朝陽科技大學
資訊工程系

【專題成果報告】

自動送餐系統

指導教授：洪士程　　博士
專題組員：張譽弘　(10627084)
　　邱名榕　(10627439)
　　王昱祺　(10627438)

中華民國 109 年 12 月

自動送餐系統

洪士程　張譽弘　邱名榕　王昱祺
朝陽科技大學資訊工程系
schong@cyut.edu.tw

摘要

我們這一組的專題是製作了一套自動送餐系統，可以讓消費者減少會被 COVID-19 傳染的風險的同時，可以讓店家節約人力成本的。我們利用 LINE Bot designer 設計了一個 Line 機器人以及簡潔的圖形化 UI 界面來作為我們的前端的點餐系統。

我們後端使用近期比較熱門的 server：heroku 作為我們的雲端伺服器，只需要將我們的程式碼部署到 heroku 上便可讓他 24h 不間斷運作，適合任何店家作使用。

然後使用 heroku 的附屬資料庫 ClearDB 作為資料庫，來存取消費者的點餐紀錄及細項，ClearDB 是使用 mysql 這個比較大眾的資料庫語法，工程師們可以輕鬆上手及編譯。最後，用 c++來讀取在 ClearDB 的消費者資料及餐點，再根據消費者所購買的品項以及數量，控制 KINOVA 機械手臂送到客戶的面前。

關鍵字：*Line bot*、*heroku*、*MySQL*、*C++*、*KINOVA*、*ClearDB*、*LINE*

Abstract

The topic of our group is to create an automatic food delivery system that can save stores labor costs and reduce the risk of consumers being infected by COVID-19. We use Line bot designer to design a Line robot and a simple graphical UI interface as our front-end ordering system. Line is the most familiar communication software for the public. Using line as a food ordering system can be easy for the general public to use. Both the elderly and children can easily control this ordering system.

Our backend uses the recently popular server: heroku as our cloud server, we only need to deploy our code to heroku to allow it to operate 24 hours a day, suitable for any store. Then use heroku's subsidiary database ClearDB as a database to access consumers' order records and details. ClearDB uses mysql, a more popular database syntax, which engineers can easily use and compile. Finally, use c++ to read the consumer information and meals in ClearDB, and then according to the items and quantities purchased by the consumers, the KINOVA robotic arm is sent to the customer.

目　　錄

圖 目 錄

第一章 簡介

COVID-19 肆虐全球，在沒有疫苗及有效防護的狀況下，由於 COVID-19 可以透過飛沫傳染，而用餐時沒有口罩的保護，會大幅增加被傳染的風險，所以大家都不太想要出門用餐和外出購物，導致全球服務業的營業額大幅度的下降。要怎麼樣才能讓大家出去用餐或是外帶食物的時候比較安心呢？我們想到最近最為熱門的無人化服務，在這幾年間，無人化服務相當的熱門，大到無人商店、自動化工廠，小到自動搖飲料機，甚至自動販賣機以及自動繳費機都算是無人化服務的一種。那如果店家可以無人化服務來達到來送餐和點餐，這樣可以避免很多的服務生和顧客之間的接觸，同時也可以為店家節省人力資源，節約成本，從而達到雙贏的局面。也因此，我們想要開發一套無人化的點餐取餐服務。

當我們在提出想要製作無人化的點餐取餐服務，洪士程教授有提及到日本有開發出自動送餐的小火車應用在旋轉壽司店，顧客先通過點餐系統，進行壽司的加點，然後師傅會將做好的壽司放到小火車，再由小火車直接送到點餐客人的座位前面，讓客人自己拿下來。我們受到這個作品的啓發，想要製作一套完整點餐-機器人拿餐點-顧客取餐及結賬的服務。而這樣一套無人化點餐系統最理想的情況會是 24 小時可以運作，點餐界面可以讓大眾輕鬆上手，無需閱讀一本厚厚的使用說明書，以及有良好的升級及擴充性以應對不同的情景及所需要的服務。因此我們選擇了 line 作為前端，heroku 作為雲端平台來架設伺服器，使用 mysql 的 ClearDB 作為資料庫，最後利用 KINOVA 機械手臂送到客戶的面前。

第二章 使用元件介紹

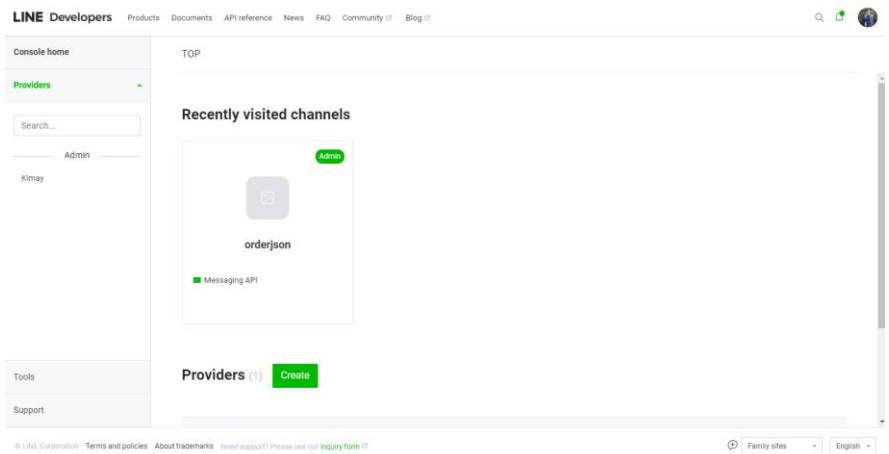
2.1 LINE Bot 介紹

聊天機器人最大的特點就是能給予用戶最即時的應對，LINE Bot 對話通常與常人無異，除了幾項特別的信息格式外，幾乎都是與人相仿的行為能力，但是 LINE Bot 能夠間接透過數據分析來達成近乎瞬間的回傳結果，同時也能利用串接外部的服務來滿足用戶的功能需求。

