中華大學資訊工程學系

107 學年度專題製作期末報告

駕駛生理及環境監控系統

Driver and environmental monitoring system

指導老師：顏金泰 教授

學生：

B10402206 陳嘉宏

B10402202 張佳瑋

B10402137 文永鈞

中華民國 108 年 1 月

摘要

透過本書，我們可以了解到此專題的研究動機以及實作的方法。本書將分為四大章，首先會介紹專題的研究動機，以及研究方向，再來會講解我們實現專題的研究流程。

研究流程會詳細介紹本專題所使用的各種硬體感測器。

第三章程式流程會仔細講解 Arduino 的內部程式運作，APP 的功能以及主機端接收程式的運作，最後會展示成果，並將開源碼貼在網路上供參考使用。

本專題是以物聯網的概念為基礎，透過蒐集感測器的數據，並對這些數據作資料處理，來達到警訊的目的。

由於物聯網概念已經趨於成熟，所以我們能夠獲取許多的資源，並透過 Arduino 達成我們的目的，去解決日常中需要解決的問題。

而台灣的交通車禍死亡率在亞洲地區相較於其他國家，一直處於偏高的狀態，所以我們希望能將物聯網的概念，應用到解決部分車禍問題上。

目錄

第一章　概論

1. 專題簡介

1-1 研究動機

根據交通部統計，2013 年至 2016 年平均每年因交通事故，而造成死傷的人數高達 40.15 萬人，相當於一個基隆市的人口。

道路交通事故在 30 日內的死亡人數，雖然有逐年下降，但 2016 年的造成死亡的人數仍高達 2877 人，相當於 1 次 921 大地震的死亡人數，其中酒駕在 2016 年造成 533 人死亡，1859 以上的人受傷。

分心對駕駛而言是造成交通事故主要原因之一，台灣每年因駕駛分心或疲勞而發生事故比例約占車禍總事故的 20%，高居各類事故原因的第二名。105 年國內因疲勞駕駛而肇事多達 2304 人次，造成 2675 人次傷亡。

2018-04-24 就發生大貨車駕駛因為疲勞打瞌睡，造成追撞國道警察的事故。

儘管運輸事故死亡率呈現下降的趨勢，但是與其他國家相比，台灣交通事故死亡率依舊偏高。

肇事與死亡者，皆以機車騎士比例最高，除了遊覽車的檢驗與駕駛安全之外，如何具體有效降低交通事故刻不容緩。

歐美各國的交通事故統計分析表明，交通事故中 80%～90%是個人因素造成的。

根據美國國家公路交通安全署的統計，在美國的公路上，每年由於司機在駕駛過程中陷入睡眠狀態而導致大約 10 萬起交通事故，約有 1500 起直接導致人員死亡，711 萬起導致人員傷害。

在歐洲的情況也大致相同，如在德國境內的高速公路上 25%導致人員傷亡的交通事故，都是由疲勞駕駛引起的。

疲勞駕駛和**酒駕**一直被列為引發交通事故的主要原因之一，必須要高度的警惕。

對於一般民眾的交通事故，只能透過政府機關的執法以及宣導，才能一點點降低，像是新竹和台北位學童教育所規劃的簡易型兒童交通公園，研議駕訓班強化管理、加長考照時數、科技執法，無人機、監控設備等積極推動，都在導正駕駛人行為。但是對於運輸業方面的公司，就可以透過感測器去蒐集駕駛員工的生理狀況回傳給公司，達到隨時監控駕駛員工的生理狀況，並有系統的處理當下發生的情況。在數據異常時能夠及時回報給公司以及員工本身，盡可能防止員工因疲勞或是酒後駕駛所釀成的事故。

1-2 研究方法

我們選擇 Arduino 為硬體平台，將各種感測器的參數，透過 HC － 06 藍芽模組和使用 Android Studio 撰寫的 APP，進行手機和 Arduino 的連接，讓手機能夠讀取觀看當前的數據。

並且同使藉由 GPS 的功能取得當前所在地的經緯度，最後以 TCP 的方式傳遞資訊給電腦主機。

主機端我們也撰寫了一個 JAVA 程式來讀取目前手機端所傳遞的資料，一旦資料異常就會發出警訊提醒後台人員。

1-3 研究目標

本專題以物聯網的概念為基礎，探討如何以科技的方式，監控司機的生理狀況以及車輛目前的環境狀況。

最終目的是讓系統能夠即時**對異常資料發出警訊**，達到**防止疲勞駕駛及酒後駕駛**，以及**隨時掌握公司貨車狀況**的目的，以促成值得客戶信賴的運輸公司！

2. 軟體平台簡介

2-1 Arduino 簡介

Arduino 是一個單晶片微控制器，它使用了 Atmel AVR 單片機，建構於簡易輸出/ 輸入（simple I/O）介面板，並且具有使用類似 Java、C 語言的開發環境，我們使用 的編譯器版本為 1.8.7。

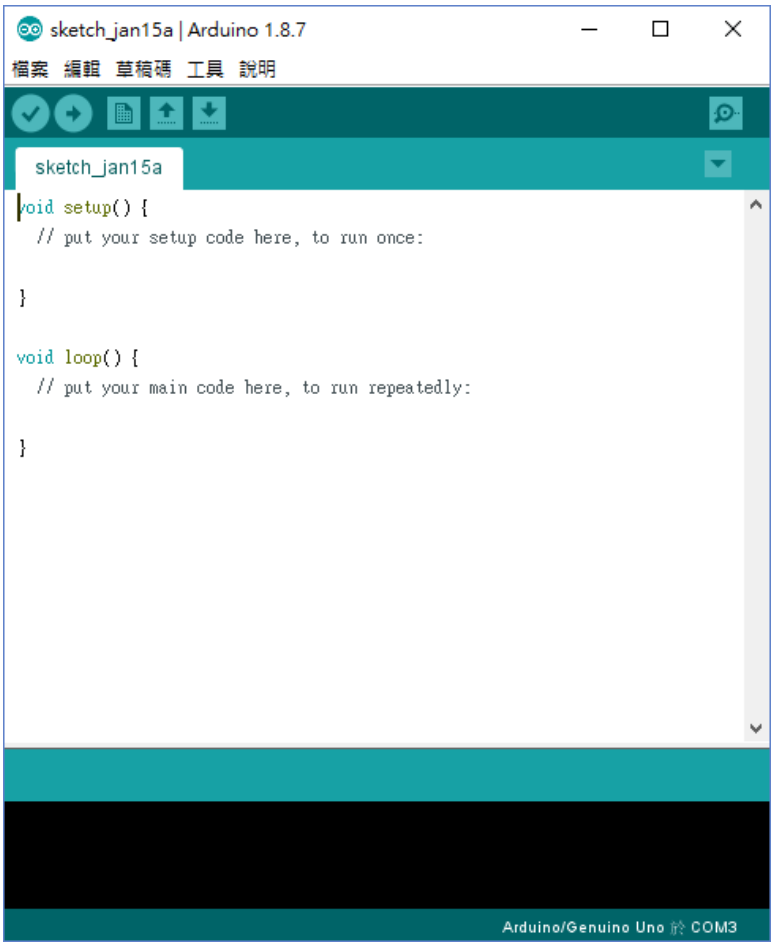


圖 1-1　Arduino 使用者介面

圖 1-2　Arduino Logo

2-2 Arduino 的特色

1. 開放源碼：

不僅軟體是開放源碼，硬體也是開放的。軟體的開發環境可在網上免費下載，而 Arduino 的電路設計圖也可從官方網站自行下載，依據自身之需求進行修，但須要符合創用 CC 授權條款(創用 CC 授權條款)。

