Final-term Project

自動跟車系統

第四組

資工三 110590028 黃冠鈞

資工三 110590034 楊榮鈞

資工三 110590035 張庭瑋

資工三 110590048 陳俊諺

一、 簡介

自動跟車系統是一項涉及硬體和軟體整合的複雜工程，旨在實現車輛的自動駕駛能力。本文介紹了基於 Arduino 和 Raspberry Pi 的自動跟車系統，包括其硬體設計、軟體架構、功能展示以及未來發展的可能性。

二、 硬體設計

Arduino 開發板是本專案的關鍵硬體基礎。我們選用 Arduino Uno 45 作為主控單元，它擁有大量的輸入輸出腳位、足夠的計算能力，能夠驅動全車各項感測器和執行器。除此之外，我們也整合了各類擴展模組如馬達驅動、超聲波 等，以實現自動駕駛所需的各項功能。

整體硬體架構採用模組化設計，各個部件之間以標準接口相連，便於維護和升級。所有零件均選用堅固可靠的工業級元件能夠在艱苦的環境中長期穩定運行。

主要硬體組件包括：

1. **HC-SR04 超音波傳感器**：超聲波測距模組在自動車中的應用非常廣泛，主要用於感知環境和輔助導航。超聲波測距模組可用於檢測車輛前方的障礙物。當自動車接近障礙物時，超聲波模組發出超聲波脈衝並測量反射回來的時間，計算出與障礙物的距離。

2. **SG90 微型伺服馬達**：用於車輛的方向控制。SG90 是一款微型伺服馬達，適合於有限空間內的應用，並能提供足夠的轉矩來控制方向盤的轉動。

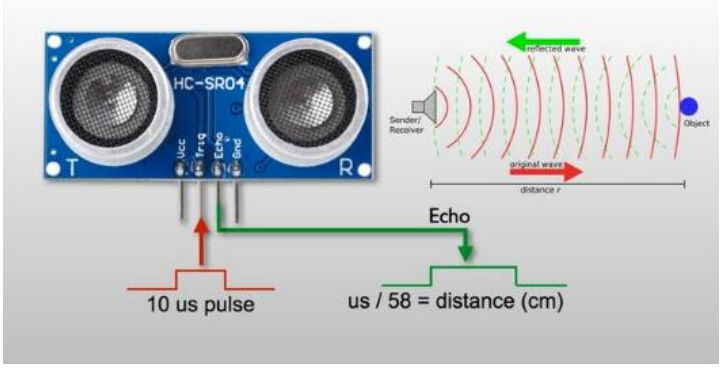
尺寸通常為 22.2mm x 11.8mm x 31mm，非常適合安裝在空間有限的設備中。 重量約為 9 克，對於需要輕量化設計的應用非常理想。 工作電壓範圍為 4.8V 至 6V，常用

5V。 在 4.8V 下，扭力約為 1.8kg/cm，在 6V 下，扭力可達 2.2kg/cm 通常轉動角度為 0 至 180 度，但也可以通過編程控制其轉動範圍。

這些硬體元件之間通過標準化接口進行連接，確保了整體系統的可維護性和可擴展性。

三、 感測器原理

感測器會向前方發送超音波，當超音波碰撞到物體時會反射，這時感測器收到反射回來的超音波進行計算後，就能得知感測器和物體的距離，如下圖所示。



四、 軟體架構

自動跟車系統的軟體部分主要由 Arduino 和 Raspberry Pi 協同運作。Arduino 負責傳感器數據的採集和初步處理，並根據數據生成控制信號。這些信號被傳送到 Raspberry Pi，Raspberry Pi 則負責進一步的數據處理和動作執行。

主要軟體模塊包括：

1. **傳感器數據處理**：Arduino 接收來自 HC-SR04 傳感器的數據，計算障礙物距離，並生成相應的控制信號。

2. **控制信號傳輸**：Arduino 將生成的控制信號傳送給 Raspberry Pi。

3. **動作執行**：Raspberry Pi 接收控制信號，通過 GPIO 引腳輸出 PWM 信號來控制 SG90 伺服馬達的轉動角度，從而實現車輛的方向控制。

