東海大學資訊工程學系
專題報告書

搭載電磁砲自動化發射控制之車輛
Automatically Firing Control Vehicle with Electromagnetic Gun

中華民國　　110 年　　12 月 15 日

摘要

科技蓬勃發展的同時，隨之而來的是愈發講究安全性和遠端遙控的便利性，搭配之後 5G 技術發展，這種遠端遙控的即時性與便利性將顯得更加重要，甚至有人考慮過遙控無人機外送的行業，要是在這些遠端遙控的載具再加上各式各樣的功能，想必會有非常廣的用途。

現今影像辨識以及物聯網技術逐漸發展的同時，也有許多企業將自家產品融入部分影像辨識或是物聯網技術，將遠端遙控載具、影像辨識以及物聯網技術加以結合。

在實作上先以軍事用途作為假設，自行製作整個車體結構並搭載電磁砲，將頭盔配合傳感器將頭盔旋轉之數據傳送至載具，並在載具上運算發射裝置應以什麼角度以及力道發射，以及透過操控 App 的方式遠端遙控載具之移動、發射和發射裝置之充能等等。

關鍵詞：電磁砲、物體追蹤、射控、OpenCV。

目錄

圖目錄

第1章 緒論

1.1 研究背景

在無人載具技術趨近成熟的現今，各式各樣的載具推陳出新，像是特斯拉的「智慧召喚」功能、無人化的自駕車物流服務、甚至是結合軌道的新穎無人外送「智慧點餐」，就連 Google、Apple 等企業也有意投入無人載具發展，未來很可能會有更多無人載具的應用，為此我們想知道在已如此多樣化的發展下，能否搭配所學再有更多新穎的變化，研究並製造或嘗試改良更新一代的無人載具。

1.2 研究動機與目的

在網路上看到有公司以無人載具為基礎，研發有關智慧教育機器人的產品用於教育事業，如 DJI 發售的 RoboMaster S1 便是一個例子，又看到有影音創作者嘗試手動製作一台迷你電磁砲，效果雖不大卻令我們感到十分興奮有趣，在上網查證過相關科學原理後，我們也想憑藉自身所學，配合改良過後的射擊功能，從無到有不假手他人依靠自己組裝實作出一台功能相似甚至是性能更佳的無人車。

1.3 研究方法與流程

本次研究目的是實作一台遙控車，與一般常見遙控車不同之處在於，用微型開發版樹莓派 4B 作為主體單元，提供電力、運算能力及整體控制單元，以 3D 列印客製需要之部件，添加以電磁能量作為動力發射之電磁砲，輔以鏡頭提供視訊串流，並利用網路技術實現遠端及時操控之功能，實現整體無人載具目的。

第2章 相關軟硬體介紹

2.1 樹梅派 4B

樹莓派（英語：Raspberry Pi）英國樹莓派基金會開發的微型單板電腦，體積小巧且功能豐富，目的是以低價硬體及自由軟體促進學校的基本電腦科學教育。

每一代開發版均使用博通（Broadcom）出產的 ARM 架構處理器，如今生產的機型（樹莓派 4B）記憶體在 2GB 和 8GB 之間，以 TF 卡作爲系統儲存媒介（初代使用 SD 卡），配備有 USB 介面、HDMI 的視訊輸出（支援聲音輸出）、3.5mm 音源孔與 GPIO 腳位，內建 Ethernet/WLAN/Bluetooth 網路連接的方式（依據型號決定），並且可使用多種操作系統，包含官方推出的 Raspberry Pi OS、各式 Linux 發行版以及嵌入式 Windows IOT。

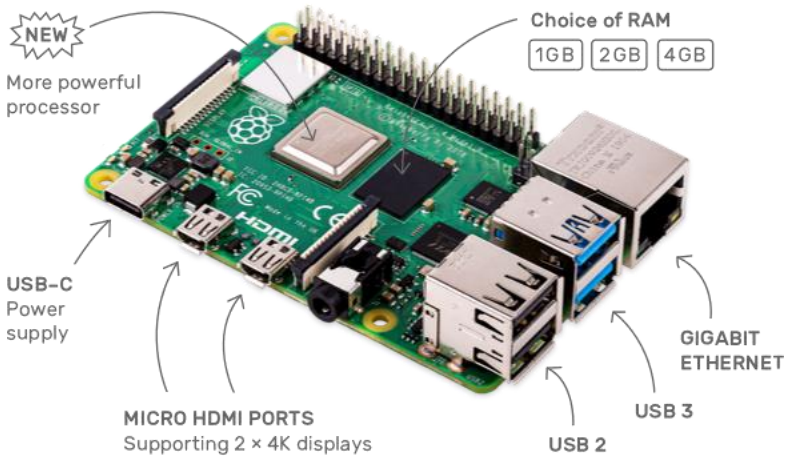


圖 2.1 樹梅派 4B

2.2 NVIDIA JETSON XAVIER NX

NVIDIA® Jetson Xavier™ NX 為小尺寸模組系統 (SOM)，但卻具備極致的超級電腦效能，更提供多種介面接口，滿足開發者的不同需求。

NVIDIA® Jetson Xavier™ NX 有高達 21 兆次運算的加速運算，擁有 384 個 NVIDIA CUDA® 核心、48 個 Tensor 核心、6 個 Carmel ARM CPU 與兩個 NVIDIA 深度學習加速器（NVDLA）引擎的效能，適合在嵌入式和邊緣系統中進行高效能運算，結合 8GB 共 128 位元的 LPDDR4x 記憶體超過每秒 59.7 GB 的記憶體頻寬、影片編碼與解碼功能，能以平行方式執行多個現代類神經網路，並同時處理多感應器的高解析度資料。

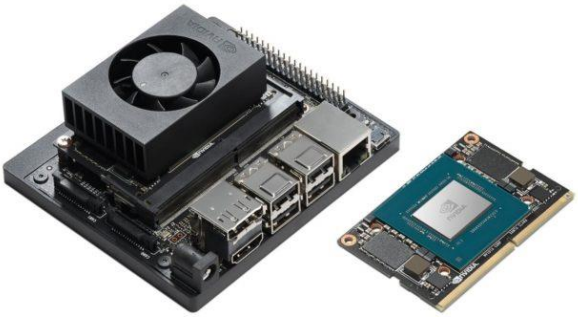


圖 2.2 NVIDIA JETSON XAVIER NX

2.3 樹梅派專用攝影鏡頭模組（Raspberry PI RPI Camera）

樹梅派專用鏡頭模組，使用軟性排線與樹梅派上的 CSI 端口連接傳遞訊號，隨插即用的特性，再加上重量極輕，非常適合移動或延時攝影、影片錄製、動作檢測、安全應用等需求。

圖 2.3 Raspberry PI RPI Camera

2.4 步進馬達（28BYJ-48-5）與驅動板

步進馬達為「無刷直流馬達」的一種，藉由切換流向線圈中的電流，用一定角度逐步轉動馬達，且用脈波訊號切換電流，不需檢測位置與速度的回傳裝置，又依訊號比例轉動，達成精確的位置與速度控制，穩定性佳。

圖 2.4 5v 步進馬達（28BYJ-48-5）與驅動板

2.5 微型金屬減速馬達（GA12-N20）

減速馬達是指減速機和馬達的合成體，又稱為齒輪馬達、齒輪電動機，連接對應直流馬達後，可有效降低輸出轉速並增大輸出轉矩，藉由齒輪互相連結而獲得較大扭力，更可依據齒輪減速箱的結構而有不同表現。

圖 2.5 微型金屬減速馬達（GA12-N20）

2.6 麥克納姆輪

由瑞典的麥克納姆公司發明而得名，在車輪外環安裝與軸心成 45 度角排列輥筒，運行時輥筒上之摩擦力會與輪軸產生相應 45 度反推力，此斜向推力會分為縱向及橫向兩個向量，而車體各自車輪產生之相應向量，其共同合力會決定車體最終運動方向。

透過調節各車輪轉速與運行方向，得以實現前進、後退、橫移、斜行、自體旋轉等移動方式，適用於空間狹窄，路徑多直角等場地。

圖 2.6 麥克納姆輪

2.7 L298N 馬達驅動模組

L298N 是一款能接受高電壓的馬達驅動板，不論是直流馬達或步進馬達都能驅動，單一片驅動芯片就能同時控制兩個直流減速馬達，並且具有過熱自斷和反饋檢測功能，防止電路燒毀。

L298N 可對馬達直接控制，設定完板上的 I/O 輸入，就可控制馬達的正反轉運行，不但操作簡單、穩定性好，更能滿足部分直流馬達的高電流驅動要件。

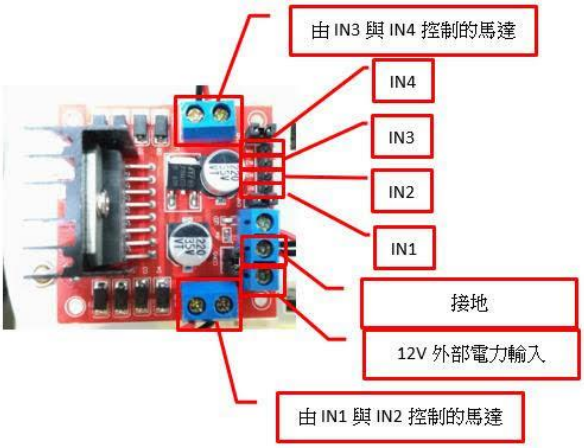


圖 2.7 L298N 馬達驅動模組

2.8 50V 升壓模組

升壓器也稱為升降壓轉換器，常用以提昇電壓的 DC-DC 轉換器，其輸出（負載）電壓會比輸入（電源）電壓要高。

圖 2.8 50V 升壓模組

2.9 繼電器

繼電器是一種電子控制元件，具有控制和被控制系統，通常應用於自動控制電路中，是用較小的電流去控制較大電流的一種「自動開關」，在電路中起到自動調節、安全保護、轉換電路等作用。

圖 2.9 繼電器

2.10 矽控整流器（Silicon Controlled Rectifier）

矽控整流器，也可稱作半導體控制整流器、晶閘管、SCR，是一種三端固態電子元件，具備陽極、陰極和閘極，廣泛應用於各種電子電路，與一般二極體相比，可對電流進行控制，以小電流（電壓）控制大電流（電壓），體積小、重量輕、功號低、效率高及開關迅速等優點。

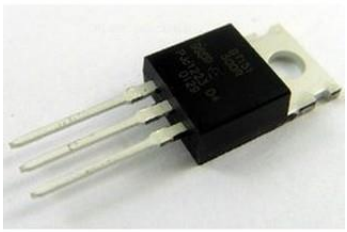


圖 2.10 控整流器（Silicon Controlled Rectifier）

2.11 數位類比轉換器（MCP3008）

MCP3008 是一種可以將類比的電壓訊號轉成數位訊號的電子元件，並透過 SPI 協定與 Raspberry Pi 進行互動，其數位訊號的解析度為 10 位元，最多可以有 2^{10}=1024 種數字組合。

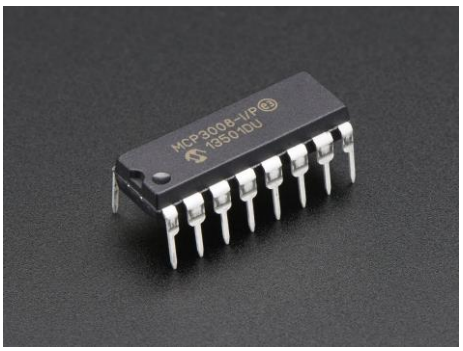


圖 2.11 數位類比轉換器（MCP3008）

2.12 高效整流二極體（HER308GW）

二極體具備一種 P 型半導體與 N 型半導體結合的「PN 接面」，「P 型」端稱為陽極，「N 型」端稱作陰極，電流只能由陽極流往陰極，擁有「順向特性」，達到「整流作用」，也就是能將交流電轉換為直流電使用。

