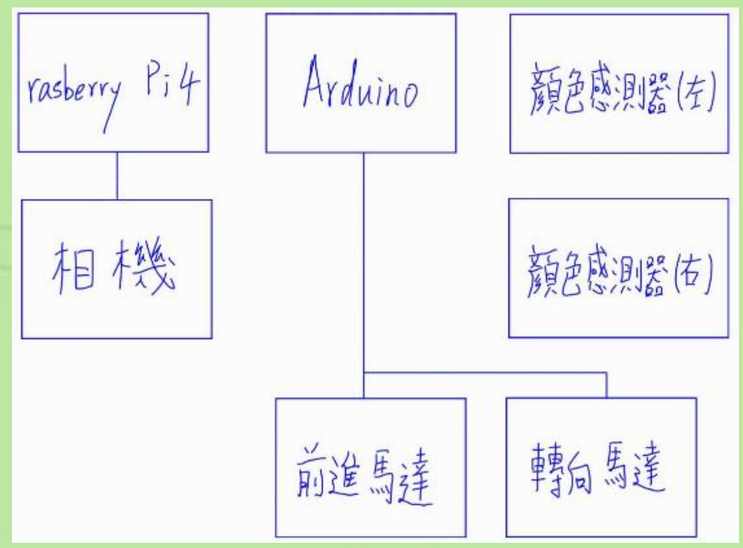
明志科大電機系113學年度專題製作競賽

紅綠燈自走車

組別: 第二組-13　組員:王輊鈞、吳忠陽、李介民

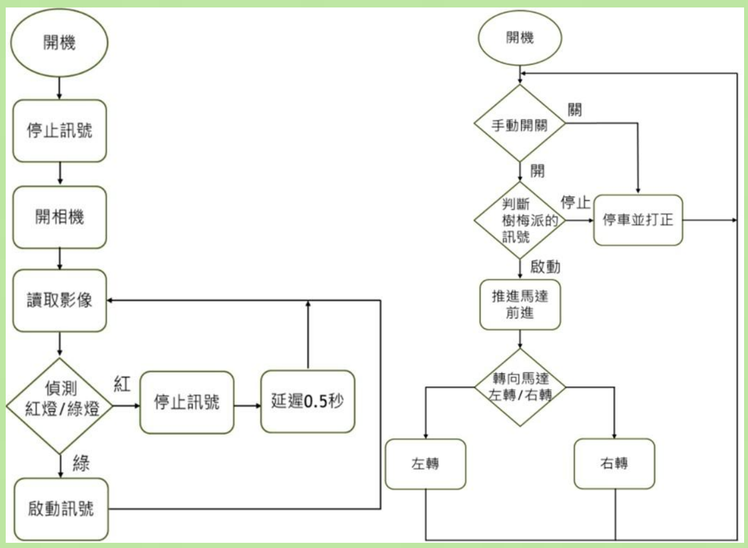
台灣的交通環境日益複雜，自駕車技術的發展也成為重要的研究方向。然而，自動駕駛在實際應用中仍面臨許多挑戰，像是如何在多變的交通情境中準確辨識周遭環境、做出即時反應，都是關鍵的問題。尤其在行駛過程中，自走車需要依賴影像辨識系統來判斷車道、交通號誌等…，但這些技術若缺乏充分測試與驗證，可能會導致誤判，進而影響行車安全。因此，我們希望透過模擬交通情境，來結合自走車系統。本自走車系統整合了相機模組、Arduino控制器Raspberry Pi與馬達驅動模組，具備多項基礎自駕功能，能夠在模擬交通情境中進行自主行動與判斷。系統啟動後，首先透過相機模組進行影像偵測，判斷前方是否存在明確的行駛軌道，若未偵測到軌道，則系統將停止馬達以避免偏離路線。當軌道明確時，自走車會進一步辨識是否出現紅燈，若紅燈亮起則自動停止，待解除後重新啟動馬達前進。此外，自走車也具備轉彎判斷與執行能力，當偵測到路線出現轉彎需求時，系統會根據轉彎方向進行左右判斷，並控制車輛完成對應的轉向動作。整個控制流程皆由Arduino負責統整執行，包含啟動、停止、轉向等命令。

系統硬體結構圖:



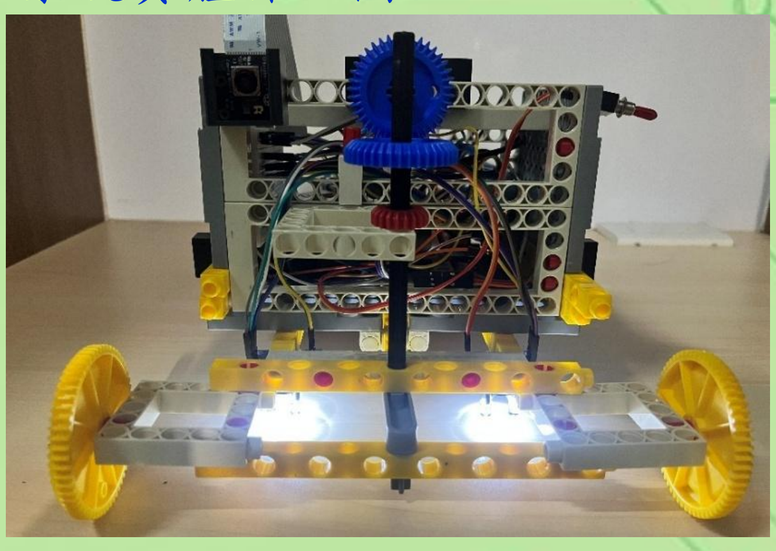
圖一

系統流程圖:



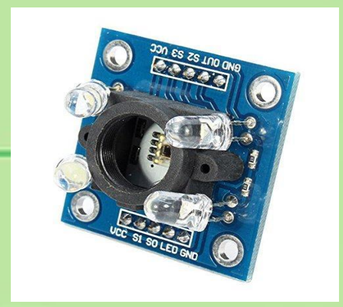
圖二

系統實體作品圖:



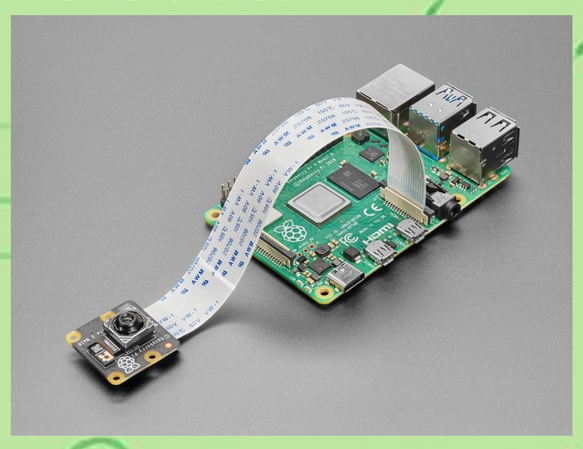
圖三

系統重要元件圖:



圖四
GY-31 TCS3200

圖五 伺服馬達



圖八 Rasberry pi 與相機

圖九 Arduino mega2560

Rasberry pi程式:

```
# 色彩範圍
lower_red = np.array([0, 0, 70])
upper_red = np.array([50, 50, 255])
lower_green = np.array([0, 70, 0])
upper_green = np.array([50, 255, 100])

# 建立遮罩
red_mask = cv2.inRange(rgb_frame, lower_red, upper_red)
green_mask = cv2.inRange(rgb_frame, lower_green, upper_green)

# 計算像素數量
red_pixels = cv2.countNonZero(red_mask)
green_pixels = cv2.countNonZero(green_mask)
```

圖六 判別紅、綠色

```
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        center = (i[0], i[1])
        radius = i[2]

        # 驗證遮罩佔比夠高
        area_mask = np.zeros_like(mask)
        cv2.circle(area_mask, center, radius, 255, -1)
        pixels_inside = cv2.bitwise_and(mask, mask, mask=area_mask)
        filled_area = cv2.countNonZero(pixels_inside)
        total_area = np.pi * radius * radius
        ratio = filled_area / total_area if total_area > 0 else 0

        #符合條件則劃一個圓到圖上
        if ratio > 0.5 and radius >= 5:
            cv2.circle(display_img, center, radius, (0, 255, 255), 2)
            cv2.circle(display_img, center, 2, (0, 0, 255), 3)
            valid_count += 1
```

圖七
確認圓型是否為一定程度的實心圓

Arduino程式:

```
void loop() {
  if(digitalRead(switchPin) == LOW || digitalRead(piPin) == LOW){
    re0();
  }
  else{
    action();
  }
}

//歸零
void re0()
{
  servo_34.write(90);
  delay(10);
  myServo.write(90);
}

//動作
void action(){
  //前進
  servo_34.write(180);

  // 讀取第一顆感測器（右側）的顏色數據
  red_1 = readColor(S2_1, S3_1, OUT_1, LOW, LOW);       // 紅色
  green_1 = readColor(S2_1, S3_1, OUT_1, HIGH, HIGH);   // 綠色
  blue_1 = readColor(S2_1, S3_1, OUT_1, LOW, HIGH);     // 藍色

  // 讀取第二顆感測器（左側）的顏色數據
  red_2 = readColor(S2_2, S3_2, OUT_2, LOW, LOW);       // 紅色
  green_2 = readColor(S2_2, S3_2, OUT_2, HIGH, HIGH);   // 綠色
  blue_2 = readColor(S2_2, S3_2, OUT_2, LOW, HIGH);     // 藍色

  Serial.println(red_1);
  Serial.println(green_1);
  Serial.println(blue_1);
  Serial.println(red_2);
  Serial.println(green_2);
  Serial.println(blue_2);
```

```
  // 判斷第一顆（右側）是否是綠色
  bool isGreen_1 = (green_1 + 20 < red_1);

  // 判斷第二顆（左側）是否是綠色
  bool isGreen_2 = (green_2 + 20 < red_2);

  // 控制伺服馬達
  if (!isGreen_1 && isGreen_2)
  {
    // 第一顆非綠色，第二顆綠色 -> 正轉 35 度
    myServo.write(125);
    Serial.println("伺服馬達正轉 20 度");
  }
  else if (isGreen_1 && !isGreen_2)
  {
    // 第二顆非綠色，第一顆綠色 -> 逆轉 35 度
    myServo.write(55);
    Serial.println("伺服馬達逆轉 20 度");
  }
  else if (isGreen_1 && isGreen_2)
  {
    // 兩顆都感測到綠色 -> 回到中立位置
    myServo.write(90);
    Serial.println("伺服馬達回到中立位置");
  }
  else
  {
    // 其他情況停止
    myServo.write(0);
    Serial.println("伺服馬達保持中立位置");
  }
}
```

圖十 arduino 重要程式內容