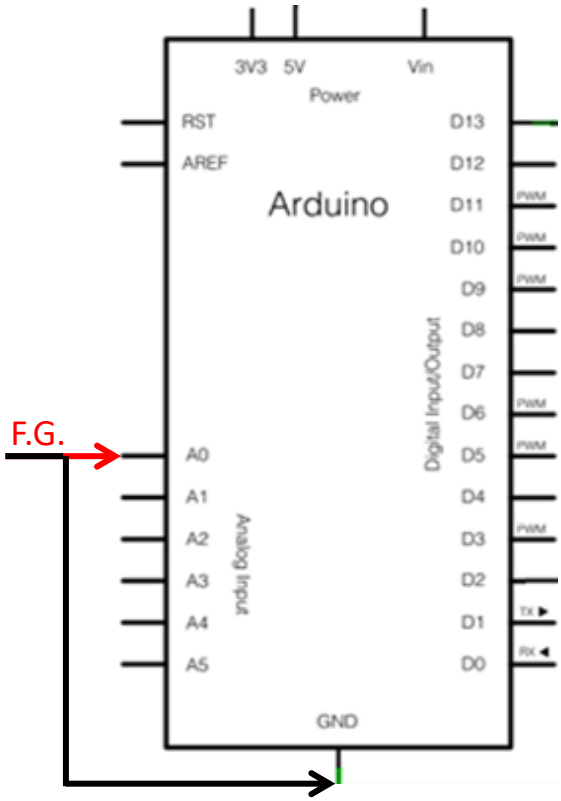
REPORT

Experiment 1: Sampling and Aliasing.



Arduino 本實驗新函式介紹與程式分析:

函式介紹:

1. analogReference(mode):

   依照 mode 來設置 analog input 的 reference 電壓，mode 的模式在 Arduino AVR Boards 系列有以下五種:

   -DEFAULT (預設狀態，依照 Arduino 板子型號可能是 5V 或 3.3V)

   -INTERNAL (使用 Arduino 板子內的 built-in reference ATmega168 和 ATmega328P 是 1.1V；ATmega32U4 和 ATmega8 是 2.56V；Arduino Mega 則不可使用此模式)

   -INTERNAL1V1 (使用 built-in 的 1.1V；只有 Arduino Mega 可以使用此模式)

   -INTERNAL2V56 (使用 built-in 的 2.56V；只有 Arduino Mega 可以使用此模式)

   -EXTERNAL (使用 AREF 處的電壓大小當作 reference 電壓，範圍限制在 0~5V)

2. analogRead(pin):

   從指定的輸入引腳(pin)讀取數值(信號)，通過 Arduino 板含有的一個多通道、10 位元模擬數位轉換器，將輸入電壓從 0V 至操作電壓(預設為 5V 或 3.3V，也可透過 analogReference() 函式決定由外部或內部獲取）映射到介於 0 到 1023 之間的整數值。例如，在實驗中使用的 Arduino UNO 上，若操作電壓為 5V，則可以產生解析度為：5 伏特 / 1024 單位或每單位 0.0049 伏特（4.9 毫伏）的數位訊號。例如:數位訊號 100 則可對應為 0.49V 的信號

程式分析:

首先定義了廣域變數 MAX_SIZE 和 data，前者用來決定要抓取幾筆輸入訊號對應出的數位訊號，後者用來儲存抓取到數位訊號。

setup 函式中設定了 Serial.begin(9600)來讓 Arduino 板子上的資訊可以傳送給電腦在序列埠監控視窗看到。

loop()函式中用 if-else 判斷是否繼續抓取資料，而抓取資料的方式是使用 analogRead()函式，接著會把抓取到的資料存入 data 中並且透過 Serial.println()顯示在序列埠監控視窗。

實驗 2-1、2-2、2-3 Arduino sketch:

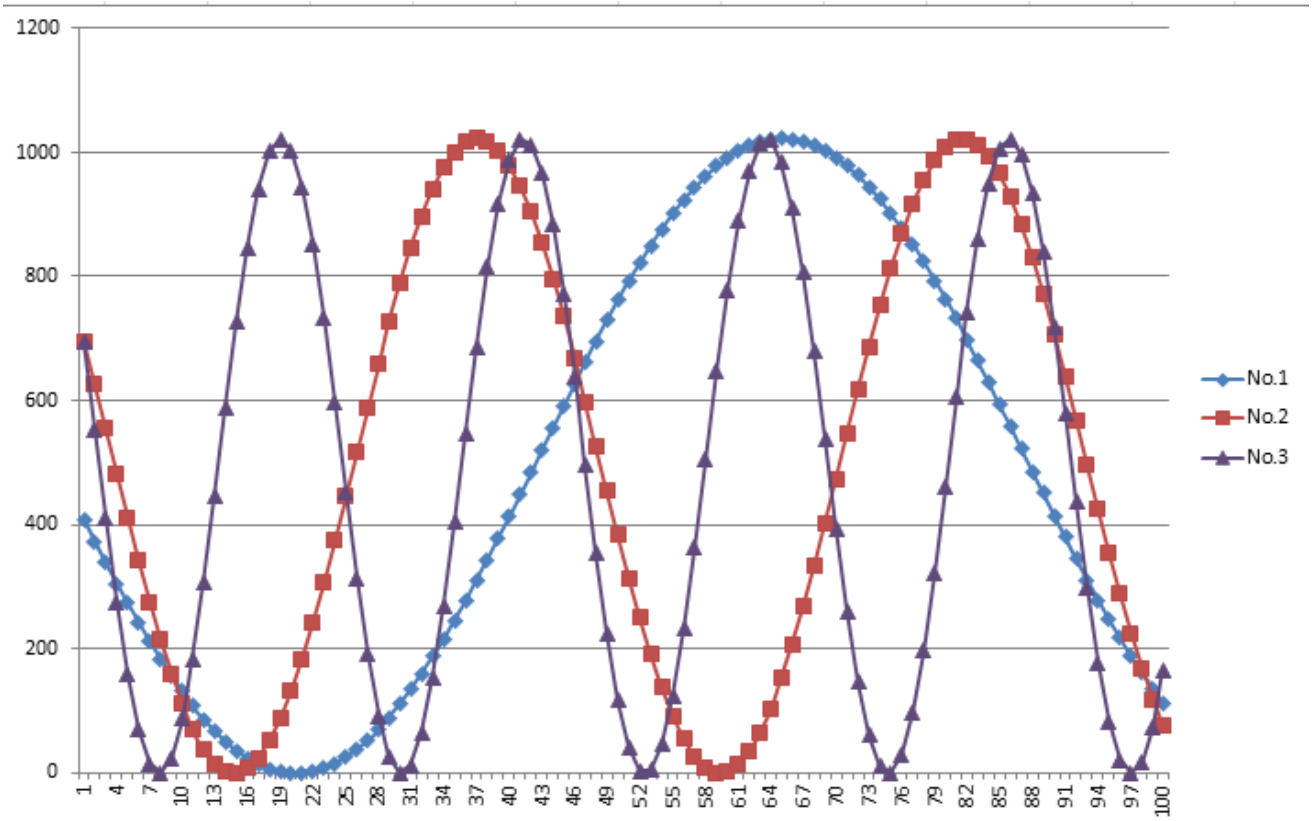
```
//condition for exp1
const int MAX_SIZE = 100;

//condition for exp2
//const int MAX_SIZE = 400;

//condition for exp3
//const int MAX_SIZE = 200;

//array stores A/D data
int data[MAX_SIZE];

int out[MAX_SIZE];
static int cnt=0,fcnt=0;

void setup()
{
  //assign serial communication baud rate
  Serial.begin(9600);

  //uncomment in exp2-4 and exp3-3 connect 3.3V to "AREF" pin
  //adjust the dynamic range of the A/D From 5V to 3.3V
  //analogReference(EXTERNAL);
}

void loop()
{
  if(cnt < MAX_SIZE)
  {
    //store data from A/D A0 pin
    for(cnt = 0;cnt < MAX_SIZE;cnt++)
      {
        data[cnt] = analogRead(A0);
      }
    //print data
    for(fcnt = 0;fcnt < MAX_SIZE;fcnt++)
    {
      Serial.println(data[fcnt]);
    }

    //print end mark
    Serial.println("finish!!");
  }
}
```

電路分析:

函數產生器正端串接 Arduino analog input A0 pin，負端串接 Arduino GND pin。

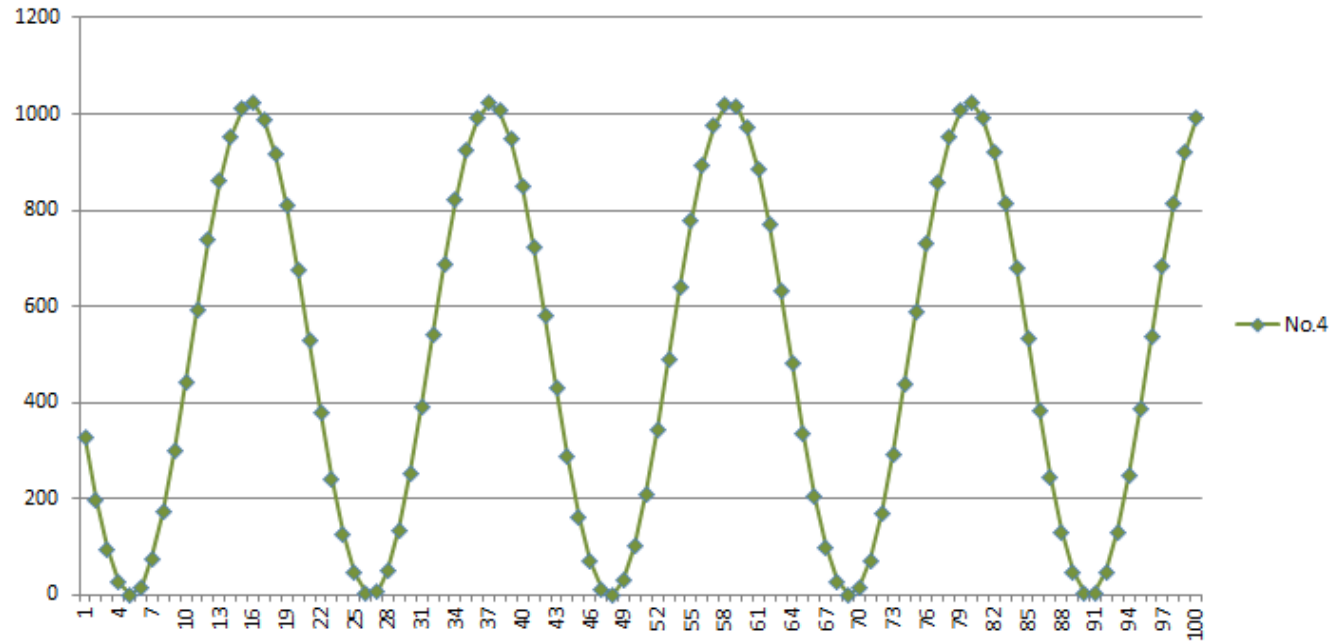
1.