2.開發簡單：

傳統上在以往的硬體環境中，要開發微控制器的程式，開發者需要具備電子、電機及相關科系的背景，一般人需花費大量時間才能有機會進入這個開發環境中。

Arduino 學習門檻較為簡單，不需要電子電機相關科系的背景，也可以很容易學會 Arduino 相關互動裝置的開發。

3.資源開放、豐富：

由於 Arduino 以公開共享為基礎，多數人都樂於分享自己的的創品，網路上能找的創作案子非常豐富。

以此會基礎，有時只需要參考分享者的作品，依據自身的需求行調整，就可以在短時間內完成自己的創作。

4.成本較低：

由於 Arduino 使用低價格的微處理控制器(Atmel AVR)，一般一張微控制器板子要好幾千塊，但 Arduino 控制板只需要數百台幣。

2-3 Android Studio 簡介

Android Studio 是官方 Android 程式編寫的開發環境，藉由圖形化介面的編譯與程式碼撰寫間的合作，讓開發難度大幅降低，編輯彈性更好。

以 Open Source IntelliJ IDEA 為基礎所發展，在 Windows、OS X 和 Linux 平台上均可執行。

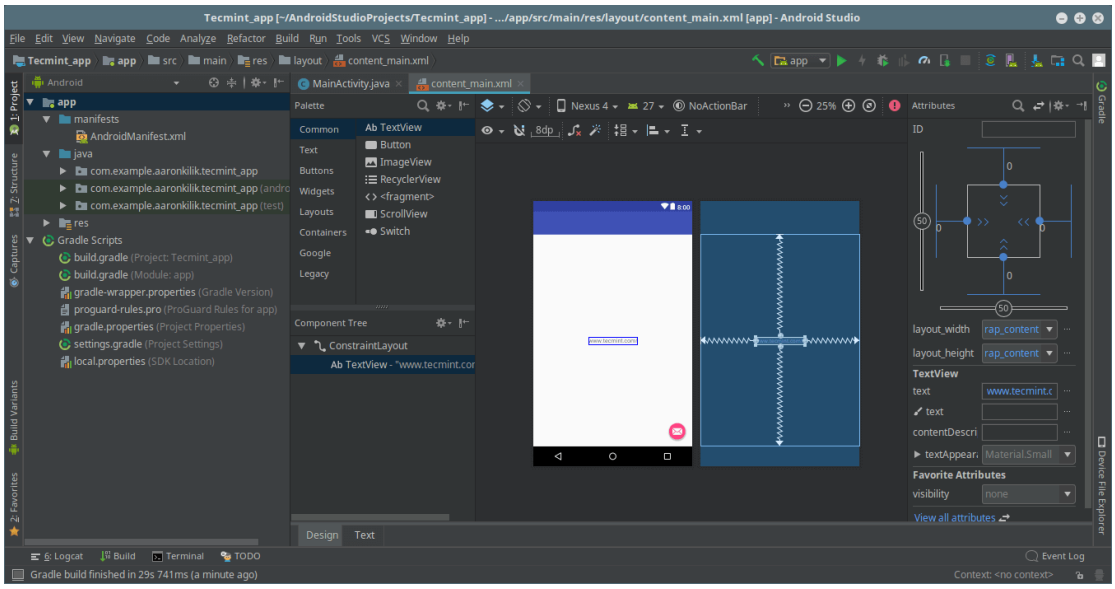


圖 2-1　Android Studio 使用者介面

圖 2-2　Android Studio　Logo

2-4 NetBeans 簡介

NetBeans IDE 是一個開發環境，供程式設計師撰寫、編譯、除錯和部署程式的一個工具。

他採用 Java 編寫而成，但能夠支援各種程式語言。另外，也有相當龐大的模組來擴充 NetBeans IDE。

本專題中我們用於開發 JAVA 程式，使用版本為 8.0.2。

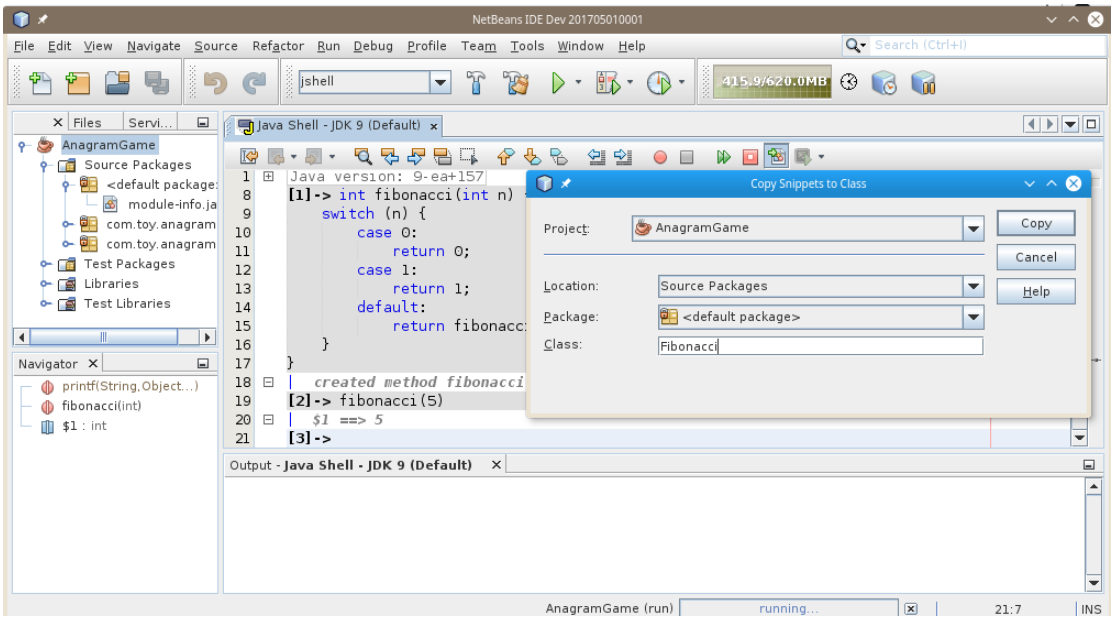


圖 2-3　NetBeans 使用者介面

圖 2-4　NetBeans Logo

3. 硬體平台簡介

3-1 Arduino UNO R3

Table 1. Specifications of Arduino Board

Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 Ma
DC Current for 3.3V Pin	50mA
Flash Memory	32 KB of which 0.5KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Length	68.6 mm
Weight	25g

圖 3-1　Arduino UNO 規格

我們選用 Arduino UNO R3 為開發板，Arduino UNO 為入門級的開發板，由於 IO 部份我們並不會使用太多、更高階的功能我們並不會使用到，加上成本考量，這張便宜又實用的板子是我們的好選擇。

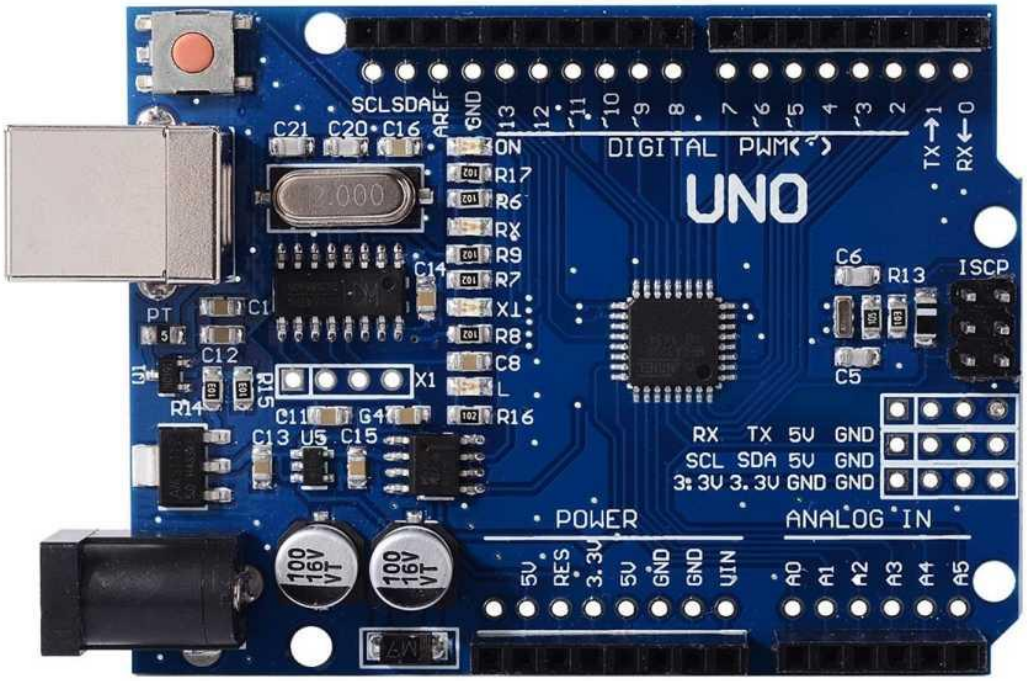


圖 3-2　Arduino UNO 圖片

3-2 Android 手機

由於我們是使用 Android Studio 開發軟體，所以需要一台 Android 手機展示成果。

我們使用的版本為 Android 8.0.0

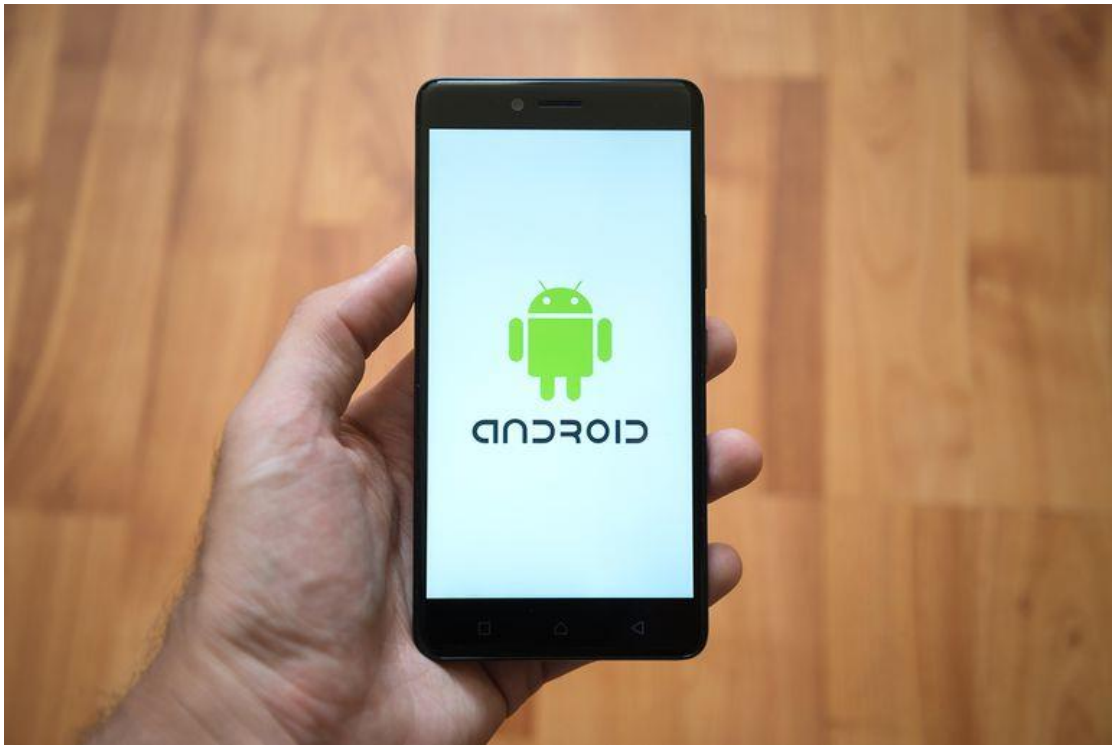


圖 3-3　Android Phone 圖片

3-3 筆電

為了模擬主機接收手機傳遞的資訊，我們使用筆記型電腦當作主機平台，並且使用 JAVA 撰寫程式接收手機傳送的資料。

圖 3-4　筆電圖片

第二章　研究步驟

1. 系統規劃

1-1 系統流程

實際使用的時候，我們會將 arduino 與各種 sensor 放置車內。

我們會將蒐集到的數據傳至手機，經由 APP 做資料處理以後，最後再傳回給公司電腦，方便電腦端能夠隨時監控數據有沒有異常。

如果有異常就跳出警告提醒後台人員，以便後台人員在發現異常之後，能夠迅速與司機構通，確認情況。

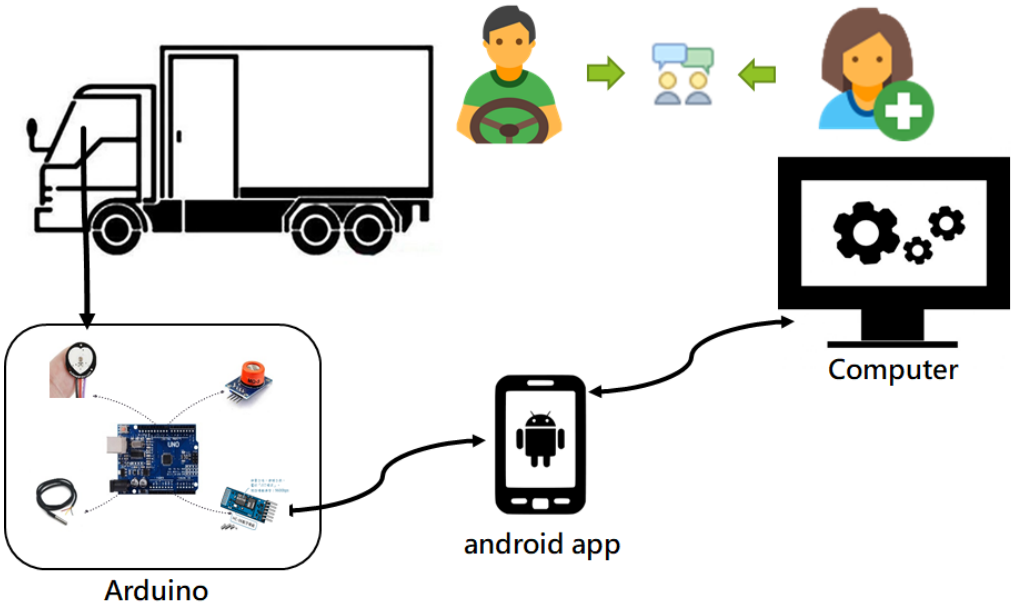


圖 1-1　系統流程圖 (1)

