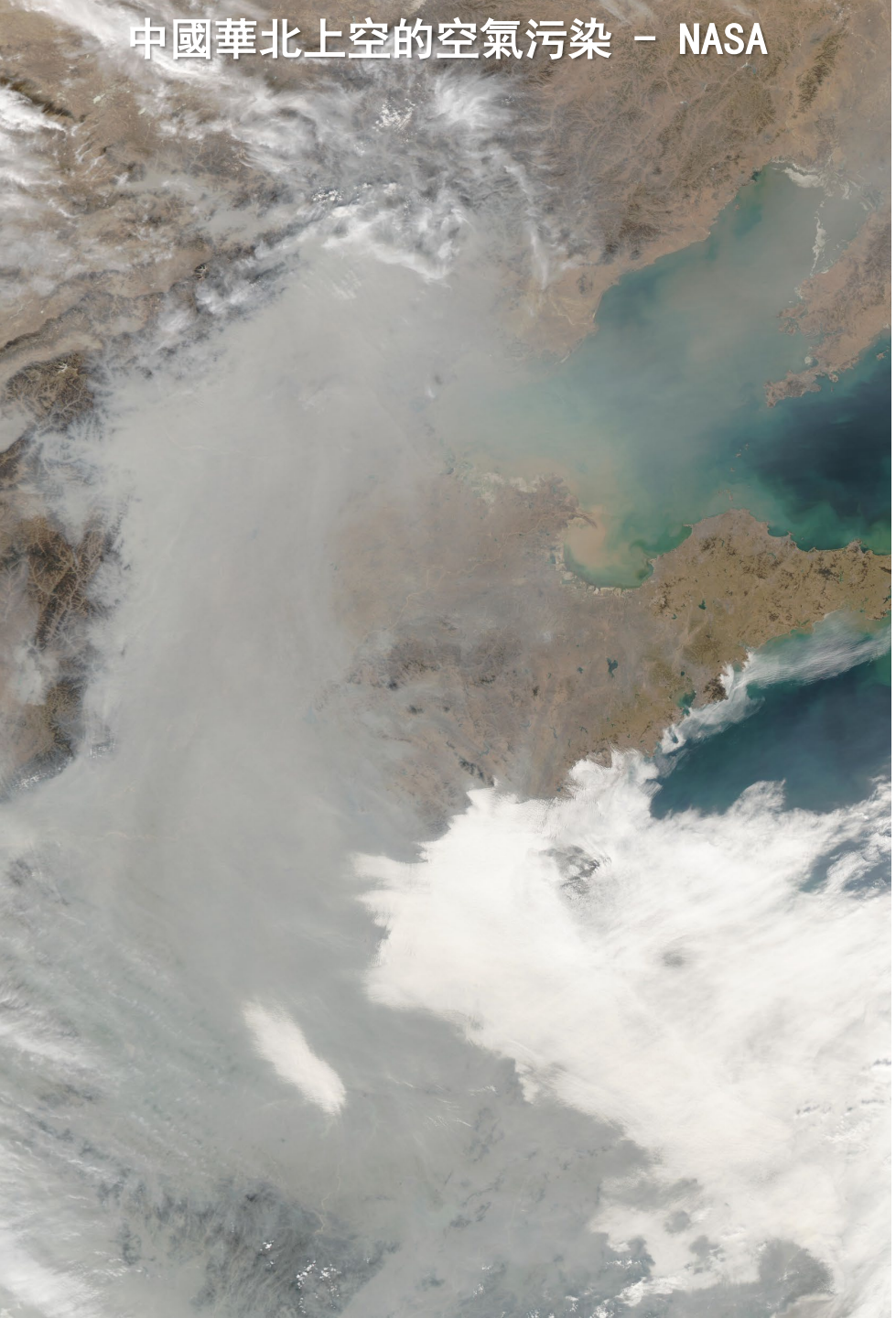
好空氣挑戰賽
CLEAN AIR CHALLENGE

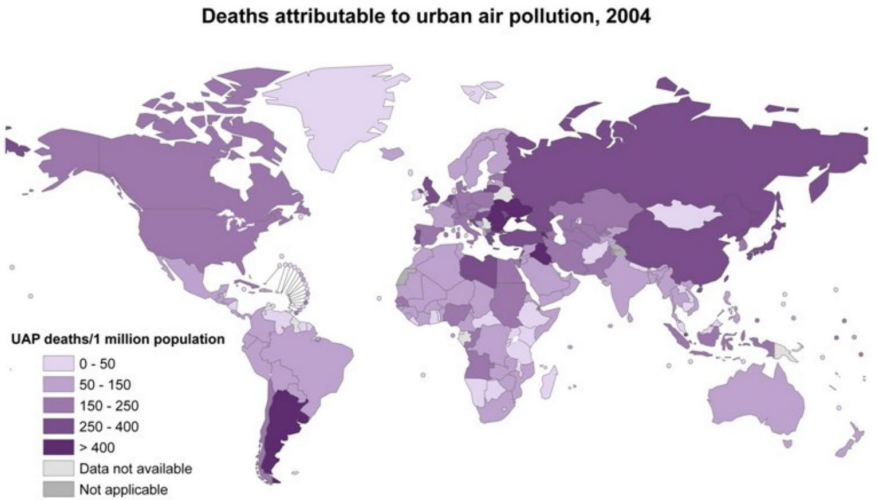
空氣質素傳感器DIY工作坊

講師：香港科技大學環境研究所 鄧昊晴先生 Mr Sunny Deng


中國華北上空的空氣污染 - NASA

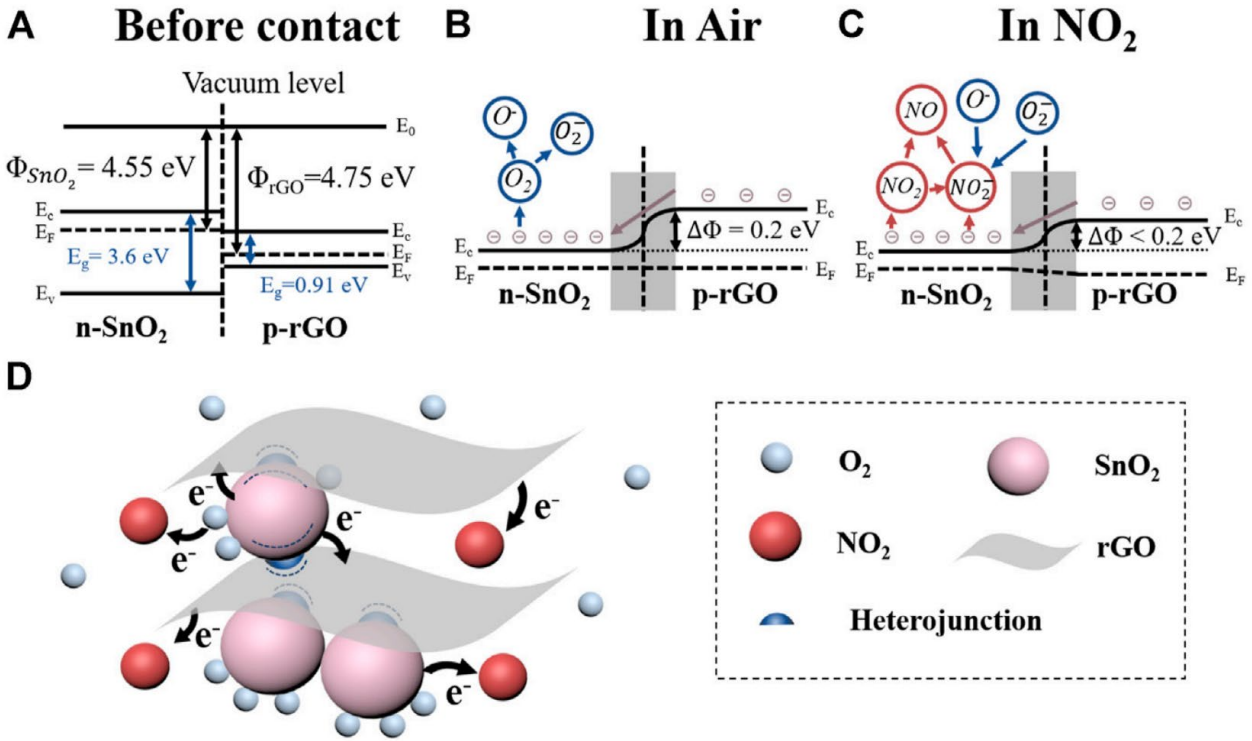
空氣污染

- 危害人體健康及周邊環境物質所對大氣層造成的污染
- 氣體、固體或液體懸浮物都可能成爲對人體有害的污染物
- 對污染物進行準確有效的測量對大氣污染防控預測尤爲重要。



常見的污染物傳感器

- 化學傳感器
  - 利用半導體特性製作表面活性材料，實現對特定污染物的高選擇性檢測
  - 通常用於NO2，O3等化學污染物的檢測



$$NO_2 + e^- \rightarrow NO_2^-$$

$$NO_2 + e^- \rightarrow NO + O^-$$

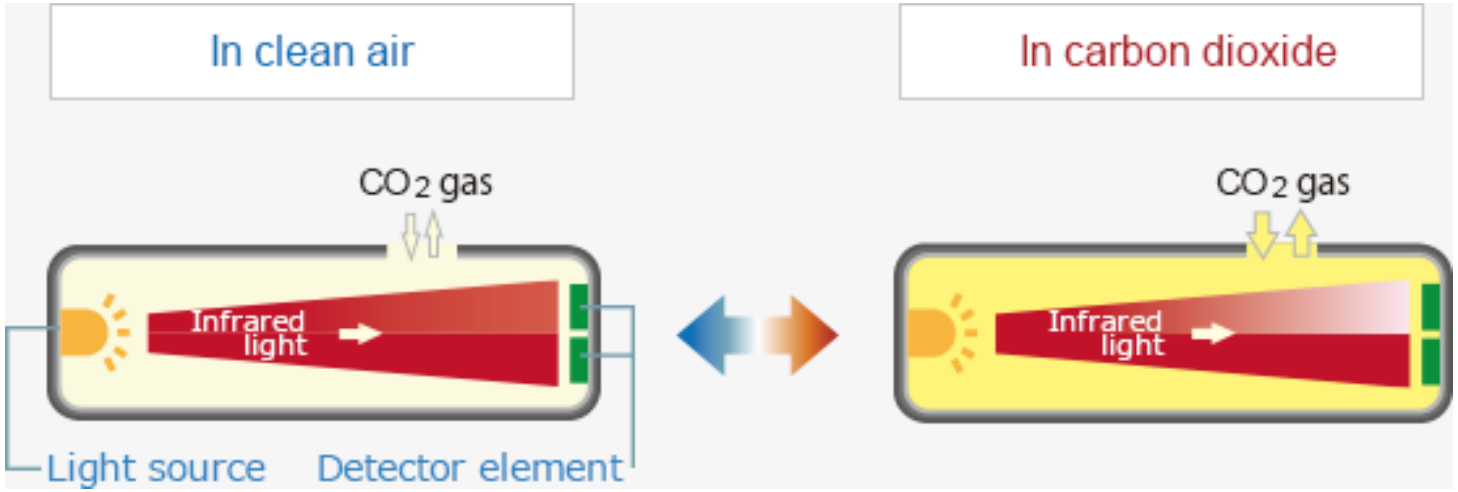
$$2NO_2 + O_2^- + e^- \rightarrow 2NO_2^- + O_2$$

$$NO_2^- + O^- + 2e^- \rightarrow NO + 2O^{2-}$$

Wang, Yingyi, et al. "Humidity-Insensitive NO2 Sensors Based on SnO2/rGO Composites." Frontiers in chemistry 9 (2021): 384.

