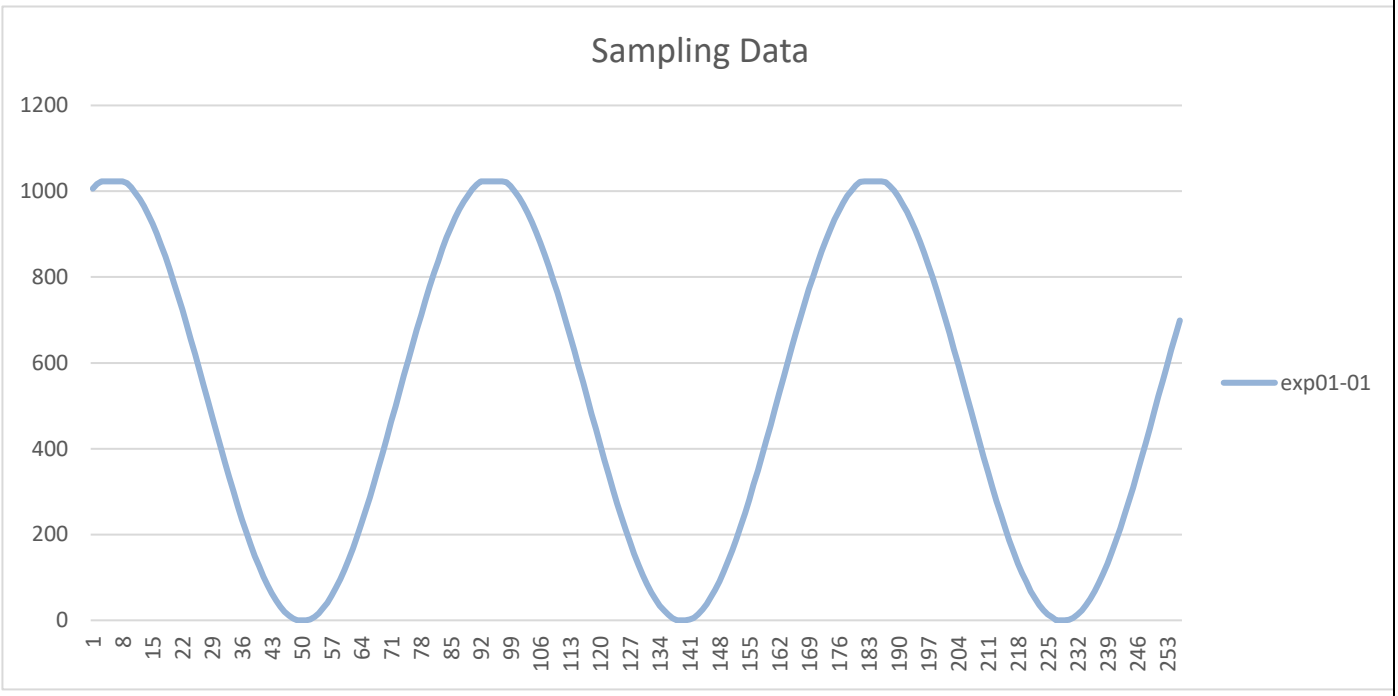
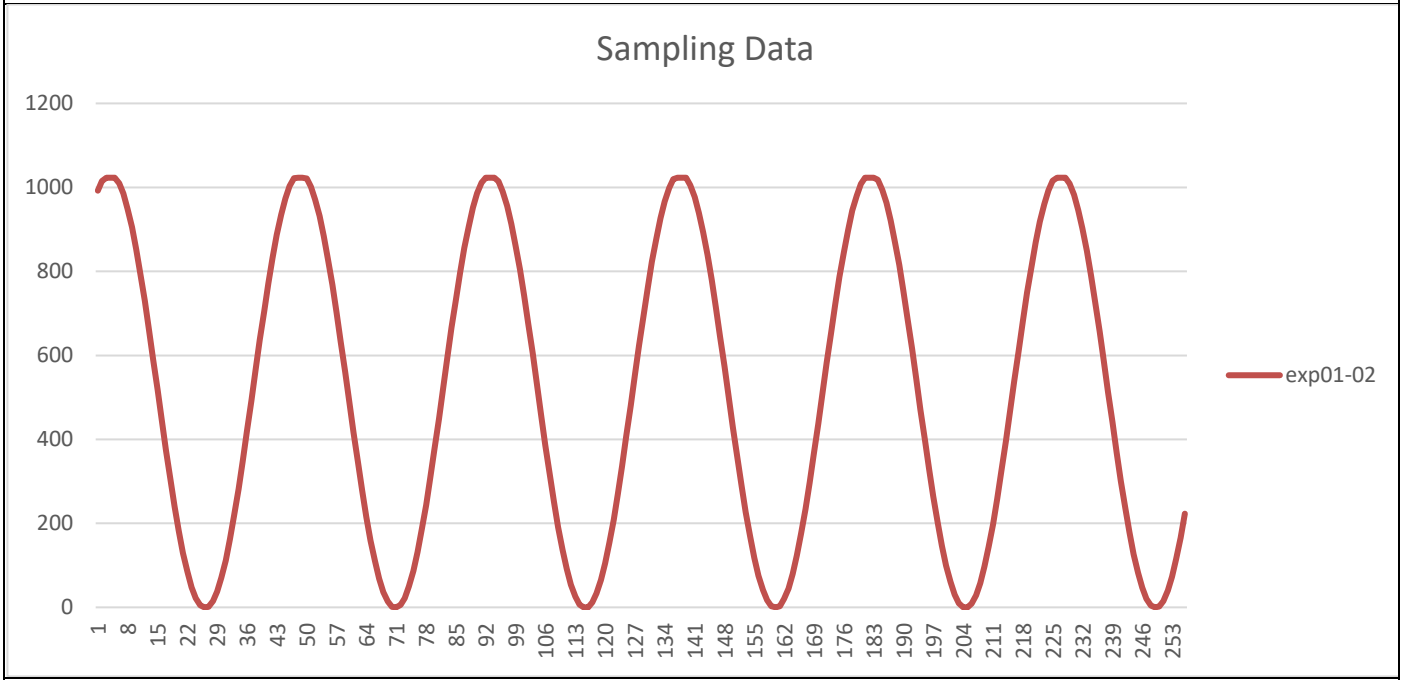
REPORT