圖 2.12 高效整流二極體（HER308GW）

圖 2.12

2.13 舵機驅動板（PCA9685）

PCA9685 採用 I2C 通訊介面，內置一個 16 路的 PWM 產生器和 clock，解析度為 12bit，開關頻率範圍為 24Hz 到 1526Hz。

圖 2.13 舵機驅動板（PCA9685）

2.14 電磁鎖

電磁鎖或稱磁力鎖（Magnetic lock），是一種利用電生磁原理製造的電子設備，當電流通過鋼片，電磁鎖會產生強大的吸力緊緊吸附達成鎖門效果。一般而言弱電型居多，只需 12 伏電壓搭配非常小的電流，就能產生非比尋常的磁吸力，內部也無複雜機械結構和鎖舌，能有效控制開與關。

圖 2.14 電磁鎖

3D 列印，又稱立體列印、增材製造、積層製造，以各式 3D 圖檔為基礎，多種方式列印，相較傳統製造工法，3D 列印更省材料成本，成品質量精巧細緻且應用廣泛。

其中以 FDM 機型最易入門，FDM 是熱熔融層積成型的一種 3D 列印技術，將可熱塑的線材加熱透過噴頭擠出，堆疊列印成形，冷卻固化後不需加工處理即可獲取成品。

圖 2.15 Ping 270 DIY

2.15 Rasberry Pi OS

Raspberry Pi OS（原為 Raspbian）是樹莓派基金會以 Debian 為基底核心開發之作業系統。自 2015 年起，樹莓派基金會正式將其作為樹莓派的官方作業系統。

Raspberry Pi OS 目前在最新版本中提供新的桌面環境：PIXEL（Pi Improved X-Window Environment, Lightweight），由修改過的 LXDE 桌面環境和 Openbox 視窗管理員組成，提供用戶簡易、輕量化且對硬體需求降至最低的基礎圖形化介面。

2.16 24 路電路旋轉連接器

電路旋轉連接器是將電子訊號從靜態結構的一端，傳遞至另一轉動結構的電機裝置，可應用於各種需要傳輸功率、類比訊號或數位訊號的轉動系統，能夠有效改善機械性能，簡化操作系統及降低晃動帶來的機械損害。

2.17 Python

Python 是一種易於學習、功能強大且廣大使用的高階直譯式語言，屬於通用型程式語言（未限定特殊用途而設計），支援多種程式設計用法，函數式、指令式、結構化、物件導向與反射式程式，擁有動態型別系統和垃圾回收機制，可自動管理記憶體使用，自身還設有一巨大廣泛的函式庫。

Python 設計哲學強調程式碼可讀性和簡潔好懂的語法，例如以空格縮排取代傳統大括弧區分程式碼區塊。相較傳統的 C 語言或 Java ， Python 優雅的語法和動態型別，讓開發者用更少代碼清楚明瞭表達想法，程式架構一目了然。

2.18 Flask

Flask 是一個使用 Python 撰寫的輕量級 Web 應用程式框架，由於其輕量化特性與僅實現 Web 部分應用的簡單核心功能，也稱作「Micro-Framework（微框架）」，其餘功能都需額外擴充套件支援實現。

2.19 HTML

HTML 全名 HyperText Markup Language，是一種「超文本標記語言」，用於組織並描述網頁結構，由一系列元素組成，像是段落、圖片、清單或表格，用於告知瀏覽器如何呈現網頁並定義內容片段。

2.20 CSS

CSS 全名 Cascading Style Sheets，又稱階層式樣式表、串樣式列表、級聯樣式表，用於為結構化文件，例如 HTML 文件，添加字型、顏色、排版的風格頁面語言。

2.21 JavaScript

JavaScript 是一種進階、直譯式程式語言，能在前端網頁實現複雜華麗的特效，提供像是及時更新、地圖互動、繪製 2D/3D 圖形、動畫呈現和影片播放控制等功能。

現也能透過像是「Node.js」等環境運行於後端伺服器內，提供「應用程式介面（API）」，提供程式額外接口。

2.22 Dart

Dart 是由 Google 開發的物件導向程式語言，可跨平台進行開發，設計上優先考慮多平台環境，Web 端、移動端、桌面端，提供開發者高品質環境體驗。擁有預先編譯 AOT（Ahead Of Time）特色，編譯成各平台原生程式碼，進一步提升效能。

Dart 富含充沛的函式庫，為日常撰寫程式碼提供許多必要工具，而將所有東西視為物件的特性，讓所有物件皆從類別延伸拓展，一切事物皆為物件，易於佈局和代碼閱讀。

2.23 Flutter

以「Dart」為基礎進行應用的軟體開發套件框架，擁有跨平台特性，撰寫一份代碼就能於 Android 和 IOS 運行，同時保有趨近於原生應用程式的效能。

與 Dart 相似之處在於將所有事物視為小部件「Widget」，一切皆由「Widget」構成，任何功能都能透過組合排列多個 Widget 實現。

熱重啟功能（Hot Reload）讓開發者不需重新等待程式碼編譯執行時間，編輯程式後，儲存或熱重啟即可立即更新至正在運行設備上，第一時間得知修改後呈現樣式，大幅縮短開發時間，獲得絕佳開發效率。

2.24 Blender

Blender 是開放原始碼（Open-Source）的開源 3D 繪圖軟體，具備 3D 建模、算圖、動畫繪製、影片剪接等功能，甚至能跨平台運行，而 Blender 相較其他軟體，不但產生檔案體積小、功能齊全，甚至完全免費，在所有平台都能提供用戶一致的體驗。

2.25 切片軟體（Ping Slicer）

切片軟體用於完成建模後，分層切割 STL 檔，並調整詳細參數設定，而參數設定大多影響成品外觀，例如噴頭溫度和移動速度、平台溫度、接縫位置等細項，存成 gcode 檔後，提供機器識別打印成品。

第3章 研究方法與架構

3.1 系統功能概述

本專題自走砲以單一嵌入式微電腦處理硬體控制，並且透過各種零組件達成移動、瞄準、充能、射擊等功能；而使用者端則會有一套介面來無線控制自走砲並且觀看串流畫面及車體資訊。

3.2 系統架構

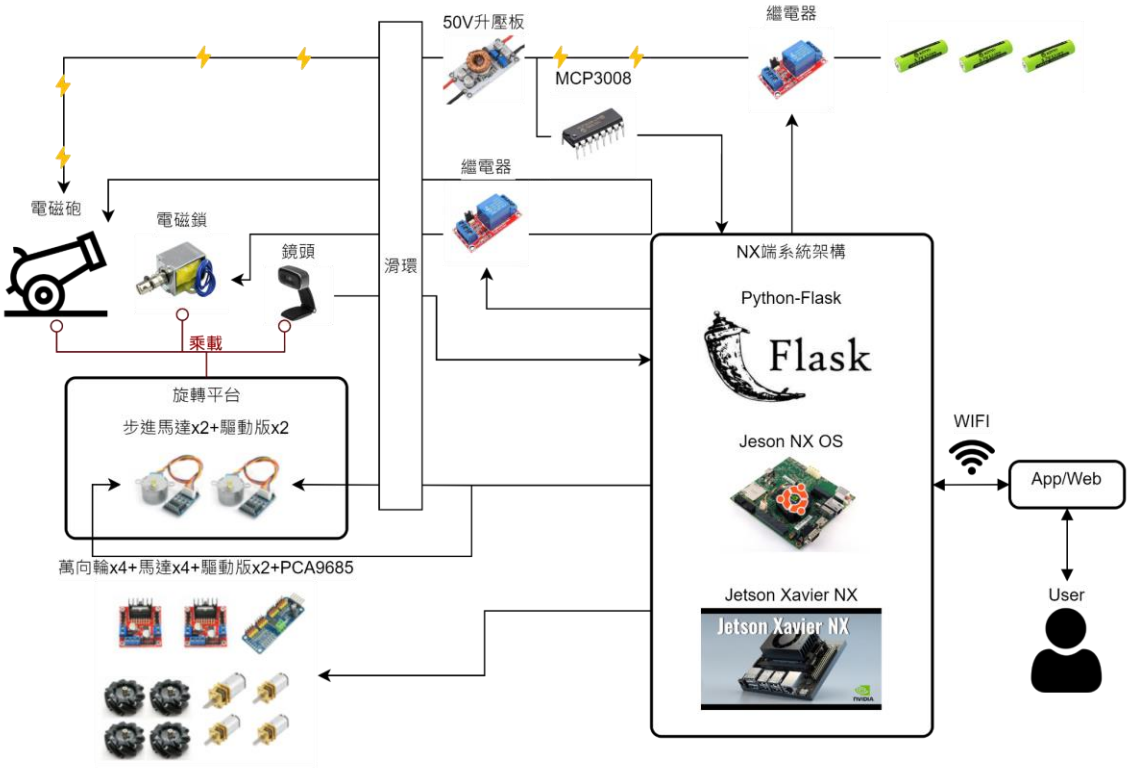


圖 3.1 系統架構示意圖

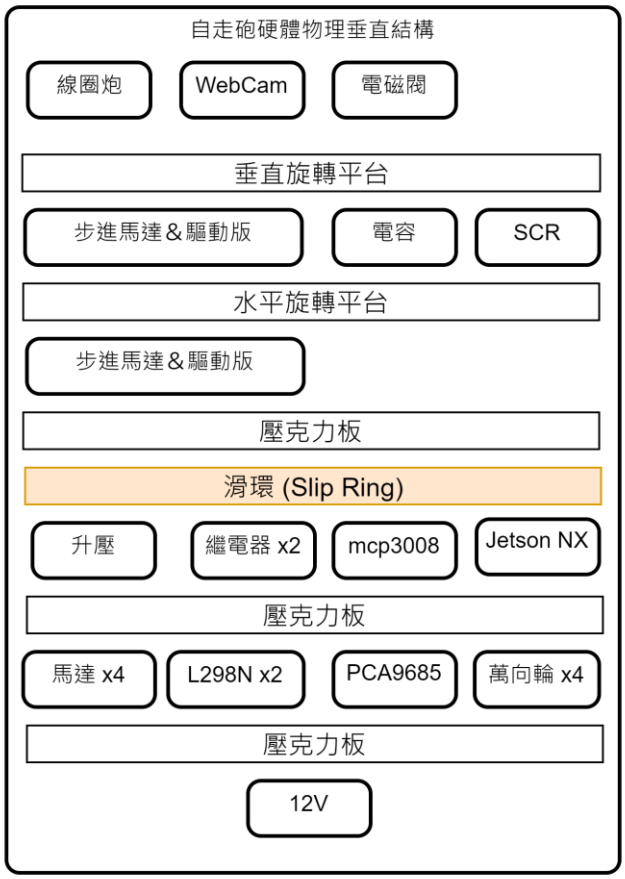


圖 3.2 硬體物理垂直結構

專題以樹莓派為系統主體，並以 Python 為主要使用語言，Python 負責了整個樹莓派的操控程式碼以及其他的運算，以 Html 和 Javascript 為網頁端的架構。

3.3 研究方法

3.3.1 建置 Flask 搭配鏡頭伺服器

隨著參考資料，在樹莓派上建置 Python 檔，並且利用 Python 的 Flask 套件建構一個網頁的架構，讓使用者可以透過網頁看到串流畫面，以 Html 撰寫網頁讓 Flask 提供頁面。

