附件 3

國防大學理工學院電機電子工程學系

無線通訊應用專題實作書面報告

實作題目：

運用影像辨識技術實現無人車的道路循跡

課程名稱：無線通訊應用專題實作(四)

報告學期：111 學年度第 2 學期

指導老師：王學誠、蘇英俊 教授

	學　號	姓　名
小組成員	1120530063	黃鈺詰
	1120530111	楊博翔
	1120530148	陳朋睿
	1120530180	陳一儒

中　華　民　國　一　一　二　年　四　月　十　日

摘要

本研究旨在探討影像辨識技術在無人車應用中的可行性。目前，無人車技術仍面臨許多問題，例如車輛自身安全性、與其他車輛和行人的協調、故障排除等。因此，開發一款智慧型的無人車，以提升傳統無人車的控制方法，具有迫切性與必要性。

本研究使用賽靈思公司所開發的 KV260，一種集成了視覺感知、目標檢測和行為決策等多種功能的硬體平台，其低功耗和高可靠性特點可滿足無人車長時間運行的需求。本研究使用 KV260 進行 CNN 深度學習運算，將鏡頭所拍攝到的照片導入模型中進行分類，產生對應編碼，讓 Arduino 接收訊息後針對不同狀況對馬達進行速度與旋轉控制，實現無人車在道路上的自駕能力。

目錄

圖目錄

1. 研究背景動機與目的

1.1 研究背景

在現代科技的發展下，讓無人車行走於道路上已經不再是一個不可達成的目標。然而，實現無人車的商業化應用仍存在著諸多問題，如技術、法規和安全等方面的挑戰。為了實現無人車在真實環境中的運行，需要開發先進的感知、控制和決策技術，並滿足無人車在實際應用中長時間運行的要求。

1.2 研究動機

要讓車輛在實際道路上行駛，需要考慮到許多問題，例如障礙物、車道、交通號誌等等。為了研究無人車對車道邊緣和車道線的識別能力，以及對其識別結果的應變方式，我們計劃實現一輛能夠循跡的無人車。透過實驗，我們可以了解無人車辨識車道線的能力，以及其應對的反應方式。

1.2.1 背景

為了實現無人車的道路循跡，我們需要面對多重挑戰。除了影像辨識的技術外，我們還需要考慮如何實時讀取影像，以及如何控制無人車的動力。

1.2.2 解決構想

王學誠老師曾經對 Duckietown 進行研究，因此我們選

擇以小鴨車為主要實作對象，以降低結構、電源、電路配置等方面的複雜度，讓我們可以基於此大架構進行研究和改進。[1]

1.3 研究目的

為了實現無人駕駛技術，我們的專題旨在探討無人車對車道的辨識及其相對應的處理。具體而言，我們將使用影像辨識技術，使無人車能夠在道路上實現自主行駛，並在辨識出車道的方向後進行相應的操作。這不僅需要考慮車輛控制方面的技術，還需要充分考慮感知技術、運動控制、路徑規劃等方面的問題，使無人車能夠更加全化地在道路上運行。因此，本專題將對影像辨識技術及其在無人駕駛技術中的應用進行研究，以期提高無人車的高效化水平及行駛安全性。

2. 文獻回顧與探討

2.1 ROS 相關參考文獻論述

ROS，Robot Operating System，是一種機器人操作系統，ROS 能夠以節點的方式自由加入新的節點與刪除不需要的節點以達到模組化，並且對於處理器的要求不高，而且可以使用開源程式碼，省去了從頭開發過程，可謂站在巨人的肩膀上。ROS 是現在作為機器人開發的主要架構。

ROS 是一個用於編寫機器人軟體的靈活框架，它集成了大量的工具、資料庫、協議，提供了類似操作系統所提供的功能，包括硬件抽象描述、底層驅動程序管理、共用功能的執行、程序間的消息傳遞、程序發行包管理，可以極大簡化繁雜多樣的機器人平臺下的複雜任務創建與穩定行爲控制。ROS 使開發者不再需要從零開始，大大降低了研究機器人的開發門檻，我們可以利用 ROS 的基礎框架配合選定的功能包快速實現系統原型，從而將重心集中於核心算法研究上，當然我們也可以單獨選用某些功能包，將其集成在已有的產品中，實現特定功能。ROS 還集合了全世界開發人員的開源程式，就是這項優點帶來了機器人爆炸性發展，在 ROS 出現之前，每種機器人的作業系統跟軟體都不一樣，即使要開發一個有特定功能的機器人，就算曾經有人做過，還是要重新設計機器人的軟硬體。[2]

2.2.1 ROS 開發理念

ROS 開發者在開發 ROS 時，希望 ROS 具備以下的特質：

- 點對點：ROS 系統是由許多 process 組成，這些程序可以以對等的方式在不同的主機上連接。這樣可以避免當計算機連接在不一樣的網路時，中央服務器容易出現問題。
- 多語言：因為每個人對於 Coding Language 都不一樣，因此開發者將 ROS 設計為支持多種語言。目前 ROS 可支持四種

語言：C++、Python、Octave、LISP。

- 模組化設計：因為 ROS 的複雜度高，因此開發者選擇使用大量小工具來執行各種 ROS 組件。
- 獨立和輕量化：大部分現有的機器人軟體，難以分離程式代碼的部分功能，並使用在其他的程式碼中，因此 ROS 使用 CMake 這個輕量的編譯工具，將所有複雜的程式碼分開放在 ROS 函式庫中，更容易提取和重用代碼。
- OPEN-SOURCE：開源的代碼對於開發者來說，更可以促進各種軟件的功能與進步。

2.2.2 ROS 重要觀念

- Node（節點）

節點是 ROS 中執行運算任務的基本單位。ROS 進程與系統內其他進程本質沒有區別，當你打開一個記事本、一個畫圖板的時候，都是開始了一個新進程，ROS 節點也是如此，稍微有點區別的是 ROS 節點之間通訊很方便，不像記事本和畫圖板之間毫無聯繫。

一個系統一般可以拆分為多個模組，每個模組由單獨的節點組成。節點的概念使得 ROS 程序設計更加清晰，也更加利於維護，多個節點在功能上相互配合，從而實現複雜的

機器人控制。

- Master（主機）

主機可以視為 ROS 在運行時主要的數據庫程式，每個節點在初始化的時候，都要和主機登記。而主機會將所有資訊傳遞給各個節點，並且每個節點都能夠存取。

- Publisher & Subscriber（發布器跟訂閱器）

發布器跟訂閱器是節點要向其他節點傳遞資訊的方法。首先必須先向主機登記要發布的主題（Topic），再向 ROS 申請發布。若有其他節點向主機 Subscribe 同樣的 Topic，主機會把訊息傳至該節點中。

