

Training Workshop on

Arduino-based Automatic Weather Station (AWS)

Hong Kong Observatory

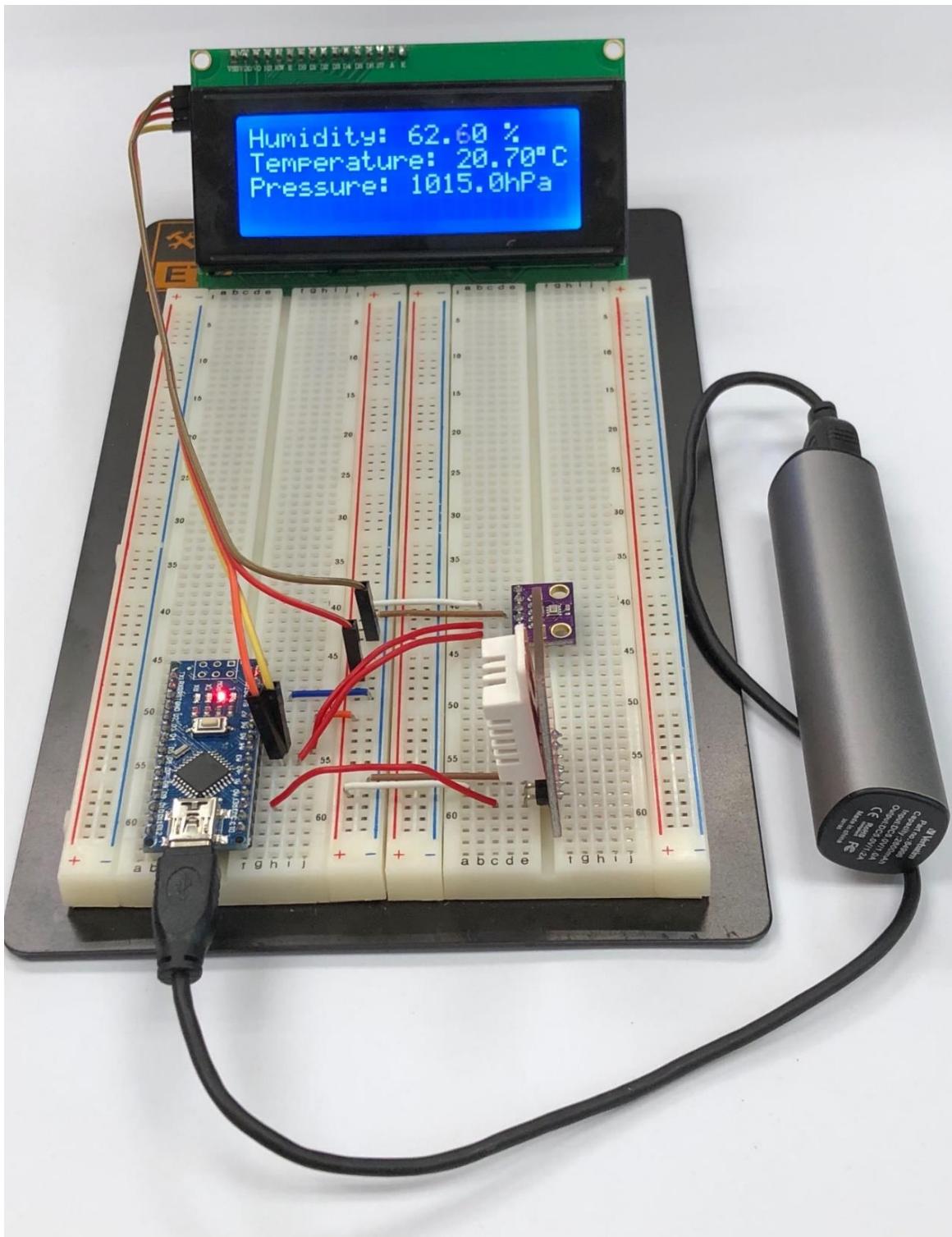


Table of Content

- 1. Project Objectives**
- 2. Getting started**
- 3. Creating your first AWS project**
- 4. Installing a temperature and humidity sensor DHT22 module**
- 5. Installing a Pressure sensor BMP280 module**
- 6. Installing a Real-Time Clock module**
- 7. Installing a LCD Display**

(1) Project Objectives

The development of Automatic Weather Stations (AWS) Network was becoming more universal in recent decade, prior to setting up AWS by commercials and individual uses, AWS mainly served for meteorological monitoring application by governmental institutes.

Starting from 2007, with the establishment of The Community Weather Information Network (Co-WIN) in Hong Kong, more than 150 educational institutions and community organizations had joined to install AWS in their own places for promoting education on meteorology and environmental science through the Co-WIN platform. With the genuine support of our Co-WIN partners, our sensor networks were widely spread across our entire Hong Kong, and quality assurance standard-formatted meteorological data have been received in enriching our database.

With the Co-WIN sensor network, Hong Kong Observatory and partner institutes could access the database in performing some micro-climate research and investigating the climatology in a designated area, which could help in having a better understanding on our living environment.

Co-WIN 2.0, 10 years after the establishment of the Co-WIN network, a phase 2 public education was commenced in promoting meteorology and climate change to public citizens with more multi-disciplinary interactions and public engagement, with the promotion of STEM education by HKSAR government in 2016, we would like to implicate science, technology, engineering and mathematics into a DIY Weather Station project in helping students and public parties involve in the designing and assembling processes of a AWS and innovate from it.



COMMUNITY WEATHER OBSERVING SCHEME
社區天氣觀測計劃

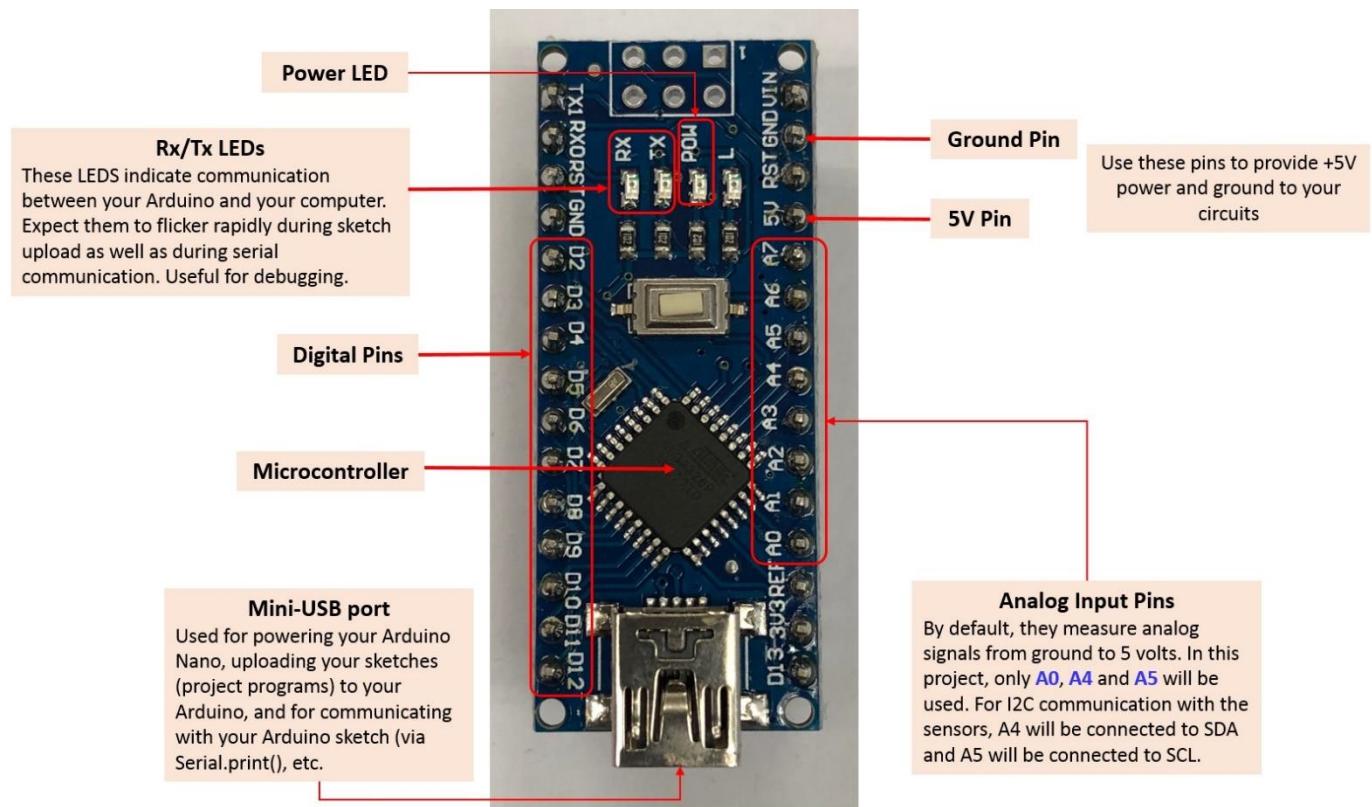


香港中文大學
The Chinese University of Hong Kong
未來城市研究所
Institute of Future Cities

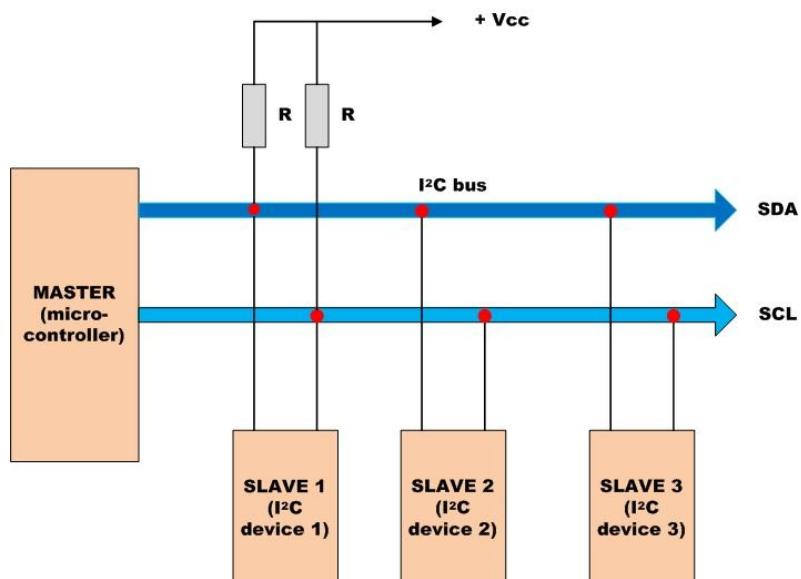
THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學
DEPARTMENT OF APPLIED PHYSICS
應用物理學系

(2) Getting started

2.1 The Arduino Nano Board



I²C Architecture



Multiple devices on common I²C bus

Serial data (SDA)
Serial clock (SCL)

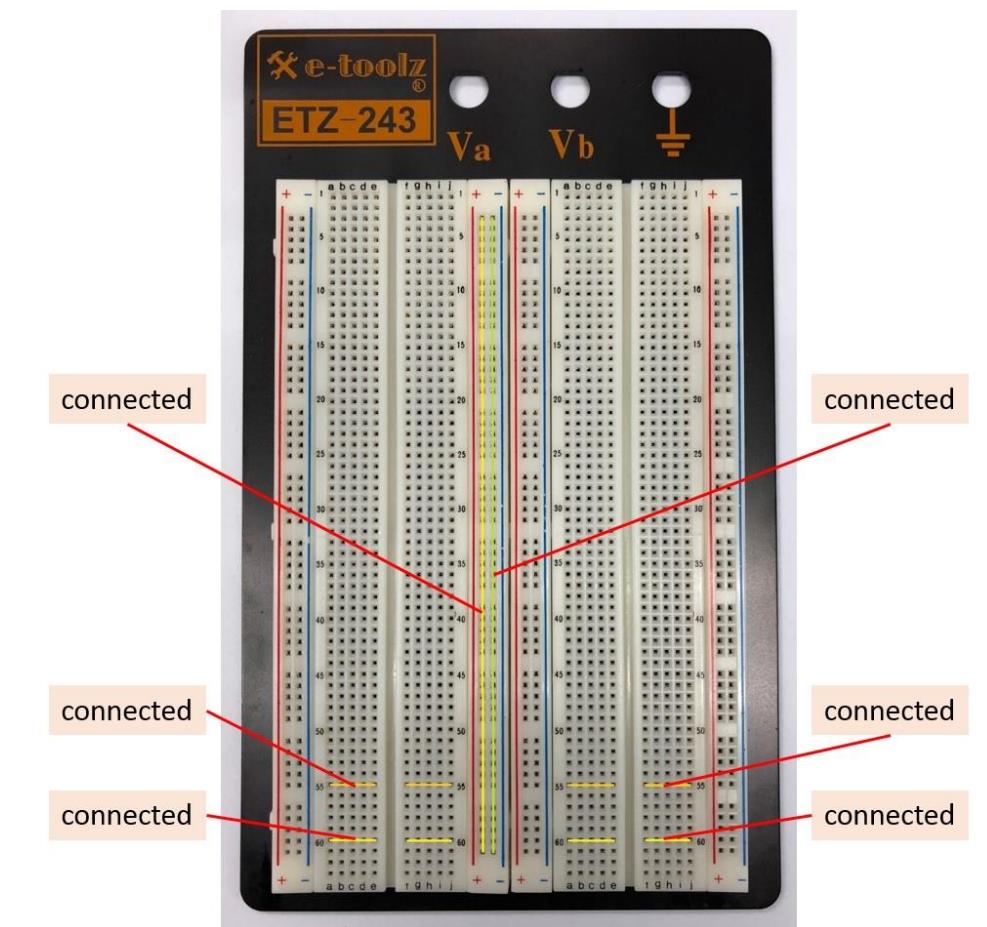
2.2 The Arduino IDE

Before you start controlling the world around you using the Arduino, you'll need to download the Arduino IDE (Integrated Development Environment). The Arduino IDE allows you to write programs and upload them to your Arduino. You can download the latest version of the IDE from: <https://www.arduino.cc/en/Main/Software>

