結報評分標準

圖表	數據	發現問題	電路分析	心得+結論	Reference
15%	15%	10%	30%	20%	10%

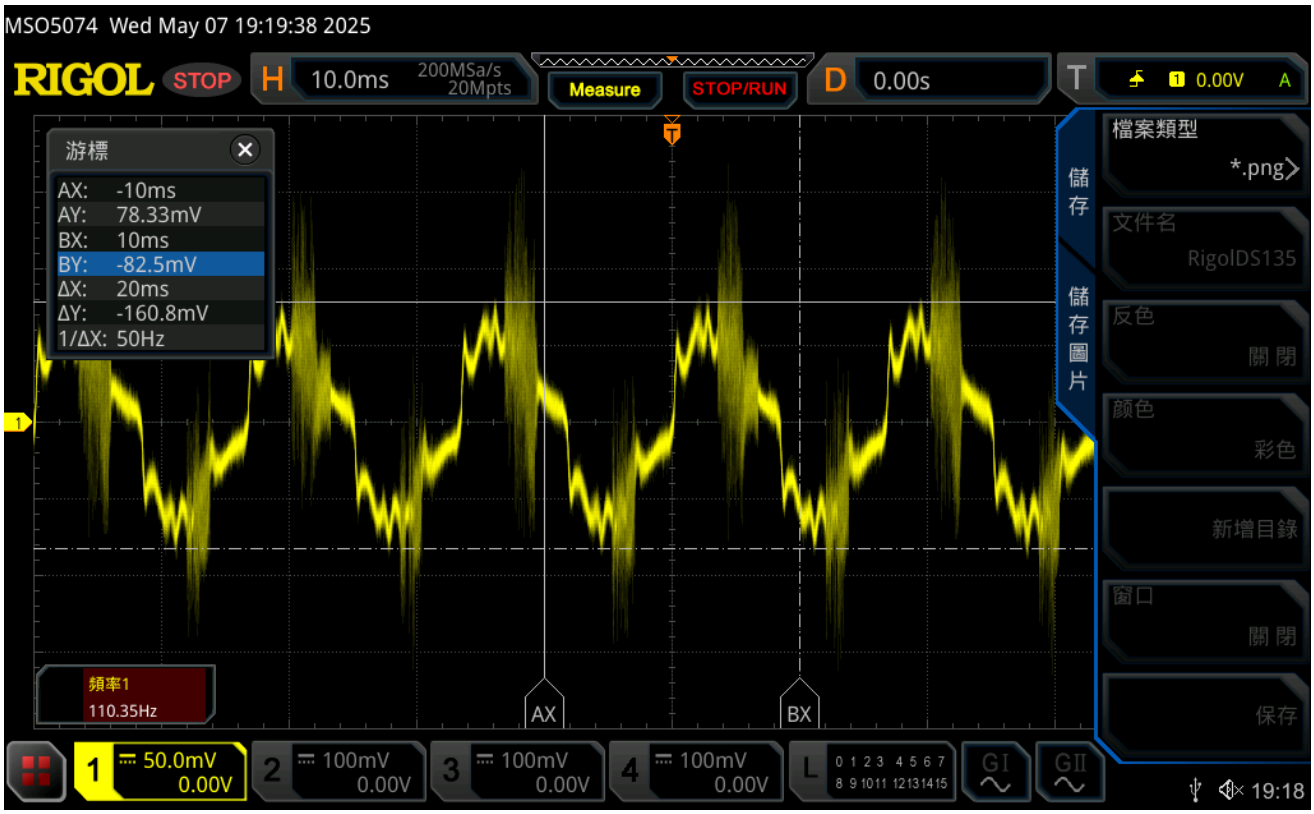
請假後補交結報的規定

1. 請假需依規定提出假單申請，並安排時間補做實驗並將核準過的假單截圖貼於下方，助教才會進行結報的批改。
2. 以請假日計算；需在一星期內完成補做驗，二個星期內補交結報(將結報交至 Delay 區)。逾時不進行結報批改。例如:3/1 請假，需在 3/8 前完成補做實驗，3/15 前補交結報。

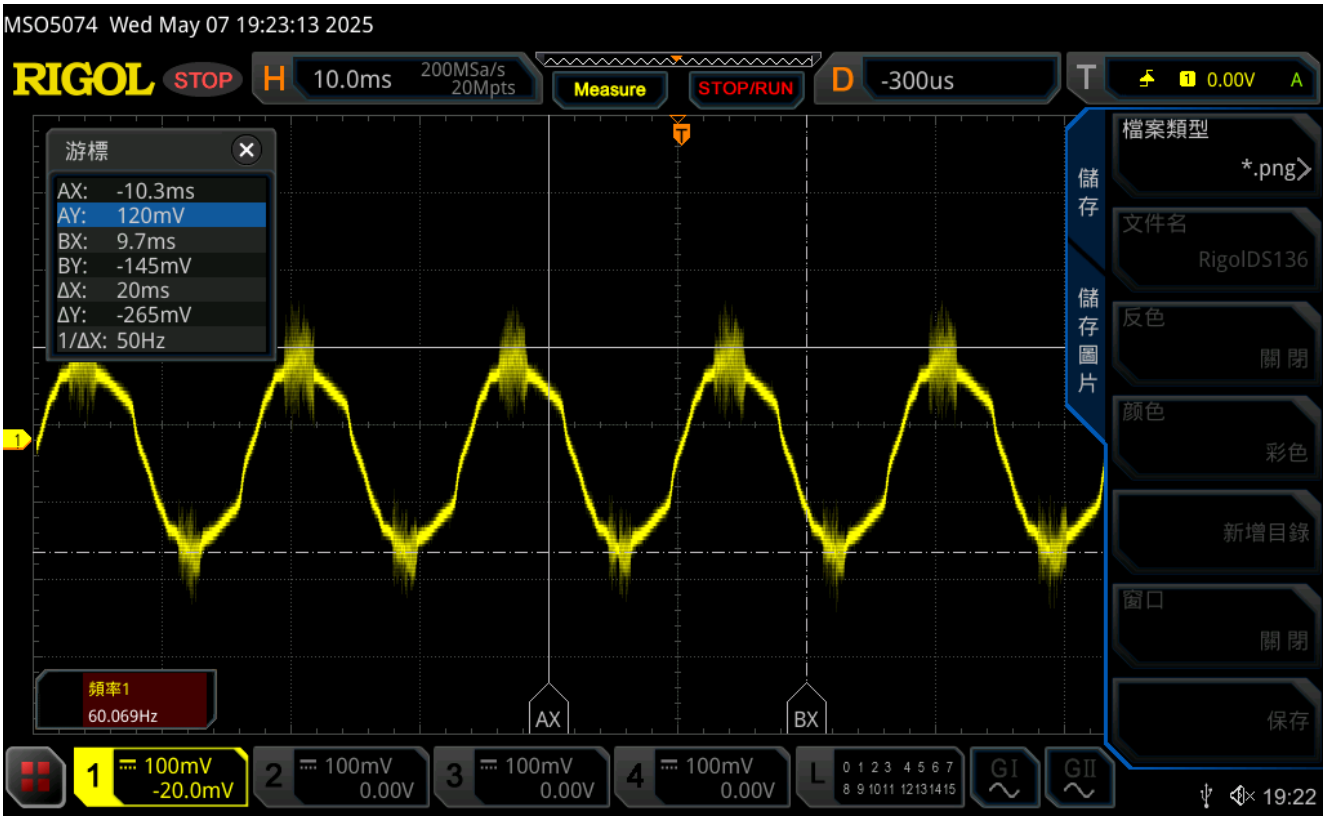
REPORT

Experiment 1: Noise Measurement

1. floating status waveform



2. With hand waveform



訊號分析

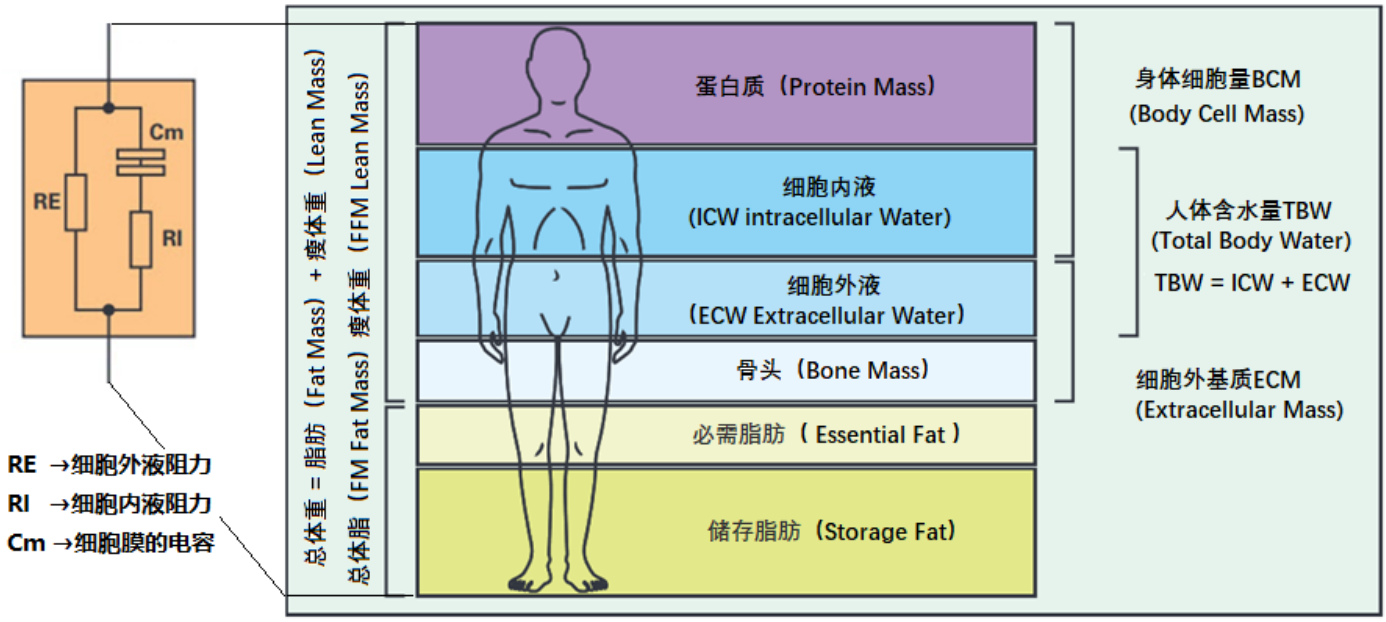
- **圖一**

  從圖一來看我們可以發現波型是非常的雜亂、隨機、不規則的，振幅相較於圖二來說較小、但是頻率較高，推測應該是受到環境的噪音影響，像是電磁干擾(EMI)、BNC cable 導線的雜訊、示波器內部的雜訊等種種原因。

- **圖二**

  當我們雙手各自握住 BNC cable 時，可以觀察到振幅有很明顯地增強、頻率從 110 Hz 衰弱到 60 Hz，我覺得造成這原因主要有以下兩種:

  1. 阻抗的差距: 空氣的阻抗近乎無限大，但人體的阻抗大致位於 2k ~ 2T Ω，所以才會造成振幅上的差異。
  2. 生物電訊號: 在頻率上，因為圖二的波型是有經過人體，所以相較於電磁干擾，人體的生物電訊號影響更鉅，像是心電訊號(ECG)、肌電訊號(EMG)等等，但是仍然小於來自電源的干擾。