Arduino serial monitor results plotted by excel – No.1, 2, 3 / Sine / Vpp:0-5V / freq:100/200/400Hz



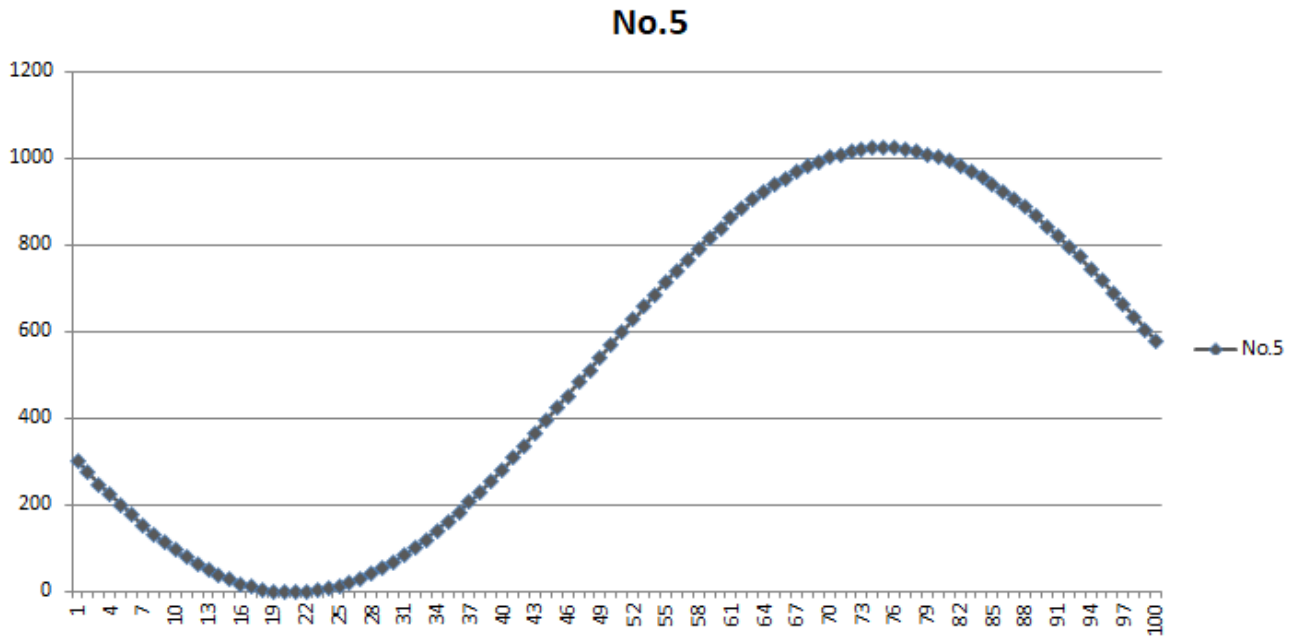
2.

Arduino serial monitor results plotted by excel – No.4 / Sine / Vpp:0-5V / freq:8.5kHz



3.

Arduino serial monitor results plotted by excel – No.5 / Sine / Vpp:0-5V / freq:9kHz



4.

Please estimate the sampling frequencies by your experiment results of No.1~No.3, and explain how you obtain it. Are the three sampling frequencies the same?

取波形週期兩端點 a、b，計算 a 點和 b 點之間有 n 點，由此可知在波形的一個週期內 Arduino 取樣了 n 次，因此可以透過將函數的週期除以 n 來獲得 Arduino 取樣週期，再將 Arduino 取樣週期倒數，即可獲得 sampling frequency: $\frac{T_{wave}}{n} = \frac{1}{f_{wave} \times n} = T_{sampling} = \frac{1}{f_{sampling}}$ ，

所以$f_{sampling} = \frac{1}{T_{sampling}} = f_{wave} \times n$

No.1：$f_{sampling} = f_{wave} \times n \cong 100 \times 89 = 8900\ Hz$

No.2：$f_{sampling} = f_{wave} \times n \cong 200 \times 46 = 9200\ Hz$

No.3：$f_{sampling} = f_{wave} \times n \cong 400 \times 23 = 9200\ Hz$

經過計算可以發現，取樣頻率大概在 9000Hz 上下，可以說是相當接近，若可以加取樣點，並用更多個週期來取 a 和 b 點，則可以獲得更準確的取樣頻率，誤差也會隨之減少。

Question:

Use the result of No1., No.2, and No.3 to estimate the correct sampling rate (calculate the average of these three conditions). And apply this correct sampling rate to derive the signal frequency of No.4 and No.5 respectively. (Notice that the frequencies you calculated are false signal frequency.)

由上面所計算的 8900Hz、9200Hz 和 9200Hz 取平均可獲得 9100Hz，接著按照上面推得的公式:

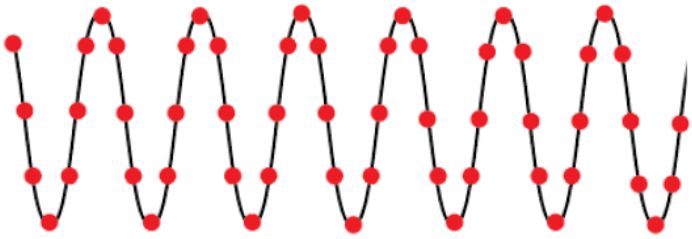
$f_{sampling} = \frac{1}{T_{sampling}} = f_{wave} \times n$來計算 No.4 和 No.5 訊號的頻率可獲得為如下:

No.4: $f_{wave} = \frac{f_{sampling}}{n} = \frac{9100}{21} \cong 433.3Hz$ (原訊號為 8.5k Hz)

No.5: $f_{wave} = \frac{f_{sampling}}{n} = \frac{9100}{113} \cong 80.5Hz$ (原訊號為 9k Hz)

為什麼用 sampling rate 反推的 No.4 和 No.5 頻率會與原訊號不同?該如何避免?

這是由於取樣頻率與原訊號頻率接近，這會導致 Arduino 每次取值時，訊號已經跑完一個完整波形左右，因此每次抓取到的數值其實都是新的一個波的同個位置往後或往前一點點。

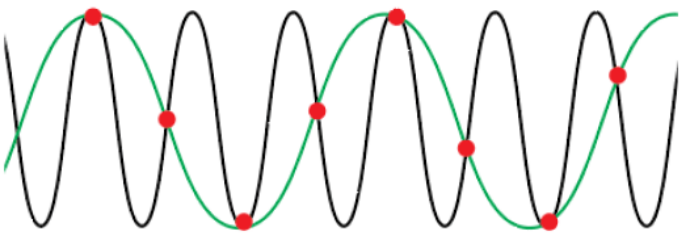


Sampling reflects true signal

這種現象稱為 aliasing，通常是因為取樣頻率太低，或訊號頻率過高導致。

如左上圖，可以發現當取樣頻率足夠抓取一個訊號一個週期內的波形多點時，就可以在不失真的情況下反映出原有訊號大致上的波型。

而當取樣頻率太低，如左下圖，就會導致只能抓取每個訊號源一次週期波形的一個點甚至在多次週期下只能抓取一個點時，就會導致 aliasing，而反推的訊號就會失真。

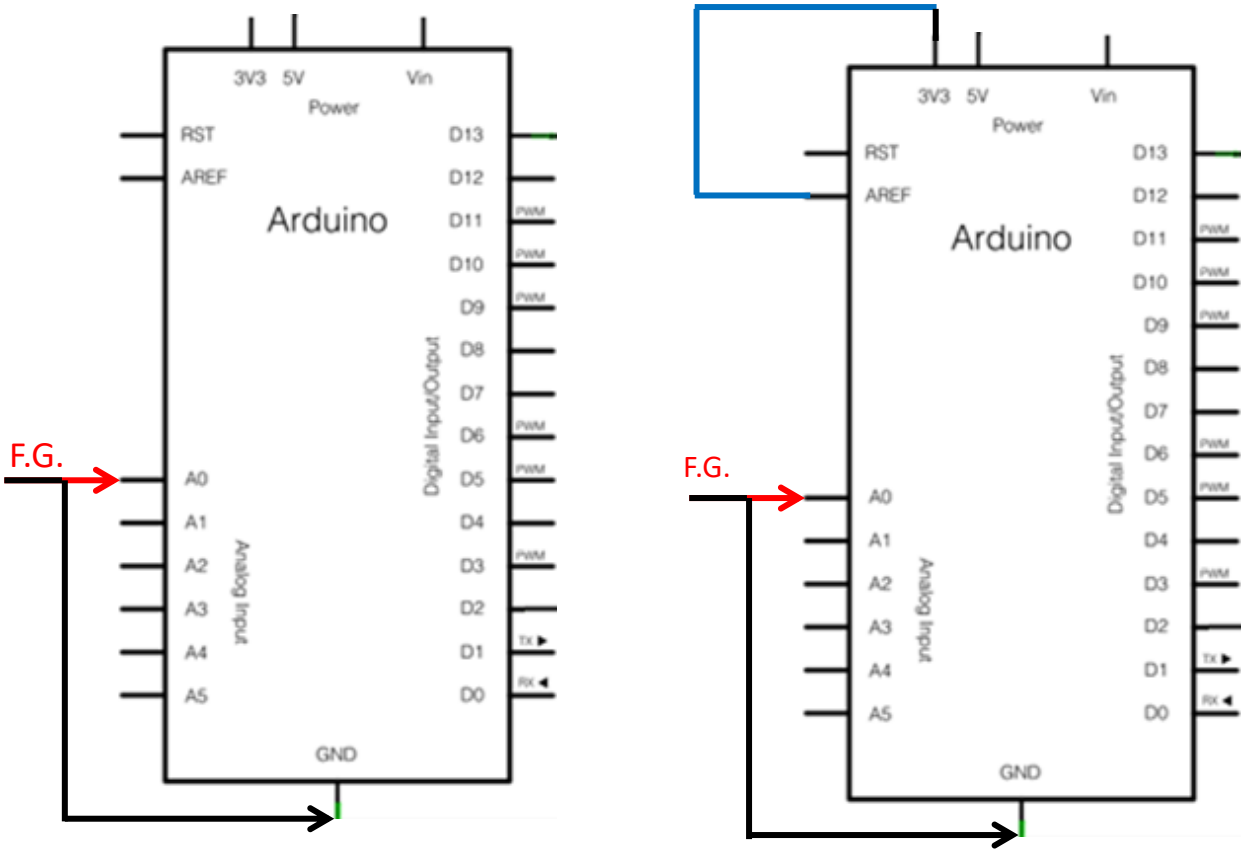


Low sampling rate leads to aliasing (false longer wavelength)

若想要解決 Aliasing 的問題，那麼我們勢必要提高 sampling rate 或是降低訊號的頻率，這時我們會提到一個專有名詞，Nyquist Frequency，此頻率是 sampling rate 的一半，只要輸入的訊號最高頻率低於 Nyquist Frequency，就可以在一個週期內至少被取樣兩個點以上，以此來避免 aliasing 的發生，因此，我們可以將先對輸入訊號進行高頻率波的動作，來將頻率高於 Nyquist Frequency 的訊號過濾掉，即可避免 aliasing 發生。

Please try to explain them in your report

Experiment 2: Reference Voltage for Analog Input.



