國 立 交 通 大 學

電信工程研究所

碩 士 論 文

Android 平台上之位置感知

行事曆提醒系統

A Location-aware Calendar Reminder System

Based on Android Platform

研究生： 陳柏丞

指導教授： 林亭佑博士

中 華 民 國 一零一 年 十二 月 四 日

Android 平台上之位置感知行事曆提醒系統

A Location-aware Calendar Reminder System Based on Android Platform

研究生： 陳柏丞 Student： Po-Cheng Chen

指導教授： 林亭佑博士 Advisor： Ting-Yu Lin

國立交通大學

電信工程研究所

碩士班論文

A Thesis
Submitted to Institute of Communications Engineering
College of Electrical Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
In

Communications Engineering

December 2012

Hsinchu, Taiwan, Republic of China

中 華 民 國 一 零 一 年 十二 月

Android 平台上之
位置感知行事曆提醒系統

學生： 陳柏丞　　指導教授： 林亭佑博士
國立交通大學電信工程研究所碩士班

摘　要

一般普遍以時間為依據的行事曆提醒器，需要使用者在不考慮位置關係的情況下，輸入各個事件在特定的日期。因為內建 GPS 功能的智慧型手機，在市場上的流行普及不斷增長，我們發現一個有潛力的趨勢，就是將位置感知資訊併入智慧型手機的 APP。

在這篇論文，以 Android 平台為基礎，我們提出並實行一種位置感知的行事曆提醒器(LACR)APP。不像傳統的行事曆提醒器，我們的 APP 能讓事件或想要的項目在特定一段時間內而不是的固定日期被配置，所有被配置的事件都和某些地點有所關聯。無論何時，當使用者接近一個明確的地點，我們的 APP 將彈出提醒訊息，以通知使用者去辦理相關的事情或購買相關的項目。

此外，當使用者要去辦理多種事情或購買多種項目時，我們的 APP 中開發了一個演算法，用以找尋花費最小的 cost 且通過全部相關的地點且每個點只能通過一次的漢米爾頓路徑。由於漢米爾頓路徑是 NP-complete 的問題，因此在演算法中我們限制了搜索範圍的輸入大小，所以可在合理的執行時間內發現漢米爾頓路徑。

在我們的行事曆提醒 APP 中，藉由結合位置資訊和演算法的智能，我們的目的是使智慧型手機的使用者日常生活更加便利。根據各個使用者的回應，新釋出的 LACR APP 在軟體可靠性和提供便利性上，獲得很多正面的評價。

A Location-aware Calendar Reminder System Based on Android Platform

Student： Po-Cheng Chen Advisor： Ting-Yu Lin
Institute of Communications Engineering
National Chiao Tung University

ABSTRACT

Conventional time-based calendar reminders require users to specify each event tagged on a certain date without considering the location relationship. With the ever-growing popularity of GPS-enabled smartphones available on the market, we observe a promising trend of incorporating location awareness into smartphone applications (APP). In this thesis, we propose and implement a location-aware calendar reminder (LACR) APP based on the Android platform. Unlike traditional calendar reminders, our APP allows an event (or wanted item) to be configured across a certain period of time (instead of a fixed date). All configured events are associated with certain locations. Whenever a user approaches a specified location, our APP will pop up a reminder to notify him/her to perform the associated event (or purchase the associated item). Furthermore, when a user has multiple events/items to perform/purchase, we develop an algorithm in our APP for finding a minimum-cost Hamiltonian path to traverse all intended locations exactly once. Since the Hamiltonian problem is NP-complete, we limit the input size of searched graph in our algorithm, so that a Hamiltonian path can be discovered within reasonable execution time. Our goal is to facilitate daily lives of smartphone users by combining location information with algorithmic intelligence in our calendar reminder APP. According to various users' feedback, the newly-released LACR APP receives several positive appraisals in terms of software reliability and provided convenience.

誌　謝

能夠順利完成此篇論文，首先要感謝我的指導教授林亭佑博士。在我的碩士生涯中，老師悉心的教導並指點我研究的方向，讓我在這段日子中獲益匪淺，尤其是在我遇到對挫折時，多虧有老師的勉勵，讓我能夠渡過難關順利完成研究。

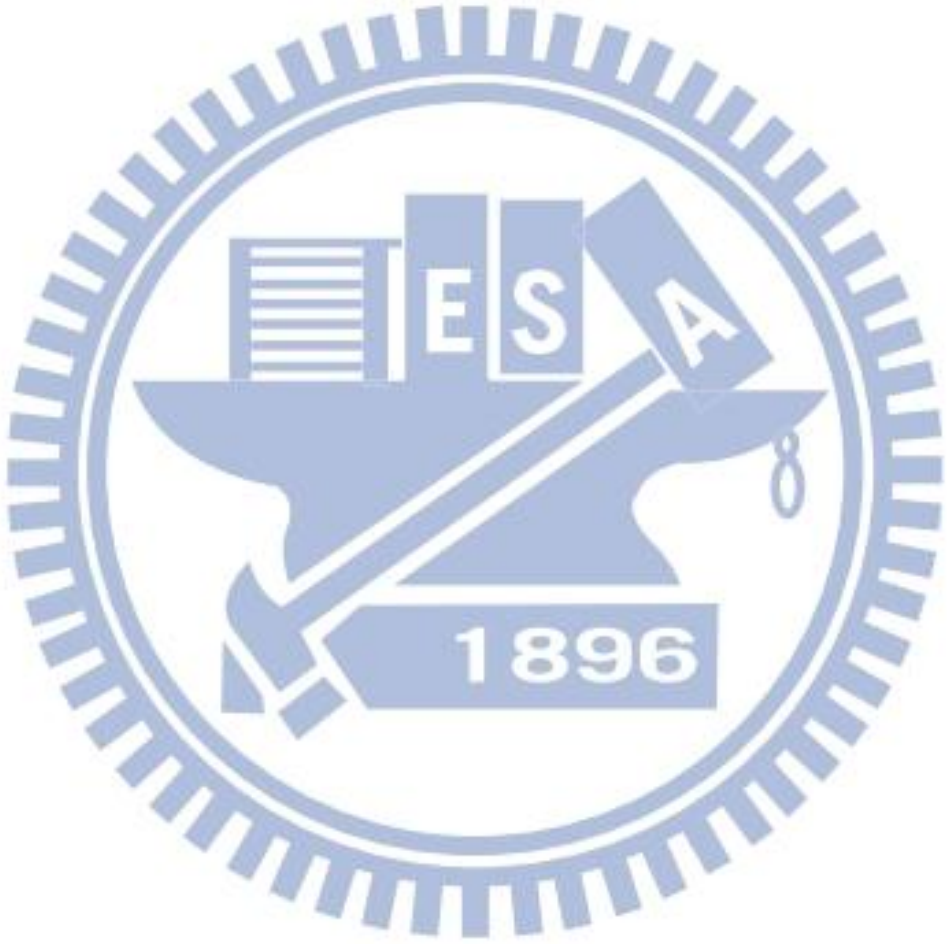
再來，感謝昆儒學長在完成此APP期間給予的協助，讓我的研究能夠順利進行。接著，感謝實驗室的同學們，嘉振、裕捷、嘉甫，在這段時間相互砥礪，一起學習、成長，讓我在研究路上不孤單寂寞。此外，也要感謝Bun Lab的所有成員，為我的碩士生活增添許多歡笑。

最後，感謝我的父母，感謝我的家人以及朋友，不斷的給我支持與鼓勵，讓我能夠有力量堅持在自己的研究上。感謝所有生命中幫助過我的人，因為你們才會有我。

誌於2012.12 新竹交大

柏丞

目錄



表目錄

圖目錄

第一章 緒論

每個人購物習慣都不相同，主要分為以下幾種購物方式：

(1) 寫下購買清單，安排時間到商家購物
(2) 想到就去購物
(3) 特地去逛街，想到什麼就買什麼

這樣對現今凡事都講求效率的社會來說，是非常浪費時間的，而且要特地出門採買非常的不方便。但由於現今社會是人手一隻智慧型手機時代，若我們能將問題透過手機的方式解決，就能改善這個問題。

現今智慧型手機幾乎有手機上網的功能，不管是使用 3G 還是 WIFI 的方式。因此有了這個特性，我們就能將資料放在雲端上，若手機這時發生連線中斷，可以設計同步的功能來解決此問題。

因此，我們透過手機開發應用程式，讓使用者透過介面設定將未來想要買的東西記下來，並利用使用者出門後(e.g. 上班或下班)，在使用者每天行經的路線上接近有販賣購物清單中品項之商家時，自動提醒告知，附近商家有販賣購物清單中之品項且可以前往商家購買。例如: 使用者出門後，行經的路線上剛好有在販賣購物列表清單中物品的商家，便會提醒使用者說附近有要買東西，讓使用者能節省時間。

本論文「Android 平台上之位置感知行事曆提醒系統」，主要的目的是因為內建GPS功能的智慧型手機在市場上越來越流行，我們發現一個有潛力的趨勢，就是將位置感知資訊併入智慧型手機的APP將使用者手機與商家做結合，只要是使用者之前設置的購物清單中，剛好附近有在販賣購物清單中品項的店家且使用者自訂的提醒時間區間符合今天的情況下，都透過開啟提醒的功能，來提醒使用者，建議使用者前往此商家購物。而且有個更方便的功能，若出現很多商家都賣一樣的東西，這時系統會將有在販賣此物品的商家之價格排列出來，讓使用者進行選擇要去哪間商家購物。還有一項很實用的功能，我們可以讓使用者去新增店家，不但能方便使用者輸入資料庫沒有的商家，而且未來能跟零售商合作，增進實用性。

另外有一創新功能，因為大部份手機的網路比較不穩定，無法連線時將造成無法使用資料庫的情況，因此我們將手機內建的資料庫與雲端資料庫做同步，來加強資料儲存的功能。

最後我們結合 Hamilton Path 演算法，來取得待去購物之商家的最佳路徑，該使用者能夠依此路徑，來解決需要繞路折返太花時間的問題。

1.1 本論文功能

本系統 LACR APP 開發之目的主要要達成以下之目標，以使本 LACR APP 的應用價值以及其便利性能夠明顯的凸顯與目前市面上行事曆提醒相關 APP 之不同以及優越之處。

本系統 APP 特性功能如下：

- 普通記事功能
- 新增店家
- 新增購物清單
- 自行設置欲提醒的時間範圍區間
- 獲取實際路徑的道路情況
- 根據實際道路情況進行商家最佳路徑規劃
- 離線資料庫
- 能將附近商家都有賣的東西，列出每家商家所販賣之價格
- 商品分類庫
- 震動提醒功能

- 顯示目前使用者與店家的地圖相對位置

1.2 與其他 APP 之比較

我們所開發之 LACR APP 與一般目前市面上所能看到的行事曆提醒 APP 比較起來有許多過人與不同之處，我們就以下表來詳細說明，讓讀者更加了解我們開發此系統的目的。

表 1.1 與其他 APP 之比較表

	本系統	其他 APP
行事曆	不但能指定一段時間區間和要買的東西，能夠當使用者接近商家或在此時間區間時，發出提醒	只能指定時間和買的東西
路徑規劃	不但能根據路況選擇店家，而且也能將所有要去的商家規劃出一條最佳路徑	只能規劃到商店的路徑
同步功能	提供本地和雲端同步功能	只能存取雲端，很少 APP 會用到本地加雲端功能
商品分類	提供完整的分類，能讓商品的廠商使用，也能自己新增店家	只提供讓使用者隨意填寫

第二章　系統架構

2.1 系統架構

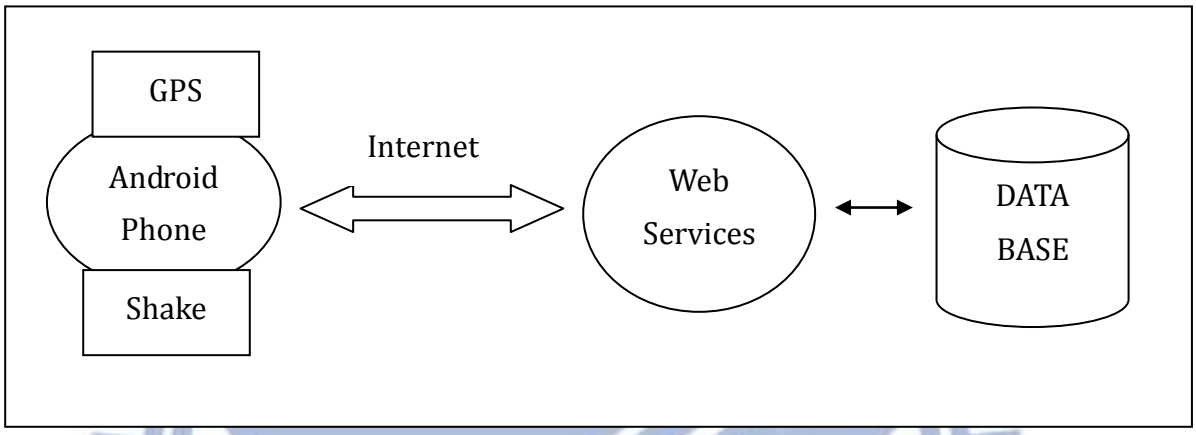


圖 2.1 系統架構圖

如圖 2.1 所示，本系統透過 Android 手機開發應用程式[13]，當使用者帶著手機出門，手機會一直接收 GPS 訊號，若這時使用者之前透過介面設置好購物清單要提醒的商品，剛好行經路線上有店家有賣(e.g. 手機透過 GPS 來知道使用者的位置[12]，然後在去比對資料庫的資料，最後透過距離演算法，來計算是否離使用者很近，若很近就可提醒使用者) ，且若時間也是在使用者所自訂的提醒範圍中，就控制 Shake 模組，讓它震動提醒。

另外，為了防止手機突然無法連上網路，本系統另有一項同步功能，讓手機在離線時，也能正常操作。若連絡恢復時，程式就自動同步某位使用者的資訊、所有商家跟購物資料。這樣一來就不怕會有斷線的情況，造成應用程式出錯。

2.2 Android Phone

Android[1]是由 Google 所開發出來的手機平台，它主要是由 Linux 和 KVM 所組成。Android 最令人嘉獎的地方是提供整合介面，讓開發者能在手機上撰寫自己的應用程式，也能應用在各個領域，像是 GPS 或是記帳等相關之應用。

由於智慧型手機越來越多人使用，滿多創新的應用程式也被開發，本論文是一個例子，若使用者都能透過 Android Phone 來使用本論文應用程式，就可省去很多時間且又可方便的購物。

Android 手機的設計是很人性化的地方，開發者只要透過 Android API 就能取得 vibrate 震動提醒的權限，並對手機下震動的指令。此功能對本論文來說很重要，因為有時若是在騎車或是開車，若有接近的店家，就可透過震動提醒使用者。

選擇最佳 Android 手機作為開發平台主要的原因是，只要學過 JAVA 的語言，不用再重新學習就能輕易使用，而且 Android 開發工具 SDK 可以跨平台，在任何作業系統下都能輕易安裝開發，與 Apple 的 iPhone 比較起來相對的會容易很多，比對如下表所示：

	Android	iPhone
開發公司	Google	Apple
開發工具	JAVA	Object C
市場佔有率	高	低
IDE	Eclipse + Android (cross-platform)	Xcode (only MAC OS)
APP Market	Google Play	App Store

表 2.1 Android 與 iPhone 比較表

另外 Android 開發方便的地方就是有大量的 API 能使用，例如：若需要震動讓使用者知道新訊息，開發者只要透過幾行程式就能做到，相當容易且具有強大的整合性。

因此本論文選擇使用 Android 作業環境來開發 APP[8]，未來透過 iPhone 也能利用相同觀念開發應用程式。

2.3 Web Services

Web Services[2]是一種網路服務，它是透過 Port 80 的方式，透過穿越 Port 來存取背後的資料庫，其目的就是要當作雲端，提供 Android 透過網路交換資料或備份之用。

我們舉一個 Web Services 最常看到的例子：假設我們要建立一個氣象網站，網站提供的服務包括了氣象資訊查詢等等，我們只要能找提供服務的 Web Services，然後將它們整合到網站中即可，不需要再花費時間和成本來建立自己的資料庫系統等。更重要的是，若透過 Web Services，不必擔心這些服務是如何建立的，我們只要將格式設置好，它自然就會回應我們要的資料，因此這些 Services 是很適合用在手機的雲端溝通上，只要手機能拿到 Services，就能很輕鬆的與雲端交換資料。

總結以上之優點，本論文決定透過架設 Web Services 來放置遠端的資料，使用的方法是用 HTTP 的 port 80 來存取資料庫的資料，並與手機端作同步的動作。

2.4 資料庫(Database)

本論文的實作資料庫採用的是 MS-SQL 2008，它優點分為三個：

1. 可信度很高且具有安全性、可靠性和可擴展性。
2. 高效能，可以降低開發和管理資料基礎設施的時間和成本。

本論文主要建了多個資料表來存取資料，以下將介紹所有資料表的 scheme 等特性。

```
CREATE TABLE `categoryinfo` (
   `ID` bigint(100) not null,
   `CategoryName` varchar(100) not null',
   `Remark01` varchar(100) Not null,
   `Remark02` varchar(100) Not null,
   `Remark03` varchar(100) Not null,
   `Remark04` varchar(100) Not null,
   `Remark05` varchar(100) Not null,
   `Remark06` varchar(100) Not null,
   `Remark07` varchar(100) Not null,
   `UserID` varchar(100) Not null,
   `UpdateTime` varchar(100) Not null
);
```

圖 2.2 分類資訊資料表

如圖 2.2 所示為本論文的分類資訊資料表，透過寫入分類資訊資料表，我們可以將每個商品分為最多七類。透過這種寫法，我們能讓使用者在找尋商品時更加方便，並也能記錄 Update Time 更新時間等資訊。

```
CREATE TABLE `noteinfo` (
   `ID` bigint(100) Not null,
   `Data` varchar(100) Not null,
   `Type` varchar(100) Not null,
   `Date` varchar(100) Not null,
);
```

圖 2.3 記事資料表

記事功能必須有自己的資料表，如圖 2.3 所示，為本論文所用的記事資料表。比較重要的欄位是去用 Data 記下所要提醒的項目，也用 Date 告知系統要提醒的時間。這裡提醒功能不包含是記事，還有購物提醒，本論文之應用與之前手機應用程式的功能，只能在單一時間的時間點做提醒，本論文透過定位驅動，來主動讀取這個資料表，取得要提醒的部份。

```
CREATE TABLE `storeinfo` (
  `ID` bigint(100) Not null,
  `StoreName` varchar(100) Not null,
  `City` varchar(100) Not null,
  `Address` varchar(100) Not null,
  `LAT` varchar(100) Not null,
  `LNG` varchar(100) Not null,
  `UserID` varchar(100) Not null,
  `UpdateTime` varchar(100) Not null,
  `length` varchar(100) Not null,
);
```

圖 2.4 商家資訊資料表

如圖 2.4 為商家資訊資料表，這個資料表記下所有商家的部份，並透過 Address、LAT 和 LNG 來取得現在地址，這個資訊主要是用來與手機現在位置的 GPS 做比對，若是接近，就查詢這家商家，是否有我們要賣的東西。另外 Update Time 對這個應用來說也很重要，我們可以知道店家更新日期，就能推測商家是否還在。另外 length 是本系統在推測距離之用。

第三章 功能架構

3.1 程式架構

本論文的程式架構可分為 Android 手機端和 Web Services 端。Android 手機端的主要功能是在顯示畫面、讀取 GPS 計算分析、提醒和資料庫存取等功能，它是整個應用程式主要的部份；Web Services 端主要是在存取遠端資料庫的功能，讓手機端能同步資料庫等等。

3.1.1 Android 手機端

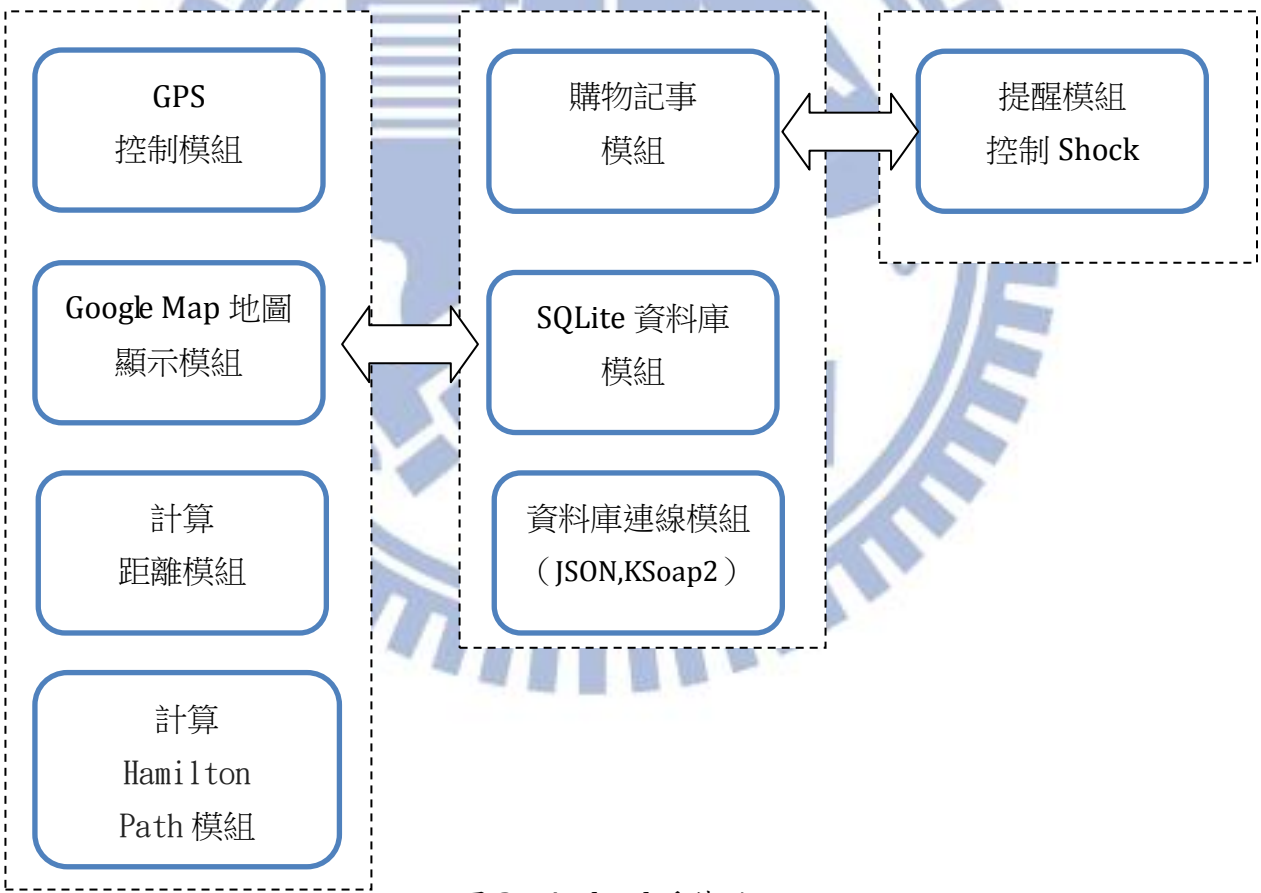


圖 3.1 Android 手機端

如圖 3.1 所示，此 Android 手機端主要分為三個項目：GPS 地圖相關模組、購物資料庫模組和控制震動模組，這裡的架構主要是由 Android 系統架構來進行模組化，比較重要的是 GPS 地圖相關模組、購物資料庫模組，因為它是整個應

用程式的核心，因此接下來的介紹，我們將著重在這兩個模組的部份。如下說明：

(1) 與 GPS 地圖相關模組可分為下列幾項子模組：

- GPS 控制模組

  接收 GPS 的訊號，將經緯度回傳給 GPS 地圖模組。

- Google Map 地圖顯示模組

  負責將 GPS 的點顯示在地圖上，並提供使用者查詢之功能，也能透過此模組標出商家的位置，同時也能做路徑規劃等應用。

- 計算距離模組

  計算商店和現在使用者位置的距離，本論文預設的距離為預估 300M 內，代表接近可以提醒使用者。因為每個人對距離的觀念不同，所以這個數值並非固定，可由使用者去調整。

- 計算 Hamilton Path 模組

  此為本論文應用上最創新的演算法，只要透過這個模組就能去抓取使用者現在位置，至所有商家的路線，為此來找尋一條最佳路徑。這條路徑最主要的目的是讓使用者把欲前往購物之商店在不需折返的情況之下都拜訪過一次，把之前在購物清單的物品做採買的動作。

(2) 購物資料庫模組可分為下列幾項子模組：

- 購物記事模組

  此模組能協助使用者，在某個自定的範圍時間內買購物清單上的東西。當使用者將購物清單設置好之後，只要等到使用者的手機 GPS 有更新時，系統就會讀取此模組，將所有相關的商家資料進行比對。若 GPS 偵測在附近，就可開啟控制震動提醒；若沒有則略過，直到有找到為止。另外這個模組很重要功能，就是將某個商品所有附近有賣的店家都列出，並把價格標示出來，由使用者選擇想去的商家。

- SQLite 資料庫模組

手機端本地的資料庫，存取包含購物記事資料、商品項目、商家等等。它確保兩邊與雲端資料庫同步，也會主動去檢查兩邊的資料是否相同。

- 資料庫連線模組

  與 Web Services 連線，透過 JSON 來交換格式資料，並存取最新的資料或進行上傳之動作，來讓資料同步。

(3) 提醒模組

可依購物記事模組的操作，來震動或發出聲音提醒使用者，已經接近有販賣購物清單上之品項的商家，現在可前往購買東西。這個功能主要是使用 Android SDK 提供的 API 來實作，只要對一個函數下動作，就會震動或發出聲音提醒使用者。

手機端的主要流程是透過 GPS 控制模組，能抓取最新目前的點座標，程式會主動去抓取資料庫中的商品，並自動比對列出符合商家的資料，並觸發提醒模組去震動手機或發出聲音來提醒使用者。接著就由使用者決定要不要前往，而且系統也提供貼心功能，能提供使用者一條最佳購物路徑，建議使用者按照這個路徑前往購物，就能節省更多時間。

同步功能也很重要，主要的技術是透過資料庫連線模組來與本地資料庫做存取，進行網路同步之動作，它主要透過 JSON 和 KSoap2 兩個 API。

JSON（JavaScript Object Notation）

JSON[3]（JavaScript Object Notation）是一種資料交換語言，是透過文字基礎，將屬性連結起來。會透過文字的方式連結，是因為可降低語系的問題，而且資料量相對也小很多，所以目前許多網路服務、API 都喜歡使用 JSON 格式的字串做為交換格式。

JSON 格式的資料以下列方式描述：

- 大括號（{）開始、大括號（}）結束
- 各屬性以逗號（，）分隔
- 每個屬性的名稱和值以冒號（：）分開

例如：{firstname:"Chen",lastname:"JJ",gender:"M",country:"Taiwan"}
其中值的部份可以有下列形式：

字串，"test"、陣列，[2,2,2,2] ，數字，39；

另一個格式，將屬性表示出來：

```
{"test":
   {"name": "JJ Chen", address":
     {"street": "5 Main Street city":
"Pingtung, TA", "zip": 91912,},
"Phone Numbers":
["08 7392806",
"08 7395393"]
     }
  }
}
```

以上面的例子屬性 name 值就是 JJ Chen，屬性 street 值就是 5 Main Street city 等等，它可以包含很多階層的方式，就能完整的表現資料庫的形式。JSON 優點是要求只需通過一個簡單的括號即可抓取到資料。若是使用傳統的 XML 標記將會比較花時間。

本系統是採用 JSON 來讀取 Web Service 來的資料，並用解析的方式，把資料讀出來，顯示在手機端。

KSoap2[4]

在 Android 平台調用 Web Services 需要使用 KSoap2，它是一個 SOAP Web services，我們是透過 KSoap2 Android 的 API 來實作。KSoap2 Android 是 Android 平台上一個高效、輕量級的 SOAP 開發包，等同於 Android 平台上 KSoap2 的移植版本。透過此 API，我們可以將 Web Services 所提供的格式，輕易的解析出來。

3.1.2 Web Services 端

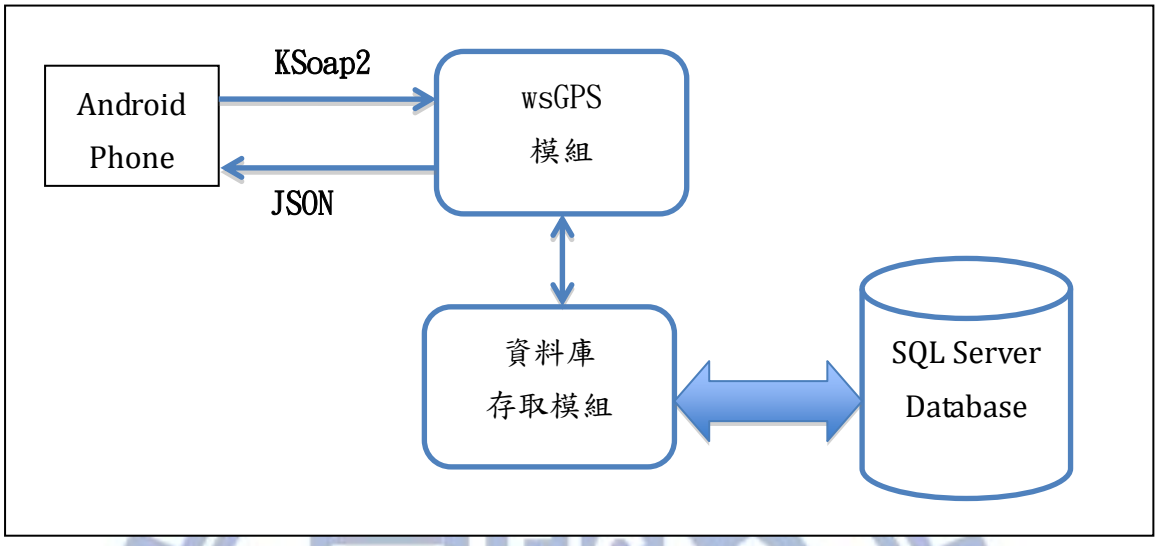


圖 3.2 Web Services 架構

如圖 3.2 所示，此 Web Services 端主要分為幾個項目：

(1) wsGPS 模組

處理由 port80 進來的網路資料，手機端程式會使用 KSoap2 通訊方式來求取資料，然後它會去問資料庫存取模組是否有對應的資料，若有的話會收集資料並轉換成 JSON 格式回傳給手機端。

(2) 資料庫存取模組

為負責存取資料庫的模組，不管是商品項目、商家資料等等，都能透過下 SQL 指令的方式與 SQL Server Database 作溝通。

(3) SQL Server Database

雲端的資料庫，它會與手機端的本地資料作同步之動作，以防到時使用者在別地方安裝本系統 APP，資料會無法同步。

本論文的 Web Services 流程是由手機端透過 Web Services 在 Port 80 利用 KSoap2 通訊，交由模組來做資料存取。基本上 Web Services 端還是主要在存取資料用，重要的運算和判斷都是在手機端作處理。

3.2 實作規格

我們的實作設備分成兩個部分，一個為手機端，另一個為 Server 端，如下表 3.1、3.2 所示：

手機	HTC
手機 OS	Android 2.2 以上版本
開發軟體	Eclipse + Android SDK
通訊方式	搭配 Google 的 JSON 與 KSoap2 與伺服器溝通
資料庫	SQLite

表 3.1 手機端實作規格

開發軟體	採用 Visual Studio2008 與 C#
OS	Windows XP
通訊方式	JSON 與 KSoap2 與伺服器溝通
資料庫	採用 MS-SQL2008

表 3.2 Server 端實作規格

我們使用 Android 手機來撰寫應用程式，然後透過 Eclipse 和 Android SDK 的 API 來開發，主要是用 JAVA 來撰寫，它透過裡面眾多的 API，能將我們 APP 所需要的功能輕易就實現，也能快速開發來驗證實驗的部份。手機的資料庫我們採用 SQLite，而網路連線我們採用 JSON 與 KSoap2 做法來傳送資料。

Server 端我們是採用個人電腦來架設，作業系統用 Windows XP。因為微軟在台灣企業的市占率極高，也不斷地更新漏洞。另外在安裝與維護時簡易且圖像化的操作介面，可以十分方便輕鬆的管理伺服器。此外透過 VS2008 和 MS-SQL 就能建立雲端資料庫，選用 MS-SQL 的理由是他有預儲程序的一個功能，而且用起來效能極佳，此外我們所建立帳戶的管理方法簡單且易用，另外當有支援事件觸發時，刪除關聯性的資料我們可以不用寫死，且可以利用觸發來寫邏輯交易的一個部分，最重要的是當伺服器無法動作的時候，MS-SQL 可以很容易的進行自動排程備份，所以可以縮短搶救的時間。進來的資料就由 KSoap2 做溝通來接收資料，並透過 C#把出去的資料轉換成 JSON 格式，因此就能跟手機端溝通，而達到同步的效果。

3.2.1 建置 Eclipse + Android SDK 環境[5]

透過下面的步驟，就能夠建置 Android 的環境，來讓開發者使用：

(1) 下載 Eclipse classic
(2) 下載 Java SDK(JDK)
(3) 下載 Android SDK
(4) 安裝 Eclipse 與 Android SDK
(5) 安裝 Eclipse ADT(Android Development Tools)
(6) 安裝後重新啟動
(7) 安裝完後，就能開始開發 Android APP

3.2.2 建置 Visual Studio 2008 + SQL Server 2008

由於我們 Web Services 的部份我們採.NET 的平台來實作，用的資料庫是 MS-SQL2008 的部份，而程式語言我們是採用 C#，必須安裝以下軟體：

(1) 安裝 Visual Studio 2008 專業版
(2) 安裝 SQL Server 2008
(3) 透過 lib 項目做連結

3.3 演算法說明

本系統主要由兩個演算法負責整個系統應用程式的運算，第一個為距離演算法，第二個為 Hamilton path 演算法，用於我們安排最佳路徑所使用，我們將會在以下章節詳細的說明此兩個主要的演算法。

3.3.1 距離演算法

由於地球為一個橢圓球體，所以經緯線就是在這個橢圓球體表面的地理座標參照系格網(Grid)。又因為我們的經緯度座標系是球面座標系，單位是度、分，所以我們要算兩點之間的距離時，並不能用度和分這些單位去計算長度。

因此我們需要透過公式將球面座標轉換成二維平面座標，這個轉換的方法就是投影法，以下的演算法就是透過投影法公式算出兩點的直線距離。

如圖 3.3 所示，將兩點相減用 toRadians 取得弧度再透過數學公式，就能取得兩點距離如下公式：

```
double dLat = Math.toRadians(lat2 - lat1);
double dLon = Math.toRadians(lng2 - lng1);

double a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
Math.cos(Math.toRadians(lat1))
                    * Math.cos(Math.toRadians(lat2)) * Math.sin(dLon / 2) *
Math.sin(dLon / 2);

double c = 2 * Math.asin(Math.sqrt(a));
EARTH_RADIUS= 6371.00; // Radius in Kilometers

return EARTH_RADIUS * c;
```

圖 3.3 距離演算法

透過以上的演算法，輸入兩點的 GPS 座標，就能算出兩點之間的距離。接近的定義是由是否小於某個距離數值來認定，若進入這個範圍且剛好商家有販賣使用者需要購買的東西時，就會由提醒模組來告知，而 EARTH_RADIUS 是個定值，設為 6371。

實際路徑選擇

由 3.3.1 可知目前兩點 GPS 座標之間的距離，但在實際的地圖上，最短並不代表是最佳路線，因為每條路線上都有一些因素，像是紅綠燈數量(等待的秒數)，或是這條路現在有沒有塞車之類的議題(時速)。

因此，本論文先以虛擬紅綠燈數量以及要等的秒數來考量，如圖 3.4 所示。它的想法是若有一距離最短，但紅綠燈秒數很長的路線，跟一個距離較長，但紅綠燈秒數很短的路線，距離長的到達時間和紅綠燈秒數相加，若與距離短的到達時間和紅綠燈秒數相加互相比較之後，假如距離長的那條路所花費的 Cost 最

小，我們就選擇這條路線，以下就是演算法的判斷：

```
//亂數產生紅綠燈個數和秒數
store.count_light = (int)(Math.random()*10);
store.light_sec = (int)(Math.random()*120);

//計算兩點之間紅綠燈秒數 + 到達時間(時速換算成秒數)
tsec = storeData.get(i).light_sec - storeData.get(j).light_sec;
length = storeData.get(i).length - storeData.get(j).length;

// avrspeed 現在平均時速
m_sec =    (Case05Activity.my.avrspeed    * 1000)/3600;
t1_sec = (storeData.get(i).length / m_sec) + storeData.get(i).light_sec;
t2_sec = (storeData.get(j).length / m_sec) + storeData.get(j).light_sec;

//選小的那個，代表比較快到達
if (t2_sec > t1_sec)
…
```

圖 3.4 實際路徑評估

判斷是否發出提醒

如圖 3.5 所示，由 3.3.1 算出目前使用者與商家的直線距離後，若使用者本身與商家的距離小於 300 公尺，就加入提醒清單，並透過使用者所設定的提醒模式做 trigger 的方式，若距離大於 300 公尺則什麼都不做。

```
store.length =
GpsFuns.GetDistance(lat, lng, Double.parseDouble(store.LAT),
Double.parseDouble(store.LNG)) * 1000;

if (storeInfo.length > 300) {
   break;
}
```

圖 3.5 判斷是否發出提醒

3.3.2 Hamilton Path 演算法

西元 1859 年愛爾蘭數學家漢米爾頓(William Hamilton)想解決路徑行走問題，目的是找出一個路徑，能夠把每一個城市中要拜訪的客戶都串起來，而且不能回溯，這類的路徑我們稱它為漢米爾頓路徑(Hamilton path)，它的定義為給一圖 G=(V,E)，通過途中的每一頂點且只通過一次的路徑稱為漢米爾頓路徑[6]。

漢米爾頓路徑在本論文應用上極為重要，程式如附錄 A 所示，因為我們能透過它的特性來產生一條不重覆的路徑，並達到路徑上的最佳情況。舉本論文所使用的例子來說，若我們假設使用者在 A 點，附近半徑 1400 公尺內有 B、C、D、E、F 五間商店，在這五間商家中能買到所有使用者購物清單上欲購買的所有品項，因此就可利用此演算法來實作，來幫使用者規劃一條路徑去買到購物清單上的所有商品。

漢米爾頓路徑這個觀念在本論文中是有些不相同處，因為在真實的地圖上，商店跟商店之間是連通的，我們一定有路可以到達，並非漢米爾頓路徑所提到，必須限制只有某點跟點之前才有路能連通，由於本論文所解的問題為 NP complete 的問題[10]，因此在演算法中我們限制了搜索範圍的輸入大小，所以可在合理的執行時間內發現漢米爾頓路徑，而解出來的解為近似解[11]。

以下為本論文使用之例子：

如圖 3.6 所示，若我們假設使用者現在位於 A，附近有 B(7-11 交大店)、C(好吃家)、D(家樂福)、E(真好味)、F(全家交大店)幾間商店，這五家商家能買到使用者購物清單上的所有物品，使用者將某距離內觸發提醒使用者的參數距離設為 1400 公尺，意思就是說這些店家都在使用者的半徑 1400 公尺範圍內，因此我們就可以套用 Hamilton path 演算法來實作並去建議使用者要怎麼走才會是花費最少 cost 的路徑近似解；例子中各路線上的數字代表所花費的 cost，也就是考慮到實際路徑的紅綠燈秒數總和以及使用者與此間商家的直線距離這些因素所得到的。

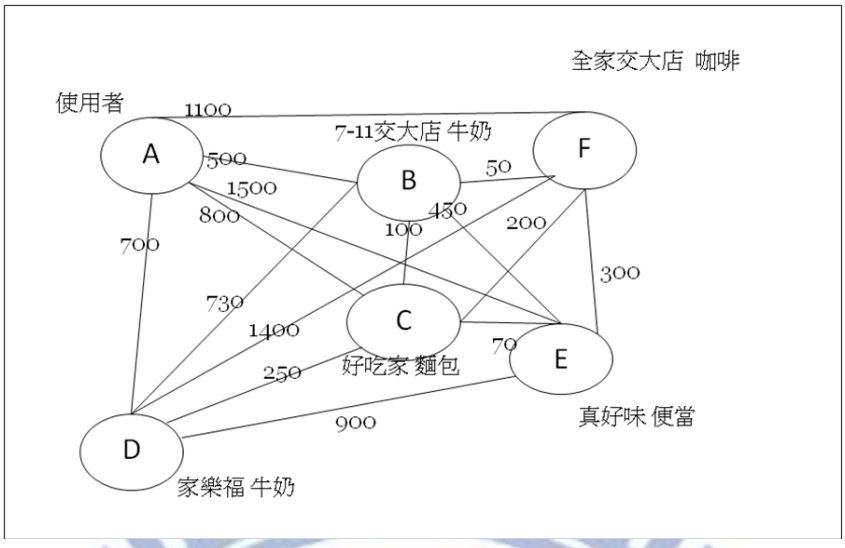


圖 3.6 商家示意圖

設 A 為起始點(現在位置)，然後開始依演算法，如以下步驟：

(1) 如圖 3.7 所示，A 點出發，透過 Greedy[7]的方式，先選花費 cost 最少的，因為各個店家路線都連通，所以選到的是花費 500 cost 到 B 點(7-11 交大店)的路徑，而且也因為沒有重覆走，所以不會回頭。

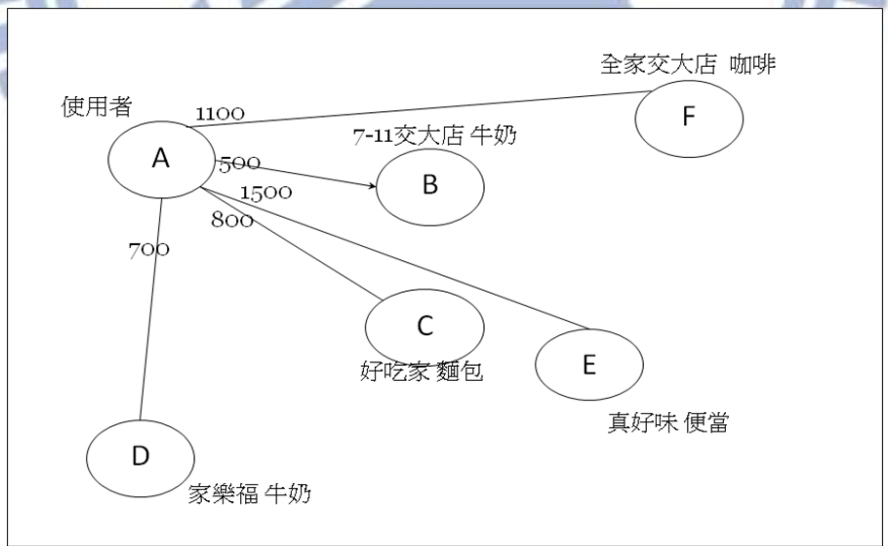


圖 3.7 Step1 示意圖

(2) 如圖 3.8 所示，到達 B 點(7-11 交大店)後，一樣採 Greedy 的方式找花費 cost 最少的，所以會選 cost 花費 50 到 F 點(全家交大店)的路徑走，也因為沒有重覆走，所以不會回頭。

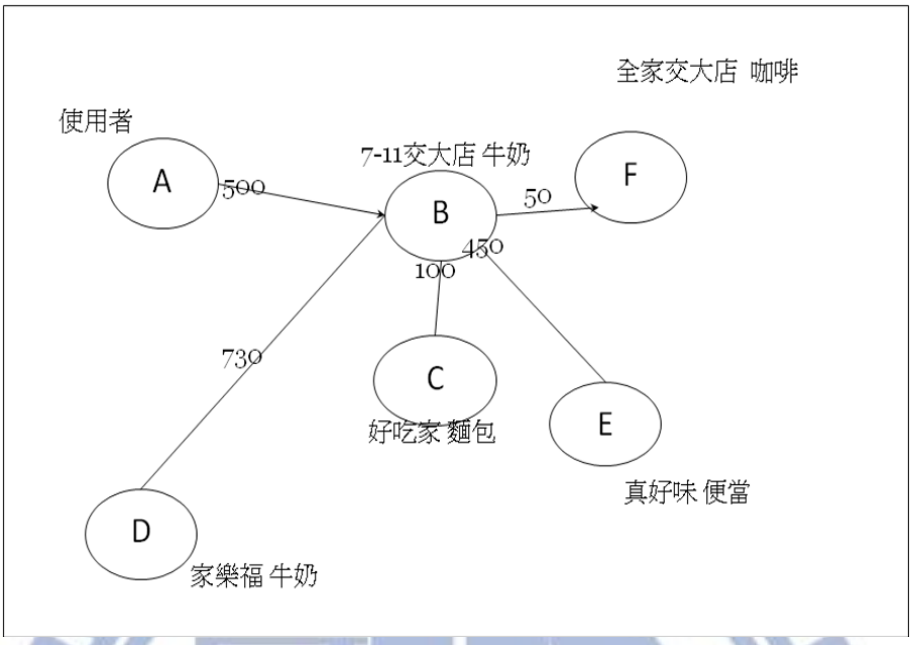


圖 3.8　Step2 示意圖

(3) 如圖 3.9 所示，到達 F 點(全家交大店)後，一樣採 Greedy 的方式找花費 cost 最少的，所以會選花費 200 cost 到 C 點(好吃家)的路徑走，也因為沒有重覆走，所以不會回頭。

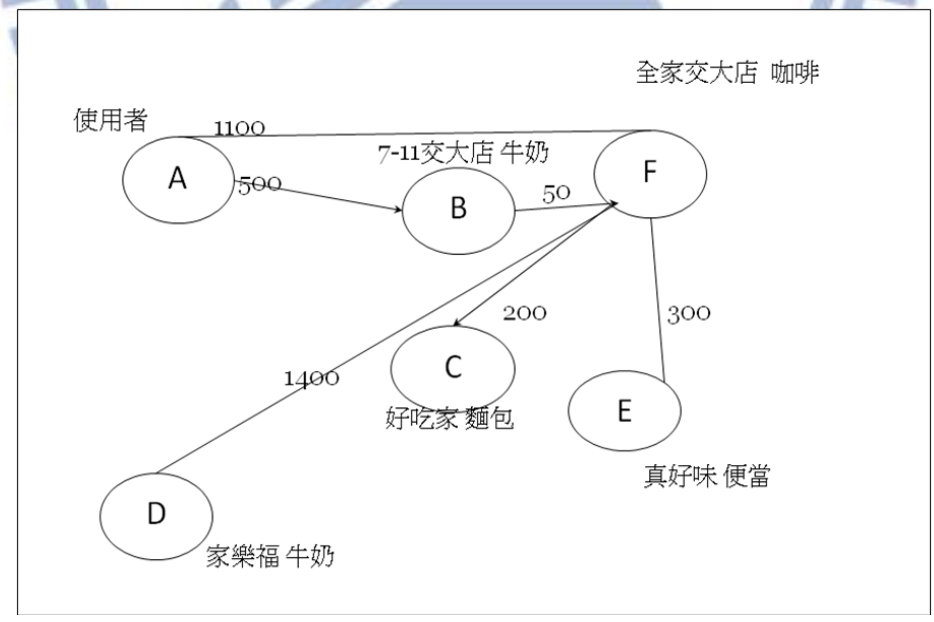


圖 3.9　Step3 示意圖

(4) 如圖 3.10 所示，到達 C 點(好吃家)後，一樣採 Greedy 的方式找花費 cost 最少的，所以會選花費 70 cost 到 E 點(真好味)的路徑，也因為沒有重覆走，所以

不會回頭。

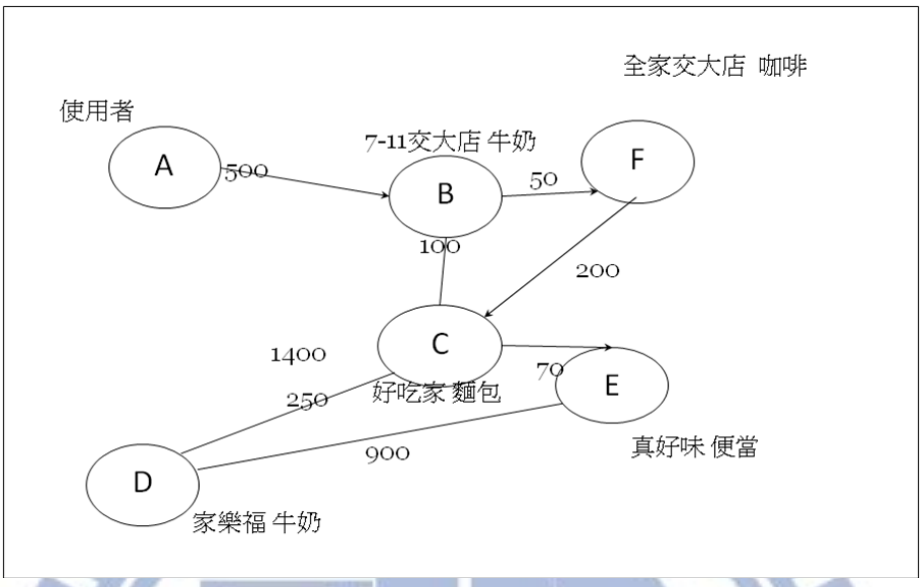


圖 3.10　Step4 示意圖

(5) 到達 E 點(真好味)後，一樣採 Greedy 的方式找花費 cost 最少的，所以會選花費 300 cost 到 F 點(全家交大店)的路徑，但因為發生重覆走的情形，所以退回 E 點繼續找花費 cost 次高 450 的 B(7-11 交大店)路徑，但是也因為發生重覆走的情形，所以我們再度退回到 E 點選擇 cost 花費 900 的路線到 D(家樂福)。

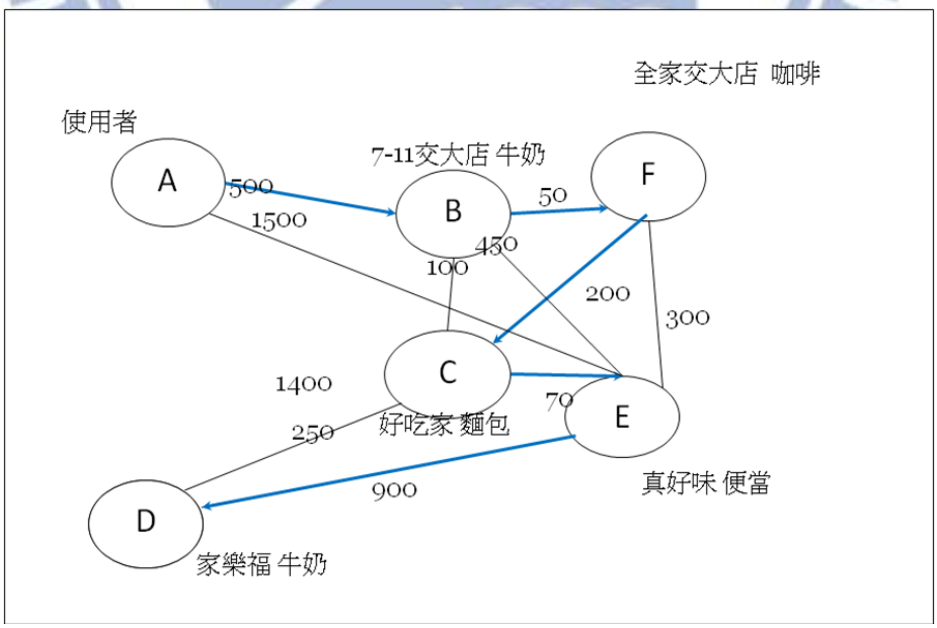


圖 3.11　Step5 示意圖

因此，這條花費最少 cost 路徑的近似解為 500+50+200+70+900 = 1510

(A>B>F>C>E>D)，若走其他條路徑，都會比這條路徑更長，比方說若選擇走以下這條路徑 A>B>D>C>F>E，成本為 500+730+200+300 =1730，都大於我們的近似解。因此我們由這個例子可得知，當使用者透過這個路徑，就能更節省時間，並買到所有想買的東西。

3.4 運作機制流程圖

3.4.1 新增資料

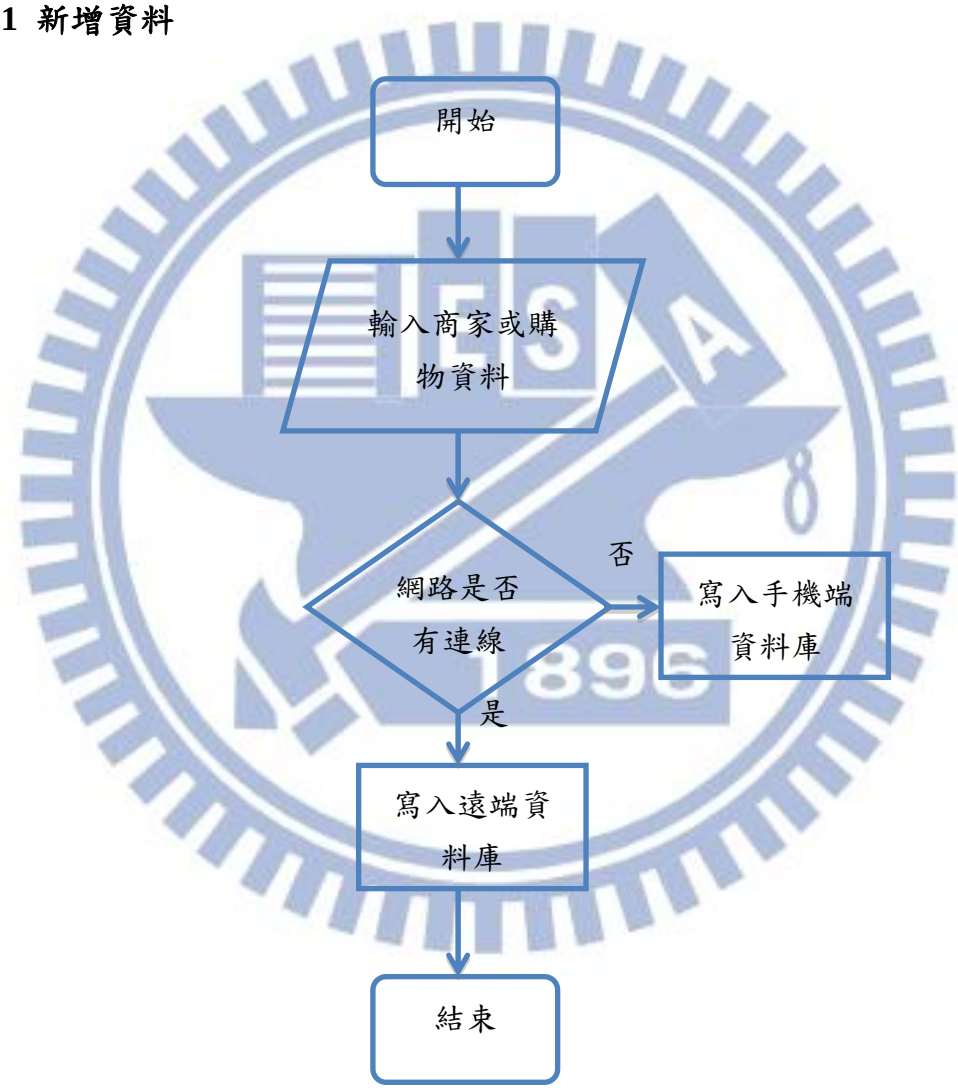


圖 3.12 新增資料流程

如圖 3.12 所示，我們可以由手機輸入要購買的品項清單或做普通記事功能，兩個功能都會牽涉到雲端資料庫的存取，因此我們必須將資料經由 Web Services，透過走 Port 80 的方式，將資料寫入；也能透過 Json 格式的方式來交換資料，進行同步等動作。主要的目的在於當使用者換一支新手機時，我們只要安

裝手機應用程式，就能透過資料取得的方式，同步手機資料庫的資料，而且不怕資料遺失等問題，是手機最常使用的方法。

3.4.2 提醒判斷

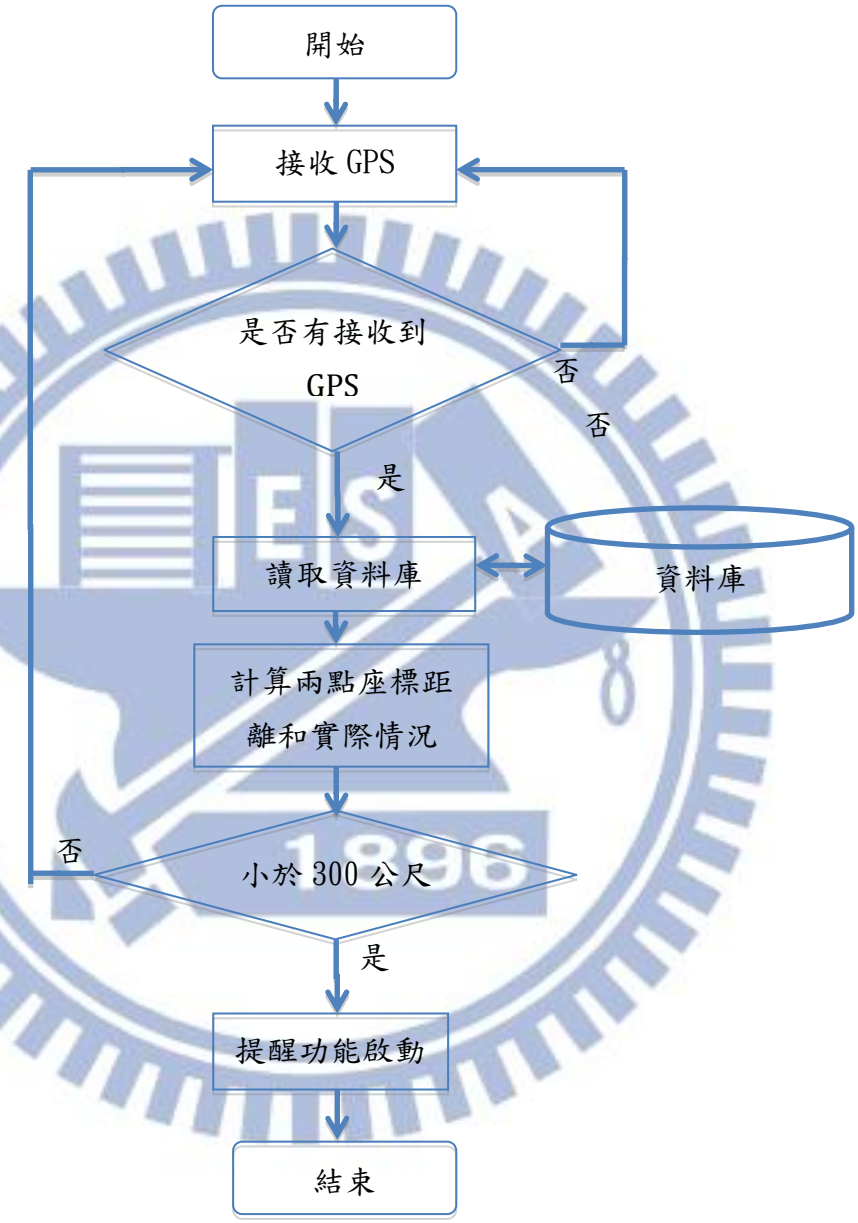


圖 3.13 提醒功能

如圖 3.13 所示，當手機開始接收 GPS 後判斷是否有接收到 GPS 訊號，若有則透過計算兩點直線距離的方式，判斷是否接近資料庫中的商家。要這麼做的原因就是要提醒使用者說，有販賣欲購買物品的商家接近，使用者可以順路去買。

這裡最主要的判斷方法是當使用者與商家距離小於 300 就發出提醒，而由於每個人對距離遠近的定義不同，這裡可以由使用者去決定，要在什麼距離範圍來說，算是順路去進行。

另外提醒功能也有兩者可說，一個是用聲音來提醒，一個是用震動來提醒，這個選項也能同時開啟，都是由使用者決定。

3.4.3 存取資料庫流程

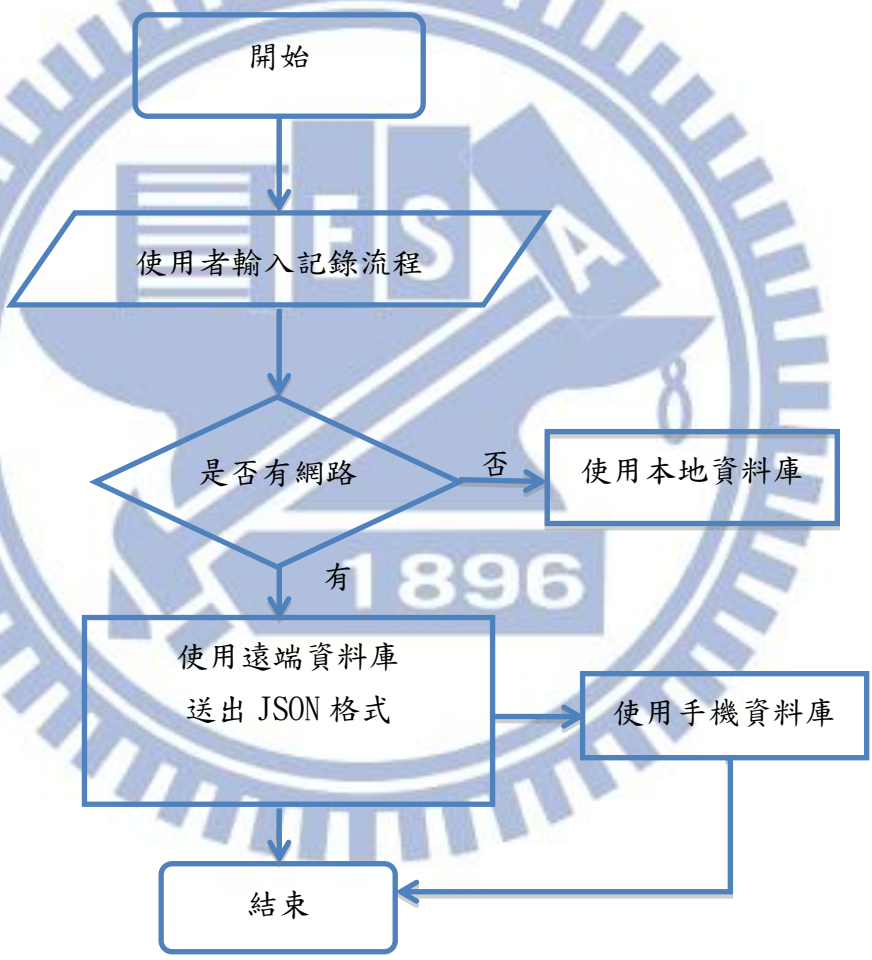


圖 3.14 存取資料庫流程

如圖 3.14 所示，透過存取資料庫流程，來讀寫資料。雖然 3G 網路廣泛的被使用，但訊號強度影響網路品質一直是個問題，因此主要的想法就是希望能讓雲端資料庫和手機本地資料庫有同步的動作。當使用者使用到一半，突然沒有網路訊號時，本系統會自動切換資料庫為本地資料庫，可維持系統的穩定度，避免當

機的情況發生，最後當網路回來的時候，系統會自動同步回去，讓本地和雲端資料庫相同，這樣就能提升系統的可用性與實用性，不像有些手機應用程式都要依賴網路才能使用，造成系統很不實用。

3.4.4 地圖顯示流程

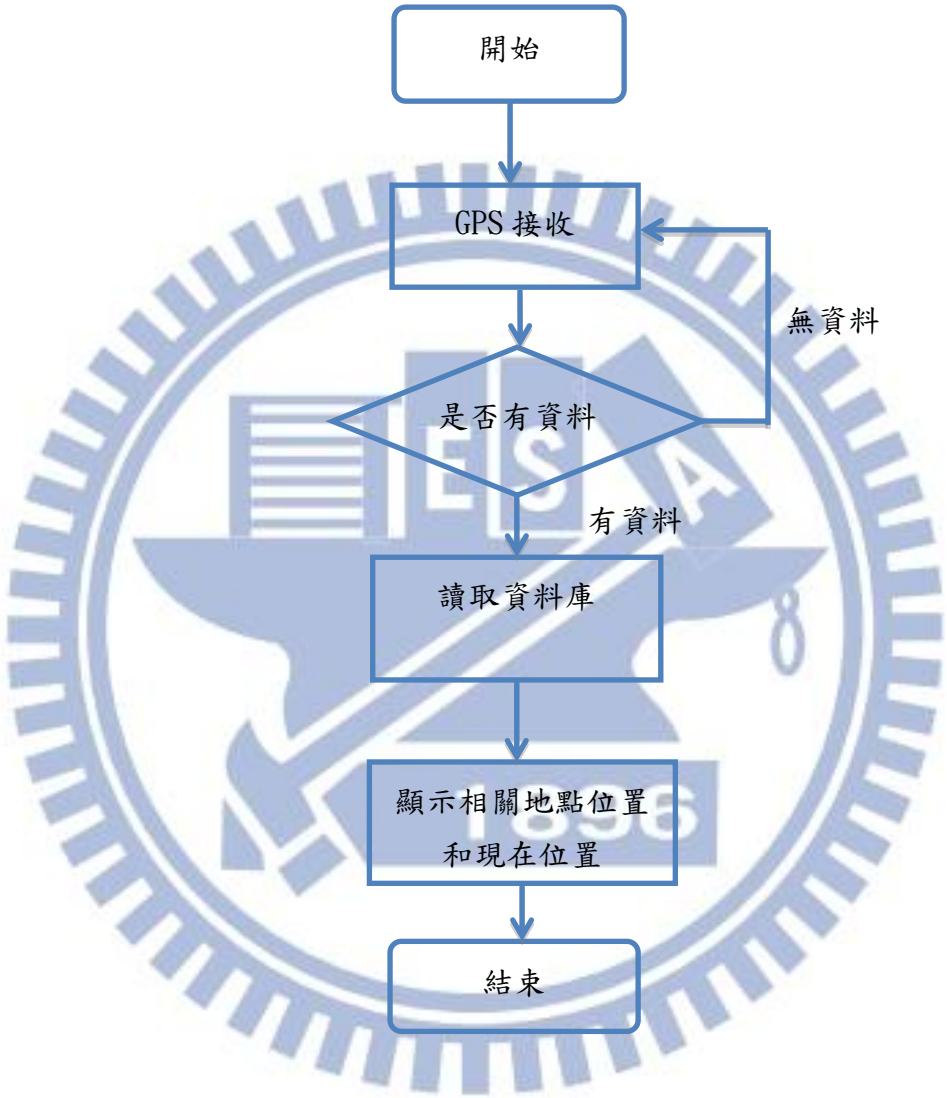


圖 3.15　地圖顯示

如圖 3.15 所示，若使用者要求觀看此點的地圖座標，就會依以下流程來處理。這邊實作的想法是因為手機有 GPS，我們能讀取 GPS 來知道自己在那裡，然後透過資料庫的方式，將所有因為距離演算法而被定義成在使用者附近的商家顯示其地址，且標記在地圖上，並把價格、詳細地址以及所需等待的紅綠燈秒數相關的資訊都表列出來，這樣使用者就可以依不同的價格和喜好點選想要去的店家。

另外這部份的實作比較重要的是，我們能顯示路徑給使用者，透過 API 的方式就能做到，將現在使用者所在的點與商家地址標出路徑，並讓使用者觀看，加強本系統之實用功能，而不是只提供地址。

3.4.5 Hamilton Path 流程

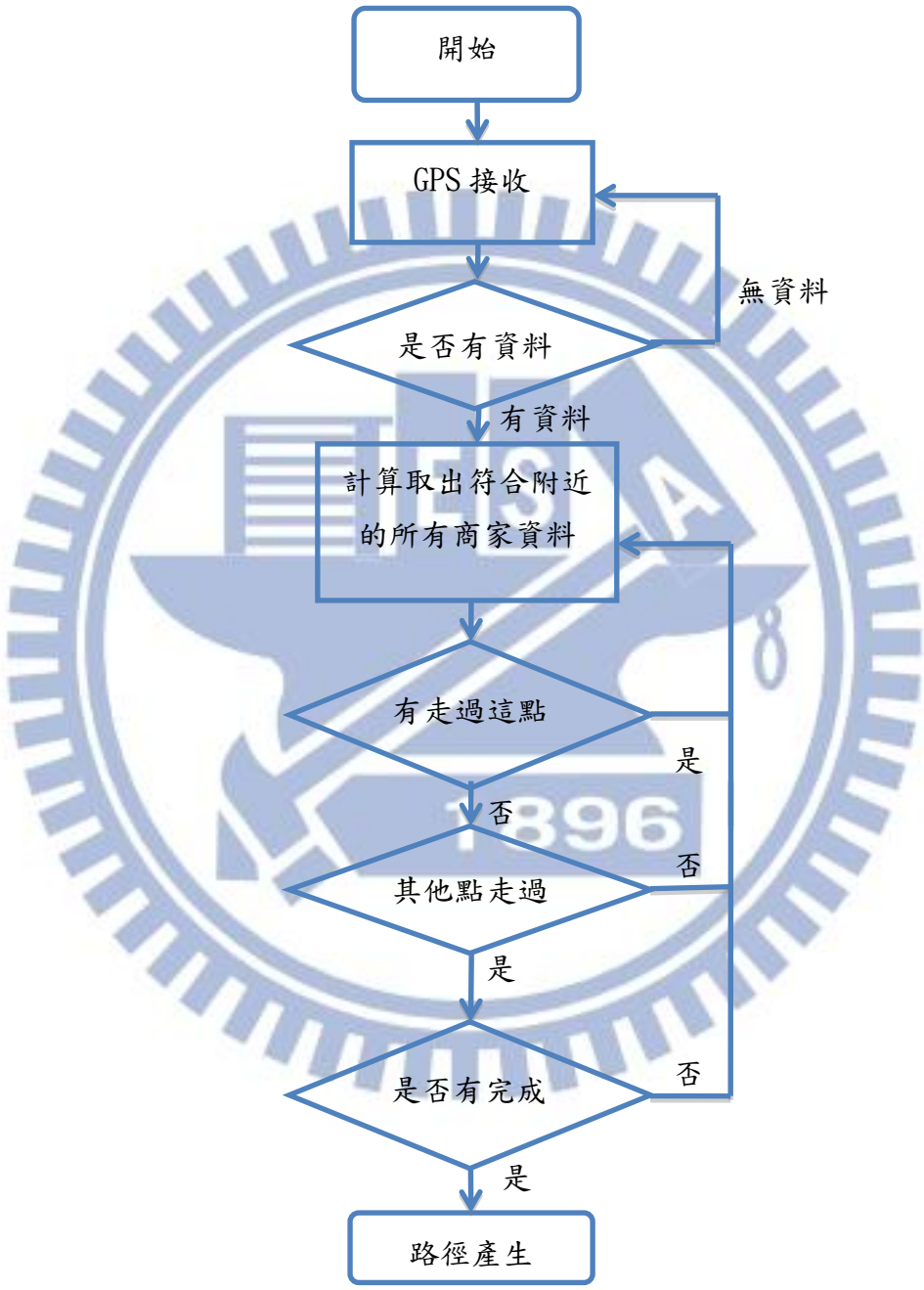


圖 3.16　Hamilton Path

如圖 3.16 所示，由開始點出現至結束點，中間不能有重覆的走法，直到走到目的才完成整個路徑。

第一步就是接收現在的 GPS 訊號，等使用者目前所在的方位建立後，可透過計算直線距離公式知道所有點(商家)跟點(商家)之間的距離。接下來由使用者目前所在的位置開始找路徑，程式會先找花費最少 cost 的那條路徑，並檢查有沒有走過，若沒有就完成現在位置到第一間商家的路徑，然後依續完成每點。過程中只要注意是否有走過的判斷，若有的話就必須退回去，直到所有點走完為止。

根據上述的步驟我們就能為使用者安排一條路徑，並提供給使用者知道要怎麼走才是最節省時間的，使用者就能很順利的將所有要去的商家都去過，並買到所有想買的東西。

第四章 實作成果

4.1 操作流程

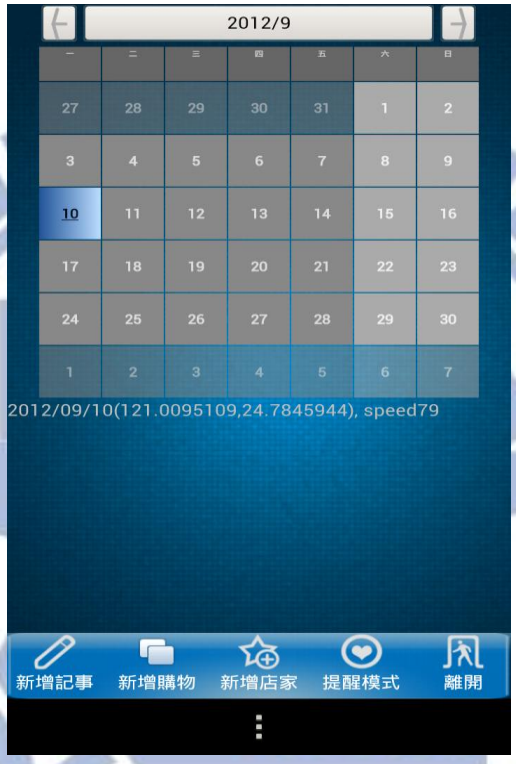


圖 4.1 APP 主畫面

當手機安裝好應用程式的時候，主畫面如圖 4.1 所示，日曆是用來記事或安排購物清單，並能點選日期。選日期的方式也相當容易，使用者可以點選日曆上的日期來安排那天要做的事情，透過很直覺的方式就能操作成功，加強本系統的實用性和可用性。

主畫面下面有幾個功能：新增記事、新增購物、新增店家、提醒模式和離開，這幾個功能會在下面介紹，比較重要的是新增購物、新增店家這兩個功能。本系統利用分類做法，將物品透過分類就能快速取得物品資訊，並將它輸入進資料庫中，而且也能加快選的速度，不會讓使用者沒有耐心，節省時間完成設定等功能。例如我們選日期 9/10 後再點選新增購物，會得到商品的分類。

圖 4.2　商品類別

如圖 4.2 所示，使用者可以依商品分類選取要買的商品，這個功能很快就能知道要買的東西在哪，而分類都可以由合作之商家輸入，價格也能馬上就知道，有了這些分類資訊，我們就能簡單的完成輸入的動作，增加使用者的愛用程度。

圖 4.3　商品內容

如圖 4.3 所示，選取要買的商品，透過分類，很清楚就能完成點選的動作，未來商家可以輸入更多資訊，像是優惠的情況，若使用者在點選此功能就看到那項東西有優惠，就能擴充本系統之優點。

圖 4.4 新增店家

如圖 4.4 所示，我們在主選單按下新增店家的時候，可由使用者輸入商家名稱。GPS 會根據目前使用者的座標位置去顯示目前所在的地址。這項資訊也可以由商家自己建立，達到商家和客戶之間的互動來增加更多的店家資訊。

圖 4.5 日期範圍

如圖 4.5 所示，在新增購物的時候，使用者可自行選擇要提醒的日期範圍 9/10

到 9/12。此功能的想法是因為每個人對買東西的習慣不同，若能設日期範圍，就能完全符合實際上的情況。例如像是 9/10 才發薪水，但現在可能才 9/1，我們提醒時間範圍若設在 9/10 的前幾天在還沒拿到薪水的狀況下就很不符合實際情況也很沒必要性，若有一個範圍，使用者就可以自己調配時間買東西跟預算分配，這是一個很貼心的設計。

圖 4.6 符合當下的購物列表

如圖 4.6 所示，當使用者出門後經過某個地方，剛好提醒的日期範圍符合且有販賣購物清單中物品的商家在範圍內，系統就會顯示出所有符合的列表。使用者可以直接點選商品旁的附近店家，查看店家的詳細資訊。

圖 4.7 商品在商店的資訊，並告知遠近

如圖 4.7 所示，當按下附近店家時，可以看到所販賣物品的價格和目前使用者離這些店家實際的距離是多少，若使用者找到想去的商家或找到那間價格最便宜，可點選上面商家列表，就能進入地圖模式中，如圖 4.8 所示，可以看到店家在地圖上的位置和這間店有提供於購物清單的商品。

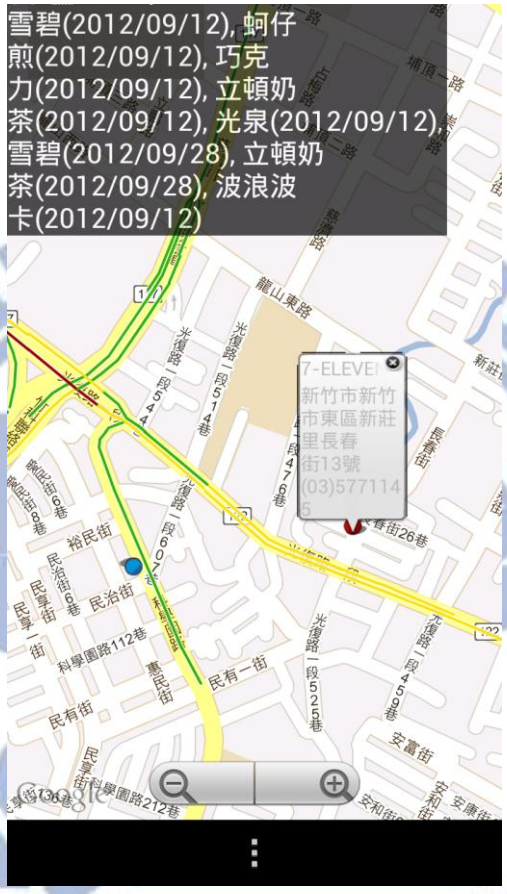


圖 4.8 詳細店家資訊和購買的東西

透過這種方式，就能知道自己的位置和商家的位置，並能開啟自動路徑規劃來知道要怎麼去，也能順便帶一些想買的東西回去，這樣的方式不但便宜又節省時間，而且透過此方式不但對消費者方便，商家也能受益。

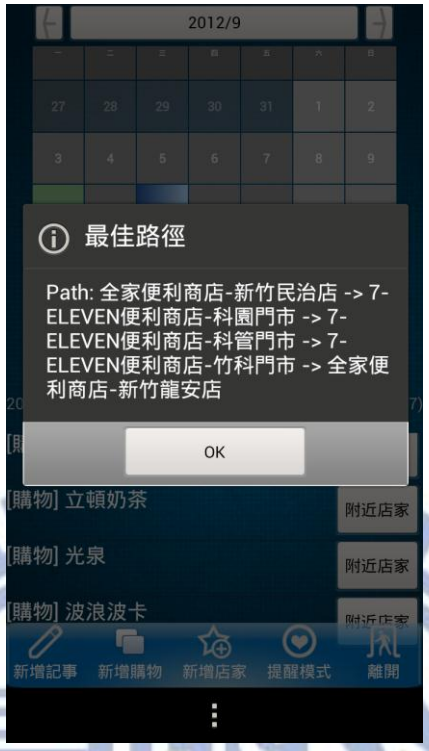


圖 4.9 最佳路徑

如圖 4.9 所示，這項功能對使用者來說非常方便，系統會透過 Hamilton path 演算法幫使用者安排一條路徑，幫使用者節省了很多交通上的時間。

目前因為時間的關係並無將資料用地圖顯示，但在未來只要加入地圖的多點路徑規劃，就能直覺的走去這幾點，更能增加真實性。這個功能創新點在於能夠將路徑串起來提供給使用者一條最佳路徑，當我們在每天行經的路線上經過某個地方的時候，假如手機發出購物提醒，使用者就能很輕易將附近所有符合購物清單上欲購買品項之商家在不折返的情況下都去過，並買到清單上的所有東西，這對使用者來講很方便且能節省很多時間，相較於在目前所有市面上的行事曆提醒系統 APP，他們並無幫使用者做一個路徑規劃的功能。這對手機應用程式來說其應用性是一大突破，不但完成智慧型手機方便的特性，而且對人們的日常生活來說也是一大貢獻。

第五章 效能評估

為了測試本系統 APP 的效能以及實際使用過後的感想，我把 APP 給實驗室的同學以及朋友們做一個使用，並請他們回報使用者的心得評估。以下為使用者的心得評估統整：

1. 當使用者第一次使用此系統時，都會覺得很容易上手，因為介面很友善。另外也因為本系統會連結到雲端上,所以若換另一隻手機也不用擔心同步問題。

2. 可自行決定要提醒的日期範圍區間，可以調配要在什麼時間區間裡提醒使用者需要買東西是很人性化的一個功能；且在每天行經的路線上面附近有販賣購物清單上的品項之店家就發出提醒，並會搭配實際的路況替我們使用者排出購物最佳路徑。相較於目前市面上的行事曆提醒系統 APP 功能上，我們使用者覺得非常的方便好用，而且會有地圖顯示本身位置和店家位置，可說是便利性十足。

3. 相同的商品在不同的店家會將價格排列出來提供使用者參考，這讓現在什麼都在漲的經濟社會來說可以替我們省錢。

4. 當我們使用者在路上自行發現某家小店在資料庫中並無資料，且我們使用者很常來這家小店光顧時，我們可以自行新增店家資訊，非常的方便。

使用者也回報了此 APP 未來能更加精進之處：

1. 物品的分類不足。

2. 無法確切掌握到實際路況。因為現在只用虛擬的紅綠燈秒數以及隨機的時速，並不符合真實的情況。

3. 由於第二點，Hamilton Path 路徑也會有些許的誤差。

4. 路徑規劃部分並無考慮到使用者是在對向或是目前這個車道上。

第六章 結論

此篇論文「Android 平台上之位置感知行事曆提醒系統」透過了一些演算法實作，包含 Hamilton Path 演算法等方法，來實作出 Android 平台上創新的應用程式。與其他市面上行事曆提醒系統的應用程式不同在於，我們系統有雲端資料庫和本地資料庫同步的功能之外，其他的手機應用程式只能在固定的日期時間顯示要買的東西，且無法在實際經過商家的情況下發出提醒，也並無幫使用者安排購物路徑的功能。

本論文最主要的功能是能夠讓使用者自行建立購物清單和要提醒的日期範圍區間，當使用者帶著手機行經在每天所會經過的路線上時，系統會判斷日期範圍吻不吻合和附近是否剛好有販賣購物清單中物品的店家，接下來就會發出提醒給使用者，讓使用者決定要不要去購買商品。

本論文不但有基本的提醒功能，最重要的創新是結合 Hamilton Path 演算法將使用者要前往購買物品的商家串起來，提供給使用者一條最佳的路徑選擇，可以讓使用者非常節省時間的完成購物。

本論文的最大貢獻是我們開發了一個便利性的手機應用程式。由我們實作的效能評估得知，此手機系統應用程式確實提供了一個方便性的購物方法，了解此系統真的能融入到生活中，這種購物方式不但能節省我們寶貴的時間，也可讓使用者覺得更加人性化以及得到非常多便利性的好處。不但如此，若大家都使用此系統，就可讓這個社會更有效率。

第七章 未來展望

由於使用者回報了幾項本應用程式系統可以更加精進之處，因此我們在這邊講述一下我們未來可以讓此系統更趨完美的一個努力方向：

(1) 物品項目能由店家提供，加入推薦的商品。增加與店家合作的機會，店家加入這個 APP 雲端的管理，若有新商品也可登入，造成雙贏的部份，若很多商家加入後，就更可能制衡價格。

(2) 可以結合實際的路況分析，本論文現在做的是兩點之間的直線距離，當遇到高山平地的時候，本系統的距離演算法就有失準確性，所以之後可以跟 google map 或 google API 結合就會有更精確的距離演算，於是就可以在路徑安排上面提高更多的準確性。

(3) 我們所使用的 Hamilton Path 演算法為暴力求取法(Greedy)，由於 Hamilton Path 為 NP Complete 的問題，目前仍然持續在研究當中，所以並無找到最佳解的方法，因此目前能解出來的都是近似解，且若提醒範圍設定的很大，可能需要耗費很可觀的時間，未來可以致力於研究更有效率的演算法去解 Hamilton Path 的問題[9]，例如巨集啟發式方法。

(4) 由於在地圖上我們只判別商家的位置，並不會由使用者目前的位置考慮到在前往商家的路徑上時是在目前這條車道上，還是對向車道上，若未來能結合這部份，勢必能更加強系統的實用性。

(5) 由於找尋安排路徑的演算法是目前很熱門的研究，所以往後可積極鑽研商家之間的路徑規劃方法，真實將地圖的點串起來，可避免遇到死路問題。

參考文獻

[1] Android wiki, http://zh.wikipedia.org/zh-hant/Android.

[2] Web Service 的應用與省思, http://www.dsc.com.tw/newspaper/46/46-1.htm.

[3] JSON wiki, http://zh.wikipedia.org/zh-hant/JSON.

[4] WebService 詳解, http://blog.csdn.net/lyq8479/article/details/6428288.

[5]安裝 Android 環境, http://www.slideshare.net/superlevin/windowsandroid.

[6] Hamilton Path wiki, http://en.wikipedia.org/wiki/Hamiltonian_path.

[7] Y. Li, Y. Yang, X. Lu, "Rules of Designing Routing Metrics for Greedy Face and Combined Greedy-face Routing". IEEE Transactions on Mobile Computing, 9(4):582--595, April 2010.

[8] E. Chin, A. P. Felt, K. Greenwood, D. Wagner, "Analyzing Inter-application Communication in Android". In Proc. 9th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), pages 239--252, June 2011.

[9] Cristina Bazgan, Miklos Santha, Zsolt Tuza. "On the Approximation of Finding A(nother) Hamiltonian Cycle in Cubic Hamiltonian Graphs". Journal of Algorithms, 31(1):249--268, April 1999.

[10] Xiao, J.T., Wang, J. "A Type of Variation of Hamilton Path Problem with Applications". In 9th International Conference for Young Computer Scientists, 2008, pages 88--93, November 2008.

[11] Hongtao Zhao, Qingde Kang. "Large Sets of Hamilton Cycle and Path Decompositions of Complete Bipartite Graphs". Graphs and Combinatorics, October 2011.

[12] Priyanka Shah, Ruta Gadgil, Neha Tamhankar. "Location Based Reminder Using GPS For Mobile (Android)". Journal of Science and Technology, 2(4): 377--380, May 2012.

[13] Xianhua Shu, Zhenjun Du, Rong Chen. "Research on Mobile Location Service Design Based on Android". In 5th Int. Conf. On Wireless Communication Networking and Mobile Computing, pages 1--4, September 2009.

附錄 A

以下為 Hamilton Path 演算法的程式碼，透過程式，我們能取得點和點之間近似解，並將結果由陣列存起來。

```
new Thread(new Runnable() {
@Override
public void run() {

//現在位置 lastGpsLocation
while (lastGpsLocation != null) {
        String txt = "";
        //GPS
        Double lat = lastGpsLocation.getLatitude();
        Double lng = lastGpsLocation.getLongitude();

        String today = (new
StringBuilder().append(calToday.get(Calendar.YEAR)).append("/")

    .append(format(calToday.get(Calendar.MONTH) +
1)).append("/").append(format(calToday.get(Calendar.DAY_OF_MONTH)))).toString
();
        //取得今日符合的清單(裡面有商品和店家資訊)
        List<NoteInfo> data = db.getMoreThenNote(today);
        ArrayList<String> butItem = new ArrayList<String>();

        //find path
        ArrayList<StoreInfo> storenode = new ArrayList<StoreInfo>();
        HashMap<String, String> hm = new HashMap<String, String>();
        Boolean hasToday = false;

         //開始找路徑
```

```
            for (NoteInfo noteInfo : data) {
                        if (noteInfo.Date.equals(today)) {
                           hasToday = true;
                        }
                        String itemName = db.getItemName(noteInfo.Data);
                        //如果是在清單裡才做下面
                        if (noteInfo.Type.equals("buyList")) {
                           List<StoreInfo> storeData =
db.getStoreNoteByNoteID(noteInfo.ID);

                     Collections.sort(storeData, new Comparator<StoreInfo>() {
                                                      @Override
                           public int compare(StoreInfo lhs, StoreInfo rhs) {
                           return lhs.length.compareTo(rhs.length);
                     }
                     });

                       //get min distance
                       StoreInfo si = storeData.get(0);
                     storenode.add(si);
                  }
               }

               //find path
               double minD = 9999;
               StoreInfo minnode = null;

               while (true)
               {
                     //現在這點，找下一點最小的部份
                     if (pathnode.size() == storenode.size()) break;
                     //若是第一點到第二點
                     if (pathnode.size() == 0) {
                           //now -> the shortest distance store
```

```
                    //找所有點跟這現在這點的距離最小的
                    for (int k=0; k<storenode.size(); k++) {
                        double dis = GpsFuns.GetDistance(lat, lng,
Double.parseDouble(storenode.get(k).LAT),
Double.parseDouble(storenode.get(k).LNG));
                        if (dis < minD)
                        {
                            minD = dis;
                            minnode = storenode.get(k);
                        }
                    }
                        //找到後就加入，pathnode 為記在 path 的陣列
                        pathnode.add(minnode);
                }
                else
                {
                    //
                    int index = pathnode.size() - 1;
                    StoreInfo nowNode = pathnode.get(index);
                    StoreInfo nextNode = getMinDjNode(nowNode, storenode,
pathnode);

                    //找到後就加入，pathnode 為記在 path 的陣列
                    pathnode.add(nextNode);
                }
            }

    }
}).start();
```