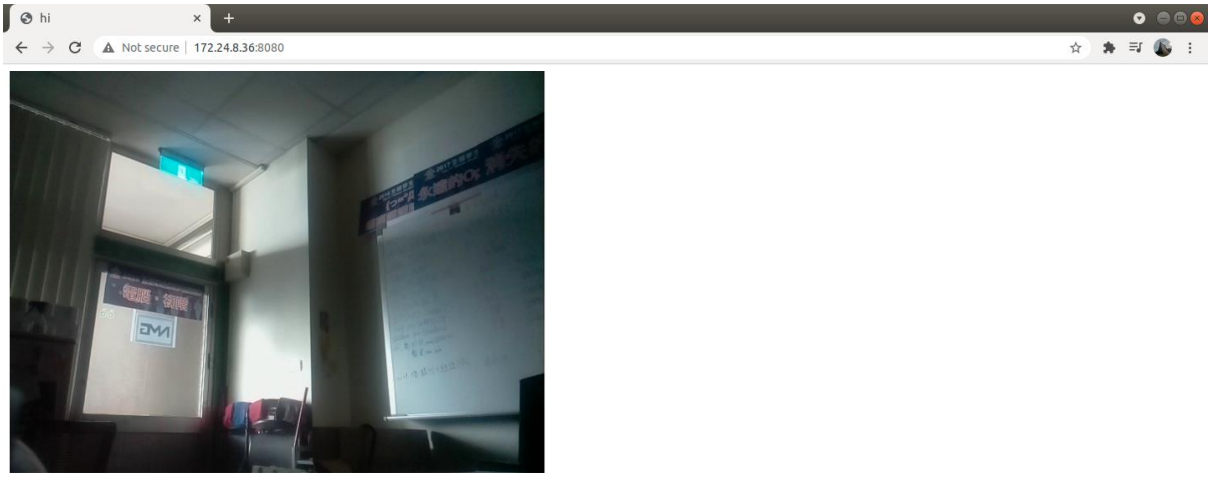


圖 3.3 串流網頁

```
import cv2
from imutils.video.pivideostream import PiVideoStream
import imutils
import time
import numpy as np

class VideoCamera(object):
    def __init__(self, flip = False):
        self.vs = PiVideoStream().start()
        self.flip = flip
        time.sleep(2.0)

    def __del__(self):
        self.vs.stop()

    def flip_if_needed(self, frame):
        if self.flip:
            return np.flip(frame, 0)
        return frame

    def get_frame(self):
        frame = self.flip_if_needed(self.vs.read())
        ret, jpeg = cv2.imencode('.jpg', frame)
        return jpeg.tobytes()
```

```
from camera import VideoCamera

@app.route('/video_feed')
def video_feed():
    return Response(gen(pi_camera),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

pi_camera = VideoCamera(flip=False) # flip pi camera if upside down.
```

圖 3.4 初始鏡頭串流程式碼

3.3.2 建置電子元件控制系統

利用 Python 控制 GPIO 接腳，並且利用 Flask 套件傳送 API，讓使用者可以透過網頁遠端操控，JavaScript 負責執行使用端的動作。初期採用將 GPIO 不同的操作放置於不同的網址底下，當進入至某個網址時，GPIO 便會自動執行相應的操作，並且採用跳轉網址的方式來操作電子元件。這導致了之後在加上鏡頭畫面時，在操作時會看見畫面重新整理。之後更改操作，採用 Javascript 提供的 fetch 語法將指令傳送給後端的 Python 並執行相應的操作，這徹底解決了使用者在操作時會有畫面重新整理的不適感。由於配合另一組需求將樹莓派更改為 NX 使用。

```python
from flask import Flask, render_template,request
import datetime
import RPi.GPIO as gpio
import time
```

```python
return render_template("index.html")
```

```python
if __name__ == "__main__":
    app.run(host='192.168.1.140', port=8001, debug=True)
```

圖 3.5 Python Flask 程式碼

```javascript
document.onkeydown = function (e) {
    let ImageHeight = window.innerHeight;
    let ImageWidth = ImageHeight / 3 * 4 / 2;
    let Scale = ImageHeight / 360;
    let horizontal = 0, vertical = 0;
    // W key
    if (!e.repeat) {
        if (e.keyCode == 87 && Forward != 1) {
            for (var i = 0; i < 4; i++) {
                MoveData[i] += 1;
            }
            Move(MoveData, speed);
            Forward = 1;
        }
        // S key
        if (e.keyCode == 83 && Back != 1) {
            for (var i = 0; i < 4; i++) {
                MoveData[i] -= 1;
            }
            Move(MoveData, speed);
            Back = 1;
        }
        // D key
        if (e.keyCode == 68 && Right != 1) {
            MoveData[0] -= 1;
            MoveData[1] += 1;
            MoveData[2] += 1;
            MoveData[3] -= 1;
            Move(MoveData, speed);
            Right = 1;
        }
        // A key
```

圖 3.6 傳送指令的程式碼

3.3.3 車子移動控制

車子採用了 4 顆減速馬達搭配 2 個由 12V 電池供電的 L298N 馬達驅動板操控 4 顆輪子，在網頁端設置 JavaScript 的 keydown 鍵盤事件，設置四個按鍵來傳送指令，分別為操控車子的前、後、向左轉、向右轉，車子的停下採用同樣四個按鍵的 keyup 來操控，讓車子可以在鍵盤放開的瞬間立即停止移動。

為求增加車子移動靈活性，將普通的輪子更換為麥克納姆輪，更換輪子後車子向左右轉的靈活度更高，並且在網頁端增加兩個按鍵為車子增加橫向移動的功能。

增加 PCA9685 控制板用於類比輸出，將 L298N 從只能輸出固定電壓更改為透過 PCA9685 控制板用 PWM 輸出操控 L298N 輸出給馬達的電壓，提供給網頁、APP 將輪子變速。

PCA9685 在 import Library 後，呼叫裡面的函式來使用。

L298N 與 PCA9685 電路接法如圖 3.8 所示。

```
try:
    pwms = PCA9685.PCA9685()
    pwms.reset()
    pwms.showInfo()
except:
    print("\x1b[31m",'Warning: PCA9685 Fail!',"\x1b[39m")

wheels_list = ['lf','rf','lb','rb']
```

```
@app.route('/wheels', methods=['POST'])
def wheels():
    data = request.json

    try:
        for key in data.keys():
            index = wheels_list.index(key)
            if int(data[key])>=0 :
                pwms.setValChOff(4*index,0)
                pwms.setValChOff(4*index+2,int(data[key]))
            else:
                pwms.setValChOff(4*index,int(-data[key]))
                pwms.setValChOff(4*index+2,0)
    except:
        print("\x1b[31m",data,"\x1b[39m")
        print("\x1b[31m",'Warning: PCA9685 Fail!',"\x1b[39m")
    return '', 200
```

圖 3.7 PCA9685 程式碼

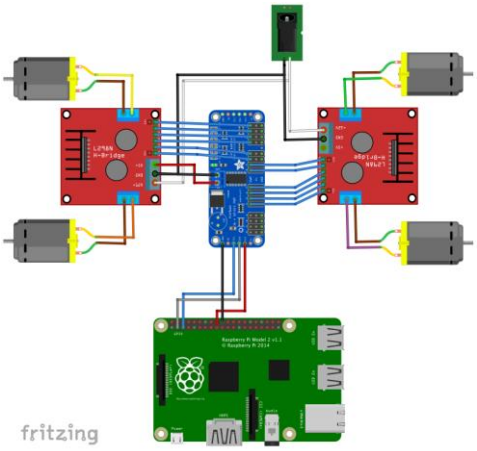


圖 3.8 PCA9685 電路圖

3.3.4 電磁砲電路圖繪製

將電磁砲的基本電路繪製完成，以電池、線圈、電容、開關和二極體構成完整結構，藉由線圈通電形成的磁力來讓金屬製子彈向前移動，電容用以儲存電池能量並對線圈達到瞬間放電的效果，藉由瞬間放電的特性讓子彈通過線圈之後不會因為磁力的干擾而停留在線圈中，以達到順利發射的目的。根據冷次定律線圈在電流突變時會產生感應電流，二極體為防止感應電流破壞其他 IC 元件的裝置。

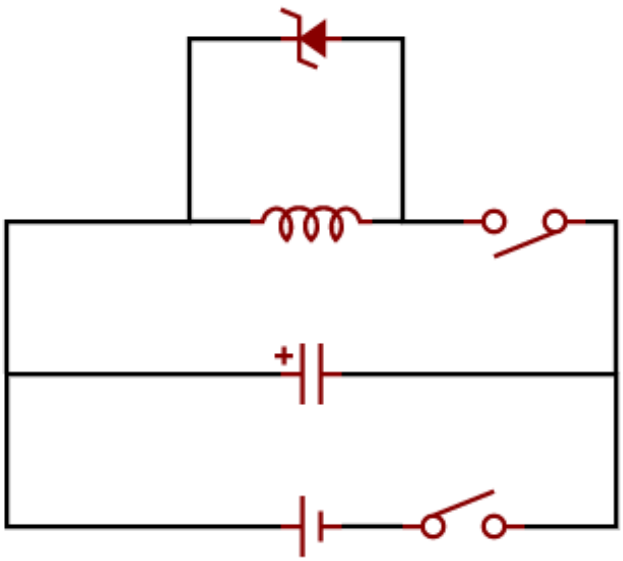


圖 3.9 電磁砲電路圖

將原先手動控制的開關更改為透過 NX 控制繼電器及 SCR，增加升壓模板及更大電壓的電容以提升線圈磁力強度，由於繼電器無法承受電容短路時的電流大小故選用可以承受較大電流的 SCR 作為控制電容放電的開關，繼電器放置於電源接至升壓模版的線路上用以控制電容的充電。在電容充電完成之後，需先將繼電器斷路才可將 SCR 通路，不然金屬子彈會無法正常發射，且因為 SCR 特性緣故無法在 SCR 通路之後主動斷路，需先將前方繼電器斷路才可將 SCR 斷路。

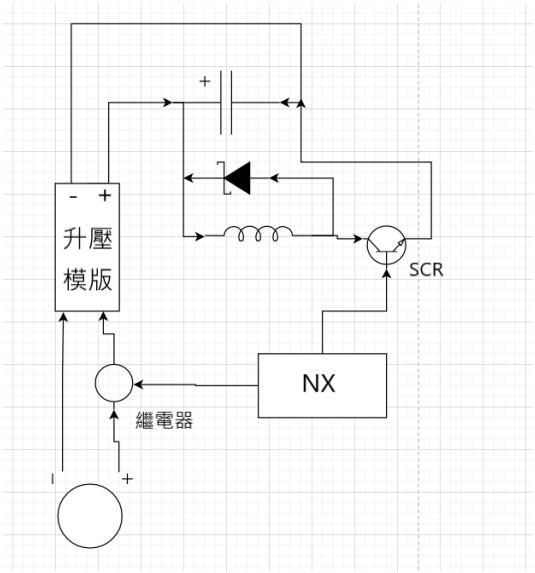


圖 3.10 由 NX 控制電磁砲之電路圖

3.3.5 設計子彈供給系統

以手槍彈匣為參考設計，利用 Blender 繪製 3D 模型再由 3D 列印機製作初版彈匣並測試可用性。

圖 3.11 初版彈匣

用彈簧將子彈送至膛室並以彈簧固定子彈。

若單純以彈匣中的彈簧將子彈推入砲管並固定住子彈，由於彈簧力道過強將無法順利發射，嘗試增加電磁鎖解決此問題，但電磁鎖的力道及行程不足以將子彈推出彈匣，參考手槍槍機將子彈送入槍管的概念重新設計彈匣結構，並以砲管、砲身、彈匣三個部件製作。