NOTE：Remember modify the sketch in Exp1, change MAX_SIZE from 100 to 400.

NOTE：Remember uncomment analogReference(EXTERNAL); when you connect pin 3.3V to AREF.

Arduino 本實驗新函式介紹與程式分析:

函式介紹:

無

程式分析:

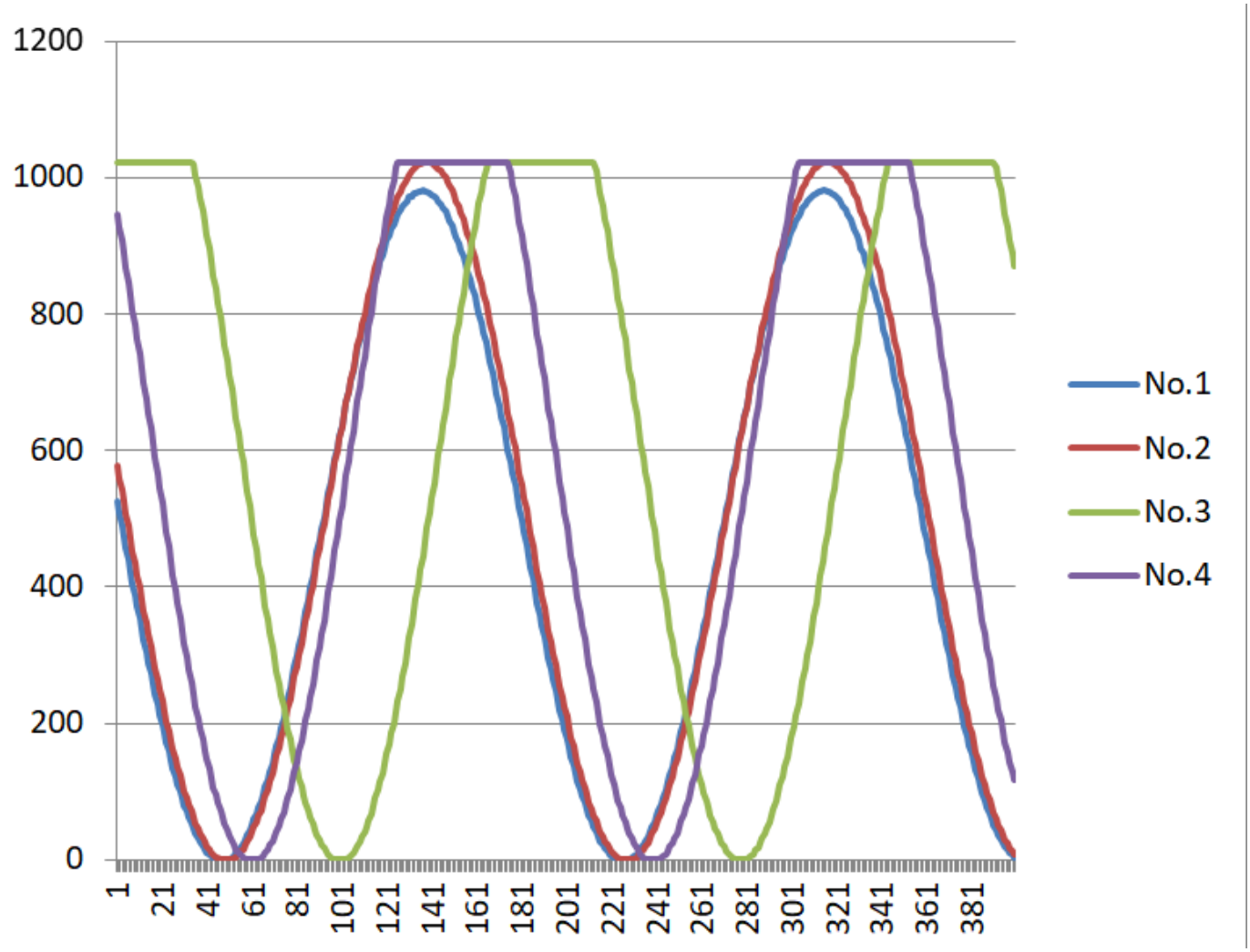
與實驗 2-1 相同，只有 MAX_SIZE 改成 400，也就是總共抓取 400 筆資料。值得注意的是在實驗 2-2-4 的時候要把 analogReference(EXTERNAL)的註解拿掉，讓 analog pin 腳的參考電壓改由從外部導入 AREF 的電壓決定。

電路分析:

函數產生器正端串接 Arduino analog input A0 pin，負端串接 Arduino GND pin。實驗 2-2-4 時將 Arduino 3.3V pin 腳與 AREF 串接。

1.

Arduino serial monitor results plotted by excel – Sine / Vpp:0-4.8V/5V/6V/4V+Aref:3.3V / freq:50Hz



Question:

1). Please estimate the full-scale input range A_{FS} of ADC of Arduino by your experiment results of No.1~No.3, and explain how you obtain it.

首先我們由實驗 2-1 函式介紹的資料，可以得到每單位的數位訊號值對應到的電壓值為 $\frac{1}{1024} \times 5V \cong 4.9mV$，所以可以大致反推 No.1 到 No.3 波形的最高電壓為如下:

No.1 (波形完整):

$$A_{FS1} = 981 * 4.9mV \cong 4.8069V \cong 4.81V$$

No.2 (波形完整):

$$A_{FS2} = 1023 * 4.9mV \cong 5.0127V \cong 5.01V$$

No.3 (波形不完整，有被截掉):

$$A_{FS3} = 1023 * 4.9mV \cong 5.0127V \cong 5.01V$$

No.1~No.3，使用 Arduino 的內部參考電壓，預設值為 5V，因此輸入的範圍理論上被限制為 0 ~ 5V。而 No.1 的波形最高值為 4.8V， No.2 的波形最高值為 5V，所以兩者皆為完整波形沒有被截掉；而 No.3 的波形最高值為 6V，所以超過 5V 的部分無法被正常表示，因此被截掉。

2). Please try to explain the experiment results of No.4 in your report

由於執行實驗 2-2-4 時，將參考電壓改成外部輸入的 3.3V，因此每單位的數位訊號值對應到的電壓值為$\frac{1}{1024} \times 3.3V \cong 3.2mV$，因此可以計算 No.4 的 full-scale input range 如下:

No.4

$$A_{FS4} = 1023 * 3.2mV \cong 3.2736V \cong 3.27V$$

可以發現當我們設定外部 3.3V 為參考電壓時，輸入訊號的限制範圍會縮到 0 ~ 3.3V，因此波形最高值為 4V 的 No.4，會出現被截掉的狀況。

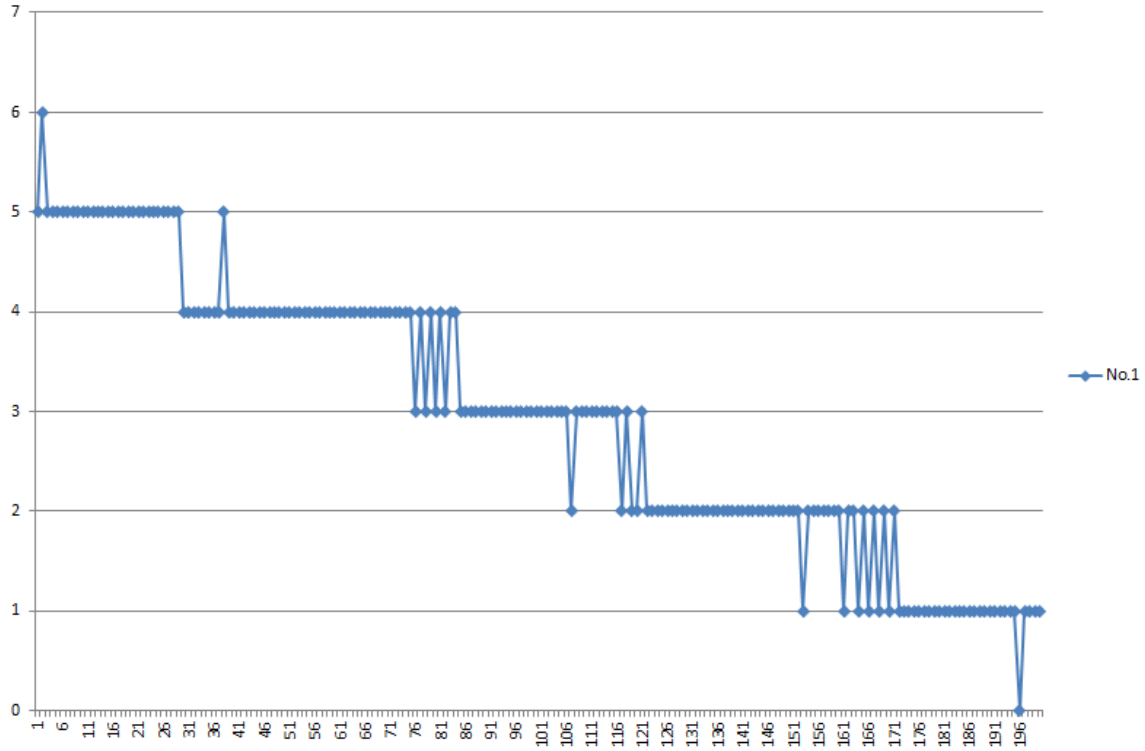
Experiment 3: Quantization Error.

NOTE：Remember modify the sketch in Exp1, change MAX_SIZE from 100 to 200.

NOTE：Remember uncomment analogReference(EXTERNAL); when you connect pin 3.3V to AREF.

