前言:

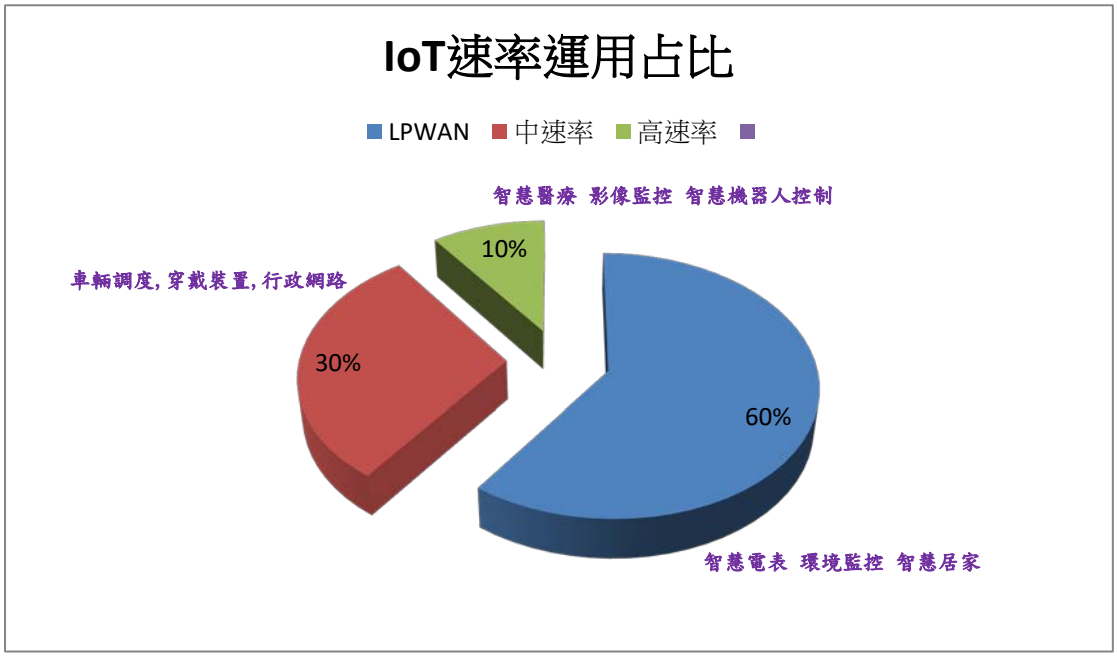
根據 IoT 應用場景不同，物聯網的需求可以分為高速率、中速率、低功率耗覆蓋三層，而 NB-IoT 在的 LPWAN 層級中又具備覆蓋 90%的佔有能力。

LPWAN（Low Power WideArea Network）：低功耗廣域網絡，就是專為低功耗、遠距離、數量多連接的物聯網應用而設計。

短距離無線網絡包含 WIFI、藍牙、ZigBee 等多種技術。

LPWAN 也包含多種技術，如 LoRa、Sigfox、NB-IoT 等。

目前 LPWAN 技術可被分為授權頻段(NB-IoT)的廣域網技術及非授權頻段(Sigox LoRa)的廣域網技術兩類，不同的 LPWAN 技術在接入網絡、部署方式、 技術特點、功耗性能及服務模式上都有所差異。



Sigfox 和 LoRa 屬於非授權頻段，應用時需要單獨建構網絡，而且使用的頻段沒有授權，在安全性上也可能存在缺陷。

NB-IoT 是 3GPP 推出的標準技術，已成為了目前被全球廣泛接受的全新窄帶物聯網技術標準。

從接入網絡上看，由於 NB-IoT 是在 LTE 基礎上發展起來的， 其主要採用了 LTE 的相關技術，並針對自身特點做了相應的修改。從技術特點上看，NB-IoT 的部署方式較為快捷、靈活，支持 3 種部署場景。此外， NB-IoT 也可以部署在 2G/3G 網絡。

NB-IoT 與 WiFi 之差異 :

	NB-IoT	WiFi
上網方式	NB-IoT 利用電信基地台連上網際網路	透過無線基地台連上網際網路
傳遞資料量	適用小資料量傳輸	適用傳輸大量資料的訊息
連線距離	由於全台基地台涵蓋率夠高，幾乎無死角	連接無線基地台的距離較短
耗損功率	採用低功率晶片， 使用一般 AA 電池可達 3-5 年以上	晶片耗用功率較高
IP 位址	使用的電信基地台提供的 IP 位址大都為 虛擬 IP 網段	連網較易取得真實 IP 位址

