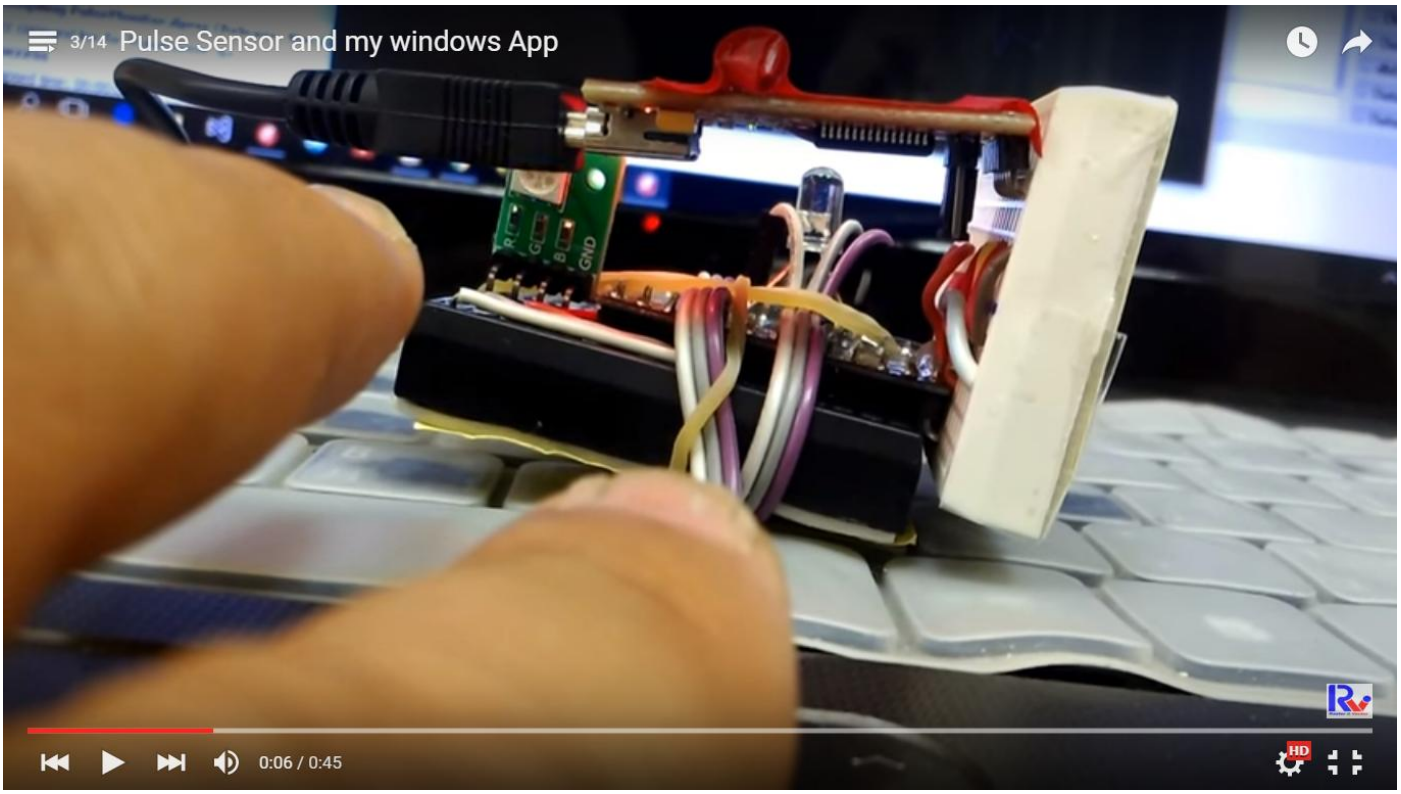
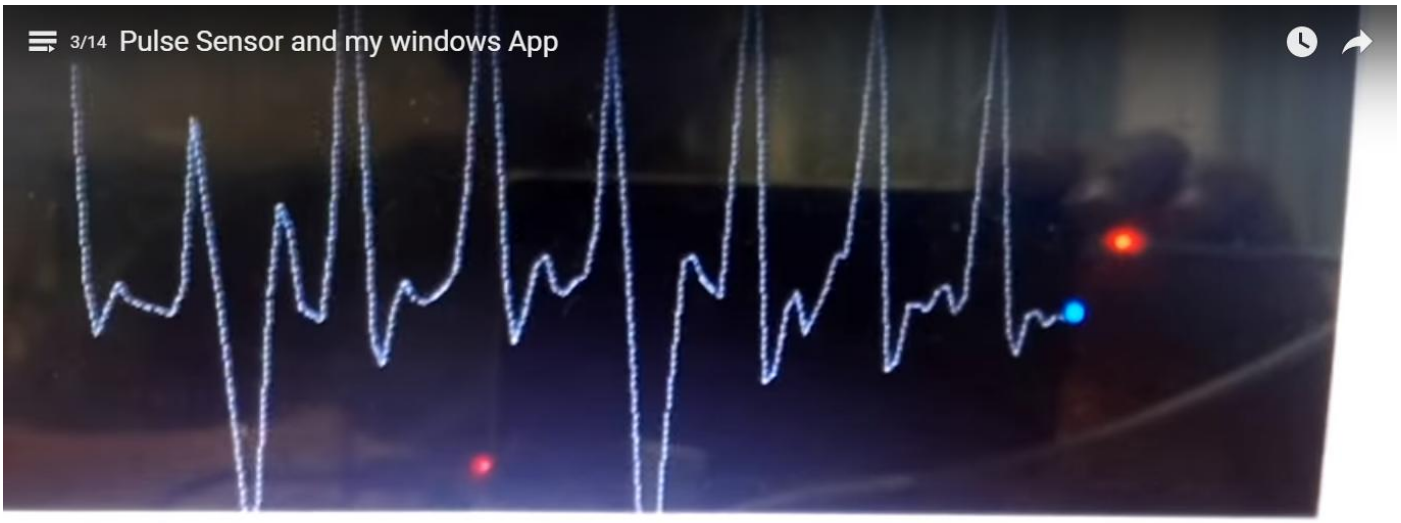
Arduino Pulse Sensor