Node（Publisher）會發布訊息至 Topic，Topic 則會將其接收到的訊息傳給已經訂閱的 Node(Subscriber)。（如圖 2-2）

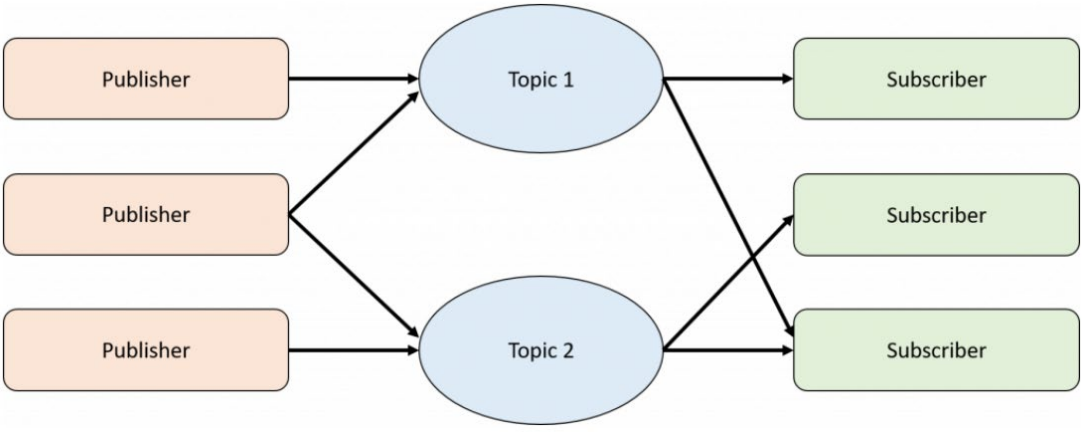

圖 2 - 1 發布器、訂閱器與主題的關係圖

- Topic（主題）

在每個 Publisher & Subscriber，在使用前都要像主機登記名稱，以及主題（Topic）的名稱，用意是讓節點能夠讀取到主題的數據。

- Message（訊息）

  主題所傳遞的資訊就稱為訊息。訊息是由複合的數據結構組成，除了 ROS 內建的訊息類別，使用者也可以用 C++建構自己的訊息。

下圖是將主題、節點、主機構成的關係圖。在 ROS 中，節點與節點之間的關係，可以透過內建的程式繪製成較容易理解的圖片。

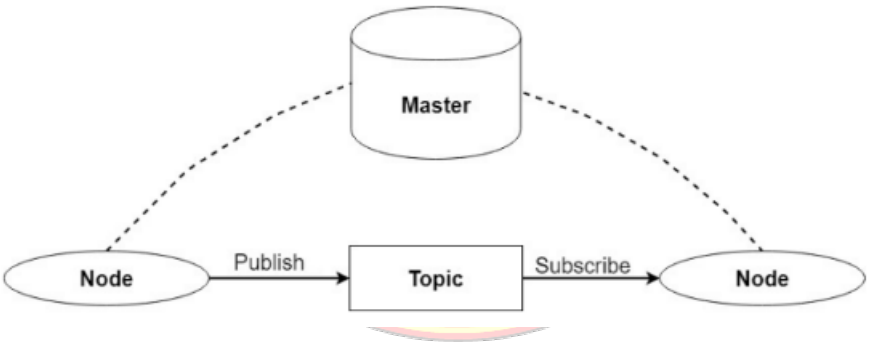

圖 2 - 2 主題、節點、主機關係圖

2.2.4 MAVLink 與 MAVROS

MAVLink（Micro Air Vehicle Link），是目前大部分無人載具與地面站溝通所用的傳輸協定，而 MAVROS 可以將 MAVLink 的傳輸功能作為一個在 ROS 上的節點，使得 ROS 也能夠具備 MAVLink 的通訊功能，因此如果要在 ROS 上與無人載具通訊，MAVROS 是必備的套件。

1. MAVLink：

MAVLink 具有以下的特性

(1) 能夠自訂回傳訊息，因此可以依照使用者去關閉不需要用到的通訊項目。

(2) 回傳資料類型的多樣性，因為無人載具的每筆通訊資料並不是一定及時回傳的，因此會以先抵達的資料來更新現有資料

(3) 支持多個通訊媒介，因此只要傳輸格式符合 Mavlink 的規範，基本上都能夠被接受，像是 serial transmission、WIFI、Bluetooth、TCP/UDP 格式皆可。

(4) 支持多語言，像是 C/C++、Python、Java 等皆可。

2. MAVROS

MAVROS 將 MAVlink 的通訊作為 ROS Topic 的 Publish 及 Subscribe，使用者可以自由地和 MAVROS 發布或訂閱，意思是能夠從 ROS 上接收到來自無人載具的訊息。同樣的，MAVROS 支援 C/C++、Python。[3]

2.2 Kria KV260 Vision AI Starter Kit

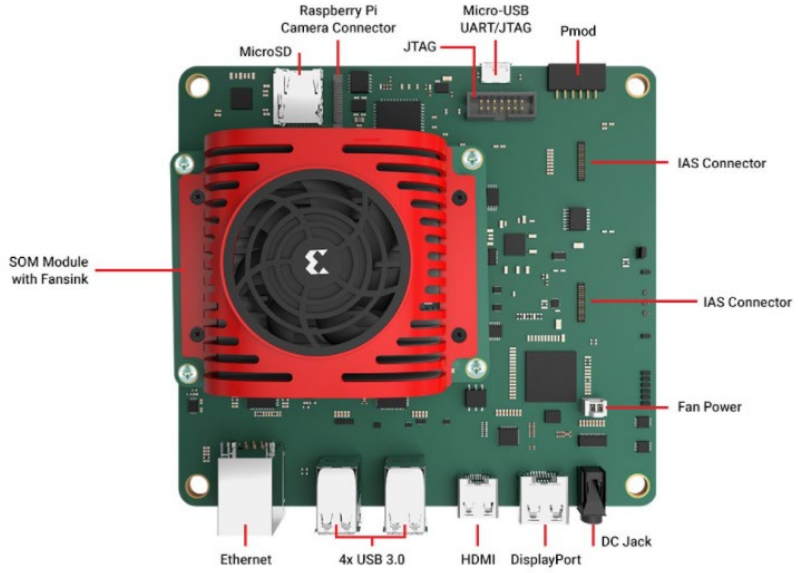

圖 2 - 3 Kria K26

Kria KV260 Vision AI Starter Kit 是由 Xilinx 公司於 2021 年 4 月推出的一款 AI 視覺辨識系統套件，其中的核心元件是被稱為 Kria K26 的 SOM(System on module)。Kria K26 是基於 Xilinx 的 Zynq UltraScale+ MPSoC 架構開發，搭載了 4 核心的 Cortex-A53 ARM 處理器，並且擁有 245 個 I/O 介面，能夠靈活地設定成不同的介面，包括記憶體介面或其他標準介面。其主要用途是進行 AI 視覺應用，是一款高度自動化的系統套件，能夠應用於多個領域，如無人車、安防監控等。

