朝陽科技大學

資訊管理系

碩士論文

實作模糊邏輯控制與智能居家系統

Implement fuzzy logic control and intelligent home systems

指導教授：陳榮靜

研 究 生 ： 丁元昱

中華民國 106 年 7 月 20 日

朝陽科技大學資訊管理系

Department of Information Management

Chaoyang University of Technology

碩士論文

Thesis for Degree of Master

實作模糊邏輯控制與智能居家系統

Implement fuzzy logic control and intelligent home systems

指導教授：陳榮靜(Rung-Ching Chen)

研 究 生 ： 丁元昱(Yuan-Yu Ting)

中華民國 106 年 7 月 20 日

中文摘要

在台灣，由於經濟水平的進步，使得國民平均 GDP 增加、教育水準高。現代人生活型態忙碌，但對於生活的品質要求越來越高。電腦硬體技術與網路速度的進步，使得晶片的成本越來越便宜，促使發展出了物聯網。生活與居住品質的提升，使得智能家電變成近年來非常重要且熱門的研究議題。智能家電除可避免電力的浪費外，也可以針對環境上的差異，來作出判斷並且執行家電控制。本研究將使用樹莓派和藍芽智慧插座來進行實作智能家居系統。使用模糊邏輯，針對環境感測資訊，作出智能家電系統開關的控制。本系統收集環境變數數據，根據室內環境的變數，透過模糊邏輯規則來控制各種家電開關，以自動開啟電子產品，達成智能家電的目標。初步實驗證明，本系統具居家環境資訊的收集、雲端分析與智能開關控制與手機操控功能，可以協助完成居家智能的初步功能。

關鍵詞：樹莓派、模糊邏輯、智慧住宅、藍芽 4.0、物聯網

Abstract

In Taiwan, due to the economic level of progress, making the national average GDP increased and high level of education. , Modern life is busy, but the quality of life requirements for more and more demanding. Due to the computer hardware technology and network speed technology advances, making the cost of the chip more and more become cheap which help to promote the development of the Internet of things. Life and living quality of the upgrade, making smart appliances has become very important in recent years and become the popular research topics. Smart home appliances are not only to avoid the waste of electricity, but also for environmental differences to make judgments and make the implementation of home appliances control actions. This study will use raspberry factions and Bluetooth smart sockets to implement smart home systems. Using fuzzy logic for environmental sensing information makes intelligent home appliance system by switch control to determine the switch for the smart home appliances system. The system collects environmental variable data. According to the indoor environment variables, the system will use fuzzy logic rules to reason the indoor environment variables to control a variety of home appliances switch.

The collected environmental variable data are used to automatically open the electronic products to achieve the goal of smart home appliances. Preliminary experiments show that the system can be used to home environment information collection, cloud analysis, intelligent switch control and mobile phone control home devices. The system can help complete the initial functions of home intelligence.

Keywords: Raspberry Pi、Fuzzy Rule、Smart Home、Bluetooth、Internet of Thing

致謝

感謝朝陽科技大學的陳榮靜老師，在我過去碩士生涯中，老師指引我的研究方向、課業上的學習，在我研究遇到的瓶頸困難也能給予我意見指導，也包容我爸爸在生病的期間能回去照顧爸爸。謝謝老師給我很多的方向

另外還要感謝同樣在朝陽科技大學的戴紹國老師，從大學中老師總是給我程式上指點教導，讓我擁有良好的程式基礎，他給予我產學合作的機會，讓我有機會在社會中實際歷練程式，給予我和廠商溝通需求的學習機會，這種經驗難得可貴，除了在程式上外的指導外，也扮演了人生導師的角色，時時常常建議我該如何做比較好

感謝大學專題的同學，讓我瞭解團隊合作的真諦，即使同學們都在打工、工作，還是很願意配合專題上的大大小小事情的開發工作，也感謝系上有機會讓我們專題院競賽，讓我們有機會參與各系的菁英高手的較量本事。最終取得了第三名頭銜。

感謝我的家人，他們支持我去做任何社會歷練的事情，給予我經濟上莫大的幫助，也讓我了解在外生活的方式，能尊重我讀碩士的決定，並給予我支持協助。

目錄

表目錄

圖目錄

第一章　前言

1.1 研究背景

隨著資訊科技與經濟的進步，科技的進步，使得處理晶片越來越便宜。而社會經濟教育的發展，讓人們越來越願意花錢追求良好生活品質，資訊通訊的進步與網際網路急速的成長下，促使物聯網的發展蓬勃發展。

物聯網可將現實記錄數位化、自動化，發展出了許多的應用領域，主要應用於物流業、健康醫療領域範圍、智慧環境（家庭、辦公、工廠）領域等等。現代人的生活型態改變，現代需要智慧化的居住環境。物聯網的導致智慧住宅的誕生，智慧住宅，利用網路之連結，使各種家庭設備其發揮整體性高效率之服務功能，不但確保居家之安全、居住環境之健康及生活之便利，並提供舒適之生活品質，創造人性化之居住環境。此外智慧住宅也可以減少電能的浪費。不但在經濟上是一大幫助，而且更能達到環保的目的。

1.2 研究動機

針對居家環境的電器裝置做一個更優化的控制，讓居家生活智慧化，利用環境感測器，收集環境數據來分析是否要開啟電器裝置，使用微電腦來操控各種電器裝置，使得家居品質更好，更能智慧化。人們在開啟電器判斷是透過人體的五感與當事人的身體狀況，判斷是否要開啟這個家電，例如人體感受到環境溫度過悶熱，會開啟冷氣的裝置或者開啟電風扇。有時人們會把冷氣開得太強，忽視電扇可以使得空氣的流通，讓整體的空間溫度更平均、讓空氣循環更好，不良的使用電器習慣會造成不必要的電力浪費，電器之間的搭配可以使得生活品質變好。

家居環境中常常人們會忘記要把電器關上，造成電力上的浪費，對於經濟、電力上是不利的影響，因此智慧住宅的發展上，節能的議題也變得相當重要。在工作環境中，人們上班下班後，往往都要開啟關閉電燈，電扇、冷氣之類的動作，如果能讓這些電器智慧化，人們就往往上下班就不用做這些動作，能讓工作環境方便、舒適。

1.3 研究目的

本研究希望使用電腦收集環境數據，利用模糊邏輯規則判斷，依據環境條件來操控藍芽智慧插座並且，透過藍芽開啟家電。此外還希望利用人體感應器收集資料分析統計人們進入室內的時機，可以利用資料事先預測開啟電器裝置，可以知道是否有人在室內中、平常的室內的環境狀況，達到讓電器產品達到全自動化。

1.4 論文架構

論文的架構分為以下五個章節，第一章節為本研究的背景、動機及目的；第二章節將會介紹與智能居家的相關研究，與模糊理論等方法的概述；第三章節為會本論文所提出的智能居家架構與方法；第四章節為智能居家架構的初步實驗結果說明；最後章節將針對論文給予結論及未來研究的方向。

第二章　　文獻探討

本章首先將會針對智能居家系統的研究情況來做介紹與分析來了解各個學者所做的研究，了解本研究相關的技術背景知識，使用的主要方法來做相關的背景知識的概述。智能居家系統早已是被討論許久的議題，如何針對室內的環境判斷出，管理室內環境的狀況，統計出每日環境狀況並從而給建立數據分析，而其中不少國內外學者也提出過相關的研究。本章節分成兩個部分，探討模糊邏輯與智慧宅相關的文獻。

2.1 模糊邏輯

在日常生活中，人們常用一些形容詞來敘述事物，例如：敘述身高用矮或高，在一般人眼中 175 公分的身高算是偏高，則以在於職業籃球員中身高算是矮，這些敘述上則是相當的模糊，而電腦中只有 0 或 1 的方式表示，對於電腦來說只有高或矮，而沒辦法處理模糊的狀態， L.A. Zadeh[11]提出了模糊邏輯的理論，將 0 和 1 二元式的表示轉換為隸屬函數作為模糊的表達。在模糊邏輯成功的結合工業領域，應用於工業上機械的精密控制，對於精準度的誤差控制，在生活中應用於空調的溫度的調整。

模糊邏輯的應用，與各領域的研究結合應用，相當的成功，在資訊安全中，施驊原[1]提出了使用模糊邏輯來以按壓的習慣速度，指壓的力道，來

做為一個密碼系統。Azim Keshtkar[12]提出使用模糊邏輯來控制能源管理應用於智慧電網上，通風和空調系統，可自動讓住宅保持在 24 C 的溫度下，和保持電力上的節省，有效帶來生活上的便利。Sajid Hussain[13]舒適模糊控制優化下調通風和空調系統，使用模糊邏輯與基因演算法結合，預測人體的舒適程度和空調的控制。周劭岳[2]提出應用模糊邏輯評估影響網路購物忠誠度之因素，設立模糊邏輯來評估網路購物的忠誠度，文中提到模糊邏輯的缺點在於無法自己利用數據去學習規則，必須依靠人的經驗去設立規則，也去比較類神經有甚麼不一樣的地方，探討有甚麼因素會影響到忠誠度的關係。陳冠竹[3]提出自主飛行無人定翼機之高性能 GPS 導引模糊邏輯控制器設計，文中主要使用模糊邏輯控制器控制飛行的穩定程度，如有偏差則會自行做調整，模糊邏輯的應用領域相當的廣泛。

2.2 物聯網

物聯網（Internet of Things）[14]是能讓所有能行使獨立功能的普通物體實現互聯互通的網路。物聯網一般是用技術為無線網路作為基礎架構，物與物的連接。在物聯網上，每個人都可以應用電子標籤將真實的物體上網聯結，在物聯網上都可以查出它們的具體位置。

物聯網發展出了許多領域，其中智慧住宅[15]的相關研究數量需多，智慧住宅是這類建築是以居住用途為目的，提供安全便利、健康舒適的居家環境為訴求，以滿足現代生活需求為導向，因此應用科技產業發展的技術，建構因應生活需求的智慧化住宅為現階段住宅建設的必然趨勢。智慧化主要在建構住宅必須提供之生活、便利功能，如安全防災、健康照護、便利舒適、及賴以永續發展之節能減碳功能。而建構這些功能，則須使用通訊科技的技術，透過網通設施平台，進行整合及管理維運，方能發揮具體功效，以提昇居住空間品質，達成住宅智慧化之目的。

智慧住宅是將各種家庭自動化設備，利用網路之連結，使其發揮整體性高效率之服務功能，以確保居家之安全、居住環境之健康及生活之便利，並提供舒適之生活品質，創造人性化之居住環境。智慧住宅也可以減少電能的浪費。對於居家生活、經濟帶來巨大的效益。

Andreas Jacobsson[16]等人提出了智慧住宅會碰到的挑戰和問題，智慧宅的基礎是使用物聯網，則大量裝置會給網路帶來大量負荷，則 Arif Ahmed[17]提出使用 Edge Computing 來減少運算量，文中提到大數據會遇到隱私和安全的問題，通信協定要解決的問題在於耗工、連接數量、傳送速度、距離。文中建議使用 Zigbee 作為互通協定，物聯網容易遭受到無線網路的攻擊，隱私安全需要一個安全機制。

Xinrong Li 提出研究中設計 Raspberry Pi 與 Arduino 無線網路監控環境系統[18]，提出使用 ZigBee 通訊協定，使用 Raspberry Pi 當作主要的控制平台，Arduino 布置在許多房間當作感測器讓 Raspberry Pi 呼叫收集環境數據。架設伺服器可以了解個房間的溫度、濕度，有效了解整層環境資訊。提出了一個監控建築物濕度、溫度的方式，良好的系統架構設計。陳世昕[4]提出物聯網智慧住宅 APP 與藍牙安全實作，提出了防止駭客入侵智慧住宅的方法與實作。

2.4 樹莓派

樹莓派(Raspberry Pi)[19]，是一款 Linux 的單板機電腦。它由英國的樹莓派基金會所開發，目的是以低價硬體及自由軟體促進學校的基本電腦科學教育。樹莓派配備（Broadcom）出產的 ARM 架構 700MHz BCM2835 處理器，256MB 記憶體（B 型已升級到 256 MB 記憶體），使用 SD 卡當作儲存媒體，且擁有一個 Ethernet、兩個 USB 介面、以及 HDMI（支援聲音輸出）和 RCA 端子輸出支援。作業系統採用開源的 Linux。

物聯網的研究中，使用樹莓派當作嵌入式電腦應用，除了本身可以運作作業系統外，本身擁有可以連接無線網路和藍芽晶片，具備了物聯網開發的條件，樹莓派本身各種硬體連接，擁有 USB 介面，可以接上攝影機和鍵盤滑鼠，以利於開發。Vladimir Vujovic[20]提出使用樹莓派作為 Web 感測器

應用在智慧住宅上，當作一個網路節點，把樹莓派當作一個小型伺服器，透過路由器連接到樹莓派，可以透過樹莓派了解房間溫度濕度的狀況。作者提出使用樹莓派的好處，使用 Linux 的作業系統，擁有豐富的開放原始碼可以利用，價格便宜、耗電量小。

柯博斌提出[5]使用樹莓派接上許多感測器，客製化成一個裝置，運算處理，全部交給樹莓派處理，結合雲端系統、Gmail，把輸入後所有的資訊上傳至雲端，可以隨時在網路上瀏覽。江元喻[6]提出使用 Raspberry Pi 與 Arduino 實作之空氣品質系統，使用樹莓派作為控制中心顯示溫度、空氣品質，Arduino 作為收集粉塵的感測裝置。

2.5 環境對人體的影響

根據余建志[7]提出的室內濕度與溫度對舒適度指標影響之研究，研究指出空氣溫度對人體感受有最直接的影響。濕度環境變數主要影響人體的代謝。氣溫適中時，濕度對人體的影響並不顯著。當氣溫較高或較低時，其波動對人體的熱平衡和溫熱感就變的非常重要。當相對濕度超過 80 %時，由於高溫高濕影響人體汗液的蒸發，因而人體會感到悶熱不適。隨著溫度的升高，這種情況將更趨明顯，反之溫度較低時，濕度較高時，人們對除了人們更容易感受到濕冷。人體舒適的溫度介於 20~26 ℃、人體最舒適的溼度介於，40%~60%的區間。