In this workshop, the Arduino IDE has been downloaded and installed for you. The link to start your Arduino IDE is shown as an Arduino icon on the desktop of your PC (see below)

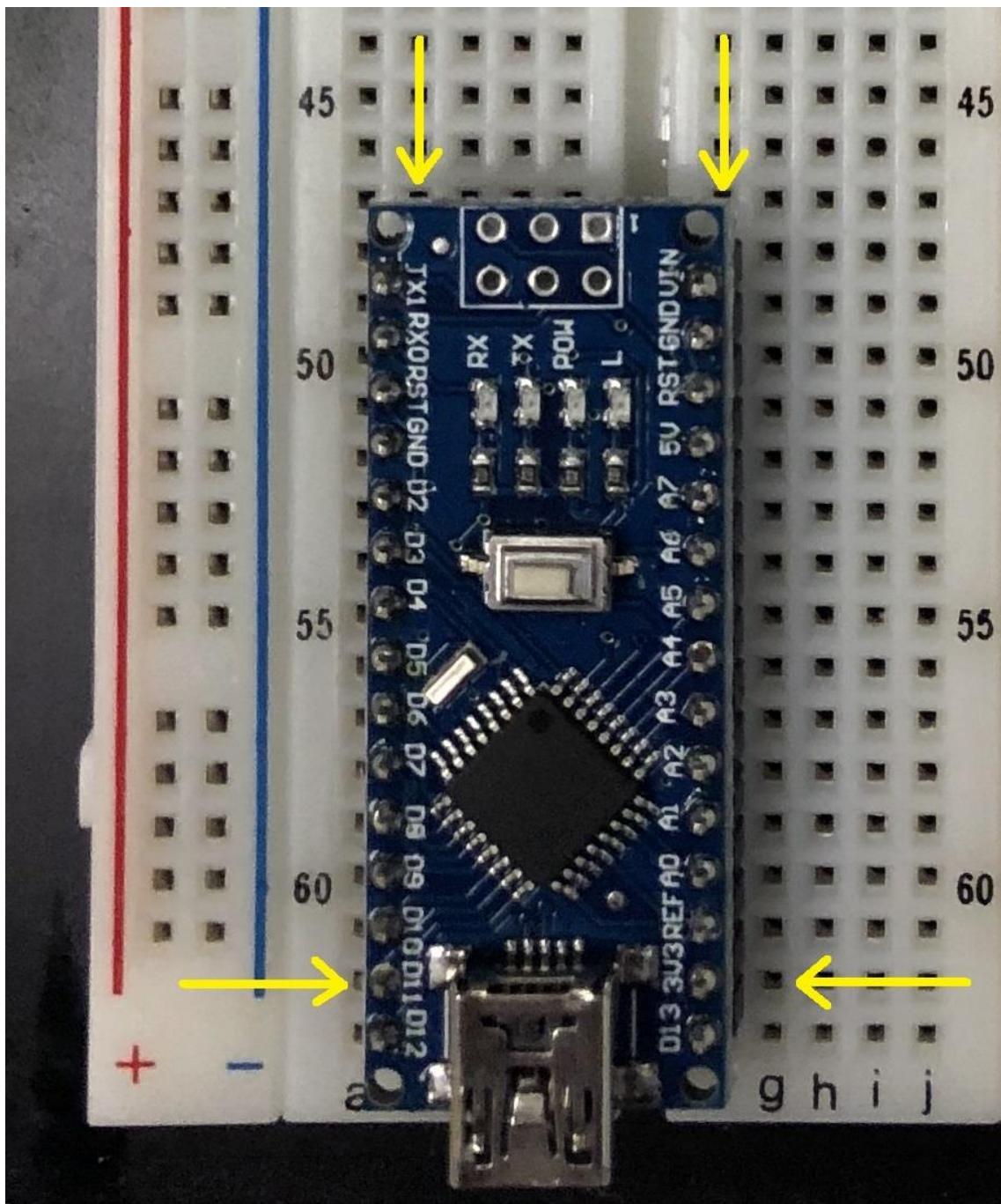


2.3 Basic information about the Breadboard used in this workshop



2.4 Connecting your Arduino Nano to a PC

Insert the Arduino Nano on the breadboard as shown below and connect it to your PC using a mini-USB cable



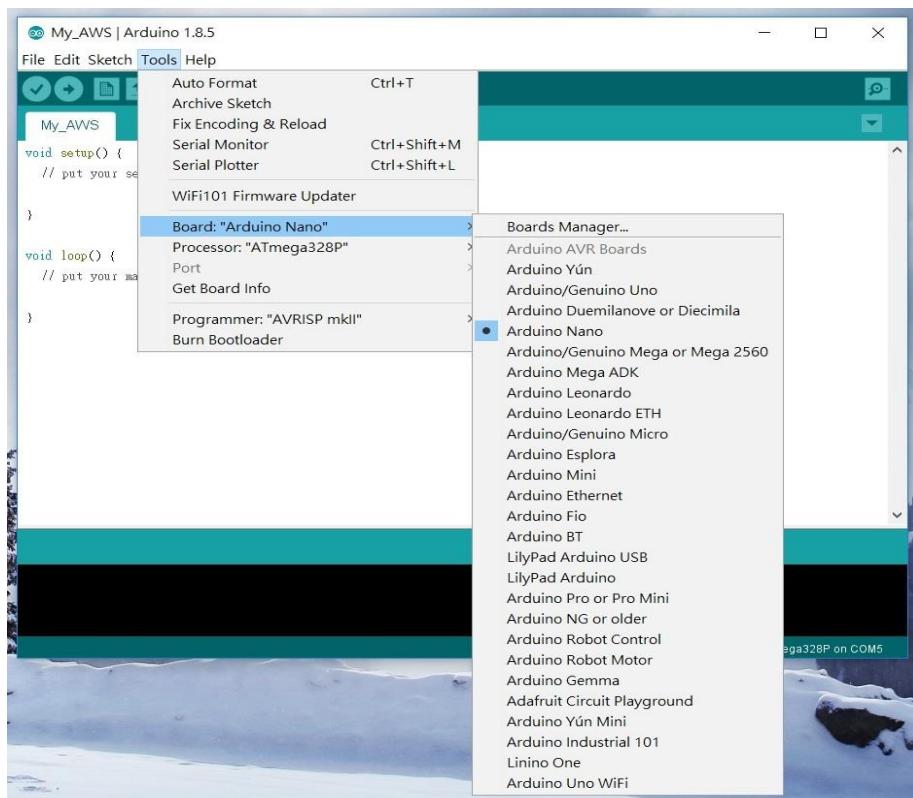
(3) Creating your first AWS project

3.1 Running the Arduino IDE

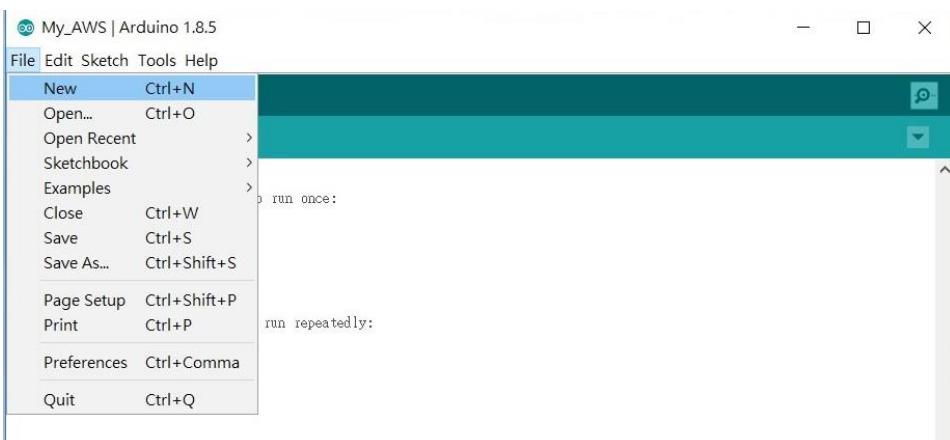
Double click on the Arduino icon on the desktop of your PC to start running the Arduino IDE.



3.2 Selecting a proper Arduino board



3.3 Creating a new project file



The basic structure of a new project file is shown below:

The screenshot shows the Arduino IDE interface with a sketch named "My_AWS". The code editor contains the following code:

```
void setup() {
    // put your setup code here, to run once:
}

void loop() {
    // put your main code here, to run repeatedly:
}
```

A red box highlights the `void setup()` block, and a callout box points to it with the text "Do any initialization steps here!".

Type the following in the `setup()` section:

```
Serial.begin(9600);
Serial.print(" This is my first AWS program! ");
```

The screenshot shows the Arduino IDE interface with the same sketch name "My_AWS". The code editor now includes the following code in the `setup()` section:

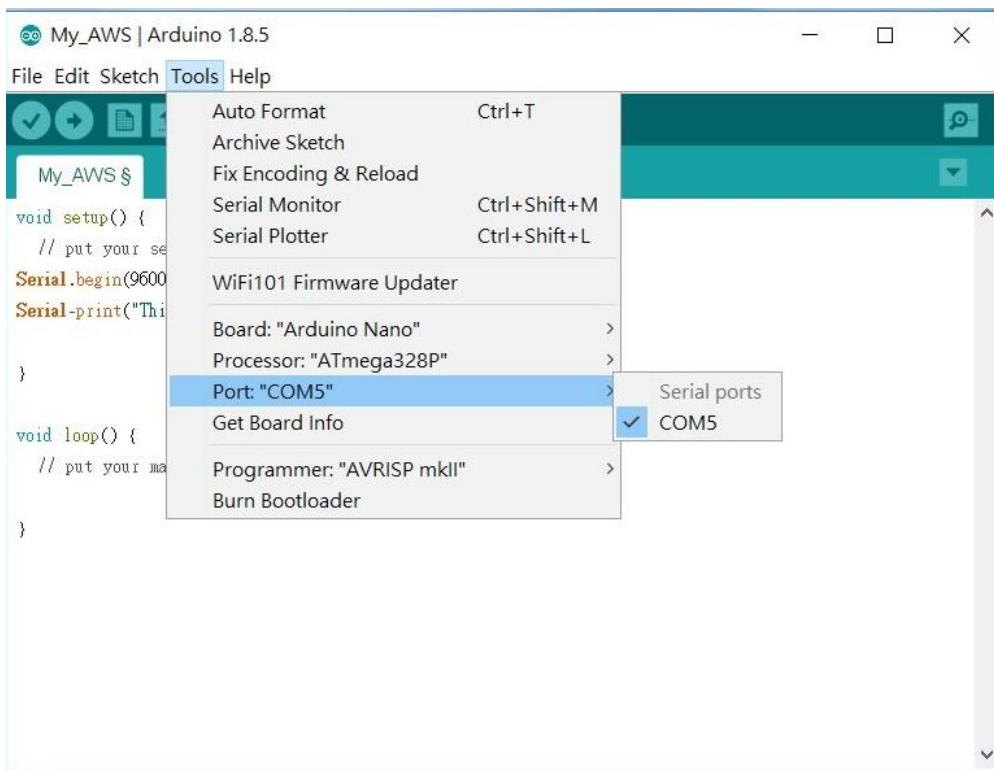
```
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.print(" This is my first AWS program! ");
}
```

Two red boxes highlight the `Serial.begin(9600);` and `Serial.print(" This is my first AWS program! ");` lines. Callout boxes point to them with the following descriptions:

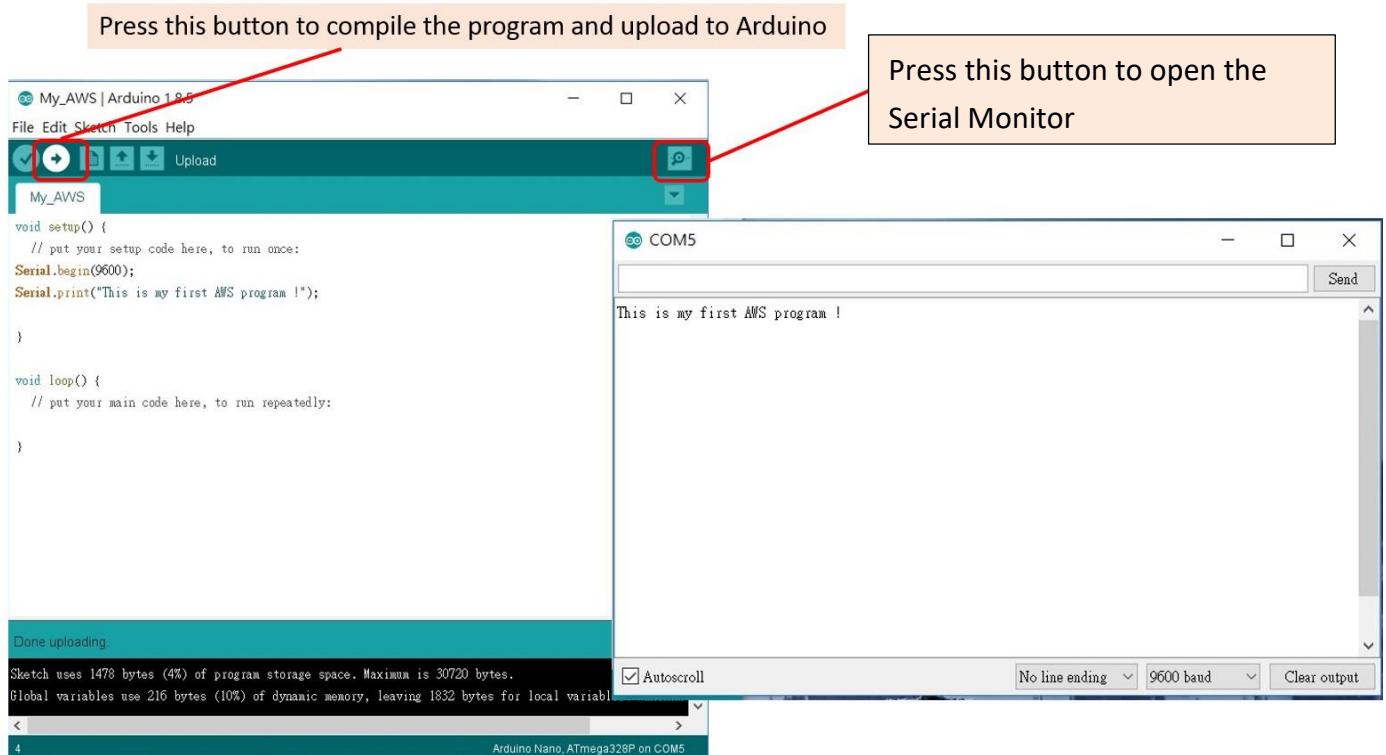
- The top callout box points to `Serial.begin(9600);` and contains the text "Opens up the USB port for serial communication and set the baud rate to 9600 bps".
- The bottom callout box points to `Serial.print(" This is my first AWS program! ");` and contains the text "Prints the value passed into the brackets (“ ”) on to the Serial Monitor".