Experiment 1: Sampling rate estimation and aliasing

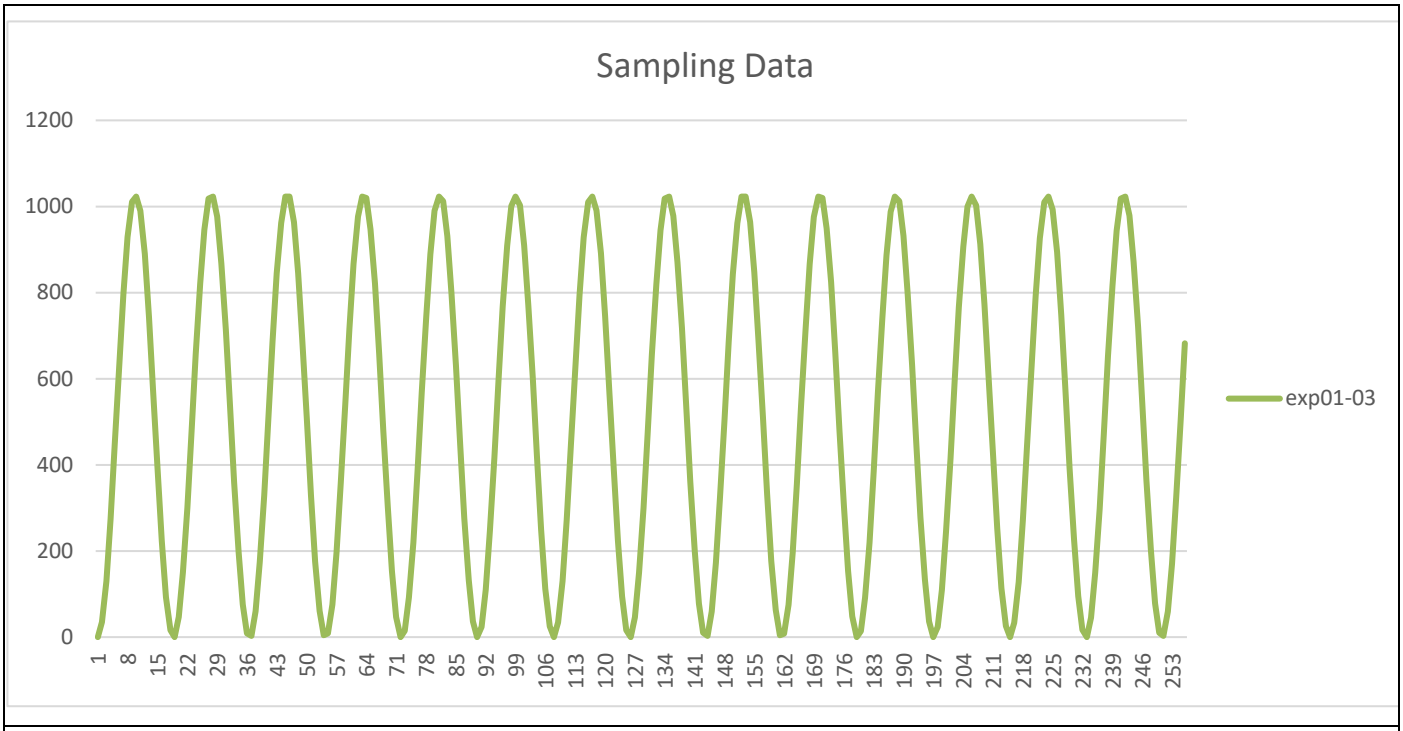
exp01-01 Sampling Data plot



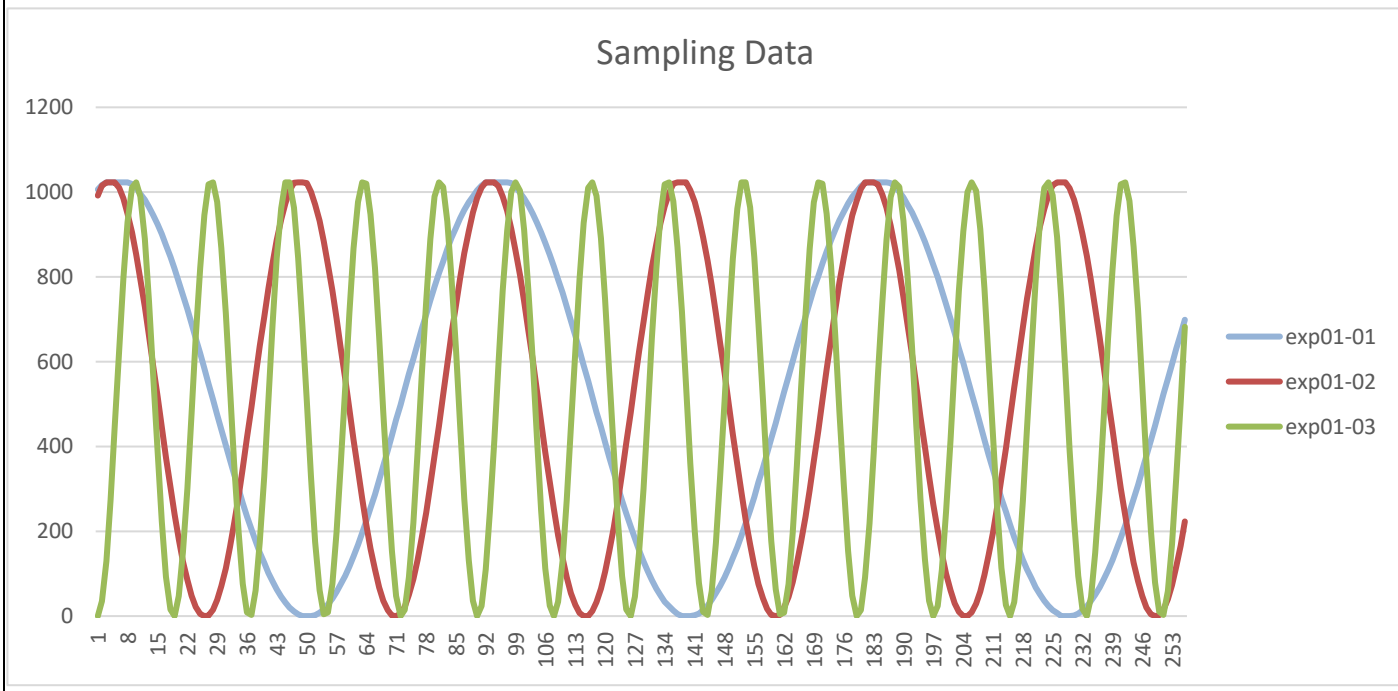
exp01-02 Sampling Data plot



exp01-03 Sampling Data plot



exp01-01 + exp01-02 + exp01-03 Sampling Data plot



Sampling Rate Estimation (sample / second)			
exp01-01	exp01-02	exp01-03	Average
9000	9400	10000	9466.7

What is aliasing?

Aliasing is a phenomenon that occurs when a high-frequency signal becomes indistinguishable from a low-frequency signal when sampled at a certain rate, specifically at a rate that is less than twice the highest frequency component of the original signal.

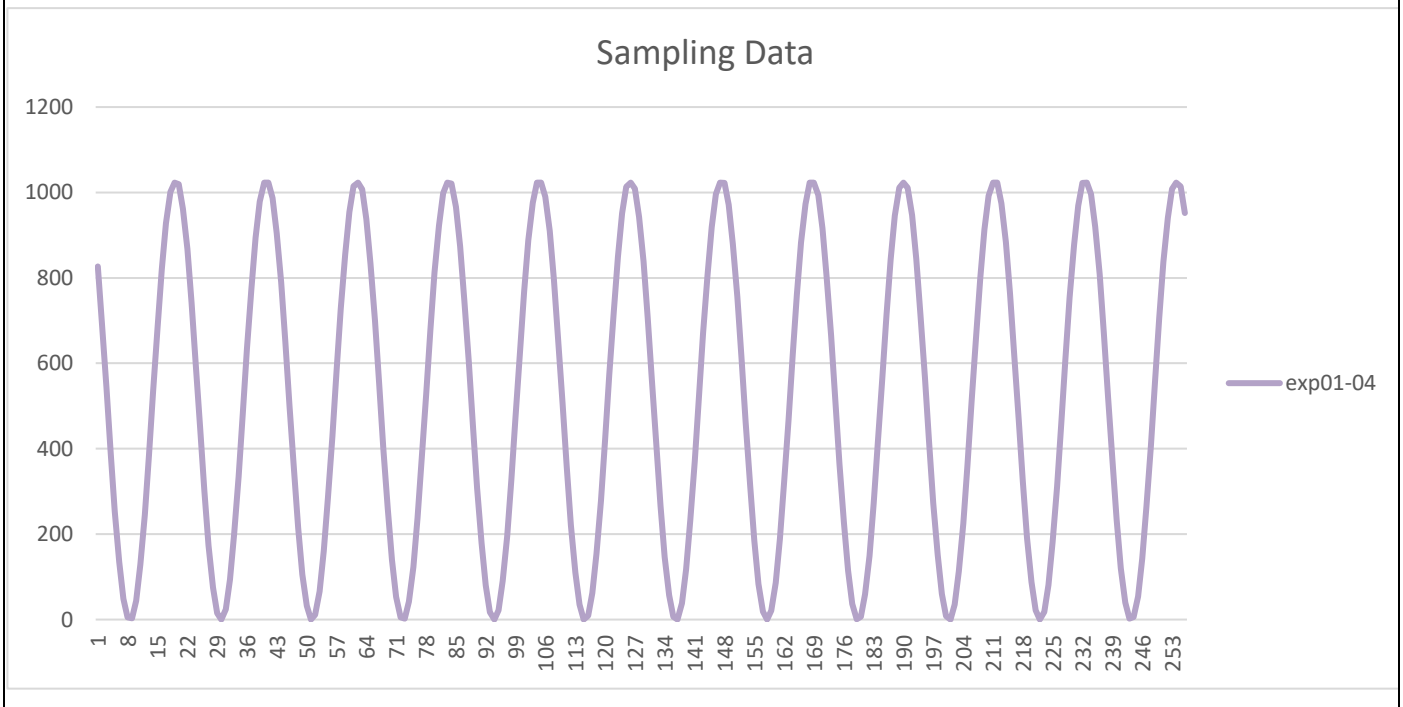
It's a kind of distortion that happens when the Nyquist-Shannon sampling theorem isn't followed. This theorem states that a signal must be sampled at least at twice its highest frequency to be properly represented in a digital format. Distorted signals will look like the ones in exp01-04 to exp01-06. The reconstructed wave will differ radically from the original signal.

What is the Nyquist frequency?