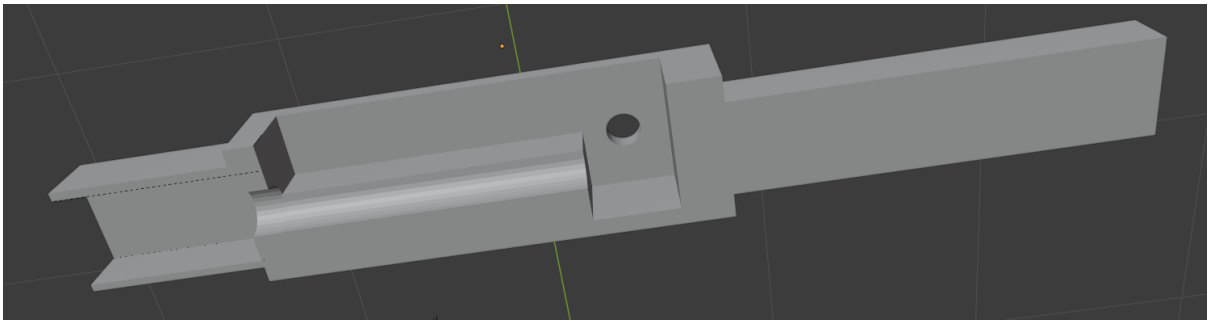


圖 3.12 砲身剖面圖

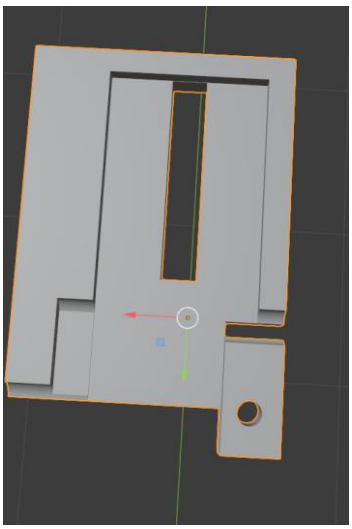


圖 3.13 最終版彈匣

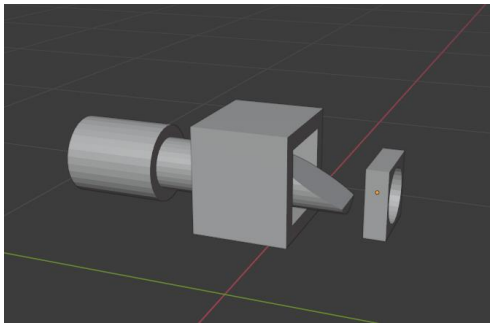


圖 3.14 彈匣卡榫

彈匣右下方圓孔為彈匣卡榫卡住彈匣的孔洞，下方左邊的凹槽為連接砲管的空間，透過彈簧將子彈推至第一個缺口固定住，上膛時由電磁鎖從後方缺口

將子彈向前推並藉由彈簧的力道將子彈送入砲管中，並在砲身下方外側黏一個磁鐵用以固定砲管中的金屬子彈。

彈匣卡榫設計配合彈簧運作，置於砲身孔洞上，平時卡榫將會因為彈簧扣進砲身的孔洞中，透過卡榫的斜面設計彈匣可直接推入砲身中並透過彈簧的力量扣入卡榫，退出彈匣時只需將卡榫拉起便可退出彈匣。

增加電磁鎖及周邊所需電子元件之電路圖如圖 3.13 所示。

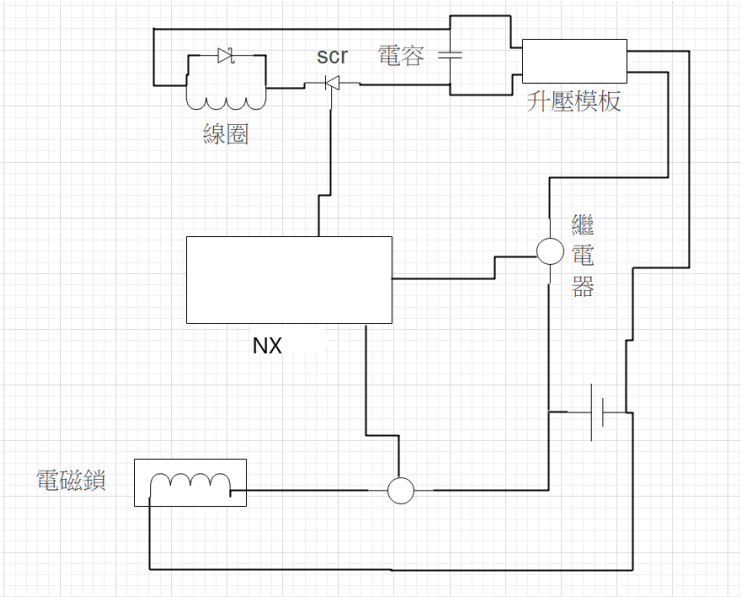


圖 3.15 砲管周邊電路圖

```
if meth == 'shot':
    gpio.output(bar,True)
    time.sleep(0.2)
    gpio.output(bar,False)
elif meth =='re':
    gpio.output(Re,True)
elif meth =='re stop':
    gpio.output(Re,False)
```

圖 3.16 電磁砲操控操控程式碼

圖 3.14

3.3.6 上方旋轉結構

為求 360 度旋轉功效，達到全方位射擊功能，嘗試模擬過各種連接方案。下圖為第一版旋轉結構，實際列印後發現經拆分後零件過於細小，不利膠合，加上尚未對 3D 列印技術有一定的熟悉度，導致成品良率偏低。

後續研究並參考坦克砲塔與底盤吊籃之內部構造連接設計，以齒輪為主要運動之機械結構，底部一端拉伸拓展成圓形平台，再往上延展固定各部件之溝槽，利用 3D 列印一體成形特性，再透過電路旋轉連接器串通上下電路，穩固整體結構。

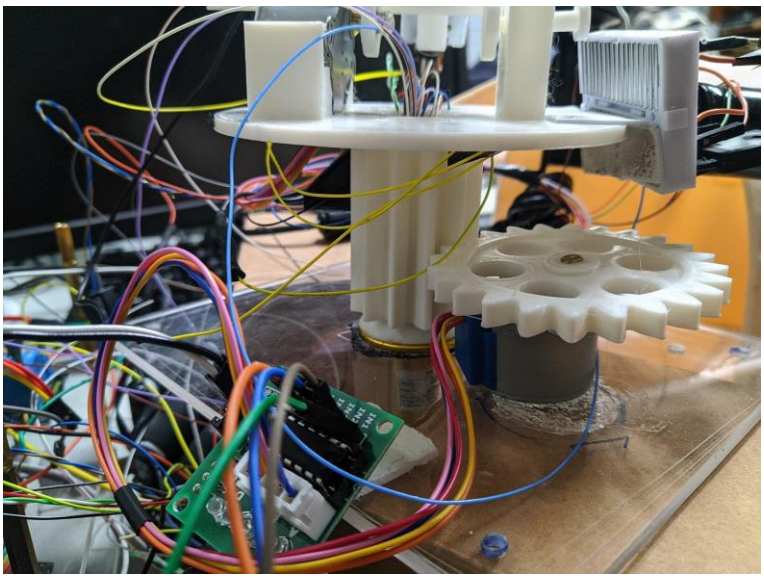


圖 3.17 旋轉平台結構（一）

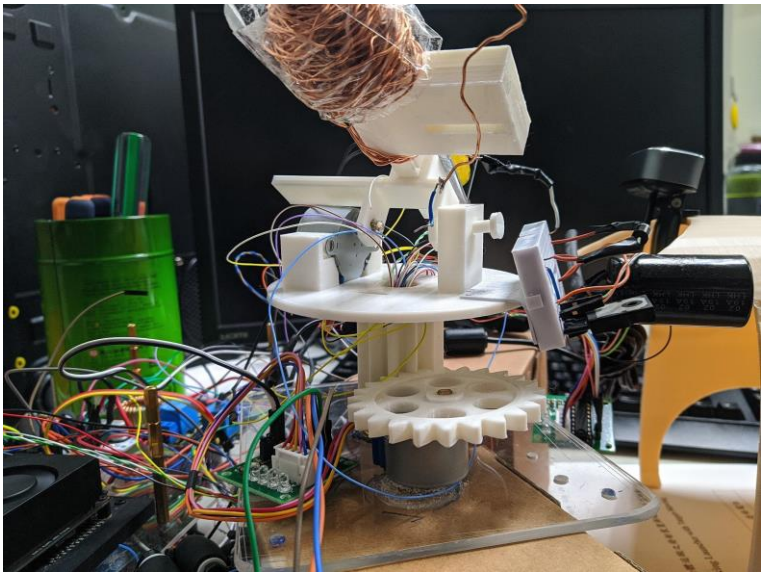


圖 3.18 旋轉平台結構（二）旋轉電路

在上方平台旋轉時，實體電路將會旋轉糾纏。因此為求連續無限制旋轉，增加了電路旋轉連接器來解決這個問題。在途中會有因線路過長導致訊號干擾過大，影響有傳輸訊號需求的元件發生問題，對此電路採用雙絞線纏繞及增加抗干擾磁環抵銷干擾。