▲ 人體組織阻抗分布

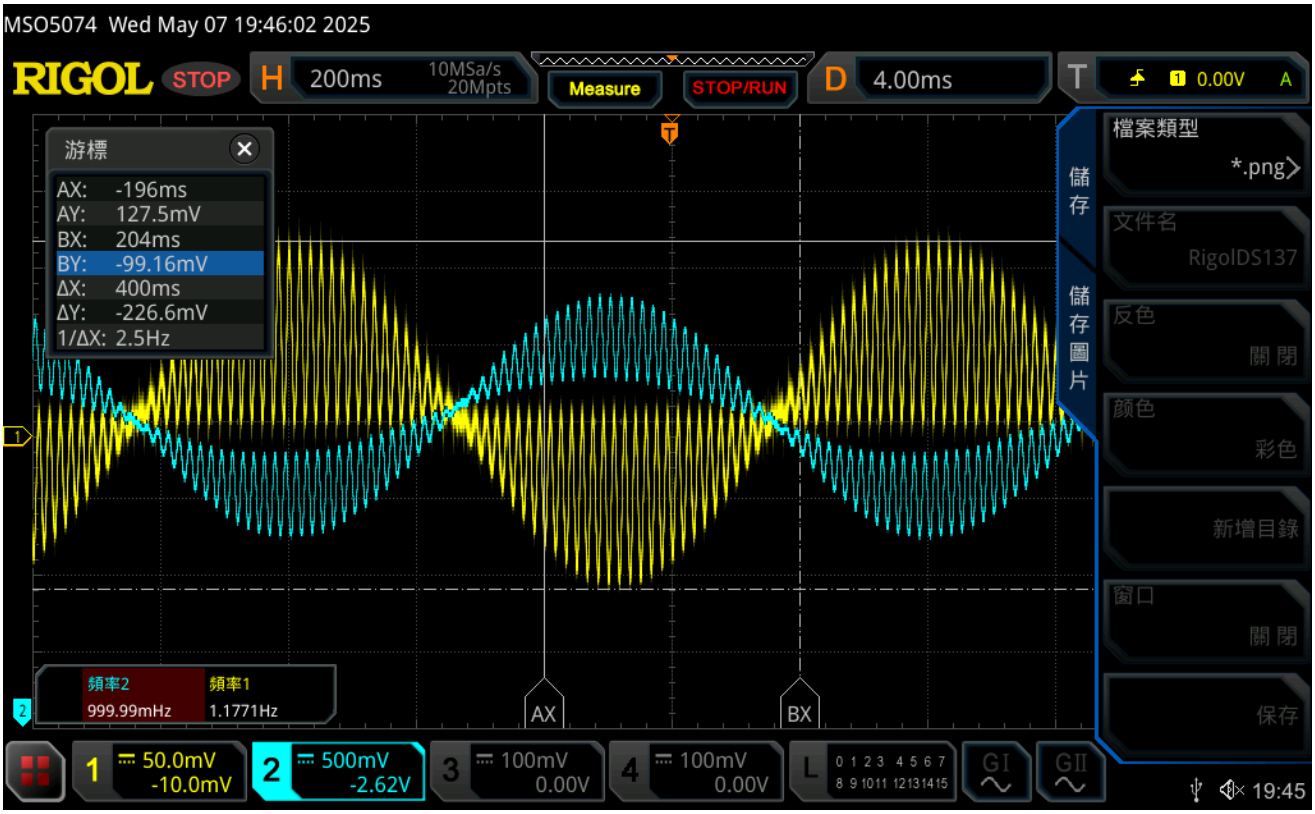
結論

在實驗一中我們將 BNC cable 分別空接、從雙手通過身體，透過操作讓我們認識到環境噪音、生物訊號的影響。

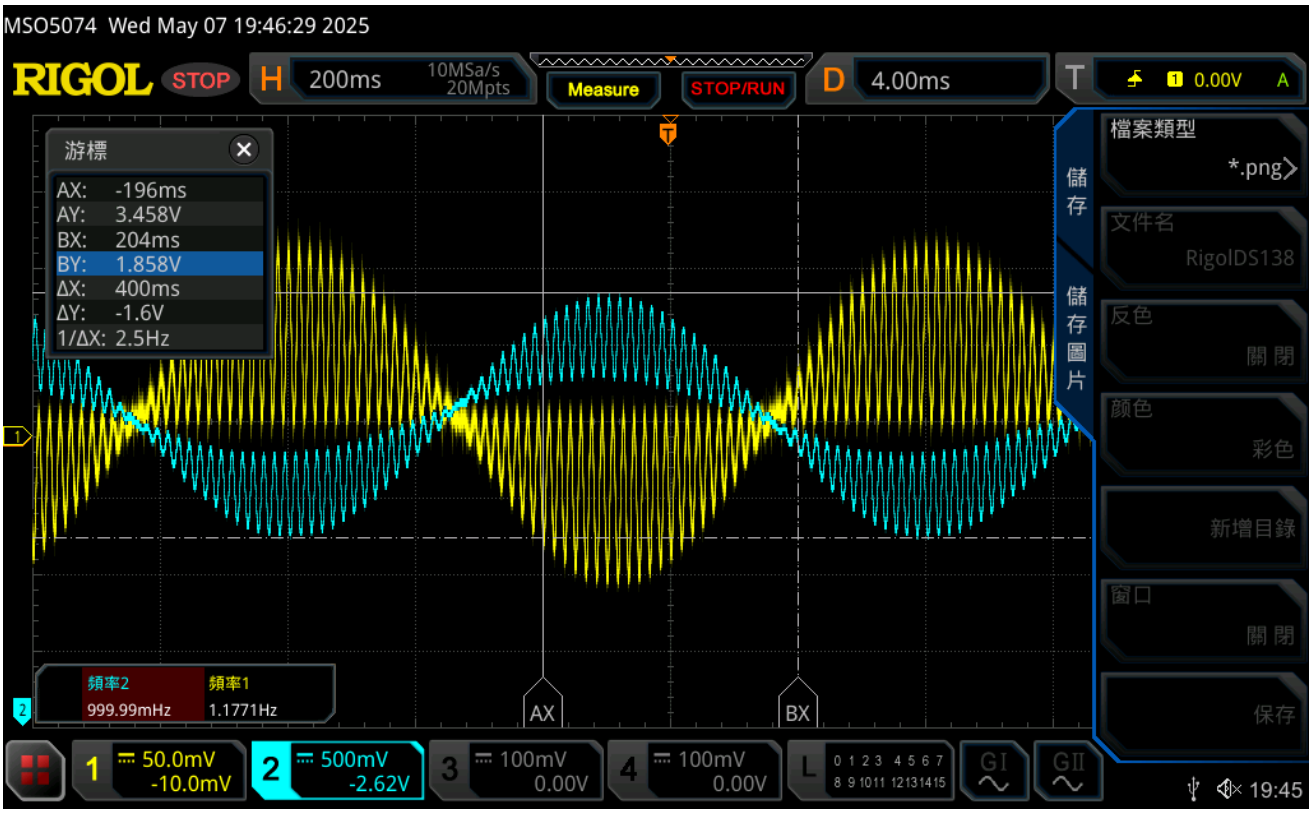
Experiment 2: Noise and filter

U1 Output & U4 Output

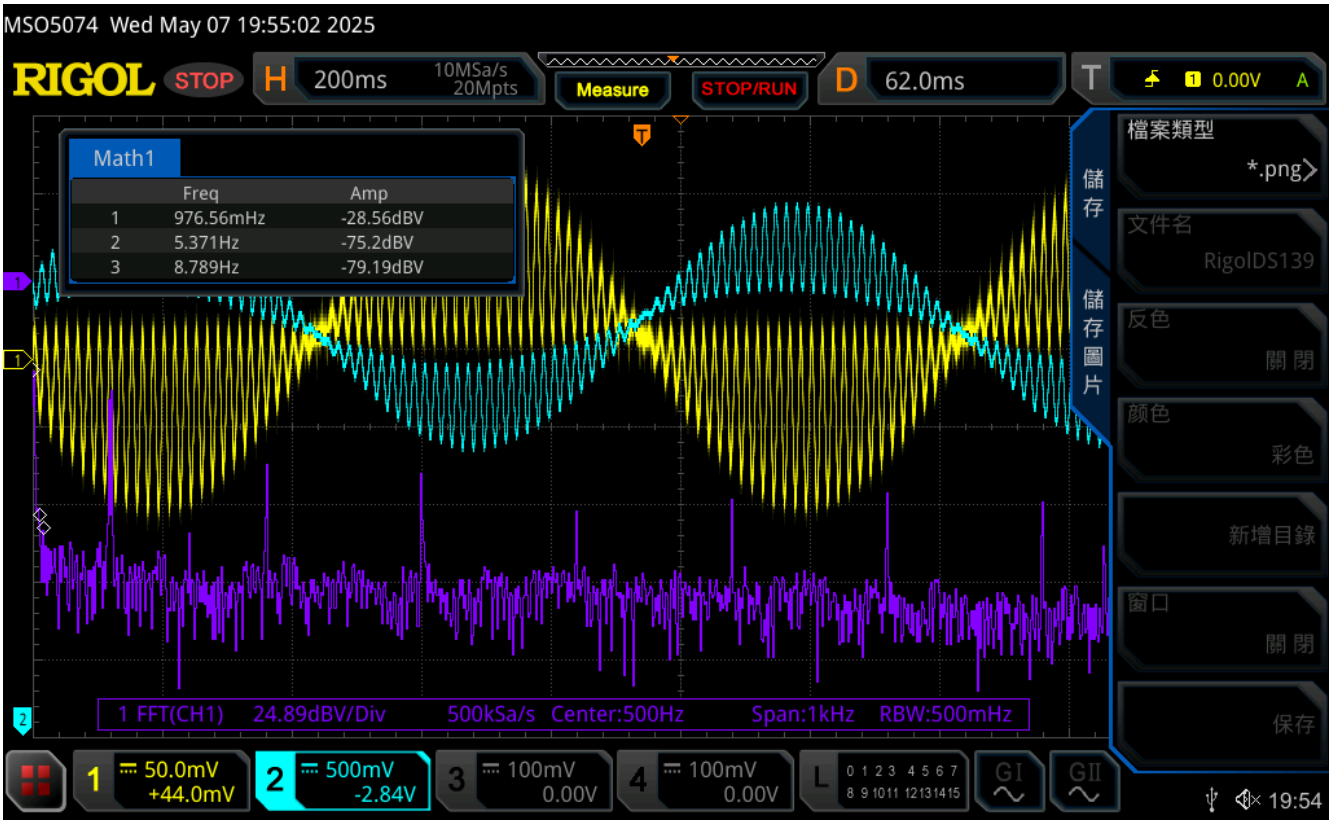
- **U1 Output 振幅測量**



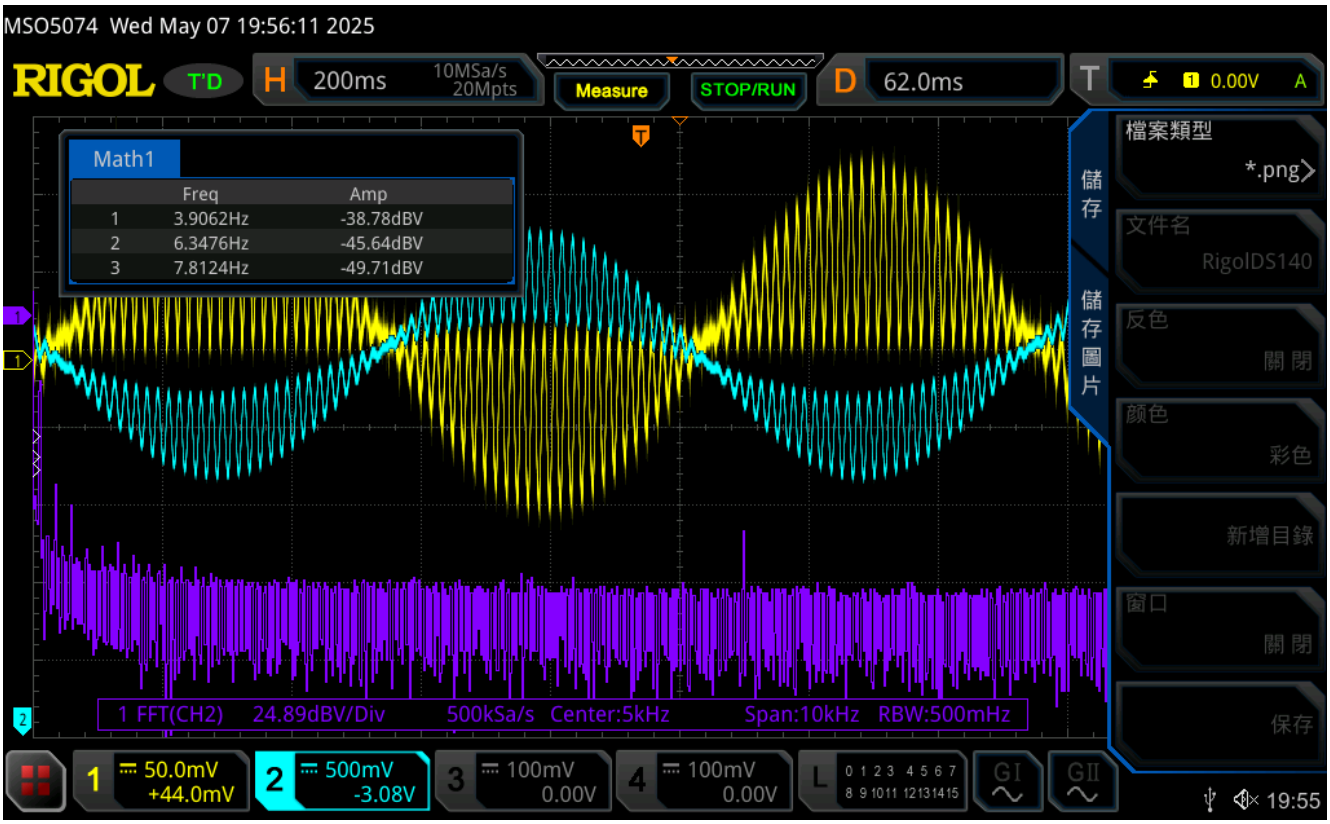
- **U4 Output 振幅測量**



U1 Output & U4 Output (FFT,U1)