3.4 Setting up an appropriate Communication Port

Set up an appropriate COM port for displaying the information: select COM5 or COM4 in some PCs.

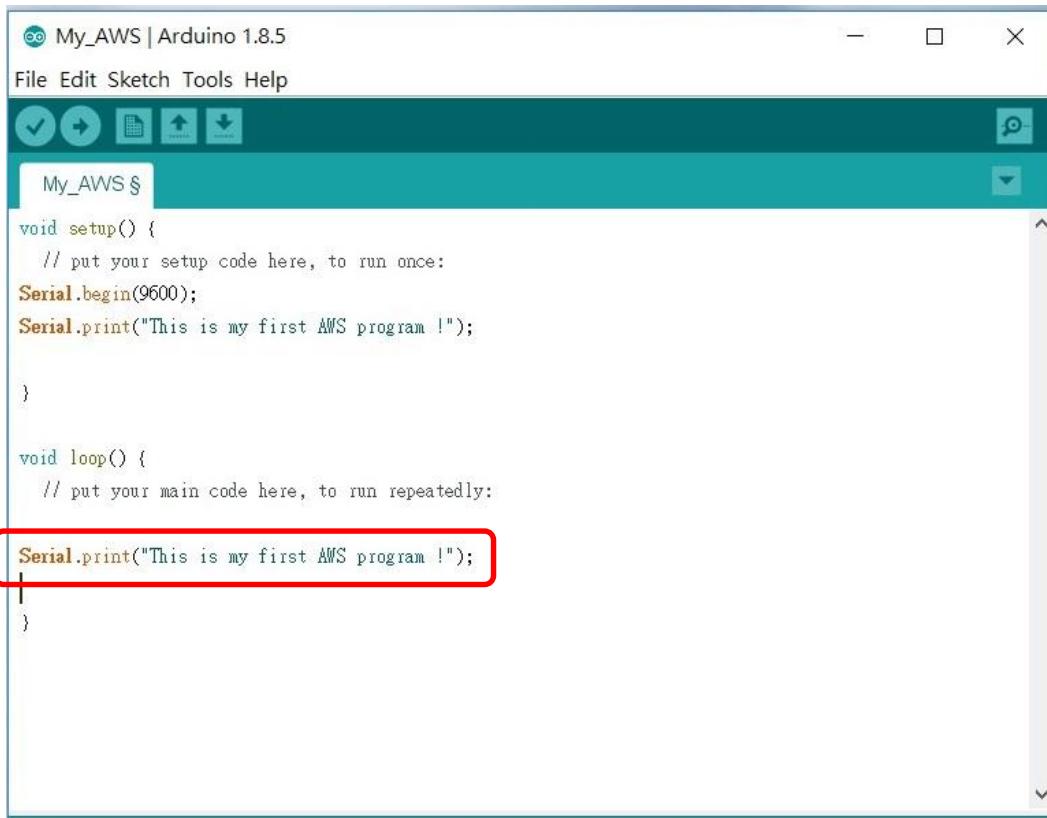


3.5 Compiling and uploading a program to your Arduino



3.6 Using the loop() function

In the loop() section, type `Serial.print("This is my first AWS program! ")` and see the result.



The screenshot shows the Arduino IDE interface with the title bar "My_AWS | Arduino 1.8.5". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, download, and serial monitor. The code editor window contains the following code:

```
My_AWS §

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.print("This is my first AWS program !");

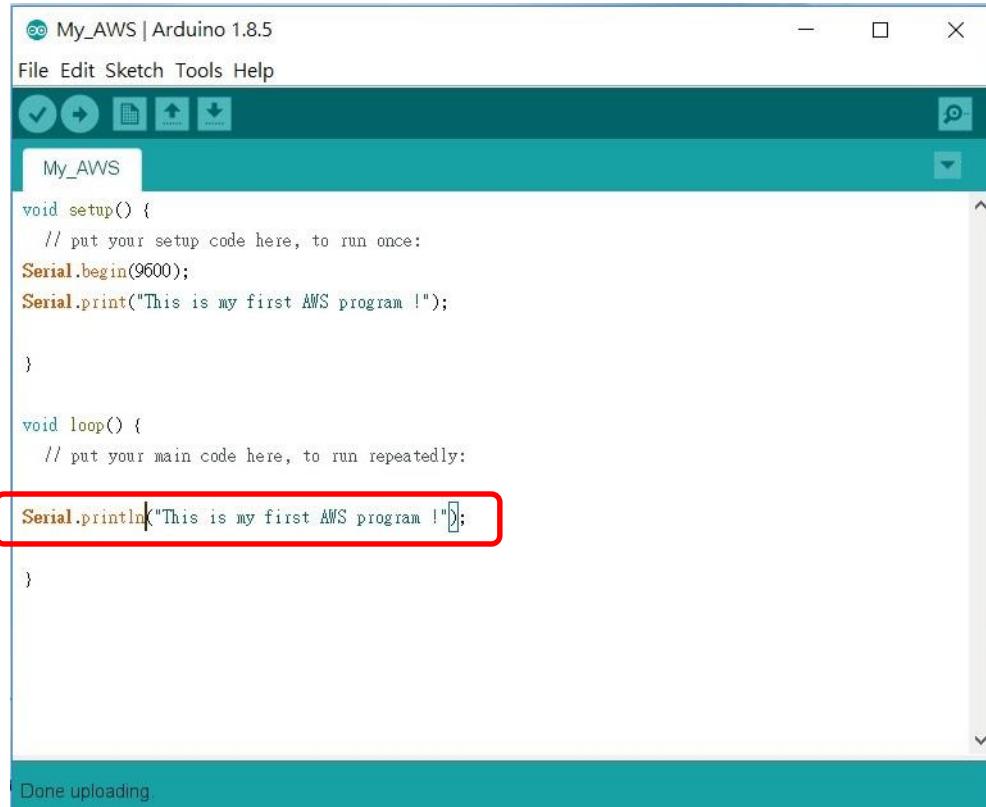
}

void loop() {
    // put your main code here, to run repeatedly:

    Serial.print("This is my first AWS program !");
}
```

The line `Serial.print("This is my first AWS program !");` in the `loop()` function is highlighted with a red rectangle.

Try using `Serial.println("This is my first AWS program! ")` instead of `Serial.print("This is my first AWS program! ")` and recompile your program and see the result.



The screenshot shows the Arduino IDE interface with the title bar "My_AWS | Arduino 1.8.5". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, download, and serial monitor. The code editor window contains the following code:

```
My_AWS

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.print("This is my first AWS program !");

}

void loop() {
    // put your main code here, to run repeatedly:

    Serial.println("This is my first AWS program !");
}
```

The line `Serial.println("This is my first AWS program !");` in the `loop()` function is highlighted with a red rectangle. At the bottom of the screen, a message "Done uploading." is displayed.

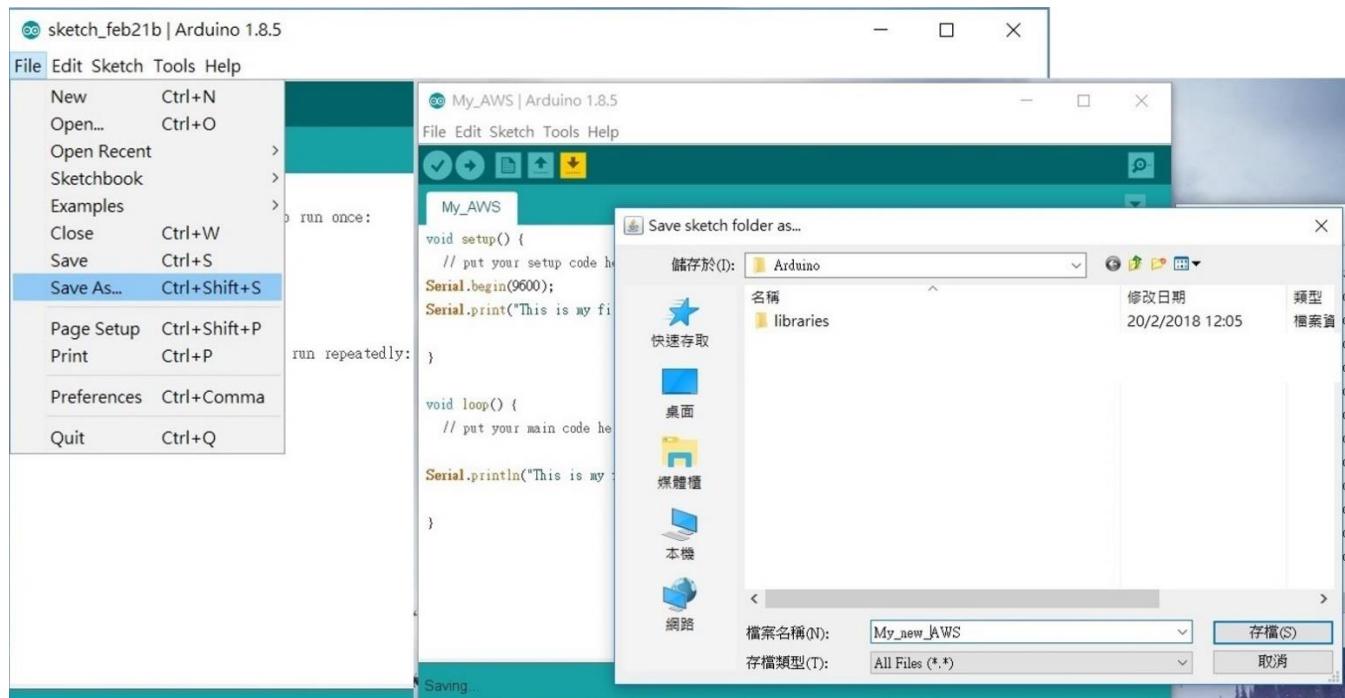
This is my first AWS program !
This is my first AWS program !