DSI2598+ 開發板介紹

DSI2598P 使用聯發科技 NB-IoT 晶片-MT2625 模組與 STM32F103C8T6 晶片，有著 PWM、I2C、SPI、ADC、UART 等多種腳位功能，簡單但 完整，可讓使用者無縫接軌任何 Arduino 程式庫，進行各項功能程式開 發，是改善 DSI2598 速度及記憶體空間不足的第三代 NB-IoT 開發板

硬體功能	第一代	第二代	第三代
NB-IoT 晶片	MT2625	MT2625	MT2625
MCU	ATMEGA328	ATMEGA328	STM32F103C8T6
I/O	16	16	37
EEPROM	1KB	1KB	可用 flash 模擬
SPRAM	2KB	2KB	20KB
Flash Memory	32KB	32KB	64KB
ADC	3	2	10
SPI	1	1	2
UART	1	1	3
I2C	1	1	2
PWM	5	2	15

開發環境設定

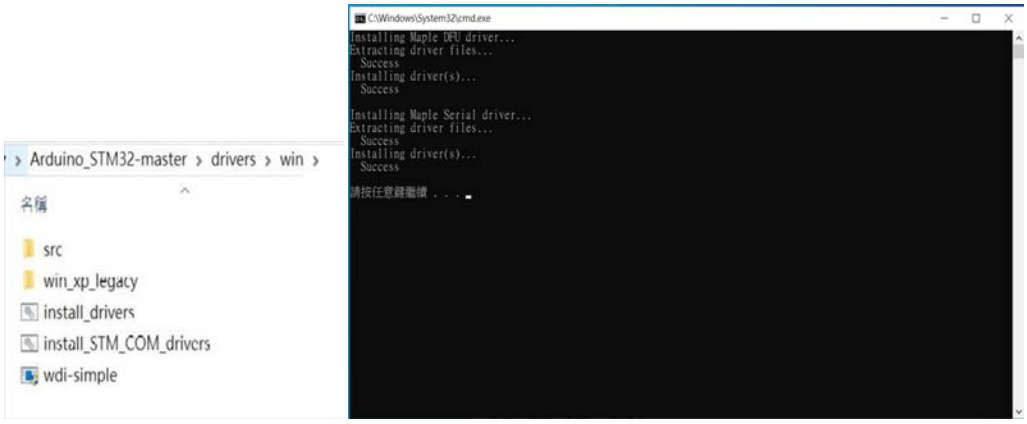
設定 Arduino DSI2598+開發板的環境: (for Windows 10 作業系統)

1. **安裝 DFU windows 的 driver :**

下載目 https://github.com/rogerclarkmelbourne/Arduino_STM32

至 Arduino_STM32-master.zip ， 解開檔案之後在目錄下用系統管理者執行

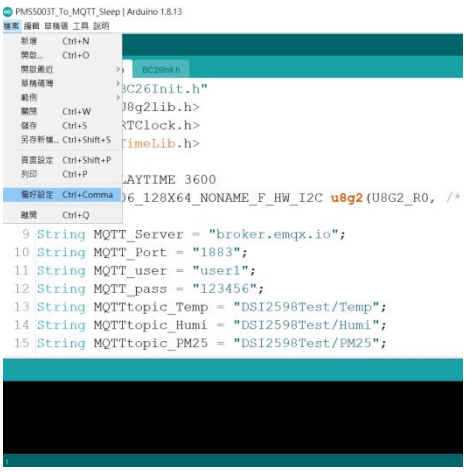
Arduino_STM32-master\drivers\win\install_drivers.bat，會出現下列畫面

