修 平 科 技 大 學
電 機 工 程 系
DEPARTMENT OF ELECTRICAL ENGINEERING
HSIUPING UNIVERSITY OF SCIENCE AND TECHNOLOGY

實 務 專 題 報 告 書

應用 iBeacon 藍牙識別技術於智慧電梯控制

指 導 老 師：林助訓

專題製作學生:

四技電三乙　黃偉翔　BD104022

四技電三乙　朱鴻旻　BD104107

中 華 民 國　107　年　6　月　25　日

修平科技大學

電機工程系

HSIU-PING UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING

指導老師：林助訓

專題製作學生：黃偉翔、朱鴻旻

製作日期：107 年 6 月 25 日

摘要

隨著時代的進步手機是每個人都必備的一個物品，又加上近幾年來物聯網的應用。而本專題使用 Beacon 及 App Invetor 2 這種設計 app 又成本便宜的訊號零件，來使生活更方便，我們這次的主題是把這 2 種東西結合在電梯控制上，避免發現現代人常常遺失瓷釦要再賠償磁扣的相關費用，以及忘記電梯密碼的困境，手機人人都有幾乎可以代替磁扣。

目錄

第一章 序言

1.1 研究動機

科技是為了使人們更方便的生活，換句話說就是科技的進步使得人們的生活可以更加便利，電梯是現在社會不可或缺的一部分，但大多數的電梯都有管制，電梯車廂內設置一密碼鍵盤，一般住戶通常要記住電梯通行密碼，男生手太大常按錯，小孩在玩亂按，老人要記住通行密碼是一個大挑戰。電梯樓層管制系統普遍使用射頻 IC 卡，攜帶困難，使用者往往需要持有多張射頻 IC 卡，造成不便，常丟卡掛失、黑名單等多個複雜的管理環節，難道沒有更方便的方法嗎?

1.2 研究目的

我們藉由 PLC 和 Arduino 藍芽裝置及手機 APP，來完成智慧電梯控制，並透過程式撰寫的方式，以達到所需的功能，現在人手一機的人們不用再擔心遺失瓷釦、密碼卡等;另外我們設有聲控模式讓行動不見者可以直接利用聲控來操控電梯裝置。藉由這次的計畫達到電梯樓層管制達到三不:

1.不必刷卡及磁釦

2.不必備密碼

3.不必怕遺失

1.3 架構圖

圖 1.1 架構圖

第二章 PLC 介紹及應用

2.1 PLC 可程式邏輯控制器介紹

可程式邏輯控制器（Programmable Logic Controller，簡稱 PLC)，是一種具有微處理器的數位電子裝置，用於自動化控制的數位邏輯控制器，可以將控制指令隨時載入記憶體內儲存與執行。可程式控制器由內部 CPU，指令及資料記憶體、輸入輸出單元、電源模組、數位類比等單元所模組化組合成。PLC 可接收（輸入）及發送（輸出）多種型態的電氣或電子訊號，並使用他們來控制或監督幾乎所有種類的機械與電氣系統。最初的可程式化序邏輯控制器只有電路邏輯控制的功能，所以被命名為可程式邏輯控制器，後來隨著不斷的發展，這些當初功能簡單的電腦模組已經有了包括邏輯控制，時序控制、模擬控制、多機通訊等許多的功能，名稱也改為可程式控制器，但是由於它的簡寫也是 PC 與個人電腦（Personal Computer）的簡寫相衝突，也由於多年來的使用習慣，人們還是經常使用可程式邏輯控制器這一稱呼，並在術語中仍沿用 PLC 這一縮寫。

在可程式邏輯控制器出現之前，一般要使用成百上千的繼電器以及計數器才能組成具有相同功能的自動化系統，而現在，經過編

程的簡單的可程式邏輯控制器模組基本上已經代替了這些大型裝置。可程式邏輯控制器的系統程式一般在出廠前已經初始化完畢，用戶可以根據自己的需要自行編輯相應的用戶程式來滿足不同的自動化生產要求。

現在工業上使用可程式邏輯控制器已經相當接近於一台輕巧型電腦所構成，甚至已經出現整合個人電腦（採用嵌入式作業系統）與 PLC 結合架構的可程式自動化控制器（Programmable Automation Controller，簡稱 PAC)，能透過數位或類比輸入/輸出模組控制機器裝置、製造處理流程及其他控制模組的電子系統。可程式邏輯控制器廣泛應用於目前的工業控制領域。在工業控制領域中，PLC 控制技術的應用已成為工業界不可或缺的一員。

2.2 PLC 架構

一般講，PLC 分為箱體式和模組式兩種。但它們的組成是相同的，對箱體式 PLC，有一塊 CPU 板、I/O 板、顯示面板、記憶體塊、電源等，當然按 CPU 效能分成若干型號，並按 I/O 點數又有若干規格。對模組式 PLC，有 CPU 模組、I/O 模組、記憶體、電源模組、底板或機架。無論哪種結構類型的 PLC，都屬於匯流排式開放型結構，其 I/O 能力可按用戶需要進行擴展與組合。PLC 的基本結構框圖如下：

電源模組:

有些 PLC 中的電源，是與 CPU 模組合二為一的，有些是分開的，其主要用途是為 PLC 各模組的積體電路提供工作電源。同時，有的還為輸入電路提供 24V 的工作電源。電源如果為交流電源通常為 220VAC 或 110VAC，若為直流電源常用的為 24V。

中央處理單元:

PLC 中的 CPU 是 PLC 的核心，它按 PLC 的系統程式賦予的功能接收並存貯用戶程式和資料，用掃描的方式採集由現場輸入裝置送來的狀態或資料，並存入規劃的暫存器中，同時，診斷電源和 PLC 內部電路的工作狀態和編程過程中的語法錯誤等。進入運行後，從用戶程式存貯器中逐條讀取指令，經分析後再按指令規定的任務產生相應的控制訊號，去指揮有關的控制電路，與個人電腦一樣，主要由運算器、控制器、暫存器及實現它們之間聯繫的資料、控制及狀態匯流排構成，還有週邊晶片、匯流排介面及有關電路。它確定了進行控制的規模、工作速度、記憶體容量等。

記憶體:

記憶體主要用於存儲程式及資料，是 PLC 不可缺少的組成單元。PLC 內部會存放撰寫完成編輯的程式指令及資料，通常也可使用 RAM 或 EEPROM 等專用記憶體卡片方式擴充，但擴充能力得依各廠牌與型號有所不同。

輸入/輸出單元:

輸入單元是用來連結擷取輸入元件的訊號動作並透過內部匯流排將資料送進記憶體由 CPU 處理驅動程式指令部分。PLC 輸入模組 PLC 系統的架構和輸入模組產品的選擇端視需要被監測的輸入訊號位準而定。來自不同類型被監測的感測器與流程控制之變量訊號，可以涵蓋從±10mV 至±10V 的輸入訊號範圍。

輸出單元是用來驅動外部負載的介面，主要原理是由 CPU 處理以書寫在 PLC 裡的程式指令，判斷驅動輸出單元在進而控制外部負載，如指示燈、繼電器、氣（油）壓閥等。

PLC 輸出模組在工業環境中用來控制制動器及馬達等的 PLC 系統類比輸出範圍包括±5V、±10V、0V 到 5V、0V 到 10V、4 到 20mA 或 0 到 20mA 等。

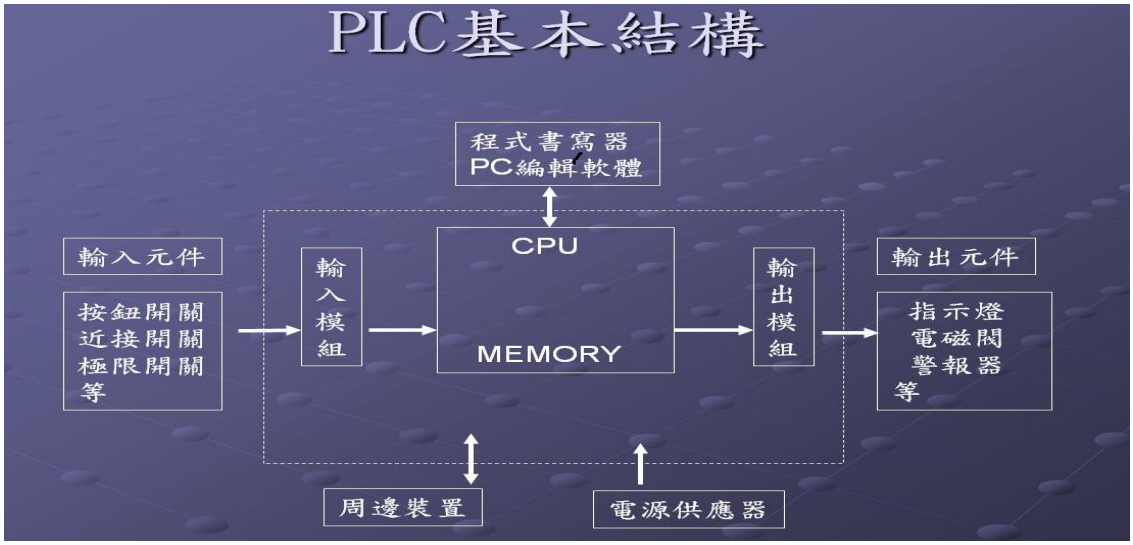


圖 2.1 PLC 基本結構

通訊:

現在 PLC 大多具有可擴充通訊網路模組的功能，簡單的 PLC 以 BUS 纜線或 RS-232 方式通訊連結，較高階的 PLC 會採用 USB 或乙太網路方式做通訊連結。它使 PLC 與 PLC 之間、PLC 與個人電腦以及其他智慧型裝置之間能夠交換資訊，形成一個統一的整體，實現分散集中控制。現在幾乎所有的 PLC 新產品都有通訊網路功能，它和電腦一樣具有 RS-232 介面，通過雙絞線、同軸電纜或光纜，可以在幾公里甚至幾十公里的範圍內交換資訊。PLC 通訊協定規格可分為 RS-232、RS-422、RS-432、RS-485…等等，其中 RS-432 最為少見。目前國際中最常用的通訊協定為 MODBUS-ASCII 模式及 MODBUS-RTU 模式，此為 Modicon 公司所制定的通訊協定。PROFIBUS 則為西門子公司所制定。

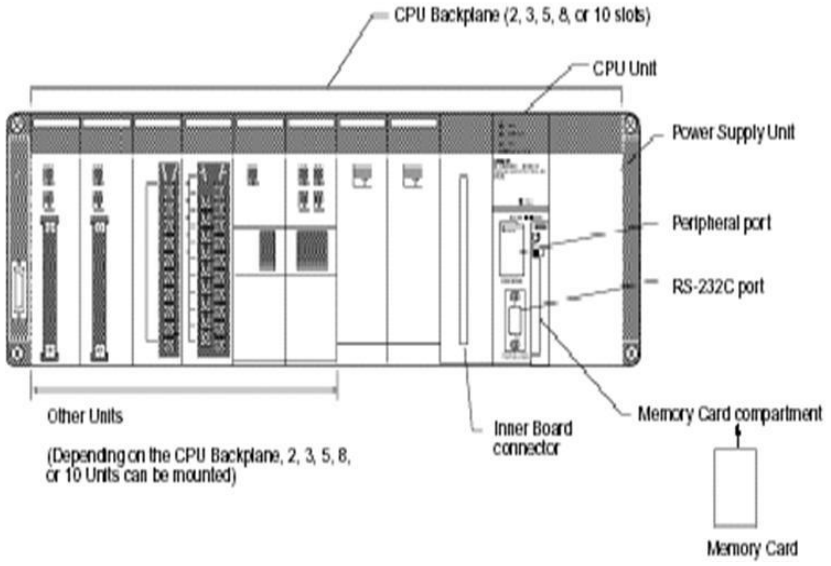


圖 2.2 PLC 硬體構成

2.3 PLC 程式撰寫

PLC 的編程程式語言與一般電腦程式語言相比，具有明顯的特點，它既不同於高階語言，也不同與一般的組合語言，它既要滿足易於編寫，又要滿足易於調試的要求。目前，還沒有一種對各廠家產品都能相容的編程語言。IEC 61131-3 是一個國際標準，它規範了 PLC 相關之軟體硬體的標準，其最終的目的是可以讓 PLC 的使用者在不更改軟體設計的狀況下可以輕易更換 PLC 硬體。IEC 61131-3 主要是提供了五種程式語言，包含：

1. **指令表**（Instruction List，IL 或 Statement List，SL）：類似組合語言的描述文字。由指令語句系列構成，如 Mitsubishi FX2 的控制指令 LD、LDI、AND、ANI、OR、ORI、ANB、ORB、MMP、MMS 與 OUT 等，一般配合書寫器寫入程式，而書寫器只能輸入簡單的指令，與電腦程式中的階梯圖比較起來簡單許多。書寫器不太直觀，可讀性差，特別是遇到較複雜的程式，更難讀；

2. **結構式檔案編程語言**（Structured Text，ST）：類似 PASCAL 與 C 語言的語法，適合撰寫較複雜的演算法，除錯上也比階梯圖要容易得多。ST 語言類似於程式語言的特性，因此可利用與微電腦及個人電腦相同的程式設計技術進行階梯式語言所難以執行的複雜計算，完成程式的建立。自保持