Th

Autoscroll No line ending 9600 baud Clear output

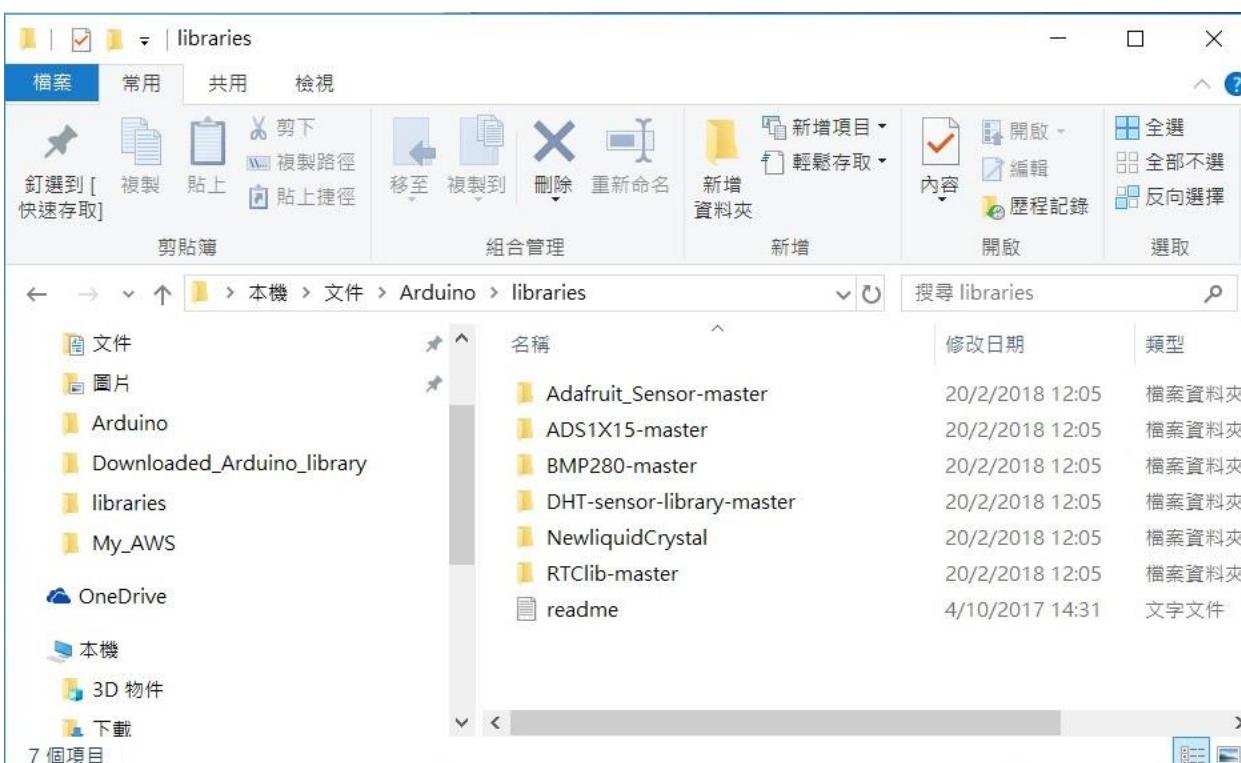
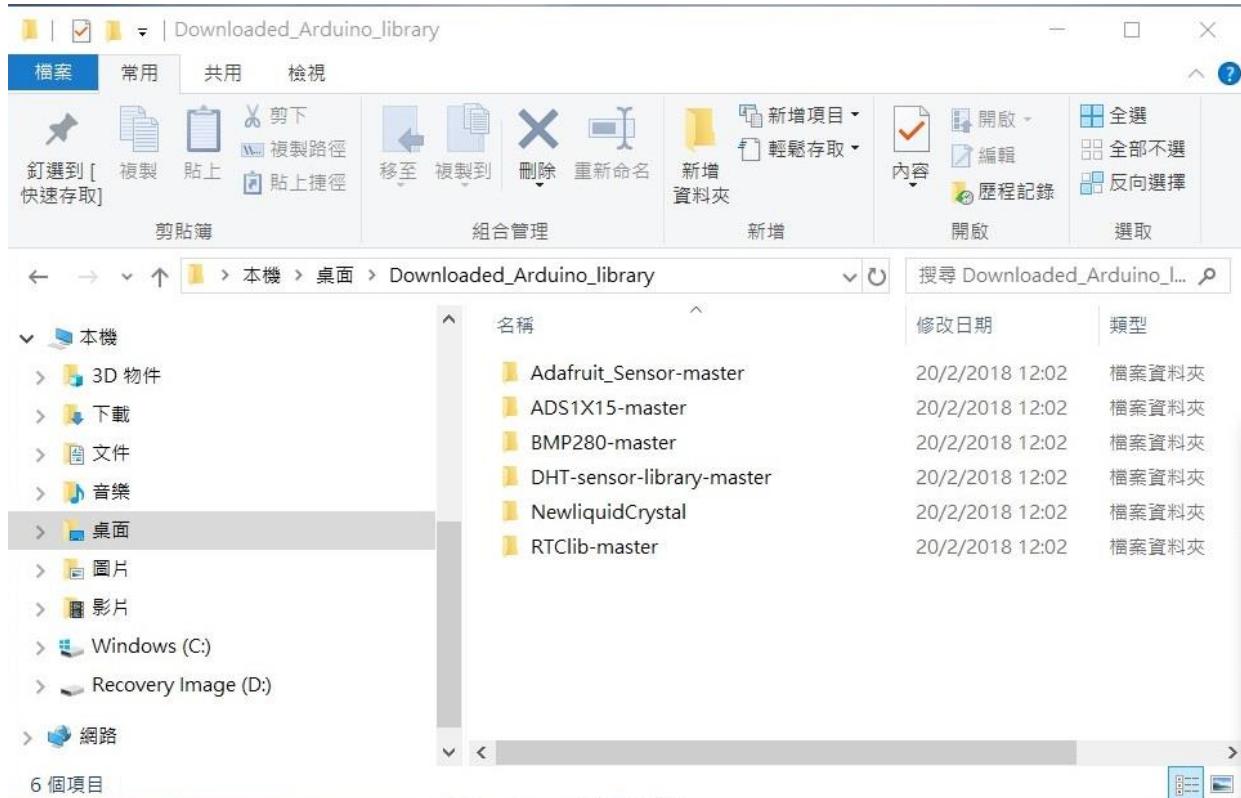
3.7 Saving your first project file

Save the program as My_new_AWS.



3.8 Downloading library files for different sensors for your Arduino projects

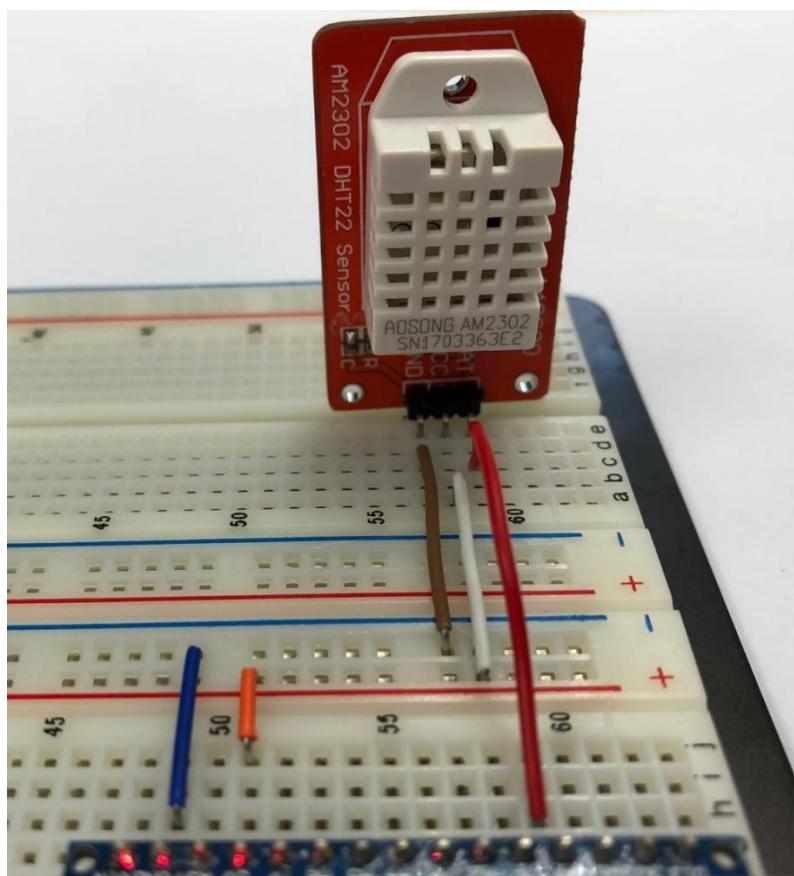
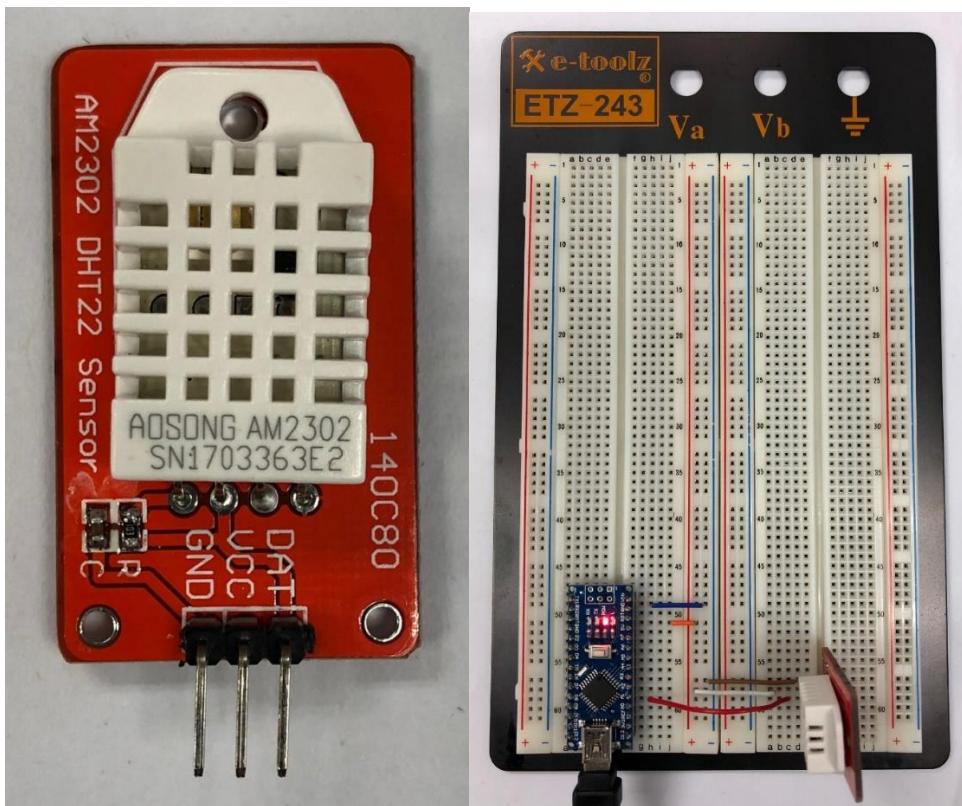
When you purchase different sensors, you need to download the appropriate library files and put them in the proper directory for Arduino IDE to access the files. The following is an example of a list of library files downloaded and stored under the folder 'Downloaded_Arduino_library'. They should be copied to the folder 'document>Arduino>libraries' as shown below:



(4) Installing a temperature and humidity sensor DHT22 module

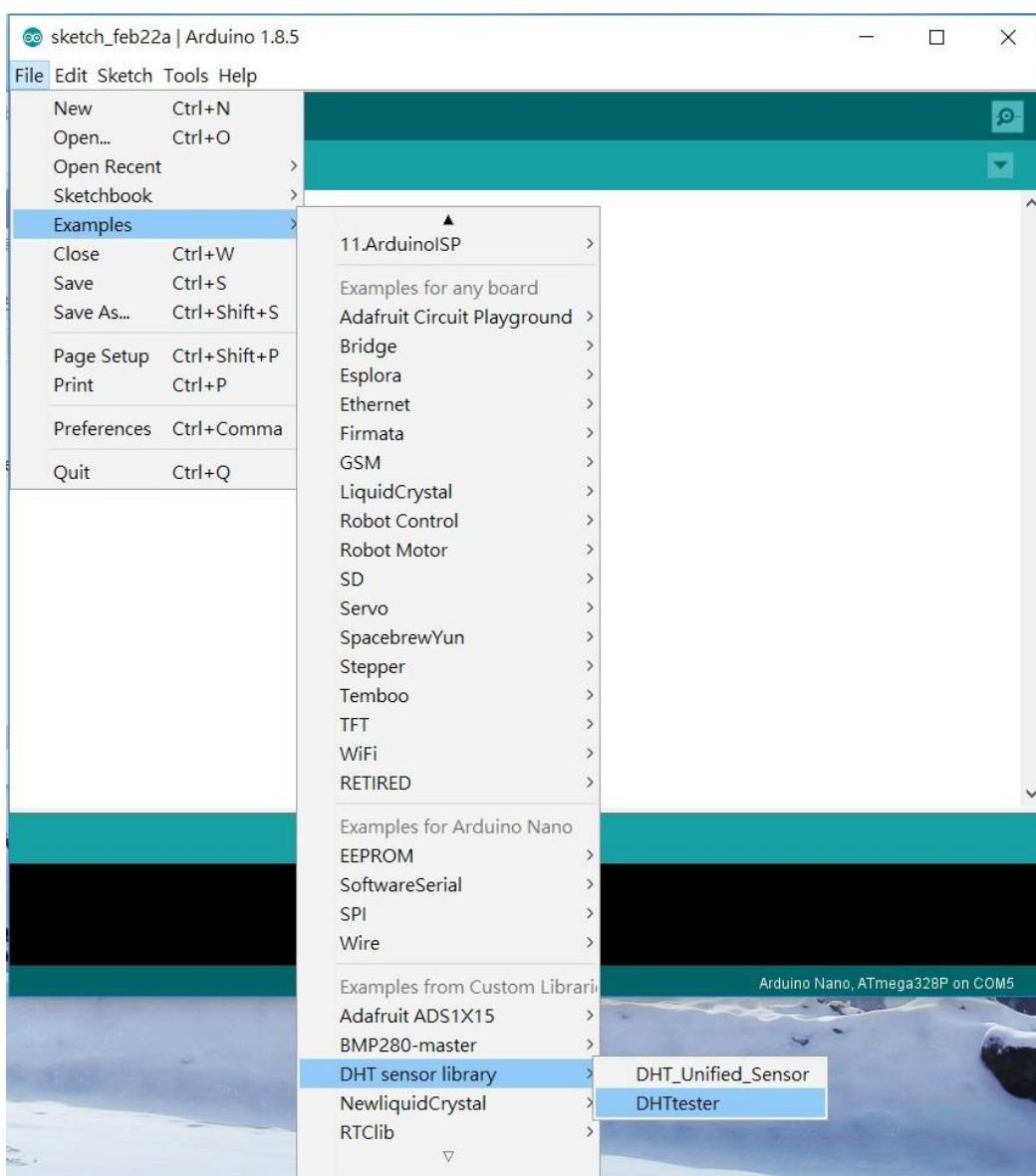
4.1 Wiring

1. Connect GND and VCC of DHT22 to the **GND** and **5V** pins of Arduino Nano respectively
2. Connect DAT of DHT22 to **A0 pin** of Arduino Nano



4.2 Choosing an Example file in the DHT sensor library

Choose the DHTtester file from the DHT sensor library as shown below:



4.3 Modifying the Analog input pin to display correctly

Modify the analog input pin from 2 to A0 as shown below:

```
DHTtester | Arduino 1.8.5
File Edit Sketch Tools Help
DHTtester
// Example testing sketch for various DHT humidity/temperature sensors
// Written by ladyada, public domain

#include "DHT.h"

#define DHTPIN A0      // what digital pin we're connected to

// Uncomment whatever type you're using!
#define DHTTYPE DHT11  // DHT 11
#define DHTTYPE DHT22  // DHT 22 (AM2302), AM2321
#define DHTTYPE DHT21  // DHT 21 (AM2301)

DHTtester | Arduino 1.8.5
File Edit Sketch Tools Help
DHTtester
// Example testing sketch for various DHT humidity/temperature sensors
// Written by ladyada, public domain

#include "DHT.h"

#define DHTPIN 2       // what digital pin we're connected to

// Uncomment whatever type you're using!
#define DHTTYPE DHT11  // DHT 11
#define DHTTYPE DHT22  // DHT 22 (AM2302), AM2321
#define DHTTYPE DHT21  // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor
```

