Training Workshop on

Arduino-based Automatic Weather Station (AWS)

Hong Kong Observatory

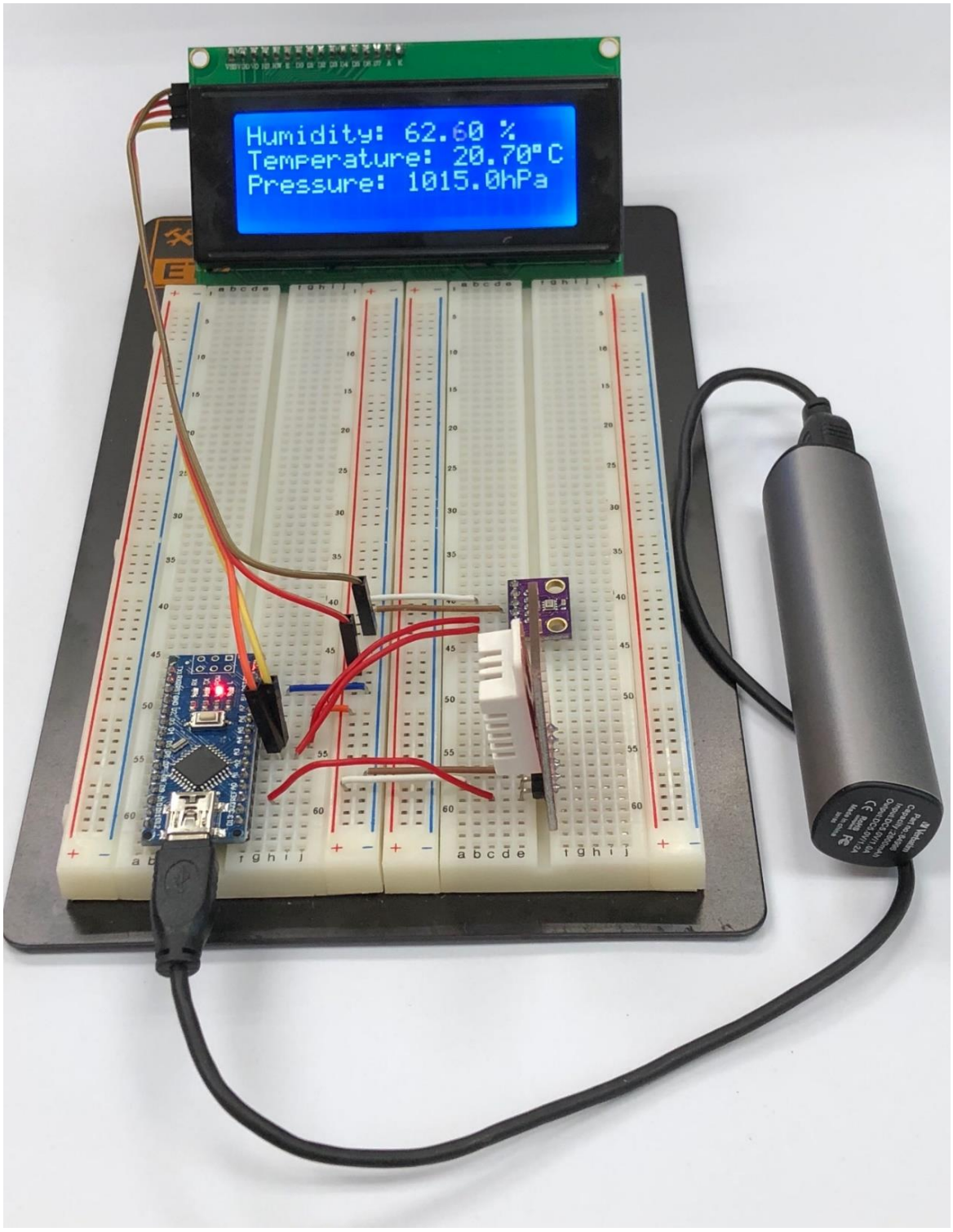


Table of Content

(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.

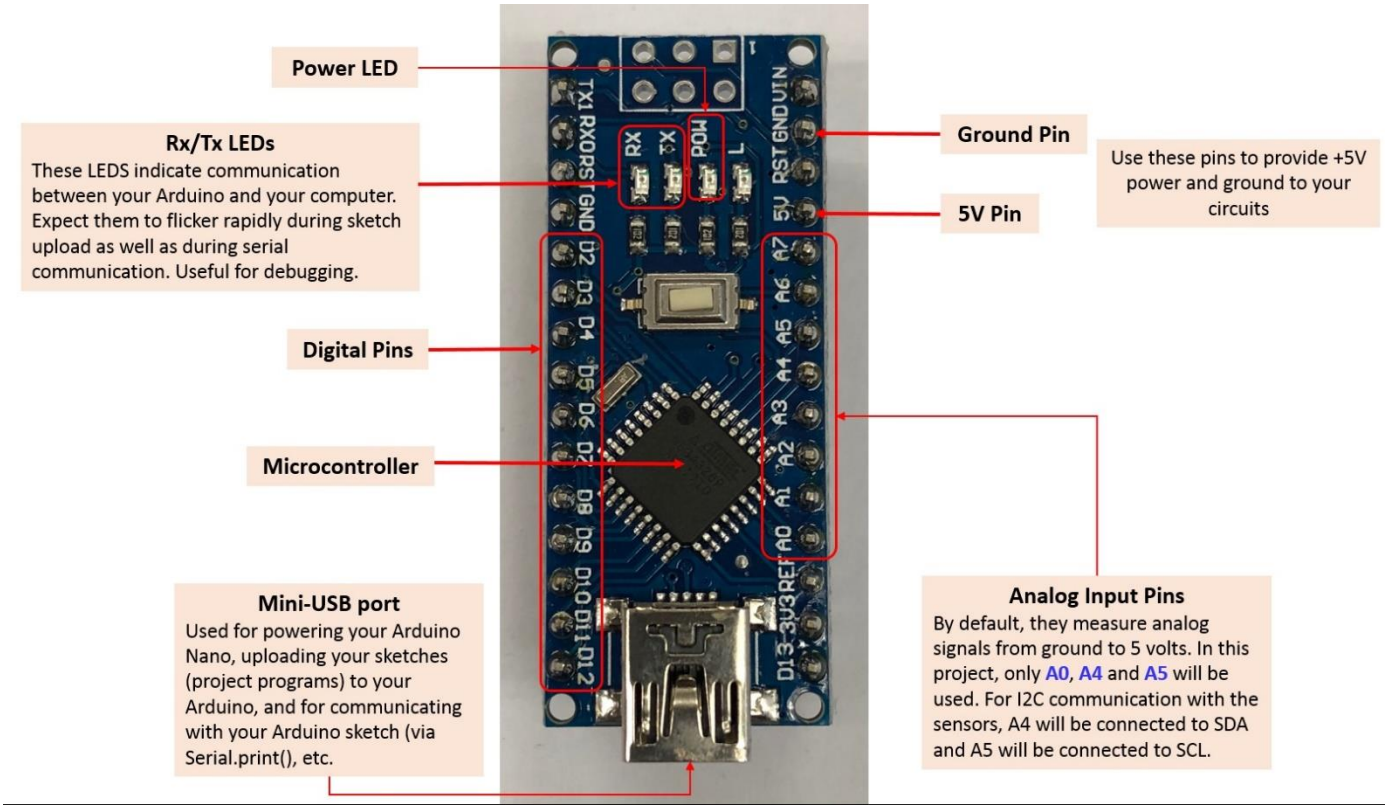
Starting from 2007, with the establishment of The Community Weather Information Network (Co-WIN) in Hong Kong, more than 150 educational institutions and community organizations had joined to install AWS in their own places for promoting education on meteorology and environmental science through the 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.

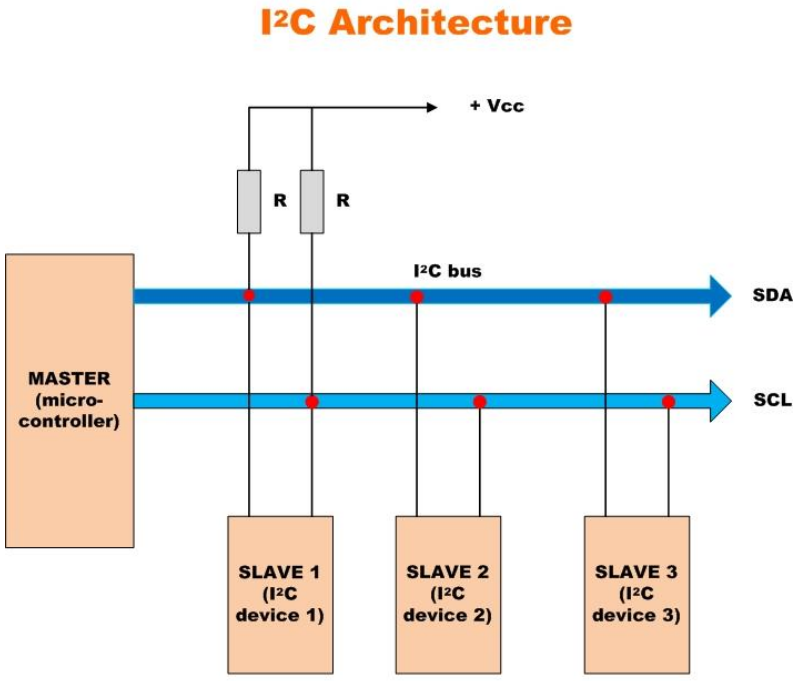
With the Co-WIN sensor network, Hong Kong Observatory and partner institutes could access the database in performing some micro-climate research and investigating the climatology in a designated area, which could help in having a better understanding on our living environment.

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.

(2)Getting started

2.1 The Arduino Nano Board

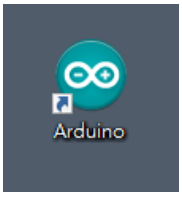




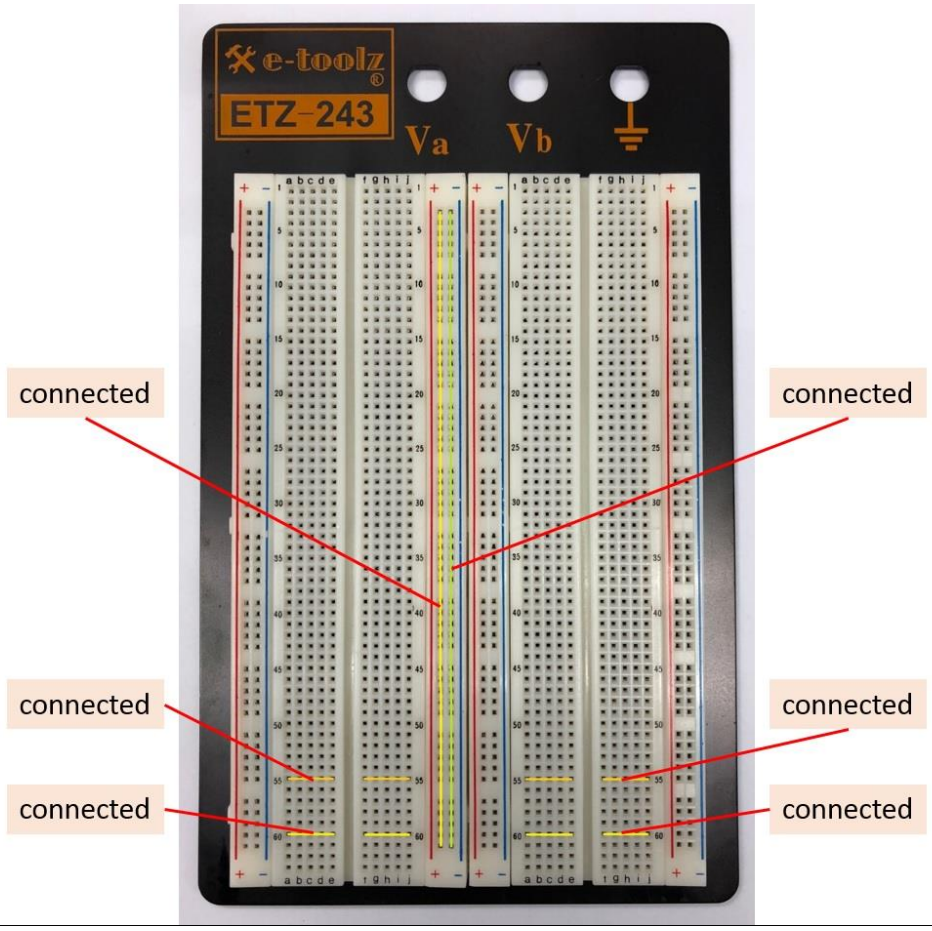
2.2 The Arduino IDE

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)

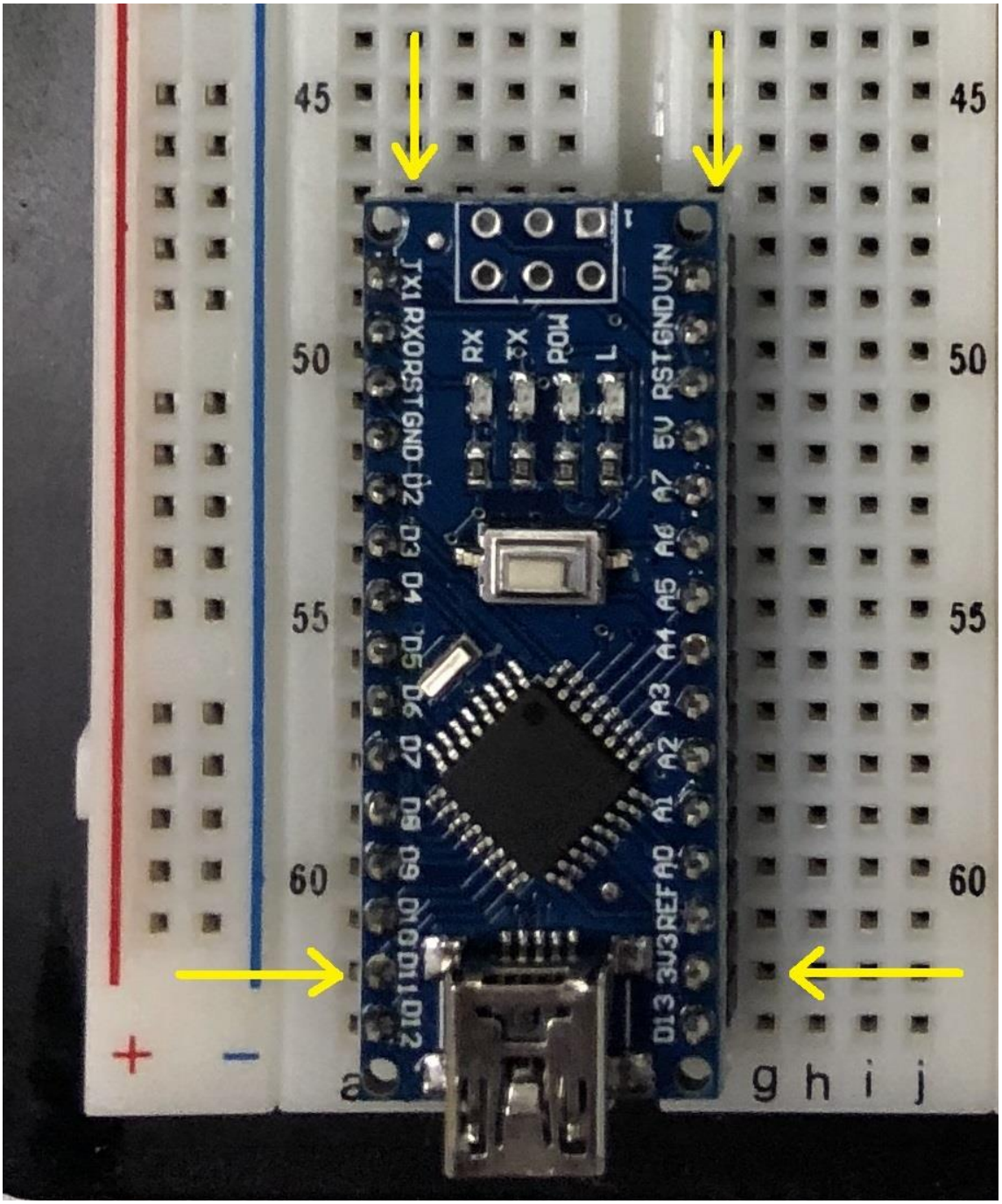


2.3 Basic information about the Breadboard used in this workshop



2.4 Connecting your Arduino Nano to a PC

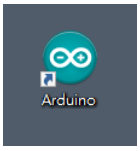
Insert the Arduino Nano on the breadboard as shown below and connect it to your PC using a mini-USB cable



