Journal of Science and Engineering Technology, Vol. 20, No. 1, pp. 47-63 (2024)

環境監測物聯網系統之開發設計

洪育緯*　盧冠華　施倫閔　蔡渙良
大葉大學資訊工程學系
515006 彰化縣大村鄉學府路 168 號
*R1206006@cloud.dyu.edu.tw

摘　要

本論文研究開發設計「環境監測物聯網系統」係遵循物聯網（IoT）/工業物聯網（IIoT）/智慧物聯網（AIoT）：智慧感知整合→Wi-Fi 通訊網路→ThingSpeak IoT 公開雲/私有雲資訊應用與 MATLAB/Simulink 資料分析的物聯網架構，運用 WeMos D1 Mini Wi-Fi 模組整合亮度、溫度/濕度、空氣品質及氣體感測器等多樣、異質整合無線環境感測模組，提供智慧製造廠域所需的無線環境監測資料，經由 Wi-Fi 接取點（Access point, AP）上傳到 ThingSpeak IoT 公開雲/自架私有雲顯示平台，結合 MATLAB/Simulink 軟體資料分析工具，運用提供智慧製造廠務管理者廠區環境監測和環安衛管理監測等資料。首先運用 WeMos D1 Mini Wi-Fi 模組整合具備序列通訊介面的 BH1750、BME680、PMS7003T 及氧氣（O_2）等多元感測器整合成一組智慧製造廠域無線環境監測整合模組，提供學校智慧製造工廠場域（工學院 H203）所需的環境監測資料，經由 Wi-Fi 接取點（AP）上傳到 ThingSpeak IoT 公開雲/ Da-Yeh AIoT 雲端顯示平台，結合 MATLAB/Simulink 軟體資料分析工具，運用提供智慧製造廠務管理者廠區環境監測和環安衛管理監測等資料。

關鍵詞：物聯網（IoT），遠端監控，ThingSpeak IoT 公開雲，Da-Yeh AIoT 雲端顯示平台

Development of Environmental Monitoring System

YU-WEI HUNG*, GUAN-HUA LU, LUN-MIN SHIH and HUAN-LIANG TSAI
Department of Computer Science and Information Engineering, Da-Yeh University
No. 168, University Rd., Dacun, Changhua 515006, Taiwan, R.O.C.
**R1206006@cloud.dyu.edu.tw*

ABSTRACT

This paper details the development of an Environmental Monitoring System that incorporates smart sensors, wireless communications, and the ThingSpeak Internet of Things (IoT) public cloud/private cloud information application. Results of MATLAB/Simulink data analysis are provided. A WeMos D1 Mini Wi-Fi module was used that integrates brightness, temperature, humidity, noise, air quality, and gas sensors. This module provides wireless environmental monitoring data for smart manufacturing plants. The data are uploaded via Wi-Fi access points to the ThingSpeak IoT public cloud/private cloud display platform, and MATLAB/Simulink software is used to analyze these data. The tools enable environmental monitoring and safety management at factories. This study proposes

the integration of all I^2C SCL/SDA, UART Tx/Rx, and SPI serial communication standard interfaces as the main innovation and creative concept of the wireless environmental monitoring integration module. No similar products are currently available, making the proposed system highly valuable in terms of development and research.

Key Words**:** IoT, remote monitoring, ThingSpeak IoT Public Cloud, Da-Yeh AIoT Cloud, display platform

一、前言

2011 年漢諾威工業展中提出工業 4.0（Industry 4.0）概念，結合智慧機械、機器人、物聯網、智慧產線及巨量資料、機器學習、人工智慧等虛實整合系統（Cyber-physical system, CPS）的智慧製造（Intelligent manufacturing）、數位製造（Digital manufacturing）正逐漸改變傳統製造產業的生產模式。參考 2018 年美國國家科技委員會（National Science & Technology Council, NSTC）未來製造（Future manufacturing）報告，符合少量多樣、多元產品等先進製造需求的智慧製造技術是下一世代的生產製造技術；同時，參考 2020 年 3 月美國國家科技基金會（National Science Foundation, NSF）未來製造（Future manufacturing）計畫徵求內容結合虛實整合系統（CPS）的數位製造技術是下一世代的生產製技術，運用布建與整合感測與控制技術，經由數位分身（Digital twin, DT）、虛實整合、人工智慧（Artificial intelligence, AI）和機器學習（Machine learning, ML）建立新的演算法，在先進的高速網路基礎設施下，以達到次世代數位製造有關開發新材料、元件、設備以及製程技術等目標。虛實整合系統（CPS）的數位製造技術結合了機械設計與智慧製造、電子電機、資通訊等三大產業，正是臺灣深耕已久且具世界競爭力的領域；為躋身先進製造國家舞台，打造新世代智慧工廠、智慧製造及數位製造是台灣製造業必須要面對的變革，行政院會議中更制訂「智機產業化」與「產業智機化」兩大方針推動我國智慧機械產業，並強調運用智慧感知、物聯網、智慧機械/機器人、巨量資料與精實管理等技術，推動產業朝向設備智能化、工廠智慧化與系統數位化、虛實整合發展，加速提升各種產業附加價值與生產力，期使台灣的精密機械與智慧製造/數位製造業更上一層樓。

自 2020 年全球受到 COVID-19 疫情影響，世界主要國家的工業製造年增率大幅下降，尤以歐美國家受到影響為最大，參考我國經濟部的統計數據顯示，我國在這波疫情中逆勢成長，本研究計畫整理 2020 年及 2021 年全球主要國家工業製造年增率比較分析。其中口罩自動化製程整廠輸出是數位製造技術的最佳典範，「口罩產業整廠輸出聯盟國家隊」整合口罩機、熔噴過濾不織布機、耳帶線機及包裝機等不同工具機，提供北美、歐洲及東南亞地區口罩及相關醫療自動化生產整廠輸出的商機。我國政府因應全球工業 4.0 推動「生產力 4.0」產業發展方案：（1）以智慧自動化為基礎，導入智慧感知、人工智慧（AI）、虛實整合系統（CPS）技術，以創造智能意識化製造業，具備預測製造、預防維修等附加價值高的智慧製造能力；（2）運用智慧感知加值集成電腦化/數位化/智能化技術，發展具備有即時感知、適應性、資源效率及人因工程的智慧製造工廠，貫穿企業夥伴流程及產業價值流程，創造產品與服務客製化供應能力。隨著智慧感知、人工智慧（AI）、物聯網（Internet of Thing, IoT）/工業物聯網（Industry IoT, IIoT）/智慧物聯網（AI IoT, AIoT）等智能化技術及 5G 高速資通訊平台的快速發展，運用智慧感知、IoT/IIoT/AIoT 智慧聯網、虛實整合系統（CPS）的數位分身（DT）技術可以加值在顧客端需求、產品設計、生產、製造的流程，當確認顧客端的需求時，即可事先了解設計、生產、製造方面的限制及規劃製造、生產所需資源和流程，在實際生產製造產品之前可有效模擬，並發現虛擬的數位分身（DT）模型中的設計缺陷從而改進。另一方面，智慧感知、大數據（Big data）分析、機器學習（Machine learning, ML）、深度學習（Deep learning, DL）等智能化技術及 5G 高速、高頻寬資通訊平台的推進，因應數位製造整合系統的多樣性，客製化製程所開發之機器學習（ML）/深度學習（DL）模型的動態調整和參數維護、建模之後的後續維護以及模型的穩定性都是重要的議題；同時，在智慧感知、智慧聯網技術提供巨量大數據資訊下，後續模型將隨著不斷輸入的資料進行機器學習（ML）/深度學習（DL）智能化學習，以符合數位化智慧製造工廠的實際運作狀況，降低後續人員維護成本。近年來，隨著邊緣運算（Edge computing）的發展，IoT/IIoT/AIoT 智慧聯網架構包含了智慧感知層、網路通訊層、邊緣運算層及雲端應用層。

回顧有關智慧感知技術的物聯網（IoT）/智聯網（AIoT）系統的研究，我們可發現智慧製造領域的環境智慧感知系統及其感測器整合模組研究並不多，2013 年 Palazon 等人 [7]利用 WSN 網路節點定位功能開發人員及機動載具避碰功能，同時運用 Mongoose 九個自由度慣性測量單元（Inertial measurement unit, IMU）提供高空作業人員防墜偵測，感測資料經由 Wi-Fi 無線網路送到中央資料庫，經 Toolbox 工具顯示；Li 與 Kara 二人 [6]在 2017 年運用 TMP36 溫度感測器、XBee 無線網路及 Data.Sparkfun 公開雲端平台進行辦公場域空調系統（Heating, Ventilation and Air Conditioning, HVAC）的溫度監測個案研究，是一個簡單標準的場域環境監測物聯網系統；Almalki [1]在 2020 年首先提出即時的工業環境監測系統設計的理論性研究，雖然提出各式周遭環境、空氣品質、水質監控感測器的解決方案，惟未論及多元感測器整合及雲端應用層的設計；González 等人 [5]運用 BME680 氣體感測器、RoLa 通訊模組開發氮氧化物（NOX）、一氧化碳（CO）、揮發性有機物（Volatile organic compounds, VOCs）有毒氣體感測節點，經由 RoLa 閘道（Gateway）上傳到“The Things Network（TTN）”雲端監測系統；Campero-Jurado 等人 [2]運用 ESP-WROOM-32Wi-Fi 模組，整合 ALS-PT19 亮度感測器、BME680 溫度/濕度/壓力/氣體多元感測器、力敏電阻（Force sensitive resistor）、MPU6050 六自由度（6-DOF）運動感測器組裝應用在工業物網（Industrial IoT, IIoT）個人防護具（Personal protective equipment, PPE）的智慧頭盔，現場人員可以經由廠域 Wi-Fi 接取點（Access point, AP）將感測資料上傳 ThingBoard 雲端平台；Ghazal 等人 [4]運用具備邊緣-霧端-雲端（Edge-fog-cloud）架構的探測機器人，加載熱像儀和視覺影像儀來偵測鋁材生產廠域的不正常熱問題，影像資料經藍芽傳送到具有早期反應的霧端無人飛行載具，再經由 Wi-Fi 無線網路後送雲端處理；Dobrilovic 等人[3]運用 WeMos D1 R2、ESPDuino、NodeMCU 等 Wi-Fi 模組整合 MQ-135、MQ-2 氣體感測器配置在紙業製造廠域進行揮發性有機物或有毒氣體的監測，感測資料經 Wi-Fi 無線網路上傳到 Mosquitto MQTT 發布/訂閱代理人（Publisher/Subscriber broker）雲端伺服器。本論文整理文獻回顧如表 1.1。本文所使用感測器 BH1750、BME680、PMS7003T 以及 O_2 感測器。