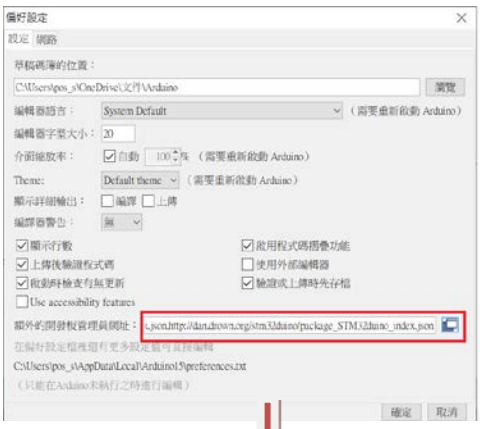


2. 安裝 **Arduino IDE for 1.8.13:**

下載區: https://www.arduino.cc/en/software

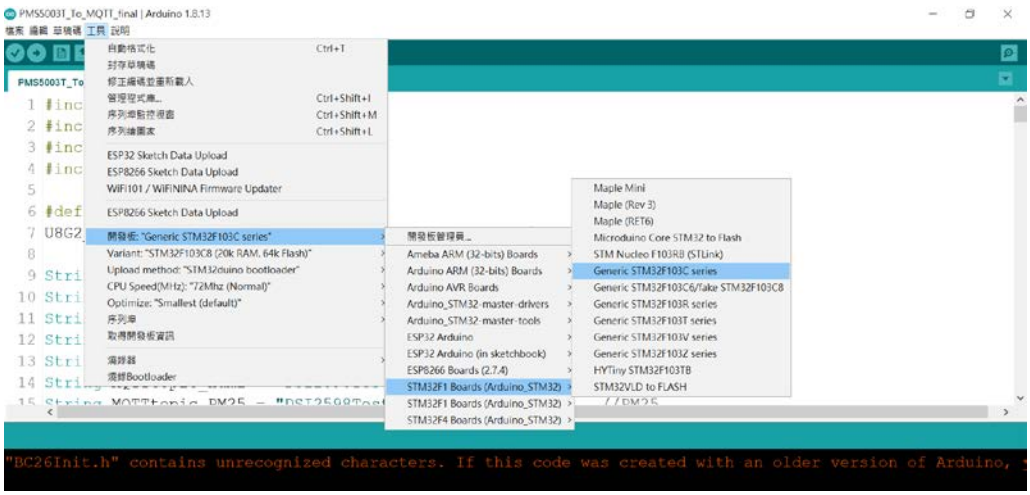
3. 設定 STM32 所需的管理員網址





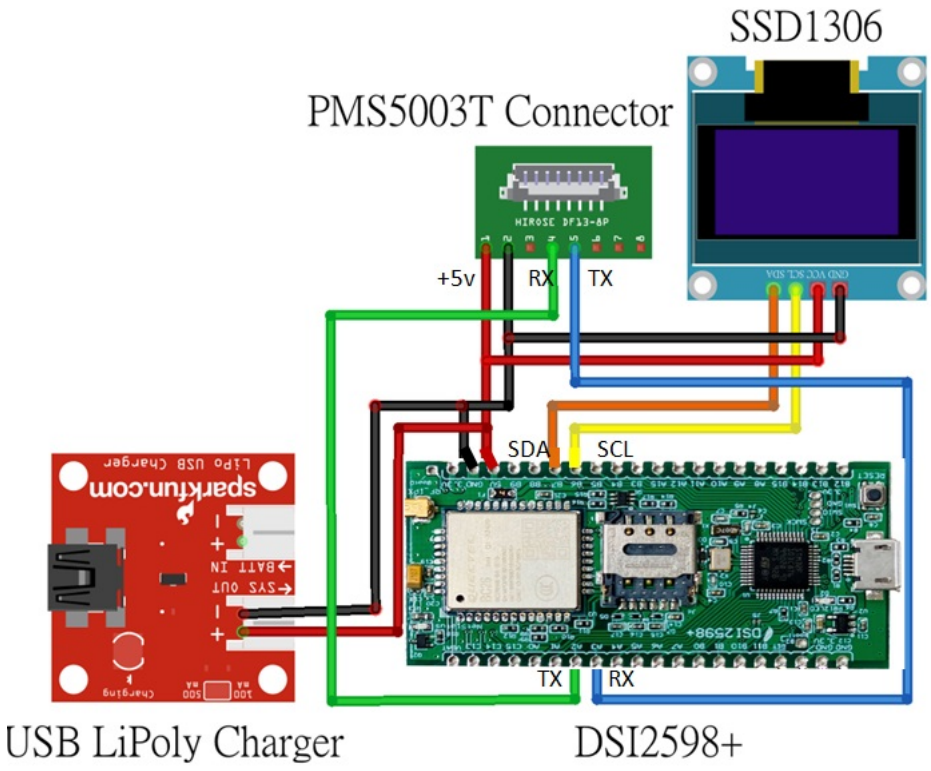
在"Additional Boards Manager URLs:" 輸入

http://dan.drown.org/stm32duino/package_STM32duino_index.json

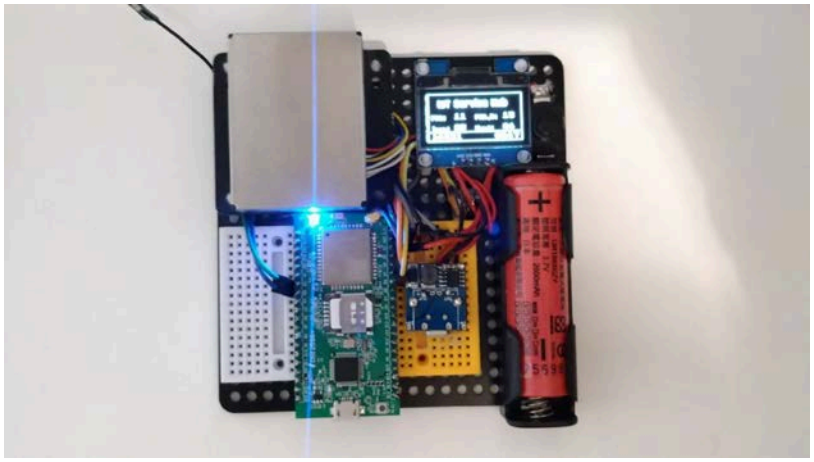


DSI2598+ 實作運用 – 落塵檢測器 (Particle Measuring System)

