國 立 台 灣 科 技 大 學

電 機 工 程 系

___105__學 年 度 實 務 專 題 技 術 報 告

專題名稱：COS(Convenient Order System)

專題編號：

研究組員

楊書維　　許名宏

組別：_____　　班別：四電機四乙

指導老師：_____陳雅淑_____教授

中華民國　　106 年　　6 月

目錄

第一章　緒論.....3
1.1 摘要.....3
1.2 功能與使用情境.....4
第二章　系統平台.....5
2.1 硬體.....5
2.1.1 行動裝置(Android 系統).....5
2.1.2 Arduino.....5
2.1.3 壓力感測器.....6
2.2 軟體.....7
2.2.1 Android Studio.....7
2.2.2 Eclipse.....8
2.2.3 Visual studio.....8
2.2.4 Ispy.....8
2.2.5 Wamp Server.....9
第三章　系統實作.....10
3.1 系統使用流程.....10
3.2 演算法.....11
3.2.1 排程演算法.....11
3.2.2 人潮演算法.....13
3.3 TCP/IP 通訊協定、Android Volley、時間預估.....14
3.3.1 TCP/IP 通訊協定.....14
3.3.2 Android Volley.....15
3.3.3 時間預估.....15
第四章　系統實作成果.....16
4.1 使用流程.....16
4.2 APP 使用介面.....17
第五章　結論與未來展望.....19
第六章　參考資料.....20

圖目錄

圖二- 1 Arduino Uno 6
圖二- 2 數位 I/O 接腳 6
圖二- 3 壓力感測器 7
圖二- 4 壓力感測器實體 7
圖三- 1 系統架構圖 10
圖三- 2 第一階段邏輯圖 12
圖三- 3 第二階段邏輯圖 13
圖四- 1 實作流程圖 16
圖四- 2 APP 使用介面 17

第一章 緒論

1.1 摘要

到餐廳點餐對外食族而言是很平常的一件事，每當到了用餐時段，用餐的地方總是大排長龍，從排隊到取餐，一份餐往往都要花上不少時間才能拿到，「排隊」這個動作在許多人眼裡是稀鬆平常的，不過，其實「排隊」是可以不必要的，舉例來說若能在遠端即能得知各項餐點資訊、人潮擁擠程度以及能做到點餐動作，當餐點完成後亦能遠端得知再前去拿取，因排隊而耗費的時間就能夠省下來去做其他事。

通常，我們在排隊時，客戶總是面對著單一窗口，在處理接收點餐訊息的效率上往往需等待前一位客戶完成後才論到下一位，而廚房所接收到餐點的順序也不一致，這互相的作用下就耗費了許多時間，在許多餐廳也都是以有限服務生，透過固定數量電腦和廚房做聯繫來為顧客做點餐，再將訂單送進廚房，在餐點的處理上都是以先進先出的方式來處理餐點，若能在這方面做些安排，讓廚房在接收餐點上以較有效率的方式處理餐點，必能大大降低客戶排隊的情況。

因此我們將設計出一套點餐系統，能將餐點做有效的安排以降低等待時間，這套點餐系統以客戶端為出發點，客戶除了能透過系統看到店家餐點作下單動作外，也能透過系統了解餐點處理情形、事先得知目前的點餐數量的狀況以及人潮擁塞情況來作為判斷如何點餐較為省時，以提升點餐的效率，並經由雲端給予數據來分析候餐時間、餐點排程處理以及叫號狀況，再發送回客戶端裝置來告知顧客，讓使用者能在系統預想的時間內取得餐點，而不再是到隊伍裡排隊面對長長人龍。

1.2 功能與使用情境

到了用餐時段，時間限制只有一小時半或是更少，之後可能要上課或是上班，不可能等到人潮散去再前去買餐，只能在不了解餐廳狀況的情形下硬著頭皮到餐廳人擠人，到了餐廳無論是要找位置或是點餐都需要尋找與等待，當真正開始用餐時，已經耗費了不少時間。

這時能透過裝置在遠端事先得知餐點資訊、人潮情形以及作遙控點餐，不必到現場人擠人，舒舒服服地坐在位置上挑選自己想吃什麼，完成點餐動作後也會有時間預估，供客戶做參考，接著在餐點準備完成後，客戶能透過裝置獲得通知，相對於一般去櫃檯點餐來說效率及時間上方便了許多，而在餐點進行中，客戶也能透過裝置追蹤餐點的完成進度，得知目前自己的餐點進度。

而在點餐之餘，若對餐點有疑惑，尤其現在食安問題層出不窮，也能透過裝置連上食材平台查看店家的餐點資訊，包含內容和食材資訊，讓客戶亦能吃得安心。

第二章 系統平台

2.1 硬體

2.1.1 行動裝置(Android 系統)

	Android	iOS
作業系統	Linux	OS X, UNIX
開發硬體需求	Android 手機以及 電腦(作業系統不限制)	iOS 手機以及 mac 電腦
開發語言	Java、c#	Object-c、c#
介面設計靈活度	大	中
硬體尺寸種類	數萬種	六、七種
發布至商店	25 美金/終身	99 美金/年