迴路的階梯圖，當開關 ON 觸動後，電燈即自我保持在輸出，直到開關 OFF 觸動才會切斷。

3. **順序功能流程圖**（Sequential Function Chart，SFC）：類似於流程設計（Flow Design），流程圖中的步驟組合而完成，主要是規劃動作順序的流程圖，故謂之順序功能流程圖。所謂步序式控制，即是一步一步控制，而這一步與上一步是有關連性的，有順序性的。必須有上一個動作(STL)，才會啟動（SET）下一個動作（STL）。

4.**功能區塊圖**（Function Chart Programming，FBD）：以畫電路圖的方式來寫 PLC 程式。常用的程式及迴路可透過 FB（功能區塊）的建立輕易地重複利用。

其他一些高檔的 PLC 還具有與電腦相容的 C 語言、BASIC 語言、專用的高階語言(如西門子公司的 GRAPH5、三菱公司的 MELSAP、富士電機的 Micrex-SX 系列），還有用布林邏輯語言、通用電腦相容的組合語言等。

2.3.1 書寫方式

階梯圖（Ladder Programming，LAD)：類似於傳統上以繼電器控制接觸器的階梯圖，梯形圖是通過連線把 PLC 指令的梯形圖符號連接在一起的連通圖，用以表達所使用的 PLC 指令及其前後順序，它與電氣原理圖很相似。它的連線有兩種：一為母線，另一為內部橫豎線。內部橫豎線把一個個梯形圖符號指令連成一個指令組，這個指令組一般總是從裝載（LD）指令開始，必要時再繼以若干個輸入指令（含 LD 指令)，以建立邏輯條件。最後為輸出類指令，實現輸出控制，或為資料控制、流程控制、通訊處理、監控工作等指令，以進行相應的工作。

