國立雲林科技大學電子工程學系

實務專題報告

GPS 物聯網應用與設計

GPS IoT Application and Design

指導教授：王　　萱　　鎰

作　　者：陳　　品　　澤

　　麥　　光　　廷

民 國 一 百 一 十 一 年 一 月 十 日

中文摘要

運用 MicroPython 與 ESP8266 進行物聯網系統的設計。使用 DHT11 溫溼度模組、SSD1306 OLED 模組、GY-GPS6MV2 模組設計出一套經緯度與溫溼度偵測，並將資料打印在 OLED 螢幕上且上傳至伺服器，伺服器則將接收到的資料繪製出行走路徑並且顯示當前溫濕度。

本次專題使用 Thonny 來撰寫 MicroPython 程式碼，讀取各感測器資料後打印在螢幕上，並用 Node.js 及 MariaDB 來架設網站伺服器及資料庫，負責接收感測器傳送的資料。

關鍵字：MicroPython、Node.js、MariaDB、GPS

ABSTRACT

IoT system design using MicroPython and ESP8266. We designed a set of latitude and longitude and temperature and humidity detection using the DHT11 module、SSD1306 OLED and GY-GPS6MV2 module, and printed the data on the OLED screen and uploaded it to the server, and the server plotted the received data into a travel path and displayed the current temperature and humidity.

This project uses Thonny to write MicroPython code, load the sensor data and print it on the screen, and use Node.js and MariaDB to set up a web server and database to receive the data sent by the sensors.

Keywords： MicroPython、Node.js、MariaDB、GPS

目錄

圖目錄

第一章　緒論

1.1 專題製作背景及目的

在現代，汽機車發展快速的年代，GPS 的需求量非常大量。不管是運用在地圖導航上，或是運動路線紀錄上都可以見到 GPS 的身影。不只是純粹讀取到 GPS 座標，可以結合地形數據或是其他資料來計算。

我們利用 MicroPython 開發版讀取到經緯度及溫溼度資料後，利用網路回傳 GPS 座標和溫溼度到伺服器中，來得知目前位置與行走路徑和當下溫溼度。

1.2 專題製作方法與步驟

專題製作大致可分為三個步驟，**MCU 控制器**、**網路設定**、**伺服器**。

MCU 控制器：使用 MicroPython 編寫程式碼到 ESP8266，使其控制各項感測器並讀取資料。

網路設定：設定 ESP8266 網路，使其連接至手機熱點。

伺服器：使用 Node.js 及 MariaDB 編寫 Web Server，使其接收 ESP8266 傳輸的資料。

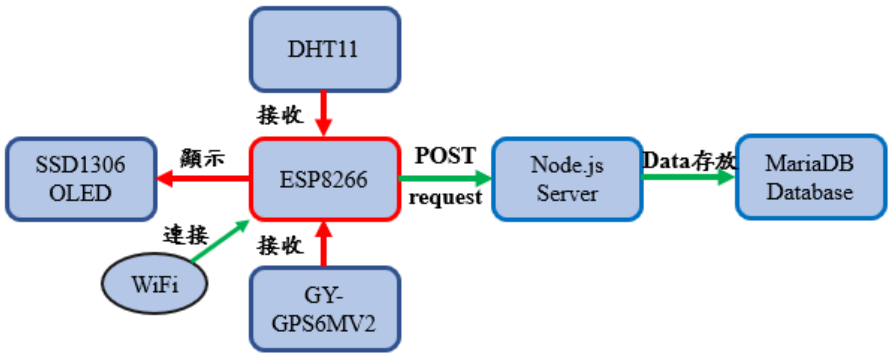

圖 1- 1 系統架構

第二章　技術探討

2.1　ESP8266 及其他模組

ESP8266 晶片由上海樂鑫公司所開發出來，是顆 32-bit 單晶片處理器，整合了支援 IEEE802.11 b/g/n 的 WiFi 晶片。它硬體規格如下：

- CPU 選用 32-bit Tensilica Xtensa LX3，處理速度 80MHz。
- ROM/RAM 方面提供 64K Boot ROM，64K Instruction RAM 與 64K Data RAM。
- 額外 Flash 擴充到 4MB，可以儲存即時作業系統與應用程式。
- 具有 WiFi 模組，可以設定為 AP、Station 或 AP+Station 等各種網路應用模式。
- 有 16 支 GPIO，支援 PWM、I2C、UART、SPI 及 10-bit ADC 等週邊功能。

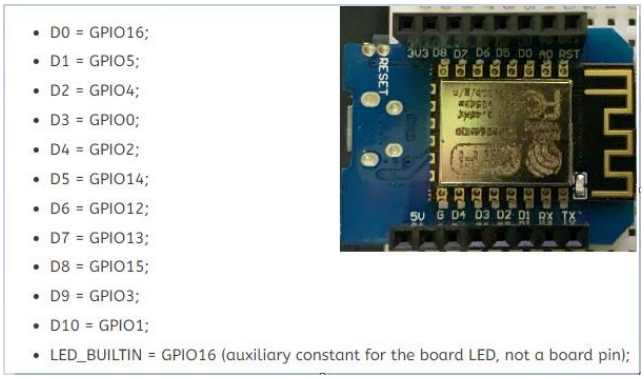

圖 2- 1 ESP8266 及 GPIO 接腳

DHT11 是一款經過校准過且直接以數字訊號輸出的溫濕度感測器。內含一個電阻式感濕元件和一個 NTC 測溫元件，並與一個 8bit 單晶片相連接。體積小、功耗低，傳輸距離最遠可達 20 公尺以上。

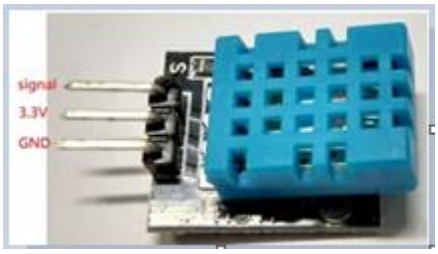

圖 2-2 DHT11 及接腳

SSD1306 是一款帶控制器的用於 OLED 點陣圖形顯示系統的單片 CMOS OLED/PLED 驅動器。 SSD1306 內建對比度控制器、顯示 RAM（GDDRAM）和振盪器，以此減少了外部元件的數量和功耗。該晶片有 256 級亮度控制。資料或命令由通用微控制器通過硬體選擇的 6800/8000 系通用並行介面、I2C 介面或序列外圍介面傳送。