五、 自動偵測流程

1. **由超音波接收距離資料**

- 超聲波感測器（如 HC-SR04）通過觸發（Trig）引腳發出一個短暫的超聲波脈衝。
- 超聲波脈衝遇到障礙物後反射回來，並被回波（Echo）引腳接收。
- Arduino 接收到回波引腳的信號，計算超聲波從發射到接收所經過的時間。

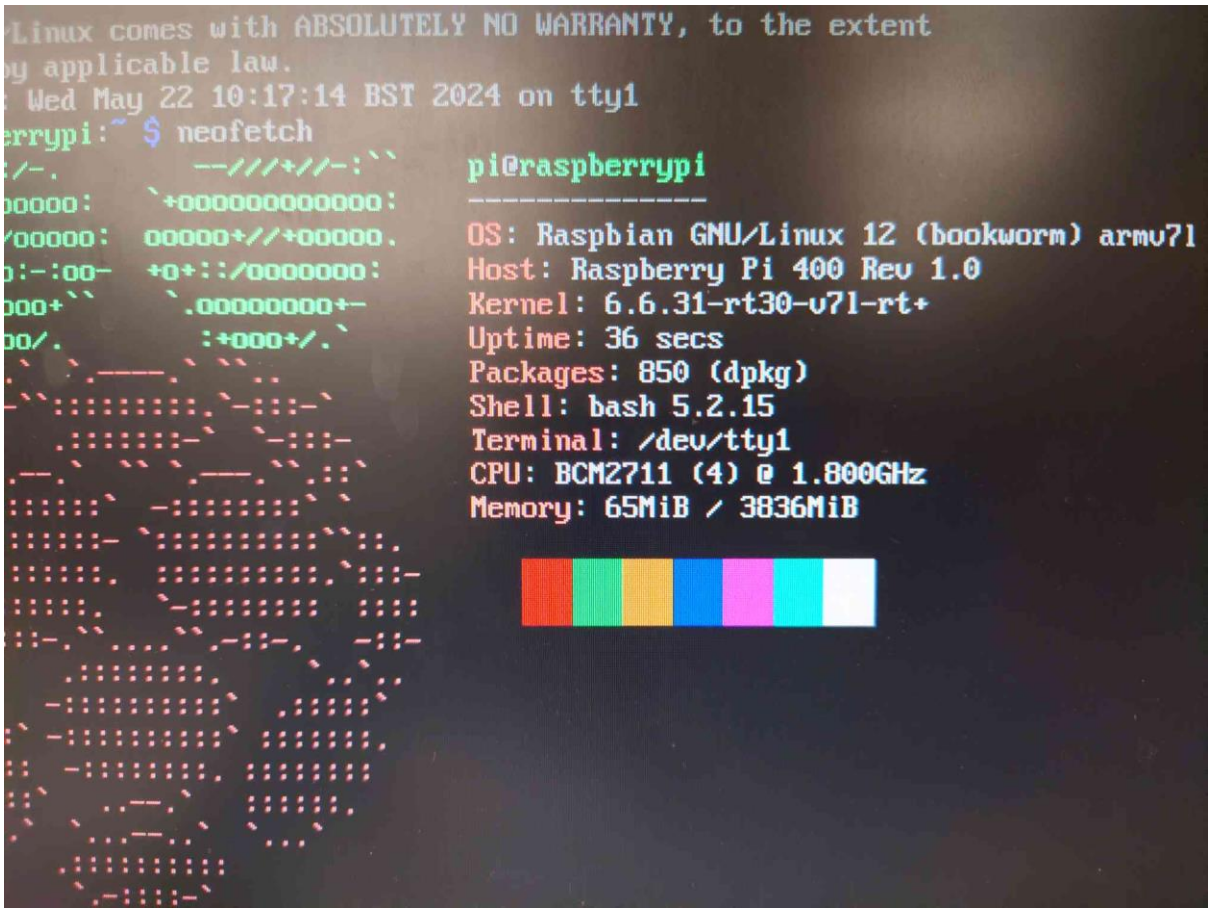
2. **數據處理與邏輯判斷**

- Arduino 根據計算出的距離數據進行判斷，例如設定不同的距離範圍來執行不同的動作。
- 根據邏輯判斷的結果，Arduino 生成適當的控制信號。

3. **動作執行與反饋**

- Arduino 將控制信號傳送至 Raspberry Pi。
- Raspberry Pi 接收 Arduino 的信號，通過 GPIO 引腳輸出 PWM 信號來控制 SG90 伺服馬達。
- 根據接收到的 PWM 信號，SG90 伺服馬達轉動至相應的角度。