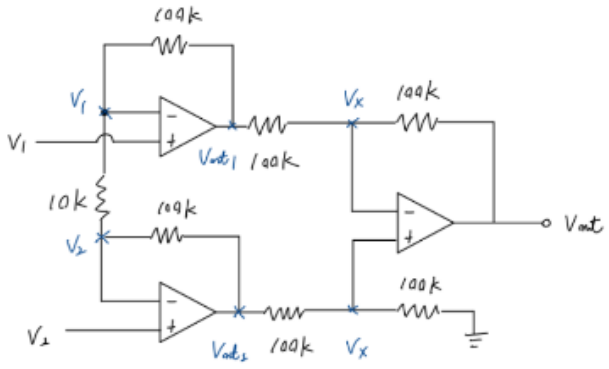


U1 Output & U4 Output (FFT,U4)



電路分析

我將電路分成兩部分討論，一個是 instrumentation amplifier，另一個是 bandpass filter，並假設 opamp 為理想$A_0 \to \infty$



$$\begin{cases} V_{out} = V_{out2} - V_{out1} \quad (\text{differential amp}). \quad ① \\ \dfrac{V_{out1} - V_1}{100k} = \dfrac{V_1 - V_2}{10k} = \dfrac{V_2 - V_{out2}}{100k} \quad ② \end{cases}$$

for ②.

$$V_{out1} - V_1 = 10(V_1 - V_2) \Rightarrow V_{out1} = 10(V_1 - V_2) + V_1$$

$$V_2 - V_{out2} = 10(V_1 - V_2) \Rightarrow V_{out2} = V_2 - 10(V_1 - V_2)$$

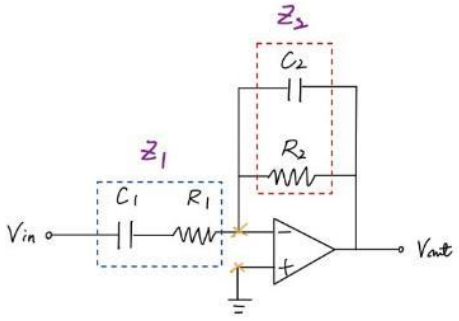
for ①.

$$V_{out} = V_{out2} - V_{out1} = V_2 - 20(V_1 - V_2) - V_1$$

$$= -21(V_1 - V_2)$$

▲instrumentation amplifier

Bandpass filter :



$$Z_1 = \frac{1}{sC_1} + R_1 = \frac{1 + sR_1C_1}{sC_1}$$

$$Z_2 = \frac{1}{sC_2} \parallel R_2 = \frac{R_2}{1 + sC_2R_2}$$

$$\frac{V_{in}(s) - 0}{Z_1} = \frac{0 - V_{out}(s)}{Z_2}$$

$$Z_2(V_{in}(s) - 0) = Z_1(0 - V_{out}(s)) \Rightarrow H(s) = -\frac{Z_2}{Z_1}$$

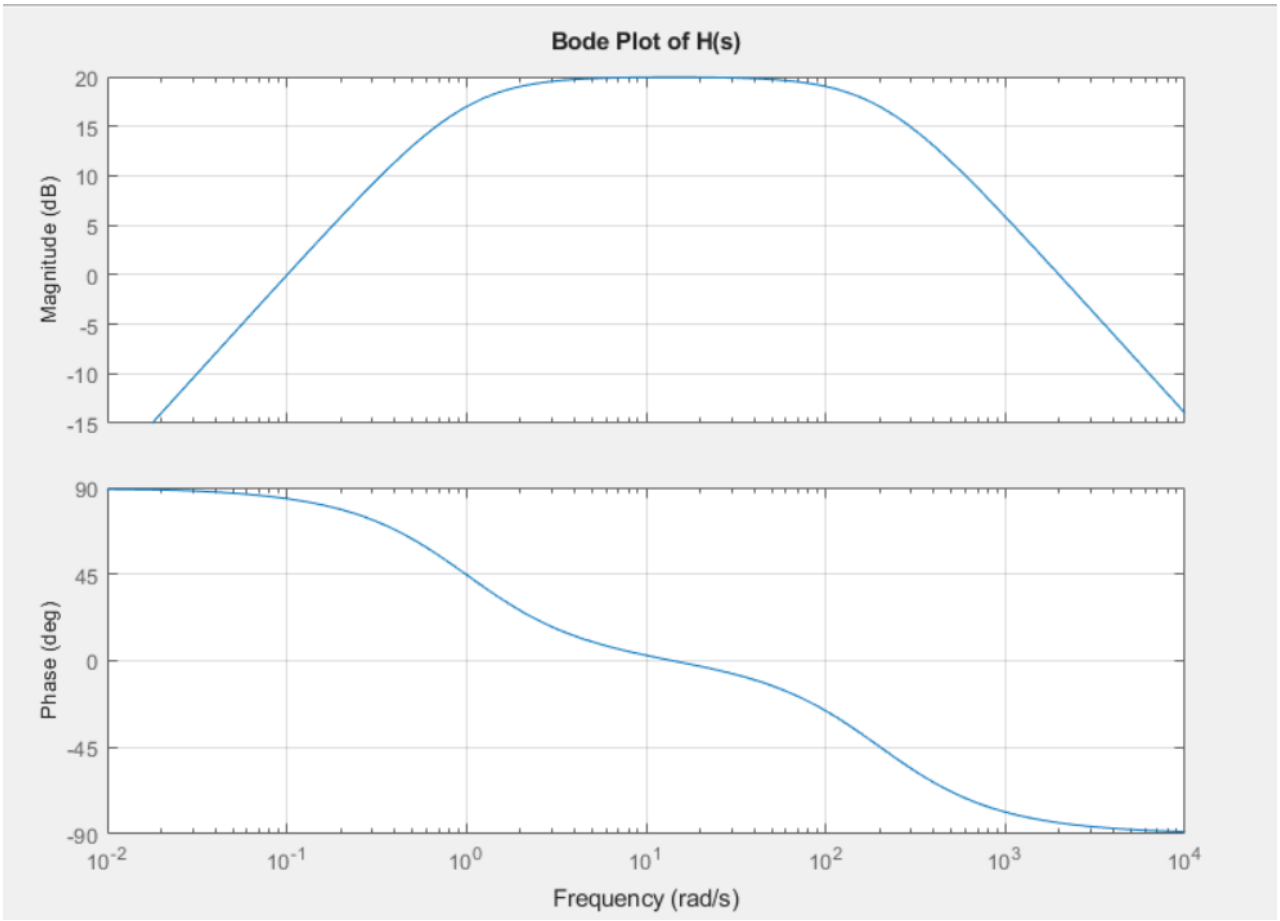
$$= -\frac{\frac{R_2}{1 + sR_2C_2}}{\frac{1 + sR_1C_1}{sC_1}} = -\frac{sR_2C_1}{(1 + sR_1C_1)(1 + sR_2C_2)}$$

透過計算我們可以得知 bandpass filter 的兩個 cut-off frequency 分別是:

$$f_{LdB} = \frac{\frac{R_2}{R_1}}{2\pi} = 1.59 \ Hz$$

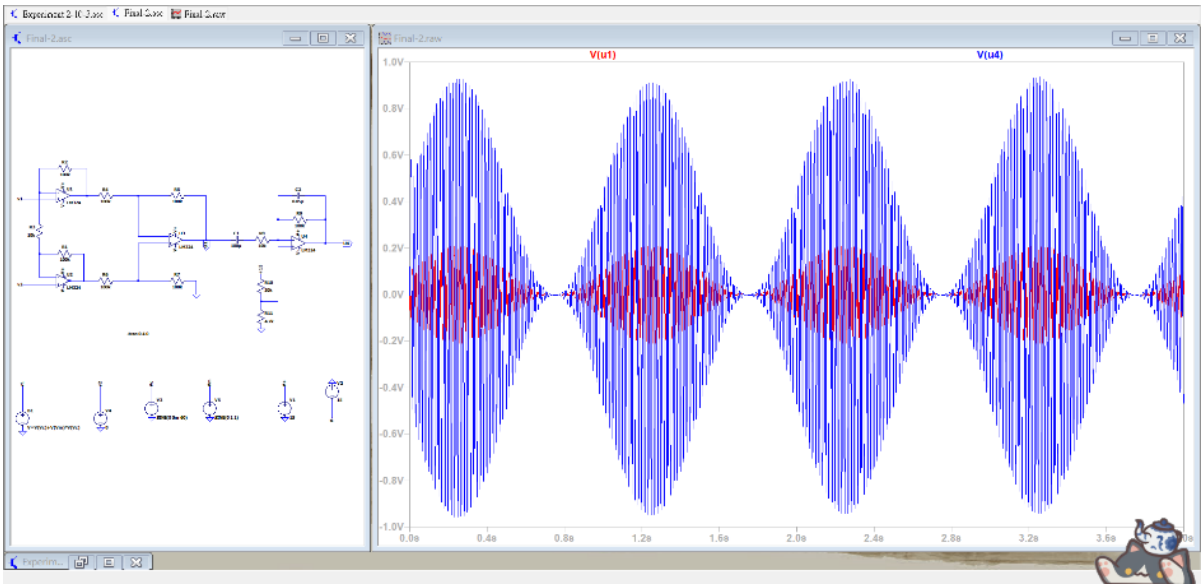
$$f_{HdB} = \frac{\frac{C_1}{C_2}}{2\pi} = 318.3 \ Hz$$