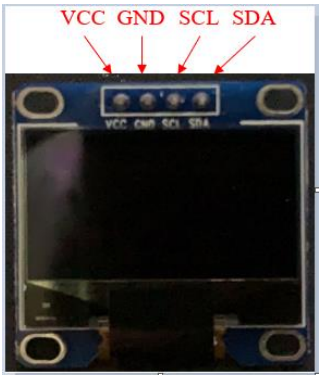

圖 2-3 SSD1306 OLED 及接腳

GY-GPS6MV2 是一款用於追蹤 GPS 衛星定位的模組。具有高靈敏度、低功耗、小型化，其極高追蹤靈敏度大大擴大了其定位的覆蓋面，在普通 GPS 接收模組不能定位的地方， 如狹窄都市天空下、密集的叢林環境，他都能接收的到訊號。適用於車載、手持設備如 PDA，車輛監控、手機、攝像機及其他移動定位系統的應用。

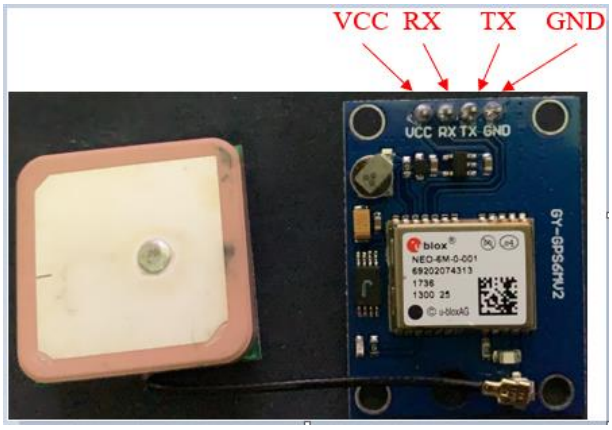

圖 2-4 GY-GPS6MV2 及接腳

2.2 MicroPython

MicroPython 是 2013 年在 Kickstarter 上募資開始建立的，顧名思義就是因為小型硬體資源有限，而將 Python 濃縮成一款小型包，載入硬體微控制器的一項開源專案，目前已經能移植於 Arduino 和 ESP8266 等板子，亦有自己專屬的開發板。

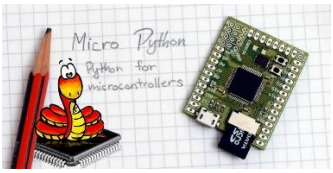

圖 2-5 MicrPython

2.3 Node.js 和 MariaDB

Node.js 是一個高效能、易擴充的網站應用程式開發框架 (Web Application Framework)。它誕生的原因，是為了讓開發者能夠更容易開發高延展性的網路服務，不需要經過太多複雜的調校、效能調整及程式修改，就能滿足網路服務在不同發展階段對效能的要求。

圖 2-6 Node.js

MariaDB 由 MySQL 的原始開發人員製作，也是一種開源軟體的資料庫，著名的用戶包括維基百科，WordPress 和谷歌，並且 MariaDB Server 是世界上最受歡迎的資料庫伺服器之一。

圖 2-7 MariaDB

第三章　專題設計與製作

3.1　ESP8266 與各項模組

DHT11 感測器控制：

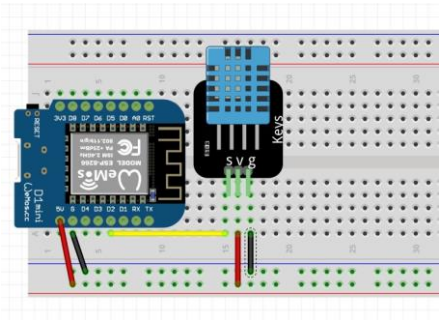

圖 3- 1 DHT11 接腳

```
69  class DHT11:
70      # 初始化DHT11
71      def __init__(self):
72          self.d = dht.DHT11(Pin(4))
73      # 偵測周邊溫濕度
74      def measureDHT(self):
75          self.measure()
76      # 讀取溫度，並回傳
77      def temperature(self):
78          self.d.measure()
79          temp = self.d.temperature()
80          return temp
81      # 讀取濕度，並回傳
82      def humidity(self):
83          humid = self.d.humidity()
84          return humid
98  d = DHT11()
109 temp = d.temperature()
110 humid = d.humidity()
```

圖 3-2 DHT11 控制程式碼

GY-GPS6MV2 感測器控制：

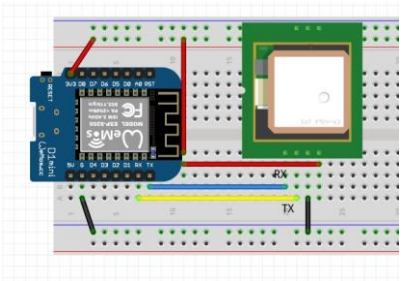

圖 3-3 GY-GPS6MV2 接腳

```
# Define GPS
class GPS:
    # 初始化GPS埠
    def __init__(self):
        self.com = UART(0, 9600)
        self.com.init(9600)
    # 定義「讀取GPS資料，並回傳」
    def getGPSInfo(self):
        data = self.com.readline()
        return data
    # 定義「緯度，並回傳」
    def latitude(self, d, h):
        if d == '':
            return 'null'

        hemi = '' if h == 'N' else '-'
        deg = int(d[0:2])
        min = str(float(d[2:]) / 60)[1:]

        return hemi + str(deg) + min
    # 定義「經度，並回傳」
    def longitude(self, d, h):
        if d == '':
            return 'null'

        hemi = '' if h == 'E' else '-'
        deg = int(d[0:3])
        min = str(float(d[3:]) / 60)[1:]

        return hemi + str(deg) + min
    # 定義「解碼GPS讀取的資料，並回傳經緯度」
    def convertGPS(self, gpsStr):
        gps = gpsStr.split(b'\r\n')[0].decode('ascii').split(',')

        lat = self.latitude(gps[3], gps[4])  # N or S
        long = self.longitude(gps[5], gps[6]) # E or W

        return (lat, long)
gps = GPS()
oled = OLED()
gpsStr = b''
gpsReading = False
data = gps.getGPSInfo()
if data and (gpsReading or ('$GPRMC' in data)) :
    gpsStr += data
    if '\n' in data:
        gpsReading = False
        lat, long= gps.convertGPS(gpsStr)
        gpsStr = b''
        break
    else:
        gpsReading = True
```