2.4 PLC 階梯圖製作

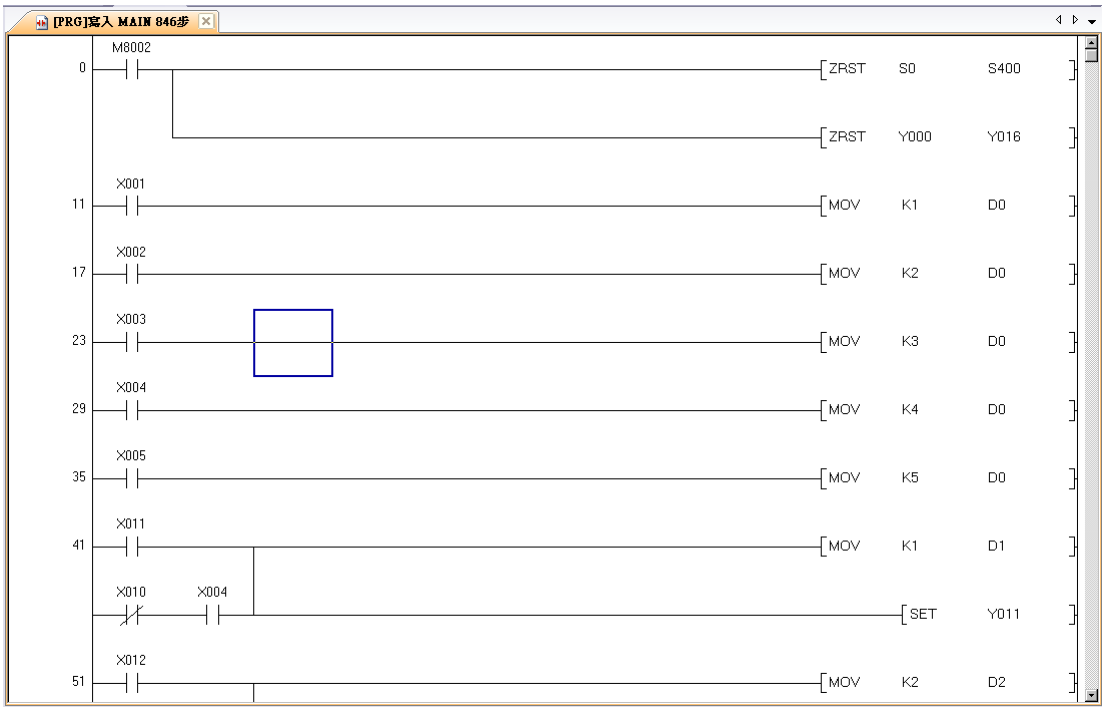


圖 2.3PLC 階梯圖

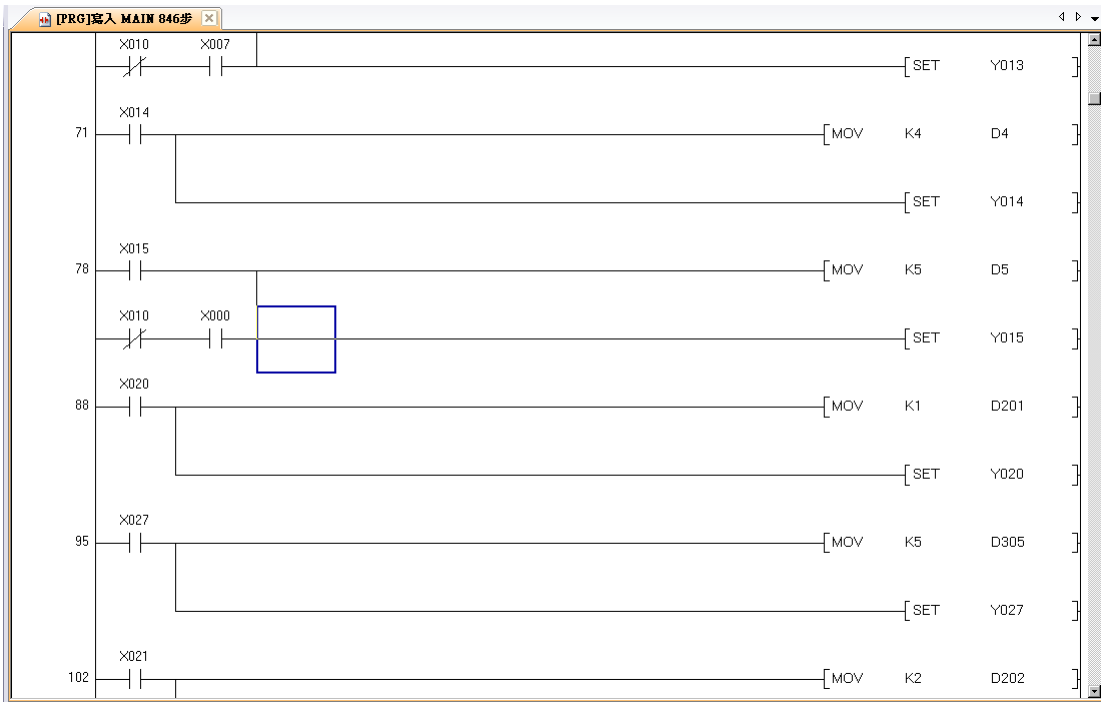


圖 2.4PLC 階梯圖

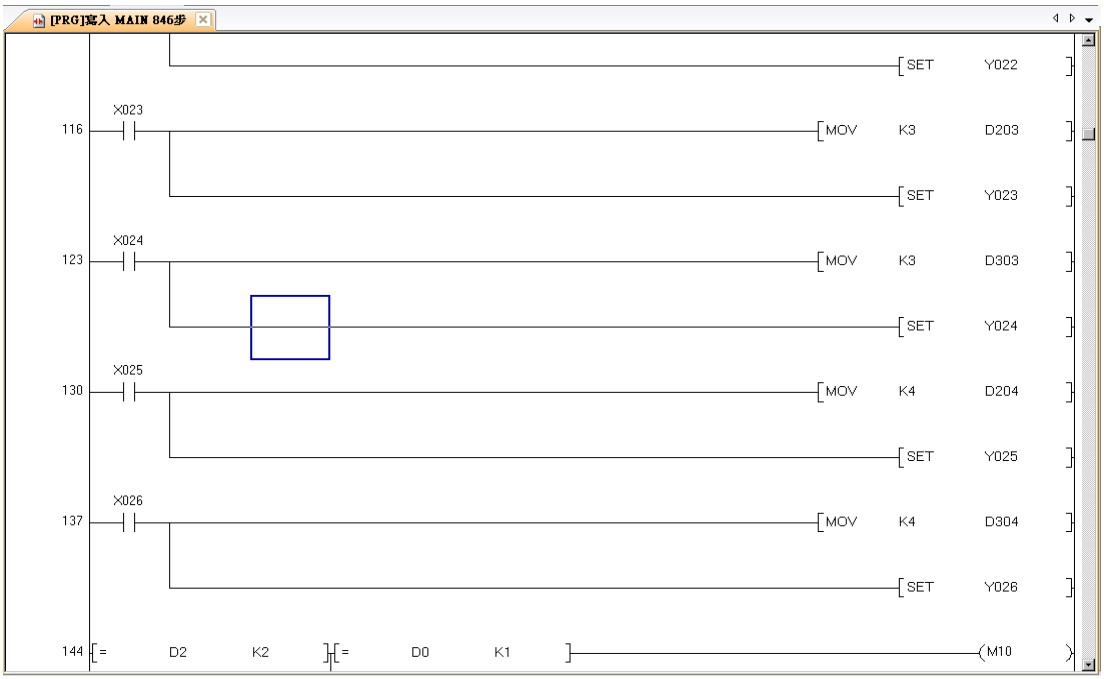


圖 2.5PLC 階梯圖

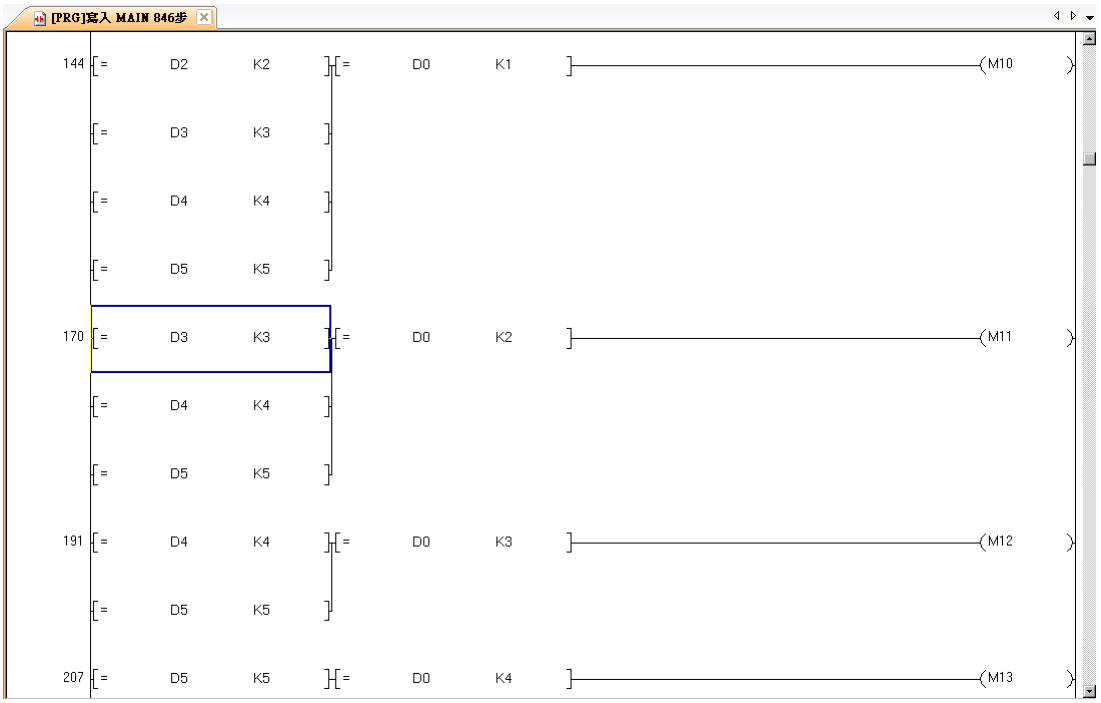


圖 2.6PLC 階梯圖

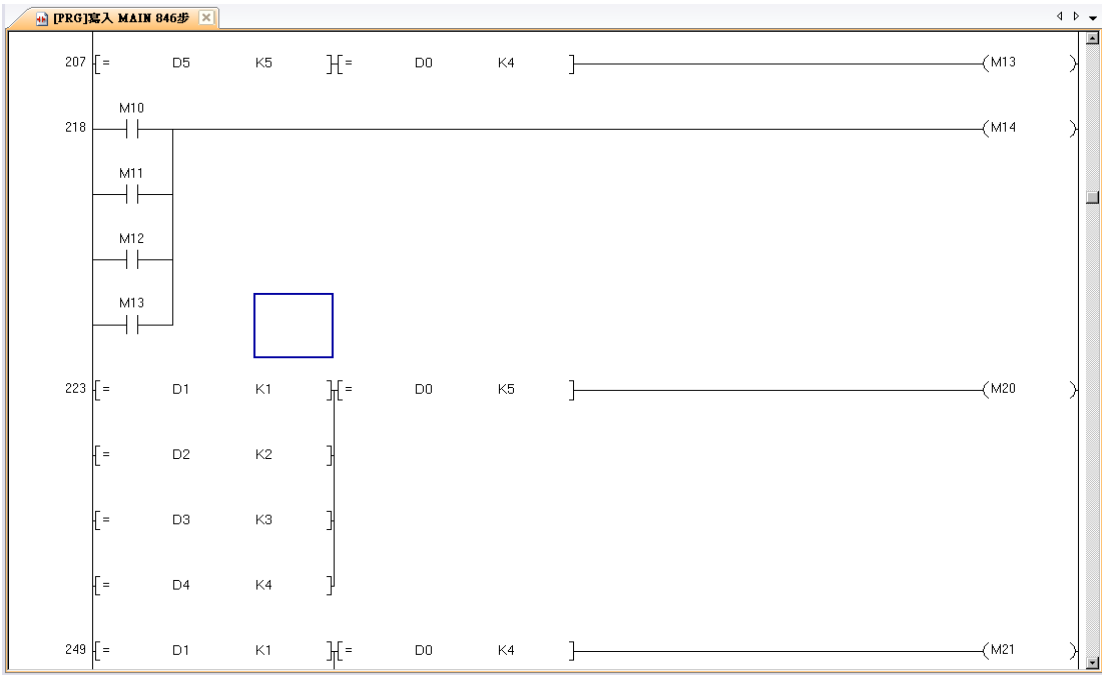


圖 2. 7PLC 階梯圖

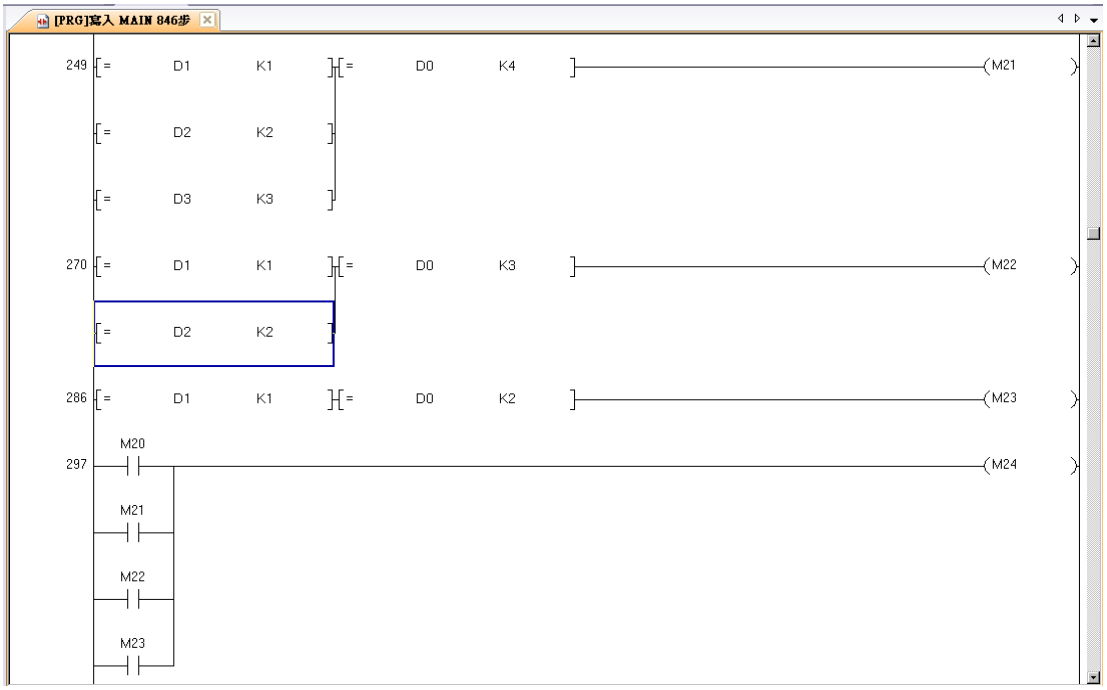


圖 2. 8PLC 階梯圖

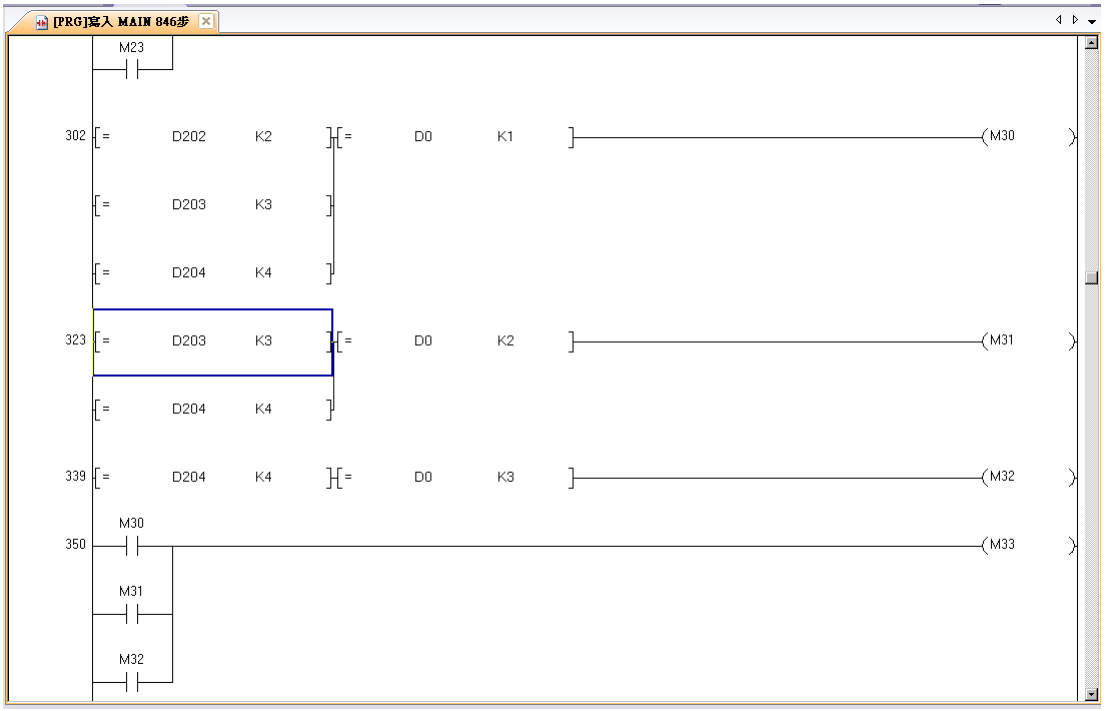


圖 2.9PLC 階梯圖

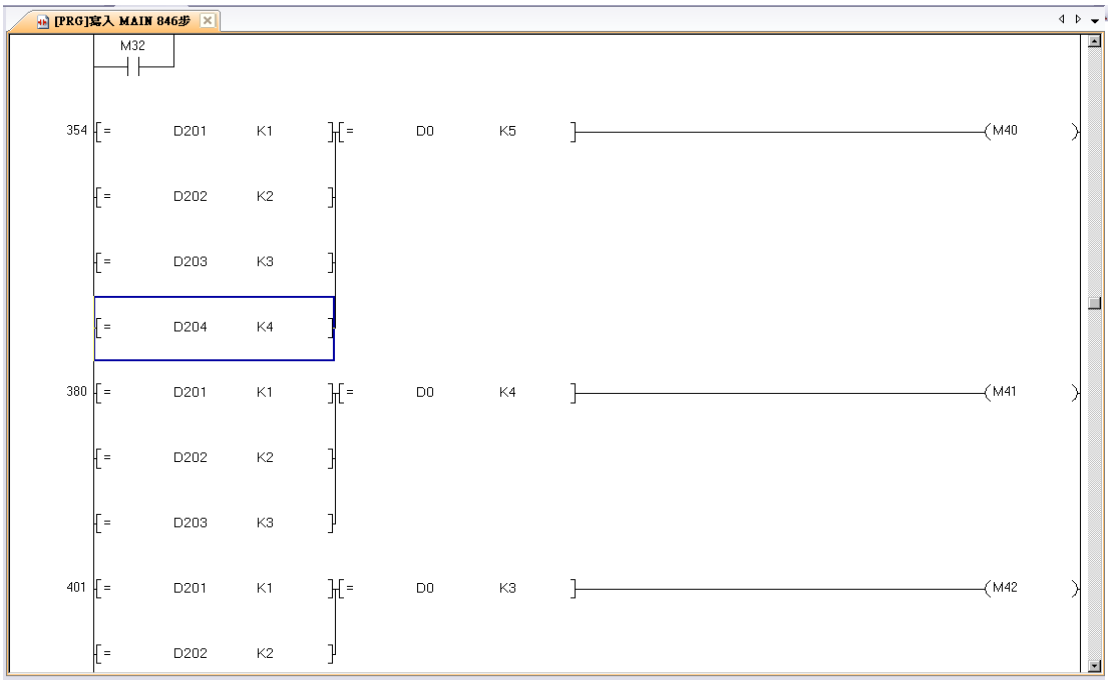


圖 2.10PLC 階梯圖

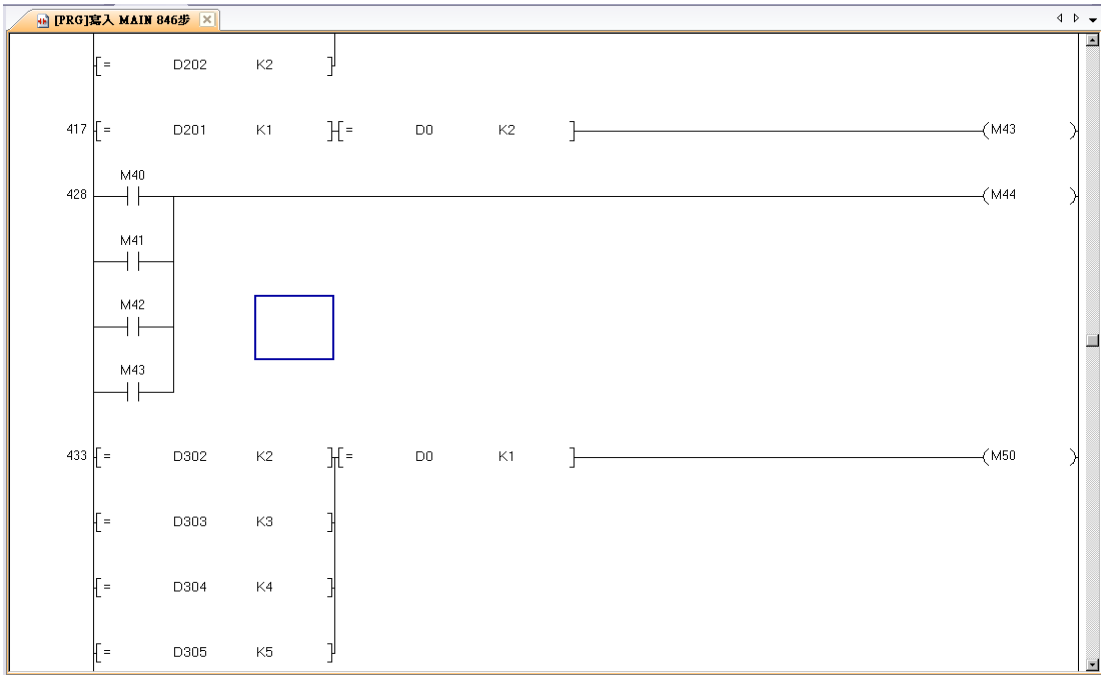


圖 2.11PLC 階梯圖

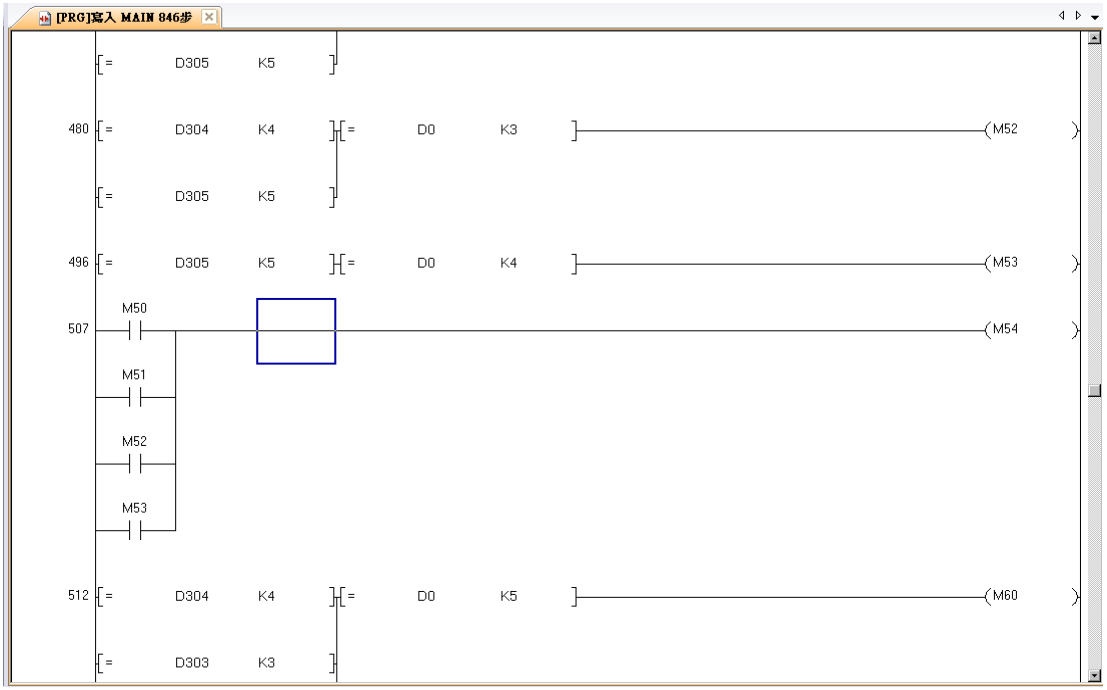


圖 2.12PLC 階梯圖

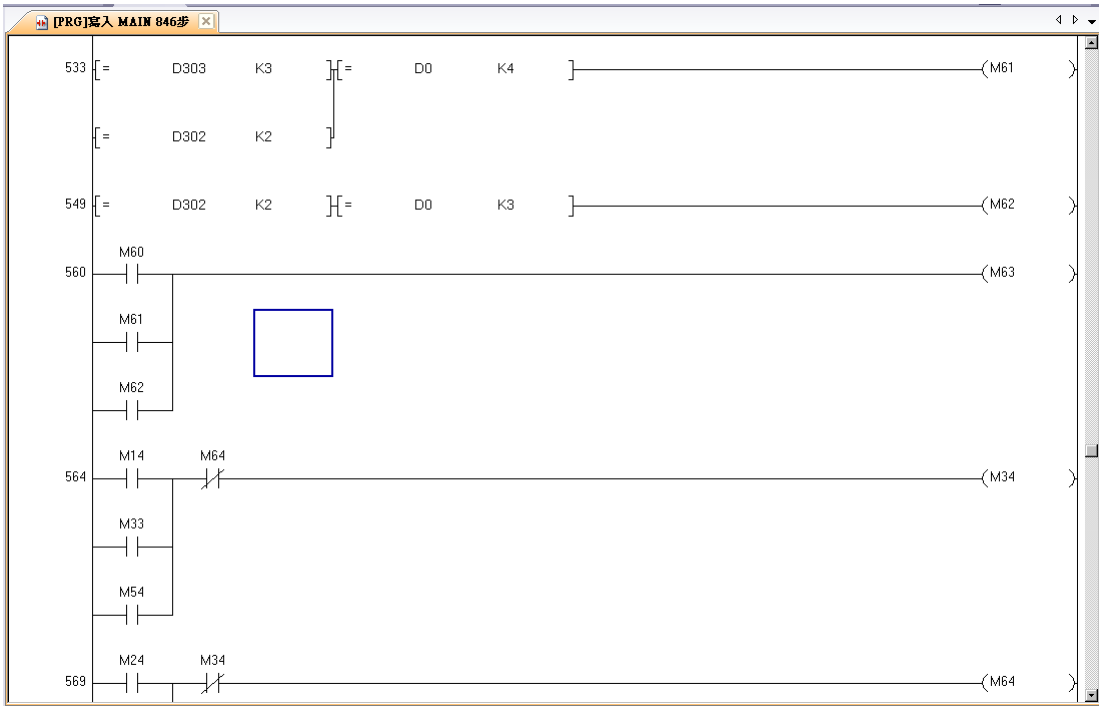


圖 2.13PLC 階梯圖

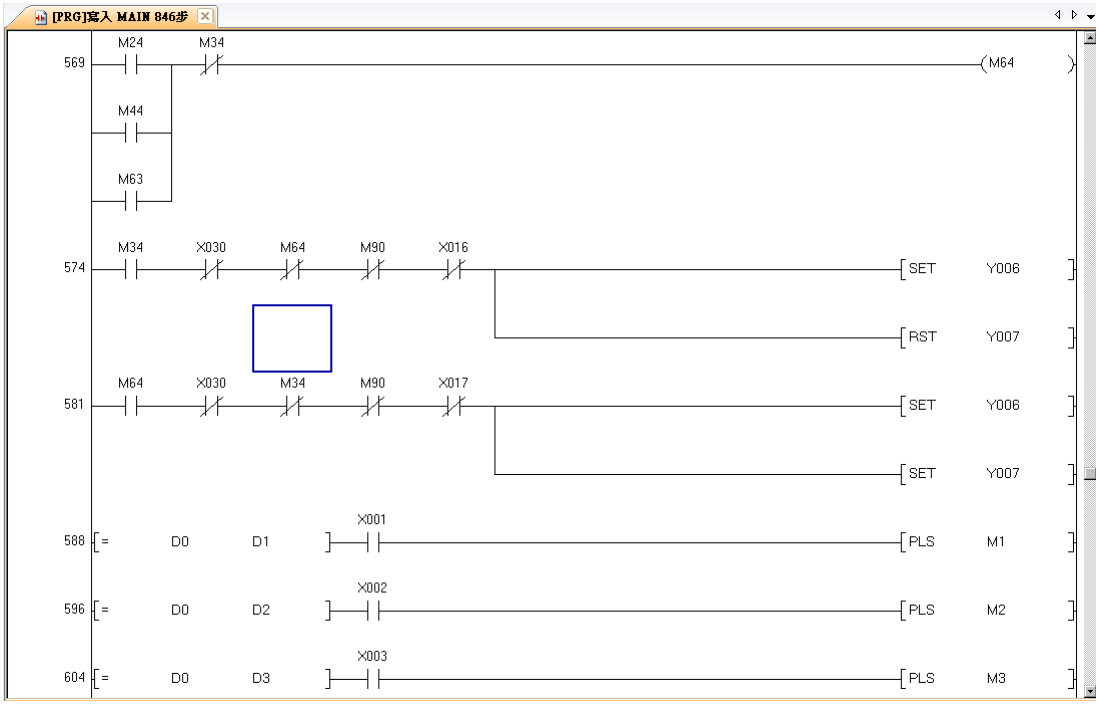


圖 2.14PLC 階梯圖

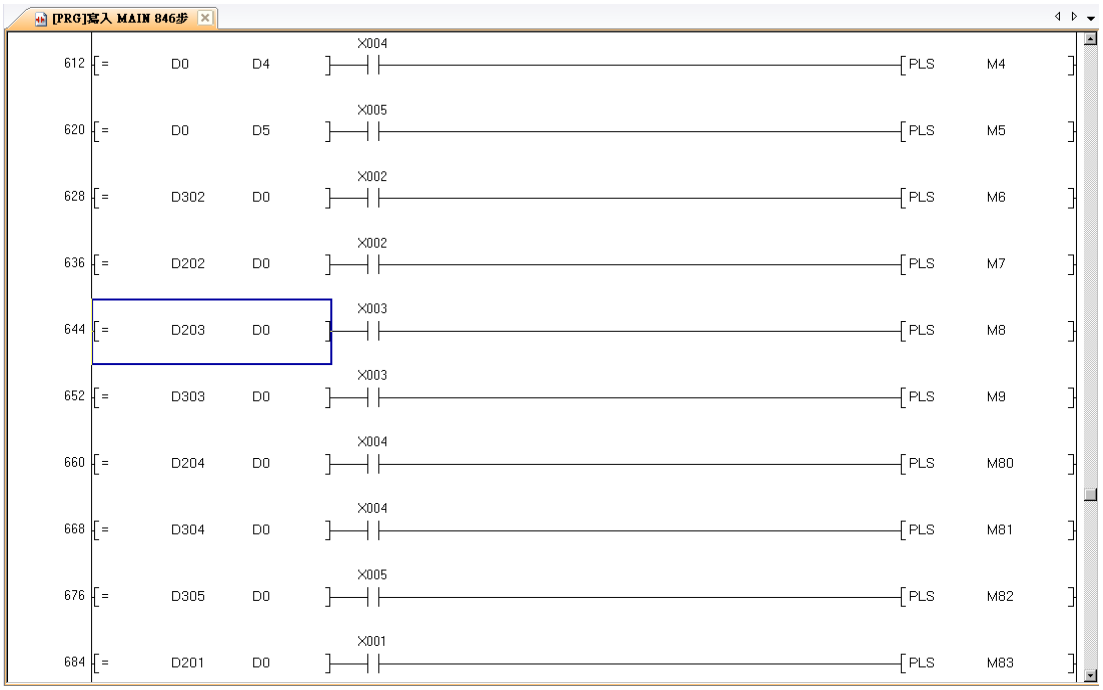


圖 2.15PLC 階梯圖

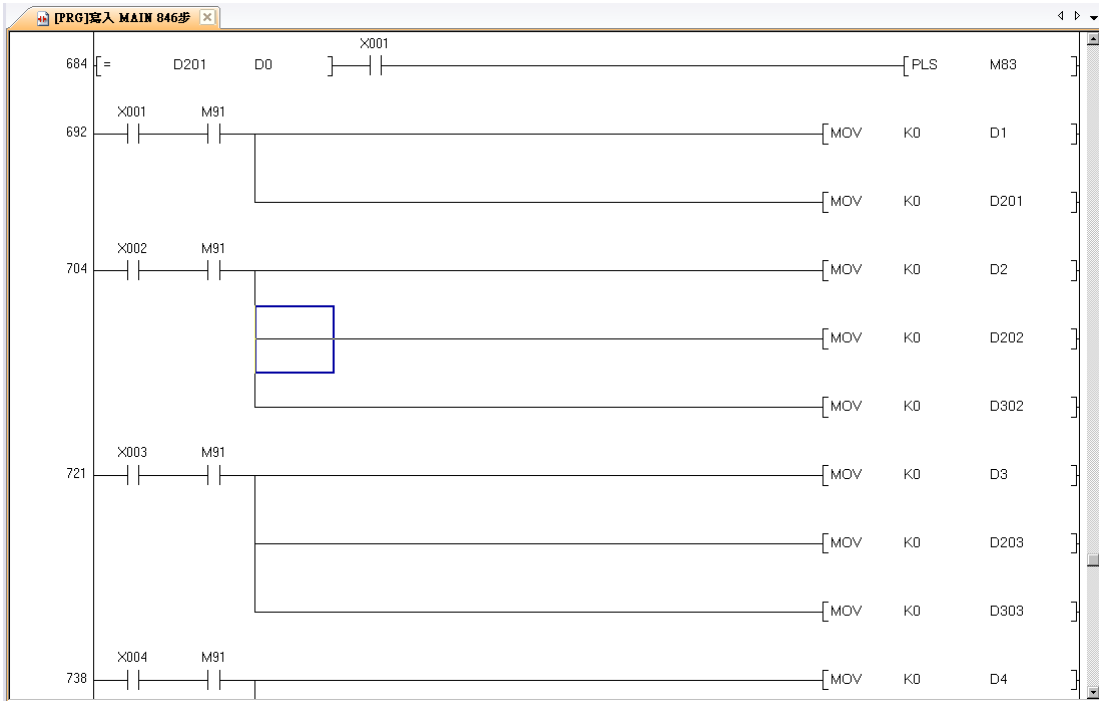


圖 2.16PLC 階梯圖

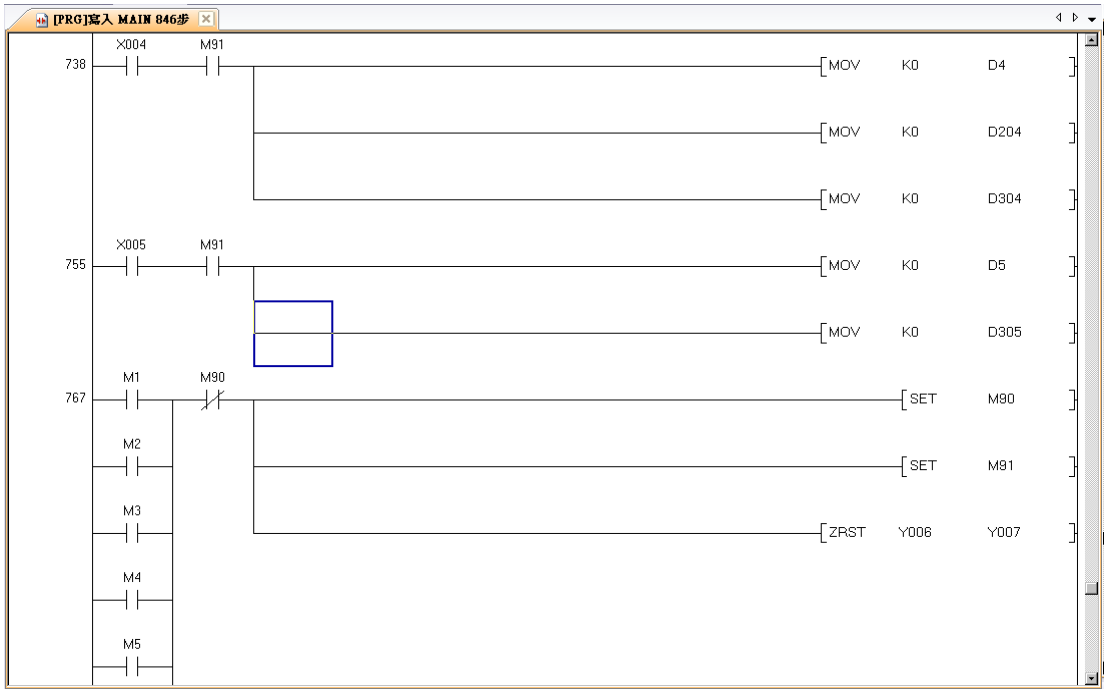


圖 2.17PLC 階梯圖

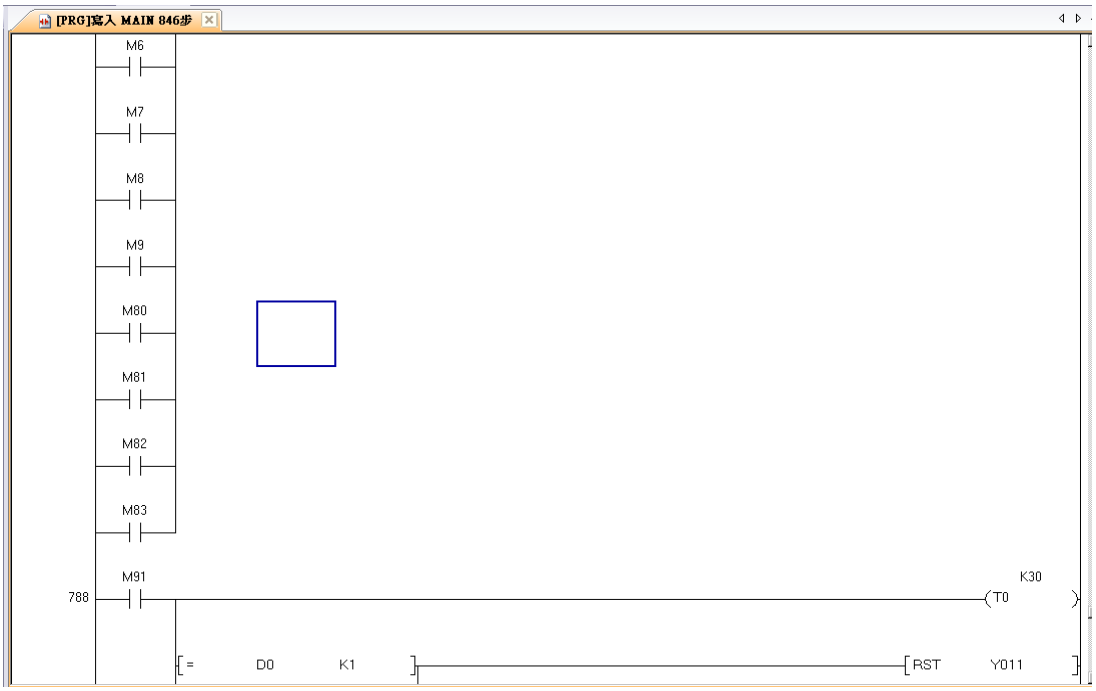


圖 2.18PLC 階梯圖

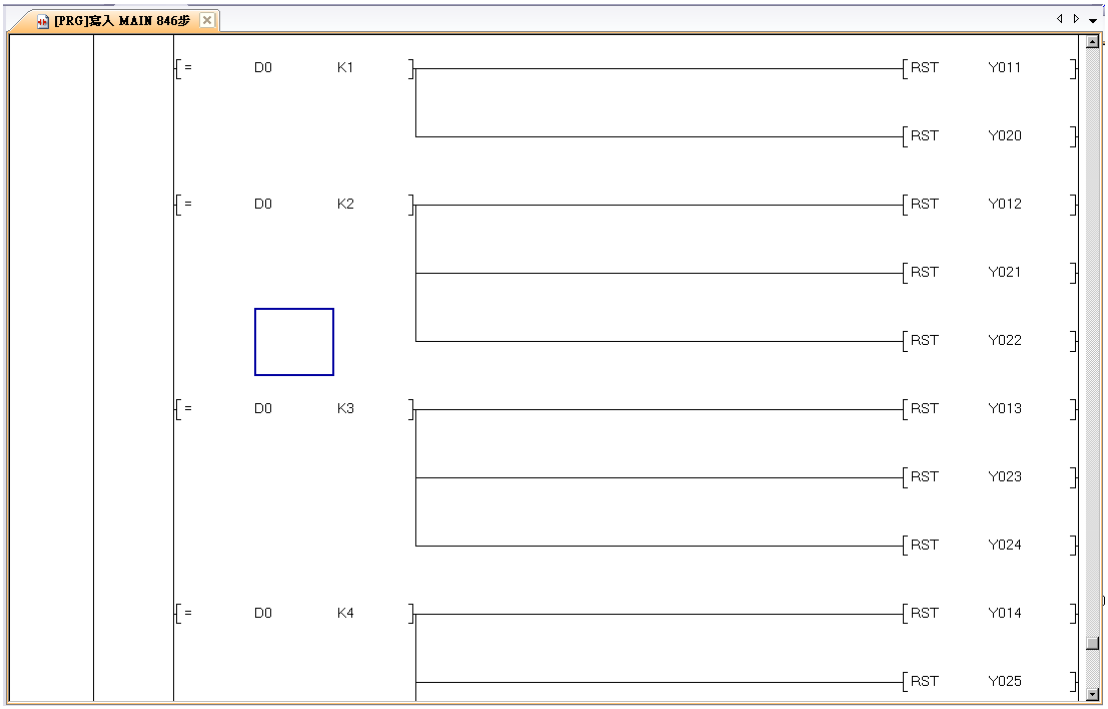


圖 2.19PLC 階梯圖