Compile and upload the program again and see the results.

```
ThDHTxx test!
Humidity: 61.70 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.54 *C 70.77 *F
Humidity: 61.40 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.75 *F
Humidity: 61.40 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.75 *F
Humidity: 61.40 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.75 *F
Humidity: 61.40 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.75 *F
Humidity: 61.40 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.75 *F
Humidity: 61.40 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.75 *F
Humidity: 61.40 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.75 *F
Humidity: 61.50 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.76 *F
Humidity: 61.50 %      Temperature: 21.70 *C 71.06 *F Heat index: 21.53 *C 70.76 *F
```

4.4 Modifying the My_new_AWS file to include the DHT22 sensor program

Open the My_new_AWS file side by side with the DHTtester file. Copy the following 4 lines to header section of My_new_AWS program:

```
#include "DHT.h"
#define DHTPIN A0      // what digital pin we're connected to
#define DHTTYPE DHT22  // DHT 22 (AM2302), AM2321
DHT dht(DHTPIN, DHTTYPE);
```

```

My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
#include <Wire.h>
#include <DHT.h>

#define DHTPIN A0
#define DHTTYPE DHT22
DHT dht(DHTPIN,DHTTYPE);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("This is my first AWS program!");

  dht.begin();
}

void loop() {
  // put your main code here, to run repeatedly:

  //get and print humidity data
  float h = dht.readHumidity();
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("%t");

  //get and print temperature data
  float t = dht.readTemperature();
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print("C %t");

  delay(1000-millis()%1000 + 10);
}

```



```

DHTTester | Arduino 1.8.5
File Edit Sketch Tools Help
DHTTester §
// Example testing sketch for various DHT humidity/temperature sensors
// Written by ladyada, public domain

#include "DHT.h"

#define DHTPIN A0          // what digital pin we're connected to

// Comment whatever type you're using!
//#define DHTTYPE DHT11    // DHT 11
#define DHTTYPE DHT22      // DHT 22 (AM2302, AM2321)
//#define DHTTYPE DHT21     // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);

```

Copy the following line to the setup section of My_new_AWS program:
dht.begin(); as shown below:

```

My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
#include <Wire.h>
#include <DHT.h>

#define DHTPIN A0
#define DHTTYPE DHT22
DHT dht(DHTPIN,DHTTYPE);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("This is my first AWS program!");

  dht.begin(); // Red arrow points here

}

void loop() {
  // put your main code here, to run repeatedly:

  //get and print humidity data
  float h = dht.readHumidity();
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("%t");

  //get and print temperature data
  float t = dht.readTemperature();
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print("C %t");

  delay(1000-millis()%1000 + 10);
}

```



```

DHTTester | Arduino 1.8.5
File Edit Sketch Tools Help
DHTTester §
//#define DHTTYPE DHT21    // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");

  dht.begin(); // Red box highlights this line

}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);
}

```

Copy the loop() section of DHTtester to My_new_AWS and modify them as shown below:

```

My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
#include <Wire.h>
#include <DHT.h>

#define DHTPIN A0
#define DHTTYPE DHT22
DHT dht(DHTPIN,DHTTYPE);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("This is my first AWS program!");

  dht.begin();
}

void loop() {
  // put your main code here, to run repeatedly:

  //get and print humidity data
  float h = dht.readHumidity();
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("%\t");

  //get and print temperature data
  float t = dht.readTemperature();
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print("C\t");

  delay(1000-millis()%1000 + 10);
}

```

```

DHTtester | Arduino 1.8.5
File Edit Sketch Tools Help
DHTtester §
void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahrenheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" C\t");
  Serial.print(f);
  Serial.print("\t");
}

```

Add a time delay for ensuring printing a data record every second.

delay(1000-millis()%1000 + 10);

```

My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
#include <Wire.h>
#include <DHT.h>

#define DHTPIN A0
#define DHTTYPE DHT22
DHT dht(DHTPIN,DHTTYPE);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("This is my first AWS program!");

  dht.begin();
}

void loop() {
  // put your main code here, to run repeatedly:

  //get and print humidity data
  float h = dht.readHumidity();
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("%\t");

  //get and print temperature data
  float t = dht.readTemperature();
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print("C\t");

  delay(1000-millis()%1000 + 10);
}

```

My_new_AWS | Arduino 1.8.5

File Edit Sketch Tools Help

Compile the program and see the results

```

My_new_AWS §

#include <Wire.h>
#include <DHT.h>

#define DHTPIN A0
#define DHTTYPE DHT22
DHT dht(DHTPIN,DHTTYPE);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("This is my first AWS program !");
  dht.begin();
}

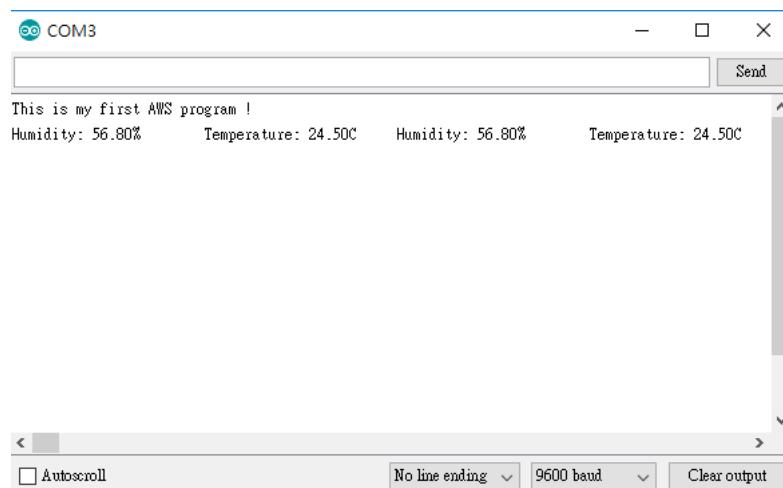
void loop() {
  // put your main code here, to run repeatedly:

  //get and print humidity data
  float h = dht.readHumidity();
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("%\t");

  //get and print temperature data
  float t = dht.readTemperature();
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print("C\t");

  delay(1000-millis()%1000 + 10);
}

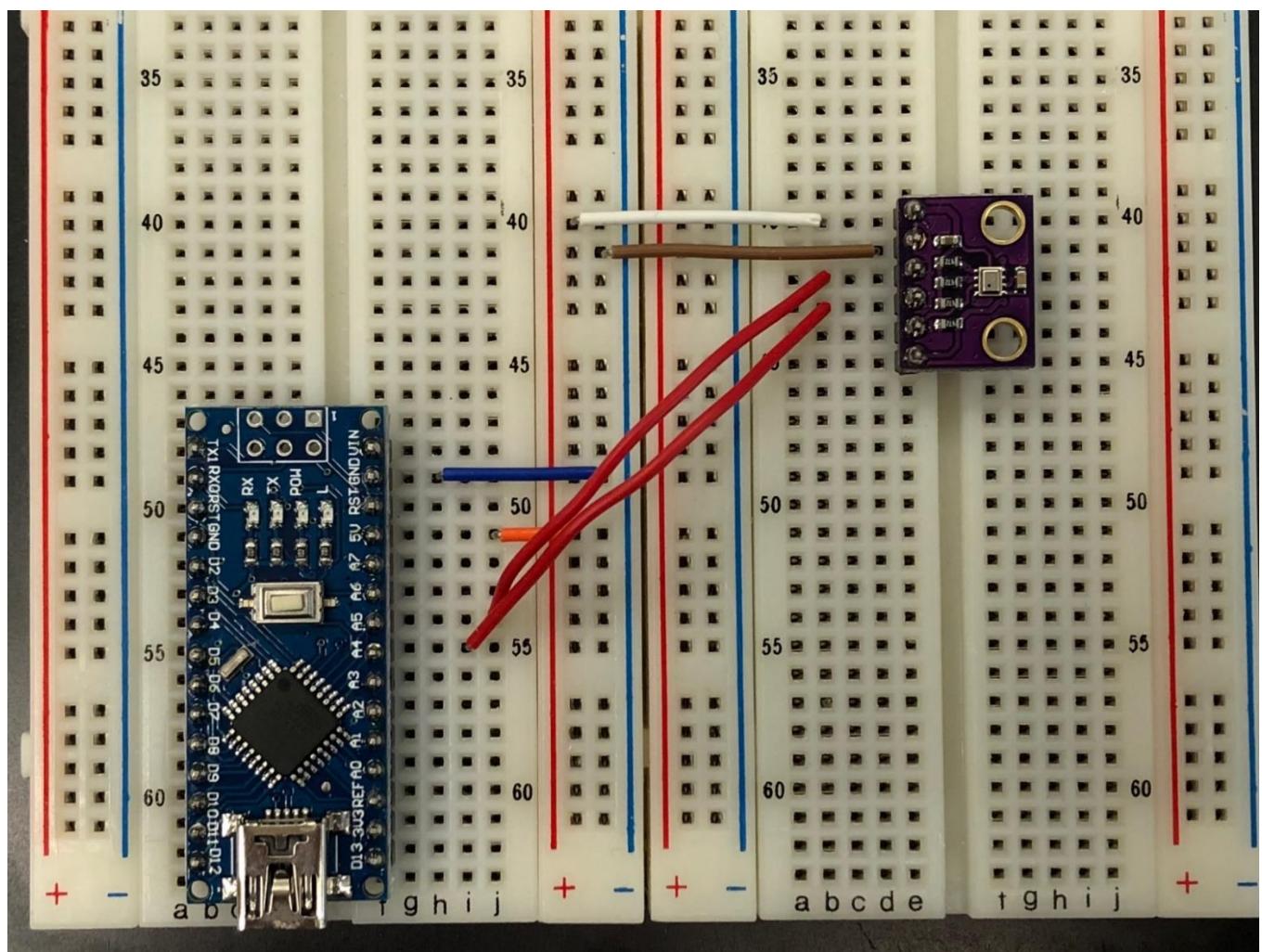
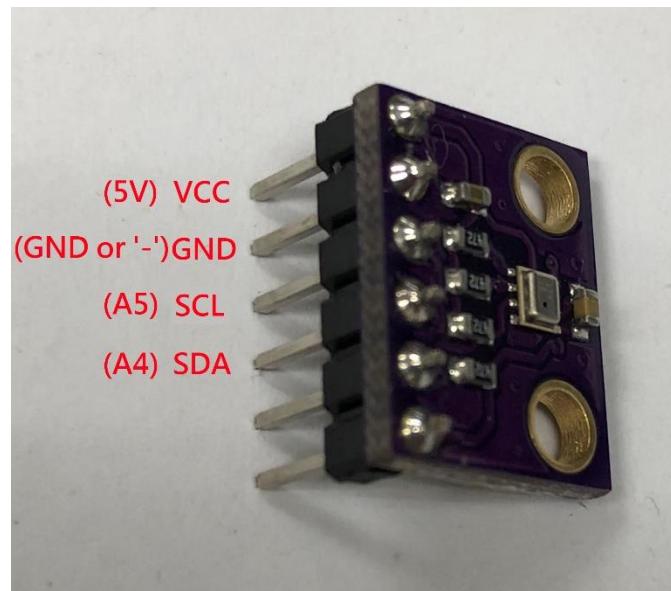
```



(5) Installing a Pressure sensor BMP280 module

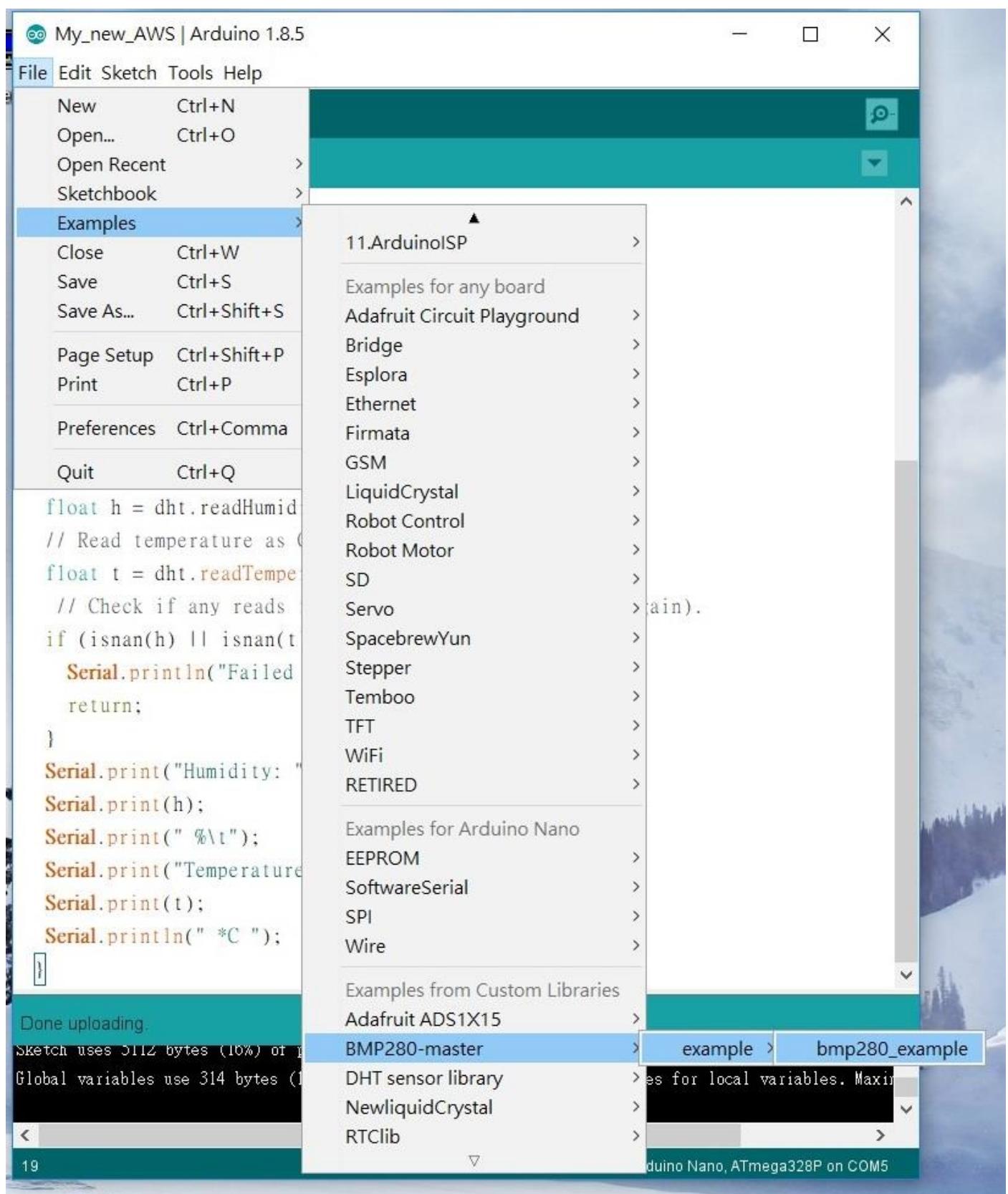
5.1 Wiring

1. Connect BMP280 Vcc to 5V and GND to ground respectively
2. Connect BMP280 SCL to Adruino A5
3. Connect BMP280 SDA to Adruino A4