2.6.藍芽智慧插座

藍芽 4.0 另一外特點在於 IOS 手機上可以不用透過 Apple 廠商的晶片認證，藍芽 4.0 有分成「低功耗藍牙」、「傳統藍牙」和「高速藍牙」三種模式，高速藍牙主攻資料交換與傳輸；傳統藍牙則以資訊溝通、裝置連線為重點；藍牙低功耗不需占用太多頻寬的裝置連線為主。藍芽 4.0 版本使用了 AES-128 CCM 演算法加密，資訊安全性是可以保證的。藍芽 4.0 版本中傳輸距離提升到 100 米以上（低功耗模式條件下）[21]。傳統藍芽與低耗電藍芽的比較表格 2-1 所示。

表格 2-1 藍芽技術規格

技術規範	傳統藍牙	低耗電藍牙
無線電頻率	2.4 GHz	2.4 GHz
距離	10 米/100 米	30 米
空中資料速率	1-3 Mb/s	1 Mb/s
應用吞吐量	0.7-2.1 Mb/s	0.2 Mb/s
發送資料的總時間	100 ms	<6 ms
網路拓撲	分散網	Star-bus
耗電量	1	0.01 至 0.5
最大操作電流	<30 mA	<15 mA

根據丁元昱[8]實作 Android 行動裝置操作藍芽智慧插座，利用藍芽 4.0 晶片和控制晶片組成藍芽智慧插座，在使用 Android 行動裝置來控制藍芽智慧插座。藍芽智慧插座是一種物聯網的裝置，此裝置有 RTC(Real Time Clock)裝置紀錄時間，利用 App 來控制裝置的開啟和關閉，可記錄時間在時間點開關或在過了一段時候之後切換開關。

第三章　實作與研究方法

本章節有四個組成要素：研究方法、系統架構、系統建置、模糊邏輯控制器、模擬運算。此章節我們會依次介紹實作模糊邏輯控制與智能居家系統的設計方法。

3.1 研究方法

本研究中，我們需要設計兩張流程圖，以利系統架構建置。因此設計了收集資料、實際使用操控藍芽智慧的流程圖，透過流程圖可以了解到如何去設計系統架構，如何能讓程式順利執行。收集資料的流程圖如圖 3-1 所示，為了要防止不能連到伺服器的情況下，就要在樹莓派本機中記錄文字檔。以利連到伺服器再把當日狀況資料上傳。

模糊邏輯控制流程圖如圖 3-2 所示，因為樹莓派可以直接收集環境變數，硬體運算能力足夠下，直接輸入參數，運算模糊邏輯規則下，並且輸出判斷是否要開啟電器產品。

控制中心-上傳文字檔資料

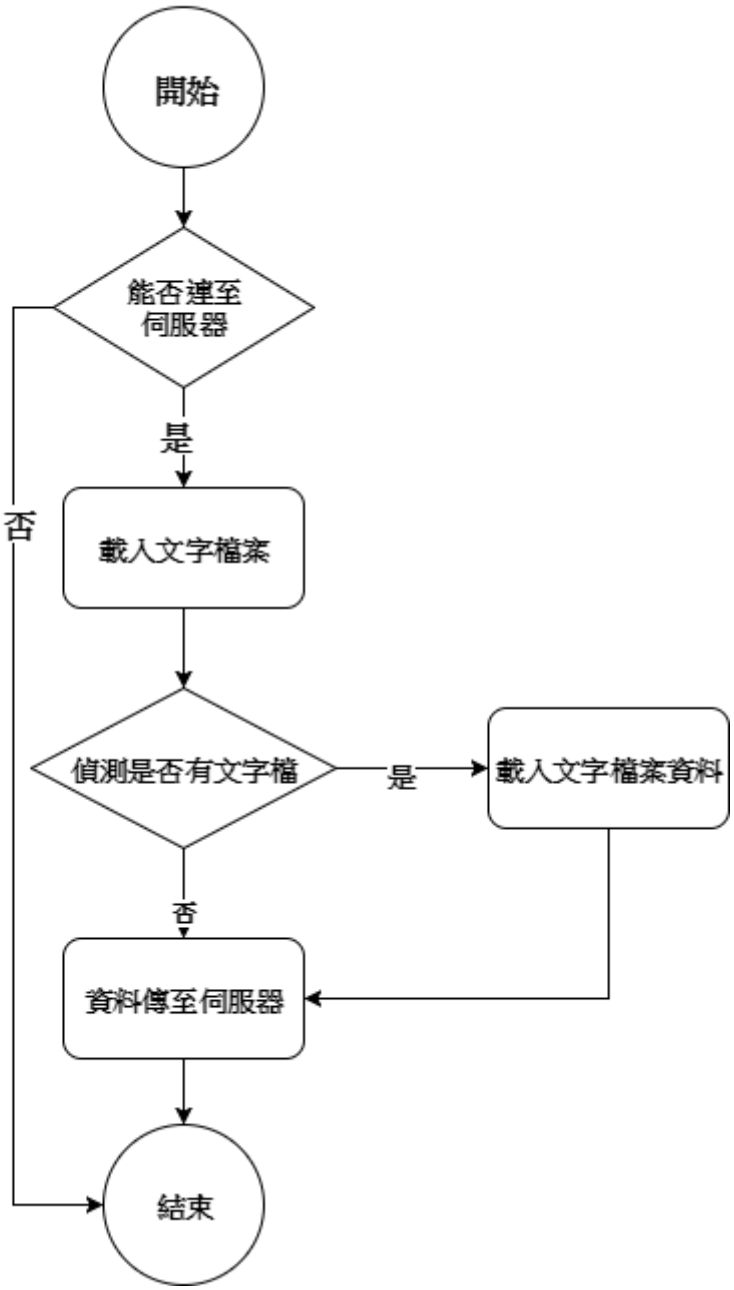

圖 3-1 環境資料紀錄流程圖

控制中心-模糊邏輯控制流程圖

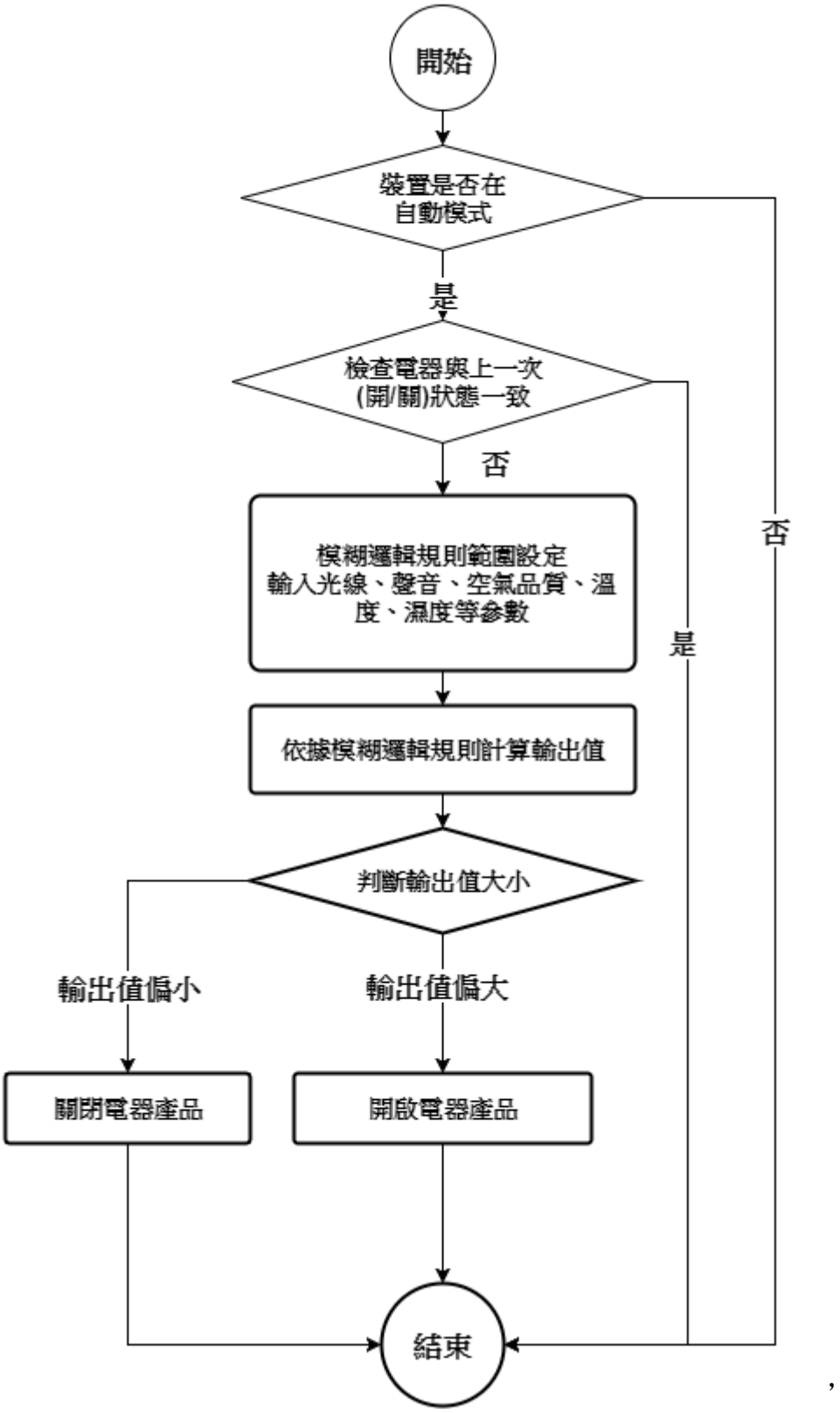

圖 3-2 模糊邏輯控制流程圖

3.2 系統架構

本研究針對室內環境來做數據的收集與分析，以三個元件組成本系統，如圖 3-3 所示，參考了 Leandro Y.Mano[22]提出系統架構，等級高到低，代表的運算程度，等級 2 負責需要高負載的運算工作伺服器主要工作是收集資料、統計資料，收集資料使用指令去呈現我們想要的資料型態。等級 1 是運算量適中的單位，負責接收環境變數和運算模糊邏輯並且發出指令，等級 0 是屬於運算量最小的單位，藍芽智慧插座負責接收指令、等待命令。

使用 Raspberry Pi 3、Grove 感應器、Grove Pi Shield 來當作控制中心，控制所有的藍芽智慧插座，控制中心負責收集環境數據傳送給伺服器。伺服器收集資料以利監控，統計出自己想要的資料。藍芽智慧插座工作是負責各項電器產品的開關。

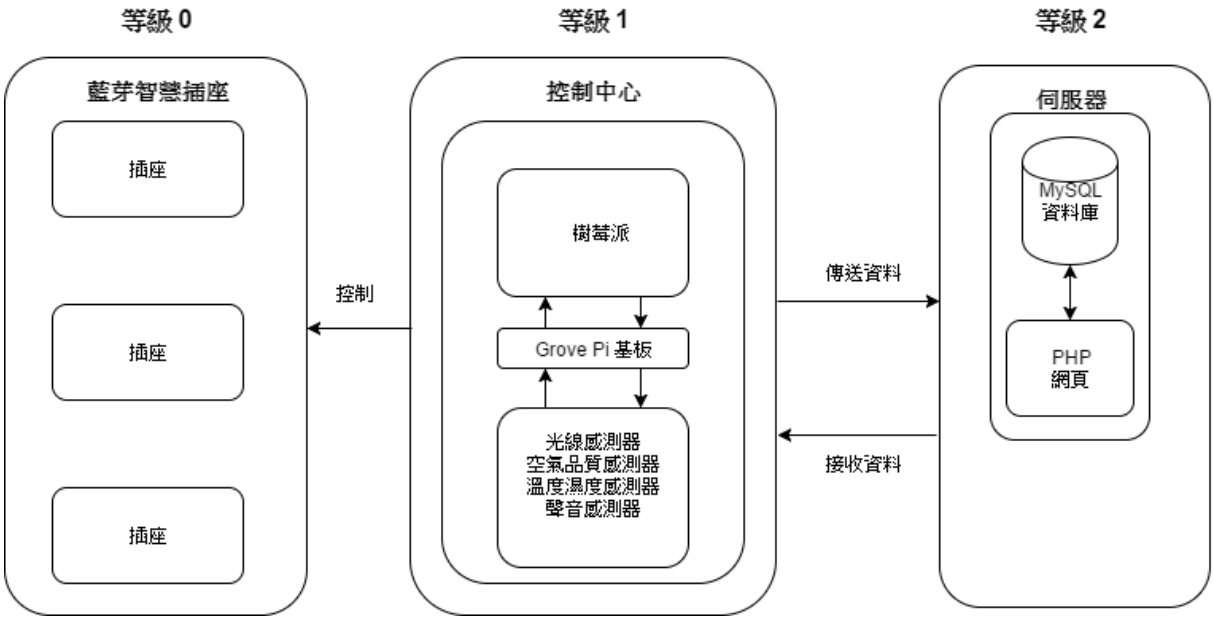

圖 3-3 系統架構圖

3.3 系統建置

本研究建置過程中， 需要準備充足的前置作業，本節將一一說明單元的規格和主要工作。

3.3.1 伺服器

伺服器，此單元的主要工作是收集資料、統計資料，負責需要高負載的運算工作，收集資料使用指令去呈現我們想要的資料型態。伺服器硬體規格為 Intel (R) Core(TM) I7-4790 CPU 3.6 GHz、Ram 22G、256 SSD、1 TB HHD、OS Win 10.0.14393。軟體所使用 Apache 2.4.17、PHP 5.6.16、MySQL 5.7.19、 JpGraph 4.0.2、 WAMPSERVICE 3 64bit 開發。監控環境圖表以平均環境資料以每小時為單位，以當天的數值呈現，方便使用透過網路了解實驗室的環境情況。

3.3.2 控制中心

在本文中實際控制藍芽插座與偵測環境資料、上傳資料的單元命名為控制中心，此單元的工作是，收集環境數據，傳送資料給伺服器。利用 Fuzzy Rule 規則去控制所有的藍芽智慧插座。由三個主要硬體所組成 Raspberry Pi 3、Grove 感應器、Grove Pi Shield。

