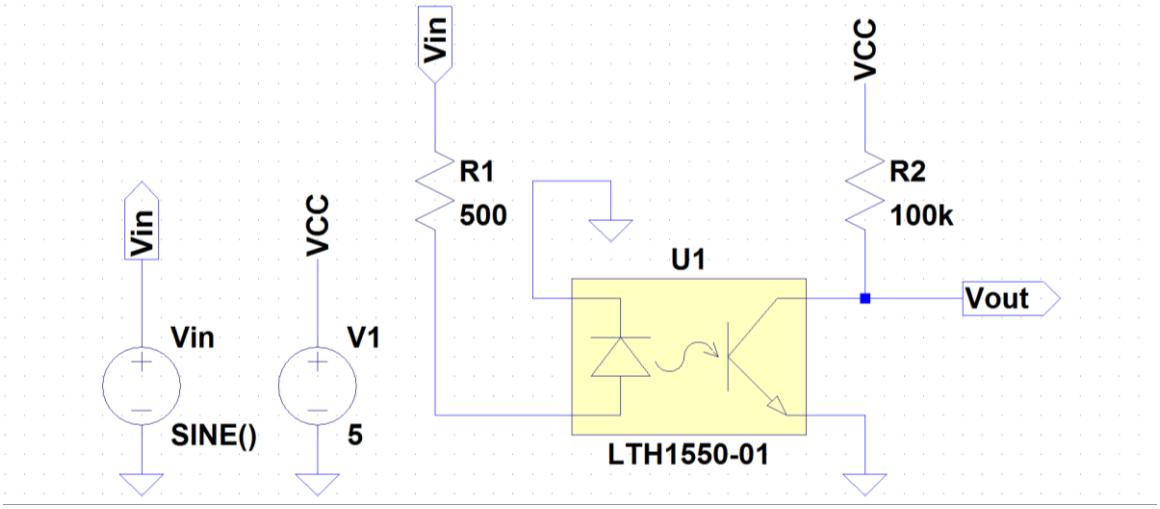
REPORT

Experiment 1: IR Driver and Sensor



電路分析:

介紹 LTH1550-01:

LTH-1550-01 是一款由 Lite-On 生產的光學開關，具有反射式設計和光晶體管輸出，由上面的簡圖可以看出其內部有一個光二極體和一個光接收器，其運作原理基於光二極體和光接收器的互動，當給予光二極體正向電壓時，它會發出光。這個光通常是紅外線（IR）光，而光接收器則用來接收光二極體發出的光信號。當有物體靠近或遮擋光路時(我們的手指)，光接收器就會檢測到光信號的變化，因而產生輸出變化。

ABSOLUTE MAXIMUM RATINGS AT TA=25℃

PARAMETER	SYMBOL	MAXIMUM RATING	UNIT
INPUT DIODE			
Power Dissipation	P_D	90	mW
Peak Forward Current (300 pps , 10 μS pulse)	I_{CP}	1	A
Continuous Forward Current	I_F	60	mA
Reverse Voltage	V_R	5	V
OUTPUT PHOTOTRANSISTOR			
Power Dissipation	P_C	100	mW
Collector-Emitter Voltage	V_{CEO}	30	V
Emitter-Collector Voltage	V_{ECO}	5	V
Collector Current	I_C	20	mA
Operating Temperature Range	T_{opr}	-25℃ to + 85℃	
Storage Temperature Range	T_{stg}	-40℃ to + 100℃	
Lead Soldering Temperature [1.6mm (.063") Form Case]	T_S	260℃ for 5 Seconds	

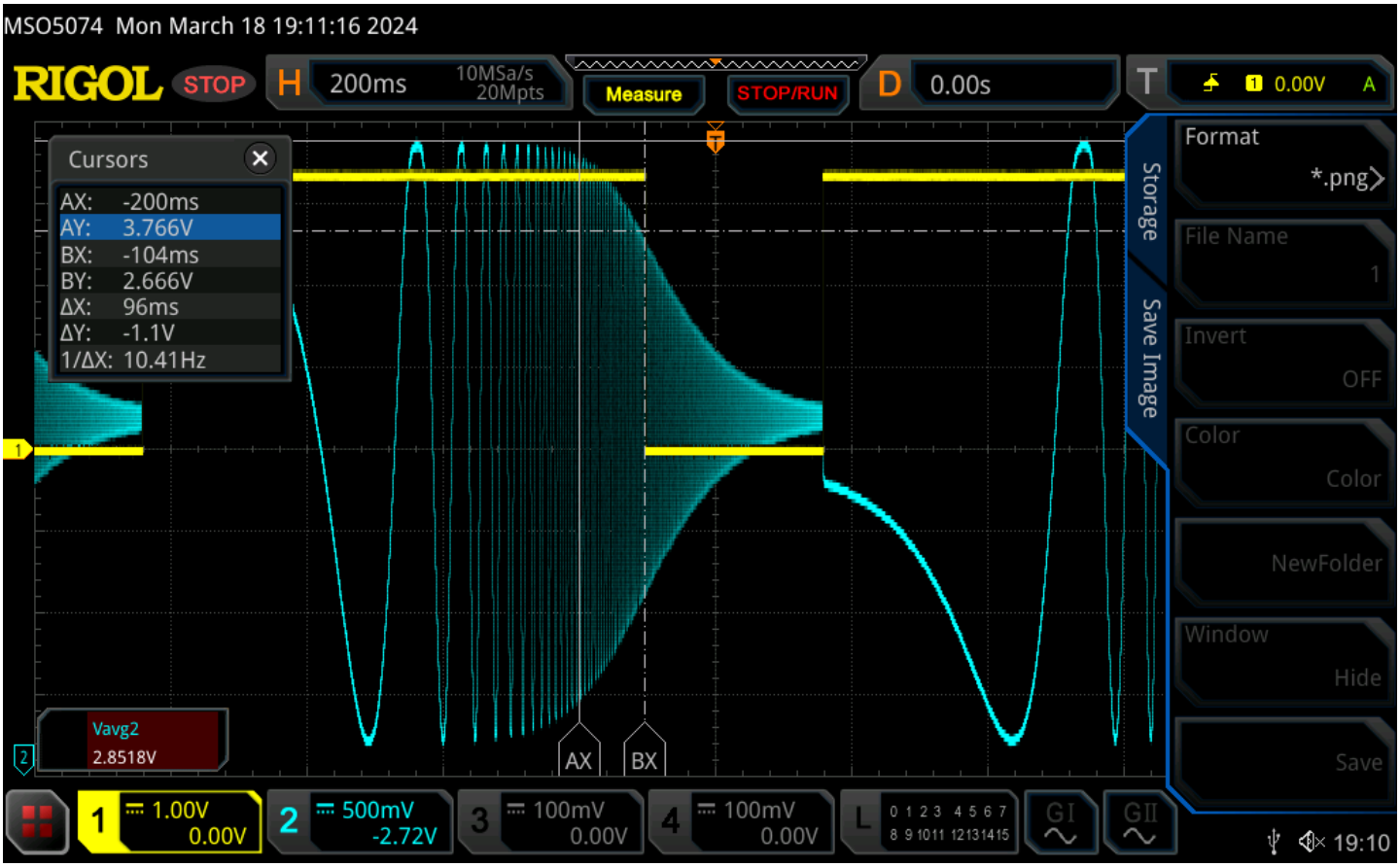
整體電路:

首先，Vin 是一個 Vpp=2V，且有 2V DC offset 的 AC sweep，所以最高電壓會到 3V，而通過限流電阻 R1 後，假設 LTH1550-01 光二極體的跨壓為 0.7V，則可以計算得出最大電流流入 LTH1550-01 光二極體為(3-0.7)V/500ohm=4.6mA，參考 LTH1550-01 的 datasheet(下圖)可以發現此電流量遠小於其輸入順向電流限制 60mA。

電流流經光二極體使其發光，而光接收器則接收訊號，產生對應的電流變化，而這些電流變化會通過 R2 電阻表現到 Vout。通過示波器觀察，可以發現此電路有類似 LPF 的效果，而整體頻率響應的平均 Vpp 為 2.82V 左右，但若忽略被過濾的高頻波段，我當時用 cursor 算出大約 12.5V 的 Vpp，所以此元件其實也有將輸入訊號進行稍微放大的作用。