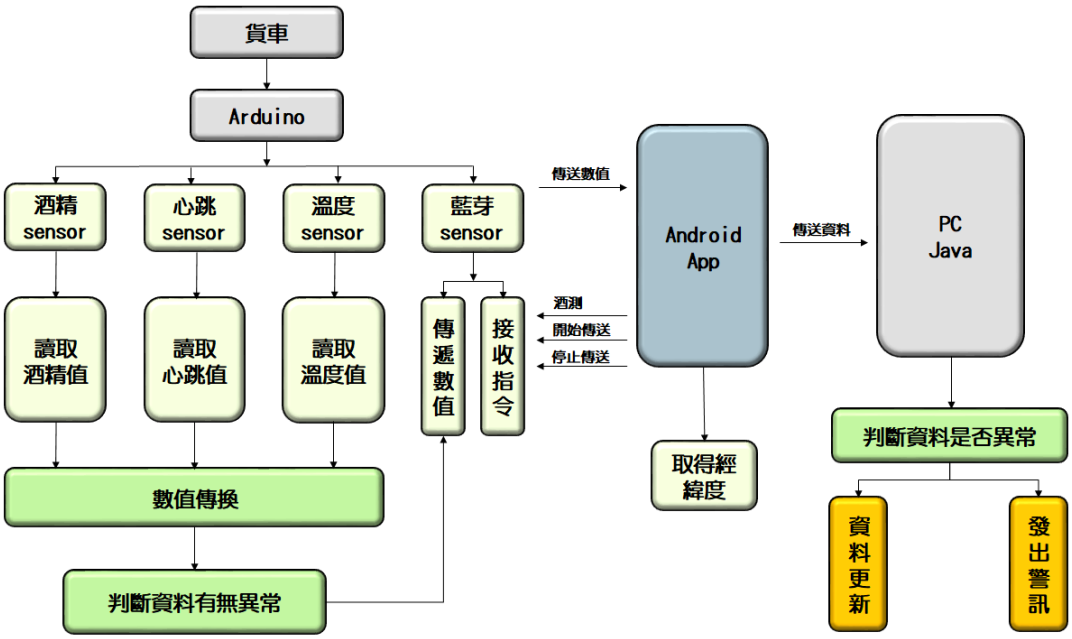

貨車
Arduino
酒精 sensor
心跳 sensor
溫度 sensor
藍芽 sensor
讀取 酒精值
讀取 心跳值
讀取 溫度值
傳遞 數值
接收 指令
數值傳換
判斷資料有無異常
傳送數值
酒測
開始傳送
停止傳送
Android App
傳送資料
PC Java
取得經 緯度
判斷資料是否異常
資料更新
發出警訊

圖 1-2　系統流程圖(2)

2. 零件與原理介紹

2-1 零件清單

我們使用了以下 arduino 零件：

1. Arduino UNO
2. Arduino Pulse sensor
3. MQ-3 氣體傳感器
4. DS18b20 溫度感測器
5. HC-06 Bluetooth sensor

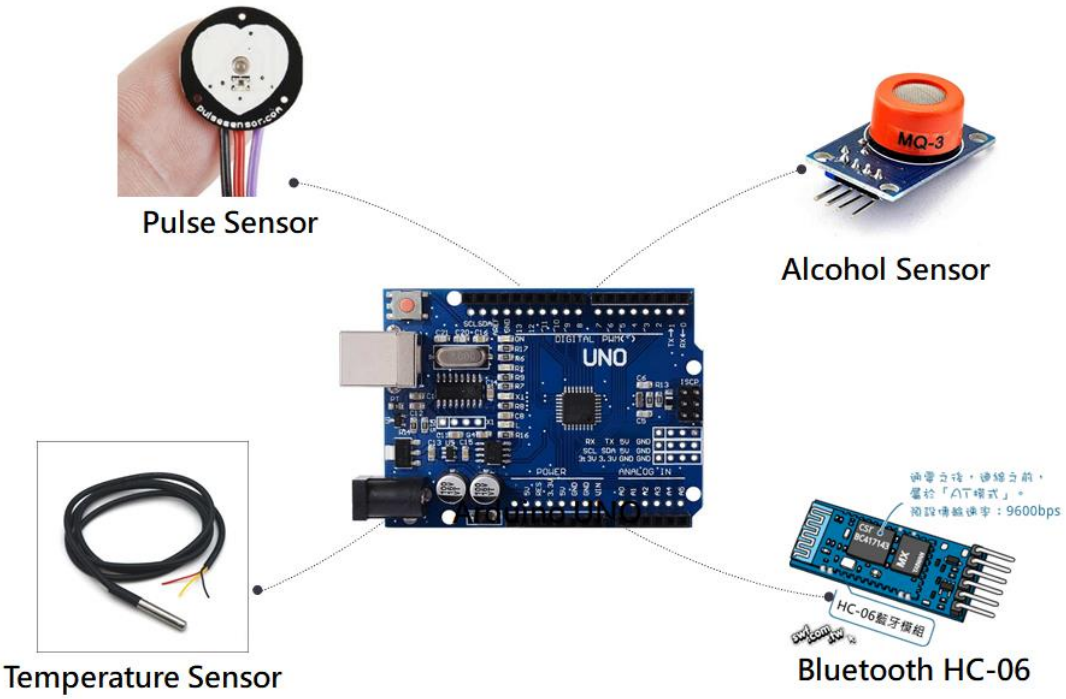


圖 2-1　arduino 零件

2-2 arduino sensor 介紹

1. Pulse sensor 心跳感測模組

規格：

供電電壓：3.3~5V
檢測信號類型：光反射信號（PPG）

輸出信號類型：模擬信號

輸出信號大小：0~VCC

電流大小：~4ma（5v 下）

圖 2-2　Pulse sensor 心跳感測模組

功能原理

PulseSensor 是一款用於脈搏心率測量的光電反射式模擬傳感器。

將其佩戴於手指、耳垂等處，利用人體組織在血管搏動時造成透光率不同來進行脈搏測量。

傳感器對光電信號進行濾波、放大，最終輸出模擬電壓值。

單片機通過將採集到的模擬信號值轉換為數字信號，再通過簡單計算就可以得到心率數值。

反射光脈搏測量原理

PPG 量測血管中血流量 （ 血紅素 ） 的變化

通常綠色光量手腕 ， 紅色光或紅外光量手指

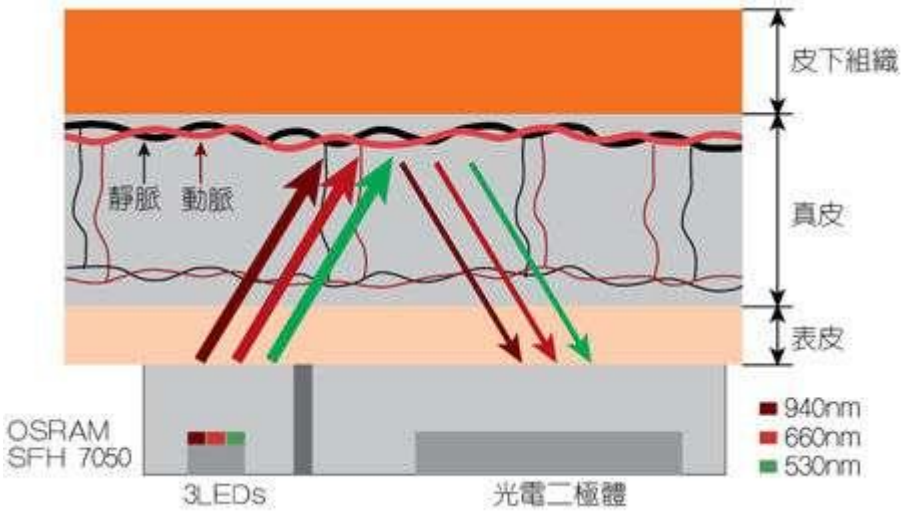


圖 2-3　感光原理示意圖

(http://sosorry.s3.amazonaws.com/raspberrypi/doc/slide/20170515_raspberry-pi-pulse-sensor.pdf)

接腳定義

傳感器只有三個引腳，分別為信號輸出 S 腳 、電源正極 VCC 以及電源負極 GND，供電電壓為 3.3V - 5V，可通過杜邦線與開發板連接。

上電後， 傳感器會不斷從 S 腳輸出採集到的電壓模擬值。需要注意的是，印有心形的一面才是與手指接觸面，在測量時要避免接觸佈滿元件的另一面，否則會影響信號准確性。

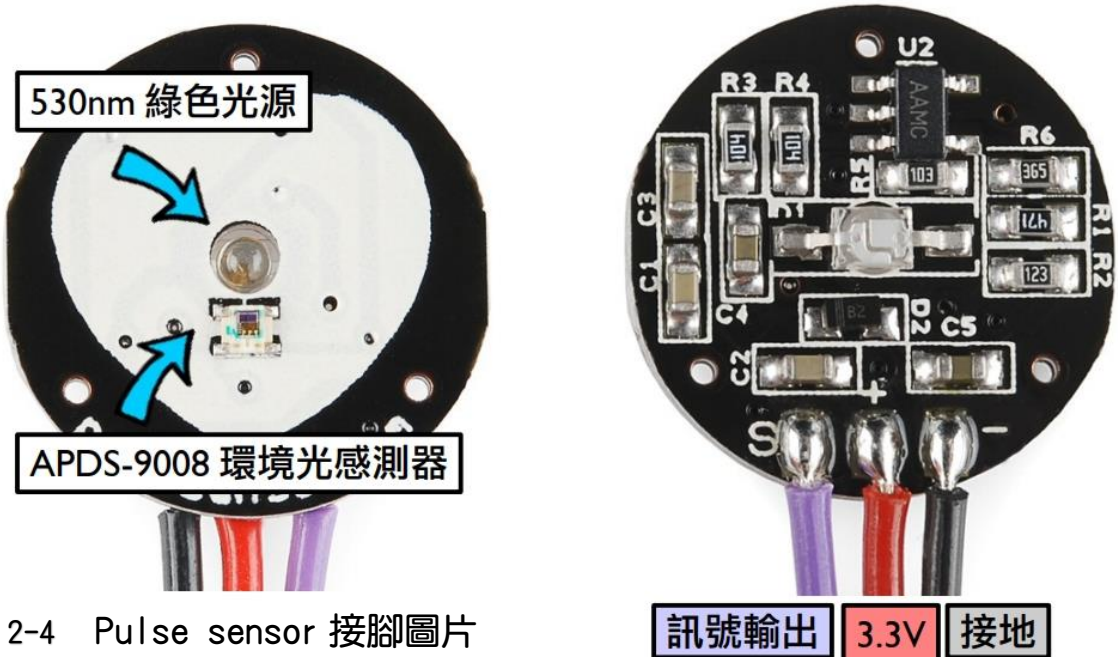


圖 2-4　Pulse sensor 接腳圖片

電壓值轉 BPM

計算心率值 —— 採樣數據處理算法

心率指的是一分鐘內的心跳次數，得到心率最笨的方法就是計時一分鐘後數有多少次脈搏。

但這樣的話每次測心率都要等上個一分鐘才有一次結果，效率極低。另外一種方法是，測量相鄰兩次脈搏的時間間隔，再用一分鐘除以這個間隔得出心率。這樣的好處是可以實時計算脈搏，效率高。