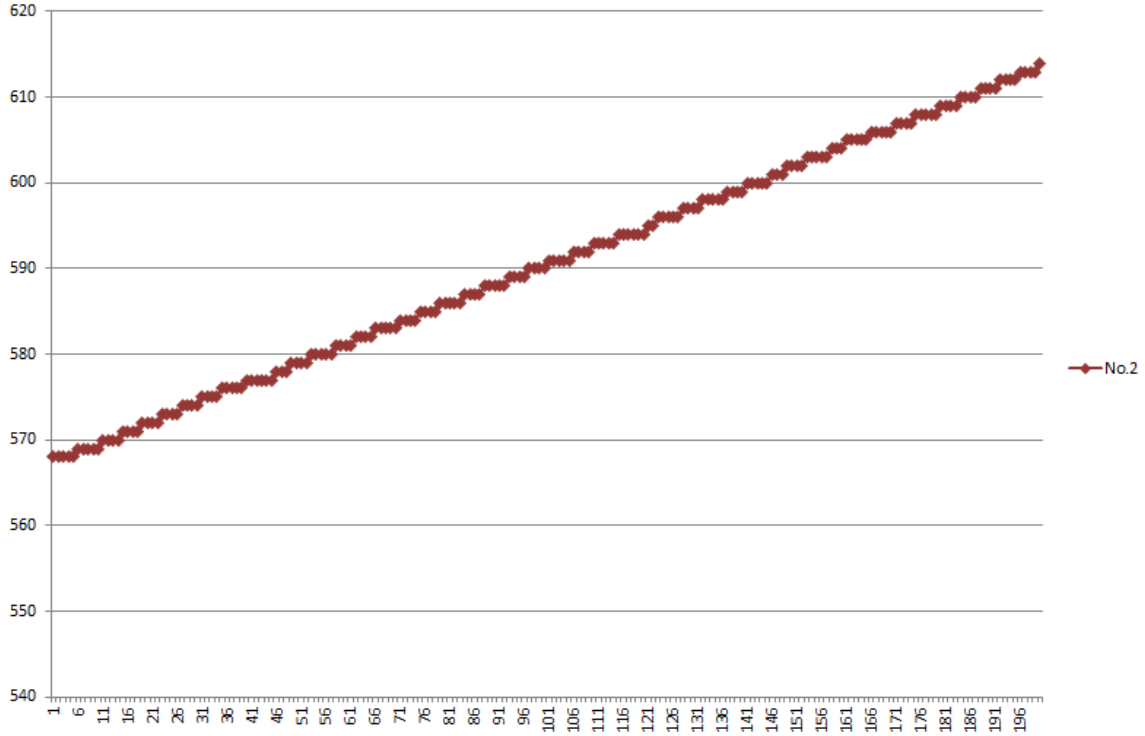
1.

Arduino serial monitor results plotted by excel –No.1 / Ramp / Vpp:0-5V / freq:0.1Hz



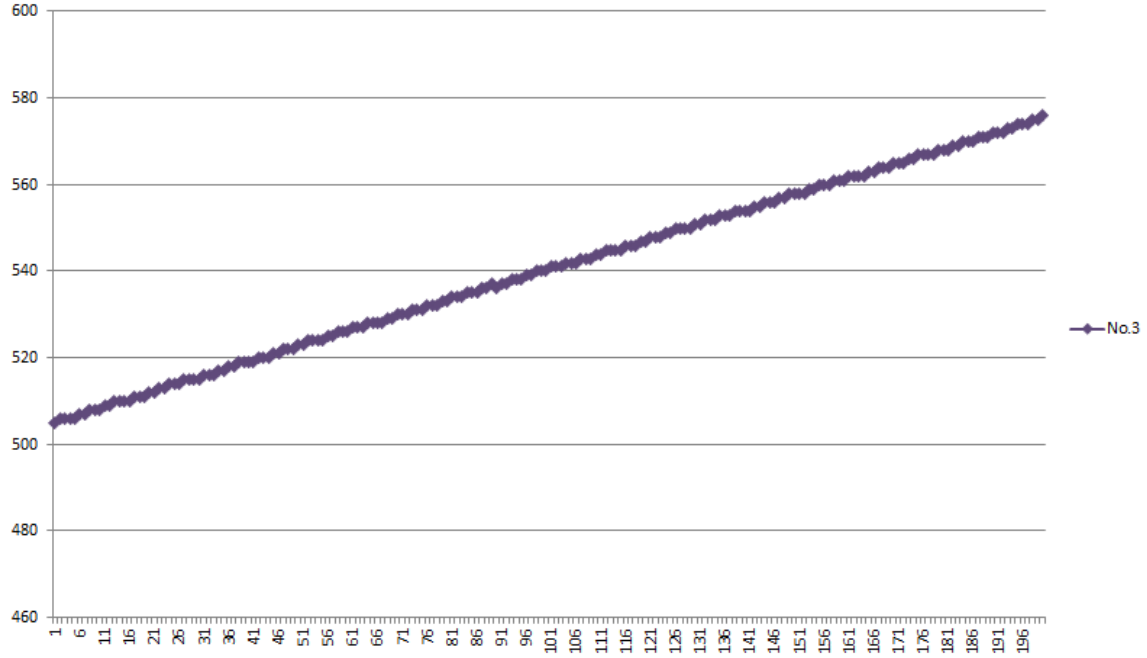
2.

Arduino serial monitor results plotted by excel –No.2 / Ramp / Vpp:0-5V / freq: 1Hz



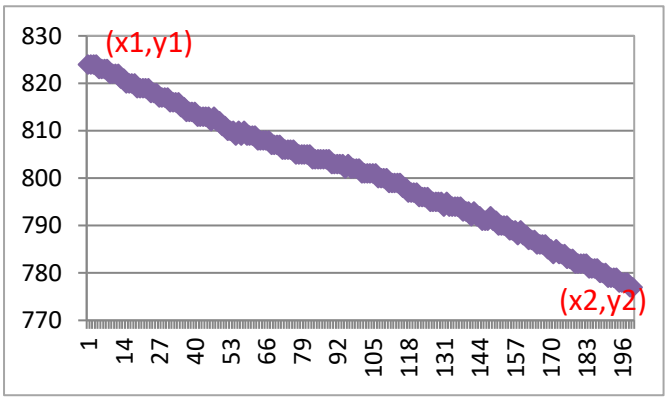
3.

Arduino serial monitor results plotted by excel –No.3 / Ramp / Vpp:0-5V / freq: 1Hz / Aref:3.3V



4.

Please calculate average LSB (Least Significant Bit) by experiment results of No.1 ~ No.3 in those intervals. Please use the average sampling frequency fs which you calculate from Exp1.



System sampling frequency: fs (from exp1)

Input signal: ramp wave,

Peak to peak voltage: Vpp

Frequency: *F* Hz

$$\text{Average LSB voltage} = \frac{Voltage\ Interval}{Digital\ Interval} = \frac{(V_{PP} \times 2F) \times \frac{x_2 - x_1}{f_s}}{|y_2 - y_1|}$$

No.	1	2	3
Average LSB voltage	0.0055	0.004782609	0.003098592

Question:

何謂 Quantization error?

量化誤差(Quantization error)是指輸入信號轉換為數位信號時所產生的誤差，其大小取決於量化級數或數位轉換器的位元數。在頻域上，可以表現為 DAC 的輸出雜波，此現象會對信號之間的轉換造成失真。量化誤差的主要來源包括分辨率限制和轉換器本身的基本誤差。典型的轉換器位元數約為 12 位，可能更高或更低，造成不同轉換器之間量化誤差也會不同。因此，量化誤差是數位信號處理與轉換中需要考慮的重要問題之一。

何謂 LSB 值?其與量化誤差有什麼關係?

LSB 值指的是最低有效位元（Least Significant Bit），在數位信號轉換中代表最小的變化。它與量化誤差密切相關，因為量化誤差即是將連續的數入信號轉換成離散的數位信號時所產生的，而每個量化級之間的差異即由 LSB 值來表示。如果量化誤差超過了 LSB 值的一半，就應該要進位或退位一個單位的數位訊號值，若沒有妥善處理則可能導致信號嚴重失真。因此，計算並了解 LSB 值的對於評估量化誤差的大小和數字信號的轉換精確度至關重要。

觀察本實驗的 No.1 到 No.3 波形的 LSB 值，可以發現什麼現象?為什麼?

由 No.1 和 No.2 做比較可以看出，在 Vpp 相同的情況下，頻率的不同會造成 LSB 值的不同，其關係為當輸入訊號的頻率越大，其 LSB 值將會越小。因為透過 LSB 值的計算公式可以發現，當訊號頻率越大時，分母的$|y_2 - y_1|$會越大，而其他部分數值則不變，因此造成 LSB 值下降。

由 No.2 和 No.3 做比較可以看出，在 Vpp 和頻率相同的情況下，參考電壓的不同會造成 LSB 值的不同，其關係為當參考電壓越小，其 LSB 值將會越小。因為透過 LSB 值的計算公式可以發現，當訊號的參考電壓下降，會讓每單位數位訊號值所代表的實際電壓值下降，而由於輸入訊號的 Vpp 和頻率不變，因此在相同時間下取樣相同數量就會發現，有著較小參考電壓的數位訊號，其斜率較陡峭，這也造就了 LSB 公式分母的$|y_2 - y_1|$會越大，而其他部分數值則不變，因此造成 LSB 值下降。

Experiment 4: Moving average filter.

Arduino 本實驗新函式介紹與程式分析:

函式介紹:

無

程式分析:

實驗 2-4-1 與實驗 2-1 相同，只有 MAX_SIZE 改成 200，也就是總共抓取 200 筆資料。到了實驗 2-4-2 的時候，引入了移動平均濾波器的概念，用來減少信號中的雜訊或高頻的成分，首先第一個 for 迴圈將原本 data 裡面的資料，取 5 筆進行平均，然後存入 out 陣列，接著第二個迴圈將 out 的資料印到序列埠監控視窗。

實驗 2-4-1、2-4-2 Arduino sketch:

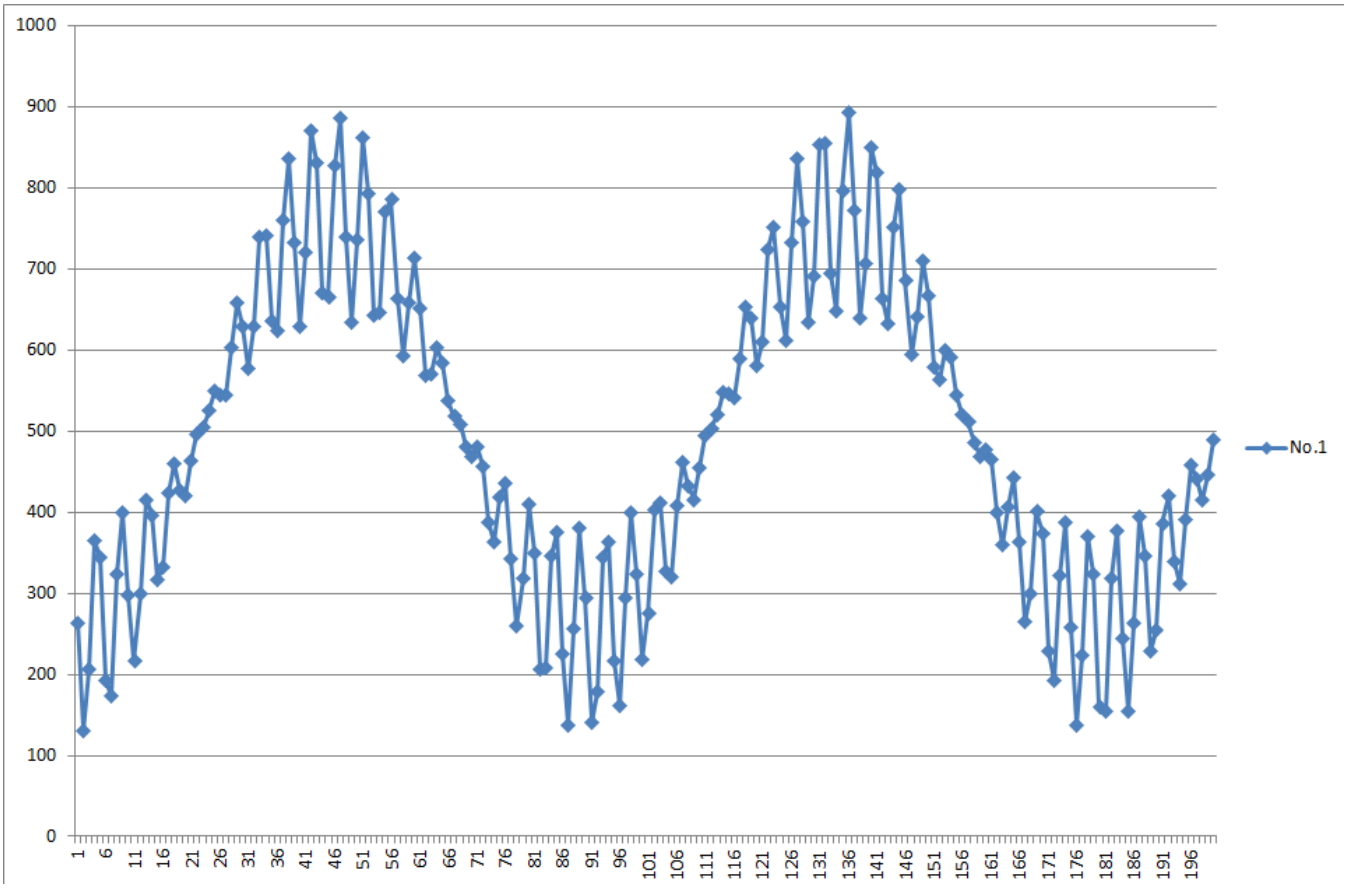
```
//condition for exp4
const int MAX_SIZE = 200;

//array stores A/D data
int data[MAX_SIZE];
int out[MAX_SIZE];

void setup()
{
  //assign serial communication baud rate
  Serial.begin(9600);
}

void loop()
{


  static int cnt=0,fcnt=0;

  if(cnt < MAX_SIZE)
  {
    //lab2-4-1 part
    for(cnt = 0;cnt < MAX_SIZE;cnt++)
      {
        data[cnt] = analogRead(A0);
      }

    for(fcnt = 0;fcnt < MAX_SIZE;fcnt++)
    {
      Serial.println(data[fcnt]);
    }

    //print end mark
    Serial.println("finish!!");
```

```
    //lab2-4-2 part
    for(fcnt = 0;fcnt < MAX_SIZE;fcnt++)
      {
        if(2 <= fcnt <= (MAX_SIZE - 3))
        {
          out[fcnt] = (data[fcnt - 2] + data[fcnt - 1] + data[fcnt] + data[fcnt + 1] + data[fcnt + 2])/5;
        }
      }

    for(fcnt = 2;fcnt <= (MAX_SIZE - 3);fcnt++)
      {
        Serial.println(out[fcnt]);
      }
  }
}
```

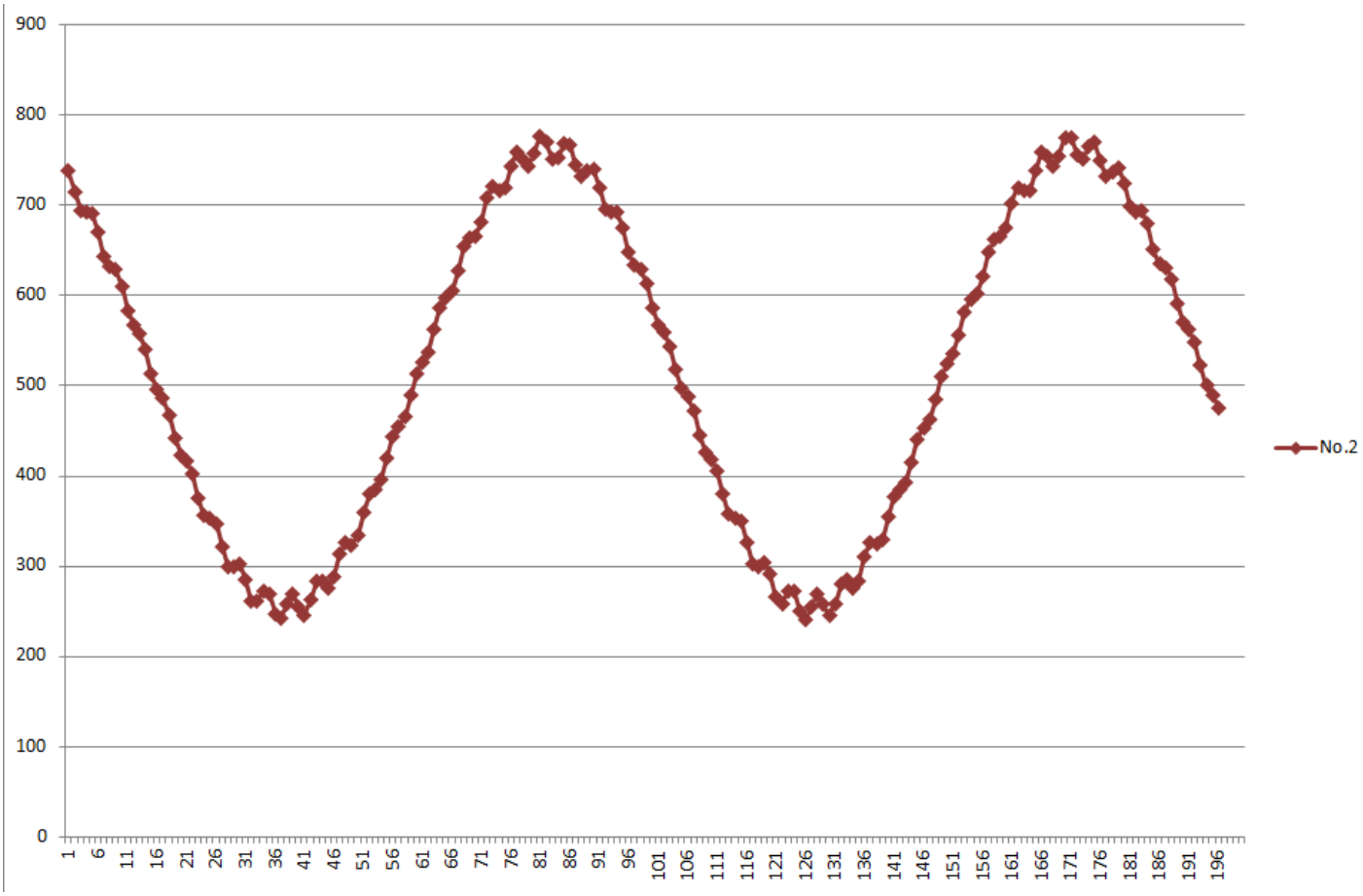
電路分析:

函數產生器正端串接 Arduino analog input A0 pin，負端串接 Arduino GND pin。

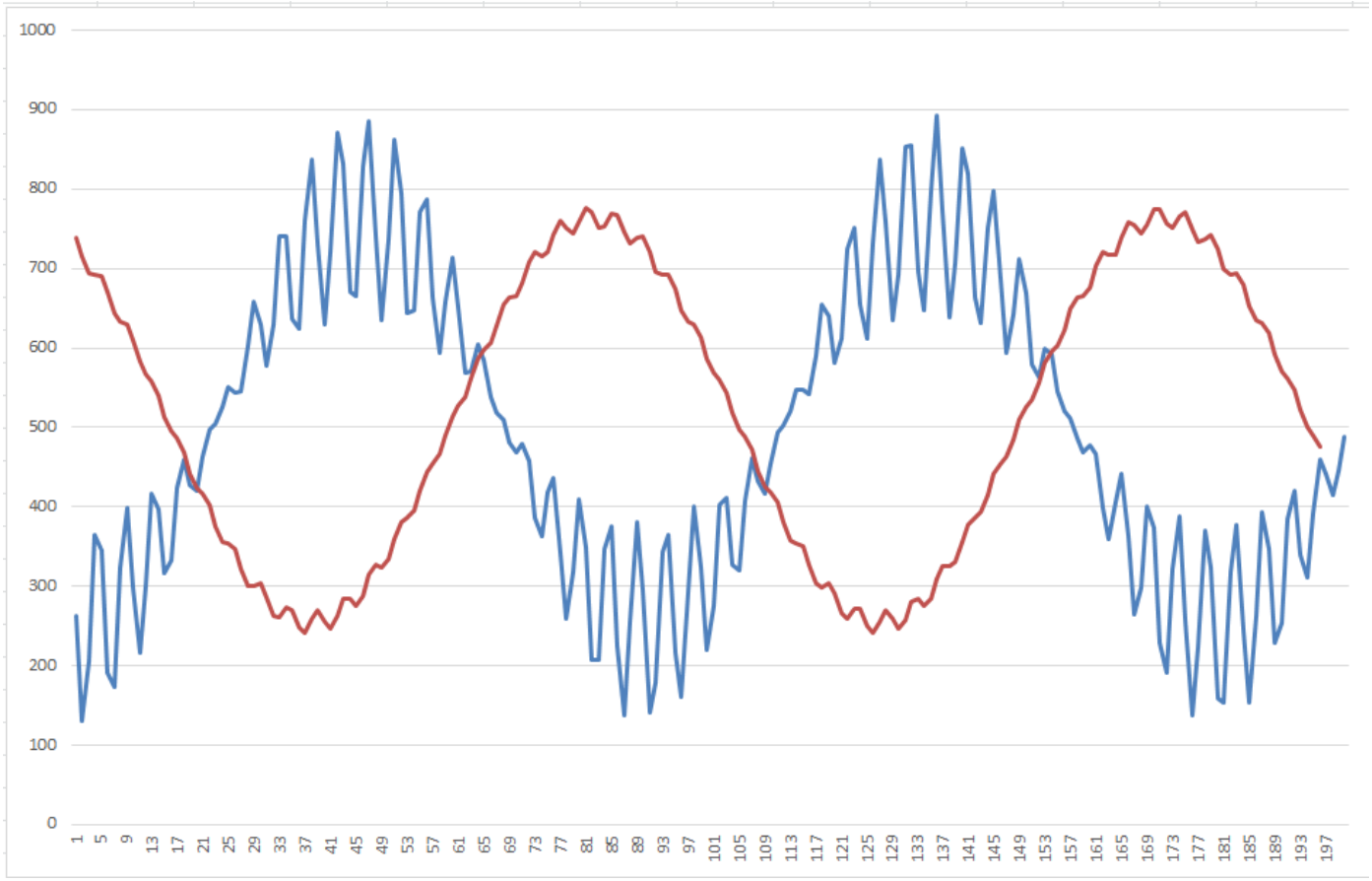
Excel plot (Before filter)



Excel plot (After filter)



Excel plot (Both condition)



Question:

甚麼是移動平均濾波? 為甚麼要做移動平均濾波?