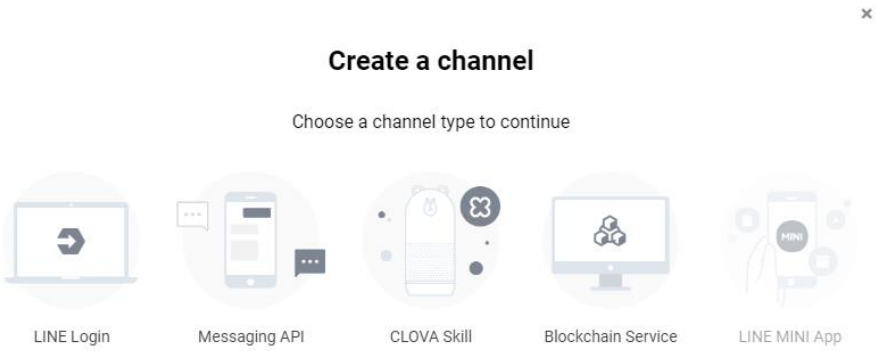
在台灣有 80%的手機用戶是使用 LINE 通訊軟體，所以 LINE Bot 很適合應用在服務業，為了讓企業可以簡單快速使用他們的平台製作聊天機器人，LINE 提供了許多開發人員工具，方便開發 LINE API ，其中我們使用以下幾項工具:

2.1.1 LINE Developers：

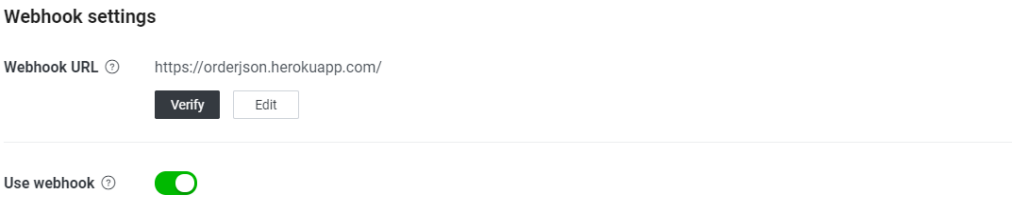
任何 LINE 用戶能夠依照需求快速的創建 Channel，設置 Channel 的 Message API、Webhook 和管理員權限，還能統計該 Channel 的數據流量。我們的目的是設定 Webhook，讓 LINE Bot 與 Heroku 溝通，並且使用 LINE Developers 產生出 QR Code，便於客戶將 LINE Bot 加為好友，進行消費等活動。



▲圖 1. LINE Developers 介面



▲圖 2. Channel 的種類

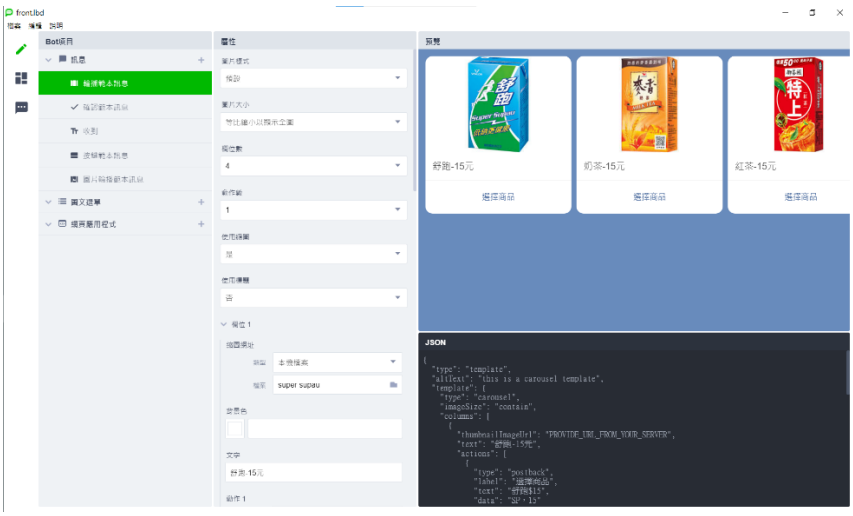


▲圖 3. 設置 Webhook

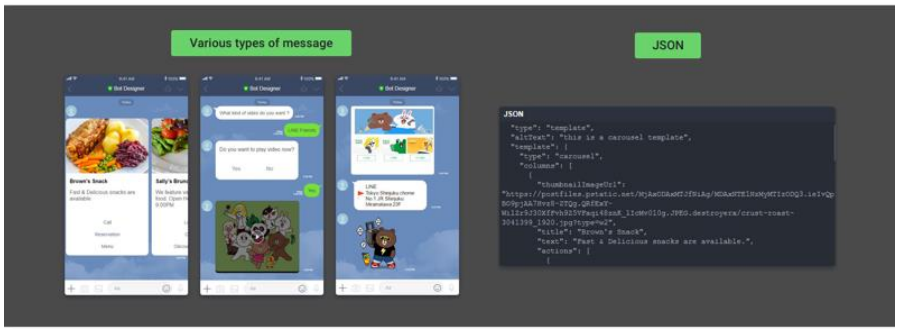
2.1.2 LINE Bot Designer：

無需任何程式相關知識，即可更快速輕鬆地設計 LINE 聊天機器人原型，依據所需的場景，自由設計聊天機器人。透過 LINE Bot Designer 產生各種類型的 LINE 訊息，並實現任何想要的聊天機器人場景。透過 LINE Bot Designer 建立訊息，所產生的 JSON 程式碼，未來可用於真實的聊天機器人開發。

我們在 LINE Bot Designer 上設計 Message API，依照設計好的 Message Box 的圖片樣式、文字格式和按鍵功能轉換成 JSON 程式碼的格式，並應用在 LINE Bot 的後端程式碼上。



▲圖 4. LINE Bot Designer 介面



▲圖 5. Message 轉換成 JSON

2.1.3 Messaging API SDK：

Messaging API 的軟體開發工具包，裡面包含工具和示例，可以更輕鬆地開始使用 Messaging API 開發聊天機器人。官方的 SDK 和社區的 SDK 都是開源的，並且有提供多種不同的程式語言。