呂芳元 2015 /12/ 2
http://www.rasvector.url.tw/

說明 ：

使用 Arduino + Pulse Sensor 擷取心脈波訊號，以 RGB LED 顯示心跳快慢程度，並將心脈波、心跳數據傳送到 PC 端顯示波形。





影片:

https://www.youtube.com/watch?v=0CKQ7as65yc&index=5&list=PLZG_AEGYW1gIMxUBlrYXnEpwFF9LIxn3T

https://www.youtube.com/watch?v=p0KdO7rtU70&list=PLZG_AEGYW1gIMxUBlrYXnEpwFF9LIxn3T

Arduino 程式碼、線路圖、PC 端程式、說明 下載:

https://www.dropbox.com/sh/iy5nacruhfk7c28/AADUfQWsaexbhZdodKG-M77Pa?dl=0

材料:

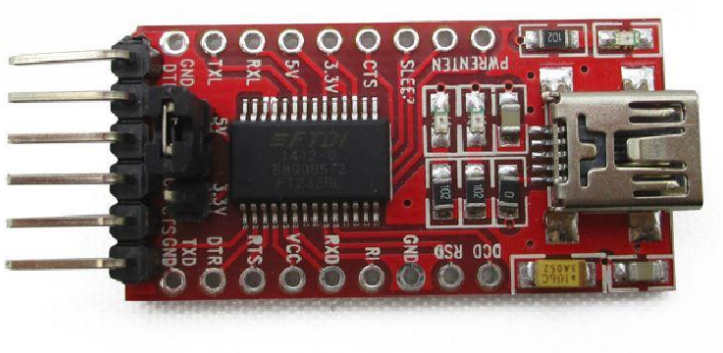
USB_To_TTL	x1	用來將 Arduino 程式碼燒錄到 Arduino，和將心跳數據傳回 PC。
Arduino ProMini	x1	用來控制 LED 閃爍和接收 Pulse Sensor 數據。
Pulse Sensor	x1	用來接收人體上的心跳、心脈波數據。
共陰極 RGB LED	x1	以 紅色(心跳快) -> 綠色 -> 藍色(心跳慢) 用來顯示心跳快慢程度。
單色 LED	x1	用來做心跳指示燈。

。

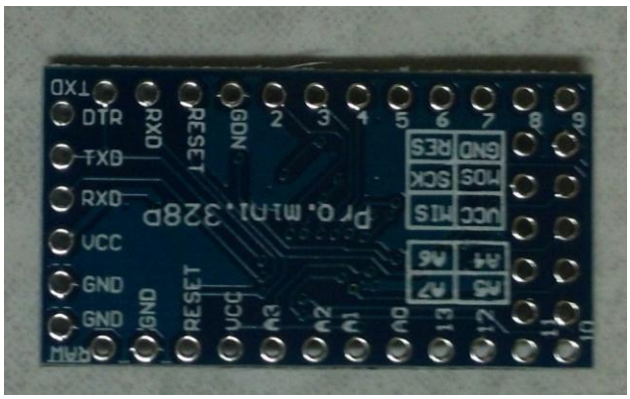
線材:

杜邦線 (公/公、公/母、母/母)各 10 條以上、麵包版。

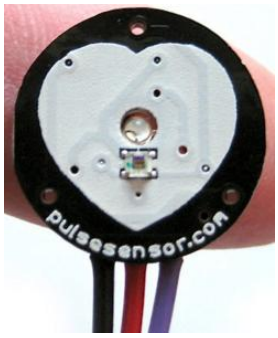
USB to TTL 模組



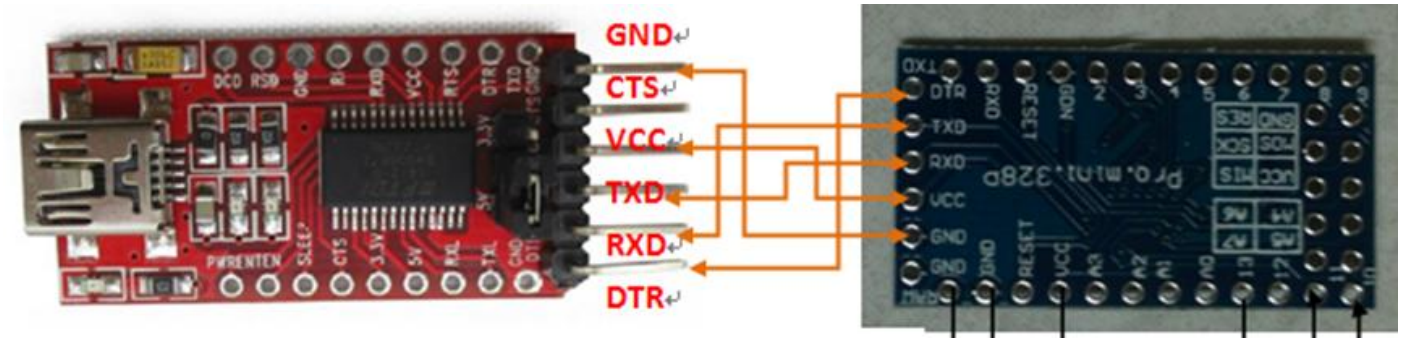
Arduino ProMini

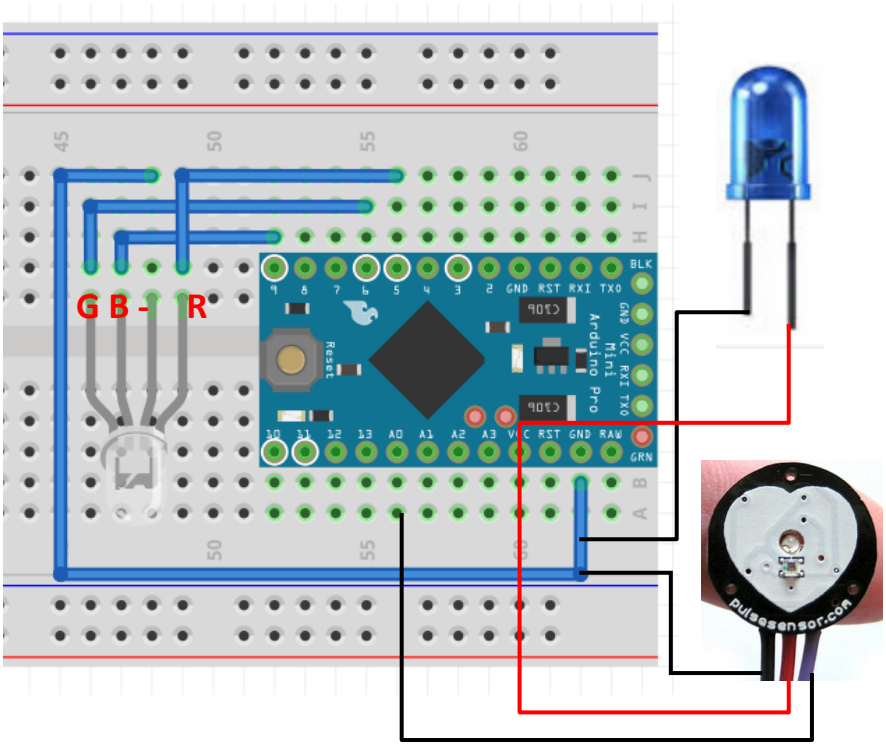


Pulse Sensor



線路:





Arduino 程式碼:

主程式 PulseSensor_BlinkLed.ino

```
/*
      使用   Timer2  中斷服務，擷取  PulseSensor  的數值

  Daniel Lu : dan59314@gmail.com
  http://www.rasvector.url.tw/
  http://www.youtube.com/dan59314/playlist

      blinkPin :  當心跳訊號出現時，用來快閃
      fadPin       :  心跳訊號出現後，用來漸暗  LED

      指令 :
Serial.print(78, BIN) gives "1001110"
Serial.print(78, OCT) gives "116"
Serial.print(78, DEC) gives "78"
Serial.print(78, HEX) gives "4E"
Serial.println(1.23456, 0) gives "1"
Serial.println(1.23456, 2) gives "1.23"
Serial.println(1.23456, 4) gives "1.2346"
Serial.println("S," + String(pulseVal) + "," + String(gTotalTime, HEX));

*/

#include <SoftwareSerial.h>

#define EnableReceiveCommand
#define AverageHeartBeat;
#define RGBColorFadding
#define UseTimer2IRS
#define CommonAnode

#define debug


typedef struct TRGB {
      byte R, G, B;
      unsigned long DelayMSec;
};
```

```
#ifdef EnableReceiveCommand
const byte cMaxCommandCount = 5;
const int cMaxCommandStringLength = 10; // length("C,255,255,255,12345");
String inputBtString = "", inputPcString = "",
       sCommand[cMaxCommandCount] = { "D","13","0","0","0" };    //將  Digitial Pin 13      設為  0;
bool stringBtComplete = false, stringPcComplete;
byte pinRx =10, pinTx =11;
unsigned long gLastListerningTime, gLastActionTime;

// BlueTooth in Pin10(RX) Pin11(TX)    PWM Pin -------------------------
SoftwareSerial BlueToothSerial(pinRx, pinTx); //10, 11); // RX, TX
#endif

#ifdef AverageHeartBeat
const int cHeartBeatCount = 10;
#endif



#ifndef UseTimer2IRS
unsigned long gLastCheckMs;
#endif

const int cRamboCount = 20;
const int cHeartBeatMin = 40, cHeartBeatMax = 120;
const int cHeartBeatStep = (cHeartBeatMax - cHeartBeatMin) / cRamboCount;
const int cFadeStep = 10;
const int cTimer2IntervalMSec=2;                  // 設定每 2 毫秒一次中斷
const byte cMinSampeDurationMsec = 250;           // 取樣最小間隔毫秒
const byte cSampleCount = 10;                     // 取樣數
const byte cOuputIntervalMSec = 50;               // 每 50 毫秒輸出一次

const byte funcOutputPulse=1;
const byte funcAveragePulse=2;
const byte funcOneColorFadding =4;



TRGB gRamboRGBs[cRamboCount];

// Timer2  掌管  PWM pin 3, 11,  因此不能使用這兩個    Pin ----------------
int pulsePin = 0;                         // 連接到  Pulse Sensor  的  A  訊號
int blinkPin = 13;                        // 用來快閃顯示心跳訊號
int fadePinR = 5, fadePinG = 6, fadePinB = 9;                   // 用來漸暗心跳訊號出現後的
```

```
// 因為這些變數使用在 ISR (中斷 Rountine 內)，因此須加 volatile ----------------------------------------

#ifdef AverageHeartBeat
volatile int gHeartBeats[cHeartBeatCount];
volatile int gCurHeartBeatId=0;
#endif

volatile int BPM;                       // 心跳，次/分
volatile int gPeakTime = 600;           // 紀錄目前的心跳間隔時間
volatile boolean blInPeakPulse = false;         // true when pulse wave is high, false when it's low
volatile boolean blOutputHeartBeat = false;             // becomes true when Arduoino finds a beat.
volatile int gMin_PeakTime = 60000 / 200;       // 假設最大心跳 200 次/分，則最小間隔時間為 60000/200
volatile int gCurSampleID = 0;                  // 目前 Sample ID
volatile int PulseDurations[cSampleCount];      // 心跳間隔時間取樣陣列
volatile unsigned long gLastBeatTime = 0;     // 上一次心跳波峰時間
volatile unsigned long gTotalTime = 0;          // 總共經過時間
volatile int gCurMaxPulse = 512;                // 最高 Pulse 波峰
volatile int gCurMinPulse = 512;                // 最低 Pulse 波峰
volatile int gLastPulseThreshold = 512;    // 紀錄最近的 Pulse Value 平均值
volatile byte decR = 255/cFadeStep, decG = decR, decB = decR;
volatile int gFadeValueR = 0, gFadeValueG = 0, gFadeValueB = 0;                 // 用來更新 FadePin 的 PWM 數值
volatile byte gFunctions =0x00000000;



void buildRamboRGBs(bool ignoreBlueToRed=true)
{
    const byte cR = 0, cG = 1, cB = 2;

    byte aR, aG, aB;
    aR = 255;
    aG = 0;
    aB = 0;

    byte aRGB = 0;
    byte rangeN = 3;
    if (ignoreBlueToRed) rangeN = 2;
    else rangeN = 3;
    byte cRamboStep = (255 * rangeN / (cRamboCount)); // 限制在 R-G, G-B, 因此 rangeN=2
```

```
 for (int i = cRamboCount - 1; i>=0; i--) //從 B 開始往前填入
 {
gRamboRGBs[i].R = aR;
gRamboRGBs[i].G = aG;
gRamboRGBs[i].B = aB;
        gRamboRGBs[i].DelayMSec = 10000 / cRamboCount;

        String s1 = "R" + String(aR) + ", G" + String(aG) + ", B" + String(aB);
        Serial.println(s1);

        if (aRGB == cR)
        {