六、 Raspberry Pi 400 RT kernel



七、 Shell Script

用於架設 RT kernel 的腳本

```
#!/bin/zsh

yay -S base-devel arm-linux-gnueabihf-gcc

git clone --depth=1 https://github.com/raspberrypi/linux


cd linux


wget https://mirrors.edge.kernel.org/pub/linux/kernel/v6.x/patch-6.6.xz
xzcat ./patch-6.6.xz | patch -p1


KERNEL=kernel7l
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2711_defconfig

menuconfig

mkdir mnt
mkdir mnt/fat32
mkdir mnt/ext4
sudo mount /dev/sdb1 mnt/fat32
sudo mount /dev/sdb2 mnt/ext4

sudo env PATH=$PATH make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- INSTALL_MOD_PATH
=mnt/ext4 modules_install
```

```
sudo cp mnt/fat32/$KERNEL.img mnt/fat32/$KERNEL-backup.img
sudo cp arch/arm/boot/zImage mnt/fat32/$KERNEL.img
# Choose one of the following based on the kernel version
sudo cp arch/arm/boot/dts/broadcom/*.dtb mnt/fat32/
sudo cp arch/arm/boot/dts/overlays/*.dtb* mnt/fat32/overlays/
sudo cp arch/arm/boot/dts/overlays/README mnt/fat32/overlays/
sudo umount mnt/fat32
sudo umount mnt/ext4


KERNEL=kernel7l
sudo cp /mnt/fat32/$KERNEL.img /mnt/fat32/$KERNEL-backup.img
sudo cp arch/arm/boot/zImage /mnt/fat32/$KERNEL.img
# Choose one of the following based on the kernel version
  # For kernels up to 6.4:
#   sudo cp arch/arm/boot/dts/*.dtb mnt/fat32/
  # For kernel 6.5 and above:
  sudo cp arch/arm/boot/dts/broadcom/*.dtb /mnt/fat32/
sudo cp arch/arm/boot/dts/overlays/*.dtb* /mnt/fat32/overlays/
sudo cp arch/arm/boot/dts/overlays/README /mnt/fat32/overlays/
sudo umount /mnt/fat32
sudo umount /mnt/ext4
```

八、Source Code

```
#include <Servo.h> //Build in function
#include <Ultrasonic.h>

Servo myservo;  // 建立SERVO物件
Ultrasonic ultrasonic(11, 12);
int distance;

void setup() {
  Serial.begin(9600);
  myservo.attach(8);  // 設定要將伺服馬達接到哪一個PIN腳
}

void motor_up(int angle){
  myservo.write(0);  //旋轉到0度，就是一般所說的歸零
  delay(1000);
  myservo.write(angle);
}

void loop() {
  distance = ultrasonic.read(); //不加參數就是輸出CM，可用read(INC)輸出英寸
  Serial.print("Distance in CM: ");
  Serial.println(distance);
  delay(500); //每次間格0.5秒
  if(distance < 16){
    motor_up(90);
    delay(1000);
  }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"a.ino" 28L, 659B                                              18,0-1        All
```

```python
import serial
import sys
import threading

COM_PORT = 'COM5'
BAUD_RATES = 9600
ser = serial.Serial(COM_PORT, BAUD_RATES)

def read_from_port(ser):
    while True:
        if ser.in_waiting:
            mcu_feedback = ser.readline().decode().strip()

            if mcu_feedback.startswith("Distance in CM: "):
                try:
                    distance = int(mcu_feedback[16:])
                    if distance > 15:
                        print('No yusha detect ... ')
                    else:
                        print('Too close, Servo speedup, it is time to go to isekai!!!')
                except ValueError:
                    print('Invalid distance value received:', mcu_feedback)

            print('Arduino reaction:', mcu_feedback)
            if mcu_feedback.lower() == 'e':
                ser.close()
                print('bye!')
                sys.exit()

def listen_for_exit():
    while True:
        end = input('press e to exit').lower()
        if end == 'e':
            ser.close()
            print('bye!')
            sys.exit()

try:
    thread = threading.Thread(target=read_from_port, args=(ser,))
    thread.daemon = True
    thread.start()
    listen_for_exit()
except KeyboardInterrupt:
    ser.close()
    print('bye!')
"b.py" 45L, 1269B                                                    29,0-1        All
```