接線圖:



實作圖: (第一版)



材料:

1. DSI2598+
2. UPS 鋰電池充電電路模組 – USB 接頭
3. SSD1306 1.3 吋
4. PMS 5003T 落塵感測器
5. 小麵包板 *2
6. 杜邦線 公對公 與 公對母 些許
7. 18650 鋰電池

功能說明:

現行存在狀況

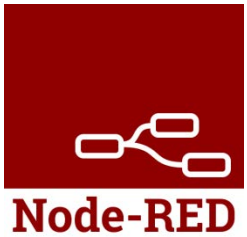
在有些 GMP 與科技工廠需要隨時監測落塵數量與狀況,之前都會透過 手持或是固定式檢測器收集落塵相關資料,然後在透過傳輸線傳回電腦做收集統計.但由於收集範圍過大,手持式需要由人員在不同地方做收集並且傳回後台系統,而固定式的檢測器則需要有插電處提供電源做收集.但固定式機台成本過高,不適合到處擺設此裝置收集落塵資訊.

解決現況問題

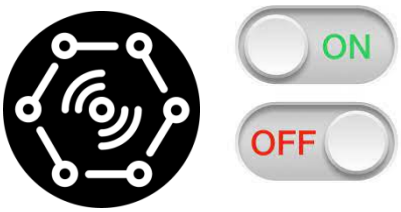
(第一版)

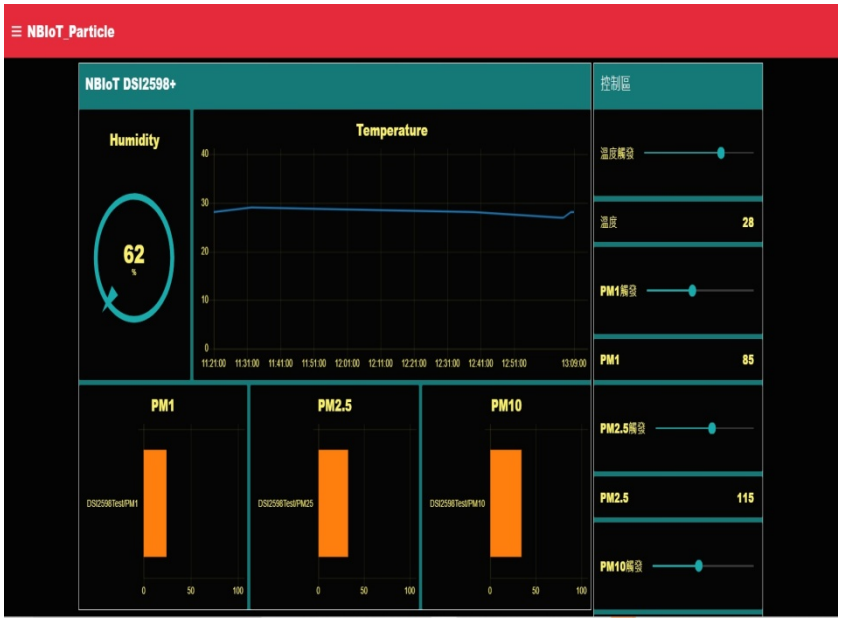
透過 NB-IoT 低耗電特性,此作品使用一節 18650 鋰電池, 操作時間上幾乎能達到 3 天以上.且透過 MQTT 傳輸機制, 可以立即傳輸 收集到的落塵資料到後台,且在後台透過 Node-RED 便能輕鬆建置 dashboard 與儲存資料,且管理者也可以透過手機或平板端來檢視收集到的落塵資訊,非常便利.

