中華民國第 62 屆中小學科學展覽會

作品說明書

高級中等學校組　工程學(一)科

052312

血氧濃度與智慧醫療

學校名稱：臺中市私立曉明女子高級中學

作者：	指導老師：
高二 蕭宜絨 高二 陳亭諭 高二 蔡函恩	林家樂

關鍵詞：血氧濃度、生物感測、智慧醫療

摘　要

本研究從自組血氧濃度計開始，進行程式撰寫後成功讀取數據，20 次的累計平均誤差約為 7.6%。進而延伸改善誤差，以 6 通道感測器搭配移動平均計算，20 次的累計平均誤差降低為 4.3%，再以類神經網路技術參入 2%雜訊進行實驗，血氧值 90-99 的累計誤差平均降低至 0.46%，並研究儲存數據建立資料庫。

續而增添血壓數值研究，並以無線通訊技術即時偵測並傳送病患的生理數據，可以大幅縮短運送病患達醫療院所的醫療準備之時空差距，能改善醫療時空差距的品質，提高搶救病患的時效。期許本研究結果能激發國內各大醫學中心在智慧醫療的提升進展。

(晉級全國賽後更新：藍芽血壓計及 APP、網路通訊卡 SIM7600X 4G HAT)

壹、前言

一、動機

急診室的檢傷分類能夠讓病患在抵達醫院時做快速的篩檢分診，增進醫療的正確性與效率。如果在救護車上、病患家裡就能做到生命數據檢測，提前傳送生命數據給醫療單位，便能大幅度爭取搶救生命的關鍵時間。因此我們展開了相關的研究，希望能將生命的血氧濃度、心律、血壓進行檢測後儲存，進而比對資料產生生命健康訊息，即時傳送給相關醫療單位，以便能拯救更多的寶貴生命。

二、目的

我們希望能藉由自組血氧濃度感測器，整合核心控制器與程式設計，進行各項實驗，達到自組血氧檢測計，同時優化量測準確度、降低誤差，並將量測數據建立資料庫，撰寫資料比對程式，將比對結果建立為健康資訊，最後將健康資訊以通訊模組傳遞到醫療機構的雲端或是終端機，用以提供即時健康資訊，大幅降低醫療時差，提高醫療的時效與對生命的尊重。因此我們建立了以下幾個具體的研究步驟與實驗的項目作為研究設計的方針：

(一)實驗一：血氧濃度套件整合實驗

(二)實驗二：血氧六通道感測器實驗

(三)實驗三：血氧類神經網路實驗

(四)實驗四：生理數據建立資料庫實驗

(五)實驗五：生理數據傳輸應用實驗

三、文獻回顧

血氧濃度與智慧醫療的技術原理及現況如下：

(一)何謂血氧濃度？

血氧飽和度（Oxygen saturation），又稱血氧濃度，即血中氧飽和血紅蛋白佔總血紅蛋白的比例。人體正常動脈的血氧濃度為 95－100%、靜脈的則為 68－77%，低於 90%有可能是低氧血症。動脈血氧濃度低於 80%可能會損害器官功能，持續的低氧水平可能導致呼吸或心臟驟停。目前在臨床上大多使用脈搏的脈衝血氧來測量人體的血氧濃度。

(二)智慧醫療的內涵與現行概況

世界衛生組織（WHO）對「智慧醫療」(eHealth) 定義為[資通訊科技(ICT) 在醫療及健康領域的應用，包括醫療照護、疾病管理、公共衛生監測、教育和研究]。內容包含(1)優化就醫病患：以物聯網的概念進行雲端掛號、查看診間進度等，(2)提升醫院營運效益：所有醫療數據上傳，決策不再是靠傳統經驗，而是科學數據，(3)強化臨床醫療效益：以生理感測器持續收集病患的數值，可以提早警備、智能醫藥管理、提升醫護效率並降低負擔。

(三)生命數值與監康狀況的應用介紹

搭載在 Arduino 和 Raspberry Pi 平台上的電子健康傳感器 kit，eHealth 健康偵測管理系統，如圖 2 所示，通過該套件為研究人員提供一種簡單的方法，來測量生物識別傳感器數據，並為每個人開啟開源醫療產品的新時代。提供九種不同的生物特徵參數傳感器，包括：脈搏、血壓、血氧 (SPO_2)、心電圖 (EKG)、氣流、血糖儀、皮膚電反應 (GSR)、患者體位和體溫。

貳、研究設備及器材

本實驗所使用的主要設備列舉說明如下表 1：

表 1 研究使用的主要設備

名稱	照片	說明
1. Arduino 控制器 Nano 版		微控制器 ATmega328、5V、22 PIN、7g、18*45mm。
2. 血氧感測器		光學傳感器：紅外和紅光 LED 與光電探測器結合，測量脈衝血液吸收，I2C 接口加 INT 引腳、3.3V 電源。
3. LCD 顯示器		128*128 像素、1.5 in、OLED Display。
4. 血壓計 (更新)		量測偏高警報：135/85 mmHg、心律值、記憶 60 組*2 區，以藍芽傳輸資料，有專用手機 APP。
5.血氧濃度計		直流電源 2.6-3.6V、解析度 1%、血氧值 60-100%、心律值 40-199 次/分。

6. 6 通道光譜感測器		六段波長分別為 610 nm，680 nm，730 nm，760 nm，810 nm 和 860 nm。
7. ESP32 無線通訊板		具有 WiFi 及 Bluetooth 功能的開發板，具有兩組 HW UART。
8. SIM7600X 4G HAT		支援 4G、GPS，外接天線、擴充記憶卡。(更新)
9.杜邦線與電路繪圖軟體		分為公腳與母腳的端子，可以免撥線快速連接元件，另有電路繪圖軟體。
10. 其他		諸如麵包版、電烙鐵、傳輸線、手機、APP 應用軟體等等。

參、研究過程與方法

一、 研究步驟流程規劃

(一)、制定研究步驟及流程

經組員小組討論過後，規劃從購買 Arduino 主機板、血氧感測器等器材，組成血氧濃度計開始，依據研究設計與規劃訂定研究步驟流程方塊圖如圖 1 所示：

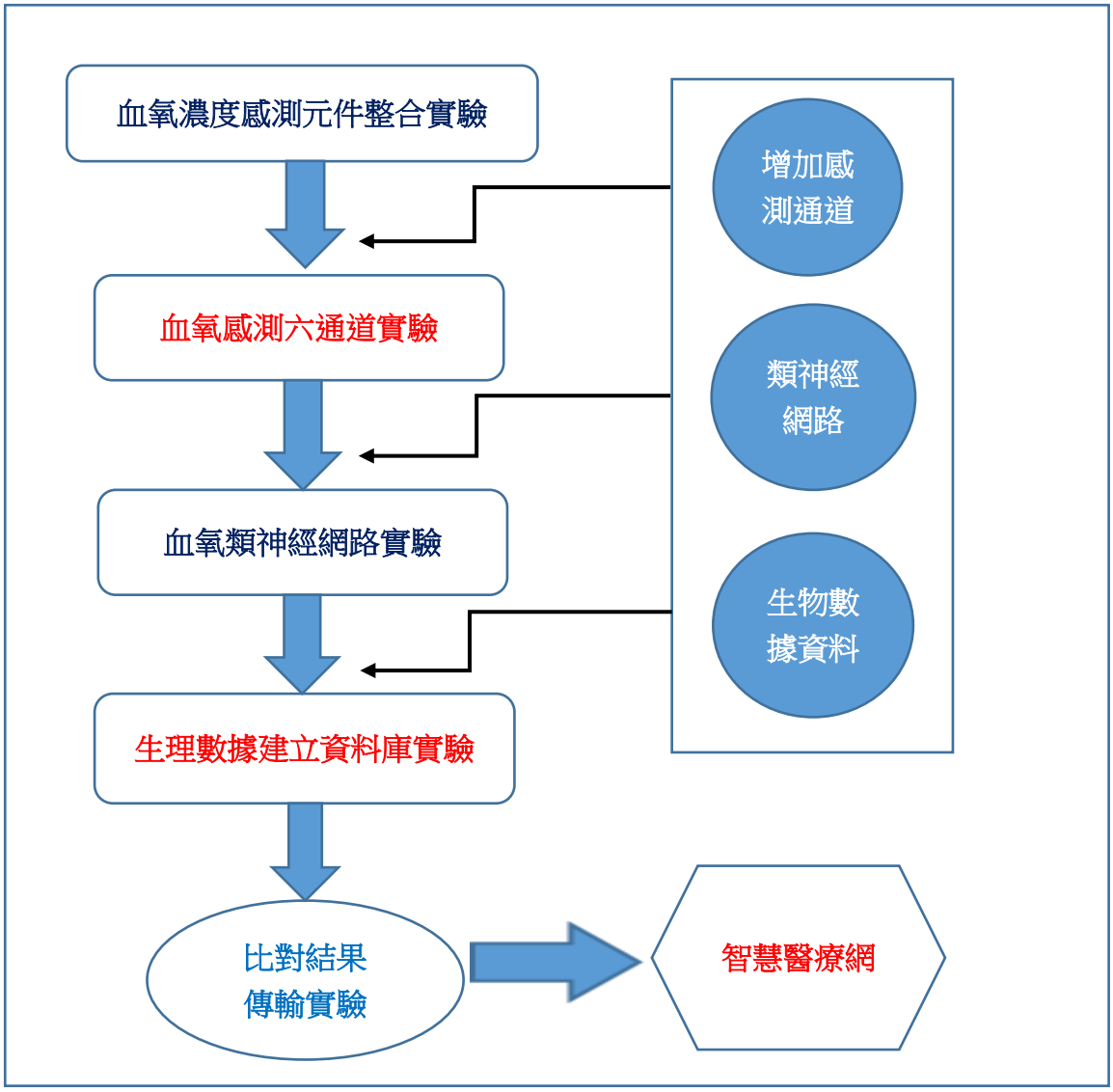


圖 1 研究流程架構圖

本研究主題最大的挑戰應該是撰寫程式的部分了，只有一位組員曾經學過 Python，所以我們積極安排了各種學習 Arduino 程式的機會，從網路上的自學到營隊課程，最後還安排了程式家教課。另外電子電路的專業知識也是一大難題，為了能看懂電子電路的元件連接圖，研究了許多電子電路連接的教學影片。

在今年的 5 月隊員到齊後開始陸續的研究，期間我們隊員除了每週共同的研究討論時間外，每個人依照分工合作的個人任務進行自主研究與學習。指導老師是一位組員的導師，所以能夠經常討論研究進度，以下我們用月份來擬訂進度表，如表 2 工作進度甘特表。

表 2 工作進度甘特表(全國賽更新)

110~111 年 項目	6 月	7 月	8 月	9 月	10 月	11 月	12 月	1 月	2 月	3 月	4 月	5 月	6 月
(1)資料收集-血氧濃度與智慧醫療的相關知識													
(2)實驗器具-自組血氧計與智慧醫療程式撰寫													
(3)實驗量測-進行生命數值偵測儲存與比對傳送													
(4)研究討論-數據整理與歸納討論													
(5)結論-綜合整理出結論													
(6)撰寫文稿-各章節分工撰文													
(7)校稿與修正-請老師指導與報告練習													

(二)、尋找資源

1.搜尋網路上有關血氧濃度的相關知識與介紹。

2.尋找智慧醫療的概念與介紹的文章與影片 。

3.瀏覽網路現有程式資料庫的研究來源。

4.與老師持續互動討論研究主題與方向。

5.上 Arduino 程式 ppt 討論並尋找大師學習研究。

二、實驗方法及先備知識

(一) Arduino 開源程式的學習與應用

Arduino 是製作開源硬體和開源軟體的公司，如圖 3 所示為微控制器 UNO 版，同時兼有專案和用戶社群，該公司負責設計和製造單板微控制器和微控制器套件，用於構建數位裝置和互動式物件，以便在物理和數位世界中感知和控制物件。該專案的產品是按照 GNU 寬通用公共許可證（LGPL）或 GNU 通用公共許可證（GPL）許可的開源硬體和軟體分發的，Arduino 允許任何人製造 Arduino 板和軟體分發。因此使用 Arduino 設計程式與功能開發可以大幅降低成本與時間，而且讓初學者也能快速上手運用程式與周邊配件，例如燈號、馬達 、時鐘、位移暫存器、繼電器、伺服馬達、蜂鳴器、光敏電阻等等。

圖 2　eHealth 健康偵測管理系統示意圖

圖 3 微控制器 UNO 版

(二)血氧濃度感測器

本研究專題使用的血氧濃度量測感測元件使用的是 MAX30102，如表 1 的第二項所示，該元件集成了完整的發光 LED 及其驅動部分，光感應和 AD 轉換部分，以及環境光干擾消除及數字濾波部分，極大地減輕了使用者的設計負擔。使用者需要使用單片機通過硬件 I2C 或者模擬 I2C 接口來讀取 MAX30102 元件的 FIFO，獲取轉換後的光強度數值，通過編寫相應的算法才可以得到心率值和血氧飽和度。

