



靜宜大學資訊傳播工程學系 畢業專題報告書

專題名稱： AI送餐員

指導教師： 蔡英德

專題學生： 資工四B 胡秉翔 411018816

資傳四A 何允衡 411018434

資傳四A 蘇家宏 411019155

資傳四A 林誼均 411019197

中 華 民 國 113 年 10 月 18 日

摘要

近年來，無人送餐服務在食品配送、醫療保健和物流等多個領域蓬勃發展，尤其是在COVID-19疫情推動下，非接觸式配送需求顯著上升。本專題旨在開發一款基於ROS框架的AI智能送餐車，應用於校園內的主顧咖啡，以提升服務效率和顧客滿意度。該系統結合激光雷達和RGB相機等多種感測器，透過避障演算法和路徑規劃技術，能夠安全地在複雜環境中進行自主導航，並計算最佳行走路線，以確保配送的精準性和效率。

此外，系統整合了搭載GPT-3.5模型的LINE聊天機器人，實現自動點餐和結帳功能，不僅突破語言溝通障礙，還大幅減少人力資源的負擔，使消費者能有更多元且便利的選擇。消費者可以透過LINE或網頁平台輕鬆點餐，系統自動處理訂單並通知後台準備餐點。整個過程自動化程度高，無需人工干預，大幅提高了配送的效率和安全性。

該系統不僅解決了校園內的服務需求問題，還為未來的無人配送技術奠定了基礎，具備良好的擴展潛力，適用於更多行業領域。

目錄

摘要	i
目錄	ii
圖目錄	iv
一、前言	1
二、文獻探討	2
第一節 系統比較分析	2
第二節 系統SWOT	2
第三節 目標客群分析	3
三、系統架構與功能	4
第一節 點餐系統	4
第二節 送餐系統	18
第三節 整合系統	24
四、系統實作結果	25
第一節 點餐系統	25
第二節 送餐系統	27
第三節 技術特色及創新點	28
五、進行方法與步驟	29
第一節 採用方法與原因	29

第二節 詳細步驟	29
六、結論與未來展望	30
七、預期完成之工作項目及具體成果	30
第一節 工作分配	30
第二節 專題時程	31
第三節 討論紀錄表	31
八、附錄	34
第一節 參考資料	34

圖目錄

圖3-1-1	LINE點餐系統架構圖	4
圖3-1-2	Render部屬網頁	5
圖3-1-3	Github資料	5
圖3-1-4	LINE系統流程圖	6
圖3-1-5	虛擬付款頁面	9
圖3-1-6	網頁點餐系統架構圖	10
圖3-1-7	網頁點餐系統流程圖	11
圖3-1-8	Firebase訂單資訊	13
圖3-1-9	雲端Excel列表	14
圖3-1-10	語音點餐系統架構圖	15
圖3-1-11	語音點餐系統流程圖	16
圖3-2-1	車身.urdf視覺與座標描述圖	18
圖3-2-2	Arduino接線圖	19
圖3-2-3	激光雷達數據圖	20
圖3-2-4	底盤架構圖	21
圖3-2-5	Cartographer	21
圖3-2-6	全局成本地圖	22
圖3-2-7	局部成本地圖	23
圖3-2-8	導航架構圖	24
圖4-1-1	交互式點餐	25
圖4-1-2	菜單詳細資料	25
圖4-1-3	線上點餐系統	26
圖4-1-4	訂單更新至雲端Excel	26
圖5-2-1	前台與後台實際情境圖	29
圖7-1-1	工作分配	30
圖7-2-1	甘特圖	31

第一章 前言

當今科技快速發展，自動化和智能化已成為飲食業的主流趨勢。我們應對此需求，開發了一款結合點餐和送餐功能的智能送餐車，旨在提高餐飲服務的便捷性和效率。配備導航及避障技術，能自主在複雜環境中行駛，並結合人工智能技術提供個性化餐飲推薦，大幅提升顧客滿意度。

隨著 AI 技術在影像識別和感知方面的進步，無人送餐車的應用快速擴展到多個領域市場價值預計將顯著增長。增長動力來自於新冠疫情增加的無接觸送達需求、勞動力短缺和電子商務活動的增加。儘管技術面臨一些挑戰，如複雜環境的導航困難，但透過系統整合和算法改進，這些問題可以得到解決，使得智能送餐車更加普及化。

我們選擇校園環境作為實驗場景，發現學校整修主顧樓一樓休息區後，學生待在休息區的時間以及頻率增加，相對也提升學生購買在休息區旁主顧咖啡的意願，主顧咖啡雖然增加許多訂單，但同時也讓原本需等待的時間變長，讓消費者必須待在原地等待叫號。有鑑於此，本專題利用 ROS 的框架搭配多種感測器(例如激光雷達、編碼器、IMU)來相互補足，利用感測器的數據以及修改 ROS 套件參數配置來開發避障演算法和路徑規劃，提高整體的導航精確度和避障能力，幫助主顧咖啡將餐點運送到指定地點。在消費者方面我們會利用 ChatGPT 搭配 LINE 聊天機器人協助點餐及結帳部分，讓消費者只需要待在指定區域休息並等待餐點送達手上即可。

第二章 文獻探討

第一節 系統比較分析

本系統基於 ROS 平台進行**動態避障與路徑規劃**，在人流多的環境中展現良好表現。相較於市面上的送餐車，本系統具備高度的開放性與彈性，使用者可依需求自由整合新感測器或功能模組，而這是商業化送餐車難以達到的。此外，系統採用開源模組，顯著降低了成本。激光雷達和 IMU 等多感測器融合，使系統在靜態與動態障礙物處理上更為可靠，提升使用安全性。

項目	AI 送餐員	市面送餐車
導航系統	ROS Navigation	預設路徑導航
感測器配置	RPLIDAR A1、IMU、編碼器	單一雷達及相機
避障能力	支援靜態與動態避障	支援靜態與動態避障
人機互動功能	Google 語音 API，支援多語系	螢幕按鈕

第二節 系統 SWOT

優勢 (Strengths)：

- 利用 ROS Navigation，使送餐車在動態環境中具備精準避障能力。
- 結合多種感測器的集合達成系統安全性與穩定性。
- 使用開源套件，降低成本，相較於市面上的商業化送餐車具備更高的彈性與擴展性。

劣勢 (Weaknesses)：

- 系統對於環境感知依賴較高，若感測器性能不佳或失效，導航與避障可能受影響。
- 初期開發與測試成本較高，對於小規模商業應用推廣可能有挑戰。

機會（Opportunities）：

- COVID-19 疫情之後，無人配送需求增加，為市場帶來新的發展機會。
- 隨著 AI 與機器人技術的快速進步，系統可在更多場景與行業中擴展應用。

威脅（Threats）：

- 商業化送餐車逐漸普及，市場競爭加劇，技術創新與迭代成為必要。
- 使用環境的不可預測性，如高人流密度或惡劣天氣，可能增加系統運行風險。

第三節 目標客群分析

學生群體： 系統主要在校園中進行實驗，學生是其主要目標群體。由於學生對於快速、高效的點餐和送餐服務需求較高，該系統將能大幅提升他們的用餐體驗，尤其是在時間緊迫的情況下。

餐飲業者： 此系統能幫助咖啡店和其他餐飲業者提升服務效率，減少人工投入，縮短顧客等待時間，並且通過科技化服務增強顧客滿意度。

無人配送服務需求者： 隨著無人配送技術的應用擴展，系統未來可以運用到更廣泛的配送場景中，針對對時間效率與安全性有高要求的企業和個人提供服務。

第三章 系統架構與功能

第一節 點餐系統

本專題提供三種點餐系統，讓使用者可以依需求選擇適合的方式：

1. **網頁點餐系統**：設計給想快速完成點餐的使用者。我們設置了清晰的商品分類和購物車檢視功能，從下單到結帳僅需幾個按鍵，即可快速完成訂單。
2. **LINE 聊天室點餐**：這種方式讓使用者可以在 LINE 聊天中以互動式方式完成點餐。只需文字輸入，即可查詢餐點、價格，並將商品加入或移除購物車；此外，還能透過虛擬付款將訂單上傳到雲端，免去瀏覽商品頁面的步驟，即便服務人員忙碌，使用者仍可享受便捷的點餐體驗。
3. **語音點餐**：只需按下送餐車螢幕上的開始點餐」，我們的線上點餐助手便會啟動。使用者可以直接對著點餐助手說出想點的餐點，詢問價格或製作方式，語音輸入比文字更加直觀、便利。

Line點餐系統

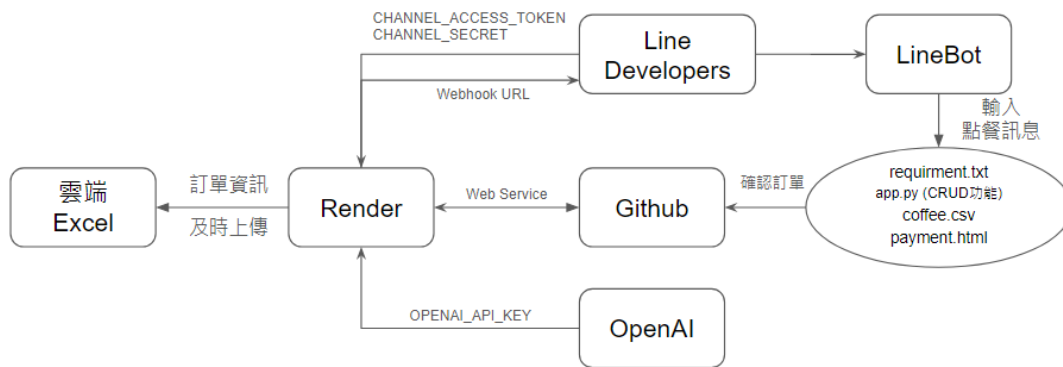


圖3-1-1 LINE點餐系統架構圖

(LineDevelopers 與 LineBot):

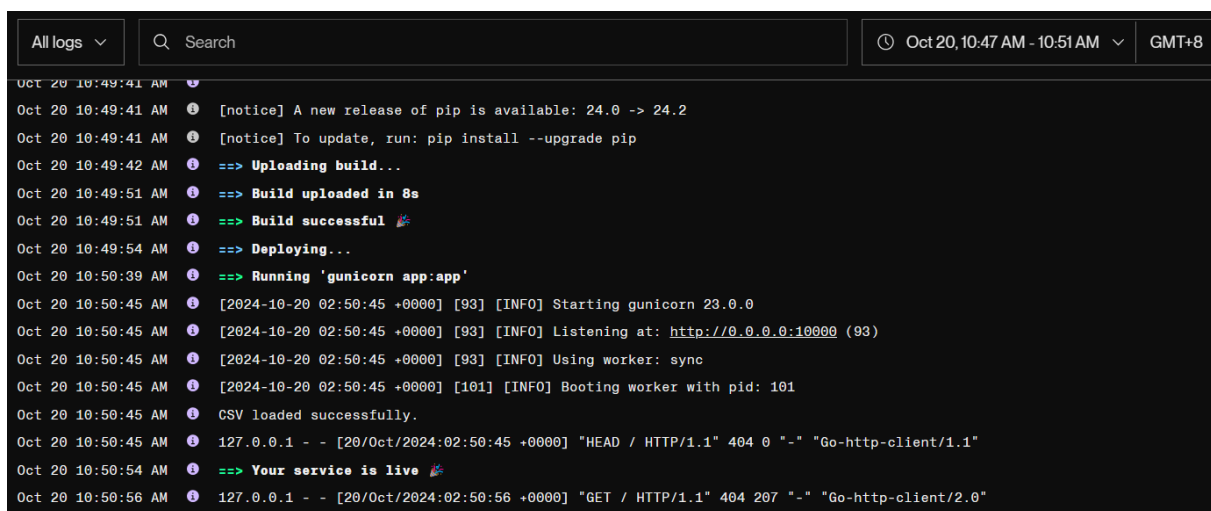
通訊: Line Developers 平台的 CHANNEL_ACCESS_TOKEN CHANNEL_SECRET 用於身份驗證和授權。Line Bot 通過 Webhook URL 與 Render 部署的服務進行通信將用戶的消息轉發給後端服務處理。

(flask):

flask 是一個輕量級的 Python 網頁框架，我們使用 flask 處理 LINE Bot 傳來的訊息請求、管理訂單邏輯、以及提供網頁端的付款確認功能。

(Render):

接收來自 LineBot 的 Webhook 請求，並將其轉發給後端服務進行處理。通過 Web Service 與 GitHub 進行結合，抓取應用程式的最新版本進行部署使用添加在 Render 上的 OPENAI_API_KEY，與 OpenAI 的 API 進行通訊，實現智能應用功能。



```
Oct 20 10:49:41 AM [notice] A new release of pip is available: 24.0 -> 24.2
Oct 20 10:49:41 AM [notice] To update, run: pip install --upgrade pip
Oct 20 10:49:42 AM ==> Uploading build...
Oct 20 10:49:51 AM ==> Build uploaded in 8s
Oct 20 10:49:51 AM ==> Build successful 🎉
Oct 20 10:49:54 AM ==> Deploying...
Oct 20 10:50:39 AM ==> Running 'gunicorn app:app'
Oct 20 10:50:45 AM [2024-10-20 02:50:45 +0000] [93] [INFO] Starting gunicorn 23.0.0
Oct 20 10:50:45 AM [2024-10-20 02:50:45 +0000] [93] [INFO] Listening at: http://0.0.0.0:10000 (93)
Oct 20 10:50:45 AM [2024-10-20 02:50:45 +0000] [93] [INFO] Using worker: sync
Oct 20 10:50:45 AM [2024-10-20 02:50:45 +0000] [101] [INFO] Booting worker with pid: 101
Oct 20 10:50:45 AM CSV loaded successfully.
Oct 20 10:50:45 AM 127.0.0.1 - - [20/Oct/2024:02:50:45 +0000] "HEAD / HTTP/1.1" 404 0 "-" "Go-http-client/1.1"
Oct 20 10:50:54 AM ==> Your service is live 🎉
Oct 20 10:50:56 AM 127.0.0.1 - - [20/Oct/2024:02:50:56 +0000] "GET / HTTP/1.1" 404 207 "-" "Go-http-client/2.0"
```

圖3-1-2 Render部屬頁面

(Github):

Render 平台從 Github 抓取程式碼(app.py、requirement.txt、coffee.csv、payment.html)，進行部署和運行。app.py 程式包含了應用程式和購物車(CRUD)的邏輯、requirement.txt 應用程式版本需求文件、coffee.csv 菜單文件、payment.html 模擬結帳頁面。

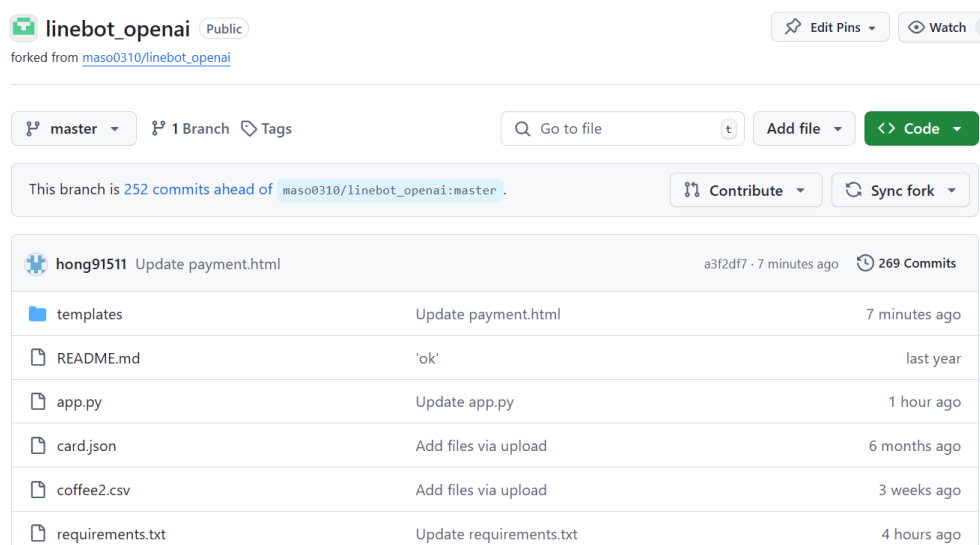


圖3-1-3 Github資料

(OpenAI):

應用程式使用配置在 Render 上的 OPENAI_API_KEY，與 OpenAI 的 API 進行通訊，當使用者傳入點餐訊息時，會調用 OpenAI 對訊息的內容進行回覆。

(雲端 Excel):

當使用者透過交互式訊息點完餐後輸入確認訂單，系統會提供一個模擬付款的連結給使用者並且可以輸入目前的桌號，在使用者完成模擬付款後，購物車中的金額、品項、餐點內容、桌號將會被上傳至雲端的 Excel 當中，並共享這個雲端 Excel 提供給商家，讓商家可以清楚的查看每筆訂單的內容。

LINE點餐系統流程



圖3-1-4 LINE點餐系統流程圖

- 使用者輸入:

使用者透過 LINE 聊天室與點餐系統互動，提供訂單或商品需求資訊。系統會調用 API，將用戶輸入的資料傳遞至後端進行處理。

- ChatGPT 回應:

系統接收來自 LINE Bot 的輸入後，透過提示工程請 ChatGPT 扮演線上咖啡廳助手讓 並對試用者輸入的內容進行回覆，包含將商品加入購物車、價格查詢、咖啡相關知識與使用者交互應用的指引。

提示工程

```
{"role": "system", "content": "你是一個線上咖啡廳點餐助手"}
```

```
{"role": "system", "content": "當客人點的餐包含咖啡、茶或歐蕾時，請務必回復品項和數量並詢問還有需要幫忙的嗎，例如：'好的，你點的是一杯美式，價格是 50 元，請問還需有要幫忙的嗎？' 或 '好的，您要一杯芙香蘋果茶，價格為 90 元。請問還有其他需要幫忙的嗎？' "}
```

```
{"role": "system", "content": "當客人說到刪除或移除字眼時，請務必回復刪除多少數量加品項，例如：'好的，已刪除一杯美式' "}
```

```
{"role": "system", "content": "answer the question considering the following data: " + info_str}
```

```
{"role": "user", "content": user_message}
```

● python 購物車程式

ChatGPT 回應的結果由 python 購物車程式解析，將回覆中相關的商品新增至購物車中，供使用者進一步確認。並擁有刪除購物車中商品、查詢購物車內容、修改購物車的功能。

(新增商品)

```
def add_item(cart, item_name, quantity):  
  
    if item_name in cart:  
  
        cart[item_name] += quantity  
  
    else:  
  
        cart[item_name] = quantity  
  
    return f"{item_name} 已加入購物車，數量: {quantity}"
```

(移除商品)

```
def remove_item(cart, item_name):  
  
    if item_name in cart:  
  
        del cart[item_name]  
  
        return f"{item_name} 已從購物車中移除。"  
  
    return f"{item_name} 不在購物車內。"
```

(查詢購物車)

```
def view_cart(cart):  
  
    if not cart:  
  
        return "購物車目前為空。"  
  
    return "\n".join([f"{item}: {quantity}" for item, quantity in cart.items()])
```

(修改商品)

```
def update_item(cart, item_name, new_quantity):  
  
    if item_name in cart:  
  
        cart[item_name] = new_quantity  
  
        return f"{item_name} 的數量已更新為 {new_quantity}。"  
  
    return f"{item_name} 不在購物車內。"
```

(匯出訂單至雲端 Excel)

```
import pandas as pd  
  
def export_to_excel(order_data, file_name="order.xlsx"):  
  
    df = pd.DataFrame(order_data)
```

```
df.to_excel(file_name, index=False)
```

```
print(f"訂單已匯出至 {file_name}")
```

- 虛擬付款頁面

在完成商品選擇後，使用者可於介面確認訂單內容，並選擇桌號或虛擬支付頁面進行付款。



圖3-1-5 虛擬付款頁面

- 訂單自動上傳 Excel

付款成功後，訂單資料自動匯出至雲端 Excel 文件，方便商家製作餐點和管理。

網頁點餐系統

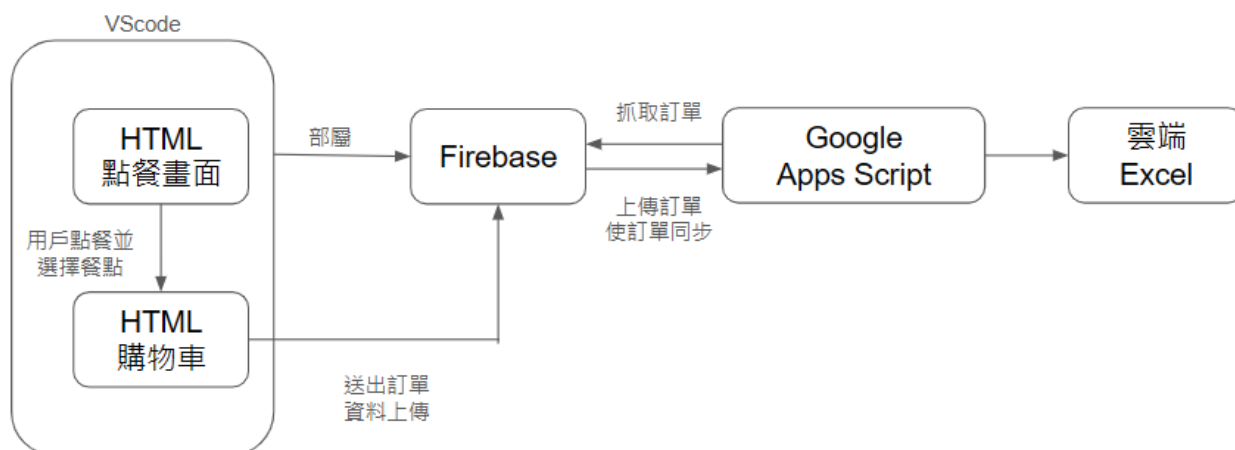


圖3-1-6 網頁點餐系統架構圖

(Visual Studio)

在開發 HTML 時，Visual Studio 提供程式碼自動輔助輸入和即時預覽的功能，助於網頁開發。並且 Visual Studio 可以輕鬆整合 CSS、JavaScript 語言，快速建立前端頁面。

(Firebase)

由 Google 提供的後端服務平台，用於開發即時互動的應用程式。Firebase 提供多項服務，如身份驗證、數據庫、儲存空間和推播通知。開發者可以快速連接應用並專注於功能開發。

(Google Apps script)

JavaScript 的雲端開發環境，可以用於 Google 服務以及其他 API 進行整合。當需要抓取 Firebase 資料庫中的資料時，Google Apps Script 可以發送 HTTP 請求來讀取和管理 Firebase 中儲存的數據。透過設置 Firebase 的 API 金鑰與資料庫網址，Apps Script 可以將 Firebase 資料直接取出並推入 Google Sheets 中，方便進行數據分析與報告。

(雲端Excel)

當使用者透過交互式訊息點完餐後輸入確認訂單，系統會提供一個模擬付款的連結給使用者並且可以輸入目前的桌號，在使用者完成模擬付款後，購物車中的金額、品項、餐點內容、桌號將會被上傳至雲端的 Excel 當中，並共享這個雲端 Excel 提供給商家，讓商家可以清楚的查看每筆訂單的內容。

網頁點餐系統流程

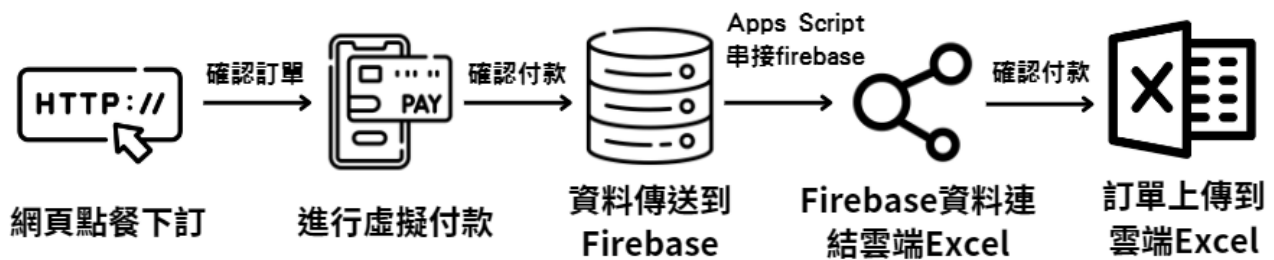


圖3-1-7 網頁點餐系統流程圖

● 使用者點餐：

使用者透過網站下訂單，可以快速又方便的點餐。首先進入網站，選擇桌號後開始檔案，選擇想吃的餐點，確認購物車內容並填寫姓名及電話送出訂單後完成付款，訂單及上傳至後台。

(HTML 購物車連結 firebase):

```
<script defer src="https://www.gstatic.com/firebasejs/8.0.1/firebase-app.js"></script>
<script defer src="https://www.gstatic.com/firebasejs/8.0.1/firebase-database.js"></script>
<script defer src="init-firebase.js"></script>
```

(收集訂單資料):

```
document.getElementById('order-form').addEventListener('submit', function (e) {
    e.preventDefault();
    const timestamp = new Date().toLocaleString();
    const tableNumber = localStorage.getItem('tableNumber');
    const name = document.getElementById('name').value;
    const phone = document.getElementById('phone').value;
    const payment = document.getElementById('payment').value;
    const notes = document.getElementById('notes').value;

    let totalPrice = 0;
    for (const item in order) {
        totalPrice += order[item].quantity * order[item].price;
    }
});
```



```
const orderData = {  
  timestamp,  
  tableNumber,  
  name,  
  phone,  
  payment,  
  notes,  
  order,  
  totalPrice  
};
```

(資料上傳到 Firebase):

```
firebase.database().ref('orders').push(orderData)  
  .then(() => {  
    // 將總金額存入 localStorage  
    localStorage.setItem('totalAmount', totalPrice);  
  
    // 清空當前訂單  
    localStorage.removeItem('order');  
    updateOrderList();  
  
    // 跳轉到付款頁面  
    window.location.href = '付款.html'; // 這裡進行頁面跳轉  
  })  
  .catch((error) => {  
    console.error('Error writing to Firebase Database', error);  
  });  
});
```

- Firebase 後台：

經過確認後的訂單資料會自動傳送到 Firebase 資料庫，Firebase 的即時資料庫能夠快速處理大量訂單資訊，確保資料的可存取性，便於後續管理和查詢。

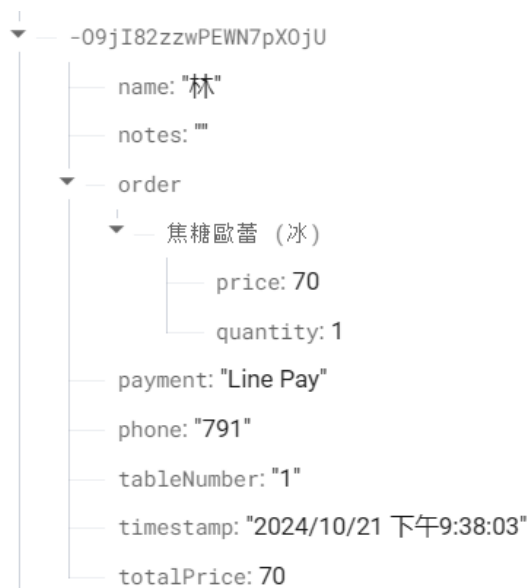


圖3-1-8 Firebase訂單資訊

- Firebase 資料庫連接雲端 Excel

為了讓網頁點餐系統訂單的資訊與 LINE 點餐系統的訂單資訊結合，我們透過 Firebase 與 Google Apps Script 的串接，資料能夠自動同步到雲端 Excel（Google Sheets）中。使用 Google Apps Script 編輯器新增程式碼，來自動化此同步過程，使得訂單資料無需手動上傳即可即時更新。

(初始化與資料庫連接)

```
function getAllData() {
  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var sheet = ss.getActiveSheet();
  var firebaseUrl = "https://coffeeshop-d6283-default-rtdb.firebaseio.com/orders";
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl);
  var data = base.getData();
}
```

(資料處理與格式化)

```
rows.push([
  order.timestamp, // 訂單時間
  order.tableNumber, // 桌號
  order.name, // 客戶姓名
  order.phone, // 客戶電話
  order.payment, // 付款方式
  formattedItems, // 格式化後的餐點資料
  total, // 訂單總金額
  order.notes // 備註
]);
}
```

(將資料寫入 Google 試算表)

```
if (rows.length > 0) {
  var dataRange = sheet.getRange(2, 1, rows.length, 8);
  dataRange.setValues(rows);
} else {
  Logger.log("No data available.");
}
} else {
  Logger.log("No data retrieved from Firebase.");
}
}
```

● 訂單上傳至Excel:

最終，處理後的訂單資料會出現在雲端 Excel 上，便於店家查看和管理。這樣的設計不僅提高了訂單處理效率，還減少了人為錯誤的發生，並為後續的分析和報表生成更加便利。

1	訂單時間	桌號	姓名	電話	付款方式	品項	總價	備註
6	2024/10/21 下午 9:38:03	1	林	791	Line Pay	焦糖歐蕾 (冰) > \$70 x 1	70	

圖3-1-9 雲端Excel列表

語音點餐系統

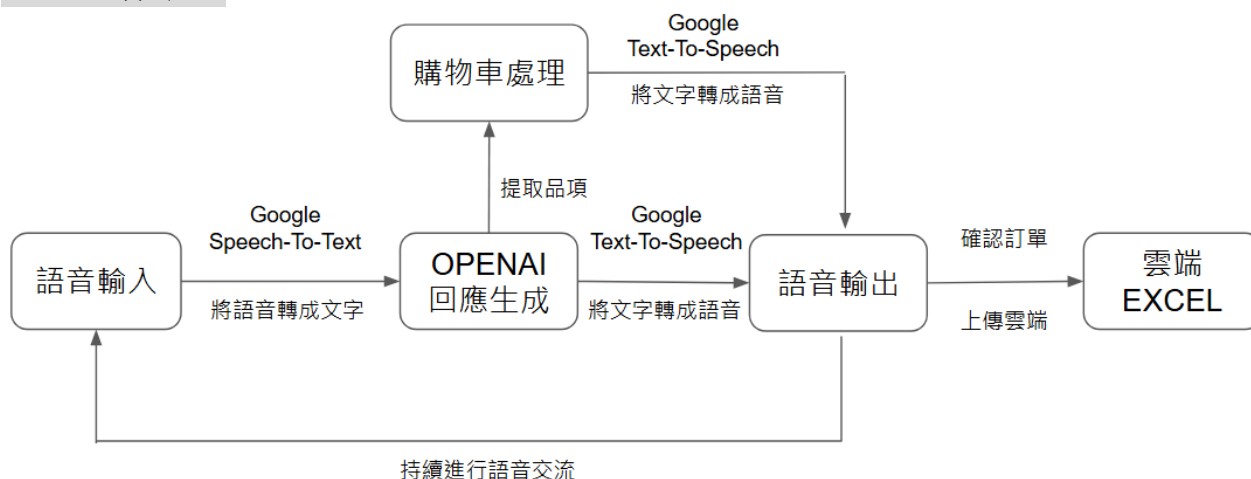


圖3-1-10 語音點餐系統架構圖

(語音輸入):

我們的語音點餐系統使用 Google Speech-To-Text 服務，將接收到的顧客語音轉換為文字。顧客可以透過語音點餐、查詢餐點等操作，這些指令會被即時轉換為文字，並傳送到 OpenAI，讓這些文字能夠被回應。

(OpenAI):

系統在接收到經由語音轉換成的文字訊息後，會使用 OpenAI 的 API Key 來處理文字內容並生成回應。無論是查詢菜單、添加至購物車，或是回答顧客的其他問題，OpenAI 都能根據提示工程的設置做出準確且友善的回應，提升顧客互動體驗。

(購物車程式):

當 OpenAI 回應顧客後，購物車程式會提取回應內容，並根據顧客的需求將相應的餐點加入或移除購物車。此外，購物車程式可以顯示最新的購物車訊息，例如已添加的餐點、數量、總價等，讓顧客隨時檢視和確認訂單狀況。

(語音輸出):

系統會使用 Google Text-To-Speech 將文字回覆轉換成語音回應，並播放給出來，使整個點餐過程更加順暢。我們的語音回應不僅包含確認添加的內容，像是「已將 1 杯美式加入購物

車」，還能提供其他互動資訊，例如當日甜點、菜單價格查詢，讓顧客使用起來更加的直覺、便利。

(雲端 Excel 上傳):

當顧客確認訂單後，購物車的完整內容（包括餐點項目、數量、總價等）會被自動上傳至雲端 Excel 作為訂單紀錄。這份雲端 Excel 會提供給商家，讓店家可以即時查看最新的內容，方便商家準備餐點、追蹤訂單以及庫存管理。

語音點餐系統流程

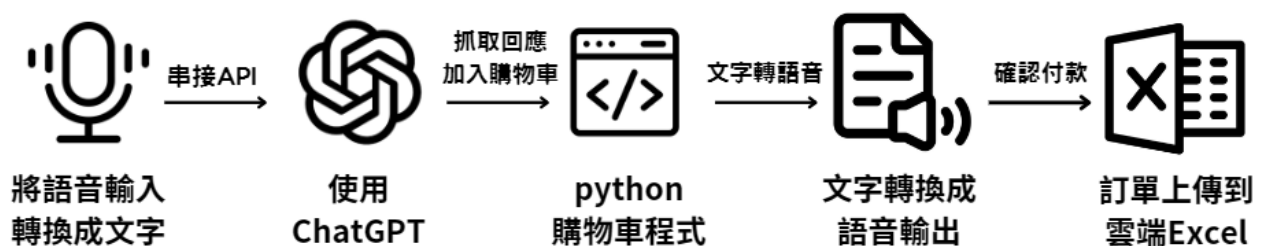


圖3-1-11 語音點餐系統流程圖

- 語音轉換文字:

在使用者按下開始點餐後，系統會開啟麥克風接收語音，並且系統會透過Google Speech-To-Text將語音轉換成文字傳入到系統中

```
def speech_to_text():
    with noalsaerr():
        recognizer = sr.Recognizer()
        with sr.Microphone() as source:
            print("請說話...")
            recognizer.adjust_for_ambient_noise(source) # 調整麥克風雜訊
            audio = recognizer.listen(source) # 聆聽用戶語音

        try:
            msg = recognizer.recognize_google(audio, language="zh-TW") # 語音轉文字
            print(f"語音輸入: {msg}") # 查看語音轉文字結果
            return msg
        except sr.UnknownValueError:
            print("無法理解語音。")
        except sr.RequestError as e:
            print(f"語音服務錯誤: {e}")
```

- ChatGPT回應:

轉換成的文字傳送至 OpenAI API，透過自然語言處理，ChatGPT 會解析使用者的需求，並進行回應。包含理解點餐內容、建議餐點、確認選項等，提供準確的餐點資訊和建議。

提示工程

```
{"role": "system", "content": "你是一個線上咖啡廳點餐助手，請以簡潔的方式回覆用戶。"} 
```

```
{"role": "system", "content": "當客人點的餐包含咖啡、茶或歐蕾時，請務必回復品項和數量並詢問還有需要幫忙的嗎，例如：'好的，你點的是一杯美式，價格是 50 元，請問還需有要幫忙的嗎？' 或 '好的，您要一杯芙香蘋果茶，價格為 90 元。請問還有其他需要幫忙的嗎？' "}
```

```
{"role": "system", "content": "當客人說到刪除或移除字眼時，請務必回復刪除多少數量加品項，例如：'好的，已刪除一杯美式' "}
```

```
{"role": "system", "content": "answer the question considering the following data: " + info_str}
```

```
{"role": "user", "content": user_message} max_tokens = 70
```

- 購物車程式

系統會抓取 ChatGPT 的回應，根據回應內容將商品加入購物車。此步驟由購物車的 Python 程式負責處理，確保選擇的餐點正確加入到購物車中。若使用者有變更需求，也可以即時刪除購物車內容。

- 文字轉語音輸出

為了完整的人機互動體驗，系統將回應內容轉換成語音，讓使用者能以語音形式聆聽點餐資訊，包括確認品項、詢問價格或加入購物車等功能。這使點餐過程更具臨場感和互動性。

- 確認訂單上傳至雲端Excel

當使用者確認訂單後，系統會自動將訂單資料匯出到 EXCEL 格式，並上傳到雲端儲存。

第二節 送餐系統

硬體架構

- 車體概述

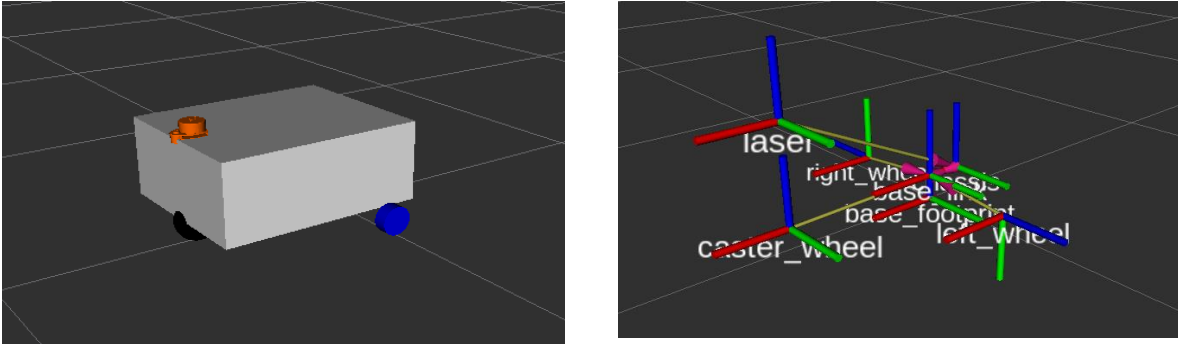


圖3-2-1 車身.urdf視覺與座標描述圖

- 核心控制器

NVIDIA Jetson Xavier AGX

作業系統：Ubuntu18.04，搭載 ROS Melodic系統。

用途：負責運算及導航路徑規劃、地圖建構與定位、語音處理與人機互動、控制指令與資料傳輸。

Arduino MEGA2560

用途：搭配DBH-1C 馬達驅動板控制JGB37-520帶編碼器減速直流馬達作為車身底盤的運動核心並且透過馬達編碼器精確測量輪子的旋轉速度與行駛距離計算里程數據，並進行資料傳輸，提供 Jetson Xavier 與底層馬達之間的即時資料傳輸與命令傳遞。

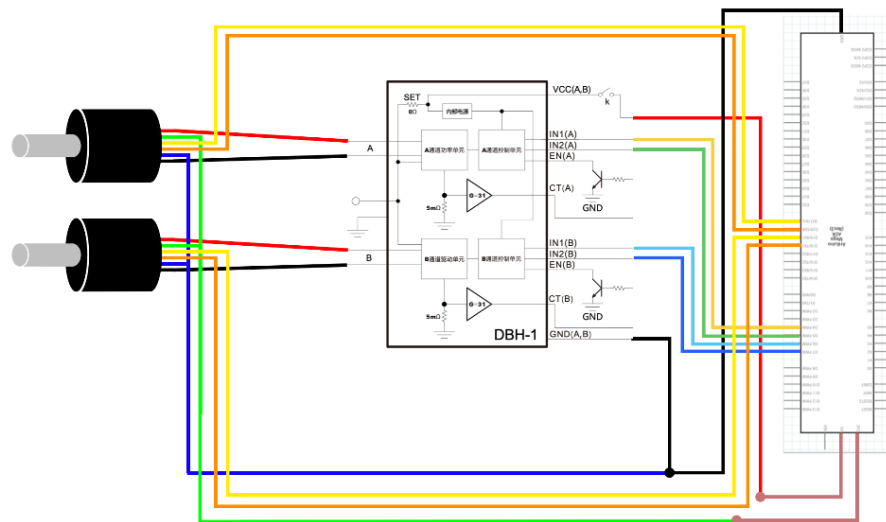


圖3-2-2 Arduino 接線圖

使用 `ros_arduino_bridge` 套件：

用途：透過修改此橋接功能套件，將 Arduino MEGA2560 與 ROS 系統整合，實現雙向的資料與命令傳遞，Arduino 收集的馬達編碼器里程數據會以 ROS topic 的形式發布，為導航與定位提供即時的輪速與里程資訊。

數據發布：

里程數據：`/wheel_odom`。

- 感測器

2D 激光雷達：Rplidar A1

功能：包含距離資訊和角度數據，分析環境中的障礙物位置並且掃描環境生成 2D 地圖，用於導航避障以及環境感知。

數據發布：

掃描數據：`/scan`

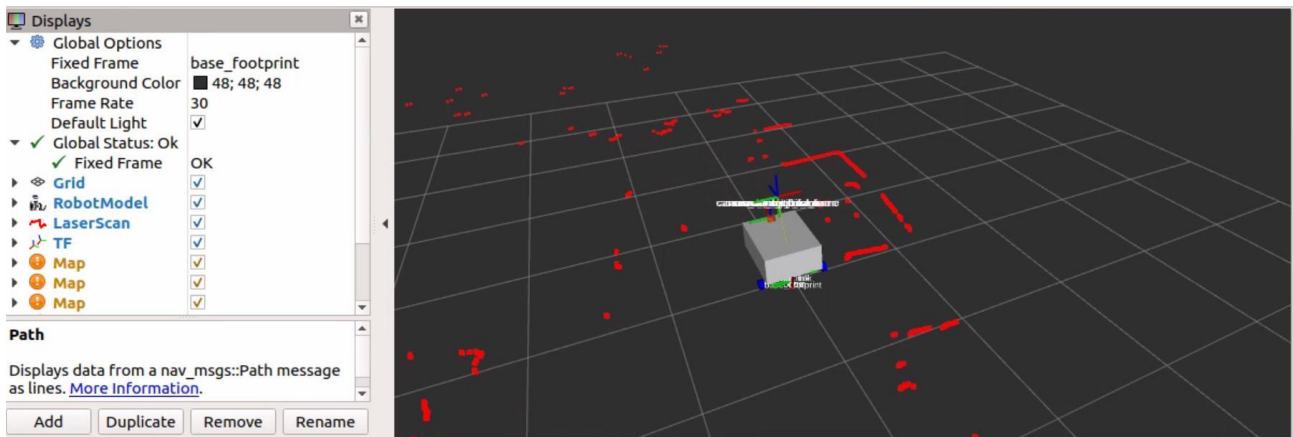


圖3-2-3 激光雷達數據圖

IMU：MPU6050

功能：測量車體的加速度與角速度，用於姿態估算和運動狀態追蹤。並且使用濾波器套件（filter）處理 `/imu/data_raw`，去除雜訊後生成穩定的運動數據。

數據發布與過濾：

原始數據：`/imu/data_raw`

過濾後數據：`/imu/data`

數據融合與應用：通過修改 EKF（Extended Kalman Filter）過濾器套件設置將過濾後的 IMU 數據 `/imu/data` 與 `/wheel_odom` 進行融合，生成精確的 `/Odom` 位姿數據，供導航讀取。

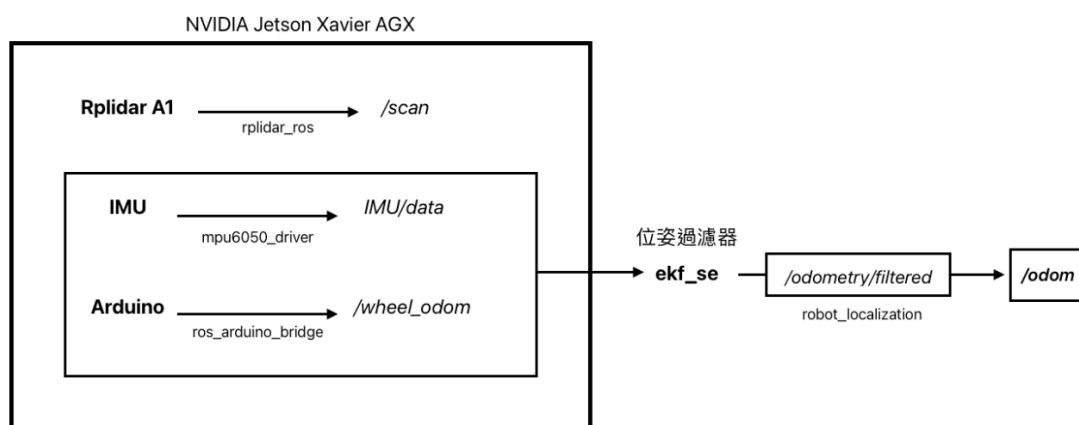


圖3-2-4 底盤架構圖

軟體架構

● 建圖模組 Cartographer

Cartographer 是由 Google 開發的 SLAM (Simultaneous Localization and Mapping) 系統，用於移動機器人在未知環境中同時進行自我定位與地圖建構。此系統能有效處理來自雷達與 IMU 的感測數據，即時更新環境地圖，並修正車體位姿，適合在室內場域使用。

運作原理：

透過接收雷達 (*/scan*) 與 IMU (*/imu/data_raw*) 數據，系統可估算機器人的即時姿態，並隨著移動過程不斷更新 2D 地圖。此外，Cartographer 採用回環偵測技術 (Loop Closure) 來減少累積誤差，確保長時間運作後的定位與地圖仍然準確。完成建圖後，生成的地圖會儲存為 .yaml 以及 .pgm 格式，建構好的地圖會以 */map* topic 供後續導航系統使用。本系統藉由 Cartographer 所建構的地圖，作為導航階段的基礎，讓車體能依據該地圖進行路徑規劃與即時避障。

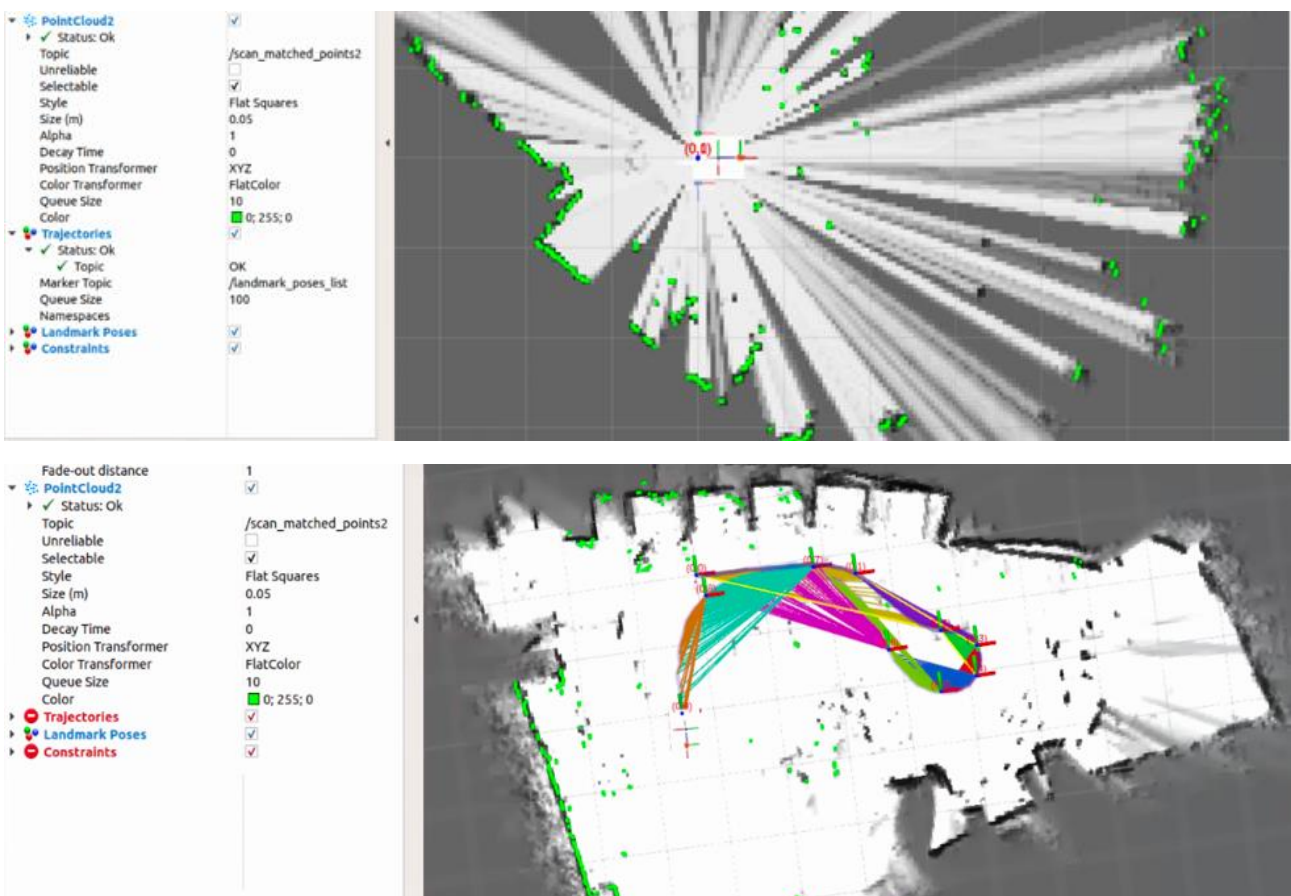


圖3-2-5 Cartographer

● 導航模組

(ROS Navigation)

使用 ROS 內Navigation套件中的 `move_base` 節點作為核心，透過配置yaml檔設定自己車身大小、感測器數據、運動控制、障礙物數據協調全局和局部路徑規劃，讓機器人能從當前位置移動至目標點。`move_base` 依賴四個主要配置檔案其節點主要是基於全局成本地圖計算從當前位置到目標點的最佳路徑再從在局部成本地圖上計算短程路徑，並進行即時避障，定位方面使用AMCL自我定位模組基於粒子濾波（Particle Filter）演算法根據雷達數據和里程計資訊，持續追蹤自己的位姿並發布(`/amcl_pose`)給`move_base`能在已知的地圖上找到自己的位置。

主要節點:

(Move_base)

用途: 協調全局路徑規劃和局部避障，將導航指令轉化為速度控制命令。

輸入：來自定位(`/amcl_pose`)、地圖(`/map`)、里程計(`/odom`)、雷達掃描(`/scan`)的即時數據。

輸出：速度指令 (`/cmd_vel`) 給車身底盤達到運動控制。

全局路徑規劃(global_planner)

用途: 基於全局成本地圖(global costmap)靜態地圖/`map`來表示環境中的已知障礙物，如牆壁、家具等透過修改演算法套件計算從當前位置到目標位置的最佳路徑。

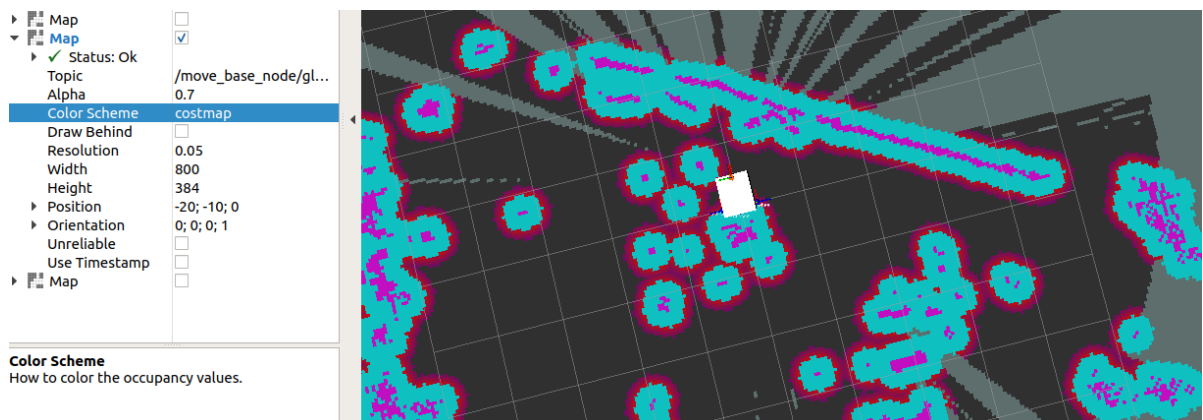


圖3-2-6 全局成本地圖

局部路徑規劃(local_planner)

用途: 根據局部成本地圖(local costmap)接收來自雷達和編碼器數據進行即時避障，以及即時更新局部成本地圖處理移動中的障礙物。透過修改Ros Navigation中的teb_local_planner插件中的參數車子每個周期的線速度與角速度且盡量符合全局路徑規劃，根據機器人的速度限制、雷達數據及局部地圖中的障礙物位置搜索躲避和行徑的多條路徑短時間內給予最佳運動方向和速度並動態規劃短程路徑。

Teb_local_planner

用途: 根據激光雷達、編碼器、IMU 等傳感器的數據，持續更新周圍環境資訊與機器人的位置。它通過優化軌跡的時間與空間參數，動態調整局部路徑。規劃過程中考慮了機器人的運動學約束（如轉向角速度、加速度）和動力學限制，並在路徑生成時兼顧障礙物的距離和未來的速度變化，確保路徑平滑且可行。

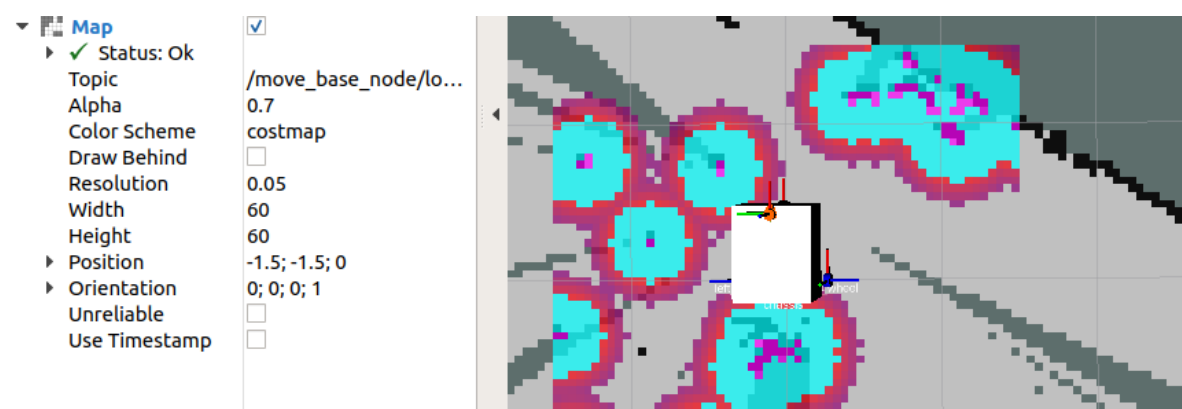


圖3-2-7 局部成本地圖

(Costmap_2d)

用途: 主要用來設置地圖解析度描述障礙物的分佈，平衡地圖的精度與運算效率以及的膨脹層，將障礙物周圍的區域設為高成本區，以創建緩衝區確保機器人與障礙物保持安全距離，避免機器人接近障礙物

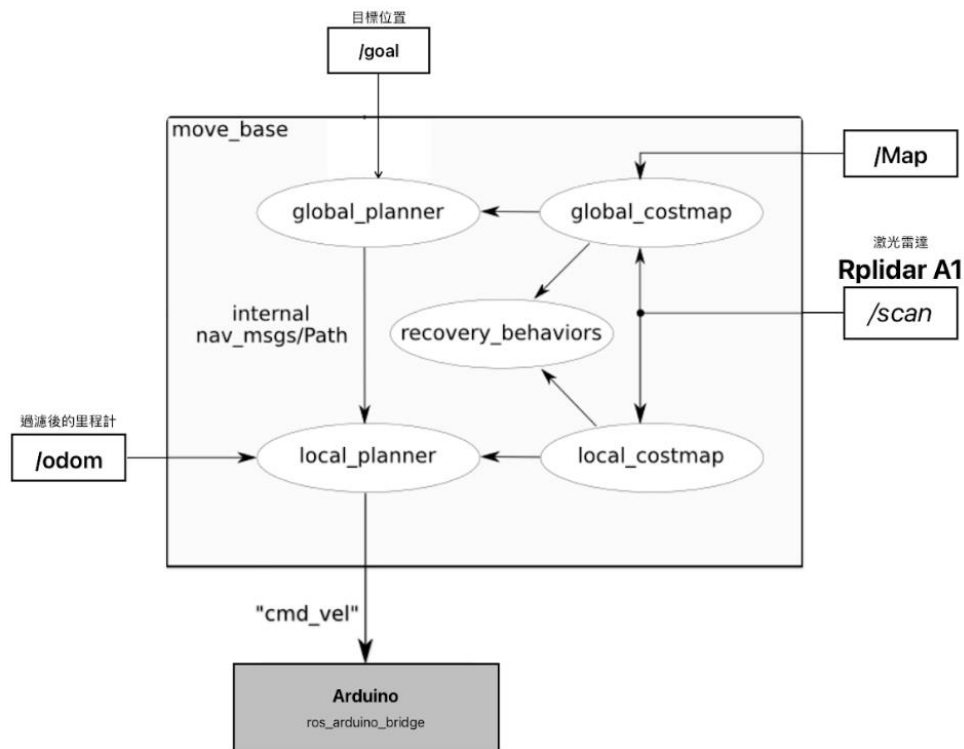


圖3-2-8 導航架構圖

第三節 整合系統

點餐與送餐的流程整合：

1. 使用者通過 LINE 或網頁點餐，ChatGPT 會解析訂單並將資料推送至後端資料庫。
2. 餐廳確認訂單後，將餐點準備完畢，並由送餐車負責配送。送餐車依照系統生成的路徑導航至顧客指定的地點。
3. 在送餐過程中，送餐車透過感測器進行即時的障礙物檢測和避障，確保餐點安全送達。當餐點到達後，系統會通知顧客取餐。

整合的優勢：

- 提高點餐與送餐的自動化程度，減少人力需求，並通過自動導航和避障技術確保送餐過程的安全性和效率。
- 結合 ChatGPT 與 LineBot 讓顧客能夠以自然語言與系統互動，提升點餐體驗。同時，所有訂單均可即時同步到後台，方便餐廳管理。

第四章 系統實作結果

第一節 點餐系統

● 交互式訂餐系統

- 建立 LINE 官方系統提供服務，串接 ChatGPT 3.5-turbo 讀取由 Excel 轉存的檔案。讓 ChatGPT 3.5-turbo 扮演線上咖啡廳點餐系統的角色讀取檔案內容提供交互式問答點餐方式。
- 使用者可以用問答方式點餐，ChatGPT 3.5-turbo 會從使用者輸入的文字中抓取使用者輸入的餐點關鍵字、數量並將餐點內容加入購物車，使用者可以查看購物車 並且可以新增、刪除品項在統計完項目後請使用者確認訂單，並且成功付款，最後將確認好的訂單及時推送到 Google Excel 中記錄時間、桌號、姓名、電話、付款方式、餐點內容、備註、價錢。

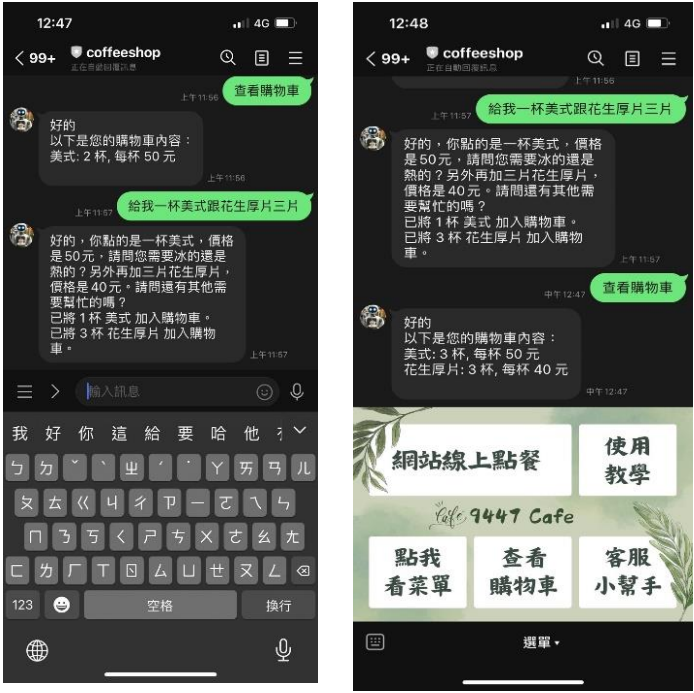


圖4-1-1 交互式點餐

	A	B	C	D
1	種類	品項	價格	標籤
2	咖啡	義式濃縮	50	#咖啡#飲品
3	咖啡	美式咖啡	50	#咖啡#熱門#飲品
4	咖啡	卡布奇諾	65	#咖啡#飲品
5	咖啡	拿鐵	70	#咖啡#熱門#飲品

圖4-1-2 菜單詳細資料

● 線上點餐系統

- 使用 HTML 和 CSS 設計網頁樣式，按下「開始點餐」後可選擇內用或外帶，並在確認桌號後，自由選擇想要的品項。對於特定品項，還能進一步選擇冰或熱的選項。
- 對於購物車的內容我們的系統已連結 Firebase 雲端資料庫，當使用者按下「確認訂單」時，購物車內容會被推送到雲端的 Excel，並完整記錄每個訂單的品項資料。

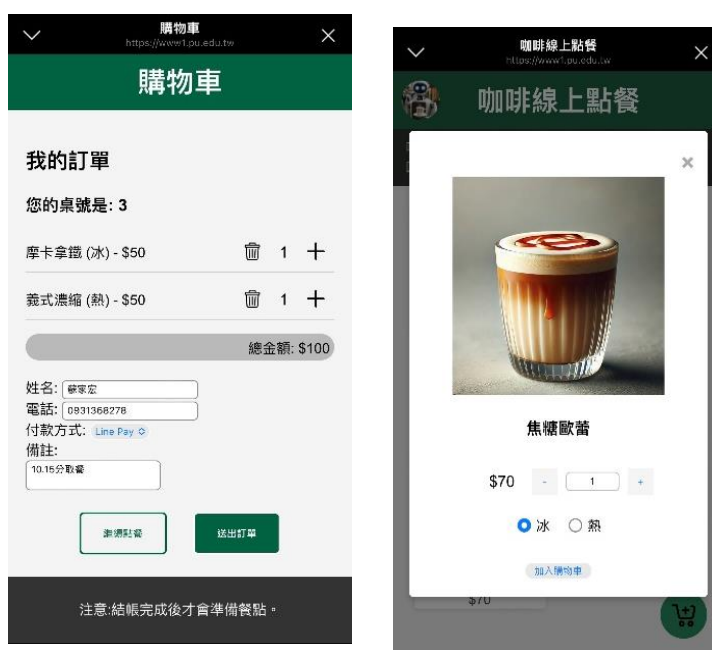


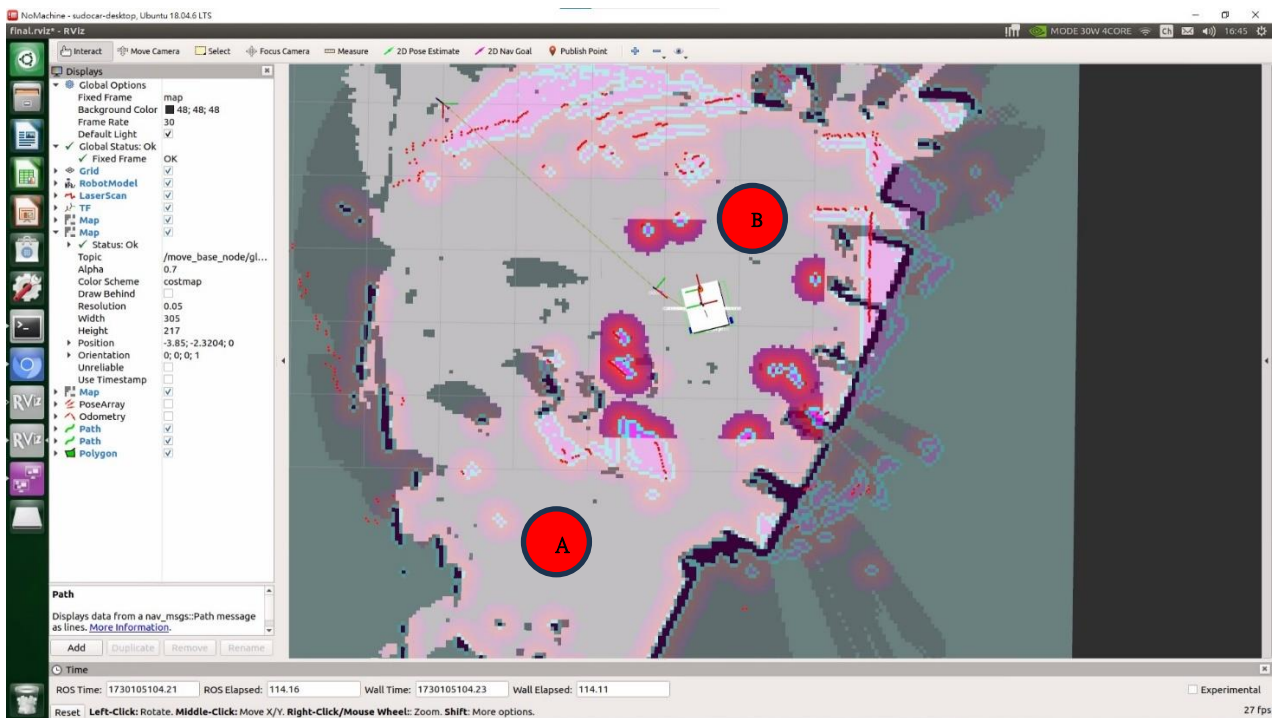
圖4-1-3 線上點餐系統

orders_coffe ☆ 雲端							
檔案 編輯 查看 插入 格式 資料 工具 擴充功能 說明							
A1 下單時間							
	A	B	C	D	E	F	G
1	下單時間	桌號	姓名	電話	備註	付款方式	餐點內容
2	2024/9/27 下午 11:28:02	外帶	龍鼠	85889999		Line Pay	摩卡拿鐵 (冰) > \$50 x 1
3	2024/9/29 下午 10:39:33	1	宏	110	你好 你好 你好你好你好	Line Pay	摩卡拿鐵 (冰) > \$50 x 3
4	2024/9/29 下午 11:27:30	外帶	林	123		Line Pay	焦糖布丁 > \$50 x 1
5	2024/9/30 下午 3:19:35	外帶	小新	333		Line Pay	美式 (冰) > \$50 x 1
6	2024/10/1 下午 10:54:42	2	濃濃	223		Line Pay	焦糖布丁 > \$50 x 1
7	2024/10/7 上午 1:01:28	3	阿寶	147		Line Pay	拿鐵 (冰) > \$50 x 1
8							花生厚片 > \$40 x 1
9							草莓氣泡飲 (少冰) > \$75 x 1
10							

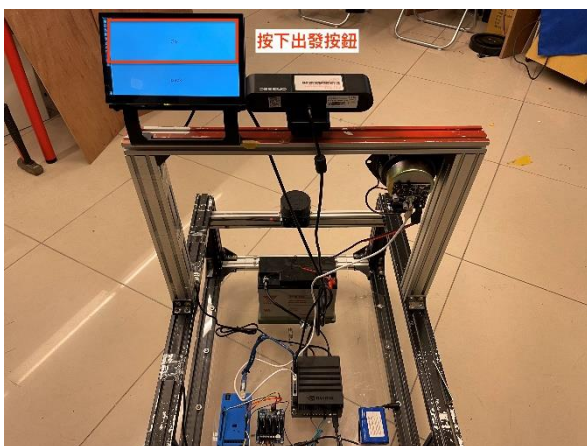
圖4-1-4 訂單更新至雲端Excel

第二節 送餐系統

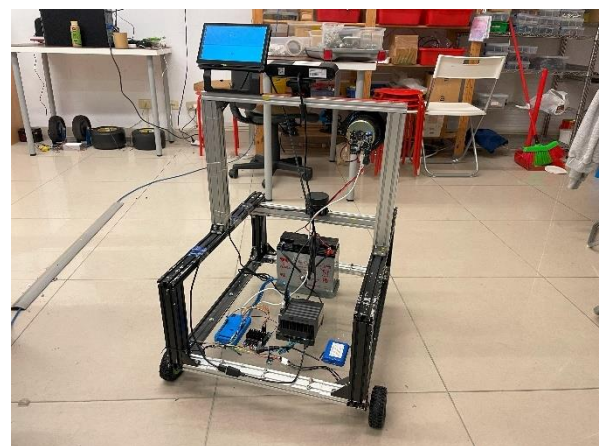
當店員將餐點放置到送餐機器人的置物籃中，並按下螢幕上的「出發」按鈕後，機器人會啟動並利用避障算法進行初步的路徑規劃。在計算完畢後，機器人開始行駛。行駛過程中，機器人透過整合雷達、編碼器和三軸感測器的數據，結合全局規劃與局部規劃的優勢，動態調整路徑。這些調整讓機器人能夠有效避開行進途中的靜態和動態障礙物，並根據預定的座標精確導航，最終將餐點送達指定目的地。



目標:選擇目的地B(由A走到B)



第一步:按下出發按鈕



第二步:經過無數障礙物抵達終點

第三節 技術特色及創新點

我們開發的送餐系統具有技術特色及創新點，與市面上的送餐機器人不同，提供了從數位化點餐到將餐點直接送到顧客手上的完整服務。透過結合ChatGPT 3.5-turbo，點餐系統能模擬人類的自然交談方式，讀取菜單項目並精確抓取消費者輸入的關鍵字與數量。這種技術大幅提升了點餐效率，特別是讓顧客無需花費時間瀏覽菜單，直接透過對話選擇商品，簡化了整個點餐流程。

在國際化的使用環境中，該技術能有效解決語言障礙，提升非母語顧客的使用體驗。透過多語系支持，外籍顧客也能方便地使用系統進行點餐，享受到流暢、自然的服務體驗，無論來自何地，都能感受到人性化的服務。

我們的系統還具備高度的擴展性與靈活性，可以根據不同餐飲業者的需求進行調整，無論是擴展更多餐點選項，還是支持多語言點餐，都能輕鬆適應，滿足各類顧客的需求。未來，我們將引入更直觀的技術來優化點餐系統，使其能夠更加簡單易用。

此外，為了確保顧客的服務品質，我們計劃對員工進行專業培訓，讓他們熟悉系統的操作流程，並能在服務過程中提供及時幫助。這不僅能提升服務效率，也能讓顧客感受到更高水準的服務品質。通過技術創新與服務提升，我們的系統將成為未來智能送餐市場中的一大亮點。

第五章 進行方法及步驟

第一節 採用方法與原因

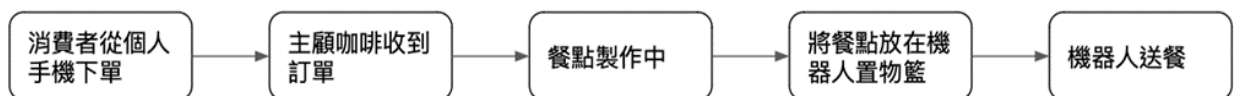
不同於市面上的送餐車，我們選擇使用自動化和人工智能技術，包括 ChatGPT 結合 LINE 聊天機器人和利用 ROS 雷達避障功能達成自動導航的自走車。這些技術不只可以提高我們的服務效率並減少人工錯誤且服務具創新性，更吸引新用戶和保持現有用戶的忠誠度。透過這些技術，我們還可以提供不間斷的服務，這對提高顧客滿意度和業績有很大的幫助。

第二節 詳細步驟

點餐機器人流程

1. 當User在官方LINE發送訂購品項時，經過ChatGPT統計品項及價格。
2. 確認訂單後產生一個付款碼。
3. 將訂單傳入主顧咖啡後台。
4. 餐點準備完成後，讓運輸車透過自動導航系統以及雷達避障功能，前往指定位置。
5. 餐點送達。

實際情境：



後台情境：

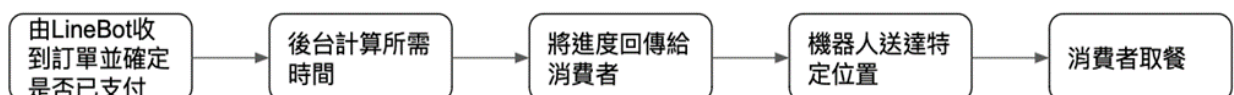


圖5-2-1 前台與後台實際情境圖

第六章 結論與未來展望

此專題開發之後，希望能運用我們高精準度的雷達避障功能及自主導航功能，還有專為餐廳設定的聊天機器人來建置整個送餐系統，並且想辦法降低成本和大量生產，希望在未來能夠將該產品發布上市，提升市場的競爭性。

第七章 預期完成之工作項目及具體成果

第一節 工作分配















































▼ 底層平台	 秉翔 胡  Nickk	March 26, 2024 → April 1, 2024	● Done
▶ 直流馬達	 秉翔 胡  Nickk	April 1, 2024	● Done
搭建紅外線攝影機環境	 秉翔 胡  Nickk	March 27, 2024 → March 29, 2024	● Done
搭建激光雷達環境	 秉翔 胡  Nickk	March 26, 2024	● Done
▼ 點餐系統APP	 家宏 蘇  林誼均 110	April 10, 2024 → May 20, 2024	● Done
(ChatGPT)建立 .json 資料檔案	 家宏 蘇  林誼均 110	April 10, 2024	● Done
(ChatGPT)LineBot導入ChatGPT	 家宏 蘇  林誼均 110	April 10, 2024	● Done
(ChatGPT)Line bot 建置	 家宏 蘇  林誼均 110	April 10, 2024	● Done
(ChatGPT)使用Render串接OpenAI跟L	 家宏 蘇  林誼均 110	April 10, 2024	● Done
(Fine-tune)編寫fine-tuning程式	 家宏 蘇  林誼均 110	April 11, 2024	● Done
編寫LINE APP程式	 家宏 蘇  林誼均 110	April 11, 2024	● Done
LINE BOT 使用 fine-tune ChatGPT測試	 家宏 蘇  林誼均 110	April 11, 2024	● Done
▶ 增添JSON資料數據集  OPEN	 家宏 蘇  林誼均 110	April 24, 2024 	● Done
改良後fine-tune 測試	 家宏 蘇  林誼均 110	April 25, 2024	● Done
建立主顧咖啡execl菜單並轉存為.csv檔	 家宏 蘇  林誼均 110	April 25, 2024	● Done
修改程式使gpt-3.5讀取.csv資料	 家宏 蘇  林誼均 110	April 26, 2024	● Done
對話測試	 家宏 蘇  林誼均 110	April 26, 2024	● Done
▼ slam建圖與自主導航	 秉翔 胡  Nickk	April 10, 2024 → April 30, 2024	● Done
gmapping環境設定	 秉翔 胡  Nickk	April 10, 2024	● Done
雷達與gmapping數據串連建圖	 秉翔 胡  Nickk	April 10, 2024	● Done
鍵盤數據與馬達驅動串連	 秉翔 胡  Nickk	April 10, 2024	● Done
雷達數據與馬達驅動串連	 秉翔 胡  Nickk	April 12, 2024	● In progress

圖7-1-1 工作分配

第二節 專題時程



圖7-2-1 甘特圖

第三節 討論紀錄表

會議記錄表			
日期時間	2024/9/9	會議次數	第 1 次
會議主席	胡秉翔	會議地點	創客吧
參與人員	何允衡 蘇家宏 林誼均	缺席人員	無
討論主題	進度報告 未來時程規劃		
下次討論事項	檢查分工項目進度		
工作項目	主持:何允衡 記錄:林誼均		

會議記錄表			
日期時間	2024/9/27	會議次數	第 2 次
會議主席	胡秉翔	會議地點	創客吧
參與人員	何允衡 蘇家宏 林誼均	缺席人員	無
討論主題	進度檢核 購物車問題討論		
下次討論事項	須購買材料清單		
工作項目	主持:胡秉翔 記錄:蘇家宏		

會議記錄表			
日期時間	2024/10/7	會議次數	第 3 次
會議主席	胡秉翔	會議地點	創客吧
參與人員	何允衡 蘇家宏 林誼均	缺席人員	無
討論主題	車身架構討論 購買材料		
下次討論事項	報告書內容		
工作項目 (Action Item)	主持:胡秉翔 記錄:何允衡		

會議記錄表			
日期時間	2024/10/20	會議次數	第 4 次
會議主席	胡秉翔	會議地點	創客吧
參與人員	何允衡 蘇家宏 林誼均	缺席人員	無
討論主題	檢視專題總體完成度 點餐、送餐功能合併		
下次討論事項	初審報告		
工作項目 (Action Item)	主持:胡秉翔 記錄:林誼均		

第八章、附錄

第一節 參考資料

Firestore連結Excel

https://www.youtube.com/watch?v=uYgbw0rW-p4&t=248s&ab_channel=AnishInTech

Render部屬

https://www.youtube.com/watch?v=-bB65g1qqTg&t=257s&ab_channel=Maso%E7%9A%84%E8%90%AC%E4%BA%8B%E5%B1%8B

Python寫入Google sheets

https://www.youtube.com/watch?v=tPfilMdhCUE&ab_channel=%E9%BE%8D%E9%BE%8DAI%E8%88%87%E7%A8%8B%E5%BC%8F%E5%AF%A6%E6%88%B0

小車連結

<https://blog.csdn.net/Drakie/article/details/126292241>

ROS環境設置

<https://blog.csdn.net/hxj0323/article/details/121215992>

深度相機

https://blog.csdn.net/qq_57061492/article/details/127584919

https://blog.csdn.net/qq_57061492/article/details/127584919

激光雷達

<https://blog.csdn.net/BAIFOL/article/details/122633384>

Gmapping

https://blog.csdn.net/m0_63647490/article/details/123130882