表 1.1 智慧環境監測智慧物聯網監控技術比較表

項次 (Items)	應用場景 (Application scenario)	感知層 (Sensing layer)		通訊層 (Communication)	資訊應用層 (information platform)	參考文獻 Reference
		感測器 (Sensors)	處理器 (Processor)			
1	汽車工廠場域 TRW automotive factory	WSN nodes Mongoose 9DoF IMU	IRIS mote	Wi-Fi	None	[7]
2	辦公場域 Office environment	TMP36 temperature sensor	Arduino®	ZeeBee	Data.Sparkfun®	[6]
3	金屬處理廠域 (Metal processing workplace)	MQ-2 CO/SO_2 sensor GP2Y1010AU0F Dust sensor DS18B20 temperature sensor Soot Particulate Partikel NO_X sensor PHE-45P pH Sensor AW2T24TEL-4A1 Shock Sensor LDR illumination level sensor	Arduino Due Arduino Nano	ESP8266 Wi-Fi SIM900 GSM	None	[1]
4	大學校園 Rovira i Virgili University campus	BME680 multi-sensor	RoLa node	TTIG RoLa gateway	The Things Network (TTN)	[5]
5	智慧頭盔 Smart helmet 5.0	ALS-PT19 light sensor BME680 multi-sensor Force sensitive resistor MPU6050 6-DOF motion sensor	ESP-WROOM-32	Wi-Fi	ThingBoard IoT	[2]
6	鋁材生產廠域 Aluminum production environment	Thermal & visual imagers	Arduino Uno	BlueTooth Wi-Fi	cloud back-end robot surveyor	[4]
7	紙業百貨公司 (Paper Haberdashery)	MQ-2 gas sensor MQ-135 gas sensor	WeMos D1 R2 ESPduino NodeMCU	Wi-Fi	Mosquitto MQTT broker	[3]
8	智慧製造示範場域 Smart manufacturing demo field	BH1750 light sensor BME680 temp/humidity/IAQ sensor SM7901 noise sensor PMS5003T dust sensor	WeMos D1 mini	Wi-Fi	ThingSpeak IoT ThingView APP	本論文

本文的主要貢獻，在於希望能夠收集到足夠多的數據並進行分析，再結合物聯網，進行即時的監測，給人們帶來安全的工作環境；本論文實作成果對社會環境、工業廠域提供一個高性價比的環境整合監控平台，同時在學術研究提供一個可以做為多元、異質感測器整合實作的實例。在本文第一章將介紹我們進行本研究的啟發以及近期相同領域的資料；第二章介紹了本裝置的各式部件，包括亮度感測器、溫濕度感測器、懸浮微粒感測器、氧氣感測器以及 WeMos D1 mini Wi-Fi 模組；第三章是我們對於研究成果的預測以及完成組裝的裝置；第四章則展示我們使用裝置所測量出來的數據經過了統計，所呈現出來的圖表；最後第五章是對於透過「環境監測物聯網系統之開發設計」的研究，做出其關鍵發現和理論貢獻的總結，同時也提出對未來研究的建議和展望。

二、研究理論基礎

本文主要探討智慧環境監測系統，規劃以 WeMos D1 Mini Wi-Fi 模組整合亮度、溫度/濕度、$PM_{1.0/2.5}$、空氣品質及氧氣氣體感測器等多樣、異質整合無線感測模組，可以提供檢測周遭環境的亮度、溫度、濕度、$PM_{2.5}$ 粉塵微粒、氧氣百分比濃度等多元感測資料，經由 Wi-Fi AP 無線通訊網路上傳到 ThingSpeak IoT 或自建雲端平台，使用者可以經由電腦或行動裝置，進行線上網頁瀏覽，或運用 ThingView APP 直接查詢相關電氣參數。

本文所使用的感測器有亮度感測、溫濕度感測、懸浮微粒感測、氧氣感測以及空氣品質感測等不同的感測器並結合 Wi-Fi 模組；亮度感測器是利用光敏元件將光轉成電信訊號，通常是由一組投光器還有受光器所組成；溫/濕度感測器是使用化學性感測來計算空氣中的濕度；懸浮微粒感測器是利用雷射光散射的原理，計算空氣中懸浮粒子的數量；氧氣感測器則是利用氧氣的磁化效率遠大於其他氣體來對空氣中的氧氣含量進行檢測；最後的 Wi-Fi 模組則是將以上所有感測器的數據利用網路上傳到數據庫，方便進行即時監測，如圖 2.1。

（一）感測器原理

如圖 2.2(a)所示，BH1750 光感應器主要採用了一個光敏二極體(photodiode, PD)，這是一種具有 p 型-本質型-n 型(p-intrinsic-n, P-I-N)結構的光敏半導體裝置，即本質半導體夾在 p 型和 n 型半導體之間。PD 元件在反向模式下工作，吸收可見光、以通過的光子撞擊效應產生電流；然後，通過兩階段 OP 放大器(一個整流 OP 放大器和一個反相 OP 放大器)將光生電流轉換為電壓。最終，經由類比轉數位轉換器（ADC）和具有 I_2C 介面輸出的數位電路，將光強度轉換為數位資料數據，符合 I^2C 串列通訊協議。

圖 2.2(b)展示了一個數位、4 合 1 BME680 感測器，可以測量溫度、濕度、壓力和氣體，其中一個單一的 MEMS 晶片帶有溫度和濕度感測器，在範圍內獲取周圍溫度和 RH 參

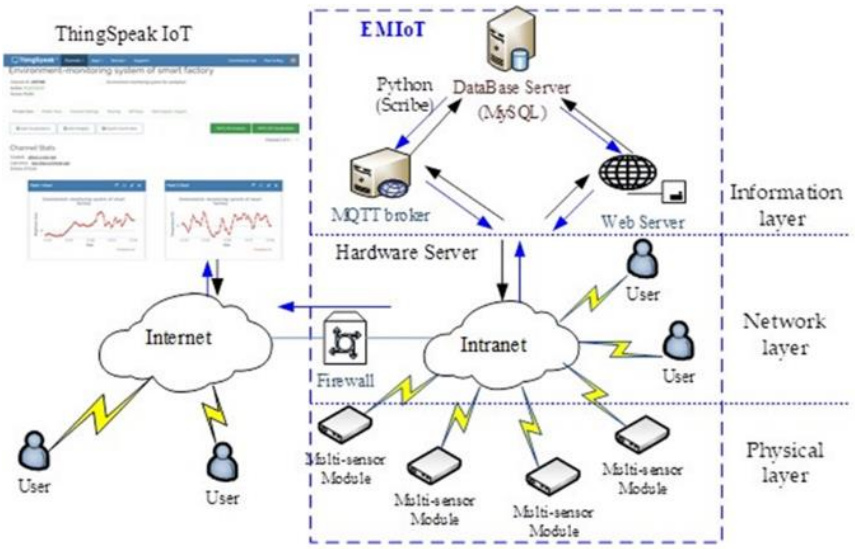


圖 2.1 廠域環境監測感測器整合模組

數，量測範圍分別－40 - 85 ˚C 和 0-100%以內，準確度分別可達到。BME680 感測器：是一款多功能環境感測器。它整合了溫度、濕度、壓力和室內空氣品質（IAQ）感測功能，能夠提供廣泛的環境數據監測在本研究主要使用室內空氣品質（IAQ）感測功能為主。BME680 感測器的主要特點和功能：BME680 能夠測量環境的溫度，提供高精度的溫度數據。溫度範圍通常在-40℃到 85℃之間。可以監測環境的濕度，提供相對濕度的數據。濕度範圍通常在 0%到 100%之間。有壓力感測功能，可以測量環境的氣壓。壓力範圍通常在 300 hPa 到 1100 hPa 之間。有室內空氣品質感測功能，可以檢測揮發性有機化合物（VOC）的濃度。它還提供室內空氣品質指數（IAQ 指數）的估算，用於評估空氣的質量。BME680 感測器在當代環境監控技術中，占據了不可或缺的角色，該裝置在氣候數據的收集、室內環境質量評估，以及智慧家居自動化中，均展現出卓越性能。它結合了溫濕度感測、大氣壓力測量以及室內空氣質量偵測等功能，並利用微電子機械系統（MEMS）技術，不僅提升了感測器在體積和能源消耗上的效率，同時也確保了在各種環境中，感測數據的準確性和可靠性。