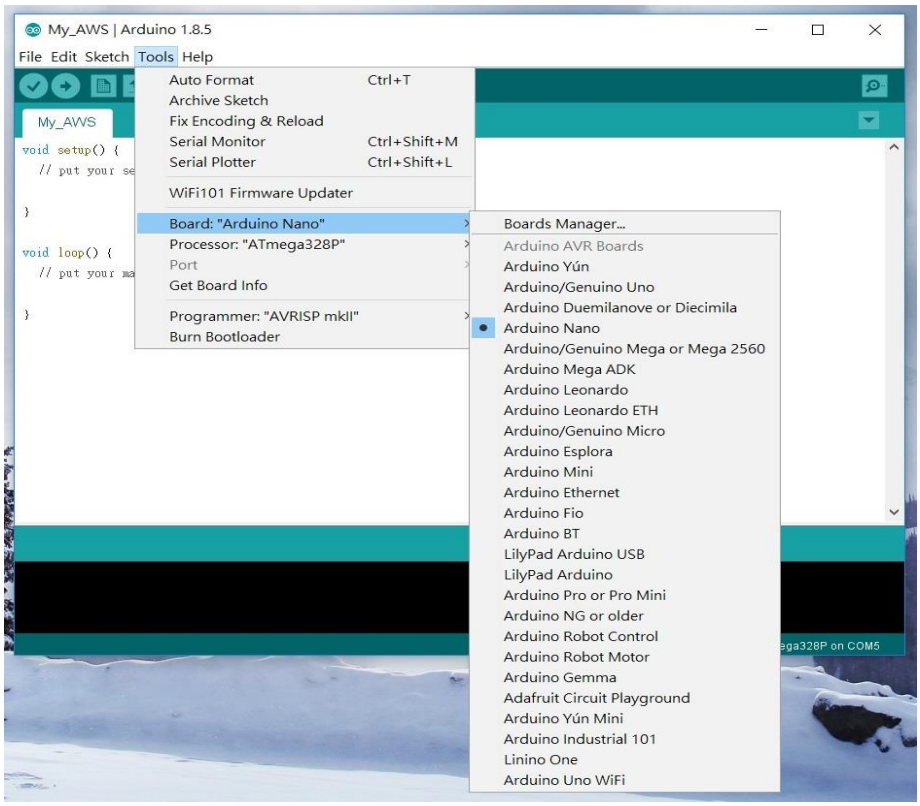
(3) Creating your first AWS project

3.1 Running the Arduino IDE

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

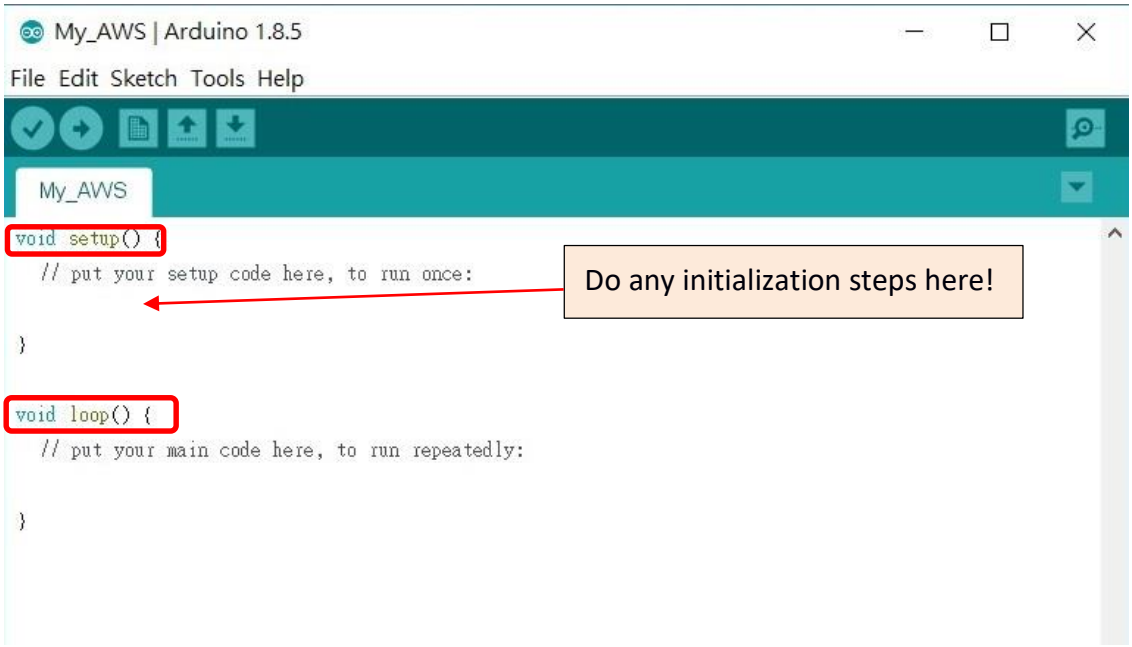


3.2 Selecting a proper Arduino board



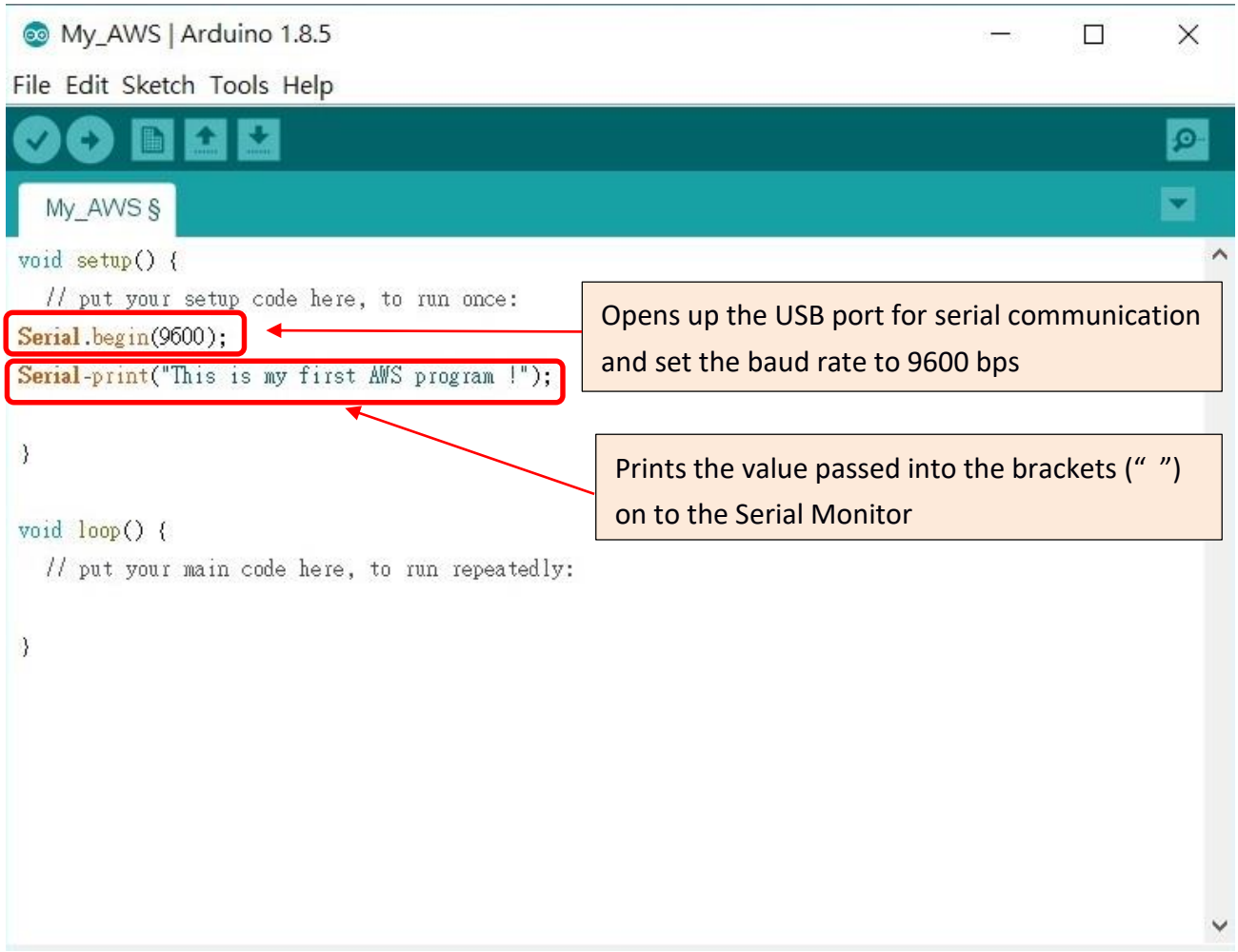
3.3 Creating a new project file

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



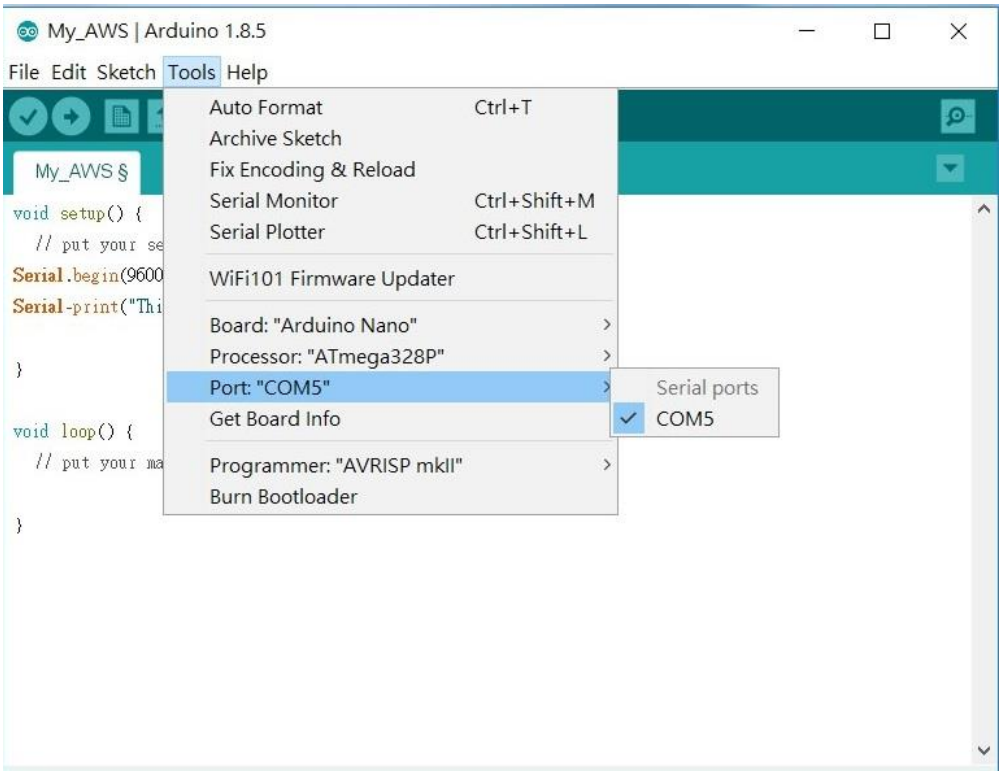
Type the following in the setup() section:

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

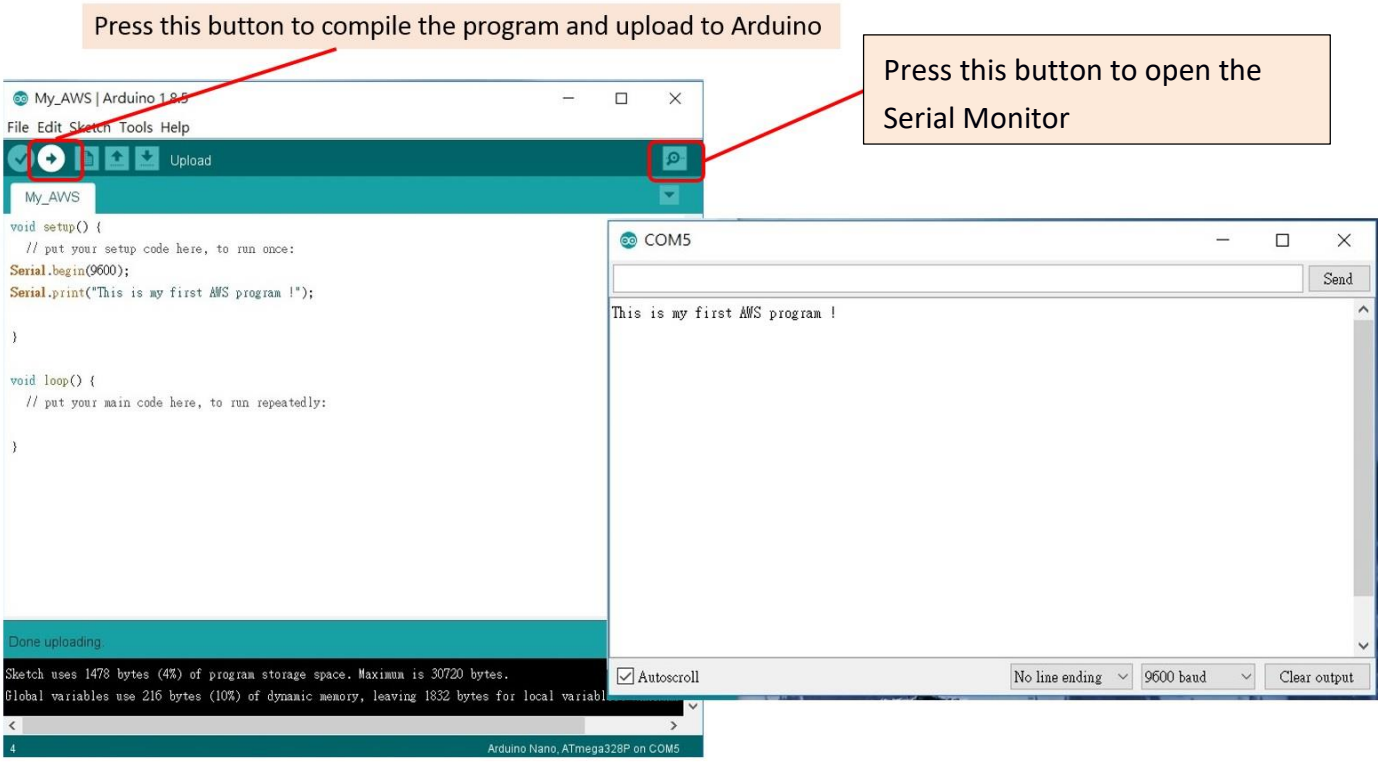


3.4 Setting up an appropriate Communication Port

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

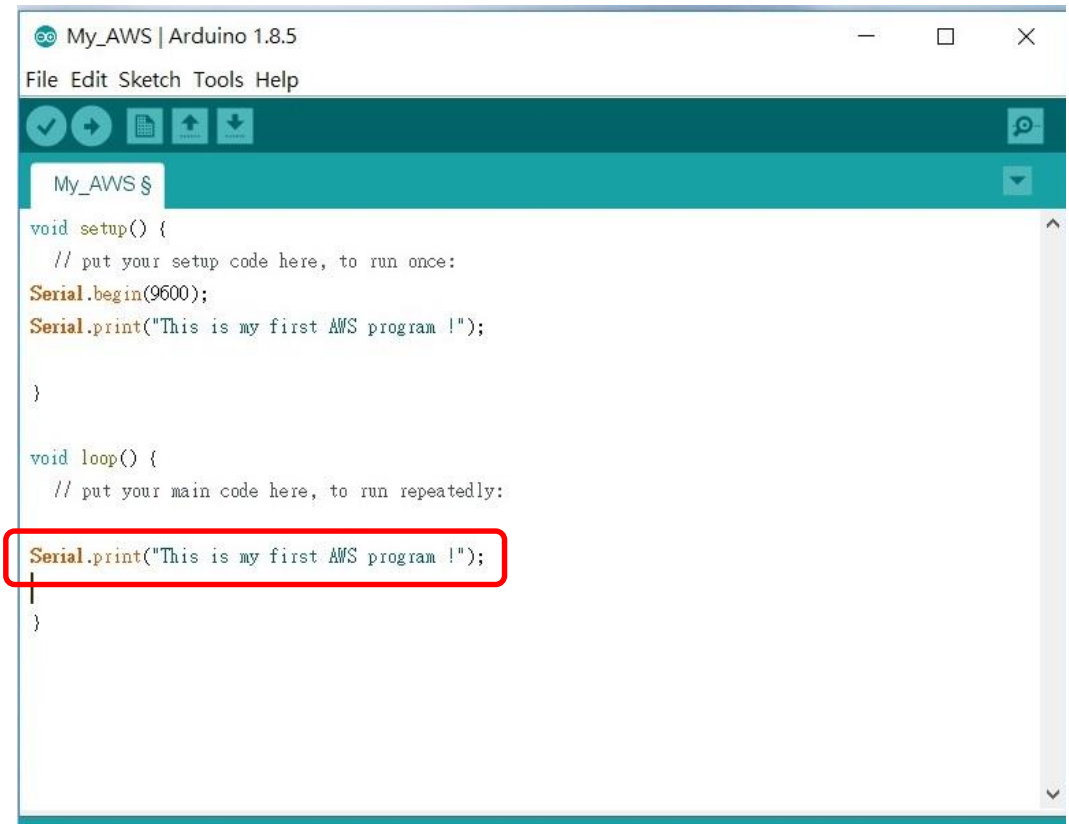


3.5 Compiling and uploading a program to your Arduino

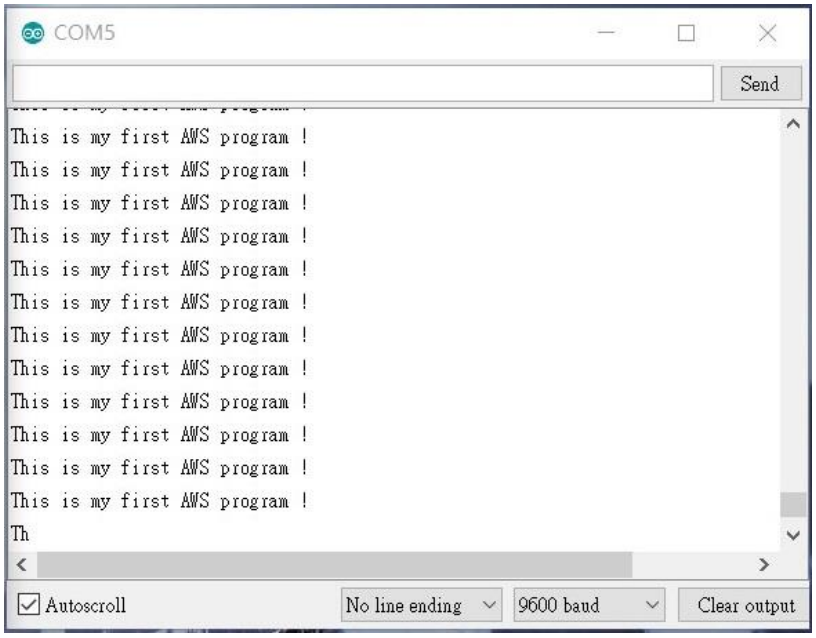


3.6 Using the loop() function

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

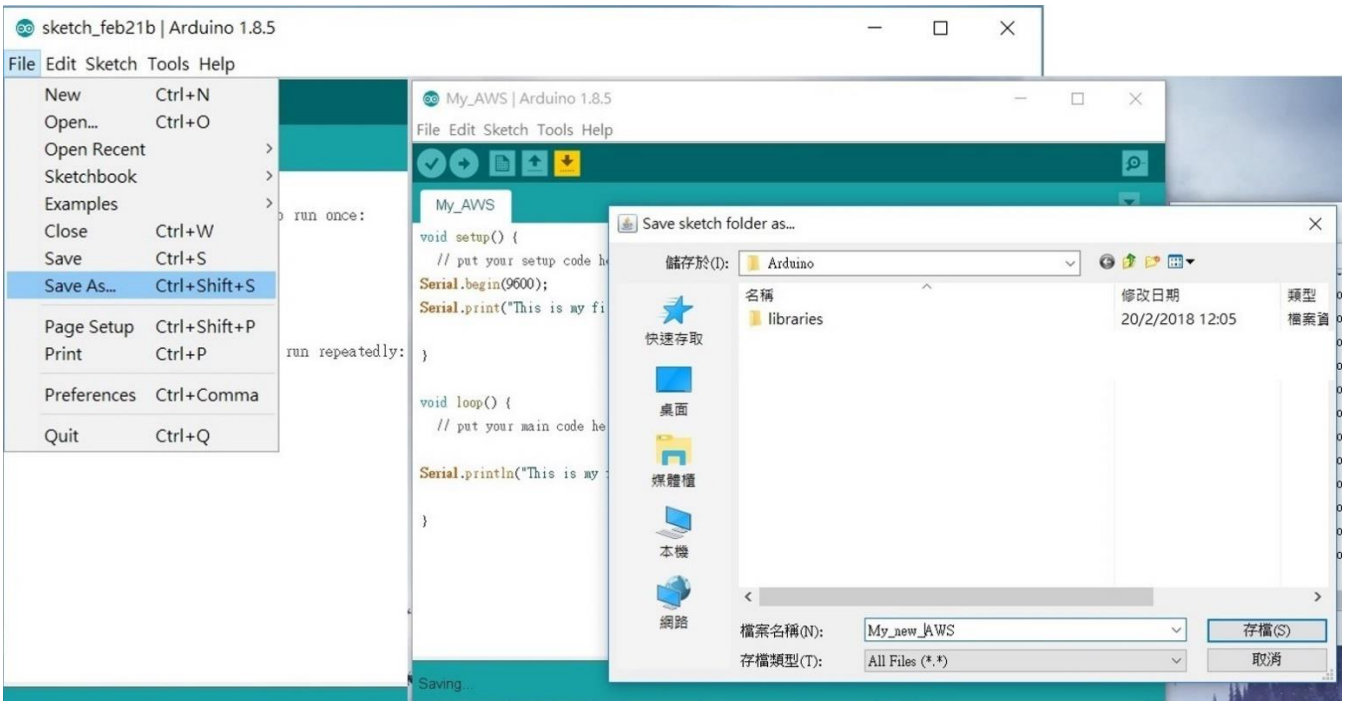


Try using Serial.println("This is my first AWS program! ") instead of Serial.print("This is my first AWS program! ") and recompile your program and see the result.



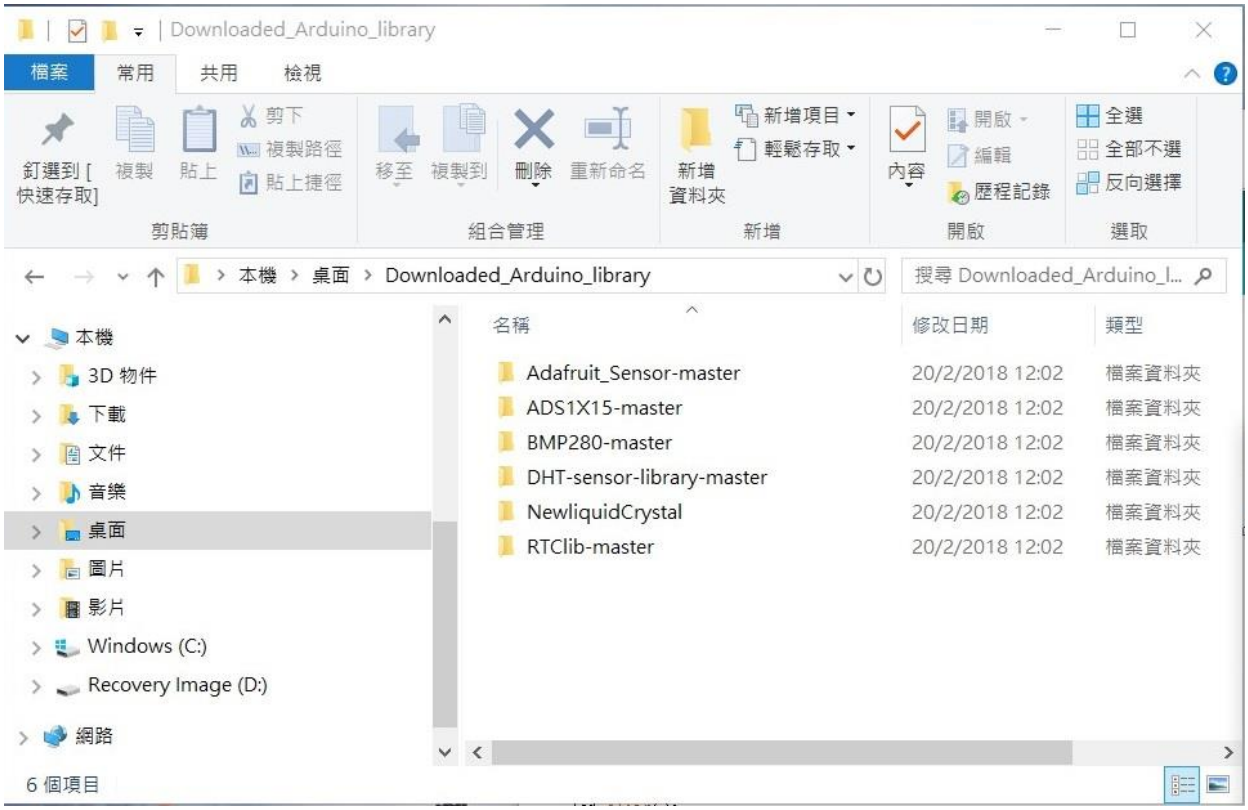
3.7 Saving your first project file

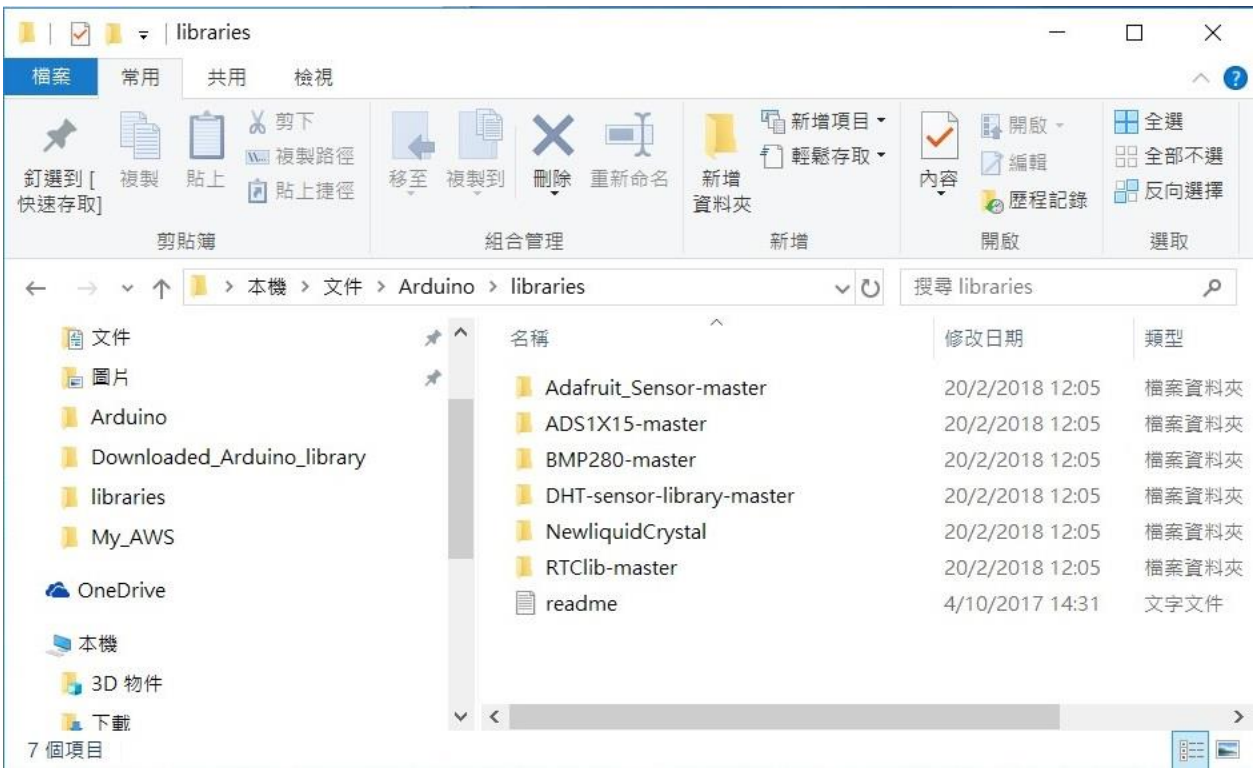
Save the program as My_new_AWS.



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:

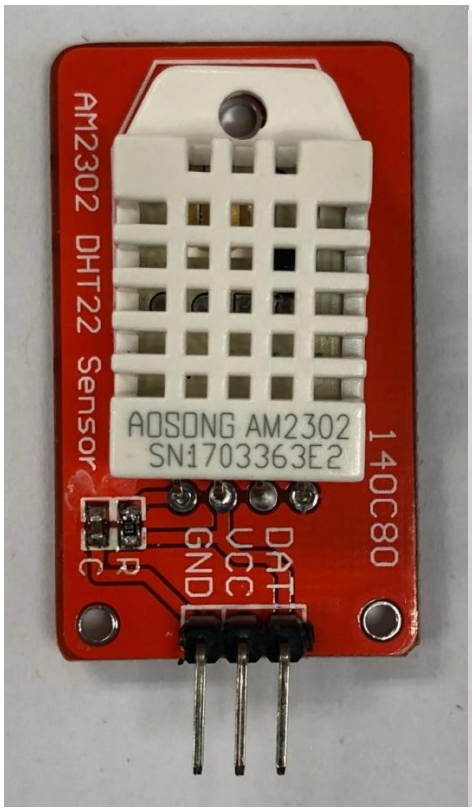


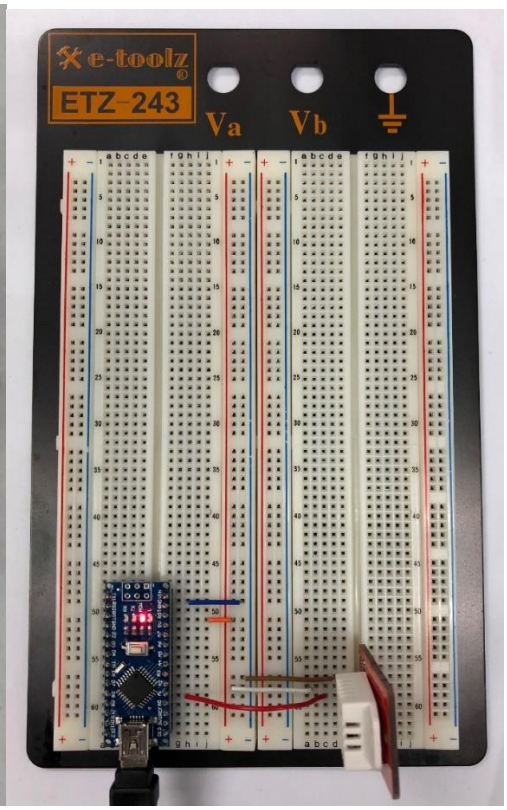


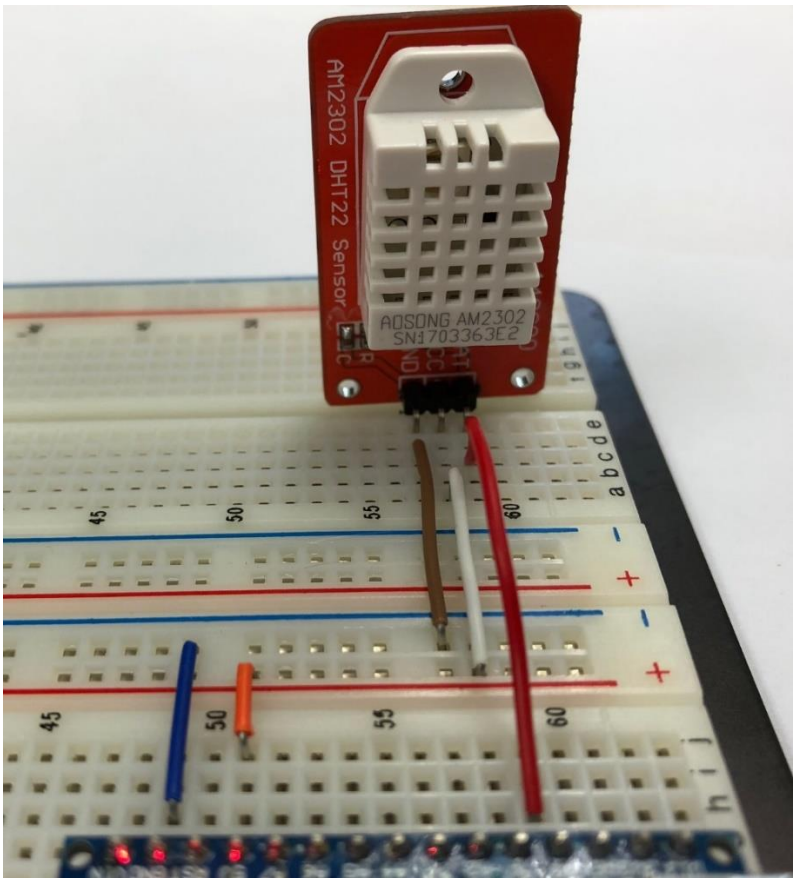
(4) Installing a temperature and humidity sensor DHT22 module

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

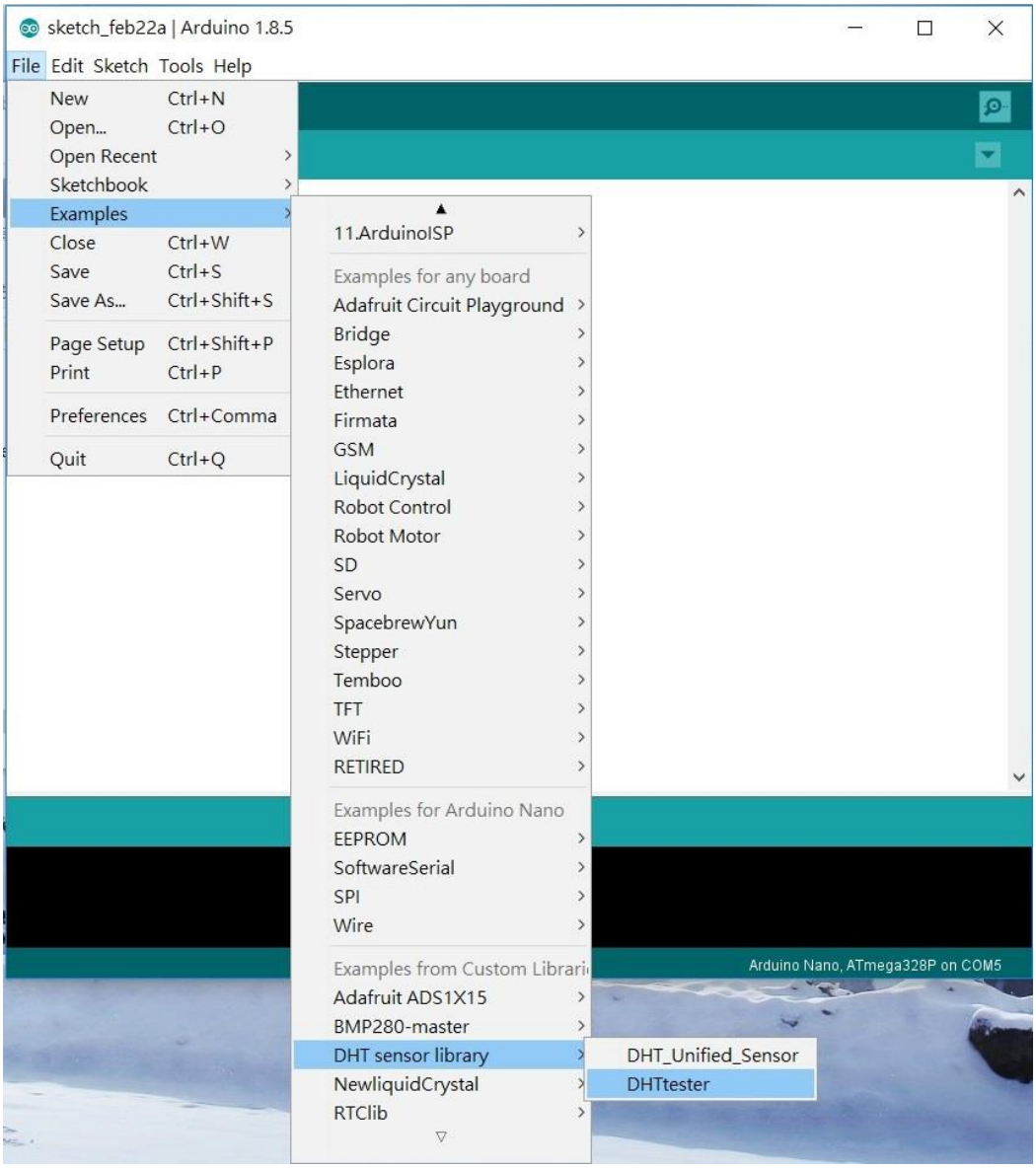






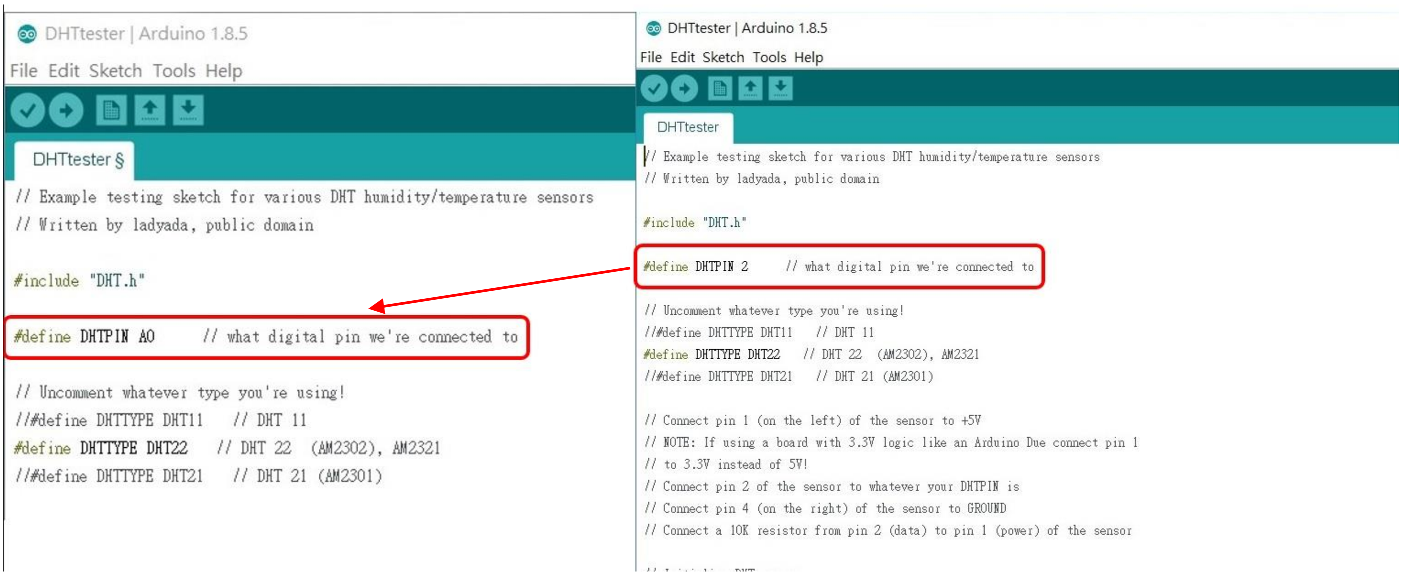
4.2 Choosing an Example file in the DHT sensor library

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

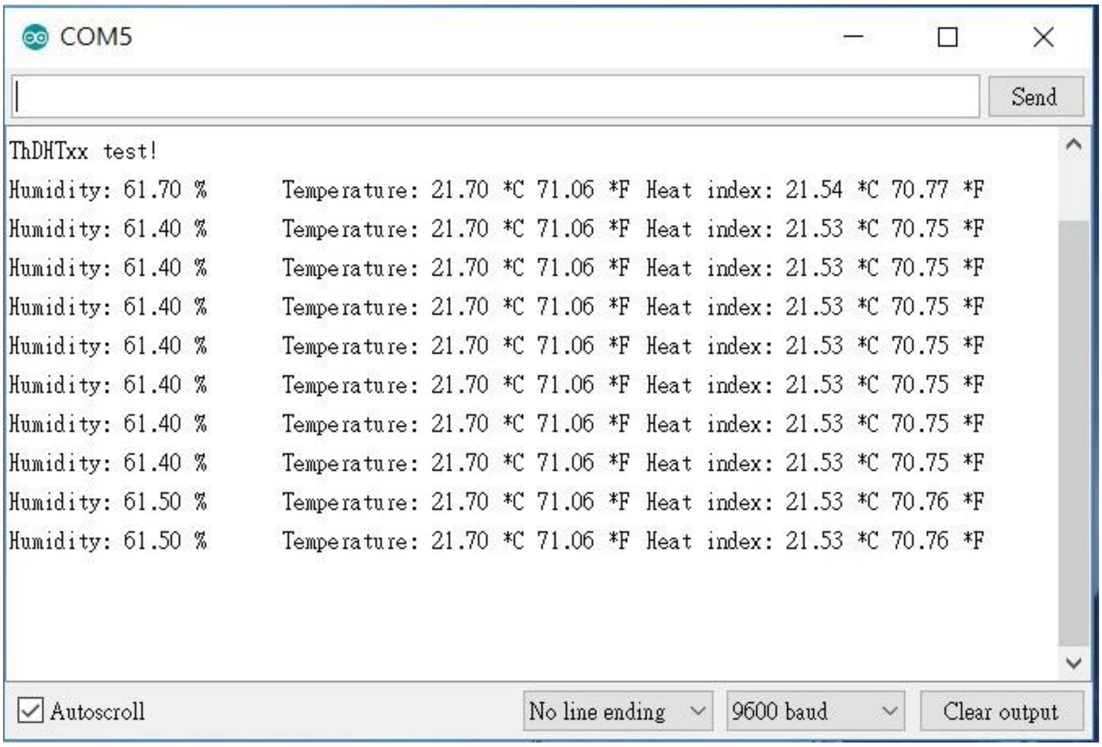


4.3 Modifying the Analog input pin to display correctly

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



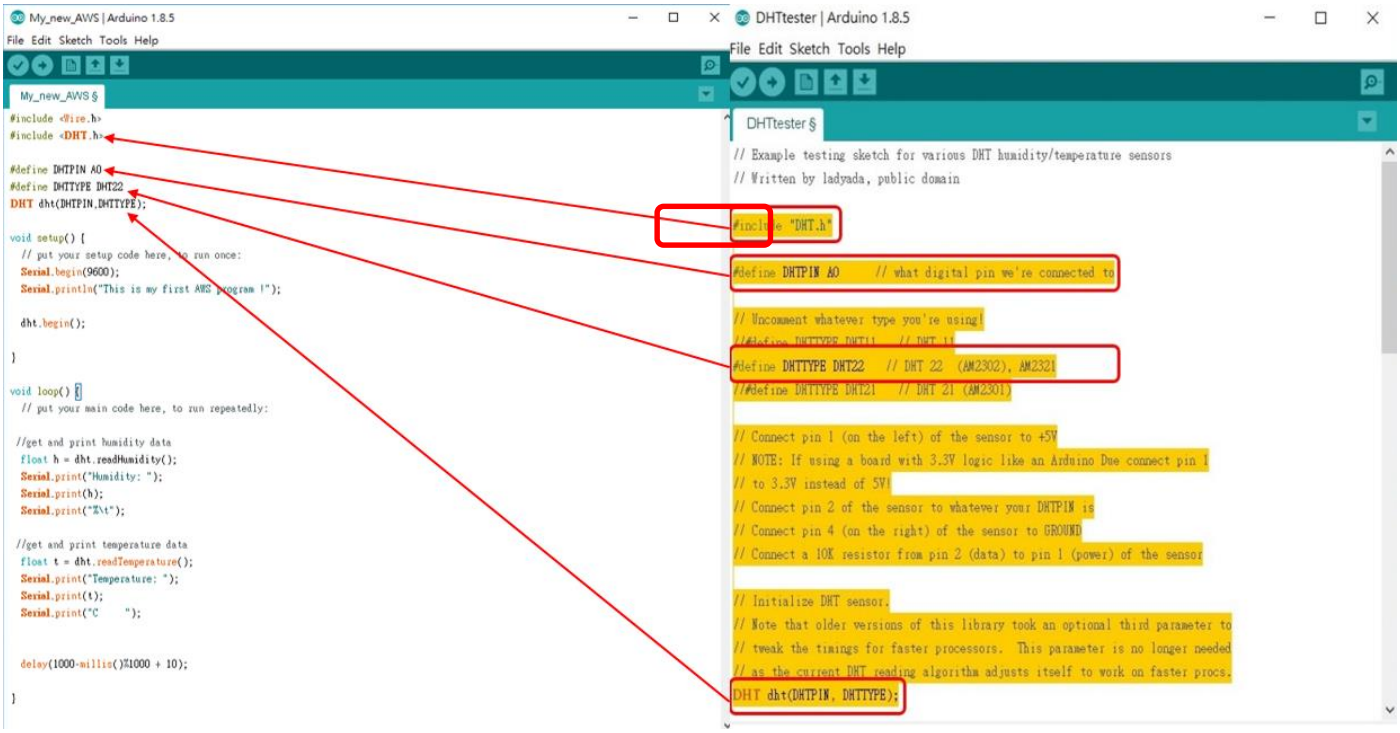
Compile and upload the program again and see the results.



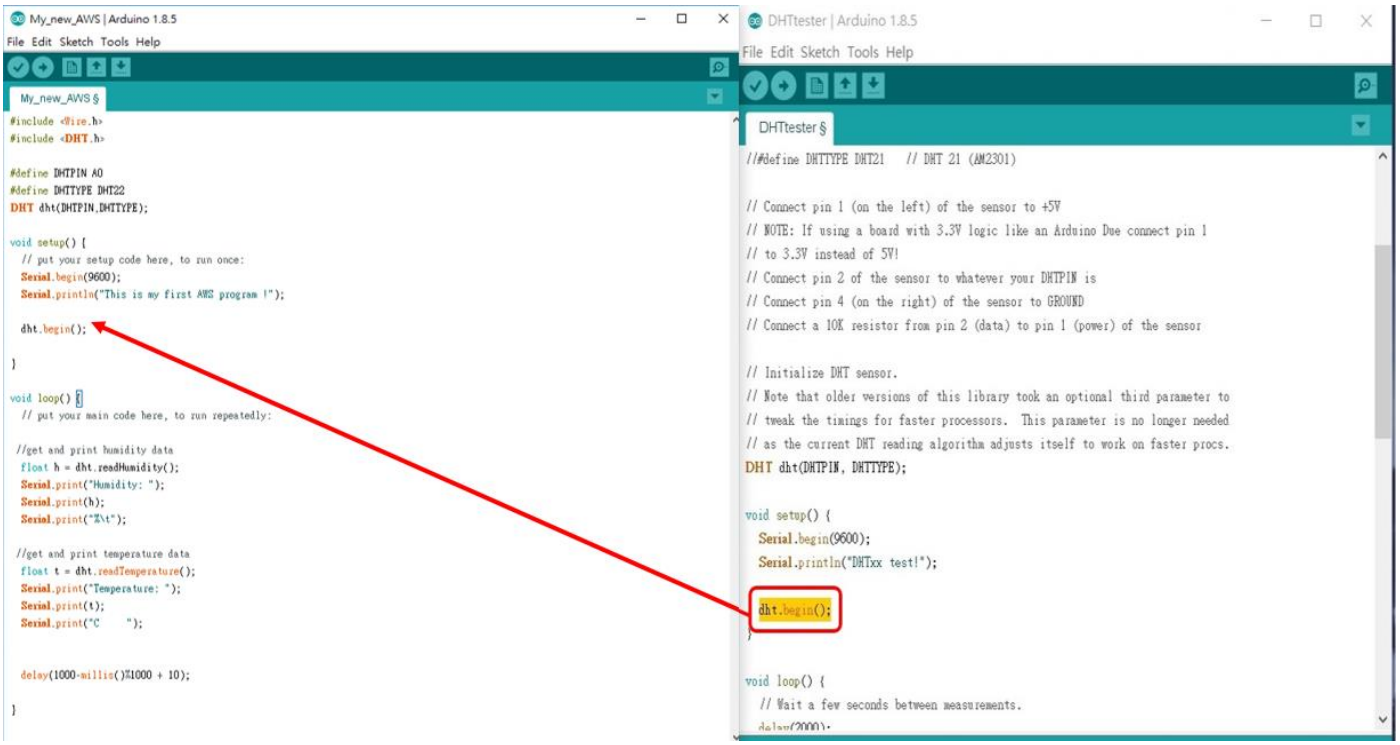
4.4 Modifying the My_new_AWS file to include the DHT22 sensor program

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:

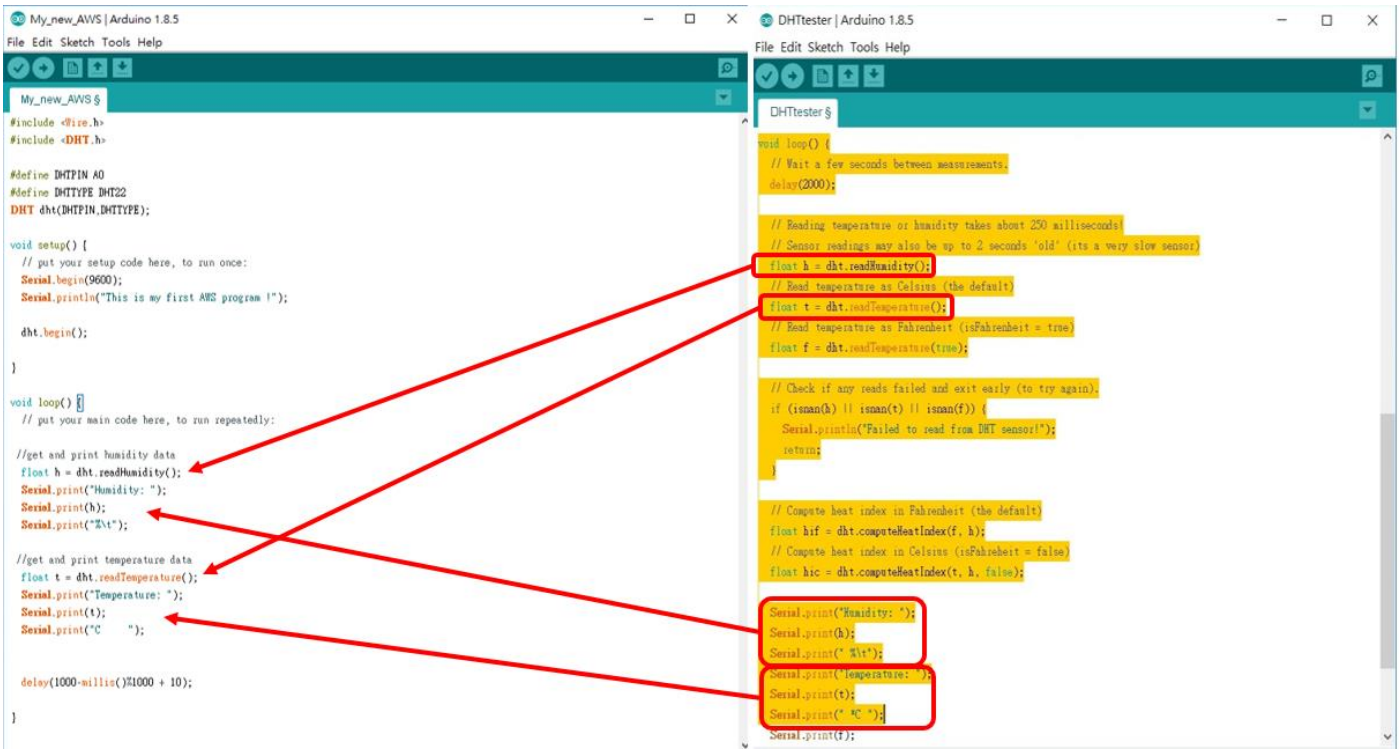
```
#include "DHT.h"
#define DHTPIN A0     // what digital pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
DHT dht(DHTPIN, DHTTYPE);
```



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

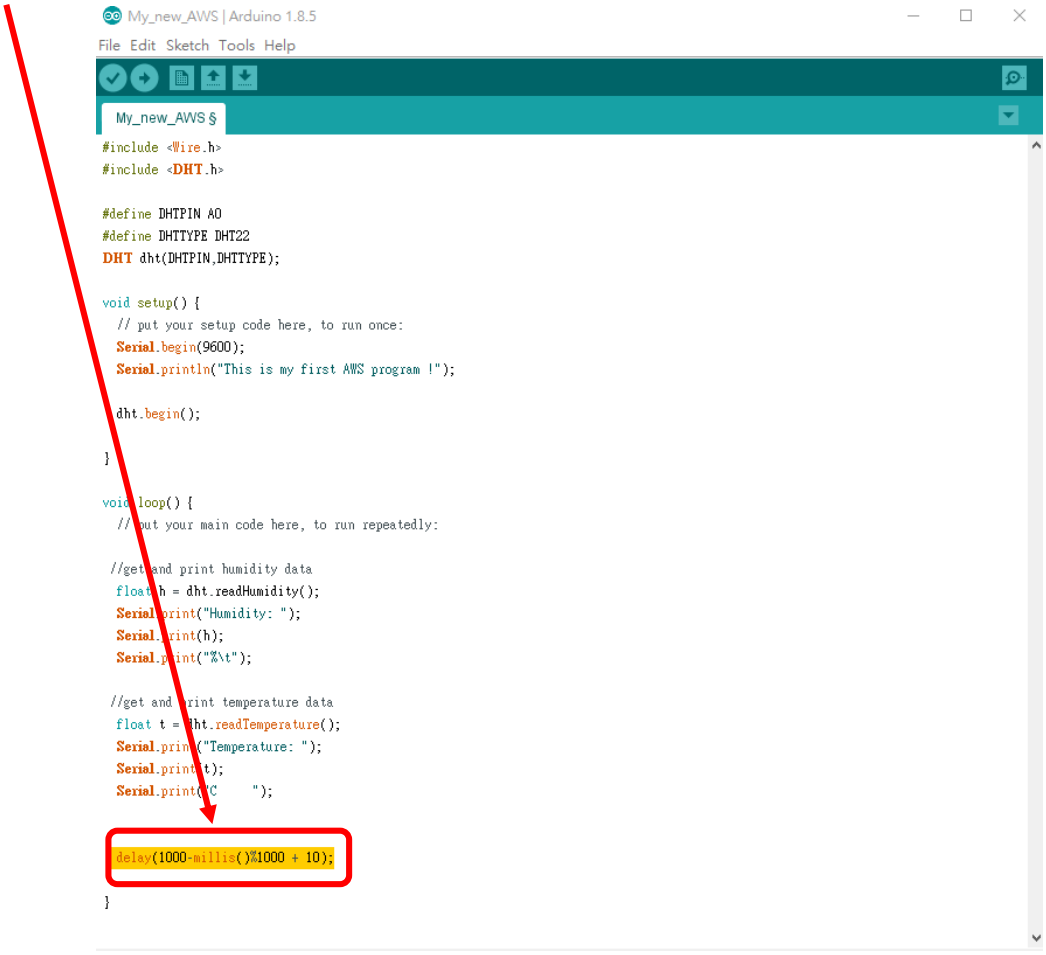


Copy the loop() section of DHTtester to My_new_AWS and modify them as shown below:



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

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



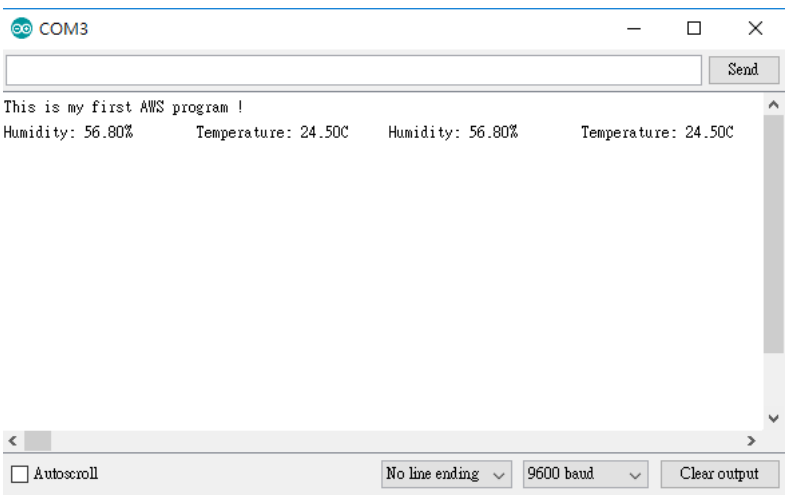
My_new_AWS | Arduino 1.8.5

File Edit Sketch Tools Help

Compile the program and see the results

My_new_AWS §

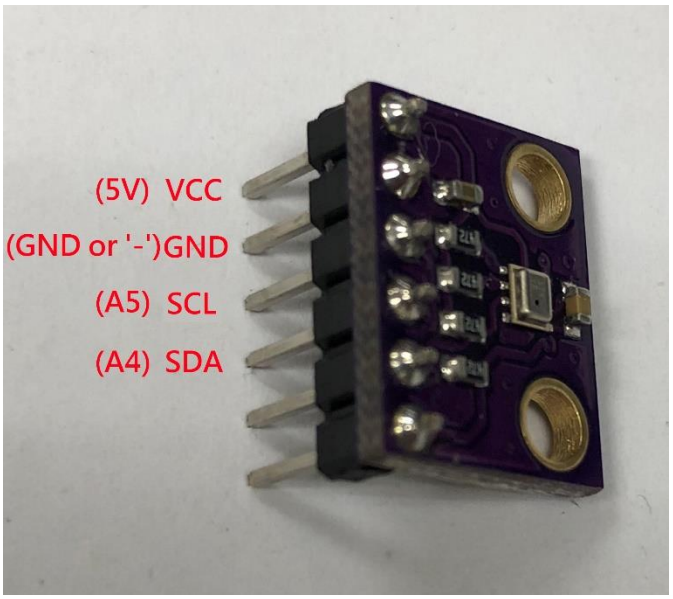
```
#include <Wire.h>
#include <DHT.h>

#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     ");


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

}
```

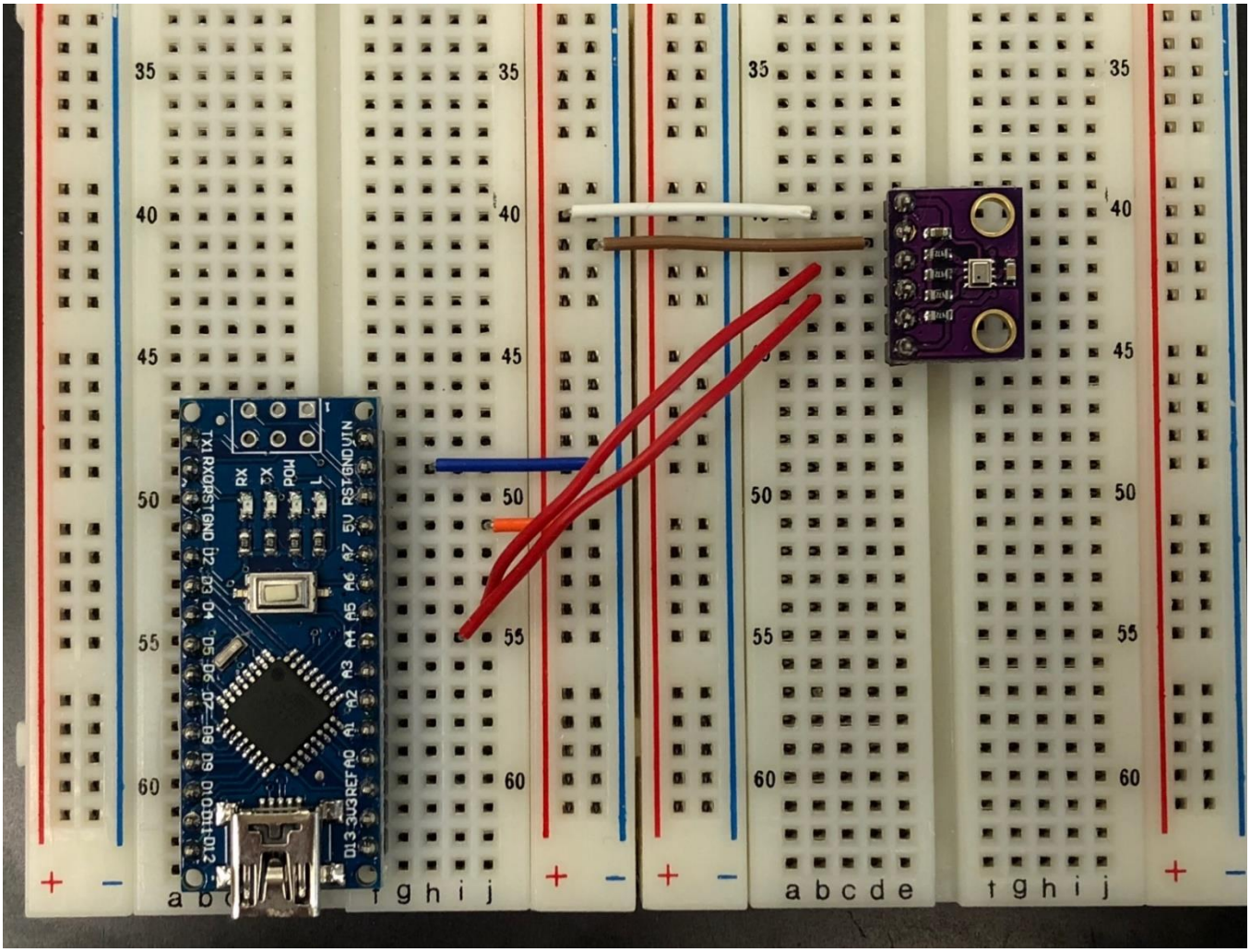


(5)Installing a Pressure sensor BMP280 module

5.1 Wiring

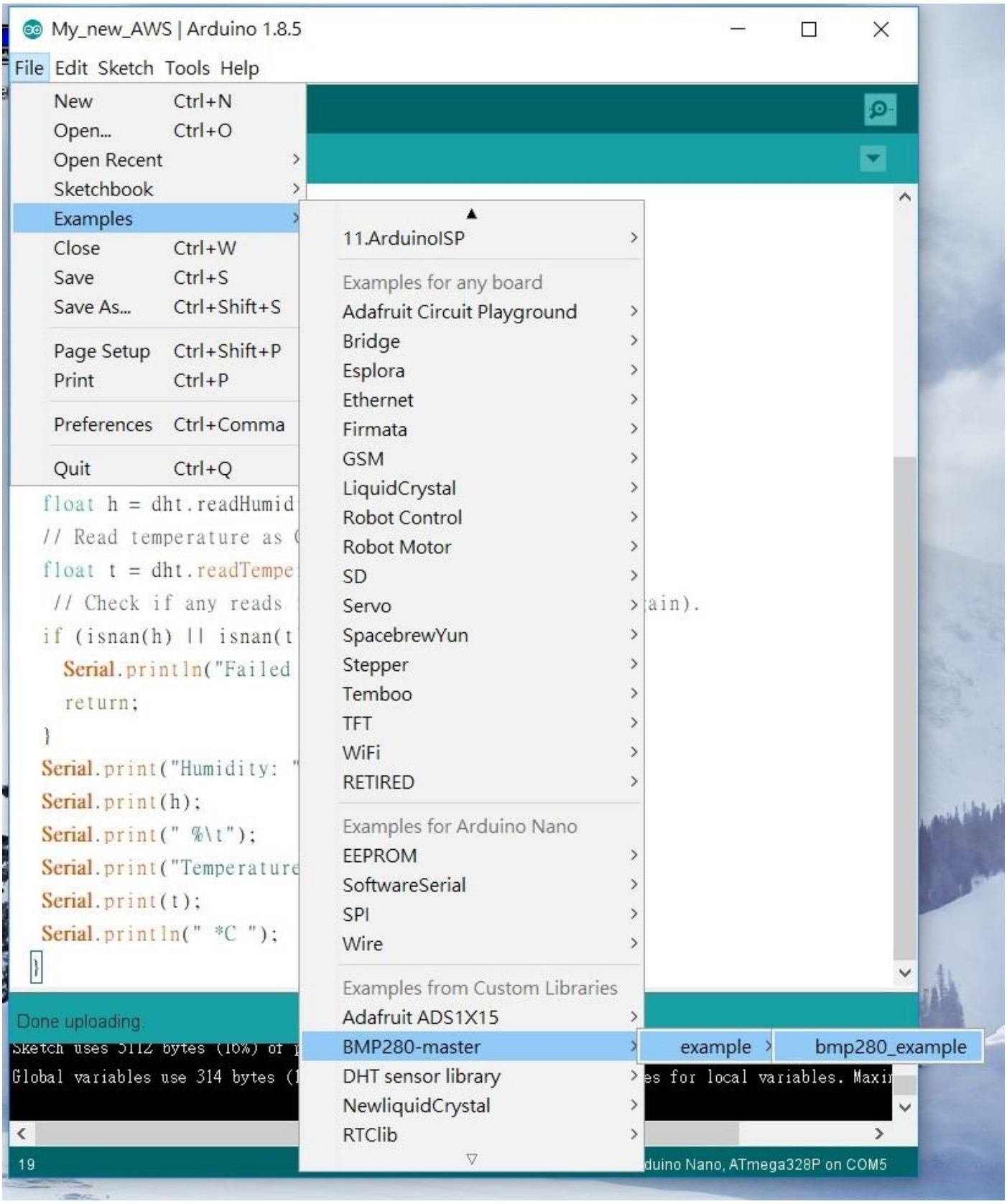
1. Connect BMP280 Vcc to 5V and GND to ground respectively
2. Connect BMP280 SCL to Adruino A5
3. Connect BMP280 SDA to Adruino A4



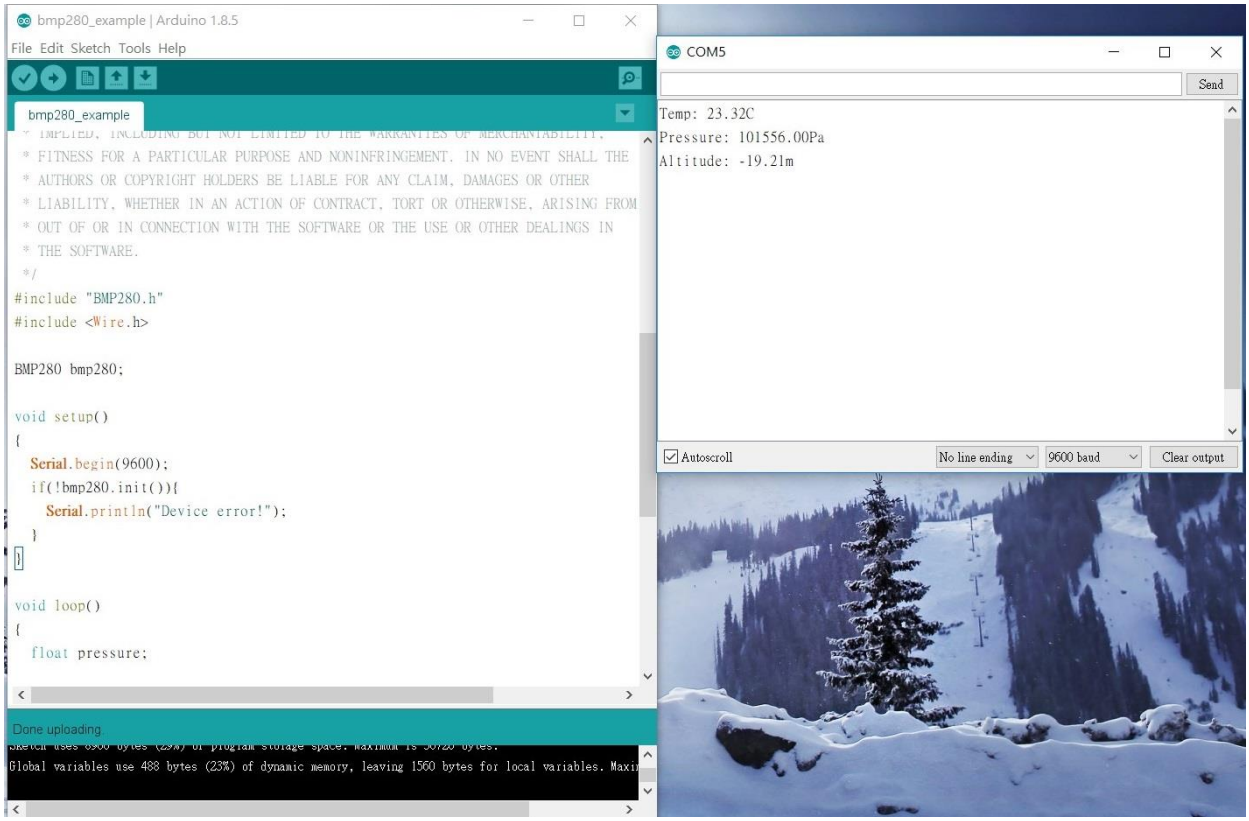


5.2 Choosing an Example file in BMP280 library

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

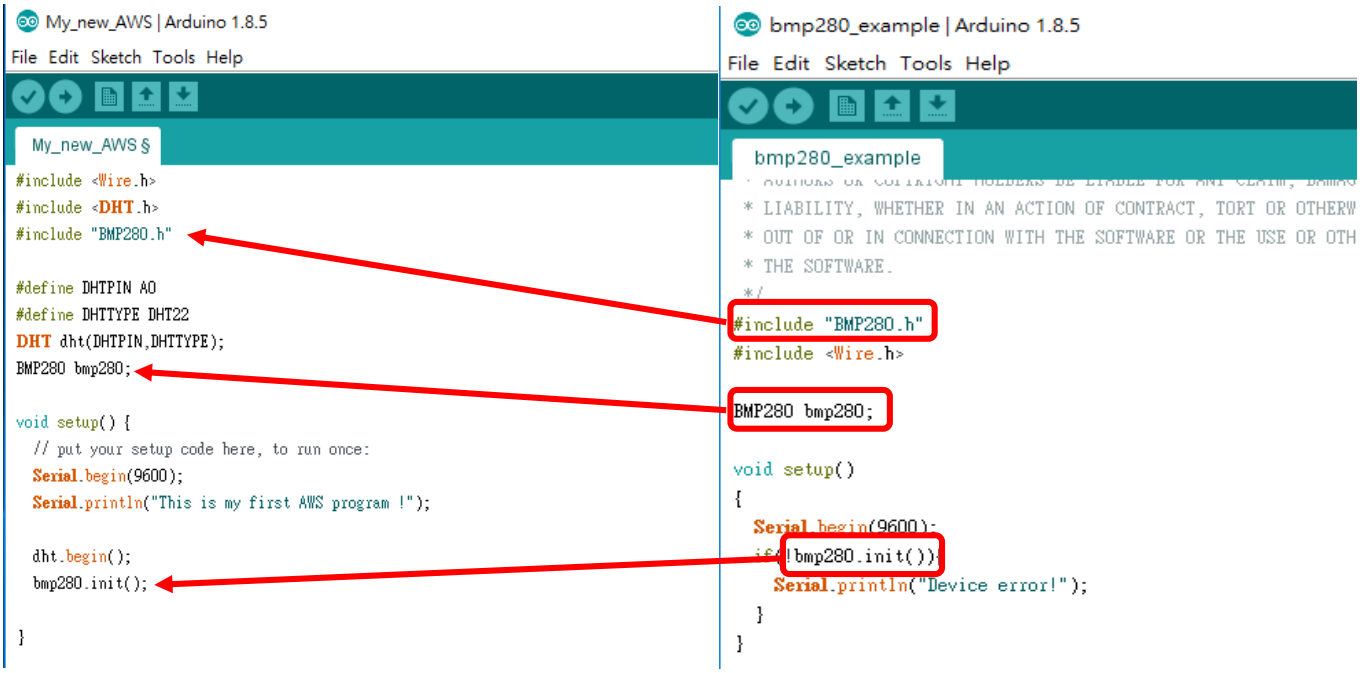


Compile and upload the program to Arduino Nano and see the results.



5.3 Modifying My_new_AWS file to include the BMP280 sensor program

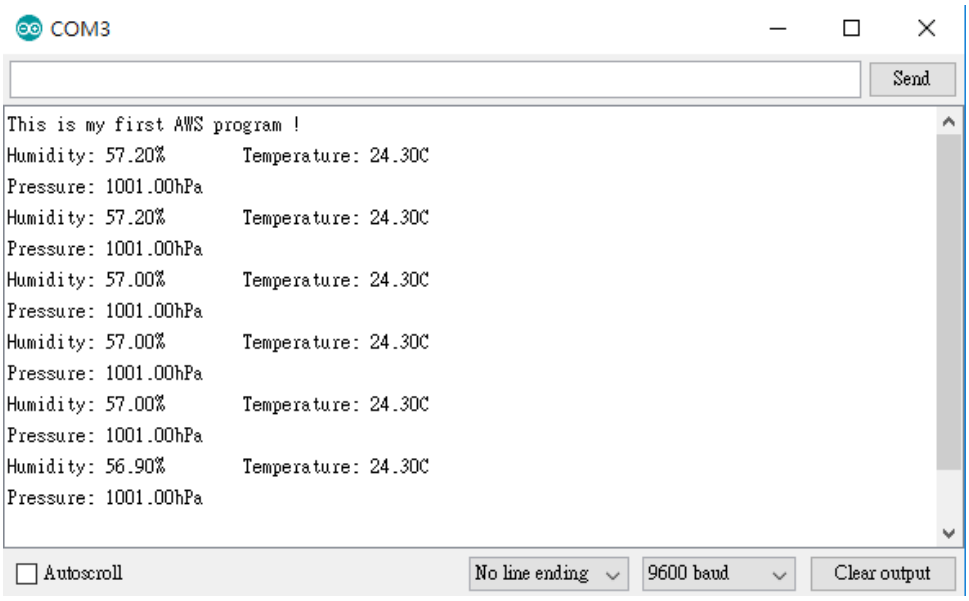
Open My_new_AWS file and copy the below lines from bmp280_example file to the file. In setup(), add bmp.init(); as shown.



```
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.print("%\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.
```

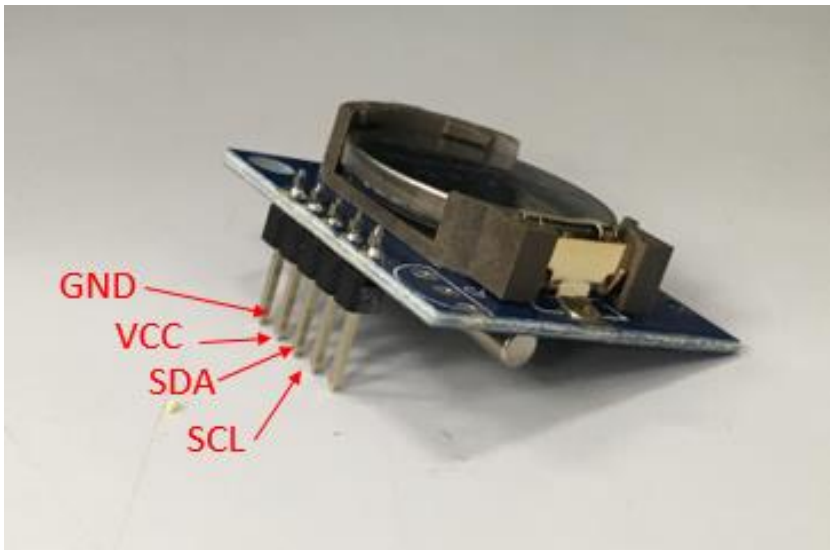
Compile and upload the program to see the results! Remember to save your My_new_AWS file again!

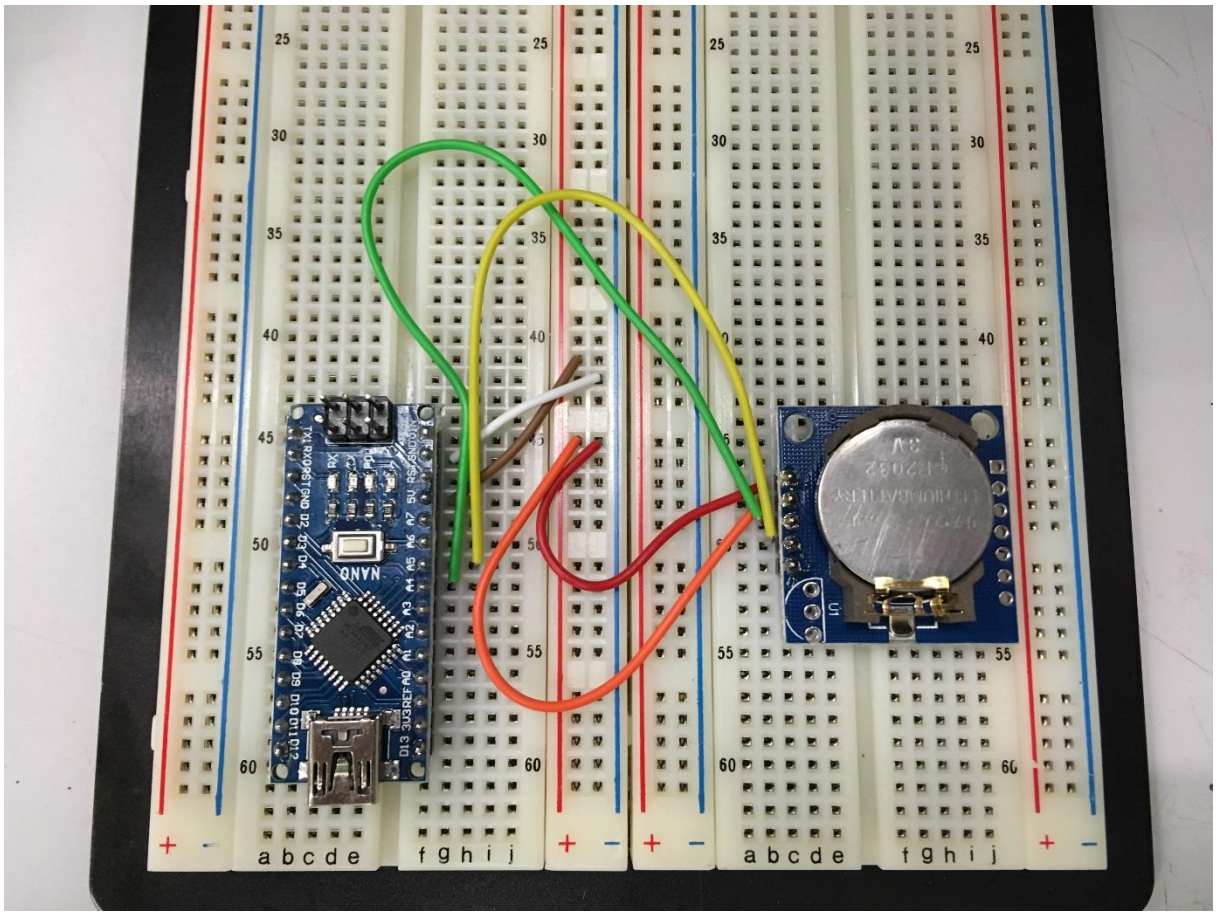


(6) Installing a Real-Time Clock module

6.1 Wiring

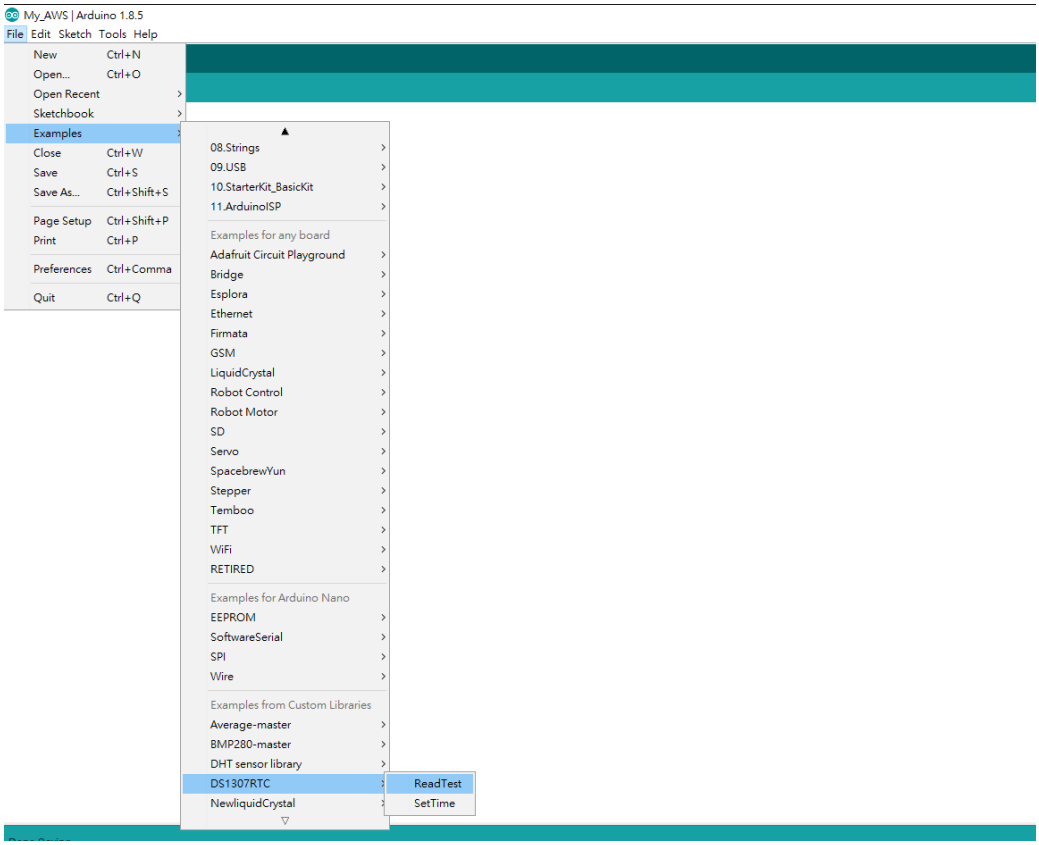
1. Connect DS1307 Vcc to 5V and GND to ground respectively
2. Connect DS1307 SCL to Adruino A5
3. Connect DS1307 SDA to Adruino A4



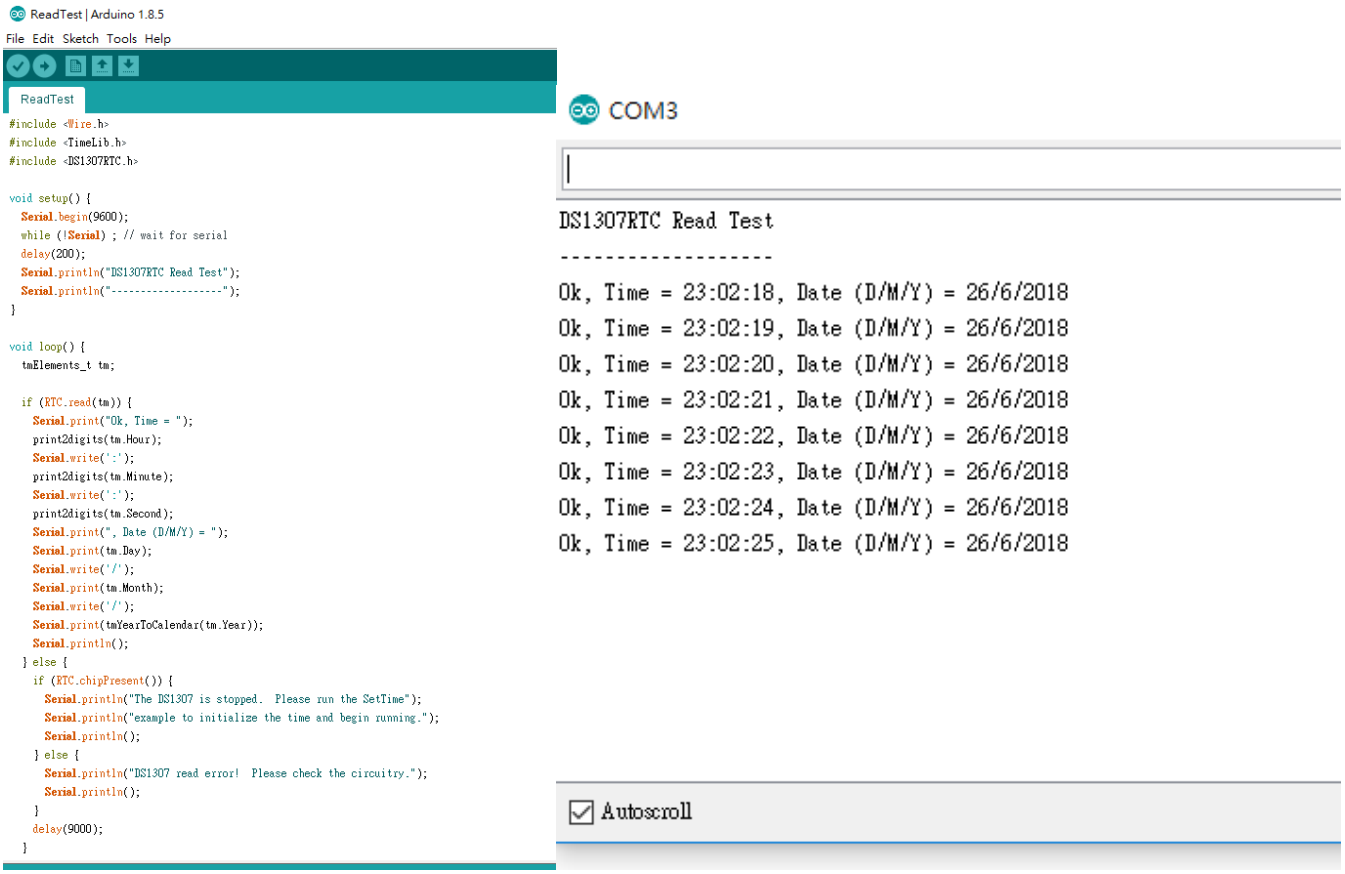


6.2 Choosing an Example file in RTC library

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



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



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:

  if (RTC.read(tm)) {
    Serial.print("Ok, Time = ");
    Serial.print(tm.Hour);
    Serial.write(':');
    Serial.print(tm.Minute);
    Serial.write(':');
    Serial.print(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();
    }
}
```

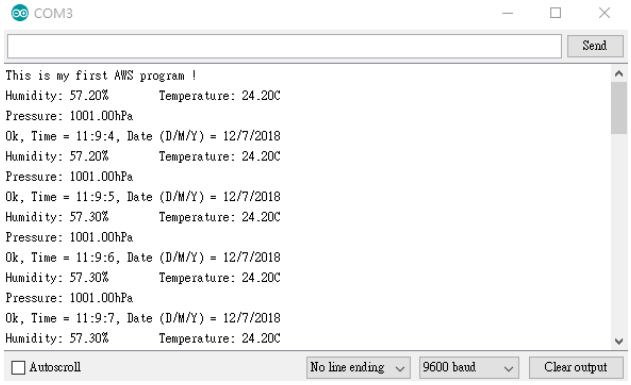
ReadTest | Arduino 1.8.5

File Edit Sketch Tools Help

ReadTest

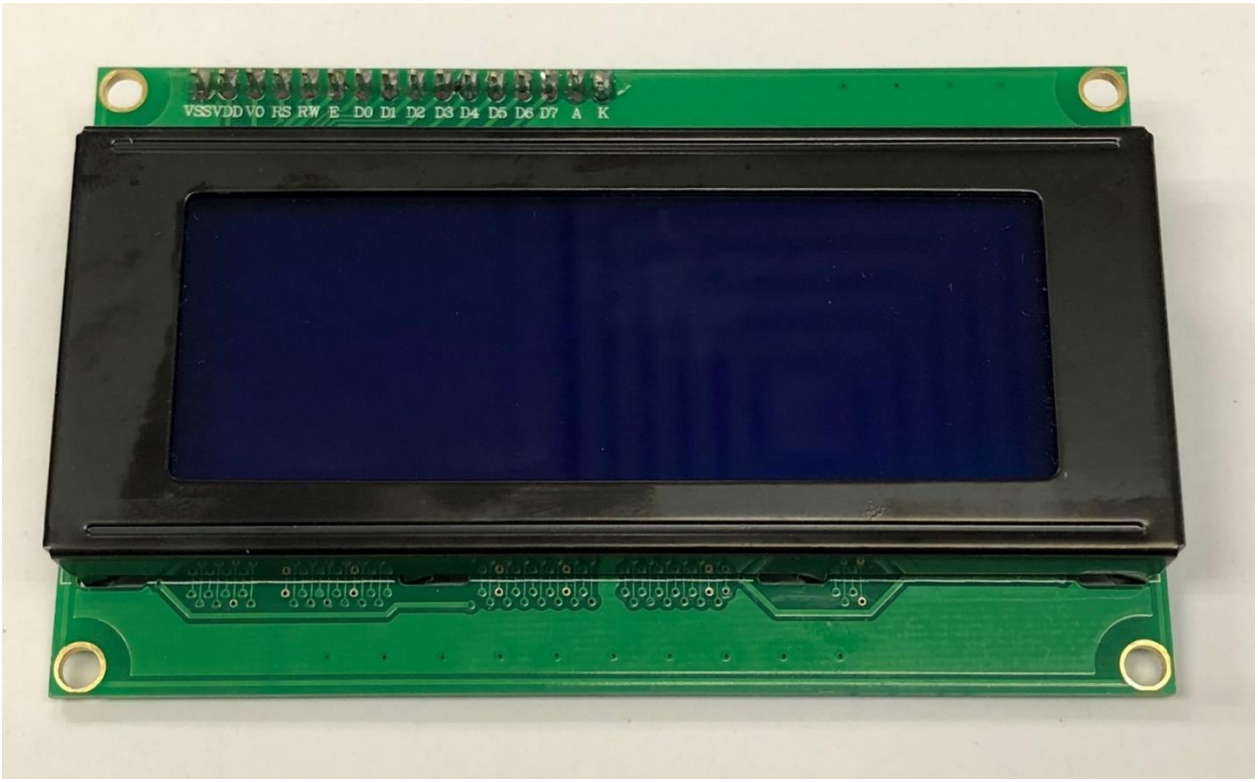
```
#include <Wire.h>
#include <TimeLib.h>
#include <DS1307RTC.h>

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!