後台 dashboard

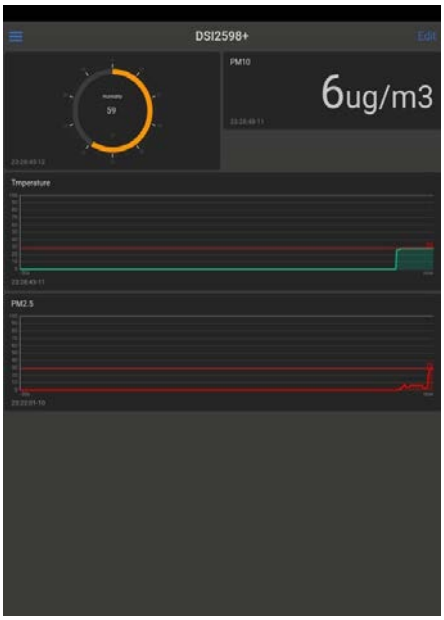


手機或平板的 App (MQTT Dash / IoT OnOff)





Node-RED dashboard

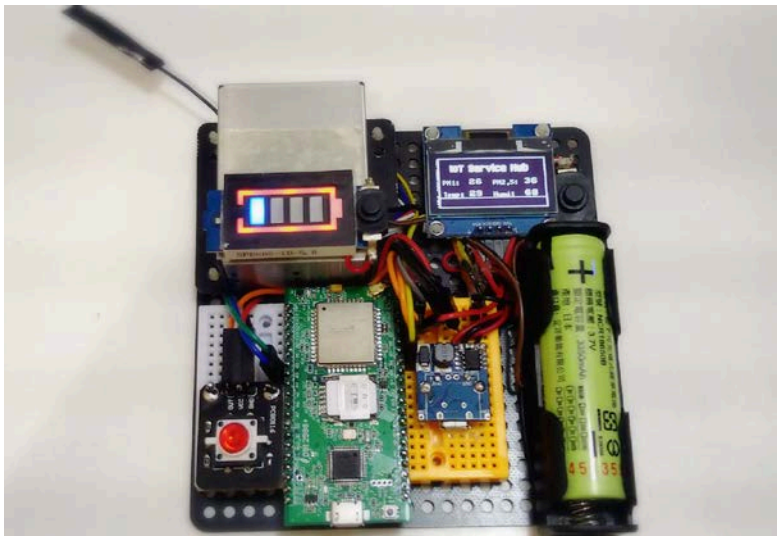


IoT OnOff (App)

(修正版本)

希望藉由 NB-IoT 省電特性,希望能將 落塵收集器的工作使用時間能延長,才能符合經濟效益.系統調整方式如下:

1. SSD1306 藉由 按鈕控制是否顯示資訊, 在需要時再開啟即可.
2. 增設 LED 的鋰電電量顯示器,一樣可藉由 按鈕來呈現目前電量狀態
3. PMS 5003 讀出落塵資料後,便把 收集落塵風扇關閉
4. BC26 (NB-IoT) 透過 MQTT 傳回資料後,便進入休眠模式,已節省電量
5. 再讓 STM32 進入休眠模式,待下次(60 分鐘後)傳回落塵資料再重新啟動.



程式說明:

PMS5003T_to_MQTT.ino (主程式)

```
#include "BC26Init.h"
#include <U8g2lib.h>
#include <RTClock.h>
#include <TimeLib.h>

#define DELAYTIME 3600
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

String MQTT_Server = "broker.emqx.io";                  //MQTT Server
String MQTT_Port = "1883";                              //MQTT Port
String MQTT_user = "user1";                             //user
String MQTT_pass = "123456";                            //password
String MQTTtopic_Temp = "DSI2598Test/Temp";             //Temp
String MQTTtopic_Humi = "DSI2598Test/Humi";             //Humi
String MQTTtopic_PM25 = "DSI2598Test/PM25";             //PM25
String MQTTtopic_PM10 = "DSI2598Test/PM10";             //PM10
String MQTTtopic_PM1 = "DSI2598Test/PM1";               //PM1
String MQTTmessage = "";

int errorFlag = 0 ;
long pmat10 = 0;
long pmat25 = 0;
long pmat100 = 0;
unsigned int temperature = 0;
unsigned int humandity = 0;
unsigned long startTime;

RTClock rt (RTCSEL_LSI); // initialise

int buttonPin = PB9;                                    //觸發喚醒 OLED 的 Pin
int ledToggle;
int previousState = HIGH;
unsigned int previousPress = 0;
unsigned int buttonFlag = 1;
int buttonDebounce = 20;

void setup() {
```

```
Serial.begin(115200);
Serial1.begin(115200);                                  //與 BC26 溝通的 UART
Serial2.begin(9600);                                    //與 PMS5003T 溝通的 UART
pinMode(buttonPin, INPUT);                              //宣告 觸發 OLED 顯示的 Pin 為 中斷服務
attachInterrupt(digitalPinToInterrupt(buttonPin), button_ISR, FALLING);

PMSpassiveMode();                                       //宣告 PMS5003T 為 passiveMode

u8g2.begin();                                           //啟動 u8g2 library
u8g2.enableUTF8Print();
u8g2.setDisplayRotation(U8G2_R2);                       //OLED 旋轉 180 度

u8g2.setFont(u8g2_font_Born2bSportyV2_tr);    //使用我們做好的字型
u8g2.clearBuffer();
u8g2.setCursor(5, 30);
u8g2.print("Connecting...");
u8g2.drawFrame(0, 0, 126, 64);
u8g2.sendBuffer();
if (!BC26init())                                        //初始化 BC26，若連線失敗等待 5 秒鐘再重新啟動
{
   u8g2.clearBuffer();
   u8g2.setCursor(5, 30);
   u8g2.print("Failed");
   u8g2.drawFrame(0, 0, 126, 64);
   u8g2.sendBuffer();
   delay (5000);
   nvic_sys_reset();
}
u8g2.setFont(u8g2_font_Born2bSportyV2_tr);
u8g2.clearBuffer();
u8g2.setCursor(5, 30);
u8g2.print("Successfully");
u8g2.sendBuffer();
delay(1000);
u8g2.setCursor(5, 30);
u8g2.print("initialization");
u8g2.clearBuffer();

u8g2.setCursor(13, 18);
u8g2.setFont(u8g2_font_Born2bSportyV2_tr);
u8g2.print("IoT Service Hub");
```

```
    u8g2.drawFrame(0, 0, 126, 64);
    u8g2.setFont( u8g2_font_5x8_tf );
    u8g2.setCursor(5, 35);
    u8g2.print("PM1: ");
    u8g2.setCursor(64, 35);
    u8g2.print("PM2.5: ");
    u8g2.setCursor(5, 50);
    u8g2.print("Temp: ");
    u8g2.setCursor(64, 50);
    u8g2.print("Humi: ");
    u8g2.drawLine(0, 51, 128, 51);
    u8g2.sendBuffer();
    Serial.println("initialization OK ....");
    Serial.println("Start Loop Program ...");
    delay(5000);
    startTime = millis();

}

void loop() {


    PMSwake();                          //開啟  PMS5003T  風扇
    delay(15000);
    retrievePmValue();                  //抓取  落塵數值與溫溼度
    while ( errorFlag )                 //若抓回數值有誤,延遲 2 秒重新抓取
    {
       retrievePmValue();
       delay(2000);
    }
    Serial.print("Fan turn off");
    PMSsleep();                         //關閉  PMS5003T  的風扇,節省電能

    String Buff      ;
    displayValue();

    Buff += temperature ;                 //數值轉為  String
    json_doc["Temp"] = Buff;

    Buff = "";
    Buff += humandity ;
    json_doc["Humi"] = Buff;
```

```
  Buff = "";
  Buff += pmat10 ;
  json_doc["PM10"] = Buff;

  Buff = "";
  Buff += pmat25 ;
  json_doc["PM25"] = Buff;

  Buff = "";
  Buff += pmat100 ;
  json_doc["PM100"] = Buff;