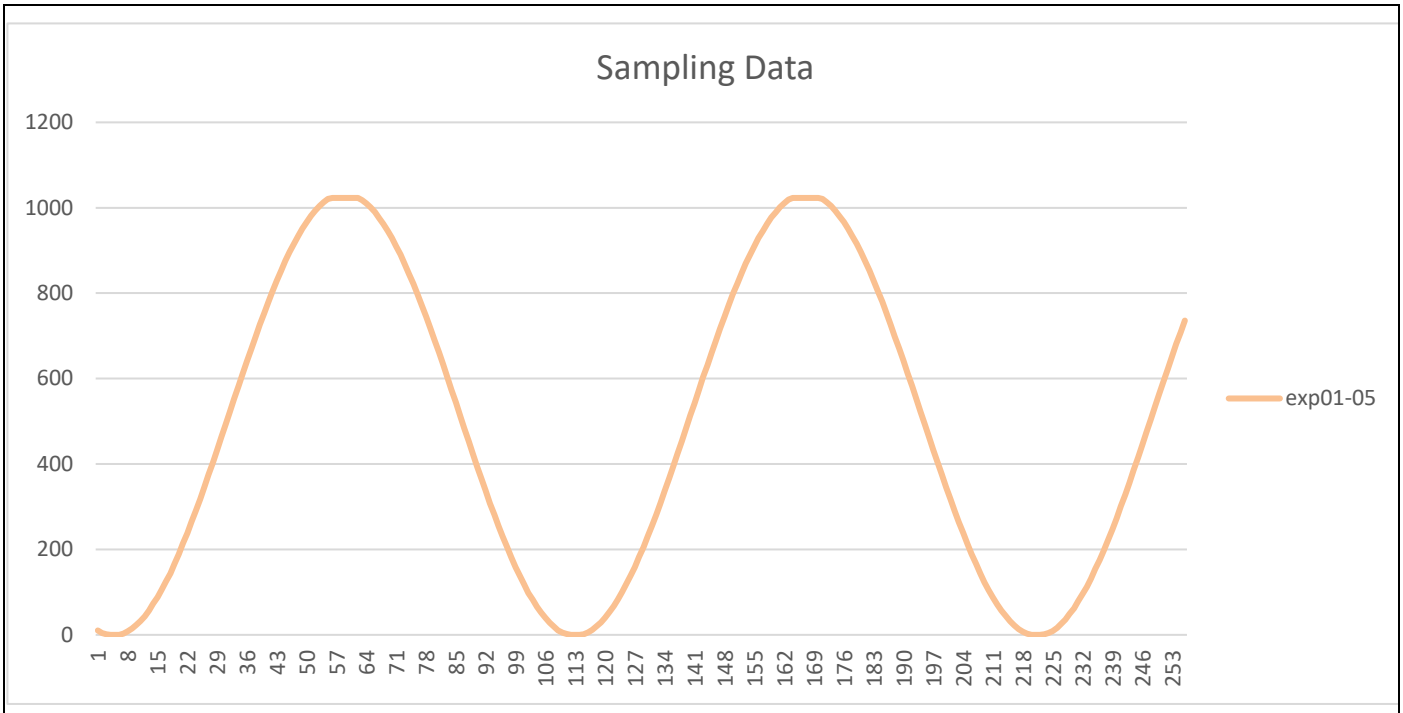
The Nyquist frequency is defined as half of the sampling rate of a digitized signal. For instance, if a continuous-time signal is sampled at 1000 Hz (i.e., a sampling rate of 1000 samples per second), the Nyquist frequency is 500 Hz. The Nyquist frequency acts as a boundary in the frequency domain. Frequencies above the Nyquist frequency are indistinguishable from lower frequencies in a discretely sampled signal due to aliasing.