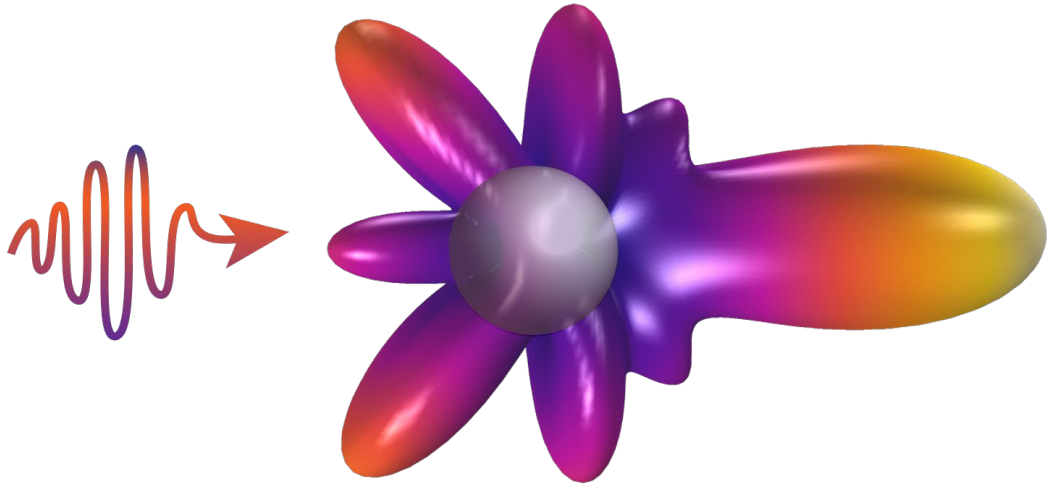
常見的污染物傳感器

- 光學傳感器
  - 利用污染物對光的吸收、反射衍射等光學特性實現對污染物濃度的定量檢測。如:
    - CO2 傳感器：由於中波段紅外綫光子能級與CO2氣體分子振動數相同，分子與紅外綫共振導致能量被氣體吸收，利用這一特性可以製作CO2傳感器

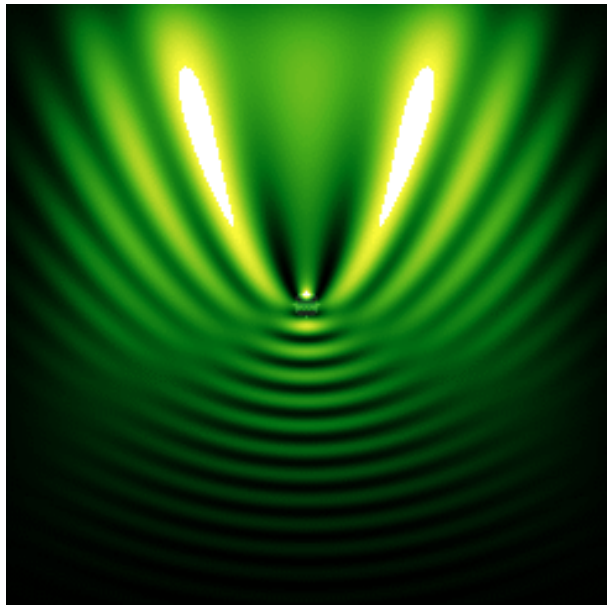


Mie Scattering(米氏散射)

- 當微粒半徑的大小接近於或者大於入射光線的波長的時候，大部分的入射光線會沿着前進的方向進行散射.



Mie scattering by 3eфp @ Wikipedia



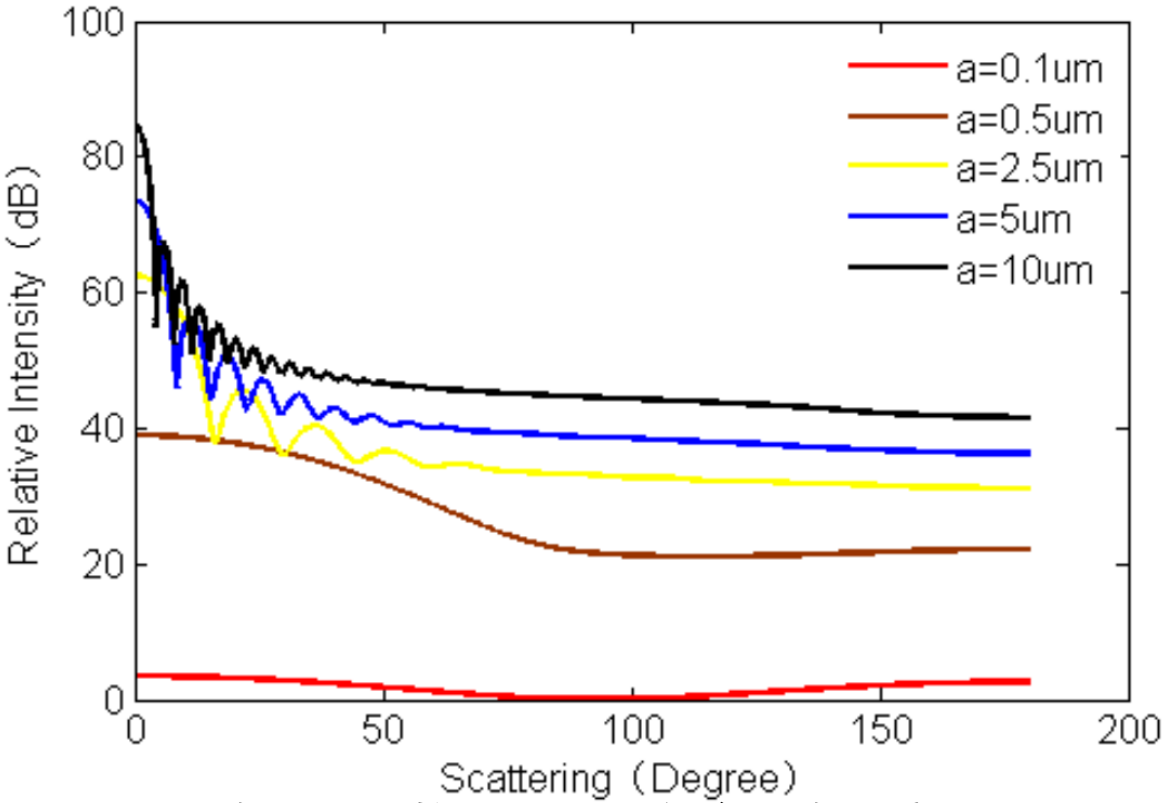
Effect of a dielectric sphere (technically a disk, as the simulation is in 2D) on an incident plane wave as a function of the radius. The patterns you see flashing in are the Mie resonances. The incident plane wave is coming from the bottom. (By Jacopo Bertolotti)

Mie Scattering(米氏散射)

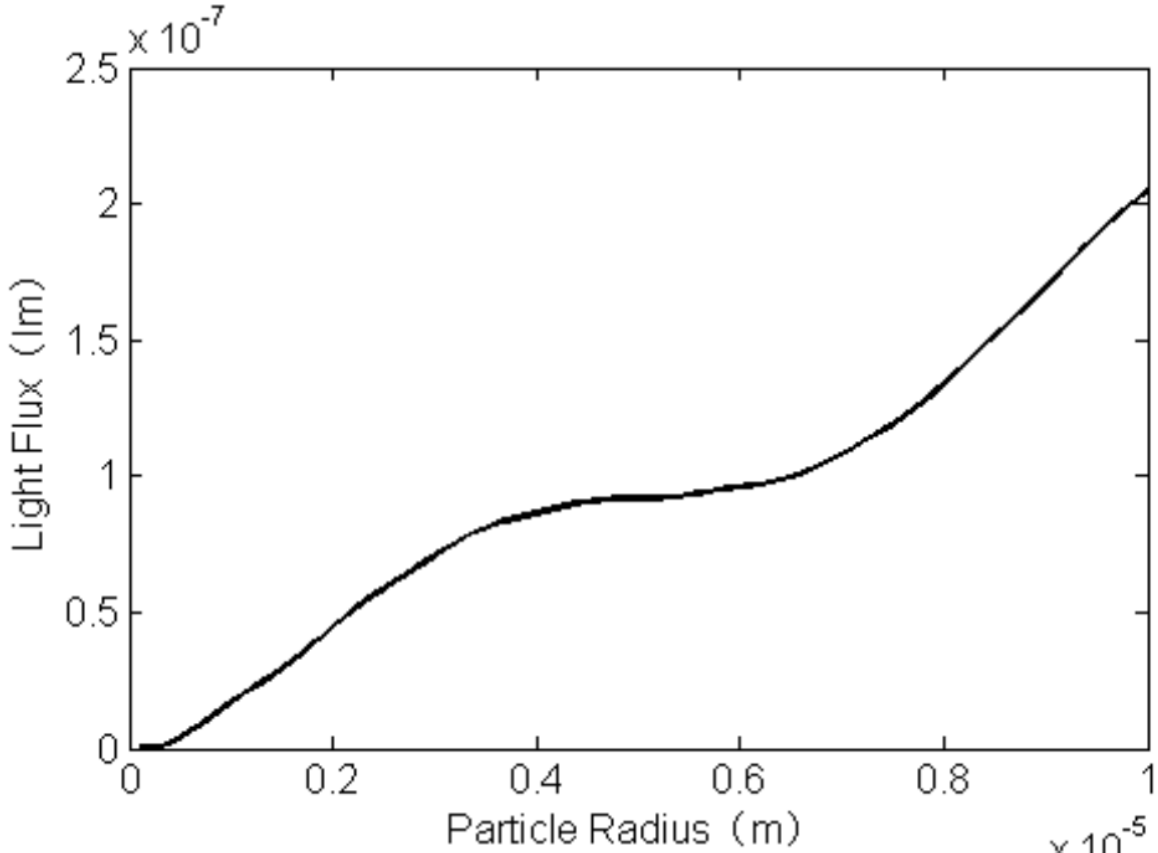
- 當微粒半徑的大小接近於或者大於入射光線的波長λ的時候，大部分的入射光線會沿着前進的方向進行散射.
- 若作用于直徑較大之顆粒物(污染物的顆粒物質，如煙霧等)則會形成丁達爾效應 (Tyndall effect)