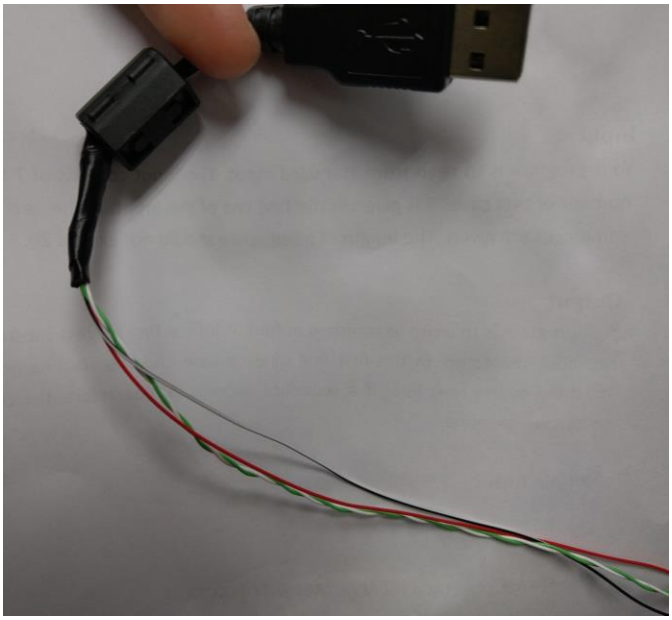


圖 3.19 雙絞線及抗干擾磁環

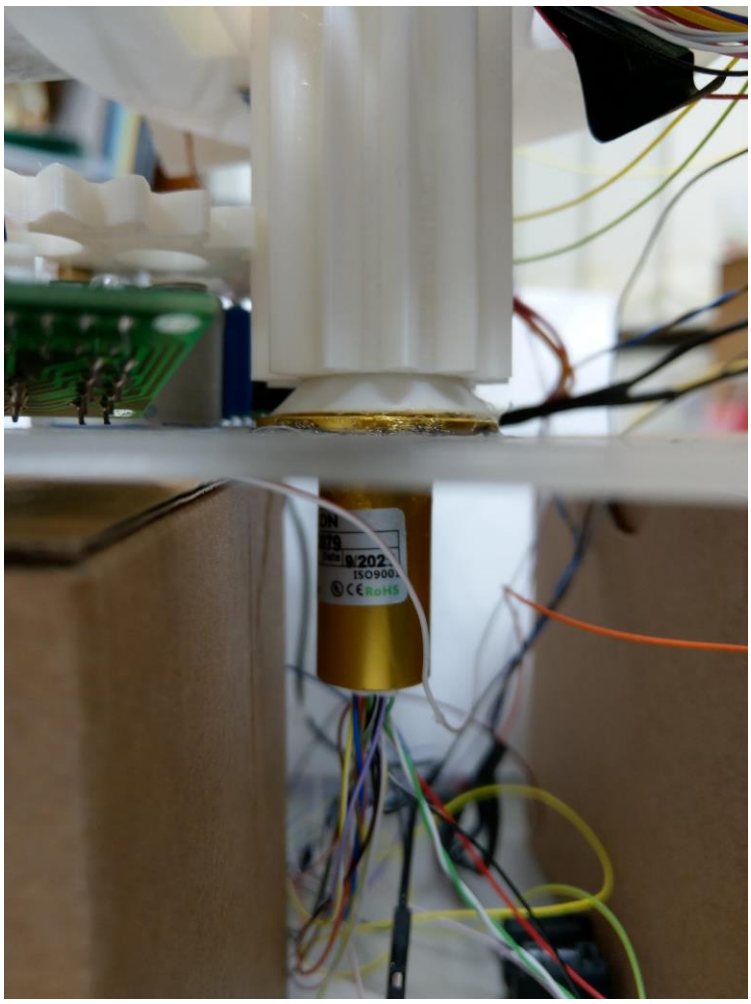


圖 3.20 滑環

3.3.7 設計整體外觀車殼

以日本原創定格動畫天竺鼠車車腳色「馬鈴薯/ポテト」為發想主題，利用 Blender 繪製 3D 模型，切片後交由 3D 列印機打印測試模型設計和尺寸。

圖 3.21 馬鈴薯/ポテト示意圖

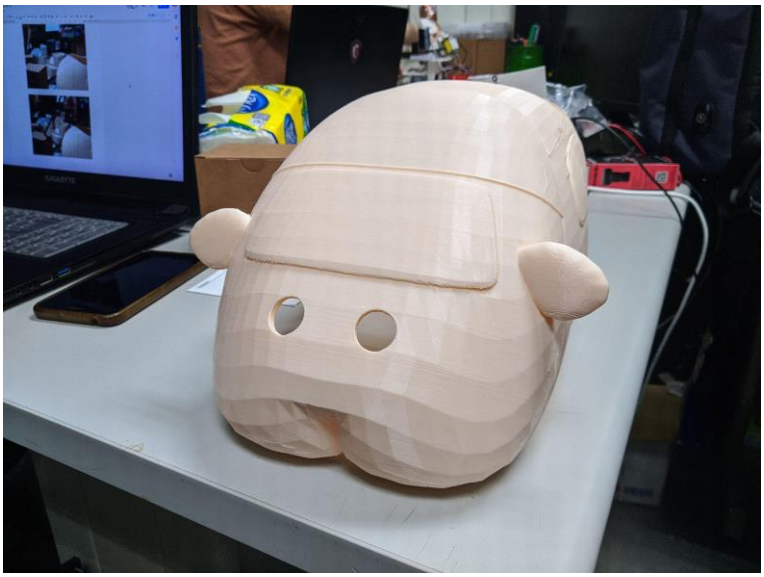


圖 3.22 車殼成品圖

3.3.8 電壓讀取

讀取電容充能的部分。透過分壓電路讓 0-50V 等比降壓成約 0-3.3V。再接入 MCP3008 數位類比轉換器的針腳，以 SPI 介面傳送 DFKai-SB 數值給 NX。

程式碼部分參考其他程式碼，再查詢電壓差讀取方式後，加以改寫成讀取電壓差的方式。

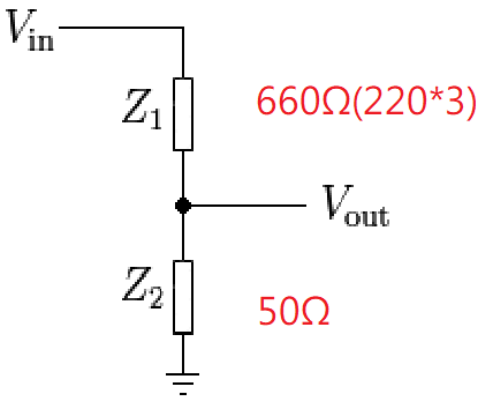


圖 3.23 分壓電路

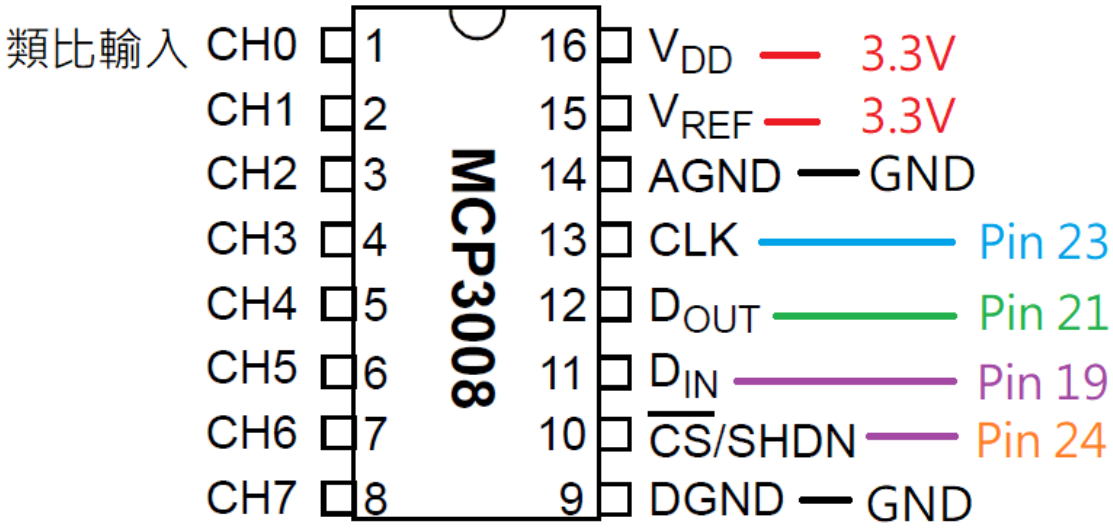


圖 3.24 MCP3008 電路圖

```
def readDiff(self, channel):
    adc = self.spi.xfer2([1,(channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2]
    return data
```

```
adc = mcp3008.MCP3008()
```

```
@app.route('/volt', methods=['GET'])
def volt():
 volt = adc.diffVolts(2)
 return str(volt)
```

圖 3.25 MCP3008 程式碼建置 WIFI 基地台

參考教學，先設定好 wlan IP，建置好 DHCP 伺服器，再來建置 hostapd 服務。

```
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 172.24.8.24
netmask 255.255.255.0
gateway 172.24.8.254

auto wlan0
iface wlan0 inet static
address 192.168.1.1
netmask 255.255.255.0
```

圖 3.26 /etc/network/interfaces 設定檔

```
default-lease-time 600;
max-lease-time 7200;
ddns-update-style none;
authoritative;
subnet 192.168.1.0 netmask 255.255.255.0 {
   range 192.168.1.2 192.168.1.50;
   option routers 192.168.1.1;
   option broadcast-address 192.168.24.255;
   default-lease-time 600;
   max-lease-time 7200;
```

圖 3.27 /etc/default/isc-dhcp-server 設定檔

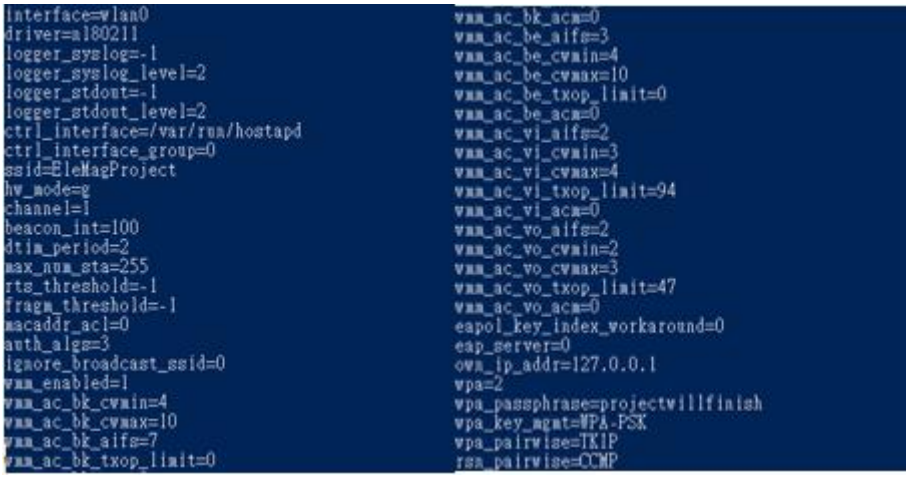


圖 3.28 /etc/hostapd/hostapd.conf 設定檔

3.3.9 建置手機應用程式

手機應用程式主要用 Flutter 開發。App 與 Server 的連線是透過 WIFI，以 Server 那端為基地台，主要連線也都是透過 http 連線 Server 端架設的 Flask 提供的 API。

首先於畫面串流，App 透過"flutter_mjpeg"這個 Package 裡的元件接收 Server 端傳出來的 MJPEG 格式，顯示於畫面中央上。

在左上方定時向 Server 取得電壓資訊；中右側是各種功能按鈕：充能切換按鈕、瞄準切換按鈕、射擊按鈕。

移動元件參考後加以改進。左下角圓形搖桿元件，給予離中心距離、方向數值；右下角滑桿元件，給予離中心距離。透過這些數據計算出四顆麥克納姆倫所應轉動之轉速，整理成 JSON 格式，持續傳送資料給 Server。

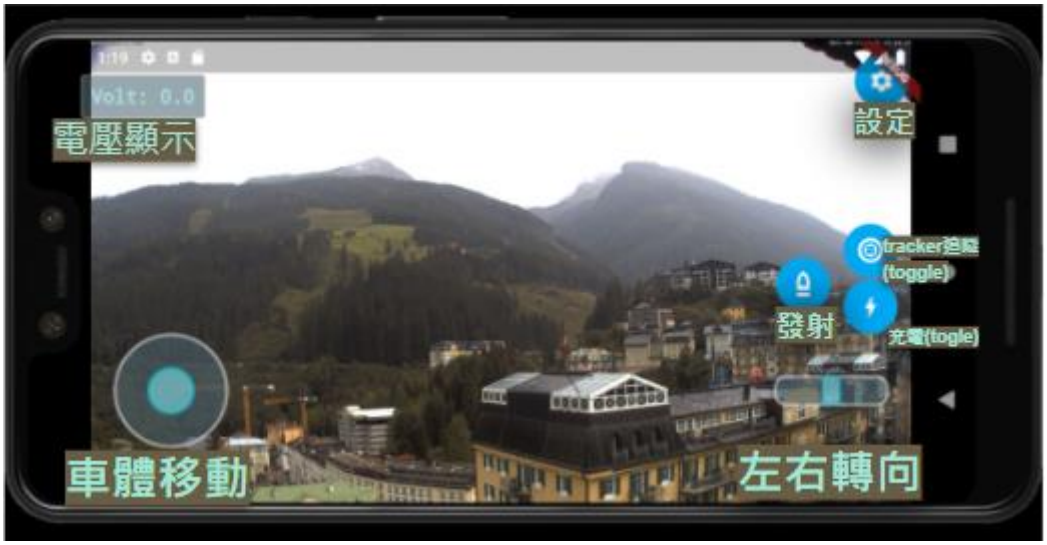
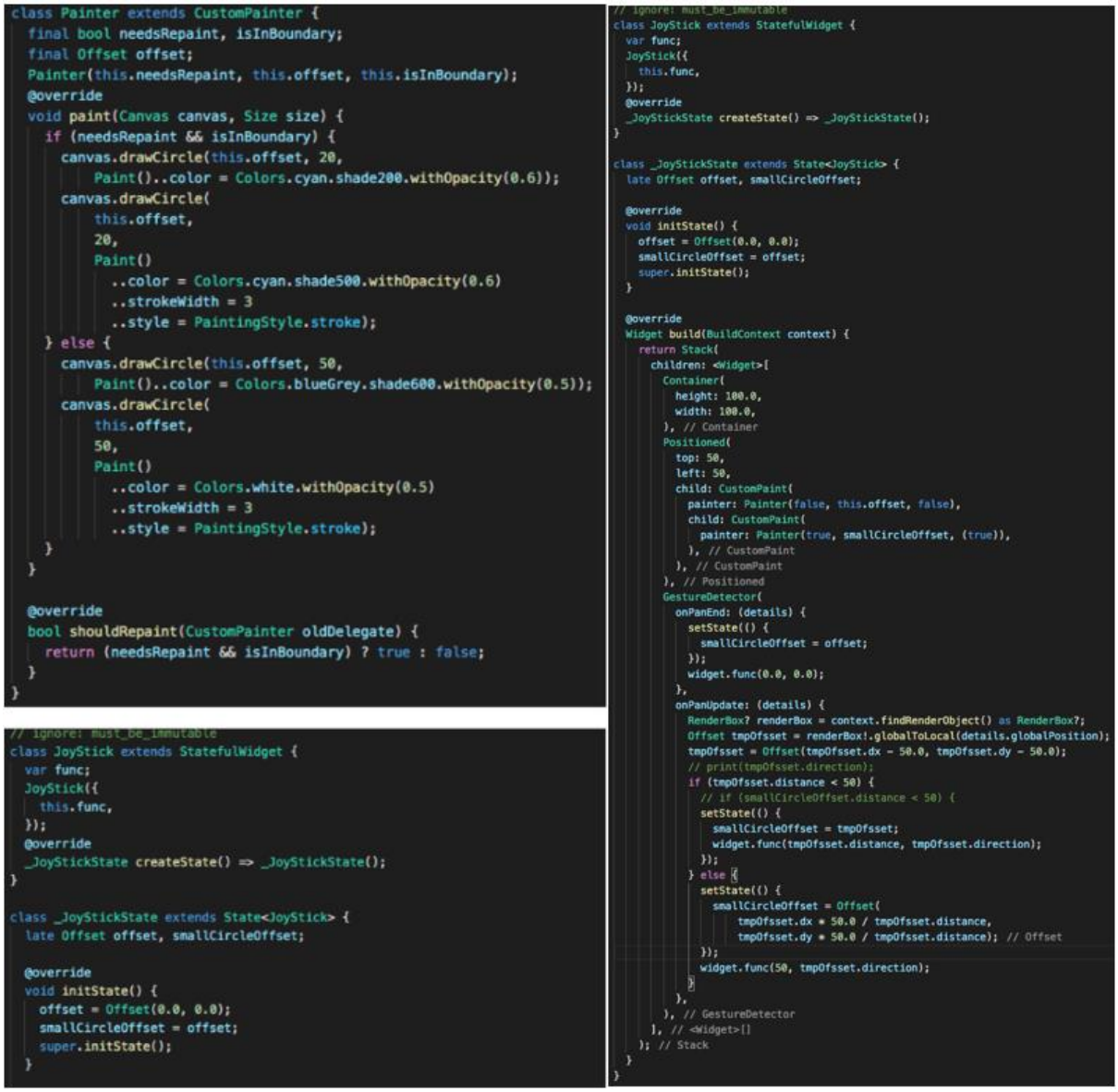