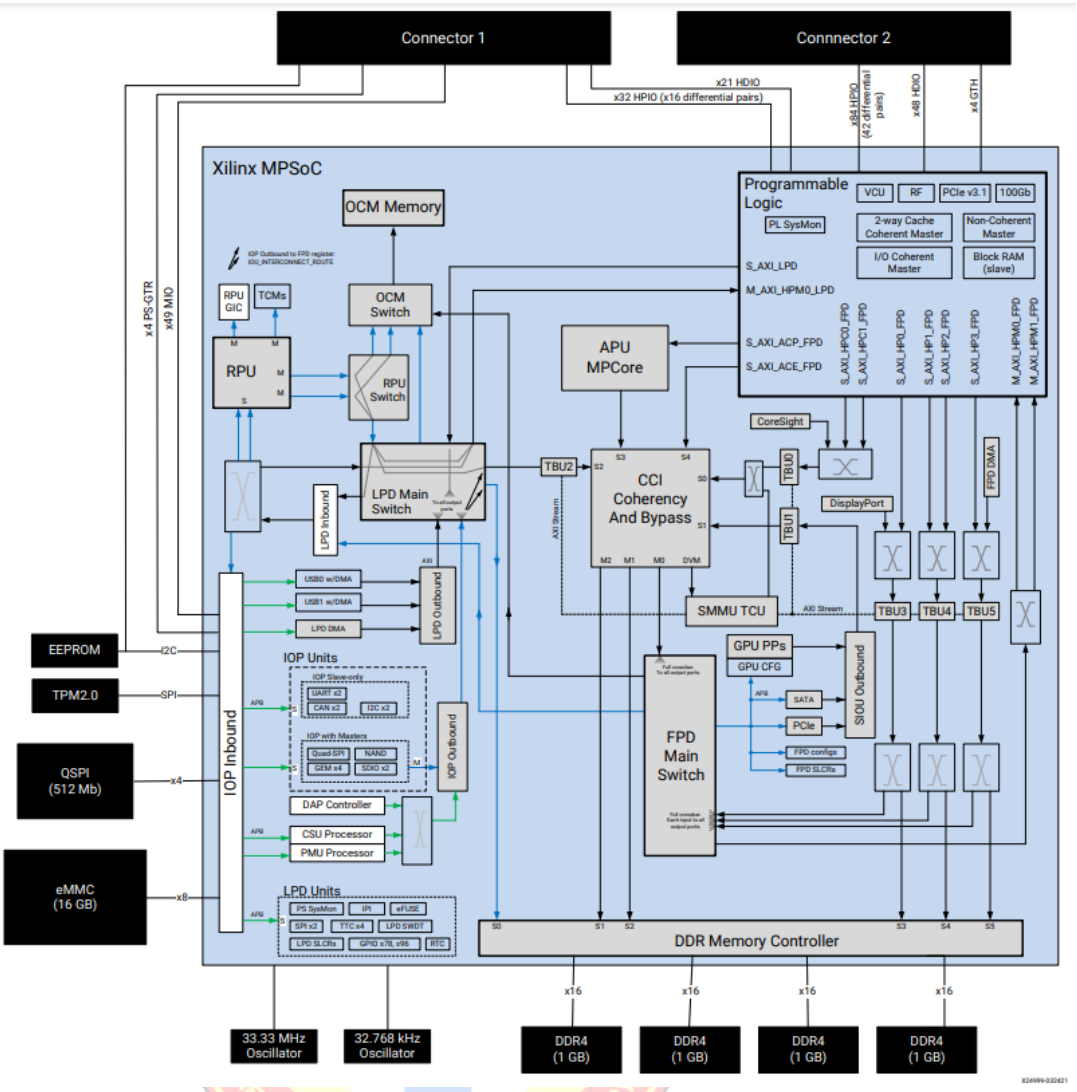

圖 2 - 4 Kria K260 架構

Kria K260 的 AI 運算能力可提供 1.4 倍的 TOPS 效能，這使得它成為一個高效能且延遲性低的選擇，尤其對於視覺導航機器人等 AI 應用具有極大的效益。其高效能的運算能力與低延遲性能夠更好地支援高密度的視覺運算，從而提供更加自動化的解決方案，並讓應用可以更好地利用視覺感知的優勢。這使得 Kria K260 成為一款高效能的 AI 視覺辨識系統套件，能夠應用於各種需要 AI 視覺應用的場景中。

2.3.1 PYNQ

- PYNQ = Python +ZYNQ

ZYNQ 是 Xillin 硬體上的運算單元，同時也是 FPGA 和 ARM 處理器的結合體。而 PYNQ 是一個基於 ZYNQ 的開源平台，提供了 Python 介面以便高階使用者可以更輕易地訪問 FPGA 硬體單元，並且使用 Python 程式語言來編寫 FPGA 的程式碼，大幅簡化了硬體開發的流程。PYNQ 不僅適用於人工智慧和機器學習上的開發，還可用於許多其他領域，如數字信號處理、嵌入式系統等等。因此，PYNQ 平台具有很高的應用價值和廣泛的應用前景。

2.3.2 Overlay

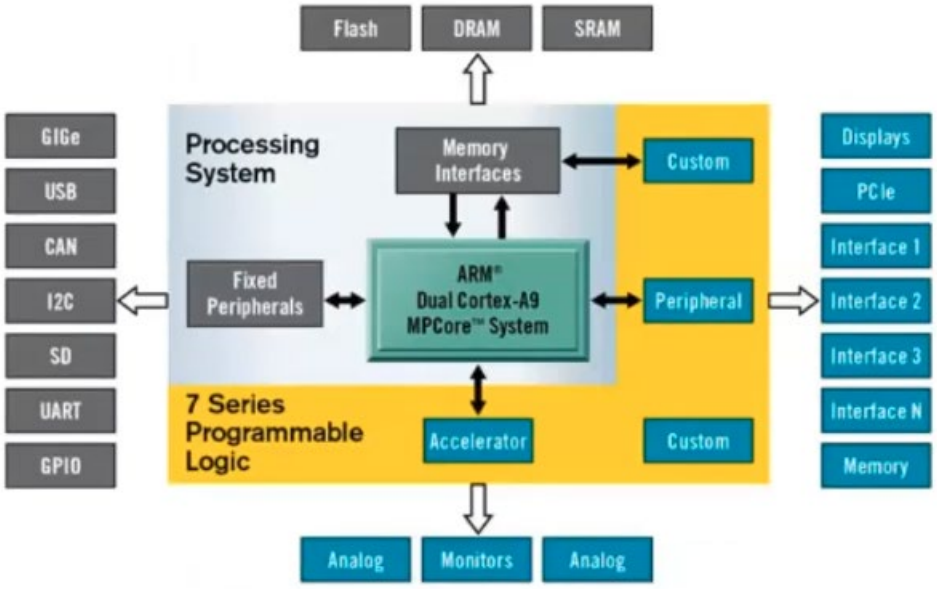

圖 2 - 5 PS PL 關係

當使用者透過 PYNQ 介面存取 ZYNQ 時，可以看到 ZYNQ 架構由兩個部分組成，分別是 PS 和 PL。其中 PS 代表 Processing

System，是 ZYNQ 中的 ARM 處理器，負責系統控制和高階應用執行；而 PL 代表 Programmable Logic，即可程式化邏輯，是 ZYNQ 中的 FPGA 部分，用於高效的硬體加速。PYNQ 提供的 Overlay 是一種硬體設計，可讓使用者將其部署至 FPGA 中，並且可使用 PYNQ Python 介面從 PS 存取 Overlay 中的 PL 部分，以達到更高效的處理能力。