  connect_MQTT(MQTT_Server, MQTT_Port, MQTT_user, MQTT_pass);   //開啟 MQTT 連線
  serializeJson(json_doc, json_output);
  Publish_MQTT(MQTTtopic_ParticleStr, json_output); //MQTT 傳出量測的值

  Serial.println( "string to json:" );
  Serial.println( json_output );
  Close_MQTT();                                          //關閉 MQTT 連結

  u8g2.setPowerSave(buttonFlag);                         //關閉 OLED 螢幕
  //Send_ATcommand("AT+QPOWD=0", 1);                     //關閉 BC26 電源
  Send_ATcommand("AT+CFUN=0",1);                         //讓 BC26 進入省電模式
  disableAllPeripheralClocks();
  sleepAndWakeUp(STANDBY, &rt, DELAYTIME);               //讓 STM32 進入 STANDBY 模式

  for (int i = 1 ; i <= DELAYTIME ; i++ )   // 控制 7200 秒傳回一次數據; 因為進入 STOP 狀況, 事件與中斷會再喚醒
  {                                                      // 透過喚醒時間延遲
    char temp[5];
    sprintf(temp, "%04d", i);
    u8g2.setPowerSave(buttonFlag);
    u8g2.setCursor(90, 62);
    u8g2.print(temp);
    u8g2.sendBuffer();
    delay(90);
    u8g2.setCursor(90, 62);
    u8g2.print("      ");
    u8g2.setCursor(5, 62);
    u8g2.print("                ");
    u8g2.sendBuffer();
```

```
    }
    nvic_sys_reset();                                   //STM32 MCU  重新啟動
}
void button_ISR()
{
    delay(20);
    buttonFlag = !buttonFlag;
    Serial.print("Button =");
    Serial.println(buttonFlag);
    delay(200);
}
void displayValue()                                     //在  OLED  顯示相關數值
{
    u8g2.setFont( u8g2_font_amstrad_cpc_extended_8f );
    u8g2.setCursor(35, 35);
    u8g2.print(pmat10);
    u8g2.setCursor(100, 35);
    u8g2.print(pmat25);
    u8g2.setCursor(35, 50);
    u8g2.print(temperature);
    u8g2.setCursor(100, 50);
    u8g2.print(humandity);
    u8g2.sendBuffer();

}
void retrievePmValue() {                                //取出  PMS5003T  相關數值  函式

    int count = 0;
    unsigned char c;
    unsigned char high;
    while (Serial2.available()) {
        c = Serial2.read();
        if ((count == 0 && c != 0x42) || (count == 1 && c != 0x4d)) {
            Serial.println("check failed");
            errorFlag = 1 ;
            break;
        }
        if (count > 27) {
            Serial.println("complete");
            errorFlag = 0 ;
            break;
        }
```

```
      else if (count == 10 || count == 12 || count == 14 || count == 24 || count == 26) {
        high = c;
      }
      else if (count == 11) {
        pmat10 = 256 * high + c;
        Serial.print("PM1.0=");
        Serial.print(pmat10);
        Serial.println(" ug/m3");
      }
      else if (count == 13) {
        pmat25 = 256 * high + c;
        Serial.print("PM2.5=");
        Serial.print(pmat25);
        Serial.println(" ug/m3");
      }
      else if (count == 15) {
        pmat100 = 256 * high + c;
        Serial.print("PM10=");
        Serial.print(pmat100);
        Serial.println(" ug/m3");
      }
      else if (count == 25) {
        temperature = (256 * high + c) / 10;
        Serial.print("Temp=");
        Serial.print(temperature);
        Serial.println(" (C)");
      }
      else if (count == 27) {
        humandity = (256 * high + c) / 10;
        Serial.print("Humidity=");
        Serial.print(humandity);
        Serial.println(" (%)");
      }
      count++;
    }
    while (Serial2.available()) Serial2.read();
    Serial.println();
}

void PMSpassiveMode()                  //PMS5003T passiveMode
{
    uint8_t command[] = { 0x42, 0x4D, 0xE1, 0x00, 0x01, 0x01, 0x70 };
```

```
    Serial2.write(command, sizeof(command));
}
void PMSwake()                          //打開  PMS5003T  風扇
{
    uint8_t command[] = { 0x42, 0x4D, 0xE4, 0x00, 0x01, 0x01, 0x74 };
    Serial2.write(command, sizeof(command));
}
void PMSsleep()                         //關閉  PMS5003T  風扇
{
    uint8_t command[] = { 0x42, 0x4D, 0xE4, 0x00, 0x00, 0x01, 0x73 };
    Serial2.write(command, sizeof(command));
}
```