將 Node.js 導入 Messaging API SDK，利用 SDK 中的函式設置圖文選單、多頁訊息、問卷調查...等，還能依指定文字自動回傳訊息。

2.2 KINOVA 協作型機械手臂

KINOVA Gan2 是由 KINOVA 公司所設計製造的輕型機械手臂，在台灣是由掌宇股份有限公司來進行進口代理。

DOF 有幾項特色，低功耗、超輕量、容易攜帶，且他的手臂程式相容 ROS 和 SDK，它的執行器中有五個感知器，有位置、溫度、力矩、電流以及加速度計，在手臂末端具備，三隻手指頭，可以輕鬆地取物品同時也可以兼顧到安全性。

控制機械手臂移動的模式分為三種：Angular velocity、Cartesian velocity 和 Cartesian position。Angular velocity 是調整速度和時間來控制每個制動器旋轉的角度，原理是將設定好的速度每隔幾秒發送一次訊息給機械手臂。而 Cartesian velocity 是調整速度和時間來控制機械手臂在座標軸上的移動。Cartesian position 是直接指定座標，發送訊息給機械手臂移動到指定座標。Cartesian 模式使用的座標系有兩個，分別為笛卡兒座標和球型座標，坐標軸分別為 X、Y、Z、ThetaX、ThetaY 和 ThetaZ。

```
            pointToSend.Position.Type = ANGULAR_VELOCITY;

            pointToSend.Position.Actuators.Actuator1 = 0;
            pointToSend.Position.Actuators.Actuator2 = 0;
            pointToSend.Position.Actuators.Actuator3 = 0;
            pointToSend.Position.Actuators.Actuator4 = 0;
            pointToSend.Position.Actuators.Actuator5 = 0;
            pointToSend.Position.Actuators.Actuator6 = 48; //joint 6 at 48 degrees per second.

            pointToSend.Position.Fingers.Finger1 = 0;
            pointToSend.Position.Fingers.Finger2 = 0;
            pointToSend.Position.Fingers.Finger3 = 0;

            for (int i = 0; i < 300; i++)
            {
                //We send the velocity vector every 5 ms as long as we want the robot to move along that vector.
                MySendBasicTrajectory(pointToSend);
#ifdef __linux__
                usleep(5000);
#elif _WIN32
                Sleep(5);
#endif
            }
```

▲圖 6. Angular velocity example

```
            pointToSend.Position.Type = CARTESIAN_VELOCITY;

            pointToSend.Position.CartesianPosition.X = 0;
            pointToSend.Position.CartesianPosition.Y = -0.15; //Move along Y axis at 20 cm per second
            pointToSend.Position.CartesianPosition.Z = 0;
            pointToSend.Position.CartesianPosition.ThetaX = 0;
            pointToSend.Position.CartesianPosition.ThetaY = 0;
            pointToSend.Position.CartesianPosition.ThetaZ = 0;

            pointToSend.Position.Fingers.Finger1 = 0;
            pointToSend.Position.Fingers.Finger2 = 0;
            pointToSend.Position.Fingers.Finger3 = 0;

            for (int i = 0; i < 200; i++)
            {
                //We send the velocity vector every 5 ms as long as we want the robot to move along that vector.
                MySendBasicTrajectory(pointToSend);
#ifdef __linux__
                usleep(5000);
#elif _WIN32
                Sleep(5);
#endif
            }
```

▲圖 7. Cartesian velocity example

```
            //We specify that this point will be an angular(joint by joint) position.
            pointToSend.Position.Type = CARTESIAN_POSITION;

            //We get the actual angular command of the robot.
            MyGetCartesianCommand(currentCommand);

            pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X;
            pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.1f;
            pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z;
            pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX;
            pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY;
            pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ;

            cout << "*********************************" << endl;
            cout << "Sending the first point to the robot." << endl;
            MySendBasicTrajectory(pointToSend);

            pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z + 0.1f;
            cout << "Sending the second point to the robot." << endl;
            MySendBasicTrajectory(pointToSend);

            cout << "*********************************" << endl << endl << endl;
        }
```

▲圖 8. Cartesian position example

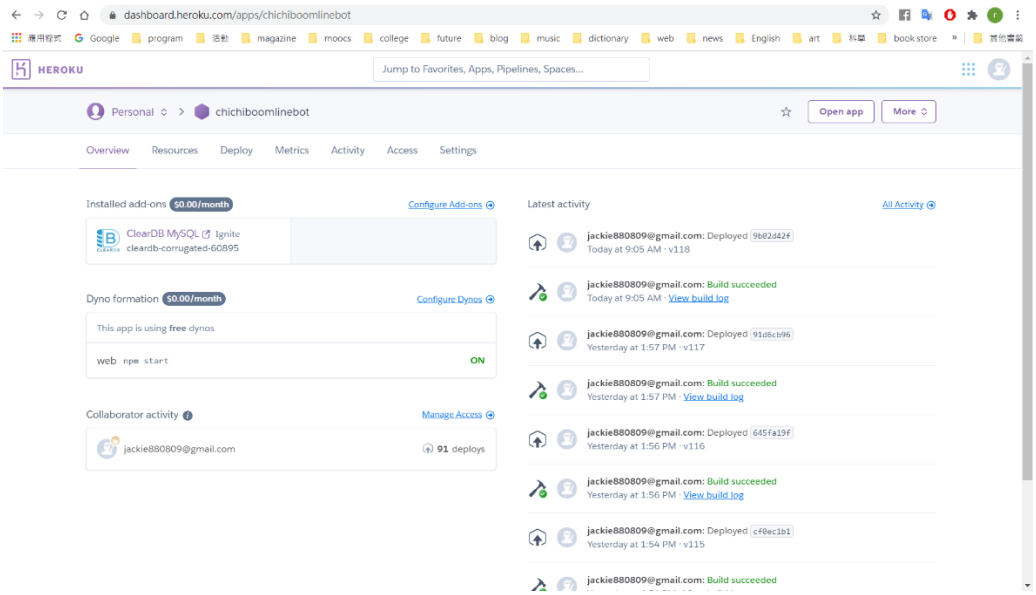
2.3 Heroku 介紹

Heroku 為一個平台即時服務（Platform as a Service ,PaaS），同時也是雲平台的始祖，平台即時服務是一種雲端運算服務，可以提供運算的平台，並且連接軟體即時服務與基礎設施服務。