Application of Mie Scattering on PM Sensor

- 米氏散射對於不同直徑粒子其散射光强分佈上有所不同。



光束通過粒子后不同角度之光强分佈

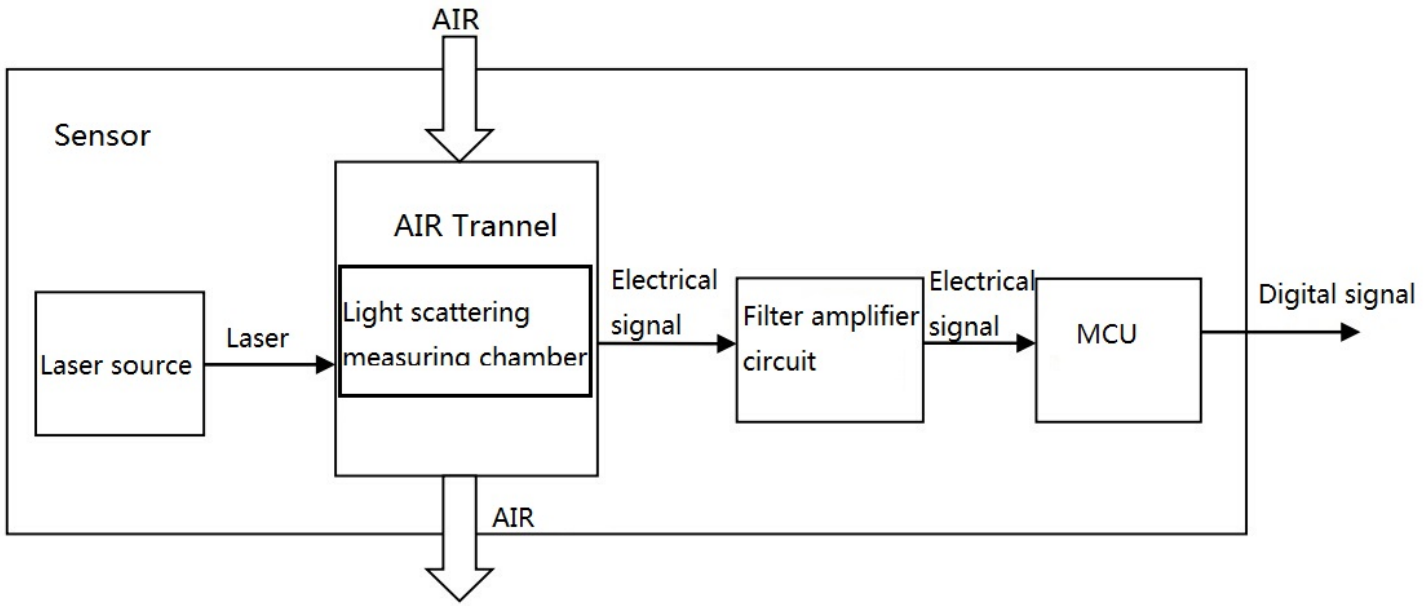


光通量（5°~30°）與顆粒物直徑的關係

Li, Yun, and Jianfeng Tang. "Design of fiber PM2. 5 and PM10 concentration detectors based on MIE scattering." *2019 International Conference on Optical Instruments and Technology: Advanced Laser Technology and Applications*. Vol. 11437. International Society for Optics and Photonics, 2020.

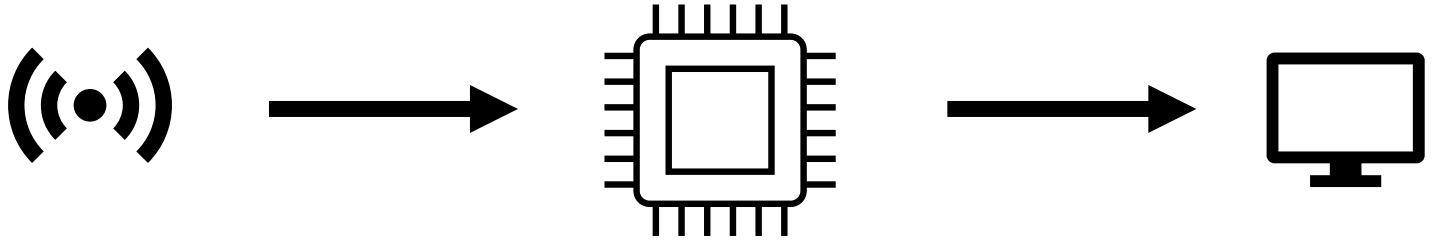
Application of Mie Scattering on PM Sensor

- 米氏散射對於不同直徑粒子其散射光强分佈上有所不同。
- 基於此原理可以檢測當前顆粒物的直徑
- 通過在一定時間區間内顆粒物直徑的判斷結果進行統計，可以計算得到顆粒物的數量，進而得到顆粒物濃度數據。

