Task 1 – Sampling in Arduino

Background 1: Fixed Signal, Varied Sampler

Sampling Theorem :
If a function $f(t)$ contains no frequencies higher than W cycles per second, it is completely determined by giving its ordinates at a series of points spaced $\frac{1}{2W}$ seconds apart.

— *From Communication in the Presence of Noise, 1949,*
Claude E. Shannon
https://ieeexplore.ieee.org/document/1697831

If we interpret this theorem in modern terminology:

- The term “cycles per second” is now referred to as **Hertz (Hz)**.
- The “points spaced $\frac{1}{2W}$ seconds apart” is now referred to as **Sampling Period**.

So, the original theorem in year 1949 can be rephrased as:

*If a signal $f(t)$ contains no frequencies higher than W Hz, it can be perfectly reconstructed from its samples taken at a rate of at least $2W$ samples/second. This rate is known as the **Nyquist rate**.*

Background 2: Varied Signal, Fixed Sampler

The previous theorem considers a setting where the signal is fixed, and the sampling rate is varying.

If we reverse it—keeping the sampling rate fixed while changing the signal—we can extend to a concept of **Nyquist Frequency**.

The Nyquist frequency is the highest frequency that can be accurately represented in a sampled signal without aliasing.

$$f_{Nyquist} = \frac{f_s}{2}$$

where f_s is the **Sampling Frequency**. The $f_s = \dfrac{1}{\text{Sampling Period}}$.

Background 1 and Background 2 are essentially the same.
Background 1 focuses on the design perspective for a good sampler, while Background 2 addresses the limits necessary to prevent aliasing among different signals.

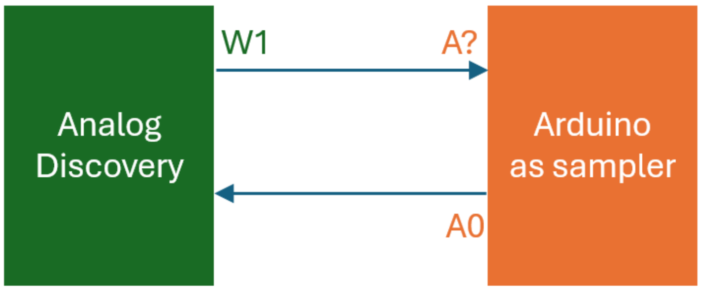
1. Set-up

In the task, we will use the Background 2 setting – Varied Signal, Fixed Sampler.

We use Analog Discovery to set signals at different frequencies. We will use the Arduino as a fixed "sampler". We will observe the aliasing effect on Analog Discovery's Scope.

To do so:

- Use Analog Discovery to generate a signal.
- Use Arduino to receive the analog signal from Analog Discovery.
- Arduino will sample the signal in code.
- Then Arduino will send the sampled signal back to Analog Discovery for measurement.
- Use 2 Scopes to observe the original signal (send-out) and the sampled signal (send-back).



First, try to connect the wires only with the previous description. Then use the PIN table to double-check:

Analog Discovery	Arduino
Pin 1+, Pin W1	Pin A?
Pin 2+	Pin A0
Pin ⏚, Pin 1-, Pin 2-	Pin G

The pins in each rows must be connected. Pin A? can be any analog pin.

2. Determine the Property of The Sampler

We use the simple code to set the Arduino as a sampler.

Arduino Code

```
void setup() {
}

void loop() {
analogWrite(A0,analogRead(A?));
delayMicroseconds(1000);
}
```

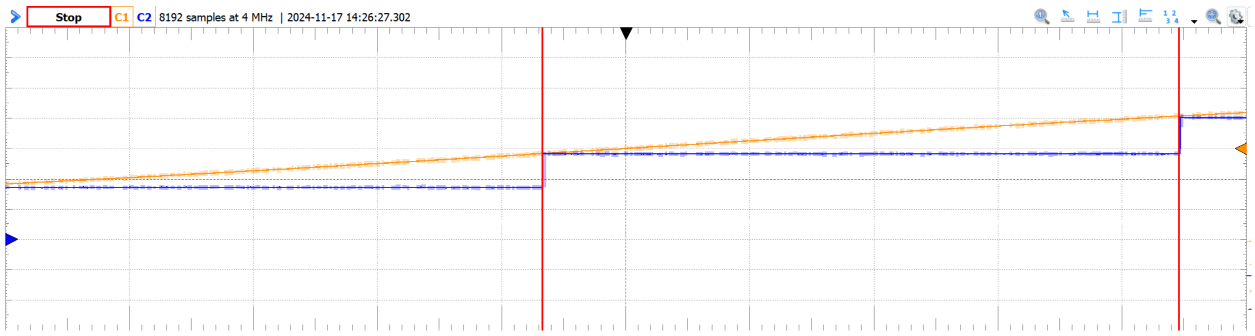
Report Item 1-a

Provide comments for the two line code inside void loop(). Each in one sentence.