由此引出了 IBI 和 BPM 兩個值的概念：

IBI： 相鄰兩次脈搏的時間間隔（單位：ms）

BPM(beats per minute)：心率，一分鐘內的心跳次數

且：BPM = 60 / IBI

由上面的分析可以得出，我們的最終目的就是要求出 IBI 的值，並通過 IBI 計算出實時心率。

核心操作 —— 識別一個脈搏信號

無論是採用計數法還是計時法，只有能識別出一個脈搏，才能數出一分鐘內脈搏數或者計算兩個相鄰脈搏之間的時間間隔。

那怎麼從採集的電壓波形數據判斷是不是一個有效的脈搏呢？

顯然，可以通過檢測波峰來識別脈搏。

最簡單的方法是設定一個閾值，當讀取到的信號值大於此閾值時便認為檢測一個脈搏。

但這裡存在兩個問題。

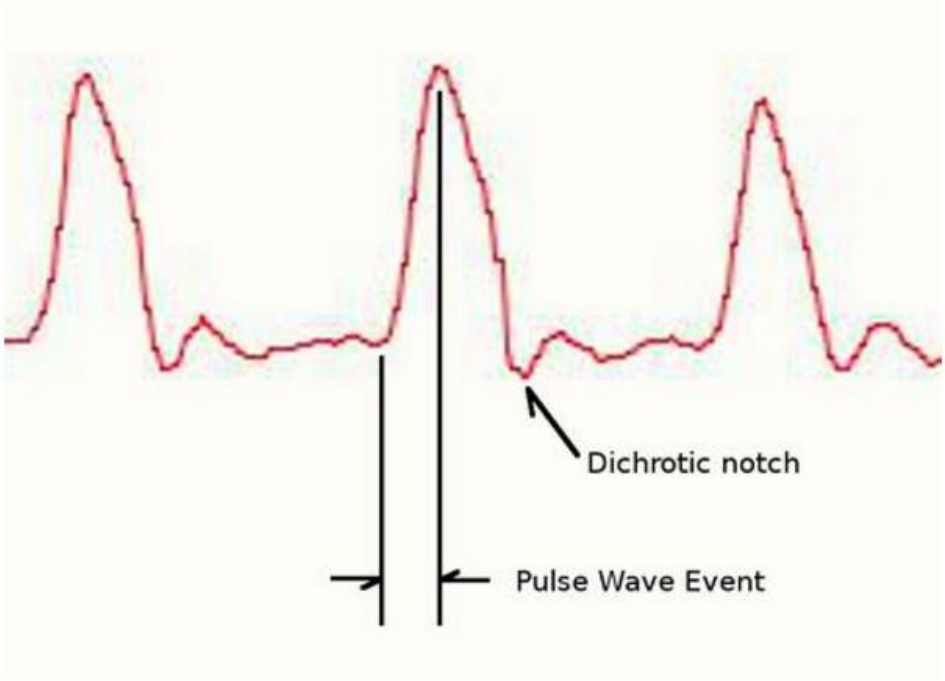


圖 2-5　閥值示意圖

問題一：閾值的選取

作為判斷的參考標尺，閾值該選多大？ 10？100？還是 1000？我們不得而知，因為波形的電壓範圍是不確定的，振幅有大有小並且會改變，根本不能用一個寫死的值去判斷。

可以看出，兩個形狀相同波形的檢測結果截然不同 —— 同樣是波峰，在不同振幅的波形中與閾值比較的結果存在差異。實際情況正是如此：傳感器輸出波形的振幅是在不斷隨機變化的，想用一個固定的值去判定波峰是不現實的。

既然固定閾值的方法不可取，那自然想到改變閾值 —— 根據信號振幅調整閾值，以適應不同信號的波峰檢測。通過對一個週期內的信號多次採樣，得出信號的最高與最低電壓值，由此算出閾值，再用這個閾值對採集的電壓值進行判定，考慮是否為波峰。

也就是說電壓信號的處理分兩步，首先動態計算出參考閾值，然後用用閾值對信號判定、識別一個波峰。

問題二：特徵點識別

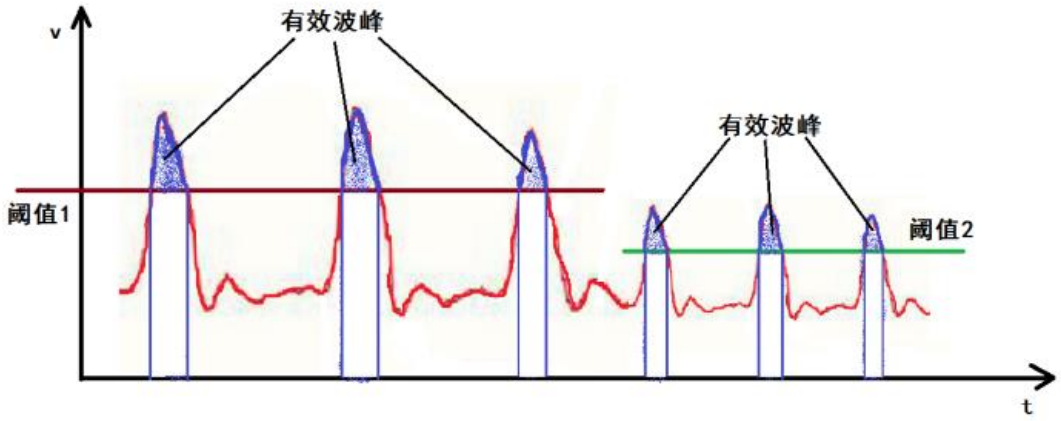


圖 2-6　峰值示意圖

上面得出的是一段有效波形，而計算 IBI 只需要一個點。

需要從一段有效信號上選取一個點，這裡暫且把它稱為特徵點，這個特徵點代表了一個有效脈搏，只要能識別到這個特徵點，就能在一個脈搏到來時觸發任何動作。

通過記錄相鄰兩個特徵點的時間並求差值，計算 IBI 便水到渠成。那這個特徵點應該取在哪個位置呢，從官網算法說明可以看出，官方開源 arduino 代碼的 v1.1 版本是選取信號上升到振幅的一半作為特徵點。

我們可以捕獲這個特徵點作為一個有效脈搏的標誌，然後計算 IBI。

算法整體框架

分析得出算法的整體框架如下：

緩存一個波形週期內的多次採樣值，求出最大最小值，計算出振幅中間值作為信號判定閾值。

通過把當前採樣值和上一採樣值與閾值作比較，尋找到「信號上升到振幅中間位置」的特徵點，記錄當前時間。

尋找下一個特徵點並記錄時間，算出兩個點的時間差值，即相鄰兩次脈搏的時間間隔 IBI。

最後由 IBI 計算心率值 BPM。

BPM = 60 / IBI。

(https://zhuanlan.zhihu.com/p/27665378)

2. MQ-3 酒精感測模組

規格：

PCB 尺寸：30mm X20mm
主要晶片：LM393、ZYMQ-3 氣體感測器
工作電壓：直流 5 伏

介紹：

具有信號輸出指示。

雙路信號輸出（類比量輸出及 TTL 電平輸出），TTL 輸出有效信號為低電平。(當輸出低電平時信號燈亮，可直接接單片機)，類比量輸出 0~5V 電壓，濃度越高電壓越高。

對乙醇蒸氣具有很高的靈敏度和良好的選擇性，具有長期的使用壽命和可靠的穩定性，快速的回應恢復特性，可應用於**機動車駕駛人員**及其他嚴禁酒後作業人員的現場檢測，也適用於**其他場所乙醇蒸氣的檢測**。

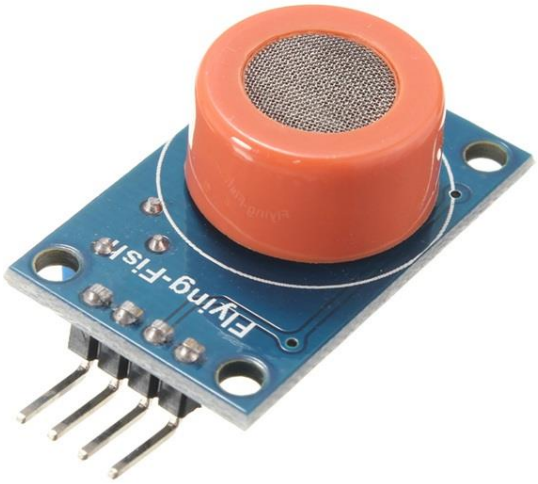


圖 2-7　MQ-3 圖片

數值轉換：

因為酒精值比較難換算成血液含酒精量，所以我們直接使用測得的值來傳輸。

我們根據網路上實測的結果來分析空氣中的酒精濃度。

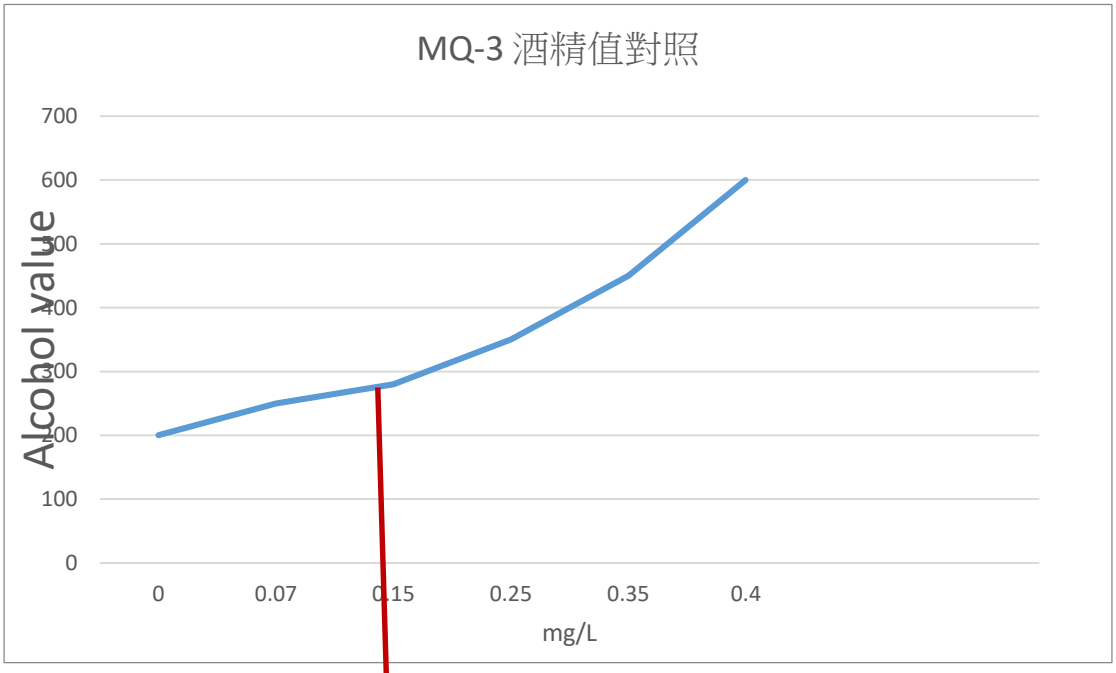
0 < Alcohol value < 200 => You are sober.

200 <= Alcohol value < 280 => You had a beer.

280 <= Alcohol value < 350 => Two or more beers.

350 <= Alcohol value < 450 => You over BAL!!

Alcohol value =>450 => you are drunk!!