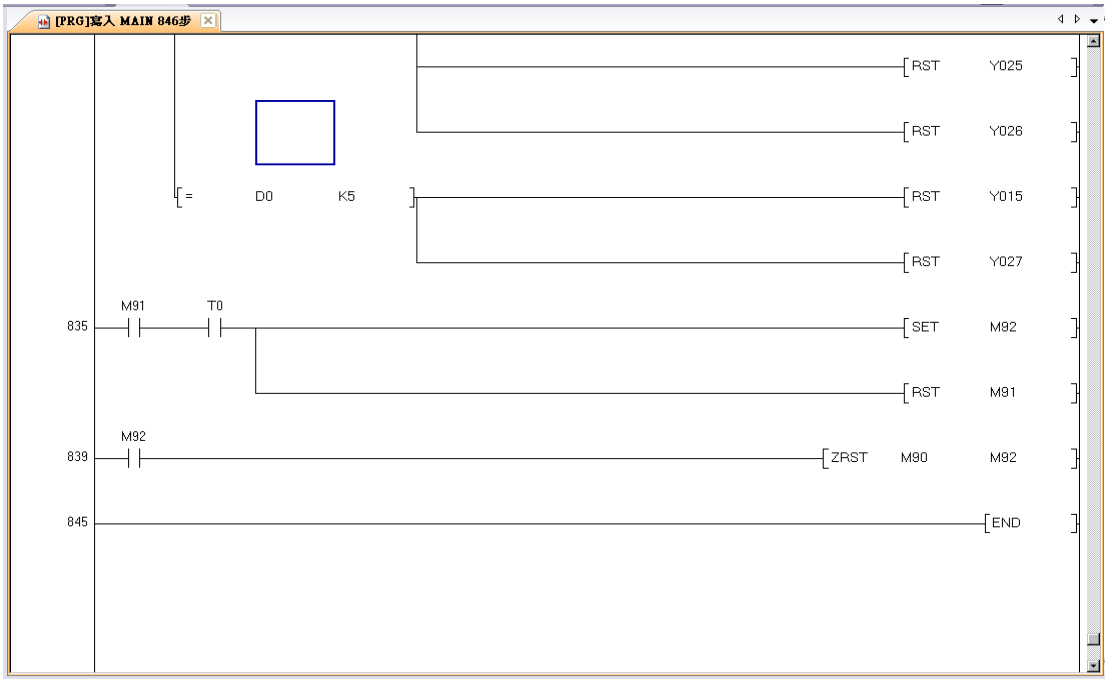


圖 2.20 PLC 階梯圖

第三章 Arduino 介紹及應用

3.1Arduino 介紹

Arduino 是一家製作開源電腦硬體和軟體的公司，同時兼有專案和用戶社群，他負責設計和製造單板微控制器和微控制器套件，用於構建數字裝置和互動式物件，以便在物理和數字世界中感知和控制物件。該專案的產品是按照 GNU 寬通用公共許可證（LGPL）或 GNU 通用公共許可證（GPL） 許可的開源硬體和軟體分發的，arduino 允許任何人製造 Arduino 板和軟體分發。Arduino 電路板設計使用各種微處理器和控制器。這些電路板配有一組數字和模擬輸入/輸出（I/O）引腳，可以連線各種擴充功能板或麵包板（遮蔽板）和其他電路。這些電路板具有串行通訊介面，包括某些型號上的通用序列匯流排（USB），也用於從個人電腦載入程式。微控制器通常使用來自程式語言 C 和 C ++的特徵的方言進行編程。除了使用傳統的編譯工具鏈之外，Arduino 專案還提供了一 Processing 語言專案的整合式開發環境（IDE）。

3.2 Arduino 特色

- 基於創用 CC 開放原始碼的電路圖設計。
- 免費下載，也可依需求自己修改，但需遵照姓名標示。您必須按照作者或授權人所指定的方式，表彰其姓名。
- 依相同方式分享，若您改變或轉變著作，當散布該衍生著作時，您需採用與本著作相同或類似的授權條款。
- Arduino 可使用 ICSP 線上燒入器，將 Bootloader 燒入新的 IC 晶片。
- 可依據 Arduino 官方網站，取得硬體的設計檔，加以調整電路板及元件，以符合自己實際設計的需求。
- 可簡單地與感測器，各式各樣的電子元件連接，如紅外線、超音波、熱敏電阻、光敏電阻、伺服馬達…等。
- 支援多樣的互動程式，如 Adobe Flash, Max/MSP, VVVV, Pure Data, C, Processing…等。
- 使用低價格的微處理控制器（Atmel AVR）（ATMEGA 8,168,328 等）。
- USB 介面，不需外接電源。另外有提供直流（DC）電源輸入。

3.3 Arduino 軟體

在 Arduino 上執行的程式可以使用任何能夠被編譯成 Arduino 機器碼的程式語言編寫。而 Atmel 也提供了數個可以開發 Atmel 微處理機程式的整合開發環境，AVR Studio 和更新的 Atmel Studio。