Open Wavegen, set the input signal as such,

Type = Sine	Frequency = 100 Hz
Amplitude = 1 V	Offset = 1.5 V
Symmetry = 50%	Phase = 0

Adjust the Scope display, until you can see a clear step in the sampled signal.



Precisely measure the horizontal distance (ΔX) of the step to 4 significant digits. This is the sampling period. (The next part result will go off if you don't measure precisely here.)

Then calculate the sampling frequency and Nyquist frequency based on Background 2.

Report Item 1-b

Measured sampling period = ??

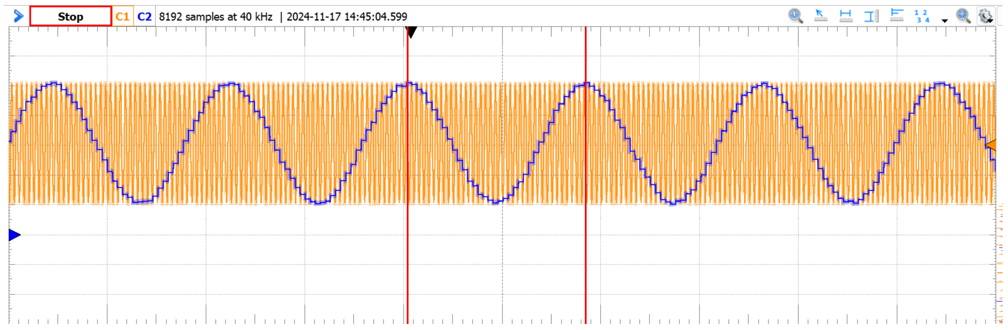
Sampling frequency f_s = ??

Nyquist frequency $f_{Nyquist} = \dfrac{f_s}{2} = ??$

3. Measure at Different Freqs

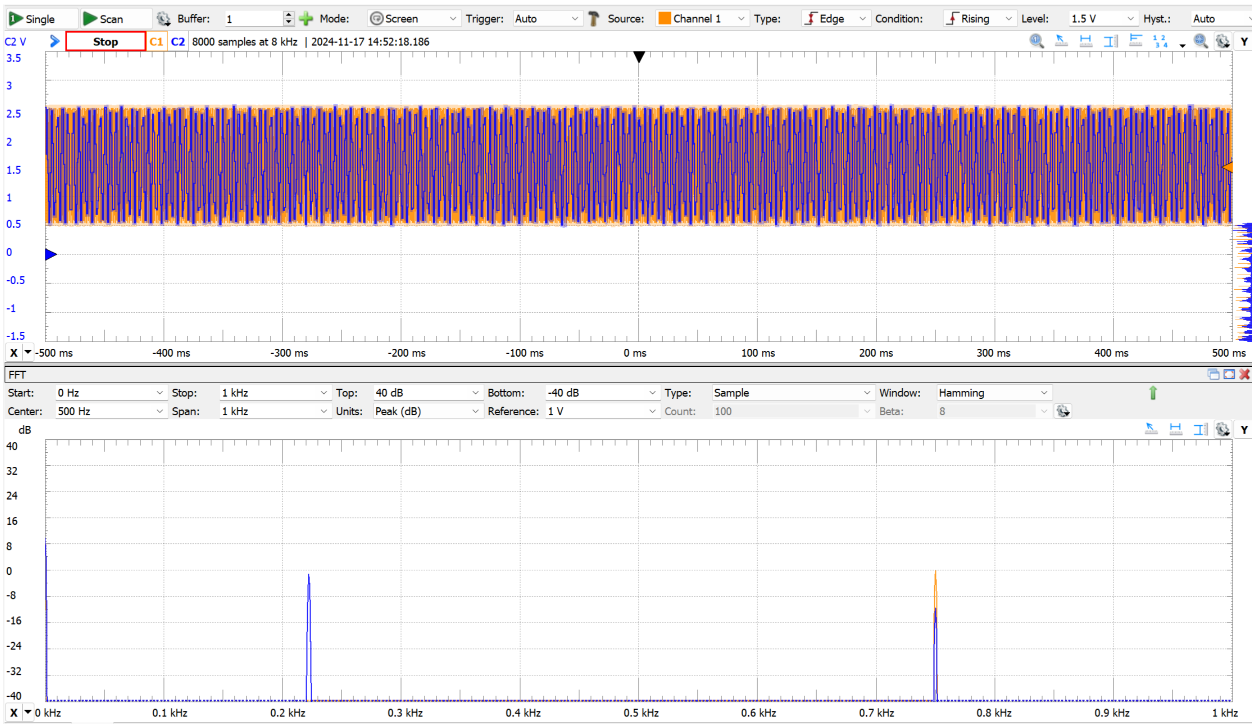
Keep the Arduino running. Now, we will vary the previous Wavegen sine signals at different freqs. We will measure the aliased freqs of the sampled signals on the Scope.

Case 1:
At certain input frequencies, a distinct low-freq sine wave (blue in the example) may appear on the scope. In such case, directly use Scope cursors to measure aliased freqs.


Stop C1 C2 8192 samples at 40 kHz | 2024-11-17 14:45:04.599

Case 2:
At some other input frequencies, it is hard to eye-ball the sine wave. Use the FFT tool to identify the aliased freq.

Start = 0 Hz	Stop = 2 kHz
Top = 20 dB	Bottom -40 dB
Type = Sample	Window = Flat Top
Units = Peak (dB)	Reference = 1 V


Single Scan Buffer: 1 Mode: Screen Trigger: Auto Source: Channel 1 Type: Edge Condition: Rising Level: 1.5 V Hyst.: Auto

Stop C1 C2 8000 samples at 8 kHz | 2024-11-17 14:52:18.186

FFT

Start: 0 Hz Stop: 1 kHz Top: 40 dB Bottom: -40 dB Type: Sample Window: Hamming

Center: 500 Hz Span: 1 kHz Units: Peak (dB) Reference: 1 V Count: 100 Beta: 8

In the FFT tool, the peak closest to 0 dB represents the primary frequency of the signal.

You may need to adjust the Time Base to obtain clear peaks in FFT, as the FFT analysis is based on the datapoints visible in the Scope display.

In the example picture, the orange is the original signal, the blue is the sampled signal. The sampled signal introduces a new low-freq peak that appears very close to 0 dB. We can measure this peak's X and record it as the aliased frequency in the table.

Recall: if the peak is 0 dB (1 V reference), then ampilitude for this specific freq is $10^{\frac{0}{20}} * 1\text{V} = 1\text{V}$.

Note: Also at some freqs, you may see 2 or more new low-freq peaks. This is because the sampled signal is actually a step-like signal, not true sine. A step-like signal has higher harmonics. These higher harmonics result to multiple new peaks.
You only need to record the highest peak. This is the main freq.

Report Item 1-c

Complete the table.
The f_s is from your Item 1-b.
In the "Original Freq" columns, you need to write down the actual numerical frequencies, rather than leaving $0.2f_s$, $0.4f_s$, ... in table.
The "Tool" columns helps you on how to perform the measurements You don't need to include "Tool" columns in the report.

Original Freq	Aliased Freq	Tool	Original Freq	Aliased Freq	Tool
$0.2f_s$			$1.2f_s$		
$0.4f_s$			$1.4f_s$		
$0.5f_s$		FFT	$1.5f_s$		FFT
$0.6f_s$			$1.6f_s$		
$0.8f_s$			$1.8f_s$		
$0.95f_s$			$1.95f_s$		
f_s		Scope Cursor	$2f_s$		Scope Cursor
$1.05f_s$			$2.05f_s$		

Report Item 1-d

Take a screenshot of your scope, either at the original frequency f_s or $2f_s$. The scope should display both original signal and sampled signal.
Make sure that the local time is included.

As you may notice in the screenshot, the high-frequency signal has been aliased to a very low frequency, which is slow enough even for the human eye to track. This is the same phenomenon demonstrated in the prelab videos.

4. Data Analysis

Report Item 1-e

In Python, plot the table data in Item 1-c.
Use Original Freq as X axis. Use Aliased Freq as Y axis.

Use `plt.plot(Xarray, Yarray, marker="o")` to display datapoints as markers along the plotted line.
Also visualize the Nyquist freq from 1-b in the Y-axis. Use `plt.axhline(y=fNyquist, color='red', linestyle='--', label='Nyquist freq')`

You may notice that, in such a plot, frequencies appear to be "folded" or "reflected by a mirror."

Check Point 1 – Final Python plot

Report Item 1-f

Based on your 1-e plot, try to infer a conclusion:

Consider an original signal with a frequency f_{orig}.
It goes thru a sampler in Nyquist freq $f_{Nyquist}$.

Suppose $f_{\text{orig}} \div f_{Nyquist} : N$ remainder f_r,
where N is the integer quotient and f_r is the remainder.

What would be the expression for the aliased frequency f_{aliased}?

You can formulate the conclusion based on whether N is odd or even.

Task 2 – Create a “Strange” Piano Music

Watching Wavegen Scope might be boring. In this task, we play with some piano music and see the sampling and aliasing work.

You will process a piece of piano music by applying simple sampling technique. With frequency aliasing, some tones may change and eventually lead to a “strange” music.

Quick background:

Each key on the piano produces a distinct frequency, making it suitable for signal analysis.

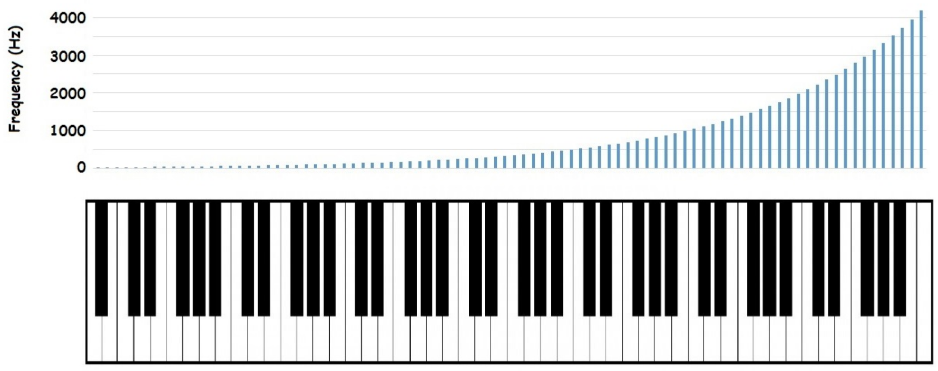


image source: Pressbooks – Understanding Sound- Chapter 12.

Open the provided “Lab11 Support.ipynb” file and copy its code into your Jupyter Notebook.

Quickly view the code to familiarize yourself with its components. The key parts include:

- Use `scipy.io` to process a .wav type music file.
- Use `scipy.signal` to analyze the music signal.

Also download the given “.wav” file into the correct location.

Run the code cell by cell. You should get a new .wav music file. Listen and compare the two music. Do you hear any difference?

Report Item 2-a

Edit the code, so that:

- It can output the value of Nyquist freq.
- It can visualize the Nyquist freq in the final plot (original VS sampled plot)

Now, it's time to process your own music file.

Find a .wav file of piano music. You can use any piano piece downloaded online. For example, you can explore the .wav files available on Wiki Commons at Piano Music Category: `https://commons.wikimedia.org/wiki/Category%3APiano_music`.

First, visualize the freq over time in the original music.

Then, determine a sampling frequency that causes part of the musical note freq to be lowered.

After sampling, you should notice that the new music sounds strange.

Report Item 2-b

Provide the following details:

- the name of the piano music, the composer, the source link.
- Sampling freq and Nyquist freq that you used.

Visualize the original signal freq, sampled signal freq, and Nyquist freq on the same figure.

Check Point 2 – Final Python plot

Lab Report

Please complete the report individually. The submission in BlackBoard must be in .pdf.

The total score for the report is 20 points.

- Format. (3 points)
- Item 1-a, 1-b, 1-c, 1-d, 1-e, 1-f (1+1+2+1+2+2 points)
- Item 2-a, 2-b (1+4 points)
- Conclusion. (3 points)