酒精值約 **300** 時會超標 **(約 0.15mg/l)**

根據 MQ3 的使用說明書，以及網路實測的結果，我們設定測得的酒精值在 300 之內都算正常，超過 300 就算酒測超標。

(https://www.youtube.com/watch?v=x0ob4BuTr0E)

3. 溫度感測模組

參數:

1. 感溫範圍寬 -55℃ ～ +125℃
2. 無需外部元件,獨特的單總線接口
3. 不銹鋼外殼(6*50mm)
4. 出線:200 公分(2 米)
5. 輸出引線:紅色(VCC),藍色(DATA),黑色(GND)
6. 供電:3.0V～5.5V
7. 可調分辨率:9～12 位

介紹：

探頭採用原裝 DS18B20 溫度傳感器芯片,芯片每個引腳均用熱縮管隔開,防止短路。內部封膠,防水防潮,不銹鋼管封裝防生鏽。

可以用於冰庫的**溫度監控**，或是**偵測高溫防止火災**。

數值轉換：

因為是直接使用廠商的函式庫，所以不需要再撰寫轉換數值的程式碼。

接腳說明：

紅色(VCC)，藍(黃)色(DATA)，黑色(GND)

圖 2-8　DS18b20 溫度感測器圖片

4. HC-06 藍芽模組

規格：

PCB 尺寸：37.3mm*15.5mm
重量：3.5g
輸入電壓：3.6V-6V
電源防反接，反接無法作用
電平：3.3V
有效距離：10 米
透明熱縮管保護

接腳說明

VCC：正極
GND：負極
RXD：接收端
TXD：發送端

簡介：

HC-06 藍牙模組，此款為被動模式，也就是說，HC-06 只能等待被連接，而無法主動連接其他藍芽產品。

不過本專題只需要一個能夠接收指令和傳遞數據的媒介，所以也能符合本專題的要求。

圖 2-9　HC-06 藍芽模組圖片

第三章　程式實作

1. Arduino 程式

Arduino 的程式設計，主要有兩樣工作。

(1) **數值轉換**：將感測器接收到的電壓值轉換為我們習慣的單位。
比如說心跳轉為 BPM，溫度轉為攝氏。

(2) **藍芽傳輸**：使用 HC － 06 模組開啟藍芽，傳送封包與手機 APP 溝通。

arduino 程式流程圖：

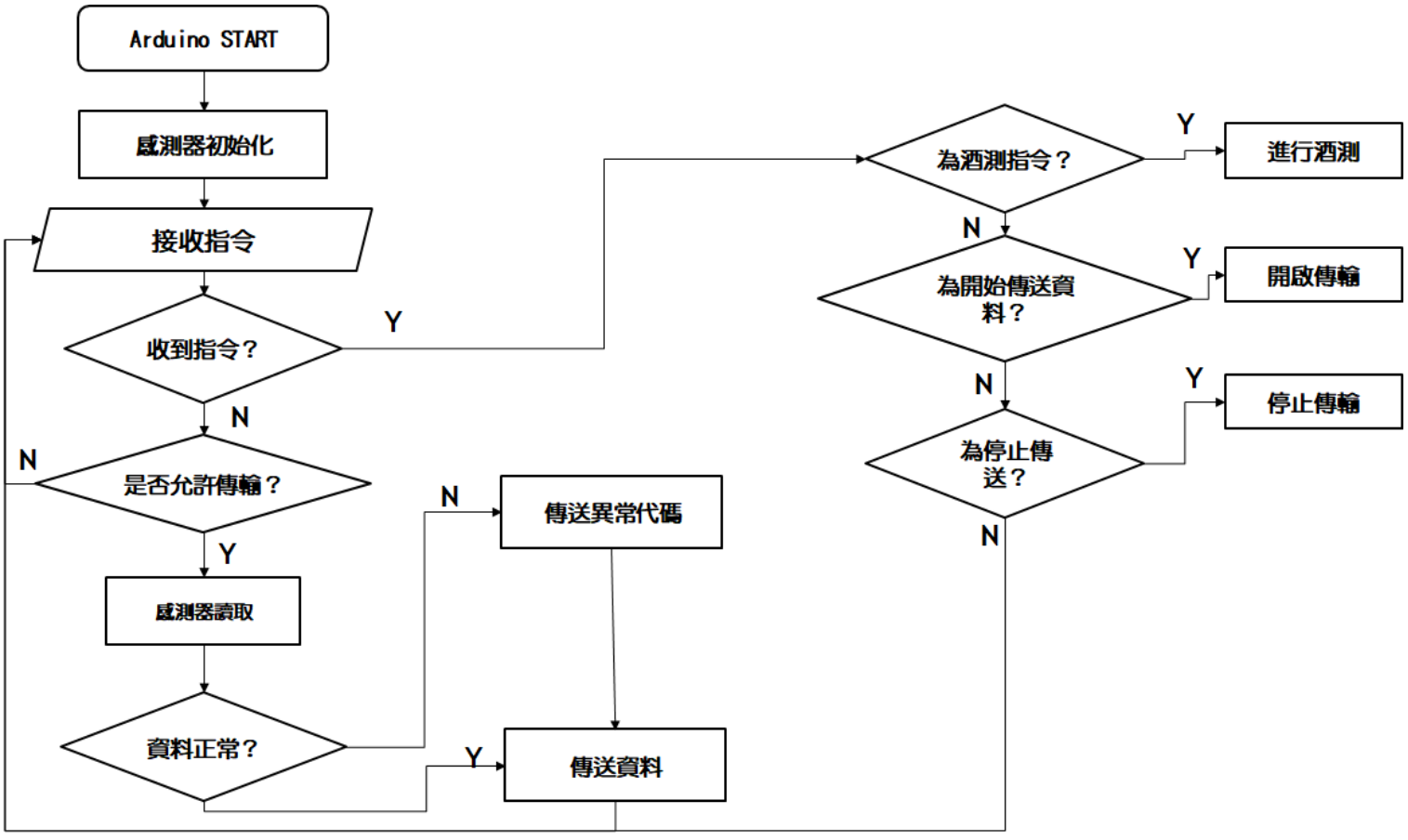


圖 1-1　arduino 程式流程圖

程式開始時，會進行初始化的動作，定義每個腳位的工作。

```
void setup() {
  //pinMode(blinkPin,OUTPUT);         // pin that will blink to your heartbeat!
  Serial.begin(9600);               // we agree to talk fast!
  BT.begin(9600); //HC-06 預設 baud
  interruptSetup();                 // sets up to read Pulse Sensor signal every 2mS
  //pinMode(DOUTpin, INPUT);//sets the pin as an input to the arduino
  pinMode(AOUTpin,INPUT);
  sensors.begin();                  // 開啟溫度感測器

  BT.println("Initialization Complete !");

}

void interruptSetup()
{
  // Initializes Timer2 to throw an interrupt every 2mS.
  TCCR2A = 0x02;     // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
  TCCR2B = 0x06;     // DON'T FORCE COMPARE, 256 PRESCALER
  OCR2A = 0X7C;      // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
  TIMSK2 = 0x02;     // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
  sei();             // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}
```

圖 1-2-1 arduino 初始化

初始化完成後，會印出一段文字，表示完成。

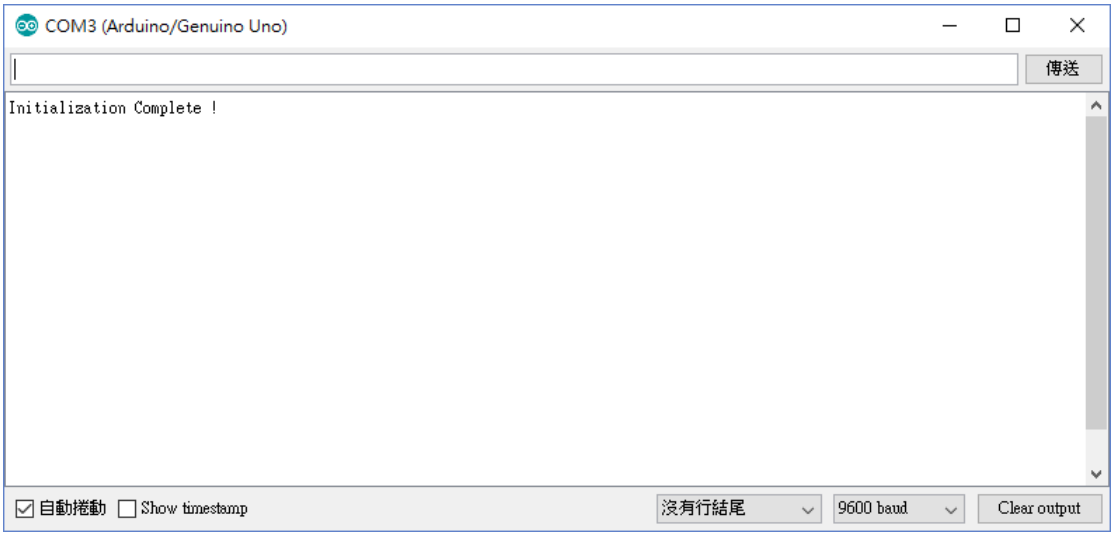


圖 1-2-2 arduino 初始化文字

接著 arduino 會等待手機端做連接，以及等待後續的指令，如果沒有接收到指令，則會一直處於待機的動作。

```
void loop() {
   RecieveData();//若有指令會收
   PrintRecieveData(startRecieve);//印出收到的指令 + 執行指令
   .
   .
   .
}


void  RecieveData(){//收資料用
    while(BT.available()) //如果有收到資就一直收到完
  {
    startRecieve = true;
    val=BT.read(); //每次接收一個字元
    recieveData += val; //字元組成字串
    delay(100);//收字延遲 太短會亂碼
  }
  }

void PrintRecieveData(bool SR){
    if(SR)
  {
  startRecieve = false;
  BT.println(recieveData); //呈現收到字串
  delay(100);
   .
   .
   .
```

圖 1-2　arduino 迴圈 接收指令

若 arduino 接收到指令後，會判斷收到的指令是否有效，一共有三種指令，分別為開啟酒測、開始傳輸與停止傳輸。

酒測指令為 ALT，開始傳輸為 SRD，停止為 SPD。

開啟酒測的功能用於駕駛員的自我檢測，開始傳輸使用於 HC-06 與手機接之後，停止傳輸則是用於 HC-06 與手機斷開連接時使用。

```
//////////////////////如果收到酒測指令
  if(recieveData == "ALT")//ALT = 酒測
  {
    //以上初始化
    int i;
    for(i=0;i<5;i++){
      Alcoholvalue= analogRead(AOUTpin);
      BT.print("酒測中，");
      BT.print(5-i);
      BT.println("秒...");//倒數

      BT.print("Alcoholvalue : ");
      BT.println(Alcoholvalue);
      if(Alcoholvalue>=300)break;
        delay(1000);//總共可以測 5s
      }
        if(i==5){//5秒內數值正常 酒測通過
        ALT = true;
        BT.println("ALPASS");
        ALtest = true; //酒測通過
        }
      else{
        ALT = false;
        BT.println("ALFAIL");
        ALtest = false;//酒測失敗
        }
    }
//////////////^^如果收到酒測指令
```

圖 1-2　arduino 酒測指令

酒測指令很單純，如果 5 秒之內數值都在 300 以內就會顯示 PASS，否則就顯示 FALL。

```
    else if(recieveData == "SRD"){//開始傳輸
     Start_Transfer_data = true;
     }
     else if(recieveData == "SPD"){//停止傳書
       Start_Transfer_data = false;
       }
   recieveData = "";//清除指令
   }
   }
```

圖 1-2　arduino 傳輸指令

傳輸指令單純改變一個布林變數“Start_Transfer_data”，此變數用於 loop 迴圈中，判斷是否要接收資料。

```
void loop() {
   RecieveData();//若有指令會收
   PrintRecieveData(startRecieve);//印出收到的指令 + 執行指令

   if(Start_Transfer_data){//如果收到開始傳資料的指令才做
    ...
   }
```