移動平均濾波(Moving Average Filtering)是一種對一維信號濾波的算法，其基本原理是將一個給定的 N 個樣本信號序列 x[n]，透過一個固定長度為 m 的窗口去進行抓取資料並計算平均值後，將結果存入另一個序列 y[n]，其公式可表達如下:

$$y[k] = \frac{x[k] + x[k-1] + x[k-2] + \cdots + x[k-m+1]}{m}$$

其中 x[k]是原樣本信號第 k 個樣本值，1/m 是歸一化因子，用於平均窗口內的樣本值。

做移動頻均濾波的主要原因就是要濾波，透過將窗口內的信號值做平均，即可減少被雜訊影響的程度，亦或是可以過濾掉高頻率的信號，讓使用者採集到真實的信號，而非受雜訊或高頻影響後的結果。

Experiment 5: Analog temperature sensor

Arduino 本實驗新函式介紹與程式分析:

函式介紹:

無

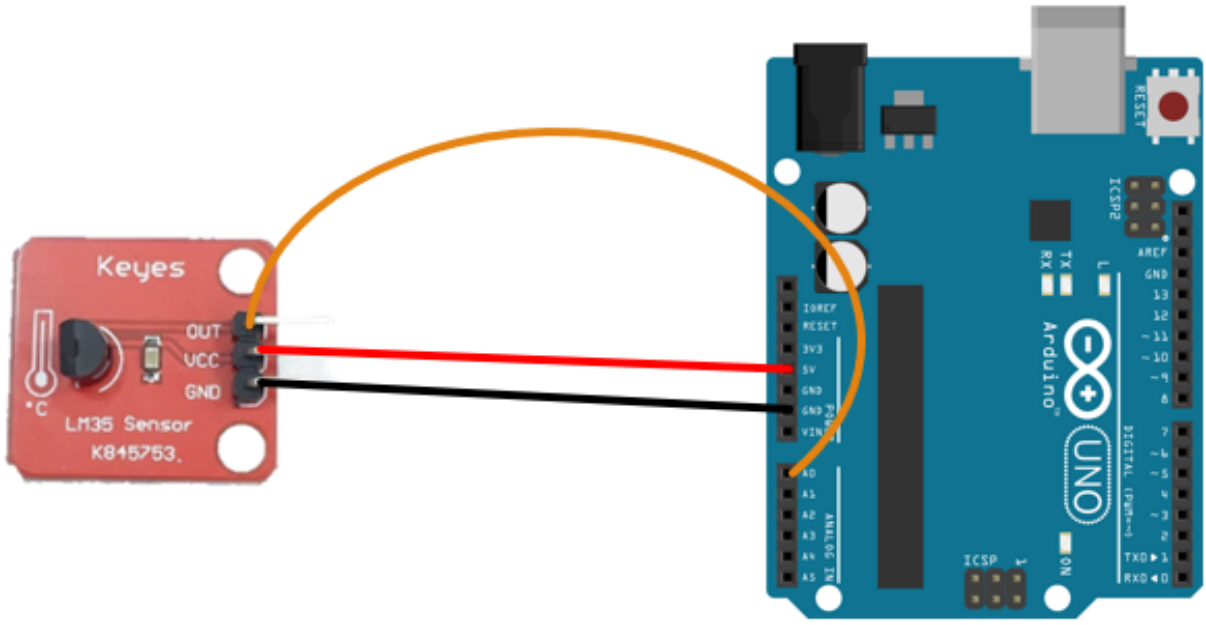
程式分析:

使用 analogRead()將 LM35 感溫 IC 的輸出電壓讀入 sensor 變數，接著透過 LM35 輸出電壓對應的溫度計算公式將計算的值存入 tem 變數，最後將 tem 也就是計算出的溫度透過 Serial.print()，顯示到序列埠監控視窗。

電路分析:

LM35 OUT 串接 Arduino analog input A0 pin，VCC 串接 Arduino 5V pin，負端串接 Arduino GND pin。

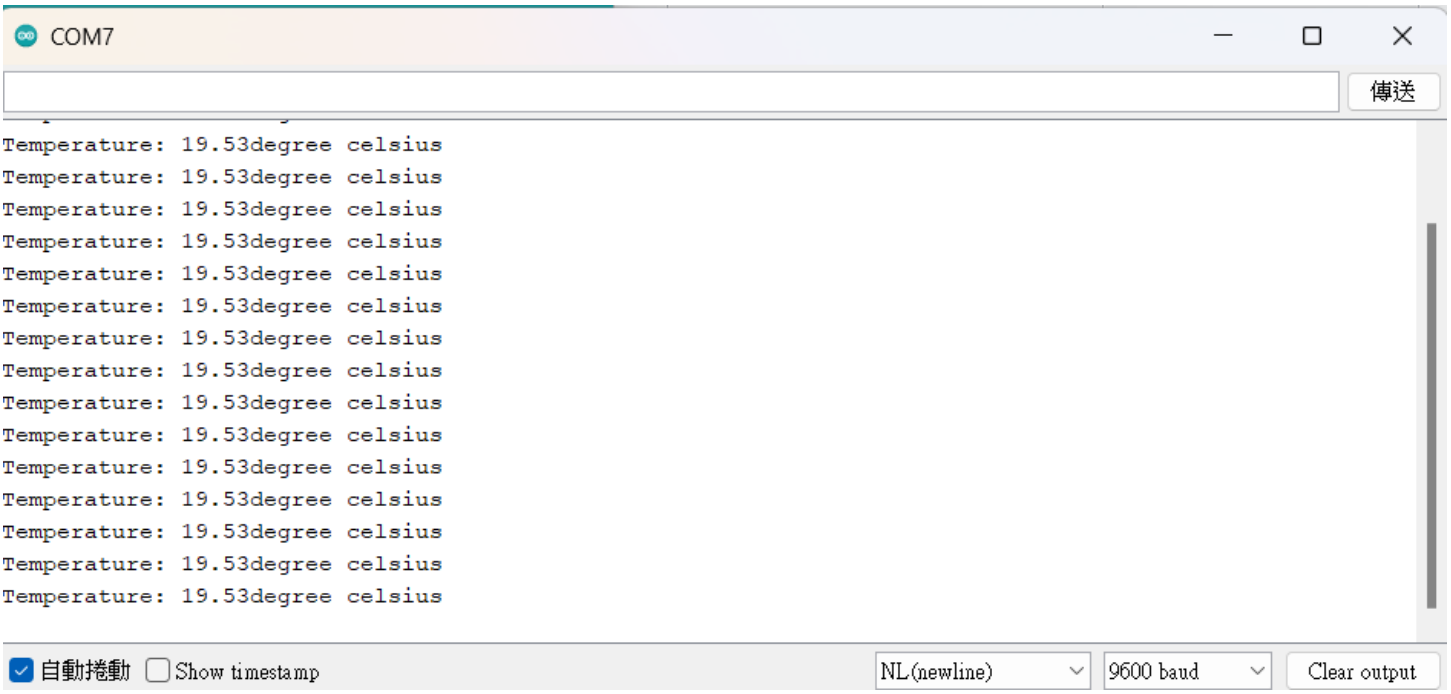
The circuit diagram of your design: (label every port clearly)



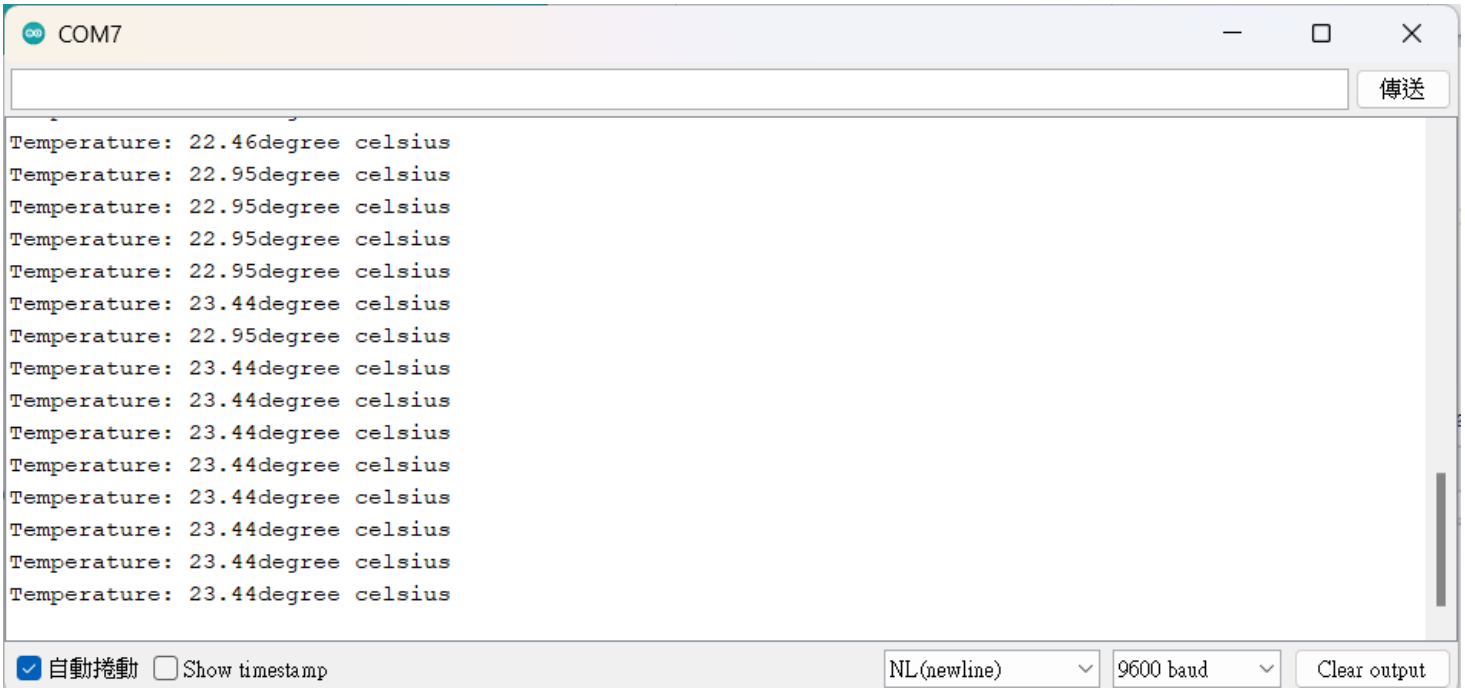
The sketch of your design: (copy from the Arduino IDE window and paste here)

```
float tem,sensor;

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

void loop()
{
  sensor =  analogRead(A0);
  //for LM35
  tem = (sensor/1024*5)/10*1000;
  //for LMP36 (tinkercad)
  //tem = (sensor/1024*5*1000 - 500)/10; //Temp = ( Veri - 500 ) / 10
  Serial.print("Temperature: ");
  Serial.print(tem);
  Serial.println("degree celsius");
  delay(1000);
}
```