由於 android 在開發上的成本考量遠低於 ios 系統，在軟硬體開發上我們選擇了 google 的 android 系統作為硬體及軟體的開發環境，加上開放的資源也較為豐富，故對於開發者而言，是入門的選擇。[1][2]

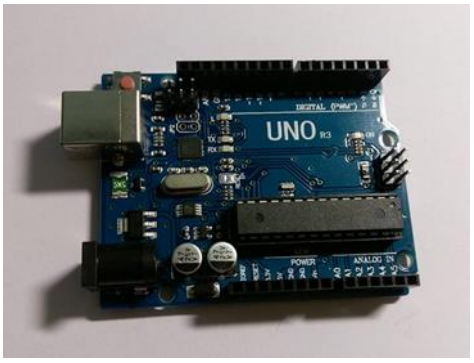
2.1.2 Arduino

Arduino 是一個開放原始碼的單晶片微控制器，為目前很常見的開發板，在編寫上不只限於本身的 IDE 還可以與其它的軟體結合例如：Java、Visual Studio ，此外只要結合其他的電子元件，便可以成為擁有許多強大的功能互動作品，此外 Arduino 也可以成為一個可以與軟體溝通的獨立運作介面。

Arduino IDE 提供開發者一個非常方便的編寫平台，除了使用類似 C 語言的簡單語法還提供了強大的 library，讓使用者能方便進行創意的實現、並且避免許多硬體開發上的麻煩。

1. 使用硬體介紹 –Arduino UNO [3]

微控制器	ATmega328P
工作電壓	5V
輸入電壓	7-12V
輸入電壓	6-20V
數字 I/O 接腳	14(含 6 個 PWM 輸出)
模擬輸入接腳	6
EEPROM	1KB (ATmega328P)
Clock Speed	16MHz



圖二- 1 Arduino Uno

2. 數位 I/O 接腳

Serial	接腳 0(RX)和 1(TX)透過這兩支接腳就能夠傳輸與接收 TTL 訊號的 資料，也是在 Arduino 使用上非常重要的兩個接腳。
LED	第 13 腳。此為 Arduino 內建的 LED 燈，透過程式的編寫便可以控制其明滅

圖二- 2 數位 I/O 接腳

3. 類比輸入接腳

Arduino 除了有數位的 I/O 接腳外，還有類比輸入腳為 A0~A5，當街上不同電子元件後，每支腳都提供不同的數值。在使用上參考電壓為 0~5V 但都可以透過 AREF 腳與 analogReference()進行更改。

2.1.3 壓力感測器

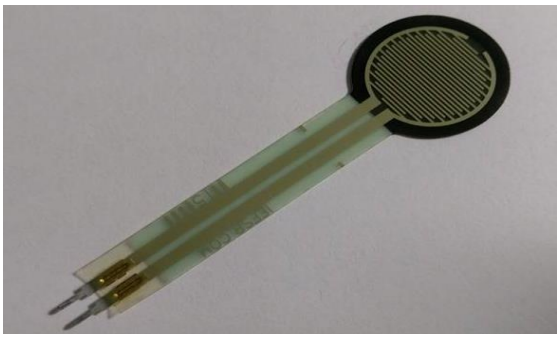
壓力感測器一種將被測件上的應變變化轉換成為一種電信號的敏感器件。金屬電阻應變片又有絲狀應變片和金屬箔狀應變片兩種。通常是將應變片通過特殊的粘和劑緊密的粘合在產生力學應變基體上，當基體受力發生應力變化時，電阻應變片也一起產生形變使應變片的阻值發生改變，從而使加在電阻上的電壓發生變化。這種應變片在受力時產生的阻值變化通常較小，一般這種應變片都組成應變電橋，並通過後續的儀錶放大器進行放大，再傳輸給處理電路（通常是 A/D

轉換和 CPU）顯示或執行機構。[4]

1. 使用硬體介紹–壓力感測器[5]

型號	Interlink 402
操作力	20gf ~ 2.0kgf (0.04lbs ~ 4.5lbs)
感測器類型	扁平薄膜
感測器樣式／尺寸	圓形、有效區 12.7 mm (整體 18.28 mm)
工作溫度	-40°C ~ 85°C

圖二- 3 壓力感測器



圖二- 4 壓力感測器實體

2.2 軟體

2.2.1 Android Studio

Adroid的開發環境上我們有Eclipse和Android Studio可選擇，但在無論是介面設計、手機上的除錯以及套件支援，皆屬於Android Studio較具有靈活性，因此Android Studio成為我們的開發環境首選。它擁有強大編輯器並提供許多增強生產力的開發工具，它有許多特色諸如:[6]

- **支援多種不同行動裝置的App開發**
- **提供效能分析工具，可以檢測App記憶體使用量**
- **豐富的APP 版面編輯器並支援圖像化版面設計，並可預覽各種不同尺寸裝置UI顯示效果**
- **完整支援Google雲端平台，易於整合APP和Google雲端服務**
- **使用Gradle可以更彈性的針對不同裝置產生APKs**
- **立即更新正在除錯APP，不用再建立新的APK**