2.3.3 Vitis-AI

Vitis-AI 是一個由 Xilinx 開發的軟體工具，它提供了一個容器化的開發環境，可以用於開發和部署深度學習模型。其中，CPU 版本是基於 CPU 的高效運行，GPU 版本則針對具備 GPU 的裝置進行了改良，可以大幅提高深度學習模型的運行速度和效率。透過 Vitis-AI，開發者可以輕鬆地將深度學習模型部署到 FPGA 上，利用 FPGA 強大的運算能力和低延遲的特點來進行高效的模型推論。由於 FPGA 具有高度的可重構性和可定制性，因此 Vitis-AI 的應用範圍非常廣泛，可用於自動駕駛、智能監控、工業自動化等多個領域。

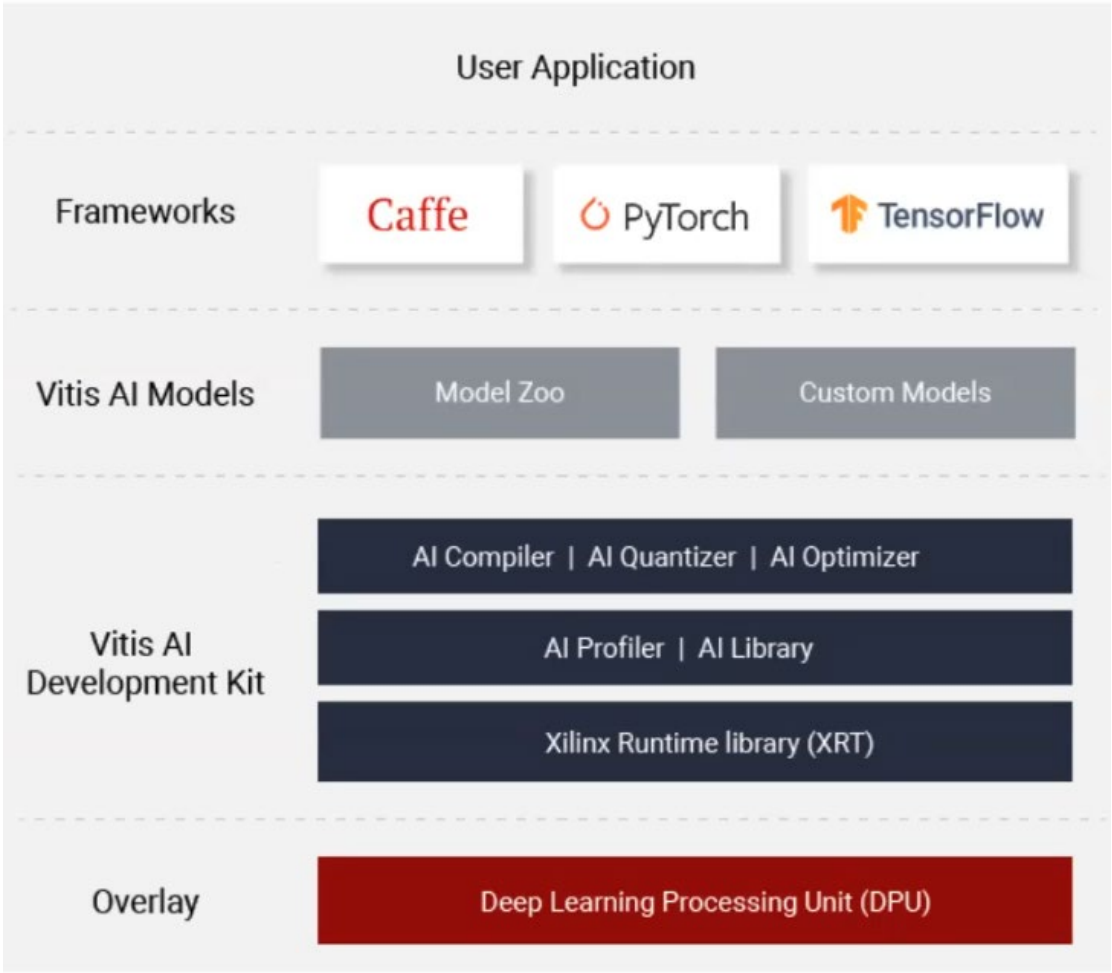

圖 2 - 6 Vitis-AI 架構

上圖為 Vitis-AI 架構。首先，它支援的 Frameworks 有 Caffe、Pytorch、TensorFlow。第二，它提供了 Model Zoo 供使用者使用，或者使用者有自己的 Models 也能夠使用。第三，它提供了一些工具能夠讓使用者使用，其中最常用的工具為：AI Compiler、AI Quantizer、AI Optimizer。最後，使用上述的工具以及 Model 時，可以把它轉成一個 DPU，也就是 Deep Learning Processing Unit，將它放至 Overlay 中，使得 FPGA 可以讀取，實現硬體加速。

Vitis AI - Model Zoo 是 Xilinx 開發的一個模型庫，收集了多個不同深度學習框架的模型，包含了諸如 input、backbone 等的相關資訊，並且提供多個版本的 Optimizer 結果，供使用者選擇最合適的應用，也可以根據自己的需求替換模型中的 Data。

AI Optimizer 可減小模型大小，複雜度可減少 5 倍至 50 倍。此工具會先進行 Pruning，刪除一些權重過低的節點，接著進行 Finetune 以提升模型的正確性。

由於在 FPGA 上執行的模型不同於在 CPU 或 GPU 上的 32 位元浮點數模型，而是採用 8 位元定點模型，因此需要使用 AI Quantizer 進行模型轉換，將模型中的權重和 Activative Function 做 Quantizing，同時可以減少記憶體使用並提升運算速度。

AI Compiler 具有解析器，可將模型的特性移除，並將模型轉換成 XIR-based 的 graph，這是 Xilinx 中介的一種表示型態。接著進行改良，並根據 DPU 的可用性判斷是否將 graph 細分為更多子圖以便在 DPU 中執行。