圖 1-3　arduino 指令用於 loop 迴圈

能夠判斷是否要傳輸指令後，就開始撰寫接收指令的程式。

第一行為心跳感測器初始化，接著載入讀取心跳值的函數，這邊要注意，由於心跳需要讀取一段時間之後判斷是否為有效心跳，否則一直讀取只是單純無意義的電壓值而已。

所以心跳值採用 triggered 的方式，若心跳判斷為有效才讀取，否則一直定義心跳值為 0。

```
.
.
.
serialOutput();//心跳初始化
error_code = 0;
error_str = "";
if (QS == true) // A Heartbeat Was Found 若收到心跳 則將心跳值存入 BPM
 {
   // BPM and IBI have been Determined
   // Quantified Self "QS" true when arduino finds a heartbeat
   serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.
   QS = false; // reset the Quantified Self flag for next time
 }
else {//若沒有就直接印 0
   BPM = 0;
 }
   ReadAlcohol();//讀取酒精值
   ReadTemperature();//讀取溫度
   .
   .
   .
```

圖 1-4　arduino sensor 讀取程式

```
ISR(TIMER2_COMPA_vect) //triggered when Timer2 counts to 124
{
  cli();                                      // disable interrupts while we do this
  Signal = analogRead(pulsePin);              // read the Pulse Sensor
  sampleCounter += 2;                         // keep track of the time in mS with this variable
  int N = sampleCounter - lastBeatTime;       // monitor the time since the last beat to avoid noise
                                              //  find the peak and trough of the pulse wave
  if(Signal < thresh && N > (IBI/5)*3) // avoid dichrotic noise by waiting 3/5 of last IBI
    {
      if (Signal < T) // T is the trough
      {
        T = Signal; // keep track of lowest point in pulse wave
      }
    }

  if(Signal > thresh && Signal > P)
    {          // thresh condition helps avoid noise
      P = Signal;                             // P is the peak
    }                                         // keep track of highest point in pulse wave

  //  NOW IT'S TIME TO LOOK FOR THE HEART BEAT
  // signal surges up in value every time there is a pulse
  if (N > 250)
  {                                   // avoid high frequency noise
    if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )
      {
        Pulse = true;                               // set the Pulse flag when we think there is a pulse
        digitalWrite(blinkPin,HIGH);                // turn on pin 13 LED
        IBI = sampleCounter - lastBeatTime;         // measure time between beats in mS
        lastBeatTime = sampleCounter;               // keep track of time for next pulse
```

```
        if(secondBeat)
        {                       // if this is the second beat, if secondBeat == TRUE
          secondBeat = false;                  // clear secondBeat flag
          for(int i=0; i<=9; i++) // seed the running total to get a realisitic BPM at startup
          {
            rate[i] = IBI;
          }
        }

        if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE
        {
          firstBeat = false;                   // clear firstBeat flag
          secondBeat = true;                   // set the second beat flag
          sei();                               // enable interrupts again
          return;                              // IBI value is unreliable so discard it
        }
      // keep a running total of the last 10 IBI values
      word runningTotal = 0;                  // clear the runningTotal variable

      for(int i=0; i<=8; i++)
        {               // shift data in the rate array
          rate[i] = rate[i+1];                  // and drop the oldest IBI value
          runningTotal += rate[i];              // add up the 9 oldest IBI values
        }

      rate[9] = IBI;                          // add the latest IBI to the rate array
      runningTotal += rate[9];                // add the latest IBI to runningTotal
      runningTotal /= 10;                     // average the last 10 IBI values
      BPM = 60000/runningTotal;               // how many beats can fit into a minute? that's BPM!
      QS = true;                              // set Quantified Self flag
      // QS FLAG IS NOT CLEARED INSIDE THIS ISR
```

```
    }
  }

  if (Signal < thresh && Pulse == true)
    {   // when the values are going down, the beat is over
      digitalWrite(blinkPin,LOW);            // turn off pin 13 LED
      Pulse = false;                         // reset the Pulse flag so we can do it again
      amp = P - T;                           // get amplitude of the pulse wave
      thresh = amp/2 + T;                    // set thresh at 50% of the amplitude
      P = thresh;                            // reset these for next time
      T = thresh;
    }

  if (N > 2500)
    {                          // if 2.5 seconds go by without a beat
      thresh = 512;                          // set thresh default
      P = 512;                               // set P default
      T = 512;                               // set T default
      lastBeatTime = sampleCounter;          // bring the lastBeatTime up to date
      firstBeat = true;                      // set these to avoid noise
      secondBeat = false;                    // when we get the heartbeat back
    }

  sei();                                    // enable interrupts when youre done!
}// end isr
```

圖 1-4　arduino 心跳感測器 triggered

以上除了讀取心跳值與判斷是否有效以外，也順便將讀值轉換為 BPM。

讀取酒精值由於不需要數值轉換，所以直接讀取就好，並且將酒精值傳入共用變數 Alcoholvalue 內。

```
void ReadAlcohol(){
Alcoholvalue= analogRead(AOUTpin);//reads the analaog value from the alcohol sensor's AOUT pin
```

圖 1-5　arduino MQ-3 感測器讀取

溫度感測器則是使用廠商內建的函式，會自動轉換為攝氏溫度，並將溫度傳入共用變數 temval 內。

```
void ReadTemperature(){
  sensors.requestTemperatures();      // 下指令要求取得溫度感測器的溫度值
  temval = sensors.getTempCByIndex(0);
  //Serial.print("Temperature for the device 1 (index 0) is: ");
  //Serial.println(temval);  // 取得並顯示溫度值
  }
```

圖 1-5　arduino 溫度感測器讀取

以上都完成後，必須判斷讀取的值是否有異常，這裡定義酒精值 300 以內為正常值，心跳介於 40~120 算正常。

```
      if(Alcoholvalue>=300)error_code++;
      if(BPM<40 ||  BPM>120)error_code+=2;
      switch (error_code) {
    case 0:
    error_str = ",D";//正常
        break;
    case 1:
    error_str = ",E1";
        break;
    case 2:
    error_str = ",E2";
        break;
    case 3:
    error_str = ",E3";
        break;
    default:
        break;
}
  /*
  真值表
  心跳  酒精  結果
  0     0       0
  0     1       1
  1     0       2
  1     1       3
  */
```

圖 1-5　arduino error code 定義

上述的 error code 可用布林代數表示，0 為正常，酒精異常為 1，心跳為 2，兩個都異常為 3，並且將 error code 寫入即將傳輸的的字串「error_str」內。

最後，將所有的值印出來，會形成一段有固定格式的字串。

```
      BT.print("D,"); //0
      BT.print(BPM);//1
      BT.print(",");
      BT.print(Alcoholvalue); //2
      BT.print(",");
      BT.print(temval);//3
      BT.println(error_str);//4
      delay(600); //  take a break
}


}
```

圖 1-6-1 arduino 印出資料程式碼

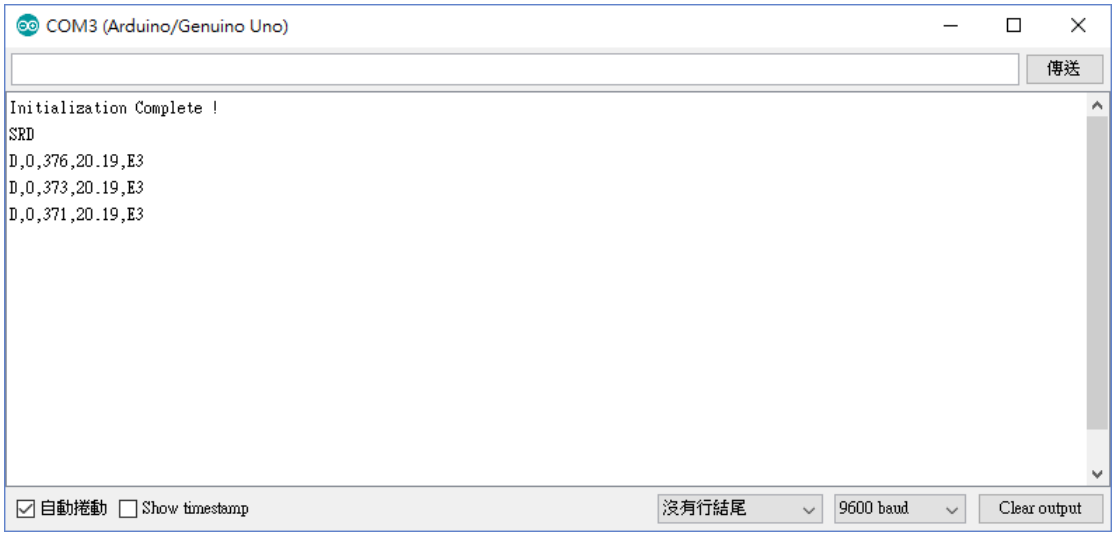


圖 1-6-1 arduino 資料格式

開頭 D,代表這是一串資料，讓 APP 辨別，接下來每筆資料用逗點隔開，分別為心跳、酒精，溫度值，最後為錯誤代碼，目前 E3 則代表心跳和酒精都異常。

2. Android APP

android app 有四個功能

(1) **藍芽傳輸**：目的是與 arduino 進行溝通，使手機可以傳遞指令與接收數據。
(2) **資料處理**：將 HC-06 傳送過來的字串進行分割，讓手機可以顯示當前的數據。
(3) **取得位置**：使用手機的 GPS 取得當前的經緯度。
(4) **資料回傳**：最後我們將收集到的數據，加上當前的經緯度，回傳給公司電腦。