圖 3.29 APP 畫面概念

```
void wheelsTimer() {
  try {
    http.post(
      Uri.parse(uri_ip + 'wheels'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
      body: jsonEncode({'start': 'begin', 'test': 1}),
    );
  } catch (e) {}
  Timer.periodic(Duration(milliseconds: 100), (timer) {
    // ignore: unused_local_variable
    var lf, rf, lb, rb;
    Map wheels = {'lf': 0.0, 'rf': 0.0, 'lb': 0.0, 'rb': 0.0};
    if (speed == 0) {
      wheels = {
        'lf': 0.0,
        'rf': 0.0,
        'lb': 0.0,
        'rb': 0.0,
      };
    } else if (direction.abs == pi / 2 || direction.abs == pi) {
      wheels = {
        'lf': speed,
        'rf': speed,
        'lb': speed,
        'rb': speed,
      };
    } else if (direction == 0 || direction.abs == pi) {
      wheels = {
        'lf': (direction.abs() / pi) * 2 - 1 * speed,
        'rf': (direction.abs() / pi) * 2 - 1 * -speed,
        'lb': (direction.abs() / pi) * 2 - 1 * -speed,
        'rb': (direction.abs() / pi) * 2 - 1 * speed,
      };
    } else if (pi / 2 <= direction) {
      wheels = {
        'lf': -((direction - pi / 4 * 3) / pi * 4) * speed,
        'rf': speed,
        'lb': speed,
        'rb': -((direction - pi / 4 * 3) / pi * 4) * speed,
      };
    } else if (pi / 2 > direction && direction >= 0) {
      wheels = {
        'lf': speed,
        'rf': ((direction - pi / 4) / pi * 4) * speed,
        'lb': ((direction - pi / 4) / pi * 4) * speed,
        'rb': speed,
      };
    } else if (-pi / 2 <= direction && direction <= 0) {
      wheels = {
        'lf': ((direction + pi / 4) / pi * 4) * speed,
        'rf': -speed,
        'lb': -speed,
        'rb': ((direction + pi / 4) / pi * 4) * speed,
      };
    } else if (-pi / 2 >= direction) {
      wheels = {
        'lf': -speed,
        'rf': -((direction + pi / 4 * 3) / pi * 4) * speed,
        'lb': -((direction + pi / 4 * 3) / pi * 4) * speed,
        'rb': -speed,
      };
    }
    print(wheels['lf']);
    wheels = {
      'lf': wheels['lf'] + turn,
      'rf': wheels['rf'] - turn,
      'lb': wheels['lb'] + turn,
      'rb': wheels['rb'] - turn,
    };
    wheels.forEach((key, value) {
      if (value > 4095) {
        value = 4095.0;
      } else if (value < -4095) {
        value = -4095.0;
      }
    });
    try {
      http.post(Uri.parse(uri_ip + 'wheels'),
          headers: <String, String>{
            'Content-Type': 'application/json; charset=UTF-8',
          },
          body: json.encode(wheels));
    } catch (e) {}
  }); // Timer.periodic
}
```

圖 3.30 麥克納姆輪的操控程式碼



```dart
class Painter extends CustomPainter {
  final bool needsRepaint, isInBoundary;
  final Offset offset;
  Painter(this.needsRepaint, this.offset, this.isInBoundary);
  @override
  void paint(Canvas canvas, Size size) {
    if (needsRepaint && isInBoundary) {
      canvas.drawCircle(this.offset, 20,
          Paint()..color = Colors.cyan.shade200.withOpacity(0.6));
      canvas.drawCircle(
          this.offset,
          20,
          Paint()
            ..color = Colors.cyan.shade500.withOpacity(0.6)
            ..strokeWidth = 3
            ..style = PaintingStyle.stroke);
    } else {
      canvas.drawCircle(this.offset, 50,
          Paint()..color = Colors.blueGrey.shade600.withOpacity(0.5));
      canvas.drawCircle(
          this.offset,
          50,
          Paint()
            ..color = Colors.white.withOpacity(0.5)
            ..strokeWidth = 3
            ..style = PaintingStyle.stroke);
    }
  }

  @override
  bool shouldRepaint(CustomPainter oldDelegate) {
    return (needsRepaint && isInBoundary) ? true : false;
  }
}
```

```dart
// ignore: must_be_immutable
class JoyStick extends StatefulWidget {
  var func;
  JoyStick({
    this.func,
  });
  @override
  _JoyStickState createState() => _JoyStickState();
}

class _JoyStickState extends State<JoyStick> {
  late Offset offset, smallCircleOffset;

  @override
  void initState() {
    offset = Offset(0.0, 0.0);
    smallCircleOffset = offset;
    super.initState();
  }
```

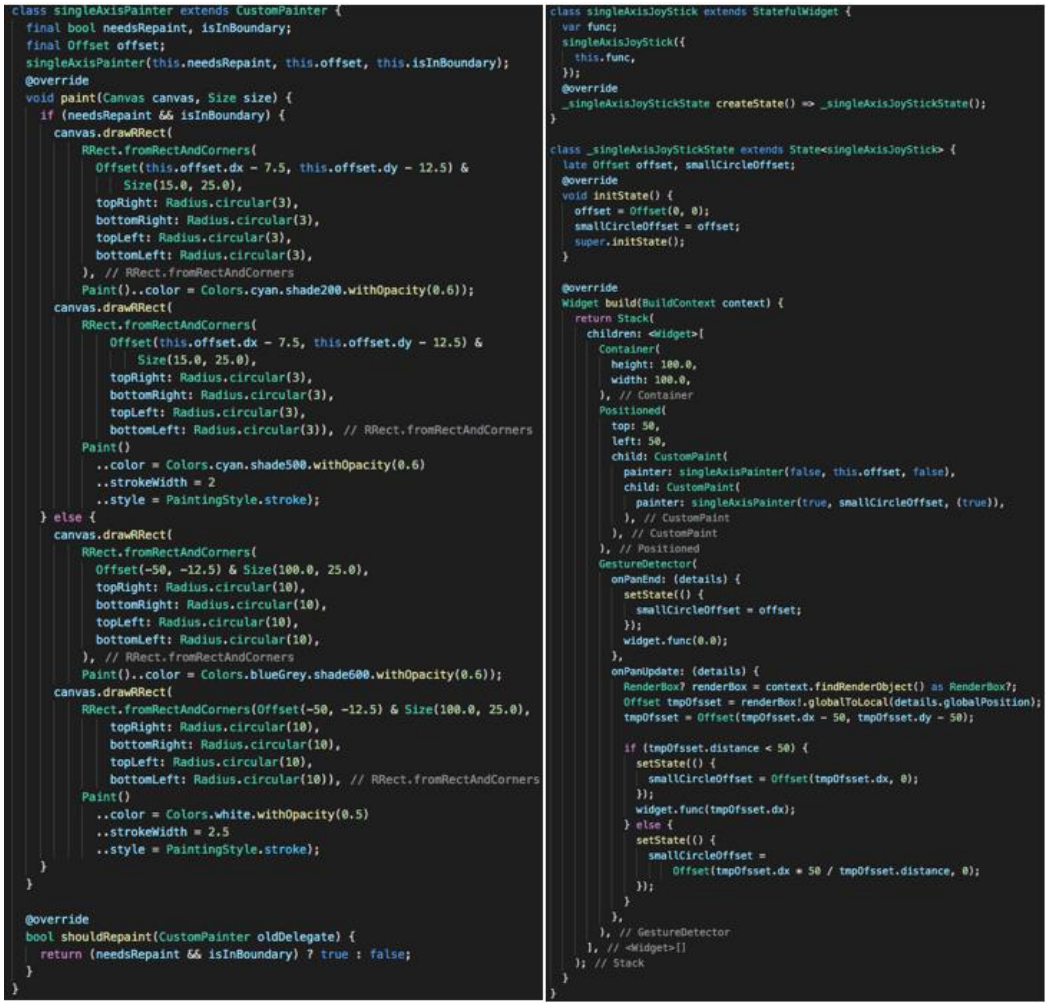
```dart
// ignore: must_be_immutable
class JoyStick extends StatefulWidget {
  var func;
  JoyStick({
    this.func,
  });
  @override
  _JoyStickState createState() => _JoyStickState();
}

class _JoyStickState extends State<JoyStick> {
  late Offset offset, smallCircleOffset;

  @override
  void initState() {
    offset = Offset(0.0, 0.0);
    smallCircleOffset = offset;
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      children: <Widget>[
        Container(
          height: 100.0,
          width: 100.0,
        ), // Container
        Positioned(
          top: 50,
          left: 50,
          child: CustomPaint(
            painter: Painter(false, this.offset, false),
            child: CustomPaint(
              painter: Painter(true, smallCircleOffset, (true)),
            ), // CustomPaint
          ), // CustomPaint
        ), // Positioned
        GestureDetector(
          onPanEnd: (details) {
            setState(() {
              smallCircleOffset = offset;
            });
            widget.func(0.0, 0.0);
          },
          onPanUpdate: (details) {
            RenderBox? renderBox = context.findRenderObject() as RenderBox?;
            Offset tmpOfsset = renderBox!.globalToLocal(details.globalPosition);
            tmpOfsset = Offset(tmpOfsset.dx - 50.0, tmpOfsset.dy - 50.0);
            // print(tmpOfsset.direction);
            if (tmpOfsset.distance < 50) {
              // if (smallCircleOffset.distance < 50) {
              setState(() {
                smallCircleOffset = tmpOfsset;
                widget.func(tmpOfsset.distance, tmpOfsset.direction);
              });
            } else {
              setState(() {
                smallCircleOffset = Offset(
                    tmpOfsset.dx * 50.0 / tmpOfsset.distance,
                    tmpOfsset.dy * 50.0 / tmpOfsset.distance); // Offset
              });
              widget.func(50, tmpOfsset.direction);
            }
          },
        ), // GestureDetector
      ], // <Widget>[]
    ); // Stack
  }
}
```