圖 3-4 GY-GPS6MV2 控制程式碼

SSD1306 OLED 顯示控制：

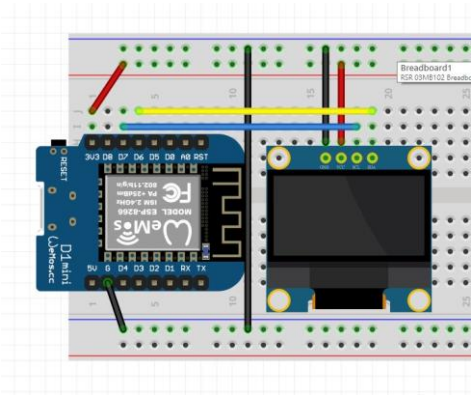

圖 3-5 SSD1306 OLED 接腳

```
class OLED:
    # 初始化OLED
    def __init__(self, scl=13, sda=12):
        self.oled = ssd1306.SSD1306_I2C(
            128, 64,
            I2C(scl=Pin(scl), sda=Pin(sda), freq=100000)
        )
        self.oled.text("GPS RUNNING...", 0, 30)
        self.oled.show()
    # 顯示經緯度及溫濕度
    def displayGPS(self, lat, long, temp, humid):
        lat = "Lat: " + lat
        long = "Long: " + long
        temp = "Temp: " + str(temp) + " C"
        humid = "Humid: " + str(humid) + "%"
        self.oled.fill(0)
        self.oled.text("Tracking", 0, 0)
        self.oled.text(lat, 0, 20)
        self.oled.text(long, 0, 30)
        self.oled.text(temp, 0, 40)
        self.oled.text(humid, 0, 50)
        self.oled.show()
oled = OLED()
oled.displayGPS(lat, long, temp, humid)
```

圖 3-6 SSD1306 OLED 控制程式碼

傳送資料：

```
# Define Web Server URL & Header
url = "http://[redacted]/web/add"
header = {'Content-Type':'application/json'}
value1 = '{"longitude":'+'"'+long+'"'+','
value2 = '"latitude":'+'"'+lat+'"'+','
value3 = '"temp":'+'"'+str(temp)+'"'+','
value4 = '"humid":'+'"'+str(humid)+'"}'
data = value1+value2+value3+value4
r = requests.post(url,data=data,headers=header)
```

圖 3-7 傳送資料的程式碼

3.2　網路伺服器和資料庫

MariaDB 資料庫設定(on CMD)：

```
create database gps;
use gps;
create table parameter(
id int(20) not null auto_increment,
longitude double(22,12) not null,
latitude double(22,12) not null,
temp int(11) not null,
humid int(11) not null,
primary key(id));
```

圖 3-8 指令新增 Database 及 Table

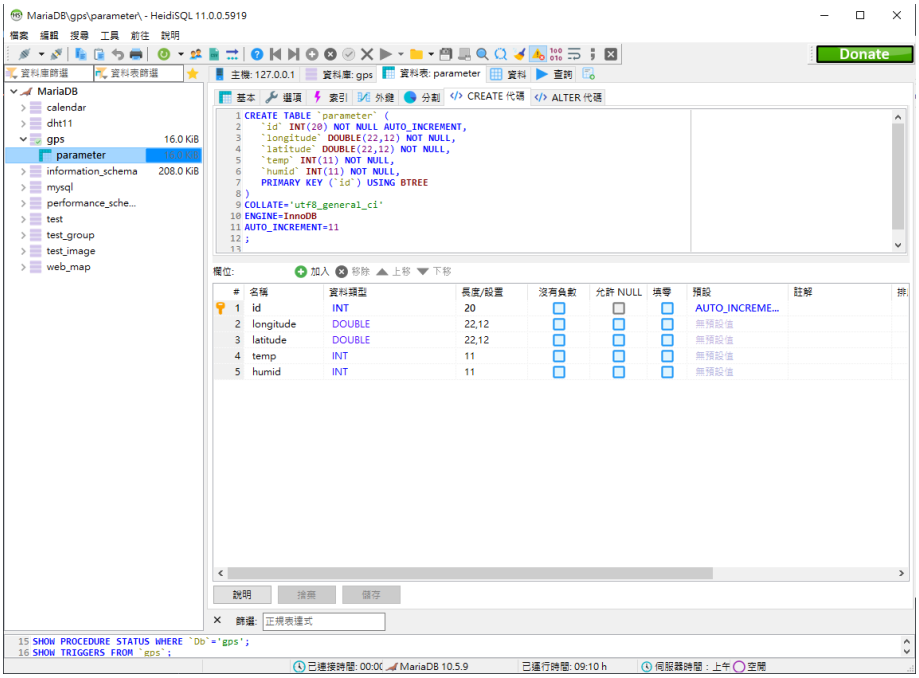

圖 3-9 資料庫表格圖示

Node.js(使用 Express 框架)：

Config.js(連接伺服器及資料庫)：

```
var config = {
    database:{
        host:'localhost',
        user:'root',
        password:'10713048',
        port:3306,
        db:'gps'
    },
    server:{
        host:'127.0.0.1',
        port:3000
    }
}
module.exports = config
```

圖 3-10 config.js 檔

App.js(伺服器主程式，設定資料庫及路由目錄)：

```
//Database
var dbOptions = {
        host: config.database.host,
        user: config.database.user,
        password:config.database.password,
        port: config.database.port,
        database:config.database.db
}
var indexRouter = require('./routes/index');
var webRouter = require('./routes/web');
//Router set
app.use(flash())
app.use('/', indexRouter);
app.use('/web', webRouter);
```

圖 3-11 app.js 檔

JavaScript：

Index.js(首頁)：

```
var express = require('express');
var router = express.Router();

//首頁js
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Welcome!!!'});
});

module.exports = router;
```