基於 MIE 散射理論，PMS7003T 顆粒濃度感測器在操作溫度下，測量範圍和精度採用激光照射空氣中的懸浮顆粒，並產生一定程度的激光散射光，取決於顆粒大小，如圖 2.2(c)所示。懸浮微粒感測器採用雷射光散射原理。即令雷射光照射在空氣中的懸浮顆粒物上產生散射，同時在某一特定角度收集散射光，得到散射光強隨時間變化的曲線。進而微處理器利用基於米氏（MIE）理論的算法，得出顆粒物的等效粒徑及單位體積內不同粒徑的顆粒物數量。如此一來便可求得入射光通過待測濃度的相對衰減率。而相對衰減率的大小基本上能線性反應在待測灰塵的相對濃度。光的強度大小和經光電轉換的電信號強弱成正比，通過測得電信號就可以求得相對衰減率，進而就可以測定灰塵的濃度。

如圖 2.2(d)所示，本研究所採用 Gravity IIC O_2（氧氣（感測器係基於電化學原理，其有效範圍是 0~25%Vol，分辨率可達到 0.15%Vol，工作電壓範圍：3.3 至 5.5 V_{DC}。能夠準確且方便地測量周圍的 O_2 濃度，具備高抗干擾能力、高穩定性和高靈敏度；這個 O_2(氧氣)感測器與 Arduino 各式平台相容，可以廣泛應用於便攜式設備、空氣品質監測設備以及工業、礦山、倉庫和其他空氣不易流通的場所。這款緊緻的 DFRobot O_2(氧氣)感測器支持 I^2C 輸出，可以在空氣中校準，

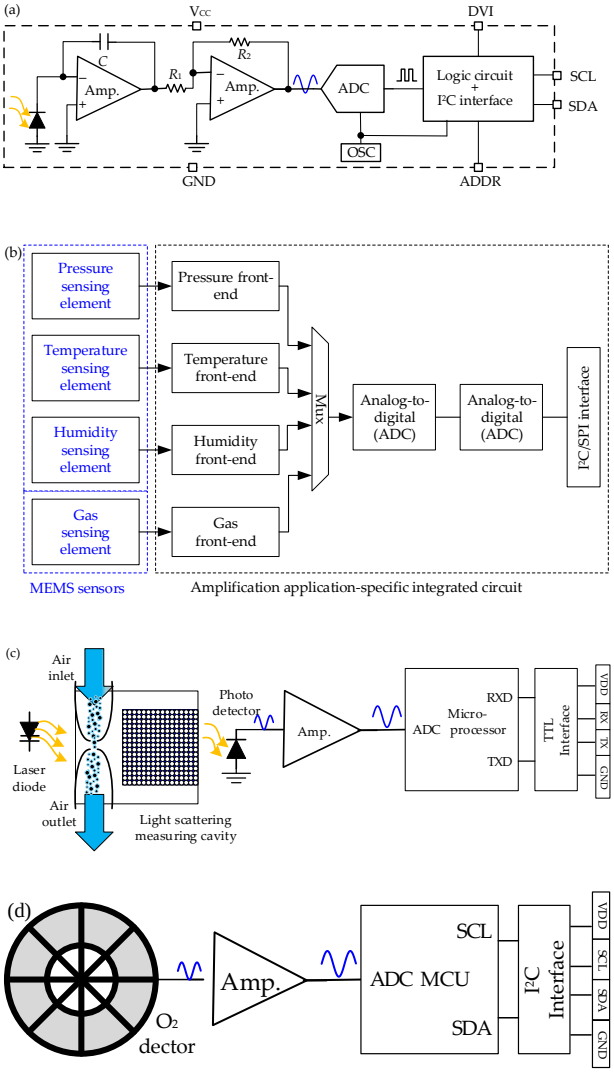


圖 2.2 多元、異質感測器方塊圖

(a) BH1750; (b) BME680; (c) PMS7003T; (d) Gravity IIC O_2

能準確測量環境中的氧氣濃度。它與多種主板兼容，如 Arduino Uno、ESP32/8266、Raspberry Pi 等。

（二）WeMos D1 Mini Wi-Fi 模組

網路是現代人生活中不可缺少的部分，網路不但協助將資料遠距離傳輸，還能夠進行通訊等不同的功能；而 WeMos D1 Mini Wi-Fi 模組最主要的功能，就是可以使設備連接到物聯網，將設備裡的數據資料，能夠透過無線網路傳輸到特定的儲存位置。所以 WeMos D1 Mini Wi-Fi 模組實現了「環境監測智慧物聯網系統」中，將所有感測器連接上網路，透過 WeMos D1 Mini Wi-Fi 模組的運作，能成功的將各種感測器所蒐集到的感測數據，上傳至 IoT 平台進行線上監測。WeMos D1 Mini Wi-Fi 模組如圖 2.3。

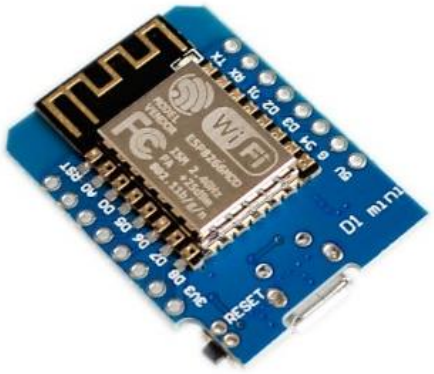


圖 2.3 WeMos D1 Mini Wi-Fi 模組

（三）ThingSpeak IoT 平台

結合物聯網是現在社會的趨勢，我們將監測的數據上傳至 ThingSpeak IOT 平台，方便我們可以即時監控當下的感測數值，在數值不穩定時，可以即時採取應變方法的措施，讓數值恢復至穩定狀態，適合人們活動的標準數值。

（四）MQTT 平台

MQTT 起初是由 IBM 公司正在開發一個物聯網項目，是需要一種輕量的二進制通訊協定，以實現設備之間的通訊。因此，由 IBM 的 Andy Stanford-Clark 及 Arcom 公司的 Arlen Nipper 於 1999 年開發了此協定的第一個版本，並在 2001 年開源，隨後在 2014 年被 OASIS 標準化。

MQTT（Message Queuing Telemetry Transport）的設計目的，是讓設備能夠透過網路傳送小型數據包，控制訊息最小只有 2 位元組的資料，將訊息用最精簡的方式傳送出去，並且使用較少的網路頻寬和電力。由於 MQTT 具備可靠性高、可擴展性強、網路傳輸效率高……等特點，故廣泛應用於各種場景中，例如物聯網、Facebook Messenger 即時通訊、遠程控制等。

HTTP（超文本傳輸協定）是一種客戶端 / 伺服器端的通訊協定，使用請求 / 回應模式進行通訊。當用戶端發送請求給伺服器時，伺服器會回應一個包含狀態碼和資料的回應。HTTP 通常用於網頁和 API 傳輸，並且設計為一種無狀態協定，每次請求都是獨立的，不會保留任何狀態資訊。

MQTT 與 HTTP 的共通點有以下幾點：（1）使用 TCP/IP 協定來傳輸資料、（2）用於實現客戶端和伺服器之間的通訊、（3）傳輸二進制或文本數據，支援不同的傳輸格式。而相異點的部分以表 2.1 所示。

表 2.1 HTTP 與 MQTT 相異點比較

	HTTP	MQTT
用途	Web 服務器之間的通訊	IoT 設備之間的通訊
傳輸方式	請求 / 回應模式	發布 / 訂閱模式
安全性	HTTPS（超文本傳輸安全協議）	SSL / TLS 和其他安全機制
可靠性	需實施附加組件 (TCP)	基於 QoS，完全可靠
可傳輸資料量	較小	較大
佔用頻寬	較大	較小

1999 年 IBM 提出第一版本的 MQTT 協定時，他們設計了一個基於「發布 / 訂閱」模式的訊息規範，來滿足簡單、網路不穩定、輕量級的需求。接下來將說明 MQTT 的「發布 / 訂閱」模式、主題（Topic）名稱以及服務品質（QoS）設定。而 MQTT 是採用 Publish / Subscribe 模式，Publisher（發佈者）發佈一個 Topic（主題）的訊息，Subscriber（訂閱者）可以透過 Subscribe（訂閱）該主題來接收相關的訊息。當 Publisher 發佈一個新的訊息時，Broker（中間人）會將該訊息轉發給所有訂閱了該主題的 Subscriber。而 Subscriber 則可以通過訂閱不同的主題來接收不同的訊息，也可以透過 Unsubscribe（取消訂閱）某個主題來停止接收相關的訊息。以圖 2.4 所示。

MQTT 的 Topic 是萬國碼 UTF－8 編碼的字串，階層（level）數量沒有限制，但不能超過 65,535 個位元，主題階層之間需使用斜線 “ / ” 分隔。

QoS 是 MQTT 中用於控制訊息傳輸質量的機制，MQTT 定義了三種不同的 QoS 等級：QoS 0、QoS 1 和 QoS 2。QoS 0 為最多傳輸一次，且不保證訊息能夠成功傳出；QoS 1 為至少傳輸一次，可以確保消息能夠到達接收端，但可能會重複傳輸；QoS 2 為確實傳輸一次，傳送訊息時會分成兩段式，並保證消息能夠到達接收端，但也可能導致較高的延遲。

總體來說，MQTT 的訊息規範非常靈活和容易擴展，可以應用於各種不同的場景。MQTT 的主題（TOPIC）和品質

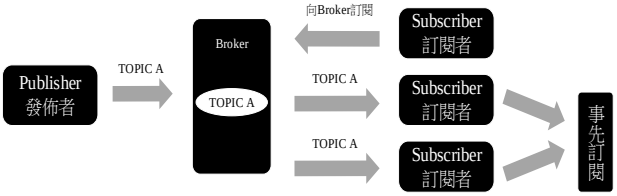


圖 2.4 MQTT Publish & Subscribe 模式示意圖

（QoS）設定使得訊息傳輸非常高效，同時也讓 MQTT 易於實現和部署。

而 MQTT 協定是一個 OASIS（Organization for the Advancement of Structured Information Standards）標準，MQTT 好處在於它是一種輕量級協議為了物聯網而設計的協定。