3.3.2.1 硬體環境架設

樹莓派硬體使用 Raspberry Pi 3 Model B 圖 3-4 所示， 硬體規格為 1.2GHz 64-bit quad-core ARMv8 CPU、802.11n Wireless LAN、Bluetooth 4.1 、1GB RAM、4 USB ports、40 GPIO pins、Full HDMI port、Ethernet port、SD Card 8GB 與 Raspberry Pi 2 差異在於多了 藍芽 4.1、無線網路模組、記憶體 1G、使用 64bit 位元 CPU，其餘硬體規格與一致。軟體則是使用 NOOBS 安裝 Raspbian(Linux 核心)。

圖 3-4 樹莓派 3 B 型主機板

控制中心由 Raspberry Pi 3 B、Grove Pi Shield、Grove 模組三樣組成，沒有適合的盒子，所以必須設計一個盒子能放置硬體。3D 模組設計所使用的軟體為 Autodesk 123D 設計，此軟體特色可以依照自己想要的尺寸輸出 STL 檔案印製 3D 模組與線上雲端備份檔案的功能。依據樹莓派的規格大小規格為(8.5 cm x 5.6 cm x 4 cm) 安裝 Grove Pi Shield 介面後是

(8.5 cm x 5.6 cm x 5 cm)。依據規格大小設計控制中心專用的機殼。機殼設計則是使用長方體的方式設計，機殼厚度均為 0.5 cm，機殼大小為(10 cm x 10 cm x 8 cm)如圖 3-5 所示。而要讓感測器能放置在外面，另外設計了一個機殼專屬的蓋子。在蓋子規格為(10.3 cm x 10.3 cm x 1 cm)，而在上打了 8 個孔洞（1 cm x 1.5 cm），設計孔洞的目的是在方便整理感測器的線，和要讓感測器能與外接環境接觸的狀態。

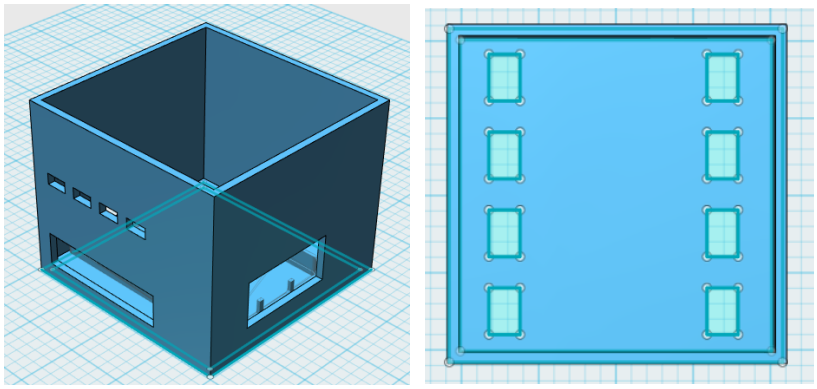

圖 3-5 控制中心機殼設計圖

印製的材料才使用 PLA 的材質印製，印製後則使用使用軟木塞加裝在殼子的上蓋，在使用釘子固定感測器，控制中心實體如圖 3-6 所示。

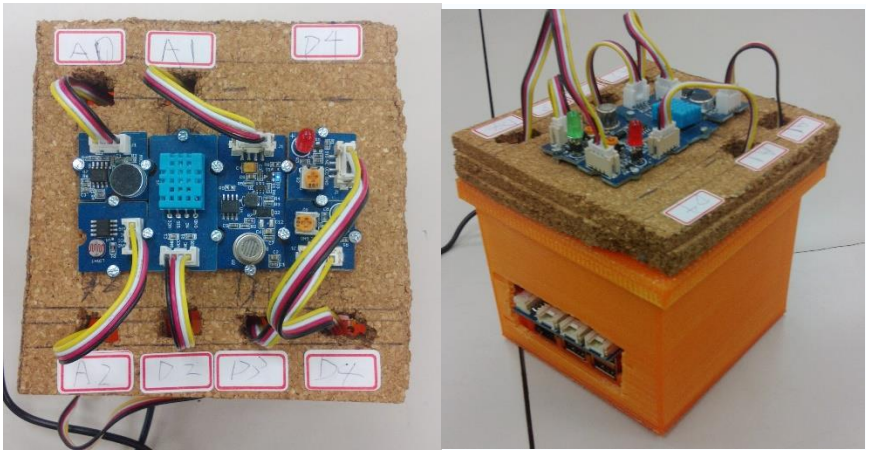

圖 3-6 控制中心實體

表格 3-1 為論文所使用的感測器種類列表，可以從感測器中了解最終會輸出 5 種分別不一樣的輸出值。

表格 3-1 傳感器列表

感測器	說明	輸出值	種類
溫度濕度傳感器	感測一般環境的溫度，不適合用於測量高溫環境與超低溫環境。	20%~90% 0c~50c	數位
聲音傳感器	LM358 放大器和駐極體麥克風，噪音閥值為 400	1~1023	類比
空氣品質感測器	一氧化碳，酒精，丙酮，稀釋劑，甲醛等	1~1023	類比
光線傳感器	GL5528 光敏電阻（光敏電阻）來檢測環境中的光	1~1023	類比

GrovePi Shield 是一種連接 Grove Pi 模組的基板設備，硬體如圖 3-7 所示。無需焊接麵包板，只需插入傳感器。把樹莓派連接到 Grove Pi Shield 上，安裝上 Grove Pi 套件，就可以撰寫程式碼。基板擁有 7 個數位埠和 3 個類比埠，提供 Grove 感測器模組連接。擁有良好的擴充性能。

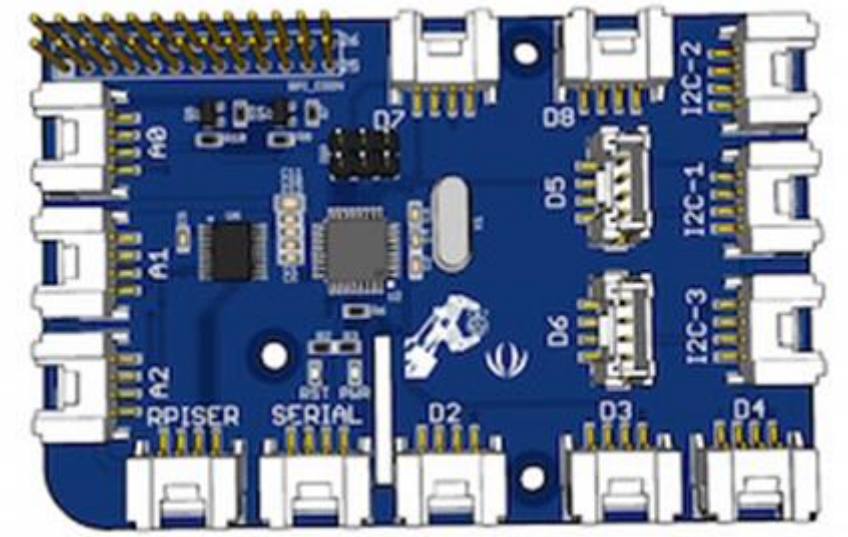

圖 3-7.GrovePi Shield 介面裝置

論文中所使用的感測器都是使用 Grove 感測器模組 ，該模組的特性為可以跨物聯網平台使用，廠商有提供各種平台的基板連接模組和提供範例程式碼，例如：Linkit、Arduino、Raspberry Pi 等平台。

空氣品質感測器[23]設計用於對室內空氣條件進行全面監測，硬體如圖 3-8 所示。能偵測的有害氣體，一氧化碳，酒精，丙酮，稀釋劑，甲醛等。由於測量機制，該傳感器不能輸出特定數據來定量描述目標氣體的濃度。但它仍然足夠能夠用於只需要定性結果的應用程序，如汽車翻新噴霧器和汽車空氣循環系統。類比數值可以區分成三個區間，400 以下為空氣品質優良、400~700 為中度汙染、700 以上為高度汙染，該感測器特性適合監控空間的空氣品質，空氣品質因為工作環境不同，作業台工具使用的不同，產生不一樣的變化。

圖 3-8 空氣品質感測器

溫度和濕度傳感器[24]提供預校準的數字輸出硬體如圖 3-9 所示。 電容式傳感器元件測量相對濕度，溫度由負溫度係數（NTC）熱敏電阻測量。它具有優良的可靠性和長期穩定性。傳感器因硬體上限制不適用於 0 度以下的溫度，此感測器適用於測量室內溫度，溫度範圍為 0°c ~50° c 之間，濕度可偵測範圍為 20%~90%，該模組屬於數位埠。

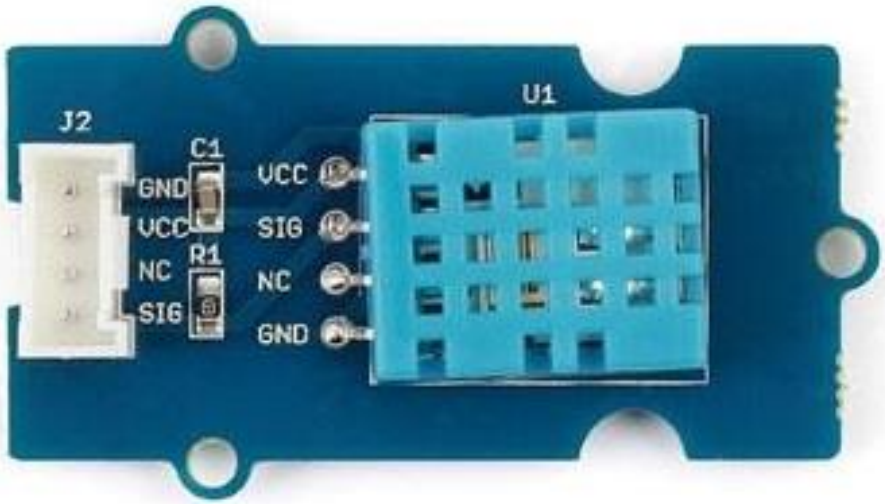

圖 3-9 溫溼度感測器

光線傳感器[25]使用 GL5528 光敏電阻（光敏電阻）來檢測環境中的光強度硬體如圖 3-10 所示。 當光強增加時，光敏電阻的電阻減小。板上的雙 OpAmp 芯片 LM358 產生對應於光強度（即基於電阻值）的電壓。 該模塊的輸出信號在亮光下為高電平，在黑暗中為低電平。該模塊可用於構建光控開關，即在白天時關閉燈並在夜間打開燈，該模組屬於類比類比埠，單位為類比值。

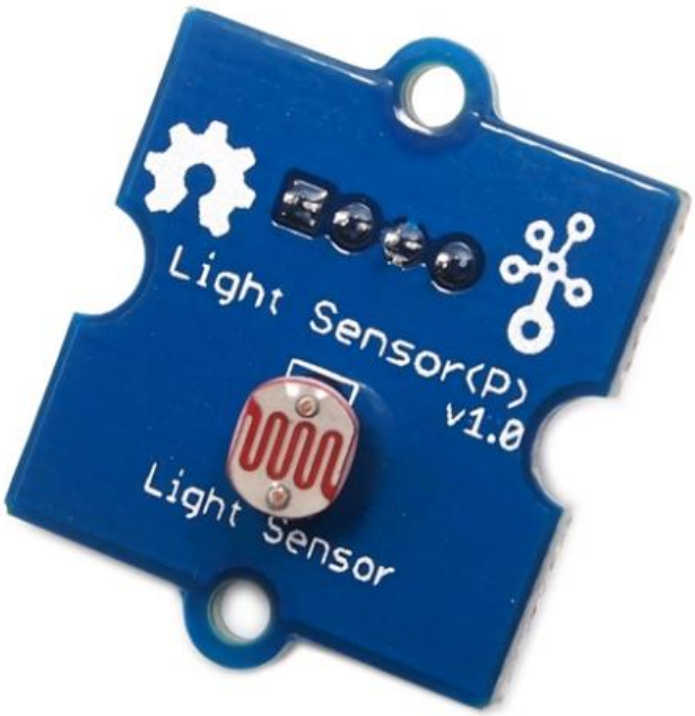

圖 3-10 光線感應感測器

聲音傳感器[26]可以檢測環境的聲音強度如圖 3-11 所示。 模塊的主要組件是一個簡單的麥克風，它基於 LM358 放大器和駐極體麥克風，該模組屬於類比埠，噪音閥值為 400，輸出值為類比值。

圖 3-11 聲音傳感器

3.3.2.2 軟體套件

論文中所使用三種軟體套件分別為 Scikit-Fuzzy、Bluepy、GrovePi，Scikit-Fuzzy 是以 Python 計算語言編寫，SciPy 是科學計算 Python 程式 Scikit-Fuzzy 是依賴 SciPy 函式庫所撰寫出的模糊邏輯算法，由 SciPy 社群開發的。Scikit-Fuzzy 安裝過程中，Scipy 包含數學函式套件庫與圖形函式庫[27]

BluePy 是一個提供 API 以 Python 連接藍牙低功耗設備的套件。 目前它只在 Linux 上運行，主要使用 Raspberry Pi 開發它，但它也將運行在 x86 Debian Linux 作業系統上。開發者作者 IanHarvey[28]在 Github 上提供。

GrovePi 套件是在 RaspberryPi 驅動 Grove Pi Shiled 軟硬體溝通的驅動程式，能辨識 Grove 感測器，以下是利用作業系統中 Command Line 下去作指令的安裝，先從 Github 下載 GrovePi 套件[29]，在資料夾切換到 Script 檔案去做解壓縮安裝的動作。

3.3.3 藍芽智慧插座

藍芽智慧插座，主要是以藍芽晶片、控制晶片、RTC 晶片而組成的，藍芽智慧插座乘載 110V~220V 電壓的電器產品，藍芽晶片所使用低功耗藍芽晶片，RTC 晶片可以讓記憶時間，藍芽智慧插座的功能是等待指令與實際電器產品的開關控制，預約時程和計時開關，可使用手機 App 直接操控插座，實體電路板如圖 3-12 所示。