Matlab 模擬

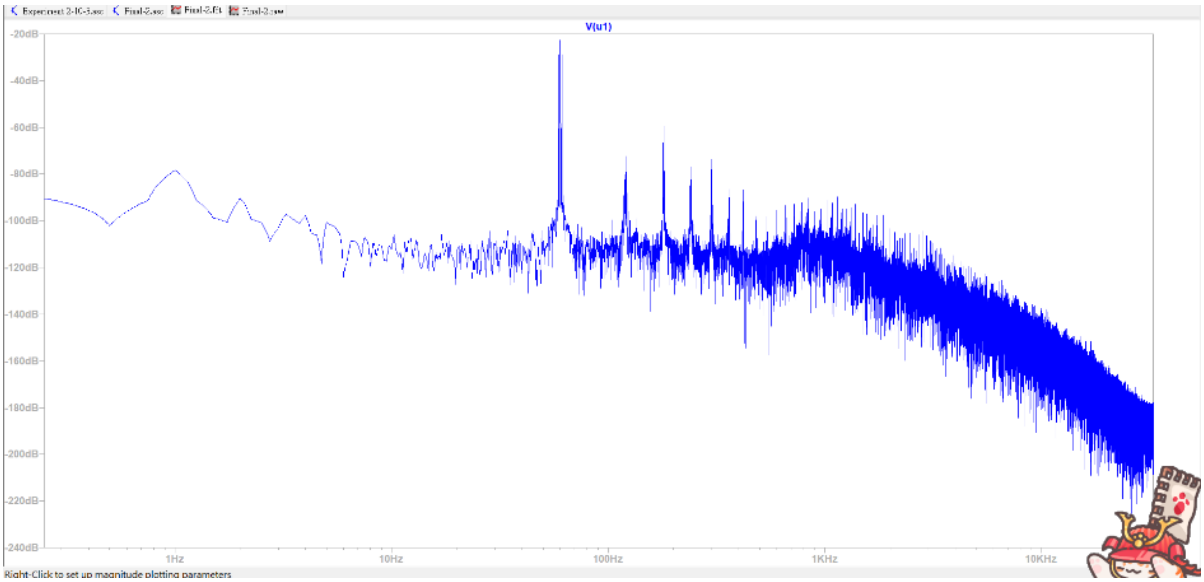


LTspice 模擬

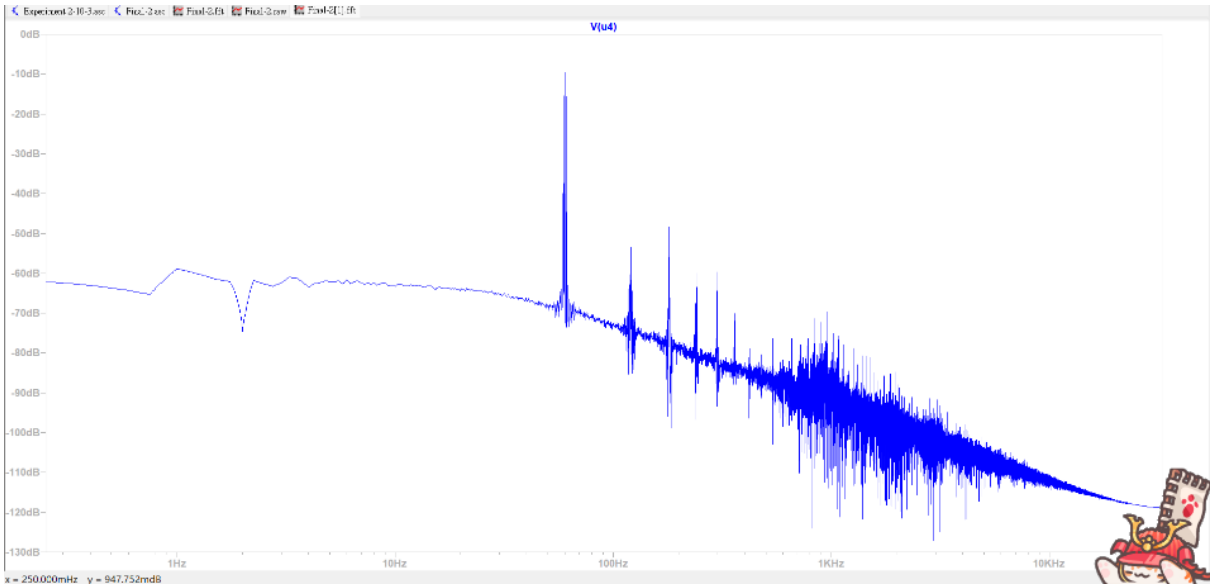
U1 Output & U4 Output



FFT,U1



FFT,U4



結論

在實驗二當中我們將 v1 輸入 AM 波、v2 輸入接地，觀察訊號經過 differential amplifier、bandpass filter 後的表現。

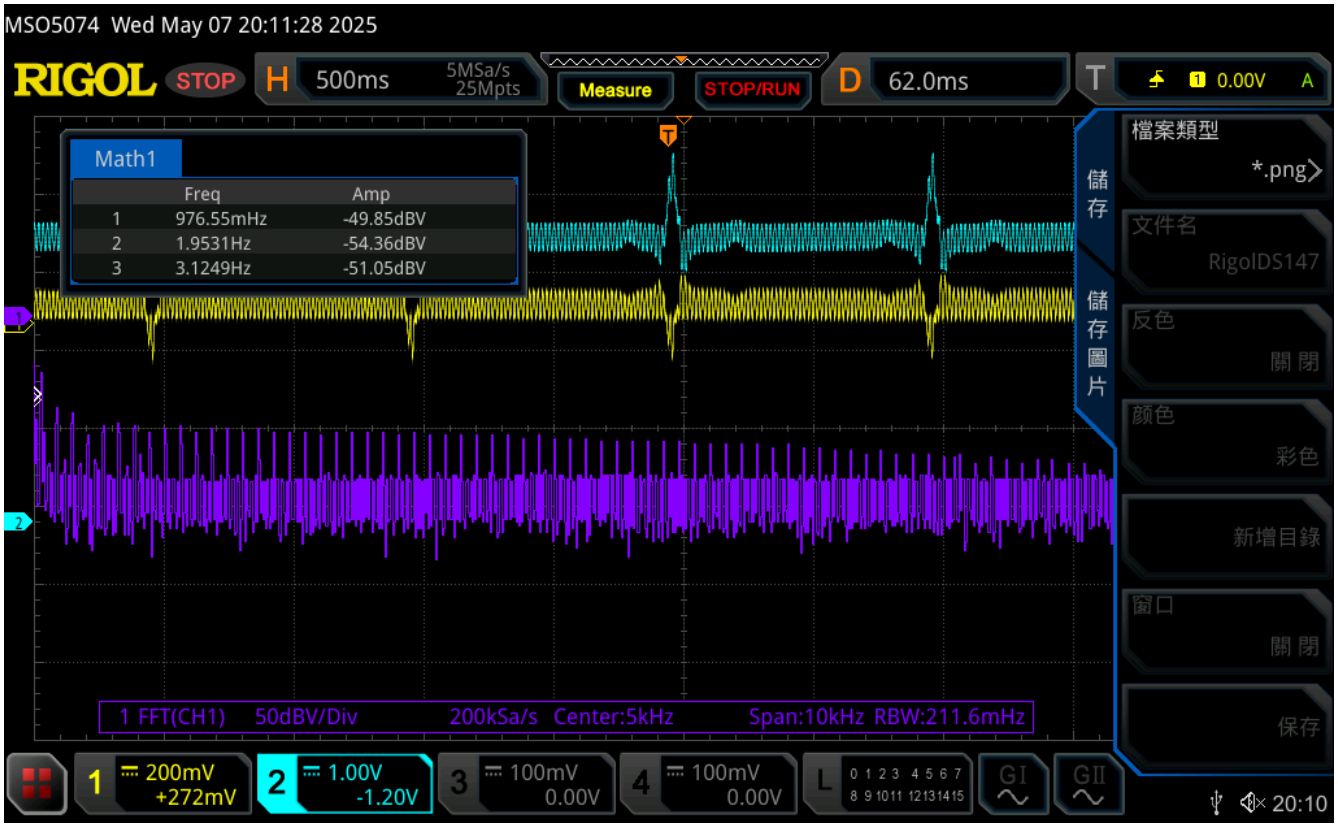
Experiment 3: Cardiac signal, Noise and filter

U1 Output & U4 Output

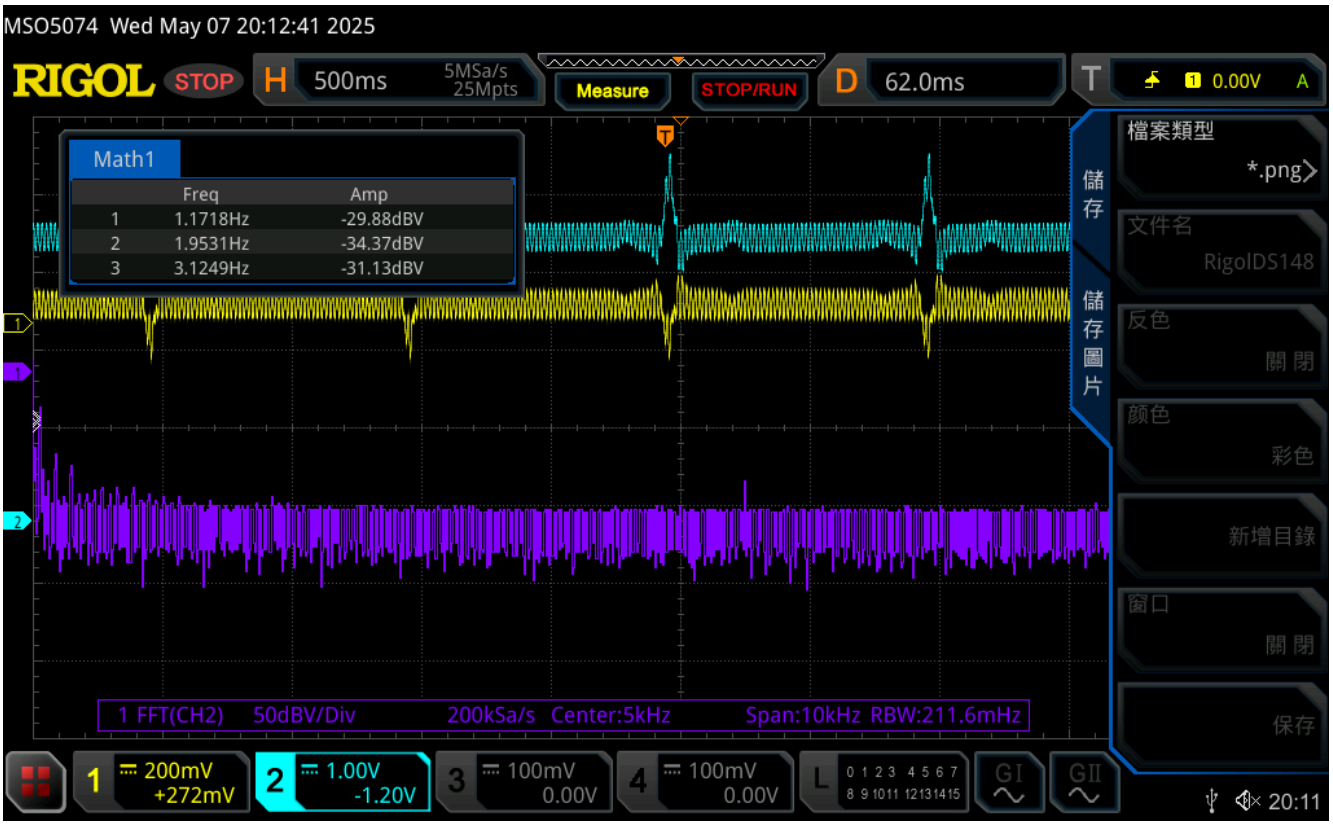
- **U1 Output 振幅測量**

- **U4 Output 振幅測量**

U1 Output & U4 Output(FFT,U1)

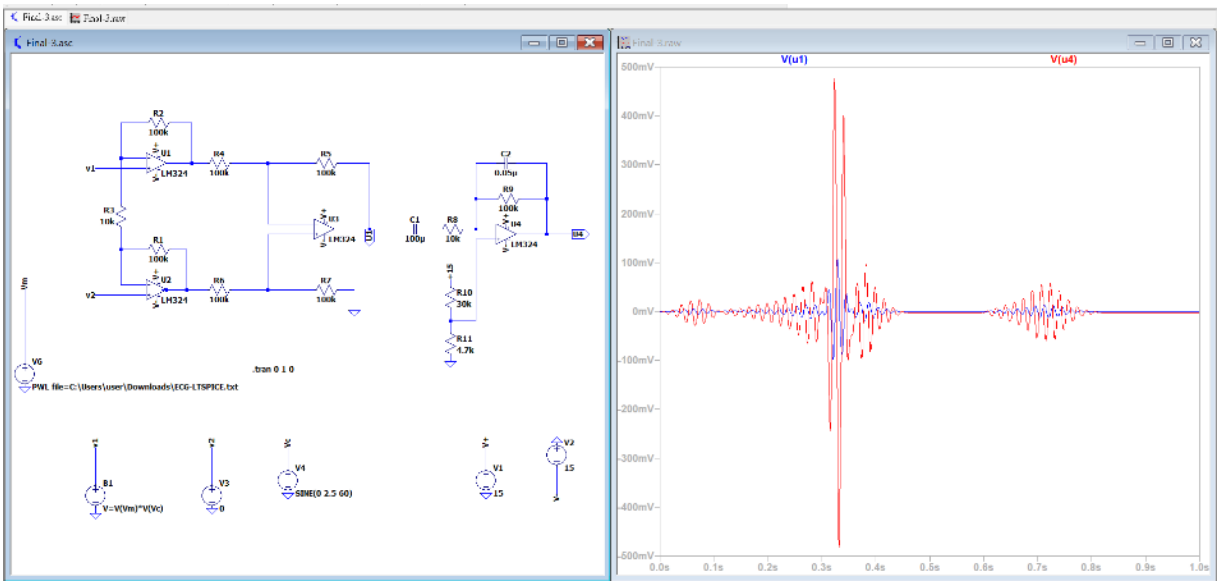


U1 Output & U4 Output(FFT,U4)