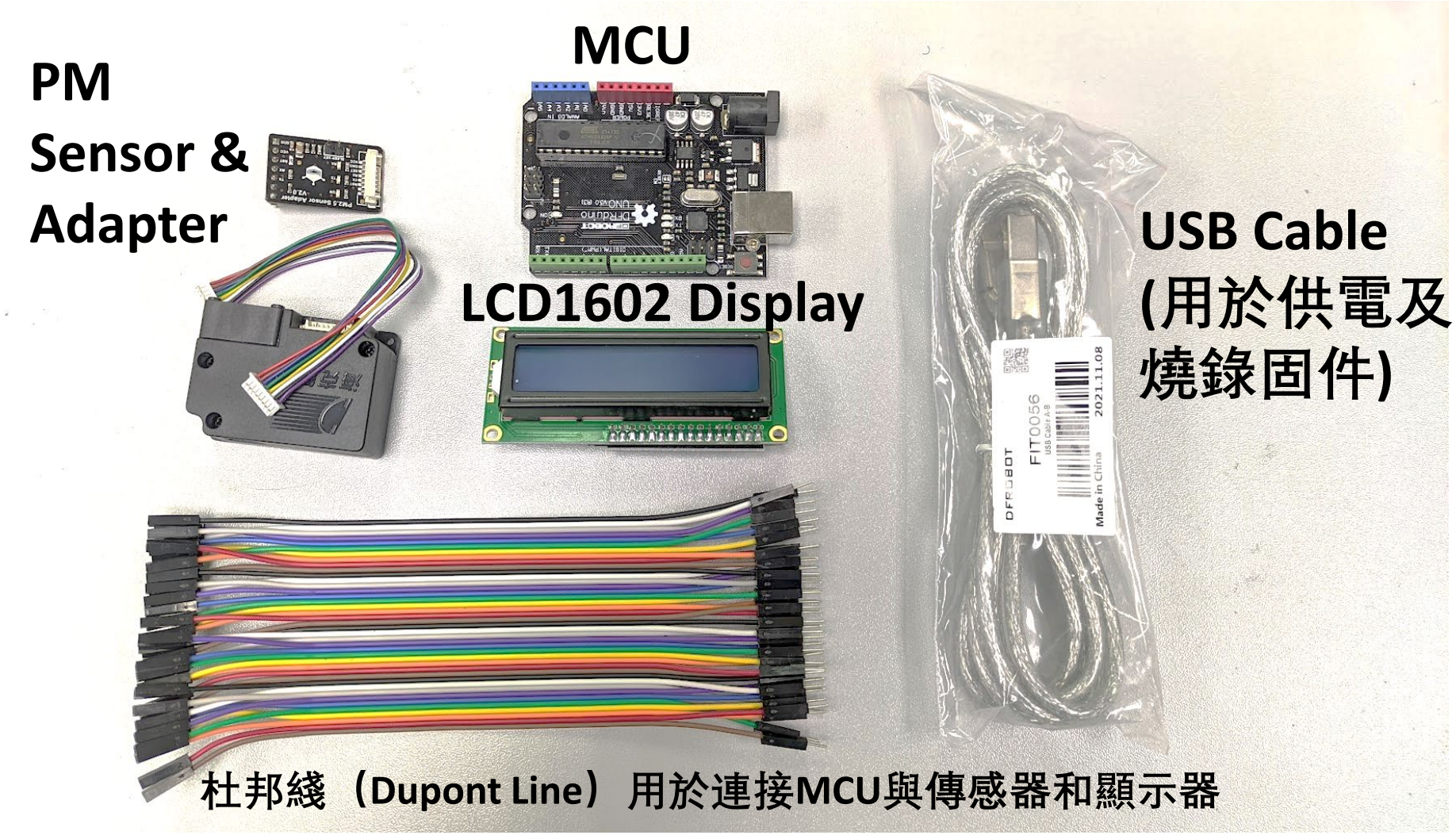


Create a particulate monitor by yourselves

Workflow



Materials that we are going to use

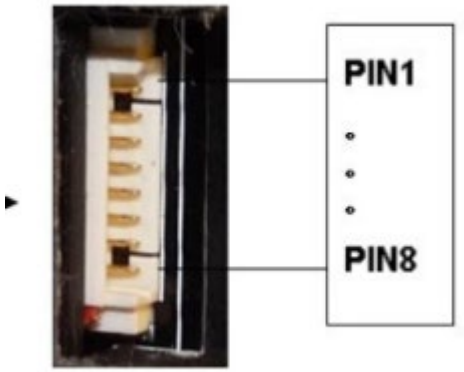


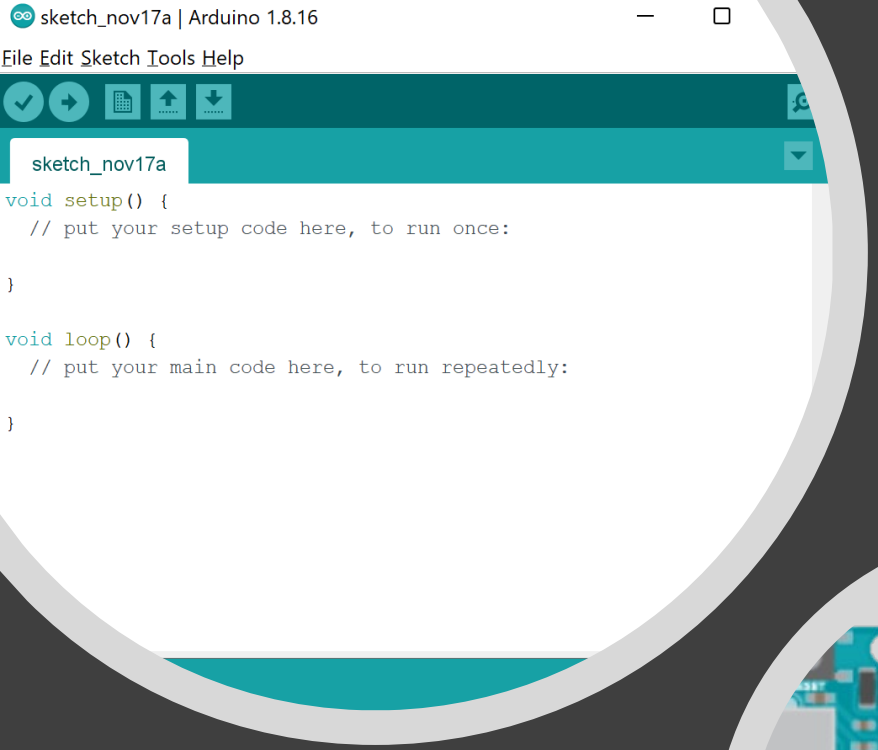
Sensor to be used

- 利用之前介紹的原理實現顆粒物濃度監測
- 能夠檢測0.3~1.0，1.0~2.5，2.5~10μm直徑的顆粒物
- 通過串口輸出測量結果

Sensor to be used

Sensor Pin	Usage	Function Description
Pin 1	VCC	Positive Power
Pin 2	GND	Negative Power
Pin 3	SET	Mode setting (More hereof later)
Pin 4	RXD	receive serial port pin (3.3V level)
Pin 5	TXD	Transferring serial port pin (3.3V level)
Pin 6	RESET	Reset
Pin 7/ 8	NC	NUll



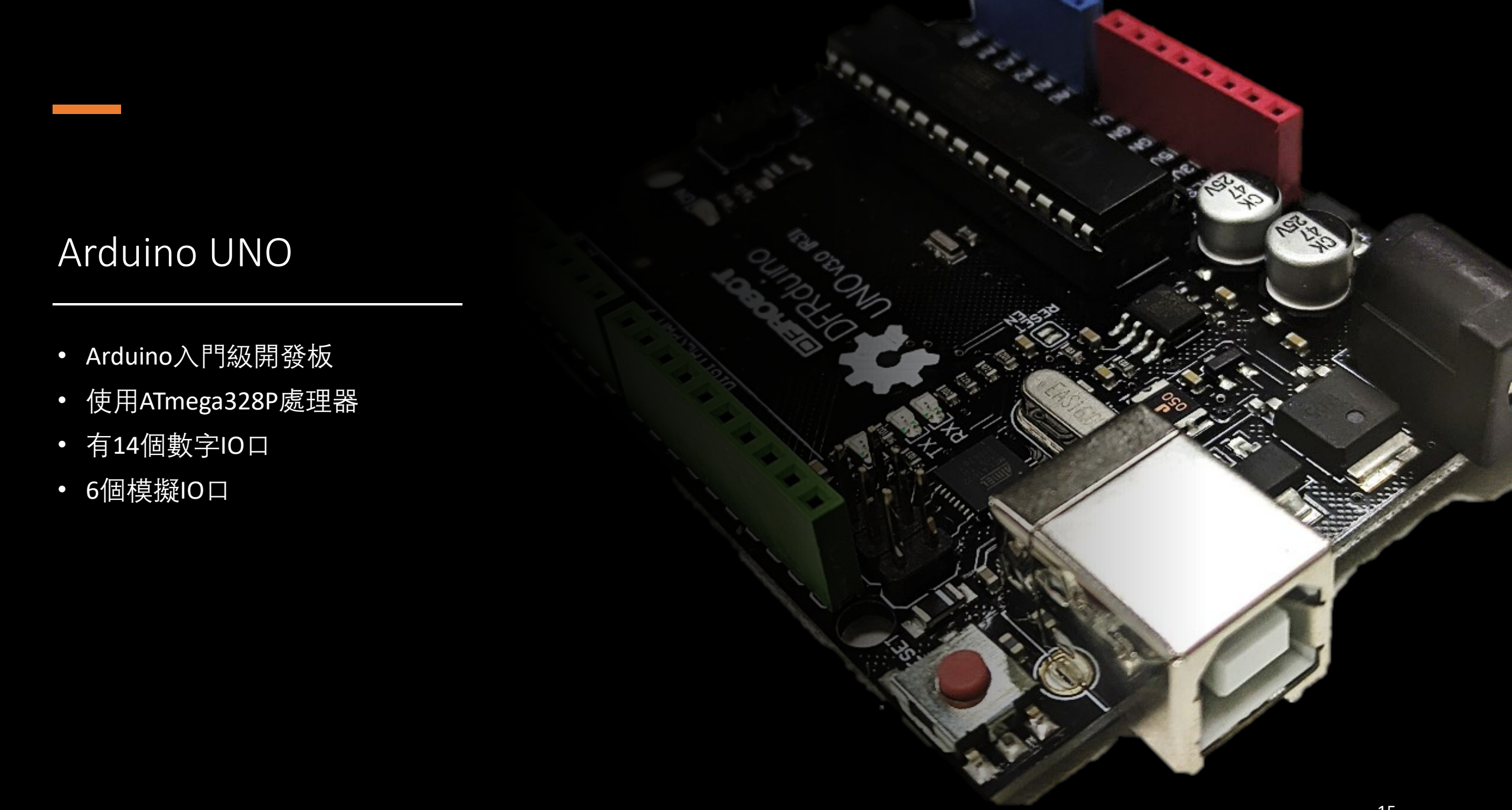


Introduction to Arduino

- 一個開源嵌入式平臺
- 大部分使用Atmel AVR系列晶片，亦有部分使用ARM Cortex-M及其他架構
- 電路設計，電路板設計以及其他應用程式都可以免費獲取，且提供Arduino IDE供初學者使用。

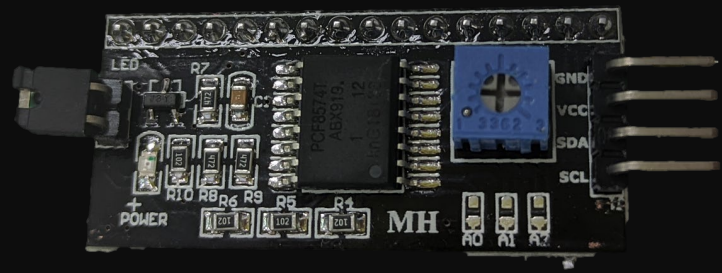
Arduino UNO

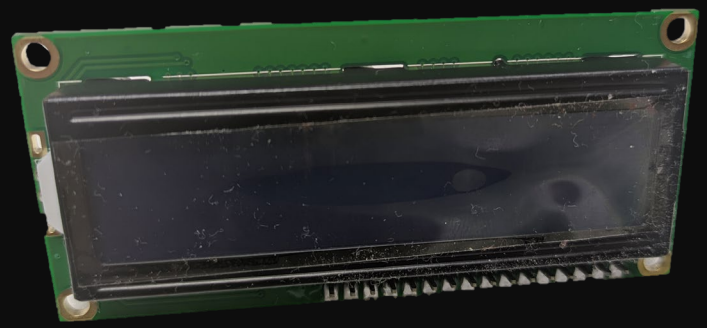
- Arduino入門級開發板
- 使用ATmega328P處理器
- 有14個數字IO口
- 6個模擬IO口