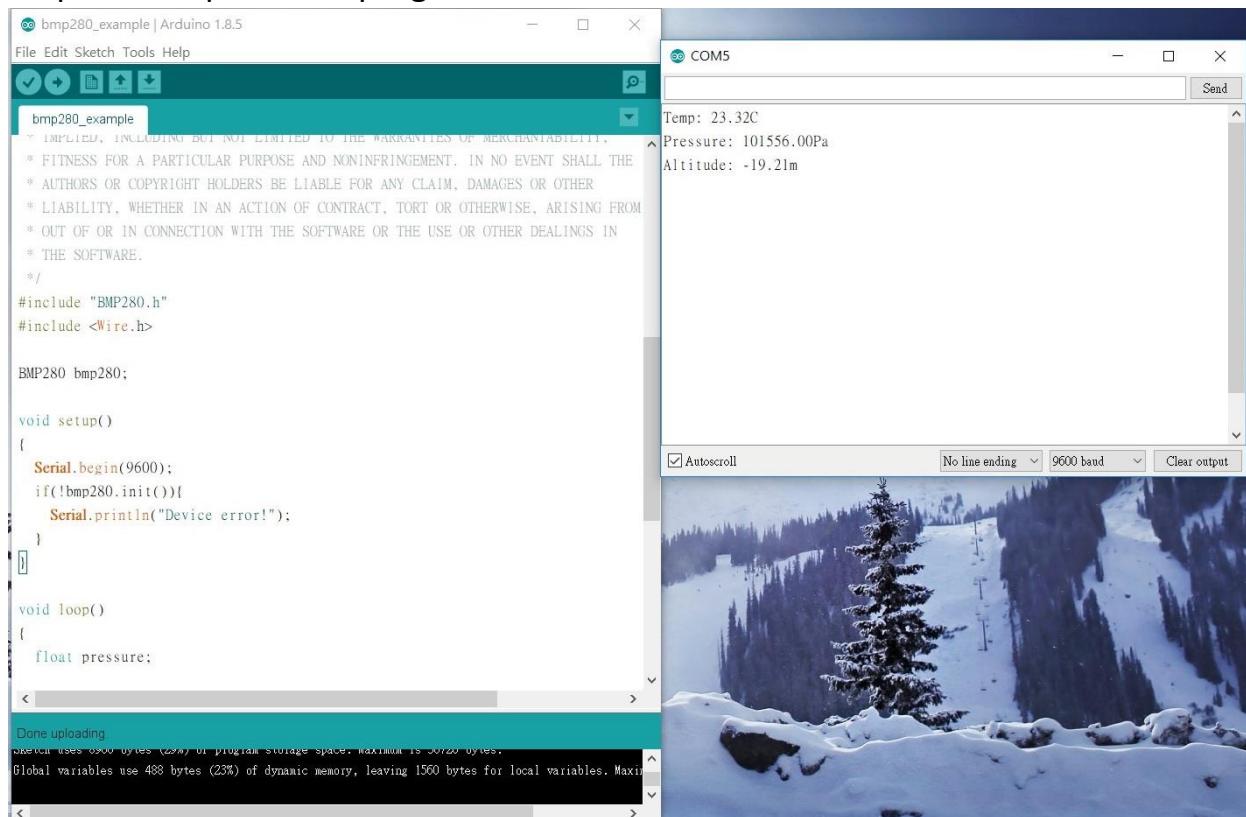


5.2 Choosing an Example file in BMP280 library

Choose the bmp280_example file from the BMP280-master library as shown below:

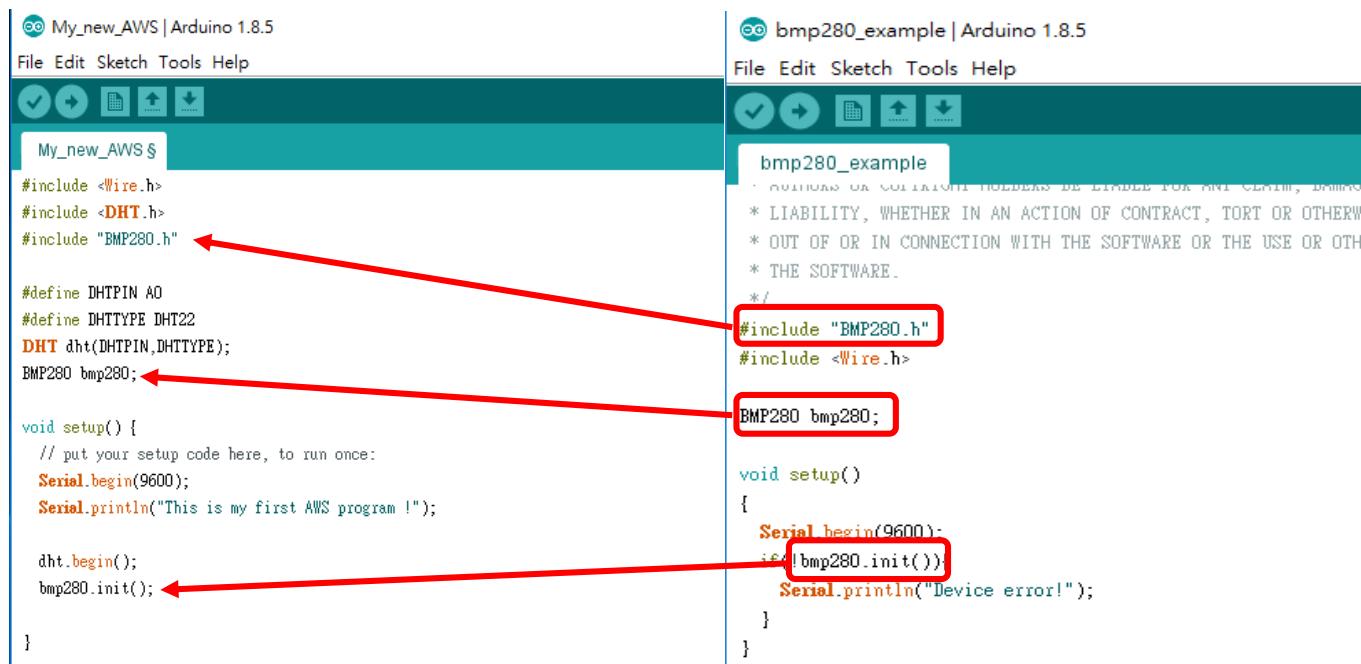


Compile and upload the program to Arduino Nano and see the results.



5.3 Modifying My_new_AWS file to include the BMP280 sensor program

Open My_new_AWS file and copy the below lines from bmp280_example file to the file. In `setup()`, add `bmp.init();` as shown.



```

void loop() {
    // put your main code here, to run repeatedly:

    //get and print humidity data
    float h = dht.readHumidity();
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.println("%\t");

    //get and print temperature data
    float t = dht.readTemperature();
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println("C      ");

    //get and print atmospheric pressure data
    float p = bmp280.getPressure()/100;
    Serial.print("Pressure: ");
    Serial.print(p);
    Serial.println("hPa");
    delay(1000-millis()%1000 + 10);

}

void loop()
{
    float pressure;

    //get and print temperatures
    Serial.print("Temp: ");
    Serial.print(bmp280.getTemperature());
    Serial.println("C"); // The unit for Celsius because original arduino don't support speical symbols

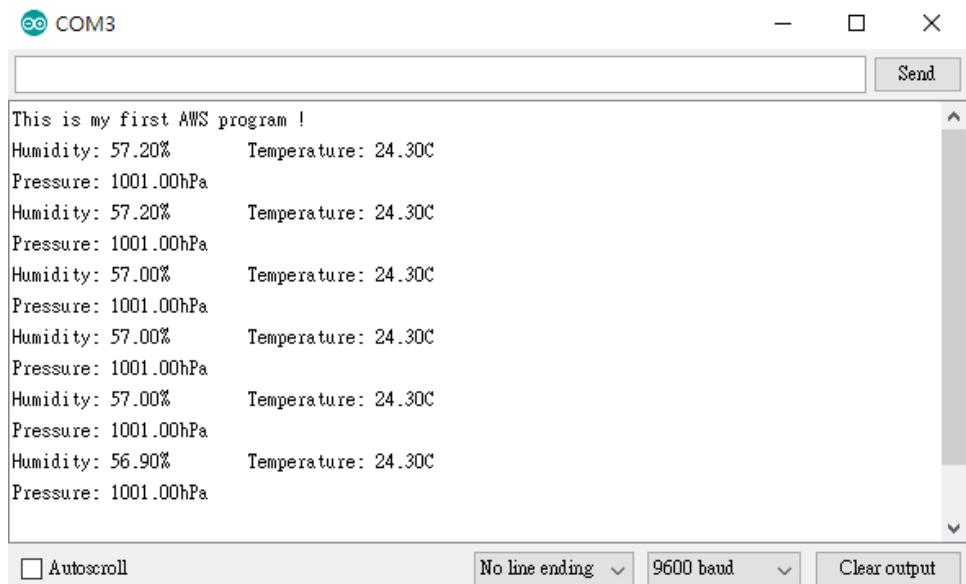
    //get and print atmospheric pressure data
    Serial.print("Pressure: ");
    Serial.print(pressure = bmp280.getPressure());
    Serial.println("Pa");

    //get and print altitude data
    Serial.print("Altitude: ");
    Serial.print(bmp280.calcAltitude(pressure));
    Serial.println("m");

    Serial.println("\n");//add a line between output of different times.
}

```

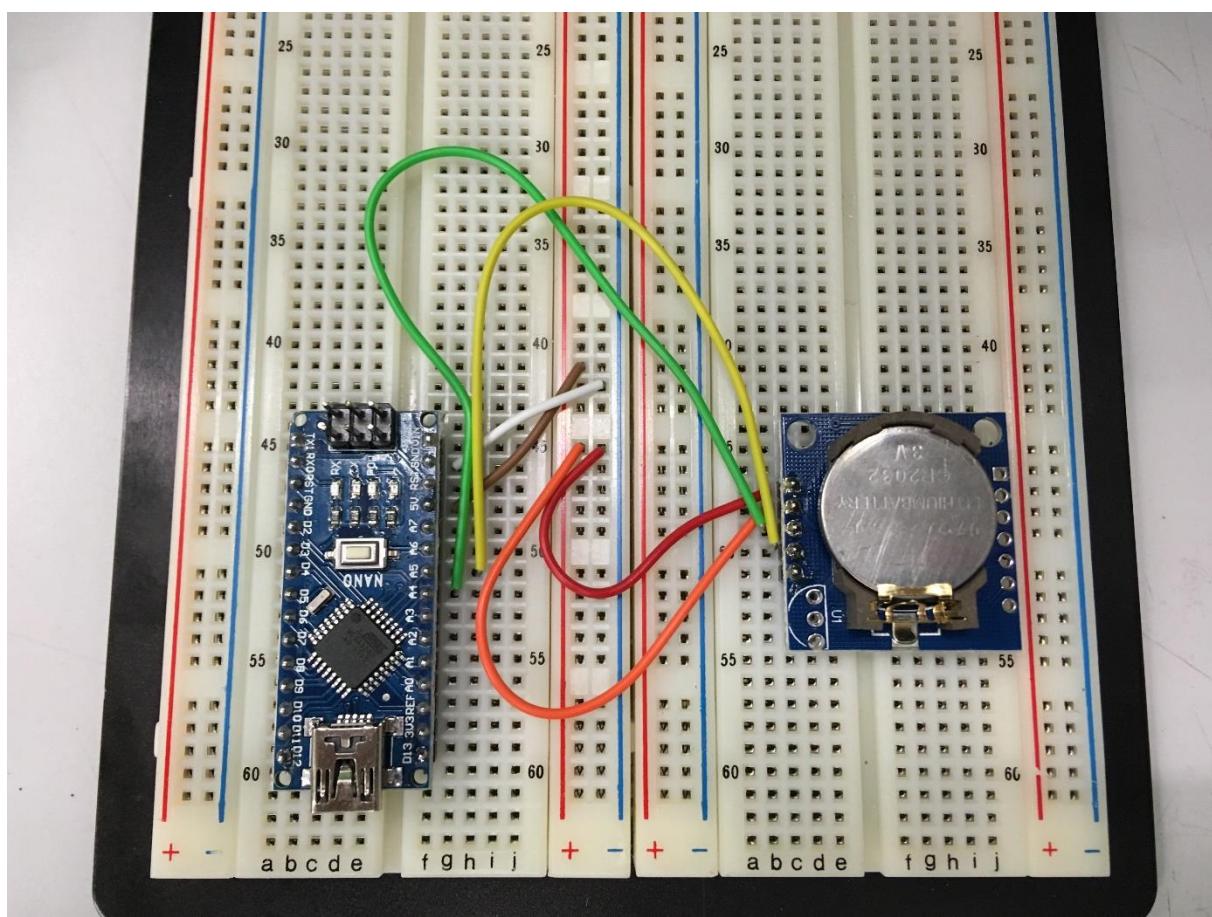
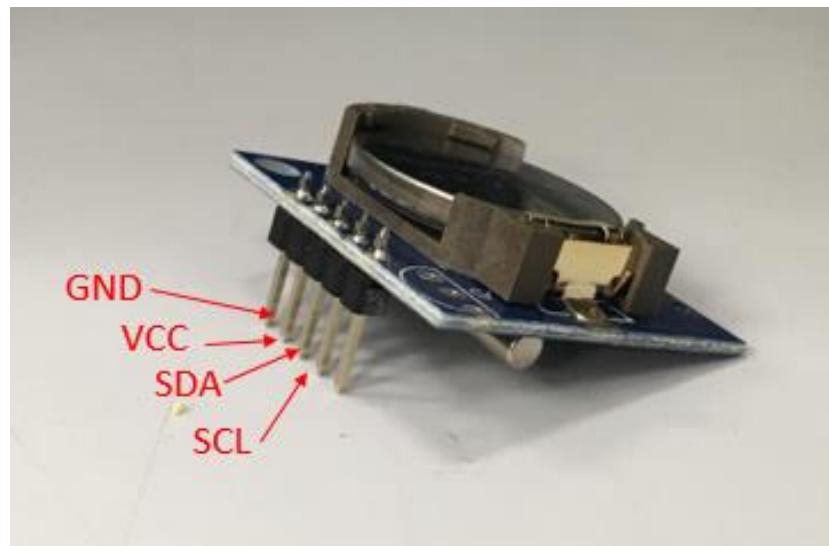
Compile and upload the program to see the results! Remember to save your My_new_AWS file again!