圖 3.31 搖桿元件程式碼



```dart
class singleAxisPainter extends CustomPainter {
  final bool needsRepaint, isInBoundary;
  final Offset offset;
  singleAxisPainter(this.needsRepaint, this.offset, this.isInBoundary);
  @override
  void paint(Canvas canvas, Size size) {
    if (needsRepaint && isInBoundary) {
      canvas.drawRRect(
          RRect.fromRectAndCorners(
            Offset(this.offset.dx - 7.5, this.offset.dy - 12.5) &
                Size(15.0, 25.0),
            topRight: Radius.circular(3),
            bottomRight: Radius.circular(3),
            topLeft: Radius.circular(3),
            bottomLeft: Radius.circular(3),
          ), // RRect.fromRectAndCorners
          Paint()..color = Colors.cyan.shade200.withOpacity(0.6));
      canvas.drawRRect(
          RRect.fromRectAndCorners(
              Offset(this.offset.dx - 7.5, this.offset.dy - 12.5) &
                  Size(15.0, 25.0),
              topRight: Radius.circular(3),
              bottomRight: Radius.circular(3),
              topLeft: Radius.circular(3),
              bottomLeft: Radius.circular(3)), // RRect.fromRectAndCorners
          Paint()
            ..color = Colors.cyan.shade500.withOpacity(0.6)
            ..strokeWidth = 2
            ..style = PaintingStyle.stroke);
    } else {
      canvas.drawRRect(
          RRect.fromRectAndCorners(
            Offset(-50, -12.5) & Size(100.0, 25.0),
            topRight: Radius.circular(10),
            bottomRight: Radius.circular(10),
            topLeft: Radius.circular(10),
            bottomLeft: Radius.circular(10),
          ), // RRect.fromRectAndCorners
          Paint()..color = Colors.blueGrey.shade600.withOpacity(0.6));
      canvas.drawRRect(
          RRect.fromRectAndCorners(Offset(-50, -12.5) & Size(100.0, 25.0),
              topRight: Radius.circular(10),
              bottomRight: Radius.circular(10),
              topLeft: Radius.circular(10),
              bottomLeft: Radius.circular(10)), // RRect.fromRectAndCorners
          Paint()
            ..color = Colors.white.withOpacity(0.5)
            ..strokeWidth = 2.5
            ..style = PaintingStyle.stroke);
    }
  }

  @override
  bool shouldRepaint(CustomPainter oldDelegate) {
    return (needsRepaint && isInBoundary) ? true : false;
  }
}
```

```dart
class singleAxisJoyStick extends StatefulWidget {
  var func;
  singleAxisJoyStick({
    this.func,
  });
  @override
  _singleAxisJoyStickState createState() => _singleAxisJoyStickState();
}

class _singleAxisJoyStickState extends State<singleAxisJoyStick> {
  late Offset offset, smallCircleOffset;
  @override
  void initState() {
    offset = Offset(0, 0);
    smallCircleOffset = offset;
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      children: <Widget>[
        Container(
          height: 100.0,
          width: 100.0,
        ), // Container
        Positioned(
          top: 50,
          left: 50,
          child: CustomPaint(
            painter: singleAxisPainter(false, this.offset, false),
            child: CustomPaint(
              painter: singleAxisPainter(true, smallCircleOffset, (true)),
            ), // CustomPaint
          ), // CustomPaint
        ), // Positioned
        GestureDetector(
          onPanEnd: (details) {
            setState(() {
              smallCircleOffset = offset;
            });
            widget.func(0.0);
          },
          onPanUpdate: (details) {
            RenderBox? renderBox = context.findRenderObject() as RenderBox?;
            Offset tmpOfsset = renderBox!.globalToLocal(details.globalPosition);
            tmpOfsset = Offset(tmpOfsset.dx - 50, tmpOfsset.dy - 50);

            if (tmpOfsset.distance < 50) {
              setState(() {
                smallCircleOffset = Offset(tmpOfsset.dx, 0);
              });
              widget.func(tmpOfsset.dx);
            } else {
              setState(() {
                smallCircleOffset =
                    Offset(tmpOfsset.dx * 50 / tmpOfsset.distance, 0);
              });
            }
          },
        ), // GestureDetector
      ], // <Widget>[]
    ); // Stack
  }
}
```

圖 3.32 滑桿元件程式碼

3.3.10 將各部件整合

裁切好符合需求大小的壓克力板作為車體底板，將車體分為三層，電源供應以三顆 3.7V 的 18650 電池串聯給需要 5V 以上的電子元件，由於 NX 的需求電壓是 19V 並且尚未找到更好的供電方式故暫時需以插頭供電，電池作為供電來源選擇安放在車體第一層底部以便更換電池。

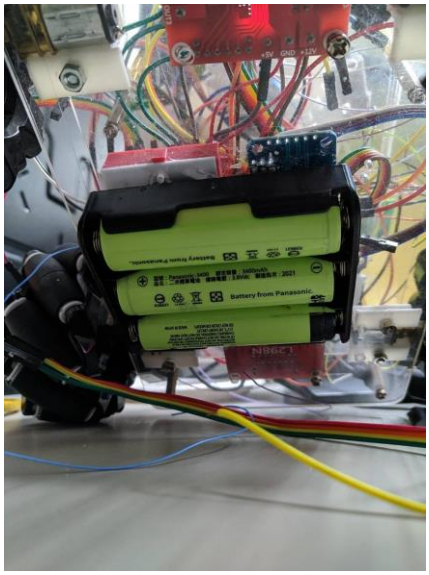


圖 3.33 電池安放位置圖

車體第一層安放兩顆 L298N 馬達驅動板、一個 PCA9685 控制板、四顆馬達與四顆麥克納姆輪一個小型麵包板馬達以 3D 列印部件固定，馬達驅動板由 12V 電池供電信號由 PCA9685 控制，PCA9685 由 12V 電池和 NX 5V 供電。

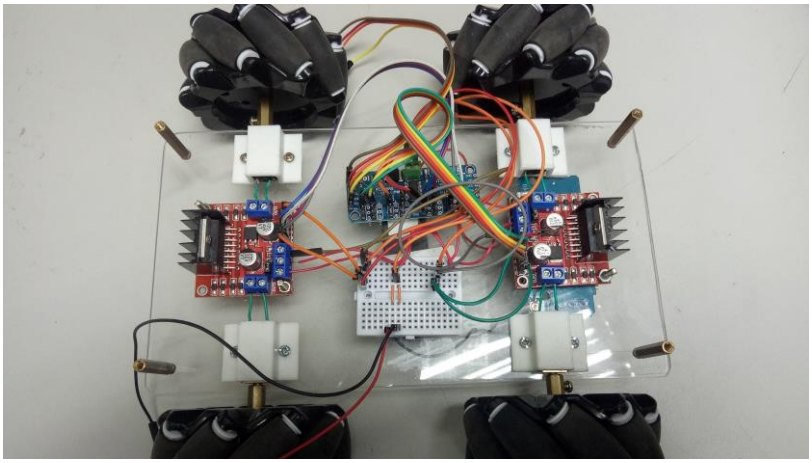


圖 3.34 車體底層

車體第二層為中間層，為線路方便連接故將所有電子元件的控制中心 NX 放置於此，這也造成此層線路繁雜，另外此層還放置兩個繼電器、一個 MCP3008、升壓模板及一個麵包板，兩個繼電器分別為電容充電和電磁鎖通電的開關，MCP3008 插在麵包板上並且透過電阻並聯著電容，12V 與 5V 供電皆連接至麵包板上，電子元件再從麵包板上獲得供電。水平旋轉的步進馬達外要連結到第三層旋轉平台之線路皆透過第二層頂部的電路旋轉連接器連接。為拍照清楚線路先行拆除。

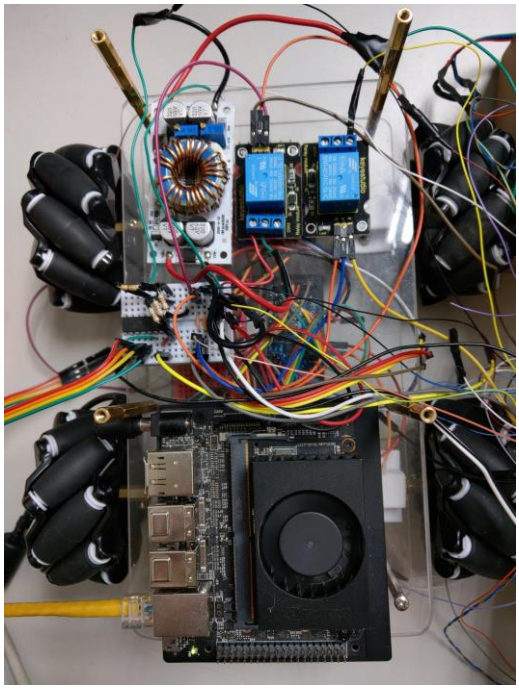


圖 3.35 車體中層圖

車體第三層為旋轉平台放置層，除水平馬達驅動板外之線路皆由電路旋轉連接器連接並透過旋轉平台之中空柱連接至平台的 SCR、電磁鎖及垂直步進馬達，電磁鎖置於砲管支架上方並列印支架補正電磁鎖與彈匣位置的差異，電容、SCR 與二極體置於砲管支架旁的空位上。垂直伺服馬達置於平台上，金屬轉軸連接砲管支架，馬達驅動板至於平台上。水平步進馬達與馬達驅動板則放置於第三層壓克力板上。為拍照方便固為單獨拍攝。

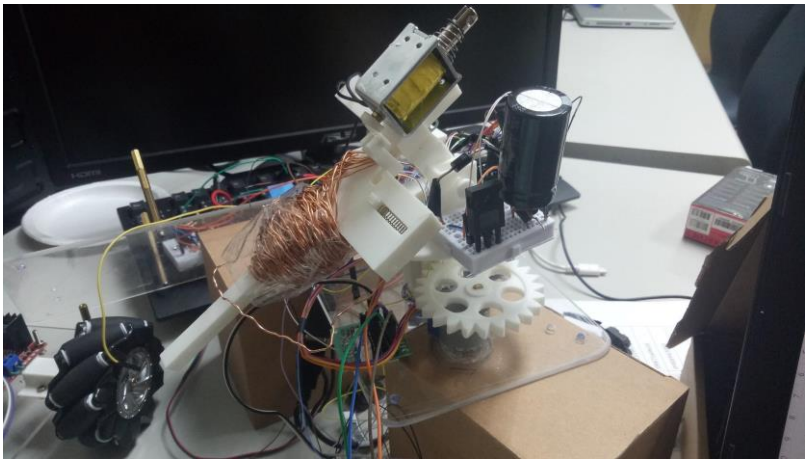


圖 3.36 車體頂層圖

第4章 成果與結論

4.1 研究方法

本專題將以 Nvidia Jetson Xavier NX 作為車體核心，將車體控制、接收電壓數據、API Server 等交由 NX 來進行處理，在車體架構方面，輪胎使用麥克納姆輪，俗稱萬向輪，以達到操控車體移動時的順暢度，車體底層皆放置車體移動馬達等相關元件， 車體各個分層則是利用切割多餘之壓克力版而製成，中層放置 NX、繼電器、升壓模板及測量電壓之 IC 元件，在水平旋轉平台中使用了滑輪，以利在水平旋轉平台旋轉時不會互相卡到線材以致功能失效，在平台旁及平台上與砲管支架的連接處分別放上控制水平旋轉的步進馬達與控制仰角高度的伺服馬達以實現砲管攻擊各個方向之目標物，砲管支架上分別固定砲管、讓子彈上膛的電磁鎖及瞄準所需之鏡頭及測距儀器，電容、SCR 及二極體至於砲管支架旁的麵包板，透過中層的繼電器控制電容充電，在充電時電磁鎖會開始運作將子彈從彈匣中推入至砲管中，並由砲管底部的磁鐵固定，在停止充電後，由 NX 向 SCR 提供發射命令，電容與線圈之間將形成通路，二極體為防止因線圈斷電產生的突波傷害其他電子元件。

