



**USER GUIDE**  
**BASIC ARDUINO TRAINING**  
**(Exercise: Automatic Weather Station)**

Version : **1.0**  
Date : **September, 2019**  
Prepared by : **Jeffrey Chang - Intern (R3/F3)**

Hong Kong Observatory  
© The Government of the Hong Kong Special Administrative Region

The contents of this document remain the property of and may not be reproduced in whole or in part without the express permission of the Hong Kong Observatory

## **TABLE OF CONTENTS**

- 1. INTRODUCTION**
- 2. GETTING STARTED**
- 3. CREATING YOUR FIRST ARDUINO PROJECT**
- 4. INSTALLING TEMPERATURE AND HUMIDITY SENSOR  
- DHT22 MODULE**
- 5. INSTALLING PRESSURE SENSOR  
- BMP280 MODULE**
- 6. INSTALLING REAL-TIME CLOCK  
- RTC1307 MODULE**
- 7. INSTALLING LCD DISPLAY  
- LCD I2C 2004A DISPLAY**
- 8. SENSORS COMPARISON**

## **1. INTRODUCTION**

### **1.1 What is Arduino**

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of **digital** and **analog input/output (I/O)** pins that may be interfaced to various expansion boards or breadboards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers can be programmed using **C and C++ programming languages**. In addition to using traditional compiler toolchains, the Arduino project provides an **integrated development environment (IDE)** based on the Processing language project.

### **1.2 Digital Signals**

Digital pins are used mainly as **output pins**. You can connect various devices (LEDs, buzzer, LCD) on the digital pins and turn them on/off by **writing HIGH or LOW on the respective pins**. Check out digitalWrite() command. Before using a digital pin, you have to set it in input/output mode using pinMode() command.

You can use a digital pin in input mode as well, especially when using interrupts in your code. Digital pins 2 and 3 of UNO can be used for interrupt detection. In general, you can check the status (on or off) of a device (or a switch) connected to a digital pin by using that pin in input mode. The command used is digitalRead().

#### **Applications: LED Flash, LCD Monitor, Fan (On/Off)**

### **1.3 Analog Signals**

Analog pins are the ADC (analog to digital converter) input pins. They are used for **reading analog voltage** (between 0-5V on arduino, by default). Check out the sample program for analogRead() command.

#### **Applications: Sensors (Signal by measuring voltage difference)**

**Two type of Electromagnetic Signals**



Analog  
Analog signal is a continuous wave denoted by a sine wave and may vary in signal strength (amplitude) or frequency (time).

Digital  
A digital signal is described as using binary (0s and 1s), and therefore, cannot take on any fractional values. This kind of signal denoted by digits that's why it is called Digital Signal.

Figure 1. Analog and Digital Signals

## 1.4 I2C Communication

The I2C communication bus is very popular and broadly used by many electronic devices because it can be easily implemented in many electronic designs which require communication between a **master** and **multiple slave devices or even multiple master devices**. Each device has a preset ID or a unique device address so the master can choose with which devices will be communicating.

The two wires, or lines are called **Serial Clock (or SCL – Pin A4)** and **Serial Data (or SDA – Pin A5)**. The SCL line is the clock signal which synchronize the data transfer between the devices on the I2C bus and it's generated by the master device. The other line is the SDA line which carries the data.

The two lines are “open-drain” which means that pull up resistors needs to be attached to them so that the lines are high because the devices on the I2C bus are active low.

### Applications: Real-time Clock, Sensors

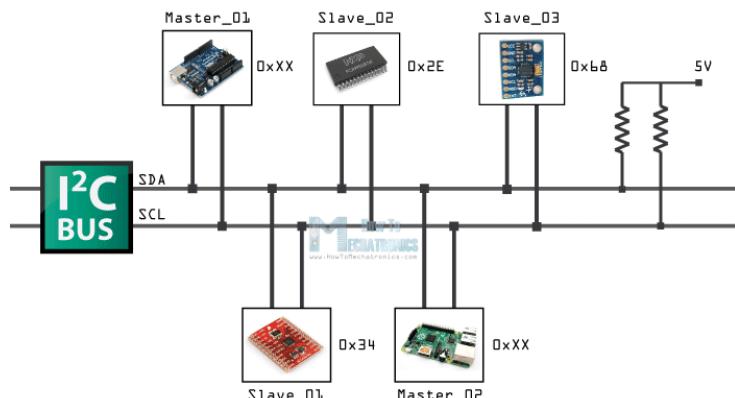
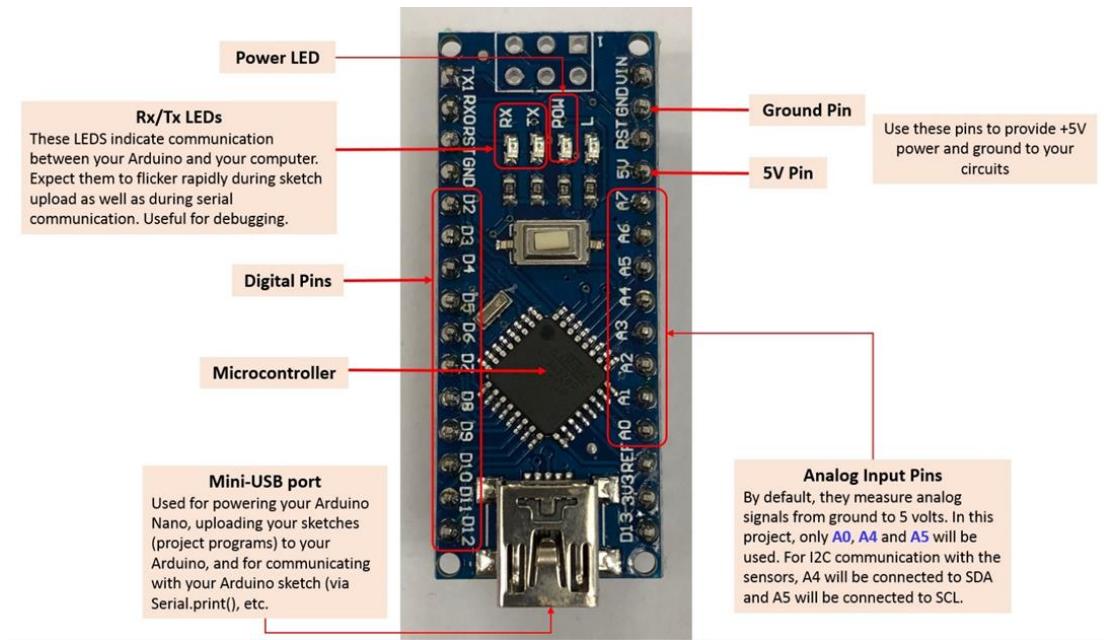


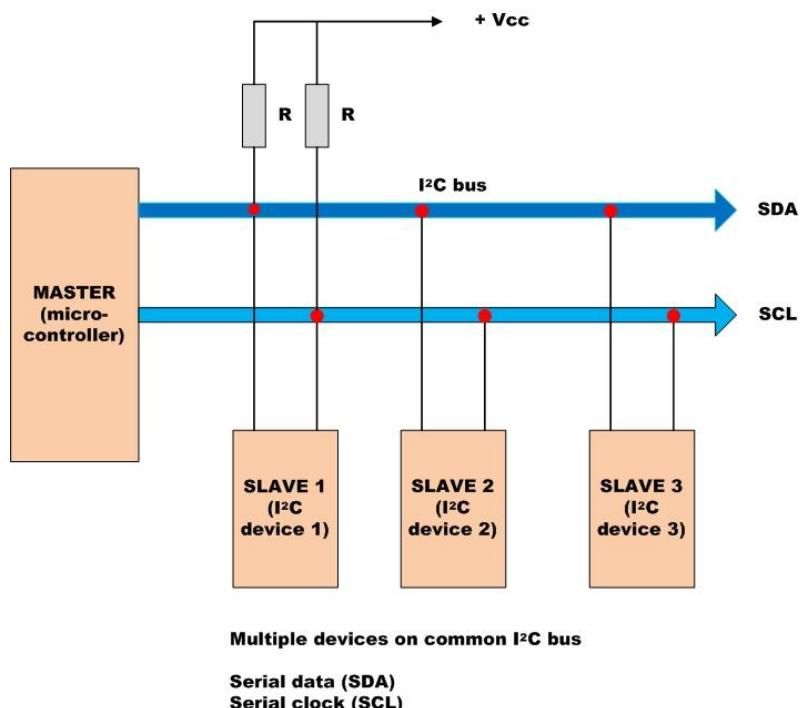
Figure 2. I2C Communication Example

## 2. GETTING STARTED

### 2.1 The Arduino Nano Board



### I<sup>2</sup>C Architecture



## 2.2 The Arduino IDE

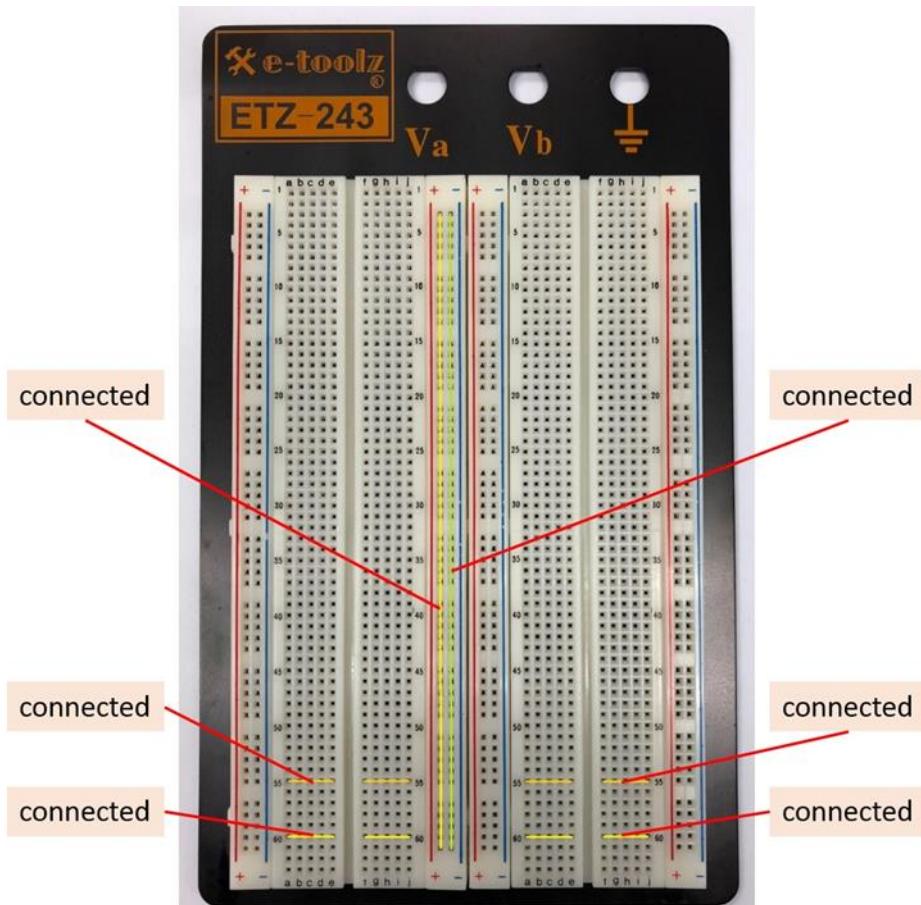
Before you start controlling the world around you using the Arduino, you'll need to download the Arduino IDE (Integrated Development Environment). The Arduino IDE allows you to write programs and upload them to your Arduino. You can download the latest version of the IDE from:

<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).

