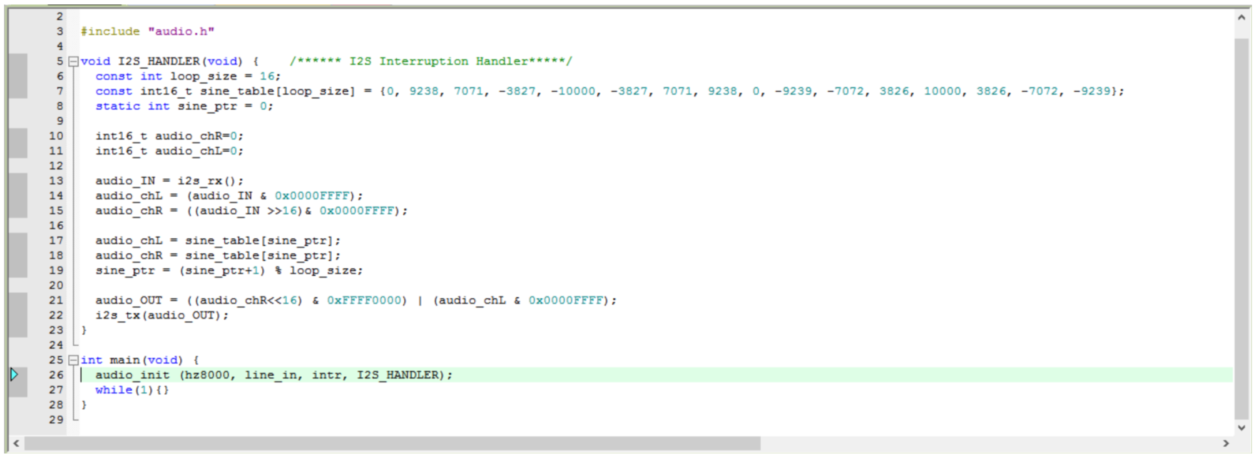
Figure 17 - Frequency Spectrum for the signal generated using fs = 8 kHz and 16 data points.

Mitchell Larson
EE 3221
12/7/17
Lab2



[✓ Channel 1]
Offset: 0 V
Range: 50 mV/div
Channel 2

Figure 18 - Continuous Time signal generated using fs = 8 kHz and 16 data points.

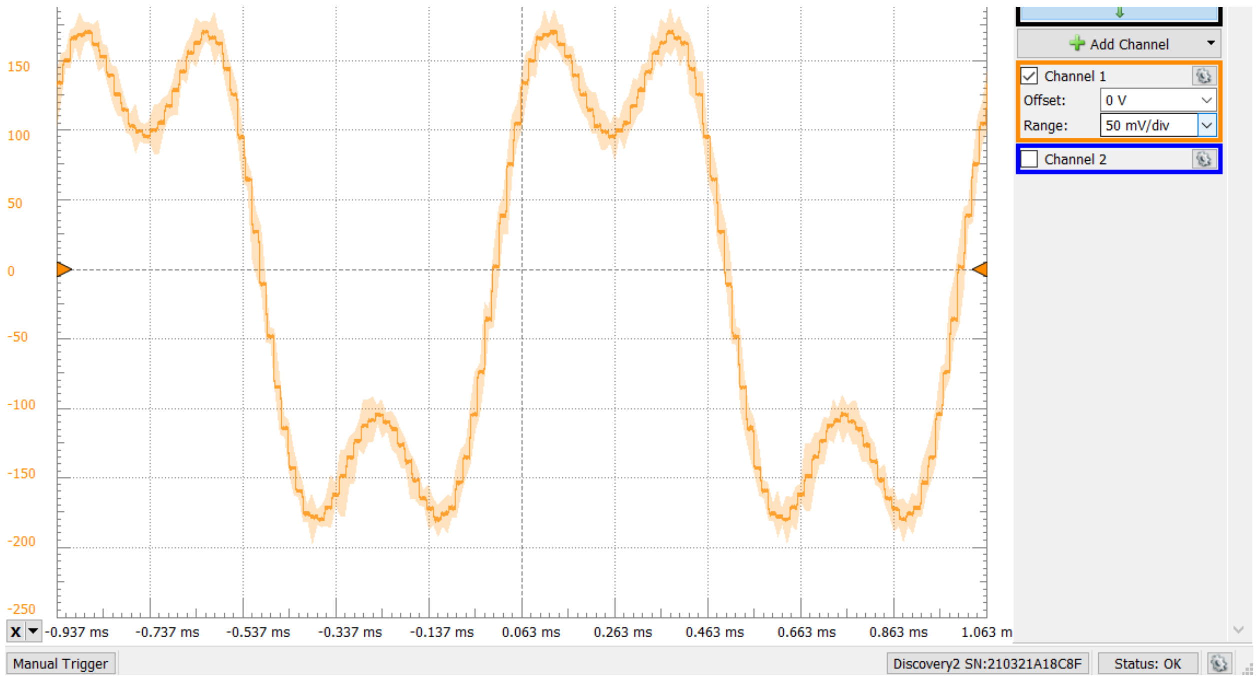


```c

#include "audio.h"

void I2S_HANDLER(void) {     /****** I2S Interruption Handler*****/
  const int loop_size = 16;
  const int16_t sine_table[loop_size] = {0, 9238, 7071, -3827, -10000, -3827, 7071, 9238, 0, -9239, -7072, 3826, 10000, 3826, -7072, -9239};
  static int sine_ptr = 0;

  int16_t audio_chR=0;
  int16_t audio_chL=0;

  audio_IN = i2s_rx();
  audio_chL = (audio_IN & 0x0000FFFF);
  audio_chR = ((audio_IN >>16)& 0x0000FFFF);

  audio_chL = sine_table[sine_ptr];
  audio_chR = sine_table[sine_ptr];
  sine_ptr = (sine_ptr+1) % loop_size;

  audio_OUT = ((audio_chR<<16) & 0xFFFF0000) | (audio_chL & 0x0000FFFF);
  i2s_tx(audio_OUT);
}

int main(void) {
  audio_init (hz8000, line_in, intr, I2S_HANDLER);
  while(1){}
}