圖 3-12 Index.js 檔

EJS：

Index.ejs(首頁畫面)：

紅框內為主要內容，其餘每個頁面皆會重複出現，故之後不再顯示。

```
<!DOCTYPE html>
<html>
        <head>
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
                <meta charset="utf-8">
                <title><%= title %></title>
                <link rel="icon" href="../images/yuntech.ico" type="image/x-icon" />
                <style>
                        html, body {
                                height: 100%;
                                margin: 0;
                                padding: 0;
                        }
                </style>
        </head>
        <body>
                <div>
                        <a href="/"><h1>Home</a> |
                        <a href="/web/list">List</a> |
                        <a href="/web/add">Add</a> |
                        <a href="/web/map">Track</a>
                </div>
                <h1><%= title %></h1>
                <h2 style="color:red">ESP8266 location tracking by MicroPython.</h2>

                <!-- 錯誤訊息 -->
                <% if (messages.error) { %>
                <p style="color:red"><%- messages.error %></p>
                <% } %>
                <!-- 正確訊息 -->
                <% if (messages.success) { %>
                <p style="color:green"><%- messages.success %></p>
                <% } %>
        </body>
</html>
```

主要內容，其餘每個頁面皆會重複。

圖 3-13 Index.ejs 檔

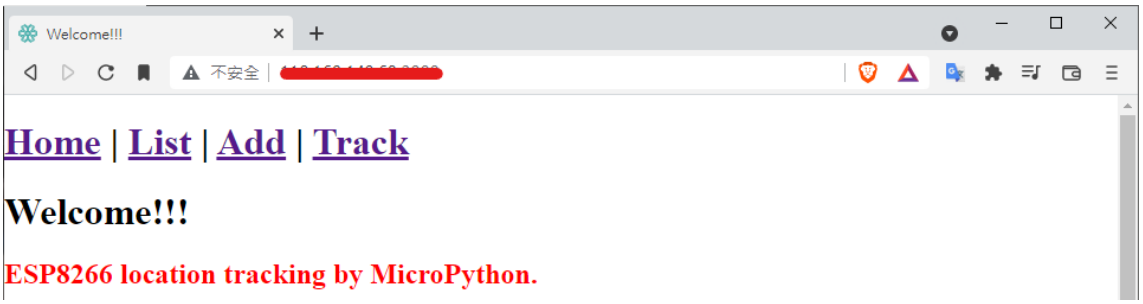

圖 3-14 首頁畫面

JavaScript：

Web.js(web 路由下的各頁面及功能，含刪除、資料列表、新增表單、地圖)：

```
//delete data
app.delete('/delete/(:id)', function(req, res, next) {

        var user = { id: req.params.id }
        req.getConnection(function(error, conn) {
                conn.query('DELETE FROM parameter WHERE id = ' + req.params.id, user,function(err, result) {
                        if (err) {
                                req.flash('error', err)
// redirect to web list page
                                res.redirect('/web/list')
                        } else {
                                req.flash('success', 'User deleted successfully! id = ' +req.params.id)
// redirect to web list page
                                res.redirect('/web/list')
                        }
                })
        })
});
```

圖 3-15 Web.js 檔之各項刪除功能

```
//delete all data
app.delete('/delete', function(req, res, next) {
        req.getConnection(function(error, conn) {
                conn.query('DELETE FROM parameter', function(err, result) {
                        if (err) {
                                req.flash('error', err)
// redirect to web list page
                                res.redirect('/web/list')
                        } else {
                                req.flash('success', 'User deleted successfully!')
// redirect to web list page
                                res.redirect('/web/list')
                        }
                })
        })
});
```

圖 3-16 Web.js 檔之全部刪除功能

```
//show list of my data
app.get('/list', function(req, res, next) {
        req.getConnection(function(error,conn) {
        conn.query('SELECT * FROM parameter ORDER BY id DESC',function(err, rows, fields) {
                if (err) {
                        req.flash('error', err)
                                res.render('web/list', {
                                        title: 'List All Data',
                                        data: ''
                                })
                } else {
// render to views/web/list.ejs
                        res.render('web/list', {
                                title: 'List All Data',
                                data: rows
                        })
                }
        })
        })
});
```

圖 3-17 Web.js 檔之資料列表功能

```
// post data form
app.get('/add', function(req, res, next){
        req.getConnection(function (error, conn){
                conn.query('SELECT * FROM parameter ORDER BY id DESC', function(err, rows, fieds){
                        res.render('web/add',{
                                title: 'Add New Data',
                                longitude : '',
                                latitude: '',
                                temp: '',
                                humid: ''
                })
                })
        })
});
```

圖 3-18 Web.js 檔之取得新增資料功能

```
// post data
app.post('/add', function(req, res, next){
        req.assert('longitude', 'longitude is required' ).notEmpty()
        req.assert('latitude', 'latitude is required').notEmpty()
        req.assert('temp', 'temp is required').notEmpty()
        req.assert('humid', 'humid is required').notEmpty()
        var errors = req.validationErrors()
        if( !errors ) {
                var     longitude = req.sanitize('longitude').escape().trim();
                var     latitude = req.sanitize('latitude').escape().trim();
                var     temp = req.sanitize('temp').escape().trim();
                var     humid = req.sanitize('humid').escape().trim();
                        req.getConnection(function(error, conn) {
                                conn.query('insert into parameter (longitude, latitude, temp, humid)'+
                                'Select ?,?,?,?',[longitude,latitude,temp,humid],
                                function(err, result) {
                                        if (err) {
                                                req.flash('error', err)
                                                res.render('web/add', {
                                                        title: 'Add New Data',
                                                        longitude : '',
                                                        latitude: '',
                                                        temp: '',
                                                        humid: ''
                                                })
                                        }else {
                                                req.flash('success', 'Data added successfully!')
                                                res.render('web/add', {
                                                        title: 'Add New Data',
                                                        longitude : '',
                                                        latitude: '',
                                                        temp: '',
                                                        humid: ''
                                                })
                                        }
                                })
                        })
        }
        else {
                var error_msg = ''
                        errors.forEach(function(error) {
                        error_msg += error.msg + '<br>'
                })
                req.flash('error', error_msg)
                res.render('web/add', {
                        title: 'Add New Data',
                        longitude: req.body.longitude,
                        latitude: req.body.latitude,
                        temp: req.body.temp,
                        humid: req.body.humid
                })
        }
```