3.3.1 IDE

而 Arduino 計劃也提供了 Arduino Software IDE，一套以 Java 編寫的跨平台應用軟體。Arduino Software IDE 源自於 Processing 程式語言以及 Wiring 計劃的整合開發環境。它是被設計於介紹程式編寫給藝術家和不熟悉程式設計的人們，且包含了一個擁有語法突顯、括號符合、自動縮排和一鍵編譯並將執行檔燒寫入 Arduino 硬體中的編輯器。

Arduino Software IDE 使用與 C 語言和 C++相仿的程式語言，並且提供了包含常見的輸入/輸出函式的 Wiring 軟體函式庫。在使用 GNU toolchain 編譯和連結後，Arduino Software IDE 提供了一個程式「avrdude」用來轉換可執行檔成為能夠燒寫入 Arduino 硬體的韌體。

3.3.2 Sketch

使用 Arduino Software IDE 編寫的程式被稱為「sketch」。一個典型的 Arduino C/C++ sketch 程式會包含兩個函式，它們會在編譯後合成為 main() 函式：

- *setup()*：在程式執行開始時會執行一次，用於初始化設定。
- *loop()*：直到 Arduino 硬體關閉前會重複執行函式放的程式碼。

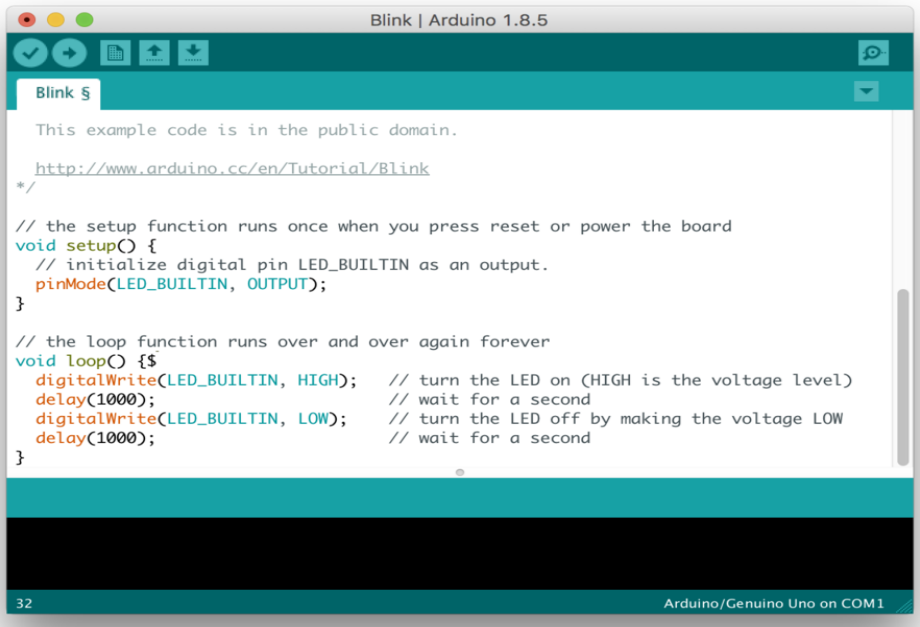


圖 3.1 Arduino IDE

3.4 藍芽介紹及應用

藍牙（英語:Bluetooth）。這是一種無線通訊技術標準，用來讓固定與行動裝置，在短距離間交換資料，以形成個人區域網路（PAN）。其使用短波特高頻（UHF）無線電波，經由 2.4 至 2.485 GHz 的 ISM 頻段來進行通訊。1994 年由電信商愛利信（Ericsson）發展出這個技術。它最初的設計，是希望建立一個 RS-232 資料線的無線通訊替代版本。它能夠連結多個裝置，克服同步的問題。

藍牙技術目前由藍牙技術聯盟（SIG）來負責維護其技術標準，其成員已超過三萬，分布在電信、電腦、網路與消費性電子產品等領域。IEEE 曾經將藍牙技術標準化為 IEEE 802.15.1，但是這個標準已經不再繼續使用