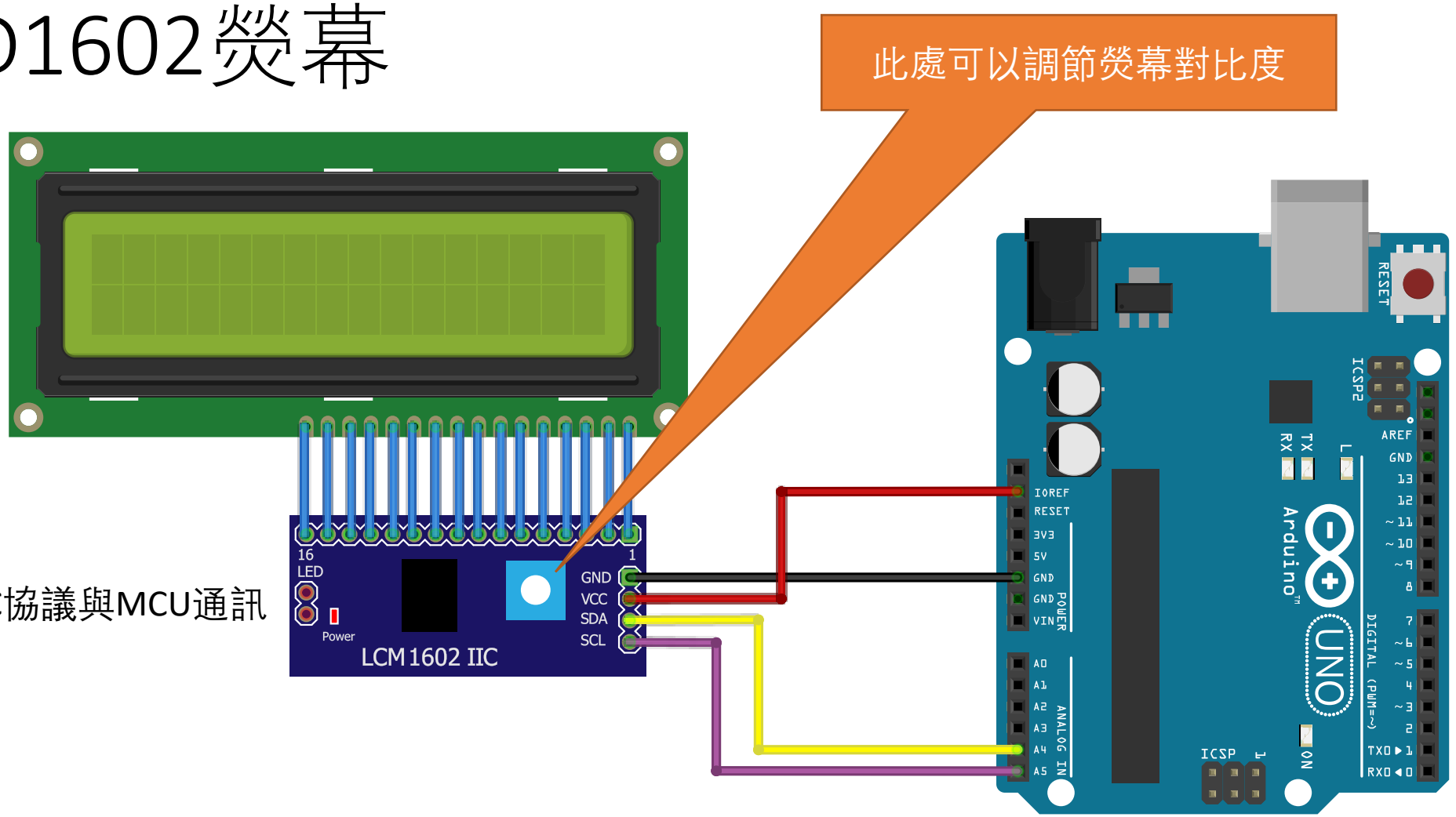
Display Module

- 我們將用1602液晶屏顯示測量的濃度數據
- LCD1602一般指2行16列之LCD模塊
- 能夠顯示英文字母、阿拉伯數字、日文片假名和一般性符號

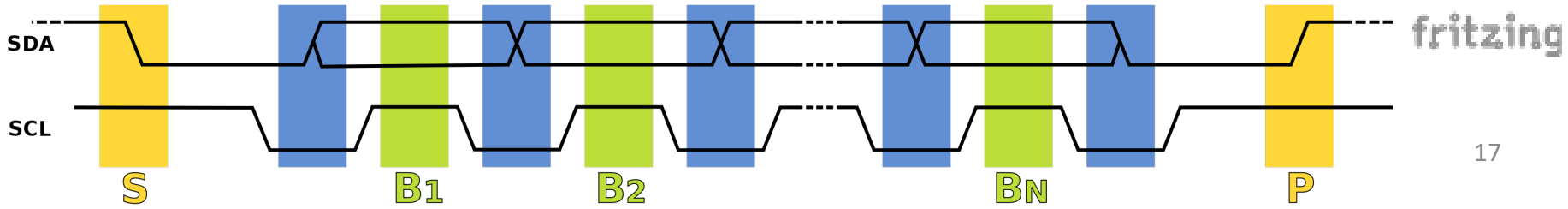




連接LCD1602熒幕

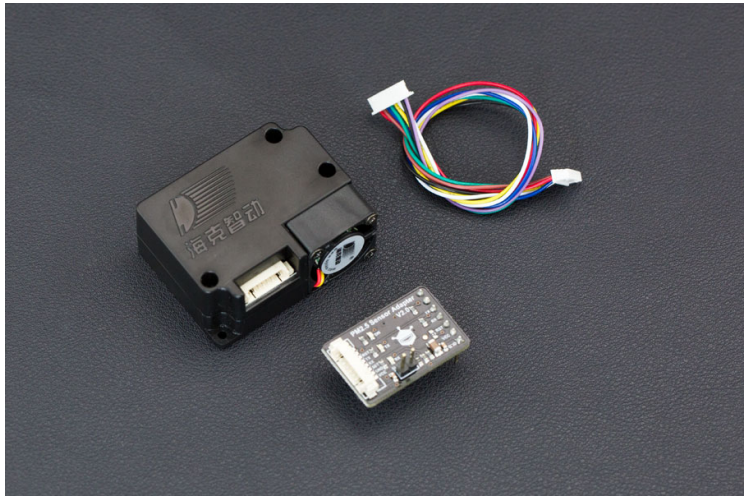


LCD1602通過轉接器利用I2C協議與MCU通訊



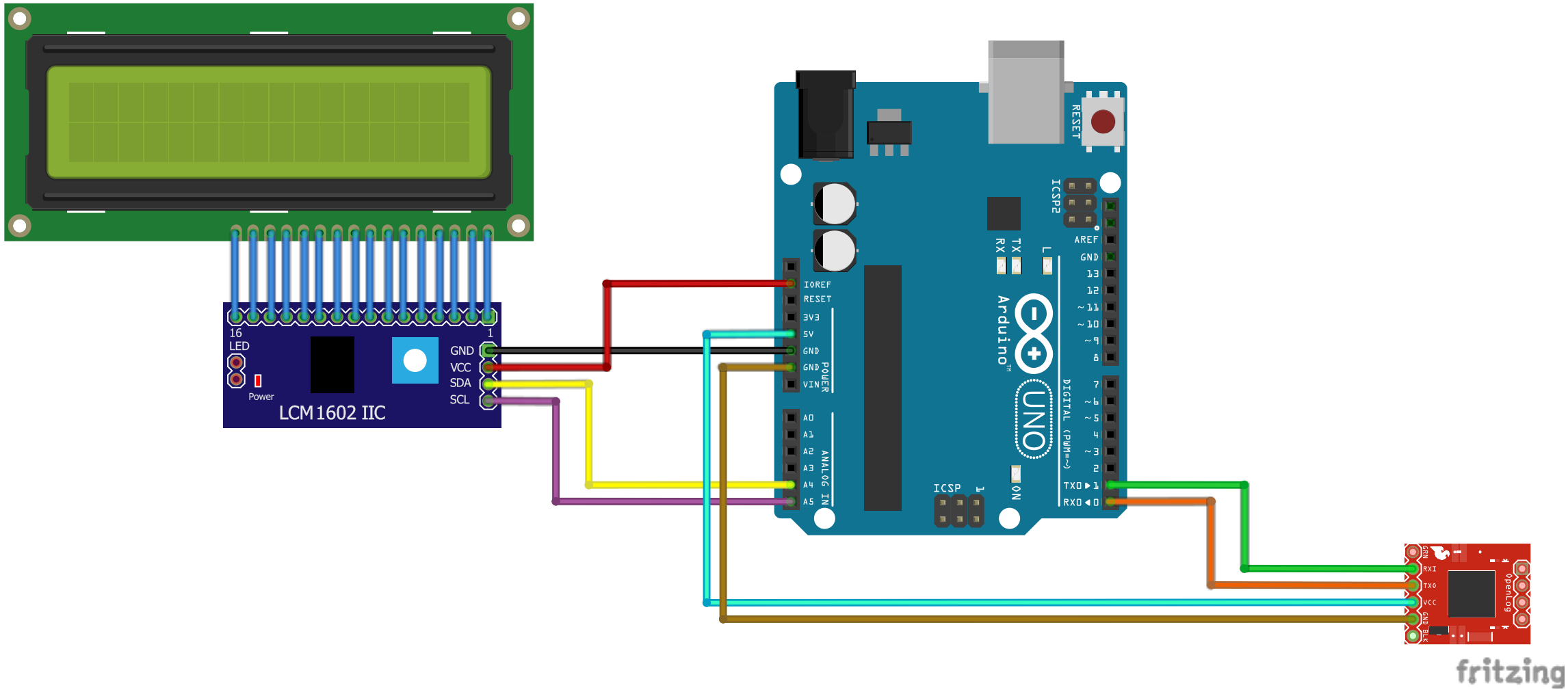
連接顆粒物傳感器

- 顆粒物傳感器通過串口與MCU通訊，該轉接器具有電平轉換，設置傳感器引脚電平至預設值等功能
  - VCC接5V
  - GND接GND
  - TX接RX
  - RX接TX