圖 3-19 Web.js 檔之新增資料功能

```
//display Map
app.get('/map', function(req, res, next){
        req.getConnection(function(error,conn) {
        conn.query('SELECT * FROM parameter ORDER BY id ASC',function(err, rows, fields) {
                if (err) {
                        req.flash('error', err)
                                res.render('web/map', {
                                        title: 'Google map',
                                        data: ''
                                })
                } else {
// render to views/web/list.ejs
                        res.render('web/map', {
                                title: 'Google map',
                                data: rows
                        })
                }
        })
        })
});

module.exports = app
```

圖 3-20 Web.js 檔之地圖功能

EJS：

```
                <div>
                        <h1><%= title %></h1>
                        <form method="post" action="/web/delete" style="float:left">
                                <input type="submit" name="delete" value='ALL Delete' />
                                <input type="hidden" name="_method" value="DELETE" />
                        </form>
                </div>
```

圖 3-21 list.ejs 檔之全部刪除按鈕

```
                <table width='100%' border=0>
                        <tr style='text-align:left; background-color:#CCC'>
                                <th>longitude</th>
                                <th>latitude</th>
                                <th>temperature</th>
                                <th>humidity</th>
                                <th>Action</th>
                        </tr>

                <% if (data) { %>
                        <% data.forEach(function(user){ %>
                                <tr>
                                        <td><%= user.longitude %></td>
                                        <td><%= user.latitude %></td>
                                        <td><%= user.temp %></td>
                                        <td><%= user.humid %></td>
                                        <td>
                                                <div style="float:left">
                                                        <form method="post" action="/web/delete/<%= user.id %>" style="float:right">
                                                                <input type="submit" name="delete" value='Delete' />
                                                                <input type="hidden" name="_method" value="DELETE" />
                                                        </form>
                                                </div>
                                        </td>
                                </tr>
                        <% }) %>
                <% } %>
        </table>
```

圖 3-22 list.ejs 檔之表格內容

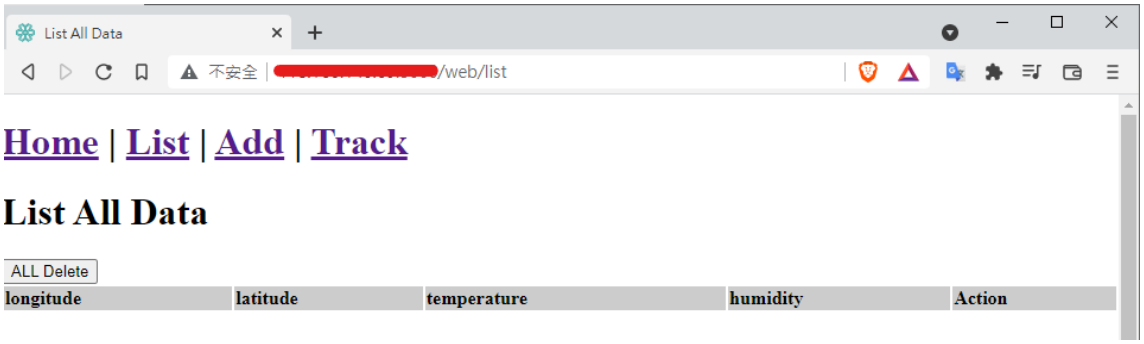

圖 3-23 資料列表畫面

```
<form action="/web/add" method="post" name="add" >
        <table width="35%" border="0">
                <tr>
                        <td>longitude</td>
                        <td><input type="text" name="longitude" id="longitude" value="<%= longitude %>" />
                        </td>
                </tr>

                <tr>
                        <td>latitude</td>
                        <td><input type="text" name="latitude" id="latitude" value="<%= latitude %>" />
                        </td>
                </tr>

                <tr>
                        <td>temperature</td>
                        <td><input type="text" name="temp" id="temp" value="<%= temp %>" />
                        </td>
                </tr>

                <tr>
                        <td>humidity</td>
                        <td><input type="text" name="humid" id="humid" value="<%= humid %>" />
                        </td>
                </tr>

                <tr>
                        <td><input type="submit" name="Submit" value="Add" /></td>
                </tr>
        </table>
</from>
```

圖 3-24 add.ejs 檔之表單格式

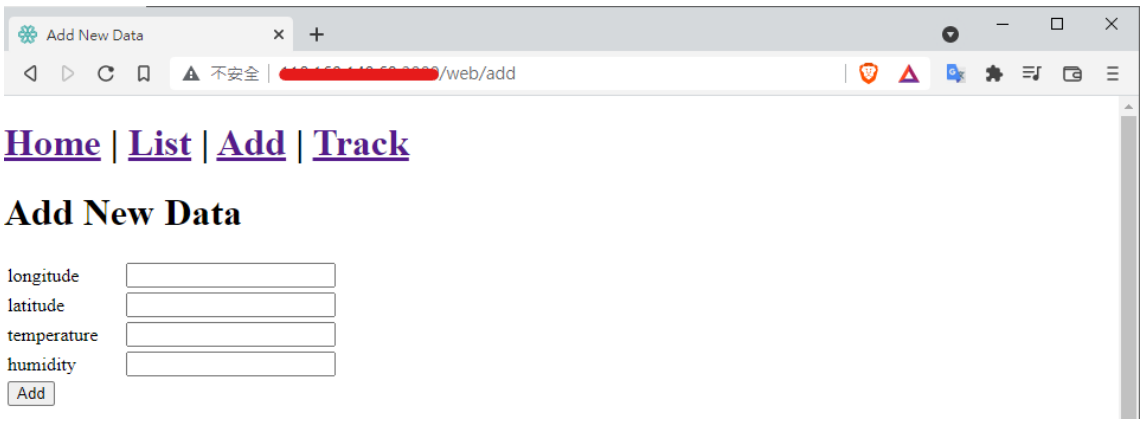

圖 3-25 新增資料畫面

```
<div>
        <h7>溫度 : <span id="temperature"></span>°C<h7>
        <h7>濕度 : <span id="humidity"></span>%<h7>
</div>
</div>
    <script>
        <% if (data) { %>
            <% data.forEach(function(user){ %>
                var temperature = <%= user.temp %>
                var humidity = <%= user.humid %>
                document.getElementById('temperature').textContent = temperature;
                document.getElementById('humidity').textContent = humidity;
            <% }) %>
        <% } %>
    </script>
```