起初 Heroku 只能支援 Ruby 但在後來慢慢的新增了 Java、Node.js、Scala、Clojure、Python，在 2011 年中後旬與 Facebook 合作，新增支援 PostgreSQL、MongoDB 等資料庫程式。

我們下載 Heroku CLI 來管理 pipeline，將 LINE Developers 取得的

Channel access token 和 Channel secret 設置 Config Vars，再利用 git 將 Node.js 程式碼部屬到 Heroku 上，這樣只要用戶從 LINE 端傳來訊息，Heroku 會呼叫函式執行設定好的動作(回傳訊息、寫入資料庫...等)。



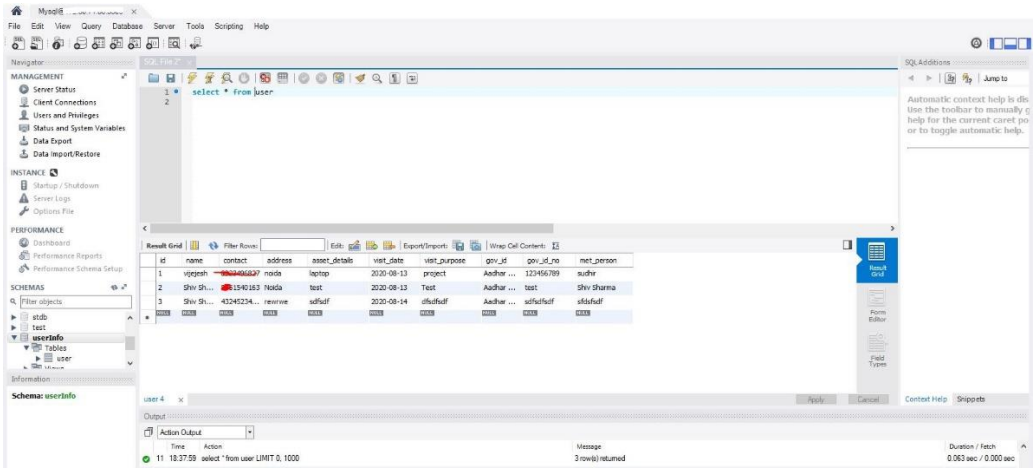
▲圖 9. Heroku 介面

2.4 MySQL 介紹

MySQL 是一種開放原始碼的關係型資料庫管理系統，是由瑞典的 MySQLAB 公司所開發，因為效能高、成本低且可靠性很好，所以成為許多中小型網站架設成用的資料庫。