3.4.1 Arduino 及藍芽架構圖

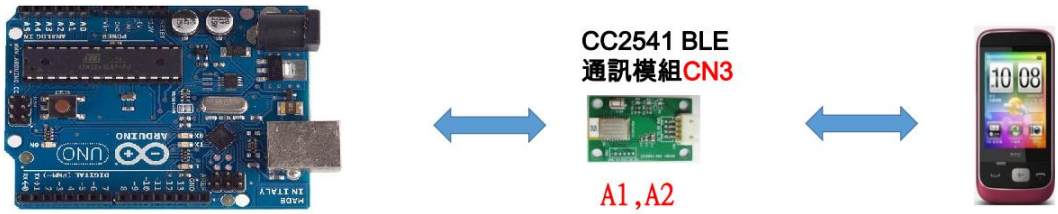


圖 3.2Arduino 架構圖

3.5 Arduino 製作過程

Five-speech-floor

```
//--------------------------------------------------------------------
// Title:Five-speech-floor
// Firmware:BLE41_LED_RGB_BZ
// Devices:NanoKIT + OLED + LED + RGB LED +Buzzer +CC2541(BLE)
// Function: OK
//UpDate:2016/11/11
//--------BLE-----------------------------------------
#include <SoftwareSerial.h>
SoftwareSerial myBLE(A1, A2);
//-------Declare-OLED---------------------------------------------------
#include <OLED.h>
OLED myOLED;
//-------Declare-LED1W-MD1----------------------------------------------
#define Lamp1 8
#define Lamp2 9
#define Lamp3 10
#define Lamp4 11
#define Lamp5 12
#define LED 3
//----------Declare-RGB LED WS2812Driver --------------------------------
#include "FastLED.h"
#define LED_PIN     A0
#define NUM_LEDS    3
#define BRIGHTNESS  64
CRGB leds[NUM_LEDS];
#define UPDATES_PER_SECOND  100
CRGBPalette16 currentPalette;
TBlendType    currentBlending;

extern CRGBPalette16 myRedWhiteBluePalette;
extern const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM;
//-------Declare- Buzzer  ----------------------------------------------
#define buzzerPIN  A3  // CN4  set Buzzer pin A3
//-------Declare-KEY----------------------------------------------
#define  KEY1  2
#define  KEY2  6
int BZState=0;
int LEDState=0;
int RGBState=0;
```

圖 3.3Arduino 程式撰寫

Five-speech-floor

```
int DemoFunc=0;
char fsgRGB='D';
void setup()
{
pinMode(Lamp1,OUTPUT);
pinMode(Lamp2,OUTPUT);
pinMode(Lamp3,OUTPUT);
pinMode(Lamp4,OUTPUT);
pinMode(Lamp5,OUTPUT);

 //-------Setup-RD1W-MD1-----------------------------------------
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH) ;delay(500);digitalWrite(LED, LOW) ;delay(1000);digitalWrite(LED, HIGH) ;delay(500); digitalWrite(LED, LOW) ;
  digitalWrite(Lamp1,LOW);digitalWrite(Lamp2,LOW);digitalWrite(Lamp3,LOW);
  digitalWrite(Lamp4,LOW);digitalWrite(Lamp5,LOW);
//-------Setup-FastLED-------------------------------------------
  delay(100); // power-up safety delay
  FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, NUM_LEDS).setCorrection( TypicalLEDStrip );
  FastLED.setBrightness(  BRIGHTNESS );
  currentPalette = RainbowColors_p;
  currentBlending = LINEARBLEND;
   SET_RGB(1 ,'D');SET_RGB(2 ,'D');SET_RGB(3 ,'D');
 //-------Setup-set OLED FONT_8x16 -------------------------------------------
  myOLED.begin(FONT_8x16); //FONT_6x8 , FONT_8x16
  myOLED.clearScreen();
  myOLED.println("BLE41_LED_RGB_BZ");
   myOLED.println("Dev:Five_Floors");  ////自訂BLE裝置訂編號
 //=========================================
   pinMode(buzzerPIN, OUTPUT);
  digitalWrite(buzzerPIN, LOW);    // turn the BUZZER off by making the voltage LOW
  //-------Setup-KEY----------------------------------------
  pinMode(KEY1, INPUT_PULLUP);
  pinMode(KEY2, INPUT_PULLUP);
  //=======SETUP BLE=============================================
  myBLE.begin(9600);
  //myBLE.begin(115200);
  BLE_setName("UnoBLExx"); //自訂BLE裝置訂編號
  Serial.begin(9600);
}
```

圖 3.4Arduino 程式撰寫

Five-speech-floor

```
// the loop routine runs over and over again forever:
void loop() {
   String uartRead="";

  if (myBLE.available()) {
    // read the most recent byte (which will be from 0 to 255):
    uartRead = myBLE.readString() ;
    ////////////////////////
    if( uartRead[0] == 'a'){
              digitalWrite(Lamp1,HIGH);
              delay(500);
              digitalWrite(Lamp1,LOW);
           }
      if( uartRead[0] == 'b')    {
              digitalWrite(Lamp2,HIGH);
              delay(500);
              digitalWrite(Lamp2,LOW);
           }
        if( uartRead[0] == 'c')    {
              digitalWrite(Lamp3,HIGH);
              delay(500);
              digitalWrite(Lamp3,LOW);
           }
          if( uartRead[0] == 'd')    {
              digitalWrite(Lamp4,HIGH);
              delay(500);
              digitalWrite(Lamp4,LOW);
           }
        if( uartRead[0] == 'e')    {
              digitalWrite(Lamp5,HIGH);
              delay(500);
              digitalWrite(Lamp5,LOW);
           }
      // set the brightness of the LED:
    myOLED.setPosi(4,0);
    myOLED.print("CMD:");
    myOLED.println(uartRead);
     Serial.println(uartRead);
    switch(uartRead[0])
    {
```

圖 3.5Arduino 程式撰寫

Five-speech-floor

```
    Serial.println(uartRead);
    switch(uartRead[0])
    {
      case 'D':
         if(uartRead[1]=='L')
            Flash_LED(uartRead[2]-0x30, 1000);
         else if(uartRead[1]=='S')
            Flash_LED(uartRead[2]-0x30, 200);
         else if(uartRead[1]=='O')
           LEDState=1;
         else if(uartRead[1]=='F')
           LEDState=0;
         else
          {
            int duty=(uartRead[1]-0x30)*100+(uartRead[2]-0x30)*10+(uartRead[3]-0x30);
            analogWrite(LED, duty);
          }
          break;
       case 'B':
           if(uartRead[1]=='L')
                Beep(uartRead[2]-0x30, 1000);
           if(uartRead[1]=='S')
                Beep(uartRead[2]-0x30, 200);
           if(uartRead[1]=='O')
                BZState=1;
            if(uartRead[1]=='F')
                BZState=0;
            break;
       case 'W':
          if(uartRead[3]=='L')
               Flash_RGB(uartRead[1]-0x30,uartRead[2],uartRead[4]-0x30 ,1000);
           else if(uartRead[3]=='S')
               Flash_RGB(uartRead[1]-0x30,uartRead[2],uartRead[4]-0x30,200);
           else if(uartRead[3]=='O')
          {
               RGBState=uartRead[1]-0x30;fsgRGB=uartRead[2];
          }
          else if(uartRead[3]=='F')
               RGBState=0;
```

圖 3.6Arduino 程式撰寫

Five-speech-floor

```
        else if(uartRead[1]=='A')
          {
              SET_RGB(1,uartRead[2]);
              SET_RGB(2,uartRead[2]);
              SET_RGB(3,uartRead[2]);
          }
        else
            SET_RGB(uartRead[1]-0x30, uartRead[2]);
        delay(500);
        break;
      case 'T':
          TrafficLight(uartRead[1]-0x30, uartRead[2]-0x30,uartRead[3]-0x30); break;
    }
    delay(1000);        // delay in between reads for stability
}
//=============================================
if(BZState==1)
       Beep(1, 200);
//=============================================
if(LEDState==1)
      Flash_LED(1, 200);
//=============================================
if(RGBState!=0)
       Flash_RGB(RGBState,fsgRGB,1,200);
  // Read Key Send String
 int  BnState = digitalRead( KEY1);
 if(BnState==0)
 {
     delay(500);        // delay in between reads for stability
     if(DemoFunc>8)
          DemoFunc=1;
      else
        DemoFunc=DemoFunc+1;
        myOLED.setPosi(6,0);
        myOLED.print("Send: S");
        myOLED.println(DemoFunc);
 }
 BnState = digitalRead( KEY2);
 if(BnState==0)
```

圖 3.7Arduino 程式撰寫

Five-speech-floor

```
  {
    delay(500);
    myBLE.print("S");
    myBLE.println(DemoFunc);
   }
   //=========================================
  if (Serial.available())
  {
     uartRead = Serial.readString() ;
    // set the brightness of the LED:
    myOLED.setPosi(6,0);
    myOLED.println(uartRead);
    myBLE.println(uartRead);
  }
}
void SET_RGB(int m, char c)
{
    m=m-1;
    if(c=='R')      leds[m] = CRGB::Red;
    if(c=='G')      leds[m] = CRGB::Green;
    if(c=='B')      leds[m] = CRGB::Blue;
    if(c=='Y')      leds[m] = CRGB::Yellow;
    if(c=='W')      leds[m] = CRGB::White;
    if(c=='D')      leds[m] = CRGB::Black;
    FastLED.show();
}

void Flash_RGB(int m, char c, int n, int finv)
{
   int i;
   SET_RGB(m, 'D');
   for(i=0;i<n;i++){
       SET_RGB(m, c);
       delay(finv);
       SET_RGB(m, 'D');
      delay(finv);
   }}

void TrafficLight(int n, int m,int t)
```

圖 3.8Arduino 程式撰寫

Five-speech-floor

```
void TrafficLight(int n, int m,int t)
{
  // 1R=n sec; 2Y flash m; 3G=t sec;
  //========step1==============
  SET_RGB(1, 'R');
  delay(n*1000);
  SET_RGB(1, 'D');
  //========step2================
  Flash_RGB(2, 'Y', m, 200);
  SET_RGB(2, 'D');
 // ==========step3 =================
  SET_RGB(3, 'G');
  delay(t*1000);
  SET_RGB(3, 'D');
}
   //=========================================
void Flash_LED(int n, int finv)
{
   int i;
   digitalWrite(LED, LOW);
   for(i=0;i<n;i++)
   {
     digitalWrite(LED, HIGH);
     delay(finv);
     digitalWrite(LED, LOW);
     delay(finv);
   }
}
void Beep(int n, int finv)
{
   int i;
   for(i=0;i<n;i++)
   {
     digitalWrite(buzzerPIN, HIGH);   // turn the BUZZER on (HIGH is the voltage level)
     delay(finv);              // wait for a second
     digitalWrite(buzzerPIN, LOW);    // turn the BUZZER off by making the voltage LOW
     delay(finv);              // wait for a second
   }
}
```

圖 3.9Arduino 程式撰寫

Five-speech-floor

```
void LED_Fade(int timing)
{
  int brightness;
   for(brightness=0;brightness<=255;brightness++)
    {
       analogWrite(LED, brightness);
       delay(timing);
    }

    for(brightness=255;brightness>=0;brightness--)
    {
       analogWrite(LED, brightness);
       delay(timing);
    }
}
//-----------------------------------------------------------------
void BLE_setName(char blename[])
{
  int i,len;

  // BLE module get mac
  myBLE.write(0xAA);
  myBLE.write(0xED);
  myBLE.write(0xBB);
  delay(100);

  // BLE module set name
  myBLE.write(0xAA);
  myBLE.write(0xE3);
  myBLE.write(0x5A);
  len=strlen(blename);
  for(i=0;i<12;i++)
  {
    if(i<len)
    {
      myBLE.write((uint8_t)blename[i]);
    }
    else
    .
```

圖 3.10Arduino 程式撰寫

```
  .
  else
  {
    myBLE.write(0x20);
  }
}
myBLE.write(0xBB);
delay(200);

// BLE module reboot
myBLE.write(0xAA);
myBLE.write(0xEF);
myBLE.write(0x5A);
myBLE.write(0xBB);
delay(200);
while(myBLE.available())
{
  myBLE.read();
}
```

圖 3.11Arduino 程式撰寫

第四章 APP Inventor2 介紹及應用

4.1App Inventor 2 介紹

App Inventor 是由 Google 實驗室（Google Lab）所設計，可用於開發 Android 系統的應用程式。App Inventor 是以拼圖式方塊來選寫程式，而用瀏覽器做為管理工具，所有程式及資源都放在雲端，設計者可在任何時間及地點進行設計工作。App Inventor 是一個完全線上開發的 Android 程式環境，拋棄複雜的程式碼而使用樂高積木式的堆疊法來完成您的 Android 程式。