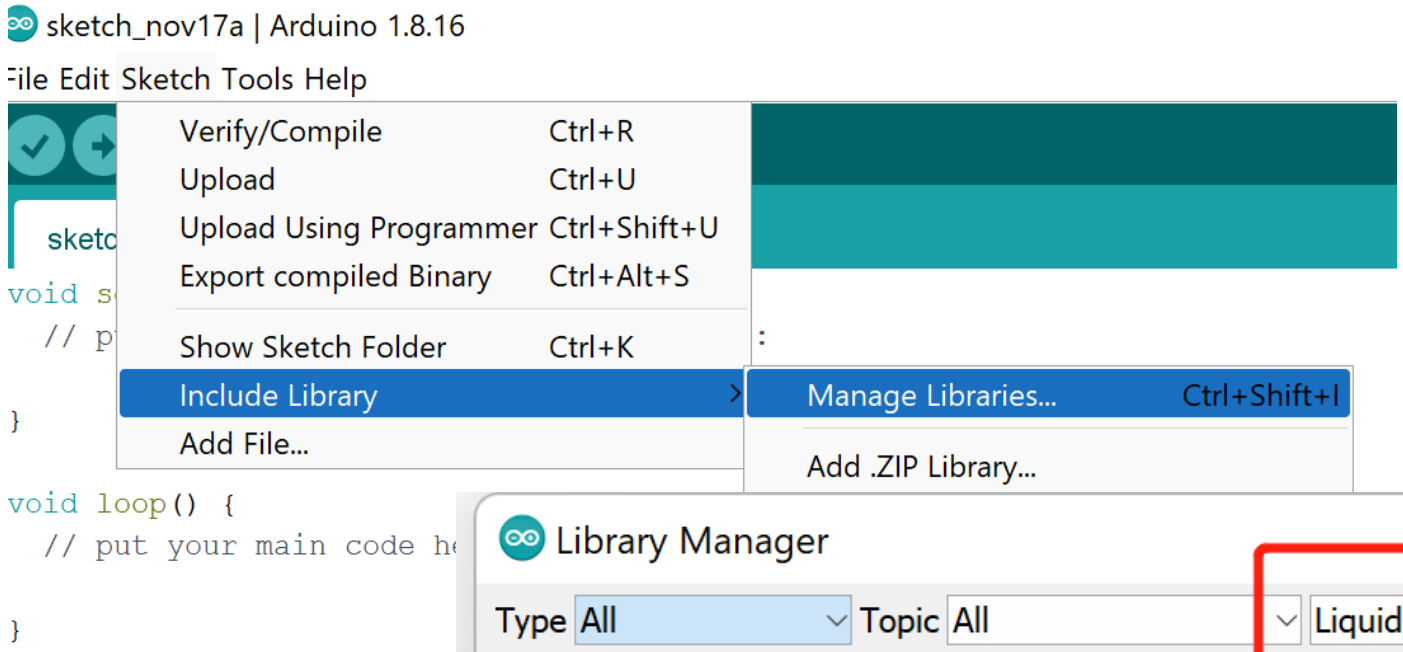


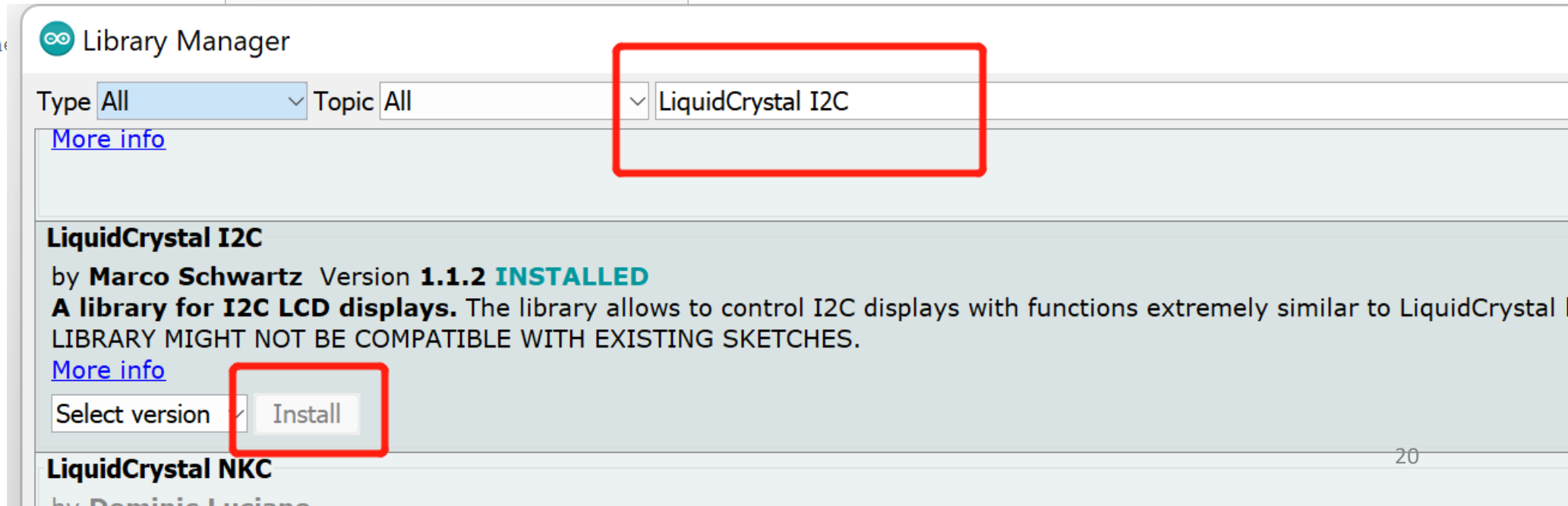
連接時請按照轉接器上的文字説明連接！

Final view


16
LED
Power
1
GND
VCC
SDA
SCL
LCM1602 IIC
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TXD▶1
RXD◀0
DIGITAL (PWM=~)
TX
RX
L
ON
ICSP
1
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
UNO
Arduino™
GRN
RXI
TXO
VCC
GND BLK
openlog
fritzing

Install library for LCD1602

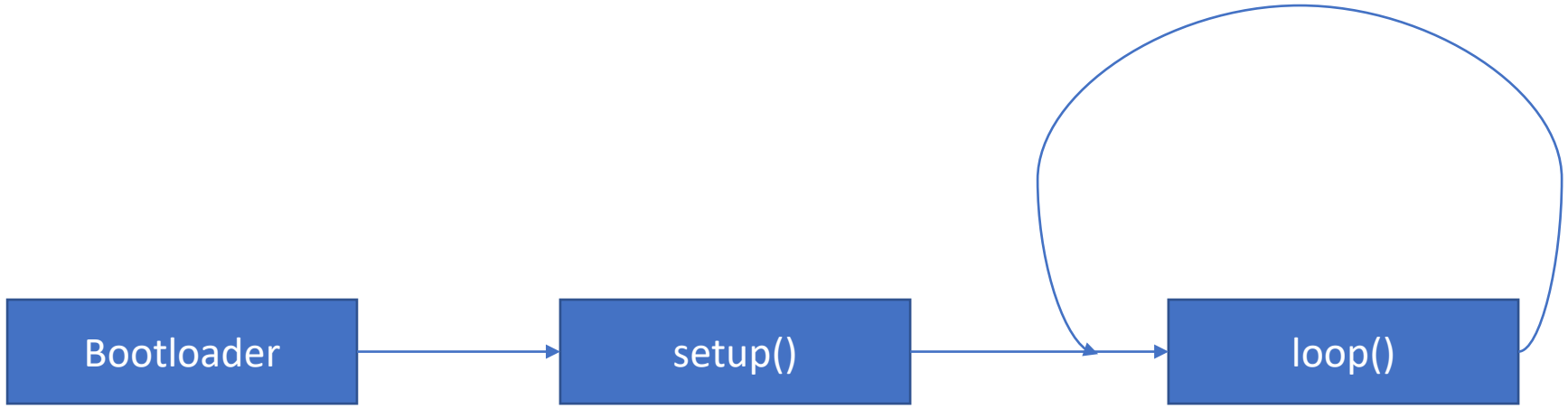




目標

- MCU讀取顆粒物傳感器的顆粒物濃度數據，並顯示在液晶屏上
- 考慮到顆粒物傳感器能夠檢測PM1.0，PM2.5以及PM10的濃度，而LCD1602無法同時顯示這麼多資訊，因此考慮第二行每隔5秒輪流顯示PM1.0與PM10的數據。
- 每1秒刷新一次數據。

Coding



Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2); // address: 0x27, cols:16, rows:2
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;

// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2); // address: 0x27, cols:16, rows:2
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;

// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

- 指定I2C通訊目標芯片的地址為0x27，同時初始化這個對象，指定顯示器參數為可以顯示2行，每行16個字符

Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2); // address: 0x27, cols:16, rows:2
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;

// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

- 用於記錄上次切換PM1.0/PM10數據顯示的時間
- 用於記錄上次刷新的時間
- 之後可以每次獲取當前時間，從而判斷是否應該刷新或切換顯示，若是的話則將showPM10設置為true

Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2);
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;
```

共32個字節，減去一個已讀字節所以為31

```
// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

Start Character 1	0x42(fixed bit)
Start Character 2	0x4d(fixed bit)
Frame Length 16-byte	Frame Length = 2*9+2 (data+check bit)
Data 1, 16-byte	concentration of PM1.0, ug/m3
Data 2, 16-byte	concentration of PM2.5, ug/m3
Data 3, 16-byte	concentration of PM10.0, ug/m3
Data 4, 16-byte	Internal test data
Data 5, 16-byte	Internal test data
Data 6, 16-byte	Internal test data
Data 7, 16-byte	the number of particulate of diameter above 0.3um in 0.1 liters of air
Data 8, 16-byte	the number of particulate of diameter above 0.5um in 0.1 liters of air
Data 9, 16-byte	the number of particulate of diameter above 1.0um in 0.1 liters of air
Data 10, 16-byte	the number of particulate of diameter above 2.5um in 0.1 liters of air
Data 11, 16-byte	the number of particulate of diameter above 5.0um in 0.1 liters of air
Data 12, 16-byte	the number of particulate of diameter above 10.0um in 0.1 liters of air
Data 13, 16-byte	Internal test data
Check Bit for Data Sum, 16-byte	Check Bit = Start Character 1 + Start Character 2 + ...all data

Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2);
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;

// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

Start Character 1	0x42(fixed bit)
Start Character 2	0x4d(fixed bit)
Frame Length 16-byte	Frame Length = 2*9+2 (data+check bit)
Data 1, 16-byte	concentration of PM1.0, ug/m3
Data 2, 16-byte	concentration of PM2.5, ug/m3
Data 3, 16-byte	concentration of PM10.0, ug/m3
Data 4, 16-byte	Internal test data
Data 5, 16-byte	Internal test data
Data 6, 16-byte	Internal test data
Data 7, 16-byte	the number of particulate of diameter above 0.3um in 0.1 liters of air
Data 8, 16-byte	the number of particulate of diameter above 0.5um in 0.1 liters of air
Data 9, 16-byte	the number of particulate of diameter above 1.0um in 0.1 liters of air
Data 10, 16-byte	the number of particulate of diameter above 2.5um in 0.1 liters of air
Data 11, 16-byte	the number of particulate of diameter above 5.0um in 0.1 liters of air
Data 12, 16-byte	the number of particulate of diameter above 10.0um in 0.1 liters of air
Data 13, 16-byte	Internal test data
Check Bit for Data Sum, 16-byte	Check Bit = Start Character 1 + Start Character 2 + ...all data

Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2);
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;

// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

Start Character 1	0x42(fixed bit)
Start Character 2	0x4d(fixed bit)
Frame Length 16-byte	Frame Length = 2*9+2 (data+check bit)
Data 1, 16-byte	concentration of PM1.0, ug/m3
Data 2, 16-byte	concentration of PM2.5, ug/m3
Data 3, 16-byte	concentration of PM10.0, ug/m3
Data 4, 16-byte	Internal test data
Data 5, 16-byte	Internal test data
Data 6, 16-byte	Internal test data
Data 7, 16-byte	the number of particulate of diameter above 0.3um in 0.1 liters of air
Data 8, 16-byte	the number of particulate of diameter above 0.5um in 0.1 liters of air
Data 9, 16-byte	the number of particulate of diameter above 1.0um in 0.1 liters of air
Data 10, 16-byte	the number of particulate of diameter above 2.5um in 0.1 liters of air
Data 11, 16-byte	the number of particulate of diameter above 5.0um in 0.1 liters of air
Data 12, 16-byte	the number of particulate of diameter above 10.0um in 0.1 liters of air
Data 13, 16-byte	Internal test data
Check Bit for Data Sum, 16-byte	Check Bit = Start Character 1 + Start Character 2 + ...all data

Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2);
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;

// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

Start Character 1	0x42(fixed bit)
Start Character 2	0x4d(fixed bit)
Frame Length 16-byte	Frame Length = 2*9+2 (data+check bit)
Data 1, 16-byte	concentration of PM1.0, ug/m3
Data 2, 16-byte	concentration of PM2.5, ug/m3
Data 3, 16-byte	concentration of PM10.0, ug/m3
Data 4, 16-byte	Internal test data
Data 5, 16-byte	Internal test data
Data 6, 16-byte	Internal test data
Data 7, 16-byte	the number of particulate of diameter above 0.3um in 0.1 liters of air
Data 8, 16-byte	the number of particulate of diameter above 0.5um in 0.1 liters of air
Data 9, 16-byte	the number of particulate of diameter above 1.0um in 0.1 liters of air
Data 10, 16-byte	the number of particulate of diameter above 2.5um in 0.1 liters of air
Data 11, 16-byte	the number of particulate of diameter above 5.0um in 0.1 liters of air
Data 12, 16-byte	the number of particulate of diameter above 10.0um in 0.1 liters of air
Data 13, 16-byte	Internal test data
Check Bit for Data Sum, 16-byte	Check Bit = Start Character 1 + Start Character 2 + ...all data

Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2);
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;

// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

Start Character 1	0x42(fixed bit)
Start Character 2	0x4d(fixed bit)
Frame Length 16-byte	Frame Length = 2*9+2 (data+check bit)
Data 1, 16-byte	concentration of PM1.0, ug/m3
Data 2, 16-byte	concentration of PM2.5, ug/m3
Data 3, 16-byte	concentration of PM10.0, ug/m3
Data 4, 16-byte	Internal test data
Data 5, 16-byte	Internal test data
Data 6, 16-byte	Internal test data
Data 7, 16-byte	the number of particulate of diameter above 0.3um in 0.1 liters of air
Data 8, 16-byte	the number of particulate of diameter above 0.5um in 0.1 liters of air
Data 9, 16-byte	the number of particulate of diameter above 1.0um in 0.1 liters of air
Data 10, 16-byte	the number of particulate of diameter above 2.5um in 0.1 liters of air
Data 11, 16-byte	the number of particulate of diameter above 5.0um in 0.1 liters of air
Data 12, 16-byte	the number of particulate of diameter above 10.0um in 0.1 liters of air
Data 13, 16-byte	Internal test data
Check Bit for Data Sum, 16-byte	Check Bit = Start Character 1 + Start Character 2 + ...all data

Coding

```
#include <LiquidCrystal_I2C.h>

// Initialize resource for LCD Display
LiquidCrystal_I2C lcd(0x27, 16, 2);
unsigned long lastSwitchDataTime = 0;
unsigned long lastUpdateTime = 0;
bool showPM10 = false;

// Initialize resource for PM Sensor
#define DATA_LENGTH 31
#define PM1D0_POS 1
#define PM2D5_POS 2
#define PM10_POS 3
#define CHECKSUM_POS 14
int pm1d0Value = 0;
int pm2d5Value = 0;
int pm10Value = 0;
char pm2d5Str[16] = {0};
char pm10Str[16] = {0};
char pm1d0Str[16] = {0};
unsigned char pmBuffer[DATA_LENGTH];
```

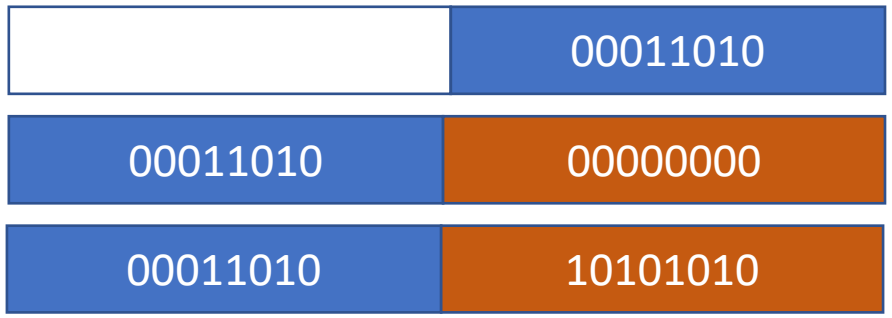
- 初始化儲存當前顆粒物濃度的內存空間
- 初始化用於輸出濃度數據的字符串內存空間
- 初始化緩存區，用於儲存傳感器通過串口傳輸到MCU的原始數據包

```
unsigned char pmBuffer[DATA_LENGTH];
```

Coding

```
int parseDataAtPosition(unsigned char* buf, int pos) {
    return (buf[1 + 2 * pos] << 8) + buf[2 + 2 * pos];
}
```

- 此函數用於讀取buffer中指定位置的數據
- 由於傳感器發過來的數據是一個16位的數據類型（2個字節），而通訊過程我們傳輸的最小是數據單元一般是8bit的單字節數據，我們需要將其進行轉換。



A=26(二進制為00011010)

A<<8 左移8位

(A<<8) + 170=6826

Coding

- 此函數用於校驗接收的數據包是否有錯誤
- 通過纍加前面接收的全部數據，並與最後一個校驗位進行校驗，如果一致則可認爲數據是完整的。

```
bool validate_checksum(unsigned char* buf, char len) {
    // calculate checksum
    int checksum = 0x42;
    for (int i = 0; i < (len - 2); i++) {
        checksum += buf[i];
    }

    // validate checksum
    if (checksum == parseDataAtPosition(buf, CHECKSUM_POS)) {
        return true;
    }
    return false;
}
```

Check Bit for Data Sum, 16-byte	Check Bit = Start Character 1 + Start Character 2 + ...all data

Coding

- 初始化LCD1602
- 啓動背光
- 清空屏幕内容

- 開啓串口，設置串口波特率為9600
- 設置串口超時為1500ms

```
void setup() {
    // Initialize LCD Display
    lcd.init();
    lcd.backlight();
    lcd.clear();

    // Initial serial for PM Sensor
    Serial.begin(9600);
    Serial.setTimeout(1500);
}
```

Coding

```
void loop() {
  // put your main code here, to run repeatedly:
  // Parse data sent from the PM sensor
  if (Serial.find(0x42)) {
    Serial.readBytes(pmBuffer, DATA_LENGTH);
    if (pmBuffer[0] == 0x4d && validate_checksum(pmBuffer, DATA_LENGTH)) {
      pm1d0Value = parseDataAtPosition(pmBuffer, PM1D0_POS);
      pm2d5Value = parseDataAtPosition(pmBuffer, PM2D5_POS);
      pm10Value = parseDataAtPosition(pmBuffer, PM10_POS);
    }
  }
```

- 此處調用串口API監控輸入數據流中的數據，若發現0x42這個數值則進入數據讀取流程，讀取數據包餘下31個字節的數據
- 數據包第二個字節為0x4d，即符合文檔中提到的包頭為0x42，0x4d；且checksum校驗通過，則開始解析數據包中的污染物數據

Coding

- 此處不斷獲取當前時間，并且與前面定義的lastSwitchDataTime和lastUpdateTime比對，如果超過5秒或者1秒，則分別執行顯示數據切換以及更新顯示屏的信息顯示。

```
    //Switch display type
    if (millis() - lastSwitchDataTime > 5000) {
      lastSwitchDataTime = millis();
      showPM10 = !showPM10;
    }

    // display data
    if (millis() - lastUpdateTime > 1000) {
      lastUpdateTime = millis();
      memset(pm1d0Str, 0, 16);
      memset(pm2d5Str, 0, 16);
      memset(pm10Str, 0, 16);
      sprintf(pm1d0Str, "PM1.0:%dug/m3", pm1d0Value);
      sprintf(pm2d5Str, "PM2.5:%dug/m3", pm2d5Value);
      sprintf(pm10Str, "PM10:%dug/m3", pm10Value);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(pm2d5Str);
      lcd.setCursor(0, 1);
      lcd.print(showPM10 ? pm10Str : pm1d0Str);
    }
}
```

Coding

- 由於我們復用了同一個顯示buffer用作顯示字符串，因此此處我們需要對buffer做一定的初始化，避免先前數據殘留導致顯示異常。

```
    //Switch display type
    if (millis() - lastSwitchDataTime > 5000) {
        lastSwitchDataTime = millis();
        showPM10 = !showPM10;
    }

    // display data
    if (millis() - lastUpdateTime > 1000) {
        lastUpdateTime = millis();
        memset(pm1d0Str, 0, 16);
        memset(pm2d5Str, 0, 16);
        memset(pm10Str, 0, 16);
        sprintf(pm1d0Str, "PM1.0:%dug/m3", pm1d0Value);
        sprintf(pm2d5Str, "PM2.5:%dug/m3", pm2d5Value);
        sprintf(pm10Str, "PM10:%dug/m3", pm10Value);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(pm2d5Str);
        lcd.setCursor(0, 1);
        lcd.print(showPM10 ? pm10Str : pm1d0Str);
    }
}
```

Coding

- 此處將格式化后的字符串寫入到對應的字符串buffer內。

```
    //Switch display type
    if (millis() - lastSwitchDataTime > 5000) {
        lastSwitchDataTime = millis();
        showPM10 = !showPM10;
    }

    // display data
    if (millis() - lastUpdateTime > 1000) {
        lastUpdateTime = millis();
        memset(pm1d0Str, 0, 16);
        memset(pm2d5Str, 0, 16);
        memset(pm10Str, 0, 16);
        sprintf(pm1d0Str, "PM1.0:%dug/m3", pm1d0Value);
        sprintf(pm2d5Str, "PM2.5:%dug/m3", pm2d5Value);
        sprintf(pm10Str, "PM10:%dug/m3", pm10Value);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(pm2d5Str);
        lcd.setCursor(0, 1);
        lcd.print(showPM10 ? pm10Str : pm1d0Str);
    }
}
```