MySQL 支援 Windoow、Linux、MacOS...等作業系統，而且還為許多種程式語言提供 API，如 C、C++、C#、PHP、Python...。

我們利用 MySQL 儲存客戶餐點的資訊，利用 MySQL Workbench 來建立 table。



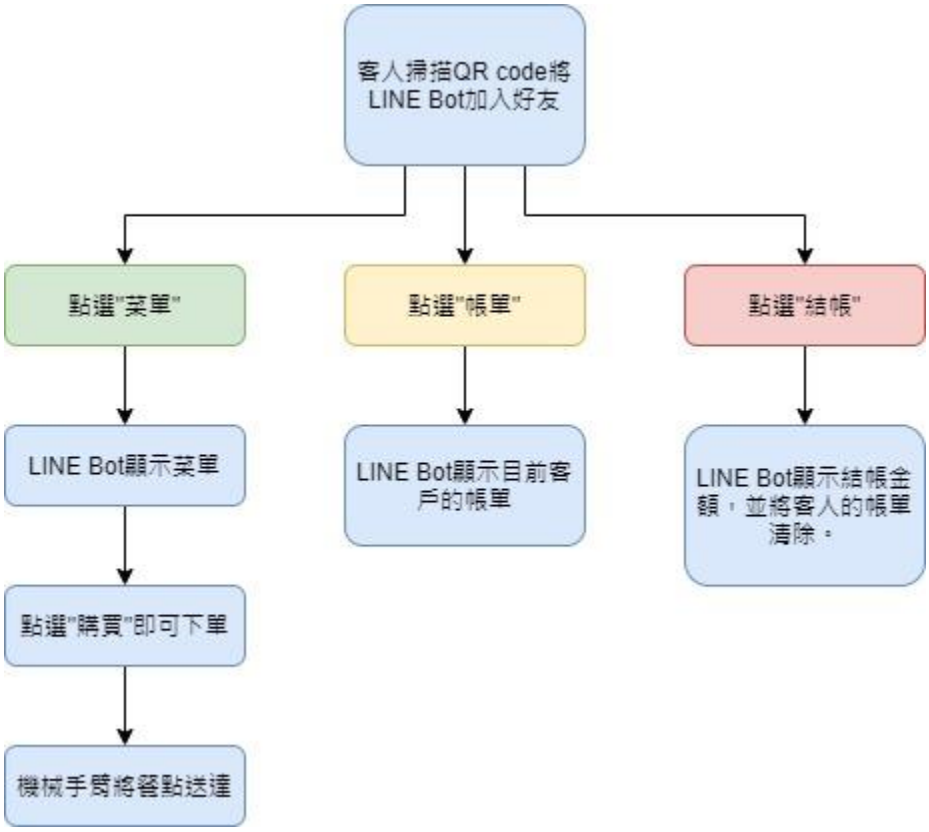
▲圖 10. MySQL Workbench 介面

2.5 Python 介紹

Python 是一種容易學習、功能強大且正在被廣泛使用的一種高階程式語言，具有 4 大特色分別是容易撰寫、功能性強大、具有跨平台的功能以及容易擴充的特性。

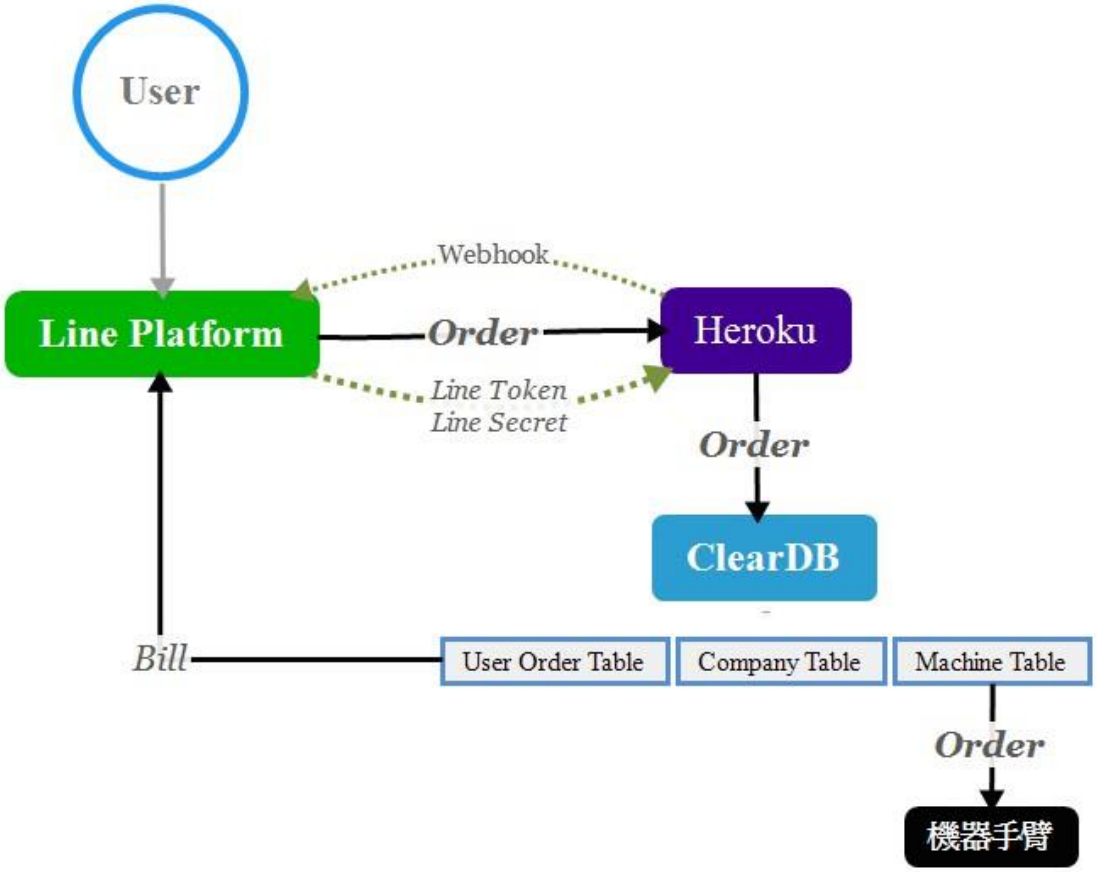
會選擇使用 Python 的原因是，它的寫法簡單且連接資料庫的部分有很多的模組可以做選擇，最後我們是使用 pymysql 這個模組來讓 python 可以操作 MySQL，如此一來就可以將 LINE Bot 裡用戶端傳過來的訊息整理後並寫入 MySQL 裡。

第三章 動作說明



▲圖 11. 使用者操作流程圖

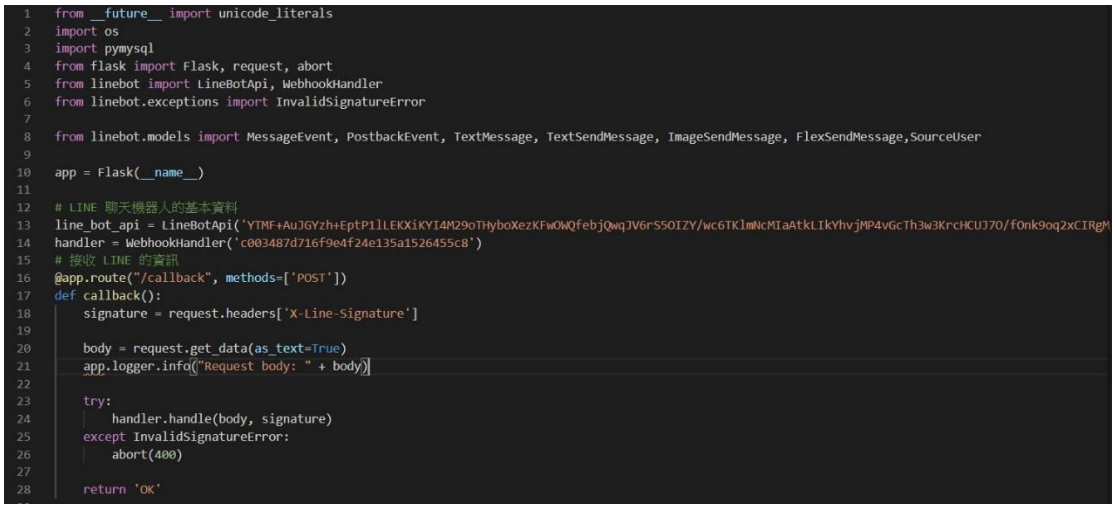
▲圖 12. 使用者介面



▲圖 13. 系統流程圖

第四章 程式設計

我們的程式設計分為兩個部分，分別控制機械手臂以及 LINE Bot。機械手臂的動作是先利用 Visual Studio 編譯 C++程式碼，再用筆電執行。而 LINE Bot 則是先使用 VS Code 編譯 python 程式碼，再用 git 部屬到 Heroku 運行。以下是 LINE Bot 的程式碼:



```python
from __future__ import unicode_literals
import os
import pymysql
from flask import Flask, request, abort
from linebot import LineBotApi, WebhookHandler
from linebot.exceptions import InvalidSignatureError

from linebot.models import MessageEvent, PostbackEvent, TextMessage, TextSendMessage, ImageSendMessage, FlexSendMessage,SourceUser

app = Flask(__name__)

# LINE 聊天機器人的基本資料
line_bot_api = LineBotApi('YTMF+AuJGYzh+EptP1lLEKXiKYI4M29oTHyboXezKFwOWQfebjQwqJV6rS5OIZY/wc6TKlmNcMIaAtkLIkYhvjMP4vGcTh3w3KrcHCUJ7O/fOnk9oq2xCIRgM
handler = WebhookHandler('c003487d716f9e4f24e135a1526455c8')
# 接收 LINE 的資訊
@app.route("/callback", methods=['POST'])
def callback():
    signature = request.headers['X-Line-Signature']

    body = request.get_data(as_text=True)
    app.logger.info("Request body: " + body)

    try:
        handler.handle(body, signature)
    except InvalidSignatureError:
        abort(400)

    return 'OK'
```

▲圖 14. LINE_Bot_API 與 handler

LINE_Bot_API 與 handler 要填上 LINE Bot 的資訊才能與聊天機器人連接。

```python
@handler.add(MessageEvent, message=TextMessage)
def echo(event):

    if event.source.user_id != "Udeadbeefdeadbeefdeadbeefdeadbeef":

        if event.message.text=='舒跑 10元 一瓶':
            if isinstance(event.source, SourceUser):
                try:
                    profile = line_bot_api.get_profile(event.source.user_id)
                    record_list=[(profile.display_name,'舒跑','10元','一瓶')]
                    line_insert_record(record_list)
                    text_list=[(profile.display_name,'舒跑',10,'一瓶')]
                    line_insert_list(text_list)

                    line_bot_api.reply_message(
                        event.reply_token,
                        TextSendMessage(text='收到')
                    )
                except:
                    line_bot_api.reply_message(
                        event.reply_token,
                        TextSendMessage(text='失敗了')
                    )
```

▲圖 15. LINE Bot SDK 的函式

```
elif event.message.text=='帳單':
    if isinstance(event.source, SourceUser):
        try:
            profile = line_bot_api.get_profile(event.source.user_id)
            profile_name=profile.display_name
            datatext=line_select_record(profile_name)
            ss=list(datatext)
            for x in range(len(ss)):
                if ss[x]==',' and ss[x+1]==')':
                    ss[x]=''
            for x in range(len(ss)):
                if ss[x]==',' and ss[x-1]==')' and ss[x+1]==' ':
                    ss[x]='\n'
            for x in range(len(ss)):
                if ss[x]=='\'' or ss[x]==')' or ss[x]=='(' or ss[x]==' ':
                    ss[x]=''
            i=0
            for x in range(len(ss)):
                if ss[x]==',':
                    ss[x]=' '
                    i=i+1
            datatext=''.join(ss)
            line_bot_api.reply_message(
                event.reply_token,
                TextSendMessage(text=datatext+"\n總金額:" + str(i*10)+"元")
            )
        except:
            line_bot_api.reply_message(
                event.reply_token,
                TextSendMessage(text='失敗了')
            )
```

```
elif event.message.text=='結帳':
    if isinstance(event.source, SourceUser):
        try:
            profile = line_bot_api.get_profile(event.source.user_id)
            profile_name=profile.display_name
            datatext=line_select_record(profile_name)
            line_delete_record(profile_name)
            ss=list(datatext)
            for x in range(len(ss)):
                if ss[x]==',' and ss[x+1]==')':
                    ss[x]=''
            for x in range(len(ss)):
                if ss[x]==',' and ss[x-1]==')' and ss[x+1]==' ':
                    ss[x]='\n'
            for x in range(len(ss)):
                if ss[x]=='\'' or ss[x]==')' or ss[x]=='(' or ss[x]==' ':
                    ss[x]=''
            i=0
            for x in range(len(ss)):
                if ss[x]==',':
                    ss[x]=' '
                    i=i+1
            datatext=''.join(ss)
            line_bot_api.reply_message(
                event.reply_token,
                TextSendMessage(text="總金額為"+str(i*10)+"元"+'\n謝謝您的惠顧')
            )
        except:
            line_bot_api.reply_message(
                event.reply_token,
                TextSendMessage(text='失敗了')
            )
```

▲圖 16、17. LINE Bot SDK 的函式

```
elif event.message.text=='name':
    if isinstance(event.source, SourceUser):
            profile = line_bot_api.get_profile(event.source.user_id)
            line_bot_api.reply_message(
                event.reply_token,[
                    TextSendMessage(text=profile.display_name)
                ]
            )
```

▲圖 18. LINE Bot SDK 的函式

圖 15、16、17、18. 利用 LINE Bot SDK 中的函式，依照接收到的指定訊息指定執行的動作。

```
def line_insert_record(record):

    db = pymysql.connect(host='us-cdbr-east-02.cleardb.com', user='b972090cd20237', passwd='145d9dee', db='heroku_35121a8821a20d5' )

    cursor = db.cursor()

    table_columns = '(username,product_name, product_much, product_quantity)'
    postgres_insert_query = f"""INSERT INTO manu {table_columns} VALUES (%s, %s, %s, %s);"""
    cursor.executemany(postgres_insert_query, record)
    db.commit()
    message = f"恭喜您！ {cursor.rowcount} 筆資料成功匯入 bill 表單！"
    cursor.close()
    db.close()

    return message
```

▲圖 19.讀取

```
def line_select_record(name):

    db = pymysql.connect(host='us-cdbr-east-02.cleardb.com', user='b972090cd20237', passwd='145d9dee', db='heroku_35121a8821a20d5' )
    cursor = db.cursor()
    postgres_insert_query = f"""select product_name,product_quantity from list where username='{name}';"""
    cursor.execute(postgres_insert_query)
    data=f"{cursor.fetchall()}"
    db.commit()
    db.close()
    return data
```

▲圖 20.寫入

```
def line_delete_record(name):

    db = pymysql.connect(host='us-cdbr-east-02.cleardb.com', user='b972090cd20237', passwd='145d9dee', db='heroku_35121a8821a20d5' )
    cursor = db.cursor()
    postgres_insert postgres_insert_query: str t where username='{name}';"""
    cursor.execute(postgres_insert_query)
    db.commit()
    db.close()
    return 0

if __name__ == "__main__":
    app.run()
```

▲圖 21.刪除

圖 19、20、21. 使用 PyMySQL 遠端連接資料庫，將讀取、寫入、刪除等功能寫成函式。

以下是機械手臂的程式碼:

```
#ifndef UNICODE
#define UNICODE
#endif
#include "KinovaTypes.h"
#include <iostream>
#include<fstream>
#include "libs\math\dot_net_example\boost_math\Stdafx.h"

#include "mysql_connection.h"
#include <cppconn/driver.h>
#include <cppconn/exception.h>
#include <cppconn/prepared_statement.h>

#ifdef __linux__
#include <dlfcn.h>
#include <vector>
#include "Kinova.API.CommLayerUbuntu.h"
#include "Kinova.API.UsbCommandLayerUbuntu.h"
#include <stdio.h>
#include <unistd.h>
#elif _WIN32
#include <Windows.h>
#include "CommunicationLayer.h"
#include "CommandLayer.h"
#include <conio.h>
#endif
```

▲圖 22. 程式碼引用的標頭檔

```
//Function pointers to the functions we need
int(*MyInitAPI)();
int(*MyCloseAPI)();
int(*MySendBasicTrajectory)(TrajectoryPoint command);
int(*MyGetDevices)(KinovaDevice devices[MAX_KINOVA_DEVICE], int& result);
int(*MySetActiveDevice)(KinovaDevice device);
int(*MyMoveHome)();
int(*MyInitFingers)();
int(*MyGetCartesianCommand)(CartesianPosition&);
```

▲圖 23.宣告機械手臂需要用到的變數、指標

```
#elif _WIN32
    //We load the API.
    commandLayer_handle = LoadLibrary(L"CommandLayerWindows.dll");

    //We load the functions from the library
    MyInitAPI = (int(*)()) GetProcAddress(commandLayer_handle, "InitAPI");
    MyCloseAPI = (int(*)()) GetProcAddress(commandLayer_handle, "CloseAPI");
    MyMoveHome = (int(*)()) GetProcAddress(commandLayer_handle, "MoveHome");
    MyInitFingers = (int(*)()) GetProcAddress(commandLayer_handle, "InitFingers");
    MyGetDevices = (int(*)(KinovaDevice devices[MAX_KINOVA_DEVICE], int& result)) GetProcAddress(commandLayer_handle, "GetDevices");
    MySetActiveDevice = (int(*)(KinovaDevice devices)) GetProcAddress(commandLayer_handle, "SetActiveDevice");
    MySendBasicTrajectory = (int(*)(TrajectoryPoint)) GetProcAddress(commandLayer_handle, "SendBasicTrajectory");
    MyGetCartesianCommand = (int(*)(CartesianPosition&)) GetProcAddress(commandLayer_handle, "GetCartesianCommand");
#endif
```

▲圖 24.取得機械手臂各項資訊

```
//Verify that all functions has been loaded correctly
if ((MyInitAPI == NULL) || (MyCloseAPI == NULL) || (MySendBasicTrajectory == NULL) ||
    (MyGetDevices == NULL) || (MySetActiveDevice == NULL) || (MyGetCartesianCommand == NULL) ||
    (MyMoveHome == NULL) || (MyInitFingers == NULL))

{
    cout << "* * * ERROR DURING INITIALIZATION * * *" << endl;
    programResult = 0;
}
else
{
    cout << "INITIALIZATION COMPLETED" << endl << endl;

    int result = (*MyInitAPI)();

    cout << "Initialization's result :" << result << endl;

    KinovaDevice list[MAX_KINOVA_DEVICE];

    int devicesCount = MyGetDevices(list, result);

    cout << "Send the robot to HOME position" << endl;
    MyMoveHome();

    cout << "Initializing the fingers" << endl;
    MyInitFingers();

    TrajectoryPoint pointToSend;
    pointToSend.InitStruct();

    cout << "Send the robot to HOME position" << endl;

    //We specify that this point will be an angular(joint by joint) position.
    pointToSend.Position.Type = CARTESIAN_POSITION;

    //We get the actual angular command of the robot.
    MyGetCartesianCommand(currentCommand);
```

▲圖 25.測試手臂能否與主機連接

```
MyGetCartesianCommand(currentCommand);
while (1)
{
    sql::Driver* driver;
    sql::Connection* con;
    sql::Statement* stmt;
    sql::PreparedStatement* pstmt;
    sql::ResultSet* result;
    try
    {
        driver = get_driver_instance();
        con = driver->connect("us-cdbr-east-02.cleardb.com",
            "b972090cd20237",
            "145d9dee");
    }
    catch (sql::SQLException e)
    {
        cout << "Could not connect to server. Error message: " << e.what() << endl;
        system("pause");
        exit(1);
    }
    con->setSchema("heroku_35121a8821a20d5");
    pstmt = con->prepareStatement("SELECT * FROM manu;");
    result = pstmt->executeQuery();

    while (result->next())
    {
        printf("Reading from table=(%s, %s, %s, %s)\n", result->getString(1).c_str(), result->getString(2).c_str(), result->getString(3).c_str(), result->getString(4).c_str());
        x=1;
    }
    delete result;
    delete pstmt;
    delete con;
```

▲圖 26.遠端連接 MySQL 資料庫

```
if (x == 1)
{
    pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X - 0.2444f;
    pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.3149f;
    pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z;
    pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX;
    pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY;
    pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ;
    pointToSend.Position.Fingers.Finger1 = 0;
    pointToSend.Position.Fingers.Finger2 = 0;
    pointToSend.Position.Fingers.Finger3 = 0;
    MySendBasicTrajectory(pointToSend);
    Sleep(1500);
    pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X - 0.2444f;
    pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.3149f;
    pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.4513f;
    pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 0.3469f;
    pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY + 0.3353f;
    pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ - 0.3702;
    MySendBasicTrajectory(pointToSend);
    Sleep(1500);
    pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X - 0.2244f;
    pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.3149f;
    pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.4513f;
    pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 0.3469f;
    pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY + 0.3353f;
    pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ - 0.3702;
    MySendBasicTrajectory(pointToSend);
    Sleep(1500);
    pointToSend.Position.Fingers.Finger1 = 4600;
    pointToSend.Position.Fingers.Finger2 = 4600;
    MySendBasicTrajectory(pointToSend);
    Sleep(1500);
    pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X - 0.2244f;
    pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.3149f;
    pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.3513f;
    pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 0.3469f;
    pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY + 0.3353f;
    pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ - 0.3702;
    MySendBasicTrajectory(pointToSend);
```

▲圖 27.Cartesian position type

```
    pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X - 0.2244f;
    pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.3149f;
    pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.3513f;
    pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 0.3469f;
    pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY + 0.3353f;
    pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ - 0.3702;
    MySendBasicTrajectory(pointToSend);
    Sleep(1500);
    pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X + 0.2526f;
    pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.2493f;
    pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.3517f;
    pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 1.3406f;0.3149f;
    pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.4513f;
    pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 0.3469f;
    pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY + 0.3353f;
    pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ - 0.3702;
    MySendBasicTrajectory(pointToSend);
    Sleep(1500);
    pointToSend.Position.Fingers.Finger1 = 4600;
    pointToSend.Position.Fingers.Finger2 = 4600;
    MySendBasicTrajectory(pointToSend);
    Sleep(1500);
    pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X - 0.2244f;
    pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.3149f;
    pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.3513f;
    pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 0.3469f;
    pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY + 0.3353f;
    pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ - 0.3702;
    MySendBasicTrajectory(pointToSend);
    Sleep(1500);
```

▲圖 28.Cartesian position type

```
pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X + 0.2526f;
pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.2493f;
pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.3517f;
pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 1.3406f;
pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY + 0.3518f;
pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ - 1.3998f;
MySendBasicTrajectory(pointToSend);
Sleep(1500);
pointToSend.Position.CartesianPosition.X = currentCommand.Coordinates.X + 0.2526f;
pointToSend.Position.CartesianPosition.Y = currentCommand.Coordinates.Y - 0.2493f;
pointToSend.Position.CartesianPosition.Z = currentCommand.Coordinates.Z - 0.4517f;
pointToSend.Position.CartesianPosition.ThetaX = currentCommand.Coordinates.ThetaX + 1.3406f;
pointToSend.Position.CartesianPosition.ThetaY = currentCommand.Coordinates.ThetaY + 0.3518f;
pointToSend.Position.CartesianPosition.ThetaZ = currentCommand.Coordinates.ThetaZ - 1.3998f;
MySendBasicTrajectory(pointToSend);
Sleep(1500);
pointToSend.Position.Fingers.Finger1 = 0;
pointToSend.Position.Fingers.Finger2 = 0;
MySendBasicTrajectory(pointToSend);
Sleep(1500);
MyMoveHome();
```

▲圖 29.Cartesian position type

圖 27、28、29. 使用 Cartesian position type，控制機械手臂的在座標軸上的移動。

第五章 專題探討

我們目前的專題成果只算是一個基本的架構，它有著眾多的缺點，它無法移動，無法做影像識別來抓取物件，沒辦法精準的送到客人手上，也沒辦法進行裝袋打包，但是他的可擴展性和發展是相當高的。

如果可以為它加上一雙“腿”，利用滾動式履帶和動線規劃，可以幫機器人設計出一條專屬於他的送餐路線，讓它可以送到各桌，如此一來才可以算是真正意義上的可以取代服務生；另外也在手臂上加裝攝像頭，使他有視覺的功能再去搭配上手臂的多軸多關節的特色，將會可以將餐點送達至更多位置，甚至可以精準送到客人手上。除此之外，也可以讓機器人在送餐的途中達到避障的功能，不要讓他去撞倒客人或是桌子之類的。

在 LINE Bot 的部分，也是有很大的開發空間，LINE Bot designer 上面有很多的圖形化界面的功能，像是在對話視窗下面的選單功能，或是可以有更多的與使用者互動的功能我們都還沒實作出來，這些是未來我們可以呈現出來的功能。

不過在自動化服務上，有個難以跨越的橫溝—無法給客戶有“溫度”的服務，一旦自動化，就會變得是由冷冰冰的機器人來服務顧客了，所以不太可能會有全自動化服務餐廳的出現，就連台北開的一間自動牛肉麵店，它也是只有煮麵的部分是自動的，還是有外場的服務生進行服務的。

第六章 結論

無人化服務是近年來一直被熱烈討論的主題，從機械手搖飲料店，再到自動牛肉面店，自動駕駛以及無人超市的出現，無人化服務已經漸漸改變我們的日常生活了。我們所做的專題是在自動化服務生這個產品之一，但是我們的特點是我們的機器手臂具有輕便，可以模擬人的手臂，可以固定在任何平面上的，這些將會讓它具有很高的發展性，給他加裝移動零件以及攝影機，輔以 AI 和影像處理能力，是真的可以利用這種機器手臂做成一個完整的機器服務生的。

第七章 參考文獻

1. 大大通-Heroku 平台介紹：https://www.wpgdadatong.com/tw/blog/detail?BID=B0236
2. KINOVA：https://www.kinovarobotics.com/en/products/gen2-robot
3. 掌宇股份有限公司：https://www.kandh-edu.com.tw/web/product-detail.php?pro_no=450
4. LINE Developers-Messaging API 介紹：https://developers.line.biz/zh-hant/docs/messaging-api/overview/
5. LINE Developer-LINE Bot Designer：https://developers.line.biz/zh-hant/services/bot-designer/
6. LINE Developers-Messaging API SDKs：https://developers.line.biz/zh-hant/docs/messaging-api/line-bot-sdk/#official-sdks
7. MySQL-Introduction to Connector/c++：https://dev.mysql.com/doc/connector-cpp/1.1/en/connector-cpp-introduction.html
8. Heroku Dev Center：https://devcenter.heroku.com/articles/cleardb
9. CSDN-Mysql connector c++：https://blog.csdn.net/bandaoyu/article/details/105975777
10. Boost：https://www.boost.org/
11. MySQL connector/C++：https://dev.mysql.com/downloads/connector/cpp/
12. Medium-Hello Python：https://medium.com/python4u/hello-python-509eabe5f5b1
13. Medium-使用PyMySQL模組：https://yanwei-liu.medium.com/python%E8%B3%87%E6%96%99%E5%BA%AB%E5%AD%B8%E7%BF%92%E7%AD%86%E8%A8%98-%E5%9B%9B-%E4%BD%BF%E7%94%A8pymysql%E6%A8%A1%E7%B5%84-79c9941088d2

14. TWCODE01-PyMySQL驅動：https://www.twcode01.com/python3/python3-mysql.html
15. Fooish：https://www.fooish.com/sql/delete-from.html
16. ITREAD：https://www.itread01.com/content/1546714024.html