MQTT 採用訂閱/推送機制，分為伺服器與客戶端兩個角色，由代理伺服器擔任所有客戶端之間溝通的橋樑，任一個客戶端都需要透過代理伺服器傳遞訊息給其他客戶端。客戶端又包含訂閱者及推送者兩種角色，推送者可向代理伺服器推送訊息，訂閱者可向代理伺服器訂閱，當推送者推送訊息時，訂閱者便會收到推送者推送的訊息。

MQTT 資訊的傳輸透過主題進行管理，主題可以依照需求自定義，當任何推送者向代理伺服器的特定主題推送訊息時，代理伺服器會向訂閱此主題的客戶端推送訊息。訂閱者及推送者皆不需要知道彼此的資訊，透過代理伺服器即可完成訊息傳遞，可有效的降低各個應用程式之間的耦合度。

MQTT 訊息格式包含三個部分，分別是固定格式封包、變動格式封包以及訊息內文。

如表 2.2 所示，包含訊息種類（Message Type）、重複訊息標記（DUP）、傳輸品質（QoS）、保留標記（Retain）及剩餘長度（Remaining Length）。訊息種類由 4 個 bits 組成，以二進位表示法可以得到 16 種不同的類型，訊息種類如表 2.3 所示。

（五）MySQL

MySQL 是一種開源的數據庫、資料庫管理系統，廣泛應用在中小型的網站中，用來配合如 PHP、ASP 或 ASP.NET 等網頁程式語言，儲存大量數據，若網站擁有後端管理程式系統（網站後台），多須配合資料庫功能。

資料庫是用來放置大量資料與檔案的一個倉庫，SQL 是跟網站倉庫溝通的管員，而 MySQL 是用來管理倉庫的系統。如此，我們可以透過 MySQL 系統請 SQL 與倉庫溝通，管理資料庫中的資料本論文用 MySQL 儲存所有感測器收集到的數據。

（六）ASP.NET Core MVC 網頁開發框架

ASP.NET Core MVC 是一個用於開發 Web 應用程序的開發框架。它是由微軟開發的，是 ASP.NET 框架的下一代版本。本論文採用 ASP.NET Core MVC 網頁開發框架係其具有跨平台支持、高性能和可擴展性、現代化的開發體驗和開放源碼等優點，可以使開發人員更輕鬆地開發高效、可靠、可擴展的 Web 應用程序。

表 2.2 MQTT **固定格式封包架構表**

bit	7	6	5	4	3	2	1	0
Byte1	Message Type				DUP	QoS		Retain
Byte2~5	Remaining Length							

表 2.3 MQTT 訊息種類

名稱	二進位	十進位	說明
Reserved	0000	0	預留
CONNECT	0001	1	向 Server 發送連線請求
CONNACK	0010	2	連線確認
PUBLISH	0011	3	推送訊息
PUBACK	0100	4	推送確認
PUBREC	0101	5	收到推送訊息
PUBREL	0110	6	釋放推送消息
PUBCOMP	0111	7	完成發佈推送消息
SUBSCRIBE	1000	8	訂閱
SUBACK	1001	9	訂閱確認
UNSUBSCRIBE	1010	10	取消訂閱
UNSUBACK	1011	11	取消訂閱確認
PINGREQ	1100	12	Ping Request
PINGRESP	1101	13	Ping Response
DISCONNECT	1110	14	中斷連線
Reserved	1111	15	預留

ASP.NET Core MVC 使用 Model-View-Controller（MVC）架構模式，這種模式將應用程序分成三個主要組件：

Model（模型）：模型代表應用程序的數據和業務邏輯。它負責處理數據存取、驗證邏輯和業務邏輯等操作。

View（視圖）：視圖是用於呈現模型數據給最終用戶的部分。它負責顯示數據，通常是通過使用 HTML、CSS 和 JavaScript 來生成動態的 Web 頁面。

Controller（控制器）：控制器處理用戶輸入和處理流程的部分。它接收用戶的請求，根據請求調用相應的模型處理邏輯，然後將結果返回到視圖進行顯示。

ASP.NET Core MVC 提供了許多功能和工具，幫助開發人員快速構建高效、可擴展的 Web 應用程序。它支援使用 C# 編程語言，並提供了強大的路由、模型繫結、過濾器、身份驗證和授權等功能。

此外，ASP.NET Core MVC 還具有良好的測試支援，開

發人員可以輕鬆編寫單元測試和集成測試，以確保應用程序的質量和穩定性。

ASP.NET Core MVC 是跨平台的，可以在多種操作系統上運行，包括 Windows、macOS 和 Linux。它也與現代 Web 開發的最佳實踐相結合，如支援 RESTful API 的開發、使用前端框架（如 Angular、React 或 Vue.js）進行 SPA（Single Page Application）開發等。

整體來說，ASP.NET Core MVC 是一個功能豐富、高效、可擴展且跨平台的 Web 應用程序開發框架，可以幫助開發人員快速構建現代化的 Web 應用程序。

三、設計與實現

（一）無線感測模組設計與實現

在最初實驗時，我們先以最原始的 Arduino Uno 模組進行電路設計實驗。首先我們分別將不同的感測器連接電路板進行程式設計實驗開始通電後，變數設定初始化為零，再來由溫/溼度感測器、亮度感測器、氧氣氣體感測器進行數據感測收集，收集的數據顯示於序列埠監控視窗與 LCD 顯示器，確認各式多元、異質感測器的工作狀態。 接著，以 WeMos D1 Mini Wi-Fi 模組取代 Arduino Uno R3 平台，進行無線感測器整合模組上網作業。圖 3.1(a)為 Arduino Uno 模組元件接線圖與實體接線圖，圖 3.1(b)為 WeMos D1 Mini Wi-Fi 模組元件接線圖與實體接線圖。相關程式請參閱：https：//reurl.cc/r6VMGy。

（二）ThingSpeak IoT 平台設計與實現

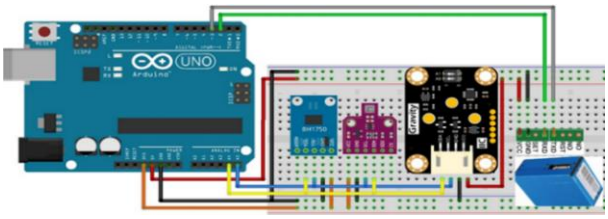
目前，現今社會網際網路科技日漸成熟，本文將感測元件結合物聯網，經由 WeMos D1 Mini 模組，將感測數據上傳至 ThingSpeak IoT 平台，如圖 3.2、圖 3.3 所示，即可透過雲端資料庫監看即時感測數據，不必重回監測現場查看顯示螢幕上的感測數值。結合物聯網是現今社會的趨勢，本文將監測的數據上傳至 ThingSpeak IOT 平台網址（https：//thingspeak.com/channels/2064595），方便即時監控當下的感測數值如圖 3.4 所示，在數值不穩定時，可以即時採取應變方法的措施，讓數值恢復至穩定狀態，適合人們活動的標準數值。

（三）自有雲環境監測平台系統開發設計

1. MQTT 平台實作

在 Da-Yeh AIoT 環境監測系統分成感測器、資料庫、網頁三大區塊，由 WeMos D1 Mini 收集所有感測器的數據經由 Wi-Fi 發送至資料庫，中間經由 MQTT 協定傳送使用者訂閱此頻道就可接收數據如圖 3.5 所示。

(a)



(b)

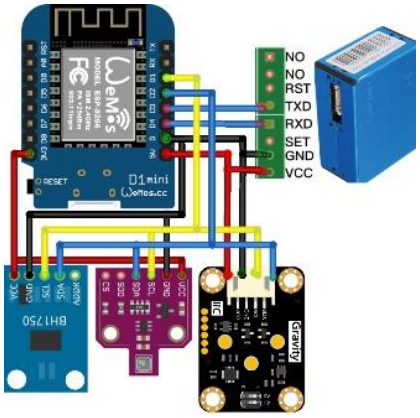
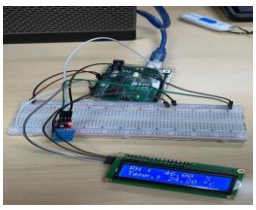
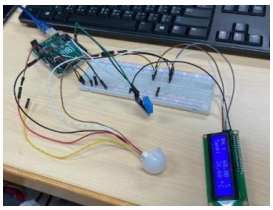


圖 3.1 多元、異質感測器元件接線圖

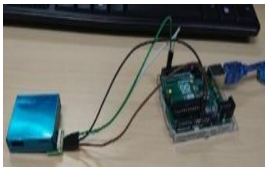
(a) Arduino Uno 模組；(b) WeMos D1 Mini Wi-Fi 模組



