無人超市

陳佑丞/王長盛/呂長遠/賴宏運/范育誌/王楷儒/丁德榮（指導教授）

國立彰化師範大學資訊工程學系 500 彰化市進德路一號

Email：alen0216@gmail.com, deron@cc.ncue.edu.tw

摘要

本專題提出一套以客戶操作方便為基礎的無人超市系統，結合網站、光達導航、MySQL 數據處理與圖像辨識，實現高效、自主的商品取貨與管理流程。利用光達掃描技術生成精確的場地地圖，並透過路徑規劃算法優化機器人導航，確保取貨過程準確且流暢也能提升營運效率。本系統旨在利用無人的方式降低人力成本並且以客戶操作方便為導向，打造一個智能化、高效率且以使用者為中心的無人超市解決方案。

關鍵詞：無人超市、MySQL、SLAM、光達、圖像辨識、ROS。

Abstract

This project proposes a customer-friendly unmanned supermarket system, integrating a website, LiDAR navigation, MySQL data processing, and image recognition to achieve efficient and autonomous product retrieval and management processes. By utilizing LiDAR scanning technology, the system generates accurate venue maps and optimizes robot navigation through path planning algorithms, ensuring precise and smooth product retrieval while enhancing operational efficiency. The system aims to reduce labor costs through unmanned operations and prioritize user convenience, creating an intelligent, high-efficiency, and user-centric solution for unmanned supermarkets.

Keywords: unmanned supermarket, MySQL data processing, LiDAR scanning and mapping, image recognition.

一、簡介

1.1 研究動機

隨著科技的進步與消費模式的變化，無人化技術已成為零售業的重要發展趨勢。在生活中可以觀察到，像是 PChome、蝦皮這類的網路商店[20][21]，目前仍未完全實現無人化的營運模式[4][5][6]，表 1 為比較表。因此，設計一套完整的系統以盡可能實現無人超市的概念具有研究價值。無人超市不僅能有效降低人力成本，還能提供 24 小時不間斷的服務，進一步滿足現代消費者對便利性的需求。然而，目前市場上的無人超市仍存在一些待改善的問題，例如商品取貨過程可能出現延遲或錯誤，商品推薦系統缺乏個性化考量，並且用戶在搜尋商品時需耗費較多時間。這些現象表明現有系統仍有提升空間。

表 1.B2C 電商平台 vs 無人平台

功能/特性	B2C 平台	無人平台
24 小時購物	V	V
即時取得商品	X	V
物流配送	V	X
無人結帳技術	X	V
需要人力支持	V	X
技術維護需求搞	X	V

1.2 研究目的

本專題的研究目的是開發無人超市系統，透過整合網頁資料庫、光達導航、MySQL 數據處理圖像辨識與 ROS [9] [10] [11] [12] [22] 等技術，設計一套更加高效且智能的無人超市系統。此系統的目的是解決現有系統在商品管理、路徑規劃與用戶體驗上的痛點，藉此提升運營效率，優化用戶體驗，並展現無人化技術在零售業的更多應用可能性。在技術成面，我們的系統特色如下：

1. 建立模組化系統，分成網頁：PHP[3]、MySQL[18]、後端：Node.js[19]、以及小車控制：ROS、YOLO v5[14]，使我們在製作時能夠各自進行以加快整合速度，並能夠簡易的將不同系統合併。維護時，也能針對各自系統做維護，不需要所有系統皆做修整。
2. 網頁能透過 API，即時獲取超市內小車理貨與運送狀態，方便管理者確認小車狀態和客戶確認訂單狀態。
3. 使用 YOLO v5 模型辨識商品，使小車能夠更精確的辨識商品，並取得正確商品。
4. 小車控制透過建立命名空間 (namespace) 與多主機架構 (Multi-Master System)[12] 來同時控制多台小車，增加超市內小車取貨效率。
5. 透過雲端服務的 PaaS(Platform as a Service)與 IaaS (Infrastructure as a Service) 使網頁與 API 雲端化，以便管理與開發並使小車控制端能夠在不同內網環境下

建設。

二、相關產品與應用技術

隨著人工智慧、物聯網、和機器視覺等技術的快速發展，零售業進入了一個智能化和自動化的新時代。無人超市作為新興的零售模式，以其高效、便捷和創新的特性，正逐漸改變消費者的購物方式。目前無人超市普遍的做法是使用 Grab-and-Go 模組化智慧商店技術[23]，這技術為利用人員追蹤、商品辨識等技術運作，也就是客戶需要進入商店挑選商品，然後用 AI 的方式進行結帳。但這樣的方式客戶依然要進入商店挑選商品，且進出商店都需要識別，在人潮多的時候還是會擠在超市裡。我們的無人超市系統將購物流程從傳統的實體挑選轉移到線上平台，客戶可透過網站完成商品瀏覽、選購和下訂單，系統隨即指派自動化機器人負責取貨並將商品存放到指定的智能貨櫃中。客戶僅需到貨櫃完成取貨，全程無需進入商店，節省時間並避免人潮擁擠的問題。

2.1 系統功能

自動化商品取貨與配送:機器人通過光達掃描生成環境地圖，結合路徑規劃算法， 自主完成商品的精確取貨與配送。

實時庫存管理:利用 MySQL 數據處理銷售數據，實現商品進出庫記錄的自動更新，並發出低庫存預警。

2.2 系統應用

零售業自動化升級:無人超市系統適用於各類零售場景，如便利商店、大型連鎖超市，減少對人力的依賴，降低營運成本。

支持疫情後非接觸式經濟模式:提供無接觸的購物環境，符合消費者對安全與衛生的需求，提升市場競爭力。

2.3 系統相關技術

2.3.1 網站技術

網站系統採用 PHP[3]、JavaScript[16]、Bootstrap[2] 、 HTML[1] 、 CSS[17] 以及 MySQL[18] 進行開發，透過完整的結構與功能設計，實現了穩定、安全且響應式的使用者體驗。以下為系統採用的相關技術：

(1) HTML (HyperText Markup Language):使用 HTML5 描述網頁的結構與內容，包含基本結構元素，如 header、form、input、button、div 等。

(2) CSS (Cascading Style Sheets):用於定義頁面的樣式與佈局，實現一致且直觀的視覺效果。此外使用 Bootstrap 5 框架，快速建立 RWD 響應式設計，以支援不同裝置適配。

(3) JavaScript:用於客戶端交互與表單驗證，增強用戶操作體驗，例如彈出提示窗等功能。此外採用 jQuery 函式庫簡化 JavaScript 操作，利用 jQuery Validation 插件進行表單驗證，如檢查帳號與密碼的長度與格式等功能。

(4) PHP:主要用於伺服端處理邏輯與資料庫交互。使用 session 記錄登入狀態、購物車內容與管理員身份等資訊，確保使用者體驗的連續性與系統管理的安全性。使用 strip_tags() 過濾用戶輸入，防止 SQL Injection、XSS 攻擊，並確保資料安全。

(5) MySQL:作為資料庫，儲存所有網站所需資訊。

(6) 資料通訊與 HTTP 請求:使用 HTTP GET 請求參數與後端系統進行資料通訊，實現快速高效的資料交互。

2.3.2 後端技術

我們採用 Node.js 開發 API，這是因為 Node.js 採用非同步、事件驅動的架構，可以在單一執行緒中同時處理大量請求，提升系統效能與擴充性並降低因為同時多請求造成資料庫錯誤的可能。其次，因為 Node.js 與前端同樣使用 JavaScript 語言，開發者無需在前後端切換不同語言環境，大幅降低學習成本並提高開發效率。此外，Node.js 有全球最大的套件管理系統 npm，可方便地整合各種第三方函式庫與工具，為專案帶來靈活性和高可維護性。這些特性使得 Node.js 特別適合需要快速迭代、即時互動或高併發量的應用程式。

2.3.3 小車控制技術

(1) ROS（Robot Operating System）[22]：開發機器人軟體常用的一套開源系統，具備可擴充性、模組化架構、以及龐大的功能資料庫，使我們能專注於整體系統開發，不必花費大量時間在韌體或次要功能開發上。

(2) JetBot：為 NVIDIA 公司的產品 Jetson nano[24]開源機器人。主要以 Jetson nano 作為運算核心，並附有避免碰撞、物件跟隨、道路跟隨等程式碼。開發者可自身條件使用各式硬體設備，如： WiFi、相機、馬達等，並安裝相對應的驅動程式，便可透過 JetBot 系統操作該機器人。除了 NVIDIA 公司有提供相關硬體

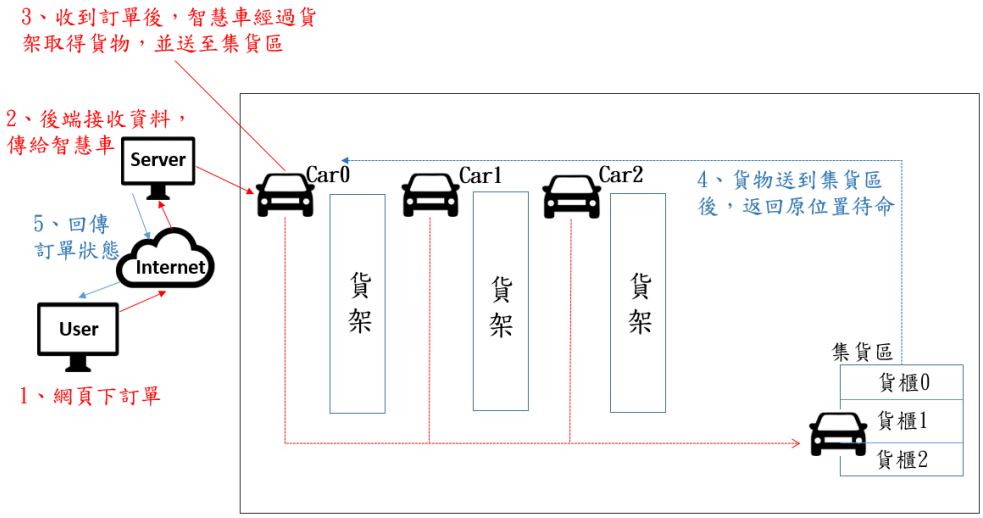

圖 1. 系統運作流程

外，也有第三方公司有製作 JetBot 機器人所需硬體，例如本專題使用的 SparkFun 公司製作的 JetBot[25]。此外，藉由 Jetson nano 的 GPU 計算能力，開發者可以在 JetBot 上使用使用 AI 模型，如：YOLO[14]，並開發出更多實用的系統或產品。

(3) YOLO（You Only Look Once）[14][26]：是一個 one-stage 的 object detection 演算法，對於一張影像只需要運算一次 DNN（Deep Neural Network）就可以一次預測出多種物體位置與其對應類別。透過 end-to-end 的架構，YOLO 能夠在實現 real-time 的情況下仍保持較高的準確度。YOLOv5 相較於 YOLOv4 使用上更加容易、具備較高的穩定性、且擁有更高準確度[26]。而對於 YOLOv5 之後的版本，如：YOLOv8、YOLOv10，雖然比起 v5 更加準確且計算速度更快，但因為 SparkFun JetBot 的版本限制而無法使用。因此，在本專題中我們使用 YOLOv5 模型。

三、系統架構

3.1 系統運作流程

我們設計系統運作流程如圖 1。用戶首先需完成系統的註冊並登入，此步驟用於識別用戶的身份。完成登入後，用戶進入商品瀏覽界面，可以瀏覽並挑選商品，若購買數量超過庫存數或是為 0 時則無法購買。當用戶選定商品後，通過系統提交訂單，系統即時接收來自用戶端的訂單資訊。在訂單生成後，系統準備安排無人車出貨。在此期間用戶可確認商品物流狀態。在無人車確認配送完成後，會自動更新客戶的訂單狀態為“已送達”。用戶需要在正確的貨櫃掃描 QRcode，輸入訂單編號與電話完成付款取貨。在付款取貨後，系統向用戶開放商品評價功能，用戶可對商品進行評價，如圖 2。

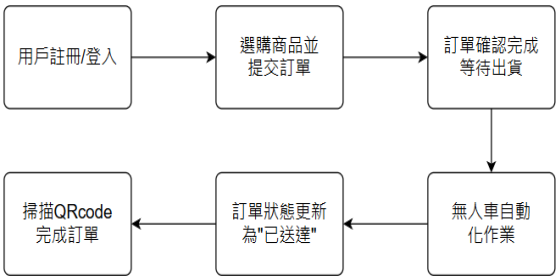

圖 2.用戶端運作流程圖

3.2 網站架構

3.2.1 網站權限

分為四個權限，分別為訪客、一般會員、訂單管理員、系統管理員，架構如圖 3。

- 訪客:僅能瀏覽首頁及查看、查詢商品。
- 一般會員:可以將商品加入購物車並下訂單，下完訂單後可查看訂單的狀態，完成取貨後可以對商品進行評價。
- 訂單管理員:查詢所有客戶的訂單，且可以修改訂單的狀態。
- 系統管理員:可以進行商品管理(新增、修改、刪除)、公告管理(新增、修改、刪除)、會員管理(刪除、修改)、訂單管理(修改狀態、刪除)、折價券管理(新增、刪除)以及查看小車當前狀態。

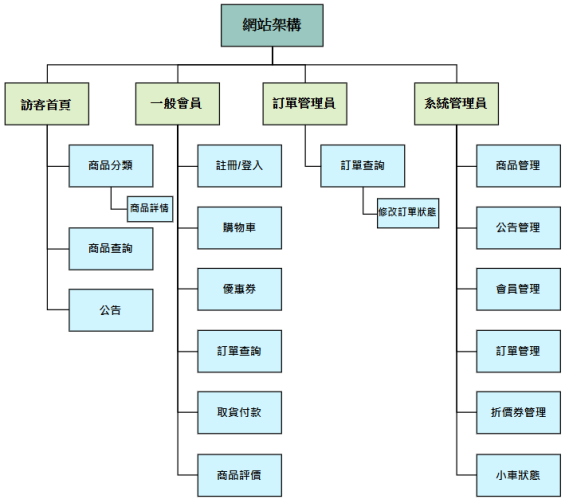

圖 3. 網站架構

3.3 後端架構

3.3.1 後端任務

以 nodejs 承接來自 php 發出的要求。透過 python 跟小車溝通，並回傳對應訊息同時更改資料庫。紀錄包含訂單資訊、小車資料、商品資料跟收貨點資料，如圖 4。

圖 4. 後端互動示意圖

3.3.2 資料庫架構

後端與前端修改同個 mysql database 如圖 5，其中後端主要有 6 個重要的資料庫表格 如圖 6。主要分成 2 類如下:

(1)車輛狀態/商品規範的紀錄:

- Carstatus:用於紀錄車輛狀態
- merchanregion:紀錄 merchanid 對貨櫃 id 的對應
- Pickupstation: 紀錄當前取貨點的狀態，用於分配

(2)訂單管理:

- Tasksinglerow:以單個 row 為單位，紀錄訂單(一個訂單可能有多個 row)
- Tasktransporting:紀錄尚未完成搬運，即因為容量限制沒有一次就被搬完的 row。但由於假設中任何 row 都能被一次完成，所以理論上不會有資料
- merchantransporting:紀錄當前小車上的貨物，有 carid 欄位用來辨別是哪台車所載的貨物

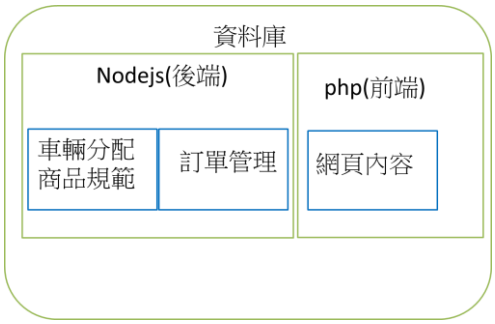

圖 5. 資料庫整體示意圖

圖 6. 後端資料庫表格及欄位

3.3.3 命令整理

以 ip+?command=…辨別任務類型，前端跟 ROS 需要發送的 Command 包括:

(1) addTask (2) takeTask (3) assignPs

(4) startTransferring (5) transferArrive

功能分別如下:

(1) addTask:接收前端請求並依照規範的格式，以單個 taskid-商品 id 為單位在資料庫內建立用於小車溝通的訂單。如圖 7。

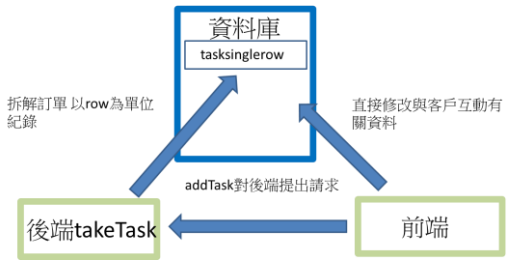

圖 7. 後端流程(前端下訂)

(2) takeTask:小車會不斷透過 API polling 這個 url 直到有 task 為止，若有可執行的 row 會回傳訂單資訊，反之則回傳錯誤訊息，如圖 8。

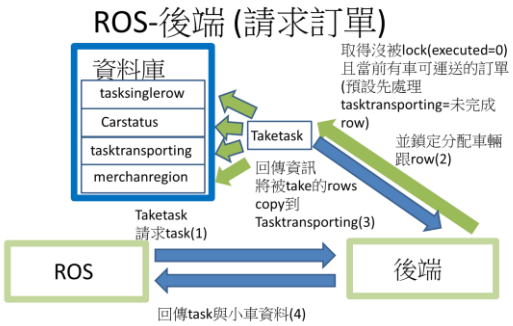

圖 8. 後端流程(ROS 請求訂單)

(3) assignPs: 分配 idle 的收貨點給 ROS。

(4) startTransferring: 貨櫃取貨後更新小車上的貨物紀錄並回傳 success，如圖 9。

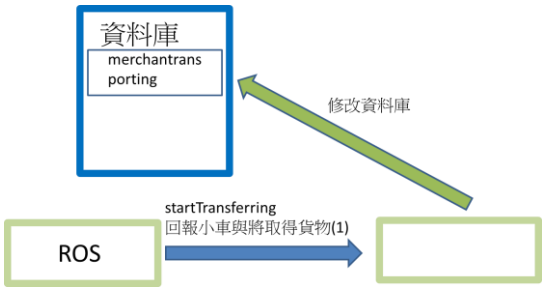

圖 9. 後端流程(ROS 回報從貨櫃取貨)

(5) transferArriving: 小車到達收貨點，則更新訂單狀況、清空小車貨物、並解鎖訂單跟小車，如圖 10。

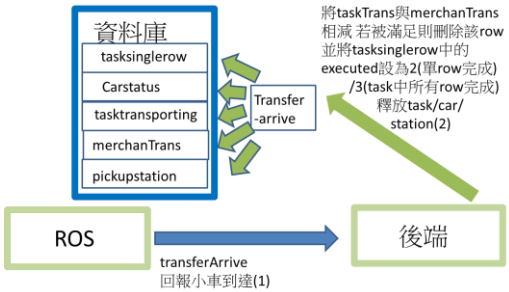

圖 10. 後端流程(小車到達收貨區)

3.4 小車控制架構

小車實際移動的方式主要是靠 ROS[22]系統完成，並分成車輛端與伺服器端。車輛端主要是發布硬體的資訊，如光達感測資訊到 ROS 的 Topic 中，或是 YOLOv5[14]辨識的結果。此外，還有從 ROS 接收些控制指令，如移動車輛、使用 YOLOv5 辨識等。伺服器端主要是透過 ROS 發布控制指令給小車，並接收小車的硬體資訊與 YOLOv5 辨識之結果。此外，還需要規劃小車的取貨與卸貨路徑。在實際架設之場地，我們放置三台 Jetbot 小車，並設定三個集貨區，分別代表不同的訂單編號。在地圖中，我們也定義了小車的 5 種區域，分別為小車初始區域、小車所屬貨櫃、小車等待區、小車排隊區、以及集貨區。實際場地的光達掃描結果與定義區域如圖 11。

小車拿取商品的流程如下：

(1) 根據訂單需求，小車拿取所屬貨櫃的商品。
(2) 小車前往等待區，並等待伺服器端下達前行指令。
(3) 小車接收前行指令並前往排隊區。
(4) 待前方小車進入集貨區後，方可根據訂單標號進入對應集貨區。
(5) 小車於集貨區卸貨完成後，回到初始地點。

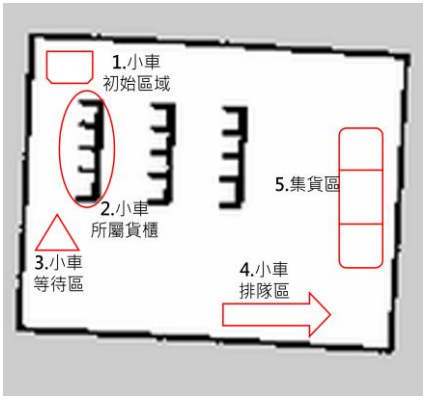

圖 11. 實際光達掃描結果與定義區域

伺服器派遣小車的系統架構圖如圖 12。我們主要分成 Main、Plan、以及 Blocking 程式。以下將介紹各程式的功能。

1. Main 程式主要負責接收 Node.js 的訂單資訊，並轉換成車輛的路徑資訊；此外，Main 程序會根據訂單所需小車運行子程序—Plan，已實做時使用平行化(Parallelize)，使小車能夠同時拿取商品，增加系統運行效率。平行化主要透過在 Main 程序使用多線程(MultiThreading)並於每一個線程中開啟 Plan 子程序，使 Main 程序能夠在每一個線程，也就是每一個小車完成拿貨與卸貨任務後才會 join，確保每次執行 Main 的原子性(Atomic)。
2. Plan 子程序會根據該車輛路徑資訊通知車輛依序前往目的地。
3. Blocking 程序主要是 Plan 子程序運行至等待區後，針對目前的車輛情況來決定車輛是否能繼續前往下一個目的地。

依據在 ROS[22]系統的通訊，Plan 會聆聽(Subscribe)個別小車的資訊，如位置資訊，並推送

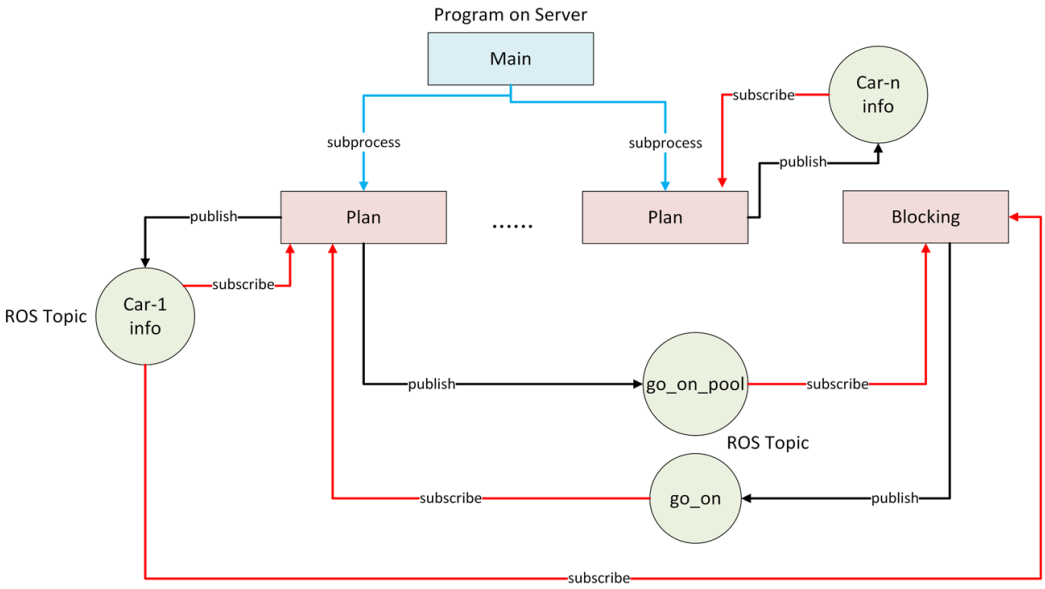

圖 12. 車輛派遣系統架構圖

(Publish)速度資訊給小車。不同 Plan 子程序都會聆聽"go_on"並推送佔用資訊至"go_on_pool"Topic，Blocking 程序則會聆聽"go_on_pool"並推送前行資訊至"go_on"Topic，使 Blocking 程序能夠規劃所有超市中的小車。

前述之 Blocking 程序流程圖如圖 13 所示。控制流程說明如下：

Step1. 需求分成「要前往等待區」、「要前往排隊區」、和「要前往集貨區」。

Step2. 對於「要前往等待區」執行 Step3，對於「要前往排隊區」執行 Step6，對於「要前往集貨區」執行 Step9。

Step3. 將該車輛編號(cid)紀錄到一陣列中並升冪排序。。

Step4. 根據排序後的陣列，紀錄其對應的集貨區至一陣列中。

Step5. 如需前往相同的集貨區，則將其排至下一順位，並回到 Step2。

Step6. 將其 cid 標記，使其不參與排序。

Step7. 根據 cid 的順序調整排隊的位置。

Step8. 如果周遭車輛與自身相差一定距離，則前往對應排隊位置並回到 Step2；否則繼續等待。

Step9. 確認是否為第一順位的 cid，是則前對應集貨區並回到 Step2。

Step10. 確認前方集貨區是否有其他車輛，是則回到 Step2。

Step11. 確認前面順序之 cid 是否已出發至集貨區，否則回到 Step2。

Step12. 確認集貨區陣列的對應集貨區的第一順位是否為自己，是則前往該集貨區；否則回到 Step2。

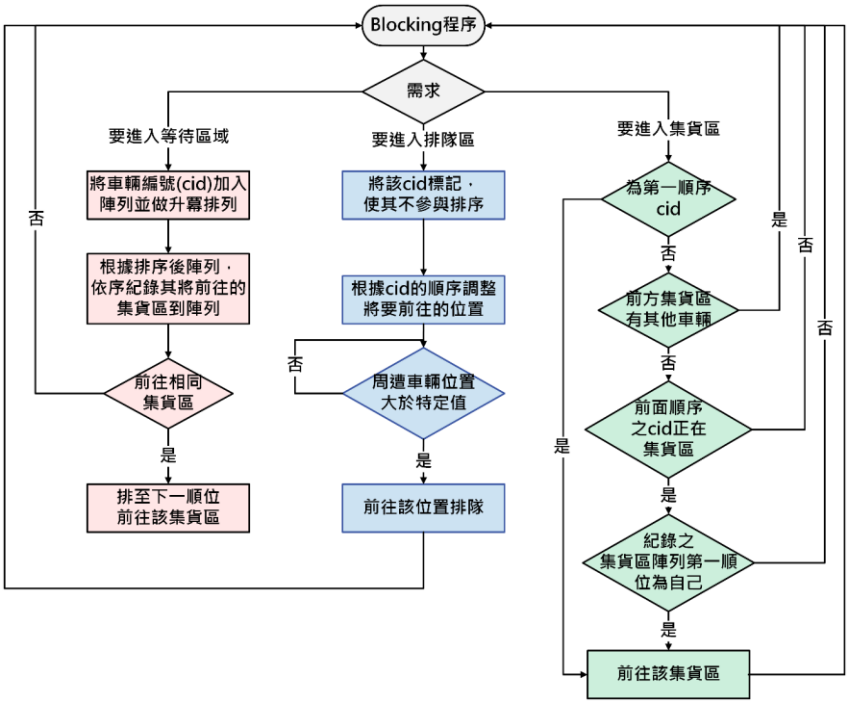

圖 13. Blocking 程序—車輛至集貨區流程圖

3.5 小車 ROS 基礎架構

3.5.1 小車架構概述

小車的主要模組包括：定位[8]與建圖模組[9][10]、路徑規劃模組[11]，以及控制指令執行模組。為了實現多小車的協同運作，系統採用了 ROS 的多主機架構 (Multi-Master System)[12]，伺服器作為主控節點負責協調多台小車的路徑規劃與任務分配。

3.5.2 單小車 TF 樹架構

在單一小車運作中，其 TF (Transform) 樹架構如圖 14 所示。小車的主要座標框架包含：

- odom：表示小車的里程計座標框架，追蹤小車的相對運動。
- base_link：代表小車的幾何中心，是小車運動學與動力學的基準點。
- scan：雷射掃描儀的座標框架，用於描述 LiDAR 感測器的數據來源。

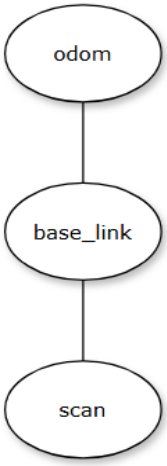

圖 14. 單小車 TF 樹架構

3.5.3 多小車 TF 樹架構

當系統運行多台小車時，伺服器需整合多台小車的資訊並在地圖框架 (map) 中協調其運動。每台小車在伺服器中擁有獨立的命名空間 (namespace)，對應的 TF 樹架構如圖 15 所示。

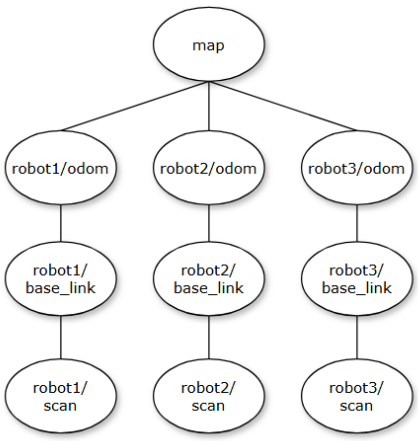

圖 15. 多小車 TF 樹架構

3.5.4 地圖生成與共享

地圖是小車自主導航的基礎。系統在部署前，利用一台小車事先掃描環境並生成靜態地圖文件，該地圖由伺服器管理，作為多車導航的全局基礎。

1.地圖建置流程：

- 利用單台小車啟動 gmapping 節點，結合 LiDAR 的 /scan 資料與小車的 odometry /odom 資訊，完成環境的地圖掃描。
- 將生成的地圖以靜態文件（如 .pgm 與 .yaml 格式）保存至伺服器。

2.地圖加載與共享：

- 在正式運行時，伺服器使用 map_server 節點載入地圖文件，並通過 /map 主題共享至所有小車。
- 每台小車的 move_base 節點通過伺服器提供的地圖，實現路徑規劃與導航。

3.5.5 ROS 節點與通訊

1.硬體節點：

- 發布 LiDAR 感測數據至 /scan 主題 (topic)。
- 發布 odometry (里程計) 資訊至 /odom 主題。

2.控制節點：

- 接收伺服器下發的導航目標點 (goal) 至 move_base 節點，並由 move_base 生成 /cmd_vel 指令來控制小車運動。
- move_base 的核心功能包括全局路徑規劃 (Global Planner) 和局部路徑規劃 (Local Planner)，通過環境感測數據和地圖信息生成動態避障的安全路徑。

3.6 YOLOv5 訓練模型

1.照片收集：使用 JetBot 小車的 CSI 攝像頭進行拍攝，調整小車位置並拍攝貨櫃中商品照片，拍攝圖片如圖 16。

圖 16. 小車 CSI 攝像頭拍攝照片

2.形成 dataset：將小車所拍攝的商品照片上傳到 Roboflow 網站[13]，此網站可將各商品照片進行標籤處理和類別命名，類別總共有 6 個，每個類別拍攝 30 張照片，並以拍攝照片為基準，生成額外處理(飽和度、模糊化、Noise、角度調整)過的照片，如圖 17 為模糊化前照片，圖 18 為經過模糊化後的照片。最後將原照片和生成的照片分類成 Train、Test、Vaild 三種分類，比例約為 8:1:1。

圖 17. 模糊化前照片

圖 18. 模糊化後照片

3.Yolov5 訓練：我們使用 YOLO v5n 模型進行轉移訓練（Transfer learning），使我們能用較少的圖片訓練出較為精確的模型。我們的訓練結果之混淆

矩陣（Confusion Matrix）如圖 19 所示，對於每一個類別的準確度（Accuracy）大多接近 100%，呈現我們的訓練結果良好。

4.Yolov5 辨識流程：在小車上運行 camera 程式已達到辨識功能並減少網路負載。流程圖如圖 20 所示。辨識流程說明如下：

Step1: 讀取訓練好的權重檔並開啟小車相機。

Step2: 等待伺服器傳送辨識指令，若接收到則進入 Step3。指令中包含要辨識商品名稱。

Step3: 將目前相機拍攝之圖片輸入至訓練後模型以進行圖像辨識。

Step4: 確認辨識結果是否為對應的商品名稱，是則通知伺服器辨識成功並回到 Step2，否則回到 Step3。

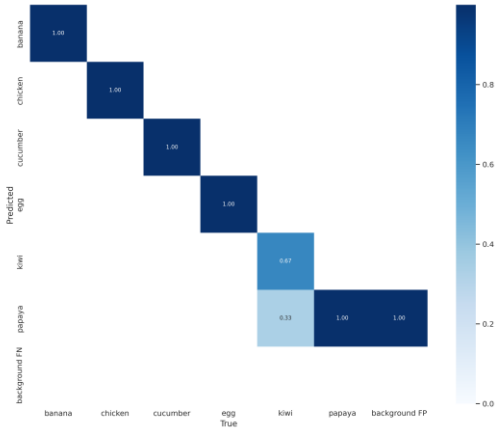

圖 19. YOLOv5 模型訓練結果圖

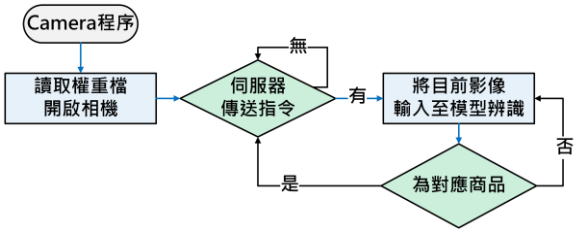

圖 20. Yolov5 辨識流程圖

四、實驗結果

在本專題中，我們設計並測試了一套基於 ROS 自動控制技術結合光達導航的無人超市小車系統。為驗證系統的導航能力，我們於模擬超市場景內進行多次實地測試。測試場地地圖，如圖 20 所示。

4.1 測試場地設計與尺寸

測試場地模擬了一個典型的小型超市，場地的實際尺寸為長 4.8 公尺，寬 3.6 公尺，內部設置包含貨架、通道以及集貨區域。小車利用光達掃描生成該環境的地圖，地圖生成結果提供了詳細的空間佈局，並作為路徑規劃的基礎。

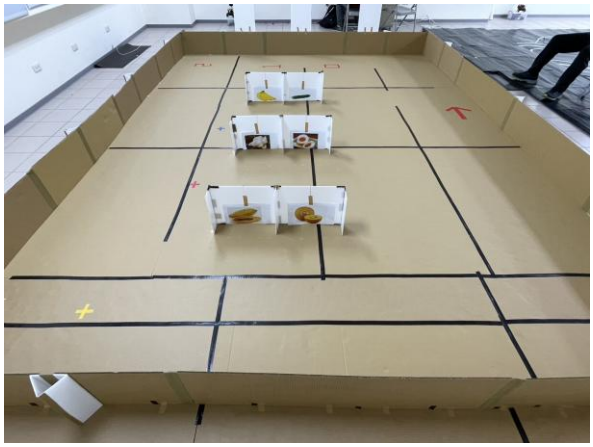

圖 21.測試場地地圖

4.2 系統運作結果：

因為較難以文字描述整體運作流程，因此，還請至附錄中觀看我們所拍攝之系統運作影片。

五、研究結論

透過這次專題做出的無人超市得出以下結論:

5.1 技術可行性

當前的機器人導航和物品檢索系統已證明其穩定性與可靠性，能滿足日常運營需求。同時，通過 PHP 前端與 MySQL 後端的緊密結合，實現了無人超市的高效用戶交互和數據管理，充分展現技術整合的優勢。

5.2 商業價值

無人超市能有效降低人力成本，提升運營效率，尤其在勞動力成本高昂的地區更具吸引力。此外，通過數據分析功能，無人超市能精準掌握用戶消費偏好，為庫存管理和促銷策略提供有力支持，進一步提升經營效益。

5.3 挑戰與改進空間

無人超市在運營層面，需建立高效的監控與維護系統，以應對機器人或系統可能發生的突發故障。在用戶體驗方面，系統在操作流暢性和友好程度上仍有改進空間，特別是在處理多筆訂單和高峰時段，需進一步提升穩定性與效率。

5.4 未來展望

1.技術升級：探索基於 AI 和計算機視覺的技術進一步提升商品識別和機器人運動規劃能力。

2.應用場景擴展：無人超市模式可推廣至更多場

景，如小型倉庫、社區便利店等。

3.可持續發展：加強節能設計和可回收包裝材料的使用，推進綠色運營模式。

六、參考文獻

[1] Html5 https://zh.wikipedia.org/zh-tw/HTML5
[2] Bootstrap https://getbootstrap.com/
[3] PhP https://www.php.net/
[4] 無人超市相關資訊 https://www.nownews.com/news/3308321
[5] 無人 711 https://www.thenewslens.com/article/88680
[6] 無人超市工研院電腦視覺自助結帳 https://www.itri.org.tw/ListStyle.aspx?DisplayStyle=01_content&MGID=1034303565072723774&MmmID=1036276263153520257&SiteID=1
[7] Google 雲端平台 https://cloud.google.com/
[8] ROS rf2o_laser_odometry https://wiki.ros.org/rf2o_laser_odometry
[9] ROS Gmapping https://wiki.ros.org/gmapping
[10] ROS Rplidar https://wiki.ros.org/rplidar
[11] ROS Navigation https://wiki.ros.org/navigation
[12] ROS MultipleMachines https://wiki.ros.org/ROS/Tutorials/MultipleMachines
[13] Robotflow Computer vision tools for developers and enterprises https://roboflow.com/
[14] Glenn Jocher et al. Yolov5. https://github.com/ultralytics/yolov5
[15] w3school https://www.w3schools.com/
[16] JavaScript https://zh.wikipedia.org/zh-tw/JavaScript
[17] CSS https://zh.wikipedia.org/zh-tw/CSS
[18] MySQL https://www.mysql.com/
[19] Node js https://nodejs.org/zh-tw
[20] PChome https://24h.pchome.com.tw/
[21] 蝦皮 https://shopee.tw/
[22] ROS https://www.ros.org/
[23] Grab-and-Go 模組化智慧商店技術 https://ictjournal.itri.org.tw/xcdoc/cont?sid=0N352578695359959792&xsmsid=0M236556470056558161&utm_source=chatgpt.com
[24] JetBot https://jetbot.org/master/index.html
[25] SparkFun JetBot AI Kit v3.0 Powered by Jetson Nano KIT-18486 SparkFun Electronics https://www.sparkfun.com/products/18486
[26] Muhammad Hussain, “YOLOv5, YOLOv8 and YOLOv10: The Go-To Detectors for Real-time Vision”, arXiv, 2024, https://arxiv.org/abs/2407.0298

附錄

圖 22. 運作流程影片連結