Coding

- 此處先清空顯示器的顯示
- 然後將對應需要顯示的字符串打印到顯示器上

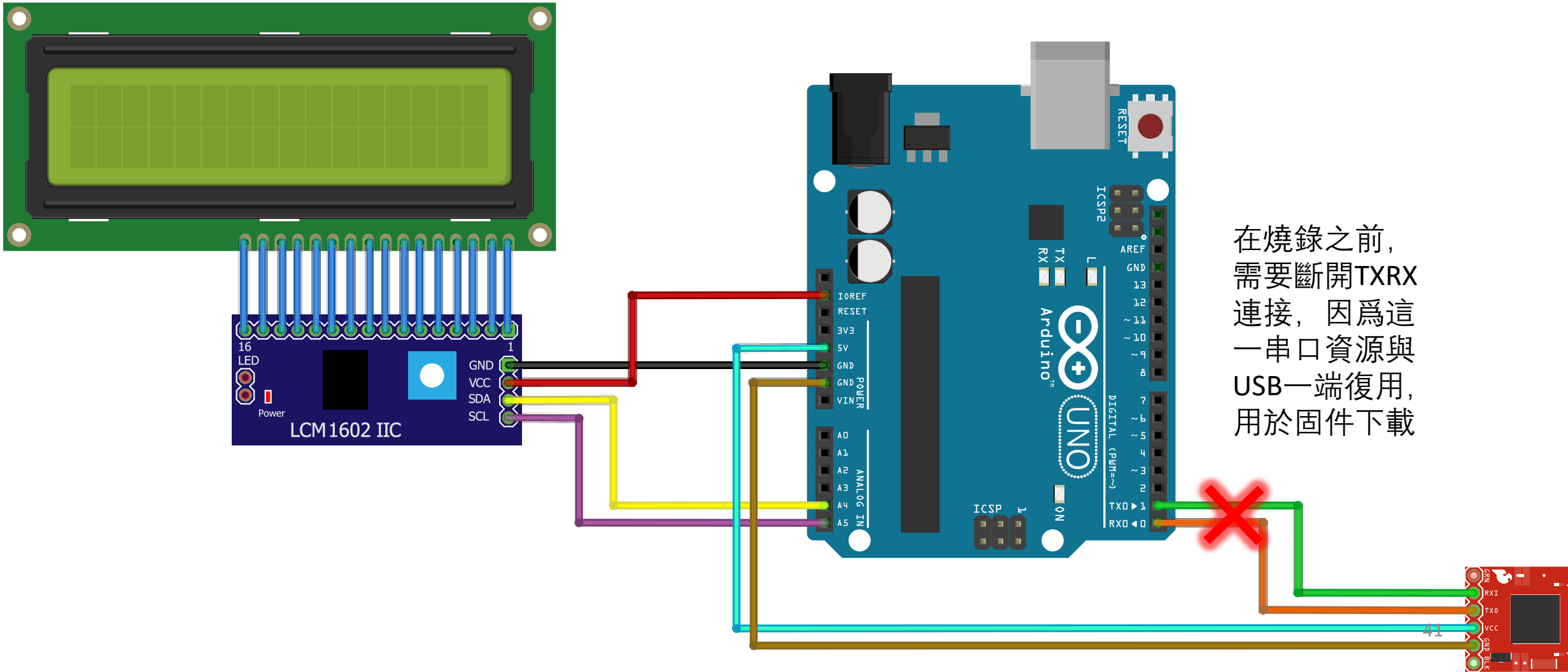
```
    //Switch display type
    if (millis() - lastSwitchDataTime > 5000) {
        lastSwitchDataTime = millis();
        showPM10 = !showPM10;
    }

    // display data
    if (millis() - lastUpdateTime > 1000) {
        lastUpdateTime = millis();
        memset(pm1d0Str, 0, 16);
        memset(pm2d5Str, 0, 16);
        memset(pm10Str, 0, 16);
        sprintf(pm1d0Str, "PM1.0:%dug/m3", pm1d0Value);
        sprintf(pm2d5Str, "PM2.5:%dug/m3", pm2d5Value);
        sprintf(pm10Str, "PM10:%dug/m3", pm10Value);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(pm2d5Str);
        lcd.setCursor(0, 1);
        lcd.print(showPM10 ? pm10Str : pm1d0Str);
    }
}
```

Coding

最終的loop函數如右圖所示

```
void loop() {
  // put your main code here, to run repeatedly:
  // Parse data sent from the PM sensor
  if (Serial.find(0x42)) {
    Serial.readBytes(pmBuffer, DATA_LENGTH);
    if (pmBuffer[0] == 0x4d && validate_checksum(pmBuffer, DATA_LENGTH)) {
      pm1d0Value = parseDataAtPosition(pmBuffer, PM1D0_POS);
      pm2d5Value = parseDataAtPosition(pmBuffer, PM2D5_POS);
      pm10Value = parseDataAtPosition(pmBuffer, PM10_POS);
    }
  }
  //Switch display type
  if (millis() - lastSwitchDataTime > 5000) {
    lastSwitchDataTime = millis();
    showPM10 = !showPM10;
  }
  // display data
  if (millis() - lastUpdateTime > 1000) {
    lastUpdateTime = millis();
    memset(pm1d0Str, 0, 16);
    memset(pm2d5Str, 0, 16);
    memset(pm10Str, 0, 16);
    sprintf(pm1d0Str, "PM1.0:%dug/m3", pm1d0Value);
    sprintf(pm2d5Str, "PM2.5:%dug/m3", pm2d5Value);
    sprintf(pm10Str, "PM10:%dug/m3", pm10Value);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(pm2d5Str);
    lcd.setCursor(0, 1);
    lcd.print(showPM10 ? pm10Str : pm1d0Str);
  }
}
```

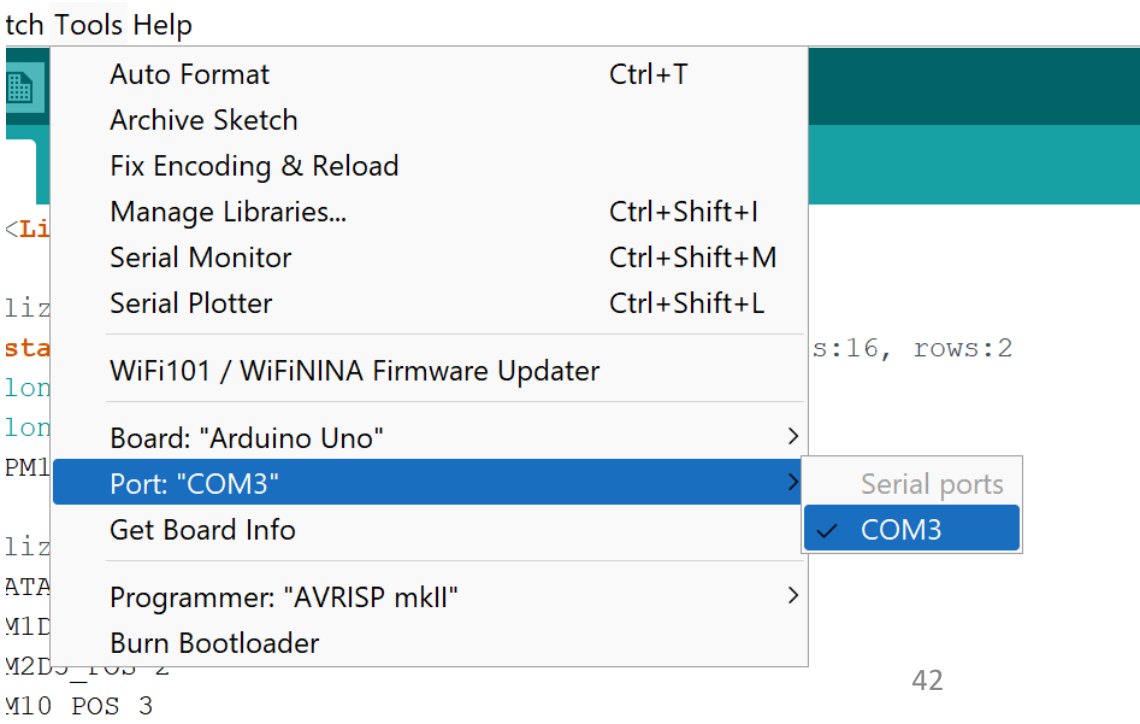
Download Program to the MCU



在燒錄之前，需要斷開TXRX連接，因爲這一串口資源與USB一端復用，用於固件下載

Download Program to the MCU

- 使用USB綫與電腦連接
- 連接后此處會新增一個新的串口設備（注意不同電腦顯示的設備名不同，有可能是是COM5，COM6，…）

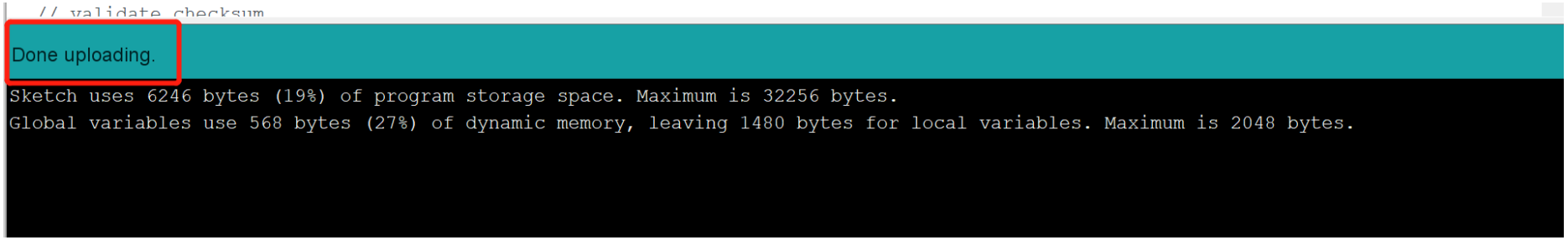


Download Program to the MCU

- 使用USB綫與電腦連接
- 連接后此處會新增一個新的串口設備（注意不同電腦顯示的設備名不同，有可能是是COM5，COM6，…）
- 選擇正確的設備后，點擊左上角的  進行燒錄

Download Program to the MCU

- 使用USB綫與電腦連接
- 連接后此處會新增一個新的串口設備（注意不同電腦顯示的設備名不同，有可能是是COM5，COM6，…）
- 選擇正確的設備后，點擊左上角的  進行燒錄
- 當顯示Done Uploading的時候則表示燒錄完成



Play with the sensor

- 此時熒幕上會顯示當前濃度數據
- 可以嘗試用能夠產生烟霧的工具測試傳感器對顆粒物的反應

Calibration

- 一般而言，校準過程通常會使用到標準設備，通過將標準設備數據與傳感器數據比較，修正誤差，實現校準
- 標準設備最簡單地可以通過將傳感器與一些路邊監測站放置在一起，對兩者數據進行比較
- 校準後的傳感器同樣地可以用來當作標準設備，給其他機器提供校準。
- 激光傳感器精確度的一些額外問題：
  - 利用激光散射方法測量顆粒物濃度之精度有可能會收到濕度等環境因素影響