                aR -= cRamboStep;
                aG += cRamboStep;
                if (aR<cRamboStep)
                {
                        aR = 0;
                        aG = 255;
                        aB = 0;
                        aRGB = cG;
                }
        }
        else if (aRGB == cG)
        {
                aG -= cRamboStep;
                aB += cRamboStep;
                if (aG<cRamboStep)
                {
                        aG = 0;
                        aB = 255;
                        aR = 0;
                        aRGB = cB;
                }
        }
        else //if (aRGB=cB)
        {
                aB -= cRamboStep;
                aR += cRamboStep;
                if (aB<cRamboStep)
                {
                        aB = 0;
```

```
                        aR = 255;
                        aG = 0;
                        aRGB = cR;
                    }
                }
            }

}



void initial_Variables()
{
#ifndef UseTimer2IRS
    gLastCheckMs = millis();
#endif

    // 給予取樣心跳陣列初值 ------------------------------------
    for (int i = 0; i<cSampleCount; i++)
        PulseDurations[i] = 60000 / 72;

#ifdef AverageHeartBeat
    // 給予 gHeartBeats 初值 ----------------------------------------
    for (int i = 0; i < cHeartBeatCount; i++)
        gHeartBeats[i]=72;
#endif
}

#ifdef EnableReceiveCommand

bool GetCompleteStringFromBlueTooth(String &str)
{
    while (BlueToothSerial.available()) {

        //  逐一加入字元直到遇到 \n--------------------
        char inChar = (char)BlueToothSerial.read();

        //  如果字元 = \n    則跳出 -----------
        if ((inChar == '\r') || (inChar == '\n') || (str.length() > cMaxCommandStringLength))
        {
            return (str != "");
        }
```

```
        else
        {
            str += inChar;
            return false;
        }
    }

    return false;
}


bool GetCompleteStringFromPC(String &str)
{
    while (Serial.available()) {
        //  逐一加入字元直到遇到 \n----------
        char inChar = (char)Serial.read();

        //  如果字元 = \n    則跳出 ----------
        if ((inChar == '\r') || (inChar == '\n') || (str.length() > cMaxCommandStringLength))
        {
            return (str != "");
        }
        else
        {
            str += inChar;
            return false;
        }
    }

    return false;
}



bool GetCommand(String inStr, String sCmd[], int &cmdCnt) //    call by reference,  陣列不需加 &
{
    //int cmdCnt=0;
    char *p = &inStr[0];
    char *str;

    cmdCnt = 0;
```

```
        //   以 ","     來拆解字串----------------------------------------
        while (cmdCnt<cMaxCommandCount && (str = strtok_r(p, ",", &p)) != NULL)
        {
                sCommand[cmdCnt] = str;
       sCommand[cmdCnt].toUpperCase();
                cmdCnt++;
        }


#ifdef debug
        //Serial.println(cmdCnt);
#endif

        return (cmdCnt>0);
}

void Process_Command(String sCmd[])
{
#ifdef debug
        /*Serial.println("Process_Command() ");
        Serial.println(sCmd[0]);
        Serial.println(sCmd[1]);
        Serial.println(sCmd[2]);
        Serial.println(sCmd[3]);*/
#endif

        // P,0/1    取消/啟動  輸出  Pulse ----------------
        if (sCmd[0].indexOf("FN") == 0 || sCmd[0].indexOf("fn") == 0) // Analog
        {
                gFunctions = sCmd[1].toInt();
#ifdef debug
      Serial.print("FN: ");    Serial.println(gFunctions, BIN);
#endif
        }
        else if (sCmd[0].indexOf("T") == 0 || sCmd[0].indexOf("t") == 0) //TI, TD
        {

                if (sCmd[0].indexOf("D") == 1 || sCmd[0].indexOf("d") == 1) // TD,200      delay msec
                {
                        //gDelayMsec = sCmd[1].toInt();
                        //String s1 = "DelayMSec : " + sCmd[1];
                        //Serial.println(s1);
```

```
            }
            else if (sCmd[0].indexOf("I") == 1 || sCmd[0].indexOf("i") == 1) // TI, 2000 Idel time msec
            {
                  //gIdleMsec = sCmd[1].toInt() % cMaxIdleMsec;

                  //String s1 = "IdleMSec : " + sCmd[1];
                  //Serial.println(s1);
            }
            else;
      }
      else
      {
#ifdef debug
            BlueToothSerial.println("Command : " + sCmd[0] + "-" + sCmd[1] + "-" + sCmd[2] + "not processed");