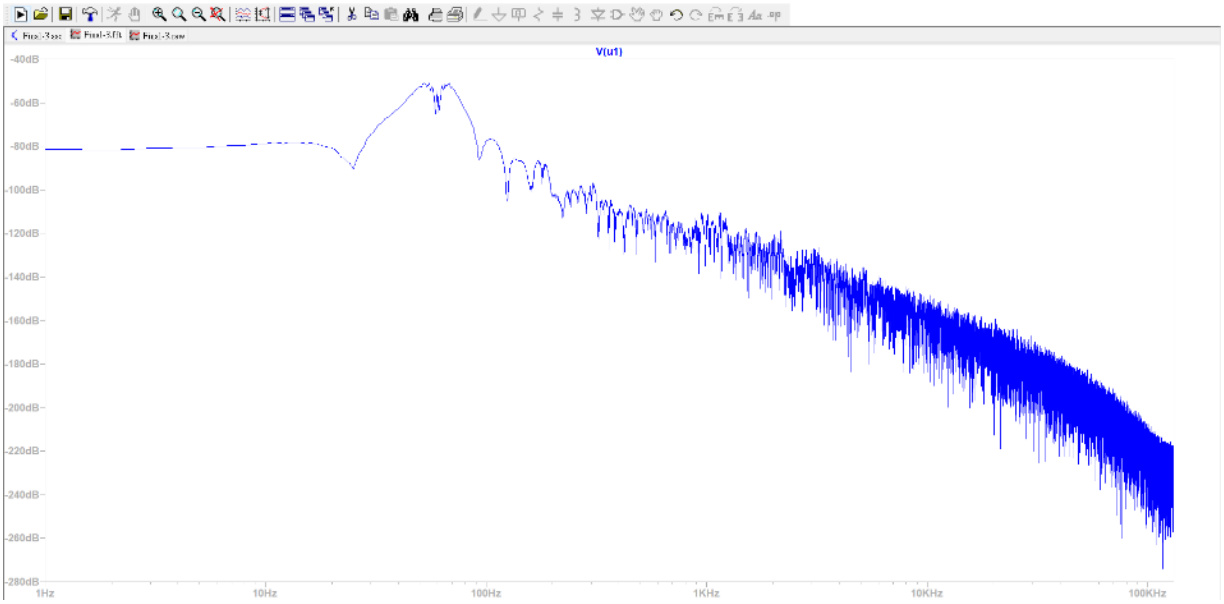


LTspice 模擬

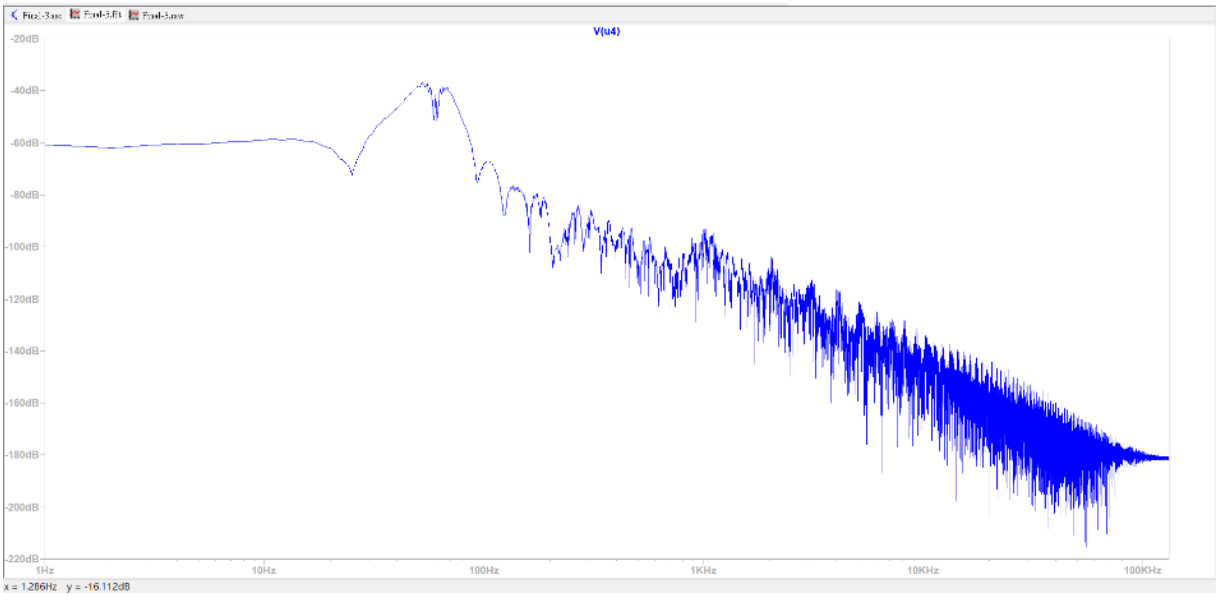
U1 Output & U4 Output



FFT,U1



FFT,U4

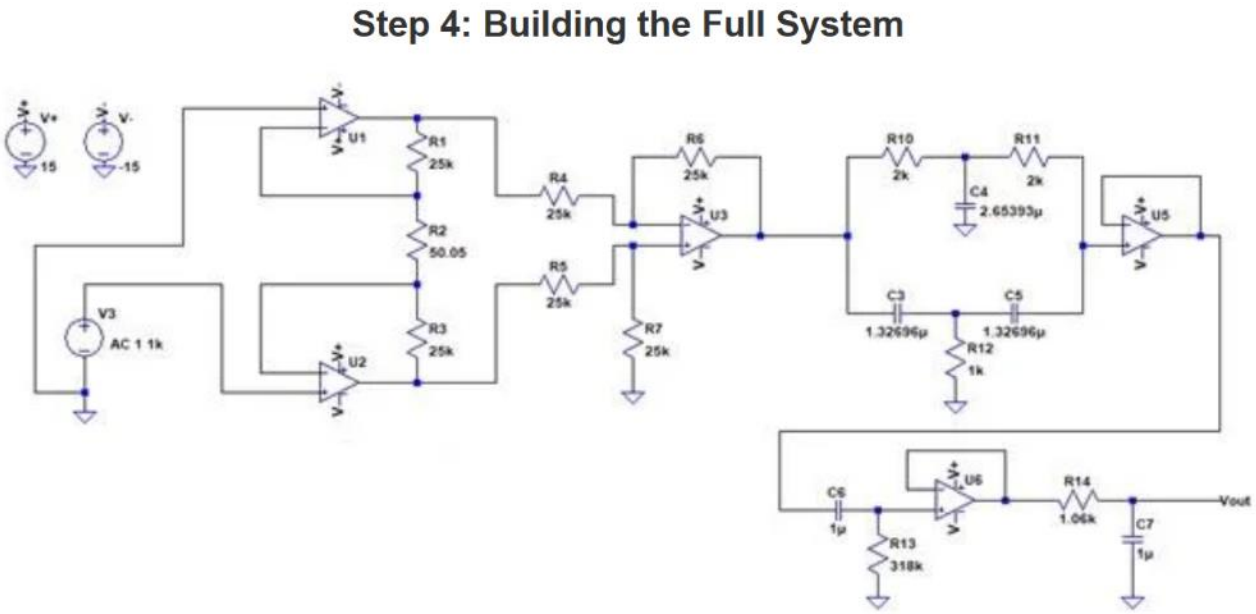


問題

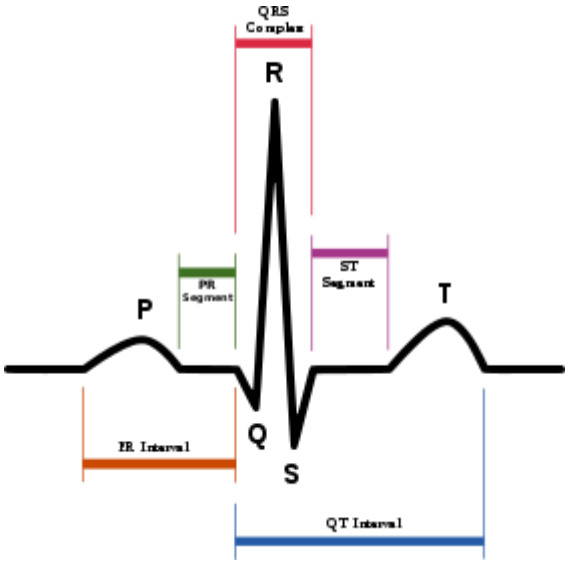
Q：如何在 LTspice 中模擬 Cardiac 訊號 ？

我主要找到三種方法 ：

- 方法 1: 用 instrumentation amplifier、bandpass filter、Notch filter 三要素構成的電路



產生的 Cardiac waveform :

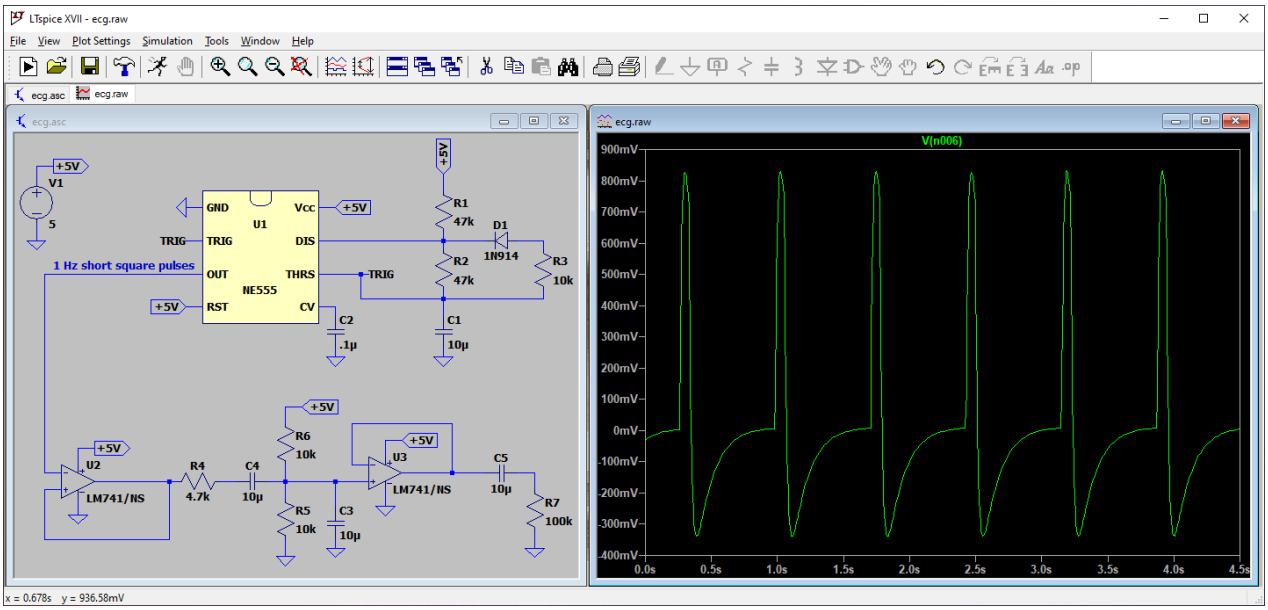


優點 ： 產生的 Cardiac 最接近函數產生器的 Cardiac 訊號。

缺點 ： 只能用 LTspice 實際模擬出來，非常耗時間。

網址 ：https://www.instructables.com/ECG-Signal-Modeling-in-LTspice/

- 方法 2: 用 555 timer 產生類似心電的波動



作者還非常好心的開源給大家使用

網址 ：

https://github.com/swharden/SWHarden.com/blob/main/content/blog/2020-09-27-ecg-simulator-circuit.md

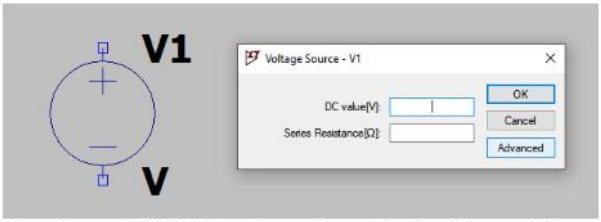
但卻不知道是什麼不知名的原因讓這 ECG 訊號產生不出這效果…

- 方法 3: 用網路整理好的 txt 檔引入作為一顆特殊 voltage component (我引用寫的 source)

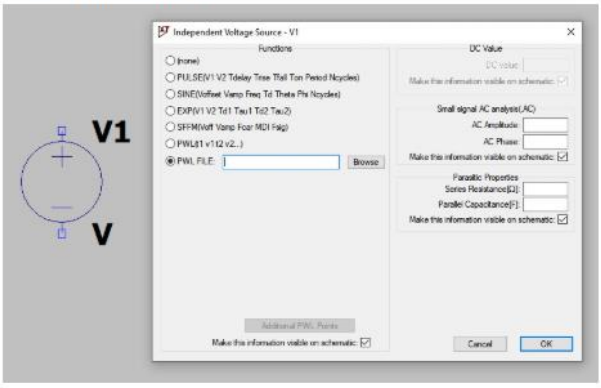
– Method: 1

The ECG source is implemented using a Piecewise Linear source. To do this we should create a text file with the values of ECG voltages for the corresponding time. I've already done that. Download the text file here.

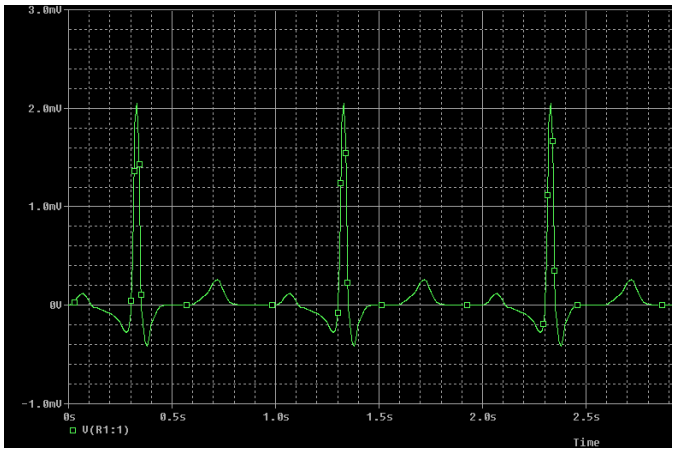
- Place a Voltage source in the schematic
- Right-click the voltage source and click 'Advanced'



- Then choose PWL FILE option and provide the address of the text file uploaded above.