```

Figure 19 - The code used to generate a 1.5 kHz signal

Mitchell Larson
EE 3221
12/7/17
Lab2

7) Time-Domain Representation of the Square Wave

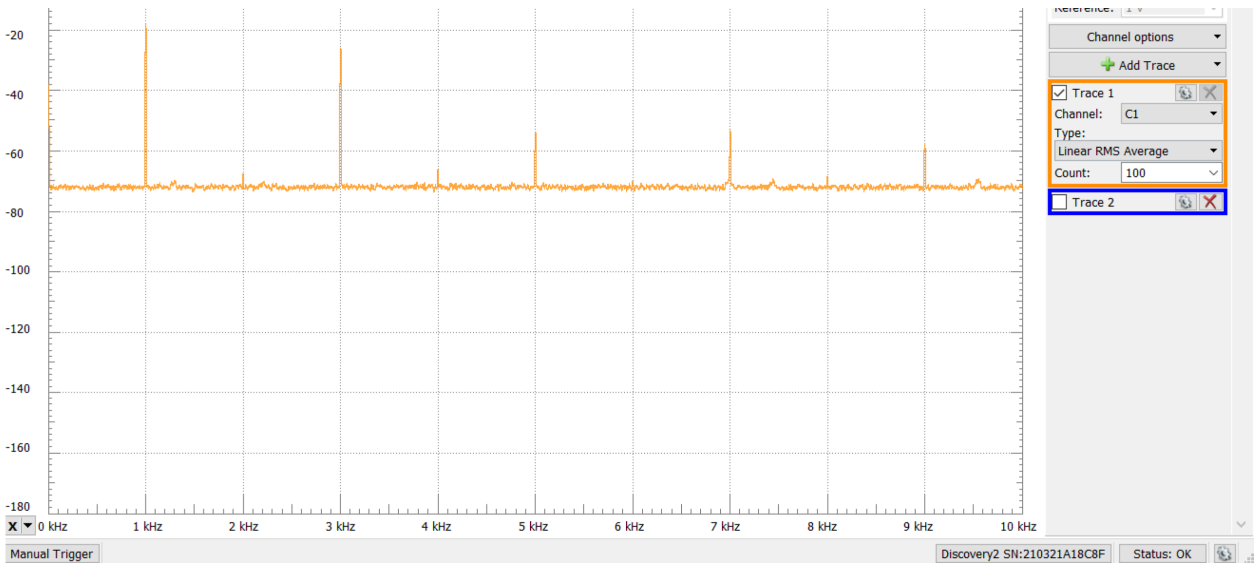


Channel 1
Offset: 0 V
Range: 50 mV/div
Channel 2
Discovery2 SN:210321A18C8F

The time domain representation of a square wave is not completely a square. This is because the signal is generated using a combination of sine waves. This is the reason that the square wave takes on a curved shape. The more sine waves used to generate the signal, the closer the signal would come to a true square wave. This differs from the Matlab version of the square wave because MatLab can use discrete values of 1 and 0 to generate the square wave. This isn't possible in real life, and sine waves must be used instead. This explains the visual difference seen when generating a square wave in MatLab vs real life.

Mitchell Larson
EE 3221
12/7/17
Lab2

8) Frequency-Domain Representation of the Square Wave



Channel: C1
Type: Linear RMS Average
Count: 100
Discovery2 SN:210321A18C8F
Status: OK

The frequency domain view of the square wave reinforces the statement in which I said that the square wave was made of several sine waves. In particular, the square wave is made of several sine waves with differing frequencies. This is illustrated in the image above. There are several spikes in the frequency spectrum with varying heights. These are all sine waves of differing frequencies and amplitudes which construct the square wave.

The Fourier transform of a square wave is $2A\tau\frac{\sin(\Omega\tau)}{\Omega\tau}$, where A is the height of the square wave, τ is the cutoff frequency of the square wave, and Ω is the frequency parameter. The square wave is thus a combination of these sine waves with frequencies of 1 kHz, 3 kHz, 4 kHz, 5 kHz, 7 kHz, and 9 kHz. The addition of all these sine wave constructs a signal that closely resembles a square wave.