優點: 1.拼圖式程式選寫
2.瀏覽器管理系統
3.強大而實用的元件庫
4.支援 NXT 樂高機器人

缺點: 1.不支援 IOS 系統
2.容易造成病毒軟體被開發

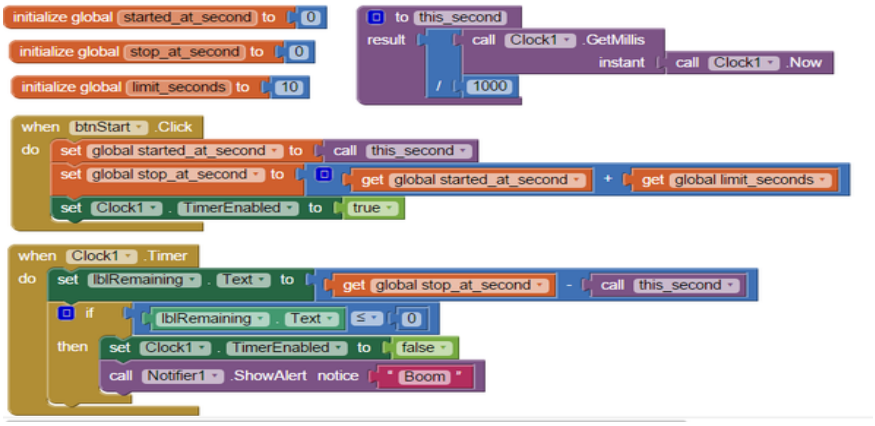


圖 4.1App 拼圖編排

4.2 App Inventor 2 介面製作

介面排列元件是一個容器，本身不會在螢幕中顯示，當其他員件加到介面排列元件裡後，會依指定方式排列，經過精心編排，就能設機出賞心悅目的介面。

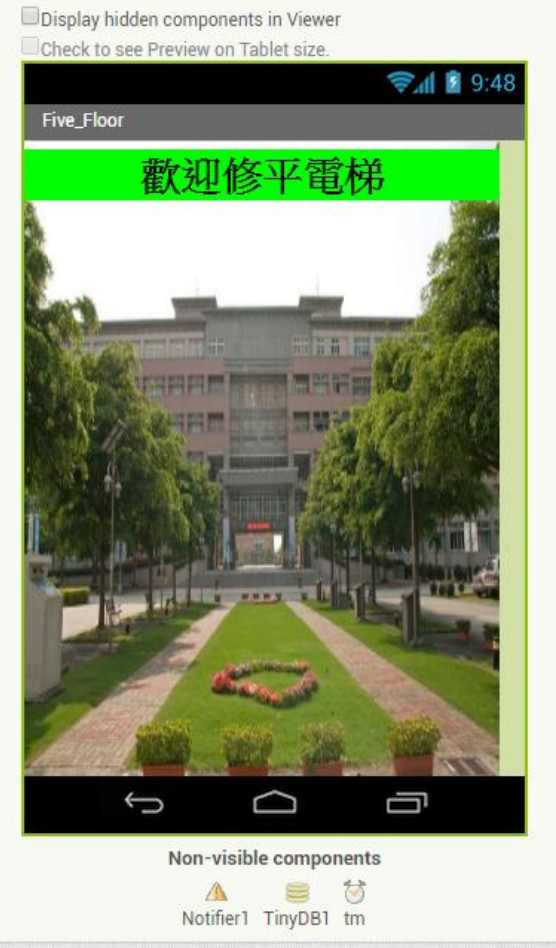


圖 4.2.app 進入介面

圖 4.3appAPP 推播介面

圖 4.4APP 操控介面

4.3 App Inventor 2 程式拼圖

執行程式有時須依情況不同而執行不同程式馬，其依據的原則就是[判斷式]。程式中用來處理重複工作的功能稱為[迴圈]，迴圈分為固定執行次數迴圈及不固定執行次數迴圈。

一般程式語言使用[陣列]來解決儲存大量同類型資料的問題，App Inventor 2 則以 Lists(清單)代替陣列。清單可說是一群性質相同變數的集合，屬於循序性資料結構，清單中的資料是一個接著一個的存放。

圖 4.5 APP 進入介面程式拼圖

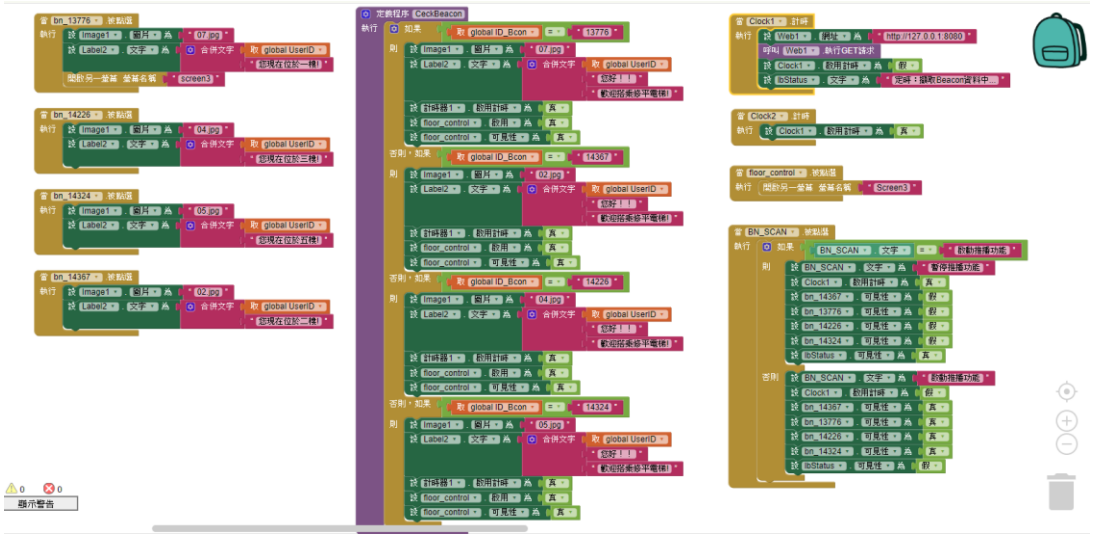


圖 4.6APP 推播界面程式拼圖

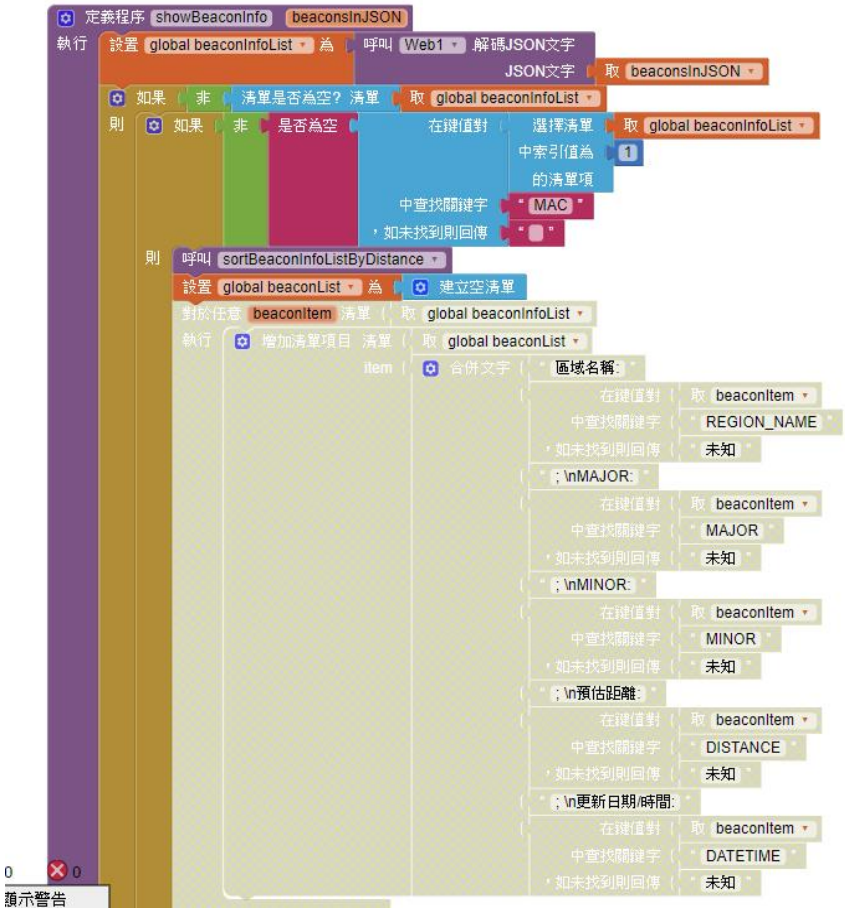


圖 4.7APP 推播界面程式拼圖

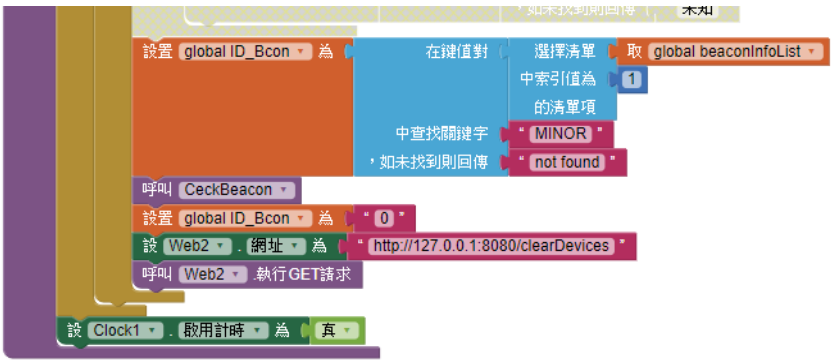


圖 4.8APP 推播界面程式拼圖

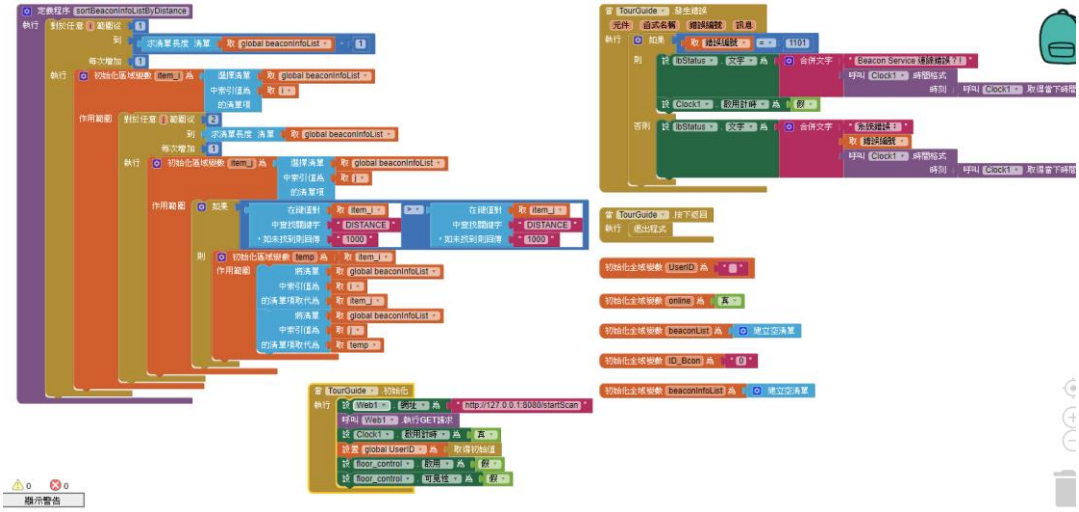


圖 4.9APP 推播界面程式拼圖

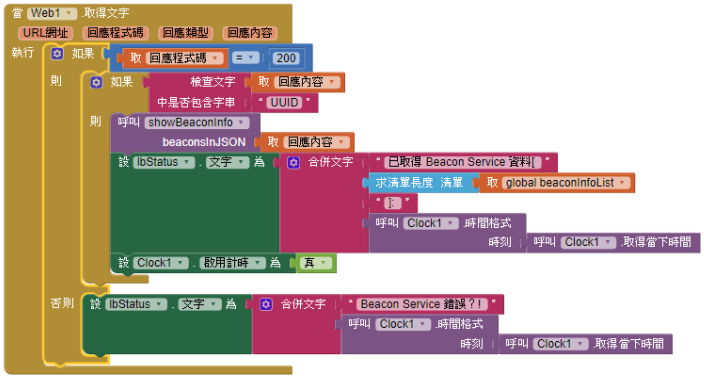


圖 4.10APP 推播界面程式拼圖

第五章 Beacon 介紹及應用

5.1 Beacon 介紹

物聯網和大數據無疑是 2014 年科技業兩大關鍵字，而這股熱潮仍會延續至 2015 年，揭示萬物連網與數據經濟時代的來臨。

圖 5.1 Beacon 運用

然而任何採用低功耗藍牙（BLE 或藍牙 4.0）的微定位訊號發射器皆能稱之為 Beacon，Beacon 與 iBeacon 的差別僅在於廣播訊號頻率的差別。簡而言之，Beacon 就像是一個不停地在廣播訊號的燈塔，當手機進入到燈塔照射的範圍內，Beacon 就會發送一串代碼給手機，手機的 App 偵測到代碼後便會觸發一連串的動作，也許是從雲端下載資訊，也可能是開啟其他 App 或連動裝置。Beacon 被業界視為明日之星的原因，在於它比 GPS 有更精準的微定位功能。以往 GPS 只能大概得知使用者所在，Beacon 則可將定位範圍

精準到 2～100 公尺內，明確辨識任何進入信號傳輸範圍內的手機。與其他無線通訊技術相比，Wi-Fi 雖然也能做到定位，但精準度低於 Beacon，且僅有 Android 系統支援 Wi-Fi 定位技術。而近兩年大為熱門的 NFC（無線近場通訊）由於必須近距離接觸才能傳輸訊息，致其先天就無法做到定位，目前 NFC 技術在行動支付領域的呼聲雖高，但後起之秀 Beacon 也能做到支付應用，因此 Beacon 又被稱為「NFC 殺手」。Beacon 同時兼具定位與支付的優勢，讓 Wi-Fi 與 NFC 分別在定位與支付的光環盡失，儼然成為實體通路虛實整合（O2O）的救世主。

圖 5.2 Beacon

5.1.2 Beacon UUID

UUID:

這是一個 32 碼的字串, 會被記錄在對應的 app 裡, 也就是所謂 app "認得的" iBeacon 識別資訊. 像是 "E2C56DB5-DFFB-4802-B060-D0F5A71096E0" 就是蘋果本家的 iBeacon UUID.