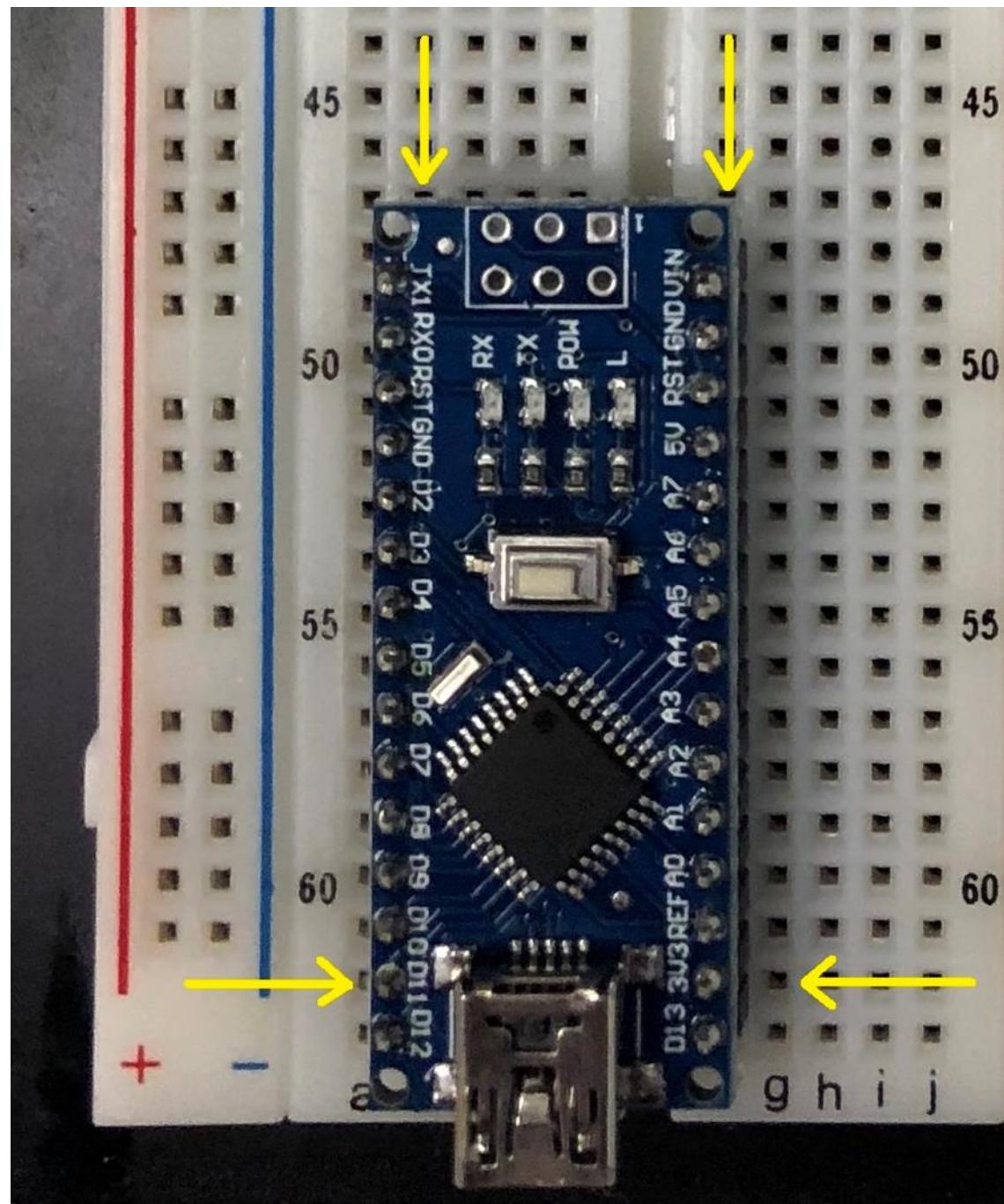


## 2.3 Basic information about the Breadboard used in this workshop



## 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



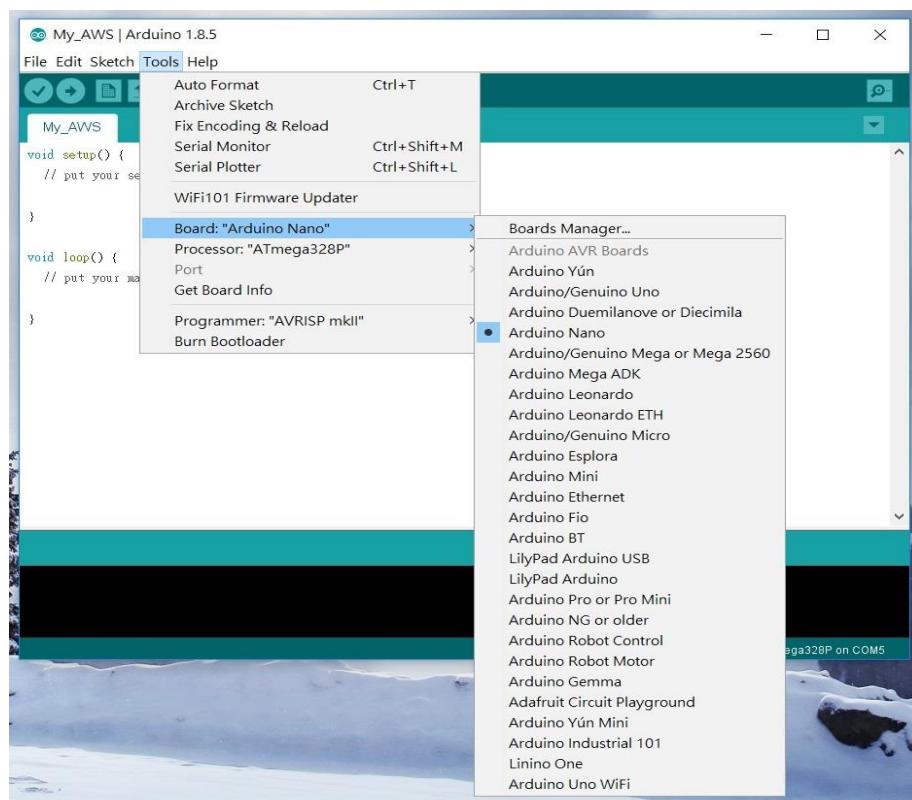
### 3. CREATING YOUR FIRST ARDUINO PROJECT

#### 3.1 Running the Arduino IDE

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



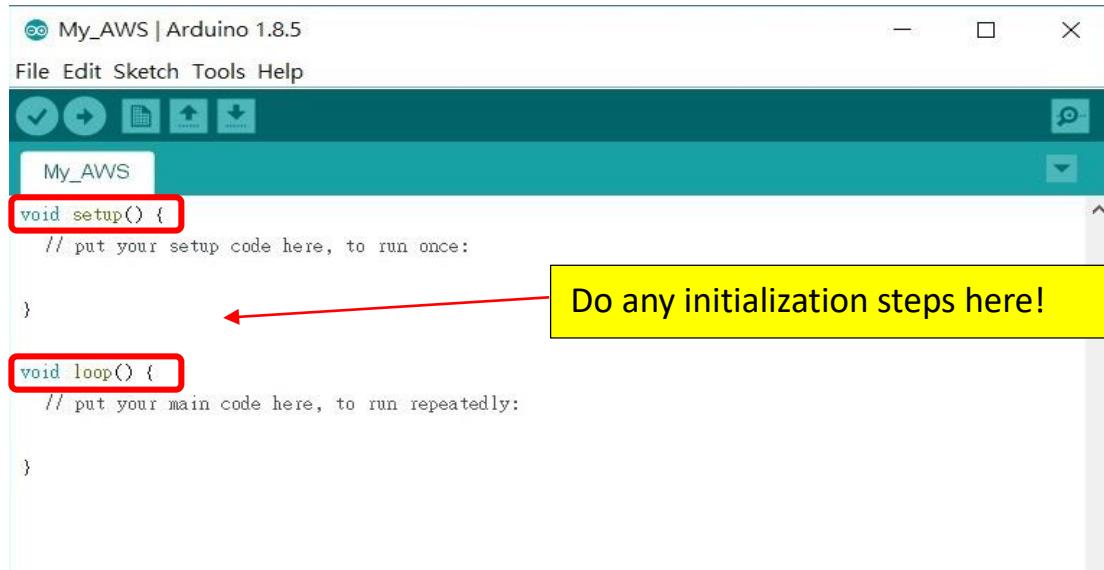
#### 3.2 Selecting a proper Arduino board



#### 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 titled "My\_AWS". The code structure is as follows:

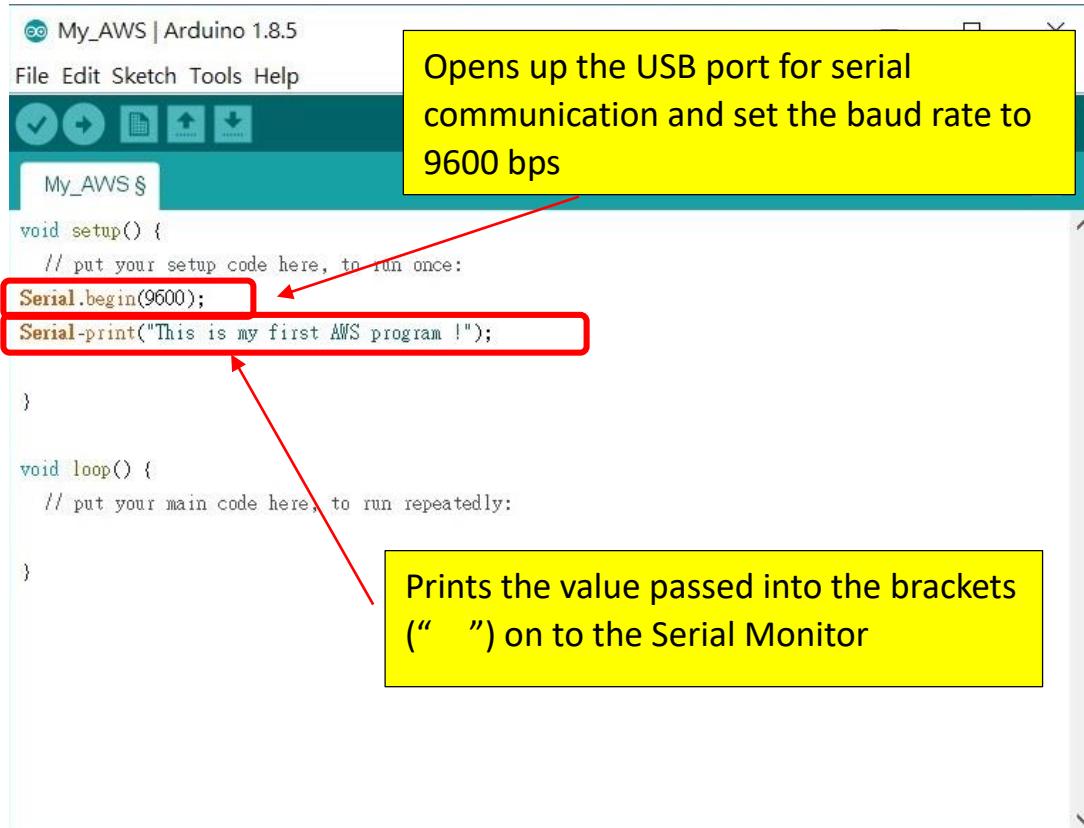
```
void setup() {
    // put your setup code here, to run once:
}

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

A red box highlights the `void setup()` block, and a yellow callout box with a red arrow points to it, containing 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 completed sketch "My\_AWS". The code now includes the following:

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

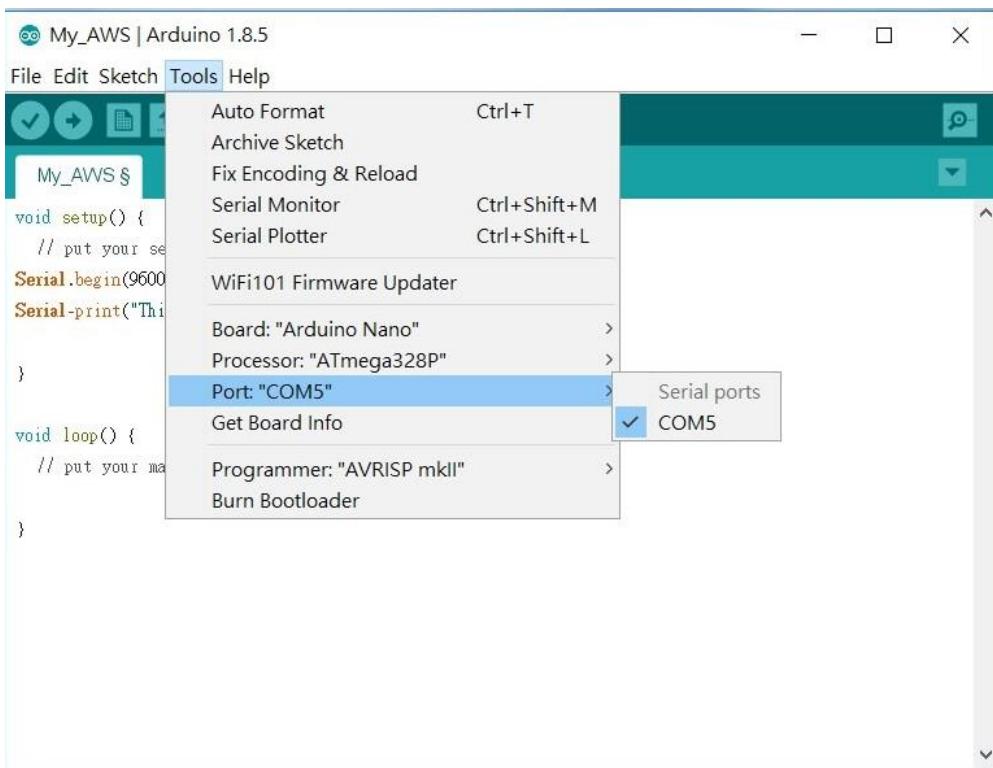
void loop() {
    // put your main code here, to run repeatedly:
}
```

Two yellow callout boxes with red arrows point to specific parts of the code:

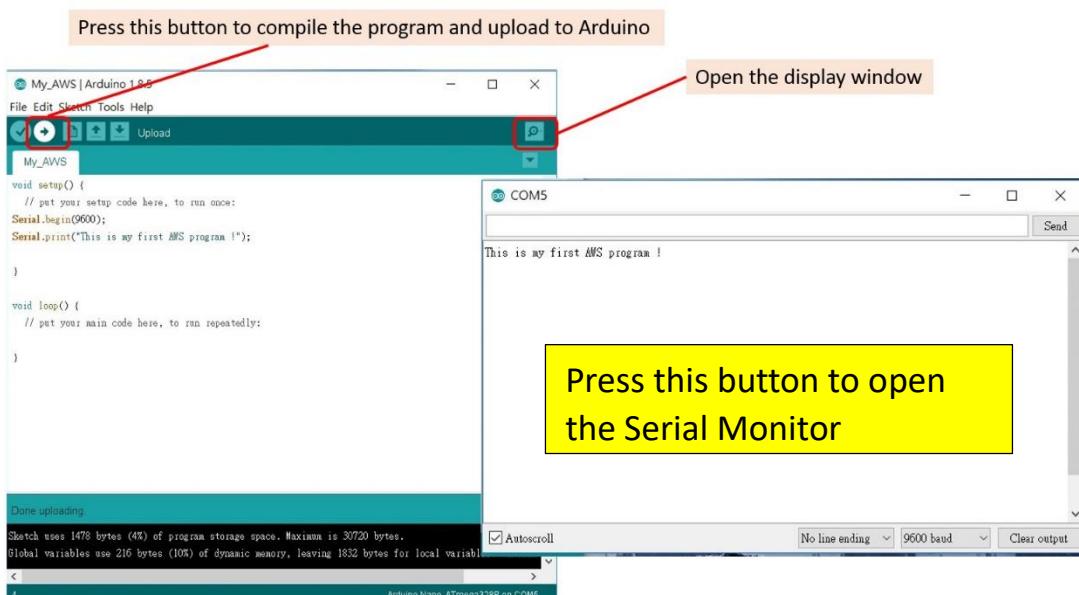
- The first callout box points to the `Serial.begin(9600);` line and contains the text: "Opens up the USB port for serial communication and set the baud rate to 9600 bps".
- The second callout box points to the `Serial.print(" This is my first AWS program! ");` line and contains the text: "Prints the 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.