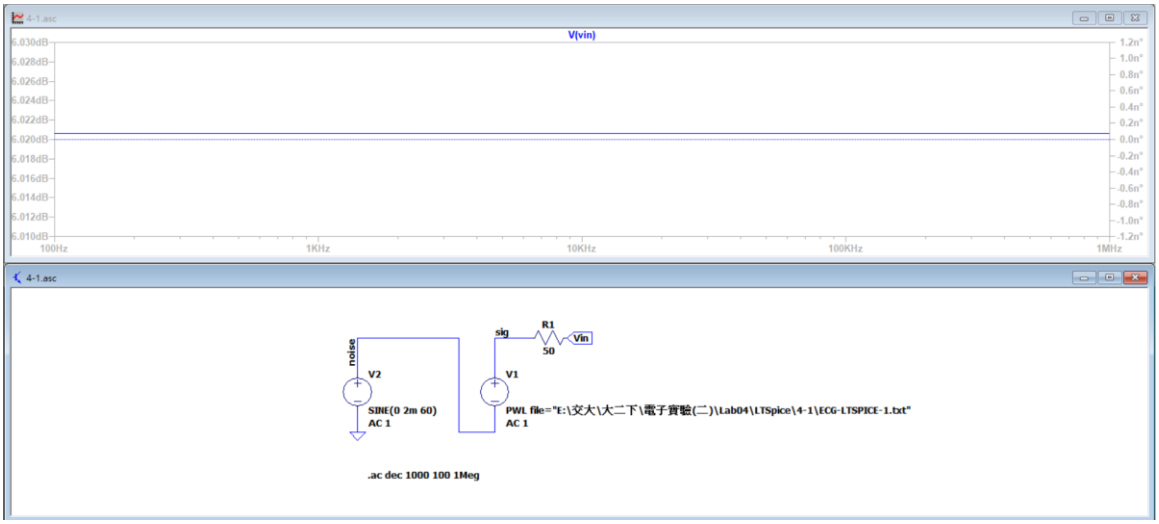
2. AC SWEEP and Bias

$f_{3dB,H}$ (Hz)	Vout average voltage (V)
494	**2.8228**

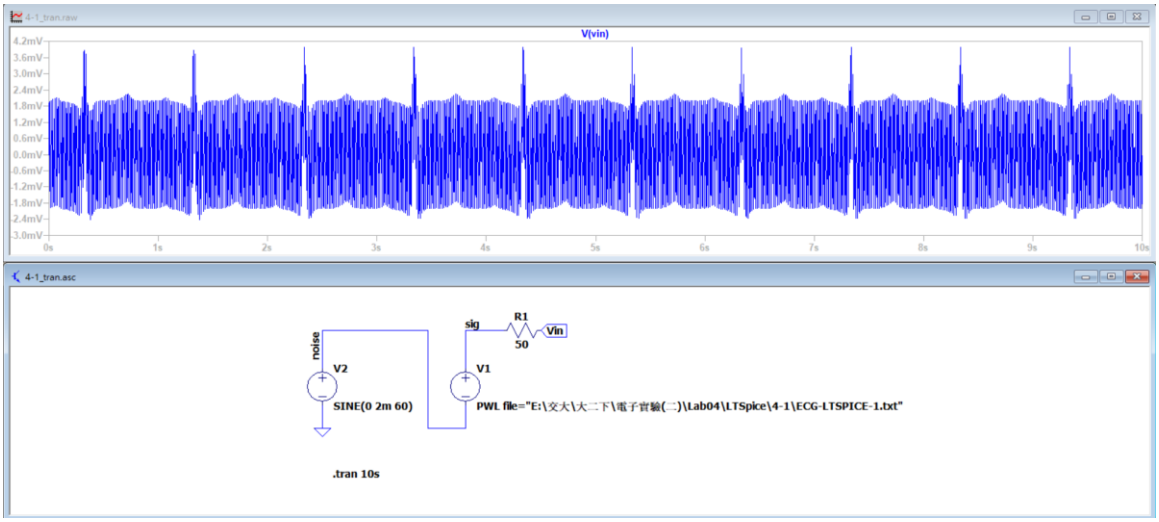


LTSpice Simulation:

心率與雜訊碟加後的輸出波德圖:



心率與雜訊碟加後的波形圖:



Question:

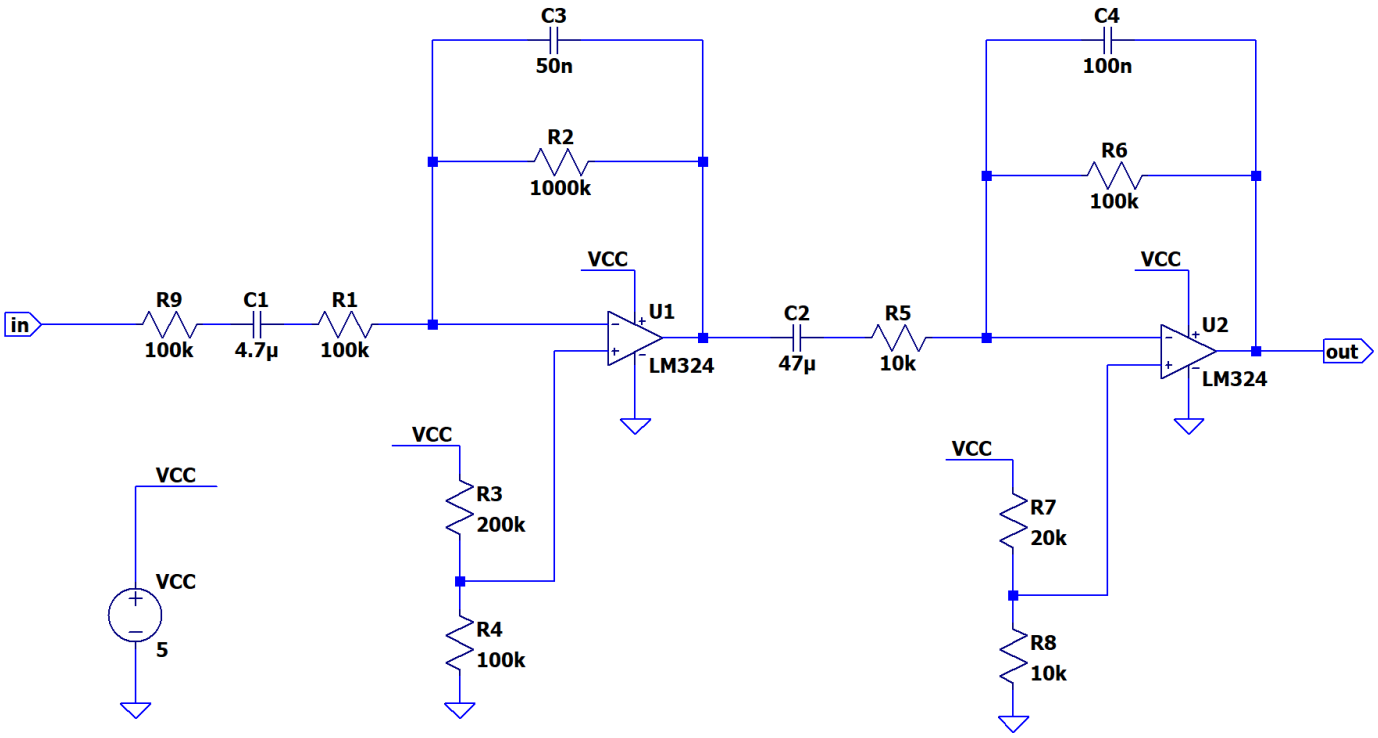
手指頭蓋住 LTH1550-01 的目的是?

由於 LTH1550-01 內的發光二極體在手指沒有蓋住時，射入光電晶體的紅外線很少，所以訊號也很弱，當手指蓋住之後，大量紅外線反射進入光電晶體，因此輸出訊號增加，而反射的紅外光也會因為血壓的高低而產生變化，這也是為什麼可以用來觀測心率的原因。

為什麼需要給函數產生器的 AC Sweep 一個 2V 的 DC offset?

由於光二極體需要順向導通一個 cut-in voltage 才會發光，因此 2V 的 offset 可以確保光二極體始終導通在工作電壓之上，才不會因此產生截波。

Experiment 2: Filter Stage

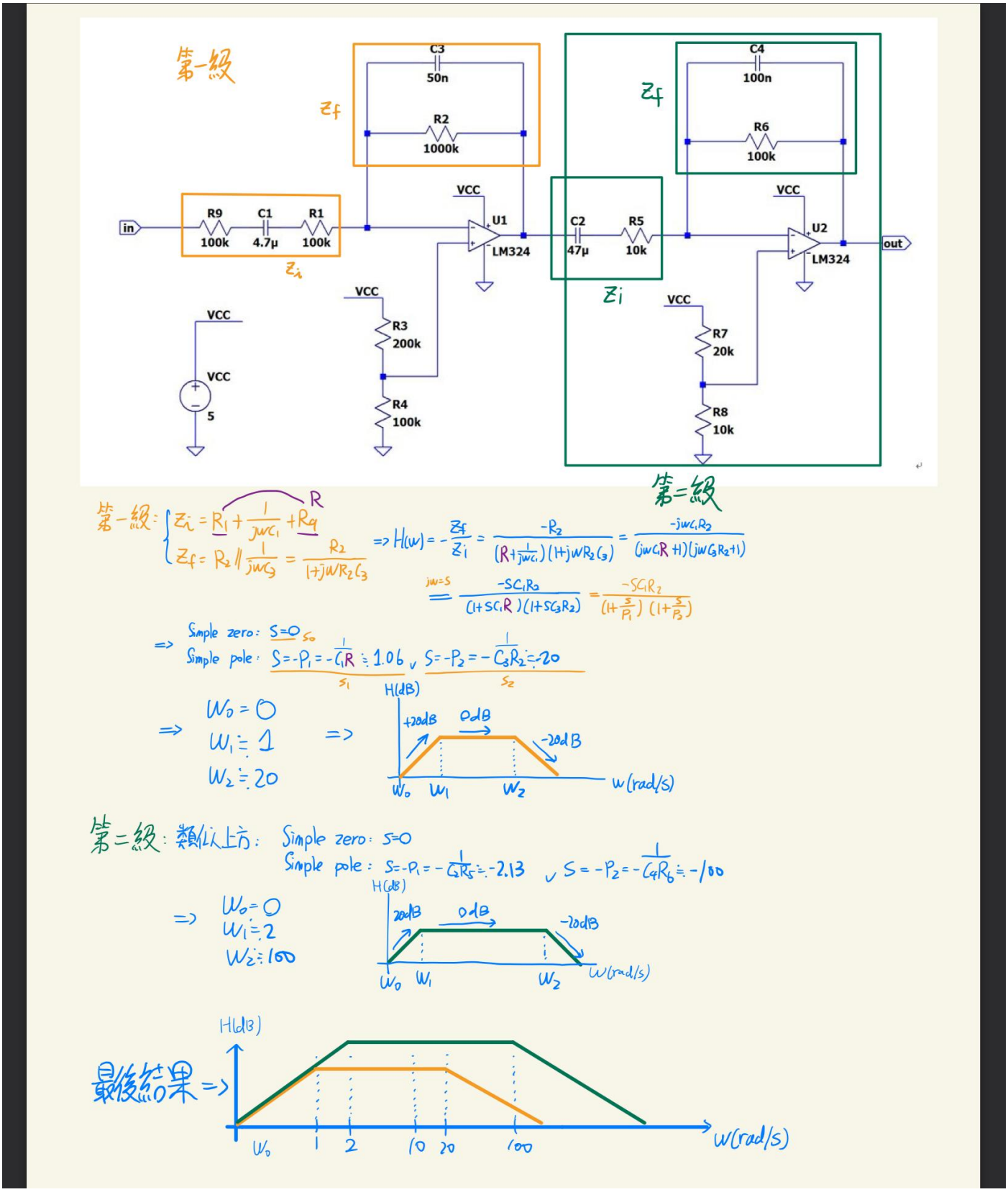


Vin : 50mVpp or appropriate value that Vo is not distorted

OSC : DC coupling

電路數學計算分析:

根據 Lab03 的經驗，可以直覺地看出這個電路左右各有一個一階的 BPF，串接起來組成一個二階的 BPF，詳細計算如與計算得出的大製波德圖如下:



再經過簡單計算就可以獲得如下的左右-3Db 點:

第一級:

$$f_1 = \frac{w_1}{2\pi} \cong 0.169Hz \ ; \ f_2 = \frac{w_2}{2\pi} \cong 3.183Hz$$

第二級:

$$f_1 = \frac{w_1}{2\pi} \cong 0.339Hz \ ; \ f_2 = \frac{w_2}{2\pi} \cong 15.915Hz$$

推導完後可以發現，這樣的濾波裝置，可以把 0.339Hz 以下和 3.183Hz 以上的波濾掉，而這樣的設計是為了符合人體心跳頻率(大約落在 60~120BPS 之間，也就是 1~2Hz 之間)的檢測。

而由於兩極濾波器彼此串聯，所以到 out2 那邊的時候，會變成+40dB/decade 和-40dB/decade(如 LTSpice 模擬的波德圖)

直流分析:

首先，直流分析將 AC 訊移除，並將電容視作斷路。由分壓公式可以得到 U1 V+電壓為:

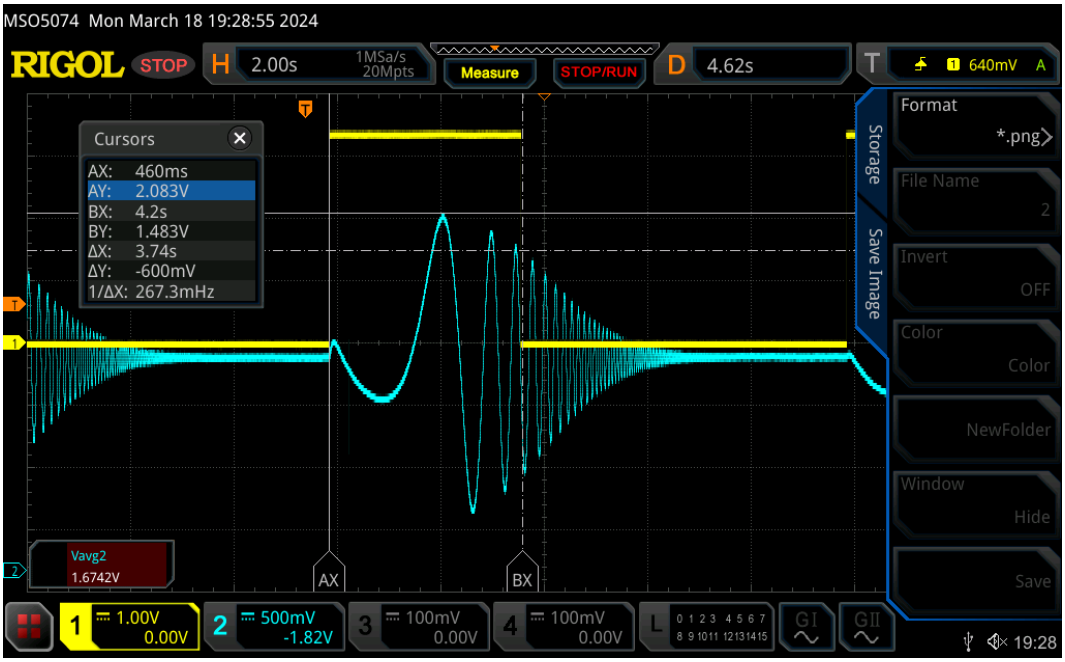
$5 * \frac{100k}{200k+100k} \cong 1.67$，接著由於虛短路，所以 U1 V-也是 1.67V，而由於放大器輸入阻抗很大，所以不會有電流流入，所以 U1 Vout 要與 U1 V-在同一電位，因此推得 U1 Vout 也是 1.67V，以上推導皆適用於 U2 的推導，因為兩者線路模式是一樣的。

2. DC Bias

U1,V+ (V)	U1,Vout (V)	U2,V+ (V)	U2,Vout (V)
1.65	**1.65**	**1.65**	**1.65**

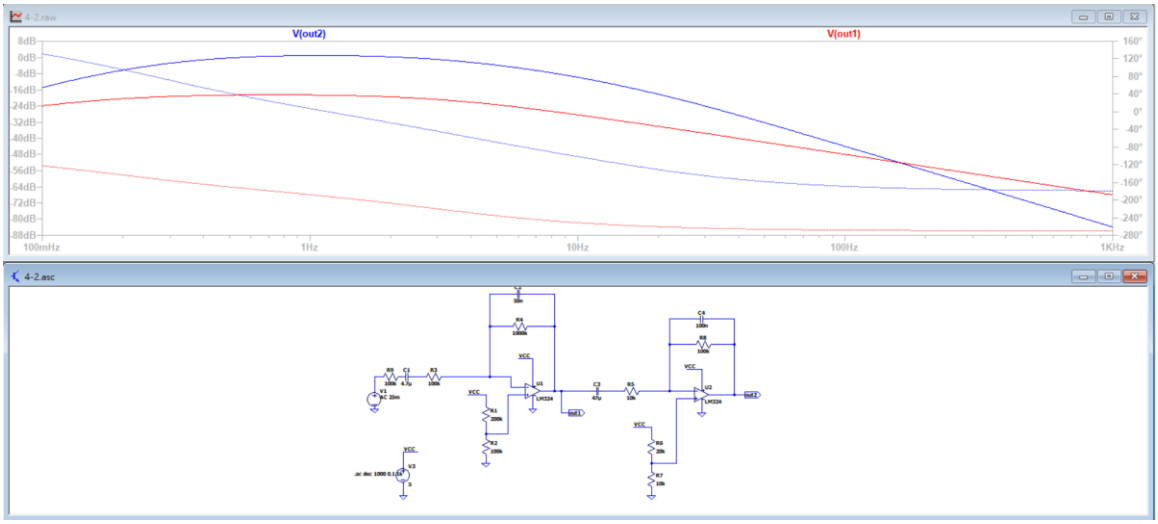
3. AC SWEEP waveform

$f_{3dB,H}$ (Hz)_____2.92_____

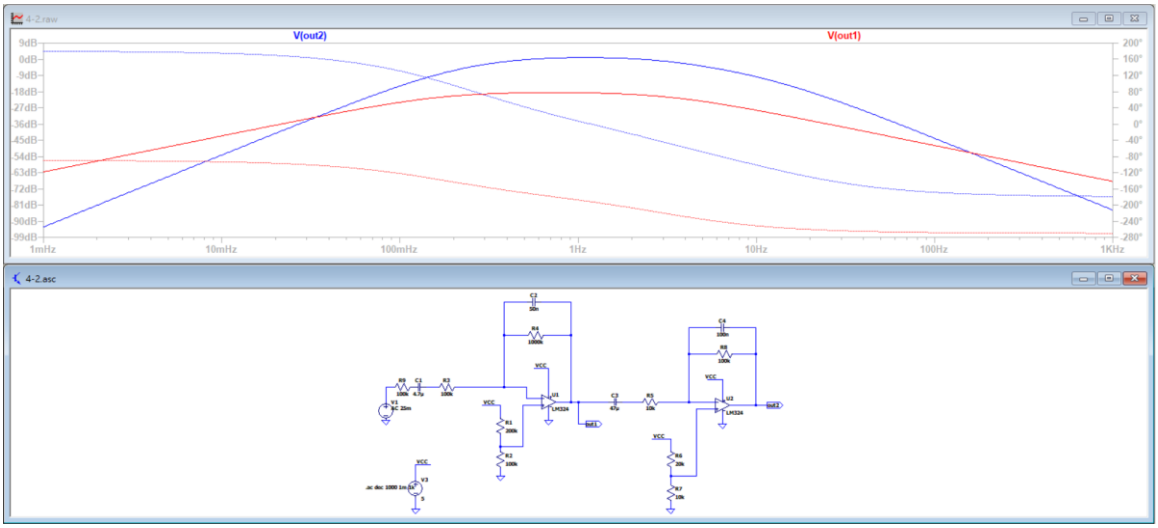


LTSpice Simulation:

模擬 out1 和 out2 的頻率響應情況之波德圖(0.1Hz~1kHz):



模擬 out1 和 out2 的頻率響應情況之波德圖(1mHz~1kHz):

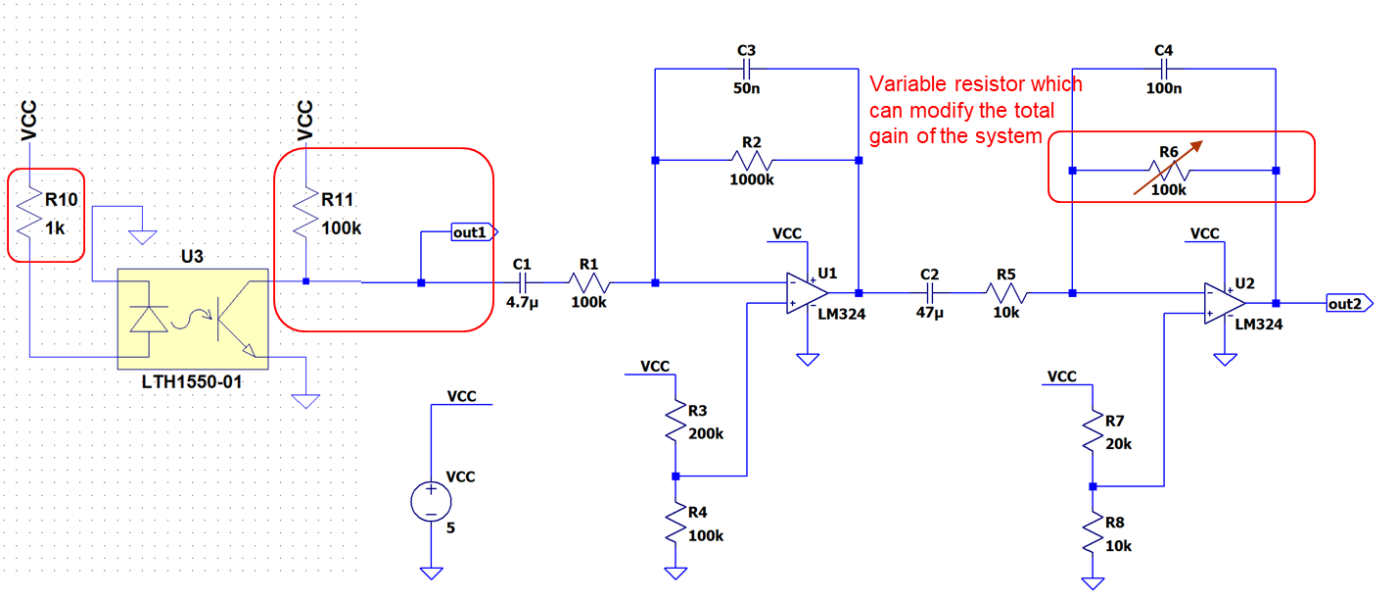


Question:

如何選定放大器正端的輸入電壓?

由於此實驗最後在第三個小實驗要將輸出訊號送給 Arduino 板上,因此必須要讓輸出訊號在 0~5V 的範圍,而正確的選擇放大器正端的電壓才可以讓輸出在所需的範圍內,經由直流分析可以發現,當放大器正端給定一個直流電壓,將會使得在 out 端有一個一樣大小的 offset,而實驗選擇的 1.6V,則可以由測量驗證,確實提供 out 端 1.6V 左右的 offset,這樣一來便可以使得乘載在上面的小訊號可以落在 0~5V 範圍內。

Experiment 3: Heart Rate Monitor

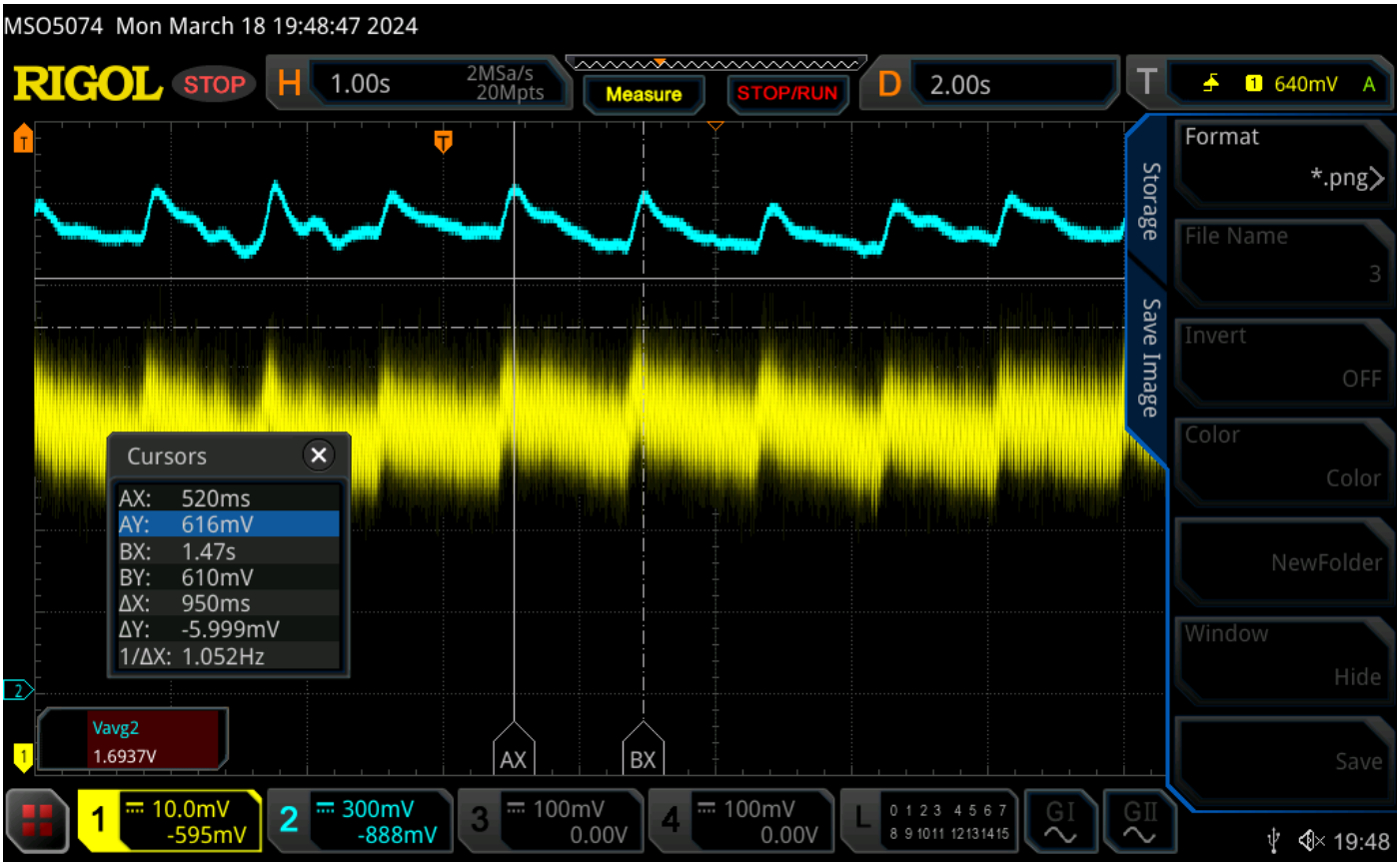


電路分析:

此電路為實驗一和實驗二電路的結合，由於我們心律造成的血管收縮對光接收器接收光的影響，進而導致 out1 點有電流變化造成的電壓變化小訊號，經由後方兩極的濾波器過濾掉雜訊以及放大後，最終在 out2 端獲得我們的心率圖。

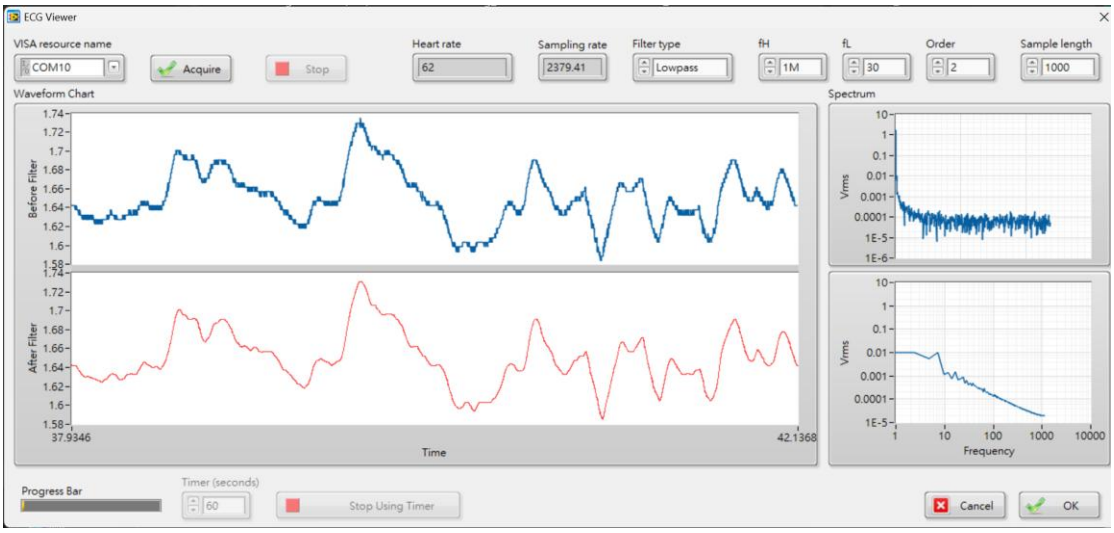
此部分調節 R6 電阻目的在於控制輸出的增益，好讓波形不會產生截波導致失真。

2. Vout1 and Vout2 waveform



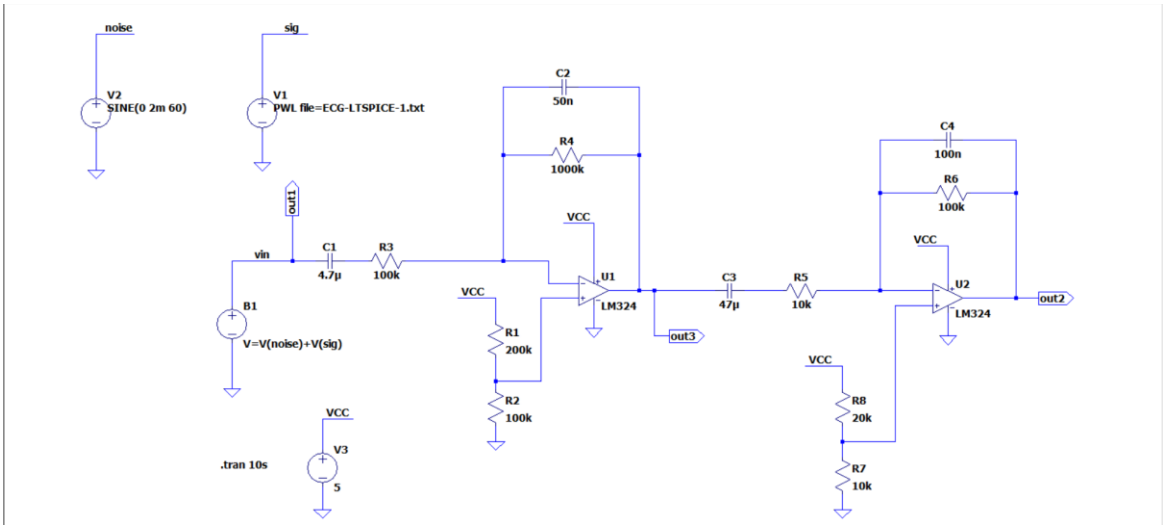
3. Record you real time heart rate = 57 BPM (beat per minute)

Take a screenshot after reading the stable heart rate.

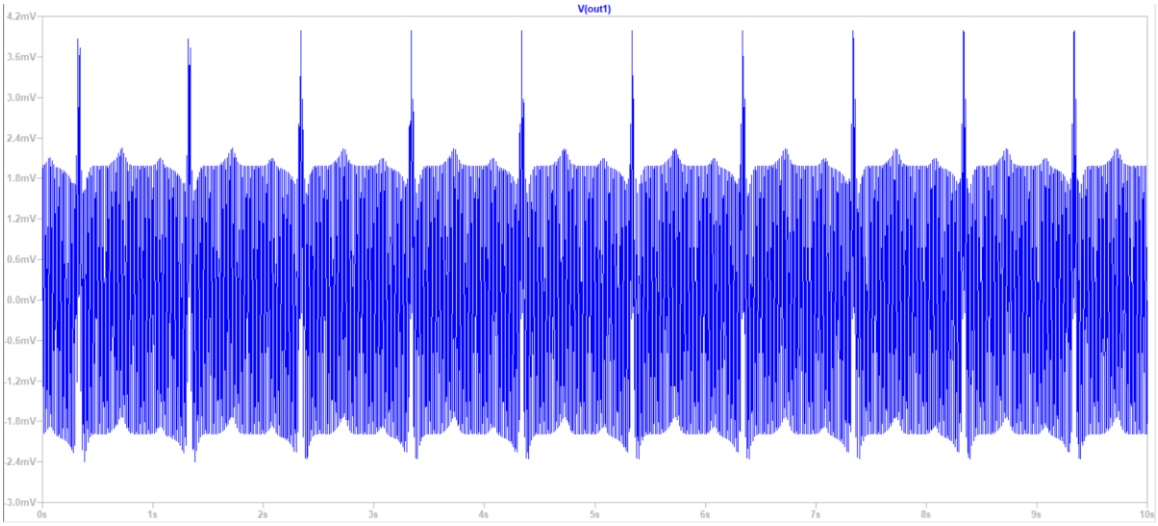


LTSpice Simulation:

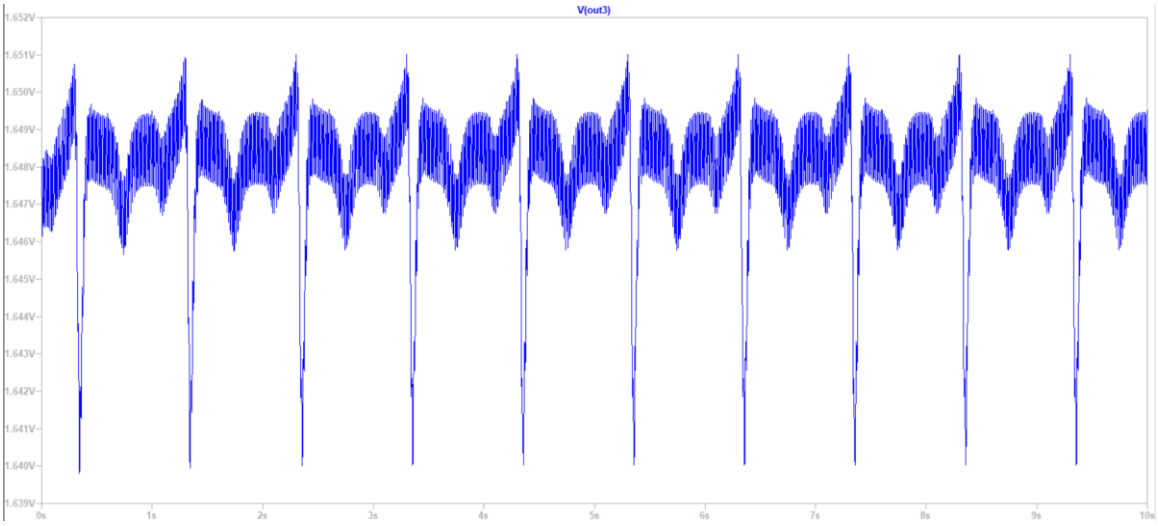
模擬電路圖:



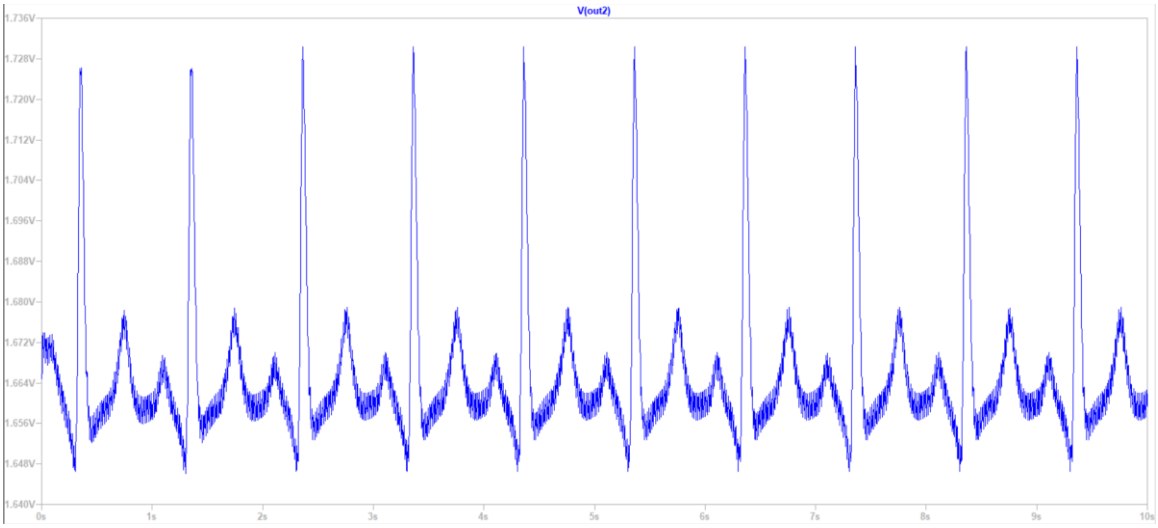
模擬 LTH1550-01 輸出點波形圖(心率與雜訊):



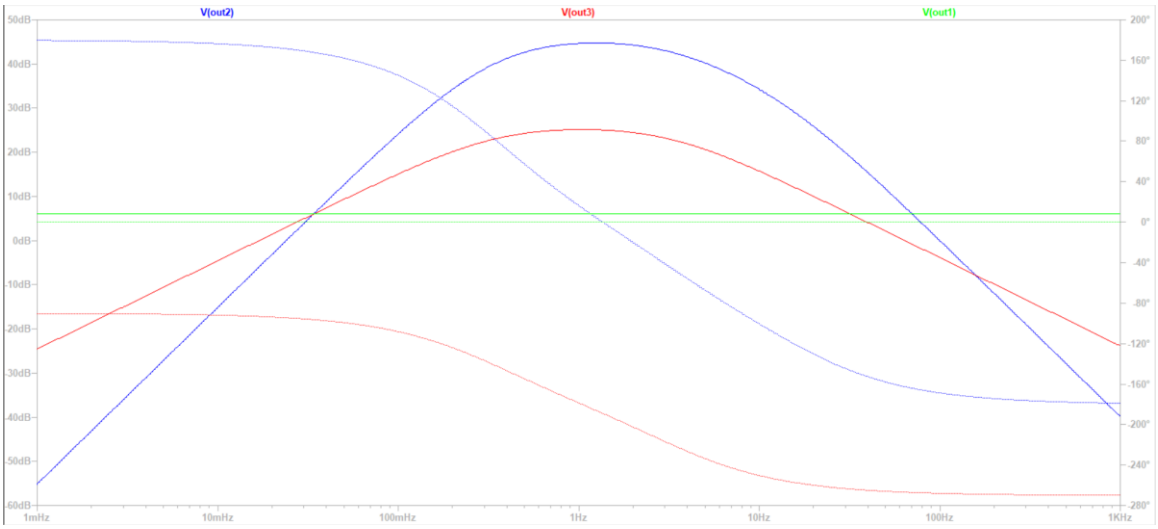
模擬 LTH1550-01 輸出訊號經過第一級率波與增益後的波形圖(圖形顛倒因為放大器增益為負):



模擬 LTH1550-01 輸出訊號經過第一級與第二級濾波與增益後的波形圖(變回正向):



模擬濾波前後頻率響應對應之波德圖(out1、out2、out3 對應上方模擬電路圖):



Question:

R1 電阻從 1000ohm 改成 500ohm 的用意?

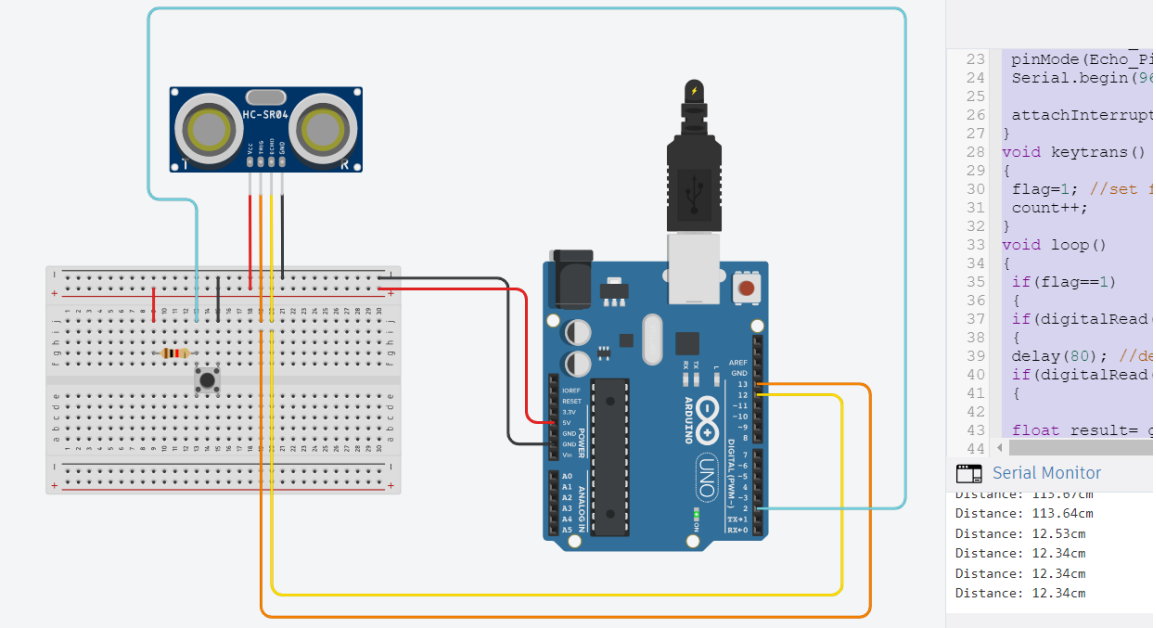
其主要目的是將低 LTH1550-01 內的光二極體亮度，而這是為了讓輸出訊號的震幅降低，以防止放大後傳到 Arduino(0~5V 限制)造成失真。

R6 電阻改成可變電阻原因?

電路分析部分有提到，R6 電阻和 R5 電阻的比例會影響 U2 放大器的輸出增益，因此改成可變電阻可以方便我們調整增益的值。

Experiment 4: Ultrasonic Sensor

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

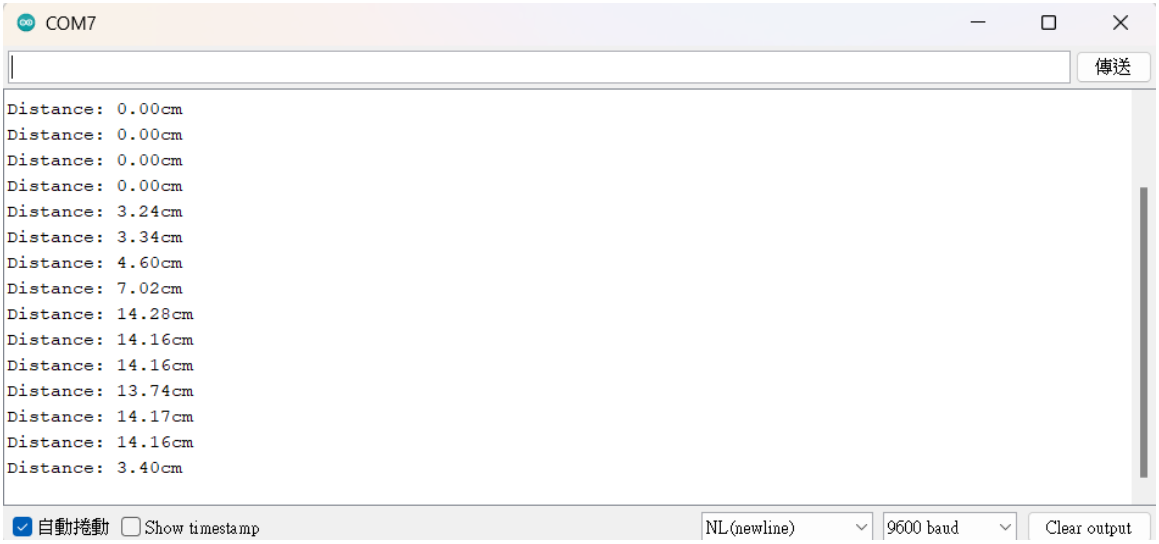


The sketch of your design: (copy from the Arduino IDE window and paste here)

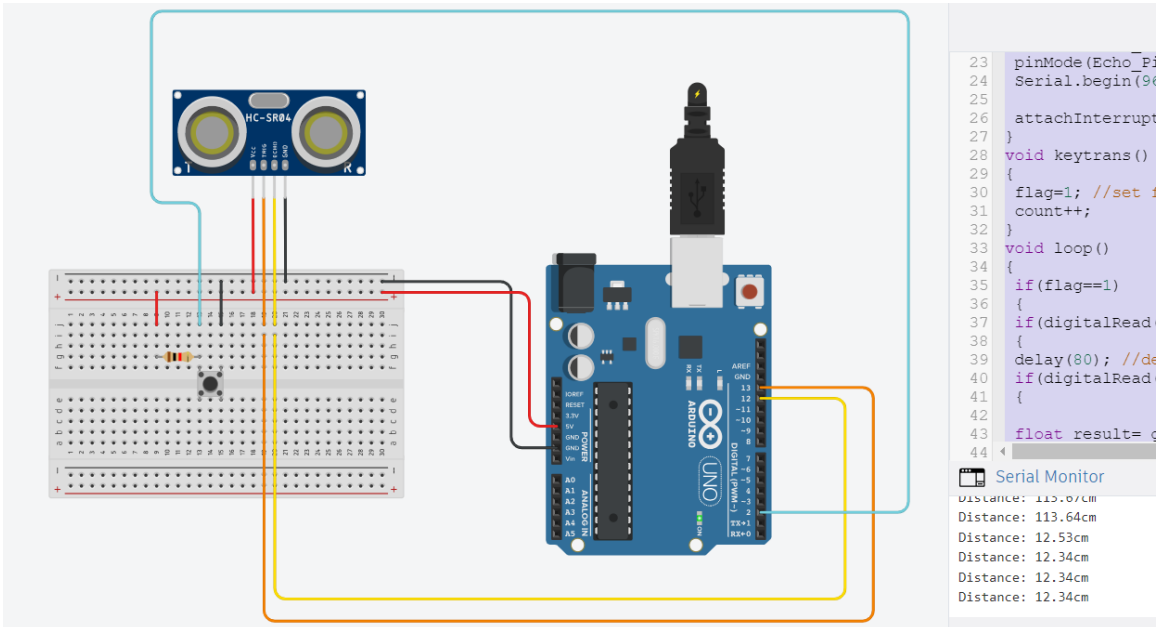
```
#define keyin 2
// defines variables

const int Trig_Pins = 13;//trig  腳位
const int Echo_Pins = 12;// echo  腳位

bool flag=0;
static long count=0;

float get_distance(int trig, int echo){    //計算距離
   float duration;
   digitalWrite(trig, HIGH);
   delayMicroseconds(10);                    // 10us TTL pulse,修復聲波發射延遲
   digitalWrite(trig, LOW);
   duration = pulseIn(echo, HIGH, 3000000);//紀錄 echo 電位從 high 到 low 的時間(超音波來回時間)，若 3 秒內沒收到超音波則回傳 0
   return duration / 29 / 2;                 //  聲速 340m/s ，換算後約每 29 微秒走一公分，超音波來回所以再除 2
}
```

```
void setup()
{
  pinMode(keyin, INPUT);
  pinMode(Trig_Pins, OUTPUT);
  pinMode(Echo_Pins, INPUT);
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed

  attachInterrupt(digitalPinToInterrupt(keyin), keytrans, FALLING); //mode= falling edge status
}
void keytrans()
{
  flag=1; //set flag=1
  count++;
}
void loop()
{
  if(flag==1)
  {
  if(digitalRead(keyin) == LOW) //check pin2 status
  {
  delay(80); //delay 80ms  解決 de-bouncing
  if(digitalRead(keyin) == LOW) //check pin2 again
  {

  float result= get_distance( Trig_Pins, Echo_Pins);
       Serial.print("Distance: ");
       Serial.print(result);
       Serial.println("cm");
       delay(1000);
  flag=0; //clear flag
  }
  }
  }
}
```

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



Tinkercad 模擬:



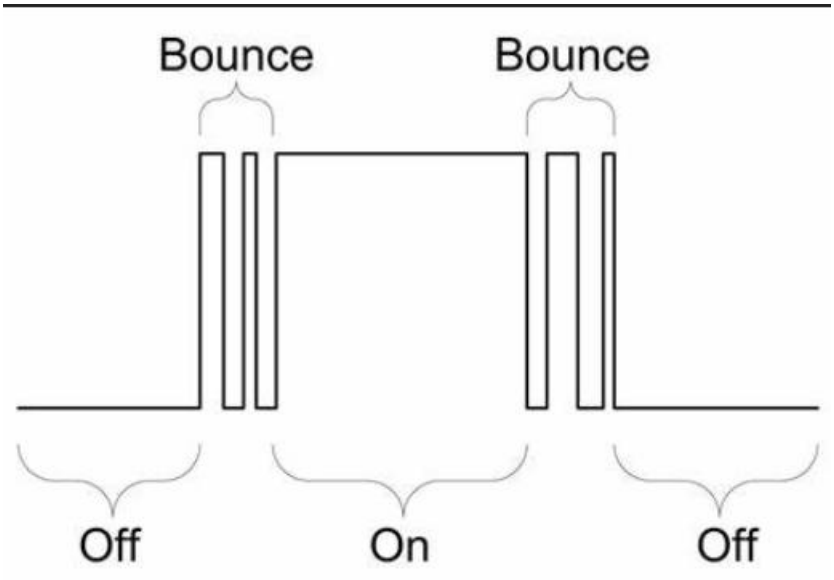
電路分析:

當按鈕為按下時，Arduino 的 D2 接腳被上拉電阻拉到高電位，而當按鈕按下後，D2 接腳就會因為接地變成低電位，而當 Arduino sketch 偵測到 D2 變成 LOW 時，就會傳送一個 pulse 訊號到 HC-SR04 的 Trig 腳位去啟動它發射超音波，而當 HC-SR04 接收到反彈的超音波後，就會從 echo 腳位回傳訊號，這中間的時間差就可以透過數學計算轉換為前方物體的距離。

Question:

為甚麼在 check pin2 是否為 LOW 時，中間要 delay 80 毫秒之後再次確認?

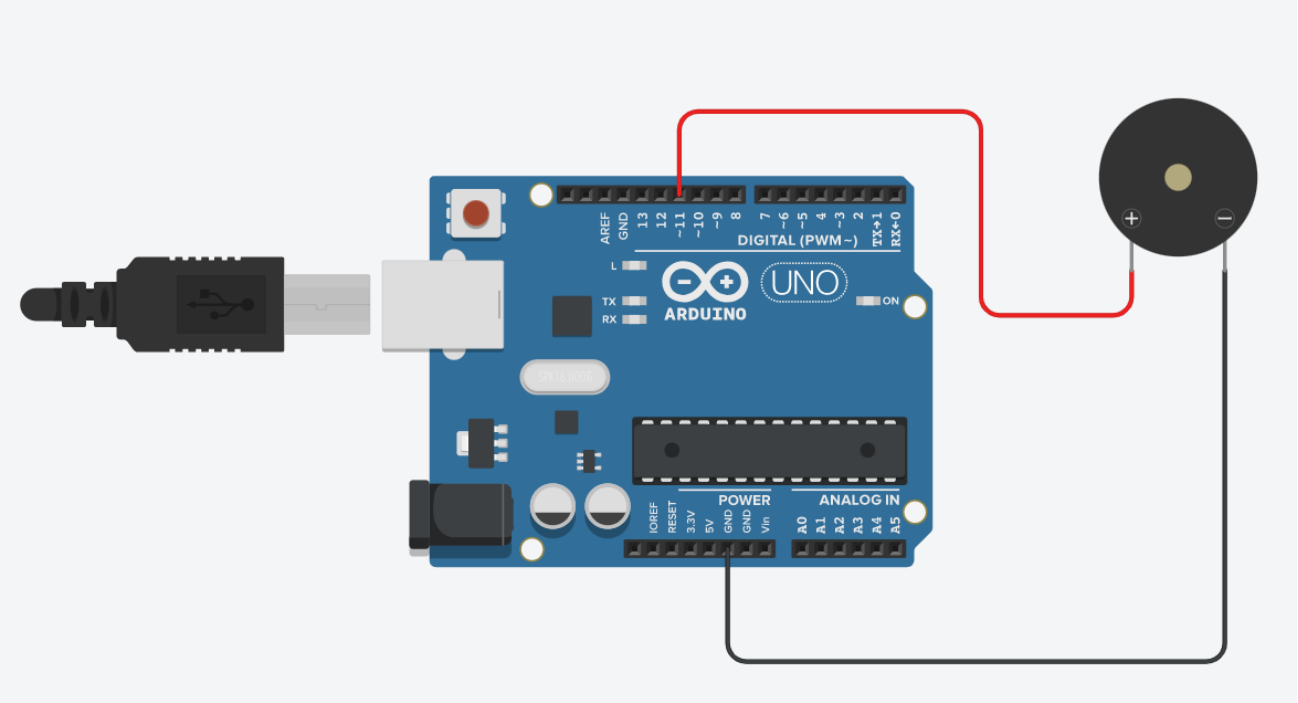
這是因為按鈕在按下時通常會因機械和物理問題而產生意外的開關過渡，而這些過渡可能在非常短的時間內被讀取為多次按下，因此會產生如下圖的訊號狀態:



而解決辦法就是先判斷一次是否進入開關按下的狀態，如果是，再等 80 毫秒後再次檢測，如果還是按下狀態，那就代表是確實按下(因為 de-bounce 是瞬間的事情，80 毫秒是大於這個狀態的)。

Experiment 5: Melody Generator

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



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

```
/*
    Hedwig's theme - Harry Potter
    Connect a piezo buzzer or speaker to pin 11 or select a new pin.
    More songs available at https://github.com/robsoncouto/arduino-songs

                                              Robson Couto, 2019
*/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
```

```
#define NOTE_D2    73
#define NOTE_DS2 78
#define NOTE_E2    82
#define NOTE_F2    87
#define NOTE_FS2 93
#define NOTE_G2    98
#define NOTE_GS2 104
#define NOTE_A2    110
#define NOTE_AS2 117
#define NOTE_B2    123
#define NOTE_C3    131
#define NOTE_CS3 139
#define NOTE_D3    147
#define NOTE_DS3 156
#define NOTE_E3    165
#define NOTE_F3    175
#define NOTE_FS3 185
#define NOTE_G3    196
#define NOTE_GS3 208
#define NOTE_A3    220
#define NOTE_AS3 233
#define NOTE_B3    247
#define NOTE_C4    262
#define NOTE_CS4 277
#define NOTE_D4    294
#define NOTE_DS4 311
#define NOTE_E4    330
#define NOTE_F4    349
#define NOTE_FS4 370
#define NOTE_G4    392
#define NOTE_GS4 415
#define NOTE_A4    440
#define NOTE_AS4 466
#define NOTE_B4    494
#define NOTE_C5    523
#define NOTE_CS5 554
#define NOTE_D5    587
#define NOTE_DS5 622
#define NOTE_E5    659
#define NOTE_F5    698
#define NOTE_FS5 740
```

```
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
#define REST 0

// change this to make the song slower or faster
int tempo = 144;

// change this to whichever pin you want to use
int buzzer = 11;
```

```
// notes of the moledy followed by the duration.
// a 4 means a quarter note, 8 an eighteenth , 16 sixteenth, so on
// !!negative numbers are used to represent dotted notes,
// so -4 means a dotted quarter note, that is, a quarter plus an eighteenth!!
int melody[] = {


  // Hedwig's theme fromn the Harry Potter Movies
  // Socre from https://musescore.com/user/3811306/scores/4906610

  REST, 2, NOTE_D4, 4,
  NOTE_G4, -4, NOTE_AS4, 8, NOTE_A4, 4,
  NOTE_G4, 2, NOTE_D5, 4,
  NOTE_C5, -2,
  NOTE_A4, -2,
  NOTE_G4, -4, NOTE_AS4, 8, NOTE_A4, 4,
  NOTE_F4, 2, NOTE_GS4, 4,
  NOTE_D4, -1,
  NOTE_D4, 4,

  NOTE_G4, -4, NOTE_AS4, 8, NOTE_A4, 4, //10
  NOTE_G4, 2, NOTE_D5, 4,
  NOTE_F5, 2, NOTE_E5, 4,
  NOTE_DS5, 2, NOTE_B4, 4,
  NOTE_DS5, -4, NOTE_D5, 8, NOTE_CS5, 4,
  NOTE_CS4, 2, NOTE_AS4, 4,
  NOTE_G4, -1,
  NOTE_AS4, 4,

  NOTE_D5, 2, NOTE_AS4, 4,//18
  NOTE_D5, 2, NOTE_AS4, 4,
  NOTE_DS5, 2, NOTE_D5, 4,
  NOTE_CS5, 2, NOTE_A4, 4,
  NOTE_AS4, -4, NOTE_D5, 8, NOTE_CS5, 4,
  NOTE_CS4, 2, NOTE_D4, 4,
  NOTE_D5, -1,
  REST,8, NOTE_AS4,4,

  NOTE_D5, 2, NOTE_AS4, 4,//26
  NOTE_D5, 2, NOTE_AS4, 4,
  NOTE_F5, 2, NOTE_E5, 4,
```

```
  NOTE_DS5, 2, NOTE_B4, 4,
  NOTE_DS5, -4, NOTE_D5, 8, NOTE_CS5, 4,
  NOTE_CS4, 2, NOTE_AS4, 4,
  NOTE_G4, -1,

};

// sizeof gives the number of bytes, each int value is composed of two bytes (16 bits)
// there are two values per note (pitch and duration), so for each note there are four bytes
int notes = sizeof(melody) / sizeof(melody[0]) / 2;

// this calculates the duration of a whole note in ms (60s/tempo)*4 beats
int wholenote = (60000 * 4) / tempo;

int divider = 0, noteDuration = 0;

void setup() {
  // iterate over the notes of the melody.
  // Remember, the array is twice the number of notes (notes + durations)
  for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {

    // calculates the duration of each note
    divider = melody[thisNote + 1];
    if (divider > 0) {
      // regular note, just proceed
      noteDuration = (wholenote) / divider;
    } else if (divider < 0) {
      // dotted notes are represented with negative durations!!
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5; // increases the duration in half for dotted notes
    }

    // we only play the note for 90% of the duration, leaving 10% as a pause
    tone(buzzer, melody[thisNote], noteDuration*0.9);

    // Wait for the specief duration before playing the next note.
    delay(noteDuration);

    // stop the waveform generation before the next note.
    noTone(buzzer);
  }
```

```
}

void loop() {
  // no need to repeat the melody.
}
```

Your demo video (play the complete melody) link: demo 過了~

實驗心得:

這次實驗的前三個小實驗有承接上一次實驗的感覺，因此做起來也比較得心應手知道自己在幹嘛，而 BPF 的 w 計算和波德圖繪畫也讓我複習了暑修的電路學知識，DC 直流分析，則讓我複習了數修的電子學(一)內容，讓我對訊號分析與電位分析更進步。

而第四個小實驗的測距裝置我之前在 Arduino 的障礙車上有用過，這次在實驗又看到也讓我想起之前在做避障車的回憶。

第五個實驗是利用程式碼來讓 Arduino 產生不同音頻的輸出訊號給喇叭，來播放音樂，相當有趣。

實驗結論:

實驗一讓我們先認識 LTH1550-01 這個元件對訊號的處理過程與結果，重點是要用電阻控制好光二極體的亮度，來讓輸出訊號可以更好的顯現出來(因此才會有從實驗一 1000ohm 換成實驗三 500ohm 的過程)。

實驗二的重點在看兩個 BPF 串接起來對訊號的處理，以及放大器增益的效果，可以透過計算得知，經過兩個 BPF 的濾波情況疊合，最終輸出端會在低頻有一個 40dB/decade 的增加，在高頻則會有一個-40dB/decade 的衰減，並且值得注意的是 R6 電阻對輸出增益的影響是否會導致截波產生，因此訊號失真。

實驗三結合了實驗一和實驗二的電路裝置，組成一個測量心率的裝置，由實驗中的分析可以清初看到，當心濾被 LTH1550-01 轉成訊號輸出時，是擁有相當高的雜訊且電壓變化非常小，透過兩次濾波與增益還有 DC offset 的調整，最終的輸出訊號被放大到 0~5V 的範圍且 0.339Hz 以下和 3.183Hz 以上的波都會被濾掉，因此可以呈現出一個清晰的心率圖。

實驗四結合了實驗一上拉或下拉電阻的開關概念，去控制測距元件的啟動，並探討和解決按鈕產生的 de-bounce 問題。

實驗五將不同頻率的訊號透過 Arduino 輸入給喇叭，呈現不同音調聲音的輸出。

Reference:

Charles Alexander. & Matthew Sadiku. (2016). Fundamentals of Electric Circuits, 6th Edition. McGraw-Hill Education

Adel S. Sedra , Kenneth C. (KC) Smith , Tony Chan Carusone , and Vincent Gaudet (2020). Microelectronic Circuits, 8th edition. Oxford University

Arduino. (2024) Debounce on a Pushbutton. Retrieved from: https://docs.arduino.cc/built-in-examples/digital/Debounce/ (2024/03/23)

LTH1550-01 Datasheet. Retrieved from: https://optoelectronics.liteon.com/upload/download/DS-55-94-0001/H1550-01new.pdf (2024/03/23)

Arduino. (2024) Play a Melody using the tone() function. Retrieved from:

https://docs.arduino.cc/built-in-examples/digital/toneMelody/ (2024/03/23)

GitHub. (2014) ArduinoCore-avr/cores/arduino/Tone.cpp. Retrieved from: https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/Tone.cpp (2024/03/23)