2.2.2 Eclipse

由於我們 app 是以 java 來做發開，故在建構伺服器時也以同樣語言來做建構，如此能減少在開發時能除錯及學習時間，而在選擇 Java 的開發環境中，有 Eclipse、NetBeans 以及 IntelliJ 等環境供選擇，我們選擇了 Eclispe，它本身只是一個框架平台，但是眾多外掛模組的支援，使得 Eclipse 擁有較佳的靈活性，所以選擇以 Eclipse 為框架開發自己的 IDE，以下介紹他各項特色：[7]

- 最初用於 java，但現在能支援多種語言，像是：C、C++、Python、PHP 和 Ruby 等語言
- 整合性的開發工具，能夠撰寫程式、建立專案、執行及除錯應用程式，並能相容多種作業系統(Windows、Linux、Solaris 等)。
- 有視覺化的外掛，可以用滑鼠拖放的方式建立 GUI 介面

2.2.3 Visual studio

Visual Studio 是用來建置 ASP.NET Web 應用程式、XML Web Services、桌面應用程式及行動應用程式的一套完整開發工具。Visual Basic、Visual C# 和 Visual C++ 都使用相同的整合式開發環境 (IDE)，如此一來便可以共用工具，並且可以簡化混合語言方案的建立程序。 其中本專題使用到的是 Visual C++ ，在 Visual C++ 中有諸多特色及優點。[8]

- 在編寫程式時頁面呈現簡單乾淨
- 有需多其他套件可加入使用 例如(opencv)
- 擁有強大的資料庫

2.2.4 Ispy

iSpy 是一套相當實用的自動監控錄影工具，它可以同時匯入、管理多個不同的攝影鏡頭、IP CAM 或其他影音輸入設備，讓我們用一台電腦就可以錄下多個不同攝影鏡頭拍下的畫面，它的特色如下。[9]

- 多鏡頭管理
- 支援視訊動態捕捉
- 音量監控
- 定時截圖

2.2.5 Wamp Server

資料庫 Server 選擇中我們選擇了 Wamp 來做開發，由於它包含了 Windows 系統、Apache 網頁伺服器、MySQL 資料庫、PHP 腳本，在使用及開發上都相當便利及直覺化，它提供了非常簡易的方法來下載 Apache、php 和 mysql 套件，免於需要個別下載的麻煩，另外一旦安裝完成，也不須做多餘的設定即可立即使用。以下為 wamp 各項特色：[10]

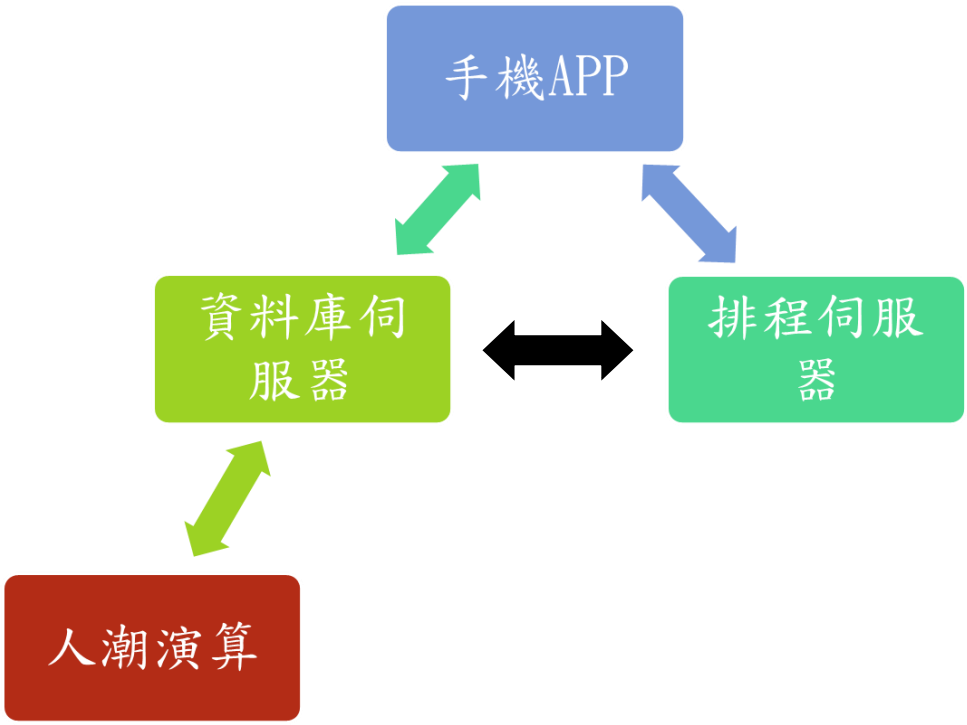
- 支援大多數 MySQL 的特點例如：
    1. 瀏覽以及刪除資料庫、資料表、欄位以及指標
    2. 支援各種 SQL 語法
    3. 管理 MySQL 使用者以及權限
- 匯出各種資料型態，例如：CSV、SQL、XML、PDF 等
- 管理多個伺服器

