國立台北大學資訊工程學系專題報告

LoRa Logistics Service

-A LoRa-Based Autonomous Vehicle with Logistics System for IOT

專題組員:張紘綸、耿楷寯、黃俊瑋、胡育旋

專題編號：PRJ-NTPUCSIE-105-003
執行期間：105 年 9 月 至 106 年 5 月

一、摘要

隨著物聯網的興起，許多無線傳輸的協議受到許多研究人員以及大眾的關注，以往日常生活中常見的藍芽、WiFi，3G 和 4G…… 等，皆是物聯網的無線傳輸技術。在近年來圍繞在物聯網中新興無線傳輸技術之一的「LoRa」也是當今新興的熱門技術與科技，得以讓開發人員實現遠距離、低耗電率的通信系統，利用這些特點，實現更多嵌入式裝置上的應用。

在現今社會中，對運送者來講，運送貨物須耗費大量人力控管貨物運送流程，造成許多管理上的不便，而對於欲寄送包裹的使用者，其寄送前所需進行的申請動作繁瑣，親跑相關單位也會造成使用者不便，因此，實現一個自動化代送系統，透過使用者以手機 App 對系統下訂單，系統便能指派附近的自走車至寄送者來幫忙代送，不但可以減少運送上的人力資源，也可節省使用者寄送包裹流程上所耗費的時間與精力。

本研究將使用 LoRa 無線傳輸技術，利用其低功耗以及長距離的特性，透過設立 LoRa 閘道(Gateway)的方式隨時接收資料的訊息，利用自走車做為運送者，使用資料訊號對車體進行控制，實現一個自走車物流系統，使用者透過簡易的使用 App 的方式填入資料，並經由 Server 進行彙整資料，在指定的時間指示車子開始送件，Server 也可以用推播方式告知 App 寄件者或收件者車子到達之通知，以及訂單之狀態，自走車上會透過路徑演算法進行分析最佳路徑，使自走車順利到達寄件與收件地點，讓使用者得以順利的取件/收件，並且要求使用者輸入後端產生的取件密碼，確保車中的郵件或包裹安全性，在未來得以便利我們的生活，並實現「智慧城市」的最初理念。

二、簡介

（一） 研製背景

近年來，IoT 一詞不斷地被應用，生活中的任何東西，都能夠與 IoT 結合，物聯網一詞最早是由 Kevin Ashton 於 1999 年提出，他用這個概念來描述說，當物品連接上網路後，對我們的生活帶來什麼樣的改變。然而 IBM 在 2014 年指出，因為物聯網目前成本過高、過於複雜，過去感測網路技術時常遇到功耗高、距離短而無法達到需求的問題，因此 IBM 提出藉由 LoRa (Long-range)無線網路技術解決。而現今智慧城市與智慧物流有著密不可分的關係，而智慧物流因無人運送機的大幅成長而興起，各廠商主打利用無人方式

運送貨物， Amazon 在 2013 年底揭露該站的 Prime Air 無人機遞送服務，瑞士郵局也選用了 Starship 無人機器車來運送包裹，凸顯智慧運送車對智慧物流的重要性。

（二） 目標

本專題計畫設計了一套系統，結合 LoRa 無線傳輸、自走車、手機及伺服器程式，達到智慧城市的目的，能夠讓使用者在手機上登記寄件資料，自走車根據資料自動抵達位置取貨、運送、交貨，使用者僅需寄放貨物、使用手機即可輕鬆監控貨物資訊、等待收貨，不必耗費大量人力運送、管理。

（三） 預估成果

1. 自走車可藉 LoRa 經由閘道與伺服器做溝通，並取得寄件資料。
2. 自走車經由路線演算法，能在指定的時間到信件出發地收取信件，並送到收件地。
3. 可以根據寄件資料，在自走車空閒的時候能使用手機 App 登記新的寄件資料，並指定信件出發地與收件地。
4. 收件者有登入手機 App 的話，將會收到取件的通知，提醒使用者取件的時間與地點。
5. 使用此系統可以節省送貨員跟包裹寄送者的時間，以提高工作的效率。

三、專題進行方式

（一） 實作平台與技術

1. 開發工具

本專題使用之開發工具如下圖：

Figure 1 開發工具

自走車以及 LoRa Gateway 使用樹莓派進行開發並且搭載 LoRa 模組在 Linux 環境下執行程式，Server 的部分使用 Microsoft Azure 雲端計算平台進行 Server 的開發並以裝載虛擬機的方式使用 Ubuntu 系統架設 MySQL 資料庫透過 C/C++程式撰寫 Server，使用 Microsoft Azure 雲端計算平台最主要的原因在於透過遠端伺服器的架設針對系統穩定度能有大量的提升，不同於使用學校所架設的 Server，而且在實驗的過程當中對於實驗的進行上也較為方便，我們可以任意的透過 SSH 的方式開啟 Server 並且執行，而在使用者端也能透過實體 IP 方式進行 Server 的連線並且進行溝通。最後，手機利用 Android Studio 進行設計，也因此使用的語言為 Java 語言。

2. LoRa 介紹

本次專題使用的無線傳輸技術為 LoRa，LoRa 是在 2014 年被 IBM 所提出的一個新興的物聯網傳輸方式，使用的技術為線性調頻擴頻調製技術，因此整體性能上擁有類似於 FSK(頻移鍵控)調製相同的低功耗以及傳輸距離長的特性，LoRa 是以 UDP 的方式進行傳輸，同時也是免許可的開放頻段。長距離的可

涵蓋範圍可從一公里到二十公里遠，也因此台北市只需要七至十台 LoRa 基地台進行佈點，便可以使訊號接受涵蓋整個城市範圍；而靠著 IOT 裝置的內建電池，即可維繫長達十年以上的使用時間，安裝部署的所需成本也較低。

（二） 系統分析與設計摘要

1. 系統架構

本專題之系統架構圖如下圖所示:

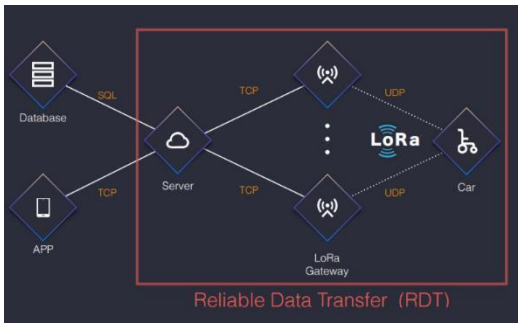

Figure 2 系統架構圖

依照系統架構圖所示，自走車會跟系統中的LoRa Gateway 透過 sx1272 LoRa 晶片進行網路溝通，而 LoRa Gateway 則會透過網路跟系統中的伺服器進行溝通，利用 LoRa 大範圍涵蓋特性，將 LoRa Gateway 佈設在所需涵蓋的環境,則自走車能夠在涵蓋範圍內收到需要的訊息，以及更新自走車自身狀態至遠端伺服器，使得伺服器因此得以監控遠端自走車使用情況與事件，讓使用者可以透過手機 App 向系統進行文件自動運送的應用，以上述的環境架設的基礎下，自走車需要一套貨物運送流程，來確保運送貨物的正確性，因此自走車會利用 state 來定義自身所處的狀態，在 LoRa 傳輸方面，LoRa 是以無線網路 UDP 的方式傳輸，這代表著在傳遞的途中資料可能會遺失，因此本專題建立了 Reliable data transfer 3.0(RDT 3.0)機制來確保資料正確，此外，APP、Server、車子之間，彼此以事件的方式進行 packet 溝通，因此也另外定義了資料溝通表格，幫助程式之間的事件溝通，底下將一一呈現 Server、車子、LoRa Gateway 和 App 之介紹，並且說明 state 機制、RDT 實作和 LoRa 封包格式之實作機制方法。

(A) Server

本專題的伺服器使用 Microsoft Azure 的雲端計算平台，使用的作業系統為 Ubuntu，以提供系統一個穩定的後端環境；在伺服器上使用 MySQL 資料庫，用來儲存系統中重要的資訊，資料庫中建立了四組表格，分別是 user、location、cars、transport：

- user：手機 App 使用者的資訊
- location：校園內各大樓的位置資訊
- cars：自走車的資訊
- transport：寄件資料的

並在伺服器上用 C/C++編寫一支伺服器程式，此程式會使用 C++ connecter 來管理資料庫，系統中的手機與車子可藉由伺服器程式來取得並使用資料庫中的系統資訊，而伺服器也可以對到了寄件時間或是收件時間的使用者發出通知，請他們要去寄件或是收件，也可以在寄件時間與收件時間前通知車子，該前往的寄件地與收件地，以確保系統的即時性。

(B) 車子

Figure 3 車子架構圖

本專題使用 Raspberry pi 來完成 LoRa 導航器實作，透過本專題所使用的 LoRa 協定基礎的 sx1272 晶片，接入於樹莓派，透過 SPI 接口來使用 LoRa 晶片進行控制，以達到網路傳輸部分之需求，而對車體的信號控制方面，透過 Raspberry pi 輸出的信號來控車子的行徑，並且在路徑規劃當中，導航系統利用 GPS 晶片，利用 GPS 做起點與終點的定位，來達成路徑規劃以及車子行走時的依據，並且使用 A*演算法來算出最短路徑，且較為快速，A*演算法之公式如下，並且這個公式遵循下表格 1 特性:

$$f(n) = g(n) + h(n)$$

Equation 1 A*演算法公式

1	如果 g(n)為 0，即只計算任意頂點 n 到目標的評估函式 h(n)，而不計算起點到頂點 n 的距離，則演算法轉化為 BFS(Breadth First Search),此時使用的是貪心策略，速度最快，但可能得不出最優解
2	如果 h(n)不為 0，則一定可以求出最優解，而且 h(n)越小，需要計算的節點越多，演算法效率越低，常見的評估函式有——歐幾里得距離、曼哈頓距離、切比雪夫距離
3	如果 h(n)為 0，即只需求出起點到任意頂點 *n* 的最短路徑 g(n)，而不計算任何評估函式 h(n)則轉化為單源最短路徑問題，即 Dijkstra 演算法，此時需要計算最多的定點

Table 1 A*演算法特性

以 LoRa、GPS、Raspberry Pi 三模組完成之 LoRa 導航器，可以獨立運作，透過 Server 給予導航器的訂單，規劃出相應路徑，而導航器再將路徑資訊換算成車體控制訊號，藉此依照導航之路徑去做移動，完成自動送貨的初步呈現。

(C) LoRa Gateway

本專題 LoRa Gateway 在架構中是一個「中繼點」的角色。透過 Raspberry Pi 並搭載 LoRa 模組進行 Gateway 的開發，LoRa Gateway 使用有線/無線網路進行與 Server 進行連線，作為車子和 Server 之間的溝通橋樑。主要的 Gateway 設計的架構是能夠在 Server 傳遞訊息時能利用 Socket 的方式得到 Server 的資料，而當車子如果需要傳遞訊息，則也能透過 LoRa 傳輸方式傳遞給 LoRa Gateway 並將資料透過有線/無線網路的方式傳遞給 Server 進行資料的更新。

在專題的初期為了確認整個環校校園皆能收到 LoRa 的訊號，也因此在實驗上我們進行 LoRa Gateway 的環校實驗，確保校園的每個角落能夠接收到 LoRa，並且透過 LoRa 進行資料的傳送，以利在專題後期的發展。接著為了完成 LoRa Gateway 和車子間資料傳送

的機制，在專題上學期的實驗當中，我們透過 LoRa Gateway 與車子建立出一個簡易的傳輸機制，並且實踐 State 機制，將車子和 LoRa Gateway 間的溝通建立了基本的架構，以利下學期在專題與 Server 進行合併時能夠更迅速的與車子進行整合，實踐整體的傳輸機制。

(D) App

本專題的 App 分為兩大部分：介面與背景服務。在介面的部分，我們希望增進使用者體驗，讓使用者可以很直覺的使用本 App，因此簡化各功能的使用流程，並用明顯的顏色以及一目了然的畫面讓使用者一眼即可得到自己想得知的訊息。

而在背景服務的部分，包含了兩個獨立的 Service：ConnectService 與 BackgroundReceiveService，各自擁有自己與 Server 的 Socket 連線進行訊息傳輸。

- ConnectService：當使用者與 App 互動、介面需要 Server 資料時，此服務會接收介面要求，透過連線將要求傳送給 Server 並分析 Server 的回應訊息，分析完後將資料傳回給介面，供使用者使用。
- BackgroundReceiveService：使用者登入或註冊後，會自動開啟此服務，等待 Server 傳來的寄件、收件通知，收到通知後開啟 Notification 提醒使用者。

2. 實現機制

(A) state 機制

為了管理包裹的運輸狀態，必須設計出一套 state 機制，運送某筆訂單的過程中，對應事件發生時，車子必須通知 Server，Server 進而改變該筆訂單的 state 值，如下圖所示：

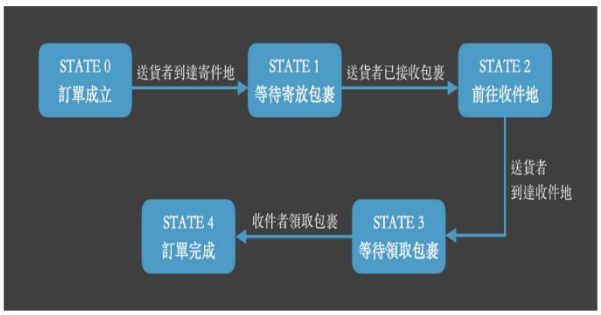

Figure 4 state 機制

(B) RDT 機制

就如系統架構圖所表示，LoRa Gateway 與車子是使用 LoRa 進行溝通，但是 LoRa 無線網路是以 UDP 的方式傳輸，這代表著在傳遞的途中資料可能會遺失，為了確保資料傳輸上的正確性，必須建立有效的傳輸協定，因此本專題在 Server 與車子之間實現了 Reliable data transfer 3.0(RDT 3.0)機制，利用有限狀態機(Finite state Machine)實現機制，其中包含 ACK、sequence Number、Timer 的實作，如下圖:

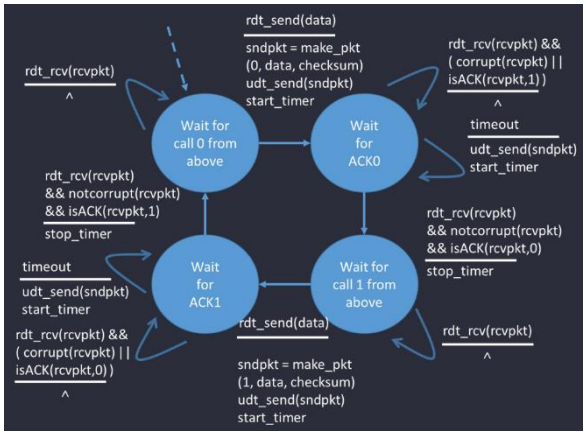

Figure 5 RDT 3.0 機制

因此，以系統架構圖呈現 RDT 效果如先前架構圖，利用 RDT 確保資料傳輸的正確性，可以有效解決 LoRa 傳輸 UDP 掉封包的不足之處。

(C) LoRa 封包格式

因 LoRa 封包的傳輸並不像 socket 的傳輸方式，每一個手機所傳輸的資料都會對應到一個作業系統給予的 socket number，而不同的車子經由 LoRa 所傳送的資料都會被同一個接口所收到，為了要有方法可以讓 Server 分辨不同的個別是來自哪一臺車，以及 Server 傳送的訊息是哪一臺車該接受的，因此制定了 LoRa 封包的傳輸格式：

Header(4)	Data

Table 2 LoRa 封包傳輸格式

Header 中會標示接收者為車子或是 Server，封包要處理的有車子 ID、Packet number，以及標出封包為 ACK 或是系統的任務。

（三） 主要困難與解決之道

1. 如何利用 GPS 達到路線規劃？路線規劃最佳化？

自走車需要根據由伺服器向閘道發出的運送要求(transport request)，並由閘道通知自走車後，自走車透過閘道傳來的座標位置來計算所需要行進的路徑。為了使車體可以根據實際的道路去做移動，因此會需要在各個重要路口以及轉角設置中繼點。

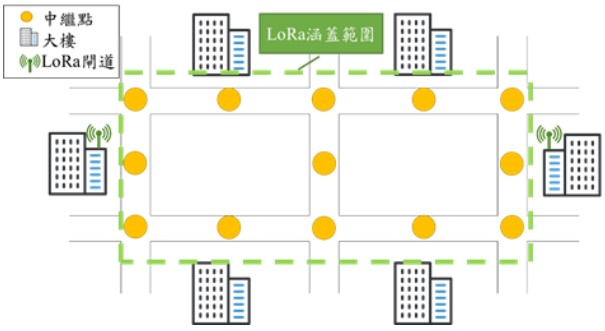

Figure 6 路線規劃中繼點

根據所擺設的中繼點（node）的地理位置，建立出一個座標空間，自走車會根據自身 GPS 晶片去偵測實際所在的地理座標，建立自身位置的資訊，根據此環境模型的中繼點與自身座標點所形成的圖，依據 A*最短路徑演算法，去實際算出最佳路徑解，紀錄全程所需經過的中繼點形成紀錄點鏈結(node link)，自走車便會根據紀錄點鏈結來確保當經過一個紀錄點時可以得知下一個要前往的紀錄點，最終到達目的地。

2. LoRa Gateway 如何佈點能夠達到最小的失誤率？

為了確保自走車在實驗環境下都可以將資料透過閘道輸出去，必須考慮到 LoRa 本身的訊號範圍以及周遭障礙物干擾，若是考慮到最少的障礙物干擾，則將閘道佈建在高樓上得以讓 LoRa 發揮距離長的特性，使 LoRa 底下區域皆能覆蓋，而 LoRa 在傳輸時會遇到大樓本身的阻擋問題如下圖所示：

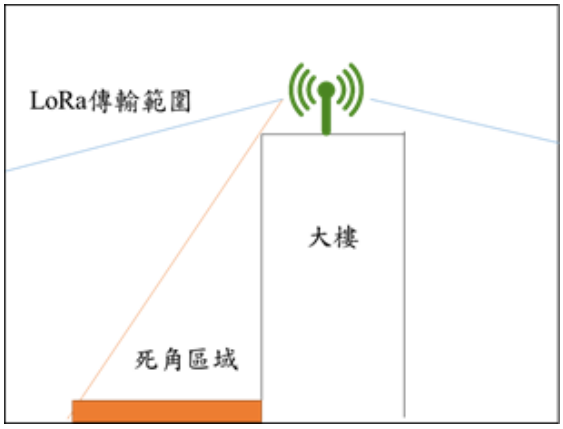

Figure 7 LoRa 傳輸阻擋問題

若是閘道擺在大樓頂的一側，相對的另一側將會被大樓本身阻擋，則在佈點方面必須考慮到上述的限制，因此要涵蓋特定範圍內的區域，需要多佈設幾個 LoRa 閘道來確保彼此間都能涵蓋到自己所不能含蓋到的死角，藉此確保自

走車能在實驗環境下透過 LoRa 完成傳輸，可以參考下圖是我們的解決方法。

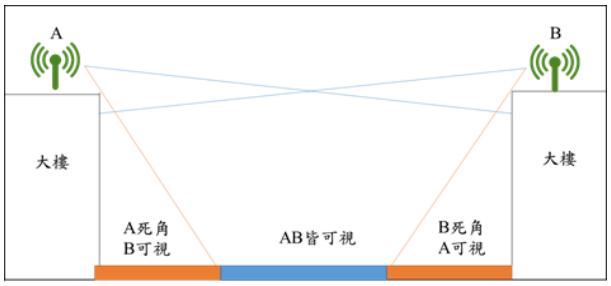

Figure 8 解決辦法

3. LoRa Gateway 問題

(A) 一開始在執行程式時，我們是個別透過兩個 Thread 方式建立 LoRa 接收/傳送的行動，但由於 LoRa 模組在實測中無法利用 Multi Thread 進行同時接收/傳送的行為，也因此後來曾經將 LoRa Gateway 變成兩個 Thread 互相輪流執行，但實際上在進行 LoRa 接收/傳遞資料的過程中呈現的結果卻不如預期能夠有效的平衡接收/傳送資料，而是大部分處於 LoRa 接收或是傳送某一個狀態，後來也曾改用 MMutex 的方式進行程式的撰寫，但結果仍然不如預期。最後，考慮到 Gateway 在 LoRa 的使用上主要在聆聽的部分，也因此程式的設計上設置一個開關，而這個公用變數能夠辨別 LoRa 模組該傳送或接收，而一旦 TCPSocket 進行資料的接收，我們也將更改開關的狀態，使下一次 LoRa 的 Thread 在執行時能夠進行傳送的行動，完成傳送後，我們也會改回開關的狀態，並且使 LoRa Gateway 進行執行聆聽的行為。

(B) LoRa 在接收資料時由於是透過 Timer 計時，若沒有收到便會跳出，並繼續執行程式內容，也因此往往必須設定一段等待 LoRa 接收資料的時間，若某一台單純的進行接收等待資料不會有問題產生，但倘若現在需要兩台同時接收，並且資料上可能會進行互相傳送，此時便會發生等待時間相同的情形，也因此在等待時間的設計上我們必須將 LoRa 的設定調成不同頻率，透過不同頻率的設定可以避免 LoRa Gateway 或者車子同時執行非剩餘的程式（理論上若要接收到訊息，程式必須有一方一定處於等待狀態）造成無法接收到訊息的結果，也因此我們必須在一開始進行等待時間的設定時將 LoRa Gateway 與車子的時間設定不同的等待時間便可避免此一問題。

(C) 與 Server 進行 TCP Socket 連線時，在連線過程中會發生非預期性的斷線，會造成程式運作進行失敗，因此在本專題中 LoRa Gateway 建立了重新連線的機制，主要的運作為當 LoRa Gateway 在一定時間內（約 1 分鐘）沒有收到來自 Server 的資訊，此時 LoRa Gateway 便會進行斷線，然後進行重新連線。透過重連機制的運作，能夠順利解決並避免未來系統運作 LoRa Gateway 與 Server 會造成斷線的問題。

4. 如何讓 BackgroundReceiveService 能夠在 app 結束執行時繼續取得 CPU、等待接收 Server 傳來的通知？

偵測使用者開機(ACTION_BOOT_COMPLETED)及螢幕解鎖(ACTION_USER_PRESENT)的事件，若事件發生，先申請電源鎖，避免系統休眠，再開啟連線接收 Server 訊息，接收並處理完訊息則關閉電源鎖。

四、主要成果與評估

（一） 個別研究成果

1. Server

伺服器能夠接受手機及車子的連

線，手機跟車子可以藉由伺服器取得並使用資料庫中的資料，而伺服器也可以在手機登記寄件的時間主動傳寄件資訊給車子，指使車子前往處理寄件事件，而在車子到達指定的地點後，伺服器也能主動通知使用者手機 App，讓使用者前往處理。

2. 車子

本專題已經將車子的 LoRa 導航器以 Raspberry Pi、GPS、LoRa module 三個元件完成，導航器可以得知目前是否有訂單，若現在有訂單，則能夠得知寄送者、收貨者、寄件地、收件地、收件密碼等資訊，而根據寄件地與收件地，以及自身 GPS 座標，導航器會在虛擬座標中利用 A*演算法計算出最短路徑，寄送員(導航器使用者)便能依據顯示於螢幕上的路徑做移動，移動途中導航器也會提示寄送員該前往的實際方向，而當寄送員跟收貨者交付包裹時，導航器可以提供密碼驗證功能，來確認領貨人之身分，從開始處理訂單到訂單送達的期間，導航器也會在每個步驟環節回傳訂單 State 至 Server，由 Server 控管訂單狀態並對包裹相關者之 APP 做推播，導航器因此有效的整合 Online-to-Offline E-Commerce Platform 之中的線下服務部分。

Figure 9 CMD 導航介面

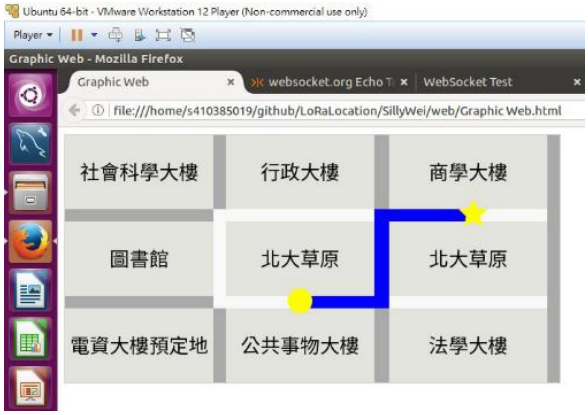

Figure 10 圖形化導航介面

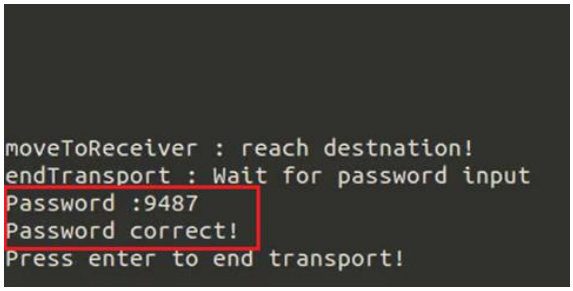

Figure 11 CMD 密碼驗證

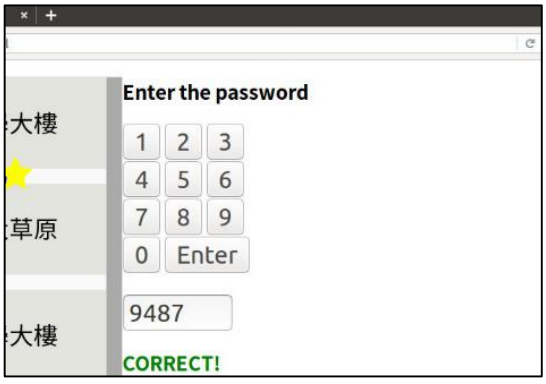

Figure 12 圖形化密碼驗證

3. LoRa gateway

本專題能使用 LoRa Gateway 透過 LoRa 和有線/無線網路完成資料的發送/接收功能，以 Multi Thread 的方式進行資料的撰寫，而最終程式架構如下圖，一開始先設定機器，設定的過程當中也包含 TCP 連線的設定、LoRa 模組的啟用，設定完成之後會透過兩個 Thread

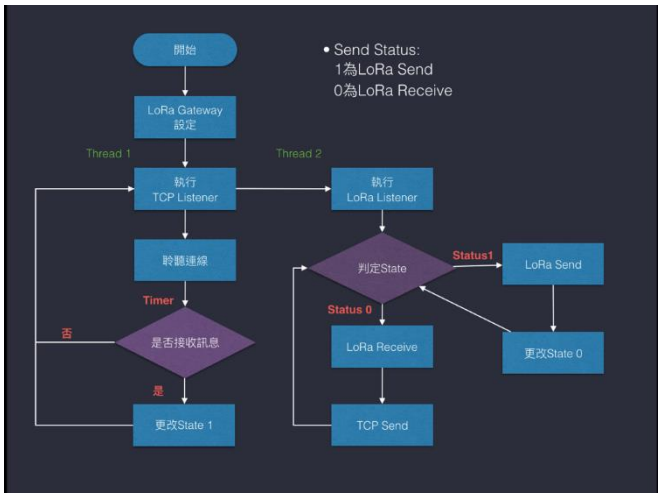

Figure 13 LoRa Gateway 程式架構

的方式進行程式的執行，其中一個 Thread 負責聆聽 Server 的訊息，而另一個負責聆聽車子的訊息，由於 LoRa 在傳輸過程無法同時收送，因此我們利用開關建立 LoRa 在單一時刻只能進行接收或傳送事件，並且在等待時間設定上必須與車子擁有不同的頻率，便能順利的完成接收訊息的任務。

4. 手機 App

本專題 App 提供以下功能：

(A) 登記寄件：使用者登記完必要資訊後，訂單即建立完成。

(B) 查看今日包裹：主畫面下方顯示今日所有包裹狀態，方便使用者查看。

(C) 寄件、收件通知：當車子到達寄件地時，手機會跳出通知提醒寄件者應寄放包裹，而車子到達收件地時，手機同樣也會跳出通知提醒收件者領取包裹。

(D) 查看寄件或收件歷史紀錄：輕易尋找歷史的寄件或收件紀錄。

(E) 更改個人資料：更改信箱、密碼以及大頭貼，讓使用者可以選擇、拍攝大頭貼讓使用者體驗更有趣。

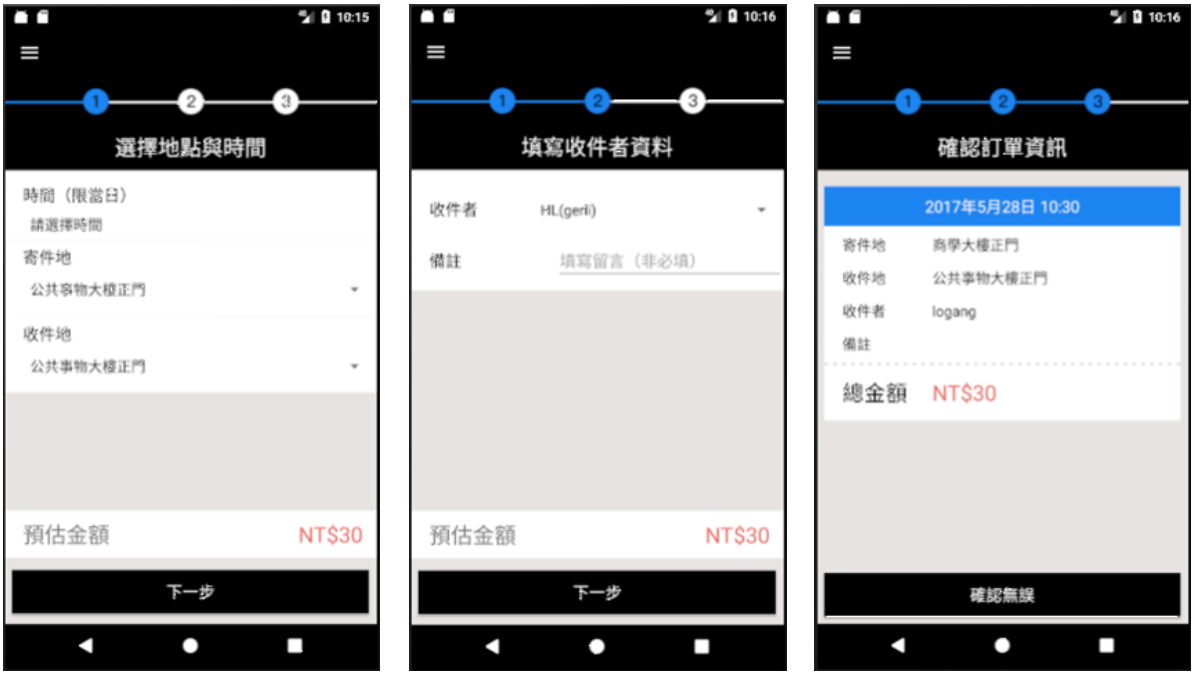

Figure 14 登記寄件

Figure 15 檢視訂單

Figure 16 個人資料

Figure 17 歷史記錄

Figure 18 貨物未處理

Figure 19 出貨處理中

Figure 20 貨物到達

Figure 21 系統通知

（二） 總系統成果

透過車子、Server、LoRa Gateway 以及手機的串聯在本次的專題中我們能夠過手機 APP 進行登記事件，透過登記事件得以將資料傳遞給 Server 進行資料處理，並將資料回送資料庫，且於登記事件的起始時間前十分鐘將資料進行發送，再透過 TCP Socket 進行連線，傳遞至 LoRa Gateway，並且再透過 LoRa 傳遞給車子，此時車子便會接收資料，並開始運作，透過我們自己設定的導航系統，顯示導航路線，而我們能夠控制車子依照導航路線的指引將到達

指定的寄件地，並等待送件者包裹，之後，車子便能繼續按照指引到達目的地，此時收件者只要能夠從手機 APP 中看到送件的密碼，便能完成一整套的寄件流程。此外，我們還能透過 Server 和資料庫的溝通儲存使用者資訊、地點資訊、訂單資訊以及車子資訊與手機 APP 進行結合運用，打造出一項 LoRa Logistic Service 物流服務。

（三） 預估與實際差距

本系統各部分(車子、Server、LoRa gateway、App)皆整合完成，在整體網路傳輸方面，除了偶爾 LoRa 因為封包 ACK 確認所造成的些許延遲之外，整體系統皆能配合運送流程完成訂單，並且正確呈現訂單資訊，而在導航器與車子控制整合部分中，導航器可以做導航以及對車體做基本控制，而 GPS 對應路徑之精準度目前存在誤差，因此車體的控制只能依循大致的路徑走，而車體之避障與路徑校正將來能透過補助 Sensor 如紅外線或電子羅盤來進行，使自走車行走更加順利，達到完全無人運輸服務。

五、結語與展望

智慧物流的興起，以及 O2O 模式電子商務(Online-to-Offline E-Commerce Platform)的實現，證明車聯網的進步可以帶給人們莫大的便利，而 LoRa Logistic Service 物流服務使 LoRa 發揮其長處，運用其長距離傳輸之特性達到車聯網系統實現，配合手機 App 給予良好的使用者體驗，使整個系統使用起來更簡易，期許未來能實現於生活當中，使整個城市更加貼近智慧物流。

六、工作分配

黃俊瑋	伺服器程式、導航圖形化介面
耿楷寧	LoRa Gateway 程式、手機 APP 介面
胡育旋	App 程式
張紘綸	LoRa 導航器程式、車體控制

七、銘謝

2016 NTPU Wireless and Mobile Network Laboratory

國立台北大學資訊工程研究所 無線暨行動網路實驗室

八、參考文獻

[1] Biggs, P., L. Srivastava, and I.T. Union, "ITU Internet Reports: The Internet of Things: International Telecommunication Union", 2005.

[2] Ashton, K., "That ‘internet of things’ thing. RFiD Journal", vol **22**, p. 97-114. 2009.

[3] Brody, P. and V. Pureswaran, "Device democracy: Saving the future of the internet of things", IBM, Sep. 2014.

[4] Centenaro, M., et al., "Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios", IEEE Wireless Communications, vol **23**, p. 60-67, 2016.

[5] Vangelista, L., A. Zanella, and M. Zorzi. "Long-range IoT technologies: The dawn of LoRa™". in Future Access Enablers of

Ubiquitous and Intelligent Infrastructures. 2015.

[6] Bor, M.; Vidler, J.E.; Roedig, U. Lora for the Internet of Things. In Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks (EWSN ’16), TU Graz, Austria, pp. 361–366, 15–17 February 2016.

[7] Mohammed, F., et al. "UAVs for smart cities: Opportunities and challenges", Unmanned Aircraft Systems (ICUAS), 2014.

[8] Adrienne Welch, Bruce Hutchinson, et al. "A cost-benefit analysis of Amazon Prime Air", University of Tennessee at Chattanooga, UTC Scholar, March 2015.

[9] Chen, W., et al. "Wits: A wireless sensor network for intelligent transportation system", Computer and Computational Sciences, 2006.

[10] Petajajarvi, J., et al. "On the coverage of LPWANs: range evaluation and channel attenuation model for LoRa technology", ITS Telecommunications (ITST), 2015.

[11] 劉孝忠，"整合超音波避障與衛星定位及羅盤定向能力之自動導航自走車設計與製作"，中華科技大學學報, p. 91-111. 2014.

[12] 林兆欣，"無人載具之避障與路徑規劃"，國立臺灣大學應用力學研究所, p. 41，2008.

[13] 彭祥瑋，"行走機器人避障路徑規劃演算法之配置與實現"，國立中興大學生物產業機電工程學系所, p. 80, 2010.