4.2 成果

在連上自走砲的 WIFI 基地台後，客戶端得以使用網頁或手機應用程式的串流影像來無線操作自走砲。在移動自走砲方面，能接收來自客戶端的前後左右和向左右旋轉等指令分別做出對應的動作，由 APP 端控制可由搖桿推移程度決定車體移動速度，網頁端則是使用按鍵變更車體移動速度，在地面上車體可以以最自由的方式移動。以步進馬達和伺服馬達分別控制著上方的兩軸旋轉平台、旋轉砲管及鏡頭。在電磁砲擊發之前要先進行電容充能，充電的同時 APP 端與網頁端將會顯示電壓數值，以方便使用者知道充電狀況，充能同時會將砲彈上膛，待充能結束便可進行擊發。

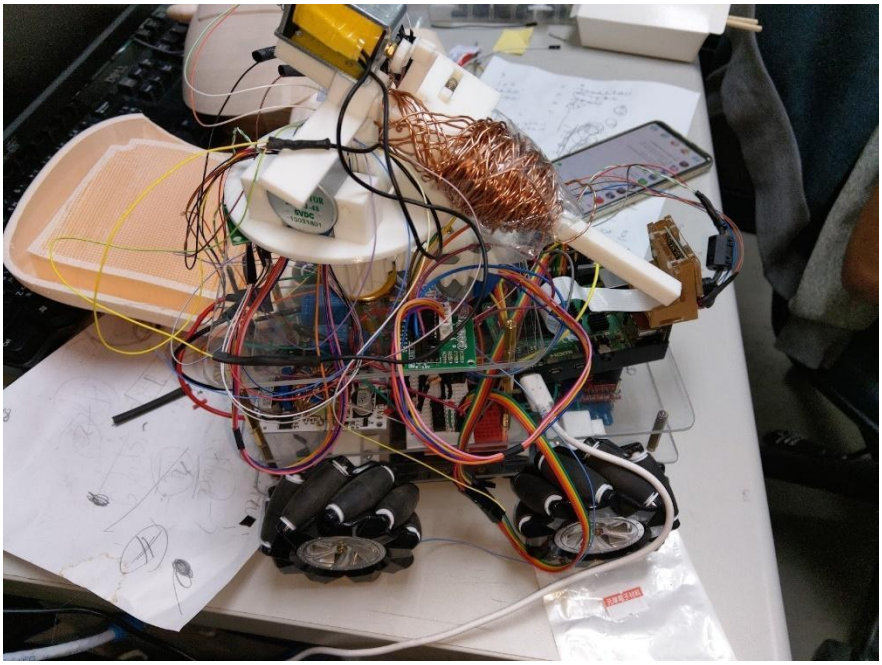


圖 4.1 自走圖砲成體

圖 4.2 實際運作 APP 畫面

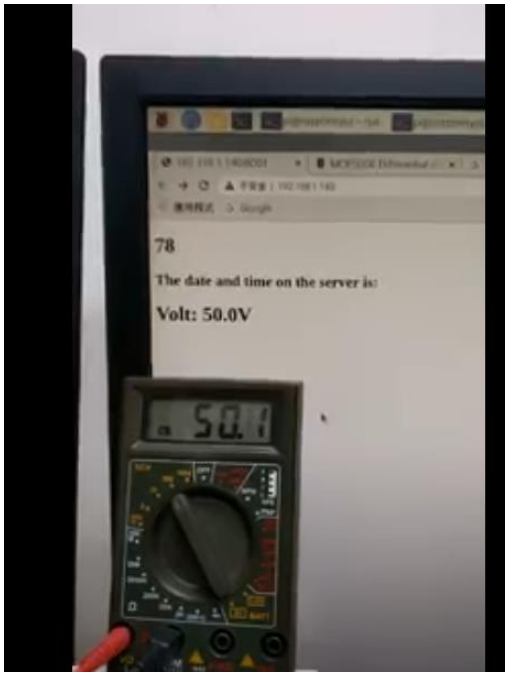


圖 4.3 網頁電壓運作圖

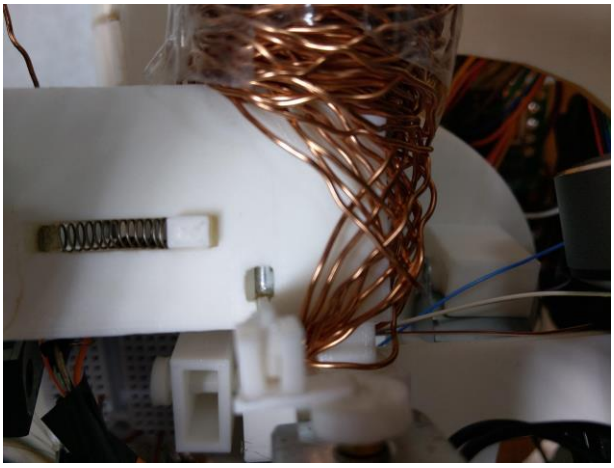


圖 4.4 上膛前

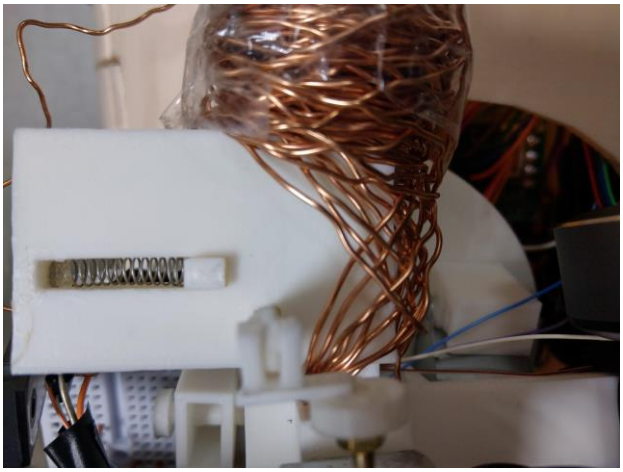


圖 4.5 上膛後

圖 4.6 車體完整圖

圖圖 4.6

4.3 未來規劃

目前本專題的研究與製作過程大抵已完成，並且最初冀望之客戶端以網頁或手機應用程式成功連線操控自走砲也順利實現，希望在未來能有更多充裕的時間最佳化整體系統與電路，甚至將客戶端操作系統，像是手機應用程式，持續研究拓展成一般射擊玩具，投以娛樂、演藝或教育產業，寓教於樂。

第5章 參考文獻

[1] **樹梅派 4B**
https://www.techapple.com/archives/30761
https://www.kodorobot.com/product/%E6%A8%B9%E8%8E%93%E6%B4%BE4%E4%BB%A3-raspberry-pi-48gb/

[2] **NVIDIA JETSON XAVIER NX**
https://www.nvidia.com/zh-tw/autonomous-machines/embedded-systems/jetson-xavier-nx/

[3] **樹梅派專用攝影鏡頭模組（Raspberry PI RPI Camera）**
https://www.taiwaniot.com.tw/product/%e6%a8%b9%e8%8e%93%e6%b4%be%e5%b0%88%e7%94%a8%e6%94%9d%e5%bd%b1%e9%8f%a1%e9%a0%ad%e6%a8%a1%e7%b5%84-raspberry-pi-rpi-camera-500%e8%90%ac%e5%83%8f%e7%b4%a0/

[4] **5V 步進馬達與驅動板**
https://www.orientalmotor.com.tw/image/web_seminar/stkiso/20130307_stkiso_seminar.pdf

[5] **減速馬達**
https://www.shinwe.com.tw/%E6%B8%9B%E9%80%9F%E9%A6%AC%E9%81%94/

[6] **麥克納姆輪**
https://www.itread01.com/content/1549544428.html

[7] L298N 馬達驅動模組
https://zhuanlan.zhihu.com/p/166119517

[8] **50V 升壓模組**
https://www.whzjedu.com/big5/jianzhuye_640794

[9] **繼電器**
https://tutorials.webduino.io/zh-tw/docs/basic/component/relay.html

[10] **矽控整流器（SCR）**
https://zh-tw.lambdageeks.com/silicon-controlled-rectifier-scr/

[11] 數位訊號轉換器（MCP3008）
https://blog.everlearn.tw/%E7%95%B6-python-%E9%81%87%E4%B8%8A-raspberry-pi/%E5%88%A9%E7%94%A8-raspberry-pi-3-model-b-%E8%88%87-mcp3008-%E9%80%B2%E8%A1%8C%E9%A1%9E%E6%AF%94%E6%95%B8%E4%BD%8D%E8%BD%89%E6%8F%9B

[12] 高效整流二極體（HER308GW）
https://resource.iyp.tw/static.iyp.tw/38747/files/06278373-6872-4b8a-a981-b46b893ed9dc.pdf

[13] 舵機驅動板（PCA9685）
https://frank1025.pixnet.net/blog/post/349725877-%5Barduino-013%5D-16%E8%B7%AF12bit-pwm%E6%8E%A7%E5%88%B6%E5%99%A8pca9685

[14] 電磁鎖
https://www.easyatm.com.tw/wiki/%E9%9B%BB%E7%A3%81%E9%8E%96

[15] 3D 列印機（Ping 270 DIY）
https://www.myfeel-tw.com/3d-printer/

[16] Raspberry Pi OS
https://fung2008.pixnet.net/blog/post/49595235

[17] 24 路電路旋轉連接器
https://canaan-elec.com.tw/html/pwork/03.php?num=22&page=1&kind1=8&kind2=23

[18] Python
https://medium.com/python4u/hello-python-509eabe5f5b1

[19] Flask
https://blog.techbridge.cc/2017/06/03/python-web-flask101-tutorial-introduction-and-environment-setup/

[20] HTML
https://developer.mozilla.org/zh-TW/docs/Learn/Getting_started_with_the_web/HTML_basics

[21] **CSS**
https://developer.mozilla.org/zh-TW/docs/Learn/Getting_started_with_the_web/CSS_basics

[22] **JavaScript**
https://noob.tw/javascript-introduction/
https://developer.mozilla.org/zh-TW/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[23] **Dart**
https://dart.cn/overview

[24] **Flutter**
https://ithelp.ithome.com.tw/articles/10215158
https://www.itread01.com/content/1561755662.html
https://ithelp.ithome.com.tw/articles/10237551

[25] **Blender**
https://underground-nest.com/?p=3287

[26] **切片軟體（Ping Slicer）**
https://seanhkchiu.pixnet.net/blog/post/312068478

[27] **Flask 搭配 Raspcam 架設 Server**
https://blog.miguelgrinberg.com/post/video-streaming-with-flask

[28] **PCA9685 Library**
https://github.com/onionys/python_code/blob/master/RasPi/modules/module_i2c/PCA9685.py

[29] **MCP3008 程式參考**
https://stackoverflow.com/questions/20849508/python-putting-two-codepieces-together

[30] **WIFI 基地台建置教學**
https://blog.gtwang.org/iot/setup-raspberry-pi-as-wireless-access-point/2/

[31] **Flutter 搖桿元件程式參考**
https://stackoverflow.com/questions/60430317/how-to-make-a-joystick-in-flutter-where-the-inner-circle-doesnt-leave-the-outer