APP 流程圖：

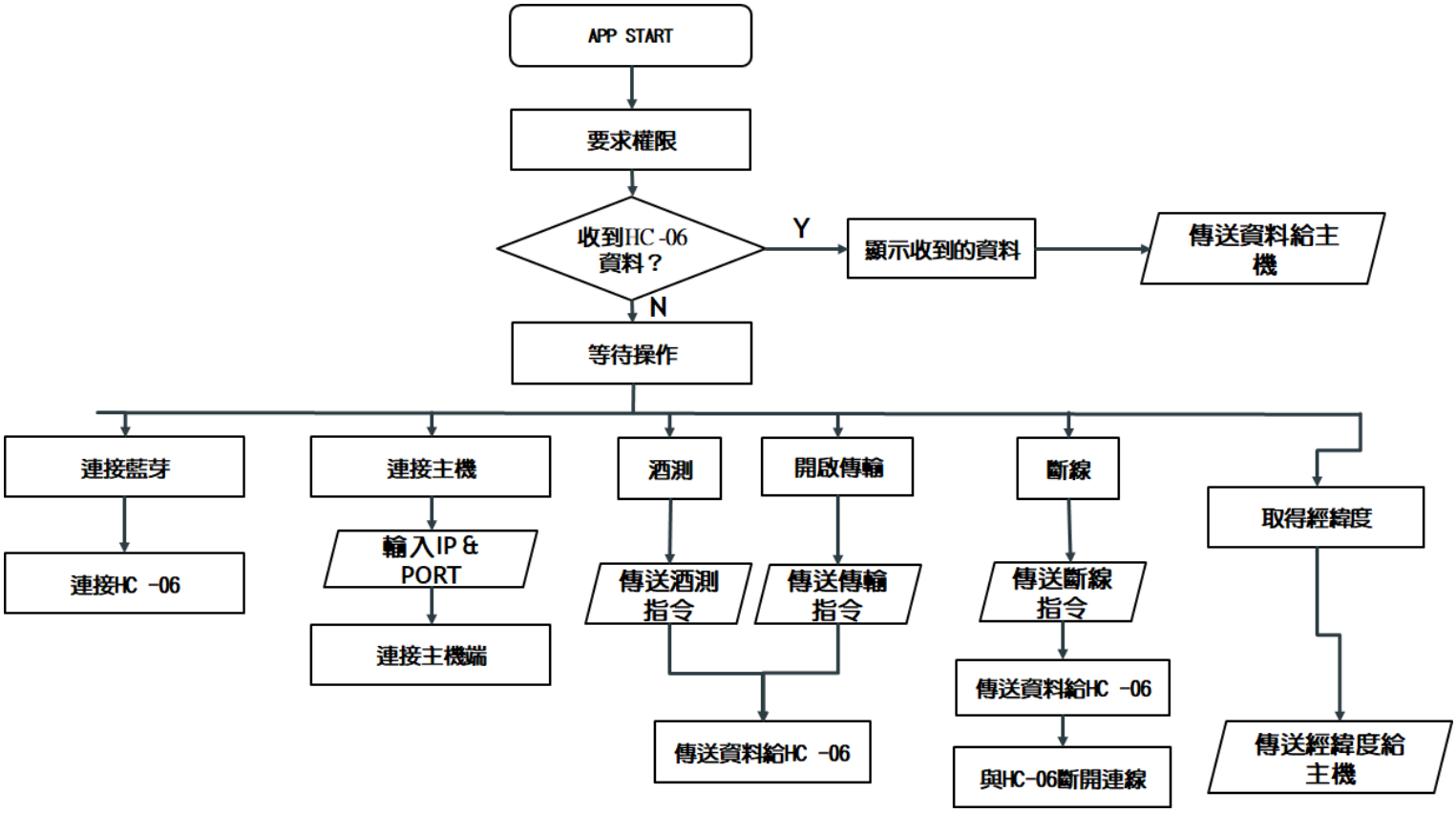


圖 2-1　APP 流程圖

由於 APP 程式碼過長，所以只放實際畫面，詳細程式已放在 GitHub，網址付在專題參考文獻中。

程式開始時，APP 會要求取得兩個手機權限，一個是開啟藍芽，另一個是要求取得位置。

圖 2-2　APP 要求權限

成功取得權限後，若 APP 判斷目前還未收到 HC-06 的資料，則會一直等待使用者操作。

功能：藍芽連接

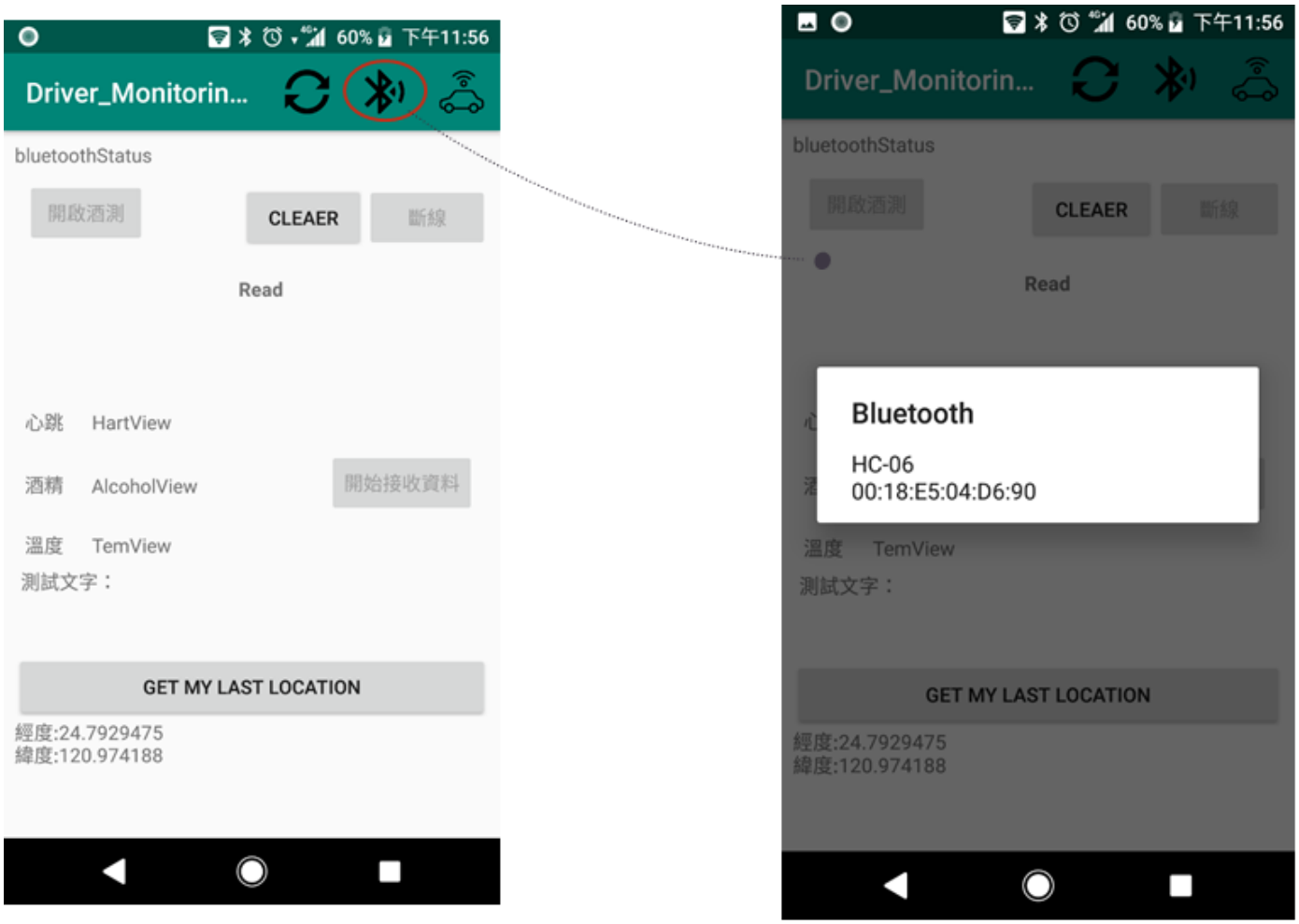


圖 2-3　APP 連接藍芽

按下 Menu 的按鈕，會列出最近取得的藍芽位置，點選後即可連線。

功能：主機端連接

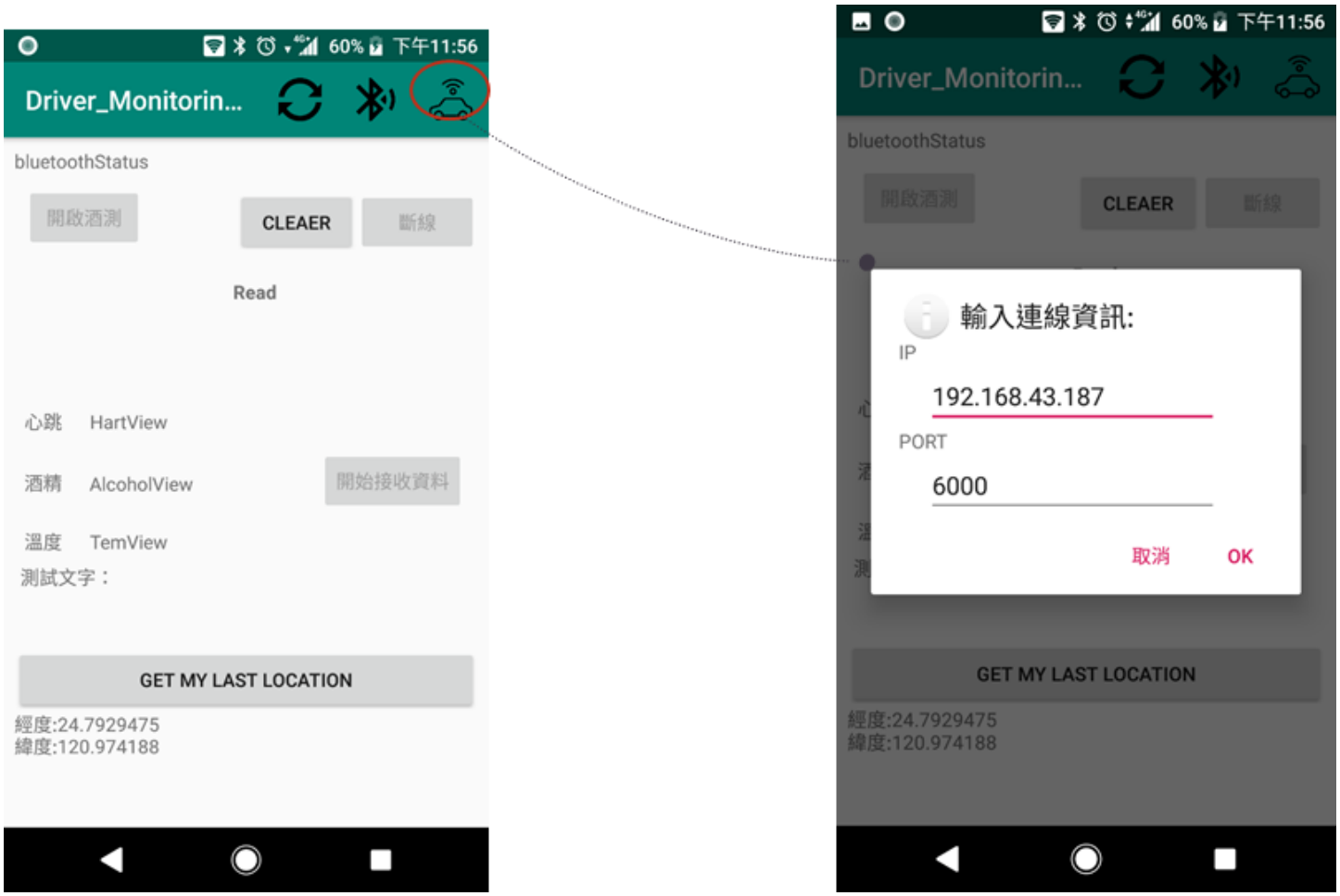


圖 2-4　APP 連接主機

連接主機端的功能需要輸入 IP 位置為 PORT 號碼。

輸入完成後會，會傳送一段文字給主機端，測試傳輸成功與否。

功能：其他

圖 2-5　APP 功能

簡易酒測功能對應到 Arduino 的酒測功能，主要就是傳輸指令「ALT」給 HC-06，令其執行酒測。

資料顯示與傳輸功能則為 SRD 與 SPD 指令，控制資料的傳輸。

資料顯示的功能為，將接收到的字串進行分割，並且呈現在手機上。

定位功能是使用手機 GPS，取得經緯度後傳送位置資訊給手機。

3. PC　　JAVA

此端的功能十分簡單，單純只是接手來自手機的資料，並且在資料異常時發出警訊。

程式流程圖：

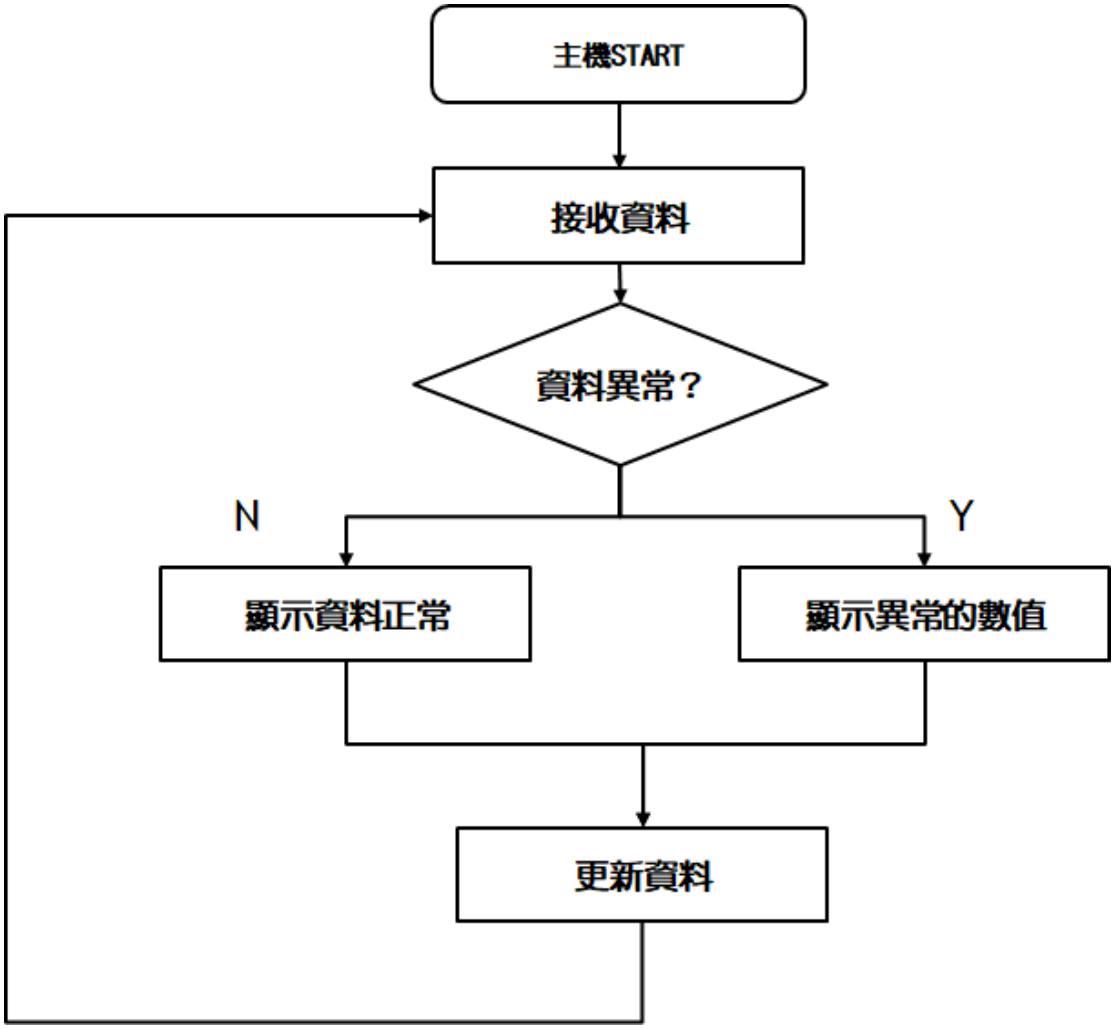


圖 3-1　主機程式流程圖

實際畫面：

程式一開始的畫面為：

圖 3-2　主機程式初始畫面

APP 端輸入完 IP 位置後，會傳送測試文字給 PC 端。

圖 3-3　主機程式連接成功畫面

當使用 APP 的定位功能時，主機端畫面如下：

這邊要注意，接收資料為 D, 開頭，接收位置為 L, 開頭，後兩位分別為經度與緯度。

圖 3-4　主機程式位址更新畫面

酒測功能：

當 APP 使用酒測功能，主機端會同步顯示酒測中。

圖 3-5　主機程式酒測畫面

最後也會將手機酒測的結果傳回主機端，時間也會隨之更新。

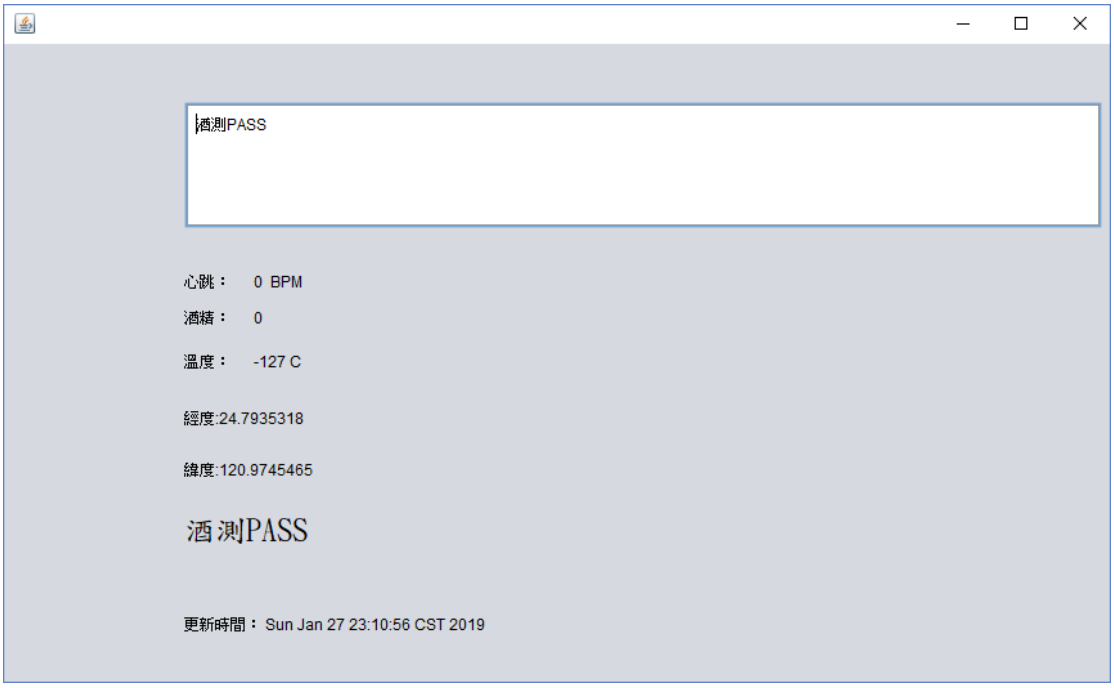


圖 3-6　主機程式酒測 PASS 畫面

接收資料的功能：

我們以容易操控的心跳值為例，當心跳值為 0，主機端會直接顯示位偵測到心跳。

圖 3-7　主機程式接收 0 心跳畫面

當所有資料都正常時，系統資料正常更新。

圖 3-8　主機程式接收正常畫面

若接收到的酒精值或心跳值異常，則會跳出異常警訊。

圖 3-9　主機程式接收異常資料畫面

斷開連線：

當手機與 Arduino 斷開連線時，也會送出訊息給主機端，讓主機端知道。

圖 3-10　主機程式斷開連接畫面

主機端程式碼和 APP 一樣放入 GitHub，付在末尾參考資料中。

第四章　討論與建議

1. 整體問題討論

當我們 Demo 成品時，我們發現幾個問題：

1. 手機端只負責接收與傳送資料，其實可以再加入警訊功能，當資料異常時就提醒司機，第一時間解決問題，會比公司端的人員與司機確認來的迅速。
2. 雖然想加入提醒司機的功能，但有組員指出，司機開車時可能會因為提醒而分心，更別提直接控制車輛靠邊停下了，可能會造成嚴重車禍，所以要如何做到提醒司機以及不打擾司機駕駛，是個難題。
3. 溫度感測器的存在有點多餘，本專題的初衷在於檢測司機生理，讓公司掌握駕駛員生理狀況，達到防止酒駕及疲勞駕駛，但後來又決定加入溫度感測器，因為公司可能也想掌握貨物運輸的情況，像是貨物所在位置、貨物的溫度濕度等等，不過最後也只做出了溫度感測器，而且並沒有具體的應用，只是檢視而已，這點有待改善。
4. 手機與主機端連線的方法是使用 TCP 連線，我們在測試同一個 WIFI 的情況下，傳輸資料都很正常，但在不同網域中卻無法順利進行，原因是我們沒有固定 IP 可以提供手機傳輸，後來也有想到幾個方法解決。一個是申請固定 IP 開網站收資料，但是要錢，另一個為用 Google 的雲端硬碟，然後讓主機端讀取裡面的資料。不夠後來因為時間與經費的問題，我們決定就使用現在的方法，以同一個網域傳輸，畢竟實際使用的時候，還是需要固定 IP 來收資料。