圖 3-26 map.ejs 檔之溫溼度顯示

```
    <script>
        var map;
        var polyline = [];
        var temp = [];
        var symbol = {
            path: 'M-44.6,22.3l0-22.3l44.6,44.6H-44.6z',
            fillColor: '#292',
            fillOpacity: 0.7,
            strokeOpacity: 0,
            scale:0.2
        };
        var font_symbol = {
            path: "M -2,0 0,-2 2,0 0,2 z",
            strokeColor: "#292",
            fillColor: "#292",
            fillOpacity: 1,
        }
        var end_symbol = {
            path: "M -2,-2 2,2 M 2,-2 -2,2",
            strokeColor: "#292",
            strokeWeight: 4,
        }
        <%if (data) { %>
            let j = 0;
            <% data.forEach(function(user){ %>
                var temperature = <%= user.temp %>
                var humidity = <%= user.humid %>
                var position = {lat:<%= user.latitude%>, lng:<%= user.longitude%>};
                temp[j]= position
                j = j + 1;
                console.log(position);
            <% }) %>
        <% } %>
        console.log("temp:",temp);
```

圖 3-27 map.ejs 檔之定義資料流及圖標

```
function initMap() {
        map = new google.maps.Map(document.getElementById('map'), {
                zoom: 16,
                center: {
                lat: 23.692544,
                lng: 120.534734
                },
                styles: [{
        var latlngbounds = new google.maps.LatLngBounds();
        for (var i = 0; i < temp.length; i++) {
                addPolyline(i);
        }
        for (var j = 0; j < temp.length; j++) {
                latlngbounds.extend(temp[j]);
        }
        map.fitBounds(latlngbounds);
}
```

圖 3-28 map.ejs 檔之初始化地圖

```
function addPolyline(e){
        polyline = new google.maps.Polyline({
                path:temp,
                geodesic: true,
                strokeColor: "#FF0000",
                strokeOpacity: 1.0,
                strokeWeight: 2,
                icons:[{
                        icon: font_symbol,
                        offset: "0%"
                },{
                        icon: end_symbol,
                        offset: "100%"
                }]
        });
        polyline.setMap(map);
        var a = 0;
        setInterval(function(){
                a = a + 1;
                if(a>100){a = 0;}
                polyline.setOptions({
                        icons:[{
                        icon: symbol,
                        offset: a+"%"
                        }]
                })
        },50);
}

        setInterval('window.location.reload()',10000)
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=&callback=initMap"async defer></script>
```