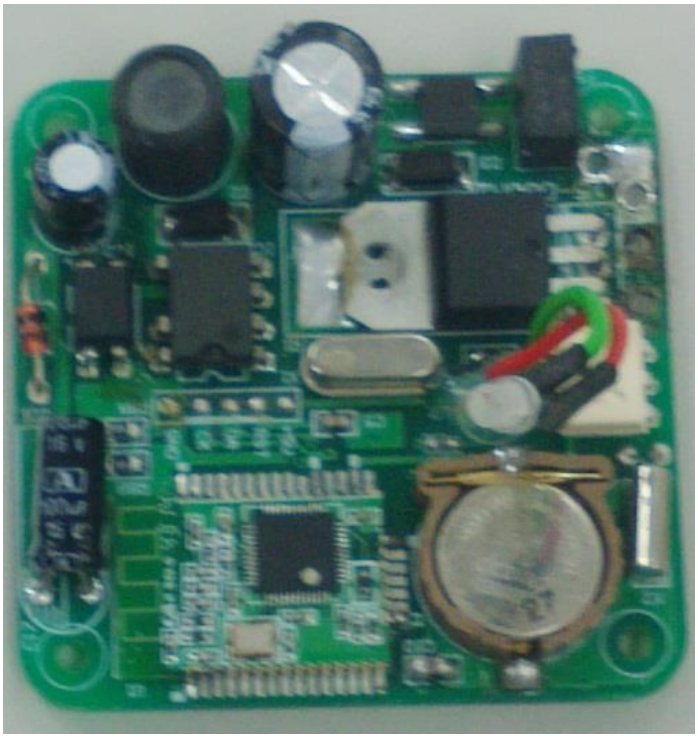

圖 3-12 藍芽智慧插座電路板

藍芽智慧插座 3D 列表機印製機殼並且安裝市售插座，設計圖如圖 3-13 所示，機殼大小規格為 15.4 cm x 5.4 cm x 5.3 cm、機殼上蓋規格為 5.6 cm x 5.8 cm x 2 cm。

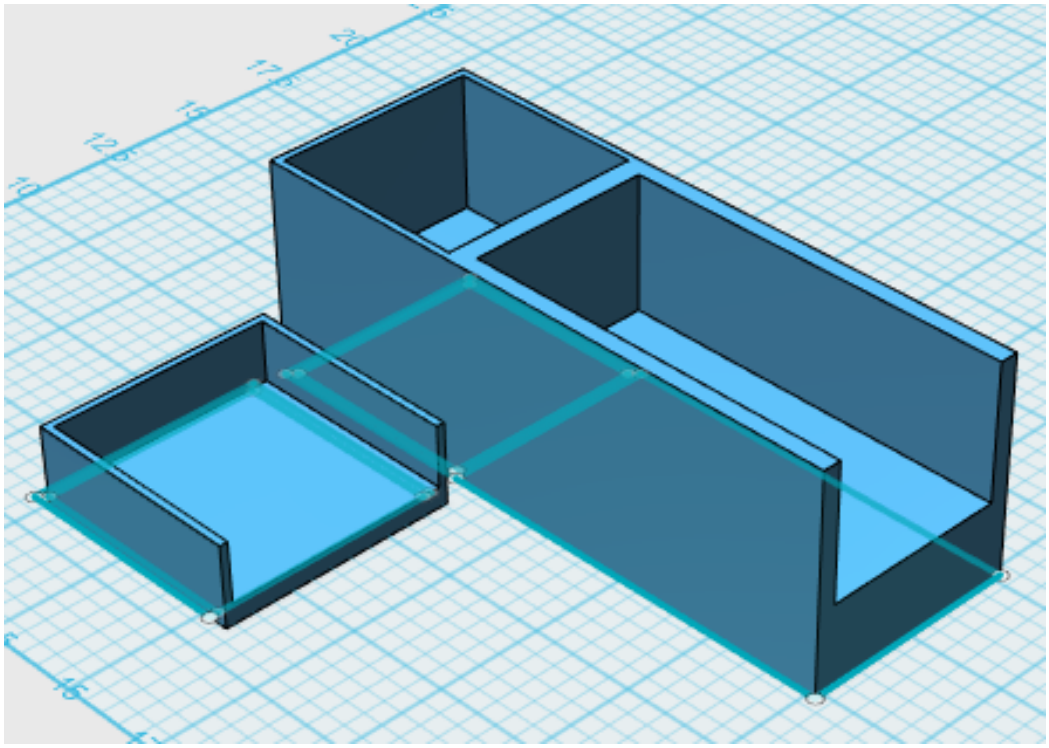

圖 3-13 藍芽智慧插座 3D 設計圖

3.3.4 環境資料的收集

建置好硬體環境後，依據各感測器是類比埠或數位埠去連接基板，程式碼建好後，使用 Cron Table[30]撰寫例行性工作自動開啟程式。讓程式不斷的收集資料，撰寫程式設定每五分鐘收集一次資料，收集光線、溫度、濕度、聲音、空氣品質數據，再把這些數值傳送給伺服器。為了防止再重開機的情況，硬體開機會自動執行例行性工作讓執行程式碼。

3.3 智慧型手機 App 控制

智慧型手機所使用 Android 系統來操控藍芽智慧插座，利用手機 App 操控，選擇連接的裝置，接著按電源鍵圖示，紅色代表該裝置關閉中，綠色代表裝置開啟中，設定操作模式則直接按下切換鍵，就會自動更新，如圖 3-14 所示。

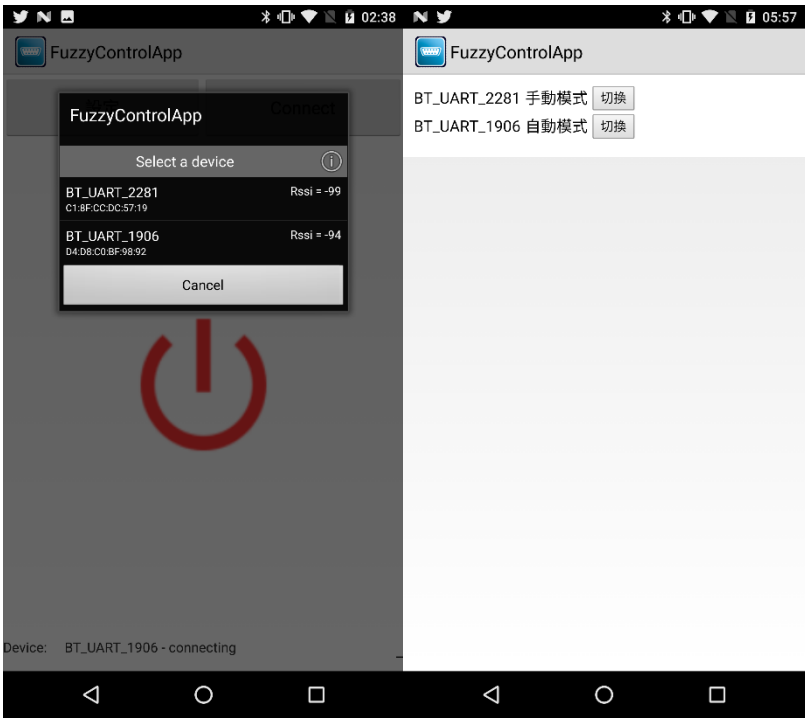

圖 3-14 藍芽智慧插座 Android App 操控介面

3.4 模糊邏輯控制器-辦公室電風扇

依據文獻，人體最舒適的溫度範圍 20°c ~26°c，根據交通部統計調查網的氣溫統計民國 87 年到 104 年區間[9]，台北站曾經觀測到的 2013 年 8 月 08 日最高氣溫是 39.3℃，台北市的都市效應造成北部的最高溫度，西元 2005 年 3 月 06 日則台北市最低溫度為 5.6°c，冬季平均溫度在於 16°c，考慮室內室外還是溫差的情況下，設定範圍設定 10°c ~40°c 區間下，使用程式自動切割成三段區塊，程式所呈現溫度的隸屬函數圖表如圖 3-15 所示。

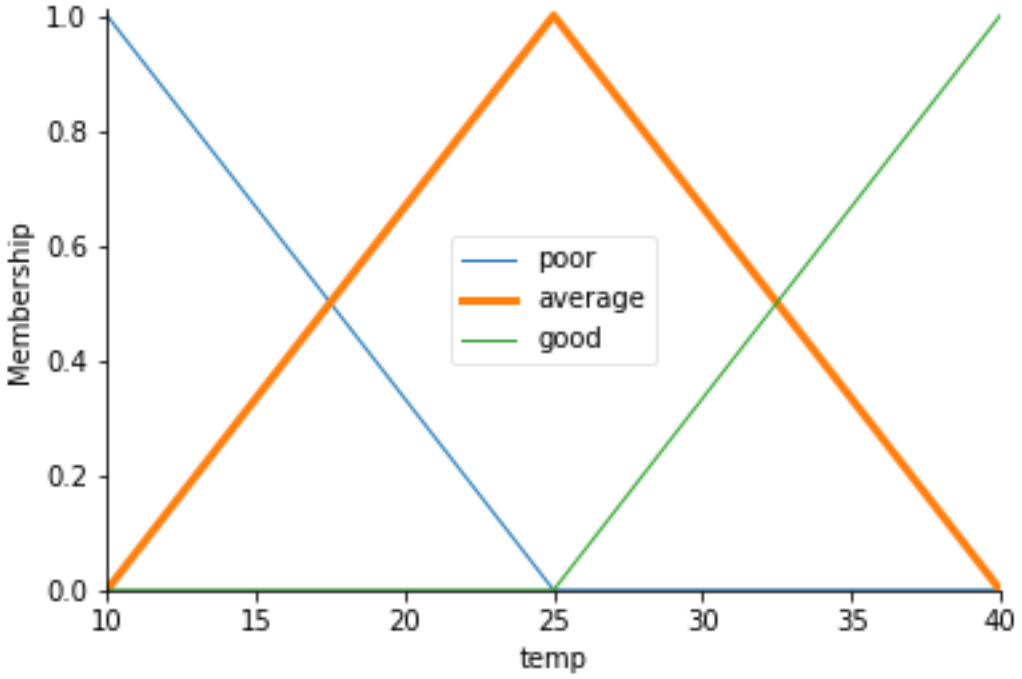

圖 3-15 溫度的隸屬函數

依據文獻，人體感受最為舒適的濕度範圍 40%~60%區間。台灣緯度位於 22 N 度至 25 N 度屬於熱帶氣候群、台灣氣候主要受到季風影響，台灣氣候雨量豐沛，日光充足的狀態下，台灣氣候容易屬於潮濕狀態。根據交通部統計調查網的溼度統計民國 87 年到 104 年區間[10]，相對溼度平均為 79 %。因此濕度範圍設定 20%~80%區間下，使用程式自動切割成三段區塊，程式所呈現濕度的隸屬函數圖表如圖 3-16 所示。

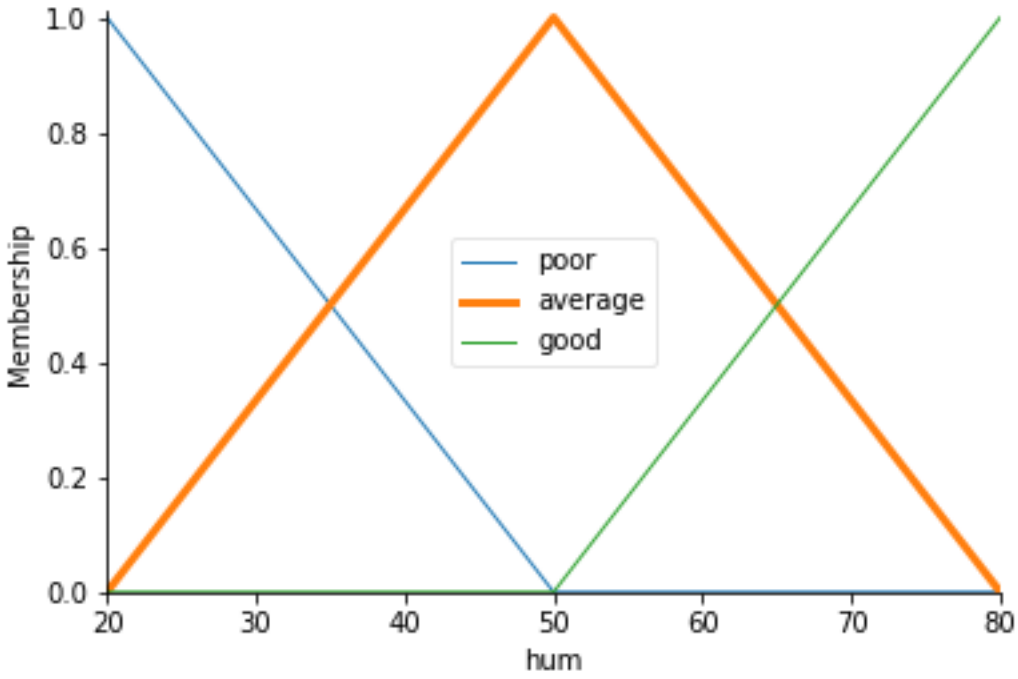

圖 3-16 濕度的隸屬函數圖表

光線的隸屬函數如圖 3-17 所示，因感測器使用類比輸出值範圍介於 0~1023 區間，設計時希望數字不要太大，因此程式碼以整除 100 的方式輸入參數進去，模糊邏輯設定光線範圍在 0~10 的區間，使用程式碼自動切割低中高區間。