SaO_2 ：廣義上的氧飽和度，常指血液樣品中的氧含量對該樣品血液最大氧含量的百分比（SpO_2是經皮血氧飽和度，SaO_2是動脈血氧飽和度，相關性好，絕對值十分接近）。如圖 4 所示為含氧及未含氧血液在光波長 600-1000 nm 的光吸收強度圖，可以用來擷取適合呈現較為顯著突出血氧數值的光波長區段。HbO_2是氧合血紅蛋白，Hb 是未含氧的。

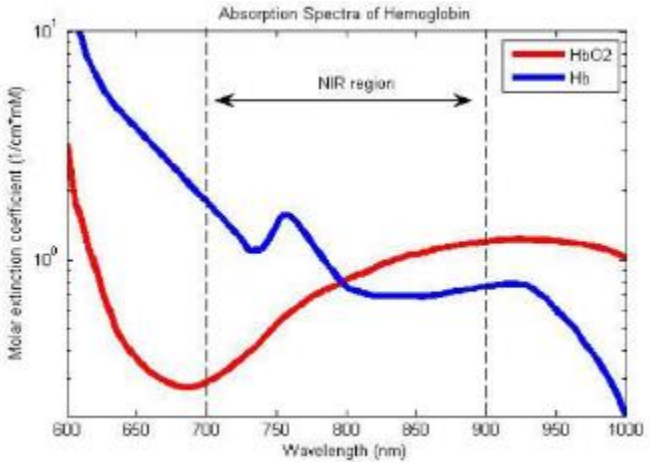


圖 4 含氧及未含氧血液在光波長 600-1000 nm 的光吸收強度圖

(三)脈搏的量測技術

傳統的脈搏測量方法有三種：1.是心電信號中提取(心電圖)，2.是從測量血壓時壓力傳感器測到的波動來計算脈率(血壓計)，3.是光容積法。前兩種方法提取信號都會限制病人的活動，如果長時間使用會增加病人生理和心理上的不舒適感。而光容積法脈搏測量作為監護測量中最普遍的測量方法之一，其具有方法簡單、佩戴方便、可靠性高等特點。光體積變化描記圖法 (光容積法 Photoplethysmography，簡稱 PPG)的基本原理是利用人體組織在血管搏動時造成透光率不同來進行脈搏和血氧飽和度測量的，其使用的傳感器由光源和光電轉換器兩部分組成。通過繃帶或夾子固定在病人的手指、手腕或耳垂。測血氧飽和度時，根據氧合血紅蛋白(Hb0₂)和血紅蛋白(Hb)對紅外光、紅外光的吸收量來計算。血管隨著心跳

舒張和收縮，舒張時血量多，吸收的紅光紅外光多，收縮時血量少，吸收的紅光紅外光少。根據反射到傳感器的光量周期性變化，可以計算出心率。

當一定波長的光束照射到指端皮膚表面，每次心跳時，血管的收縮和擴張都會影響光的透射 (例如在透射 PPG 中，通過指尖的光線) 或是光的反射 (例如在反射 PPG 中，來自手腕表面附近的光線)如圖 5 所示，當光照透過皮膚組織然後再反射到光敏感測器時，光照會有一定的衰減。像肌肉、骨骼、靜脈和其他連接組織對光的吸收是基本不變的 (前提是測量部位沒有大幅度的運動)，但是血液不同，由於動脈裡有血液的流動，那麼對光的吸收自然也有所變化。當我們把光轉換成電信號時，正是由於動脈對光的吸收有變化而其他組織對光的吸收基本不變，得到的信號就可以分為直流 DC 信號和交流 AC 信號。提取其中的 AC 信號，就能反應出血液流動的特點。如圖 5 所示為紅外光透過皮膚組織反射血液脈搏的示意圖。

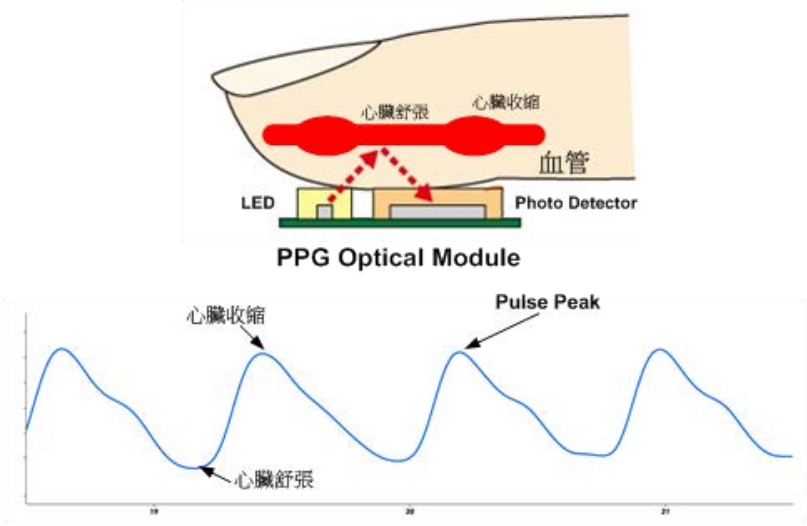


圖 5 紅外光透過皮膚組織反射血液脈搏的示意圖

肆、研究結果

一、實驗一：血氧濃度零件整合實驗

本實驗為購置血氧濃度感測零件自行組裝，並撰寫程式作為實驗組，另購置一台市售夾指型的血氧濃度計，如研究設備及器材表之第五項所示，作為標準的對照組，以雙波段（紅光 660 nm／紅外光 890 nm）光源激發，與自組的紅外光波段相近。實驗目的為建立一個可以自行撰寫程式，進行程式參數語法調整的實驗平台，作為後續實驗的起始點。實驗前準備好購置的材料如圖 6 所示，包含微控制器、麵包版、血氧感測器、LCD 顯示器、杜邦線等等。自組血氧濃度計的接線圖如圖 7 所示，圖 8 為組員組裝接線中，自組血氧濃度計接線完成圖如圖 9 所示，如圖 10 所示為量測血氧及心率讀值完成顯示於 LCD。將量測 20 次的數值紀錄於表 3 當中，並繪製成自組血氧濃度計與對照組的數值曲線對照圖，如圖 11 所示，自組血氧濃度計的程式截圖如圖 12 所示。

圖 6 自組血氧濃度計元件材料

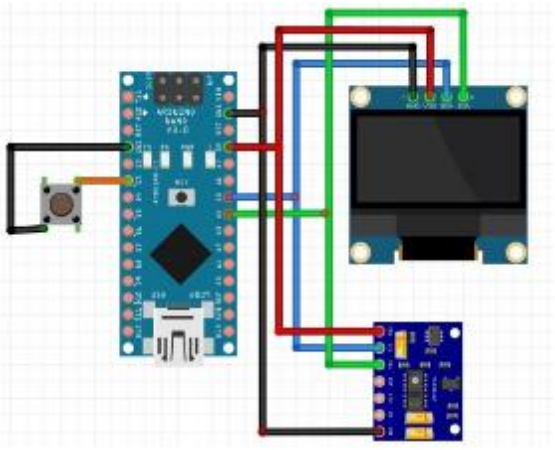


圖 7 自組血氧濃度計的接線圖

圖 8 自組血氧濃度計接線實驗

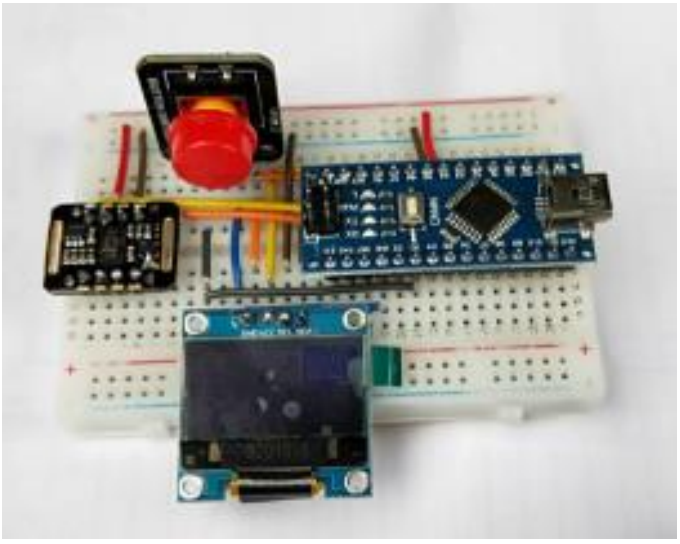


圖 9 自組血氧濃度計接線完成圖

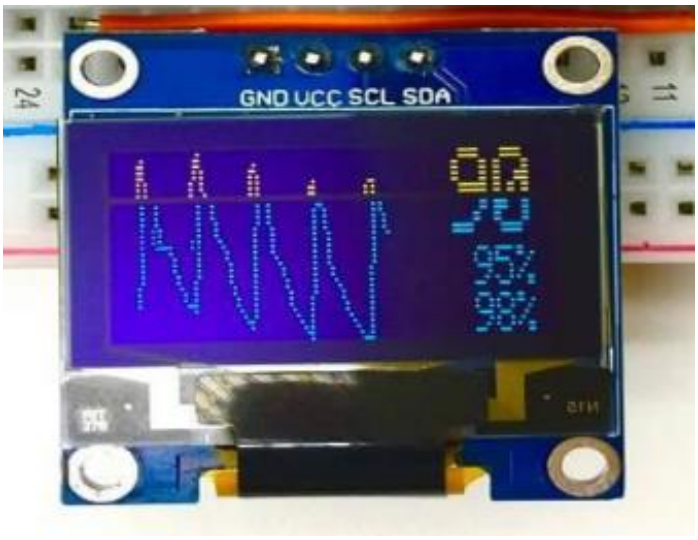


圖 10 量測血氧及心率讀值完成顯示於 LCD

表 3 自組血氧濃度計 20 組量測值紀錄表

次數	對照組	實驗組	差異值	差異百分比
1	98	68	30	30.6%
2	98	72	26	26.5%
3	96	78	18	18.8%
4	97	84	13	13.4%
5	97	92	5	5.2%
6	96	95	1	1.0%
7	98	96	2	2.0%
8	99	97	2	2.0%
9	97	98	-1	-1.0%
10	96	98	-2	-2.1%
11	95	99	-4	-4.2%
12	96	98	-2	-2.1%
13	96	96	0	0.0%
14	97	95	2	2.1%
15	98	96	2	2.0%
16	98	92	6	6.1%
17	98	90	8	8.2%
18	96	88	8	8.3%
19	96	87	9	9.4%
20	97	86	11	11.3%
累計平均誤差			152	7.6%