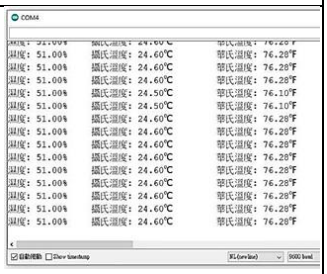
（a）溫度/濕度感測器



（b）亮度感測器



（c）$PM_{2.5}$ 感測器



（d）序列埠監控視窗

圖 3.2 連接 Arduino Uno 平台端實作照片

```
#include <Wire.h>
#include <BH1750.h>
#include "DFRobot_OxygenSensor.h"
#include <PMS.h>
#include <SoftwareSerial.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include <ESP8266WiFi.h>
char ssid[] ="H715";//"Xperia 10 II_7109";//"H715";//"MichaelTsai";//"LAPTOP-PRV61I67 1706"; //"R
char password[] = "CSIEH715";//"11111111";//"CSIEH715";// "Mike810116";//"mypassword";//
//請修改為你自己的API Key
const char* server = "api.thingspeak.com";
String apiKey = "HSUCNHKVWL8RUV2C";// "GYCJ86OBI7YID2YK";//"Q53ZSBKMFYDQ2T2V";//"27MIVN1ZZKE9VLOA
#define COLLECT_NUMBER    10             // collect number, the collection range is 1-100.
#define Oxygen_IICAddress ADDRESS_3
/*   iic slave Address, The default is ADDRESS_3.
       ADDRESS_0               0x70      // iic device address.
       ADDRESS_1               0x71
       ADDRESS_2               0x72
       ADDRESS_3               0x73
*/
```

圖 3.3 WeMos D1 Mini 模組連接 ThingSpeak IOT 平台

圖 3.4 ThingSpeak 平台監測頁面

網頁使用 ASP.NET Core MVC 開發框架去做開發，MVC 分成三大功能方便維護分工如圖 3.6 所示。

2. 大葉環境監測平台設計與實現

本研究在伺服器上架設一個 AIoT 資料庫其內容如圖 3.7 所示，包含 Brightness、Temperature、Humidity、Oxygen_concentration、$PM_{2.5}$、$PM_{1.0}$、IAQ、IAQ Level 等八種資料表。

每個資料表都表示一個感測器，其欄位包含了資料型態為 int 的 ID、資料型態為 decimal 的 sensor_data、資料型態為 timestamp 的 timestamp，各代表意義分別為感測器數據是第幾筆資料的編號、感測器的監控數據、數據的時間。

因數據是由 MQTT 發送感測器監控數據無法直接儲存進資料庫，所以利用 Python 撰寫一個目的是為了將監控數據傳送至資料庫中如圖 3.8 所示。

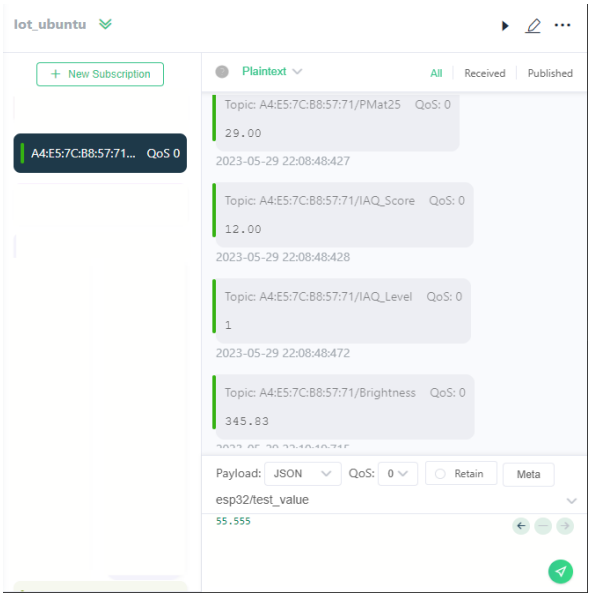


圖 3.5 MQTT 傳送監控

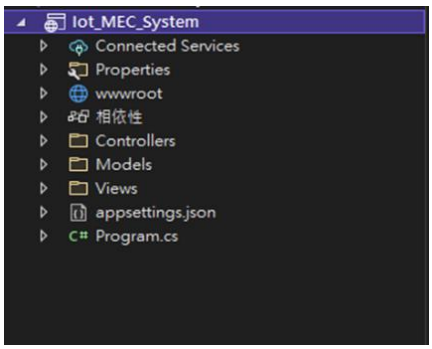


圖 3.6 ASP.NET Core MVC

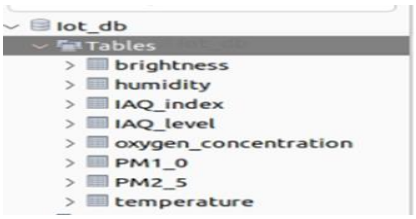


圖 3.7 IoT 資料庫

本文也使用 ASP.NET MVC 設計顯示及時數據網頁，而 MVC 架構分成 Model、View、Controller 三大功能，如圖 3.9 所示。

Model、View、Controller（MVC）的區分允許我們將軟體內部模組化，以便每個模組可以專注於不同的職責。具體而言，這表示我們將不同功能和意義的程式碼放置在不同的檔案中，以實現更清晰的程式結構和管理。

在系統中，我們為每個資料表建立各自的 Model，並透過不同感測器類型分別定義出各自的 Controller，最終透過 Controller 中的 getData（）方法取得各個感測器的資料，將資料存入 SensorDataCollection 類別的欄位中回傳。當使用者透過網頁 ChartView 瀏覽網頁時，ChartControler 將會分別使用各自感測器 Controller 中的 getData（）方法取得各自的 SensorDataCollection 物件，並使用 SensorDataCollectionList 類別將各自感測器回傳的 SensorDataCollection 物件儲存，傳回 ChartView 中，並在 ChartView 中將感測器的數值使用折線圖顯示出來，如圖 3.10 所示。

```python
try:
    # 建立Connection物件
    conn = pymysql.connect(**db_settings)
    # 建立Cursor物件
    with conn.cursor() as cursor:
        # 新增資料SQL語法(INSERT INTO {"table名稱"} ("table存數值的欄位名稱") VALUES({"數值"}))
        command = f"INSERT INTO {table} (sensor_data) VALUES({value})"
        cursor.execute(command)
        # # 儲存變更
        conn.commit()
except Exception as ex:
    print(ex)
```

圖 3.8 利用 Python 轉傳監控數據至資料庫

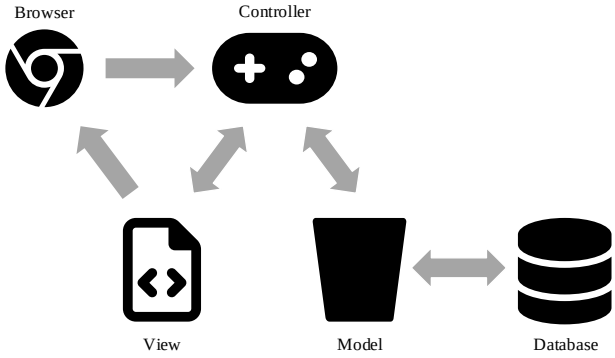


圖 3.9 MVC 架構

在應用程式開發中，模型層負責與資料庫之間的資料溝通和管理。這種區分允許我們在應用程式中執行關鍵操作，例如新增、瀏覽、修改和刪除資料。通常，這些操作需要透過模型層來從資料庫中提取必要的資料，然後將這些資料傳遞給應用程式，隨後啟動 JavaScript 以執行相應的應用程式操作。因此，Model 物件在此過程中扮演著關鍵的角色，用來統一管理資料和資料庫之間的互動，如圖 3.11 所示。

View 層在軟體開發中扮演著「表現層」的角色，其主要負責管理應用程式畫面的呈現，也就是處理 HTML 樣板

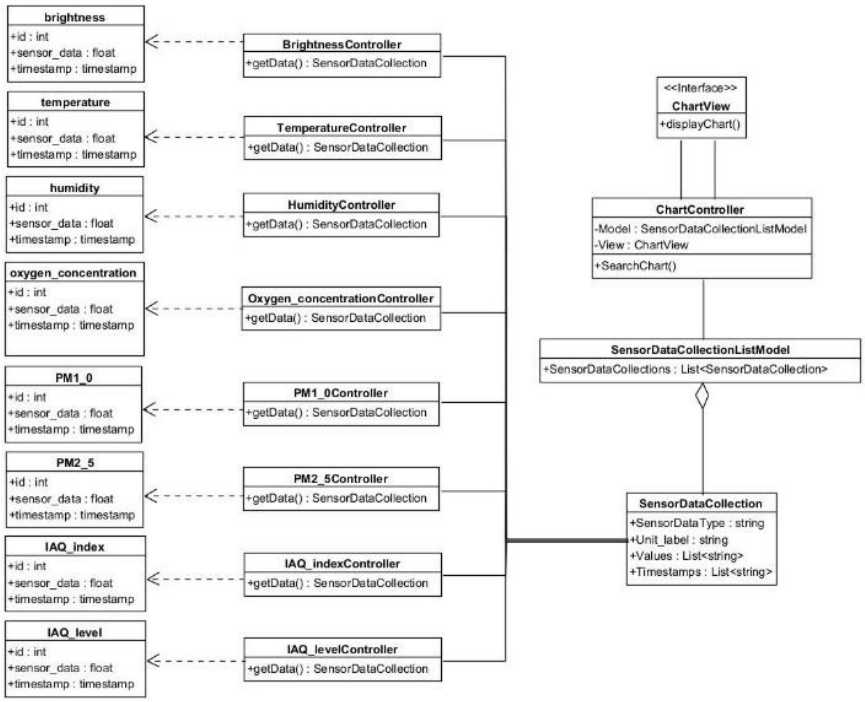


圖 3.10 MVC UML Class Diagram

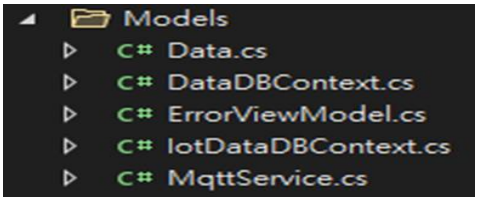


圖 3.11 模型層示例

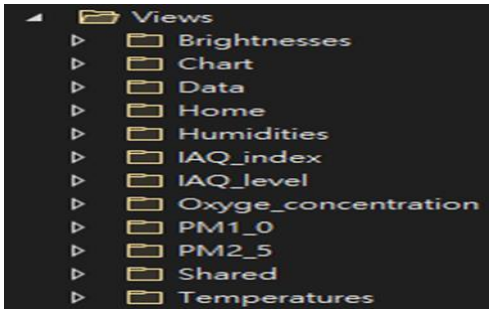


圖 3.12 View 層示例

圖 3.13 Controller 示例

的相關工作。在開發框架中，HTML 樣板通常需要以動態方式顯示資料，這些資料通常是由模型層取得的。因此，View 層會進一步使用樣板引擎來將這些資料嵌入到 HTML 樣板中，使畫面以動態方式呈現，如圖 3.12 所示。