BC26Init.h (NB-IoT function)

```
#include <ArduinoJson.h>
#include <STM32Sleep.h>

void(* resetFunc) (void) = 0;

byte Rset_Count=0;
int waitingTime = 30000;

String Check_RevData()
{
  String data= "";
  char c;
  while (Serial1.available())
  {
    delay(50);
    c = Serial1.read(); //Conduct a serial read
    data+=c;            //Shorthand for data = data + c
    if (c=='\n') break;
  }
  data.trim();
  return data;
}

byte Send_ATcommand(String msg,byte stepnum)
{
  String Showmsg,C_temp;
```

```
  Serial.println(msg);
  Serial1.println(msg);
  Showmsg=Check_RevData();
  //Serial.println(Showmsg);
  long StartTime=millis();
  switch (stepnum)
   {
      case 0:    // Reset BC26
            C_temp="+IP:";
            break;
      case 1:          // Other Data
            C_temp="OK";
            break;
      case 2:          // Check IPAddress
            C_temp="+CGPADDR:";
            break;
      case 10:         // build MQTT Server
            C_temp="+QMTOPEN: 0,0";
            break;
      case 11:         // Connect to MQTT server by username and password
            C_temp="+QMTCONN: 0,0,0";
            break;
      case 12:         // Publisher MQTT Data
            C_temp="+QMTPUB: 0,0,0";
            break;
      case 13:         // Sub MQTT Data
            C_temp="+QMTSUB: 0,1,0,0";
            break;
   }
   while (!Showmsg.startsWith(C_temp))
   {
     Showmsg=Check_RevData();
     if (Showmsg.startsWith("+")) Serial.println(Showmsg);
     if ((StartTime+waitingTime) < millis()) return stepnum;
   }
   return 99;
}

bool BC26init()    // initialization BC26
{
  Send_ATcommand("AT+QGACT=1,1,\"apn\",\"internet.iot\"",1);
  Send_ATcommand("AT+QCGDEFCONT=\"IP\",\"internet.iot\"",1);
```

```
  Send_ATcommand("AT+QBAND=1,8",1);
  Send_ATcommand("AT+QRST=1",0);
  if (Send_ATcommand("ATE0",1)==99)
    if (Send_ATcommand("AT+CGPADDR=1",2)==99) return true;
  return false;
}

bool connect_MQTT(String Broker,String port,String user,String pass) // connect MQTT Broker
{
  String tempBuff;
  tempBuff = "\"" + Broker + "\"" + "," + port;
  tempBuff="AT+QMTOPEN=0," + tempBuff;

  if (Send_ATcommand(tempBuff,10)!=99)
    return false;

  tempBuff= "\"" + user + "\"" + "," + "\"" + pass + "\"";
  tempBuff="AT+QMTCONN=0,0," + tempBuff;
  if (Send_ATcommand(tempBuff,11)!=99)
       return false;
  return true;
}

bool Publish_MQTT(String topic, String message) {
  String tempBuff;
  tempBuff = "\"" + topic + "\"" + "," + message ;
  tempBuff = "AT+QMTPUB=0,0,0,0," + tempBuff ;

  if (Send_ATcommand(tempBuff,12)!=99)
       return false;
  return true;
}

bool Sub_MQTT(String topic) // Subscribe Topic
{
  String tempBuff;
  tempBuff = "\"" + topic + "\"" + "," + "0";
  tempBuff = "AT+QMTSUB=0,1," + tempBuff;

  if (Send_ATcommand(tempBuff,13)!= 99)
    return false;
```

```
  return true;
}

bool Close_MQTT() // Close MQTT connect
{
  String tempBuff;
  tempBuff="AT+QMTCLOSE=0";
  if (Send_ATcommand(tempBuff,1)!=99)
        return false;
  return true;
}


String JSON_DEC_data (String input,String findData)
{
  int index = input.indexOf(',');
  int x = input.substring(0, index).toInt();
  String json = input.substring(index + 1, input.length());
  //Serial.println(json);
  index = json.indexOf(':');
  x = json.substring(0, index).toInt();
  json = json.substring(index + 1, json.length());
  //Serial.println(json);
  DynamicJsonDocument doc(1024);
  deserializeJson(doc, json);
  JsonObject obj = doc.as<JsonObject>();
  return obj[findData];
}

bool Sub_Ideaschain(String attrestopic)
{
  String S_temp;
  S_temp="\""+ attrestopic + "\"" + "," + "0";
  S_temp="AT+QMTSUB=0,1," + S_temp;    // Qos 0
  Serial.println(S_temp);
  Serial1.println(S_temp);
  delay (2000);
  return true;
}

String Get_Publish_MQTT(byte mode,String attreqtopic , String message) {
  String Showmsg;
```

```
  String S_temp,T_temp;
  //delay (1000);
  if (mode==0) T_temp="sharedKeys";
  if (mode==1) T_temp="clientKeys";
  S_temp="\"" + attreqtopic + "\"" + "," + "\"{\"" + T_temp + "\":\"" + message + "\"}\"";
  S_temp="AT+QMTPUB=0,0,0,0," + S_temp;
  Serial.println(S_temp);
  Serial1.println(S_temp);
  Showmsg=Check_RevData();
  long StartTime=millis();
  while (!Showmsg.startsWith("+QMTRECV:"))
  {
    delay(100);
    Showmsg=Check_RevData();
    if (Showmsg.length()>30) break;
    //Serial.println(Showmsg);
    if ((StartTime+waitingTime) < millis()) return "error";
  }
  //Serial.println(Showmsg);
  return JSON_DEC_data (Showmsg,message);
}
```

所需的額外函式庫:

#include <U8g2lib.h>

下載處 ： https://github.com/olikraus/u8g2

#include <TimeLib.h>

下載處 ： https://github.com/PaulStoffregen/Time

結語:

以 NB-IoT 設計上的概念完全不能以 wifi 的模式來思考,要盡可能的用最低的電能來發揮他最大的效益.而 DSI2598+ 是以 STM32F103C + BC26 為架構所設計的板子,對於高效能與低耗電且會去計較一分一毫電量的需求者,與在一些運用場合上無法提供 wifi 時, DSI2598+ 無非是個最佳的運用選擇.