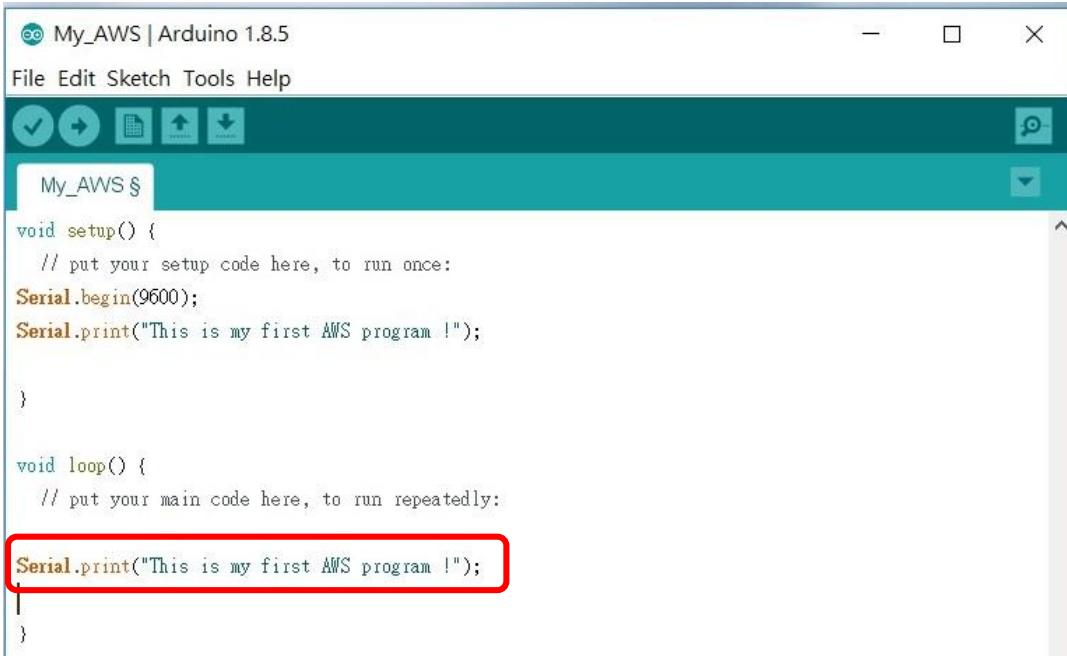


### 3.5 Compiling and uploading a program to your Arduino



### 3.6 Using the loop() function

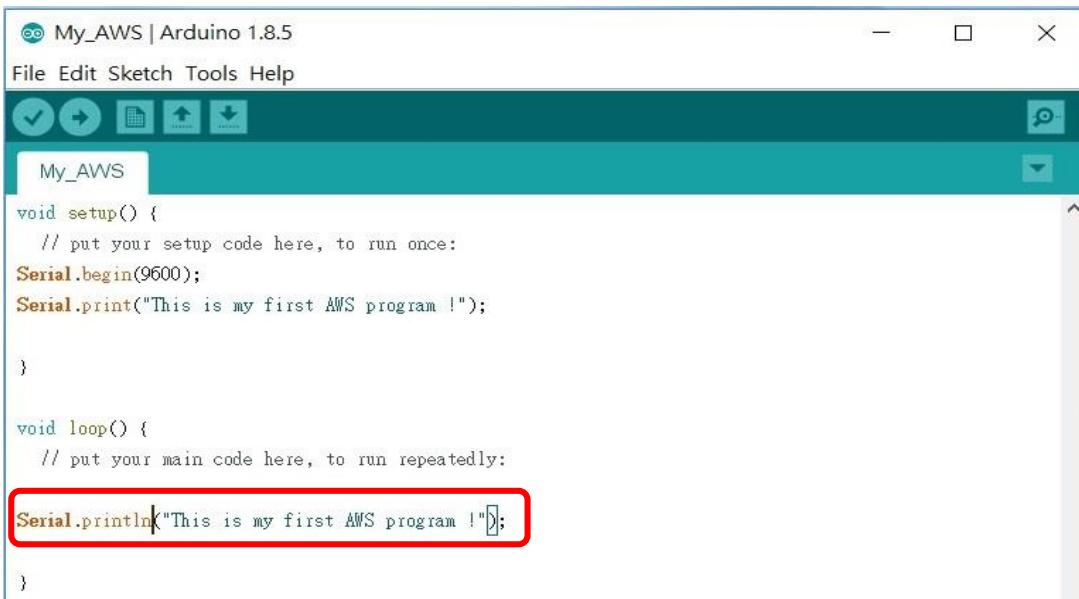
In the loop() section, type `Serial.print("This is my first AWS program! ")` and see the result.



```
My_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_AWS §
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.print("This is my first AWS program !");
}

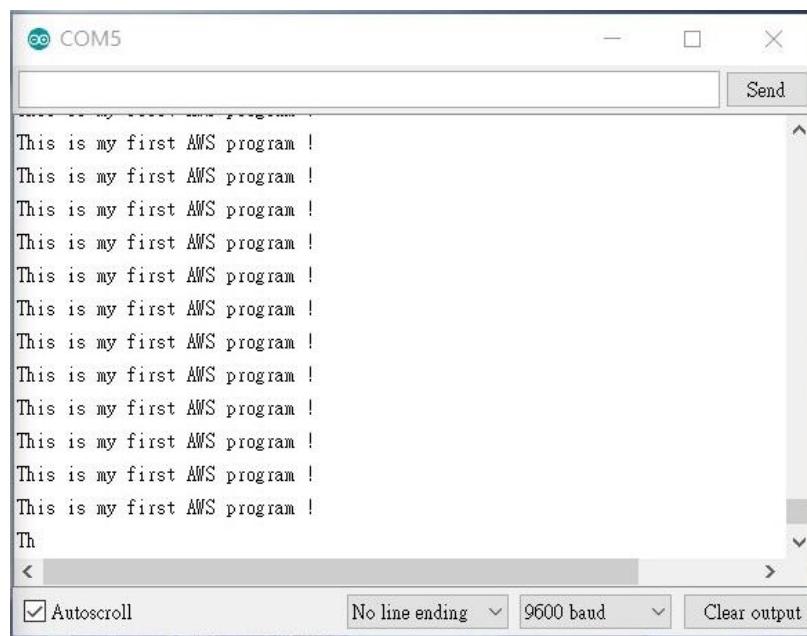
void loop() {
    // put your main code here, to run repeatedly:
    Serial.print("This is my first AWS program !");
}
```

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.



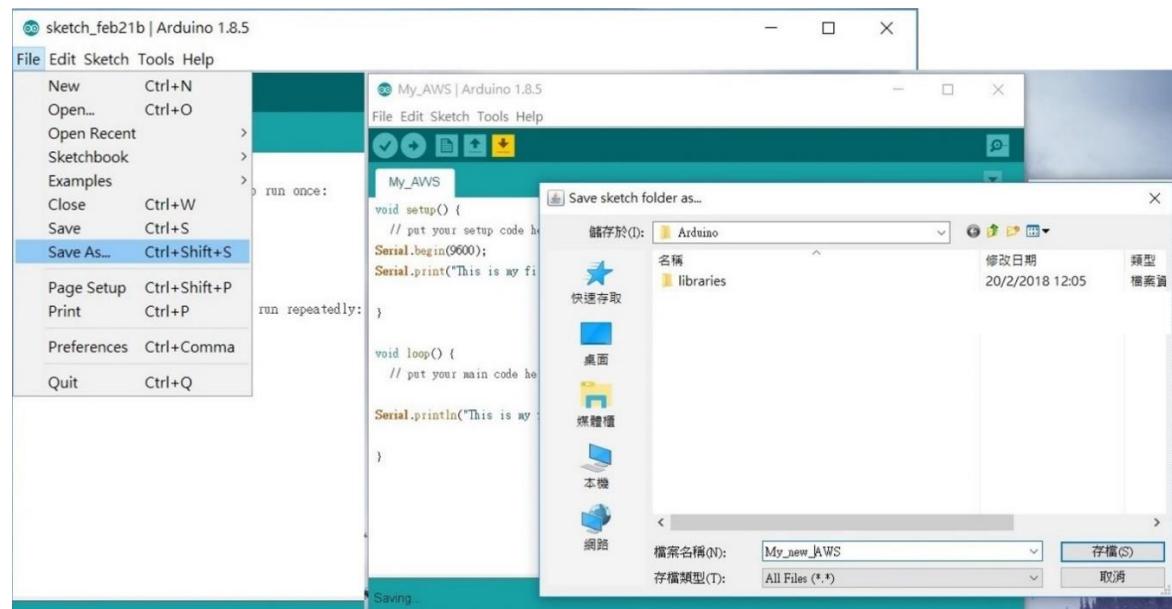
```
My_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_AWS
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.print("This is my first AWS program !");
}

void loop() {
    // put your main code here, to run repeatedly:
    Serial.println("This is my first AWS program !");
}
```



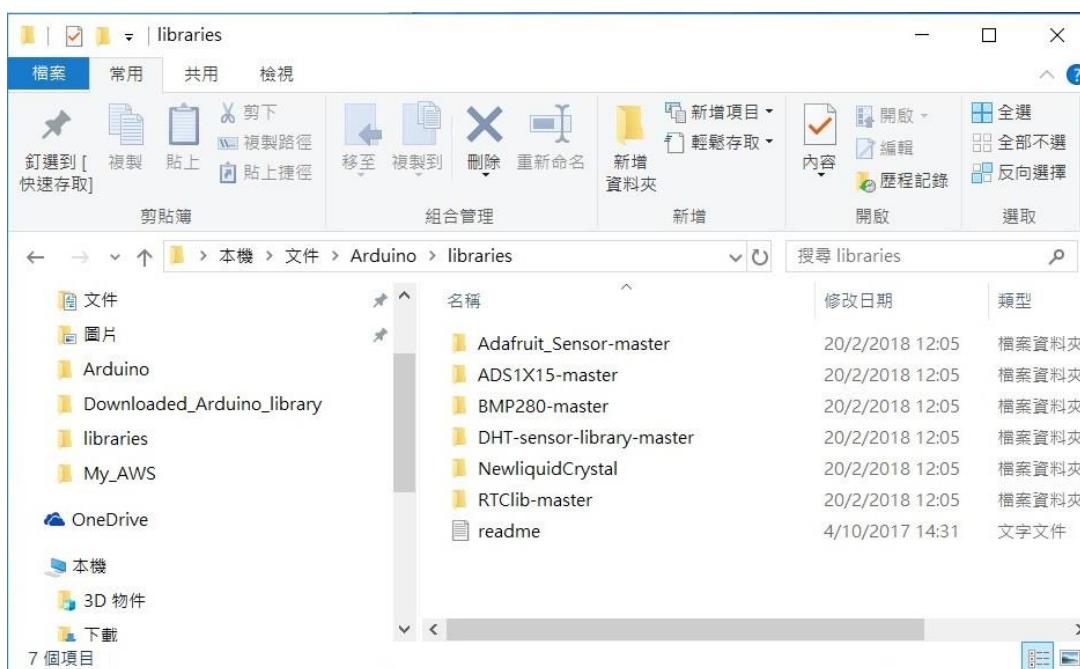
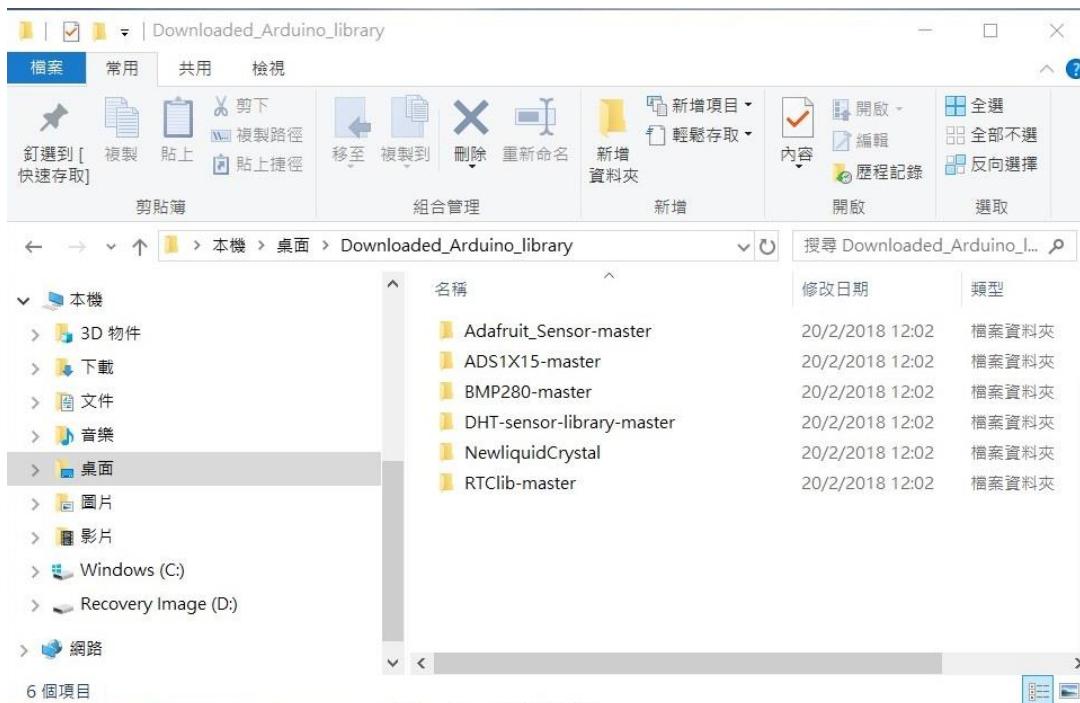
### 3.7 Saving your first project file

Save the program as My\_new\_AWS.



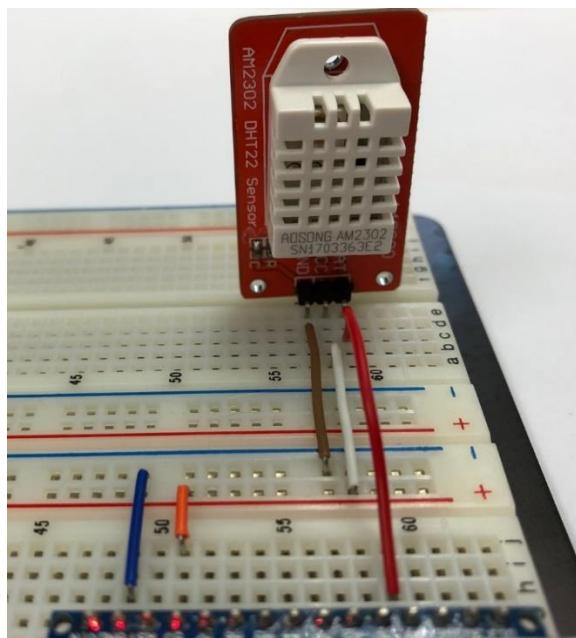
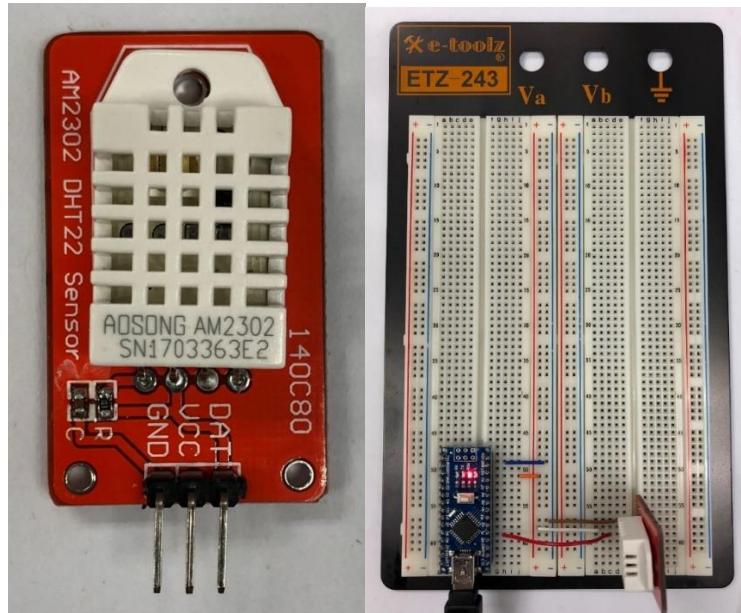
### 3.8 Downloading library files for different sensors for your Arduino projects

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:



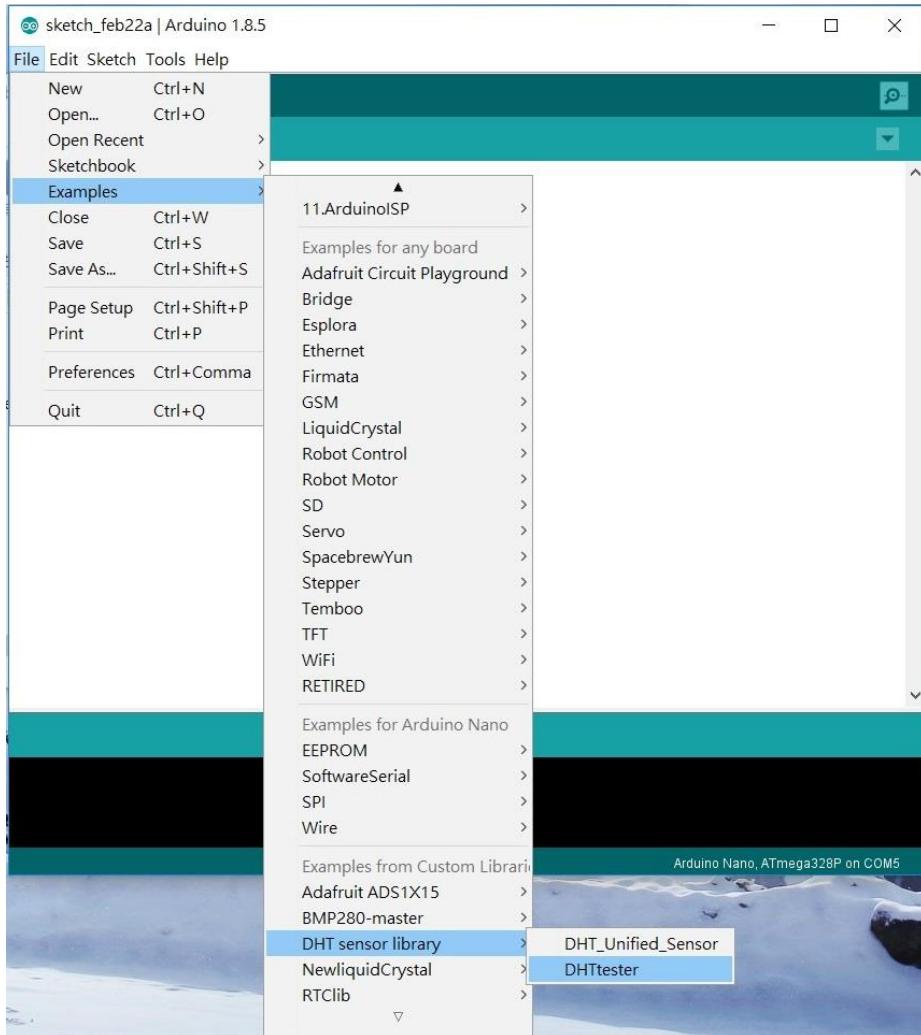
**4. INSTALLING TEMPERATURE AND HUMIDITY SENSOR****- DHT22 MODULE****4.1 Wiring**

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:

```

// 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)

```

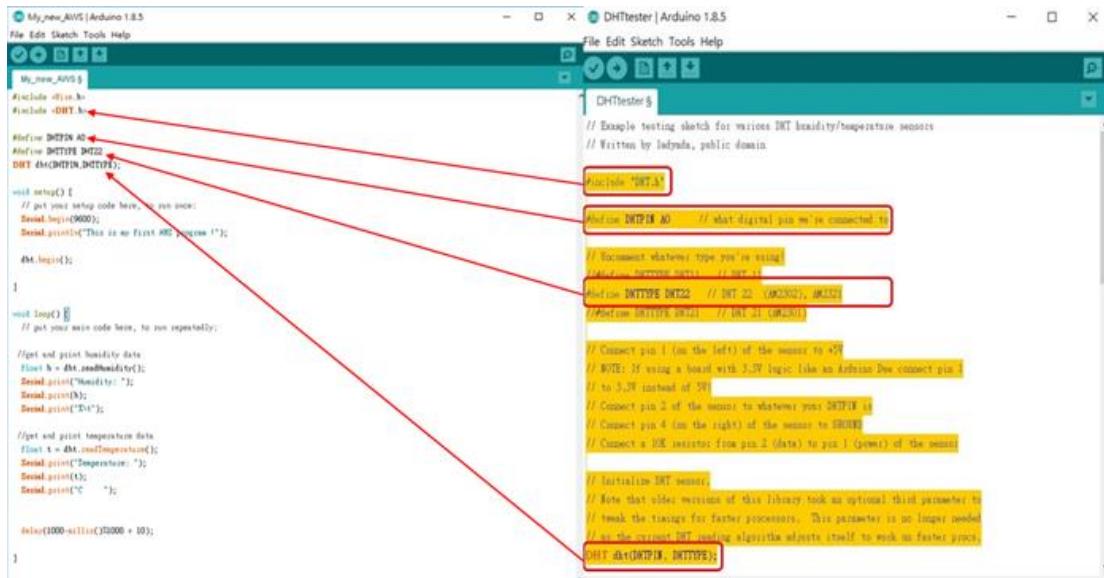
Compile and upload the program again and see the results.

#### 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 DHTester file.

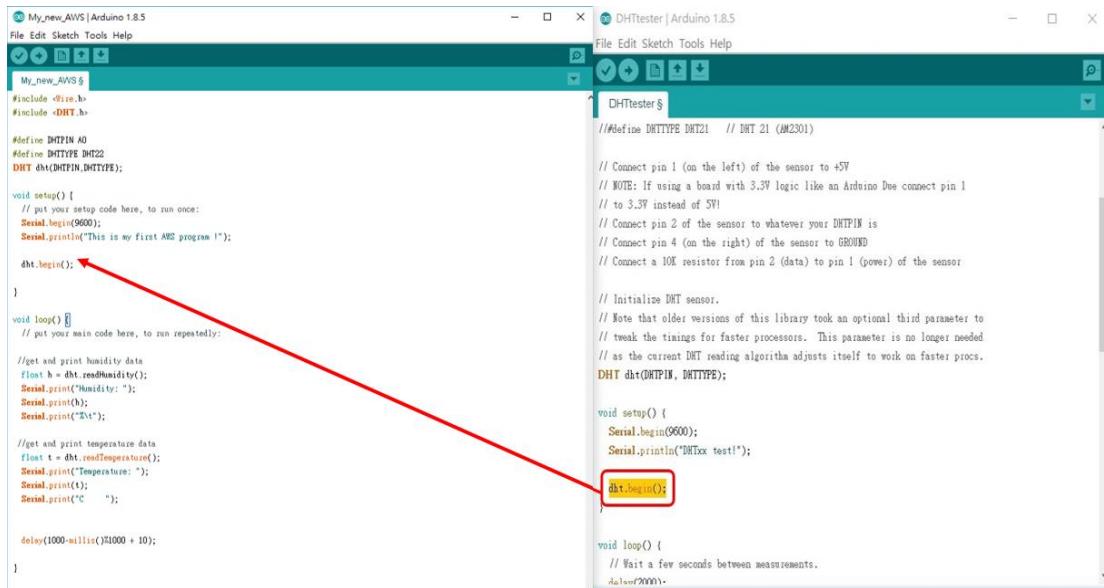
Copy the following 4 lines to header section of My\_new\_AWS program:

```
#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:



## USER GUIDE

### BASIC ARDUINO TRAINING (Exercise: Automatic Weather Station)

## CONTENTS

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("%\n");

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

    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 (if Fahrenheit = 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 best index in Fahrenheit (the default)
    float hF = dht.computeRHIndex(f, h);
    // Compute best index in Celsius (if calculate = false)
    float hC = dht.computeRHIndex(t, h, false);

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

```

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();
}

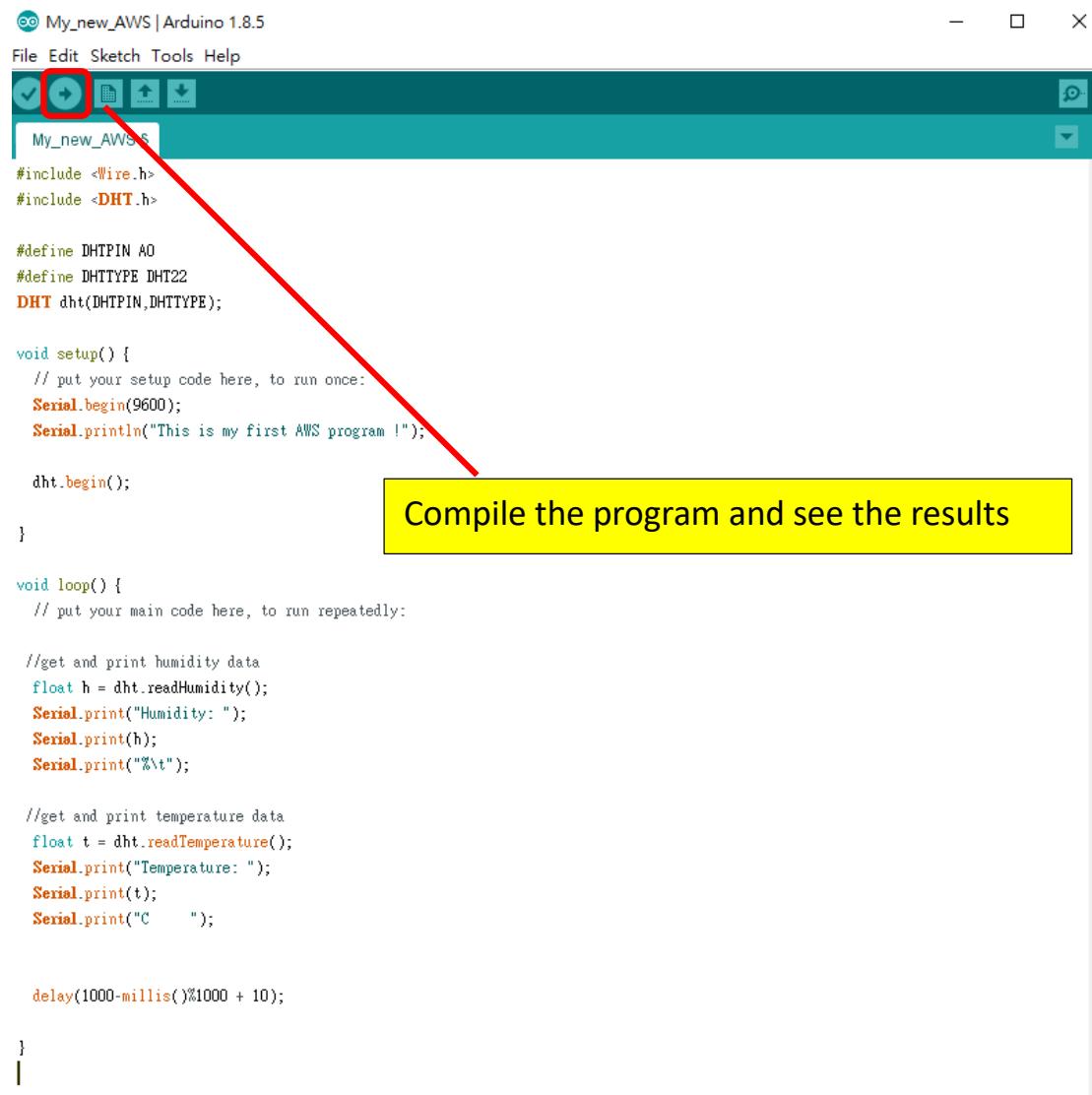
void loop() {
    // put your main code here, to run repeatedly:

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

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

    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:

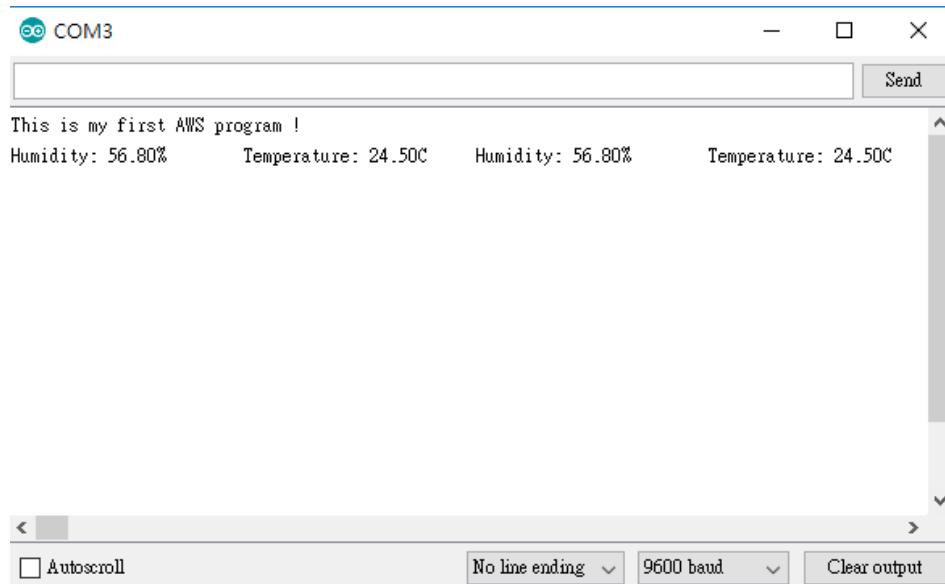
    //get and print humidity data
    float h = dht.readHumidity();
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print("%\t");

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

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

}
```

Compile the program and see the results



COM3

This is my first AWS program !

Humidity: 56.80%      Temperature: 24.50C      Humidity: 56.80%      Temperature: 24.50C

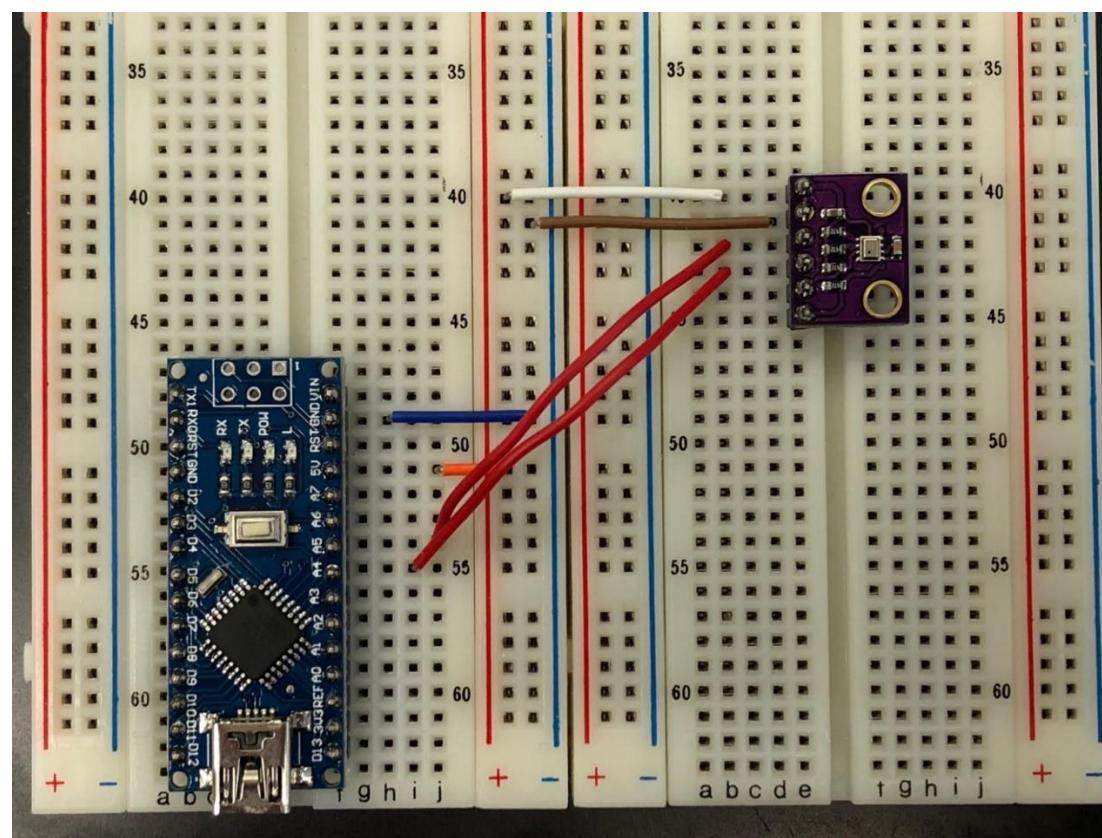
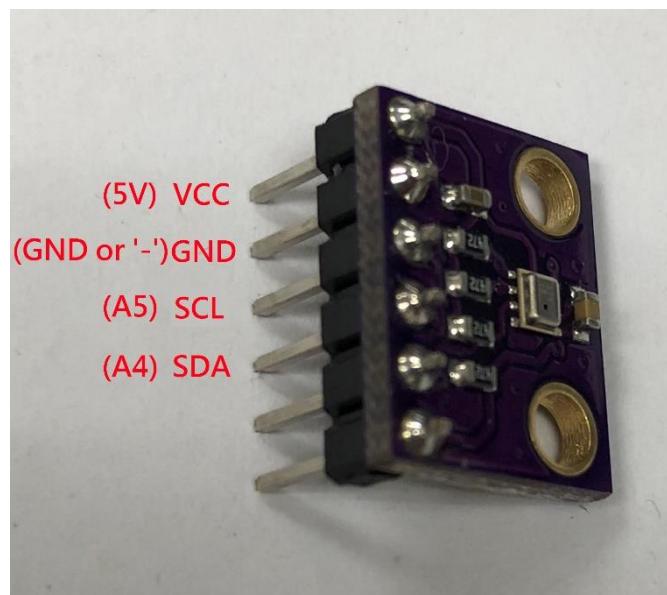
< >

Autoscroll      No line ending      9600 baud      Clear output

## 5. INSTALLING PRESSURE SENSOR - BMP280 MODULE

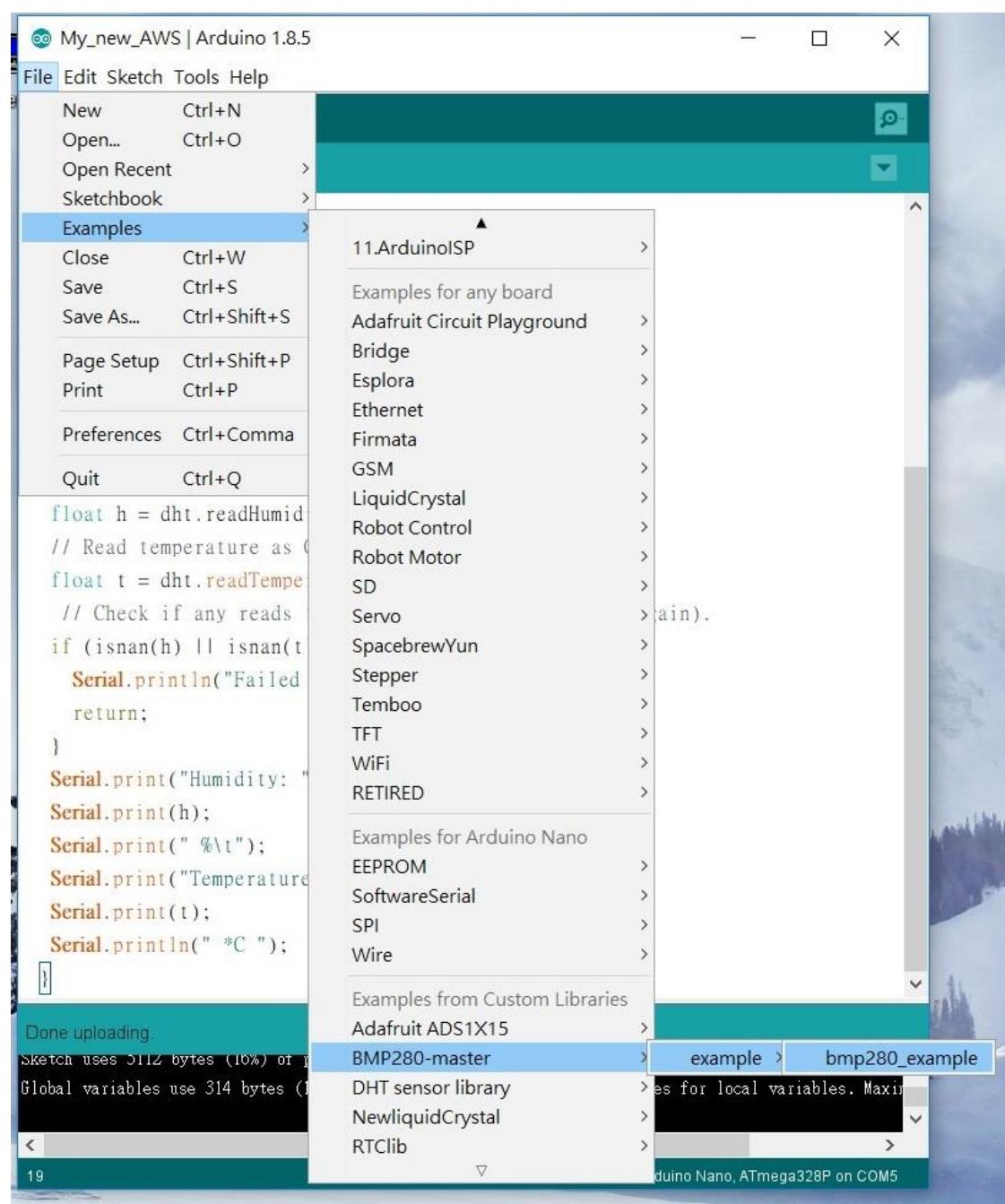
### 5.1 Wiring

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

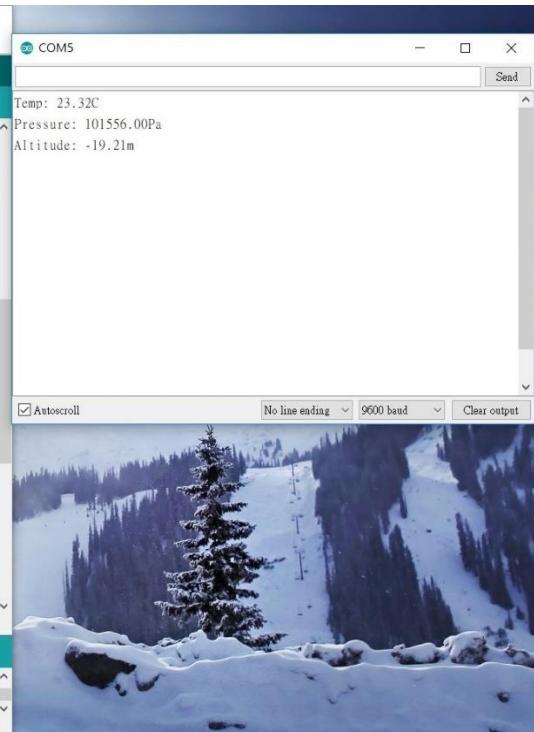
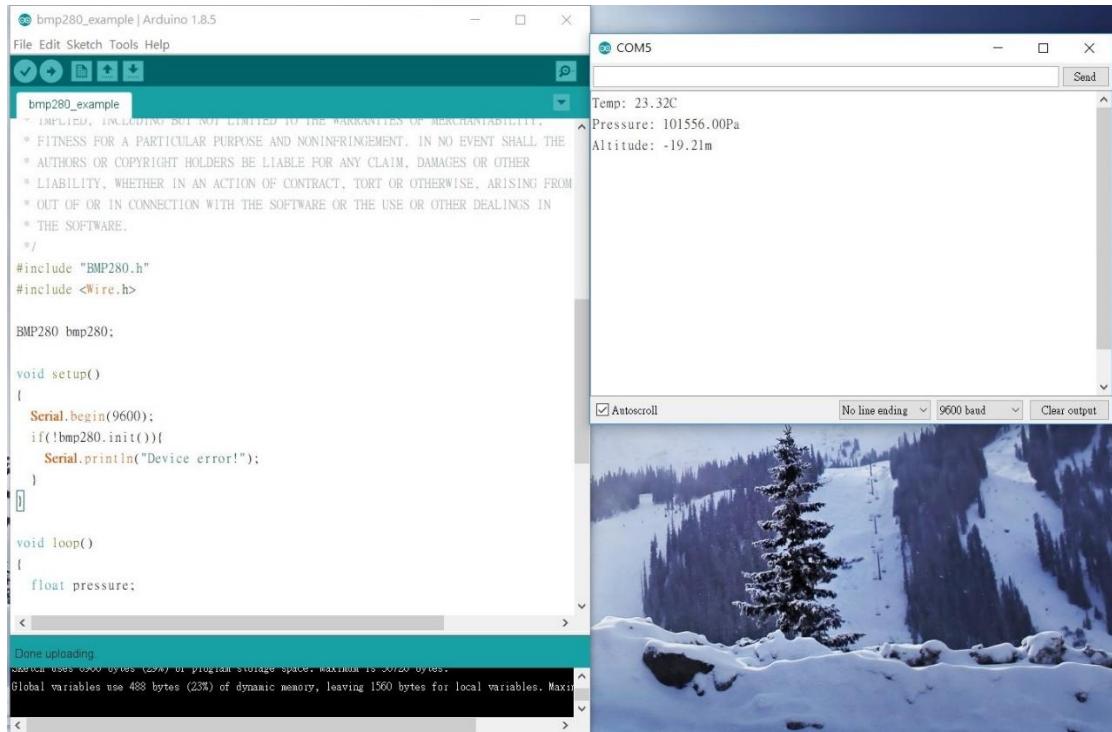
Choose the `bmp280_example` file from the BMP280-master library as shown below:



## USER GUIDE BASIC ARDUINO TRAINING (Exercise: Automatic Weather Station)

## CONTENTS

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.

```


My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
#include <Wire.h>
#include <DHT.h>
#include "BMP280.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(); // Red arrow points here
}



```

```


bmp280_example | Arduino 1.8.5
File Edit Sketch Tools Help
bmp280_example
  * THE SOFTWARE.
  */
#include "BMP280.h"
#include <Wire.h>

BMP280 bmp280;

void setup()
{
  Serial.begin(9600);
  if(!bmp280.init()){
    Serial.println("Device error!");
  }
}


```

## USER GUIDE BASIC ARDUINO TRAINING (Exercise: Automatic Weather Station)

## CONTENTS

```
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()%100+10);

}

void loop()
{
    float pressure;

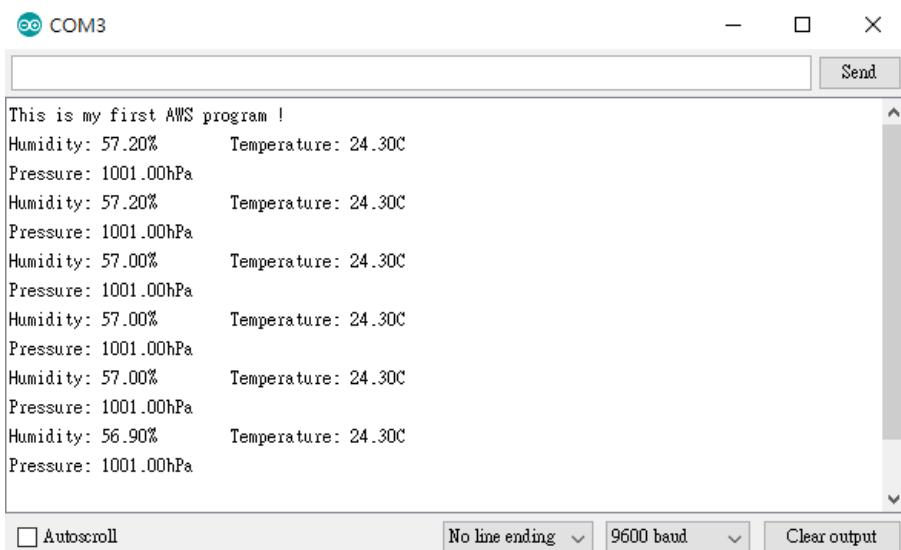
    //get and print temperatures
    Serial.print("Temp: ");
    Serial.print(bmp280.getTemperature());
    Serial.println("C"); // The unit for Celsius because original arduino don't support speical symbols

    //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.
```

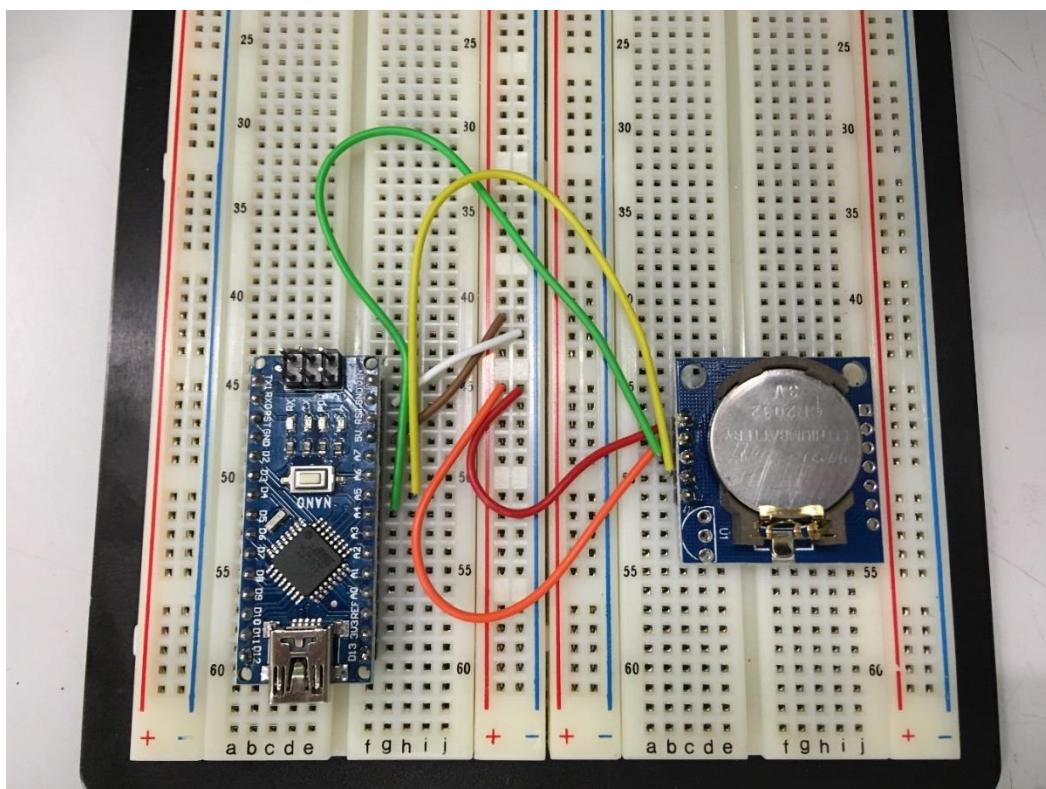
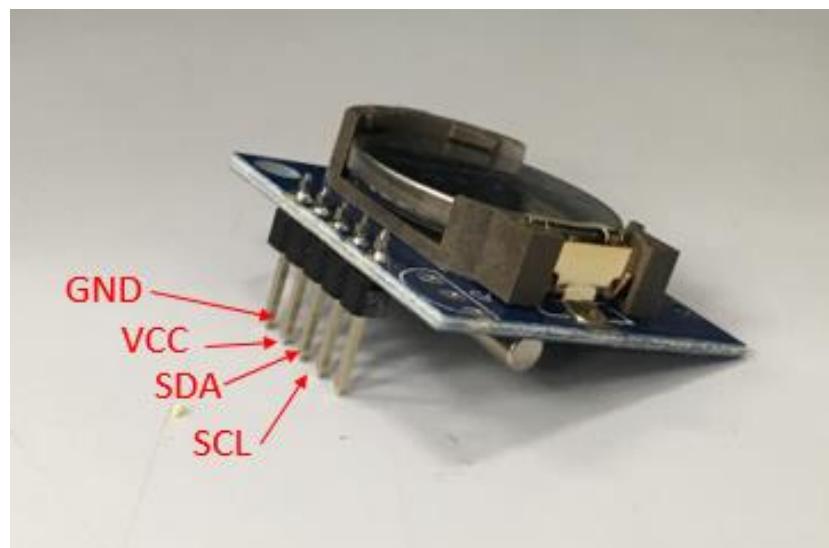
Compile and upload the program to see the results! Remember to save your My\_new\_AWS file again!



## 6. INSTALLING REAL-TIME CLOCK - RTC1307 MODULE

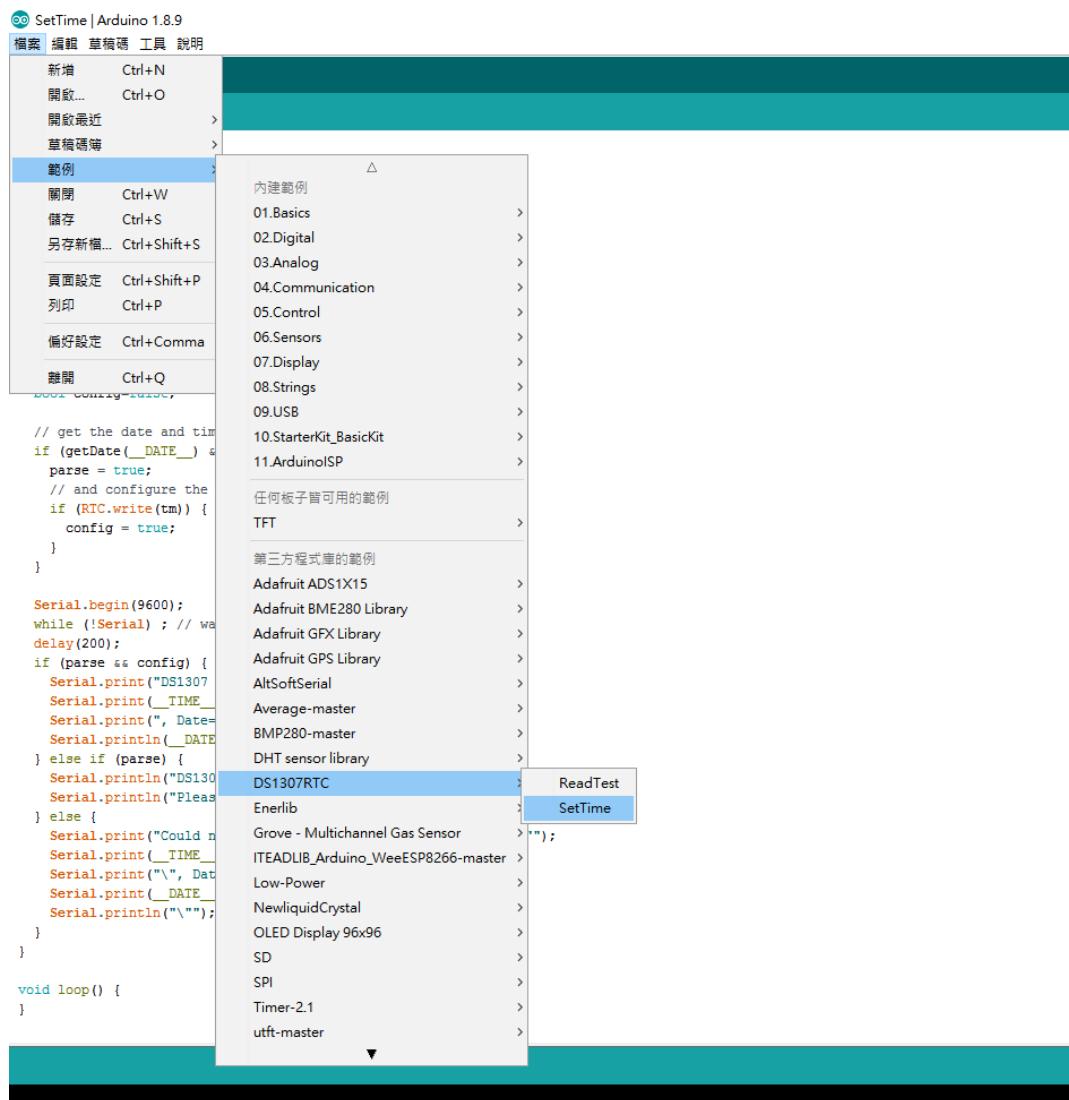
### 6.1 Wiring

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 (Set Time)

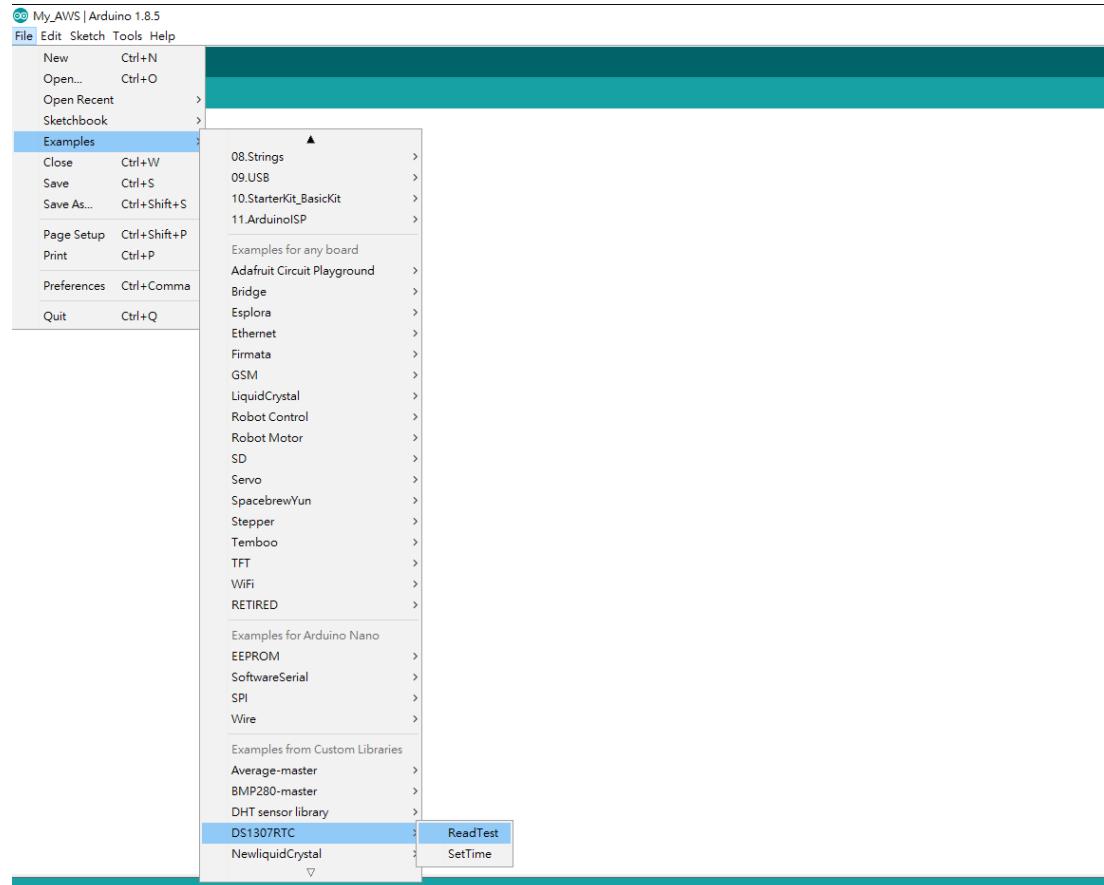
Since the clock may not be synchronizing with local time, we can run this script in synchronizing the time with computer. Choose the SetTime file from the DS1307RTC library as shown below.



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

### 6.3 Choosing an Example file in RTC library (Read Time)

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



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

The screenshot shows the Arduino IDE with the 'ReadTest' sketch open. The code includes #includes for `<Wire.h>`, `<Time.h>`, and `<DS1307RTC.h>`. The `setup()` function initializes the serial port at 9600 bps and prints a test message. The `loop()` function reads the RTC time and date, prints it to the serial monitor, and handles errors if the RTC is stopped. The serial monitor window shows the output 'DS1307RTC Read Test' followed by a series of time and date prints. A checkbox for 'Autoscroll' is checked at the bottom of the monitor window.

```

void setup() {
    Serial.begin(9600);
    while (!Serial); // wait for serial
    delay(200);
    Serial.print("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);
    }
}

```

## 6.4 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:
}

```

```

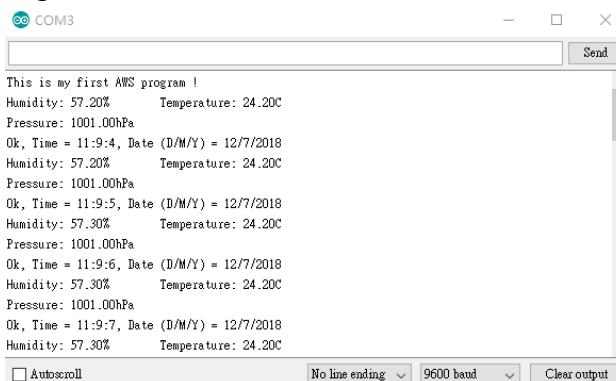
ReadTest | Arduino 1.8.5
File Edit Sketch Tools Help
ReadTest
#include <Wire.h>
#include <TimeLib.h>
#include <DS1307RTC.h>

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.print("The DS1307 is stopped. Please run the SetTime");
            Serial.print("example to initialize the time and begin running.");
            Serial.println();
        } else {
            Serial.print("DS1307 read error! Please check the circuitry.");
            Serial.println();
        }
        delay(9000);
    }
    delay(1000);
}

```

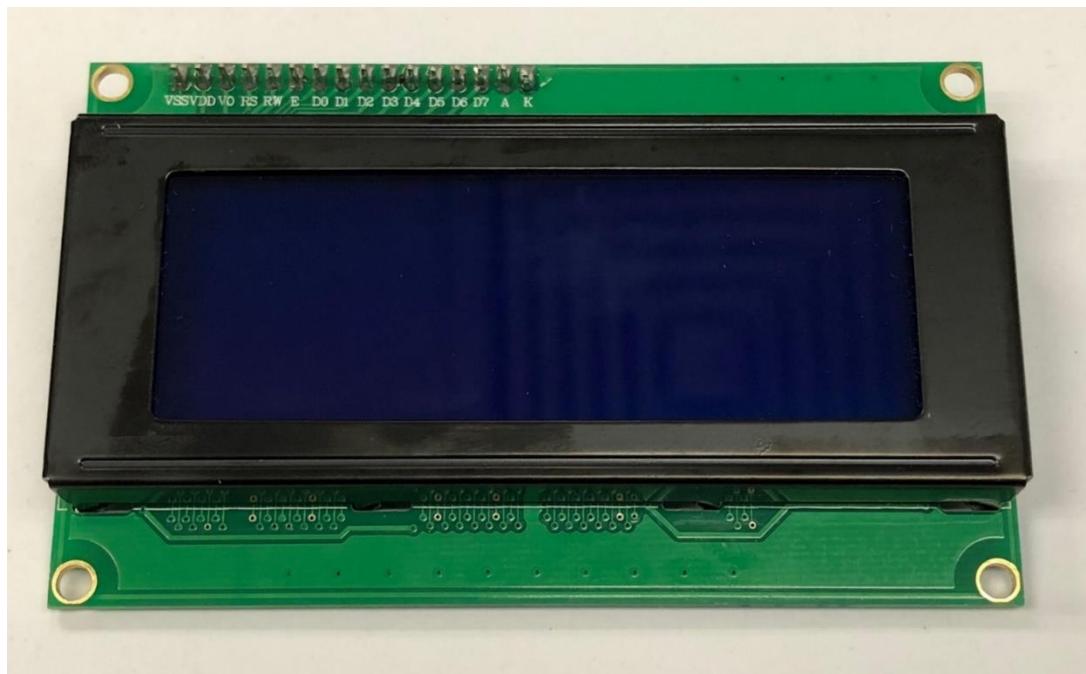
Compile and upload the program to see the results! Remember to save your My\_new\_AWS file again!



**7. INSTALLING LCD DISPLAY**

**- LCD I2C 2004A DISPLAY**

Front view

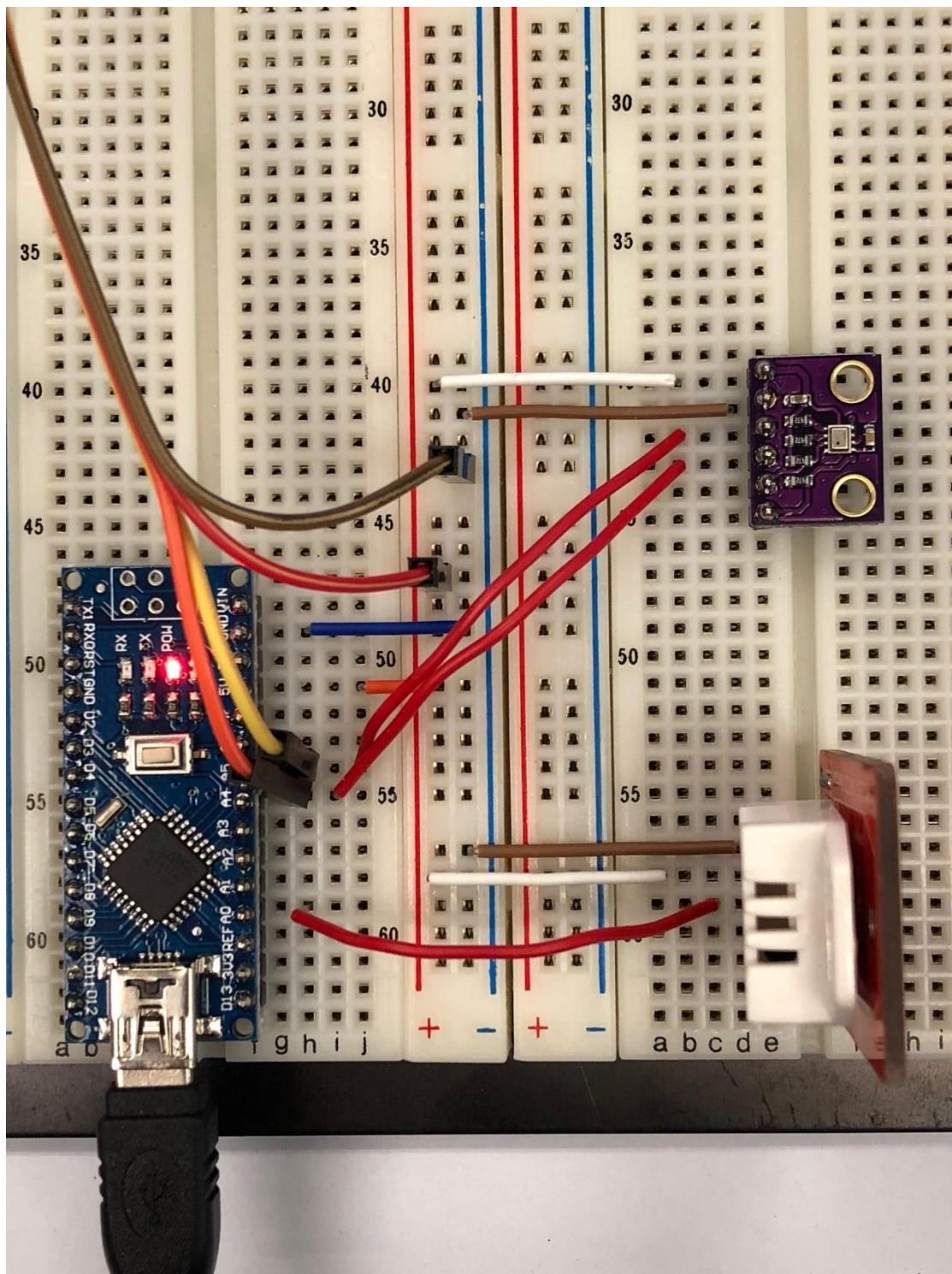


Rear view



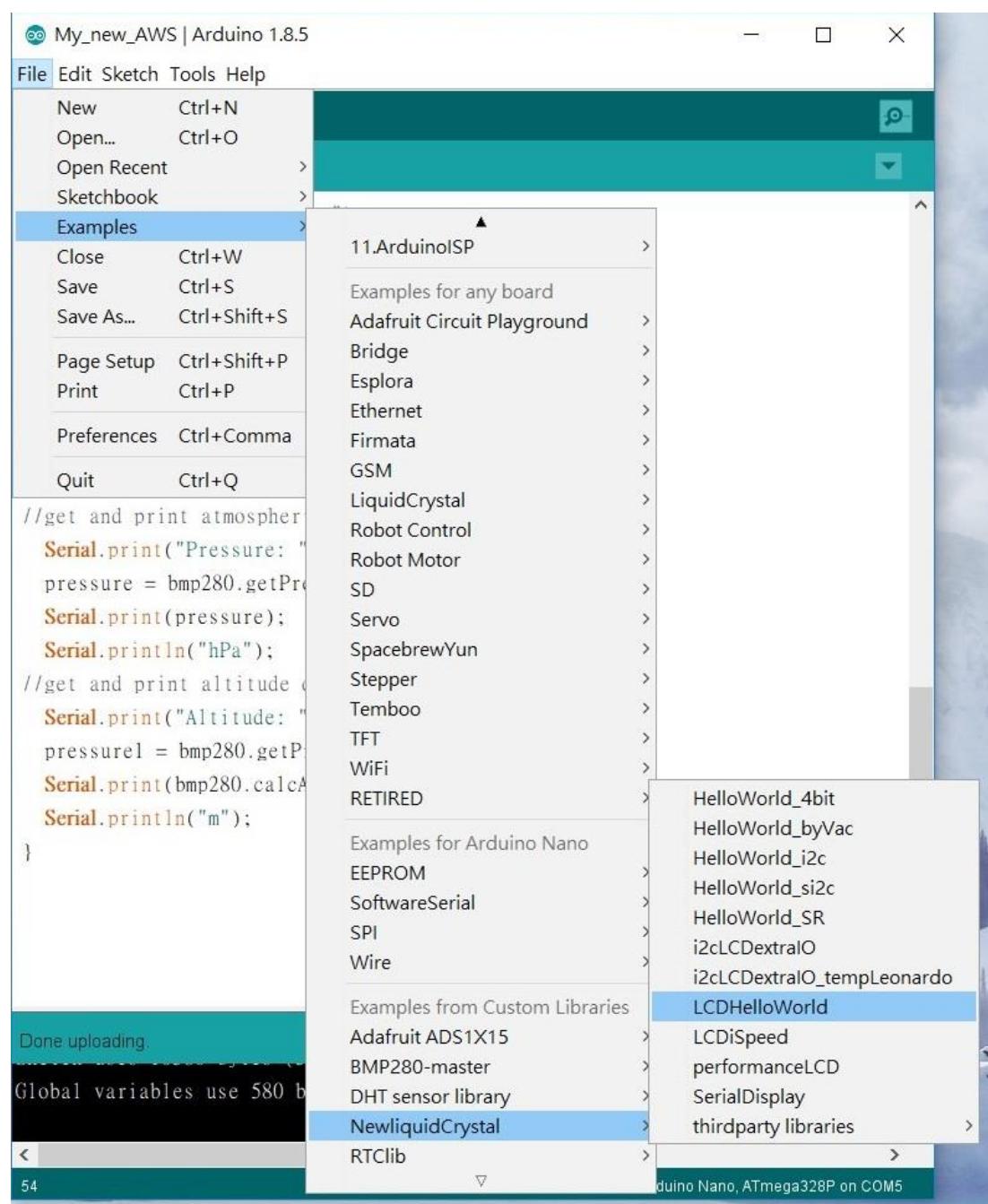
## 7.1 Wiring

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**

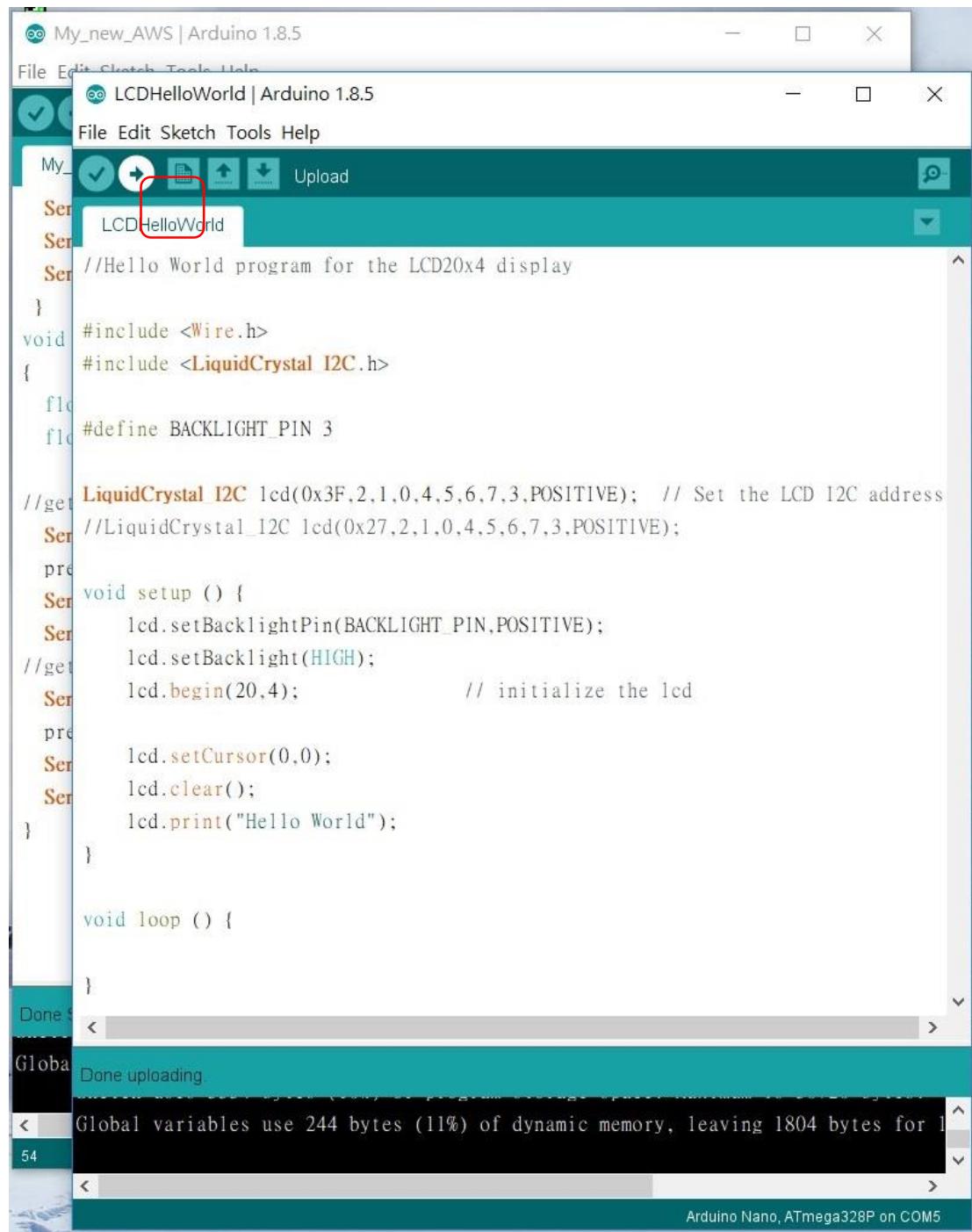


## 7.2 Choosing an Example file in the NewliquidCrystal library

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.





### 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 "BMP280.h"
#include <TimeLib.h>
#include "DS1307RTC.h"
#include <LiquidCrystal_I2C.h>

#define DHTPIN A0
#define DHTTYPE DHT22
#define BACKLIGHT_PIN 3
DHT dht(DHTPIN, DHTTYPE);
BMP280 bmp280;
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

char myArray1[19];
char myArray2[19]; Add new command

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();
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.begin(20,4);
}

void loop() {
  timeElements_t tm;
  RTC.read(tm);
  // put your main code here, to run repeatedly:

  /**** Date and Time on display ****/
  sprintf(myArray1, "2024/02/04 %A", tm.Day, tm.Month, tm.YearToCalendar(tm.Year));
  lcd.setCursor(0,0);
  lcd.print(myArray1);

  sprintf(myArray2, "%H:%M:%S", 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 LCD2024 display

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

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);

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

void loop () {
}

```

## USER GUIDE

### BASIC ARDUINO TRAINING (Exercise: Automatic Weather Station)

## CONTENTS

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/%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: ");

//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);
}

```

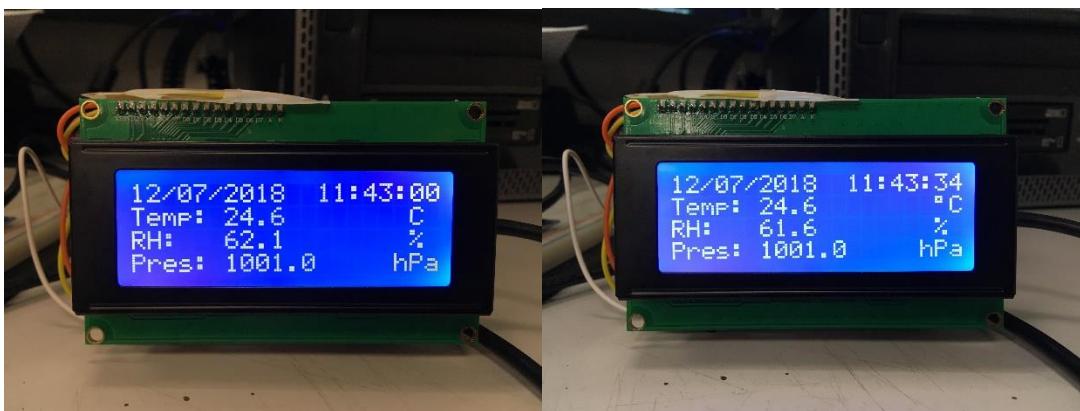
*// Integrate the abbreviation setting on display*

*// 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;



## 8. SENSORS COMPARISON

Since sensors may have their initial errors in measurement, calibration and testing are required. A script was developed to compare the DHT22 sensors in a parallel circuit.

```
// Example testing sketch for various DHT humidity/temperature sensors
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"

#define DHTPIN1 A0          // what digital pin we're connected to
#define DHTPIN2 A1
#define DHTPIN3 A2
#define BACKLIGHT_PIN 3

#define DHTTYPE1 DHT22      // DHT 22  (AM2302), AM2321
#define DHTTYPE2 DHT22
#define DHTTYPE3 DHT22

DHT dht1(DHTPIN1, DHTTYPE1);
DHT dht2(DHTPIN2, DHTTYPE2);
DHT dht3(DHTPIN3, DHTTYPE3);
LiquidCrystal_I2C lcd(0x3F,2,1,0,4,5,6,7,3,POSITIVE);

void setup() {
    Serial.begin(9600);
    Serial.println("DHT22 test!");
    lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
    lcd.setBacklight(HIGH);
    lcd.begin(20,4);           // initialize the lcd

    dht2.begin();
    dht3.begin();
}
```

```
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 h1 = dht1.readHumidity();
    float h2 = dht2.readHumidity();
    float h3 = dht3.readHumidity();
    // Read temperature as Celsius (the default)
    float t1 = dht1.readTemperature();
    float t2 = dht2.readTemperature();
    float t3 = dht3.readTemperature();

    lcd.setCursor(0,0);
    lcd.print("T1:");
    lcd.setCursor(0,1);
    lcd.print("T2:");
    lcd.setCursor(0,2);
    lcd.print("T3:");
    lcd.setCursor(4,0);
    lcd.print(t1);
    lcd.setCursor(4,1);
    lcd.print(t2);
    lcd.setCursor(4,2);
    lcd.print(t3);
    lcd.setCursor(0,0);
    lcd.print("T1:");
    lcd.setCursor(0,2);
    lcd.print("T2:");
    lcd.setCursor(0,2);
    lcd.print("T3:");
    lcd.setCursor(4,0);
    lcd.print(t2);
    lcd.setCursor(4,1);
    lcd.print(t2);
    lcd.setCursor(4,2);
    lcd.print(t3);
```

```
lcd.setCursor(10,0);
lcd.print("RH1:");
lcd.setCursor(10,1);
lcd.print("RH2:");
lcd.setCursor(10,2);
lcd.print("RH3:");
lcd.setCursor(15,0);
lcd.print(h1);
lcd.setCursor(15,1);
lcd.print(h2);
lcd.setCursor(15,2);
lcd.print(h3);

lcd.setCursor(10,0);
lcd.print("RH1:");
lcd.setCursor(10,2);
lcd.print("RH2:");
lcd.setCursor(10,2);
lcd.print("RH3:");
lcd.setCursor(15,0);
lcd.print(h2);
lcd.setCursor(15,1);
lcd.print(h2);
lcd.setCursor(15,2);
lcd.print(h3);
}
```