第三章　系統實作

3.1 系統使用流程

系統架構主要分為四個部分，**人潮演算伺服器**、**裝置應用程式**(以下稱APP)、**排程伺服器**以及**資料庫伺服器**，APP 是藉由 wi-fi 和資料庫伺服器以及排程伺服器之間連結做溝通並相互傳遞訊息，而人潮演算伺服器則是分析人潮數據後傳回至資料庫再由 APP 做讀取。

首先，客戶需註冊一組屬於自己的帳號密碼再進行其他功能，進入應用程式後能在畫面上先看到人潮狀況、時間預估以及其他功能項，點選人潮狀況文字即可做狀況更新，狀況分為鬆散、一般、擁擠、非常擁擠四種狀況，在進入點餐功能項後，裝置會連上資料庫讀取店家及餐點資料選擇並放入購物車，完成選擇後即進入結帳頁面，餐點送出後會跳至主畫面並在畫面上顯示預估時間，等候餐點完成期間能透過查詢訂單功能項查看自己目前餐點完成進度，以及透過餐點數量查詢查看各個店家的各項點餐數量，另外，能透過食材平台功選項與外部網站做連接並得知各項餐點內容，最後當餐點完成後，系統會透過推播訊息以震動和顯示在螢幕上及上方通知欄的方式來通知客戶前去領餐。



圖三- 1 系統架構圖

3.2 演算法

3.2.1 排程演算法

排程規劃方面我們分為兩階段來做安排，分別為第一階段的分類以及第二階段的分配，第一階段的排程是利用**排序的概念加上分類項目**達到目的[11]，將來自四面八方的餐點分配至指定的餐點陣列當中，並**將各個餐點陣列賦予優先權等級**[12]，當等級到達最高時即送至廚房做處理，其到達等級提高條件為該陣列達到固定數量、該陣列處於大於零但小於固定數量時間過長以及當有其他陣列送至廚房的次數到達固定數量，從這些條件**確保不會有餐點無法送出或遺漏以及達到將餐點順序條理化**，此種分類好處在於廚房接收到的餐點順序不再是以碎片狀的形式存在，而是能有一系列整齊且規劃過的形式。

第二階段的分配我們參考了**排隊理論(Queueing Theory)**的概念[13][14]，分配的順序從資料表當中的第一個 column 並從第一個 row 往下搜尋直到最後一個 row 再跳至下一個 column，我們將客戶拿到餐點的條件設為該餐點完成數量需大於客戶的被分配餐點，當完成數量大於零但不足以滿足該客戶時，第一步先繞過該客戶往下將餐點做分配，直到該 row 結束或分配餐點數量歸零，當該餐點做下一次分配時會由該位客戶開始優先做分配，直到滿足該客戶系統方會繼續往下做分配。