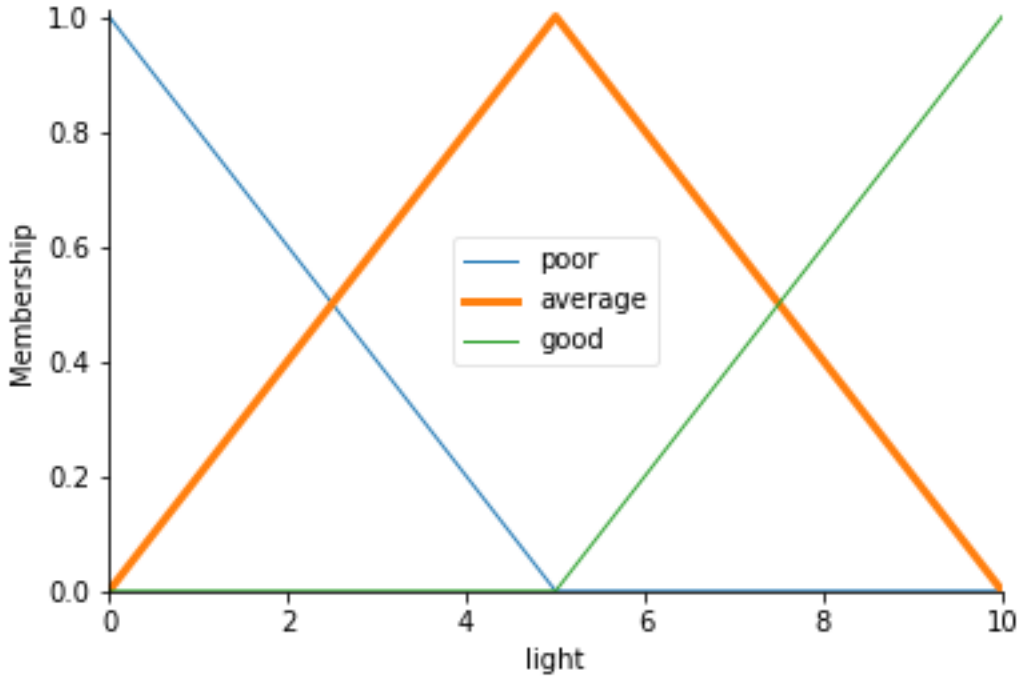

圖 3-17 光線的隸屬函數

切換開關的隸屬函數設定範圍是 0~10，低的區間為 0~5，中的區間為 0~5、5~10，高的區間則是 5~10。隸屬函數圖表如圖 3-18 所示。

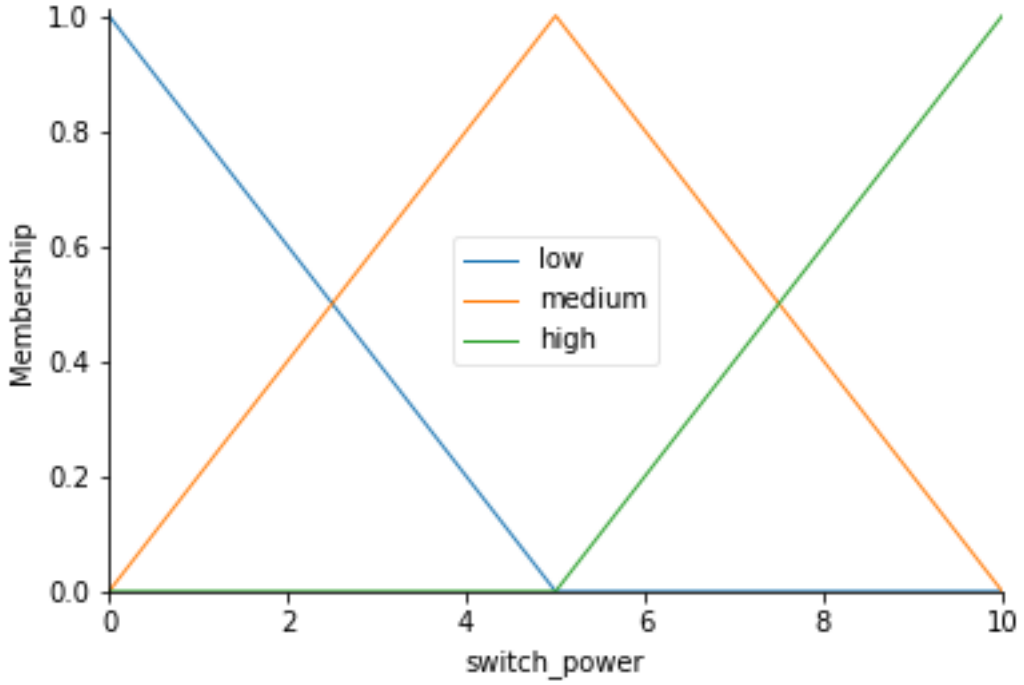

圖 3-18. 切換開關的隸屬函數

電風扇作測試範例，為辦公室工作環境設立，人不在研究室時，電風扇會關閉，則有人時，則依照溫度濕度，去開啟電扇，本文設立四條規則，規則一：當溫度熱時，會開啟電風扇、規則二；當溫度適中並且濕度過高時，會開啟電風扇、規則三：當溫度冷時，會關掉電風扇、規則四：當光線不足時，會關掉電風扇。為了判斷何時開啟電扇的時機，我們設立了一個演算法，如表格 3-2 所示。

模糊邏輯-辦公室電扇控制規則：

Rule1-IF 溫度 IS 非常熱 THEN 開啟電扇

Rule2-IF 溫度 IS 適中 AND 濕度 IS 非常高 THEN 開啟電扇

Rule3-IF 溫度 IS 非常冷 THEN 關閉電扇

Rule4-IF 光線 IS 非常暗 THEN 關閉電扇

表格 3-2 電扇模糊邏輯演算法虛擬碼

```
Function Fun Fuzzy Rule switch
01. Input:temp、hum、light、Switch_Status //溫度、濕度、光線、開關狀態
02. TempSetting(range(10, 41, 1)'temp') //設定溫度範圍 10c~40c
03. lightSetting(range (0, 11, 1), 'light')//設定光線範圍為 0~10
04. humSetting(range (20, 81, 1), 'hum') //設定濕度範圍為 20%~80%
05. switch_power=( range (0, 11, 1),'switch_power') //設定輸出值長度為 11
06. rule1 = Rule(temp['good'],switch_power['high'])
07. rule2 = Rule(temp['average'] & hum['good'] ,switch_power['high'])
08. rule3 = Rule(temp['poor'], switch_power['low'])
09. rule4 = Rule(light['poor'], switch_power['low'])
10. switch_power['low'] = fuzz.trimf(switch_power.universe, [0, 0, 5])
11 switch_power['medium'] = fuzz.(switch_power.universe, [0,5,10])
12.switch_power['high'] =fuzz.trimf(switch_power.universe, [5, 10, 10])
13.tipping.input['temp'] = float(temp) //輸入溫度參數
14.tipping.input['light'] =float(light) //輸入光線參數
15.tipping.input['hum'] = float(hum)    //輸入濕度參數
16.tipping.compute()  //依據環境參數和規則計算公式
17.N=tipping.output['switch_power'] //依據規則參數計算出來的輸出值
18.if N>5             //輸出值超過 5 時開啟開關
19.   Swich_Status=on //開啟智慧插座
20.else if N<=5
21.   Swich_Status=off //關閉智慧插座
```

3.5 模擬環境-辦公室電風扇

在實驗中，我們模擬四條規則下，使用電腦直接輸入參數，模擬輸出的過程，我們依據五種不一樣的情境模擬運算，第一種情境；低光線、溫度適中、濕度高的情況，第二種情境；光線充足但是溫度濕度偏低的情況，第三種情境；溫度高、濕度適中、光線低，第四種情境；溫度高、濕度高、光線充足，第五種情境；溫度低、濕度低，光線不充足。

參數如表格 3-3 所示，可以從表格 3-3 中了解，光線不足時輸出值偏低，若則光線充足，溫度越偏高時開啟電扇、溫度偏低時則採取不做任何動作。則只要光線不充足，就一定會關電扇，但是濕度過高、溫度過高的情況下則是會開起電風扇。

表格 3-3 情境模擬-參數輸入

溫度	濕度	光線	輸出值	命令
26	78	0.20	4.99	關閉電扇
26	65	0.20	4.96	關閉電扇
14	20	7.46	1.33	關閉電扇
14	40	7.01	1.33	關閉電扇
36	78	0.70	5.03	開啟電扇
30	65	0.70	4.95	關閉電扇
31	80	7.50	8.66	開啟電扇
28	65	7.20	8.66	開啟電扇
10	40	0.20	1.33	關閉電扇
10	20	0.20	1.33	關閉電扇

3.6 模糊邏輯控制器-空氣濾淨機

因為空氣品質感測器的關係，空氣沒辦法偵測到特定濃度的氣體，但可以知道空氣品質的程度，輸出值是類比值 1~1023 的範圍之下，所以在程式中除上 100 在做輸入的動作。在模糊邏輯的設定，低的區域在 0~5，平均的區域在於 0~5、5~10，高的區域 5~10。數字越高則汙染程度越嚴重，隸屬函數如圖 3-19 所示。

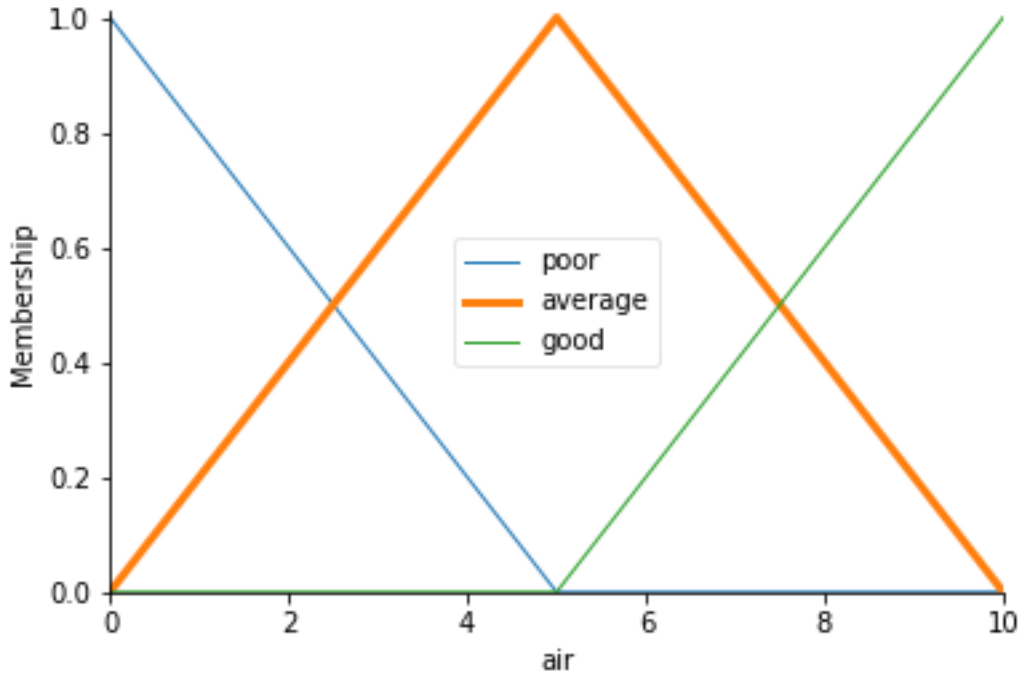

圖 3-19 空氣品質的隸屬函數

噪音感測器是類比值輸出範圍則是 1~1023，在程式中方便處理，直接除上 100 再進行輸入的動作，模糊邏輯設定則是，低的區間在 0~5，平均的區域在 0~10，高的區域在 5~10。則數字越高代表噪音越吵。切換開關的隸屬函數則跟辦公室電扇是一致的。隸屬函數如圖 3-21

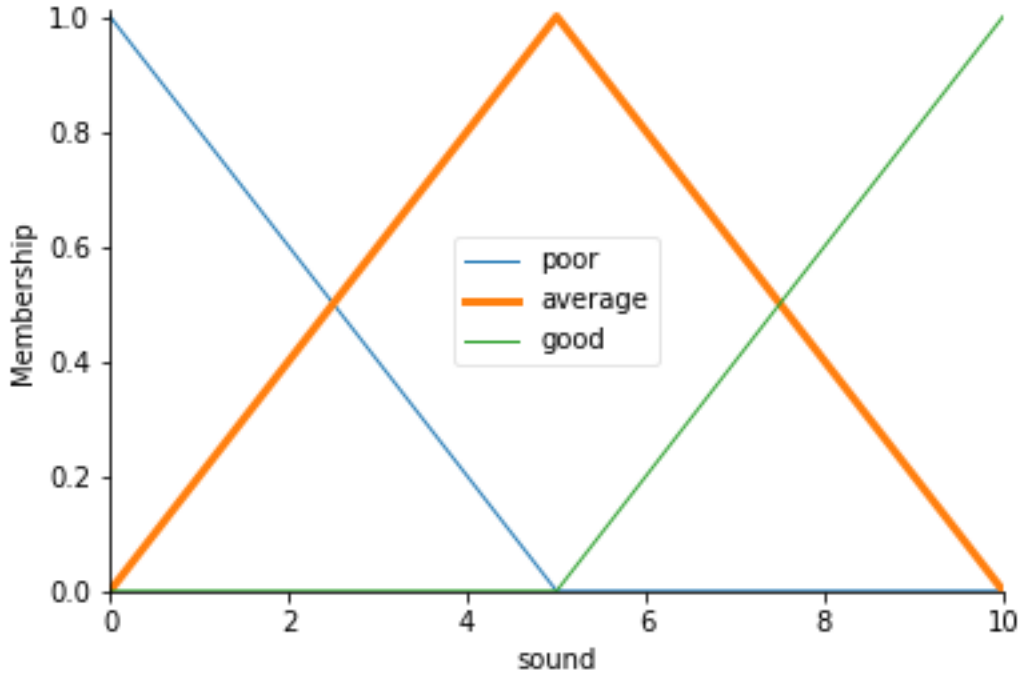

圖 3-21. 噪音的隸屬函數

以空氣濾淨機作測試範例，為辦公室工作環境設立，主要是過濾空氣和保持安靜的環境設立，本文設立四條規則，條列一：當空氣汙染高時，會開啟機器、條列二；當噪音、空氣汙染介於中間值時，會關閉機器、條列三：當空氣汙染介於中間且噪音屬於時，會開啟機器、條列四：當空氣汙染低或者噪音屬於低時，會關閉機器。為了判斷何時開啟電扇的時機，我們設立了一個演算法，如表格 3-4 所示。