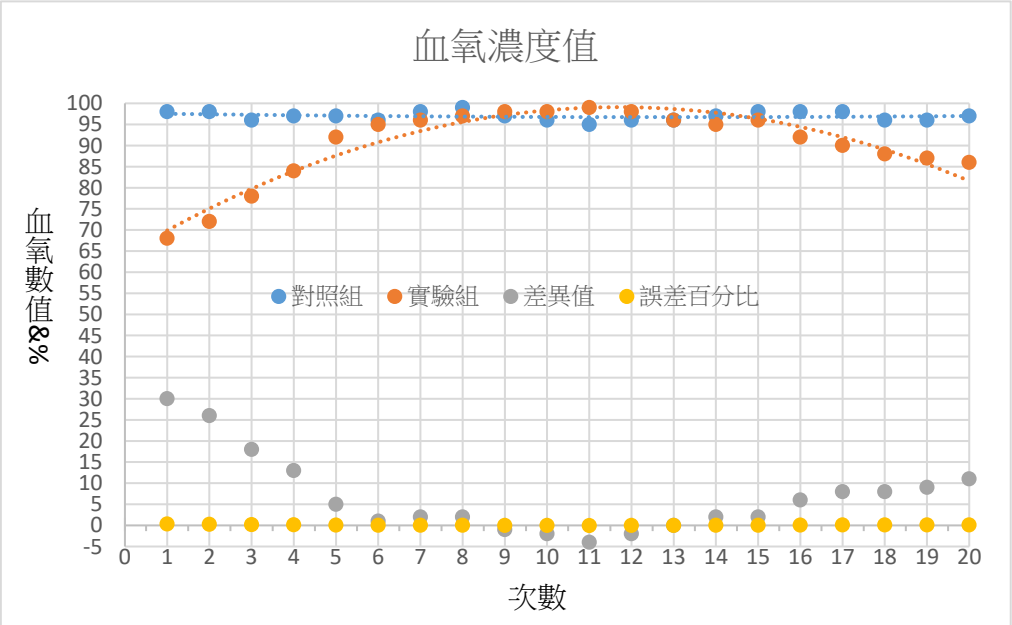


圖 11 自組血氧濃度計與對照組的數值曲線對照圖

```
Code：
#include <Adafruit_GFX.h>              //OLED libraries
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include "MAX30105.h"                  //MAX3010x library
#include "heartRate.h"                 //Heart rate calculating algorithm
#include "ESP32Servo.h"
#include <BluetoothSerial.h>
BluetoothSerial BT;


MAX30105 particleSensor;
int Tonepin = 4;
//計算心跳用變數
const byte RATE_SIZE = 10; //多少平均數量
byte rates[RATE_SIZE]; //心跳陣列
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred
float beatsPerMinute;
int beatAvg;

//計算血氧用變數
double avered = 0;
double aveir = 0;
double sumirrms = 0;
double sumredrms = 0;

double SpO2 = 0;
double ESpO2 = 90.0;//初始值
double FSpO2 = 0.7; //filter factor for estimated SpO2
double frate = 0.95; //low pass filter for IR/red LED value to eliminate AC component
int i = 0;
int Num = 30;//取樣 100 次才計算 1 次
#define FINGER_ON 7000 //紅外線最小量（判斷手指有沒有上）
#define MINIMUM_SPO2 90.0//血氧最小量

//OLED 設定
#define SCREEN_WIDTH 128 //OLED 寬度
#define SCREEN_HEIGHT 64 //OLED 高度
#define OLED_RESET     -1 //Reset pin
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET); //Declaring the display name (display)

//心跳小圖
static const unsigned char PROGMEM logo2_bmp[] =
{ 0x03, 0xC0, 0xF0, 0x06, 0x71, 0x8C, 0x0C, 0x1B, 0x06, 0x18, 0x0E, 0x02, 0x10
0x0C, 0x03, 0x10,              //Logo2 and Logo3 are two bmp pictures that
display on the OLED if called
  0x04, 0x01, 0x10, 0x04, 0x01, 0x10, 0x40, 0x01, 0x10, 0x40, 0x01, 0x10, 0xC0,
0x03, 0x08, 0x88,
  0x02, 0x08, 0xB8, 0x04, 0xFF, 0x37, 0xC8, 0x01, 0x30, 0x18, 0x01, 0x90, 0x30,
0x00, 0xC0, 0x60,
  0x20, 0x60, 0xC0, 0x30, 0x31, 0x80, 0x00, 0x1B, 0x00, 0x00, 0x0E, 0x00, 0x00,
0x04, 0x00,
};
//心跳大圖
static const unsigned char PROGMEM logo3_bmp[] =
{ 0x01, 0xF0, 0x0F, 0x80, 0x06, 0x1C, 0x38, 0x60, 0x18, 0x06, 0x60, 0x18, 0x10,
0x01, 0x80, 0x08,
  0x20, 0x01, 0x80, 0x04, 0x40, 0x00, 0x20, 0x02, 0x40, 0x20, 0x00, 0x02, 0x20,
0x00, 0x08, 0x03,
  0x80, 0x00, 0x08, 0x21, 0x80, 0x00, 0x18, 0x01, 0x80, 0x20, 0x1C, 0x01, 0x80,
0x00, 0x14, 0x00,
  0x80, 0x00, 0x14, 0x20, 0x80, 0x00, 0x14, 0x00, 0x40, 0x10, 0x12, 0x00, 0x40,
0x10, 0x12, 0x00,
  0x7E, 0x1F, 0x23, 0xFE, 0x03, 0x31, 0xA0, 0x04, 0x01, 0xA0, 0xA0, 0x0C, 0x00
0xA0, 0xA0, 0x08,
  0x20, 0x60, 0xE0, 0x10, 0x00, 0x20, 0x60, 0x20, 0x06, 0x00, 0x40, 0x60, 0x03,
0x00, 0x40, 0xC0,
  0x01, 0x80, 0x01, 0x80, 0x00, 0xC0, 0x03, 0x00, 0x00, 0x60, 0x06, 0x00, 0x00,
0x30, 0x0C, 0x00,
  0x20, 0x08, 0x10, 0x20, 0x00, 0x06, 0x50, 0x00, 0x00, 0x03, 0xC0, 0x00, 0x00,
0x01, 0x80, 0x00
};
//氧氣圖示
static const unsigned char PROGMEM O2_bmp[] = {


  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x3f, 0xc3, 0xf8, 0x00,
0xff, 0xf9, 0xfc,
  0x03, 0xff, 0xff, 0xfe, 0x07, 0xff, 0xff, 0xfe, 0x0f, 0xff, 0xff, 0xfe, 0x0f, 0xff, 0xff,
0x7e,
  0x1f, 0x80, 0xff, 0xfc, 0x1f, 0x00, 0x1f, 0xb8, 0x3e, 0x3e, 0x3f, 0xb0, 0x3e, 0x3f,
0x3f, 0xc0,
  0x3e, 0x3f, 0x1f, 0xc0, 0x3e, 0x3f, 0x1f, 0xc0, 0x3e, 0x3f, 0x1f, 0xc0, 0x3e, 0x3e,
0x2f, 0xc0,
  0x3e, 0x3f, 0x0f, 0x80, 0x1f, 0x1c, 0x2f, 0x80, 0x1f, 0x80, 0xcf, 0x80, 0x1f, 0xe3,
0x9f, 0x00,
  0x0f, 0xff, 0x3f, 0x00, 0x07, 0xfc, 0xfc, 0x00, 0x0b, 0xfc, 0x3c, 0x00, 0x1d, 0xff,
0xf8, 0x00,
  0x1e, 0xff, 0xe0, 0x00, 0x1f, 0xff, 0x00, 0x00, 0x1f, 0xf0, 0x00, 0x00, 0x1f, 0xe0,
0x00, 0x00,
  0x0f, 0xe0, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00
};
void setup() {
  Serial.begin(230400);
  Serial.println("System Start");
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //Start the OLED display
  display.display();
  delay(3000);
  //檢查
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port,
400kHz speed
  {
    Serial.println("找不到 MAX30102");
    while (1);
  }
  byte ledBrightness = 0x1F; //亮度 Options: 0=Off to 255=50mA
  byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
  byte ledMode = 2; //Options: 1 = Red only(心跳), 2 = Red + IR(血氧)
  //Options: 1 = IR only, 2 = Red + IR on MH-ET LIVE MAX30102 board
  int sampleRate = 800; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
  int pulseWidth = 215; //Options: 69, 118, 215, 411
  int adcRange = 16384; //Options: 2048, 4096, 8192, 16384

  //顯示血氧數值
  if (beatAvg > 30) display.print(String(ESpO2) + "%");
  else display.print("--- %");
  display.display();//顯示螢幕
  tone(Tonepin, 1000);//發出聲音
  delay(10);
  noTone(Tonepin);//停止聲音
  Serial.print("beatAvg="); Serial.println(beatAvg);//將心跳顯示到序列
  BT.println("value of pulse is");BT.println(beatAvg);BT.println("BPM");
  long delta = millis() - lastBeat;//計算心跳差
  lastBeat = millis();
  beatsPerMinute = 60 / (delta / 1000.0);//計算平均心跳
  if (beatsPerMinute < 255 && beatsPerMinute > 20) {
    //心跳必須為 20-255 之間
    rates[rateSpot++] = (byte)beatsPerMinute; //儲存心跳數值陣列
    rateSpot %= RATE_SIZE;
    beatAvg = 0;//計算平均值
    for (byte x = 0 ; x < RATE_SIZE ; x++) beatAvg += rates[x];
    beatAvg /= RATE_SIZE;
  }
                    };
}

  //計算血氧
  uint32_t ir, red ;
  double fred, fir;
  particleSensor.check(); //Check the sensor, read up to 3 samples
  if (particleSensor.available()) {
    i++;
    red = particleSensor.getFIFOIR(); //讀取紅光
    ir = particleSensor.getFIFORed(); //讀取紅外線
    //Serial.println("red=" + String(red) + ",IR=" + String(ir) + ",i=" + String(i));
    fred = (double)red;//轉 double
    fir = (double)ir;//轉 double
    avered = avered * frate + (double)red * (1.0 - frate);//average red level by low
pass filter
    aveir = aveir * frate + (double)ir * (1.0 - frate); //average IR level by low pass
filter
```

圖 12 自組血氧濃度計的程式截圖

二、實驗二：6 通道感測器實驗

在實驗一中的實驗組數值呈現爬升後下降的趨勢，我們添購了一塊 6 通道的 AS7263 感測器如使用的設備及器材表中的第 6 項所示，希望能藉由多通道的交叉補償修正後的數值，進行移動平均值的計算，可以降低誤差、增進準確度。AS7263 擷取的六段波長，分別為 610 nm，680 nm，730 nm，760 nm，810 nm 和 860 nm，其半峰全寬(FWHM)為 20 nm 如圖 13 所示，傳感器將獲得的信號通過杜邦線傳輸到 Arduino Nano 控制器，同時考慮到光感測器的積分時間與邊緣運算裝置的計算時間，我們設定光傳感器的讀取時間間隔為 0.7 秒。將量測 20 次的數值紀錄於表 4 當中，並繪製成 6 通道血氧濃度計與對照組的數值曲線對照圖，如圖 14 所示。

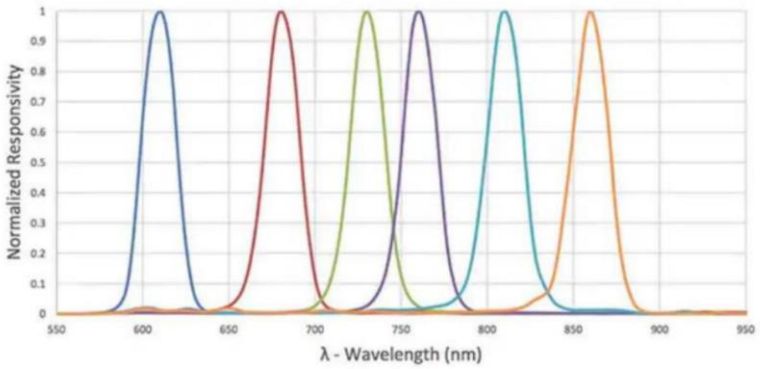


圖 13　6 通道的 AS7263 感測器的六個波長頻段

表 4　6 通道血氧濃度計的實驗數值

次數	對照組	實驗組	差異值	差異百分比
1	96	85	11	11.5%
2	97	89	8	8.2%
3	98	92	6	6.1%
4	98	93	5	5.1%
5	97	95	2	2.1%
6	96	96	0	0.0%
7	98	97	1	1.0%
8	98	97	1	1.0%
9	97	97	0	0.0%
10	96	96	0	0.0%
11	95	94	1	1.1%
12	96	95	1	1.0%
13	96	95	1	1.0%
14	95	95	0	0.0%
15	97	96	1	1.0%
16	97	96	1	1.0%
17	96	95	1	1.0%
18	96	95	1	1.0%
19	98	95	3	3.1%
20	97	95	2	2.1%
累計平均誤差			46	2.3%

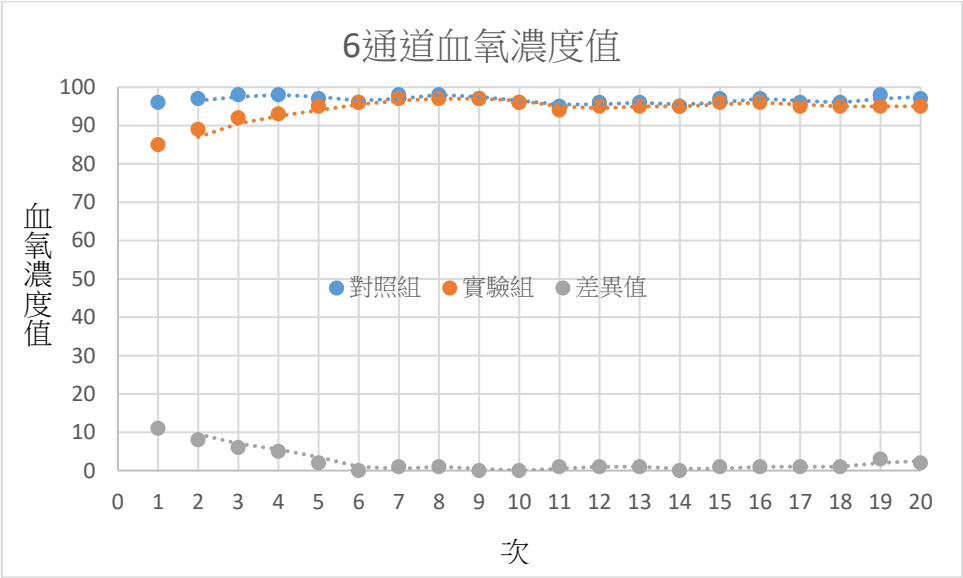


圖 14　6 通道血氧濃度計與對照組的數值曲線對照圖

實驗三：類神經網路實驗

類神經網路是一種用於資料叢集的收集與條件設定修改，進而分析何種條件設定的資料處理可以獲得較佳的模式，因為條件交連有點像神經網路，因此稱之為類神經網路如圖 15 所示為 3 維的，我們使用的是一維的。事實上是一種資料處理的分析法，我們希望加入這種分析法繼續來改善實驗二的誤差值。對一個成功的神經網路模型而言，具有合適且高品質的訓練資料是必要的，在資料集的選用方面，由於 HbO_2 與 Hb 在可見光與紅外光的波段所表現出不同的吸收特性，我們將採用包含不同區域的波段與光譜數量並套用做訓練。一維卷積神經網路(1D-CNN)相較於深度神經網路的全連接特性，透過權值共享的方式對個別區域(特定神經元)先進一步擷取輸入資料的有效特徵，再透過全連接層來解決分類問題，這樣將能夠在同時減少運算規模，不降低模型的準確性。式 1.1 為 1D-CNN 運算的數學表達式，其中 Hk 為卷積核的長度，Uk 為序列數據的長度，k 為掃描單個數據所需的步數。此外，hk-i 是卷積核的第 k-i 行，ui 是序列數據的第 i 行。另外，1D-CNN 的卷積核寬度是固定的，不需要指定寬度。因此，1D-CNN 的輸入形狀是一個 3D 張量，依次是批量大小、序列資料長度和輸入資料的深度。