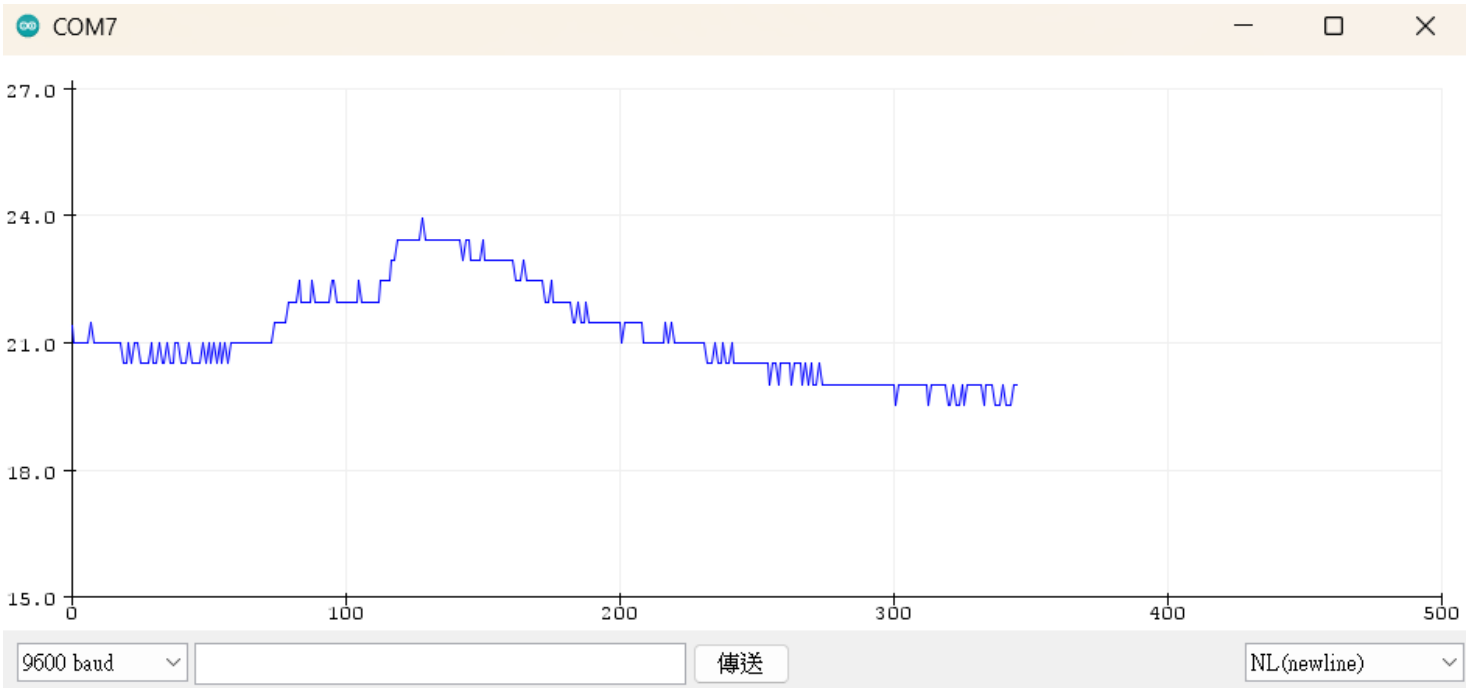
The screen capture of the serial monitor: (show the room temperature value on the window)



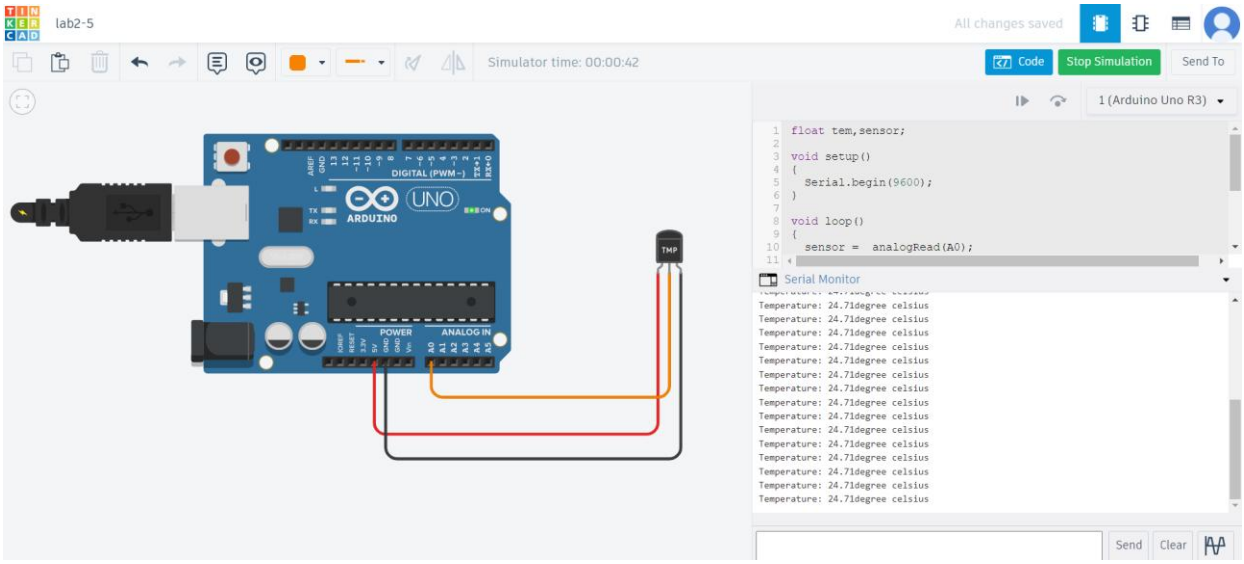
The screen capture of the serial monitor: (show the heated temperature value on the window)



The screen capture of the serial plotter: (show the change of the temperature)



Tinkercad 模擬:



Tinkercad 的溫度感測器中沒有 LM35，因此使用 TMP36 來代替，其溫度計算公式為:

$$V_{out} = (T \times 10 + 500)\, mV/℃$$

$$\rightarrow T = \frac{V_{out} \times 1000 - 500}{10}\ ℃$$

Question:

LM35 是甚麼? 其工作原理是?

溫度感測器像是 LM35 是一種廣泛用於測量溫度的元件。它的運作原理基於熱電偶效應，這是一種物理現象，當兩個不同金屬連接時，它們會產生一個電位差。LM35 採用了溫度和電壓之間的線性關係，當溫度上升時，LM35 內部的溫度感測元件會感知到內部金屬的電壓變化，並將它轉換為電壓信號從 OUT 端腳輸出。

LM35 的輸出電壓與攝氏溫度成正比，每攝氏 1 度的溫度變化約對應著 10 毫伏（0.01 伏）的電壓變化。例如，當溫度為 25 攝氏度時，LM35 的輸出電壓約為 250 毫伏（0.25 伏）。

Experiment 6: Breathing light

Arduino 本實驗新函式介紹與程式分析:

函式介紹:

1. analogWrite(pin, value):

將類比值（PWM 波形）寫入一個腳位。可用於以不同亮度點亮 LED 燈或驅動馬達以不同速度運轉。在呼叫 analogWrite()後，根據不同 value 的數值(0 always off ~255 always on)該腳位將產生指定佔空比的穩定方波，直到下一次呼叫 analogWrite()（或在同一腳位上呼叫 digitalRead()或 digitalWrite()）為止。

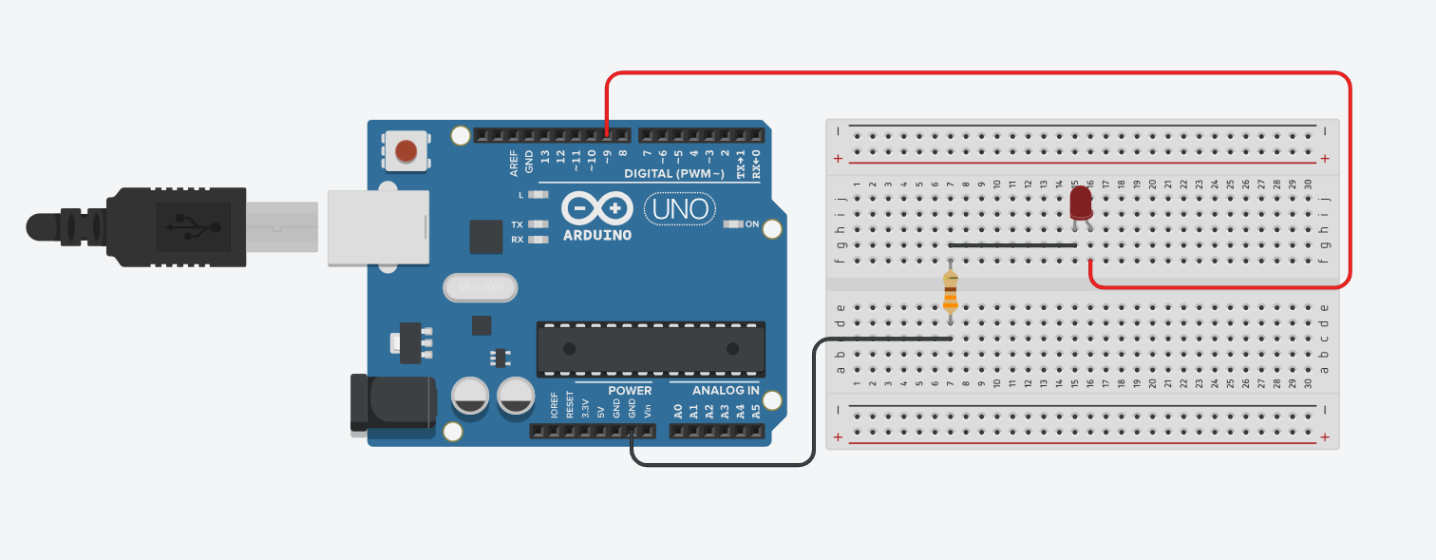
程式分析:

首先在 setup()中定義 pin 9 為輸出模式，接著在 loop()中使用一個 if-else 判斷 initial 這個變數是否為 0 或 255，若 initial 達到 0 或 255 則將 gap 變數乘上負號，其目的是控制 initial 變數在 0 到 255 之間，每 0.045 秒以 5 個單位做變化，用來控制 pin 9 的輸出方波佔空比，進而控制 LED 燈的亮暗變化。

電路分析:

Arduino pin9 輸出方波信號到 LED 的 anode，LED 的 cathode 串接一顆 330ohm 的電阻，最後電阻另一端串接回 Arduino 的 GND pin 腳。

The circuit diagram of your design: (label every port clearly)



The sketch of your design: (copy from the Arduino IDE window and paste here)文字版統一放在結報最後

```
int led = 9,gap = 5, initial = 255;

void setup()
{
  pinMode(led,OUTPUT);
}

void loop()
{
  if(initial == 0 || initial == 255)
  {
    gap = -gap;
  }
  analogWrite(led,initial);
  initial += gap;
  delay(45);
}
```

Your demo video(both LED and waveform) link: 大助說有現場 demo 過就不用放影片 :D

Question:

若在 loop()中使用不同的 delay()時間會造成 LED 的亮暗變化有甚麼樣的改變?

