長庚大學資訊管理學系

畢業專題報告

醫醫靈 — 智慧醫療手環

指導老師：林維昭 副教授

專題學生：林鉅全 (B0503202)

郝家緯 (B0644223)

陳詳鈞 (B0644127)

王鈞任 (B0644141)

李孟晟 (B0644251)

呂學霖 (B0644153)

109. 1

中華民國 一一O 年 一 月

**長庚大學資訊管理系畢業專題報告**

**指導老師同意書**

資訊管理系大學部

|  |  |
| --- | --- |
| B0503202 | 林鉅全 |
| B0644223 | 郝家緯 |
| B0644127 | 陳詳鈞 |
| B0644141 | 王鈞任 |
| B0644251 | 李孟晟 |
| B0644153 | 呂學霖 |

君等所提之畢業專題報告

醫醫靈 — 智慧醫療手環

業經本人審閱完畢，同意提付系辦存查。此致

系主任

指導老師 (簽名)

年 月 日

# 中文摘要

當病人在住院的時候，除了醫生和護理師會定期關心以外，通常會有家屬或是其他親朋好友過來探望、守夜，這時候發生任何問題，病人都會有其他人能輔助；相反地，當然也會出現只剩病人獨自待在病房的情況，這時候病人的最後手段就只剩醫院提供的緊急按鈕，緊急按鈕是有線的，所以能支援到的範圍大概就只有病床的周圍，當遠離病床時發生危險，病人將會失去求救的手段。

不過，近年來隨著物聯網(Internet of Things)技術的蓬勃發展，藉由各種嵌入式感測器和 API 等裝置，透過連上網際網路，使實際物體間也能夠相互溝通和進行訊息的交換。因此，我們欲利用 Arduino， 以手勢感測元件作為開發版，並利用IFTTT、LINE、Visual Studio 等開發工具，令使用者即時地傳遞動態資訊給醫護人員，透過減少例行性工作和清楚傳達病人目的的產品特性來減輕護理師的負擔，讓我們能以智能化的方式去提升工作的效率。

**目錄**

[長庚大學資訊管理系畢業專題報告 i](#_Toc61478700)

[中文摘要 ii](#_Toc61478701)

[目錄 iii](#_Toc61478702)

[圖目錄 vi](#_Toc61478703)

[第一章、緒論 1](#_Toc61478704)

[1.1 研究背景 1](#_Toc61478705)

[1.2 研究動機與目的 1](#_Toc61478706)

[1.3 特點設計 3](#_Toc61478707)

[第二章、產品介紹 4](#_Toc61478708)

[2.1 設計理念 4](#_Toc61478709)

[2.2 手勢感測元件原理圖 5](#_Toc61478710)

[第三章、功能概述 6](#_Toc61478711)

[3.1 流程圖 6](#_Toc61478712)

[3.2 硬體元件介紹 7](#_Toc61478713)

[3.2.1 建奕科技 gesture\_V2 開發版 7](#_Toc61478714)

[3.2.2 TTL to USB 模組 8](#_Toc61478715)

[3.3 軟體介紹 9](#_Toc61478716)

[3.3.1 IFTTT (If This Then That) 9](#_Toc61478717)

[3.3.2 App Inventor 2 11](#_Toc61478718)

[第四章、操作說明 14](#_Toc61478719)

[4.1 手勢「更換點滴」 14](#_Toc61478720)

[4.2 手勢「緊急呼叫」 15](#_Toc61478721)

[4.3 手勢「查閱行程」 16](#_Toc61478722)

[第五章、系統特色 17](#_Toc61478723)

[5.1 實體 17](#_Toc61478724)

[5.2 網路 19](#_Toc61478725)

[5.2.1 多功能App 19](#_Toc61478726)

[5.2.2 手機推播 20](#_Toc61478727)

[5.3 產品概念 21](#_Toc61478728)

[第六章、總結 22](#_Toc61478729)

[6.1 研究目標之達成 22](#_Toc61478730)

[6.2 未來展望 22](#_Toc61478731)

[6.3 結論 23](#_Toc61478732)

[參考文獻 24](#_Toc61478733)

[附錄 25](#_Toc61478734)

[A. ARDUINO 程式碼 25](#_Toc61478735)

[函式庫 25](#_Toc61478736)

[定義宣告 25](#_Toc61478737)

[設定指令內容 26](#_Toc61478738)

[設定姿態角度 26](#_Toc61478739)

[初始化 27](#_Toc61478740)

[迴圈 29](#_Toc61478741)

[LED燈控制 29](#_Toc61478742)

[手勢判斷 30](#_Toc61478743)

[尤拉角公式 34](#_Toc61478744)

# 圖目錄

圖1、102年12月至103年2月的護理人員總數及離職率 2

圖2、產品外觀正面 4

圖3、手勢感測元件原理圖 5

圖4、流程圖 6

圖5、建奕科技 gesture\_V2開發版的主要組件、介面圖 8

圖6、TTL to USB 模組 8

圖7、IFTTT 設定頁面 ..10

圖 8、IFTTT 觸發事件 ..10

圖9、IFTTT 設定LINE 的訊息 .11

圖 10、手機App登入介面示意圖 .12

圖 11、手機App主要介面示意圖 .12

圖 12、手機App輔助介面示意圖 .13

圖13、更換點滴情境示意圖 .14

圖14、緊急呼叫情境示意圖 .15

圖 15、查閱行程情境示意圖 .16

圖 16、手勢動作圖 .17

圖 17、App 手機介面圖 .18

圖 18、手機推播示意圖 .19

# 第一章、緒論

## 研究背景

有賴於互聯網技術蓬勃發展，物聯網(Internet of Things)已成為現今的熱門關鍵詞。我們希望透過此次專題，能深入了解物聯網技術的原理和應用，並期望在學習之餘，也能同時解決社會問題。

## 研究動機與目的

現今的物聯網技術日新月異，對各個產業來說，不管在效率還是效果上都獲得了豐碩的成果，而對使用者來說，更是為生活、工作增添一位得力的助手；然而，我們發現到物聯網發展的趨勢偏向讓行動方便的人變得更便利，而忽略了那些行動不便的人，因此本次作品的動機就是想讓物聯網這項技術為了真正需要的人而使用。經過醫院實際探訪之後，我們發現了一些問題，並且認為我們有能力為此作改善，因此決定了病人與護理師為我們作品的使用者，以下為我們想解決的問題：

1. **病人的獨立性**

病人共同的特徵就是虛弱，在這種狀態下，他們真的有能力獨自完成緊急求救之類的行為嗎？難道緊急時刻發生時，他們只能「坐以待斃」嗎？因此為了預防悲劇發生，病人的獨立性是我們要面對的問題。

1. **護理人員的短缺**

護理人員的短缺源自於過高的離職率，而主要的問題就是工作量過大、工作壓力大以及超時工作這三項，因此我們思索著能否利用我們所學的物聯網技術來使護理師的工作簡化或是減輕負擔。

若再更進一步觀察，聚焦在護理人員的「離職率」上，護理執業環境惡化主要因素在於護理人力嚴重的不足，圖 1 是我們 103 年中華民國護理師公會全國聯合會發行的全聯護訊，統計出的護理人員總數及離職率。由圖1可知，全年離職總數8,227人，總離職率12%，與以往96~100年(離職人數9931~11365人)總離職率(12.89%，12.01%，10.44%，11.59% ，12.56%)比較，雖比100年的12.56%低，但近6年的比較差距並不大，護理人員的短缺實屬大問題。

一張含有 文字, 收據 的圖片

自動產生的描述

圖 1、102 年 12 月至103 年 2 月的護理人員總數及離職率1

1 中華民國護理師公會全國聯合會全聯護訊

http://www.nurse-newsletter.org.tw/index.php/099b/09905

(擷取日期：2020/12/29)

因此，我們專題最主要的目的是希望能藉由物聯網的技術，來降低因工作量過大、工作壓力大以及超時工作這三項問題衍生出的護理人員的短缺。我們預期能透過本專題，讓護理師能透過減少例行性工作和清楚傳達病人目的的產品特性來減輕負擔，同時藉由攜帶方便的行動式裝置與簡易快速的手勢，提升病人求救的效率，藉此解決病人的獨立性問題。

## 特點設計

為了達成上述的目的，我們的裝置擁有以下三個特點，分別是：

1. **手勢感測**

與過往按緊急按鈕相比，傳遞的訊息更多樣化，並運用富含意義

的手勢，不僅應用方便，也讓病人更方便記憶。

1. **行動化設計**

可隨身攜帶，因此不會有使用範圍的問題。

1. **目的性明確**

當狀況發生時，藉由不同手勢的即時性推播通知，來清楚得知病

人需求，並得以區分輕重緩急，避免護理人員白跑一趟，來減輕

護理人員負擔。

# 第二章、產品介紹

## 設計理念

智慧穿戴裝置我們秉持「輕便、便宜、簡易化」的設計理念，希望能透過我們的產品帶給配戴者一個「完善的照護體驗」。

我們整體的外觀是利用3D列印所製成，產品表面質地使用的是PLA材質。在外觀的選色上，我們除了選用霧面白色為基底，使整體風格簡約乾淨，卻又不失美感。此外，盒子還設計了滑蓋式的蓋子，可以輕鬆取出手勢元件維修。其外觀構造如下圖 2 所示。

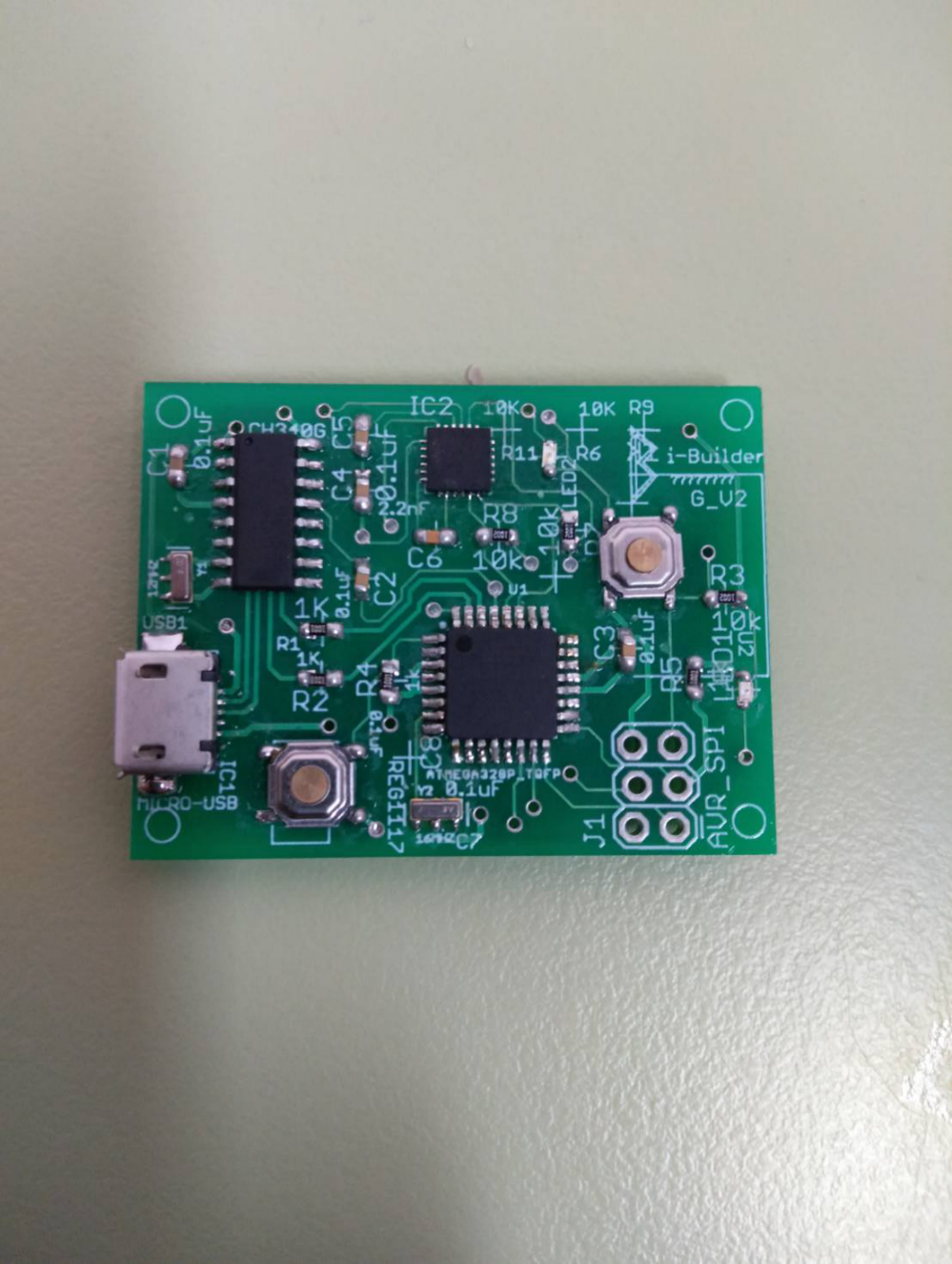
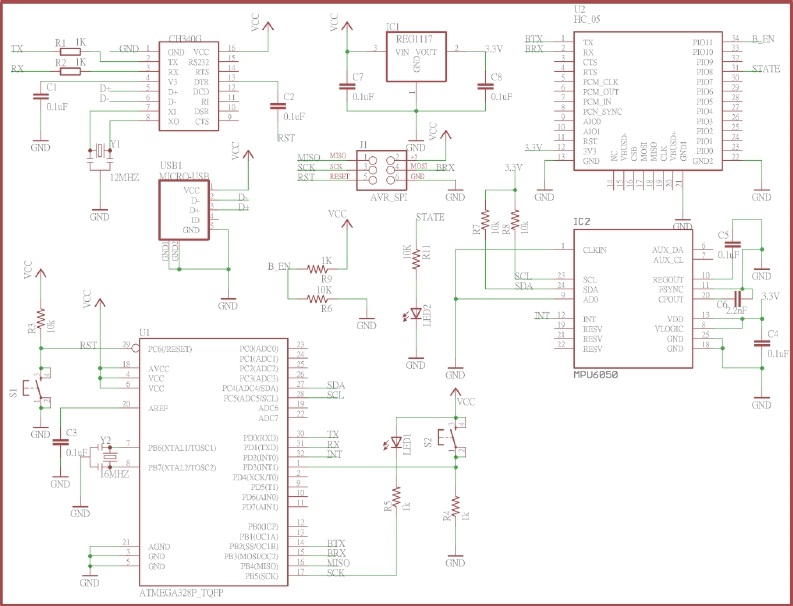


圖 2、產品外觀正面

## 手勢感測元件原理圖

圖 3、手勢感測元件原理圖

****

# 第三章、功能概述

## 流程圖

首先，手勢感測元件會自動連結藍芽。在確定已經連結藍芽後，使用者需要先按下重設按鍵，則手勢感測元件會重新校準，接下來手勢感測元件會進行判斷，判斷使用者是否有做出「緊急求救」、「更換點滴」、「查閱行程」這三種手勢，當使用者做出對應手勢後，手勢後感測元件會透過藍芽傳送手勢對應值到使用者的App端，然而透過IFTTT推播信息到LINE以及在App欄位內顯示回饋信息，然後重新回到手勢判斷。

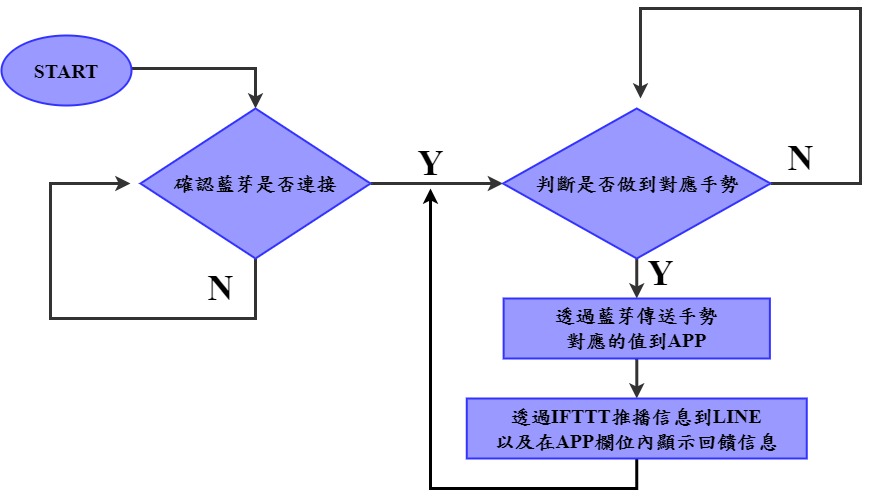


圖 4、流程圖

## 硬體元件介紹

### 建奕科技 gesture\_V2 開發版

Arduino是一個開放源代碼的單芯片微電腦，它使用了Atmel AVR 8-bit單晶片處理器，採用了基於開放源代碼的軟硬件平台，構建於開放源代碼Simple I/O介面，並具有類似Java、C語言的Processing/Wiring 開發環境。並且Arduino擁有以下四個優點：

1. **跨平台**

Arduino IDE 能夠在主流作業系統上運行，使開發人員可自由選擇自己鍾愛的平台上進行開發。

1. **價格低廉**

其價格不貴，且開發軟體亦為免費開源。

1. **開源的軟硬體**

Arduino本身提供許多開源的硬體元件，提供創客們進行採購；且在網路(例：GitHub)亦有許多的專案、程式碼分享，資源非常豐富。

1. **簡單的編程方式**

Arduino 的編程語言相較於其他程式語言 (例如：C 語言、Java)較為簡單，適合程式初學者撰寫。

因此，Arduino 非常適合作為物聯網設備的入門學習使用。

我們使用的是由建奕科技 gesture\_V2 開發模組，其外觀構造如下圖5 所示。

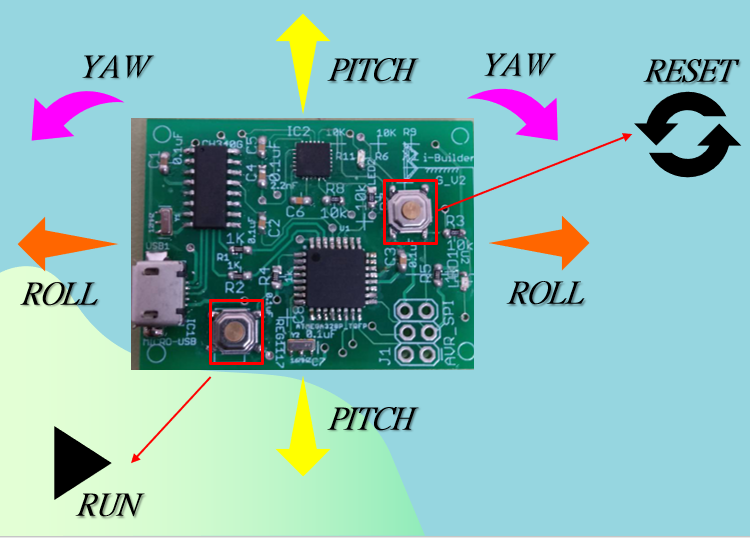


圖 5、建奕科技 gesture\_V2開發版的主要組件、介面圖

### TTL to USB 模組

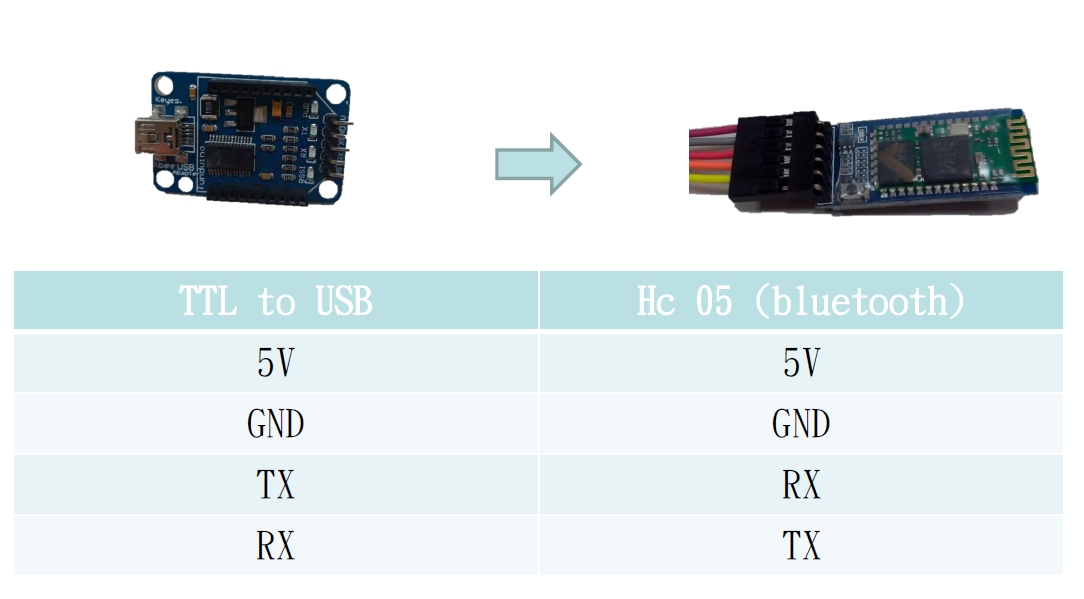
 USB TTL 串行電纜是一系列 USB 到串列轉換器電纜，提供 USB 和串列 UART 介面之間的連接。提供一系列電纜，可在 5V、3.3V 或使用者指定的信號電平內提供各種連接器介面。所有電纜均採用整合在電纜 USB 類型「A」連接器中的 FTDI FT232R 裝置，該連接器提供對 UART 傳輸（Tx）、接收（Rx、RTS#、CTS#、VCC（5V）和GND連接的存取。所有電纜都完全符合 RoHS 標準，並經過 FCC/CE 認證，如圖6 所示。

圖 6、TTL to USB 模組

## 軟體介紹

### IFTTT (If This Then That)

IFTTT是一個「網路自動連結」的服務平台，基於某個任務條件被觸發， 而執行某個動作，觸發的條件稱作「Triggers」，觸發條件後所執行的任務稱為「Actions」。IFTTT 的優點即為它讓各個網路服務之間不再是靜態的分佈，它把不同的網路服務串連成一系列的「行動」， 當「如果」A 服務發生什麼事情，「那麼」B 服務就做出什麼反應。 IFTTT 目前有 Android、iOS 和網頁版，目前支援許多網路免費服務，例如：LINE、 Facebook、Twitter 等軟體。我們系統連動至手機端LINE 推播的部分，即是透過 IFTTT 這個平台進行的。

我們事先在 IFTTT 上設立一個名叫「command」的觸發事件，並選擇 LINE 為其觸發的項目，下如圖 7、8 所示。連動自己的LINE 帳號、完成綁定的動作後，我們根據其不同的瓦斯使用狀況，如 4.1~4.3 節所述的三種狀況，分別給予它們不同的 Value 值， 如圖 9 所示，作為 LINE 的推播內容，其 LINE 推播的訊息內容如

5.2.2 節所示。其程式碼如附件 A.5 所示。

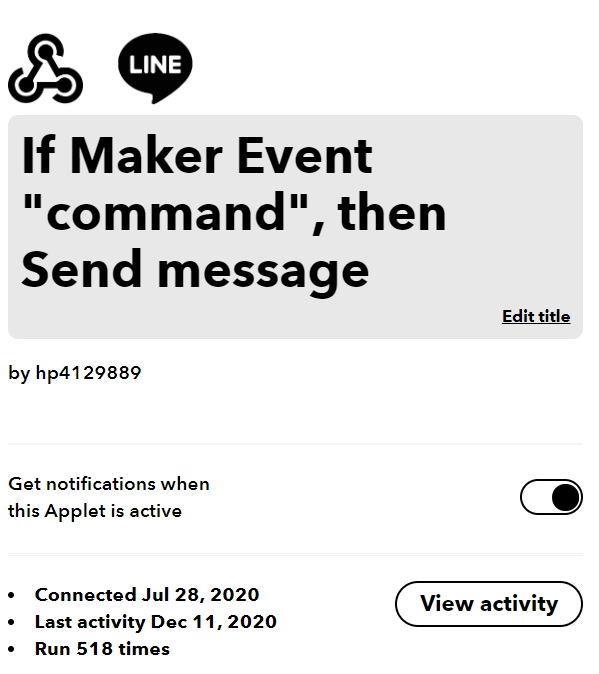


圖 7、IFTTT 設定頁面

Z

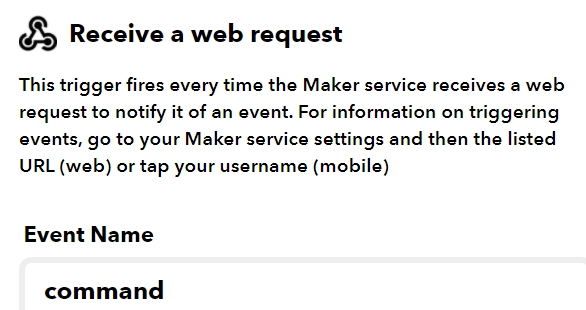


圖 8、IFTTT 觸發事件

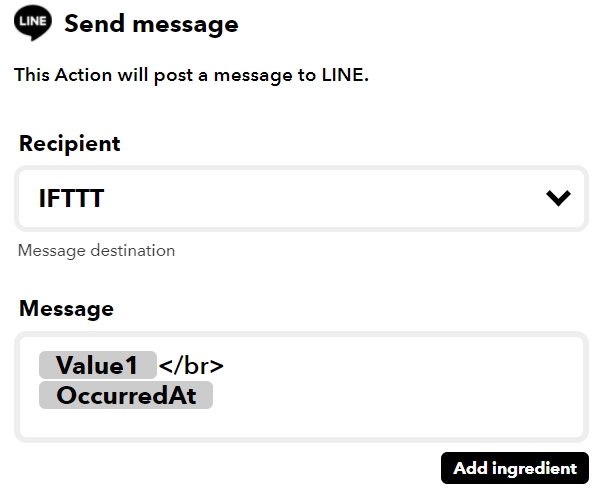


圖 9、IFTTT 設定 LINE 的訊息

### App Inventor 2

App Inventor 原是Google實驗室（Google Lab）的一個子計畫，由一群Google工程師與勇於挑戰的Google使用者共同參與。Google App Inventor是一個完全線上開發的Android程式環境，拋棄複雜的程式碼而使用樂高積木式的堆疊法來完成您的Android程式。除此之外它也正式支援樂高NXT機器人，對於Android初學者或是機器人開發者來說是一大福音。因為對於想要用手機控制機器人的使用者而言，他們不大需要太華麗的介面，只要使用基本元件例如按鈕、文字輸入輸出即可。下圖10為本專題實作的App介面。

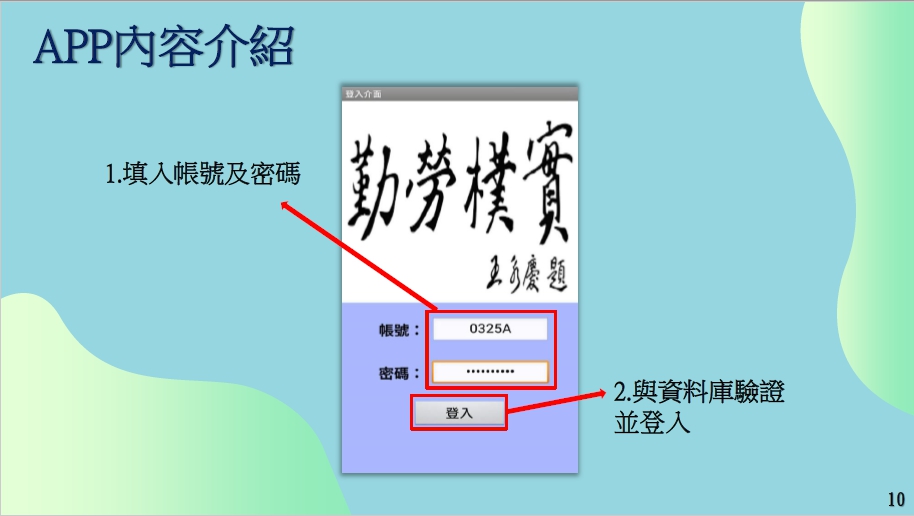


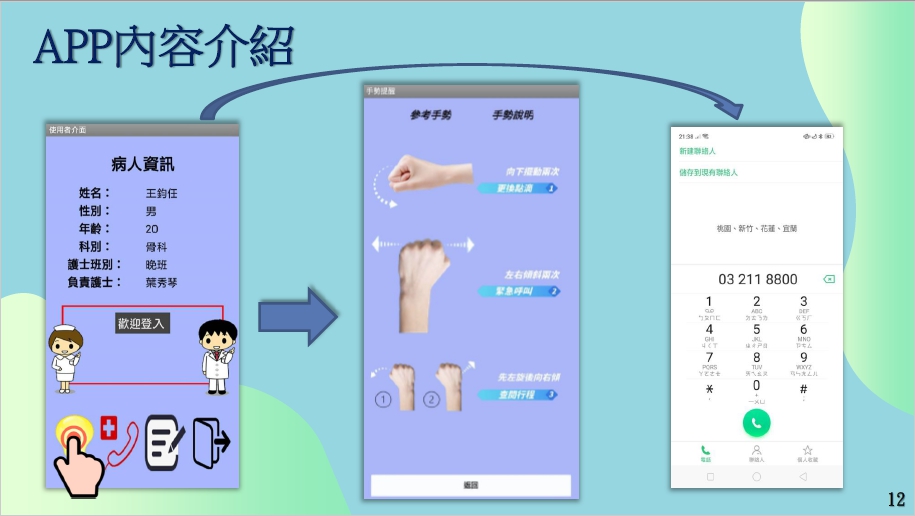
圖 10、手機App登入介面示意圖

圖20為本專題登入畫面，輸入完帳號密碼後，按下登入按鈕，與資料庫驗證登入。由於App是病人在操作，所以帳號密碼在本專題中設定為病人的病床號和身分證字號。



圖 11、 手機App主要介面示意圖

圖11為本專題主要介面，(1)為病人資訊是和資料庫連結直接呈現。中間即時回饋，其運作原理為手勢元件所發送的指令將會經由Web service傳送到App上，並在即時回饋欄呈現，呈現後也會推播至醫院端的LINE上，讓護理師即時收到病人需求。



(A)功能介紹—手勢提醒、緊急電話



(B)功能介紹—留言板

圖 12、手機App輔助介面示意圖

圖(A)第一項為手勢提醒，在病人不熟悉我們的系統，或是忘記手勢，都可以隨時進來手勢提醒複習。圖A第二項為緊急電話，在護理師照顧病人時，可能會有些突發狀況需要通知家屬，可以使用此功能快速聯絡家屬。

圖(B)為留言板，此功能屬於休閒向，病人在住院期間需要長時間待在病床上，時間一久病人的心情可能會逐漸煩悶，而留言板則可以讓他們與其他病人聊天，一同度過煩悶的住院時光。

# 第四章、操作說明

本產品主要依照不同情形作相對應的安全措施，就其本實驗我們分成三個狀況，依其實驗的本質與目的來達到減少護理人員重複工作效果。

## 手勢「更換點滴」

首先，手勢感測元件會自動連結藍芽。在確定已經連結藍芽後，若使用者做出「更換點滴」手勢時，使用者會在App端上看到發出「更換點滴」信號，為了方便使用者了解自己是否有發出信號，並且會推播訊息到醫護端LINE使護理人員了解使用者在哪一時間發出「更換點滴」信號。



圖 13、更換點滴情境示意圖

## 手勢「緊急呼叫」

首先，手勢感測元件會自動連結藍芽。在確定已經連結藍芽後，若使用者做出「緊急呼叫」手勢時，使用者會在App端上看到發出「緊急呼叫」信號，為了方便使用者了解自己是否有發出信號，並且會推播訊息到醫護端LINE使護理人員了解使用者在哪一時間發出「緊急呼叫」信號。



圖 14、緊急呼叫情境示意圖

## 手勢「查閱行程」

首先，手勢感測元件會自動連結藍芽。在確定已經連結藍芽後，若使用者做出「查閱行程」手勢時，使用者可以在App上看到未來排定之檢查行程等。



圖 15、查閱行程情境示意圖

# 第五章、系統特色

首先我們將產品分成實體跟網路，來去進行內部功能及效用的說明。

## 實體

實體的部分主要為手勢感測功能，根據使用者的需求自行做出對應的手勢之後再給予回饋。手勢主要分為三個：向下擺動兩次以進行呼叫護士進行更換點滴、左右傾斜兩次以傳送緊急呼叫的訊息、先左搖後向下可以在App上查閱今日的行程，如圖16所示。

手勢的動作主要是參考國際通用手勢以及手語來做參考的動作，有方便好記的特色以及較不容易做出對應動作的特性來避免無意呼叫卻發出請求指令的動作。



圖16、手勢動作圖

## 網路

網路的部分主要有多功能App及手機推播，就App遠端察看的部分，主要是顯示病人資訊以及指令是否有發送出與查看形成的部分，方便讓使用者可以及時確認動作是否確實以傳送對應的指令。而手機推播主要利用訊息通知的方式，當使用者發出請求訊息時，護理師就能馬上收到通知到病房進行對應的處置。

### 多功能App

而多功能App的介面特色有可以即時觀看是否有傳送出指令的功能，以及查閱手勢、緊急電話與聊天室等其餘輔助功能。當使用者一開始進行輸入完帳號登入後，就能看到自己的資訊以及各式功能，如圖17 所示。



圖 17、App 手機介面圖

### 手機推播

手機推播我們主要是利用LINE 去發送訊息的方式，因為 LINE 是最廣泛的社交軟體，他可以不改使用者習慣的去讓使用者方便使用，而且他可以根據使用者偏好去更改訊息內容，可以針對不同狀況發送不同的訊息通知，更因為目前的手機中都有此配備，因此在設備的導入當中是比較沒有問題的，如圖18所示。

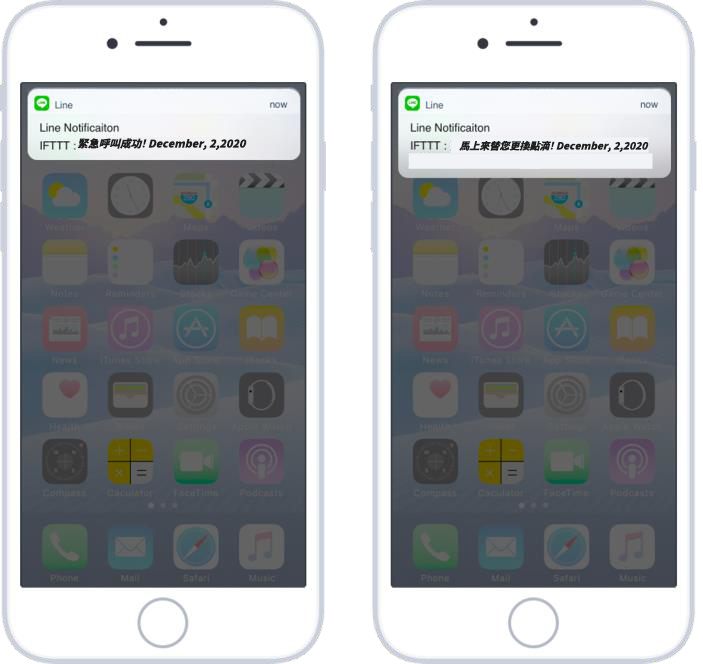


圖 18、手機推播示意圖

## 產品概念

產品概念主要為物聯網的想法，且因為是可以隨身配戴在手上的，相較於現在所使用的緊急按鈕只能在病床上使用，我們希望的是產品在任何時刻皆能產生作用。而物聯網則是因為在現今科技發達的世代，許多產品都會開始結合實體與網路去讓人們可以相對簡單的去使用任何產品，像是掃地機器人、智慧電燈等等，各式各樣的產品來改善的生活，我們也藉由這個特性來改善了病人的住院品質。

# 第六章、總結

本專題透過現今正流行的智慧型手環，希望能取代掉以往舊式的緊急鈴等，期許未來在物聯網更加蓬勃發展的時代，可以使護理人員的工作壓力減輕，也讓我們這項作品的應用能更加親民。

## 研究目標之達成

本專題針對智慧型手環來作為主軸，平常日時還是由護理人員作照料，但有可能因為護理人員正在繁忙無暇顧及，病人疏於照顧或許會有突發意外的狀況發生。為了防止這種情況的發生，我們在智慧型手環上使用了手勢感測器能讓病人可以迅速告知護理人員自身的需求，希望能讓緊急鈴的作用作出更多發展，我們也透過不一樣的手勢來做出更多的反饋及，且結合手機推播及App來做查看，讓親屬及護理人員都可以簡單的掌握院區病人狀況。

## 未來展望

有了初步的雛型架構後，為了使系統更加完善，於此我們希望在未來智慧家居應用上可新增以下三個項目應用：

1. **安全防護及手勢防呆機制**

我們希望未來可以利用此產品，讓突發意外的發生頻率降低，並且減少手環使用者可能做出誤觸時的反饋效果。

1. **手機推播**

利用了現今普羅大眾的社群軟體，可以立即發送警示通知，使得護理人員可以馬上處理危機。

1. **App 遠端確認**

期望親屬與醫院可以透過手機App來了解目前病人的狀況。

## 結論

在醫院的與病人的陪伴是親屬們最安穩的一段時間了，但我們又該如何保證病人在無人照料的情況下能夠，為了防止一些突發意外的發生，我們想透過目前發展趨勢也相對簡單的智慧手環，希望能對醫院環境能有最直接的輔助，也期許能藉此減少護理人員們的工作壓力。我們結合網路App可以隨時掌握狀況，透過手勢感測器偵測回傳資料至 App的方式，讓用戶能迅速得知是否有異狀，並對其作出反映。

# 參考文獻

〔1〕黃建庭，《輕鬆玩 Arduino 程式設計與感測器入門》，初版， 台北，碁峰資訊，民國 107 年。

〔2〕蔵下まさゆき著，吳炘廷譯，《Arduino 連上網好好玩!：手機遠端遙控•即時監測•雲端智慧通知》，初版，台北，旗標，民國106 年。

# 附錄

## A. ARDUINO 程式碼

### 函式庫

//#include "I2Cdev.h"

#include "MPU6050\_6Axis\_MotionApps20.h"

#include <SoftwareSerial.h>

SoftwareSerial bluetooth(10, 11); // RX, TX

#include "Wire.h"

MPU6050 mpu;

**定義宣告**

#define INTERRUPT\_PIN 2 // use pin 2 on Arduino Uno & most boards

#define button 3

#define led 13

bool dmpReady = false; // set true if DMP init was successful

uint8\_t mpuIntStatus; // holds actual interrupt status byte from MPU

uint8\_t devStatus; // return status after each device operation (0 = success, !0 = error)

uint16\_t packetSize; // expected DMP packet size (default is 42 bytes)

uint8\_t fifoBuffer[64]; // FIFO storage buffer

unsigned long Time;

unsigned long gestureTime;

float duration;

Quaternion q; // [w, x, y, z] quaternion container

VectorFloat gravity; // [x, y, z] gravity vector

float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector

char last\_action;

int flip\_array=8;

bool flip[8];

bool led\_state;

int flip\_cnt=0;

volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin has gone high

float forward\_Detect,Turn\_Detect,cw\_Detect;

int result;//processing接收手勢感測的結果

**設定指令內容**

const char\* Emergency="N1";

const char\* ChangeDiandi="N2";

const char\* Schedule="P1";

const char\* forward="1";

const char\* backward="2";

const char\* right="3";

const char\* left="4";

const char\* Right\_turn="5";

const char\* left\_turn="6";

const char\* Rest="0";

### 設定姿態角度

const int forward\_angle=30;

const int backward\_angle=-30;

const int right\_angle=30;

const int left\_angle=-30;

const int Right\_turn\_angle=30;

const int left\_turn\_angle=-30;

const int transfer\_speed=200;// unit mS

**初始化**

void setup() {

// join I2C bus (I2Cdev library doesn't do this automatically)

#if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE

Wire.begin();

Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties

#elif I2CDEV\_IMPLEMENTATION == I2CDEV\_BUILTIN\_FASTWIRE

Fastwire::setup(400, true);

#endif

bluetooth.begin(9600);

Serial.begin(115200);

while (!Serial); // wait for Leonardo enumeration, others continue immediately

//Serial.println(F("Initializing I2C devices..."));

mpu.initialize();

pinMode(INTERRUPT\_PIN, INPUT);

pinMode(button, INPUT);

pinMode(led, OUTPUT);

// verify connection

//Serial.println(F("Testing device connections..."));

//Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));

//read\_button();//------------------------------------------------------------------

//Serial.println(F("Initializing DMP..."));

devStatus = mpu.dmpInitialize();

//mpu.setRate(5);

// supply your own gyro offsets here, scaled for min sensitivity

/\* mpu.setXGyroOffset(220);

mpu.setYGyroOffset(76);

mpu.setZGyroOffset(-85);

mpu.setZAccelOffset(1788); // 1688 factory default for my test chip\*/

// make sure it worked (returns 0 if so)

if (devStatus == 0) {

// Calibration Time: generate offsets and calibrate our MPU6050

mpu.CalibrateAccel(6);

mpu.CalibrateGyro(6);

mpu.PrintActiveOffsets();

// turn on the DMP, now that it's ready

//Serial.println(F("Enabling DMP..."));

mpu.setDMPEnabled(true);

// enable Arduino interrupt detection

//Serial.print(F("Enabling interrupt detection (Arduino external interrupt "));

//Serial.print(digitalPinToInterrupt(INTERRUPT\_PIN));

//Serial.println(F(")..."));

attachInterrupt(digitalPinToInterrupt(INTERRUPT\_PIN), dmpDataReady, RISING);

mpuIntStatus = mpu.getIntStatus();

// set our DMP Ready flag so the main loop() function knows it's okay to use it

//Serial.println(F("DMP ready! Waiting for first interrupt..."));

dmpReady = true;

// get expected DMP packet size for later comparison

packetSize = mpu.dmpGetFIFOPacketSize();

} else {

// ERROR!

// 1 = initial memory load failed

// 2 = DMP configuration updates failed

// (if it's going to break, usually the code will be 1)

//Serial.print(F("DMP Initialization failed (code "));

//Serial.print(devStatus);

//Serial.println(F(")"));

digitalWrite(led,led\_state=false);

}

gestureTime=millis();

}

### 迴圈

void loop()

{

read\_mpu6050();

if((millis()-gestureTime)>=5000){

bluetooth.print(result);

gestureTime=millis();

}

}

**LED燈控制**

void read\_button()

{

unsigned long led\_Time;

float led\_duration;

Serial.println("wait button.....");

while(!digitalRead(button)) //等待按按鈕，此狀態下，LED會0.5秒閃爍一次

{

led\_duration = (millis() - led\_Time);

if(led\_duration>=500)

{

digitalWrite(led,led\_state=!led\_state);

led\_Time=millis();

}

}

while(digitalRead(button));

digitalWrite(led,led\_state=false);

Serial.println("一秒後校準");

delay(1000);

}

**手勢判斷**

void gesture()

{

duration = (millis() - Time);

if(duration>transfer\_speed)

{

digitalWrite(led,led\_state=!led\_state);

/\*Serial.print("ypr\t");

Serial.print(cw\_Detect);

Serial.print("\t");

Serial.print(Turn\_Detect);

Serial.print("\t");

Serial.print(forward\_Detect);

Serial.print("\t");\*/

int movement=0;

Time=millis();

move\_flip();

//判斷是否有旋(YAW)的動作//

if(cw\_Detect>=Right\_turn\_angle) //旋轉角度>=30(預設右旋)

{

//bluetooth.print(Right\_turn);

//Serial.println("無指令RT");

}

else if(cw\_Detect<=left\_turn\_angle) //旋轉角度<=-30(預設左旋)

{

if(forward\_Detect>=forward\_angle) //向前角度>=30(預設)

{

//bluetooth.print(Schedule);

//Serial.println(3);//您今天的行程是...

result=3;//查看行程

}

else

{

//bluetooth.print(left\_turn);

// Serial.println("無指令LT");

}

}

else movement++;

//判斷是否有左右傾斜(ROLL)的動作//

if(movement==1)

{

flip[flip\_array-1]=1;

if(Turn\_Detect>=right\_angle) //左右傾斜角度>=30(預設右傾斜)

{

//bluetooth.print(right);

last\_action='3';

//Serial.println("無指令R");

}

else if(Turn\_Detect<=left\_angle) //左右傾斜角度<=-30(預設左傾斜)

{

//bluetooth.print(left);

last\_action='4';

// Serial.println("無指令L");

}

else movement++;

}

//判斷是否有前後傾斜(PITCH)的動作//

if(movement==2)

{

flip[flip\_array-1]=1;

if(forward\_Detect>=forward\_angle) //前後傾斜角度>=30(預設向前)

{

//bluetooth.print(forward);

last\_action='1';

//Serial.print(5);

//result=5;//僅有往前

}

else if(forward\_Detect<=backward\_angle) //前後傾斜角度<=-30(預設向後)

{

//bluetooth.print(backward);

//Serial.println("無指令B");

}

else movement++;

}

if(movement>=3)

{

flip\_cnt=0; //次數

flip[flip\_array-1]=0; //第7位為0

for(int cnt=0;cnt<=flip\_array-2;cnt++) //檢查第0~6位 @@為什麼不是檢查0~7? >>因為第6位已經跟第7位檢查到了，而且第七位沒有下一位

{

if(flip[cnt]!=flip[cnt+1]) //若相鄰兩位不相同

flip\_cnt++; //則次數加一

}

if(flip\_cnt>=4) //當次數>=4，且不需要連續，因為人難以做到要求的動作(電腦判斷間隔太短)

{

switch(last\_action)

{

case '1':

//bluetooth.print(ChangeDiandi);

//Serial.print(2);//更換點滴

result=2;//更換點滴

break;

case '3':

// bluetooth.print(Emergency);

//Serial.print(1);//緊急求救

result=1;//緊急求救

break;

case '4':

//bluetooth.print(Emergency);

//Serial.print(1);//緊急求救

result=1;//緊急求救

break;

} //3&4 -->> 因為以上面的判斷標準來看，好像是只看最後一個動作來分辨翻滾的方向，而不是原本的連續兩次相同方向的要求，所以為了方便記憶，就將緊急求救設定為「將手左右甩動」

}

else

{

//bluetooth.print(Rest);

//Serial.print(4);

//result=4;//原點

}

}

}

}s[2] = 2;

}

else

{

//bluetooth.print(left\_turn);

// Serial.println("無指令LT");

}

}

else movement++;

//判斷是否有左右傾斜(ROLL)的動作//

if(movement==1)

{

flip[flip\_array-1]=1;

if(Turn\_Detect>=right\_angle) //左右傾斜角度>=30(預設右傾斜)

{

//bluetooth.print(right);

last\_action='3';

//Serial.println("無指令R");

}

else if(Turn\_Detect<=left\_angle) //左右傾斜角度<=-30(預設左傾斜)

{

//bluetooth.print(left);

last\_action='4';

// Serial.println("無指令L");

}

else movement++;

}

//判斷是否有前後傾斜(PITCH)的動作//

if(movement==2)

{

flip[flip\_array-1]=1;

if(forward\_Detect>=forward\_angle) //前後傾斜角度>=30(預設向前)

{

//bluetooth.print(forward);

last\_action='1';

//Serial.print(5);

//result=5;//僅有往前

}

else if(forward\_Detect<=backward\_angle) //前後傾斜角度<=-30(預設向後)

{

//bluetooth.print(backward);

//Serial.println("無指令B");

}

else movement++;

}

if(movement>=3)

{

flip\_cnt=0; //次數

flip[flip\_array-1]=0; //第7位為0

for(int cnt=0;cnt<=flip\_array-2;cnt++) //檢查第0~6位 @@為什麼不是檢查0~7? >>因為第6位已經跟第7位檢查到了，而且第七位沒有下一位

{

if(flip[cnt]!=flip[cnt+1]) //若相鄰兩位不相同

flip\_cnt++; //則次數加一

}

if(flip\_cnt>=4) //當次數>=4，且不需要連續，因為人難以做到要求的動作(電腦判斷間隔太短)

{

switch(last\_action)

{

case '1':

//bluetooth.print(ChangeDiandi);

//Serial.print(2);//更換點滴

result=2;//更換點滴

break;

case '3':

// bluetooth.print(Emergency);

//Serial.print(1);//緊急求救

result=1;//緊急求救

break;

case '4':

//bluetooth.print(Emergency);

//Serial.print(1);//緊急求救

result=1;//緊急求救

break;

} //3&4 -->> 因為以上面的判斷標準來看，好像是只看最後一個動作來分辨翻滾的方向，而不是原本的連續兩次相同方向的要求，所以為了方便記憶，就將緊急求救設定為「將手左右甩動」

}

else

{

//bluetooth.print(Rest);

//Serial.print(4);

//result=4;//原點

}

}

}

}s[2] = 2;

//判斷是否有前後傾斜(PITCH)的動作//

if(movement==2)

{

flip[flip\_array-1]=1;

if(forward\_Detect>=forward\_angle) //前後傾斜角度>=30(預設向前)

{

//bluetooth.print(forward);

last\_action='1';

//Serial.print(5);

//result=5;//僅有往前

}

else if(forward\_Detect<=backward\_angle) //前後傾斜角度<=-30(預設向後)

{

//bluetooth.print(backward);

//Serial.println("無指令B");

}

else movement++;

}

if(movement>=3)

{

flip\_cnt=0; //次數

flip[flip\_array-1]=0; //第7位為0

for(int cnt=0;cnt<=flip\_array-2;cnt++) //檢查第0~6位 @@為什麼不是檢查0~7? >>因為第6位已經跟第7位檢查到了，而且第7位沒有下一位

{

if(flip[cnt]!=flip[cnt+1]) //若相鄰兩位不相同

flip\_cnt++; //則次數加一

}

if(flip\_cnt>=4) //當次數>=4，且不需要連續，因為人難以做到要求的動作(電腦判斷間隔太短)

{

switch(last\_action)

{

case '1':

//bluetooth.print(ChangeDiandi);

//Serial.print(2);//更換點滴

result=2;//更換點滴

break;

case '3':

// bluetooth.print(Emergency);

//Serial.print(1);//緊急求救

result=1;//緊急求救

break;

case '4':

//bluetooth.print(Emergency);

//Serial.print(1);//緊急求救

result=1;//緊急求救

break;

} //3&4 -->> 因為以上面的判斷標準來看，好像是只看最後一個動作來分辨翻滾的方向，而不是原本的連續兩次相同方向的要求，所以為了方便記憶，就將緊急求救設定為「將手左右甩動」

}

else

{

//bluetooth.print(Rest);

//Serial.print(4);

//result=4;//原點

}

}

}

}s[2] = 2;

if(flip[cnt]!=flip[cnt+1]) //若相鄰兩位不相同

flip\_cnt++; //則次數加一

}

if(flip\_cnt>=4) //當次數>=4，且不需要連續，因為人難以做到要求的動作(電腦判斷間隔太短)

{

switch(last\_action)

{

case '1':

//bluetooth.print(ChangeDiandi);

//Serial.print(2);//更換點滴

result=2;//更換點滴

break;

case '3':

// bluetooth.print(Emergency);

//Serial.print(1);//緊急求救

result=1;//緊急求救

break;

case '4':

//bluetooth.print(Emergency);

//Serial.print(1);//緊急求救

result=1;//緊急求救

break;

} //3&4 -->> 因為以上面的判斷標準來看，好像是只看最後一個動作來分辨翻滾的方向，而不是原本的連續兩次相同方向的要求，所以為了方便記憶，就將緊急求救設定為「將手左右甩動」

}

else

{

//bluetooth.print(Rest);

//Serial.print(4);

//result=4;//原點

}

}

}

}s[2] = 2;

//bluetooth.print(Rest);

//Serial.print(4);

//result=4;//原點

}

}

}

}s[2] = 2;

**尤拉角公式**

void read\_mpu6050()

{

if (!dmpReady) return;

if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet

mpu.dmpGetQuaternion(&q, fifoBuffer);

Turn\_Detect=asin(2\*q.x\*q.z-2\*q.w\*q.y)\*57.3;

forward\_Detect=atan2(2\*q.y\*q.z+2\*q.w\*q.x,-2\*q.x\*q.x-2\*q.y\*q.y+1)\*57.3;

cw\_Detect=atan2(2\*(q.x\*q.y+q.w\*q.z),q.w\*q.w+q.x\*q.x-q.y\*q.y-q.z\*q.z)\*57.3\*-1;

gesture();

}

}