Note: paste plot in the excel file

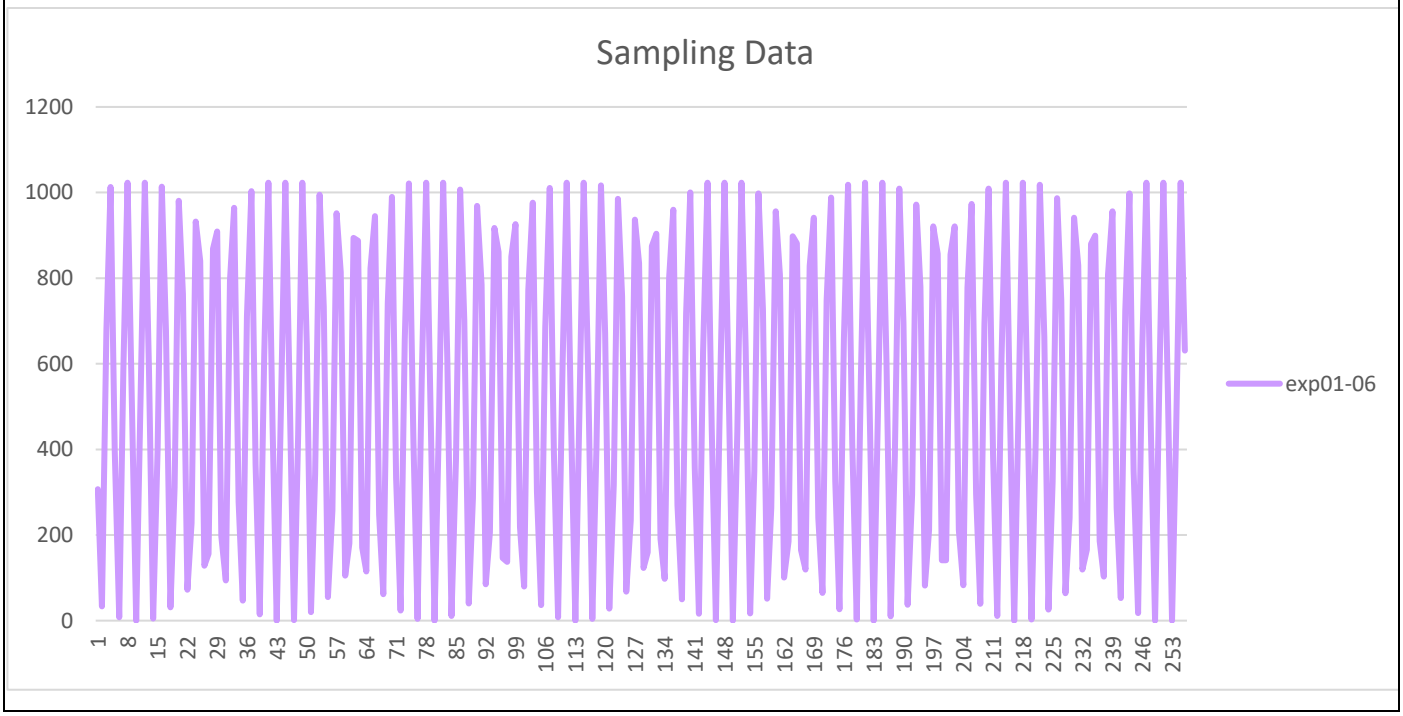
exp01-04 Sampling Data plot



exp01-05 Sampling Data plot

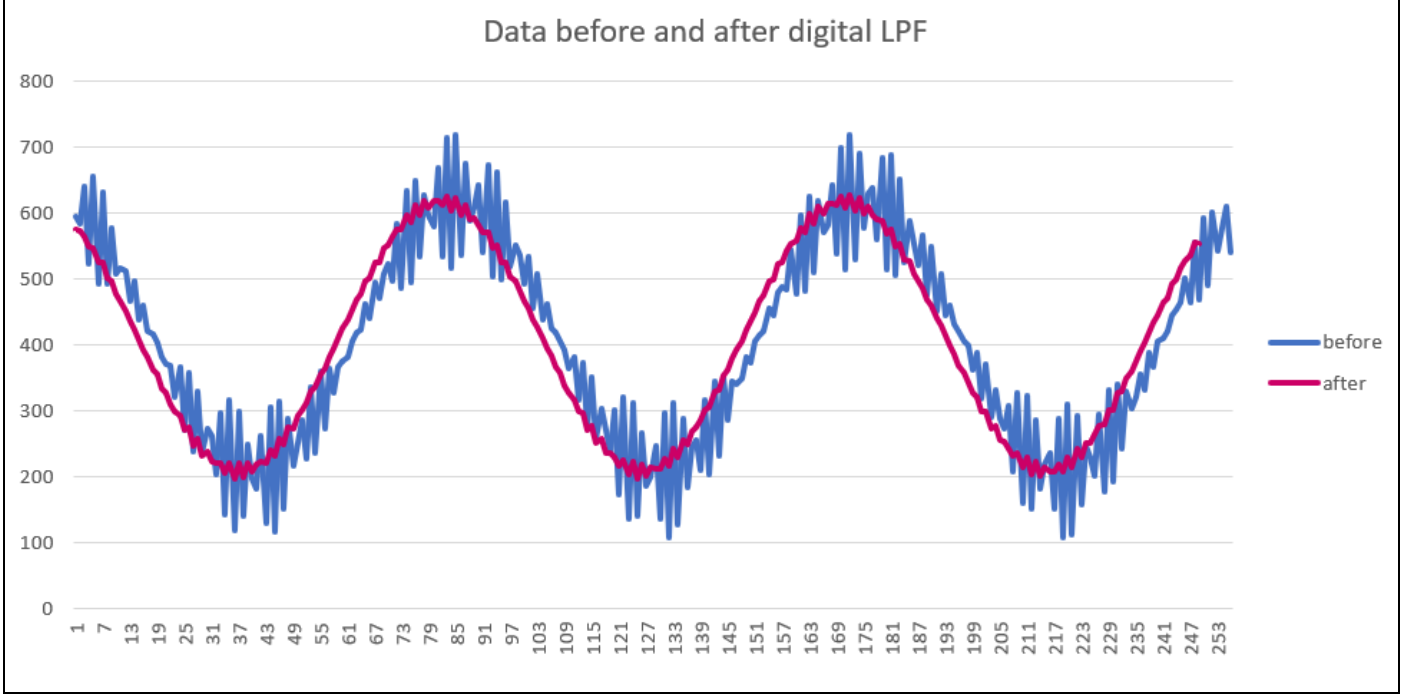


exp01-06 Sampling Data plot



Experiment 2: DSP Application (Low-Pass Filter)

Data Plot (Before and after LPF)



What is a moving average filter?

A moving average filter is a type of finite impulse response (FIR) filter used for smoothing data in a signal or time series. It works by taking the average of a certain number of consecutive data points from the input signal to produce each point in the output signal.

The moving average filter operates by sliding a window of fixed size over the input data and computing the average of the data points within that window. This average value becomes the output for the current position of the window. The window then shifts one data point, and the process is repeated to obtain the next output value.

A moving average filter can be denoted mathematically as the following:

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k]$$

- y[n]: the output signal at time n
- x[n]: the input signal at time n
- N: the window size or the number of samples used for averaging

The key properties of a moving average filter are:

1. Smoothing effect: It helps to reduce random noise and fluctuations in the signal by averaging neighboring data points. This smoothing effect is stronger for higher window sizes.
2. Lag introduction: The filter introduces a lag equal to (M-1)/2 samples, where M is the window size.

This means that the output signal is delayed compared to the input signal.

3. Low-pass filtering: The moving average filter acts as a low-pass filter, attenuating high-frequency components while preserving low-frequency components.

What is a finite impulse response filter?

An FIR (Finite Impulse Response) filter is a type of digital filter that uses only the current and past input samples to calculate the output signal.

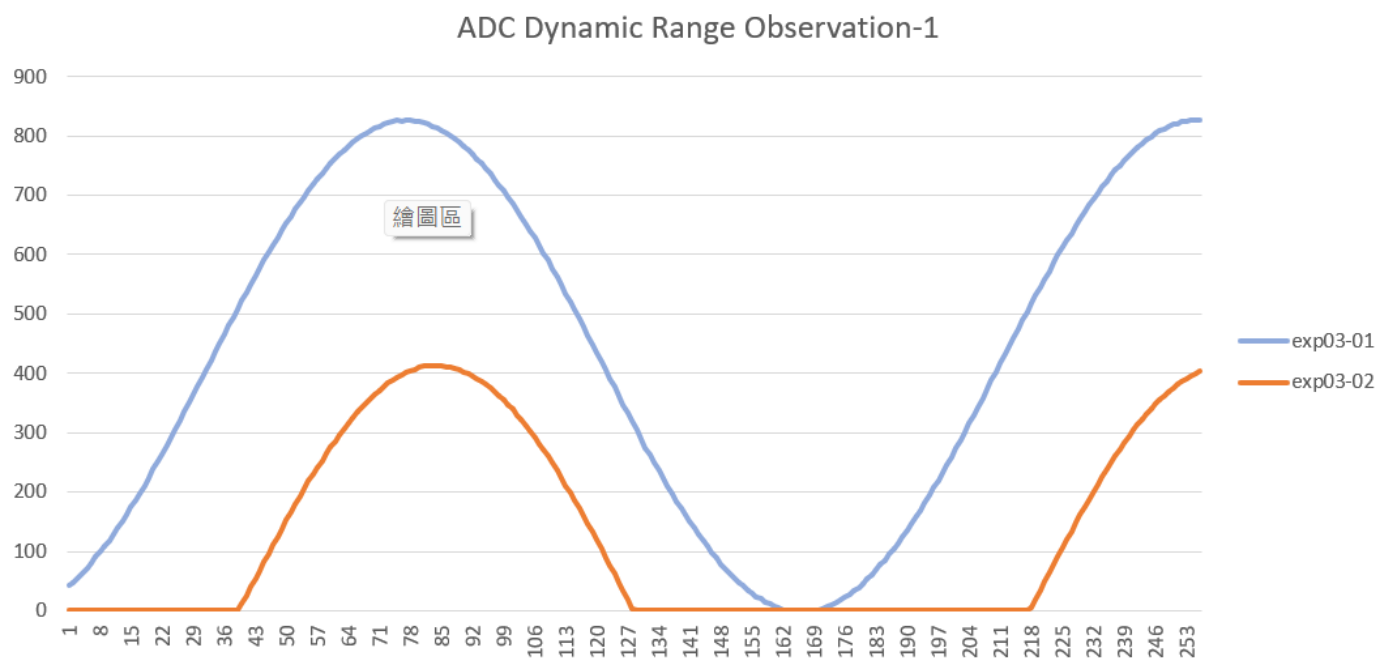
What is an infinite impulse response filter?