模糊邏輯-空氣濾淨機邏輯控制規則：

Rule1-IF 空氣汙染程度 IS 非常髒 THEN 開啟機器

Rule2-IF 空氣汙染程度 IS 平均值 AND 噪音 IS 非常吵 THEN 關閉機器

Rule3-IF 空氣汙染程度 IS 平均值 AND 噪音 IS 平均值 THEN 關閉機器

Rule4-IF 空氣汙染程度 IS 非常低 AND 噪音 IS 非常吵 THEN 關閉機器

表格 3-4 空氣濾淨機邏輯演算法虛擬碼

```
Function Fun Fuzzy Rule switch
01.sound =ctrl.Antecedent(np.arange(0, 11, 1), 'sound')空氣品質範圍長度
02.air =ctrl.Antecedent(np.arange(0, 11, 1), 'air') 噪音範圍長度
03.switch_power = ctrl.Consequent(np.arange(0, 11, 1), 'switch_power')
04.switch_power['low'] = trimf(switch_power.universe, [0, 0, 5])
05.switch_power['medium'] = trimf(switch_power.universe, [0, 5, 10])
06.switch_power['high'] = trimf(switch_power.universe, [5, 10, 10])
07.rule1 =Rule(air['good'], switch_power['high'])
08.rule2 =Rule(air['average'] & sound['average'], switch_power['low'])
09.rule3 =Rule(air['average'] & sound['poor']  ,switch_power['high'])
10.rule4 =Rule
   (air['poor']  | sound['good'],switch_power['low'])
11.tipping_ctrl = ctrl.ControlSystem([rule1,rule2,rule3,rule4])
12.tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
13.tipping.input['sound'] = float(sound) //輸入聲音參數
14.tipping.input[' air '] =float(air) //輸入空氣參數
15.tipping.compute()  //依據環境參數和規則計算公式
16.N=tipping.output['switch_power'] //依據規則參數計算出來的輸出值
17.if N>5             //輸出值超過 5 時開啟開關
18.    Swich_Status=on //開啟智慧插座
19.else if N<=5
20.    Swich_Status=off //關閉智慧插座
```

3.7 模擬環境-空氣濾淨機

在實驗中，我們模擬四條規則下，使用電腦直接輸入參數，模擬輸出的過程，我們依據五種不一樣的情境模擬運算，第一種情境；高汙染、高噪音的情況，第二種情境；低汙染、低噪音情況，第三種情境；噪音低、汙染高，第四種情境；噪音高、汙染低，第五種情況：噪音、汙染在於平均值。 參數如表格 3-5 所示，可以從表格中了解，若只要噪音只要超過 5 基本上就會關閉機器，但是過高空氣汙染時就會開啟機器。

表格 3-5 情境模擬-參數輸入

空氣品質	噪音	輸出值	命令
9.00	9.00	5.00	開啟
6.00	6.00	3.36	關閉
1.00	1.00	3.36	關閉
4.00	4.00	3.36	關閉
6.00	1.00	6.63	開啟
9.00	4.00	6.63	開啟
4.00	9.00	1.72	關閉
1.00	6.00	1.72	關閉
5.00	5.00	1.66	關閉
5.00	5.00	1.66	關閉

第四章　實驗與結果

4.1 實驗環境

實驗環境位於台中市朝陽科技大學的資訊大樓一樓實驗室,M117-Lab 屬於不對流的密閉空間，該環境中使用 2 部空調機使空氣對流，環境中共有 14 台電腦運作，電腦運作時會有些許的廢熱產生、空氣品質良好、研究室環境安靜，實驗的環境為大小為 15 坪(7.5 m x 6.7 m x 3.2 m)的空間，平面圖和 3D 圖如 4.1 所示，日期取至 2017-01-11，圖表中生產 X 軸是小時、Y 軸是平均數值。將控制中心建置在實驗室中插上電源供電，讓控制中心依據環境變化來實際操控電器產品。本研究所使用的電器產品則是電風扇、空氣濾淨機。

圖 4-1 實驗室環境圖

4.2 實驗與結果

實驗記錄期間 2017-01-01 到 2017-04-30 共有 34,848 筆資料，監控圖表擷取自 2017-01-11 日期當天研究室的每小時狀況，透過瀏覽器連接到我們的監控介面，如圖 4-2 所示，可以了解整體實驗室的小時平均溫度介於 25c~28c 之間，實驗室室內中因為有冷氣空調。溫度上對於人體舒適上，相當符合文獻中的標準。整體平均濕度位於 20%的狀態，屬於比較乾燥的環境，人們不適合過長時間屬於在乾燥的環境中。

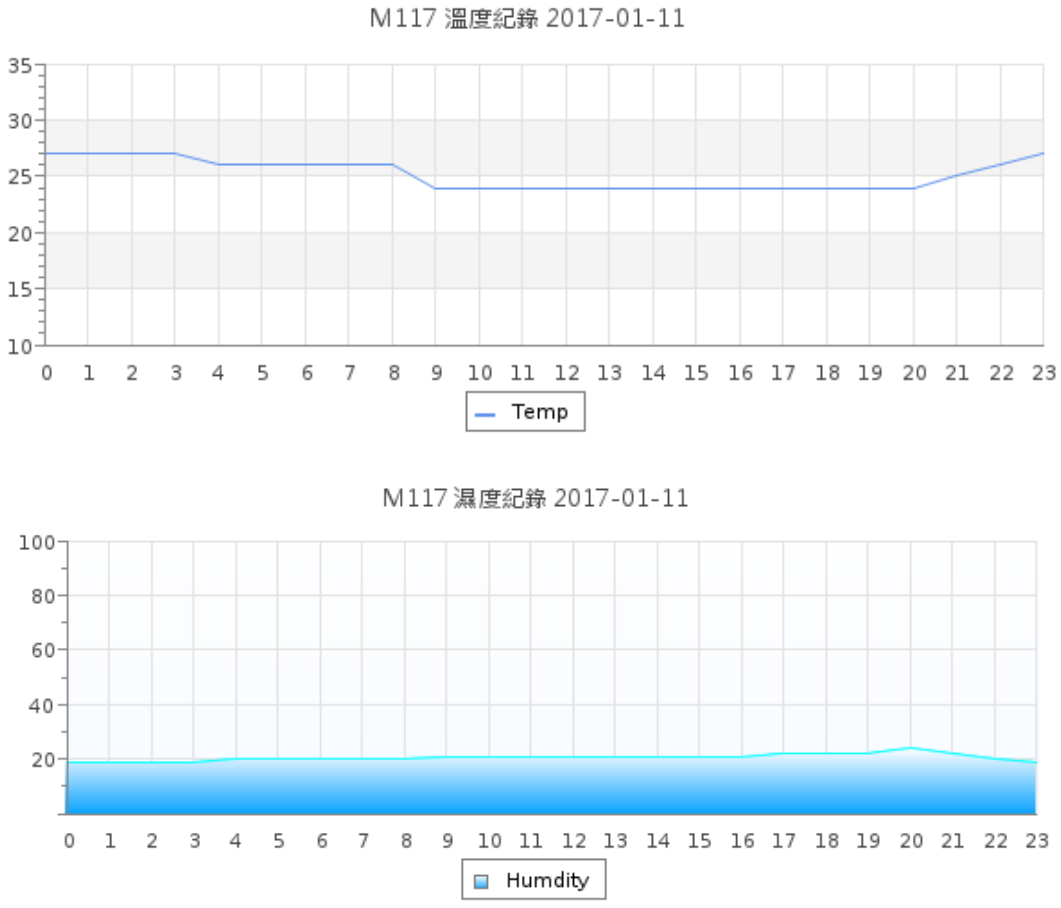

圖 4-2 2017 年 01 月 11 日 溫度濕度圖表

圖 4-3 所示在光線數值的監測中，可以了解當日 0 點到 1 點之間和 8 點到 20 點之間，類比值一直保持在 700，依據類比值，可以推測空間是否有人使用。聲音監控所收集的類比值，聲音的類比值位於 200，屬於低強度聲音環境下。

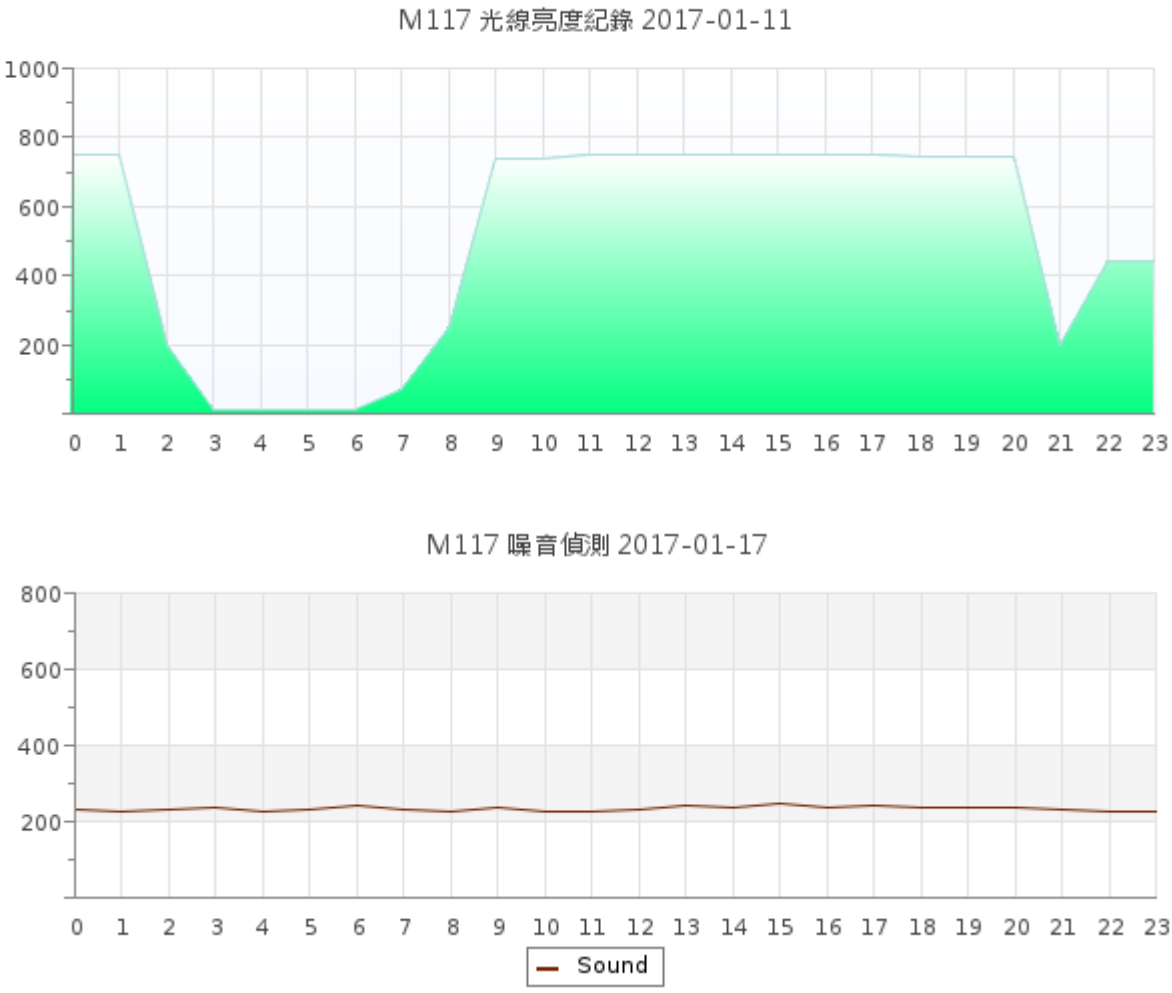

圖 4-3 2017-01-17 光線聲音監控圖表

依據感測器文件紀錄超過 700 以上為高汙染、700~400 為中度汙染、400 以下為無汙染，如圖 4-5 所示空氣品質 24 小時位於 100 以下，空間環境無有害氣體存在。

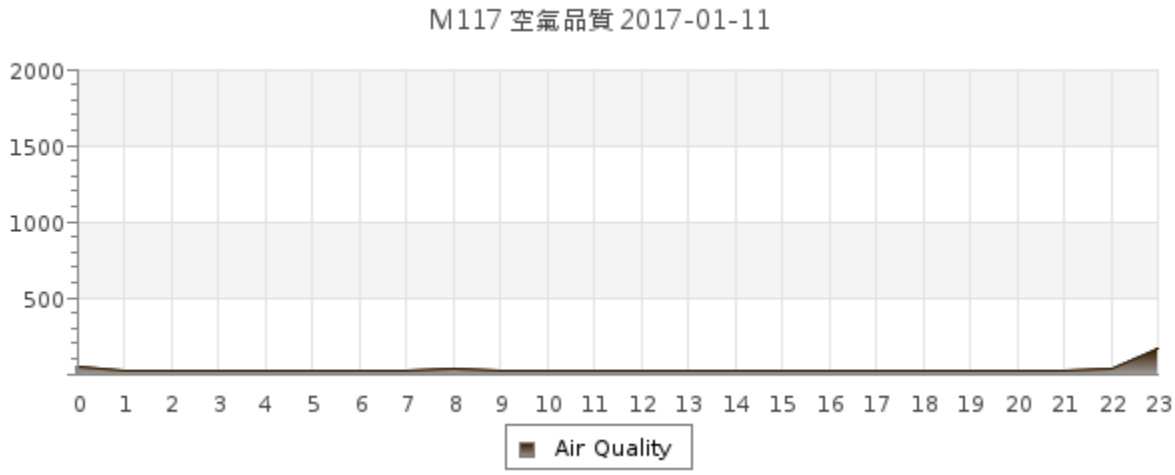

圖 4-5 空氣品質

實驗針對電風扇設備去紀錄所需要的環境資訊，溫度、濕度，光線類比值單位。表格 4-1 記錄著 70 筆資料，記錄著電風扇開啟關閉的時段，可以觀測出我們的四條規則下，參數輸入的狀況以及輸出值的狀況，在四條規則中最明顯的是光線對於輸出值的影響，只要電燈一關，電扇也會跟著關閉。但在於兩筆資料下一筆 2017/4/26 09:45　光線 5.99 溫度 26.04 輸出值 8.32584 和另一筆 2017/4/28 09:25 光線 4.86 溫度 29.11 輸出值 7.73744，在兩筆資料中，光線屬於適中的情況下，溫度偏高的情況下，控制中心基於模糊邏輯規則下，輸出值為偏高，超過了 5 所以自動讓電器產品開啟。