我們透過 voltage advanced 選項裡勾選 PWL 引入別人寫好的包裝並使用，然後就能夠吳統產生 ECG 訊號!



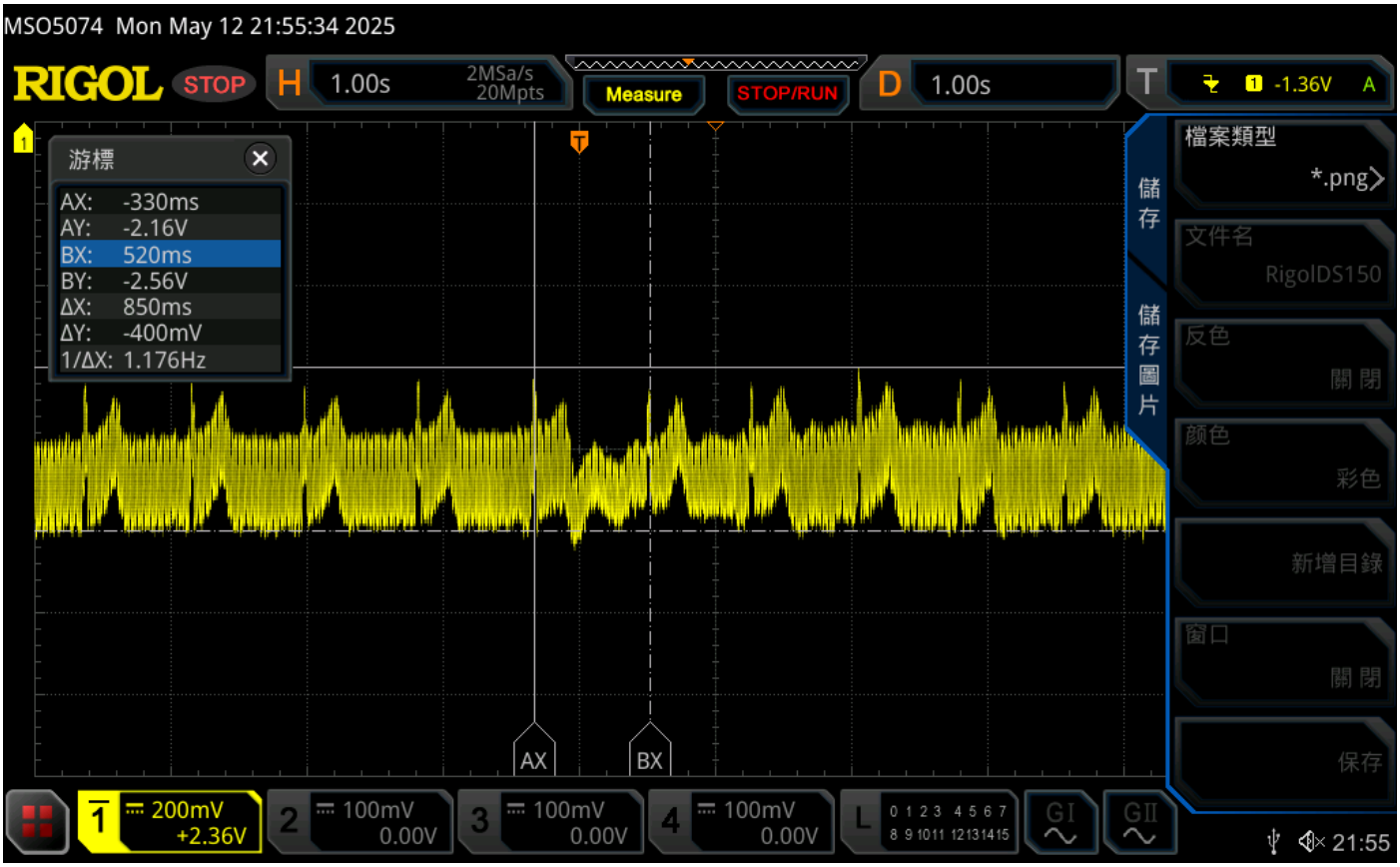
▲以上是產生的 Cardiac function

結論

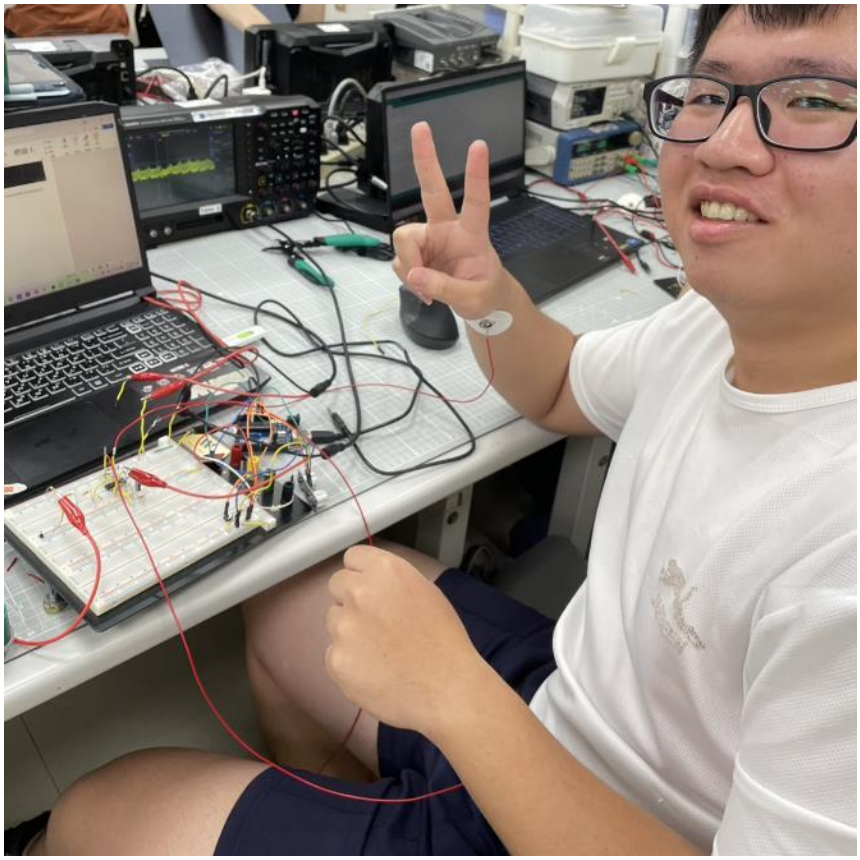
實驗三與實驗二的設置差不多，同樣是 instrumentation amplifier & bandpass filter 的組合，但不同的是這次是接上 function generator 的 Cardiac(心臟)訊號模擬真實的心電訊號。

Experiment 4: human body Cardiac signal

1. Cardiac waveform:



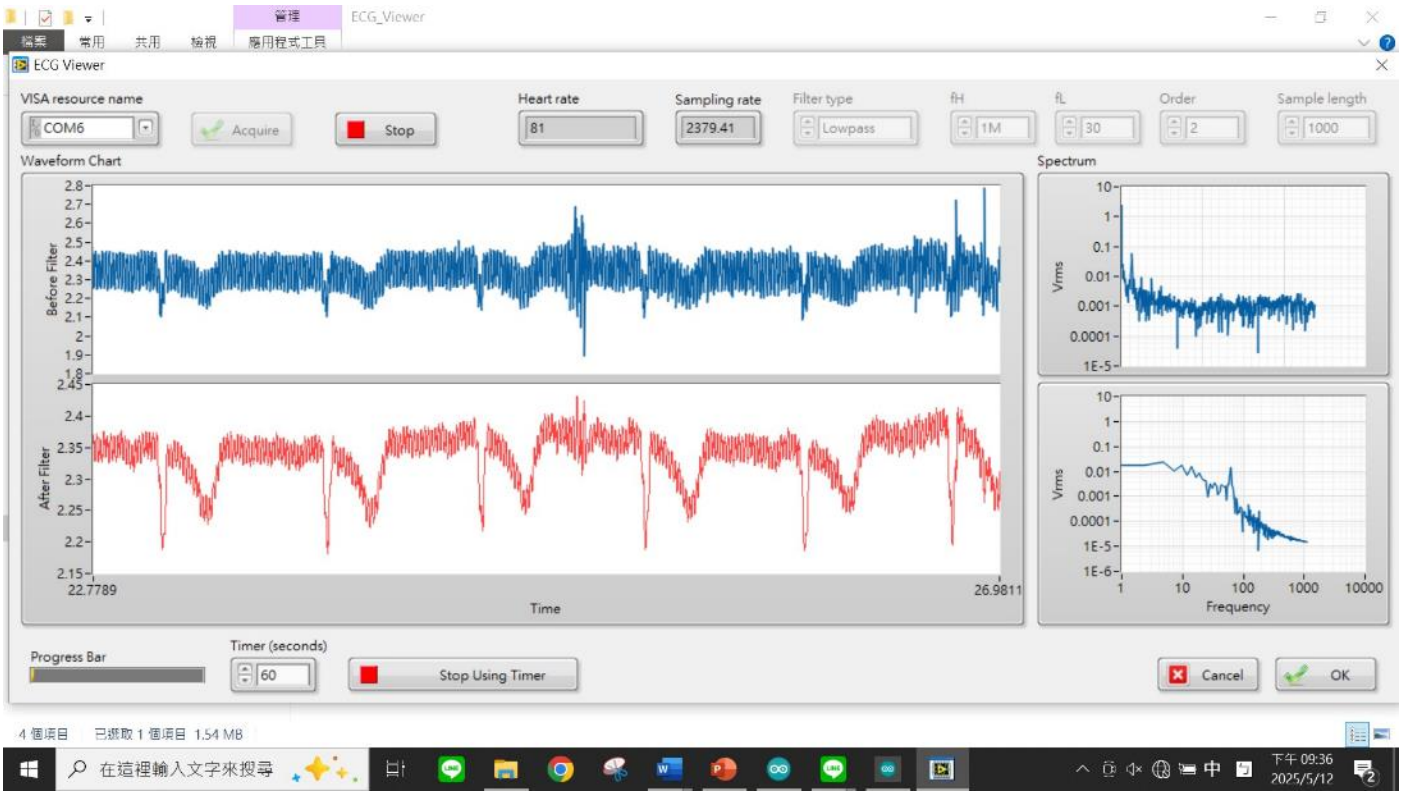
2. A photo about real experiment environment. (including yourself, your circuit and the instruments you used.)



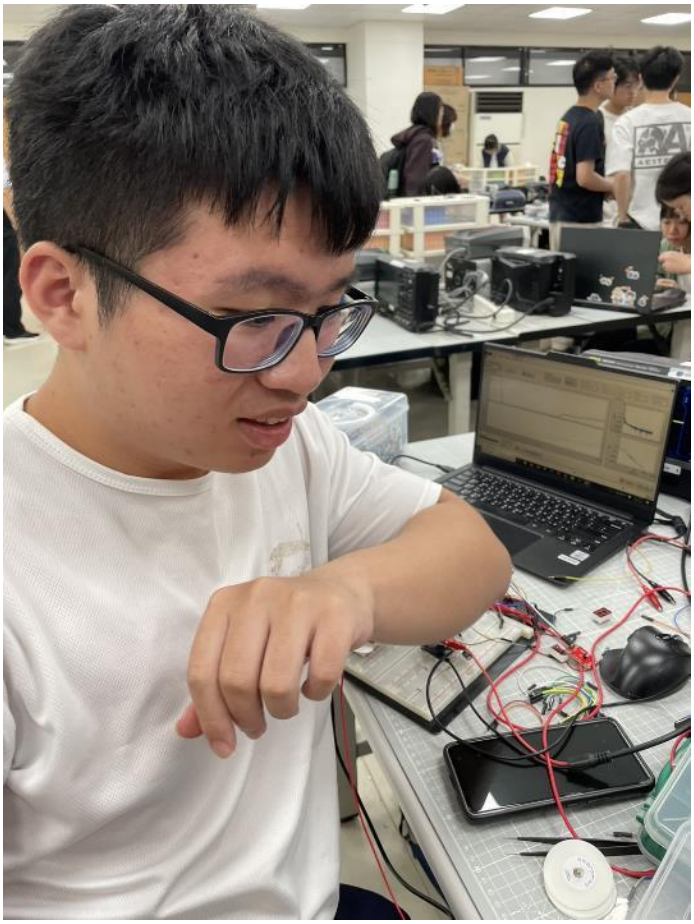
Experiment 5: human body Cardiac signal and Arduino ADC

1. Screenshot of your lab result (including Cardiac waveform, heart rate, etc.)

Heart rate = 81 bpm



2. A photo about real experiment environment. (including yourself, your circuit and the instruments you used.)

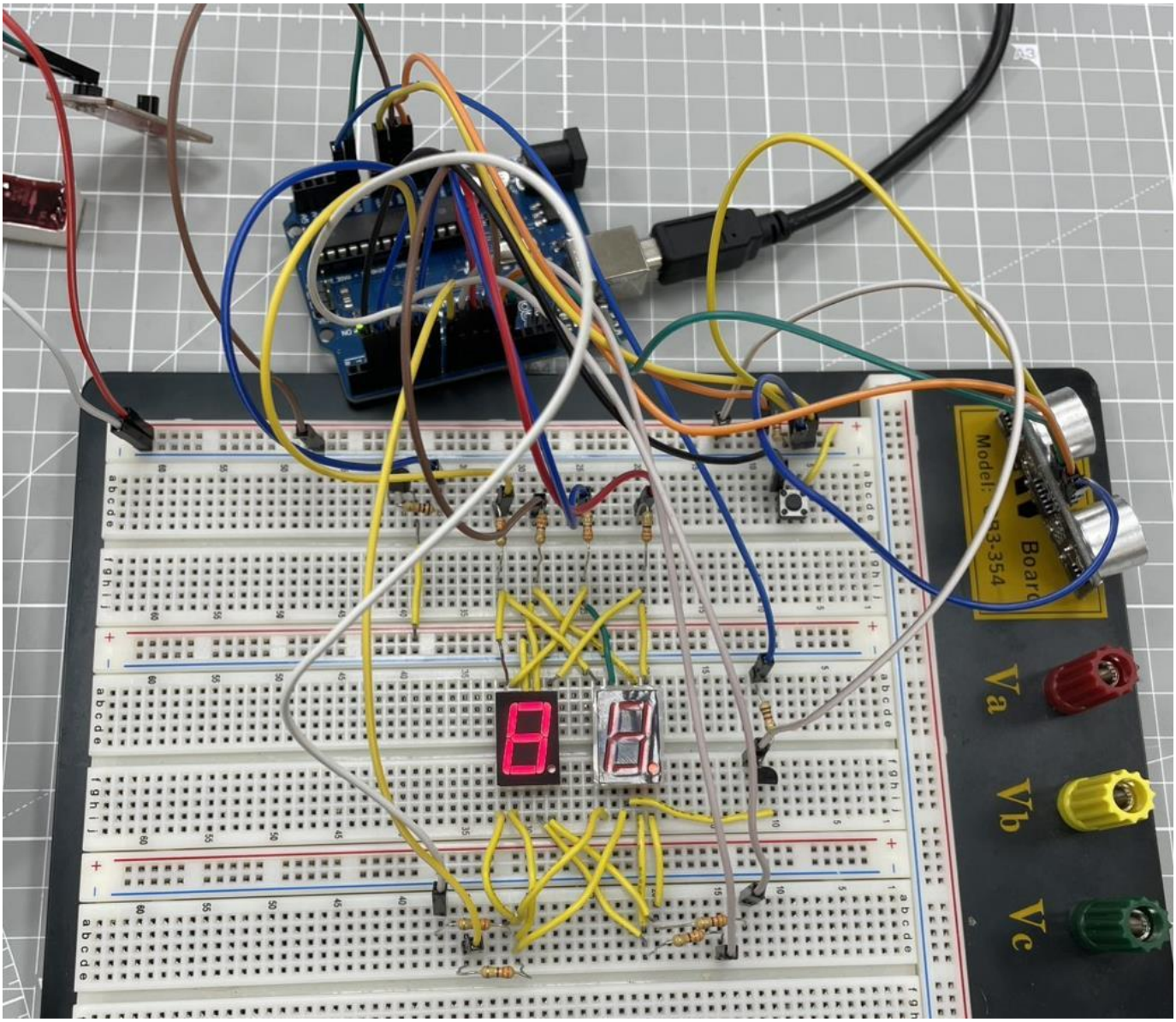


結論(4&5)

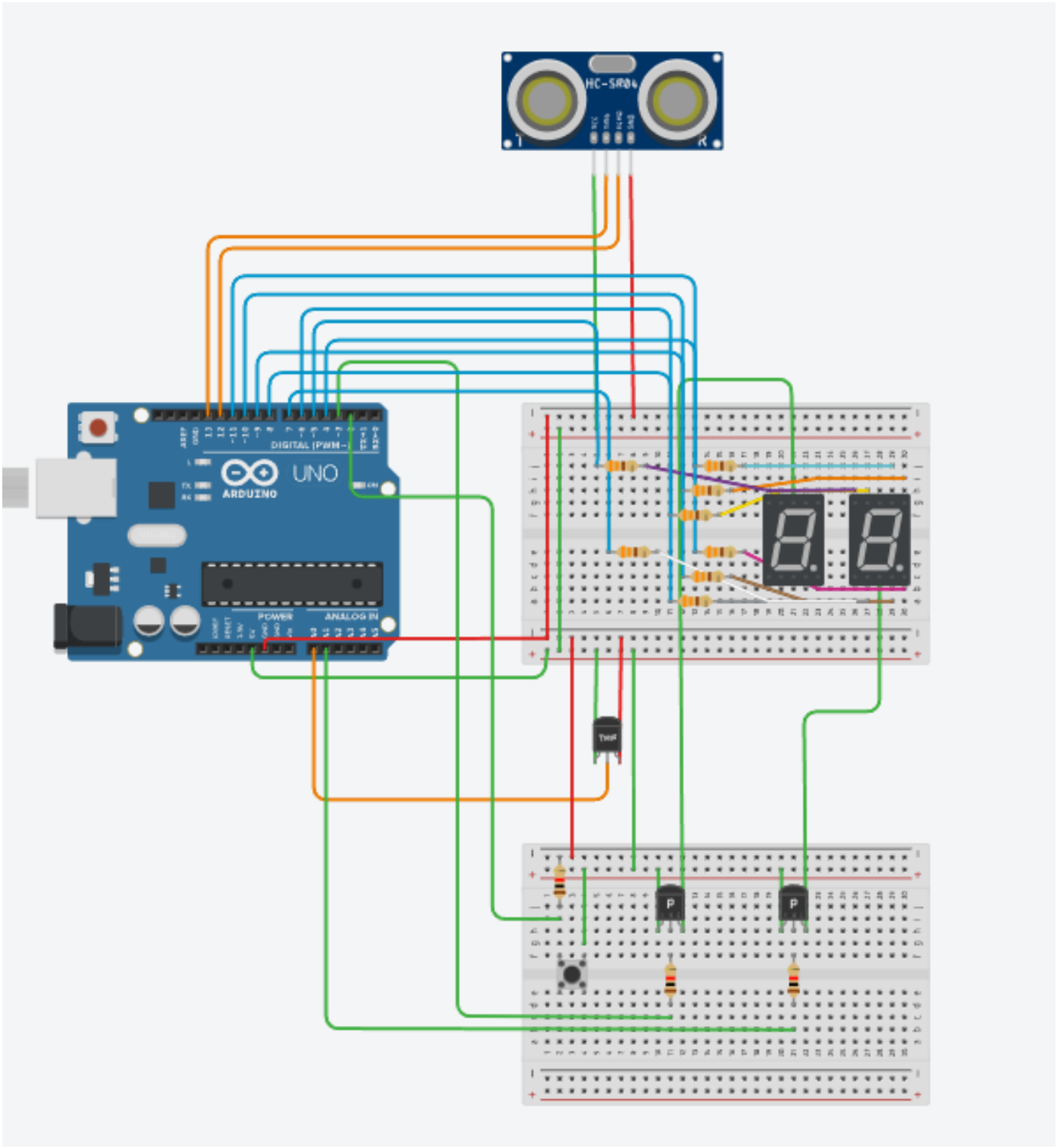
在實驗四和五當中我們把函數產生器的訊號移除，實際接上電極貼片測量生體的 ESG 訊號，並用示波器顯示出來，確認完訊號的正確與否後，換接上 Arduino 將訊號顯示在 ESG viewer 當中，觀察並測量 HR 等等資訊。

Experiment 6: Arduino project

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



▲ 以上是電路接線實際狀況



▲ 以上是 Tinkercad 模擬的電路狀況

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

程式碼分析

```
#define KEY_PIN 2
#define TRIGGER_PIN 13
#define ECHO_PIN 12
#define A 11
#define B 10
#define C 9
#define D 8
#define E 7
#define F 6
#define G 5
#define DP 4
#define Control_left 3
#define Control_right A1
```

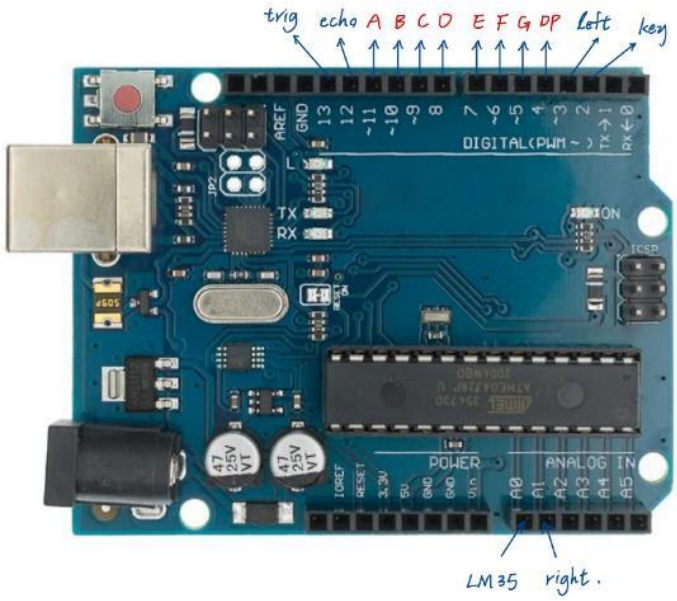
▲ 圖(一)

```
bool light_state;
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(9600);
    pinMode(A0, INPUT);
    pinMode(KEY_PIN, INPUT);
    pinMode(TRIGGER_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    pinMode(Control_left, OUTPUT);
    pinMode(Control_right, OUTPUT);
    // 設定腳位 11 到 3 為 OUTPUT
    for (int pin = 4; pin <= 11; pin++)
    {
        pinMode(pin, OUTPUT);
    }
    light_state = false;
    digitalWrite(Control_left, HIGH);
    digitalWrite(Control_right, HIGH);
}
```

▲ 圖(二)

說明

- 圖(一)：在這裡我用 # define 定義字元將 Arduino UNO 版上的腳位命名成我們實際控制時所稱的代號，pin 13、12 主要是超音波感測器用(triger 腳位控制超音波發送，echo 腳位感測之前 triger 所發送的超音波)，pin 11 ~ 4 主要是負責控制七段顯示器的顯示位置，依序對入顯示 A~DP 的位子，pin 3、A1 是用來控制 PNP 開關，在這裡因為 digital pin 不夠所以就用 analog pin 代替(A1)。
- 圖(二)：在 setup function 當中我先將 PC 端與 Arduino UNO 板間加上連結，以利之後用 serial monitor debug，並在這裡加上 light_state 變數，這變數紀錄、控管七段顯示器是要顯示溫度感測的資料還是超音波測具的資料。



```
36  void loop()
37  {
38      if (digitalRead(KEY_PIN) == HIGH)
39      {
40          light_state = !light_state;
41      }
42      delay(200);
43
44      if (light_state)
45      {
46          float distance = getDistance(TRIGGER_PIN, ECHO_PIN);
47          Serial.print("Distance: ");
48          Serial.print(distance);
49          Serial.print(" cm ");
50          Serial.print(int(distance / 100));
51          Serial.print(" ");
52          Serial.print(int(distance) % 100 / 10);
53          Serial.print(" ");
54          Serial.println(int(distance) % 10);
55          showDistance(distance);
56      }
57      else
58      {
59          float tmp = getTemperature();
60          getTemperature();
61          Serial.print("tmp: ");
62          Serial.print(tmp);
63          Serial.print(" C ");
64          Serial.print(int(tmp / 10) % 10);
65          Serial.print(" ");
66          Serial.print(int(tmp) % 10);
67          Serial.print(" ");
68          Serial.println(int(tmp * 10) % 10);
69          showTemp(tmp);
70      }
71  }
```

在 loop function 當中我先判別是否有按下按鍵，如果有就改變 light_state 的狀態(也就是改變七段顯示器要顯示的數據對象)，改變完再進入顯示數據的環節。

再更新完狀態(前後相反)後進開始判斷邀輸出的數據是哪個?，如果是 1 的話(有按下)就顯示超音波測距的內容，先顯示再 serial monitor 上再用包裝的 showDistance() 在七段顯示器上輸出所測量到的值。

相反地，如果是 0 的話(並沒有按下)就顯示溫度感測的內容，採用的方法也是先顯示在 serial monitor 上後再用包裝的 showTemp 在七段顯示器上輸出所測量到的溫度。

在這裡我透過這樣的設計達成以下的效果 ： 當我按下按鈕時會改變型態，不按就不會，配合之前先在 setup 中宣告的 light_state = false，我們可以達到 default 為溫度感測，等到按下按鍵才會是超音波測距的功能。

```
224  float getDistance(int trigPin, int echoPin)
225  {
226      digitalWrite(trigPin, LOW);
227      delayMicroseconds(2);
228      digitalWrite(trigPin, HIGH);
229      delayMicroseconds(20);
230      digitalWrite(trigPin, LOW);
231
232      long duration = pulseIn(echoPin, HIGH, 30000);
233      return duration * 0.034 / 2;
234  }
```

getDistance 顧名思義，就是用來將 HC-SR04P 超音波測距所測到的時間差轉換成距離，首先我們先知道超音波測距測出來的時間是微秒，因此在轉換成公分時要考慮進去這因素的影響:

音速公式 $V = 331 + 0.9 \times T$（攝氏溫度）

令環境溫度為 $15^\circ C \Rightarrow V = 340\ m/s$

$$path\ (cm) = \Delta T \times 340\ (m/s) \times 10^{2}\ (cm) \times 10^{-6}\ (cm/\mu m) \div 2\ (ech.)$$

$$= \Delta T \times 0.034 / 2$$

透過以上的關係式我們能將讀取到的時間差轉成距離(cm)。

```
float getTemperature()
{
    float tmp;
    tmp = analogRead(A0) * (500.0) / 1024;
    return tmp;
}
```

接下來我們來探討 LM35 讀取的電位該如何轉換成攝氏溫度，首先我們先知道 analog 讀到的數值會介在 0~1023 之間，然後讀取到的溫度與電位之間的關係為:

$$V_{amt} = T \times 10\ mV$$

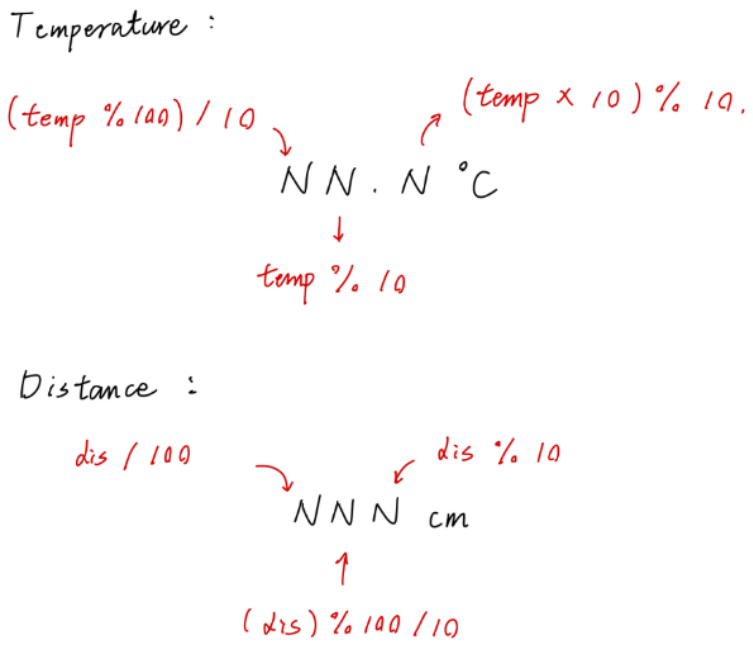
再根據 LM35 板上 Vcc 端接的是 5V，我們可以把電訊號轉成攝氏溫度:

$$T = \underbrace{A_0 / 1024 \times 5}_{\Delta V\ (V)} \times \underbrace{100}_{\frac{1}{10m}}$$

```
73  void showDistance(float distance)
74  {
75      for (int i = 0; i < 50; i++)
76      {
77          digitalWrite(Control_left, HIGH);
78          digitalWrite(Control_right, LOW);
79          showsevenseg(int(distance / 100));
80          delay(10);
81      }
82      for (int i = 0; i < 25; i++)
83      {
84          digitalWrite(Control_left, LOW);
85          digitalWrite(Control_right, HIGH);
86          showsevenseg(int(distance / 100));
87          delay(10);
88          digitalWrite(Control_left, HIGH);
89          digitalWrite(Control_right, LOW);
90          showsevenseg(int(distance) % 100 / 10);
91          delay(10);
92      }
```

```
129 void showTemp(float tmp)
130 {
131     for (int i = 0; i < 50; i++)
132     {
133         digitalWrite(Control_left, HIGH);
134         digitalWrite(Control_right, LOW);
135         showsevenseg(int(tmp / 10) % 10);
136         delay(10);
137     }
138     for (int i = 0; i < 50; i++)
139     {
140         digitalWrite(Control_left, LOW);
141         digitalWrite(Control_right, HIGH);
142         showsevenseg(int(tmp / 10) % 10);
143         delay(5);
144         digitalWrite(Control_left, HIGH);
145         digitalWrite(Control_right, LOW);
146         showsevenseg(int(tmp) % 10);
147         delay(5);
148     }
```

以上兩張圖都是我用來顯示 Distance、Temperature 所包裝好的函數，透過以下的規則將顯示變成跑馬燈的形式輸出:



顯示上我會先開啟左邊的七段顯示器顯示 10ms，再關掉左邊開啟右邊七段顯示器 10ms，根據人類眼睛 fps 大概是 50~60 Hz 之間，我這樣閃爍頻率大概在 100 Hz，可以讓跑馬燈流暢地顯示，並在每次顯示完左右後就會往左推送一格，並遞補心的字元顯示。

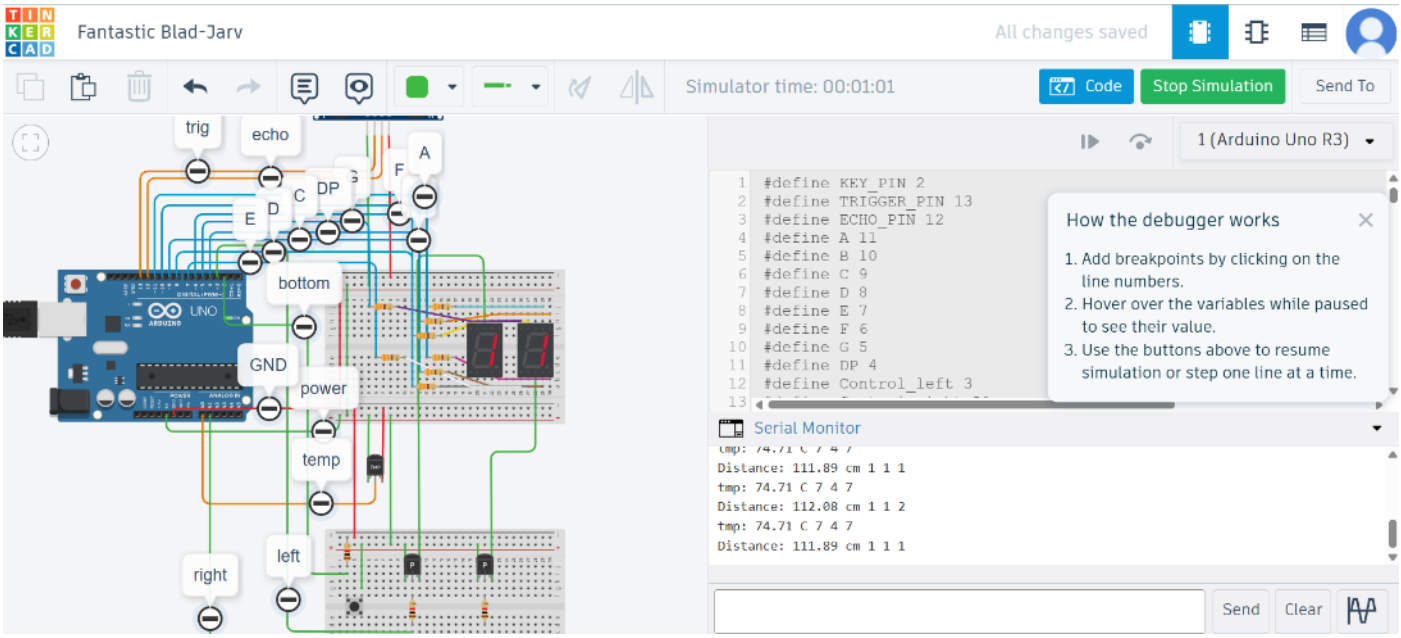
```
void showsevenseg(int num)
{
    // 清除所有段（共陽極：輸出 HIGH 表示關閉）
    for (int i = 4; i < 12; i++)
    {
        digitalWrite(i, HIGH); // 先關閉所有段
    }

    switch (num)
    {
    case 0:
        digitalWrite(A, LOW);
        digitalWrite(B, LOW);
        digitalWrite(C, LOW);
        digitalWrite(D, LOW);
        digitalWrite(E, LOW);
        digitalWrite(F, LOW);
        break;
```

最後我將要顯示數字顯示在七段顯示器上，所以要控制 A~DP 的腳位(digitalWrite)，我將這一整個部分都寫到 showsevenseg 函數中，透過這個函數控制左右的七顯應該要顯示哪些區段。

Your demo video (play the complete melody) link: https://youtube.com/shorts/qy7vO08ah7o?feature=share

Tinkercad 模擬



Tinkercad 連結: https://www.tinkercad.com/things/7RGkkIPdo7d-final-project

結論

透過這次實驗我們將先前的實驗所用過的 Arduino 電子元件整合起來，透過跑馬燈流暢地顯示我們所測量到的數字。

心得

這次的實驗確實貫徹了本學期所學的內容。前半部分的 Arduino 專題延續了實驗二的溫度感測、實驗四的超音波感測，以及實驗五的七段顯示器，我們綜合運用這些元件，設計出一個具有跑馬燈效果的顯示裝置。在程式設計初期，我遇到了不少困難，例如如何實現跑馬燈的動態效果、如何在 delay 延遲期間偵測按鈕的狀態等。然而後來發現，其實這些問題並沒有我想像得那麼複雜，只要搭配 demo 時的手勢控制，再加上設定空白顯示的技巧，就能輕鬆解決。

後半部分則是差動放大器與 AM 調變的應用，對應到 Part 1 的內容。透過這些操作，我們成功濾除噪音並以頻域分析的方式處理心電訊號。不過在實際操作時，我卻在這裡遇到了最大的 bug：當時在示波器上觀察心電訊號時，誤將微弱的心電訊號接到 +15V 的端點，而非簡報中助教指示的接地端。直到重新檢查簡報才發現這個錯誤，為了這個問題我花了將近兩個小時才成功除錯……

引用

- **七段顯示器**
  https://milktea132.blogspot.com/2018/05/8051-4-timer.html
- **心電圖感應器**
  https://kuangtien.yida-design.com.tw/2/file603f14b17f312.pdf?n1=
- **LM35**
  https://www.ti.com/product/LM35?utm_source=google&utm_medium=cpc&utm_campaign=asc-sens-null-44700045336317599_prodfolderdynamic-cpc-pf-google-tw_zhtw_int&utm_content=prodfolddynamic&ds_k=DYNAMIC+SEARCH+ADS&DCM=yes&gad_source=1&gad_campaignid=8991900621&gbraid=0AAAAAC068F0F_ZN4v190FoIQo9rIz9jTm&gclid=Cj0KCQjwucDBBhDxARIsANqFdr0Dck8iwzC9m63iXpm249agMynTVd2mSlrBRAAS3wKKnFE9vKB32lsaArVVEALw_wcB&gclsrc=aw.ds
- **人眼 fps**
  https://as-creative.com.tw/%E6%9C%AA%E4%BE%86%E6%8A%80%E8%A1%93%E5%B1%A4/%E6%88%91%E5%80%91%E7%9A%84%E8%82%89%E7%9C%BC%E6%AF%8F%E7%A7%92%E8%83%BD%E5%A4%A0%E6%8D%95%E6%8D%89%E5%A4%9A%E5%B0%91%E7%95%AB%E9%9D%A2%EF%BC%9F/
- **人體電阻**
  https://lp-cn1-wechat-production.merklechina.com/post/digikey/technical_article/1106/
- **LTspice ECG 訊號**
  https://glenzac.wordpress.com/2018/11/09/ecg-sources-for-pspice-tina-multisim/