第一階段-分類

分類部分共分為三項 thread 持續進行[15]，分別為

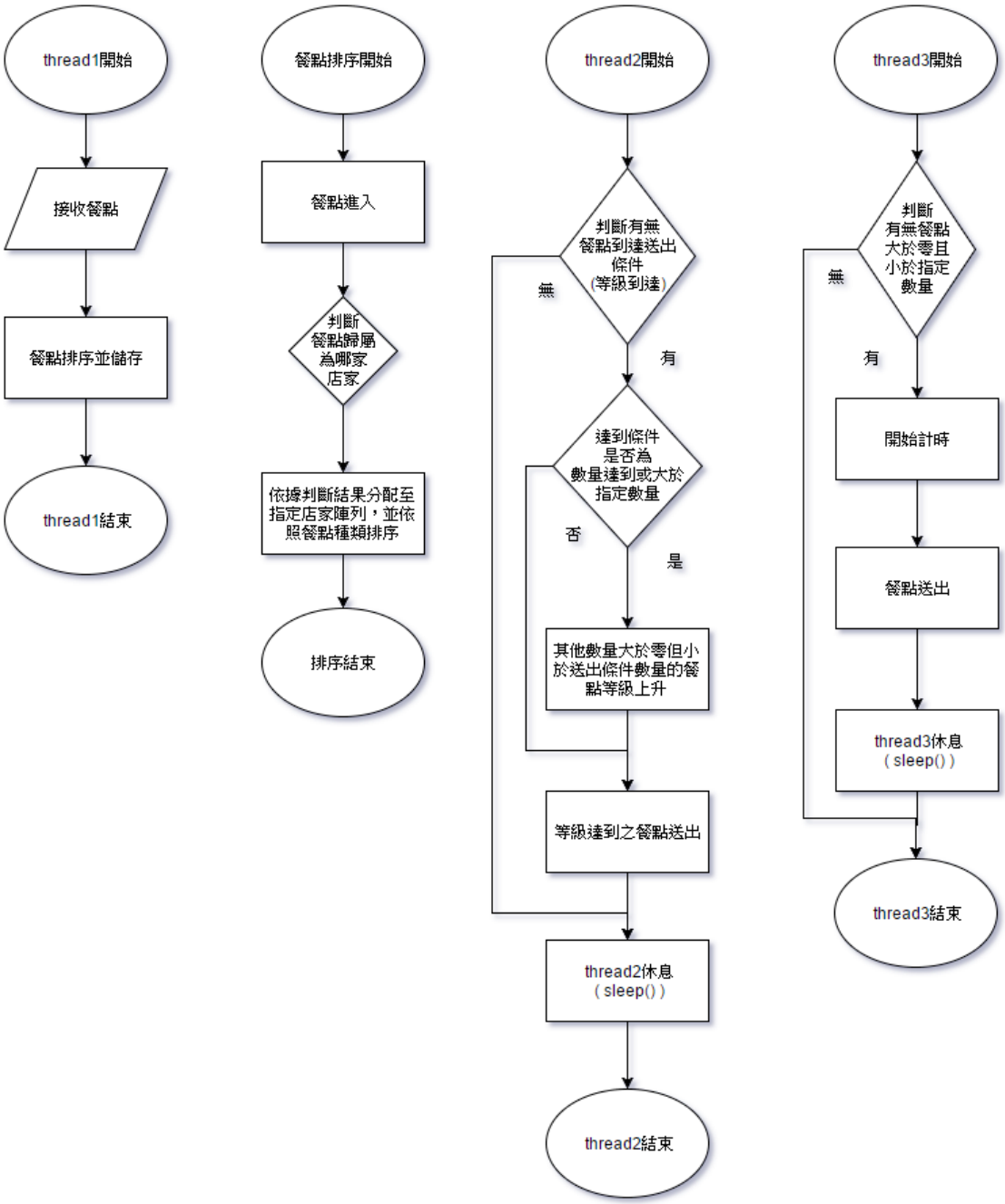
thread1：接收餐點(accept())並呼叫分類類別的 function(Arrange())做分類及排序動作。

thread2：檢視送出條件是否滿足，滿足送出條件後，將餐點做送出，其餘餐點陣列等級上升。

thread3：餐點滯留是否過久， 若過久則餐點等級上升。

藉由各個 thread 的交互使用，以達到點餐在接收上的效率提升以及餐點的順序條理化，另外，為避免硬體設備在 thread 的持續使用下會導致 cpu 被占用過多影響效能，除了接收餐點(thread1)的執行緒外，**其餘的 thread 皆會在結束時休息(sleep())一段時間，讓出空間給其餘的 thread 執行**。