Controller 層負責處理使用者互動的邏輯，同時也是應用程式中處理請求和回應的核心。當來自路由的請求進入應用程式時，它們首先會被傳遞給 Controller 層，然後 Controller 層負責協調模型層以擷取所需的資料，接著將這些資料傳遞給 View 層以生成樣板，最終將包含呈現資料的 HTML 頁面回傳給客戶端。這個過程由 Controller 層來協調和管理，如圖 3.13 所示。

Controller 在 MVC 架構中充當了中介角色，其主要負責決定應用程式的工作流程（Workflow），並收集不同元件的工作成果，將其統一回傳給使用者。在 ASP.NET MVC 設計中，Controller 負責處理顯示即時數據的網頁，維護工作流程，並協調各個元件的互動，以實現有效的應用程式功能。如圖 3.14 所示。

四、實作場域驗證與結果分析

本節介紹本研究使用的軟硬體、運用包含亮度感測、溫/濕度感測、懸浮微粒感測、氧氣感測等不同的感測器來

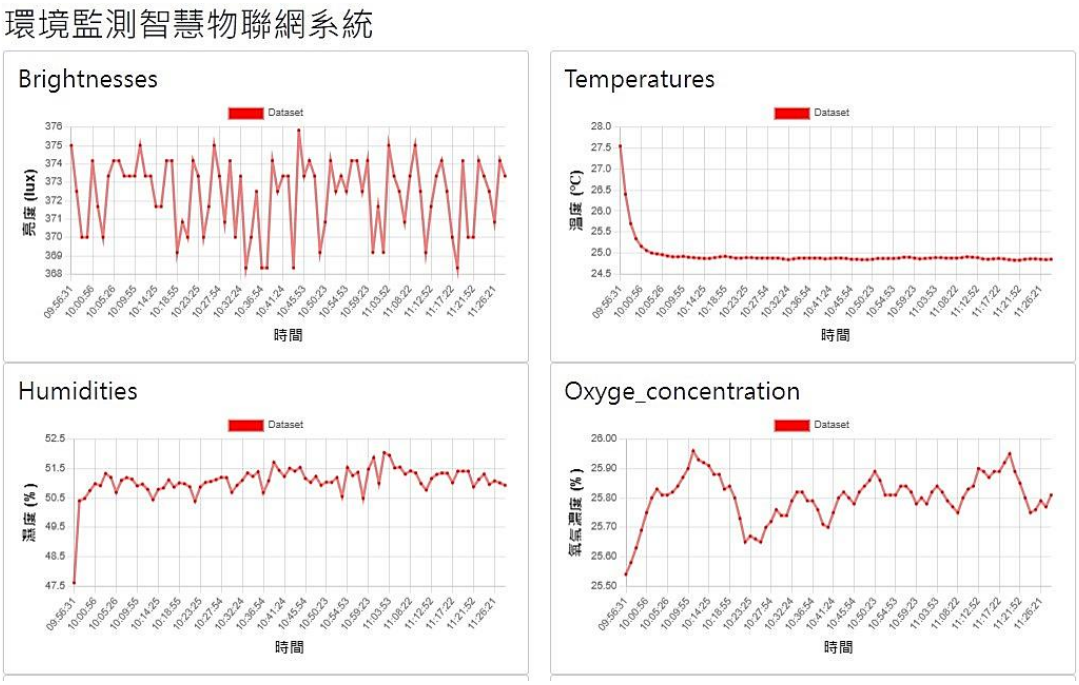


圖 3.14 環境監測智慧物聯網網頁

進行對環境的監測，再利用感測到的數據傳上網路，組合成數據庫，以方便對設備所在的環境進行分析，確保設備的周圍環境適合人們生存。本實驗的場景係以大葉大學工學院 H715 實驗室實作量測場域，做為實驗室的環境品質監測，以確保所有在實驗室的學生、助教及老師的身心健康及提供一個舒適的工作環境。

（一）Arduino Uno 與感測器實作量測

本研究初期使用 Arduino Uno 模組連接各個感測器進行監測環境，使用 LCD 顯示器顯示所屬空間的溫濕度、亮度…等等的數據，如圖 4.1 所示。

（二）WeMos D1 Mini 與 ThingSpeak IoT 平台

在實驗過程中發現，將 LCD 顯示器用於監測數據的顯示存在了一些不便之處，而且無法實現對環境數據偵測的遠端監控。因此，將 Arduino Uno 替換為 WeMos D1 Mini 模組是一個較合適的選擇，因為它可以通過 Wi-Fi 將感測器數據上傳到 ThingSpeak IoT 平台，實現了遠端監控環境數據的目的，如圖 4.2、圖 4.3 所示。

（三）Da-Yeh AIoT 環境監測系統

本研究採用了 ASP.NET MVC 和 MySQL 作為 Da-Yeh AIoT 環境監測系統的前後端技術基礎。以 ASP.NET MVC 架構，將系統劃分為 Model、View 和 Controller 三大模組，以便於後期開發與維護。至於資料庫選擇，基於現行系統運行環境為 Ubuntu Server，同時考慮到 MySQL 的相容性和以往的使用經驗，因此選擇了 MySQL 作為資料儲存的資料庫，如圖 4.4、圖 4.5 所示。

圖 4.1 Arduino Uno 模組

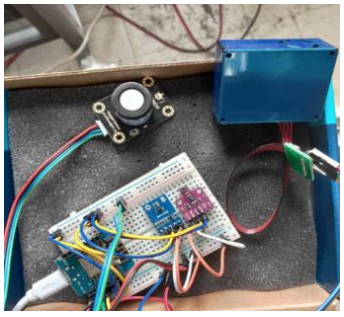


圖 4.2 WeMos D1 Mini Wi-Fi 模組

圖 4.3 ThingSpeak IoT 監測頁面

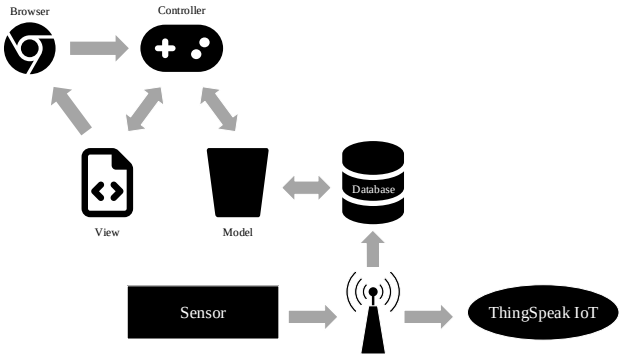


圖 4.4 Da-Yeh AIoT 環境監測架構圖

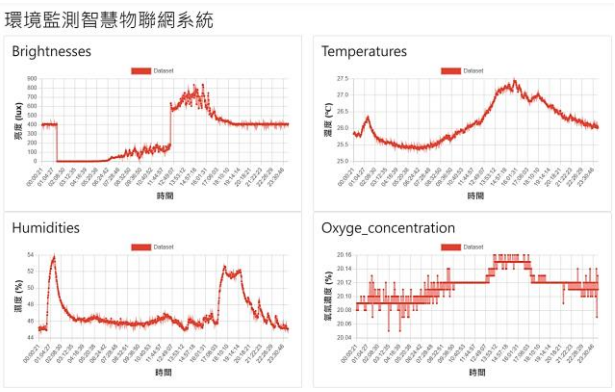


圖 4.5 環境監測智慧物聯網系統

1. 網頁設計

當使用者開始瀏覽網頁時，會透過 ChartView 瀏覽網頁，網頁會向 ChartController 請求圖表需要顯示的資料，ChartController 會分別向各個感測器的 Model 取得各個感測器的數值，當 Model 回傳數值後，將使用 CharController 儲存，最終將收集完成的感測器資料傳回 ChartView，並在網頁上繪製出各自感測器的折線圖，展示感測器收集之數值隨時間的變化趨勢，使用者可輕鬆地透過折線圖的變化，閱讀生硬的資料，並能夠進行更深入的分析和決策達到資料可視化的目的，如圖 4.6 所示。

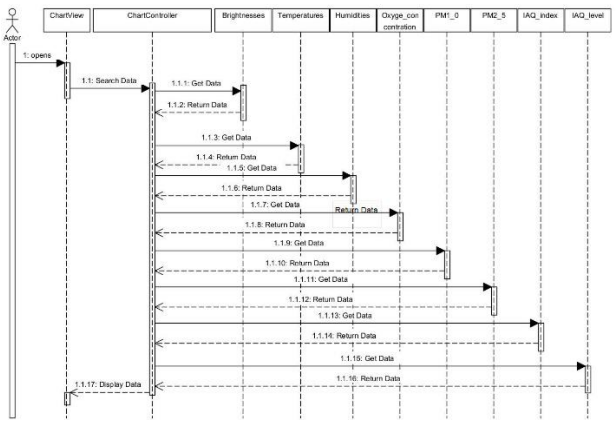


圖 4.6 MVC UML sequence diagram

Model 主要用於與資料庫中的資料表進行資料交換，如圖 4.7 所示。在圖中，Model 與資料庫對應的綠色框線表示了資料庫中的資料表名稱，而紅色框線則對應了資料表中的欄位以及其資料類型，如圖 4.8 所示。