(6) Installing a Real-Time Clock module

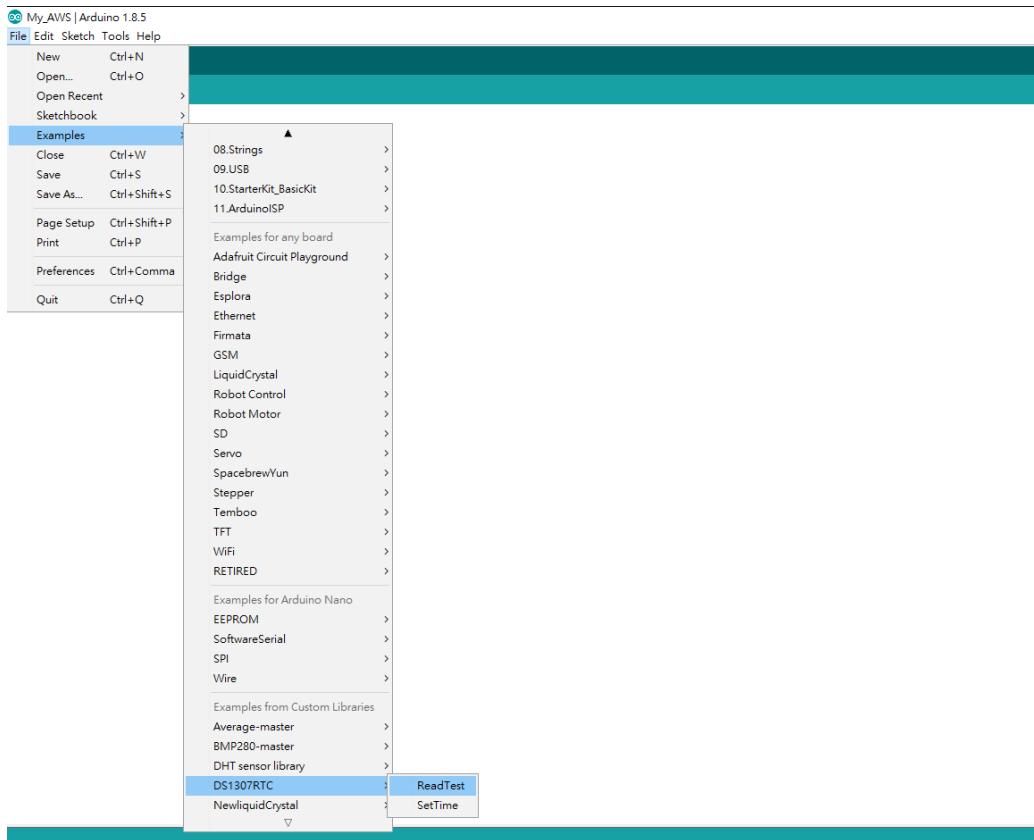
6.1 Wiring

1. Connect DS1307 Vcc to 5V and GND to ground respectively
2. Connect DS1307 SCL to Adruino A5
3. Connect DS1307 SDA to Adruino A4



6.2 Choosing an Example file in RTC library

Choose the ReadTest file from the DS1307RTC library as shown below:



Compile and upload the program to Arduino Nano and see the results.

A screenshot of the Arduino IDE showing the "ReadTest" sketch. The code includes #include <Wire.h>, #include <i2cLib.h>, and #include <DS1307RTC.h>. The setup() function initializes the serial port at 9600 bps and prints the start message. The loop() function reads the RTC time and date, prints them to the serial monitor, and handles errors if the RTC is stopped. The serial monitor window shows the output of the "DS1307 RTC Read Test" followed by a series of "Ok, Time = 23:02:18, Date (D/M/Y) = 26/6/2018" messages repeated 10 times. At the bottom right of the monitor window, there is a checked checkbox labeled "Autoscroll".

```
#include <Wire.h>
#include <i2cLib.h>
#include <DS1307RTC.h>

void setup() {
  Serial.begin(9600);
  while (!Serial); // wait for serial
  delay(200);
  Serial.println("DS1307RTC Read Test");
  Serial.println("-----");
}

void loop() {
  tmElements_t tm;

  if (RTC.read(tm)) {
    Serial.print("Ok, Time = ");
    print2Digits(tm.Hour);
    Serial.write(':');
    print2Digits(tm.Minute);
    Serial.write(':');
    print2Digits(tm.Second);
    Serial.print(", Date (D/M/Y) = ");
    Serial.print(tm.Day);
    Serial.write('/');
    Serial.print(tm.Month);
    Serial.write('/');
    Serial.print(tmYearToCalendar(tm.Year));
    Serial.println();
  } else {
    if (RTC.chipPresent()) {
      Serial.println("The DS1307 is stopped. Please run the SetTime");
      Serial.println("example to initialize the time and begin running.");
      Serial.println();
    } else {
      Serial.println("DS1307 read error! Please check the circuitry.");
      Serial.println();
    }
    delay(9000);
  }
}
```

6.3 Modifying My_new_AWS file to include the RTC ReadTest program

Open My_new_AWS file and copy the below lines from ReadTest file to the file, minor modification from **print2digits** to **Serial.print** would be needed for simplifying the code.

```

// My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS
#include <Wire.h>
#include <DHT.h>
#include "BMP280.h"
#include <TimeLib.h>
#include <DS1307RTC.h>

#define DHTPIN A0
#define DHTTYPE DHT22
DHT dht(DHTPIN,DHTTYPE);
BMP280 bmp280;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.println("This is my first AWS program !");

    dht.begin();
    bmp280.init();

}

void loop() {
    tmElements_t tm;
    // put your main code here, to run repeatedly:
}

```



```

// ReadTest | Arduino 1.8.5
File Edit Sketch Tools Help
ReadTest
#include <Wire.h>
#include <TimeLib.h>
#include <DS1307RTC.h>

void setup() {
    Serial.begin(9600);
    while (!Serial); // wait for serial
    delay(200);
    Serial.println("DS1307RTC Read Test");
    Serial.println("-----");
}

void loop() {
    tmElements_t tm;

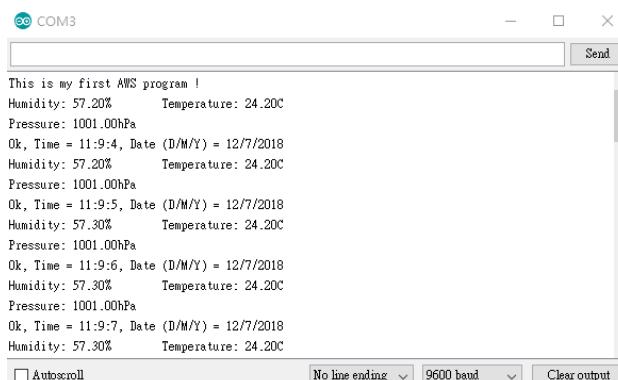
```

```

if (RTC.read(tm)) {
    Serial.print("Ok, Time = ");
    print2digits(tm.Hour);
    Serial.write(':');
    print2digits(tm.Minute);
    Serial.write(':');
    print2digits(tm.Second);
    Serial.print(", Date (D/M/Y) = ");
    Serial.print(tm.Day);
    Serial.write('/');
    Serial.print(tm.Month);
    Serial.write('/');
    Serial.print(tmYearToCalendar(tm.Year));
    Serial.println();
} else {
    if (RTC.chipPresent()) {
        Serial.println("The DS1307 is stopped. Please run the SetTime");
        Serial.println("Example to initialize the time and begin running.");
        Serial.println();
    } else {
        Serial.println("DS1307 read error! Please check the circuitry.");
        Serial.println();
    }
    delay(9000);
}
delay(1000);

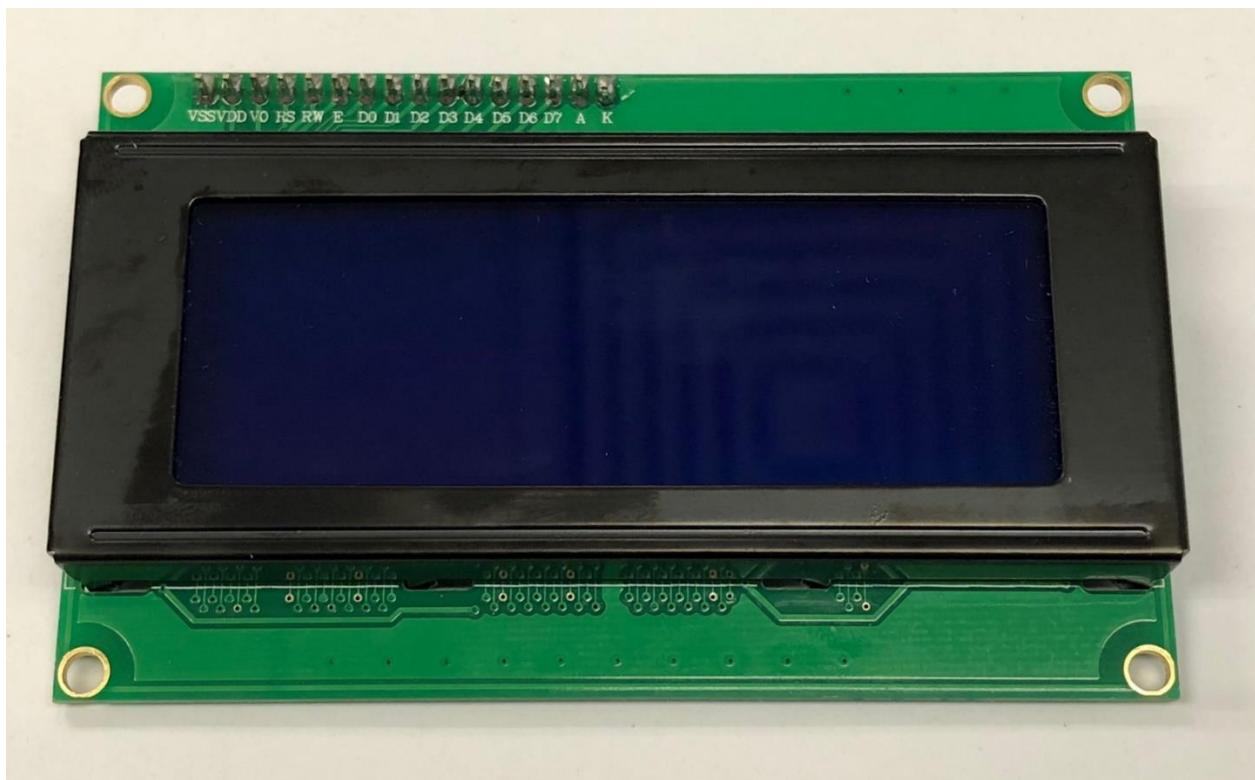
```

Compile and upload the program to see the results! Remember to save your My_new_AWS file again!



