**超音波測距雷達系統設計與實作報告**

B1121141 葉彥辰；B1121149 張嘉宸

一、專題簡介

本專題旨在設計並實作一個簡易的雷達系統，結合伺服馬達與超音波測距模組（HC-SR04），透過 Raspberry Pi 控制伺服馬達旋轉，並同步進行距離測量，將掃描結果即時顯示於網頁儀表板上（Dashboard）。本系統實現了低成本的簡易 2D 掃描雷達功能，並結合 Node-RED 與 Chart.js 技術呈現視覺化雷達圖。

二、系統組成

Raspberry Pi

HC-SR04 超音波測距模組

SG90 伺服馬達

Node-RED 平台與 Dashboard 插件

Python 控制程式（mixed\_new.py）

Linux 系統與 pigpio 函式庫

三、Python 控制程式（mixed\_new.py）解析

此程式負責控制伺服馬達角度、進行距離測量，並將測得的資料輸出供 Node-RED 擷取。

程式碼功能說明摘要：

import time, signal, sys, pigpio, RPi.GPIO as GPIO

匯入必要的模組與函式庫。

def stop\_handler(signum, frame):

global running

    print(">>> 收到中斷訊號，結束程式")

    running = False

signal.signal(signal.SIGINT, stop\_handler)

signal.signal(signal.SIGTERM, stop\_handler)

處理中斷訊號（如 Ctrl+C）以安全地停止掃描與釋放資源。

pi = pigpio.pi()

if not pi.connected:

    print("pigpiod 尚未啟動")

    sys.exit(1)

SERVO\_GPIO = 12

TRIG = 23

ECHO = 24

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG, GPIO.OUT)

GPIO.setup(ECHO, GPIO.IN)

GPIO.output(TRIG, False)

time.sleep(1)

設定 pigpio 用來控制伺服馬達的 PWM。

設定 GPIO 模式並初始化超音波模組所需的腳位。

def set\_servo\_angle(angle):

    pulse = max(600, min(2400, int(600 + angle \* 10))) # 0~180 -> 600~2400

    pi.set\_servo\_pulsewidth(SERVO\_GPIO, pulse)

根據角度設定對應的脈衝寬度，控制伺服馬達旋轉到指定角度。

def get\_distance():

    GPIO.output(TRIG, True)

    time.sleep(0.00001)

    GPIO.output(TRIG, False)

start = time.time()

    while GPIO.input(ECHO) == 0:

        if time.time() - start > 0.05 or not running:

            return 0

    pulse\_start = time.time()

    while GPIO.input(ECHO) == 1:

        if time.time() - pulse\_start > 0.05 or not running:

            return 0

    pulse\_end = time.time()

    duration = pulse\_end - pulse\_start

    return round(duration \* 17150, 2)

try:

    angle = 0

    step = 2

發送超音波並接收回波時間，計算距離（單位為公分）。

while running:

        set\_servo\_angle(angle)

        dist = get\_distance()

        print(f"{angle - 90} {dist}", flush=True)

        angle += step

        if angle >= 180 or angle <= 0:

            step \*= -1

        time.sleep(0.05)

except Exception as e:

    print(f"[錯誤] {e}")

finally:

    print("清理中...")

    pi.set\_servo\_pulsewidth(SERVO\_GPIO, 0)

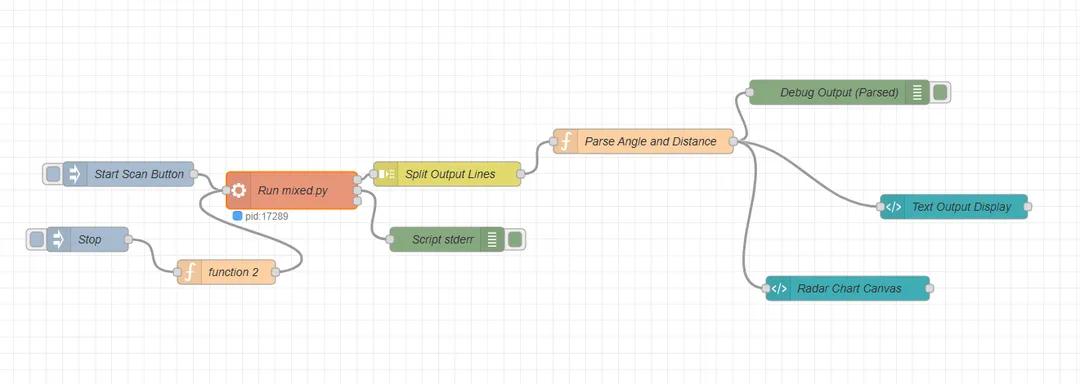
    pi.stop()