如圖 4.9 所示，Model 主要負責將各個感測器的數值以及時間傳遞給圖表視圖。在圖中，紅色框線表示了該 Model 的名稱，綠色框線則表示感測器的名稱，通常與 Controller 的名稱相同。黃色框線代表感測器的數值，而紫色框線表示感測器數值的時間軸。

如圖 4.10 所示，這是多重圖表 View 的 Model，其主要任務是收集各個感測器的數值和時間資料，並在 Controller 中將其傳遞給圖表視圖。在圖中，紅色框線表示了該 Model 的名稱，綠色框線則表示該 Model 將單一圖表 View 的 Model 組成一個列表（List），以便於將其傳遞給多重圖表 View。Controller 主要用途為控制資料從資料庫到 view 的過程，資料方向：資料庫→ Model→Controller→View。

在「/Controllers/BrightnessesController.cs」紅框為將資料庫相依性導入 controller 中，以便存取資料庫資料使用，如圖 4.11 所示。

如圖 4.12 所示，為所有資料排列成列表的 Action 紅框處為存取資料庫中 Brightness 資料表的方法。

如圖 4.13 所示，紅框中表示了讓圖表只顯示當天監測數值，所以要以當天日期範圍查詢數據資料。

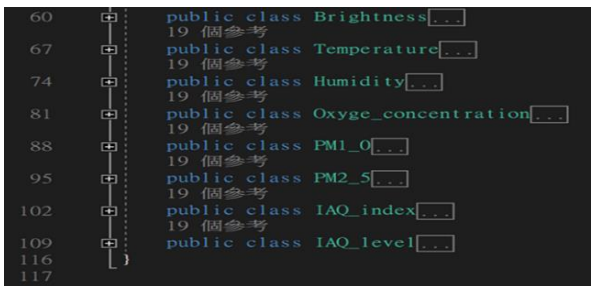


圖 4.7 Models Data.cs

圖 4.8 Model 與資料庫

圖 4.9 單一圖表 View 的 Model

圖 4.10 多重圖表 View 的 Model

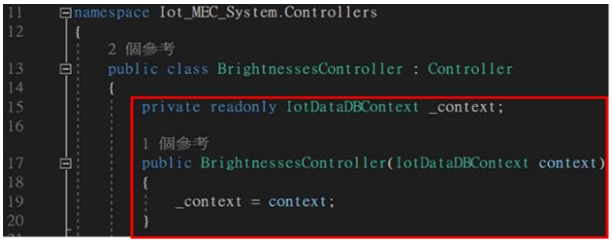


圖 4.11 Controller（Brightnesses）

圖 4.12 Controller 中存取資料庫資料

圖 4.13 Contoller 提供圖表資料的動作

「./Controllers/ChartController.cs」中，多重圖表最終要顯示資料的地方在 MutiChart 的 Action。

使用 getData（）取得所有感測器的數值、時間以及資料，再以多重圖表 View 的 Model 組合成 List，導入多重圖表的 view，如圖 4.14 所示。

View 用來顯示使用者看到的頁面如圖 4.15 所示是多重圖表。綠框使用 partial view 技術（@Html.PartialAsync( )），讓圖表可以重複利用，紅框為排版程式示例，如圖 4.16 所示。

Partial view 在「./Views/Shared/_ChartPartialNew1.cshtml」中，使用 javascript 產生圖表，綠框會作用於「<canvas id= “XXX” ></canvas>」，紅框單個圖表可以看做一個 Card，如圖 4.17 所示。

資料庫連接字串在「./appsettings.json」定義連接資料庫時，使用到的連接字串。以及定義提供主程式存取連接字串的參數名稱，紅框為參數名稱、綠框為 MySQL 資料庫的連接字串如圖 4.18 所示。

在「./Models/Data.cs」內提供連接資料庫的 Class，紅框為供存取資料庫中各個資料表的各個實體。如圖 4.19 所示。

在「./Program.cs 中紅框為 appsettings.json 中定義的連接字串參數名稱將連接字串 Class 以相依性插入（DI）的方式註冊進主程式中，如圖 4.20 所示。

2. 資料庫設計

架設 Da-Yeh AIoT 環境監測系統 IoT 資料庫 brightness、Temperature、Humidity、Oxygen_concentration、$PM_{2.5}$、$PM_{1.0}$、IAQ、IAQ Level 等八種資料表，如圖 4.21 所示。

```
public async Task<IActionResult> MutiChart()
{
    SensorDataCollectionList sensorDataCollectionList = new SensorDataCollectionList();

    sensorDataCollectionList.SensorDataCollections.Add(await new BrightnessesController(_context).getData());
    sensorDataCollectionList.SensorDataCollections.Add(await new TemperaturesController(_context).getData());
    sensorDataCollectionList.SensorDataCollections.Add(await new HumiditiesController(_context).getData());
    sensorDataCollectionList.SensorDataCollections.Add(await new Oxyge_concentrationController(_context).getData());
    sensorDataCollectionList.SensorDataCollections.Add(await new PM1_0Controller(_context).getData());
    sensorDataCollectionList.SensorDataCollections.Add(await new PM2_5Controller(_context).getData());
    sensorDataCollectionList.SensorDataCollections.Add(await new IAQ_indexController(_context).getData());
    sensorDataCollectionList.SensorDataCollections.Add(await new IAQ_levelController(_context).getData());

    return View(sensorDataCollectionList);
}
```

圖 4.14 多重圖表的 Controller

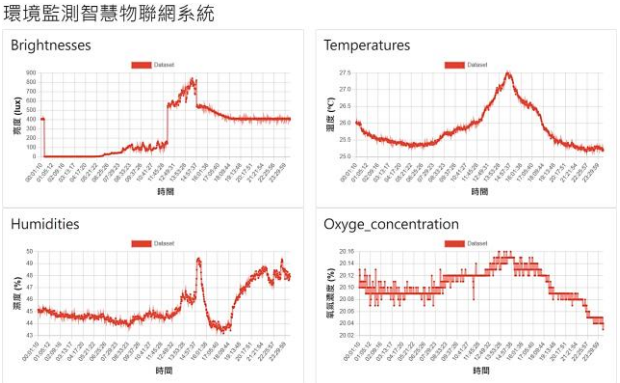


圖 4.15 多重圖表的 View

```
<h1>環境監測智慧物聯網系統</h1>

<div class="row">
    @for (int i = 0; i < @Model.SensorDataCollections.Count; i = i + 2)
    {
        if (@Model.SensorDataCollections.Count % 2 == 0)
        {
            <div class="row">
                <div class="col-sm-6">
                    @await Html.PartialAsync("_ChartPartialNew2", @Model.SensorDataCollections[i])
                </div>
                <div class="col-sm-6">
                    @await Html.PartialAsync("_ChartPartialNew2", @Model.SensorDataCollections[i+1])
                </div>
            </div>
        }
```

圖 4.16 多重圖表排版的 View

```
<div class="card">
    <div class="card-body">
        <h3 class="card-title">@Model.SensorDataType</h3>
        <canvas id="@Model.SensorDataType"></canvas>
    </div>
</div>
```

圖 4.17 Partial View

```
"ConnectionStrings": {
  "IotDataDBContext": "server=xxx.xxx.xxx.xxx;uid=xxxx;pwd=xxxxxx;database=Iot_db"
}
```

圖 4.18 連接資料庫

```
public class IotDataDBContext : DbContext
{
    0 個參考
    public IotDataDBContext(DbContextOptions<IotDataDBContext> options)
        : base(options)
    {
    }
    13 個參考
    public DbSet<Brightness> brightness { get; set; }
    13 個參考
    public DbSet<Temperature> temperature { get; set; }
    13 個參考
    public DbSet<Humidity> humidity { get; set; }
    13 個參考
    public DbSet<Oxyge_concentration> oxygen_concentration { get; set; }
    13 個參考
    public DbSet<PM1_0> PM1_0 { get; set; }
    13 個參考
    public DbSet<PM2_5> PM2_5 { get; set; }
    13 個參考
    public DbSet<IAQ_index> IAQ_index { get; set; }
    13 個參考
    public DbSet<IAQ_level> IAQ_level { get; set; }
}
```

圖 4.19 連接字串 class

```
//資料庫相依性插入
builder.Services.AddDbContext<IotDataDBContext>(options =>
    options.UseMySQL(builder.Configuration.GetConnectionString("IotDataDBContext")));
```

圖 4.20 註冊連接字串進主程式

3. 量測結果

本論文所有的感測器在配置前均以商用儀表校準，如 TES-1337B 高度計（light intensity meter）、TES-1364 溫度/濕度（temperature and humidity meter）、TES-1370H 二氧化碳濃度計（CO2 meter）、TES-1370H 粉塵計數器（particle

counter）及 TES-5321A 空氣品質計（IAQ meter），經確認無誤後再進行實作場域量測。Da-Yeh AIoT 環境監測系統運用不同的感測器對環境進行監測，再利用收集的數據進行整合分析，成功完成了對環境的監控，一旦環境發生重大變化，監控網頁第一時間就會顯示出來，以便於人們處理問題，方便尋找場域出現什麼問題。以下為我們量測 24 小時數據的折線圖與分析。