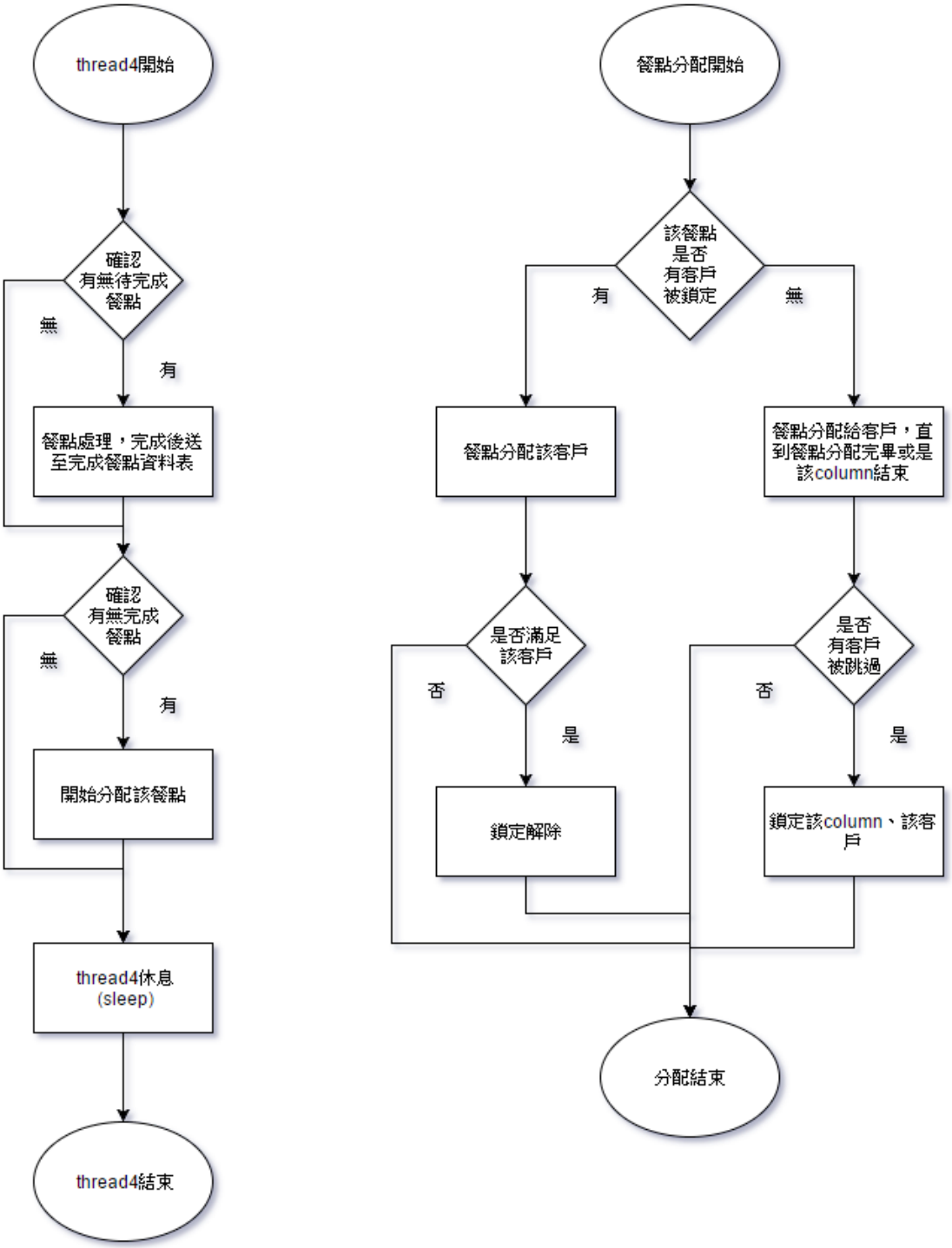
下圖為演算法流程圖：



圖三- 2 第一階段流程圖

第二階段-分配

此階段演算法一間店家以一個 thread 做處理，其中將為完成餐點做處理，完成後放進餐點完成表單中即為廚房做菜模擬，接著為分配工作，當該餐點分配結束時，若有客戶在這一次的分配中被跳過則**下一次做分配時會鎖定該 column 以及該客戶**，持續**對該客戶做分配直到滿足該客戶**方繼續往下做該餐點分配。



圖三- 3 第二階段流程圖

3.2.2 人潮演算法

在人潮演算法部分我們參照了兩個目前大型遊行最常使用的計人數方法，1. **雅各布斯法**[16] 2.**定點點算法**[16]，在設計上我們將整個餐廳分為入口，走道及座位部分。

1. 入口：

採用定點點算法，**以 1 分鐘得到人數密度，然後平均 5 分鐘人潮的密度**。在入口人潮是流動的因此變數較大，而透過攝影機可以抓取到五分鐘內還算精準的人數，再透過結合雅各布斯法就能得到 5 分鐘內入口部分人潮的密度。

2. 走道:

參照雅各布斯法，透過人臉偵測得到 1 分鐘單位面積人的數量，一樣是平均 5 分鐘的結果在走道的部分因為人潮屬於靜態的，可能因買餐點餐等位子…等情況而在走道停留因此採用靜態的雅各布斯法來收集人潮的密度。

3. 座位 SENSOR:

透過 SENSOR 得到目前座位佔有的數量，再透過和總數相除得到占用比例，這個部分直接顯示目前使用中的比例給使用者。

4. 標準參照雅各布斯法:

非常擁擠- 3.9 (人/M*2)以上
擁擠-3.9(人/M*2)以下 2.1 (人/M*2)以上
一般-2.1(人/M*2)以下 1.1 (人/M*2)以上
鬆散-1.1 (人/M*2)以上

最後**將得到的三個部分各自乘上一個修正係數後再進行一次的平均**，最後得到的數值再進行一個標準的檢測，就能得到一個目前餐廳人潮的狀況，在上傳到伺服器供給使用者最為參考的數據。

3.3 TCP/IP 通訊協定、Android Volley、時間預估

3.3.1 TCP/IP 通訊協定

應用程式與排程伺服器溝通我們採用 Java 本身提供的 ServerSocket 和 Socket 來進行連線的動作，但 ServerSocket 的接受連線方法 accept()會讓程式停頓(直到成功連線為止)，所以必須使用 Thread 來做執行[17]，而傳輸分為 Server 和 Client 兩部分的程式碼，這裡的排程伺服器是 Server 的角色來接收來自四面八方 Client 的連線訊息；Socket 是以 TCP/IP 通訊協定來做溝通，在這裡我們用於傳送點餐資訊，連線步驟如下：

1. 建立 TRY 和 CATCH 來包住整個網路運作，接收網路運作當中有例外狀況發生。

2. 連線建立: 建立 socket 類別物件(ServerSocket)並設置 Server 的 port
3. 開始連線(socket.accept())：不斷的等待並接受連線。
4. 以串流方式接收訊息，這裡用 DataInputStream 來接收的訊息(getInputStream())，並用內建函示 readUTF()來讀出訊息。