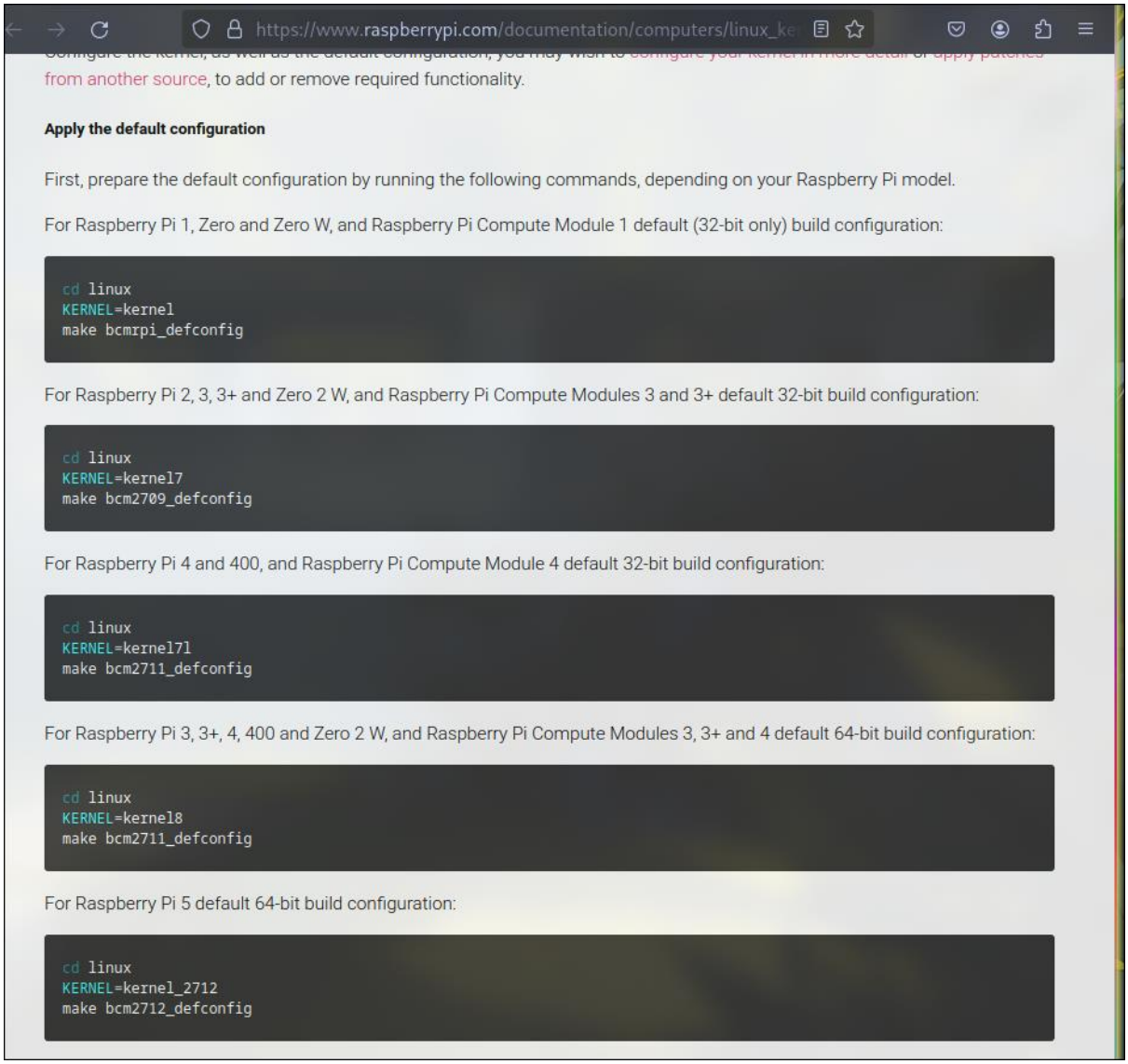
九、 DEMO 展示

https://youtu.be/K3sg3x7Sc48?si=B5fBvf0u1ZVtLvP1

十、 遇到問題

環境架設問題

我們參考的文件內容中，其提供的 kernel 的 value 是錯誤的，導致我們架設 RT kernel 時出現問題，如下圖所示。



硬體問題

1. 靈敏度感應差

2. 會檢測到異常數據

3. 延遲響應

十一、 環境問題解決方式

為了解決 RT kernel 的問題，我們 overwrite 了啟動時要用的 image 到可用的版本，讓 RT kernel 能夠運行。

```
File: temp.diff
+kernel=/boot/firmware/kernel7l.img
```

十二、 硬體問題改進

為了提升自動跟車系統的性能，我們進行了多方面的改進：

1. **改善感測器靈敏度和增強信號接收天線**：通過增強信號接收天線，可以提高感測器的靈敏度，使其能夠更準確地檢測環境中的物體和障礙物。

2. **定期校準**：感測器需要定期進行校準，以確保其測量數據的準確性和可靠性。定期校準可以避免感測器因環境變化或長時間使用而產生的誤差。

3. **異常數據檢測**：系統需要具備異常數據檢測功能，能夠及時發現和處理感測器輸出的異常數據，確保系統運行的穩定性和安全性。

4. **使用穩壓電源**：使用穩壓電源可以確保感測器和系統其他組件穩定工作，避免電壓波動對系統性能的影響。

5. **延遲和響應速度問題**：系統需要優化感測器的延遲和響應速度，確保其能夠快速、準確地響應環境變化，提供及時的數據輸出以進行相應的控制決策。

十三、 未來發展

自動跟車系統的未來發展方向廣泛，包括但不限於以下幾個方面：

應用領域：

1. **軍事用途**：自動跟車系統可以用於軍事任務中的無人車輛自主導航和偵察，提升作戰效率和人員安全。

2. **警匪對峙**：在警匪對峙中，自動跟車系統可以用來偵察和追蹤嫌疑人，減少人員危險，提高執法效率。

3. **災難救援**：自動跟車系統可以在災難救援中發揮作用，進入危險或難以接近的區域進行搜索和救援，提供必要的救援工具和物資。