由圖 4.22 數據圖得知，早上 6 點太陽出來，亮度漸漸上升，由於感測器是放在室內，光線容易遭到遮擋，所以不太穩定，不過亮度有在持續上升，中午到達了最頂峰，一路持續到了下午 5 點開始下降，突然的走高是因為將實驗室的燈打開形成了光源造成亮度上市，關閉之後，亮度如預期一般下降至 0。

由圖 4.23 數據圖可以得知，濕度在太陽出來之前，濕度幾乎變化，在上午 6 點時，太陽剛出來形成露水，讓濕度有些微上升，不過在太陽完全升起後，濕度就漸漸下降；其中，因為快接近中午時段，室溫過高有開冷氣，中午時段濕度達到量測值最低點，濕度最低量測值持續到下午 5、6 點，太陽下山後，濕度才又漸漸上升。

由圖 4.24 數據圖可以得知，在太陽出來前溫度大致不變，在 6 點太陽出來後開始上升，並在中午到達了頂點，持續到了下午 5 點開始下降，並在太陽完全下山後，趨近穩定。

由圖 4.25 數據圖中得知，監測環境中的氧氣濃度不太有變化，皆保持在約 19~20 Vol%之間保持穩定。

如圖 4.26、圖 4.27、圖 4.28 所示，根據觀察發現，$PM_{1.0}$ 與 $PM_{2.5}$ 以及室內空氣指標這三項數據與濕度成正比。

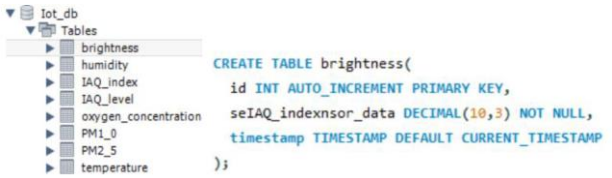


圖 4.21 IoT 資料庫設計

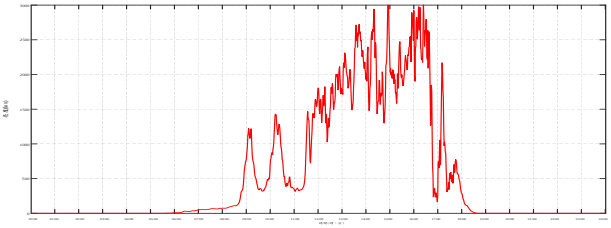


圖 4.22 亮度 24 小時曲線圖

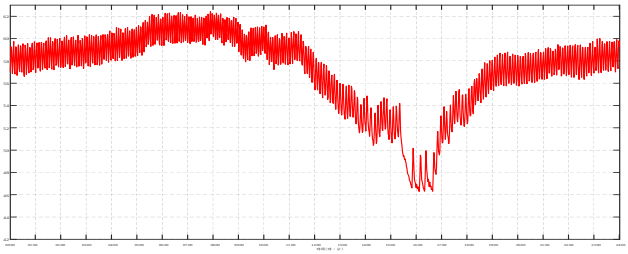


圖 4.23 濕度 24 小時曲線圖

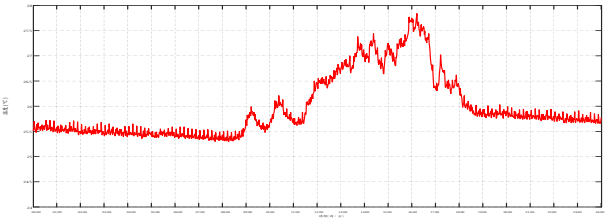


圖 4.24 溫度 24 小時曲線圖

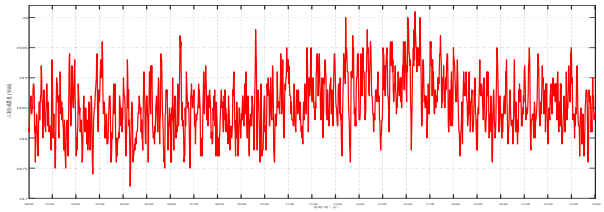


圖 4.25 氧氣濃度 24 小時曲線圖

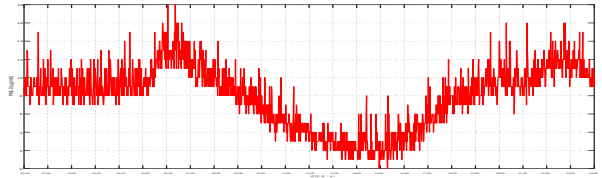


圖 4.26 PM 1.0 24 小時曲線圖

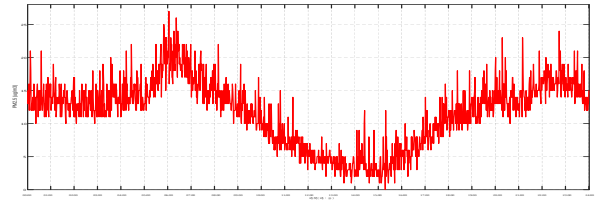


圖 4.27 PM 2.5 24 小時曲線圖

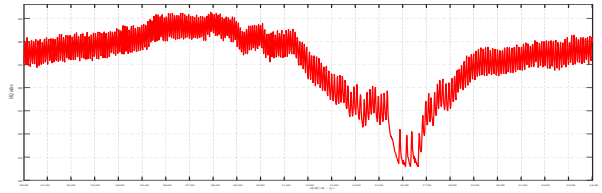


圖 4.28 室內空氣指標 24 小時曲線圖

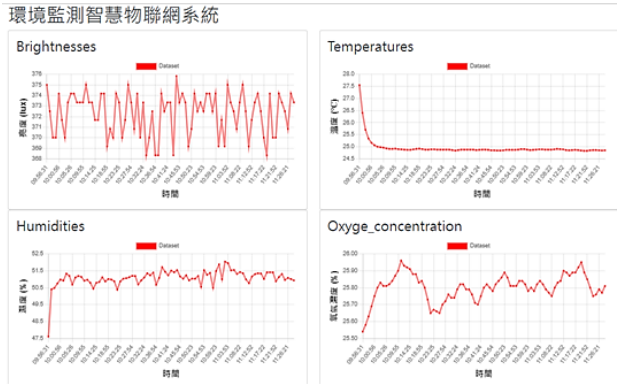


圖 4.29 Da-Yeh AIoT 環境監測系統即時監測圖

以上即時數據圖，皆可在本研究開發的網頁監測做即時查詢，如圖 4.29 所示。

五、結論

環境監測系統的建置方式，使用公有雲或自行開發的私有雲，各自具有一系列優點和待考慮因素，最終還是需要根據特定需求和約束條件來做出適當的選擇。首先，公有雲提供了簡單易用的平台和工具，使得平台建置和管理環境監測系統相對簡單。使用公有雲，可以快速部署環境監測系統，無需投入大量時間和資源來搭建和配置基礎設施。公有雲具有高度的擴展性和彈性，可以根據需要自動調節計算和儲存資源，以適應環境監測需求的變化。此外，公有雲的提供商擁有豐富的基礎設施和數據中心管理經驗，能夠提供穩定可靠的服務和數據備份。然而，使用公有雲也存在一些缺點。將環境監測數據存儲在第三方服務器上可能引起安全和隱私方面的考慮，因此需要仔細評估公有雲提供商的安全措施和合規性。此外，依賴於公有雲服務和平台可能會對環境監測系統造成影響，如果服務商出現故障或服務中斷，可能會對數據的控制和所有權產生憂慮。相對而言，自行開發的私有雲允許完全掌控數據，提供給使用者更高的安全性和隱私保護，並可以根據具體需求進行客製化。然而，自行開發和管理私有雲需要投入更多的時間和資源。

在未來，我們期望環境監測系統將受益於新興技術的發展。物聯網的應用將改善數據收集和傳輸的效率，大數據和人工智能將提高數據處理和分析的能力，雲計算和邊緣運算將提供更多計算和儲存資源，傳感器技術的進步將降低成本並提高可擴展性。這些發展將有助於更好地應對環境監測的挑戰，確保生態平衡和人類健康。

參考文獻

1. Almalki, H. M. (2020) Real-time industrial environment monitoring system design. *International Journal of Scientific & Technology Research*, 9(2), 821-827.
2. Campero-Jurado, I. , J. M. Corchado, J. Quintanar-Gómez, S. Márquez-Sánchez, and S. Rodríguez (2020) Smart helmet 5.0 for industrial internet of things using artificial intelligence. *Sensors*, 20(21), 6241, 1-27.
3. Dobrilovic, D. , D. Perakovic, G. Jotanovic, G. Jausevac, V. Brtka and Z. Stojanov (2021) A model for working environment monitoring in smart manufacturing. *Application Science*, 11, 2850, 1-19.
4. Ghazal, M., A. S. El-Baz, M. Alkhedher, M. Mahmoud, M. Yaghi and T. Basmaji (2020) Cloud-based monitoring of thermal anomalies in industrial environments using AI and the internet of robotic things. *Sensors*, 20, 6348, 1-26.
5. González, E., A. Romero, E. Llobet, J. Casanova-Chafer, J. Mitrovics and X. Vilanova (2020) LoRa sensor network development for air quality monitoring or detecting gas leakage events. *Sensors*, 20, 6225, 1-20.
6. Li, W. and S. Kara (2017) Methodology for monitoring manufacturing environment by using Wireless Sensor Networks (WSN) and the Internet of Things (IoT). *Procedia CIRP*, 61, 323-328.
7. Palazon, J. A., J. Gozalvez, J. L. Maestre and J. R. Gisbert (2013) Wireless solutions for improving health and safety working conditions in industrial environments. Proceedings of the 2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013), Lisbon, Portugal, 544-548.

收件：112.11.28　修正：113.01.02　接受：113.03.19