(7) Installing a LCD Display

Front view

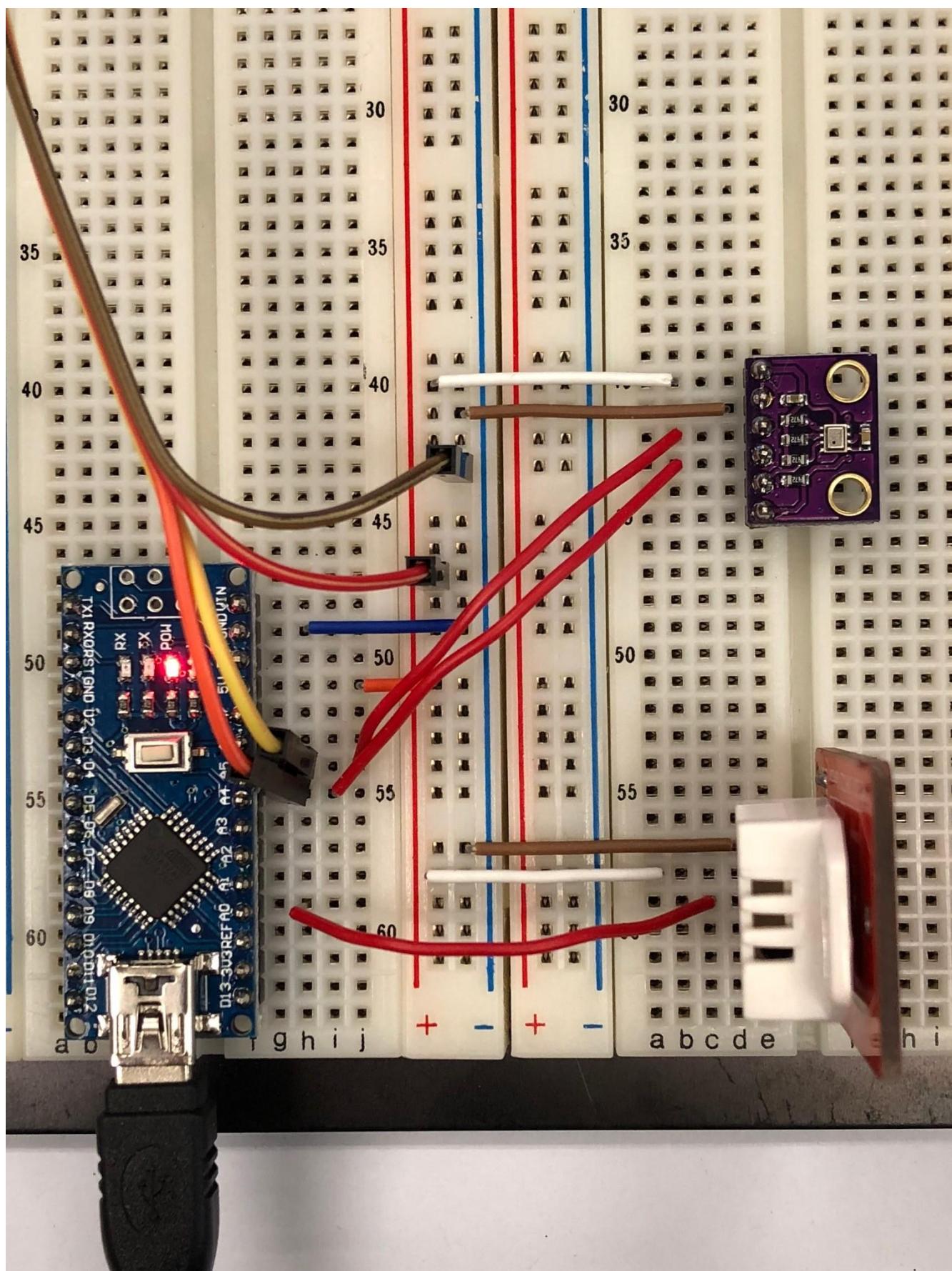


Rear view



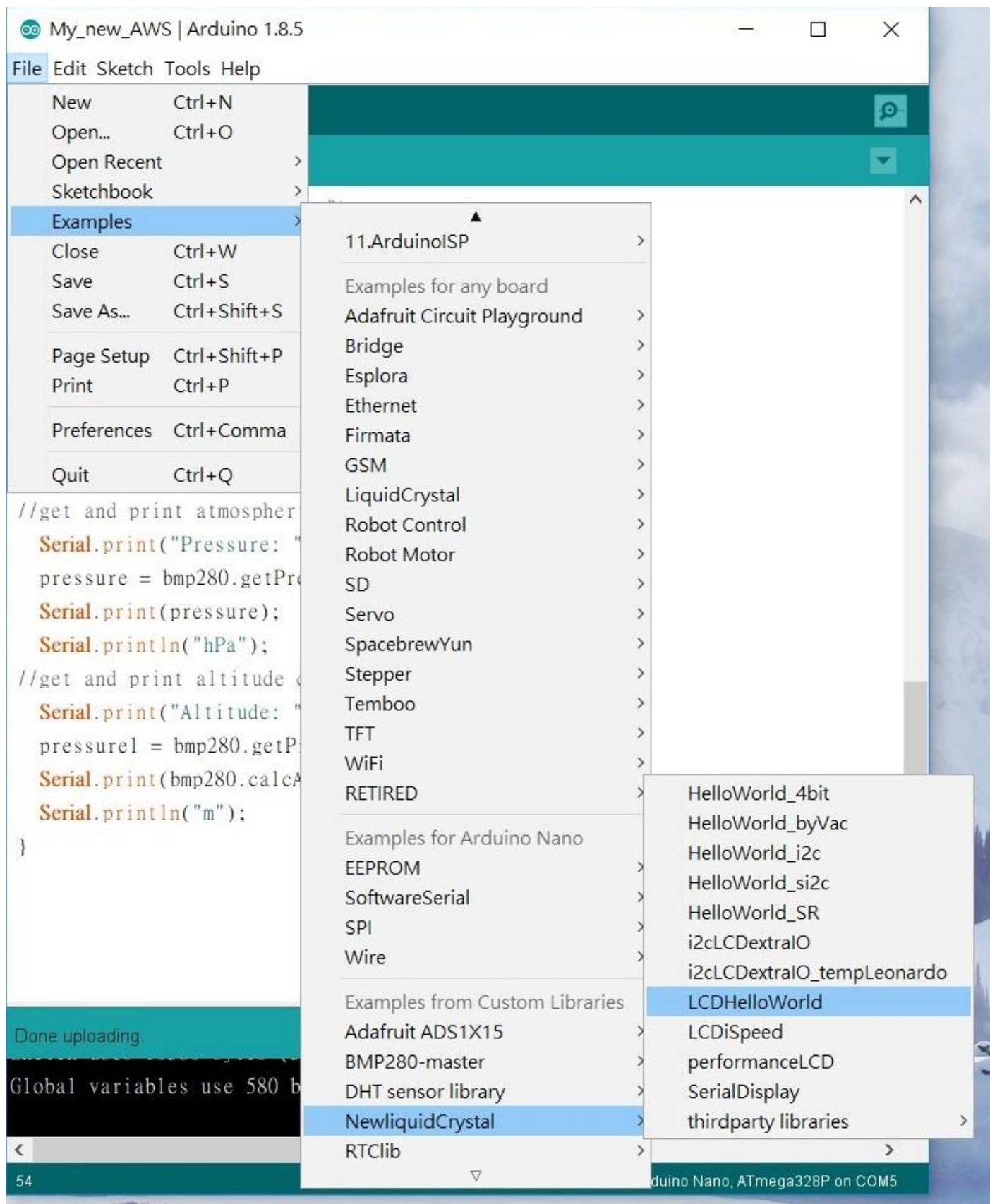
7.1 Wiring

1. Connect LCD Vcc to 5V and GND to ground respectively
 2. Connect LCD SCL to Adruino A5
 3. Connect LCD SDA to Adruino A4



7.2 Choosing an Example file in the NewliquidCrystal library

Choose the LCDHelloWorld example file from the NewliquidCrystal library as shown.



Compile and upload the LCDHelloWorld program to Arduino Nano and see the output shown on the LCD display.

The screenshot shows the Arduino IDE interface with two windows open. The top window is titled 'My_new_AWS | Arduino 1.8.5' and contains a sketch named 'LCDHelloWorld'. The sketch code is as follows:

```
//Hello World program for the LCD20x4 display

}
void
{
    #include <Wire.h>
    #include <LiquidCrystal_I2C.h>

    #define BACKLIGHT_PIN 3

//get
//LiquidCrystal_I2C lcd(0x3F,2,1,0,4,5,6,7,3,POSITIVE); // Set the LCD I2C address
//LiquidCrystal_I2C lcd(0x27,2,1,0,4,5,6,7,3,POSITIVE);

pre
Ser
void setup () {
    Ser
        lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
    Ser
        lcd.setBacklight(HIGH);
    Ser
        lcd.begin(20,4);           // initialize the lcd
    pre
    Ser
        lcd.setCursor(0,0);
    Ser
        lcd.clear();
    Ser
        lcd.print("Hello World");
    }

void loop () {

}

Done
Global Done uploading.

Global variables use 244 bytes (11%) of dynamic memory, leaving 1804 bytes for 1
54
```

The bottom window is also titled 'LCDHelloWorld | Arduino 1.8.5' and shows the status message 'Done uploading.' and memory usage information. A red circle highlights the 'Upload' button in the toolbar of the main window.



7.3 Modifying the My_new_AWS file to display information on the LCD display

Copy the highlighted lines in the header and setup() sections from LCDHelloWorld to the My_new_AWS. Add 2 new commands into the script in having the time format as XX:YY:ZZZZ HH:MM:SS, since if we print the date out directly, when having a single digit time (i.e. 10:09:04, the display will show 10:9:4 instead of 10:09:04), therefore we need to add a command in keeping the format.

```

My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS
#include <Wire.h>
#include <DHT.h>
#include "MP280.h"
#include <TimeLib.h>
#include <DS1307RTC.h>
#include <LiquidCrystal_I2C.h>

#define I2C_PIN A0
#define I2C_TYPE I2C2
#define BACKLIGHT_PIN 3
DHT dht(BACKLIGHT_PIN, DHTTYPE);
MP280 mp280;
LiquidCrystal_I2C lcd(0x2F, 2, 0, 4, 5, 6, 7, 3, POSITIVE);

char myArray1[19];
char myArray2[19];

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.println("This is my first AWS program :)");
    dht.begin();
    mp280.init();
    lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
    lcd.setBacklight(HIGH);
    lcd.begin(20,4);
}

void loop() {
    tmElements_t tm;
    RTC.read(tm);
    // put your main code here, to run repeatedly:
    //Print Date and Time on display
    sprintf(myArray1, "%02d/%02d/%04d", tm.Day, tm.Month, tm.Year);
    lcd.setCursor(0,0);
    lcd.print(myArray1);
    sprintf(myArray2, "%02d:%02d:%02d", tm.Hour, tm.Minute, tm.Second);
    lcd.setCursor(12,0);
    lcd.print(myArray2);
}

```

```

LCDHelloWorld | Arduino 1.8.5
File Edit Sketch Tools Help
LCDHelloWorld
//Hello World program for the LCD20x4 display

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define BACKLIGHT_PIN 3

LiquidCrystal_I2C lcd(0x2F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Set the LCD I2C address
//LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

void setup() {
    lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
    lcd.setBacklight(HIGH);
    lcd.begin(20,4); // initialize the lcd

    lcd.setCursor(0,0);
    lcd.clear();
    lcd.print("Hello World");
}

void loop () {
}

```

```

@@ My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
void loop() {
tmElements_t tm;
RTC.read(tm);
// put your main code here, to run repeatedly:

//Print Date and Time on display
sprintf(myArray1, "%02d/%02d/%04d", tm.Day, tm.Month, tmYearToCalendar(tm.Year));
lcd.setCursor(0,0);
lcd.print(myArray1);

sprintf(myArray2, "%02d:%02d:%02d", tm.Hour, tm.Minute, tm.Second);
lcd.setCursor(12,0);
lcd.print(myArray2);

//Abbreviation setup
lcd.setCursor(0,1);
lcd.print("Temp: ");
lcd.setCursor(0,2);
lcd.print("RH: ");
lcd.setCursor(0,3);
lcd.print("Pres: ");

//Print Temperature information on display
float t = dht.readTemperature();
lcd.setCursor(6,1);
lcd.print(t,1);
lcd.setCursor(18,1);
lcd.print("C");

//Print Humidity information on display
float h = dht.readHumidity();
lcd.setCursor(6,2);
lcd.print(h,1);
lcd.setCursor(18,2);
lcd.print("%");

//Print Pressure information on display
float p = bmp280.getPressure()/100;
lcd.setCursor(6,3);
lcd.print(p,1);
lcd.setCursor(17,3);
lcd.print("hPa");
delay(1000-millis()%1000 + 10);
}

```

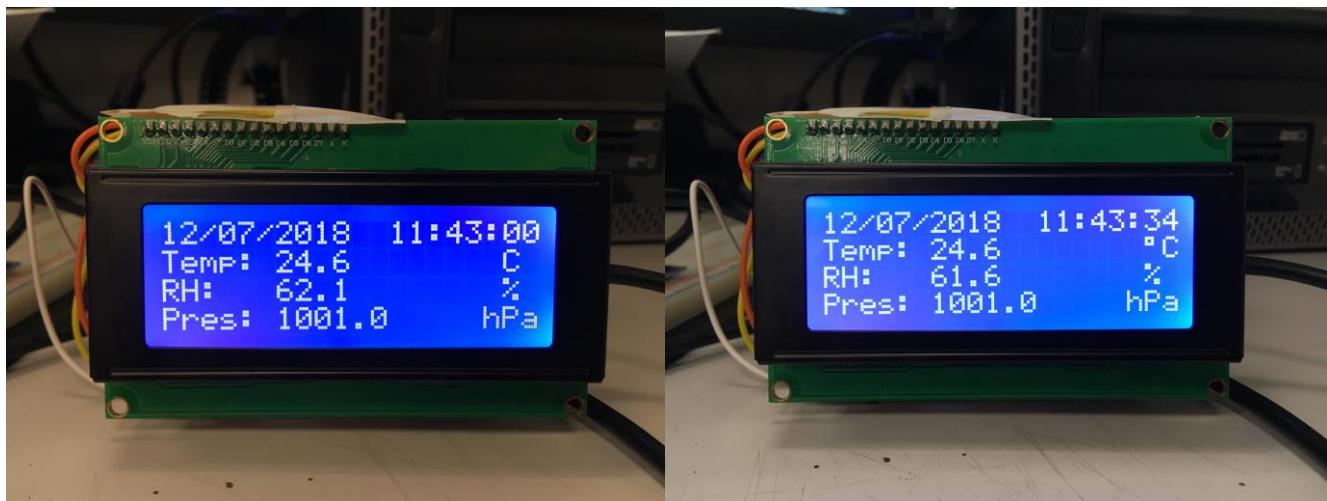
// Integrate the abbreviation setting on display

// Showing temperature info. on display

// Showing humidity info. on display

// Showing pressure info. on display

If you want to output the degree symbol °C, try to change the command `lcd.print("C");` to `lcd.print("\337C")` in temperature module command;



Congratulation! You have now completed your My_new_AWS project!