    GPIO.cleanup()

    print("結束完成")

不斷旋轉伺服馬達，邊旋轉邊測距，並即時將「角度 距離」以文字方式印出。

四、Node-RED 流程解析



功能流程總覽：

透過 inject 節點觸發 Python 腳本。

Python 腳本每輸出一行，即由 split 節點分解每行資料。

使用 function 節點解析角度與距離，轉換為物件形式。

將資料輸出至：

Debug 視窗

文字顯示欄位（顯示當前角度與距離）

雷達圖（Chart.js）以雷達形式即時顯示。

**Radar chart program:**

<!-- 引入 Chart.js 套件 -->

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<!-- 畫布設定 -->

<div style="width: 300px; height: 300px; margin: auto;">

<canvas id="radarCanvas" width="300" height="300"></canvas>

</div>

<script>

(function(scope) {

// === 建立角度標籤（-180° 到 180°，每 10° 一格） ===

const angleLabels = [];

for (let i = -180; i <= 180; i += 10) {

angleLabels.push(i + "°");

}

// === 初始化各角度對應距離值，預設為 0 ===

const dataArray = Array(angleLabels.length).fill(0);

// === 取得畫布 context 並初始化雷達圖 ===

const ctx = document.getElementById('radarCanvas').getContext('2d');

const radarChart = new Chart(ctx, {

type: 'radar',

data: {

labels: angleLabels,

datasets: [{

label: 'Distance (cm)',

data: dataArray,

backgroundColor: 'rgba(255, 99, 132, 0.2)',

borderColor: 'rgba(255, 99, 132, 1)',

borderWidth: 1,

pointRadius: 3

}]

},

options: {

animation: false,

responsive: true,

elements: {

line: { tension: 0 } // 直線連接

},

scales: {

r: {

beginAtZero: true,

max: 200,

ticks: {

stepSize: 40

}

}

},

plugins: {

legend: { display: false }

}

}

});

// === 資料更新邏輯：監聽 Node-RED 傳入的資料 ===

scope.$watch('msg', function(msg) {

if (!msg || !msg.payload) return;

const angle = msg.payload.angle;

const distance = msg.payload.distance;

// 角度轉換為陣列索引：angle → index

// 說明：角度 \* -1 後 +180，映射到索引範圍 [0, 36]

const index = Math.round((angle \* -1 + 180) / 10);

if (index >= 0 && index < dataArray.length) {

dataArray[index] = distance;

radarChart.update(); // 更新圖表

}

});

})(scope);

</script>

重要節點說明：

inject：手動啟動掃描。

exec：執行 mixed\_new.py 腳本。

split：依照每一行輸出（換行符號）分割資料。

function：解析字串為數字並轉成 JSON 格式。

ui\_template (雷達圖)：使用 Chart.js 繪製雷達圖表。

ui\_template (文字顯示)：顯示當前角度與距離。

inject (Stop) + function (SIGTERM)：終止腳本。

五、實作成果與心得

<https://drive.google.com/file/d/1faGkxDzbCGwENh8t-3xbpiKV4n6XhftE/view?usp=sharing>

<https://drive.google.com/file/d/1fX4zZfWdYfWbvlRTt7PX423LFt1SdCRl/view?usp=sharing>



透過本專題，我學會了如何整合感測器與伺服馬達，並以 Python 控制硬體裝置。同時，也熟悉了 Node-RED 的流程設計與 UI 元件操作。透過 Chart.js 我能將資料視覺化，進一步提升介面的易用性與資料的可解讀性。

在開發過程中，遇到過以下幾點挑戰：

超音波測距模組在角度改變過快時資料不穩定。

伺服馬達 PWM 控制需精準對應角度與脈衝寬度。

Chart.js 中角度資料對應索引需進行角度換算與修正。

最終成果呈現了一個可以即時顯示掃描角度與距離資訊的簡易雷達裝置，具有教育示範與進階擴充的潛力。