2.3 本實作可能應用之本系專業課程內涵

在電機系的課程中，可能會應用到的課程有

1. 高等程式設計，在 ROS 系統中，許多的檔案都是使用 C 語言撰寫的，因此要熟悉 C 語言是必要的。

2. 演算法，演算法在影像辨識中扮演著重要的角色，它是訓練模型、提高辨識準確率的關鍵。通過選擇適合的演算法，我們可以讓電腦更快地學習並提高其辨識能力。此外，不同的演算法可以針對不同的辨識任務進行改良，例如目標檢測、圖像分割等。因此，對於影像辨識相關的研究與應用而言，熟悉各種演算法是至關重要的。
3. 單晶片，利用課堂上學到的程式邏輯，並連接各項感測器，透過這些感測器來獲取我們所需的資料，將這些資料傳給我們後端的工作站進行處理分析，來判斷是不是指定的目標。
4. 通訊系統、確保無人機及無人船還有工作站的連線穩定，並測試我們無人機最遠能接收的距離，以確保我們在執行任務的時候不會中途斷線。
5. 人工智慧，在人工智慧的領域中，利用大量數據來訓練電腦辨別目標物已經成為常見的方法之一。透過機器學習演算法的應用，我們可以利用收集到的資料來建立訓練模型，讓電腦能夠更快速、更準確地辨識物體。例如在影像辨識中，透過標註大量的圖片來訓練模型，可以幫助電腦更好地識別圖像中的物體。

3. 研究方法規劃

3.1 ROS 與 KV260

在我們的設計中，主要關注點是如何有效地整合鏡頭、影像處理和馬達控制，以實現我們的目標。因此，我們決定使用 ROS 系統作為傳輸不同組件之間資料的媒介。ROS 是一個開源機器人操作系統，使用節點（node）的方式進行資料交換。在之前的學期中，我們學習到 ROS 的節點功能，可讓不同部位的機器人系統建立節點並進行資料傳輸。透過 Publisher-Subscriber 模式，Publisher 將資料傳送給 Subscriber，後者則進行資料訂閱。

在我們的設計中，鏡頭所拍攝到的即時影像作為 Publisher，而 KV260 作為 Subscriber，進而傳送影像至內部的 DPU（Deep-Learning Processing Unit）進行影像辨識處理。透過 ROS 系統的整合，我們能夠有效地傳送資料並實現我們的目標。[4]

運用 ROS 在無人車的系統架構

3.2 量化處理

進到 KV260 的模型，因為 DPU 無法支援 32bits 浮點數模型，因此我們需要一個工具—AI Quantizer，也就是量化器。這部分我們參

考了 Xilinx 的官方所釋出的 quantizer 架構去做撰寫。並加入了 calib 與 test 模式，以使得模型量化的流程更加完整。

其中，calib 模式是指 CNN 模型經過量化之前，需要先進行校準，這個過程可以生成一個 py 檔案，以作為下一個 test 模式使用。test 模式則是指將校準過的模型丟進去，進行量化，生成一個 xmodel 檔案，這個檔案便可以在 KV260 上進行執行。其中，X 代表 Xilinx，這個名字也代表著這個模型是基於 Xilinx 硬體平台所進行開發和運行的。

```python
def quantize(build_dir, quant_mode, batchsize):
    float_model = build_dir + '/float_model'
    quant_model = build_dir + '/quant_model'
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    print("Running on device: {}".format(device))
    model = CNN_Model().to(device)
    model.load_state_dict(torch.load(os.path.join(float_model, 'lane_following.pth'), map_location=device))
    if (quant_mode=='test'):
        batchsize = 1
    rand_in = torch.randn([batchsize, 3, 480, 640]) # inout size (batchsize, C, H, W)
    quantizer = torch_quantizer(quant_mode, model, (rand_in), output_dir=quant_model)
    quantized_model = quantizer.quant_model

    test_data = datasets.ImageFolder(
        'dataset/Trail_dataset/test_data',
        transform = transforms.Compose([transforms.ToTensor()])
    )
    test_loader = DataLoader(dataset=test_data,
                             batch_size=batchsize,
                             shuffle=True,
                             num_workers=2)
    test(quantized_model, device, test_loader)
    if quant_mode == 'calib':
        quantizer.export_quant_config()
    if quant_mode == 'test':
        quantizer.export_xmodel(deploy_check=False, output_dir=quant_model)
    return
```

圖 3 - 1 量化處理

3.3 CNN Model 與分類

在此次的實作中，我們需要處理大量的照片，並將其分類到不同的類別中。為了達成這個目標，我們選擇使用卷積神經網路(CNN)模型進行影像處理和分類。在建立 CNN 模型時，我們最初使用了兩個卷積層，但經過多次實驗後，我們將卷積層的數量增加到四層，這樣能夠更精準地進行分類。透過不斷的調整和優化，我們的 CNN 模型已經能夠準確地分類出各種不同的圖像。[5]

```
class CNN_Model(nn.Module):
    def __init__(self):
        super(CNN_Model, self).__init__()

        self.conv1 = nn.Sequential(
            nn.Conv2d(
                in_channels=3,
                out_channels=32,
                kernel_size=4,
                stride=1,
                padding=0,
            ),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(
                in_channels=32,
                out_channels=32,
                kernel_size=4,
                stride=1,
                padding=0,
            ),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.conv3 = nn.Sequential(
            nn.Conv2d(
                in_channels=32,
                out_channels=32,
                kernel_size=4,
                stride=1,
                padding=0,
            ),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.conv4 = nn.Sequential(
            nn.Conv2d(
                in_channels=32,
                out_channels=32,
                kernel_size=4,
                stride=1,
                padding=0,
            ),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.fc1 = nn.Linear(31968, 200)
        self.fc2 = nn.Linear(200, 18)
```

圖 3 - 2 CNN Model

卷積層

第一層:

這層是由一個卷積運算和一個池化運算構成，使用 4x4 的 filter 對輸入圖像進行卷積運算，並且為了保留較多圖像的細節，所以我們將 stride 設為 1，讓其每次移動 1。理論上特徵圖的數量越多，提取特徵的能力也就越強，但是考量模型的參數量和計算複雜度。因此，設定第一層輸出特徵圖數量為 32 個，表示有 32 個不同的卷積特徵。

經過卷積運算後，對特徵圖進行 2x2 的池化計算，我們採用最大池化的方式進行採樣，目的在於強化圖像中最首要的特徵，抑制次要的特徵，同時減少特徵圖的大小，從而進一步降低計算量，提高模型的運行速度。

第二到第四層:

這幾層的構成與第一層基本上是一樣的，都是由一個卷積運算和一個池化運算構成，就是為了進一步的特徵提取，使其特徵間的相關性小一點。與第一層不同的是，第一層因為是承接輸入的照片，所以通道是 RGB 的三個通道，而從第二層開始都是承接前一層的輸出 32 個特徵圖。

全連接層

在 CNN 中，卷積層和池化層通常用於提取圖像特徵，而全連接層則用於對特征特徵進行分類。我們在這個模型的最後兩層使用全連接層來分類。

第一層:

用來將第四層卷積層輸出攤開成一維向量，第一層全聯層就是讓特徵圖轉化成向量。

第二層:

最後輸出的 18 就是我們要的分類種類，而這 18 個就是我們後續辨別不同道路情況的依據。

3.4 馬達控制

當我們將照片分類完之後，需要讓 Arduino 知道這個照片是屬於哪一類，並且要產生相對應的馬達控制訊號。因此我們透過 KV260 以及 Arduino 的 UART 來進行溝通。

分類的結果會依照不同的類別產生不同的字串，其中字串設計為 8 碼：

0 or 1	右輪轉速 (百位數)	右輪轉速 (十位數)	右輪轉速 (個位數)	0 or 1	左輪轉速 (百位數)	左輪轉速 (十位數)	左輪轉速 (個位數)
0	1	2	3	4	5	6	7

字串共分為兩部分：右輪控制與左輪控制，分別為第 0~3 及 4~6 字元，每一部分的第一個字元代表著車輪旋轉方向，0 表示逆時針，1 表示順時針；第二至四字元代表著車輪轉速。

產生字串後，透過 UART，TX 為 KV260，RX 為 Arduino，將字串送進 Serial Port 中。而在 Arduino 的程式中，我們設計一個解碼器的程式，來將收到的字串轉換為 L298N 的馬達控制訊號。

字元變成整數的部分，我們採用了 ASCII 碼的方式來去做處理，因為對於 0~9 的 ASCII 碼而言，只要取跟 0 的 ASCII 碼之差值，就能得到原本的數字。而因為我們是先接收到百位數的字元，因此每次取完差值後，我們必須將數字乘上 10，讓其往左移位。反覆操作 2 次後，就會得到我們所需要的三位整數。

舉例而言，當我們收到 12551150 時，首先先將最前面的 1 儲存當作右輪的順時針旋轉信號。接著我們透過接收到的第一個字元 2 與 0 的 ASCII 碼做相減，便可得到 2 這個值。

'2'-'0'=2

再將此數字儲存至一變數中。

temp=2

接著，我們收到了 5 這個字元，一樣先與 0 的 ASCII 碼做相減，完成之後必須先加上前一組數字乘以 10 的數字，再次儲存至變數中。乘以 10 的動作就像是將原本在個位數的數字往前挪動一格，並空出

個位數儲存下一個數字。

		2
	2	5

temp = temp*10 + '5'-'0'

同理，第三輪也必須先將前一組數字乘以 10，再加上新進數字與 0 的 ASCII 碼做相減的值，便可得到 255 這個整數。

獲得了旋轉方向以及旋轉速度後，我們便可以將此訊號匯入 L298N 中，讓其控制馬達進行運轉，從而實現結果。

4. 實作成果說明與分析

4.1 硬體架構

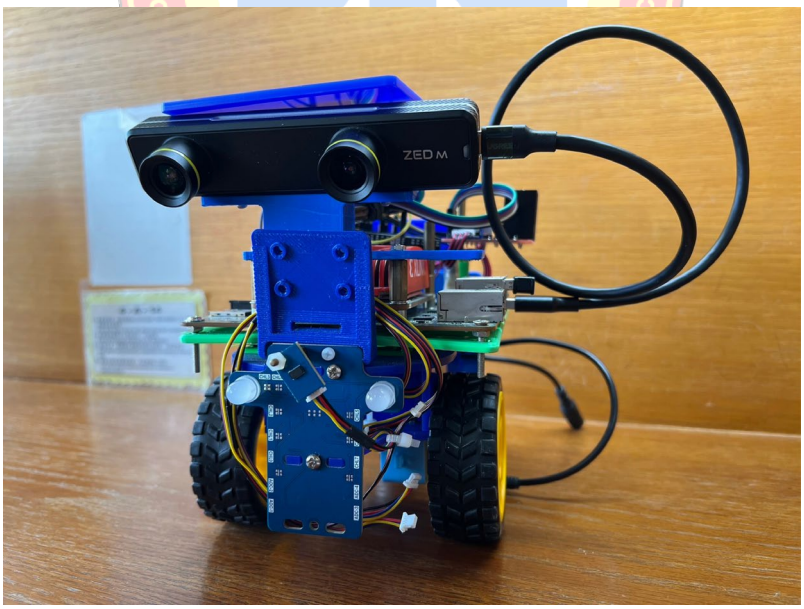

圖 4 - 1 前視圖

在我們的硬體架構中，我們特別擺放了鏡頭，以便能夠即時捕捉道路狀況，並傳輸給影像處理系統。

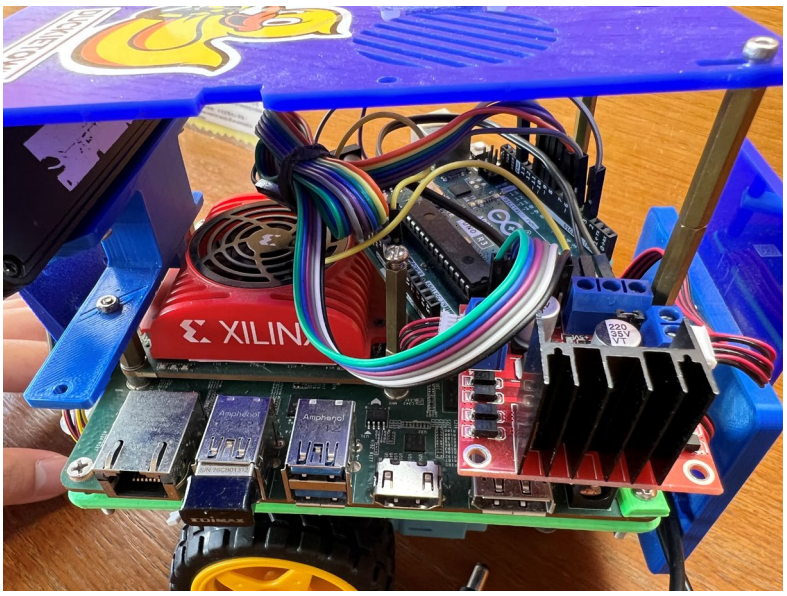

圖 4 - 2 內部圖

在硬體架構的本體中，我們主要搭載了 KV260，並將 Arduino 與 L298N 架高，以減少橫向空間擴展過大的問題。這樣的設計能夠使轉彎時更加順利，同時也有助於提高整體系統的穩定性和可靠性。