由 Arduino 官網對 analogWrite()的介紹(下圖)，可以發現不同型號的板子對應 pin 腳產生的方波有不同的頻率，我們使用的 Uno A3 板子對應的頻率為 490Hz，也就是週期為 0.002 秒左右，因此若 delay()的時間設定小於這個週期，會導致無法顯示出完整的方波，因此會導致 LED 不會有亮暗的變化。

實驗心得:

這次的實驗透過 Arduino 讓我對類比訊號轉數位訊號有更深刻的了解，平常在上微分方程和信號與系統的時候就有碰到將連續訊號轉換為離散訊號的問題，而這次的實驗與之有些許相同之處，也讓我更能體會 aliasing 所造成的訊號失真與量化誤差和 LSB 值對數位訊號轉換為類比訊號的影響，以及如何將誤差太大的訊號透過 Moving Average 的方式去進行過濾，很有趣的實驗。

實驗結論:

實驗 2-1 到 2-3 的重點是觀察類比訊號轉到數位訊號時，會因為取樣頻率與輸入信號的關係而導致 alasing 的產生，使得將數位訊號轉回類比訊號時，產生失真。其關係為當輸入信號的頻率超過取樣頻率的一半時，就會產生 aliasing。

實驗 2-4 的重點是觀察 Moving Average Filter 對一個 AM sine 訊號的過濾效果，經由實驗可以發現，經過過濾所產生的波型確實較沒有過濾的波形來的平滑，因此確定使用 Moving Average Filter 可以過濾雜訊。

實驗 2-5 的重點是將溫度感測器得到的輸出信號，透過 Arduino 的類比 pin 腳讀取，並最終將讀入的電壓信號轉換為實際的溫度。

實驗 2-6 的重點是選取適當的 delay()值，來使 Arduino 的輸出 pin 腳可以輸出至少一個完整的方波，並透過控制輸出方波的占空比，來達到控制 LED 燈的亮暗變化產生呼吸燈的效果。

Reference:

Arduino. (2024) Arduino. Retrieved from: https://www.arduino.cc/ (2024/03/09)

HAPLESS PIGEON. (unknown) SIGNAL PROCESSING: ALIASING AND FILTERING. Retrieved from: https://haplesspigeon.com/signal-processing-aliasing-and-filtering/ (2024/03/09)

中文百科。 (未知) 量化誤差。檢自: https://www.newton.com.tw/wiki/%E9%87%8F%E5%8C%96%E8%AA%A4%E5%B7%AE#google_vignette (2024/03/09)

CSDN. (2023/07/18) 信号采样基本概念 —— 4. 移动平均滤波（Moving Average Filtering）。檢自: https://blog.csdn.net/poisonchry/article/details/131158726 (2024/03/09)

維基百科。(2023/04/08) 移動平均。檢自: https://zh.wikipedia.org/zh-tw/%E7%A7%BB%E5%8B%95%E5%B9%B3%E5%9D%87 (2024/03/09)

百科知識。(未知) LM35。檢自: https://www.jendow.com.tw/wiki/lm35#:~:text=LM35%20%E6%98%AF%E7%94%B1National%20Semiconductor%20%E6%89%80%E7%94%9F%E7%94%A2%E7%9A%84%E6%BA%AB%E5%BA%A6%E6%84%9F%E6%B8%AC%E5%99%A8%EF%BC%8C%E5%85%B6%E8%BC%B8%E5%87%BA%E9%9

B%BB%E5%A3%93%E7%82%BA%E6%94%9D%E6%B0%8F%E6%BA%AB%E6%A8%99%E3%80%82%20LM35%E6%98%AF%E4%B8%80%E7%A8%AE%E5%BE%97%E5%88%B0%E5%BB%A3%E6%B3%9B%E4%BD%BF%E7%94%A8%E7%9A%84%E6%BA%AB%E5%BA%A6%E6%84%9F%E6%B8%AC%E5%99%A8%E3%80%82,%E7%94%B1%E6%96%BC%E5%AE%83%E6%8E%A1%E7%94%A8%E5%85%A7%E9%83%A8%E8%A3%9C%E5%84%9F%EF%BC%8C%E6%89%80%E4%BB%A5%E8%BC%B8%E5%87%BA%E5%8F%AF%E4%BB%A5%E5%BE%9E0%E2%84%83%E9%96%8B%E5%A7%8B%E3%80%82%20LM35%E6%9C%89%E5%A4%9A%E7%A8%AE%E4%B8%8D%E5%90%8C%E5%B0%81%E8%A3%9D%E5%9E%8B%E5%BC%8F%E3%80%82%20%E5%9C%A8%E5%B8%B8%E6%BA%AB%E4%B8%8B%EF%BC%8CLM35%20%E4%B8%8D%E9%9C%80%E8%A6%81%E9%A1%8D%E5%A4%96%E7%9A%84%E6%A0%A1%E6%BA%96%E8%99%95%E7%90%86%E5%8D%B3%E5%8F%AF%E9%81%94%E5%88%B0%20%C2%B11%2F4%E2%84%83%E7%9A%84%E6%BA%96%E7%A2%BA%E7%8E%87%E3%80%82 (2024/03/09)

OMRON. (未知) 溫度感測器。 檢自: https://www.omron.com.tw/solution/guideDetail/115#:~:text=%E2%97%8F%E5%8E%9F%E7%90%86%20%E5%88%A9%E7%94%A8%E4%B8%8D%E5%90%8C%E7%A8%AE%E9%87%91%E5%B1%AC%E9%96%93%E7%94%A2%E7%94%9F%E7%9A%84%E7%86%B1%E9%9B%BB%E5%8B%95%E5%8B%A2%E7%8F%BE%E8%B1%A1%EF%BC%88%E5%A1%9E%E8%B2%9D%E5%85%8B%E6%95%88%E6%87%89%EF%BC%89%E3%80%82,%E6%AD%A4%E7%A8%AE%E9%87%91%E5%B1%AC%E7%B7%9A%E7%9A%84%E7%B5%84%E5%90%88%E6%90%AD%E9%85%8D%E4%BE%BF%E7%A8%B1%E7%82%BA%E7%86%B1%E9%9B%BB%E5%81%B6%E3%80%82%20%E4%B8%AD%E9%96%93%E6%BA%AB%E5%BA%A6%E6%B3%95%E5%89%87%E8%88%87%E4%B8%AD%E9%96%93%E9%87%91%E5%B1%AC%E6%B3%95%E5%89%87%E7%86%B1%E9%9B%BB%E5%8B%95%E5%8B%A2%E7%9A%84%E5%A4%A7%E5%B0%8F%E5%8F%96%E6%B1%BA%E6%96%BC2%E7%A8%AE%E9%87%91%E5%B1%AC%E7%B7%9A%E7%9A%84%E6%9D%90%E8%B3%AA%E3%80%81%E6%B8%AC%E6%BA%AB%E6%8E%A5%E9%BB%9E%EF%BC%88%E7%86%B1%E6%8E%A5%E9%BB%9E%EF%BC%89%E5%8F%8A%E5%9F%BA%E6%BA%96%E6%8E%A5%E9%BB%9E%EF%BC%88%E5%86%B7%E6%8E%A5%E9%BB%9E%EF%BC%89%E7%9A%84%E6%BA%AB%E5%BA%A6%E5%B7%AE%EF%BC%8C%E8%87%B3%E6%96%BC%E4%B8%AD%E9%96%93%E9%83%A8%E4%BD%8D%E5%8D%B3%E4%BD%BF%E7%94%A2%E7%94%9F%E6%BA%AB%E5%BA%A6%E5%B7%AE%E4%BA%A6%E4%B8%8D%E6%9C%83%E9%80%A0%E6%88%90%E5%BD%B1%E9%9F%BF%E3%80%82 (2024/03/09)

各實驗所使用 Arduino code:

實驗 2-1、2-2、2-3:

```
//condition for exp1
//const int MAX_SIZE = 100;

//condition for exp2
//const int MAX_SIZE = 400;

//condition for exp3
const int MAX_SIZE = 200;

//array stores A/D data
int data[MAX_SIZE];

int out[MAX_SIZE];
static int cnt=0,fcnt=0;

void setup()
{
   //assign serial communication baud rate
   Serial.begin(9600);

   //uncomment in exp2-4 and exp3-3 connect 3.3V to "AREF" pin
   //adjust the dynamic range of the A/D From 5V to 3.3V
   analogReference(EXTERNAL);
}

void loop()
{
   if(cnt < MAX_SIZE)
   {
      //store data from A/D A0 pin
      for(cnt = 0;cnt < MAX_SIZE;cnt++)
         {
            data[cnt] = analogRead(A0);
         }
      //print data
      for(fcnt = 0;fcnt < MAX_SIZE;fcnt++)
      {
```

```
        Serial.println(data[fcnt]);
      }

      //print end mark
      Serial.println("finish!!");
   }
}
```

實驗 2-4:

```
//condition for exp4
const int MAX_SIZE = 200;

//array stores A/D data
int data[MAX_SIZE];
int out[MAX_SIZE];

void setup()
{
   //assign serial communication baud rate
   Serial.begin(9600);
}

void loop()
{


   static int cnt=0,fcnt=0;

   if(cnt < MAX_SIZE)
   {
      //lab2-4-1 part
      for(cnt = 0;cnt < MAX_SIZE;cnt++)
         {
            data[cnt] = analogRead(A0);
         }

      for(fcnt = 0;fcnt < MAX_SIZE;fcnt++)
      {
         Serial.println(data[fcnt]);
      }
```

```
    //print end mark
    Serial.println("finish!!");

    //lab2-4-2 part
    for(fcnt = 0;fcnt < MAX_SIZE;fcnt++)
      {
        if(2 <= fcnt <= (MAX_SIZE - 3))
        {
          out[fcnt] = (data[fcnt - 2] + data[fcnt - 1] + data[fcnt] + data[fcnt + 1] + data[fcnt + 2])/5;
        }
      }

    for(fcnt = 2;fcnt <= (MAX_SIZE - 3);fcnt++)
      {
        Serial.println(out[fcnt]);
      }
  }
}
```

實驗 2-5:

```
float tem,sensor;

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

void loop()
{
  sensor =    analogRead(A0);
  //for LM35
  tem = (sensor/1024*5)/10*1000;
  //for LMP36 (tinkercad)
  //tem = (sensor/1024*5*1000 - 500)/10; //Temp = ( Veri - 500 ) / 10
  //Serial.println("Temperature: ");
  Serial.println(tem);
  Serial.print("");
  //Serial.print("degree celsius");
  delay(100);
```

```
}
```

實驗 2-6:

```
int led = 9,gap = 5, initial = 255;

void setup()
{
   pinMode(led,OUTPUT);
}

void loop()
{
   if(initial == 0 || initial == 255)
   {
      gap = -gap;
   }
   analogWrite(led,initial);
   initial += gap;
   delay(45);
}
```