#endif
      }
}
#endif


void setup()
{
      pinMode(blinkPin, OUTPUT);          // 心跳時閃一下
      pinMode(fadePinR, OUTPUT);          // 心跳後漸暗
      pinMode(fadePinG, OUTPUT);          // 心跳後漸暗
      pinMode(fadePinB, OUTPUT);          // 心跳後漸暗
      Serial.begin(115200);               // Serail BaudRate

#ifdef   UseTimer2IRS
      // 設定 Timer2 Interrupt，每 2 毫秒中斷一次 ---------------------------------
      interruptTimer2Setup(cTimer2IntervalMSec);
#endif

      initial_Variables;

      buildRamboRGBs();
}



void loop()
{
```

```
#ifdef EnableReceiveCommand
	if (GetCompleteStringFromBlueTooth(inputBtString))
	{
		//PlayMode = cListerning;
		gLastListerningTime = millis();
		stringBtComplete = true;
	}
	else if (GetCompleteStringFromPC(inputPcString))
	{
		//PlayMode = cListerning;
		gLastListerningTime = millis();
		stringPcComplete = true;
#ifdef debug
		//Serial.println(inputPcString);
#endif
	}
	else
	{
	}


	int cmdCnt;
	if (stringBtComplete)
	{
		//   將字串拆成  Pin,Id,Value, Ex: D,13,0     就是將  Digitial Pin 13     設為  0
		if (GetCommand(inputBtString, sCommand, cmdCnt))
		{
			Process_Command(sCommand);
		}

		// clear the string:
		inputBtString = "";
		stringBtComplete = false;

	}
	else if (stringPcComplete)
	{
		//   將字串拆成  Pin,Id,Value, Ex: D,13,0     就是將  Digitial Pin 13     設為  0
		if (GetCommand(inputPcString, sCommand, cmdCnt))
		{
			Process_Command(sCommand);
```

```
            }

            // clear the string:
            inputPcString = "";
            stringPcComplete = false;
        }
#endif


        // 漸減 fadPin 的 pwm 值，漸暗 ---------------------------------
        ledFadeToBeat();


#ifdef   UseTimer2IRS
        delay(20);
#else
        unsigned long nowMs = millis();
        if (nowMs - gLastCheckMs >= cTimer2IntervalMSec)
        {
            OnTimer2Interrupt();
            gLastCheckMs = nowMs;
        }
#endif
}

void ledFadeToBeat()
{
        gFadeValueR -= decR;                                        // 遞減 PWM 值
        gFadeValueR = constrain(gFadeValueR, 0, 255);         // 將 PWM 值限制在 0~255
#ifdef CommonAnode
   analogWrite(fadePinR, 255-gFadeValueR);            // 送出 PWM 值
#else
   analogWrite(fadePinR, gFadeValueR);            // 送出 PWM 值
#endif

#ifdef RGBColorFadding
        if (gFunctions&funcOneColorFadding)
            ;
        else
        {
            gFadeValueG -= decG;
            gFadeValueG = constrain(gFadeValueG, 0, 255);         // 將 PWM 值限制在 0~255
```

```
#ifdef CommonAnode
      analogWrite(fadePinG, 255-gFadeValueG);            // 送出 PWM 值
#else
     analogWrite(fadePinG, gFadeValueG);             // 送出 PWM 值
#endif

              gFadeValueB -= decB;
              gFadeValueB = constrain(gFadeValueB, 0, 255);          // 將 PWM 值限制在 0~255
#ifdef CommonAnode
      analogWrite(fadePinB, 255-gFadeValueB);            // 送出 PWM 值
#else
     analogWrite(fadePinB, gFadeValueB);             // 送出 PWM 值
#endif
        }
#endif

}

void sendDataToProcessing(String sCmd, int data)
{
       Serial.print(sCmd+",");                        // 開頭字元
       Serial.println(data);                         // 數值
}



void OnTimer2Interrupt()
{
       int pulseVal = analogRead(pulsePin);               // 讀取 PulseSensor 數值

       gTotalTime += cTimer2IntervalMSec;                      // 因為 Timer2 每次中斷間隔為 cTimer2IntervalMSec
       int dTime = gTotalTime - gLastBeatTime;          // 用來篩選過近時間內的雜訊

       bool blIsOverMinPeakTime = (dTime > gMin_PeakTime);

       // 更新 波峰和波谷的數值 ------------------------------------------------------------------------------------------
       if (pulseVal < gLastPulseThreshold &&              // 如果 pulse 小於上次的平均值
              blIsOverMinPeakTime)                  // 且間隔時間也超過上次間隔時間的 3/5
       {
              if (pulseVal < gCurMinPulse)  // 如果 pulse 比上次的波谷還小
                     gCurMinPulse = pulseVal;      // 更新波谷數值
       }
```

```
    if (pulseVal > gLastPulseThreshold) //    &&      // 如果 pulse 大於上次的平均值，過濾掉雜訊
        // dTime > gMin_PeakTime)          不過濾時間，所有的波峰都取，以免漏掉真正的波峰
    {
        if (pulseVal > gCurMaxPulse)                // 且比上次的波峰值還大
        {
            gCurMaxPulse = pulseVal;                    // 更新波峰值
        }
    }


    // 開始計算心跳數值 ----------------------------------------------------------------