表格 4-1 設備開啟資料與環境數據-辦公室電扇

藍芽裝置	日期	溫度	濕度	光線	輸出值	命令
BT_UART_2281	2017/4/13 21:45	27.12	19.00	7.49	8.30	on
BT_UART_2281	2017/4/13 22:25	28.00	19.00	0.03	4.90	off
BT_UART_2281	2017/4/14 10:05	27.12	19.00	7.52	8.30	on
BT_UART_2281	2017/4/14 10:10	28.03	19.00	0.70	4.95	off
BT_UART_2281	2017/4/14 10:15	27.05	19.00	6.08	8.31	on
BT_UART_2281	2017/4/14 19:25	24.00	21.00	0.01	4.99	off
BT_UART_2281	2017/4/15 09:05	27.11	24.00	7.46	8.30	on
BT_UART_2281	2017/4/15 09:30	28.00	20.00	0.65	4.95	off
BT_UART_2281	2017/4/15 12:10	28.11	20.00	7.45	8.27	on
BT_UART_2281	2017/4/15 21:15	24.00	21.00	0.01	4.99	off
BT_UART_2281	2017/4/16 11:25	28.11	20.00	7.44	8.27	on
BT_UART_2281	2017/4/16 22:30	24.00	24.00	0.01	4.99	off
BT_UART_2281	2017/4/17 08:55	28.11	23.00	7.46	8.27	on
BT_UART_2281	2017/4/17 19:05	24.00	25.00	0.01	4.99	off
BT_UART_2281	2017/4/18 06:35	28.11	23.00	7.45	8.27	on
BT_UART_2281	2017/4/18 19:35	26.00	26.00	0.02	4.99	off
BT_UART_2281	2017/4/18 20:40	25.11	23.00	7.44	8.33	on
BT_UART_2281	2017/4/19 17:20	26.00	23.00	0.07	4.99	off
BT_UART_2281	2017/4/20 09:55	29.11	23.00	7.47	8.24	on
BT_UART_2281	2017/4/20 18:00	26.00	26.00	0.03	4.99	off
BT_UART_2281	2017/4/21 09:25	29.11	19.00	7.47	8.24	on
BT_UART_2281	2017/4/21 19:25	28.00	19.00	0.02	4.90	off
BT_UART_2281	2017/4/22 08:25	28.11	19.00	7.46	8.27	on
BT_UART_2281	2017/4/22 08:40	29.01	18.00	0.26	4.84	off
BT_UART_2281	2017/4/22 08:50	29.11	18.00	7.47	8.24	on
BT_UART_2281	2017/4/22 10:30	26.02	20.00	0.31	5.00	off
BT_UART_2281	2017/4/22 11:40	25.11	20.00	7.43	8.33	on
BT_UART_2281	2017/4/24 17:50	24.00	21.00	0.07	4.99	off
BT_UART_2281	2017/4/24 19:30	27.11	19.00	7.46	8.30	on
BT_UART_2281	2017/4/24 21:25	27.00	19.00	0.01	4.96	off
BT_UART_2281	2017/4/25 08:15	28.11	19.00	7.45	8.27	on
BT_UART_2281	2017/4/25 15:35	29.01	31.00	0.27	4.84	off
BT_UART_2281	2017/4/25 16:40	24.11	21.00	7.43	7.72	on
BT_UART_2281	2017/4/25 19:10	24.00	10.00	0.03	4.99	off

BT_UART_2281	2017/4/25 21:35	27.11	19.00	7.46	8.30	on
BT_UART_2281	2017/4/25 21:40	27.00	0.00	0.01	4.96	off
BT_UART_2281	2017/4/26 09:45	26.04	20.00	5.99	8.33	on
BT_UART_2281	2017/4/26 18:30	27.00	19.00	0.02	4.96	off
BT_UART_2281	2017/4/27 09:50	28.11	29.00	7.46	8.27	on
BT_UART_2281	2017/4/27 18:30	26.00	20.00	0.02	4.99	off
BT_UART_2281	2017/4/28 09:25	27.04	19.00	5.98	8.31	on
BT_UART_2281	2017/4/28 09:40	22.03	26.00	0.60	4.03	off
BT_UART_2281	2017/4/28 09:55	27.11	20.00	7.45	8.30	on
BT_UART_2281	2017/4/28 17:35	26.01	20.00	0.11	4.99	off
BT_UART_2281	2017/4/29 13:00	27.11	29.00	7.46	8.30	on
BT_UART_2281	2017/4/29 21:00	24.00	21.00	0.02	4.99	off
BT_UART_2281	2017/4/30 11:45	27.11	19.00	7.44	8.30	on
BT_UART_2281	2017/4/30 18:50	24.00	21.00	0.01	4.99	off
BT_UART_2281	2017/5/1 10:05	25.11	9.00	7.45	8.33	on
BT_UART_2281	2017/5/1 17:50	24.00	21.00	0.05	4.99	off
BT_UART_2281	2017/5/2 09:50	28.04	19.00	5.94	8.28	on
BT_UART_2281	2017/5/3 21:50	27.00	21.00	0.01	4.96	off
BT_UART_2281	2017/5/4 08:50	30.11	19.00	7.45	8.19	on
BT_UART_2281	2017/5/5 19:40	24.00	24.00	0.02	4.99	off
BT_UART_2281	2017/5/6 10:00	27.11	23.00	7.44	8.30	on
BT_UART_2281	2017/5/6 21:15	24.00	21.00	0.02	4.99	off
BT_UART_2281	2017/5/7 12:30	29.12	23.00	7.50	8.23	on
BT_UART_2281	2017/5/7 18:55	24.00	21.00	0.01	4.99	off
BT_UART_2281	2017/5/8 08:40	29.12	23.00	7.48	8.23	on
BT_UART_2281	2017/5/8 20:05	24.00	22.00	0.02	4.99	off
BT_UART_2281	2017/5/9 01:10	28.00	22.00	5.21	8.28	on
BT_UART_2281	2017/5/9 01:30	28.00	19.00	0.02	4.90	off
BT_UART_2281	2017/5/9 08:45	29.11	21.00	7.46	8.24	on
BT_UART_2281	2017/5/9 16:45	24.01	27.00	0.13	4.99	off
BT_UART_2281	2017/5/9 16:50	24.11	25.00	7.44	7.72	on
BT_UART_2281	2017/5/9 19:05	24.00	21.00	0.01	4.99	off
BT_UART_2281	2017/5/9 20:15	27.11	23.00	4.86	7.99	on
BT_UART_2281	2017/5/9 21:40	28.00	20.00	0.02	4.90	off
BT_UART_2281	2017/5/10 09:25	29.11	18.00	7.47	8.24	on
BT_UART_2281	2017/5/12 17:15	24.01	21.00	0.15	4.99	off

實驗針對空氣濾淨機設備去紀錄所需要的環境資訊，空氣品質、噪音類比值單位。表格 4-2 記錄著約 25 筆資料，記錄著空氣濾淨機開啟關閉的時段，因在實驗室的空氣品質一直處於不錯的狀態和噪音環境不高的狀況，所以指令偏向屬於關閉的狀態。

表格 4-2 設備開啟資料與環境數據-空氣濾淨機

藍芽裝置	日期	聲音	空氣品質	輸出	命列
BT_UART_1906	2017/7/2 17:55	2.27	0.41	4.30	off
BT_UART_1906	2017/7/2 17:50	0.42	0.41	2.50	off
BT_UART_1906	2017/7/2 17:45	0.41	0.42	2.50	off
BT_UART_1906	2017/7/2 17:40	0.42	0.42	2.50	off
BT_UART_1906	2017/7/2 17:35	2.27	0.41	4.30	off
BT_UART_1906	2017/7/2 17:30	2.26	0.42	4.30	off
BT_UART_1906	2017/7/2 17:25	2.26	0.42	4.30	off
BT_UART_1906	2017/7/2 17:20	0.41	0.41	2.49	off
BT_UART_1906	2017/7/2 17:15	2.27	0.42	4.30	off
BT_UART_1906	2017/7/2 17:10	2.28	0.42	4.31	off
BT_UART_1906	2017/7/2 17:05	0.40	0.40	2.47	off
BT_UART_1906	2017/7/2 17:00	0.41	0.41	2.49	off
BT_UART_1906	2017/7/2 16:55	0.41	0.41	2.49	off
BT_UART_1906	2017/7/2 16:50	0.42	0.42	2.50	off
BT_UART_1906	2017/7/2 16:45	2.27	0.43	4.30	off
BT_UART_1906	2017/7/2 16:40	2.30	0.40	4.32	off
BT_UART_1906	2017/7/2 16:35	0.39	0.40	2.47	off
BT_UART_1906	2017/7/2 16:30	2.28	0.40	4.30	off
BT_UART_1906	2017/7/2 16:20	0.37	0.37	2.42	off
BT_UART_1906	2017/7/2 16:15	0.37	0.37	2.42	off
BT_UART_1906	2017/7/2 16:10	2.29	0.37	4.31	off
BT_UART_1906	2017/7/2 16:05	2.29	0.37	4.31	off
BT_UART_1906	2017/7/2 16:00	0.37	0.38	2.43	off
BT_UART_1906	2017/7/2 15:55	2.28	0.37	4.30	off
BT_UART_1906	2017/7/2 15:50	2.28	0.38	4.30	off

第五章 結論及未來研究

5.1 結論

隨著科技的發展，也因此近幾年來針對智慧居家的議題的討論也越來越多。對於現代人來說生活的品質越來越重視，如果能用低廉的成本和價格去推廣智慧住宅的發展，對於現代人生活是有幫助的，增進現代人的生活上的便利。因此本研究提出一個基於模糊邏輯針對環境數據的實作模糊邏輯控制與智能居家系統，透過模糊邏輯來判斷的是否要開啟這項電子產品，紀錄環境變數與電子產品的開關紀錄。賦予電子產品自動化智慧化的能力。

5.2 未來研究

而在未來研究方面，會使用智慧手錶記錄人體的身體狀況，依據身體的狀況去做讓電子產品判斷是否要開，是否要打開這項電子產品，使用智慧手表來連接控制中心表示，人有近來這研究室，讓控制中心更有智慧的執行電器產品的開關時段，並且讓控制中心加入不斷電系統。

參考文獻

[1]施驊原，2017年，基於模糊邏輯與決策樹使用螢幕觸壓之手機身分識別系統，資訊管理系，朝陽科技大學，台中。

[2]周劭岳，2011年，應用模糊邏輯評估影響網路購物忠誠度之因素，資訊經營學系，大同大學，台北。

[3]陳冠竹，2010年，自主飛行無人定翼機之高性能GPS導引模糊邏輯控制器設計，電機工程系碩士班，國立雲林科技大學，雲林。

[4]陳世昕，2015 年，物聯網智慧住宅 APP 與藍牙安全實作，碩士論文，僑光科技大學，資訊科技研究所，台中。

[5]柯博斌，2014 年，嵌入式感測盒的實作議題-Raspberry Pi 系統模組案例，碩士論文，國立臺灣科技大學，自動化及控制研究所，台北。

[6]江元喻，2014年，以 Raspberry Pi 與 Arduino 實作之空氣品質系統，碩士論文，聖約翰大學，電機工程研究所，淡水。

[7]余建志，2010年，例室內濕度與溫度對舒適度指標影響之研究，碩士論文，國立屏東科技大學，環境工程與科學系，屏東。

[8]丁元昱，2016 年，實作 Android 手機操作藍芽智慧插座，第十屆資訊科技國際研討會，朝陽科技大學，台中。

[9]交通部統計調查網，統計氣溫圖表，
http://stat.motc.gov.tw/mocdb/stmain.jsp?sys=100
[10]交通部統計調查網，統計濕度圖表，
http://stat.motc.gov.tw/mocdb/stmain.jsp?sys=100
[11]L.A. Zadeh(1965), “Fuzzy Sets”, Information and Control,Vol.8,No 3,pp. 338-353.
[12]Azim Keshtkar , Siamak Arzanpour, “An adaptive fuzzy logic system for residential energy management in smart grid environments,” Applied Energy ,Vol. 186,pp. 68–81, November 2016.
[13]Sajid Hussain, Hossam A. Gabbar, Daniel Bondarenko, Farayi Musharavati, Shaligram Pokharel, “Comfort-based fuzzy control optimization for energy conservation in HVAC systems,” Control Engineering Practice, Vol. 32,pp. 172-182, September 2014.
[14]Internet of Things ,Luigi Atzori,Antonio Iera, Giacomo Morabito. “The Internet of Things: A survey (PDF). Computer Networks,” 2010.
[15]smart home 定義，
http://www.t3.com/features/the-smart-home-guide

[16]Andreas Jacobsson,Martin Boldt, Bengt Carlssonb, “A risk analysis of a smart home automation system,” Future Generation Computer Systems
,Vol. 56,pp 719-733, June 2015.

[17]Arif Ahmed , Ejaz Ahmed,” A Survey on Mobile Edge Computing,” India: 10th IEEE International Conference on Intelligent Systems and Control(ISCO』16), India, November 2016

[18]Sheikh Ferdoush, Xinrong Li, “Wireless Sensor Network System Design using Raspberry Pi and Arduino for Environmental Monitoring Applications,”The 9th International Conference on Future Networks and Communications (FNC-2014),pp.103－110, University of North Texas, Denton, Texas, September 2014

[19]Raspberry Pi,https://www.raspberrypi.org/help/faqs/

[20]Vladimir Vujovic, Mirjana Maksimovic, “Raspberry Pi as a Sensor Web node for home automation,” Computers and Electrical Engineering, Vol. 44, pp. 153－171, May 2015.

[21]Bluetooth,https://www.bluetooth.com/what-is-bluetooth-technology

[22]Leandro Y. Mano ,Bruno S. Faical ,Luis H.V. Nakamura ,Pedro H. Gomes,Giampaolo L. Libralon ,Rodolfo I. Meneguete, “Exploiting IoT technologies for enhancing Health Smart Homes,” Computer Communications, Vol. 89－90, pp. 178－190, September 2016.