圖 3-29 map.ejs 檔之繪製路線圖

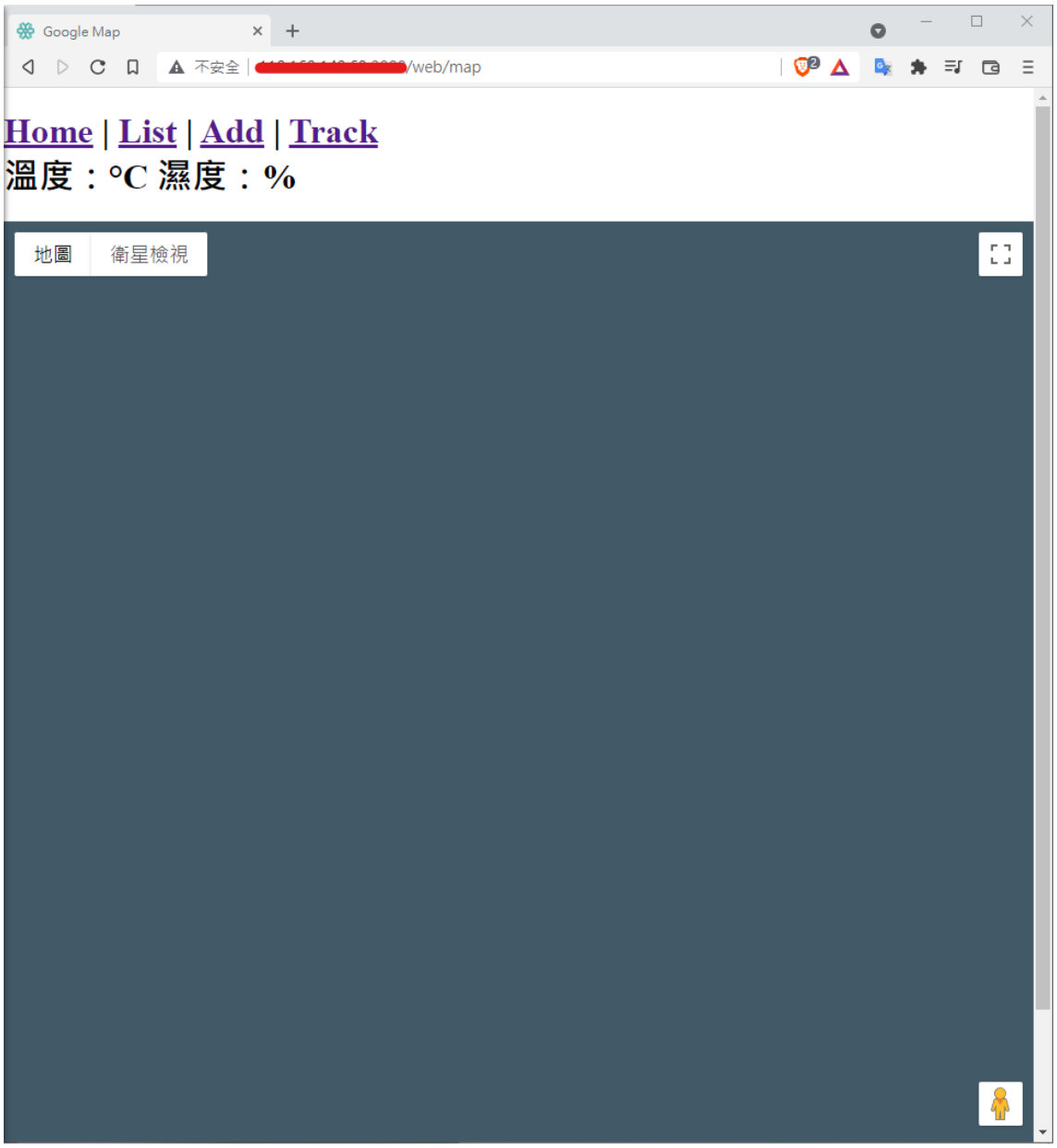

圖 3-30 地圖畫面

第四章　專題成果

4.1　ESP8266

接線：

VCC：3V3

GND：GND

DHT11-S：D2

GY-GPS6MV2-Tx：Rx

GY-GPS6MV2-Rx：Tx

SSD1306 OLED-SCL：D7

SSD1306 OLED-SDA：D6

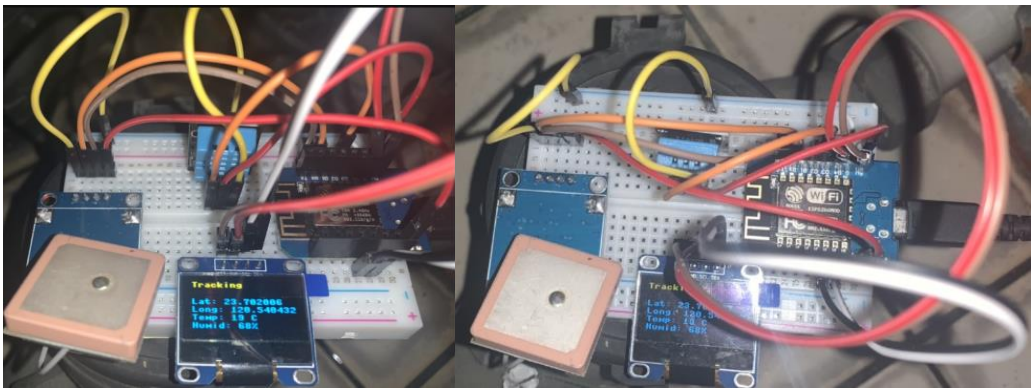

圖 4- 1 完成品

圖 4-2 OLED 顯示畫面

4.2 Web Server

首頁：

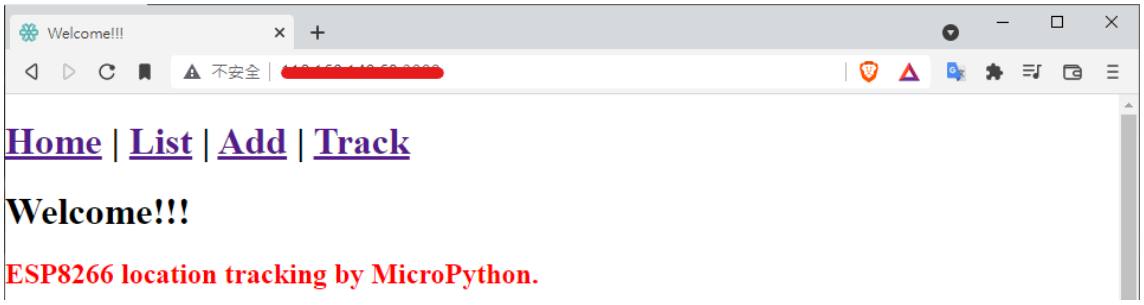

圖 4-3 首頁顯示畫面

資料列表：

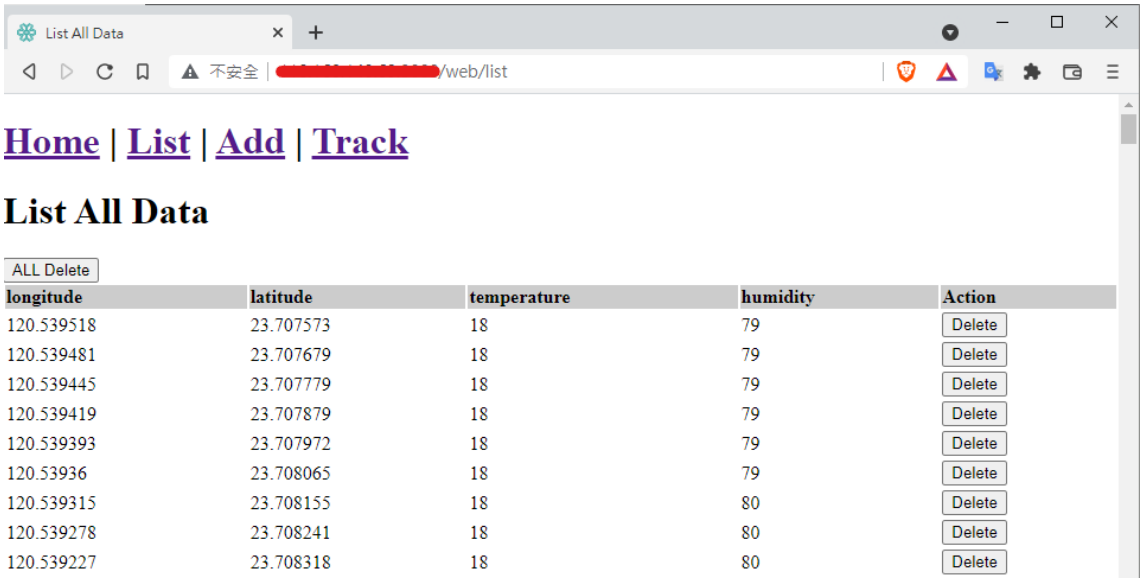

圖 4-4 資料列表顯示畫面

新增資料(用於與 ESP8266 串接)：

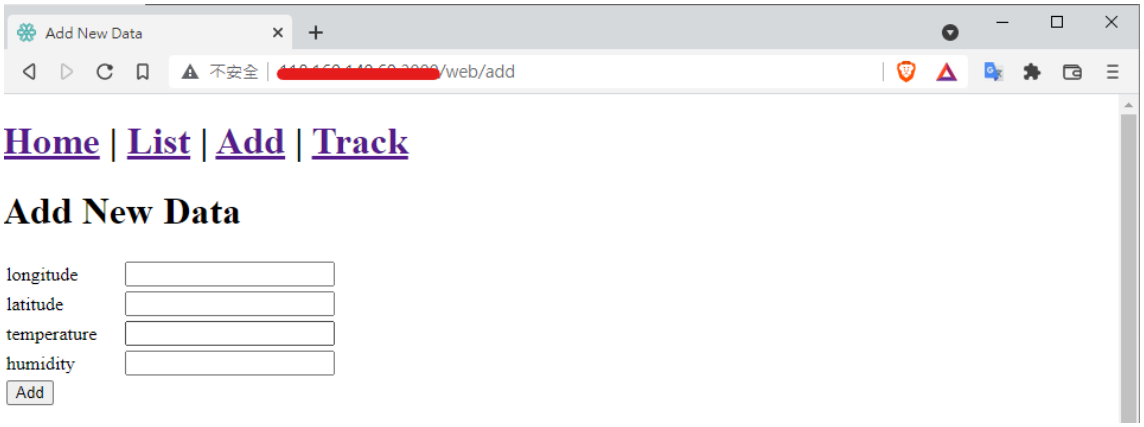

圖 4-5 新增資料顯示畫面

地圖及溫度顯示：

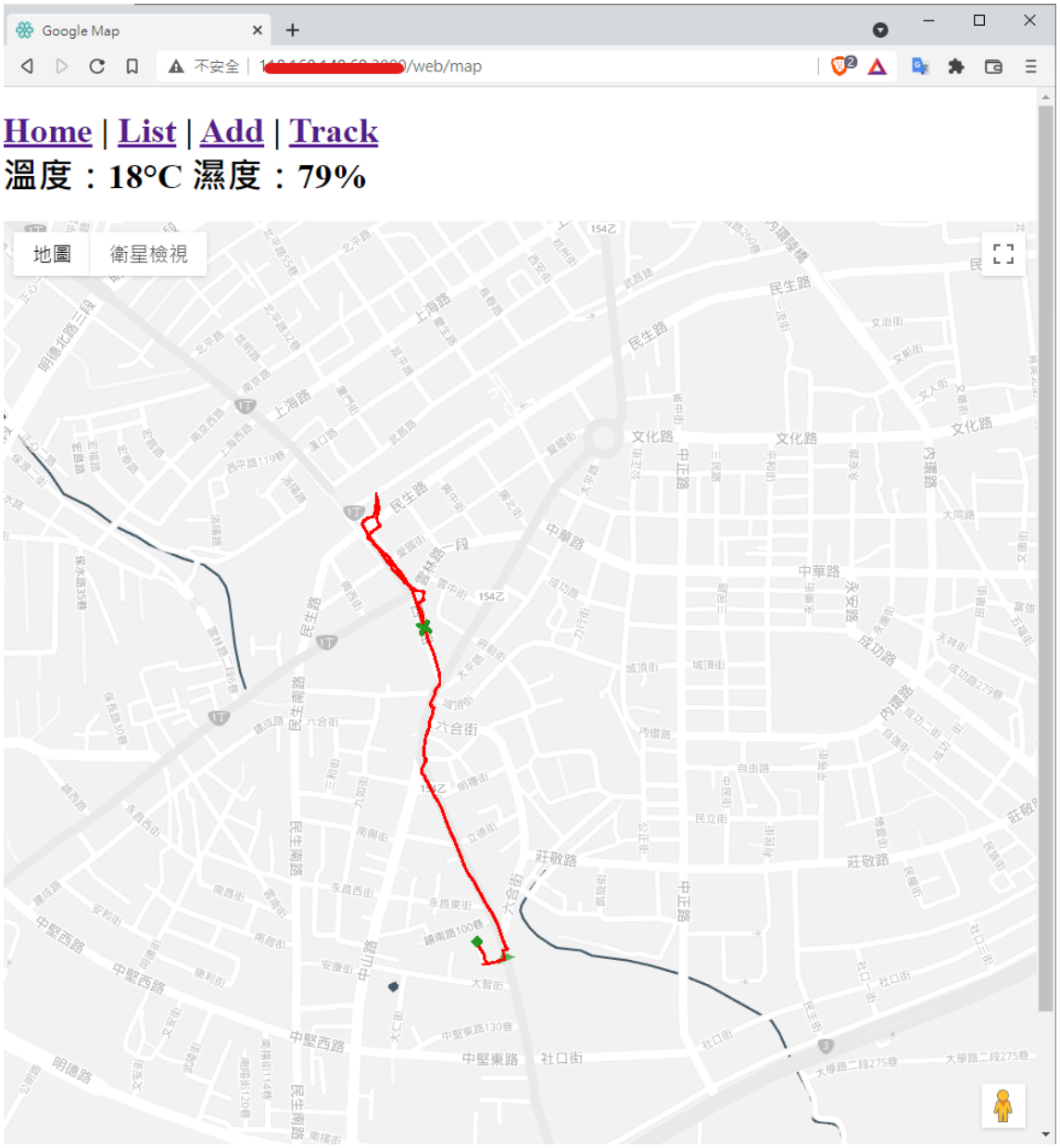

圖 4-6 地圖及溫度顯示畫面

第五章　結論

本次專題使用的程式語言有 MicroPython、JavaScript、MySQL 及 EJS。通過 ESP8266 這個 MCU 來控制 DHT11 溫濕度感測器及 GY-GPS6MV2 飛控 GPS 模組，藉由這兩個感測器去偵測外界的資訊(溫度、濕度、經度及緯度)，再使用 SSD1306 OLED 顯示器來顯示讀取到的資料，最後再藉由 urequest 的函式庫與自架的 Web Server 進行串接。

Web Server 將接收到的資料存放至 Database 中，當使用者點選地圖頁面時，就會讀取資料。溫溼度的資料是顯示在最上方，負責告訴使用者當下的溫度及濕度；經緯度則通過 Google Map API 來顯示在地圖上，並且依據各個位置畫製折線圖，告訴使用者 ESP8266 的行走路徑。

這個專題能夠應用在需要對特定目標進行追蹤的情況下，如重要物品或特定人物，通過追蹤結果，可以得出目標的當前位置及附近環境狀況，並且也可追蹤目標的行進路徑。

參考資料

[1] https://developers.google.com/maps/documentation/javascript/overview

[2] https://randomnerdtutorials.com/micropython-oled-display-esp32-esp8266/