    if (dTime > cMinSampeDurationMsec)      // 過濾掉太相近的雜訊
    {

        if (gFunctions&funcOutputPulse) //送出 pulseSender 讀到的數值
        if (gTotalTime % cOuputIntervalMSec == 0)
        {
            Serial.println("S," + String(pulseVal) + "," + String(gTotalTime, HEX)+",");
        }

        if ((pulseVal > gLastPulseThreshold) &&         // 如果目前的 pulse 超過上次的平均值
            (blInPeakPulse == false) &&         // 且還沒找到新的 Pulse
            blIsOverMinPeakTime)                // 且間隔時間也超過上次間隔時間的 3/5
        {
            blInPeakPulse = true;                                   // 找到 Pulse
            digitalWrite(blinkPin, HIGH);               // 開啟閃爍的 Pin13 LED
            gPeakTime = dTime;                                      // 紀錄距離上次 Pulse 的時間間隔

#ifdef Debug
            //sendDataToProcessing("Q", gPeakTime);     // 送出波峰間的時間間隔 dT
#endif

            gMin_PeakTime = (gPeakTime / 5) * 3;    // 更新新的 時間間隔 Threshold
            gLastBeatTime = gTotalTime;                             // 更新最後 Pulse 時間

            gCurSampleID %= cSampleCount;                       // 目前填入的陣列位置
            PulseDurations[gCurSampleID] = gPeakTime;   // 填入目前的間隔時間
            gCurSampleID++;                                         // 到下一個陣列空間

            // 計算所有心跳間隔時間加起來的總數 ------------------------------------
            word runningTotal = 0;
            for (int i = 0; i < cSampleCount; i++)
```

```
            runningTotal += PulseDurations[i];

        runningTotal /= cSampleCount;       // 平均每次心跳的間隔時間
        int heartBeat = 60000 / runningTotal;           // 將 60000 毫秒 / 每次間隔 -> 每分鐘心跳數

#ifdef AverageHeartBeat
        if (gFunctions&funcAveragePulse)
        {
            gCurHeartBeatId %= cHeartBeatCount;
            gHeartBeats[gCurHeartBeatId] = heartBeat;
            gCurHeartBeatId++;
            int allHb = 0;
            for (int i = 0; i < cHeartBeatCount; i++)
                allHb += gHeartBeats[i];
            BPM = allHb / cHeartBeatCount;
        }
        else
#endif
            BPM = heartBeat;


        //#ifdef debug
        //sendDataToProcessing("B", BPM);               //送出心跳資料到 Serail
        Serial.println("B," + String(BPM) + "," + String(gTotalTime, HEX) + ",");
//#endif

        int aId = (BPM - cHeartBeatMin) / cHeartBeatStep;
        //constrain(aId, 0, cRamboCount - 1);   // 無效
        // BPM, cHeartBeatMin, cHeartBeatStep...必須 int, 否則 aId 會超出邊界
        if (aId < 0) aId = 0; else if (aId >= cRamboCount) aId = cRamboCount - 1;

        gFadeValueR = gRamboRGBs[aId].R;
        decR = gFadeValueR / cFadeStep;                         //設定 PWM fadPin 從 255 開始遞減

#ifdef RGBColorFadding

        if (gFunctions&funcOneColorFadding)
            ;
        else
        {
            gFadeValueG = gRamboRGBs[aId].G;
            decG = gFadeValueG / cFadeStep;
```

```
                    gFadeValueB = gRamboRGBs[aId].B;
                    decB = gFadeValueB / cFadeStep;
                }
#endif


#ifdef debug
                /*
                sendDataToProcessing("i", aId);
                sendDataToProcessing("r", gFadeValueR);
                sendDataToProcessing("g", gFadeValueG);
                sendDataToProcessing("b", gFadeValueB);
                */
#endif
            }

        }

        if (pulseVal < gLastPulseThreshold &&     // Pulse 比上次平均值低
            blInPeakPulse == true)                // 且上一次是找到波峰，則表示目前由波峰往下走
        {
            digitalWrite(blinkPin, LOW);                        // 關閉 Pin13 LED
            blInPeakPulse = false;                                  // 離開波峰
            int amp = gCurMaxPulse - gCurMinPulse;              // 振福
            gLastPulseThreshold = amp / 2 + gCurMinPulse;       // 更新 Pulse 平均值
            gCurMaxPulse = gLastPulseThreshold;                     // 更新波峰、波谷的數值
            gCurMinPulse = gLastPulseThreshold;
        }

        if (dTime > 2500)                         // 很久沒有發現 HeartBeat
        {
            gFadeValueR = 0;
#ifdef RGBColoreFading
            gFadeValueG = 0;
            gFadeValueB = 0;
#endif

            gLastPulseThreshold = 512;          // 重設 Pulse 平均值
            gCurMaxPulse = 512;                     // 更新波峰、波谷的數值
            gCurMinPulse = 512;
            gLastBeatTime = gTotalTime;         // 更新上次心跳時間
```

```
    }
}
```