[23]Air Quality Sensor specification, http://wiki.seeed.cc/Grove-Air_Quality_Sensor_v1.3/

[24]Temperature And Humidity_Sensor specification, http://wiki.seeed.cc/Grove-TemperatureAndHumidity_Sensor/

[25]light Sensor specification,http://wiki.seeed.cc/Grove-Light_Sensor/

[26]Sound Sensor specification,http://wiki.seeed.cc/Grove-Sound_Sensor/

[27]Scikit-Fuzzy Document,http://pythonhosted.org/scikit-fuzzy/

[28]Bluepy Document,http://ianharvey.github.io/bluepy-doc/index.html

[29]GrovePi Document,https://github.com/DexterInd/GrovePi

[30]Wikipedia:Cron ,https://en.wikipedia.org/wiki/Cron

附錄

安裝軟體套件指令

```
sudo python -m pip install --upgrade pip

sudo pip3 install --user numpy scipy matplotlib ipython jupyter pandas sympy nose

sudo pip 3 install  scikit-fuzzy


sudo apt-get install git build-essential libglib2.0-dev
git clone https://github.com/IanHarvey/bluepy.git
cd bluepy
python setup.py build
sudo python setup.py install

sudo git clone https://github.com/DexterInd/GrovePi

cd GrovePi/Script

sudo chmod +x install.sh

sudo ./install.sh
```

Python 模糊邏輯控制藍芽智慧插座-程式碼

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
from grovepi import *
from grove_rgb_lcd import *
import time
import grovepi
import  datetime
import mysql.connector


from btle import UUID,Peripheral
import struct
```

```
import binascii

def sqlinsert(device,temp,hum,light,sound,air,output,command):
    try :
        device=str(device)
        senddevice='BT_UART_1906'
        today=str(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
        cnx = mysql.connector.connect(user='abc',
password='1234',host='120.110.7.25',database='smarttimersocket')
        cursor = cnx.cursor()
        add_test = ("INSERT INTO `socket1`"
                    "(`id`, `Deviceid`, `Senddevice`, `Date`, `temp`, `hum`, `light`,
`sound`, `air`, `output`, `CommandLine`)"
                    "VALUES"
                    "(NULL, %s, %s, %f,%f, %f,%f,%f,%f,%f,%s);")
        data_test=(device,senddevice,today,temp,hum,light,sound,air,output,command)

        a=(add_test % data_test)
        print (a)
        cursor.execute(add_test, data_test)
        cnx.commit()
        cursor.close()
        cnx.close()
    except (IOError, TypeError) as e:
        print ("sqlinsert IOError TypeError:"+str(e))
    except Exception as e:
        print("sqlinsert Error:"+str(e))
        f = open('file.txt','r+')
        add_test = ("INSERT INTO `socket1`"
                    "(`id`, `Deviceid`, `Senddevice`, `Date`, `temp`, `hum`, `light`,
`sound`, `air`, `output`, `CommandLine`)"
                    "VALUES"
                    "(NULL, %s, %s, %f,%f, %f,%f,%f,%f,%f,%s);")
        data_test=(device,senddevice,today,temp,hum,light,sound,air,output,command)
        a=(add_test % data_test)+'\n'
        print(f.read()+str(a))
        f.write(f.read()+str(a))
```

```
        f.close()
def sqlget():
    try :
        cnx = mysql.connector.connect(user='abc',
password='1234',host='120.110.7.27',database='smarttimersocket')
        cursor = cnx.cursor()
        #My SQL Command
        cursor.execute("SELECT * FROM `socket1` WHERE `Deviceid`=1 ORDER BY
`socket1`.`id` DESC LIMIT 0,1")
        row=cursor.fetchone()
        cursor.close()
        cnx.close()
        return row10]
    except (IOError, TypeError) as e:
        print ("sqlget IOError TypeError:"+str(e))
    except Exception as e:
        print("sqlget Error:"+str(e))
def sqlauto(deviceid):
    try :
        deviceid=str(deviceid)
        cnx = mysql.connector.connect(user='abc',
password='1234',host='120.110.7.27',database='smarttimersocket')
        cursor = cnx.cursor()
        #My SQL Command
        cursor.execute("SELECT * FROM `device_switch` WHERE `id`="+deviceid)
        row=cursor.fetchone()
        cursor.close()
        cnx.close()
        return int(row[3])
    except (IOError, TypeError) as e:
        print ("sqlauto IOError TypeError:"+str(e))
    except Exception as e:
        print("sqlauto Error:"+str(e))
##--------------bluetooth setting-------------------------
RX_SERVICE_UUID=UUID('6e400001-b5a3-f393-e0a9-e50e24dcca9e')
RX_CHAR_UUID=UUID('6e400003-b5a3-f393-e0a9-e50e24dcca9e')
TX_CHAR_UUID=UUID('6e400002-b5a3-f393-e0a9-e50e24dcca9e')
```

```
switch_staus="off"

##---------------Fuzzy Setting----------------------------#
# New Antecedent/Consequent objects hold universe variables and membership
# functions
temp =ctrl.Antecedent(np.arange(0, 41, 1), 'temp')
light =ctrl.Antecedent(np.arange(0, 11, 1), 'light')
hum =ctrl.Antecedent(np.arange(0, 81, 1), 'hum')
switch_power = ctrl.Consequent(np.arange(0, 11, 1), 'switch_power')
temp.automf(3)
light.automf(3)
hum.automf(3)
#switch_power.automf(3)
switch_power['low'] = fuzz.trimf(switch_power.universe, [0, 0, 5])
switch_power['medium'] = fuzz.trimf(switch_power.universe, [0, 5, 10])
switch_power['high'] = fuzz.trimf(switch_power.universe, [5, 10, 10])

rule1 = ctrl.Rule(temp['good'],switch_power['high'])
rule2 = ctrl.Rule(temp['average'] | hum['good'] ,switch_power['high'])
rule3 = ctrl.Rule(temp['poor'], switch_power['low'])
rule4 = ctrl.Rule(light['poor'], switch_power['low'])
tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3,rule4])
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)



# Pass inputs to the ControlSystem using Antecedent labels with Pythonic API
# Note: if you like passing many inputs all at once, use .inputs(dict_of_data)
#quality['average'].view()
#service.view()
#tip.view()
#rule1.view()
# Crunch the numbers
##---------------fuzzy setting end-----------------------------------------#

#-------------Senor Grove pi  Setting----------------------------#
#---Connect Analog Port--------------------------------
#Analog Port A0~A2
```

```
Sound_Senor_Port    = 0   # Connect the Sound sensor  to port A0
Air_Senor_Port      = 1   # Connect the Air sensor    to port A1
Light_Senor_Port    = 2   # Connect the Light sensor  to port A2
#--------------------------------------------------

#---Connect Digital Port-------------------------------
#Digital Port D2~D8
DHT_Senor_Port      = 2   # Connect the DHT sensor    to port D2
Green_LED_Port      = 3   # Connect the GREEN LED     to port D3
RED_LED_Port        = 4   # Connect the RED LED       to port D4
#--------------------------------------------------
#-------------Senor Grove pi setting End--------------------------#

try:
    Light_value=0
    temp=0
    hum=0
    Sound_value=0
    Air_value=0
    while Light_value == 0:
        Light_value = grovepi.analogRead(Light_Senor_Port)
    while Sound_value == 0:
        Sound_value = grovepi.analogRead(Sound_Senor_Port)
    while Air_value == 0:
        Air_value = grovepi.analogRead(Air_Senor_Port)
    if  Light_value != 0:
        resistance  = (float)(1023 - Light_value) * 10 / Light_value
    else :
        resistance  = 0
    while temp == 0:
        [t, h] = grovepi.dht(DHT_Senor_Port,0)
        temp          = float(t)
        hum          = float(h)

    Sound=Sound_value/100
    Air=Air_value/100
    light=Light_value/100
```

```
    tipping.input['temp'] = float(temp)
    tipping.input['light'] =float(light)
    tipping.input['hum'] = float(hum)
    tipping.compute()
    output=float(tipping.output['switch_power'])
    print('temp=%f, ‘hum=%f‘, ‘light=%f‘, ‘sound=%f‘, ‘air=%f‘, ‘output=%f‘' %
(temp,hum,light,Sound,Air,output))
    last=sqlget()
    print ('last=%s' % last)

    if output >=5 :
        print ("on")
        switch_staus="on"
        if(sqlauto(1)==1):
            print("Auto Model")
            with Peripheral('C1:8F:CC:DC:57:19','random') as p:
                ch=p.getCharacteristics(uuid=TX_CHAR_UUID)[0]
                val=binascii.a2b_hex('AA0630306655')
                ch.write(val)
                p.disconnect()
            if(last!=switch_staus):
                print ("Sql Insert On")
                sqlinsert(1,temp,hum,light,Sound,Air,output,'on')
        else:
            print("Manual Model")
    else :
        print ("off")
        switch_staus="off"
        if(sqlauto(1)==1):
            print("Auto Model")
            with Peripheral('C1:8F:CC:DC:57:19','random') as p:
                ch=p.getCharacteristics(uuid=TX_CHAR_UUID)[0]
                val=binascii.a2b_hex('AA0630316755')
                ch.write(val)
                p.disconnect()
            if(last!=switch_staus):
                print ("Sql Insert off")
```

```
                sqlinsert(1,temp,hum,light,Sound,Air,output,'off')
        else :
            print("Manual Model")
except (IOError, TypeError) as e:
    print ("Main IOError TypeError:"+str(e))
    #print (e)
except Exception as e:
    print("Main Error:"+str(e))
finally :
    print ("Finsh")
    time.sleep(10)
```

Python 模糊邏輯環境偵測紀錄-程式碼

```
from __future__ import print_function
from grovepi import *
from grove_rgb_lcd import *
import time
import grovepi
import  datetime
import mysql.connector
#----
# Get sensor value




#----
#---Port Setting Start---------------------------------

#---Connect Analog Port--------------------------------
#Analog Port A0~A2
sound_sensor_Port   = 0   # Connect the Sound sensor  to port A0
Air_Senor_Port      = 1   # Connect the Air sensor    to port A1
Light_Senor_Port    = 2   # Connect the Light sensor  to port A2
#-------------------------------------------------------

#---Connect Digital Port-------------------------------
#Digital Port D2~D8
DHT_Senor_Port      = 2   # Connect the DHT sensor    to port D2
Green_LED_Port      = 3   # Connect the GREEN LED     to port D3
RED_LED_Port    = 4   # Connect the RED LED       to port D4
#-------------------------------------------------------


#---Port Setting End-----------------------------------

digitalWrite(Green_LED_Port,1)
#---Programm Start-------------------------------------
try:
```

```
#---init Var--------
sound_sensor=0
Air_value=0
Light_value=0
temp=0
hum=0

digitalWrite(RED_LED_Port,0)

grovepi.pinMode(sound_sensor_Port,"INPUT")        # Setting Sound Senor pinmode
grovepi.pinMode(Light_Senor_Port,"INTPUT")       # Setting Light Senor pinmode
#---init Var End----
time.sleep(5)
while Light_value == 0:
    Light_value = grovepi.analogRead(Light_Senor_Port)

if    Light_value != 0:
    resistance  = (float)(1023 - Light_value) * 10 / Light_value
else :
    resistance  = 0

while sound_sensor == 0:
    sound_sensor = grovepi.analogRead(sound_sensor)
while Air_value == 0:
    Air_value = grovepi.analogRead(Air_Senor_Port)

#Temp and Hum  String
while temp == 0:
    [t, h] = grovepi.dht(DHT_Senor_Port,0)
    temp      = str(t)
    hum      = str(h)
time.sleep(.5)
light     = str(Light_value)#Light Senor   String
resistancestr = str(resistance) # Calculate resistance of sensor in light
Air       = str(Air_value)#Air   Senor   String
Sound     = float(sound_sensor)#Sound Senor   String
today     = str(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))    # DateTime
```

```
    add_test = ("INSERT INTO `dayrecored`"
           "(`id`, `Date`, `device`, `temp`, `hum`, `light`,
`dark`,`sound`,air_quality)"
           " VALUES (NULL,%s,'RaspberryPi',%s,%s,%s,%s,%s,%s)")
    data_test = ( today,temp,hum,light,resistance,Sound,Air)
    #---My SQL Start-----------------------------------------------------------------------
--------------------
    #My SQL Setting
    cnx = mysql.connector.connect(user='abc',
password='1234',host='120.110.7.27',database='smarttimersocket')
    cursor = cnx.cursor()

    #My SQL Command
    cursor.execute(add_test, data_test)
    cursor.close()
    cnx.close()
    #---My SQL End-------------------------------------------------------------------------
--------------------
    #---View Print Variors-----------------------------------------------------------------
--------------------
    print("-------------------------------------------------")
    print("DateTime=%s" %today)
    print("Temp=%s Hum=%s" % (temp,hum) )
    print("light=%s resistancestr=%s" % (light,resistancestr) )
    print("Air Value=%s " % Air)
    print("Sound Value=%d " % Sound)
    print("-------------------------------------------------")
    print("SQL-------------------------------------------")
    print(add_test % data_test)
    print("")
    #---View Print Variors-----------------------------------------------------------------
--------------------
except (IOError, TypeError) as e:
    print ("IOError TypeError:"+str(e))
    #print (e)
```

```
except Exception as e:
    print("Error:"+str(e))
    print("Error:"+str(e))
    f = open('recored.txt','r+')
    a=(add_test % data_test)+'\n'
    print(f.read()+str(a))
    f.write(f.read()+str(a))
    f.close()
finally :
    print ("Finsh")
#---Programm End-----------------------------------
    time.sleep(10)
```

Python 文字檔紀錄產生-程式碼

```
import mysql.connector


try :
    with open('file.txt','r+') as fp:
        for line in fp:
            print(line)
            cnx = mysql.connector.connect(user='abc',
password='1234',host='120.110.7.27',database='smarttimersocket')
            cursor = cnx.cursor()
            cursor.execute(line)
            cnx.commit()
            cursor.close()
            cnx.close()
        fp.close
        f = open('file.txt','w+')
        f.close
    with open('recored.txt','r+') as rep:
        for line in rep:
            print(line)
            cnx = mysql.connector.connect(user='abc',
password='1234',host='120.110.7.27',database='smarttimersocket')
            cursor = cnx.cursor()
            cursor.execute(line)
            cnx.commit()
            cursor.close()
            cnx.close()
        rep.close
        ref = open('recored.txt','w+')
        ref.close
except (IOError, TypeError) as e:
    print ("IOError TypeError:"+str(e))
except Exception as e:
    print("Error:"+str(e))
finally :
    print("Finall")
```