3.3.2 Android Volley

Android 在連上網路做溝通時，往往需要較複雜的步驟，若是沒有經過適當的封裝處理，很容易為寫出重複的程式碼，因此 GOOGLE 提出了一項新的解決辦法-Android Volley，用於處理屬於輕量級的網路溝通，除了一般的傳遞文字資訊外也可傳遞圖片,用法類似於網頁以 POST 和 GET 的方式做傳輸,以下為 Volley 的幾項優點：[18]

1. 能自行安排網路的 request
2. 能擁有多項網路連線
3. 能對特定範圍的 request 做取消請求
4. 能穩定、簡單且正確的從網路上取得資訊

藉由先建立一個發出需求的類別物件(ResquestQueue)，建立一個資料傳遞的物件(StringRequest)，在 StringRequest 中加入連線的 uri、從網路中取回的 jason 解析以及傳遞至 web 的值，來完成網路溝通。

若應用程式需要持續與網路做溝通，最有效率的方法即建立一個 RequestQueue，這個物件能夠持續運作直到應用程式結束，有很多方式來建立 RequestQueue，而最簡潔有力的方式即為建立一項屬於它的類別(class)以及其他 Volley 的函式，本專題利用此方法達到許多網路溝通，像是餐點查詢、訂單查詢以及推播等功能。

3.3.3 時間預估

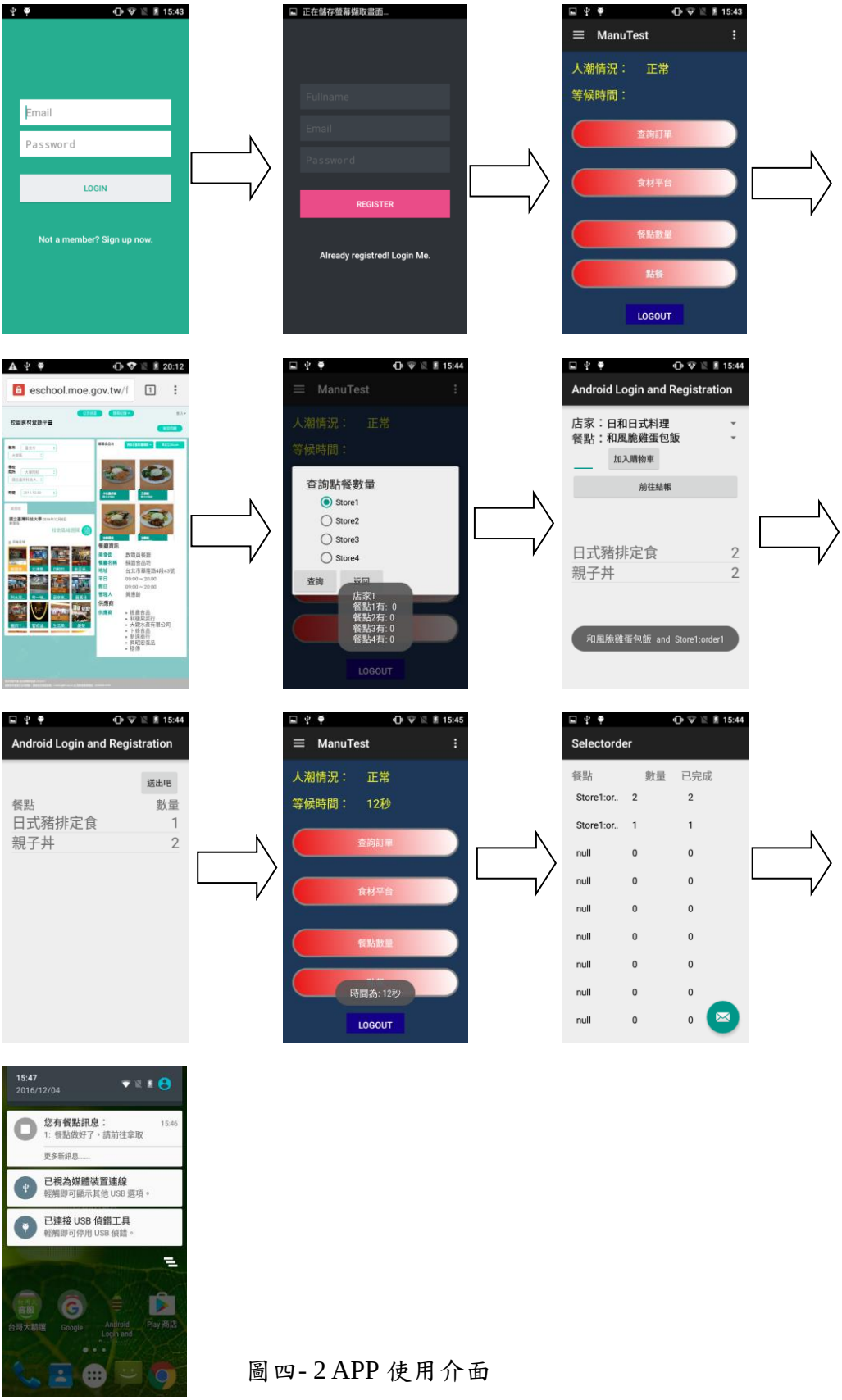
時間預估上，我們讓裝置自行做時間的估算，裝置中**各個店家的時間採用平均時間作為基準**，數據方面則由資料庫作提供，公式採用**最大時間**來做估算，送出餐點後裝置會分析出該客戶的餐點關聯了哪些店家,並**和資料庫做溝通讀回相關店家的餐點數量**，計算出各個店家的做餐時間找出最長等待時間，接著再**加上客戶本身的餐點作業最大時間**，即為時間預估。

第四章　系統實作成果

4.1 使用流程

圖四-1 實作流程圖

4.2 APP 使用介面



圖四- 2 APP 使用介面

如上頁圖四-2 所示，一開始會進入登入畫面，若為第一次使用則可進入註冊畫面做註冊，登入後即能看到人潮情況，可在食材平台選項連上食材平台網站做查看，以及在餐點查詢功能頁面中看到各個店家目前的餐點數量，在進入點餐頁面後，會有資料庫提供的店家以及餐點資料供選擇，加入購物車後會顯示在下方清單，若要做修改可點選下方清單做修改，完成選擇後進入送出頁面做最後確認並送出，接著會回到主畫面並會顯示等待時間，等待餐點期間可進入訂單查詢頁面查看餐點進度，餐點完成後，系統會推播至裝置通知客戶前來取餐。

第五章　結論與未來展望

物聯網時代發展至今，許多不同面向的應用持續的被開發出來，人類的生活也越來越趨向於 ALL IN ONE 的形式發展，讓原本需要靠經驗或是要到達現場才能得知的資訊只需在裝置上就能獲得，像是等公車 APP、捷運系統班次時間和 UBIKE 空位 APP(台北微笑單車)等，都是透過資料庫及演算法運用讓資訊相互做傳遞、運算以及儲存，並透過 APP 的方式呈現給使用者，讓使用者感受到其便利性，這同時也是物聯網的特性之一，以物聯網時代的思維來看，物聯網、雲端技術、大數據、行動裝置和社群媒體，是不可或缺的重點元素，同時也是驅動所有需求最關鍵的力量。

而我們的專題亦是利用物聯網特性，將原本需要前往櫃檯點餐的動作搬至雲端平台上運作，甚至藉由網路連線的橋樑事先提供給客戶許多必須到現場才會面臨到的情況，像是人潮、排隊情形以及候餐時間等，透過行動裝置讓每個人手上都能擁有一個像是小型櫃台隨時服務，也利用演算法將餐點的分類、分配做控管，使得整體的隊伍效率提升。

本專題的系統如果針對演算法、APP 使用者體驗以及食材平台做進一步的開發，將準確性以及實用型做提升，相信能在市場上做更多的應用，尤其是在台灣的人口密集，不只是餐廳還有賣場以及遊樂園等人潮眾多的大型地點，光是在等待上就耗費了許多時間，利用演算法能有效地降低這些時間以及達到疏通的效果，此外金融科技的話題也不斷浮上檯面，支付的方式不在侷限於現金或是傳統金融載具，系統 APP 上若能以 NFC 或是其他感應技術做到支付動作或是遠端付費，勢必能成為未來的革命性趨勢。

第六章　參考資料

[1] http://www.androidauthority.com/android-vs-ios-682005/

[2] http://www.androidauthority.com/developing-for-android-vs-ios-697304/

[3] Arduino　Arduino UNO & Genuino UNO From
https://www.arduino.cc/en/main/arduinoBoardUno

[4] 壓力感測器簡介: 工作原理
http://eshare.stust.edu.tw/EshareFile/2016_4/2016_4_7507878e.pdf

[5] FSR 402 Short From　http://www.interlinkelectronics.com/FSR402short.php

[6] https://developer.android.com/studio/intro/index.html#gradle_build_system

[7]
https://www.csee.umbc.edu/courses/undergraduate/341/fall08/Lectures/Eclipse/intro-to-eclipse.pdf

[8] Visual Studio 簡介 From
https://msdn.microsoft.com/zh-tw/library/fx6bk1f4(v=vs.100).aspx

[9] 錄影監控 iSpy v6.5.1.0 From　https://briian.com/7654/

[10] http://wamptutorials.blogspot.tw/

[11] https://en.wikipedia.org/wiki/Sorting_algorithm

[12] http://mail.tsu.edu.tw/~hjsin/courses/OS/Chap4.pdf

[13] http://episte.math.ntu.edu.tw/applications/ap_poisson/index.html

[14] http://consulteam.com.tw/Stanley/QueuingTheory/QueuingTheory.php

[15]
https://github.com/JustinSDK/JavaSE6Tutorial/blob/master/docs/CH15.md#151-%E5%9F%B7%E8%A1%8C%E7%B7%92%E5%85%A5%E9%96%80

[16] 點算人數方法系列 From
https://www.hkupop.hku.hk/chinese/columns/columns48.html

[17] http://www.nex1.com.tw/old_version/document/paper17.htm

[18] https://developer.android.com/training/volley/index.html