$$Y(k)= H_k U_k = \sum_{i=0}^{N} h_{k-i}\, u_i \qquad \text{(式 1.1)}$$

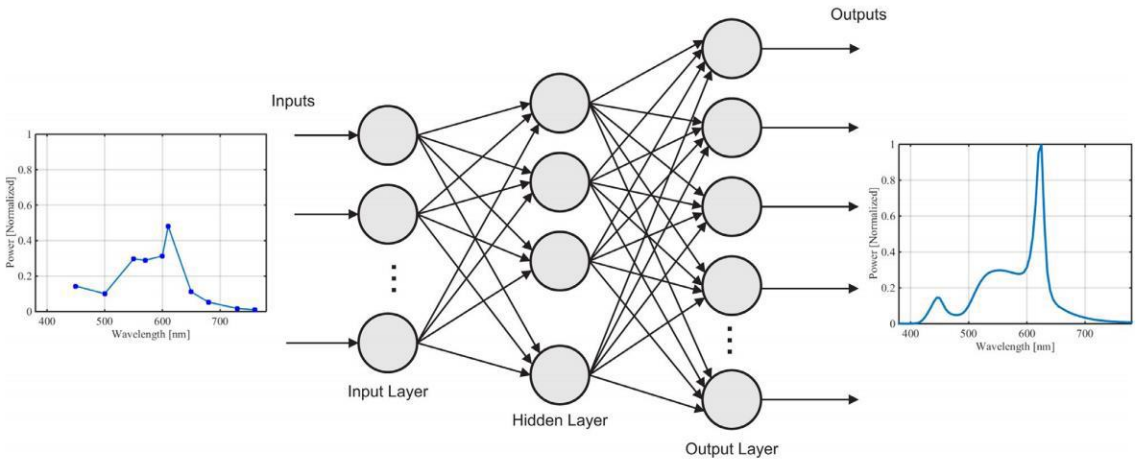


圖 15 藉由類神經網路技術改善數值得到最佳的分析模式

資料蒐集階段，為了取得可靠且有效的數據，我們右手使用 Pulse Oximeter SPO_2血氧分析儀，左手使用量測裝置同時進行，為了模擬缺氧當下的 SpO_2狀況，我們透過憋氣的方式來達成，以取得相同時間下的光譜與血氧濃度。透過量測裝置將各波長量測的光強度與對應時間記錄在 Excel 中以方便之後標籤化階段有所對應，如此才能將各筆資料準確的標籤上 SpO_2。而在 SpO_2標籤紀錄方面，將血氧分析儀的數值記錄下來並透過記錄時間來確認各時間點所對應的光譜值，最終我們取得了 SpO_2從 81%到 99%的血氧光譜如圖 16 所示。

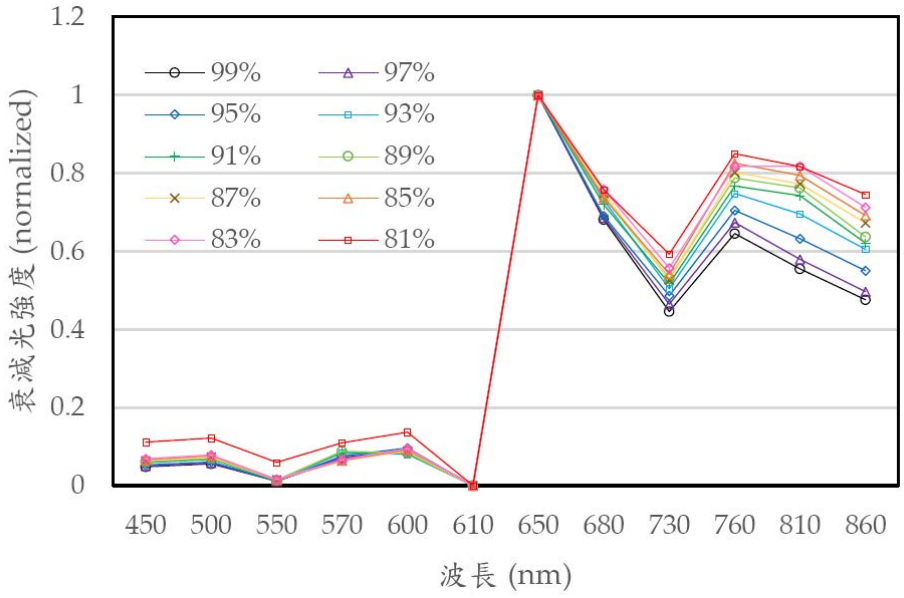


圖 16 SpO_2從 81%到 99%的血氧光譜圖

我們針對每組添加雜訊以增加訓練，藉此觀察神經網路對雜訊的免疫力，即泛化能力將得到極大的提升。在圖 17 中顯示，當雜訊添加率增加時，模型精度得到提升。但是，準確率 2%後下降，因此在此量測模型中，必須添加大於 2%的隨機雜訊才能顯著提升準確率。從圖

17 可以看出，2%的雜訊添加將對我們優化的模型產生最大的好處。綜上所述，充分了解訓練數據的特點，適當添加雜訊有助於提高神經網絡的抗干擾能力。

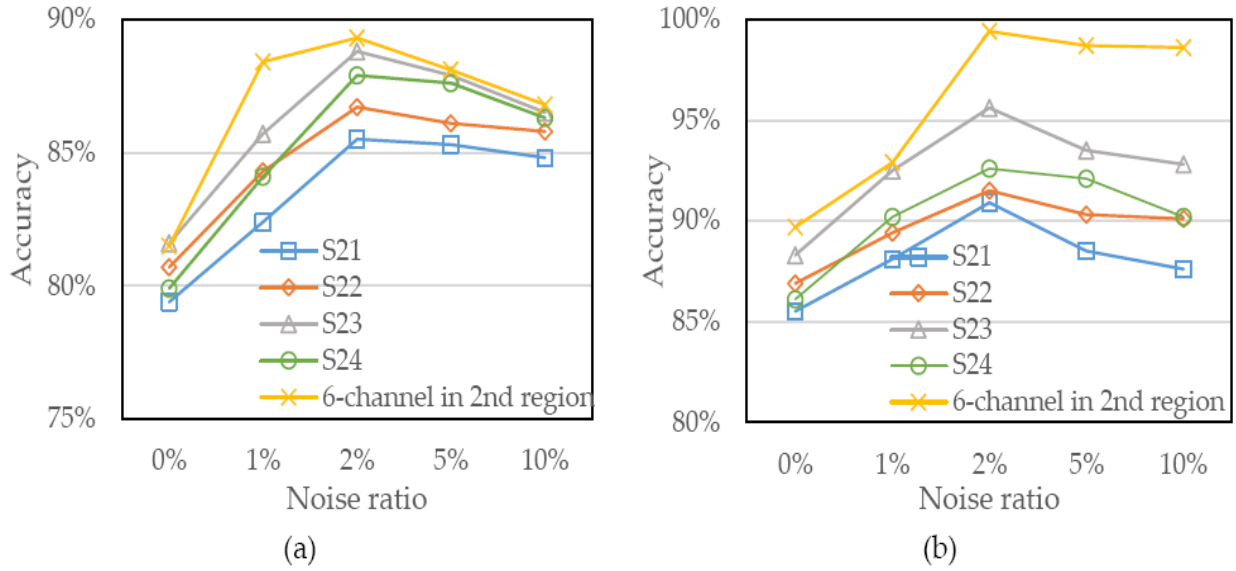


圖 17　2%的雜訊添加對於準確度的影響曲線圖

選擇具有 2%雜訊的第二區域數據集訓練的模型，顯示了對模型在 SpO_2 從 99%到 90%的量測能力的更深入討論和評估，以及最大誤差、平均誤差和標準差，便於我們清楚地了解每個濃度的準確性與穩定性，將數值製作成表 5。如表 5 所示，經過類神經網路加上 2%雜訊的訓練下，差異值已經可以調整降低到 0.46%。

表 5 經過類神經網路加上 2%雜訊的訓練下的差異值

SpO_2 (%)	Maximum deviation(%)	Average deviation ± Standard deviation(%)
99	0.41	0.31 ±0.18
98	0.77	0.46 ±0.38
97	0.54	0.35 ±0.27
96	1.08	0.45 ±0.58
95	0.79	0.44 ±0.36
94	1.14	0.56 ±0.39
93	0.61	0.43 ±0.22
92	1.51	0.58 ±0.67
91	0.72	0.41 ±0.34
90	1.43	0.61 ±0.51
Total error : 0.46%		

實驗四：生物數據資料庫實驗

本實驗嘗試撰寫不同的程式語法以及結構，討論驗證所建立數據資料庫的樣態，哪一種較符合智慧醫療所需的形式。第一種是 Arduino 收集數據並分析，程式架構節錄如圖 18 所示，所得的資料在 1000 比之後會產生覆蓋的情形，如圖 19 所示。

```
void loop(void)
{
  // 要求匯流排上的所有感測器進行溫度轉換
  sensors.requestTemperatures();

  // 參數 0 代表匯流排上 第 0 個 1-Wire 裝置

  Serial.print("DATA,TIME");
  Serial.print(",");
  Serial.print("TIMER");
  Serial.print(",");
  Serial.print(sensors.getTempCByIndex(0));
  Serial.print(",");
  Serial.println(",");

  row ++;

  //超過 1000 行資料後就覆蓋舊資料
  /*
  if (row >= 1000)
  {
    row=0;
    Serial.println("ROW,SET,2");
  }
  */
```

圖 18 第一種是 Arduino 收集數據並分析的程式架構節錄

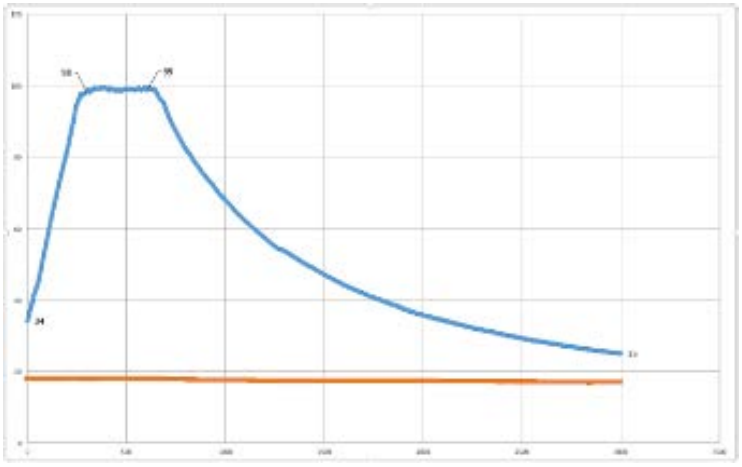


圖 19 第一種是 Arduino 收集數據並分析的程式資料紀錄曲線

第二種是從 Arduino 連線到資料庫，程式架構節錄如圖 20 所示，資料庫位於連線電腦中的試算表資料夾。

```
#include <Ethernet.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>

byte mac_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

IPAddress server_addr(x,x,x,x);  //the IP address to that of the system on which m
MySQL server is set up
char user[] = "bob";              // MySQL user login username
char password[] = "secret";       // MySQL user login password


// Sample query
char query[] = "SELECT id FROM world.city";

EthernetClient client;
MySQL_Connection conn((Client *)&client);

void setup() {
  Serial.begin(9600);
  while (!Serial); // wait for serial port to connect
  Ethernet.begin(mac_addr);

  //while(!conn.connect(server_addr, 3306, user, password));

  Serial.println("Connecting...");
  if (conn.connect(server_addr, 3306, user, password)) {
    delay(1000);
```

圖 20 第二種是從 Arduino 連線到資料庫的程式節錄

第三種是 Arduino 使用 PMSA003 讀取數值，並使用 ESP8266 透過 PHP 儲存資料到資料庫，程式架構節錄如圖 21 所示，資料庫位於連線電腦中的試算表資料夾。程式為每 10 秒讀取 DHT11 一次，上傳資料庫，完成上傳後就立即斷線資料庫，等候下一個 Loop。

```
void setup() {
  Serial.begin(9600);
  Serial.println("PMSA003 Detect PM2.5");
  Serial2.begin(9600);              //啟始 Arduino Mega 硬體串列埠 Pin16 與
Pin17 (PMSA003)
  Serial1.begin(9600);              //啟始 Arduino Mega 硬體串列埠 Pin18 與
Pin19 ( ESP8266)
  Serial1.println("AT+RST");        //傳送 AT 指令起始 ESP8266
}

void loop() {
  unsigned char chrData[30];
  unsigned int pm1,pm25,pm10;
  if (Serial2.available())   //PMSA003
  {
    chrRecv = Serial2.read();
    if (chrRecv == START_1 && bytCount == 0)   bytCount = 1;
    if (chrRecv == START_2 && bytCount == 1)   bytCount = 2;
    if (bytCount == 2)
    {
      bytCount = 0;
      for(int i = 0; i < 30; i++) {
        chrData[i] = Serial2.read();
      }
      for(int i = 0; i < 30; i++){
        chrData[i] = Serial2.read();
```

PHP 程式

```
<?php
//http://127.0.0.1:808/pm.php?pm1=2&pm25=3&pm10=20&temp=25&humi=50
DEFINE ('DBServer', '127.0.0.1');
DEFINE ('DBName', 'pm');
DEFINE ('DBUser', 'root');
DEFINE ('DBPw', '');

$conDb = mysqli_connect(DBServer,DBUser,DBPw);
if (!$conDb) {
    die("Can not connect to DB: " . mysqli_error($conDb));
    exit();
}

$selectDb = mysqli_select_db($conDb, DBName);
if (!$selectDb) {
    die("Database selection failed: " . mysqli_error($conDb));
    exit();
}

$temp = mysqli_real_escape_string($conDb, $_GET['temp']);
$humi = mysqli_real_escape_string($conDb,$_GET['humi']);
$pm1 = mysqli_real_escape_string($conDb,$_GET['pm1']);
$pm25 = mysqli_real_escape_string($conDb,$_GET['pm25']);
$pm10 = mysqli_real_escape_string($conDb,$_GET['pm10']);
$datetime = date_create()->format('Y-m-d H:i:s');

$query = "INSERT INTO pm (pm1, pm25, pm10, temp, humi, datetime) VALUES
('$pm1', '$pm25', '$pm10', '$temp', '$humi', '$datetime')";
```

圖 21　第三種是 Arduino 使用 PMSA003 讀取數並使用 ESP8266 透過 PHP 儲存資料到資料庫，程式架構節錄圖

第四種為無線監控數據，Arduino 配合 raspberry pi，Arduino 和 ESP8266 連線當作 SERVER 端，並讓擔任 CLIENT 端的 raspberry pi 來進行連接，並將 DHT11 收集到的數據由 SERVER 傳送到 CLIENT 端，最後則是上傳到資料庫進行資料保存，方便日後要需要用到資料時可以隨時查看，程式架構節錄如圖 22 所示。

```
//ESP8266 連接無線網路
#define Name "111"//私人網路名稱： "111"
#define Pass "11111111"//網路密碼： "11111111"
//#define Name "TANetRoaming"//公開網路名稱：
"TANetRoaming"
void setup(){
pinMode(ledPin, OUTPUT);// 設定 ledPin 為輸出
pinMode(ledPin2, OUTPUT);// 設定 ledPin2 為輸出
pinMode(ledPin3, OUTPUT);// 設定 ledPin3 為輸出
digitalWrite(ledPin, LOW);// 設定 ledPin 輸出電位為低電位
12
digitalWrite(ledPin2, LOW);// 設定 ledPin2 輸出電位為低電位
digitalWrite(ledPin3, LOW);// 設定 ledPin3 輸出電位為低電位
Serial.begin(_baudrate);//設定 Arduino 序列埠通訊鮑率
ESP.begin(_baudrate);//設定 ESP-8266 序列埠通訊鮑率
void AE(){//初始化
ESP.println("AT"); re();//進入 AT 模式
ESP.println("AT+CWMODE=3"); re();//Station+AP 模式
//String cmd="AT+CWJAP=\"";cmd+=Name;cmd+="\",";//連線公開 Wi-Fi 網路
String
cmd="AT+CWJAP=\"";cmd+=Name;cmd+="\",\"";cmd+=Pass;cmd+="\"";ESP.printl
n(cmd);//連線私人 Wi-Fi 網路
ESP.println(cmd); re();
ESP.println("AT+CIPMUX=1"); re();//設定多重連線
ESP.println("AT+CIPSERVER=1,8087"); re();//開啟伺服器 設定連接埠
8087
ESP.println("AT+CIPSTO=7200"); re();//設定伺服器主動斷開前連時時間
ESP.println("AT+CIFSR"); re();//取得 ESP8266 IP 位址
}
void re(){//判斷初始化是否 OK
while(b == 'A'){
```

圖 22 第四種為無線監控數據，Arduino 配合 raspberry pi，Arduino 和 ESP8266 連線

實驗五：生理數據傳輸應用實驗

現今救護醫療系統尚未全面加裝即時連線系統，僅有少數醫學中心等級的院所才會建置院內的病患生理數據即時傳輸。一般如救護車、居家環境、安養院所等，如有遇到立即性的急救需求，大多無法立即式的回報訊息，往往會錯失搶救的黃金時間。因此生理數據即時傳輸的應用顯得相當重要，而且是必要全面普及實施。所以相較於現在救護車的一般模式，僅有移動載具能儲存病患的生理數值，以隨車攜帶的方式帶回醫療院所，我們在實驗五探討兩種可行的傳輸方式。

一、雲端傳輸

如圖 23 所示為雲端傳輸架構圖，感測器讀取的生理數值經由 EPS32 通訊板以 UART 的方式傳給 SIM7600X 4G HAT 的通訊板，再以 HTTPS 的模式傳送到雲端 Google Apps Script，在儲存至 Google Sheets 備查。如圖 24 的步驟建立 App Script 及 Google sheet，這個需要醫療院所配合雲端資料連線共享，便能達到即時生理數據的傳送，救護車或病患居家一上傳數據，醫療院所便能立即做好醫療的相關準備。

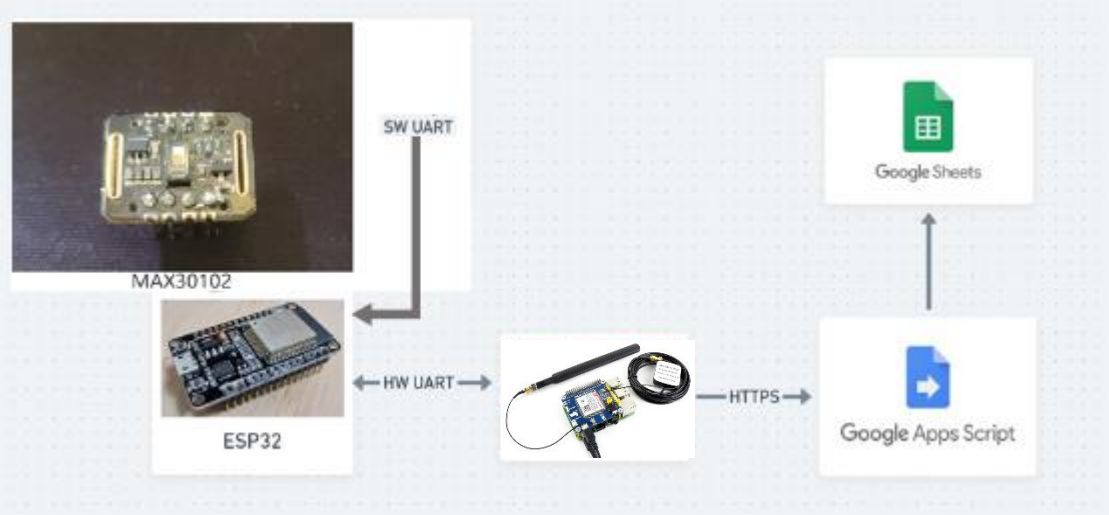


圖 23 雲端傳輸架構圖

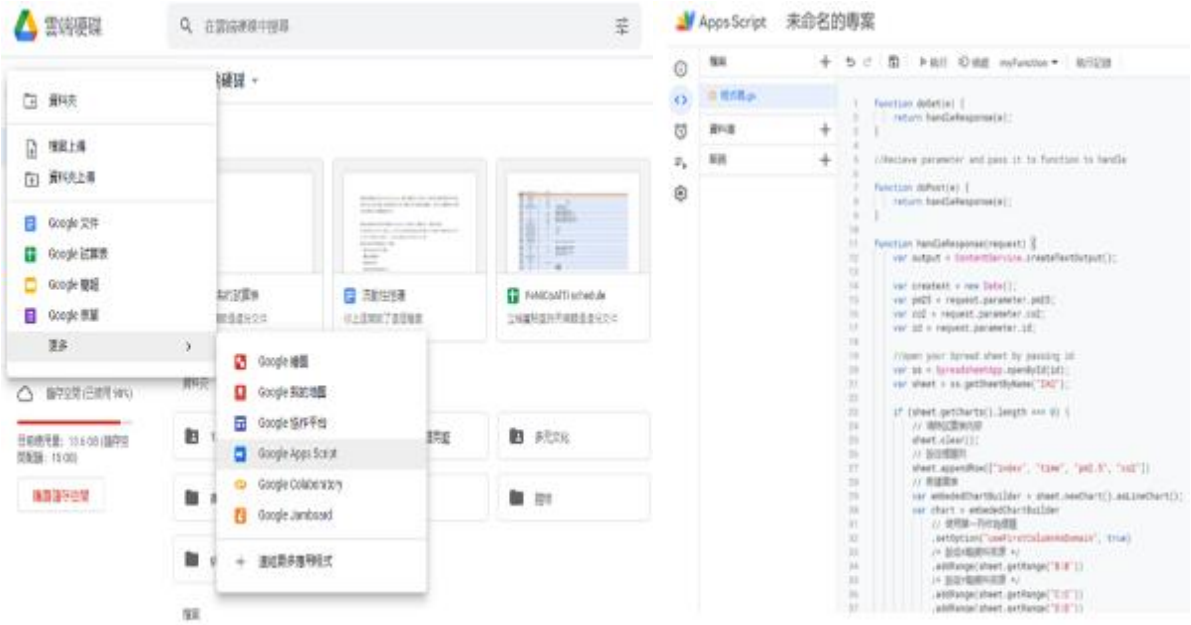


圖 24 建立 App Script 圖

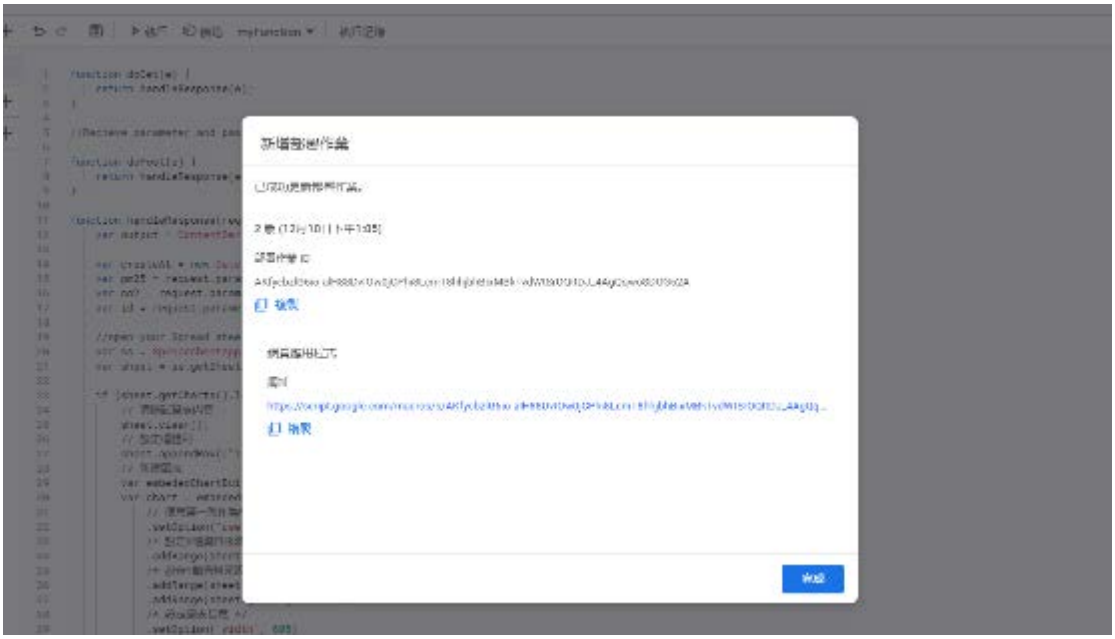


圖 25 建立 Google sheet 圖

二、傳數值給 Android 手機

傳輸使用 ArduinoBlueControl 如圖 26，，操作步驟如下 1. ~4. 說明，如圖 27 可在手機上獲得數據，需要醫療院所端有配置專屬的單位人員及時接收手機訊息。

1. 藍牙列表的設定－when List_BT.BeforePicking：在藍牙連線到裝置前，先將之前配對好的藍芽裝置匯入到準備連接的列表中。

2. 藍牙連線到裝置－when List_BT.AfterPicking：選好要連接的藍牙裝置後，將藍牙列表關閉、啟動計時器開始每隔一段時間讀值，並把斷線的按鈕設定為可執行。

3. 藍牙通訊（發送 key 與接收資料）－when Clock1.Timer：計時器啟動後，每隔一段時間(100ms)就會啟動一次，一開始我們會先發送一個 key(=97)給 Arduino，如果 Arduino 確任無　誤的話也會發送一個 key(=97)給手機，手機確認無誤後，會開始收兩個 byte(1byte = 8 bits)的數值，因為 Arduino 端送出的數值是採用 8bits 的封包，所以在這邊要將它還原回來必須將它從 8bits(2^8 = 256)轉回 10 進制，最後在除上 100 將單位轉成 cm。

4.斷線按鈕的設定－when Btn_Disconnect.Click：按下斷線後就會斷開藍牙通訊，並把按鈕配置回到一開始的狀態。

ArduinoBlueControl 的程式節錄如圖 28 所示。

圖 26 傳輸使用 ArduinoBlueControl

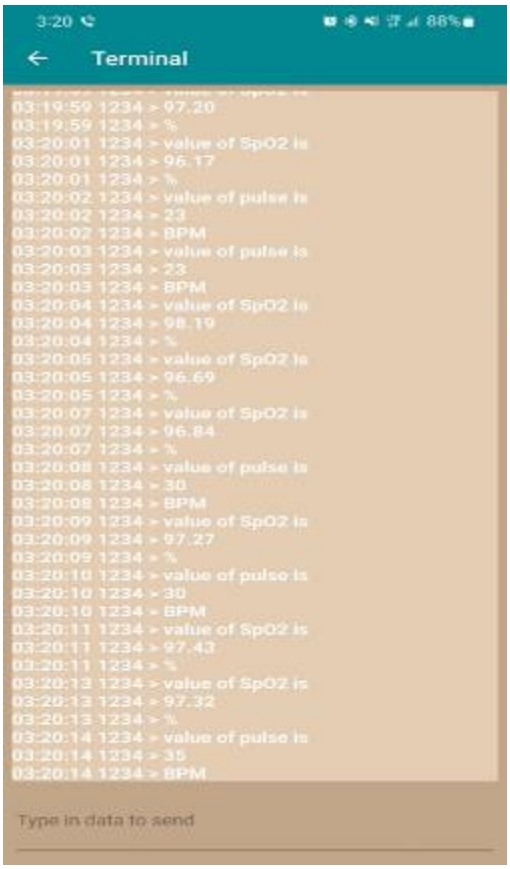


圖 27 可在手機上獲得數據

```
#include <Adafruit_GFX.h>       //OLED libraries
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include "MAX30105.h"           //MAX3010x library
#include "heartRate.h"          //Heart rate calculating algorithm
#include "ESP32Servo.h"
#include <BluetoothSerial.h>
BluetoothSerial BT;


MAX30105 particleSensor;
int Tonepin = 4;
//計算心率用變數
const byte RATE_SIZE = 10; //多少平均數值
byte rates[RATE_SIZE]; //心率陣列
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred
float beatsPerMinute;
int beatAvg;

//計算血氧用變數
double avered = 0;
double aveir = 0;
double sumirrms = 0;
double sumredrms = 0;

double SpO2 = 0;
double ESpO2 = 90.0;//初始值
double FSpO2 = 0.7; //filter factor for estimated SpO2
double frate = 0.95; //low pass filter for IR/red LED value to eliminate AC component
int i = 0;
int Num = 30;//取樣100次才計算1次
#define FINGER_ON 7000 //紅外線最小量 (判斷手指有沒有上)
#define MINIMUM_SPO2 90.0//血氧最小量

//OLED設定
#define SCREEN_WIDTH 128 //OLED寬度
#define SCREEN_HEIGHT 64 //OLED高度
#define OLED_RESET    -1 //Reset pin
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //Declaring
```

```
if (checkForBeat(irValue) == true) {
  display.clearDisplay();//清除螢幕
  display.drawBitmap(0, 0, logo3_bmp, 32, 32, WHITE);//顯示大的心跳圖示
  display.setTextSize(2);//設定文字大小
  display.setTextColor(WHITE);//文字顏色
  display.setCursor(42, 10);//設定游標位置
  display.print(beatAvg); display.println(" BPM");//顯示心跳數值
  display.drawBitmap(0, 35, O2_bmp, 32, 32, WHITE);//顯示氧氣圖示
  display.setCursor(42, 40);//設定游標位置
  //顯示血氧數值
  if (beatAvg > 30) display.print(String(ESpO2) + "%");
  else display.print("---- %" );
  display.display();//顯示螢幕
  tone(Tonepin, 1000);//發出聲音
  delay(10);
  noTone(Tonepin);//停止聲音
  Serial.print("beatAvg="); Serial.println(beatAvg);//將心跳顯示到序列
  BT.println("value of pulse is");BT.println(beatAvg);BT.println("BPM");
  long delta = millis() - lastBeat;//計算心跳差
  lastBeat = millis();
  beatsPerMinute = 60 / (delta / 1000.0);//計算平均心跳
  if (beatsPerMinute < 255 && beatsPerMinute > 20) {
    //心跳必須再20-255之間
    rates[rateSpot++] = (byte)beatsPerMinute; //儲存心跳數值陣列
    rateSpot %= RATE_SIZE;
    beatAvg = 0;//計算平均值
    for (byte x = 0 ; x < RATE_SIZE ; x++) beatAvg += rates[x];
    beatAvg /= RATE_SIZE;
                }
}
```

圖 28 ArduinoBlueControl 的程式節錄圖

伍、討論

回顧整個實驗流程，我們從實驗一到實驗六逐步設計逐步改善，以下就各個實驗階段來進行討論。

一、實驗一：在本實驗中建立一個具有實驗組與對照組的血氧檢測初始模板。

二、實驗二：以 6 通道的感測器加上移動平均的數值計算，可以明顯降低誤差值。

三、實驗三：以複雜的類神經網路技術來進行條件設定於演算學習，再加上參入誤差值的訓練，得到較佳的低誤差結果。

四、實驗四：本實驗探討四種資料庫的建立，其實各有優缺點，最終選擇的建立形式主要是端看使用者發射端與管理者接收端的管理方式而定。

五、實驗五：本實驗測試了 2 種遠距傳輸病患生理數據的方法，猜測可能以第二種方式較能貼近目前的大眾使用手機習慣的方便性。

六、原本在開發過程中所使用的 arduino uno/nano 板因為傳輸資料的需求，改而使用功能以及腳位更多元的 ESP32(DEVKIT V1),原以為能使用相同的擴充套件，根據多次查詢網路資源，如圖 29 所示。其中 Arduino forum 裡的討論區一位網友道破一切，他說：“That error indicates that the library is written for use with the AVR architecture (e.g. Uno, Leonardo, Mega, Nano, etc.). It can’t be used with ESP8266/32.”，如圖 30 所示。言下之意即為除了 uno,leonardo,mega,nano…等相似類型的開發版能使用，ESP8266/32 是不能使用此擴充套件的，當下立即重新以其他語法及套件去修正這套生理數據傳輸應用。

七、另外修改為 ESP32 開發板，每個腳位都需參照說明書上之腳位指示連接輸出輸入，和之前 uno/nano 板相較起來雖然功能較多，但也較為複雜，需特別注意。

八、這次的開發讓我們發現 Arduino 不僅僅只有單一玩法，配合各種零組件，甚至配合更多擴充套件，能延伸出更多玩法，體積雖然小，但能做出這麼多元化的物品，想必未來在科技應用上會是不可或缺的一環。

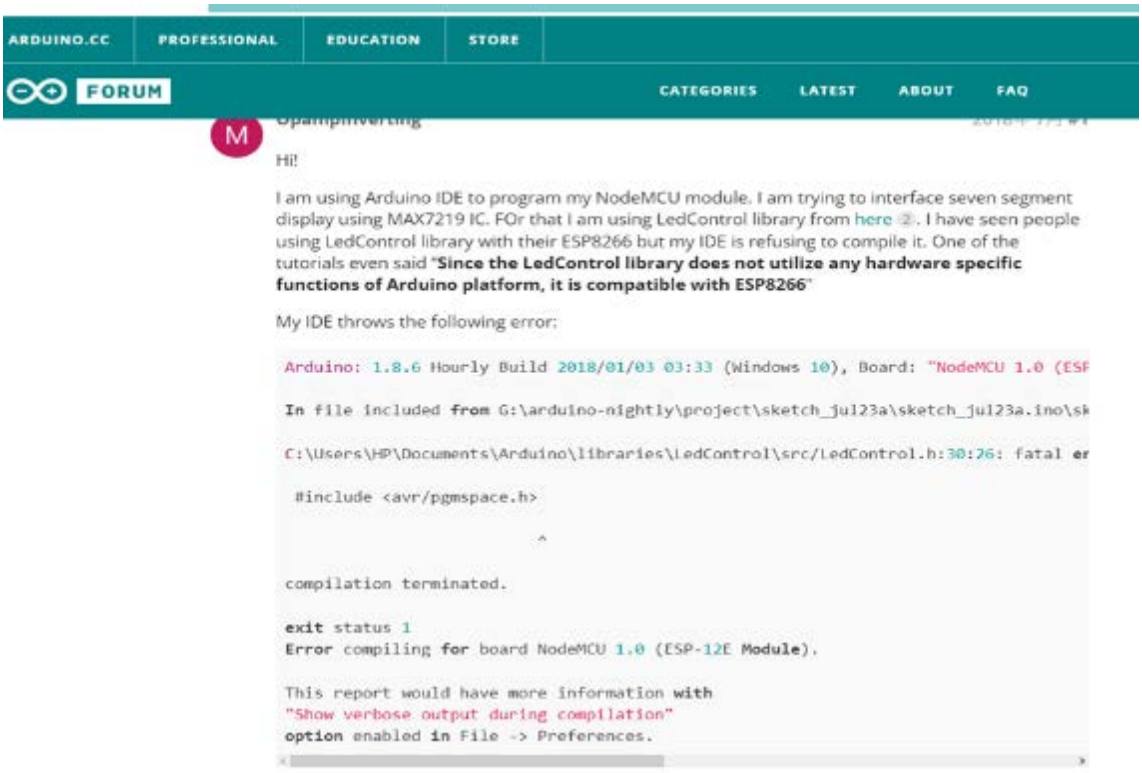


圖 29 查詢網路的程式語言討論區

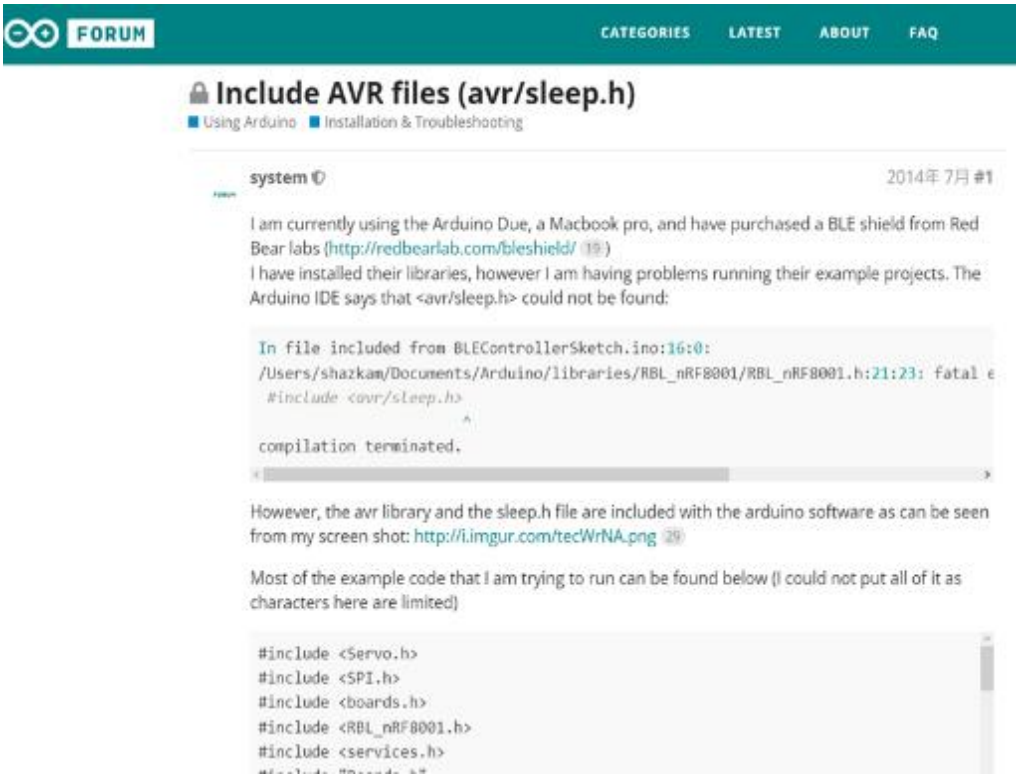


圖 30 在 Arduino forum 裡的討論區內容圖

陸、結論

針對本實驗的綜合結論如下：

一、可以順利自行組裝血氧濃度計，並撰寫程式測試成功，量測 20 次的數值平均誤差為 7.6%。

二、證實多通道的改善方法是有效的，量測 20 次的數值平均誤差降低為 2.3%。

三、以類神經網路加上 2%的誤差值參入訓練，讓數據結果誤差大幅降低至 0.46%。

四、資料庫的建立方法有多種，可以挑取適合想建立的架構來進行。

五、兩種遠距傳輸的方式均能順利進行，端看醫療院所的配合方案。

未來展望：規劃納入更多的生理數值、例如血壓、體溫、血糖等等，讓病患健康資訊能夠更加完整，更加有助於醫療照顧的完善。如圖 31 為這次實驗順道購置的血壓計，可以使用傳輸線將量測的數值傳輸到電腦，如圖 32 為電腦端的傳輸接收軟體，但是實際使用起來很不方便，量測完血壓值到電腦端接收數值的時間超過 5 分鐘，期待尋找更好的替代方案智慧醫療的發展不僅有實質的需求，更可以有效的拯救更多寶貴的生命，希望我們的研究是小小的開端，能夠在不久的將來看到理想完成的被實現。

圖 31 可傳輸血壓數值到電腦的血壓計

圖 32 傳輸線式血壓計的操作軟體

全國賽更新：持續研究下，我們找到藍芽式血壓計，如圖 33 所示，下載手機板的傳輸如圖 34 所示。實際使用結果，傳輸時間相當迅速，不到 5 秒鐘便能將收縮壓、舒張壓、心律值傳輸完成。目前能夠將量測的血氧濃度值、血壓值、心律值傳輸到手機，也可以藉由 SIM

圖 33 藍芽式血壓計

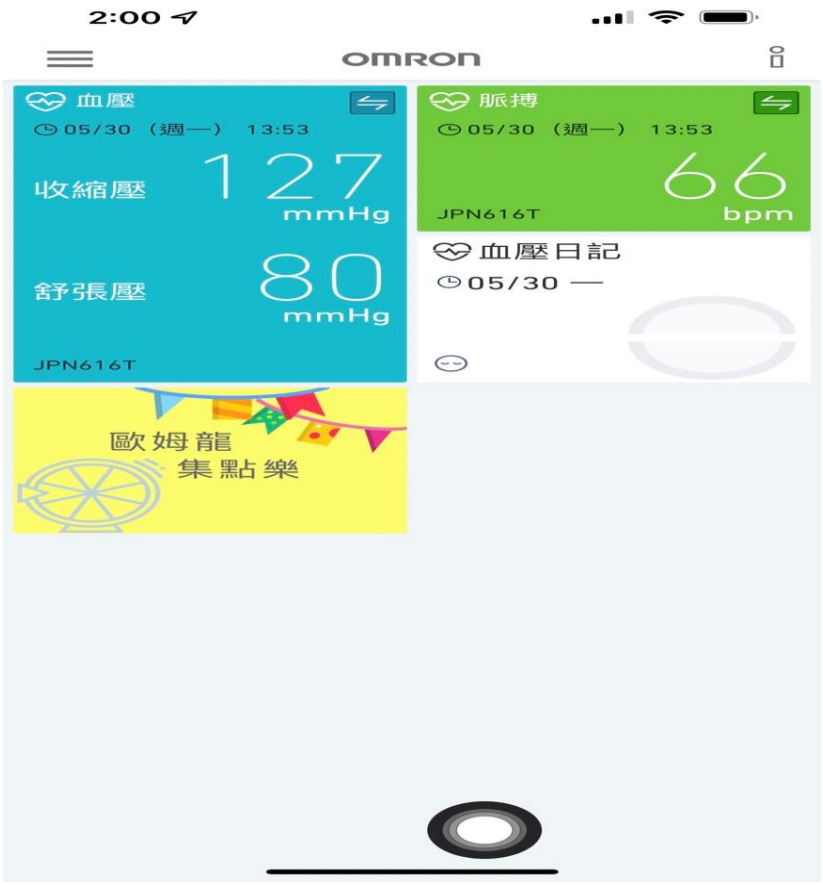


圖 34 藍芽式血壓計手機傳輸 APP

7600X 4G HAT 網路通訊板值將量測的數值直接傳送到雲端，完成初步的智慧醫療模式，須待醫療體系將醫療端地接收數據端口建置，並實際納入病患整體照顧的配套系統，才能算是完全的健全，如此便會是全國民眾、病患及家屬的美好佳音。

柒、參考文獻資料

一、非侵入式血氧濃度量測。葉國賢、黃寶震、陳怡良，第九屆離島資訊技術與應用研討會論文集。2010 Conference on Information Technology and Applications in Outlying Islands

二、非接觸式皮膚血氧濃度影像系統之應用。蔡心怡、黃國政、葉哲良，2018.03.31 科儀新知 214 期，長照與生醫工程專題。

三、應用於人體血氧濃度分析的皮膚影像檢測法。蔡心怡、黃國政、張漢釗、張中興，國科會 NSC 101-2622-E-492-004-CC3 計畫所補助，2018.03.09 第 13 屆全國 AOI 論壇與展覽。

四、生命徵象異常之居家評估及處理
https://www.chimei.org.tw/main/cmh_department/59012/info/D300/AD300110.html

五、遠端健康小幫手之研究，全國中學生小論文工程技術類，于明器、周季霖、錢思達，大安高工綜高三愛，指導老師張顯盛老師。
https://www.shs.edu.tw/works/essay/2017/03/2017032214555388.pdf

六、智慧醫療病床初探，全國中學生小論文工程技術類，陳宏瑞、黃靜文、孫茂綸，高雄市大榮高級中學電子電機群電子資訊科，指導老師周宗元老師。
https://www.shs.edu.tw/works/essay/2019/10/2019100722440935.pdf

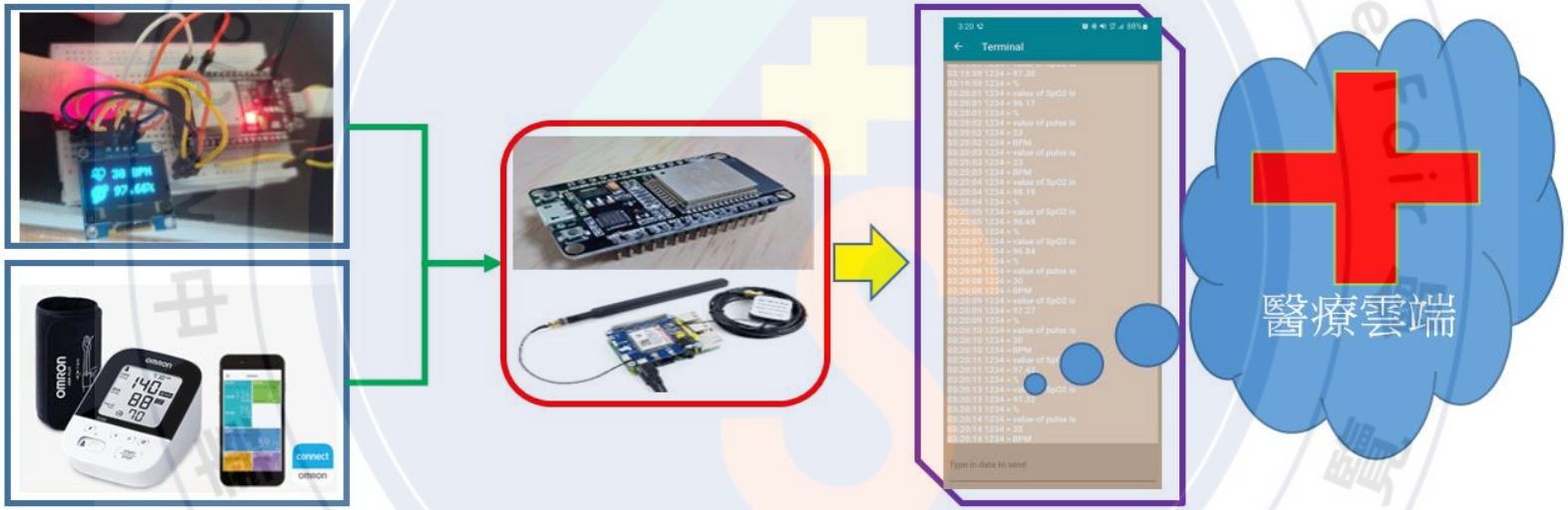
【評語】052312

本作品從自組血氧濃度計、利用六通道的交叉補償修正再加上類神經網路技術減少誤差，引入無線通訊技術及時傳輸生理數據，並建立雲端數據資料庫，已具備有智慧醫療健康管理系統的雛形，是個具巧思且完成度高的作品。團隊善用所學，循序漸進的改良作品成效，最後也達成一定成果，具備科學探究的精神。建議如下：

1. 說明如何交差補償修正多通道的訊號，相比於實驗１所用的單雙波長有何優勢?
2. 文獻回顧較缺乏說明擬進行發展相關技術目前研究的現況，呈現本作品的創新性及獨特性。
3. 實驗收案的程序以及產生誤差的原因須說明清楚。
4. 說明類神經網路的訓練樣本取得方式、訓練方法以及模型架構。
5. 本作品應和坊間商品進行比較，以確立本計畫的獨特性和貢獻。

作品簡報

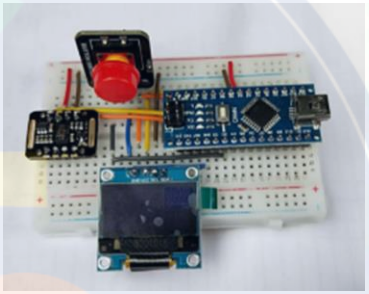
血氧濃度與智慧醫療

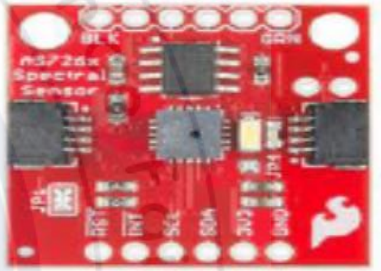


報告日期：2022/08/04

摘要 Abstract

- 自組血氧濃度計
- 六通道感測器
- 類神經網路技術
- 數據資料庫
- 無線通訊技術
- 智慧醫療

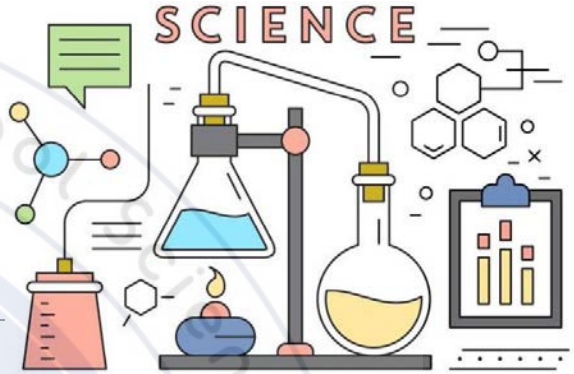




動機： **爭取搶救生命的關鍵時間**

目的： **研究將健康資訊送達雲端**

研究的實驗項目



實驗一：血氧濃度套件整合實驗

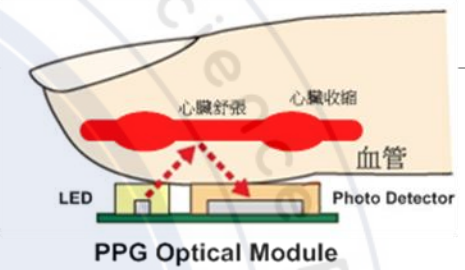
實驗二：血氧六通道感測器實驗

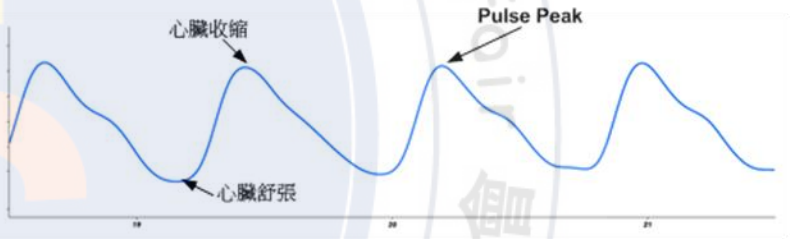
實驗三：血氧類神經網路實驗

實驗四：生理數據建立資料庫實驗

實驗五：生理數據傳輸應用實驗

文獻回顧





一：何謂血氧濃度?

二：智慧醫療現況!

三：類神經網路技術

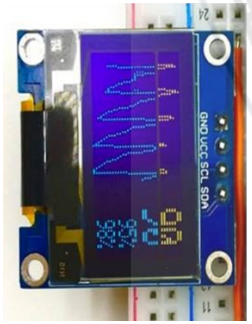
四：程式研究函數庫

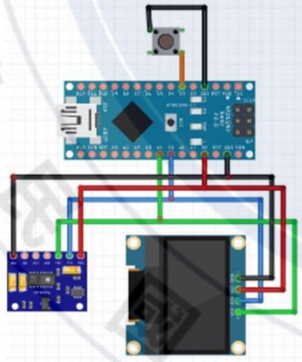
五：生命數值與電子設備結合技術

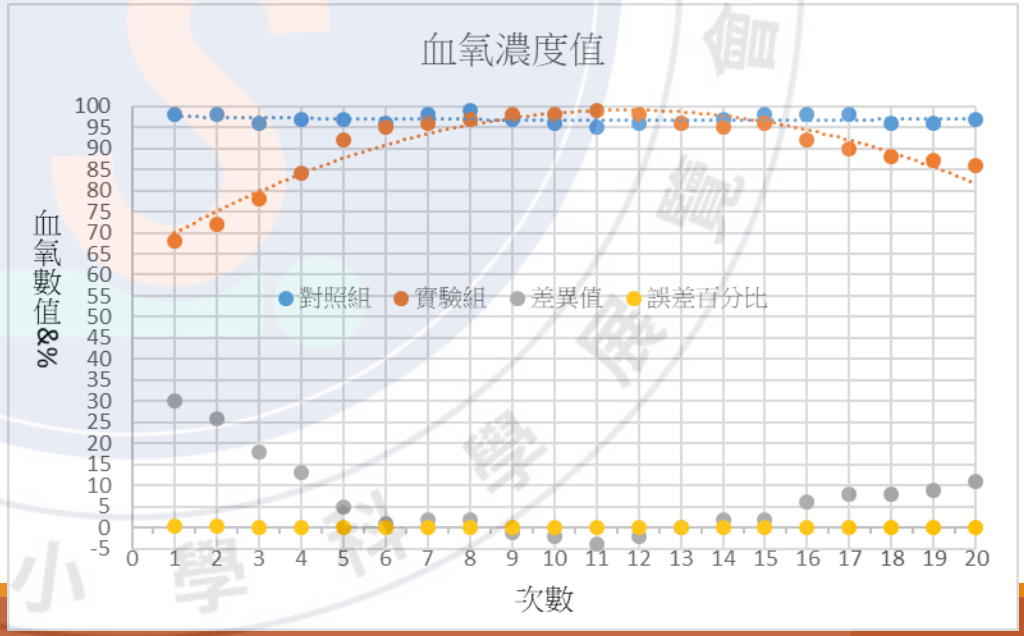
實驗一：血氧濃度套件整合實驗

結果：

可以順利自行組裝血氧濃度計，並撰寫程式測試成功，量測20次的數值平均誤差為7.6%。



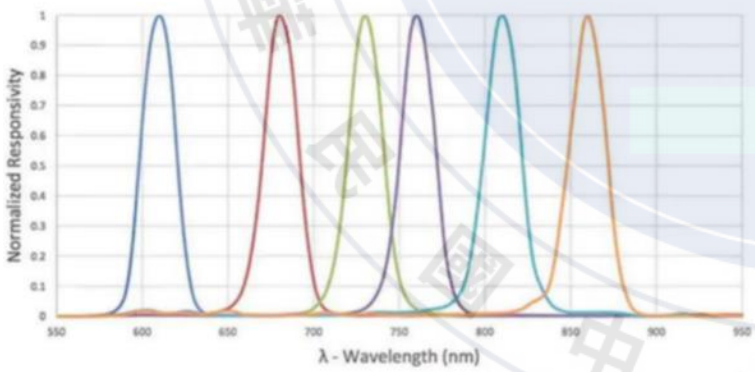


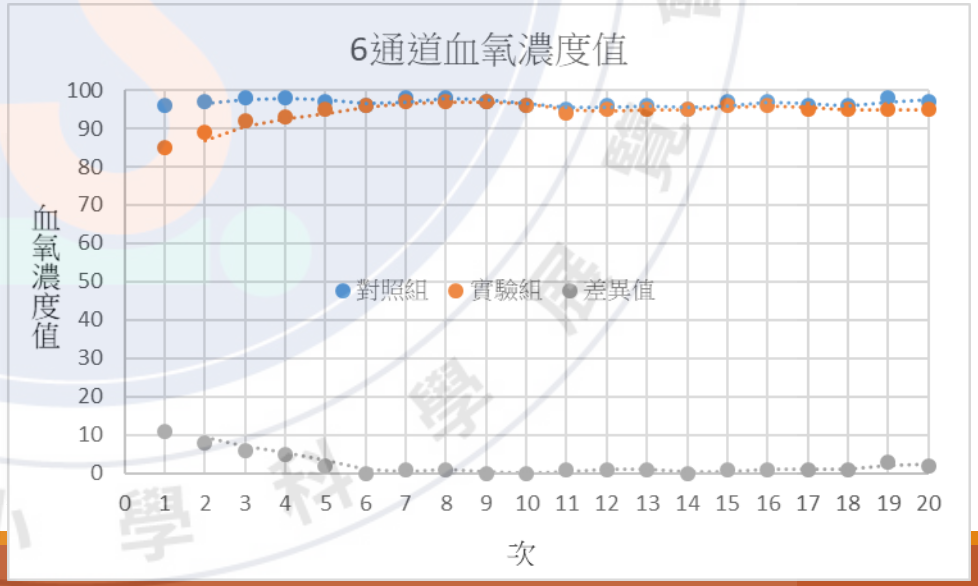


實驗二：血氧六通道感測器實驗

結果：

證實多通道的改善方法是有效的，量測20次的數值平均誤差降低為2.3%。

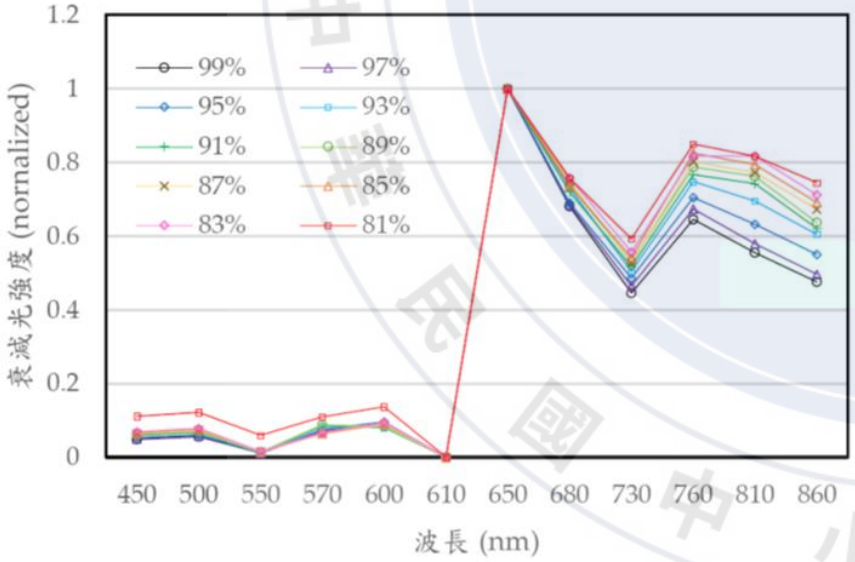




實驗三：血氧類神經網路實驗

結果：

以類神經網路加上2%的誤差值參入訓練，讓數據結果誤差大幅降低至0.46%。

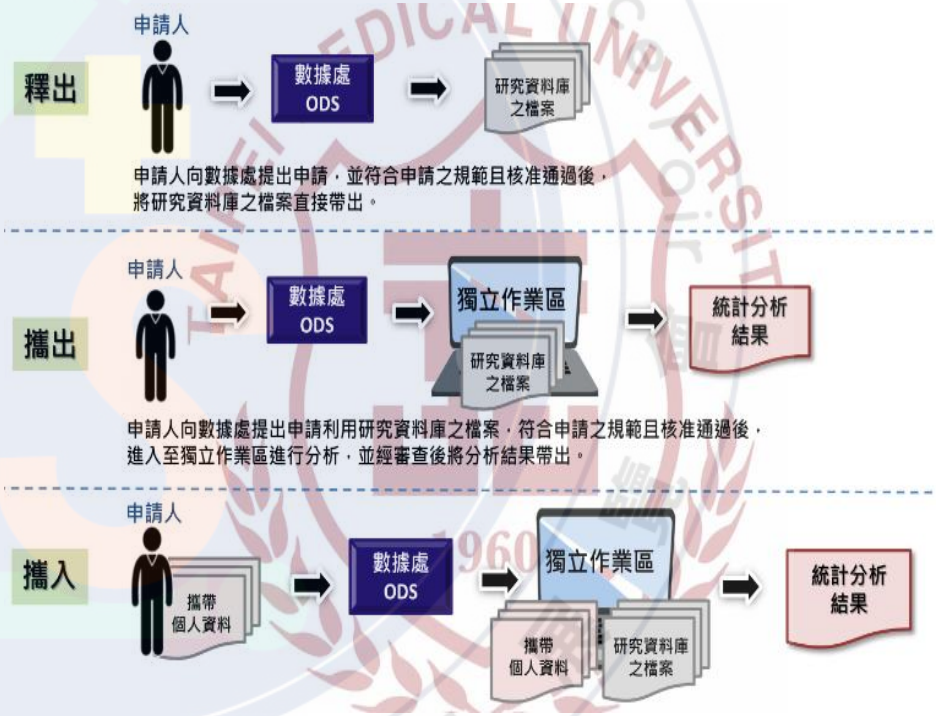


SpO_2 (%)	Maximum deviation(%)	Average deviation ± Standard deviation(%)
99	0.41	0.31 ±0.18
98	0.77	0.46 ±0.38
97	0.54	0.35 ±0.27
96	1.08	0.45 ±0.58
95	0.79	0.44 ±0.36
94	1.14	0.56 ±0.39
93	0.61	0.43 ±0.22
92	1.51	0.58 ±0.67
91	0.72	0.41 ±0.34
90	1.43	0.61 ±0.51
Total error : 0.46%		

實驗四：生理數據建立資料庫實驗

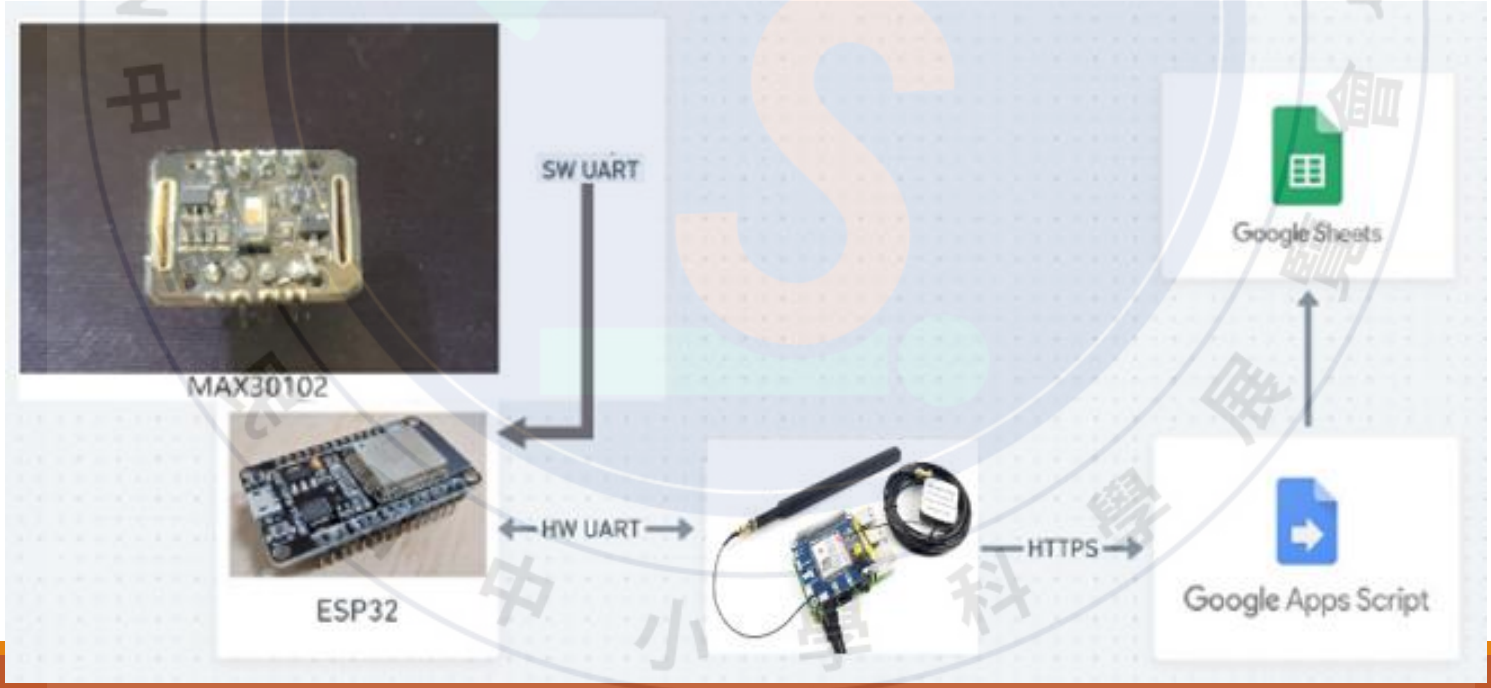
結果：

本實驗探討四種資料庫的建立，其實各有優缺點，最終選擇的建立形式主要是端看使用者發射端與管理者接收端的管理方式而定。



實驗五：生理數據傳輸應用實驗

結果：實驗測試了2種遠距傳輸病患生理數據的方法，猜測可能以第二種方式較能貼近目前的大眾使用手機習慣的方便性。



THE END

- ✓遇到的困難/解決辦法。
- ✓成長與收穫!
- ✓未來與展望!
- ✓謝謝您的聆聽!