An IIR (Infinite Impulse Response) filter is a type of digital filter that uses feedback loops to generate the output signal as a linear combination of the current and past input samples, as well as the past output samples.

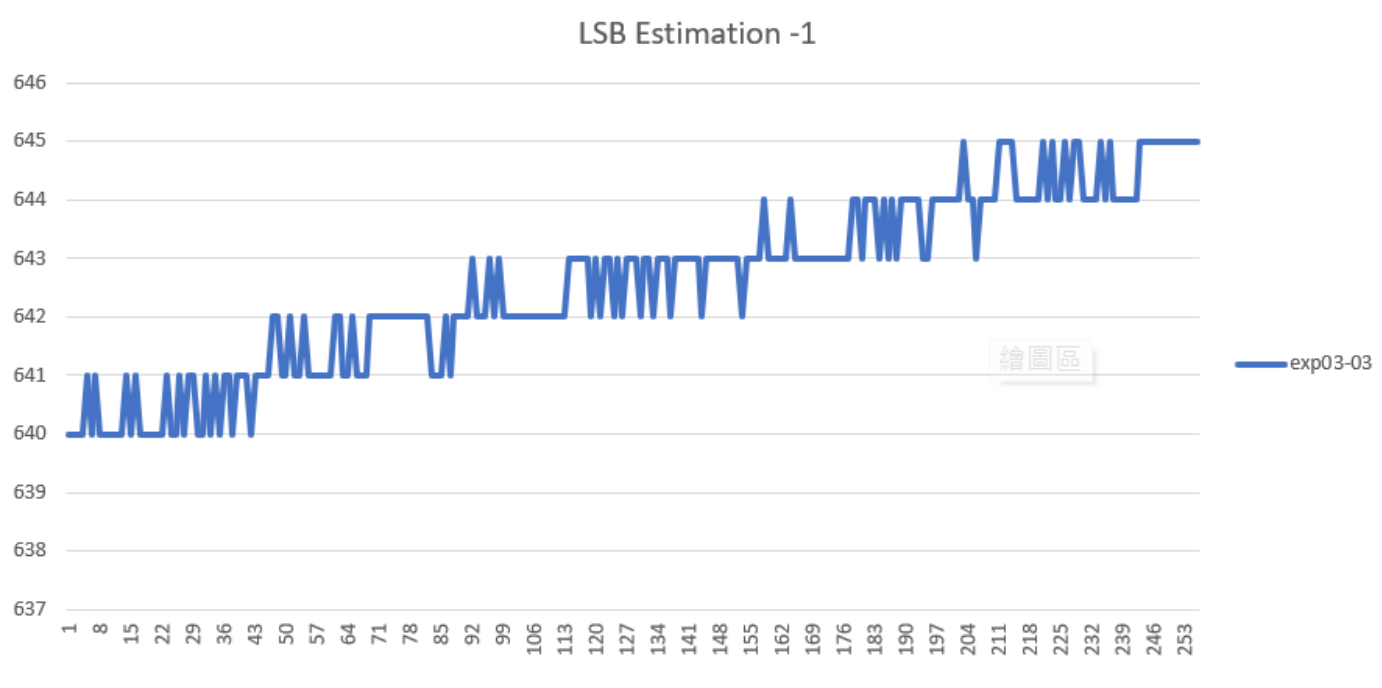
Filter Type	FIR Filter	IIR Filter
Structure	Non-recursive	Recursive (feedback loop)
Impulse Response	Finite duration	Infinite duration
Stability	Always stable	Can be unstable if not designed properly
Computational Complexity	Generally higher	Generally lower

Experiment 3: Dynamic Range, Quantization Error and LSB

exp03-01 + exp03-02 Sampling Data plot



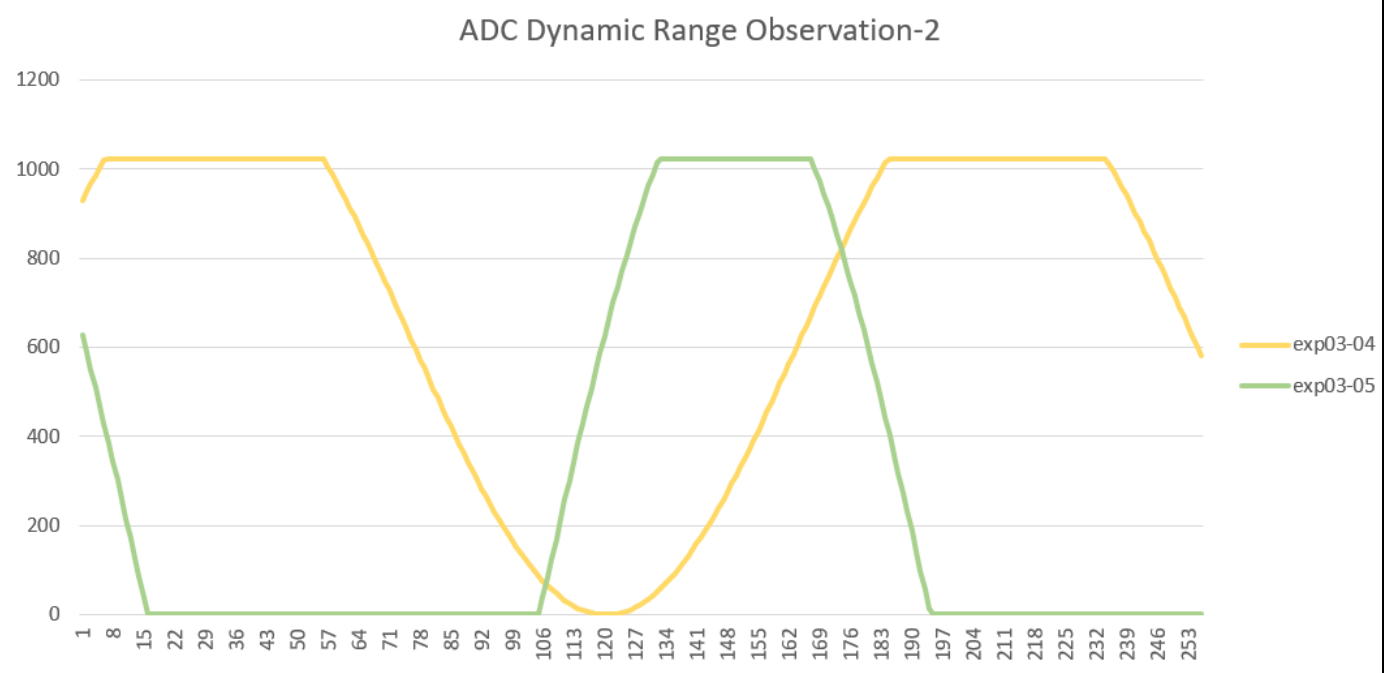
exp03-03 Sampling Data plot



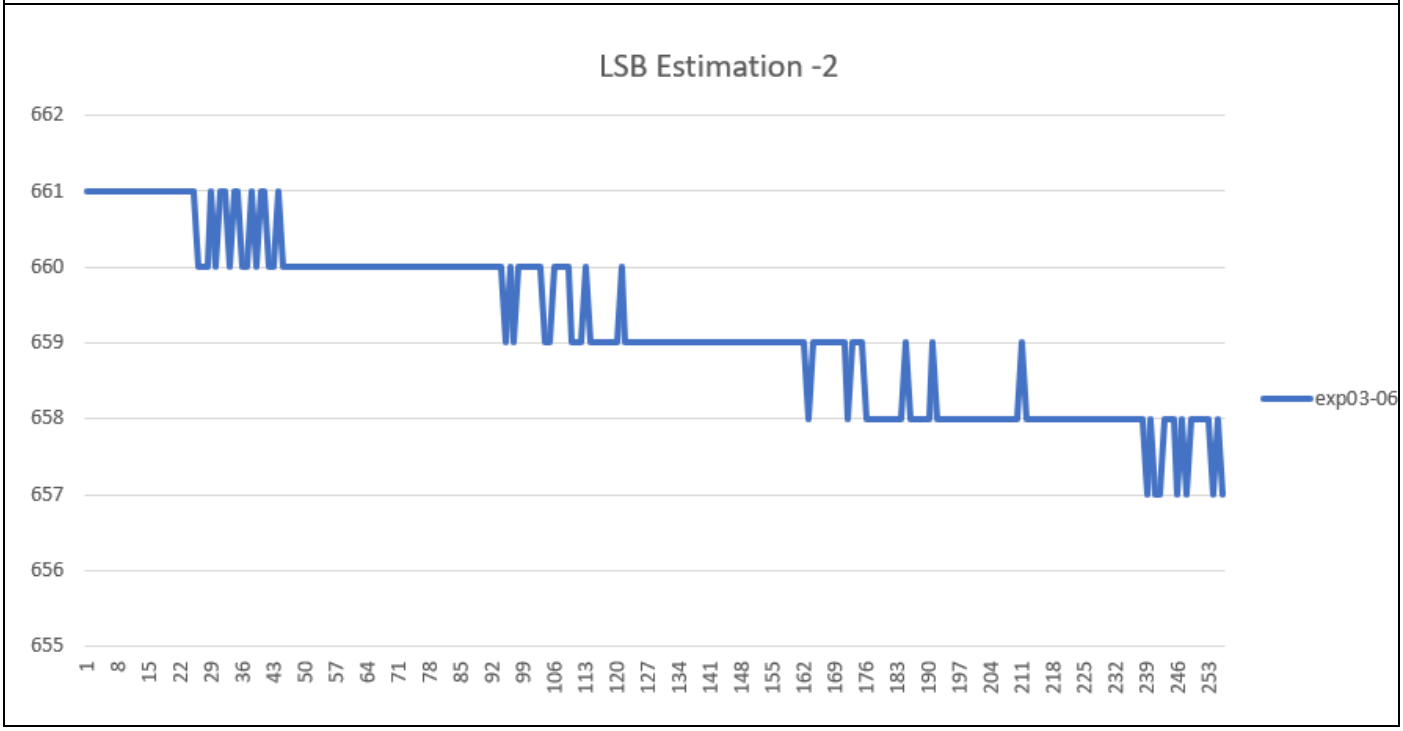
LSB = 0.0043 Volt (No analogReference() command, No AREF connection)

Ideal value = $\frac{5V}{2^{10}} = 0.0049\ V$

exp03-04 + exp03-05 Sampling Data plot



exp03-06 Sampling Data plot



LSB = 0.0027 Volt (enable analogReference() command, AREF=3.3V)

Ideal value = $\frac{3.3V}{2^{10}} = 0.0032\ V$

What's ADC (Analog-to-Digital Converter)?

ADC (Analog-to-Digital Converter) is an electronic circuit that converts an analog signal into a digital representation. Here's how it works:

1. The analog input signal is sampled at discrete time intervals, according to the sampling rate.
2. The sampled analog value is quantized into one of the digital output levels or codes.
3. The quantization is done by comparing the analog input to a set of reference voltage levels.
4. The digital output code represents the closest quantization level to the analog input.

Reference Voltage:

The reference voltage is the maximum analog input voltage that can be converted without any clipping or loss of information. In other words, it determines the full range of the ADC. When we used 3.3V as the reference voltage in exp03-04 and exp03-05, the Arduino could only fully represent signals from 0V to 3.3V. Any value not within this range will be clipped off.

Sampling Resolution:

Sampling resolution, or simply resolution, refers to the number of discrete digital output levels that an ADC can produce. It is typically expressed in bits. The resolution determines the granularity with which the ADC can represent the analog input signal.
The resolution is related to the number of quantization levels (N) by the equation: N = 2^resolution
For example, an 8-bit ADC has 2^8 = 256 quantization levels.

Least Significant Bit (LSB):

The Least Significant Bit (LSB) is the smallest possible change or step size in the digital output code of an ADC. It represents the weight or value of the least significant bit in the digital output.
The value of 1 LSB is calculated as: 1 LSB = (Analog Full-Scale Range) / (2^resolution)
For example, if an 8-bit ADC has a full-scale range of 0-5V, then its ideal LSB is:

$$LSB = \frac{5}{2^8} = \frac{5}{256} = 19.53mV$$

This means that the smallest change in the analog input that the ADC can detect and represent in the digital output is 19.53mV. However, there will be some error when we compare the actual LSB to the ideal value. Potential factors causing the error are as follow:

1. Reference Voltage Inaccuracy
2. Temperature
3. ADC Non-linearity
4. Noise
5. Quantization Error

Experiment 4: Analog Temperature Sensor

Attach your Arduino code here.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  double value;
  Serial.print("temp = ");
  Serial.print((value * 500)/1024);
  Serial.print(" degree Celsius\n");
  delay(2000);
}
```

The code is fairly straightforward. I declared a double named “value” to take the analog input from pin “A0”. Then, I applied the following conversion formula to acquire the correct temperature value. Because LM35 has a temperature reading refresh rate of 0.5Hz. I added a delay of 2 secs to make sure that LM35 has updated before the Arduino reads new data.

Conversion Formula:

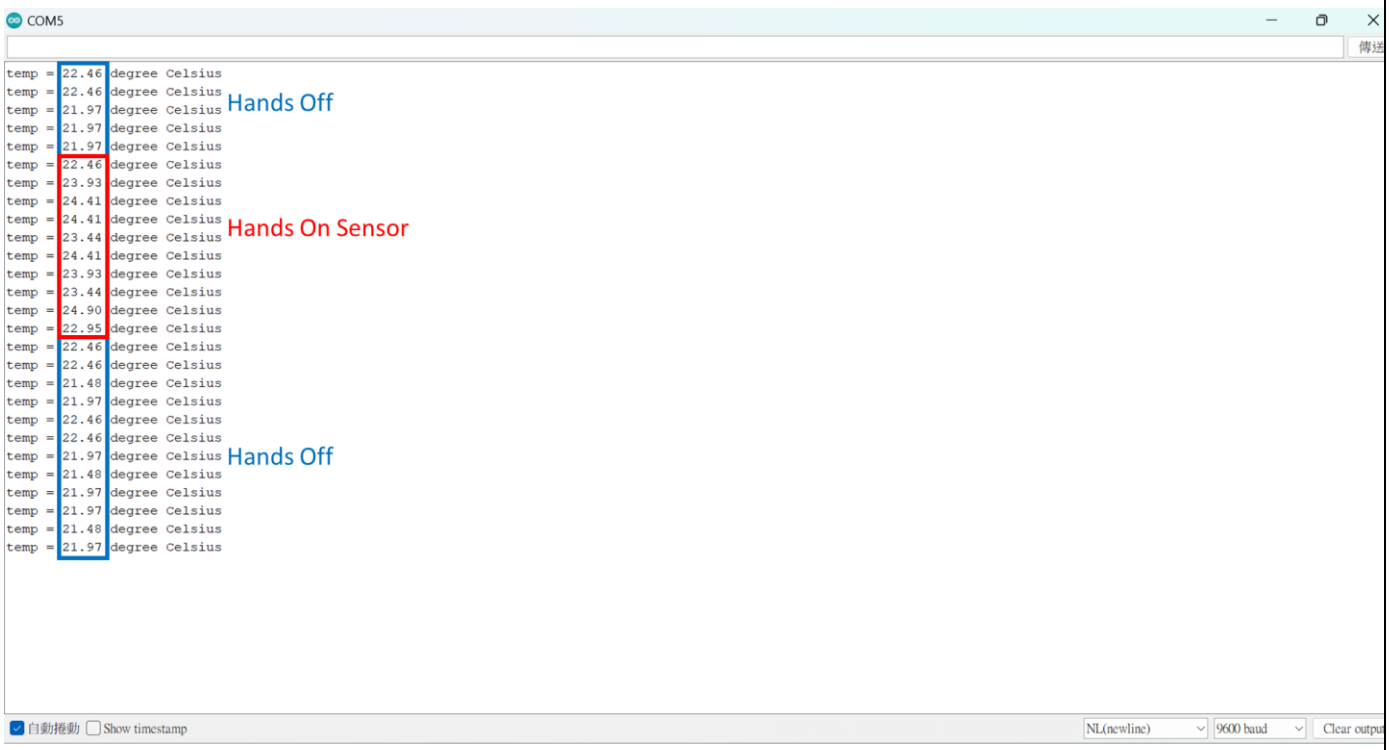
$$LM35\ input \times reference\ V \times 100 \div sampling\ resolution$$

Arduino Uno has a default reference voltage of 5V, and uses 10 bits for sampling. Therefore, the equation becomes:

$$LM35\ input\ \times 5 \times 100 \div 1024$$

Touching the LM35 with my hand led to a noticeable 2~3 °C increase in temperature.

Serial Monitor Screen Capture



Serial Plotter Screen Capture