Major 和 Minor:

Major 和 Minor 是用來做虛擬分類的。想像一個場景，例如百貨公司，全部樓層，甚至各專櫃，全部都裝設有 iBeacon 發訊器。這些 iBeacon 發訊器都會發射一樣的 UUID, 訊號滿天飛. 那 app 要怎麼區別誰是誰呢? 這時 Major 與 Minor 兩個值就派上用場了。可以先約定好， Major 代表樓層，Minor 代表專櫃。(Major=5, Minor=3) 就代表五樓第 3 號專櫃發送出來的 iBeacon 訊號。

5.2 BLE Gateway

當 BLE Gateway 開啟後，按下 Beacon 就會在 BLE Gateway 上搜尋到 Beacon UUID，就可以開始進行推播。

5.2.1 BLE Gateway App

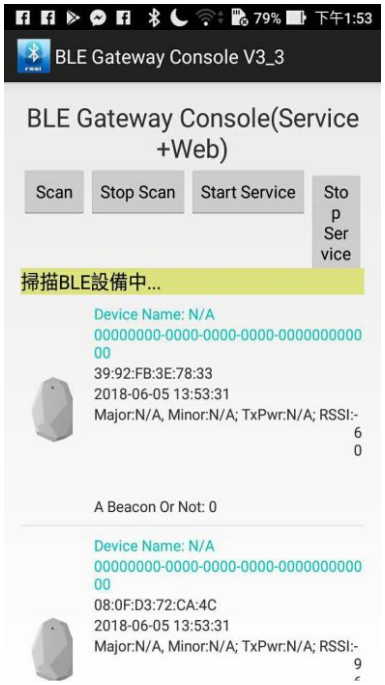


圖 5.3 BLE Gateway 沒有搜尋到 UUID

圖 5.4 BLE Gateway 搜尋到 UUID

第六章繼電器電路控制板

6.1 電晶體開關控制電路

雙極性接面電晶體(Bipolar Junction Transistor, BJT)由半導體材料組成,至少有三個對外端點(稱為極),集極(Collector)、射極(Emitter)、基極(Base),其中基極是控制極，另外兩個端點之間的伏安特性關係是受到控制極的非線性電阻關係。電晶體基於輸入的電流或電壓，改變輸出端的阻抗 ，從而控制通過輸出端的電流，因此電晶體可以作為電流開關，而因為電晶體輸出信號的功率可以大於輸入信號的功率，因此電晶體可以作為電子放大器。

6.2 2N2222 NPN 型電晶體

2N2222 是一種常見的 NPN 雙極街面電晶體管用於一般用途的低功率放大或開關應用。它設計用於低至中等電流，低功率，中等電壓，並且可以在適中的高速下運行。標準測試條件下測量包括外形尺寸，小信號電流增益，轉換頻率，耐壓最大值，電流額定值，功耗和溫度額定值等。其他零件號碼將有不同的參數。

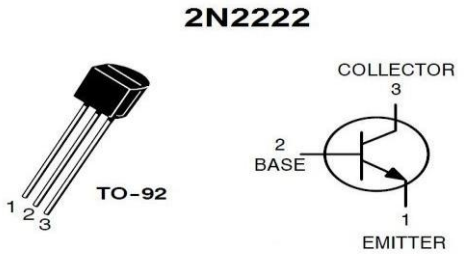


圖 6.1 電晶體 2N222

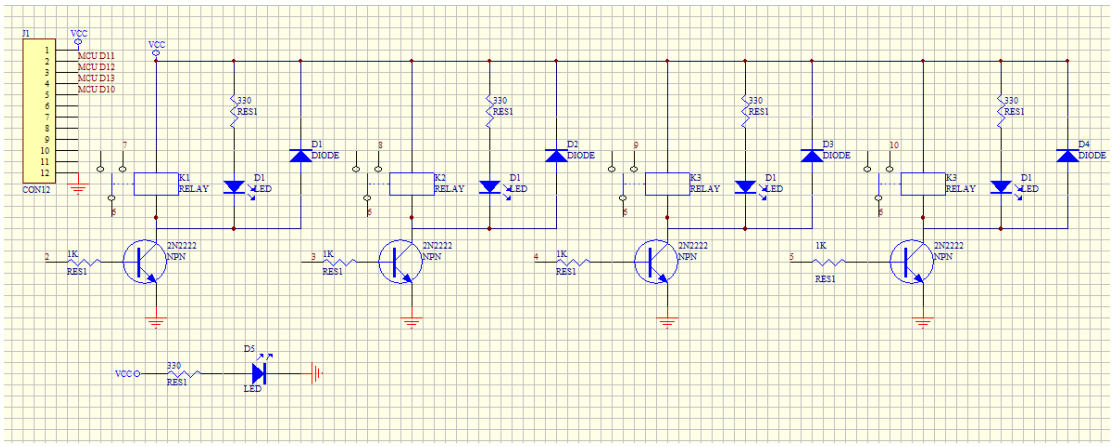
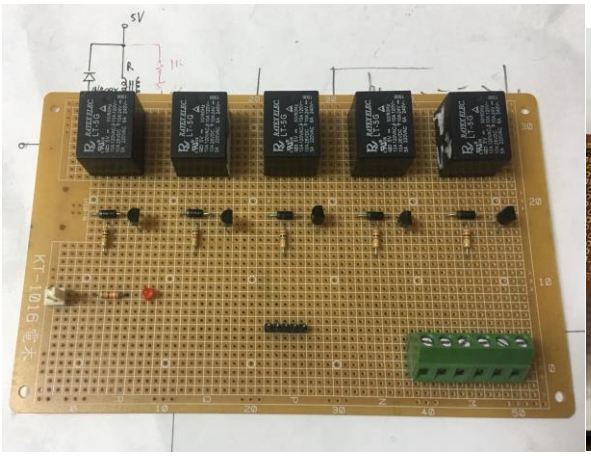


圖 6.2 電晶體開關控制電路

當電晶體處於開關工作方式時，因為電源電壓和集電極電阻的限制，集電極 I_C 不足以提供 $h_{FE}I_B$ 不能持續增加，電晶體進入飽和狀態，此時集電極和發射極之間的電壓 V_{CE} 間的電壓約為 0.1~0.2V 左右的低電壓。電晶體處於導通狀態。正是因為這個緣故，電晶體才可以作為開關來使用。電晶體作為電子開關使用時，能夠對被控對象進行控制，諸如 LED、電動機、繼電器線圈等。應為此時電晶體的集電極除被控對象外，沒有連接任何其他的電路或元件。

若在繼電器增加一個反向二極體，當電晶體為截止時，二極體可以吸收多餘的感應電勢所產生的電流，以保護繼電器線圈部燒毀。因此可以利用 $h_{fe}=\frac{IB}{IC}$，而 $I_C=I_E+I_B$，得到電晶體的工作狀況。

6.3 製作過程



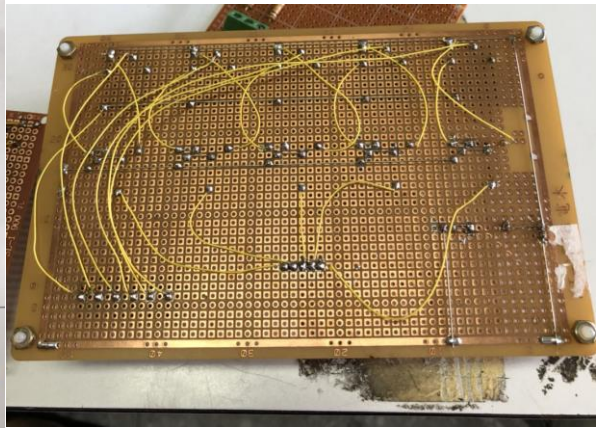


圖 6.3　電路設計圖 6.4 電路接線

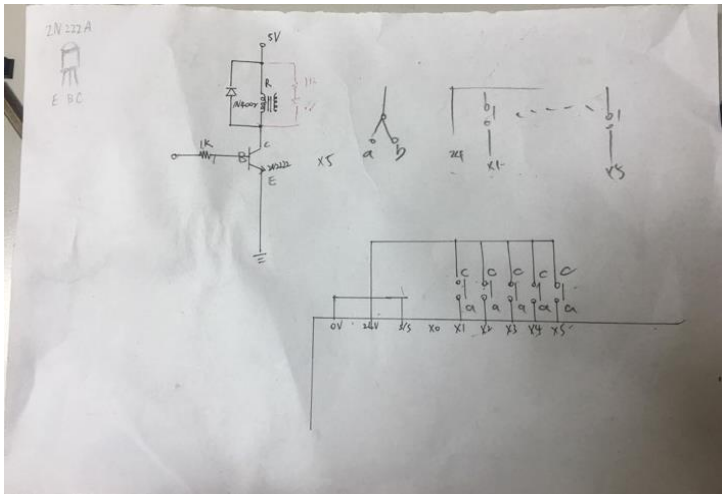


圖 6.5 電路設計圖

參考文獻

[1]http://www.how01.com/post_kXzempD3lv542.html PLC 介紹:愛經驗

[2]http://www.phd.com.tw/knowledge/app-dev/ APP 介紹:博泉科技有限公司

[3]https://zh.wikipedia.org/zh-tw/ArduinoArduino 介紹:維基百科

[4]https://tutorials.webduino.io/zh-tw/docs/basic/component/relay.html

繼電器介紹:Webduion 學習手冊

[5]http://www.ancheng.tw/wp/%E4%BB%80%E9%BA%BC%E6%98%AFbeacon%EF%BC%9F/ Beacon 介紹:安成數位科技

[6]http://gogoprivateryan.blogspot.com/2014/10/cc2540-bluetooth-low-energy-5-ibeacon.html Beacon UUID:鋼鐵人實作聯盟阿立

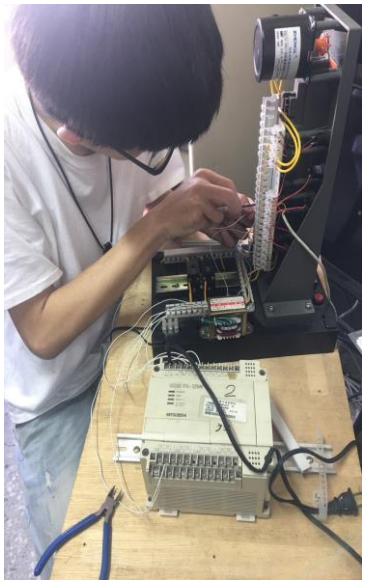
[7]http://www.appinventor.tw/ai2 App Invetor 2:

結論

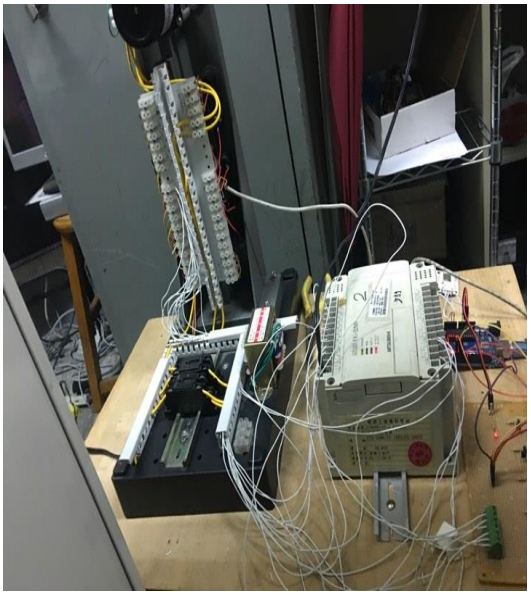
在這次專題上我們從規劃、設計及製作。我們在製作 Beacon 電梯的過程中有很多東西是我們沒碰過和做過的，我們先自己摸索但碰到不會的地方，我們詢問了老師和同學請他們交導我們。我們重配線、電路製作、App 設計….等等。

雖然在做這專題碰到很多的事情，但我們都慢慢去找出他問題的所在定研究，這次專題能夠順利做出來，但是還是有些定方要加強和補救，例如；藍芽掃描及 app，如果太多人聯會導致她擋掉，這是我們所要加強的地方。經過科技的進步和生活中的進步，可以達到意想不到的結果，這次我們利用電梯結合 App、藍芽和 Beacon 呈現，也利用手機端來控制達到一開始的目的，讓生活更方便。

專題製作過程







作者介紹

個人簡介

姓名:朱鴻旻

性別:男

生日:1997 年 3 月 12 日

E-mail:pig86312@gmail.com

學歷

修平科技大學-電機工程系(108 級)

台中市私立嘉陽高級工學-電子科(104 級)

專長

AppInventor 程式開發，電腦檢修，電路設計，焊接電路

證照與競賽證明

勞動部 工業電子 丙級

勞動部 電腦硬體裝修 乙、丙級

個人簡介

姓名:黃偉翔

性別:男

生日:1996 年 11 月 26 日

E-mail:jk901336386 @gmail.com

學歷

修平科技大學-電機工程系(108 級)

雲林縣公立虎尾農工職業高級學校(104 級)

專長

工業配線，室內配線，變壓器裝修

證照與競賽證明

勞動部 工業配線 丙級

勞動部 室內配線 丙級

勞動部 變壓器裝修 乙級