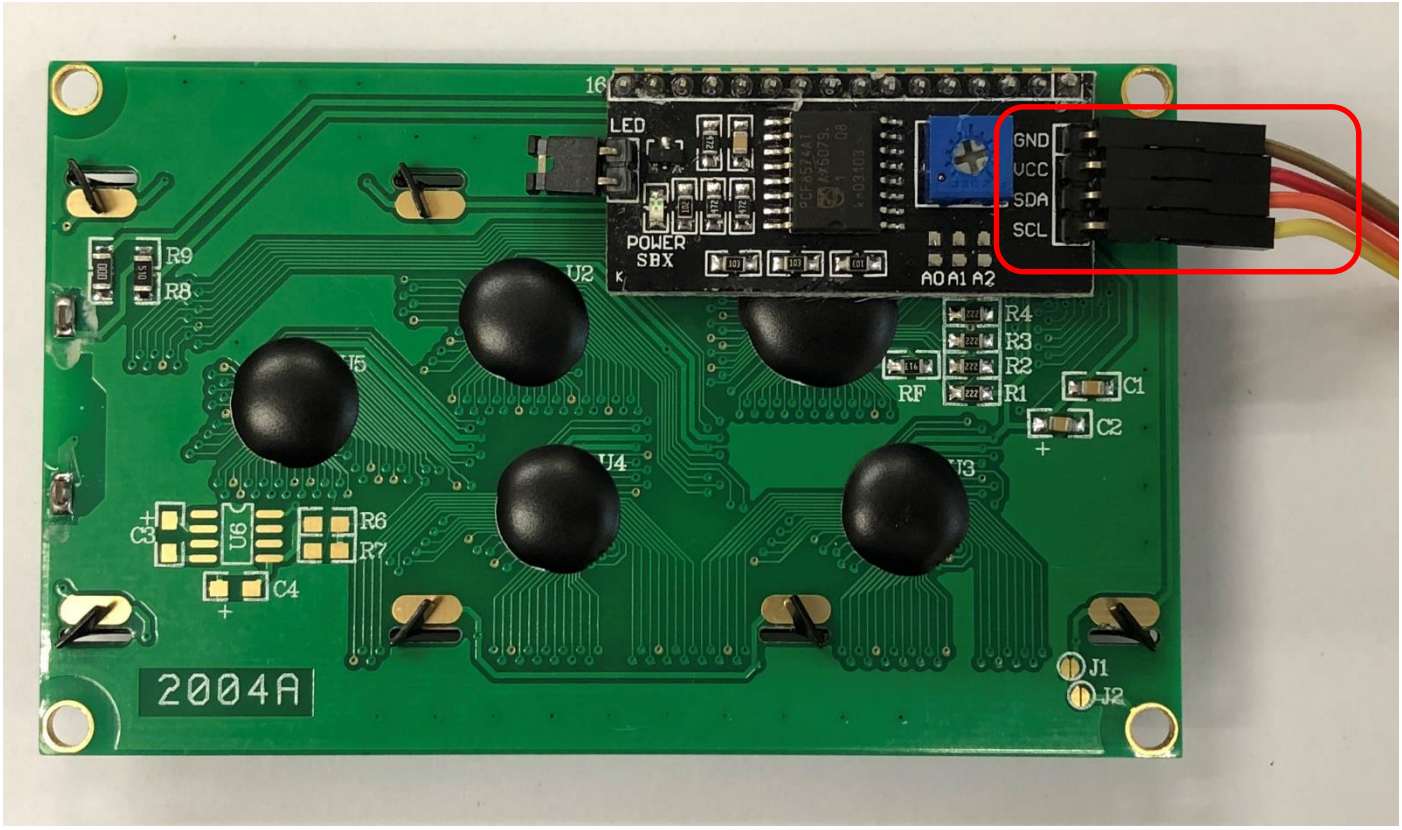


(7) Installing a LCD Display

Front view

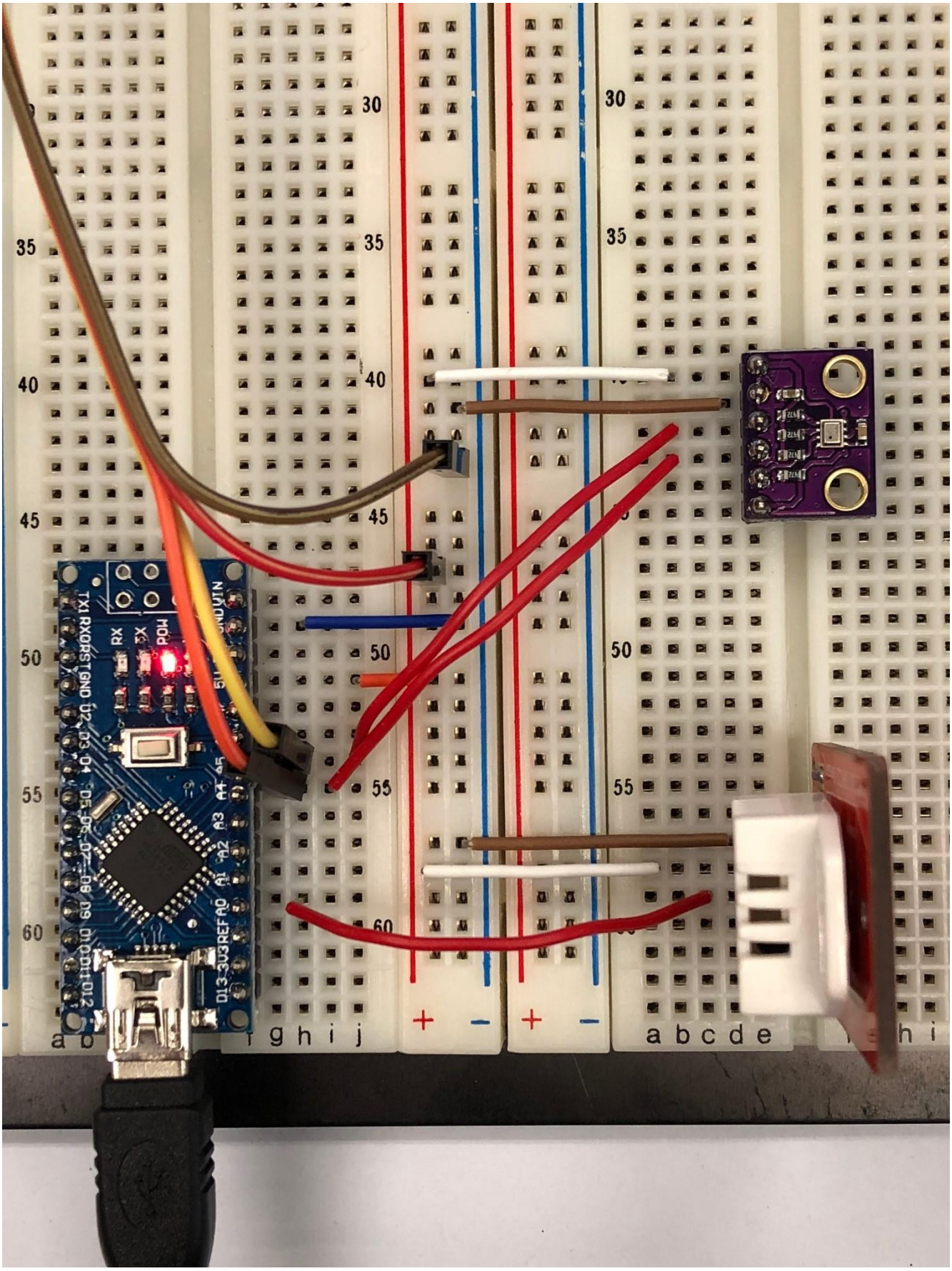


Rear view



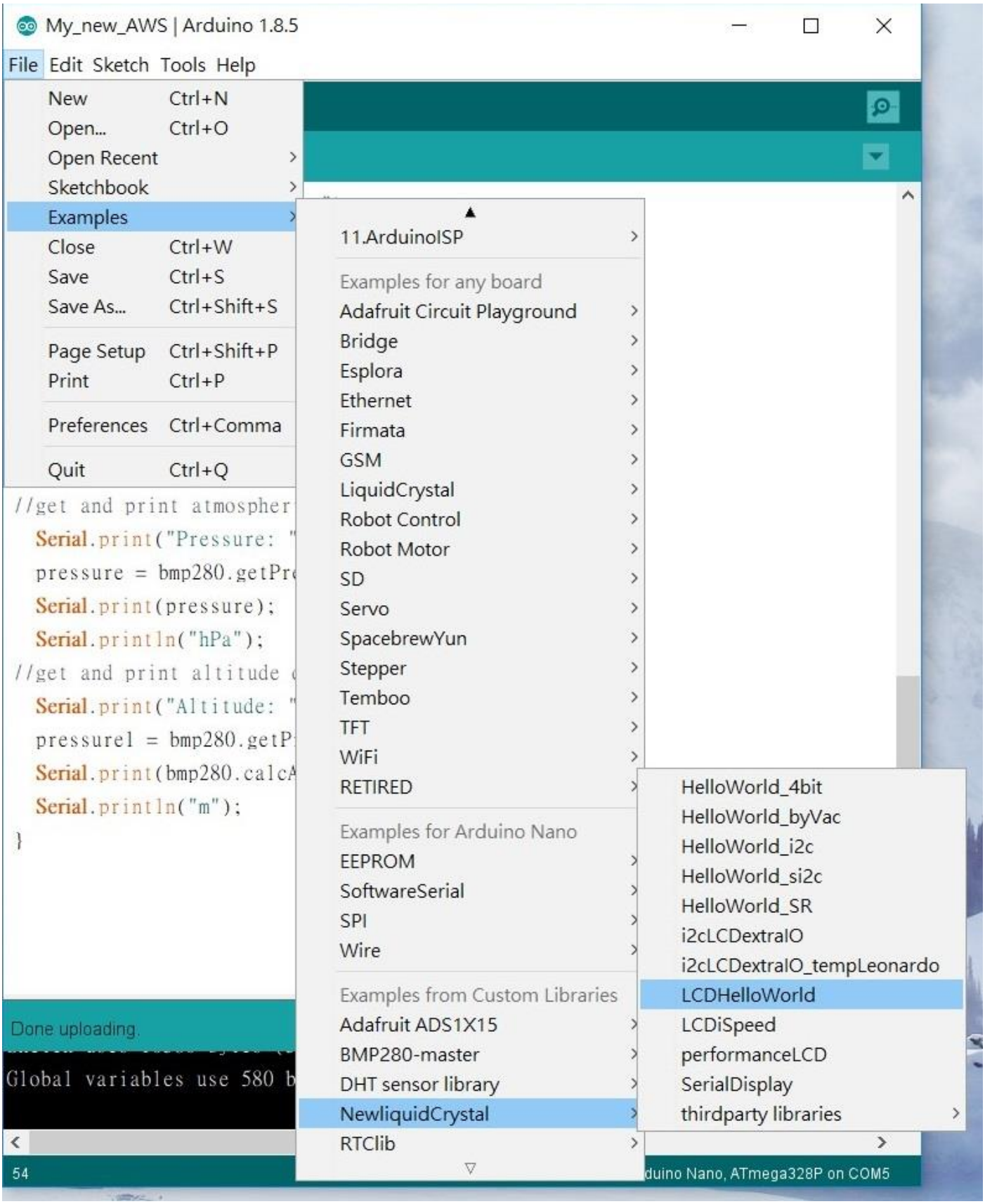
7.1 Wiring

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

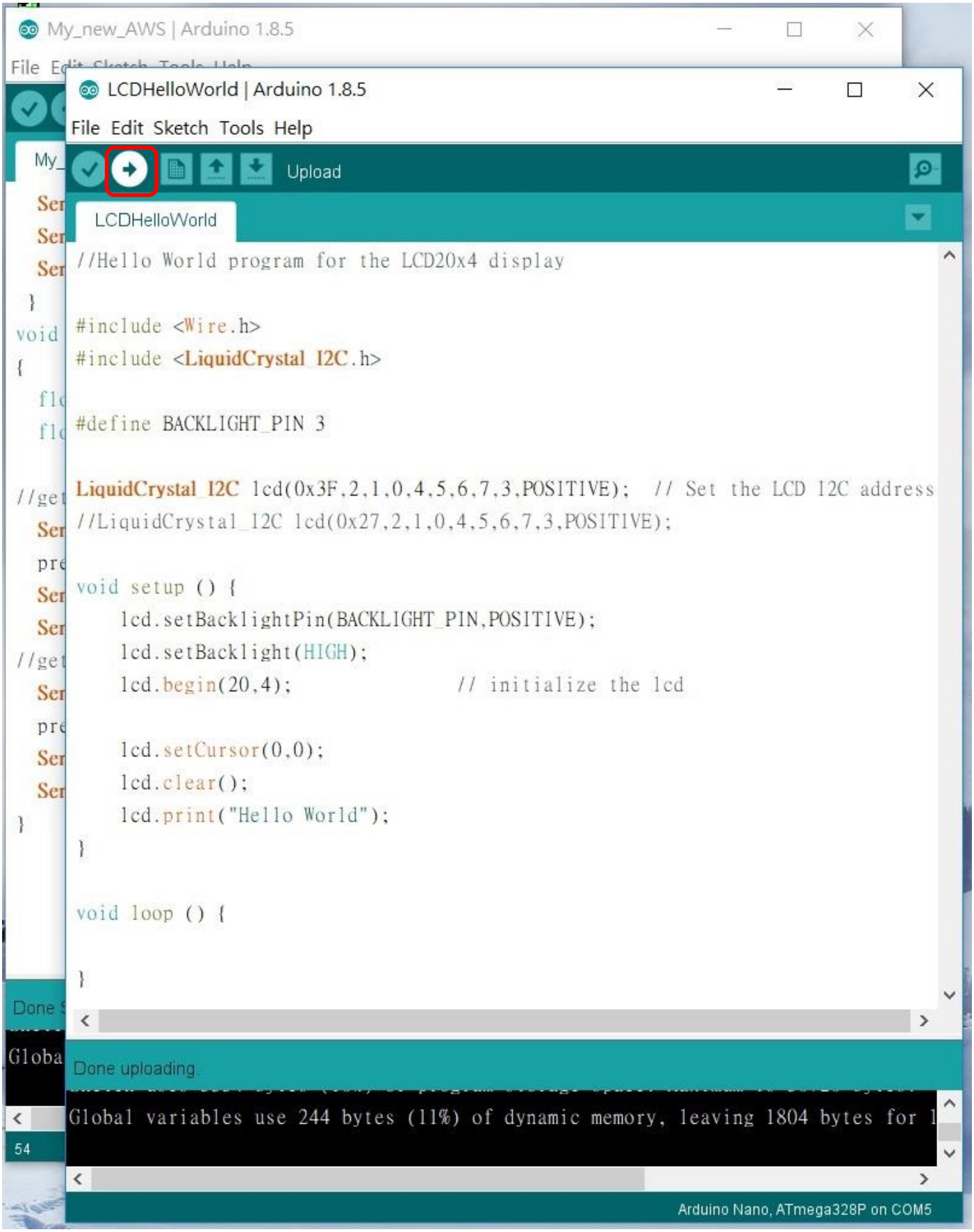


7.2 Choosing an Example file in the NewliquidCrystal library

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

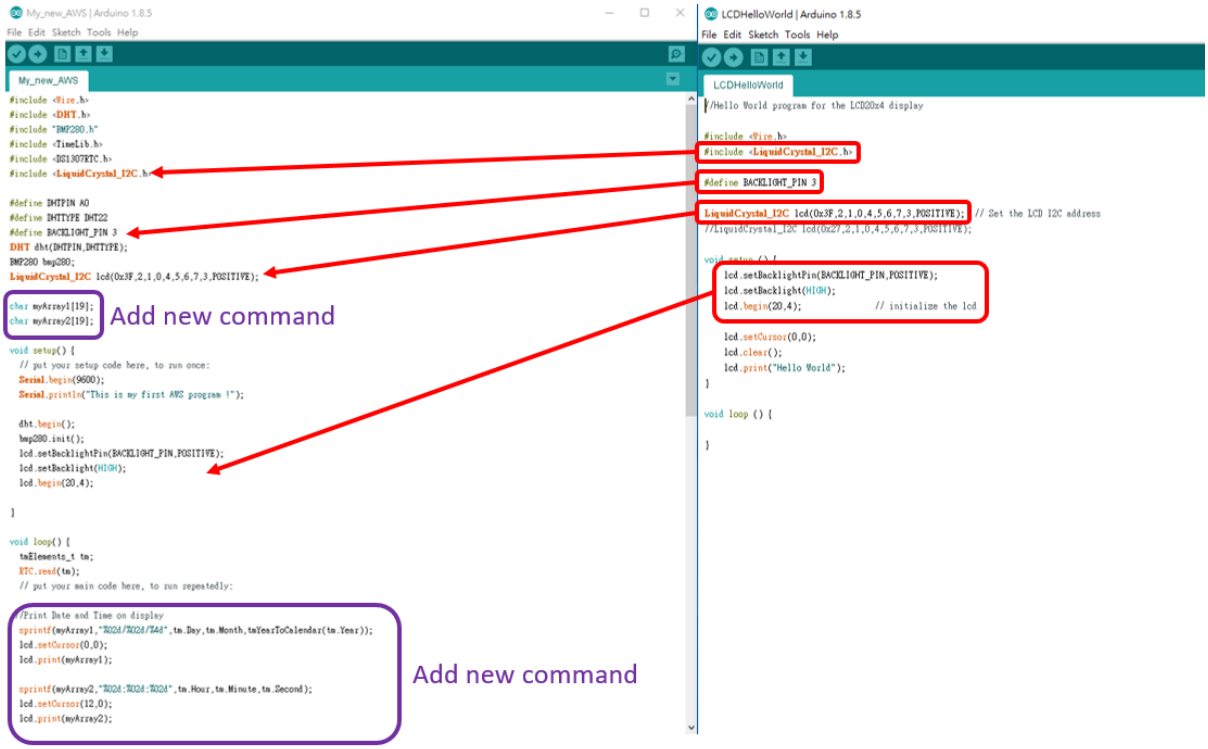


Compile and upload the LCDHelloWorld program to Arduino Nano and see the output shown on the LCD display.



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 LCDHellowWorld 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 §

```
  tmElements_t tm;
  RTC.read(tm);
  // put your main code here, to run repeatedly:

 //Print Date and Time on display
  sprintf(myArray1,"%02d/%02d/%4d",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: ");
```

// Integrate the abbreviation setting on display

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

// Showing temperature info. on display

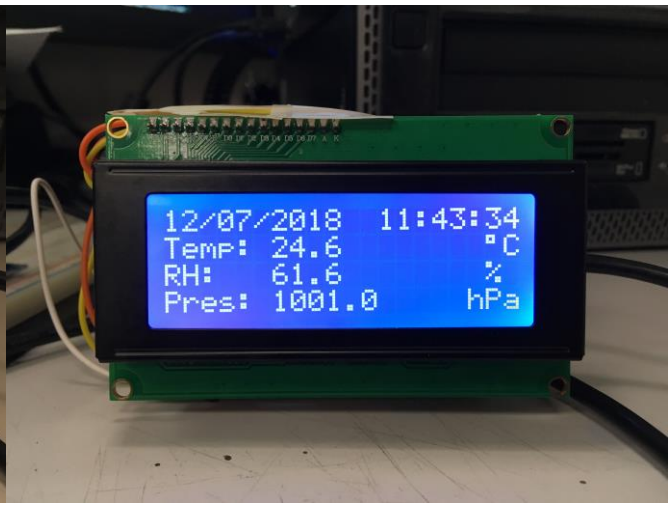
```
//Print Humidity information on display
  float h = dht.readHumidity();
  lcd.setCursor(6,2);
  lcd.print(h,1);
  lcd.setCursor(18,2);
  lcd.print("%");
```

// Showing humidity info. on display

```
//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);
}
```

// 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!