4. **探索未開發區域**：該系統能夠用於探索人類尚未涉足的區域，如深海或其他極端環境，進行科學研究和資源探測。

5. **提供轉生異世界大卡車禮包**：像是輕小說常常看到的異世界轉生大卡車，雖然這聽起來像是創意性和幻想性的應用，但它展示了自動跟車系統在虛擬或遊戲世界中的潛在應用，提供新的娛樂方式和用戶體驗。

額外功能：

1. **紅外線傳感器輔助**：添加紅外線傳感器來輔助超聲波偵測，提高偵測的準確性和可靠性。

2. **AI 影像識別**：結合 AI 影像識別技術，使系統能夠更精確地識別和分析環境中的物體。

3. **救援工具**：系統可以搭載各種救援工具，如醫療用品、食物和水，提升災難救援的效率和效果。

4. **探索工具**：系統可以配備各種探測設備，如攝影機、聲納和雷達，用於探索未知區域。

5. **攻擊性武器**：在某些情況下，自動跟車系統可以設計成攻擊性武器，用於防禦或攻擊目的。

十四、 參考資料

1. Arduino Uno Datasheet

https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

2. HC-SR04 Ultrasonic Sensor Guide

https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/

3. SG90 Servo Motor with Arduino

https://www.electronics-lab.com/project/using-sg90-servo-motor-arduino/

4. Raspberry Pi Linux Kernel Documentation

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

5. Raspberry Pi Linux Kernel Documentation (config.txt)

https://www.raspberrypi.com/documentation/computers/config_txt.html#kernel