4.2 運作流程分析

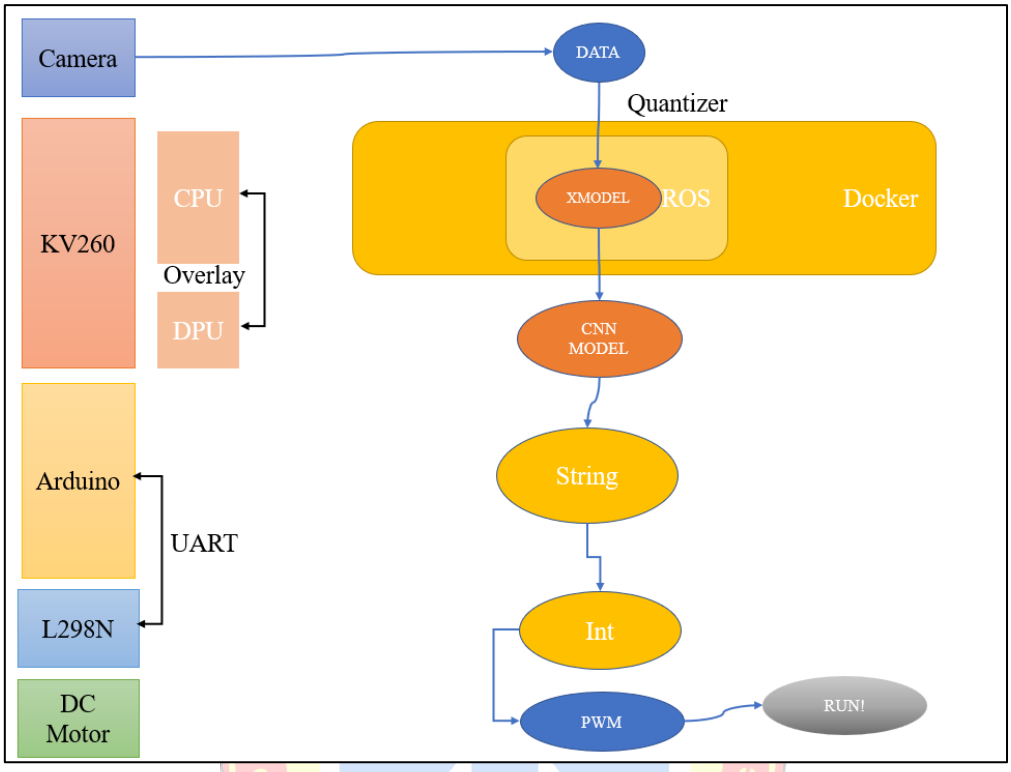

圖 4 - 3 運作流程概念圖

我們在 KV260 上，裝載了一個 Docker container，其中裡面包含到了 ROS 系統，負責將相機所拍攝到的即時畫面，透過 ROS 的訂閱機制，我們可以接收到相機所拍攝到的即時畫面，進行影像處理和分析。這樣的架構不僅可以提高系統的穩定性和可靠性，還可以方便地進行模組化設計和功能擴展。[6]

收到照片之後，我們會先將模型進行量化處理，因為在 DPU 上無法處理浮點數模型。處理好的模型會有一個 xmodel 檔案，此檔案會透過 Overlay 來傳送至 DPU 上。

接著，透過 DPU 上來進行 CNN Model 的預測，將圖片進行分類。在知道屬於哪一個類別之後，會根據其所需要的轉彎程度，產生字串。

這個字串共有 8 碼，其中，1、5 碼代表右、左輪的旋轉方向，0 代表反轉，1 代表正轉。2~4 及 6~8 碼代表右、左輪的旋轉速度，大小從 0 至 255。產生一個 8 碼的字串後，會透過 UART 的方式傳送至 Arduino 上。

在 Arduino 上接收到字串後，會進行解碼的動作，將旋轉方向以及旋轉速度帶到 L298N 上，進而給馬達控制訊號，透過控制 L298N 的輸入訊號，可以控制車輛的運動讓車體運行。

4.3 HIL 測試結果

為了能夠更加精確地模擬小鴨車的運作，我們在 Gazebo 中建立了一個具有高度真實感的場景，透過 Gazebo 的模擬，我們可以測試和改善小鴨車的運行策略和控制算法，以確保其在現實環境中的表現能夠達到最佳狀態。[7]

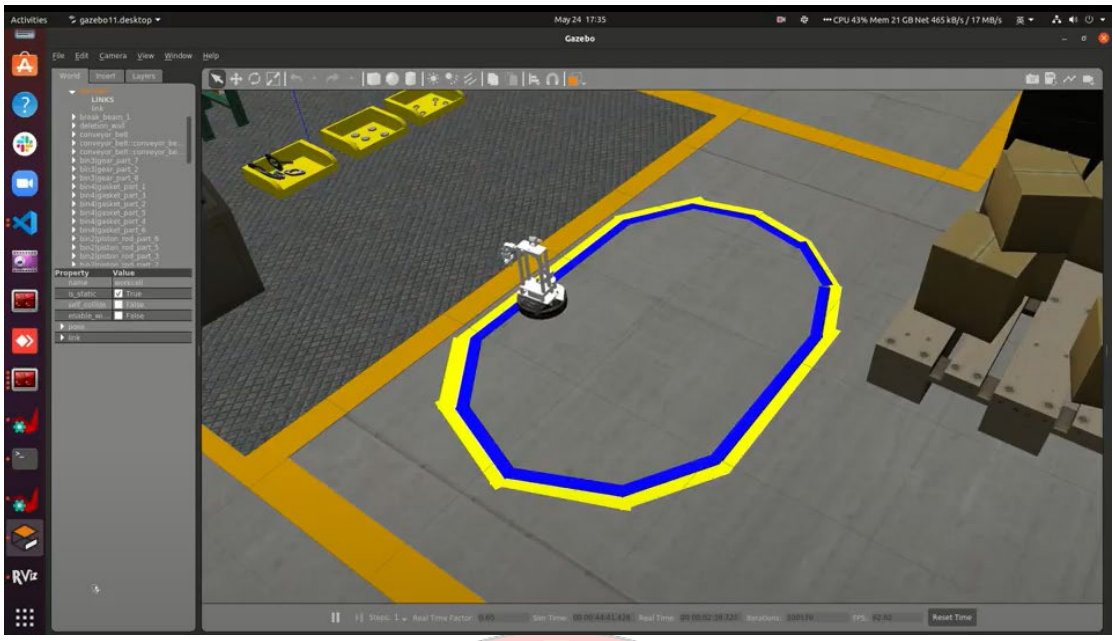

圖 4 - 4 模擬結果

4.4 成果展示

圖 4 - 5 成果展示-1

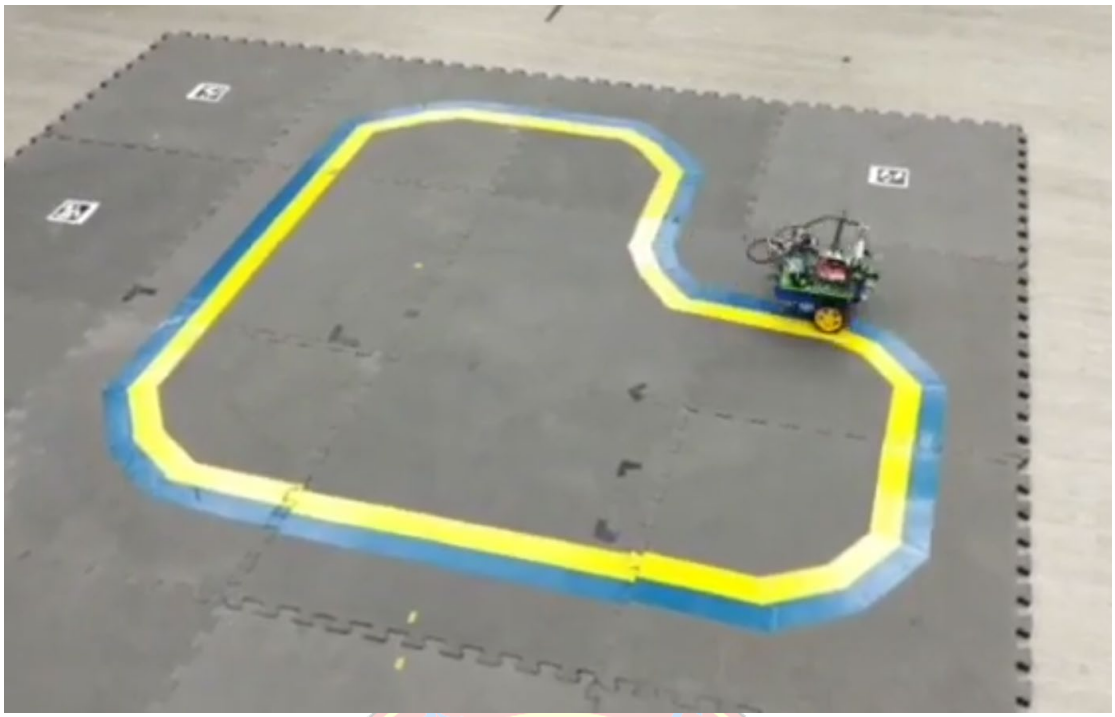

圖 4 - 6 成果展示-2

圖 4 - 7 成果展示-3

我們用了黃藍膠帶當作道路，讓小鴨車在上面運行。而小鴨車也很順利的進行了一圈道路辨識。

5. 結論與未來展望

我們成功完成了無人車循跡的部分。這個實作涉及到許多不同的技術和知識，其中大部分並非在課堂上學習到的。我們必須不斷地學習新知識，瀏覽各種資料和文獻，並與研究生學長討論實作的可行性，逐步改進才能達到今天的成果。雖然這個結果可能不是最完美的，但我們從中學到了很多。

然而，道路循跡只是無人車實際上路功能中的冰山一角。為了讓無人車真正上路，還需要涉及到多個層面。除了結合其他技術和專業知識外，還必須考慮倫理道德和法律法規等因素。此外，實際的環境變數非常複雜，為了應對不同的環境，我們需要制定更多的對策，例如如何應對行人或其他車輛等問題。這些都是需要考慮進去的。

未來，隨著技術的不斷進步，無人車將會在各個領域發揮更加重要的作用。我們希望透過這次的實作經驗，能夠為未來的無人車技術和應用做出貢獻。我們也期待在未來能夠繼續學習和改進，以實現更加完善的無人車系統。

6. 實作心得

楊博翔： 這學期最困難的地方在於要實現所有的目標，一開始遇到非常多問題，特別是在硬體設備方面，由於是實體狀態，所以會遇

到很多硬體設備上的問題，再經過老師以及學長的協助後，我們也成功地讓小鴨車運行。回顧這四個學期以來，我覺得這個專題給我的收穫很大，不僅讓我知道團隊合作的重要性，也讓我對於人工智慧方面的知識大開眼界。

陳一儒：本組專題最大的特點之一莫過於我們除了學校老師的協助以外，還有每週到陽明交大，參與王副教授的團隊，向實驗室中不同學長學習各類知識，每學期由教授給予進度指示，再透過請教學長完成每次須完成的進度，讓我們每一學期都能有不同的收穫與成果。進入最後一學期，須將這兩年的所有成果整合，並將我們規劃的種種一切付諸實現，這使得我們這學期的討論與協調更加複雜，從軟體的應用、編寫，到硬體的使用、測試，乃至於實際應用面臨到的技術、法規等，我們必須要將自己的視野放更加宏觀，全面檢視運行結果後，再針對細部利用我們過去兩年的經驗進行微調，過程很不容易，幸虧有實驗室的學長一步步指引我們，我們才能為專題畫下完整的句號。從我們這一屆開始，專題有了陽明交大帶來更多不同的可能性，除了讓我們能得到十分豐沛的學習資源外，他們傳授給我們的經驗更加珍貴，因此我們格外感謝這樣的機會，也很珍惜在這環境下完成的專題實作成果。

陳朋睿：這一次的專題，可謂是最忙亂的一次，過去我們總會有一個明確的進度目標，但這一次我們要將過去的總總進行結合，而不同硬體間的溝通就讓我們苦惱了一陣子，後續要讓車體動起來，也煞費苦心花了許多的時間與嘗試，這一次最大的心得是，用講用看的都很簡單，但真的要去弄要把軟體硬體都兜起來時，每個地方都是 bug 及 error，這一次的專題讓我對於一個機器架構有了很深刻的理解，也想特別感謝交大的學長很樂意回答我們的問題，Chat-GPT 雖然回答的答案不可靠，但對於拓荒一個領域來說，起到很好的引導效果，可以很有效的幫助我釐清我真正的問題及理解整體概念。

黃鈺詰： 我們完成無人車的循跡的部分，整個實作牽扯到許多不同的技術及知識，而且大部分都並非課內所學過的知識，我們要一直去學不同的知識，翻閱及瀏覽各種資料及文獻，並且與研究生學長討論實作的可能性，步步修正才能達到今天的成果，或許這個結果並不是最完美的，但是我們透過這個學習到了很多，道路循跡只是無人車實際上路功能中的冰山一角，要真的能夠上路，牽涉到的層面非常廣，不只要結合其他技術及專業知識，同時也要考量倫理道德及法規，而且實際的環境變數肯定更多，為了因應不同環境，就需要研擬更多對策，像是如果遇到人的時候怎麼辦或是遇到其他台車的時候該如何應對等等，這些都是需要考量進去的。

參考文獻

[1] duckietown
https://www.duckietown.org/

[2] Comparison between ROS and ROS2
https：//www.researchgate.net/figure/Comparison-between-ROS-and-ROS2_fig4_335382592

[3] 劉閔威，「空拍影像接合與辨認之研究」，碩士論文，中州科技大學智慧自動化工程學系，彰化，台灣，2020
https：//hdl.handle.net/11296/9f4p4s

[4] Vitis-AI
https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html

[5] Ahmed El-Yamany，「A Generic Approach CNN-Based Camera Identification for Manipulated Images」，Conference，Department of Electrical Engineering，University of Alexandria，Shenzhen, China，2018

[6] docker docs
https://docs.docker.com/

[7] Liam Paull，「Duckietown: An open, inexpensive and flexible platform for autonomy education and research」，論文，TTIC, ETH, MIT, NCTU，Singapore，2017
https://ieeexplore.ieee.org/document/7989179