5. 我們只做出了一台貨車對應一台電腦，因為我們只有一套 Arduino 所以主機在接收時只接收到一組訊息，實際使用的時候應該是一台主機能夠接收很多貨車的訊息，由於經費原因，我們也沒有實做出來，如果這方面要實現，那又要加入司機編號以及車輛號碼，並且加入資料庫系統才能有效的管理。
6. 原本我們的成品應該包含一輛自走車，使用另一台手機來遙控，並且模擬司機駕駛的環境，不過後來組員未完成，所以後來又臨時改為跳警訊通知後台人員。

2. 結論與未來展望

以我們的概念，可減少大貨車因酒駕或疲勞駕駛而造成的傷亡，以保障用路人的人身安全，並減少公司業者因事故而產生的理賠問題。

也可讓公司提供目前貨物的狀態給消費者，使消費者對該業者有更高的信任度。

當然，我們無法保證駕駛有無鑽漏洞的行為，該如何解決這些問題，是我們的專題將來需要改善的地方。

而且以商業的角度而言，這並非是個好產品，很多功能只是能用而已，實際使用時還有很多要解決的問題。

像是藍芽會受到干擾，有時候傳輸指令會失敗，由於手機要同時進行許多功能，如果一次要處理的資料太大，有機率會當機，雖然這些問題在經歷程式作者好幾次的改版後降低不少，但也沒有大量去測試還會不會發生。

並且我們的專題是將市面上的感測器買來並做運用，實際使用的時候不可能時用如此笨重的商品，如果能將感測器縮小化、輕便化，成本相對降低，就能變得更容易套用在貨車上，甚至每一台車上。

未來也許能結合面部感測器，由 AI 判斷駕駛員的表情，加上即時的心跳數據來判斷司機的疲勞程度，加上自動駕駛的發展，相信能夠長足的改善道路交通以及人為疏失釀成的車禍。

專題完整程式碼

GITHUB:

https://github.com/jiared1231/Driver_Monitoring_System.git

Google 雲端：

https://drive.google.com/open?id=1-JVCIb0iMLRGphbxdNUhLR_JjzmCR6Gd

參考文獻

arduino 藍芽通信功能實作

https://home.gamer.com.tw/creationDetail.php?sn=3671289

Android 與 Arduino 的藍芽通訊

https://blog.xuite.net/chycahock/EDA/225365055-Android%E8%88%87Arduino%E7%9A%84%E8%97%8D%E8%8A%BD%E9%80%9A%E8%A8%8A

Arduino Breathalyzer Project using MQ3 alcohol sensor 0.96´ 128x64 OLED display on Arduino Mega

https://www.youtube.com/watch?v=6Pus5bs3SdY

Pulse Sensor interfacing with Arduino

https://www.youtube.com/watch?v=Gb0zZLC4Zql

藍芽模組設定修改

http://blog.ilc.edu.tw/blog/index.php?op=printView&articleId=650815&blogId=868

Send & Receive data from Android to PC (TCP Sockets) in Android Studio

https://www.youtube.com/watch?v=29y4X65ZUwE

實作 Activity 上方選單 Menu 與下拉項目功能

https://litotom.com/2017/07/31/ch7-4-menu/

https://tw.saowen.com/a/e6c18d242347762cdc822b914ad70bace05d4d6720e0e92e30bbead0f51637f6

Android ： 如何取得現在現在位置的經緯度

http://hsingjungchen.blogspot.com/2017/04/android.html