中斷服務　Interrupt.ino

```
/*
Timer0 - An 8 bit timer used by Arduino functions delay(), millis() and micros().
Timer1 - A 16 bit timer used by the Servo() library
Timer2 - An 8 bit timer used by the Tone() library

Timer_output Arduino_output    Chip_pin    Pin_name
OC0A (Timer0)       6                  12            PD6
OC0B (Timer0)       5                  11            PD5
OC1A (Timer1)       9                  15            PB1
OC1B (Timer1)       10                 16            PB2
OC2A (Timer2)       11                 17            PB3
OC2B (Timer2)       3                  5             PD3

*/

void interruptTimer2Setup(int timer2IntervalMsec)
{
    TCCR2A = 0x02;      // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
    TCCR2B = 0x06;      // DON'T FORCE COMPARE, 256 PRESCALER

    // SET THE TOP OF THE COUNT TO 124 (16MHz/256/500 = 125, because index start from 0, so count=124) FOR 500Hz
SAMPLE PulseDurations, 500 Hz/sec =>    1000 msec/500 -> 2 MSec
    //OCR2A = 0X7C;
    float scaledFrequency = 16000000 / 256.0;
    float wantedFrequency = 1.0 / (timer2IntervalMsec / 1000.0);    //timer2IntervalMSec
    int tickCount = scaledFrequency / wantedFrequency;
    OCR2A = tickCount - 1;

//#ifdef debug
    //Serial.print("Scaled Frequency: ");    Serial.println(scaledFrequency);
    //Serial.print("Wanted Frequency: "); Serial.println(wantedFrequency);
    //Serial.print("TickCount: "); Serial.println(tickCount);
//#endif

    TIMSK2 = 0x02;      // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
    sei();              // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}
```

```
// TIMER 2 INTERRUPT SERVICE ROUTINE ----------------------------------------------------------------------
ISR(TIMER2_COMPA_vect)    // triggered when Timer2 counts to 124,    0~124 = 125
{
#ifdef debug
        //Serial.print('Tick: ');    Serial.println(millis());
#endif

        cli();                                                  // disable interrupts

        OnTimer2Interrupt();

        sei();                                                  // enable interrupts when youre done!


//#ifdef debug
        //Serial.print('Finish OnTimer2Interrupt(): ');    Serial.println(millis());
//#endif
}
```

監看程式: