Training Workshop on

Arduino-based Automatic Weather Station (AWS)

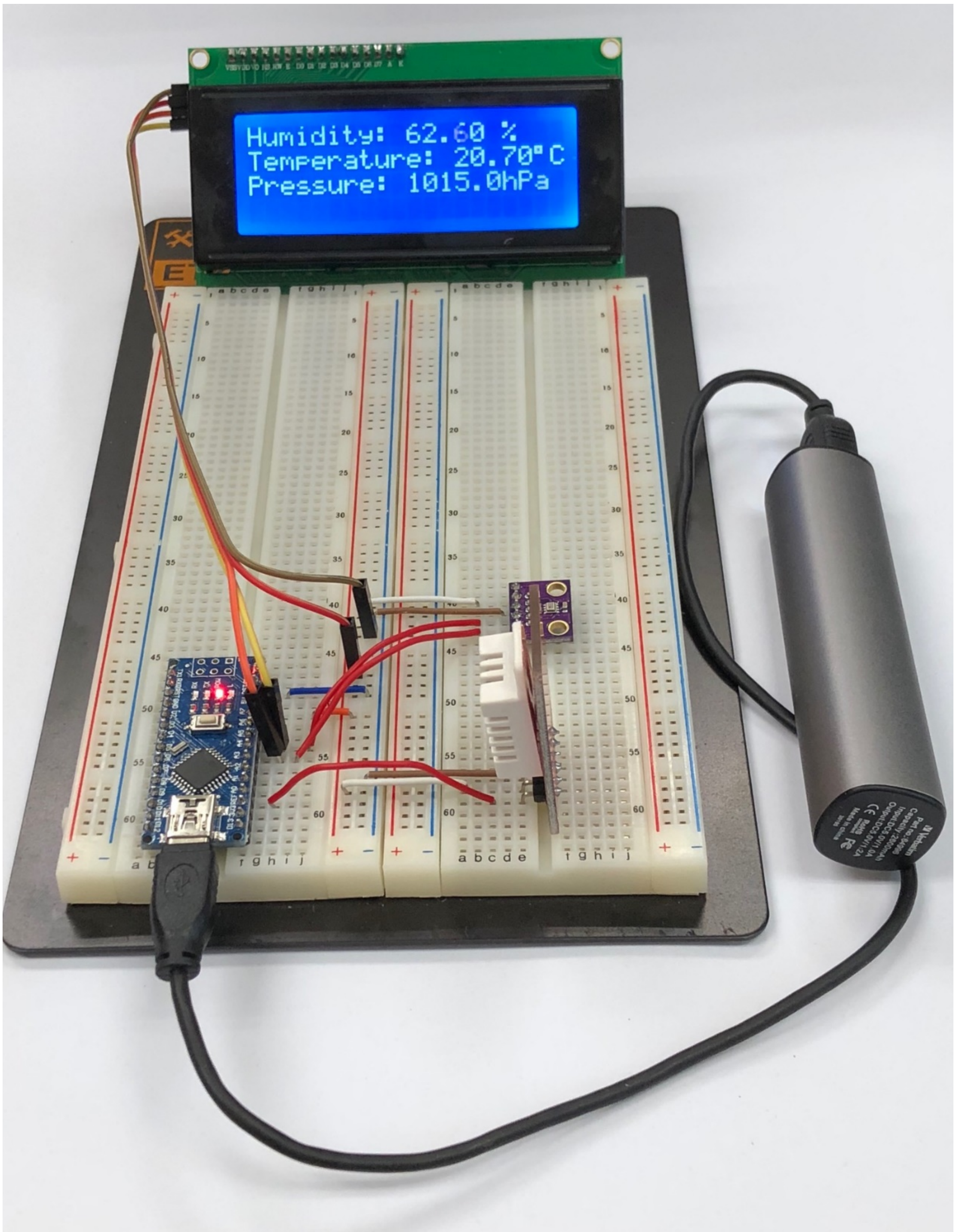
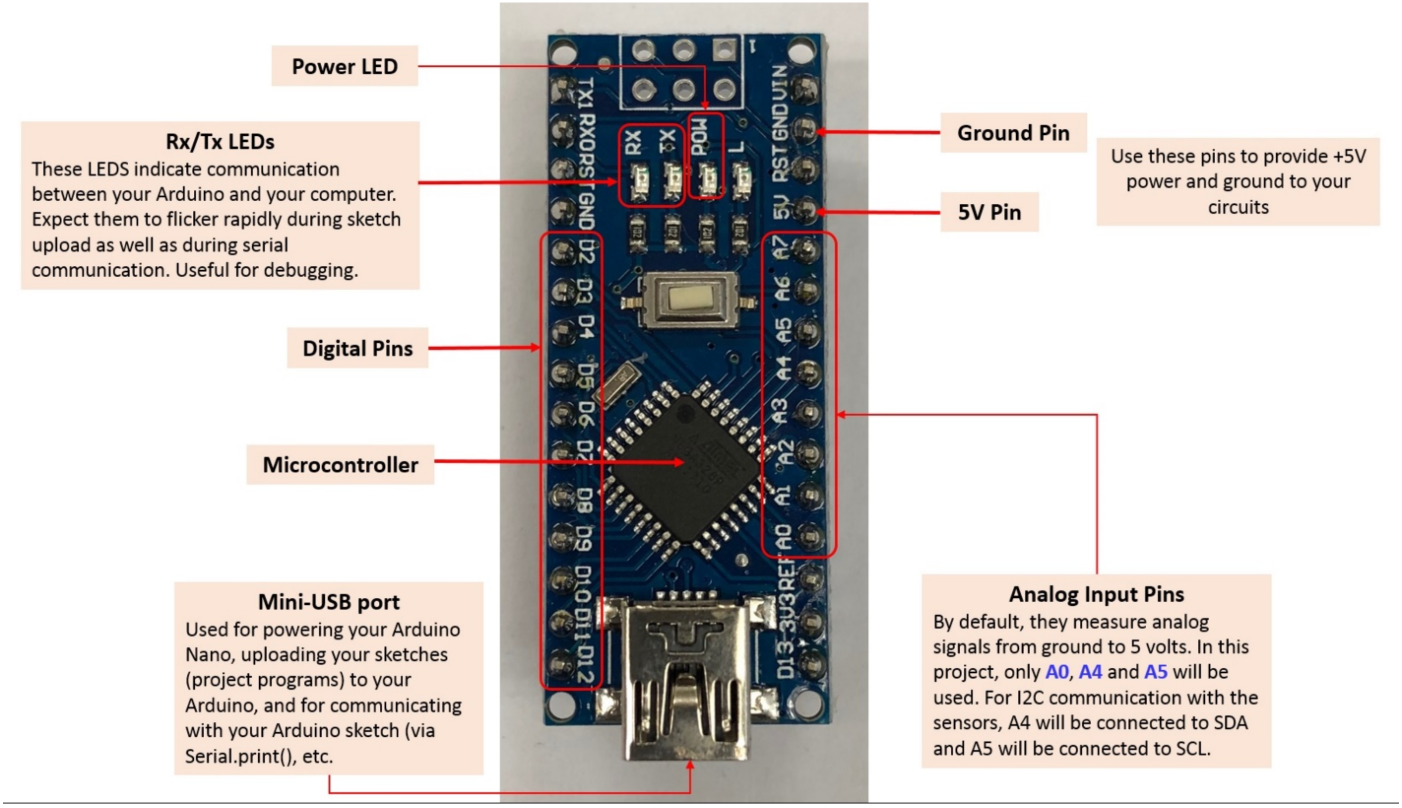


Table of Content

1. **Getting started**
2. **Creating your first AWS project**
3. **Installing a temperature and humidity sensor DHT22**
4. **Installing a Pressure sensor BMP280**
5. **Installing a LCD Display**

2. Getting started

1.1 The Arduino Nano Board



Power LED

Rx/Tx LEDs
These LEDS indicate communication between your Arduino and your computer. Expect them to flicker rapidly during sketch upload as well as during serial communication. Useful for debugging.

Digital Pins

Microcontroller

Mini-USB port
Used for powering your Arduino Nano, uploading your sketches (project programs) to your Arduino, and for communicating with your Arduino sketch (via Serial.print(), etc.

Ground Pin

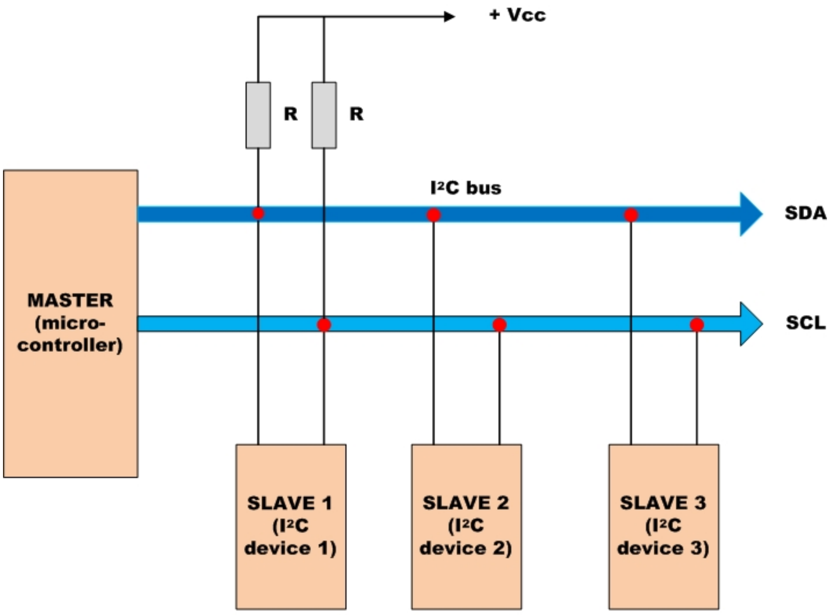
5V Pin

Use these pins to provide +5V power and ground to your circuits

Analog Input Pins
By default, they measure analog signals from ground to 5 volts. In this project, only **A0**, **A4** and **A5** will be used. For I2C communication with the sensors, A4 will be connected to SDA and A5 will be connected to SCL.

TX1 RX0 RST GND D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
VIN GND RST 5V A7 A6 A5 A4 A3 A2 A1 A0 REF 3V3 D13
RX TX POW L

I²C Architecture



Multiple devices on common I²C bus

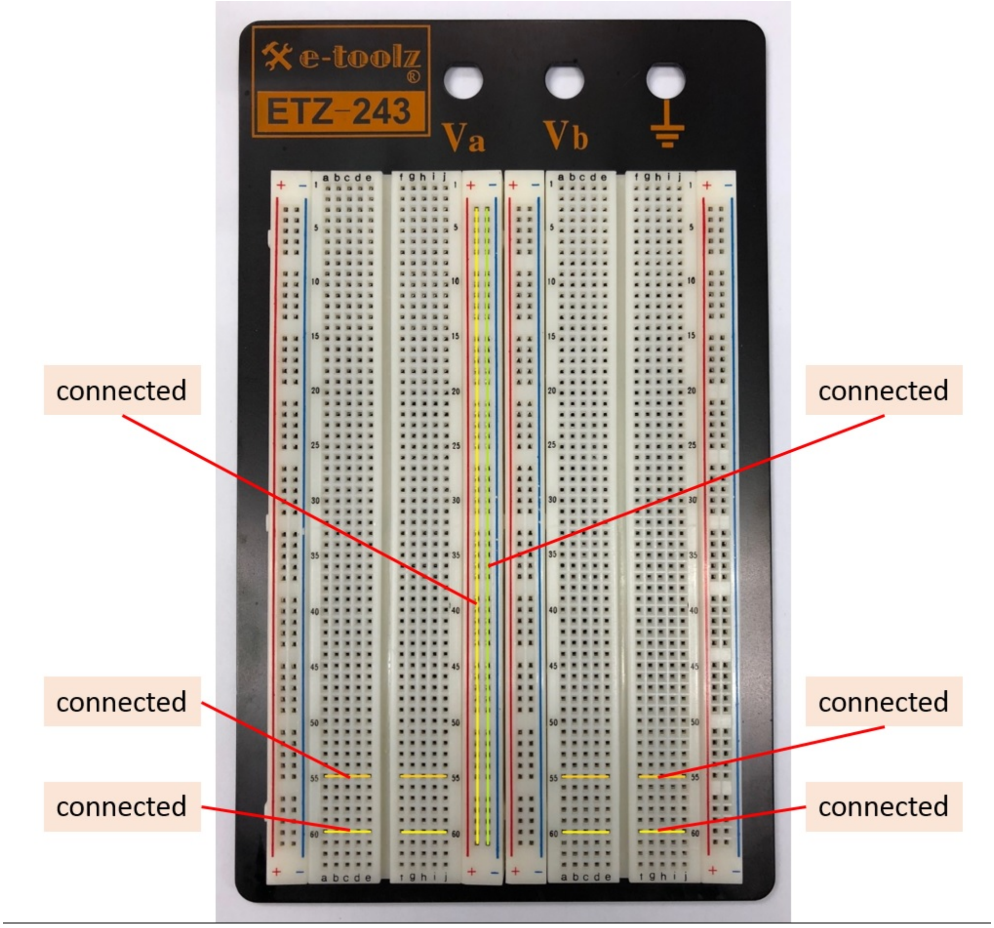
Serial data (SDA)
Serial clock (SCL)

1.2 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

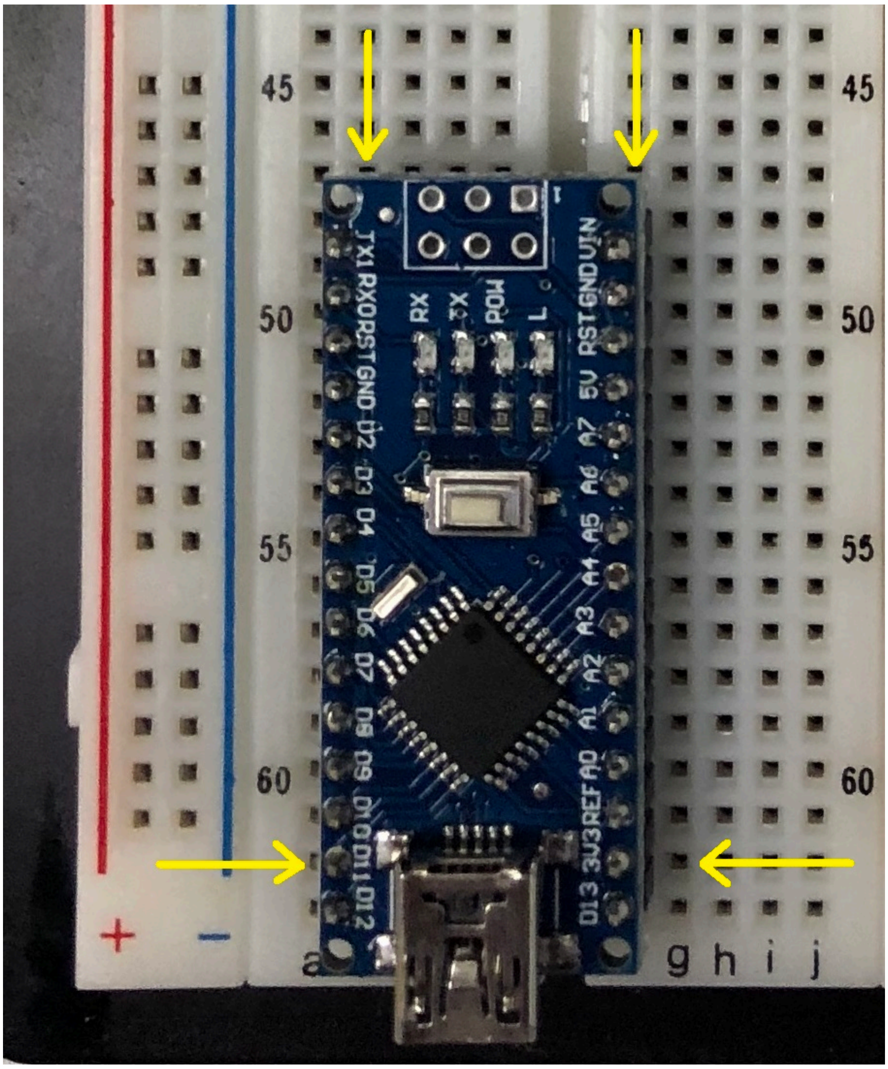
In this workshop, the Arduino IDE has been downloaded and installed for you. The link to start your Arduino IDE is shown as an Arduino icon on the desktop of your PC (see below)

1.3 Basic information about the Breadboard used in this workshop



1.4 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

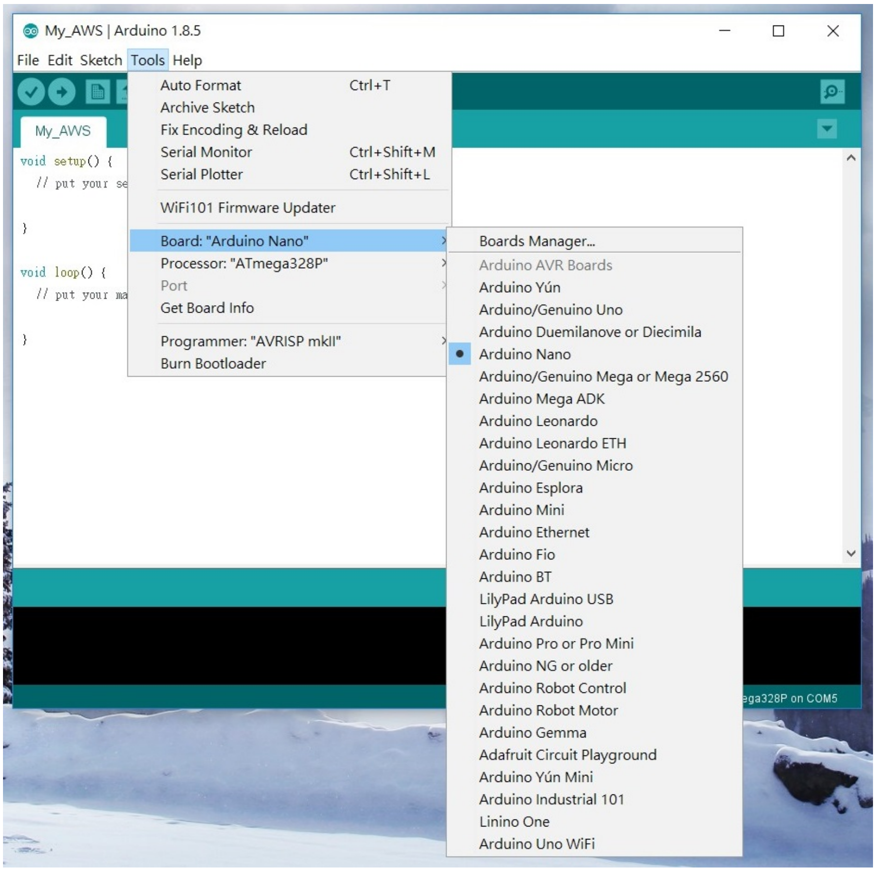

45
50
55
60
+
−
a
g h i j
TX1 RX0 RST GND D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
VIN GND RST 5V A7 A6 A5 A4 A3 A2 A1 AREF 3V3 D13
RX
TX
POW
L

2. Creating your first AWS project

2.1 Running the Arduino IDE

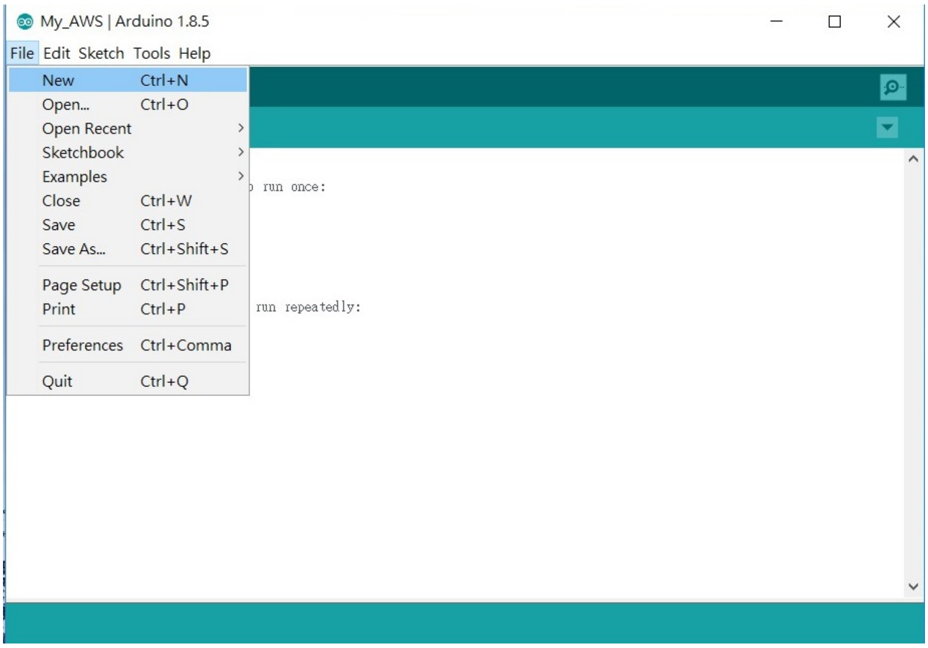
Double click on the Arduino icon on the desktop of your PC to start running the Arduino IDE.

2.2 Selecting a proper Arduino board



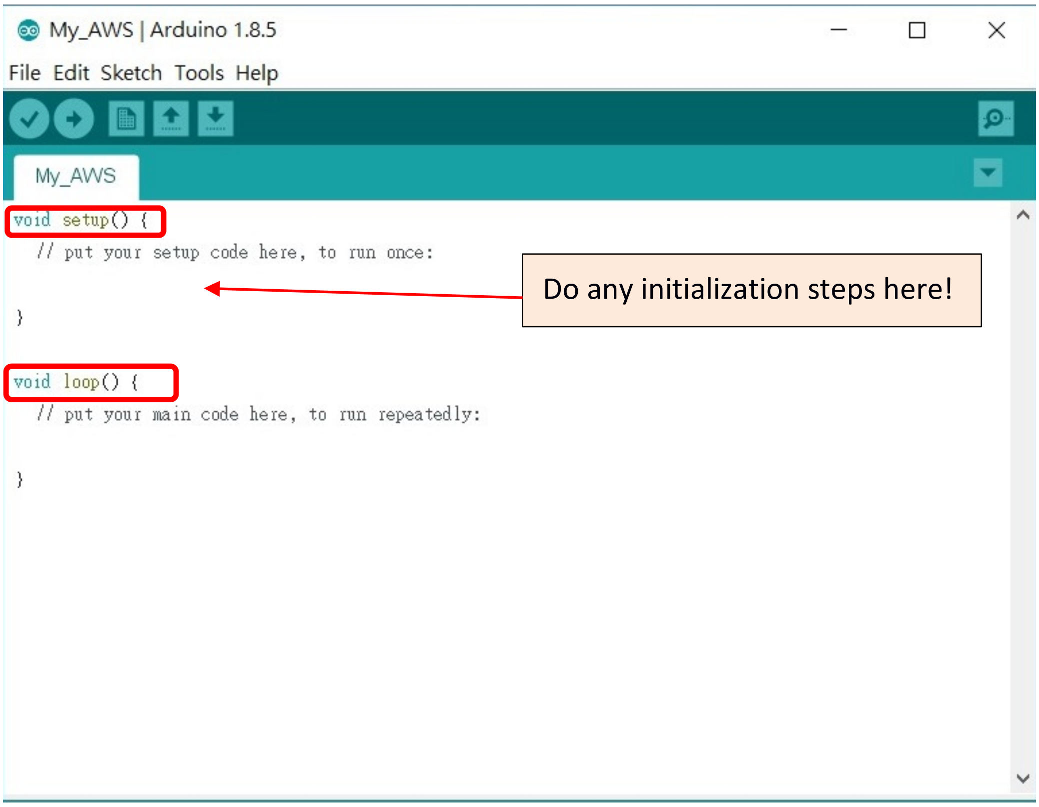
```
void setup() {
  // put your se

}

void loop() {
  // put your ma

}
```

2.3 Creating a new project file



```
o run once:

run repeatedly:
```

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



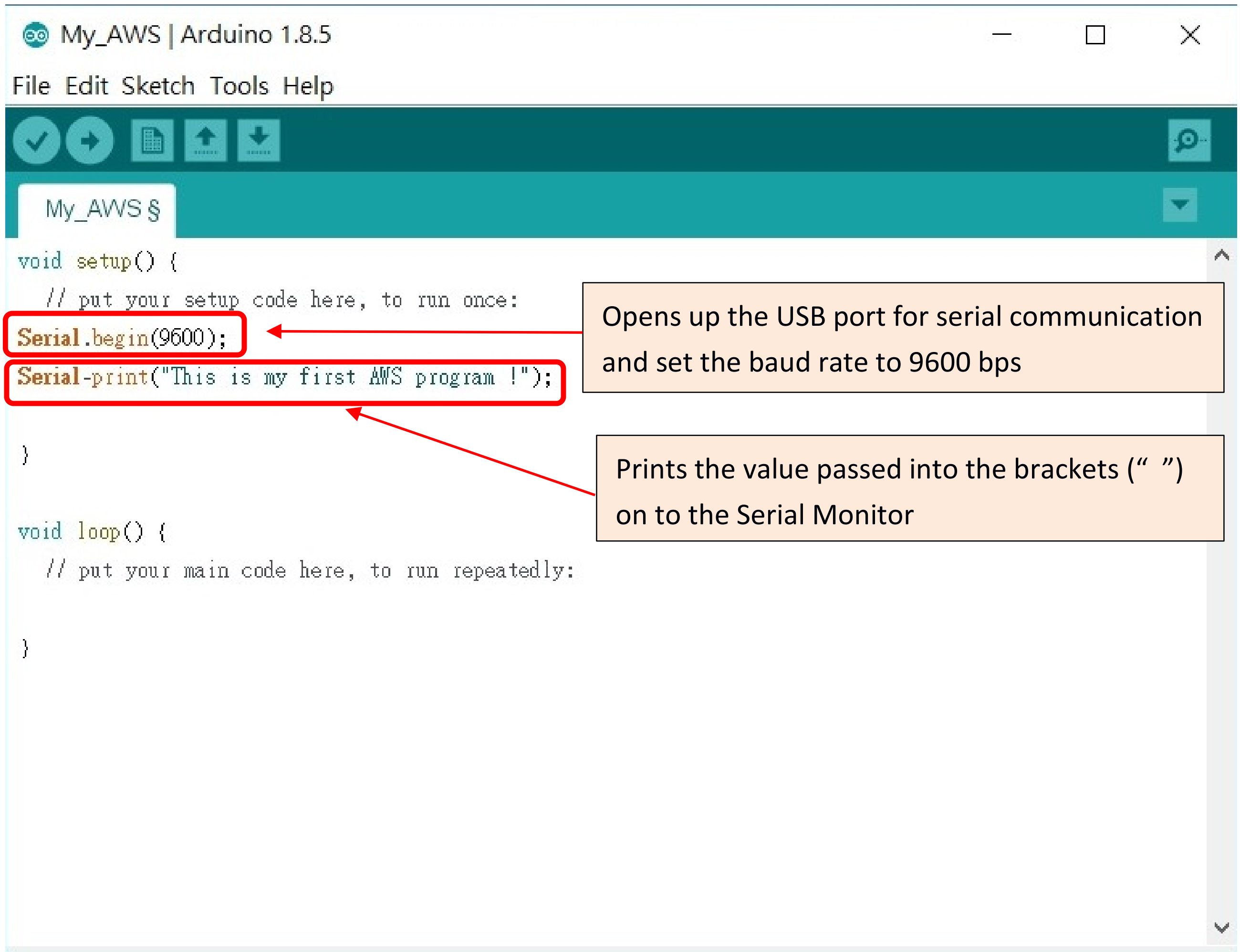
```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

Do any initialization steps here!

Type the following in the setup() section:

Serial.begin(9600);

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



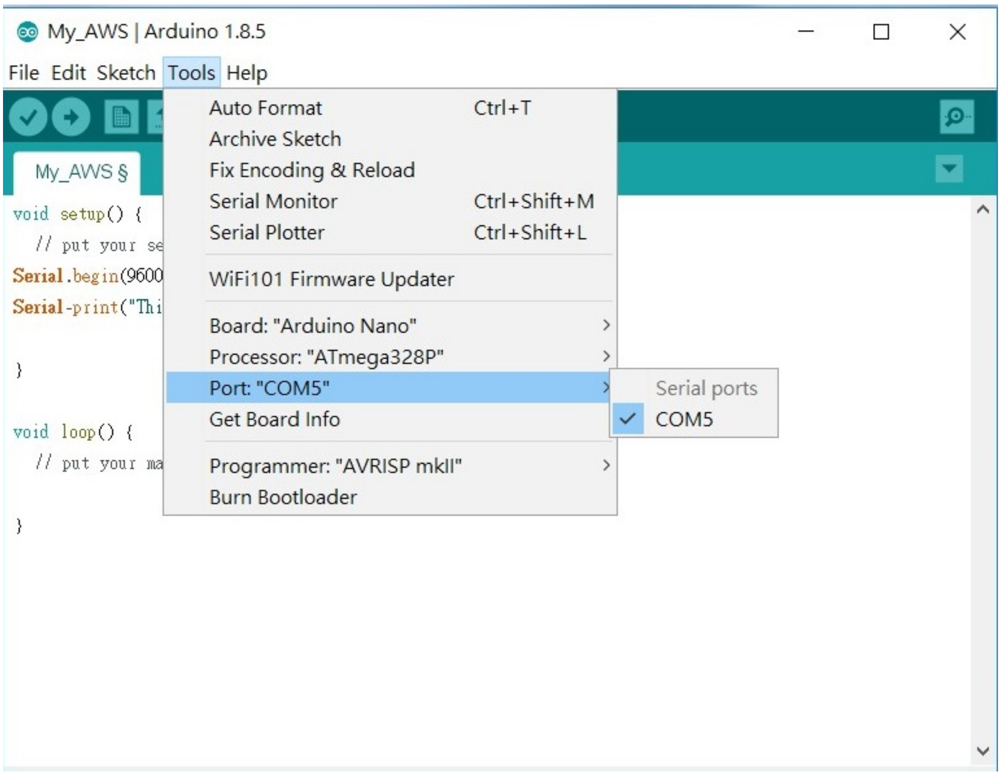
```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
Serial.print("This is my first AWS program !");

}

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

}
```

Opens up the USB port for serial communication and set the baud rate to 9600 bps

Prints the value passed into the brackets (“ ”) on to the Serial Monitor

2.4 Setting up an appropriate Communication Port

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

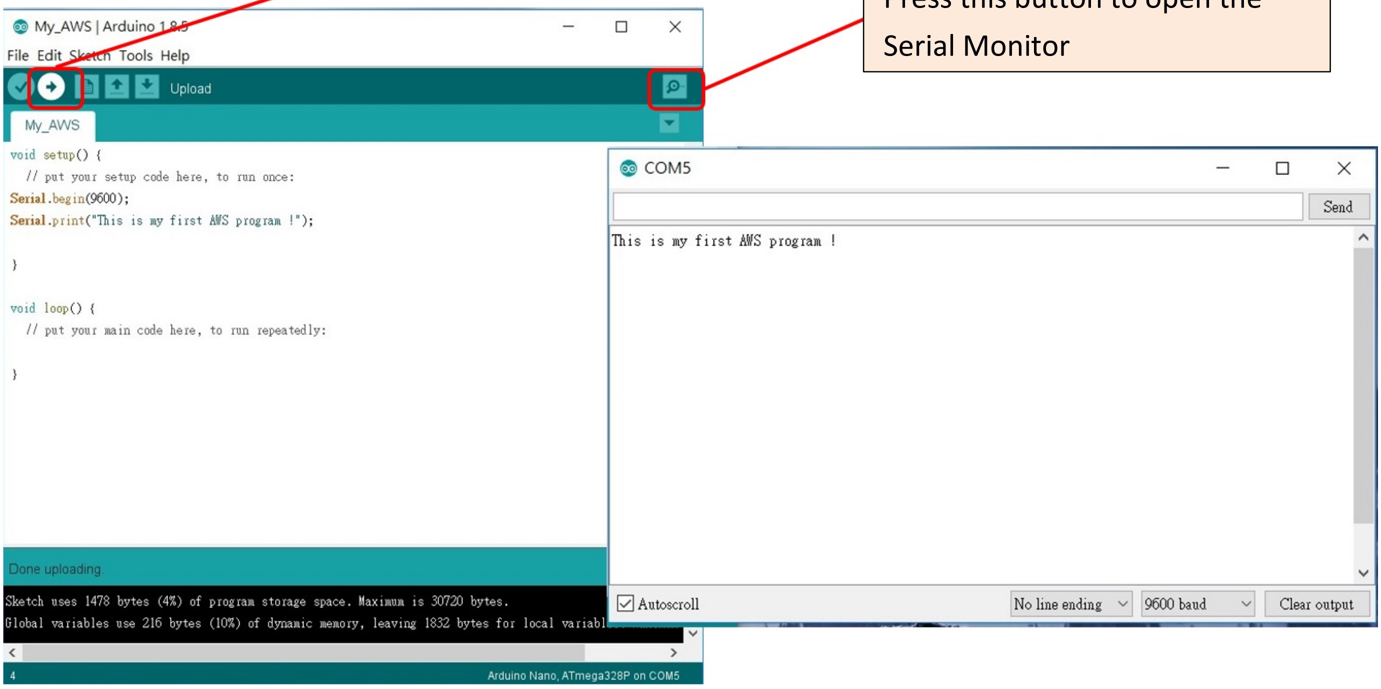


```
void setup() {
  // put your se
Serial.begin(9600
Serial-print("Thi

}

void loop() {
  // put your ma

}
```

2.5 Compiling and uploading a program to your Arduino

Press this button to compile the program and upload to Arduino

Press this button to open the Serial Monitor



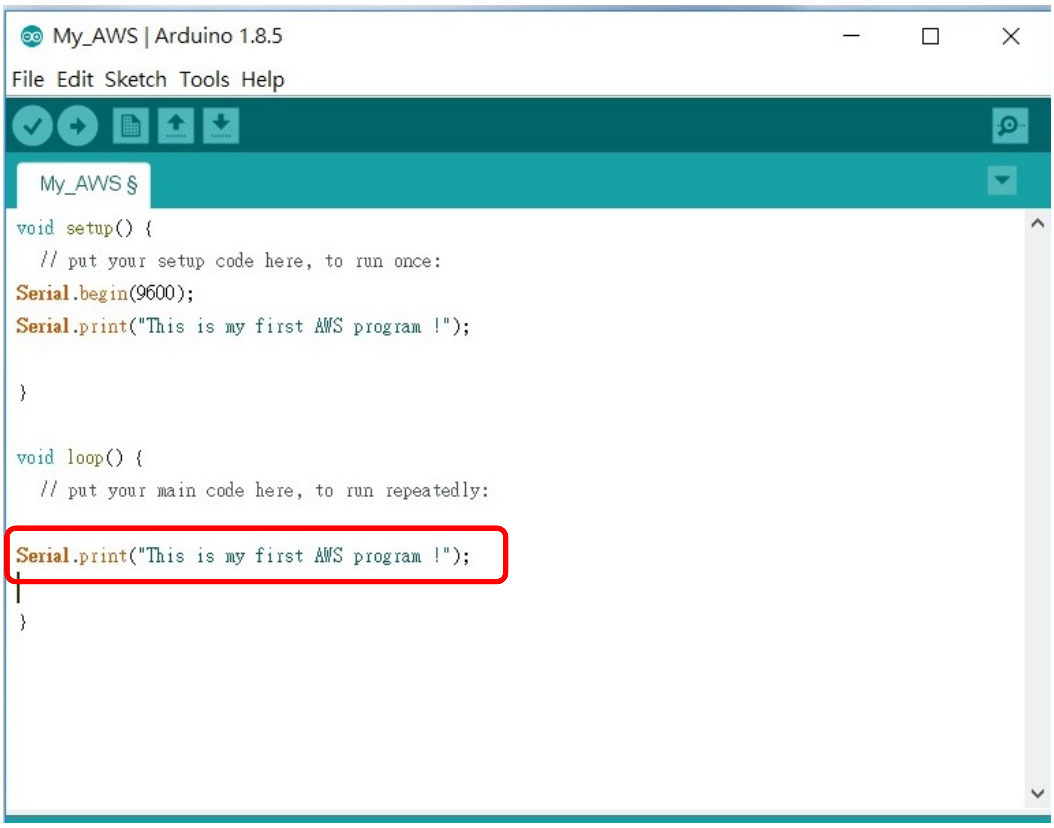
```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
Serial.print("This is my first AWS program !");

}

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

}
```

```
Done uploading.
Sketch uses 1478 bytes (4%) of program storage space. Maximum is 30720 bytes.
Global variables use 216 bytes (10%) of dynamic memory, leaving 1832 bytes for local variabl
```

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

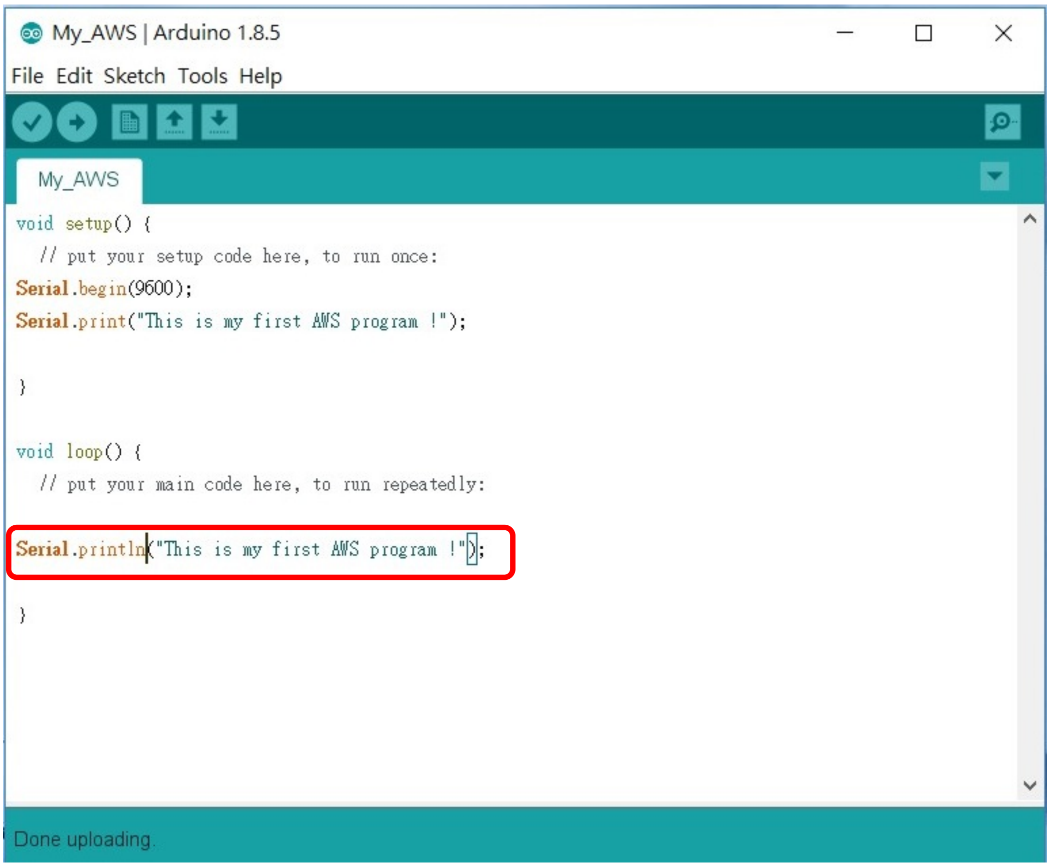
2.6 Using the loop() function

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



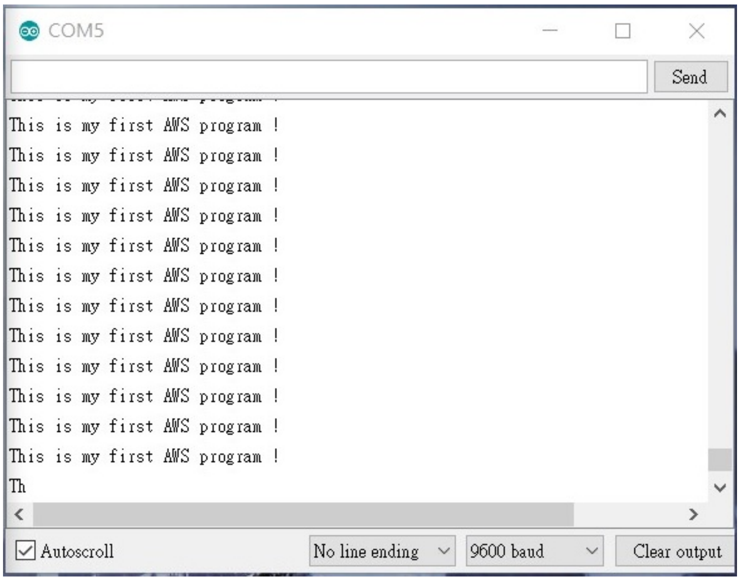
```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
Serial.print("This is my first AWS program !");

}

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.



```
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 !");

}
```

Done uploading.

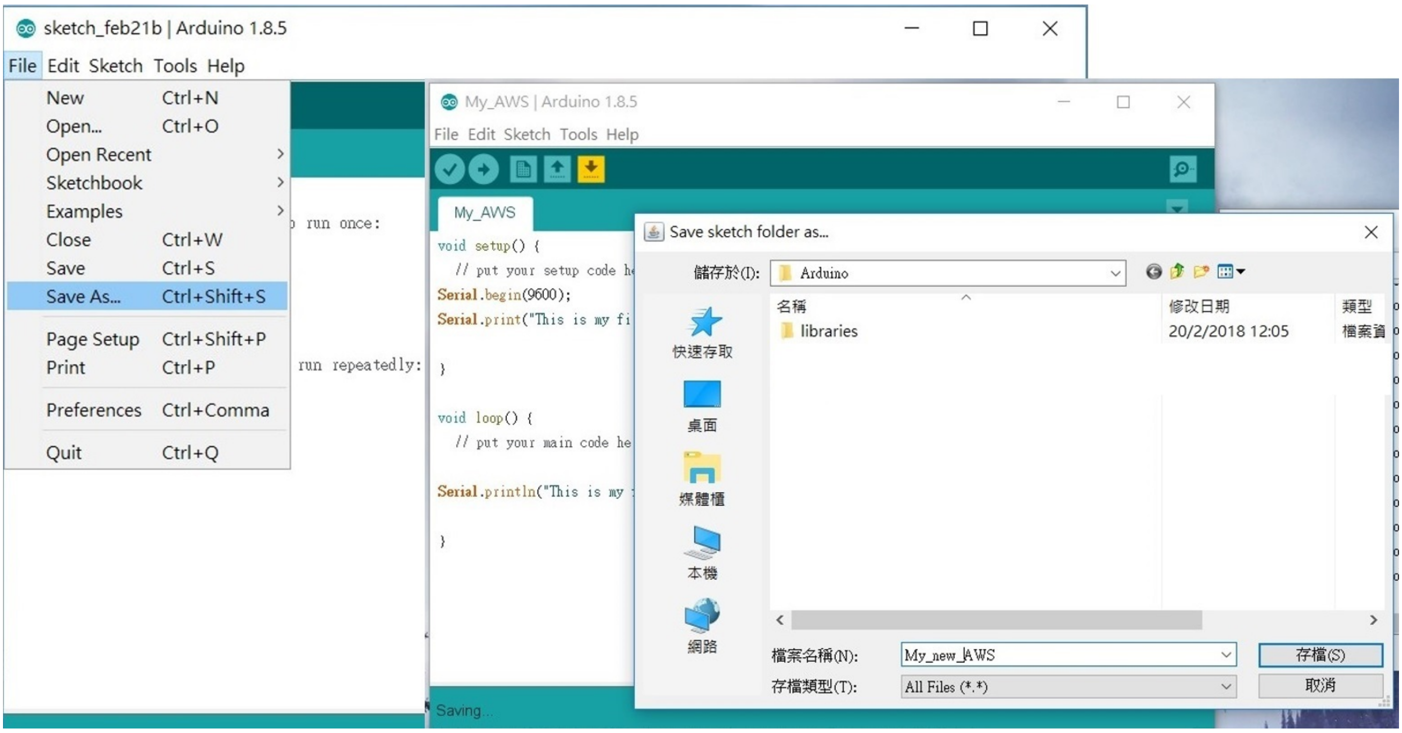


```
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
This is my first AWS program !
Th
```

☑ Autoscroll | No line ending | 9600 baud

2.7 Saving your first project file

Save the program as My_new_AWS.



```
void setup() {
  // put your setup code he
Serial.begin(9600);
Serial.print("This is my fi

}

void loop() {
  // put your main code he

Serial.println("This is my 

}
```

儲存於(I): Arduino

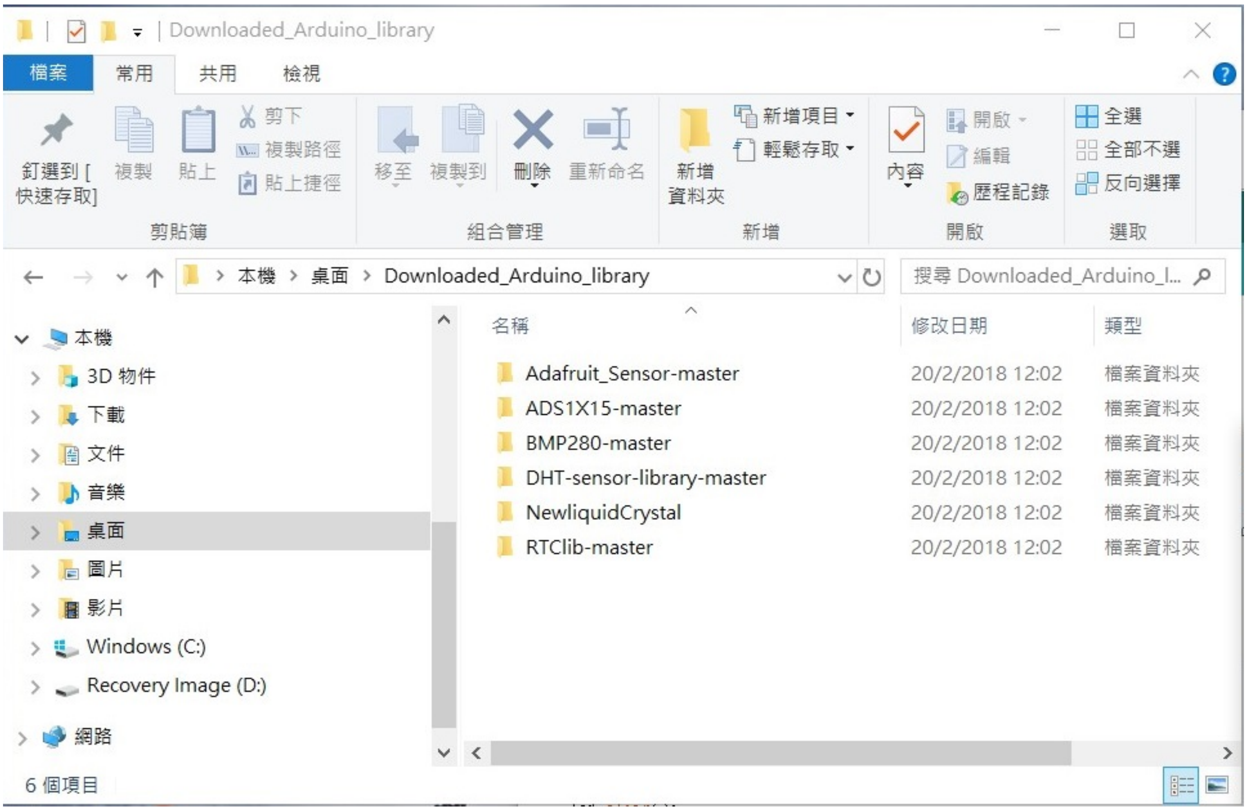
名稱	修改日期	類型
libraries	20/2/2018 12:05	檔案資

檔案名稱(N): My_new_AWS

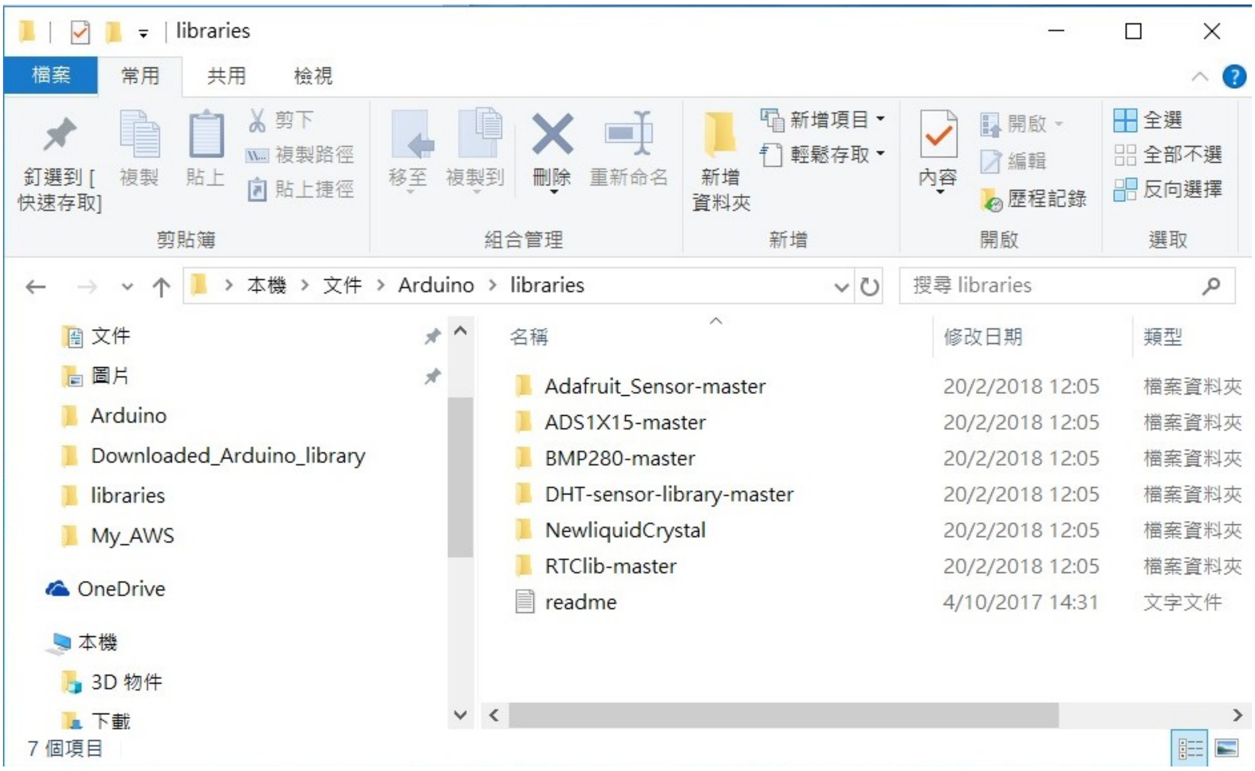
存檔類型(T): All Files (*.*)

2.8 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:

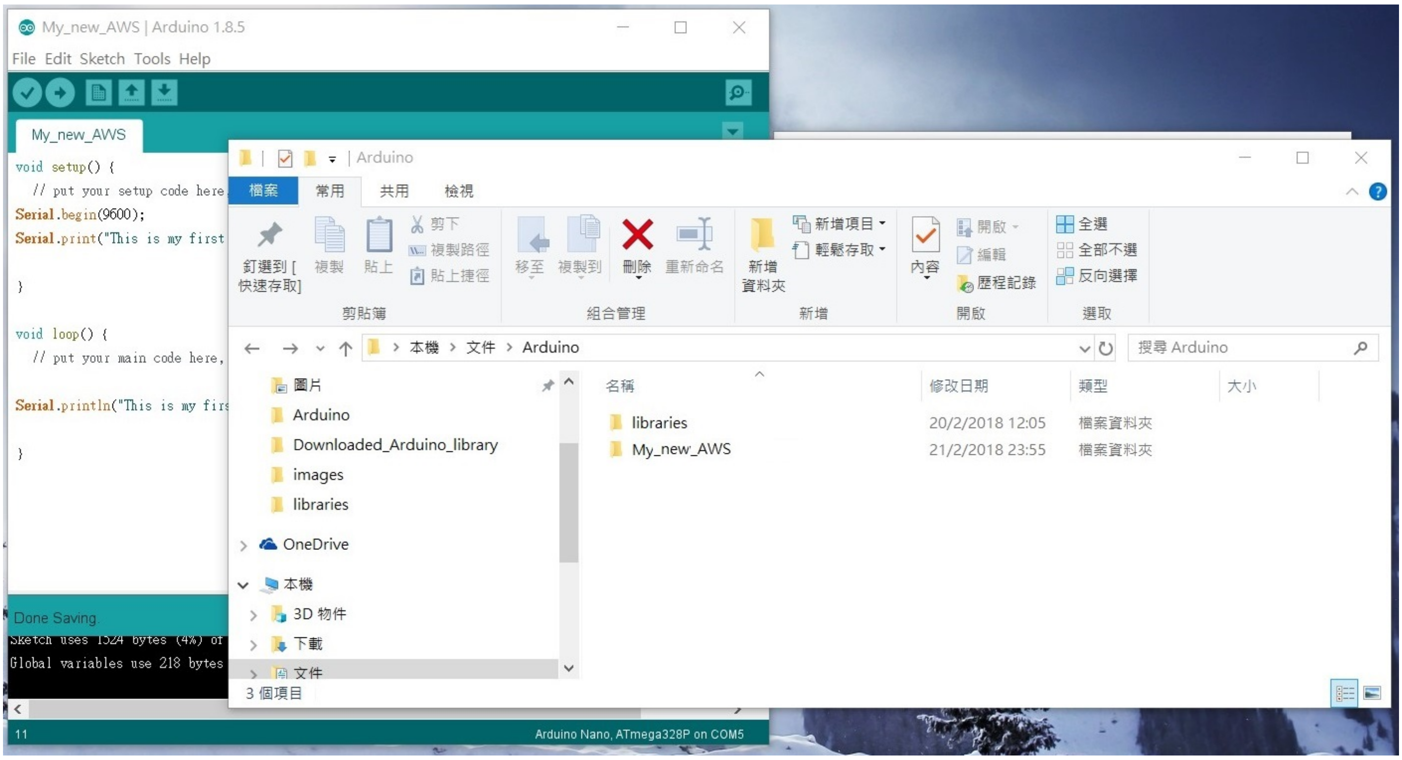


名稱	修改日期	類型
Adafruit_Sensor-master	20/2/2018 12:02	檔案資料夾
ADS1X15-master	20/2/2018 12:02	檔案資料夾
BMP280-master	20/2/2018 12:02	檔案資料夾
DHT-sensor-library-master	20/2/2018 12:02	檔案資料夾
NewliquidCrystal	20/2/2018 12:02	檔案資料夾
RTClib-master	20/2/2018 12:02	檔案資料夾



名稱	修改日期	類型
Adafruit_Sensor-master	20/2/2018 12:05	檔案資料夾
ADS1X15-master	20/2/2018 12:05	檔案資料夾
BMP280-master	20/2/2018 12:05	檔案資料夾
DHT-sensor-library-master	20/2/2018 12:05	檔案資料夾
NewliquidCrystal	20/2/2018 12:05	檔案資料夾
RTClib-master	20/2/2018 12:05	檔案資料夾
readme	4/10/2017 14:31	文字文件

Note that My_new_AWS folder is next to the libraries folder.



My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS
Arduino
檔案 常用 共用 檢視
釘選到[快速存取] 複製 貼上 剪下 複製路徑 貼上捷徑
剪貼簿
移至 複製到 刪除 重新命名
組合管理
新增資料夾 新增項目 輕鬆存取
新增
內容 開啟 編輯 歷程記錄
開啟
全選 全部不選 反向選擇
選取
本機 > 文件 > Arduino
搜尋 Arduino
圖片
Arduino
Downloaded_Arduino_library
images
libraries
OneDrive
本機
3D 物件
下載
文件
3 個項目
Arduino Nano, ATmega328P on COM5

```
void setup() {
  // put your setup code here
Serial.begin(9600);
Serial.print("This is my first

}

void loop() {
  // put your main code here,

Serial.println("This is my firs

}
```

Done Saving.

```
Sketch uses 1524 bytes (4%) of
Global variables use 218 bytes
```

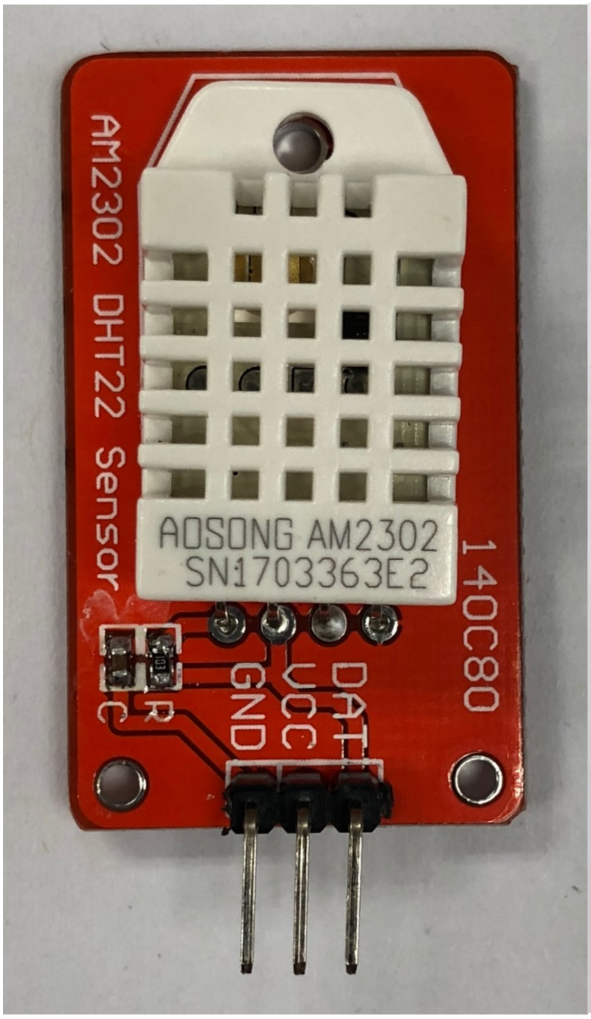
名稱	修改日期	類型	大小
libraries	20/2/2018 12:05	檔案資料夾	
My_new_AWS	21/2/2018 23:55	檔案資料夾	

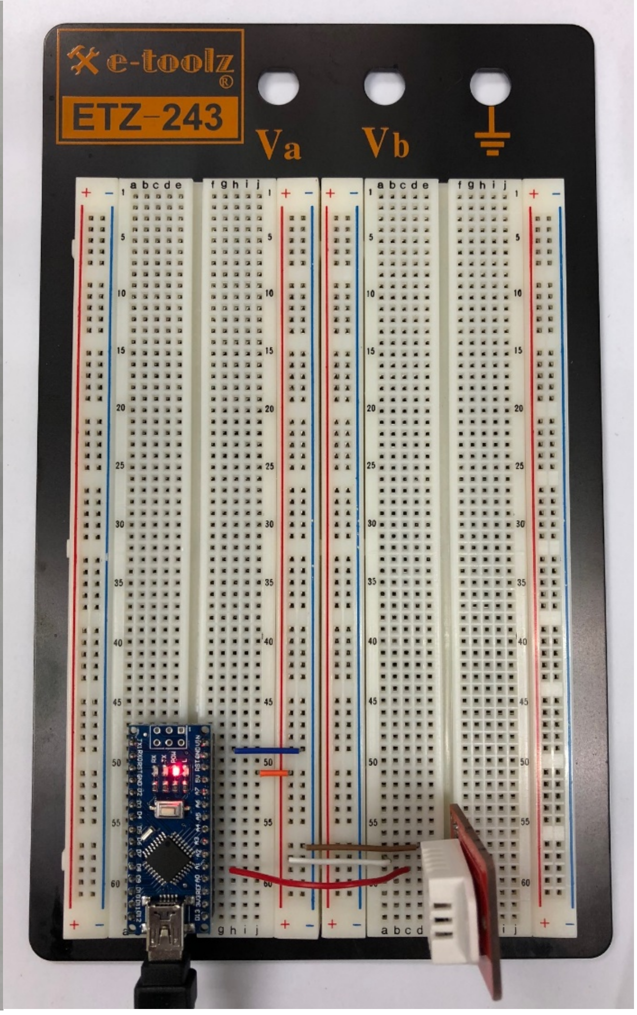
3. Installing a temperature and humidity sensor DHT22

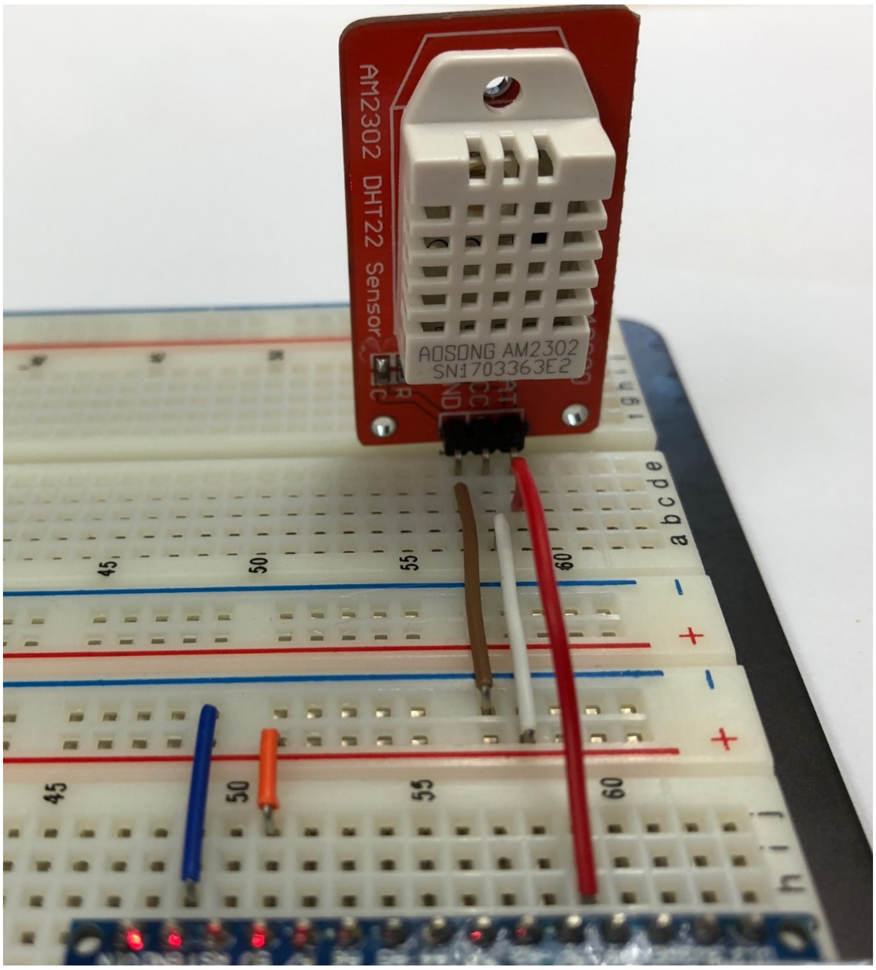
3.1 Wiring

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

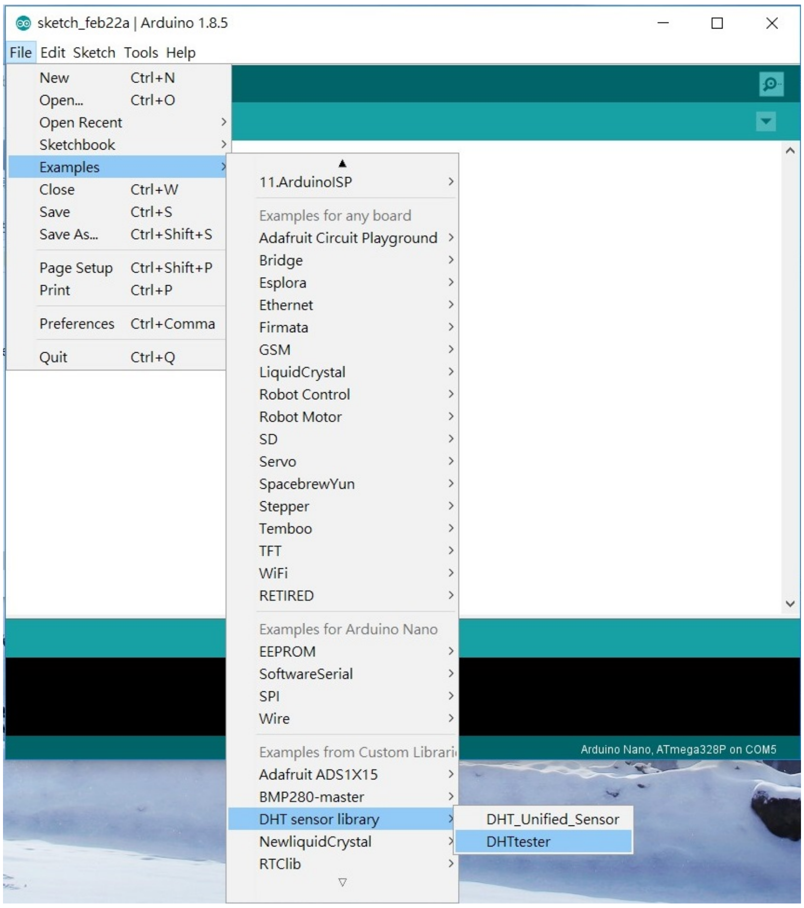






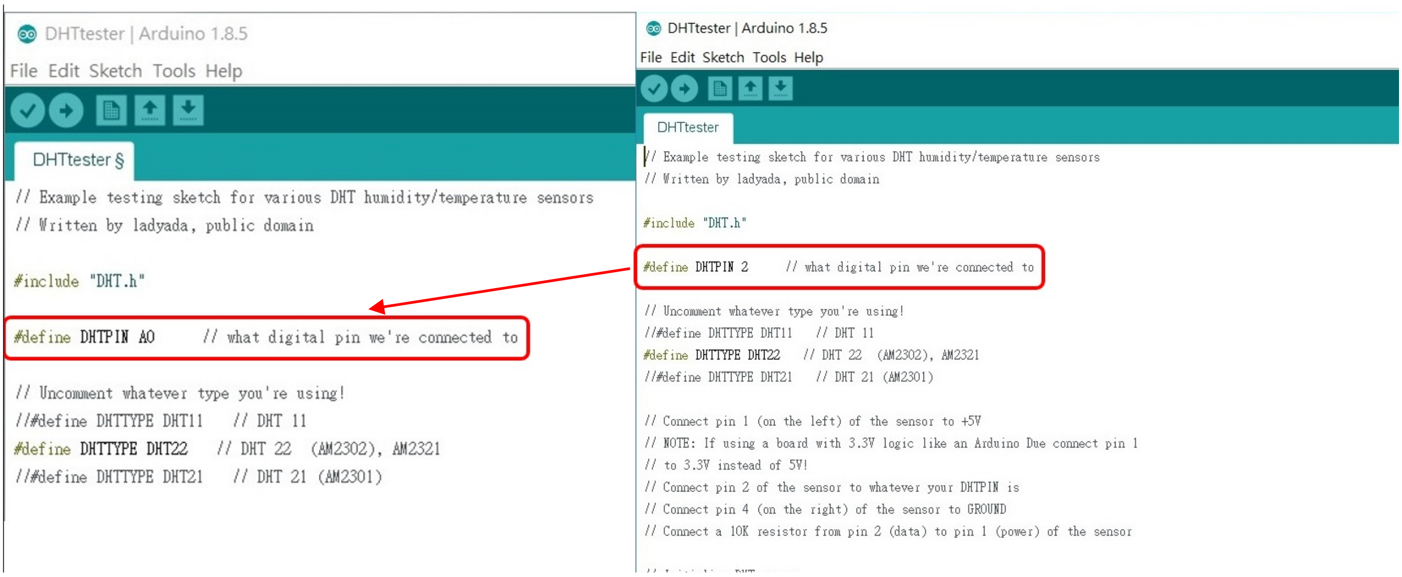
3.2 Choosing an Example file in the DHT sensor library

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



3.3 Modifying the Analog input pin to display correctly

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

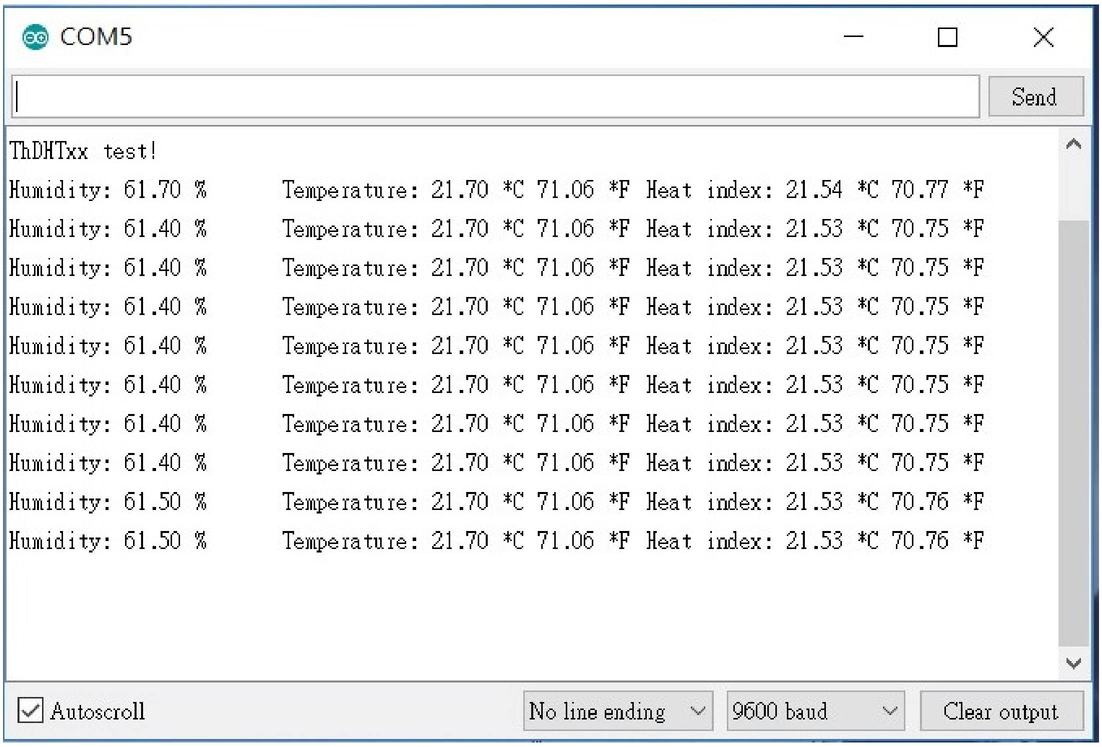

DHTtester | Arduino 1.8.5
File Edit Sketch Tools Help
DHTtester §

```
// 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.

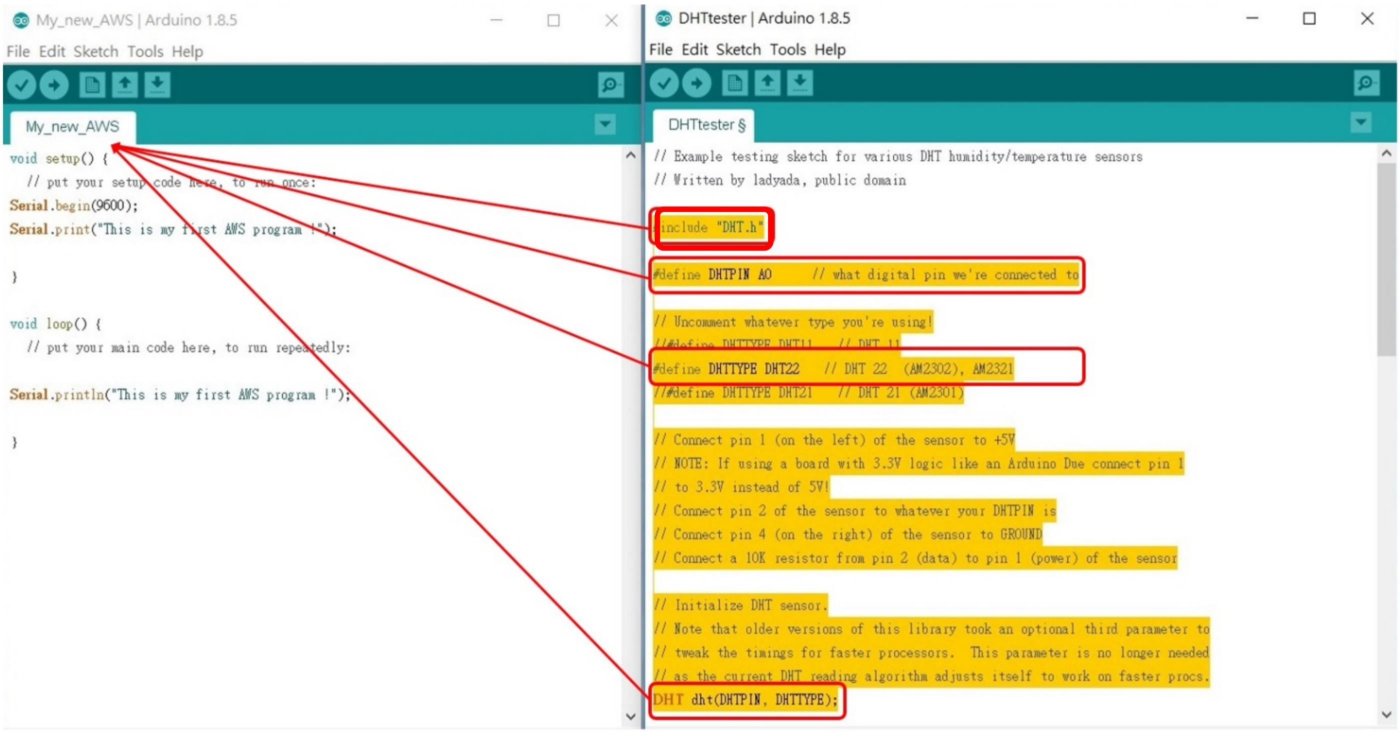


```
ThDHTxx test!
Humidity: 61.70 %	Temperature: 21.70 *C 71.06 *F	Heat index: 21.54 *C 70.77 *F
Humidity: 61.40 %	Temperature: 21.70 *C 71.06 *F	Heat index: 21.53 *C 70.75 *F
Humidity: 61.40 %	Temperature: 21.70 *C 71.06 *F	Heat index: 21.53 *C 70.75 *F
Humidity: 61.40 %	Temperature: 21.70 *C 71.06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

3.4 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:

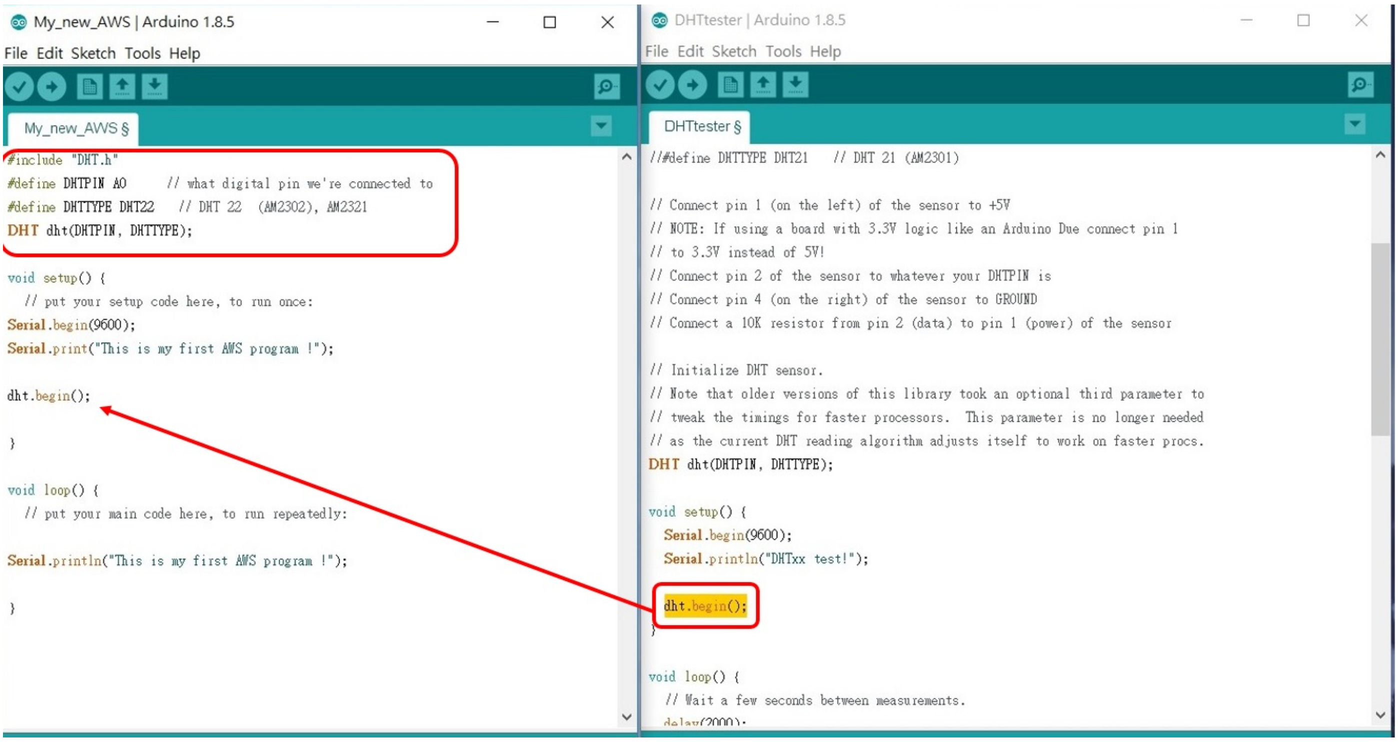
```
#include "DHT.h"
#define DHTPIN A0     // what digital pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
DHT dht(DHTPIN, DHTTYPE);
```


My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS
DHTtester | Arduino 1.8.5
File Edit Sketch Tools Help
DHTtester §

```
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 !");

}
```

```
// 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)

// 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);
```

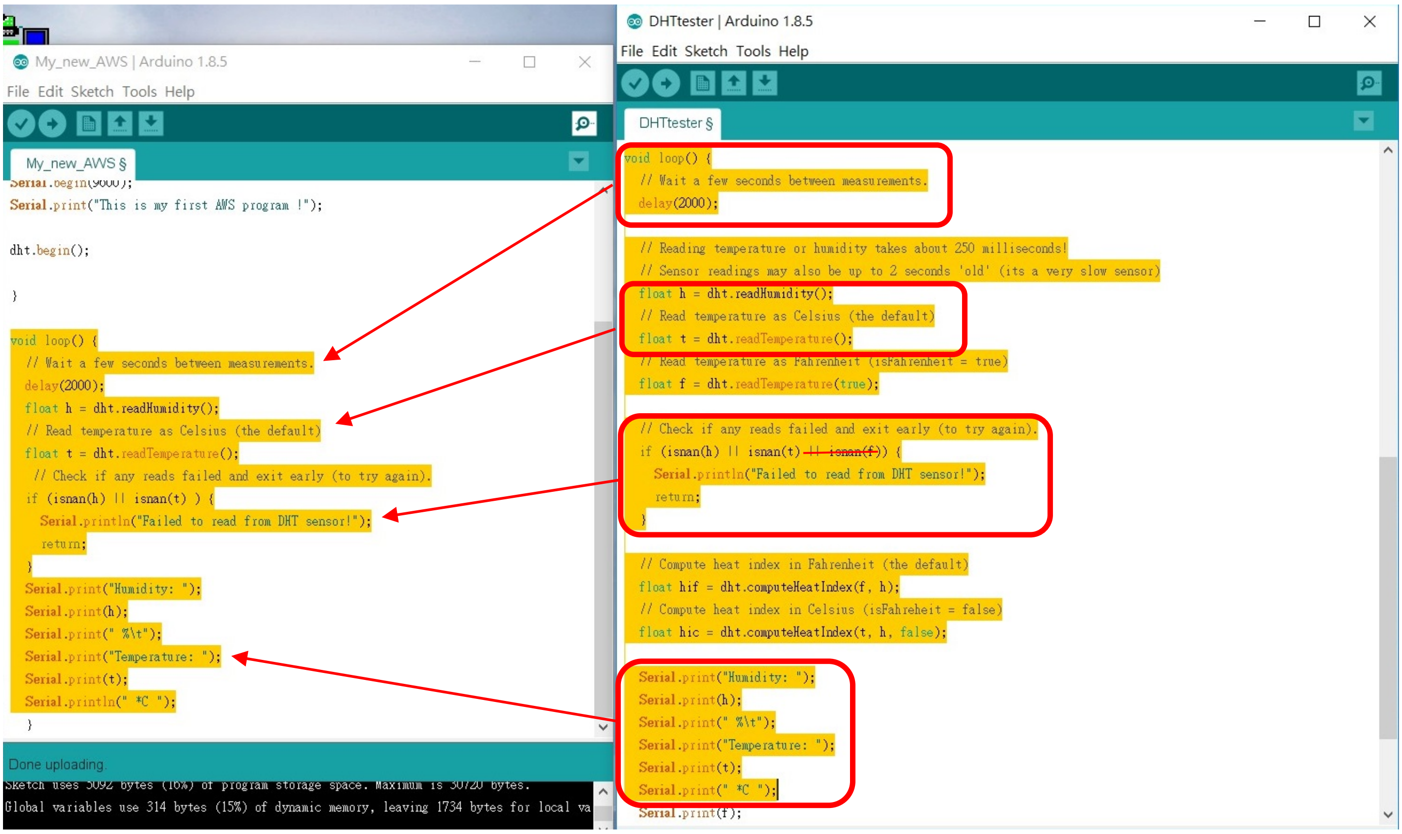
Copy the following line to the setup section of My_new_AWS program: **dht.begin();** as shown below:


My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
DHTtester | Arduino 1.8.5
File Edit Sketch Tools Help
DHTtester §

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

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

dht.begin();

}

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

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

}
```

```
//#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();
}

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

Copy the following lines to the loop() section of My_new_AWS as shown below:

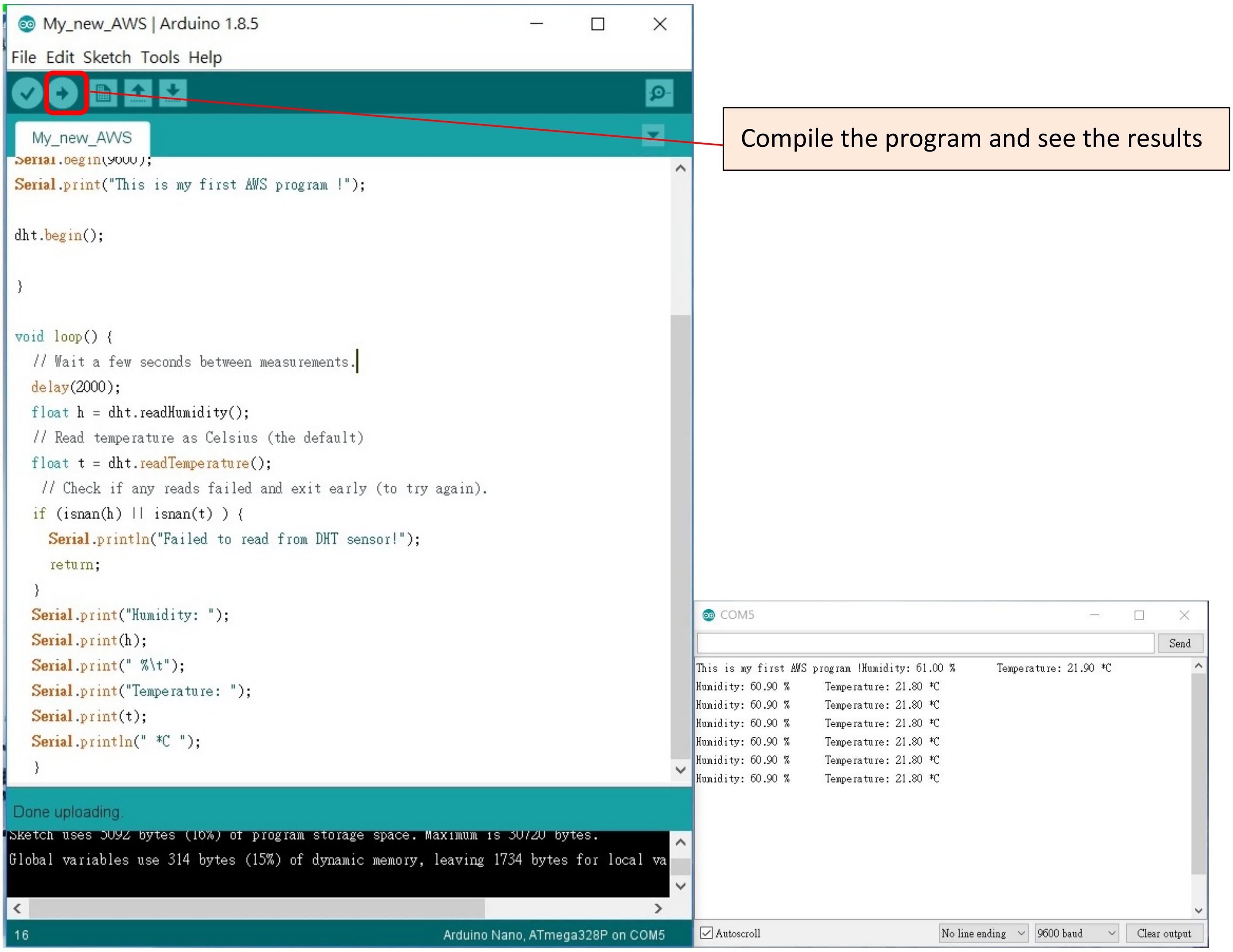

My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
DHTtester | Arduino 1.8.5
DHTtester §

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

dht.begin();

}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
   // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");
  }
```

Done uploading.

```
Sketch uses 5092 bytes (16%) of program storage space. Maximum is 30720 bytes.
Global variables use 314 bytes (15%) of dynamic memory, leaving 1734 bytes for local va
```

```
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 (isFahreheit = 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 ");
  Serial.print(f);
```



Compile the program and see the results

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

dht.begin();

}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
   // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");
  }
```

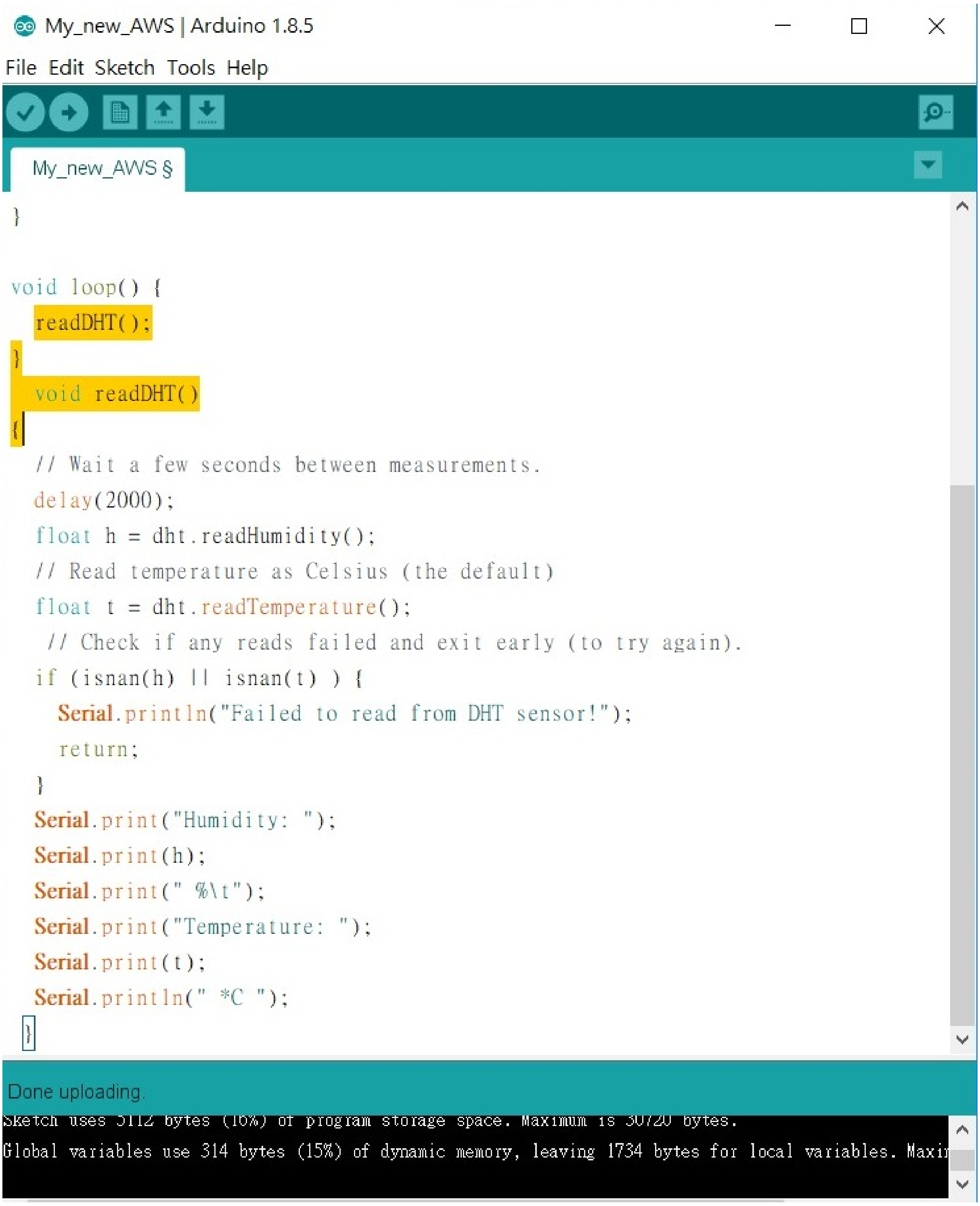
Done uploading.

```
Sketch uses 5092 bytes (16%) of program storage space. Maximum is 30720 bytes.
Global variables use 314 bytes (15%) of dynamic memory, leaving 1734 bytes for local va
```

16   Arduino Nano, ATmega328P on COM5

```
This is my first AWS program !Humidity: 61.00 %     Temperature: 21.90 *C
Humidity: 60.90 %      Temperature: 21.80 *C
Humidity: 60.90 %      Temperature: 21.80 *C
Humidity: 60.90 %      Temperature: 21.80 *C
Humidity: 60.90 %      Temperature: 21.80 *C
Humidity: 60.90 %      Temperature: 21.80 *C
Humidity: 60.90 %      Temperature: 21.80 *C
```

Create a subroutine readDHT() by adding it in loop() as shown:



```
}

void loop() {
  readDHT();
}
  void readDHT()
{
  // Wait a few seconds between measurements.
  delay(2000);
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
   // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");
 }
```

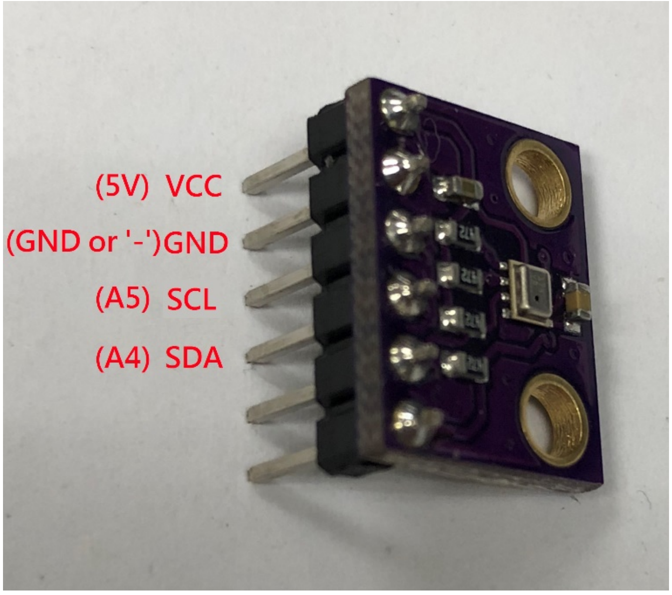
```
Done uploading.
Sketch uses 5112 bytes (16%) of program storage space. Maximum is 30720 bytes.
Global variables use 314 bytes (15%) of dynamic memory, leaving 1734 bytes for local variables. Maxi
```

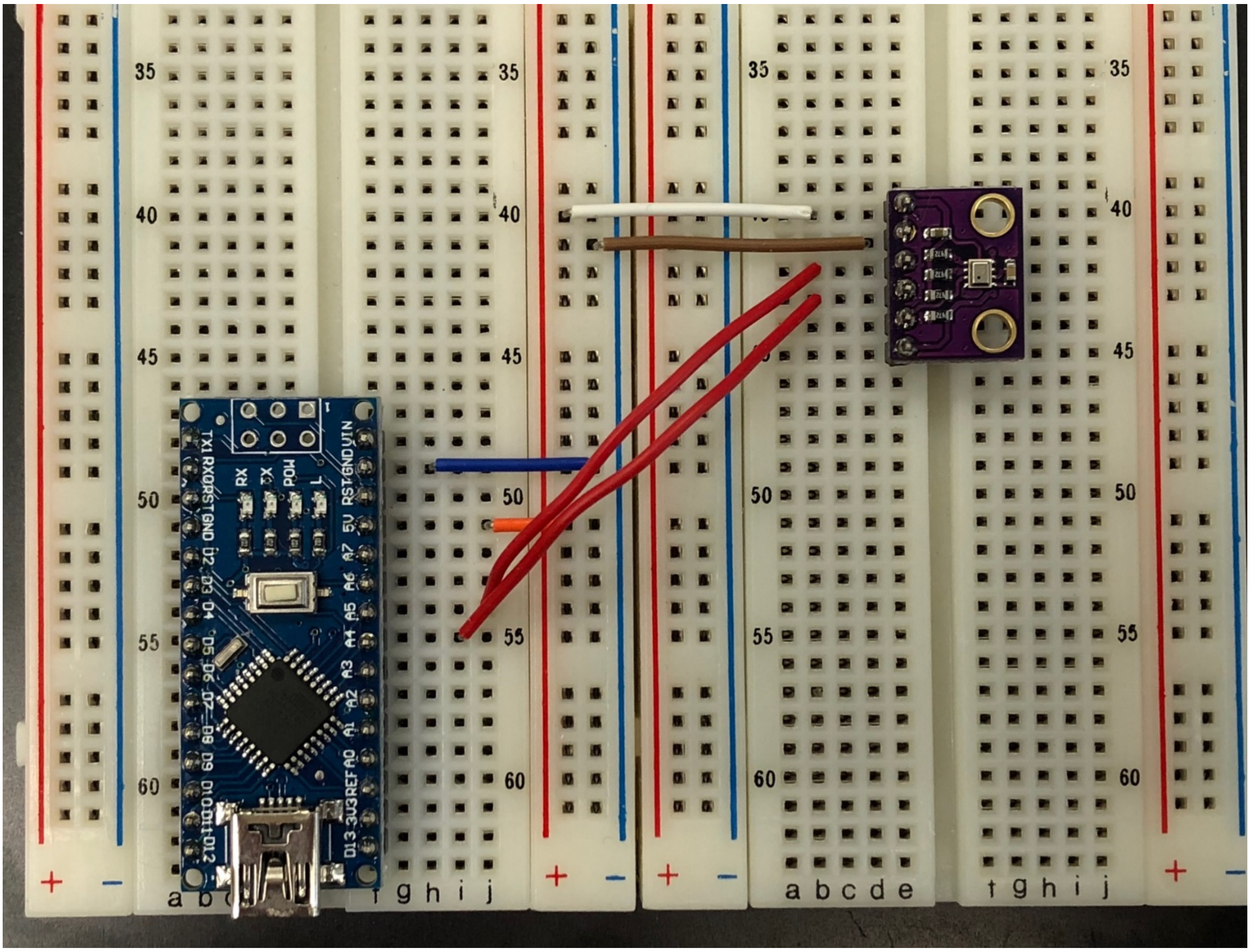
Remember to save your My_new_AWS file again!

4. Installing a Pressure sensor BMP280

4.1 Wiring

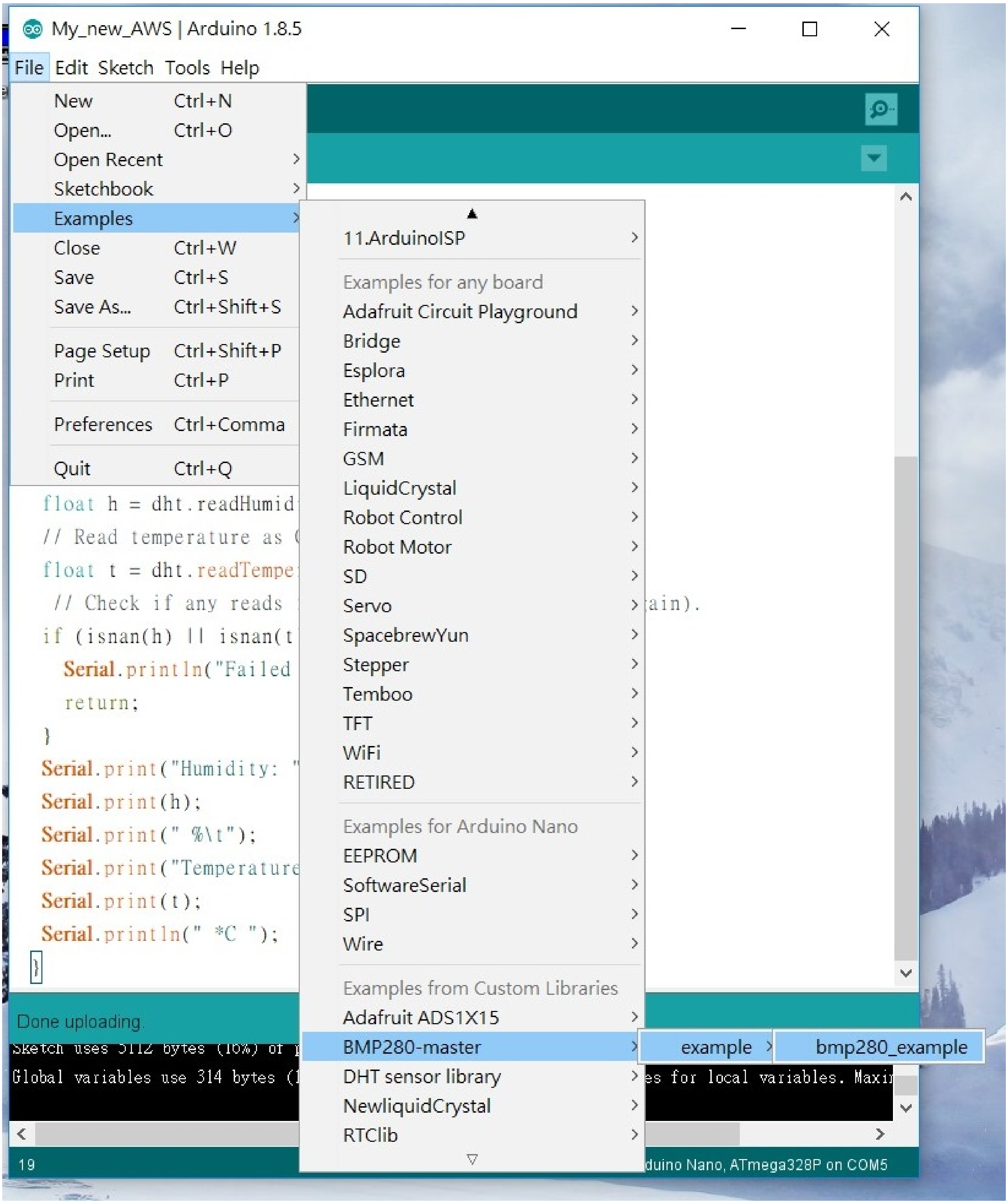
1. Connect BMP280 Vcc to 5V and GND to ground respectively
2. Connect BMP280 SCL to Adruino A5
3. Connect BMP280 SDA to Adruino A4





4.2 Choosing an Example file in BMP280 library

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

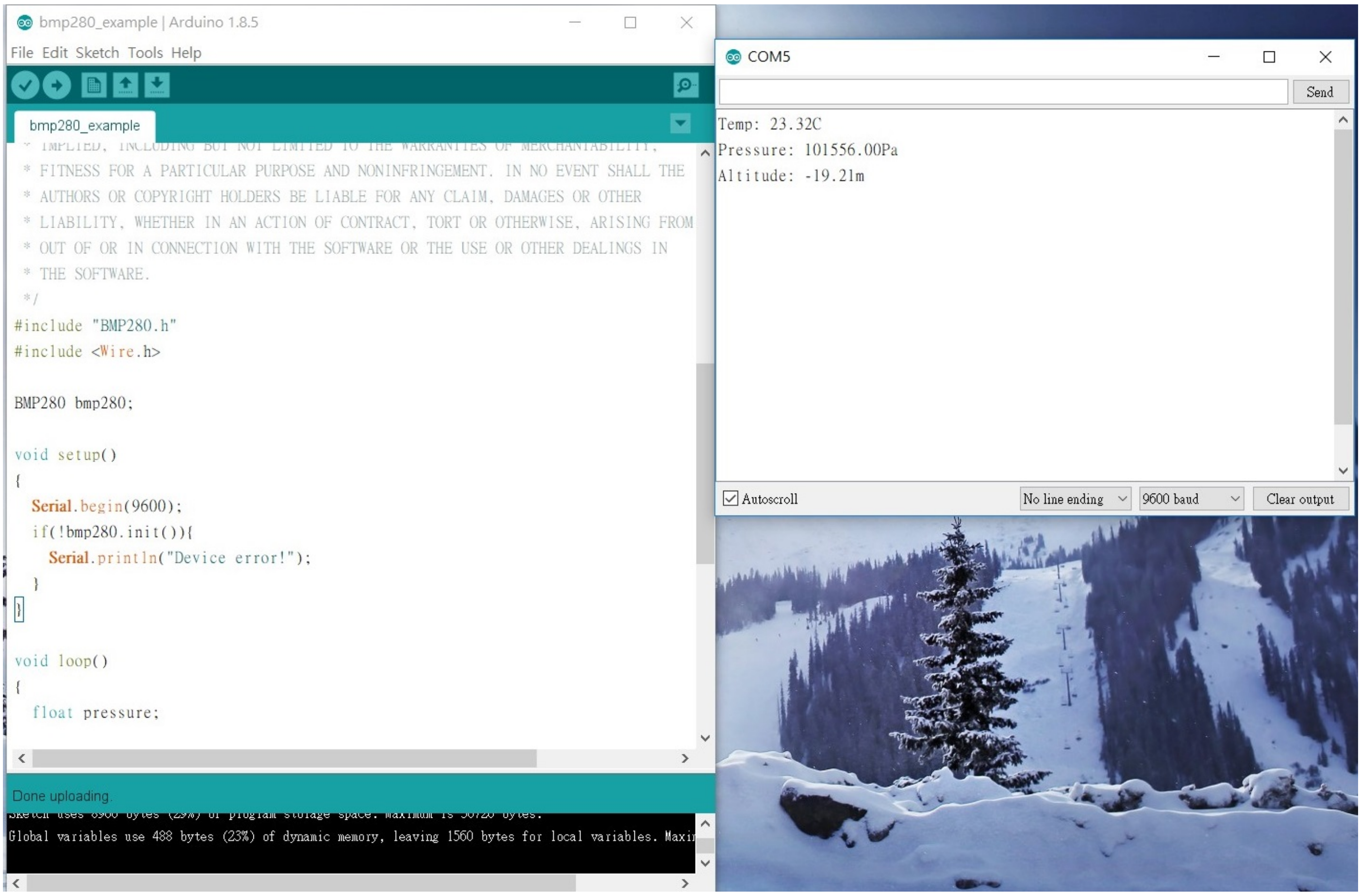


```
  float h = dht.readHumid
  // Read temperature as (
  float t = dht.readTempe
   // Check if any reads                                   ain).
  if (isnan(h) || isnan(t
    Serial.println("Failed
    return;
  }
  Serial.print("Humidity: "
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature
  Serial.print(t);
  Serial.println(" *C ");
}
```

```
Done uploading.
Sketch uses 5112 bytes (16%) of p
Global variables use 314 bytes (1                          es for local variables. Maxi
```

```
19                                                         duino Nano, ATmega328P on COM5
```

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



bmp280_example | Arduino 1.8.5

```
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
#include "BMP280.h"
#include <Wire.h>

BMP280 bmp280;

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

void loop()
{
  float pressure;
```

Done uploading.

```
Sketch uses 8900 bytes (29%) of program storage space. Maximum is 30720 bytes.
Global variables use 488 bytes (23%) of dynamic memory, leaving 1560 bytes for local variables. Maxi
```

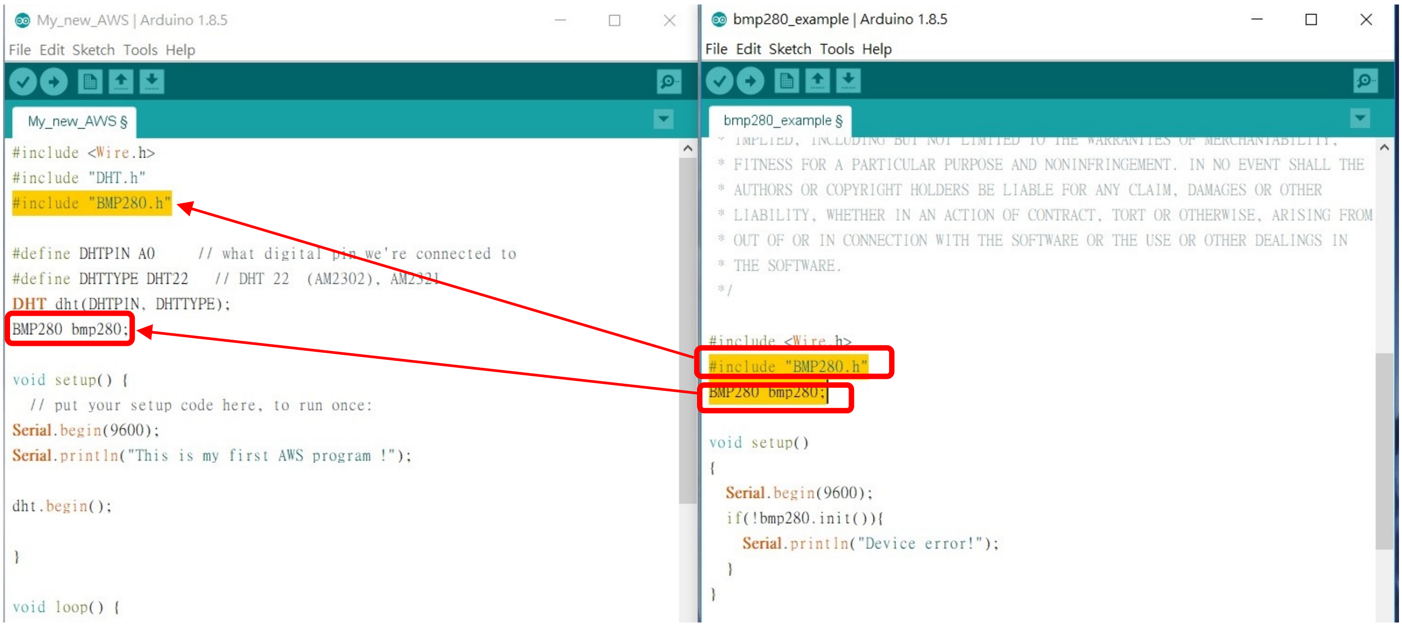
COM5

```
Temp: 23.32C
Pressure: 101556.00Pa
Altitude: -19.21m
```

☑ Autoscroll · No line ending · 9600 baud · Clear output

4.3 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.



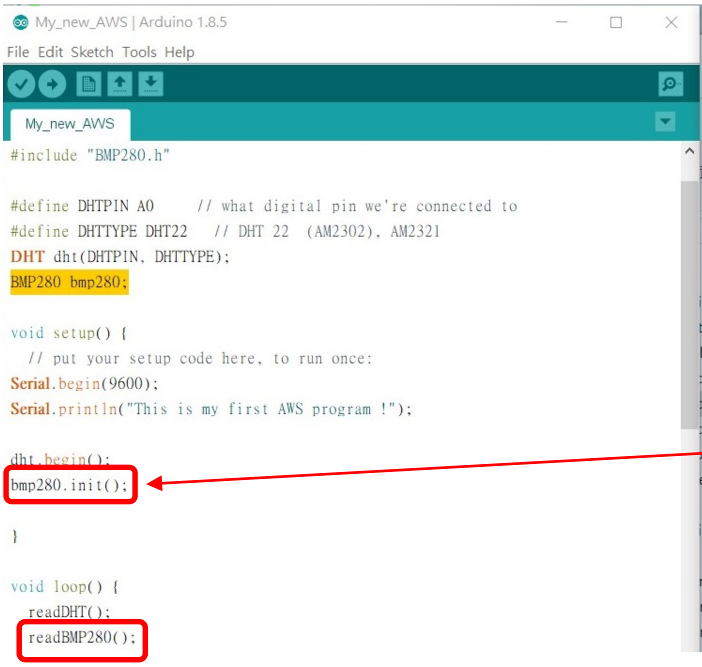
My_new_AWS | Arduino 1.8.5

```
#include <Wire.h>
#include "DHT.h"
#include "BMP280.h"

#define DHTPIN A0     // what digital pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
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();

}

void loop() {
```

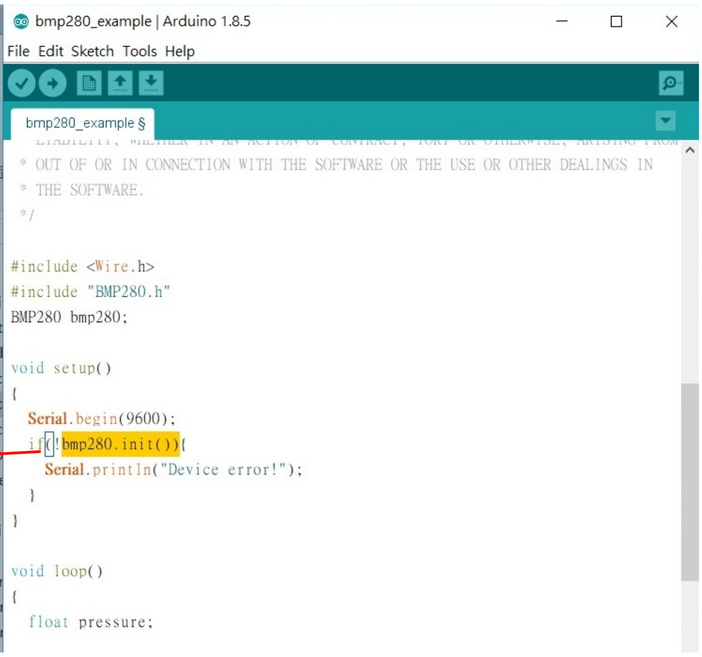
bmp280_example | Arduino 1.8.5

```
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */

#include <Wire.h>
#include "BMP280.h"
BMP280 bmp280;

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

In setup(), add bmp.init(); as shown. In loop(), add readBMP(); as shown.

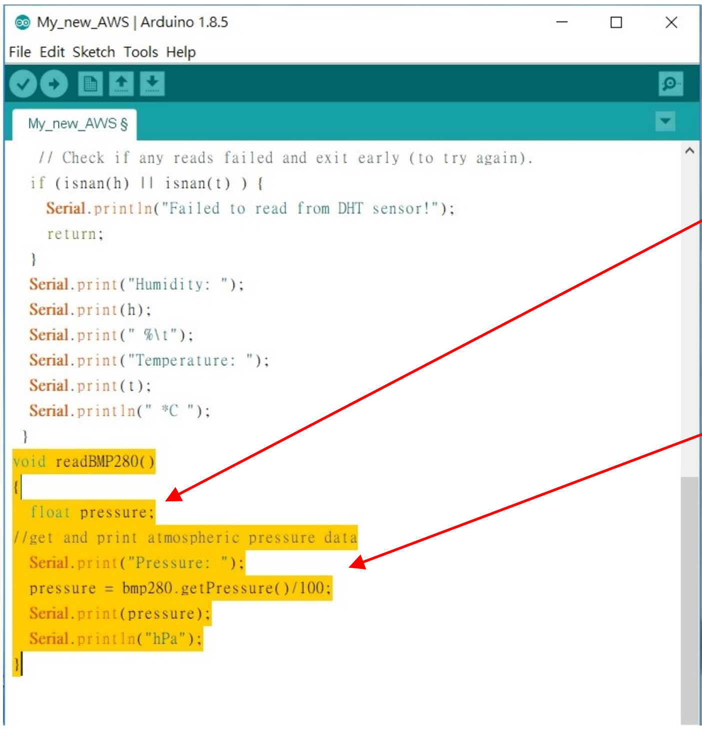


```
#include "BMP280.h"

#define DHTPIN A0     // what digital pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
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() {
  readDHT();
  readBMP280();
```

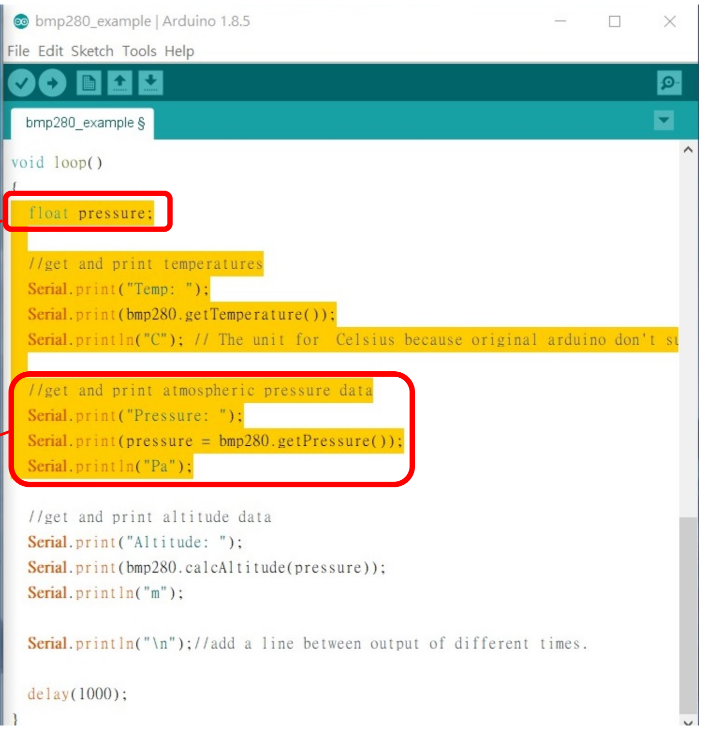


```
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */

#include <Wire.h>
#include "BMP280.h"
BMP280 bmp280;

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

void loop()
{
  float pressure;
```

Create a new subroutine void readBMP280() as shown below:



```
   // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");
 }
void readBMP280()
{
  float pressure;
//get and print atmospheric pressure data
  Serial.print("Pressure: ");
  pressure = bmp280.getPressure()/100;
  Serial.print(pressure);
  Serial.println("hPa");
}
```

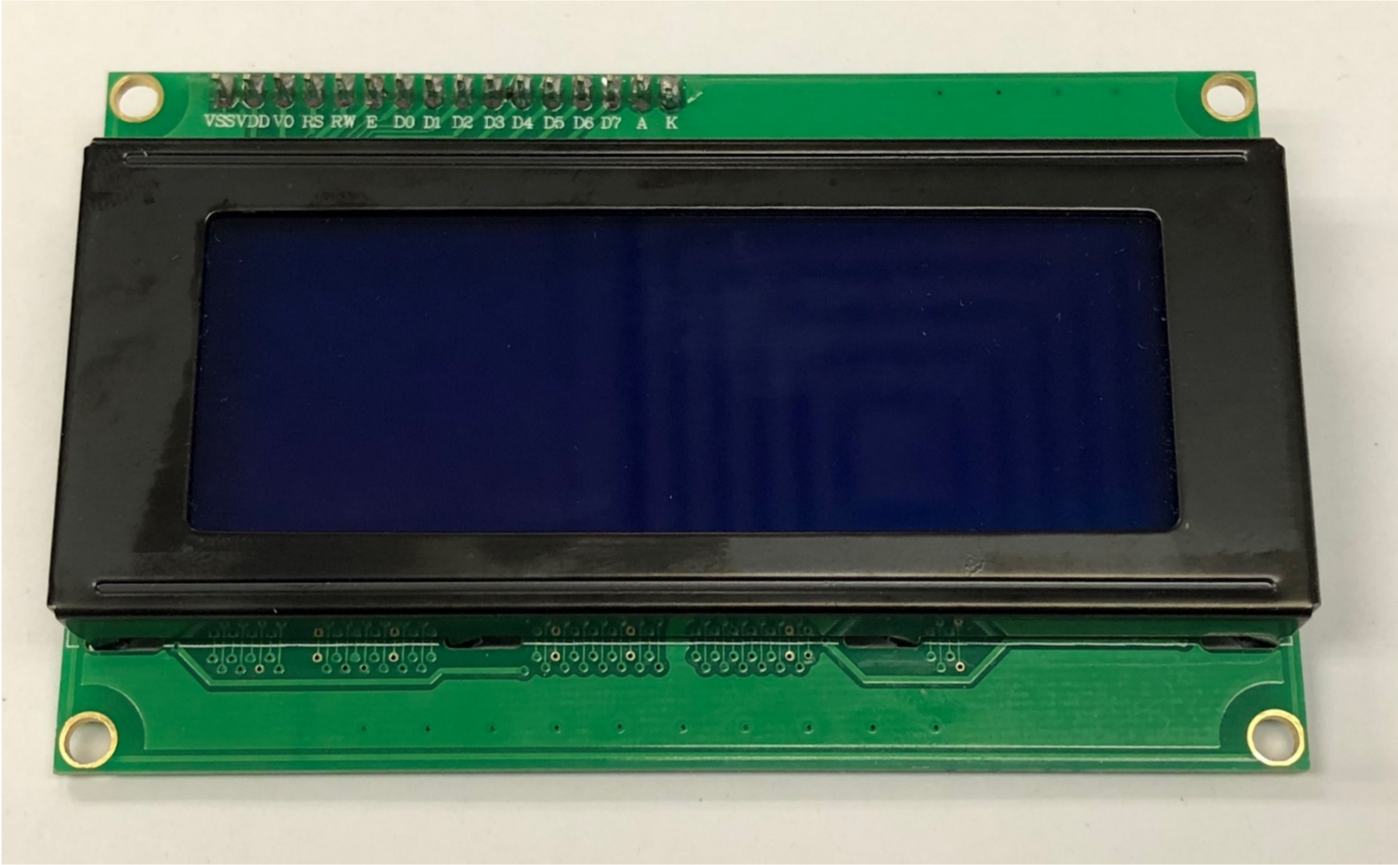


```
void loop()
{
  float pressure;

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

  //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.

  delay(1000);
}
```

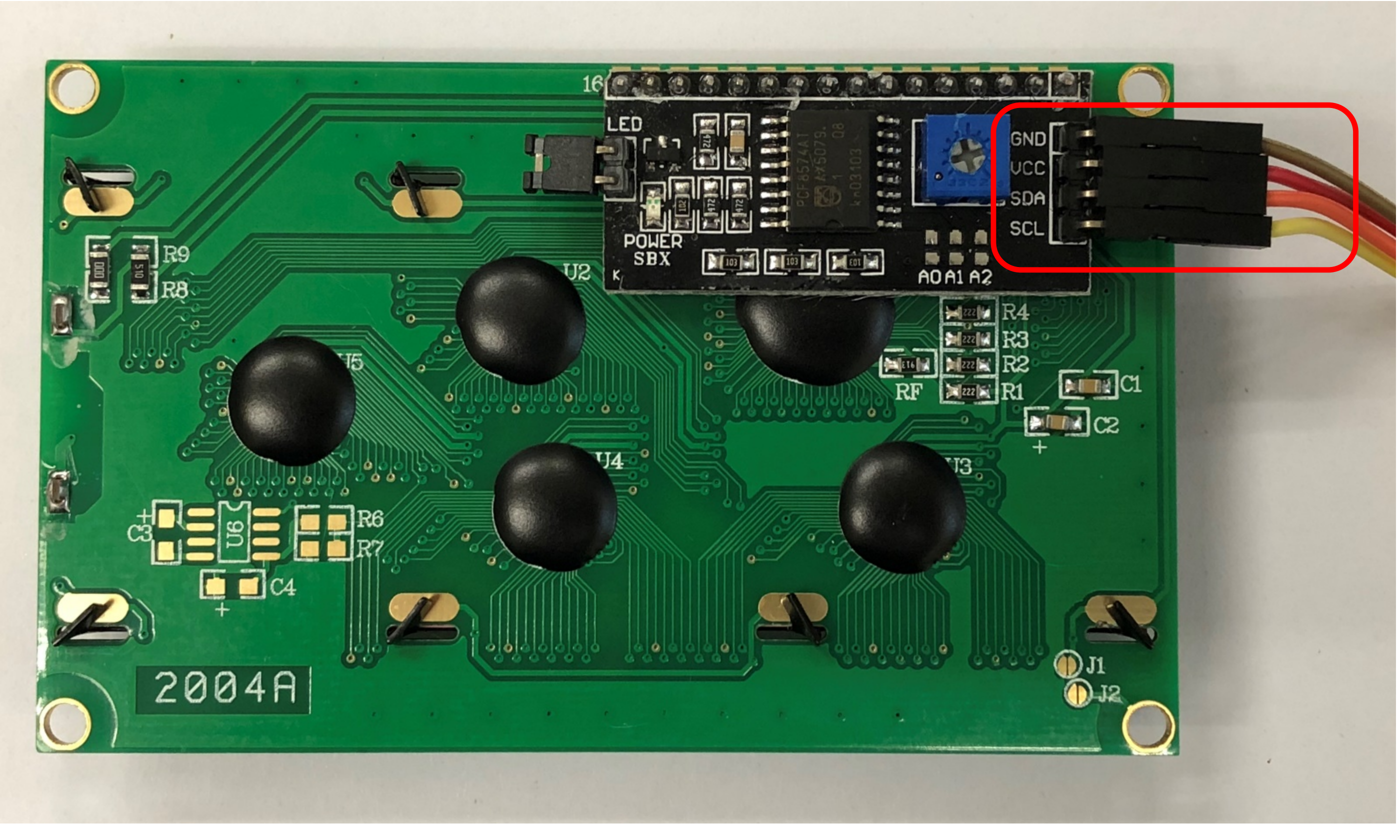
Compile and upload the program to see the results! Remember to save your My_new_AWS file again!

5. Installing a LCD Display

Front view

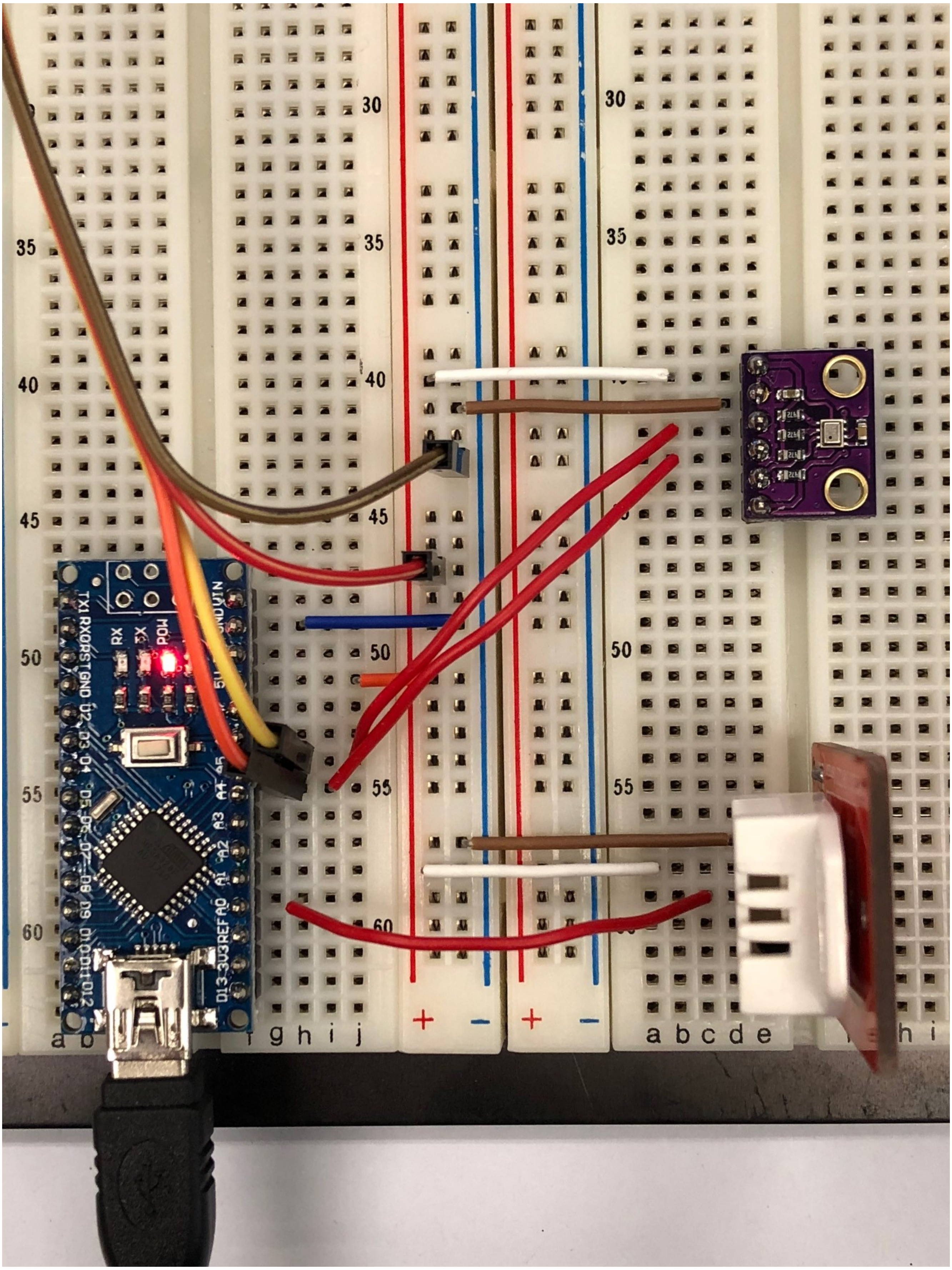


Rear view



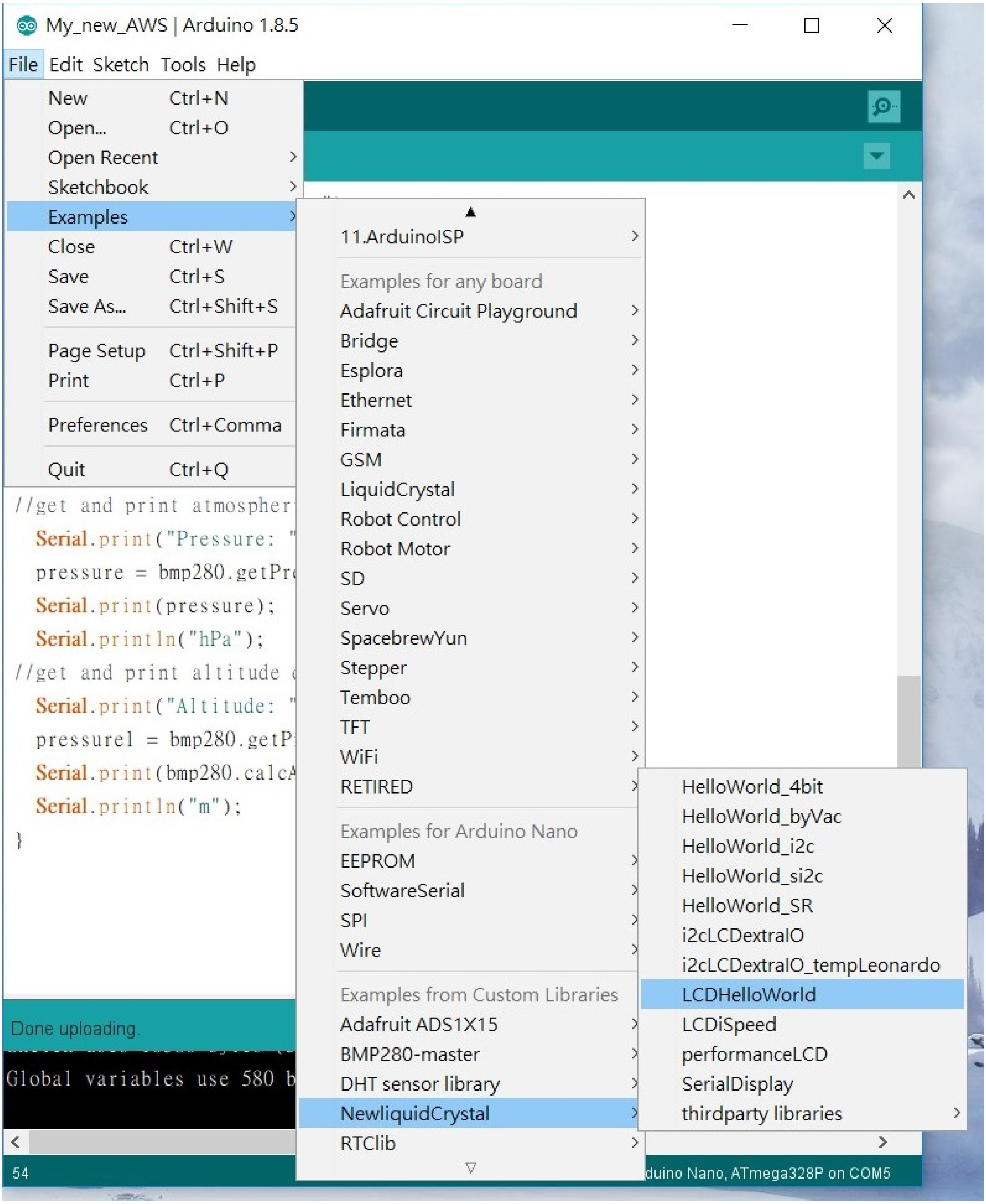
5.1 Wiring

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



5.2 Choosing an Example file in the NewliquidCrystal library

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

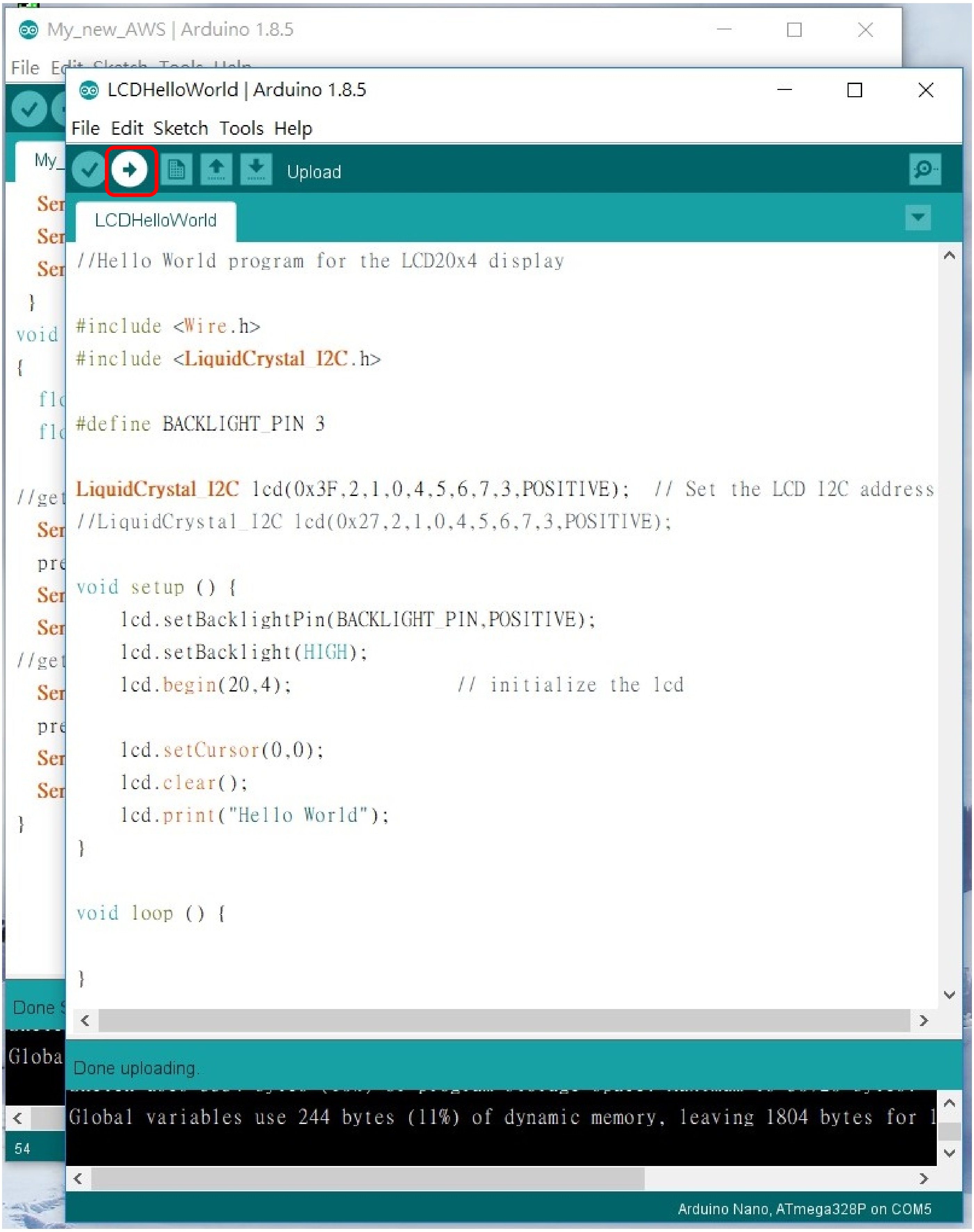


```
//get and print atmospher
  Serial.print("Pressure: "
  pressure = bmp280.getPr
  Serial.print(pressure);
  Serial.println("hPa");
//get and print altitude d
  Serial.print("Altitude: "
  pressure1 = bmp280.getP
  Serial.print(bmp280.calcA
  Serial.println("m");
}
```

Done uploading.

```
Global variables use 580 b
```

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


My_new_AWS | Arduino 1.8.5
LCDHelloWorld | Arduino 1.8.5
File Edit Sketch Tools Help
Upload
LCDHelloWorld

```
//Hello World program for the LCD20x4 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

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

void loop () {

}
```

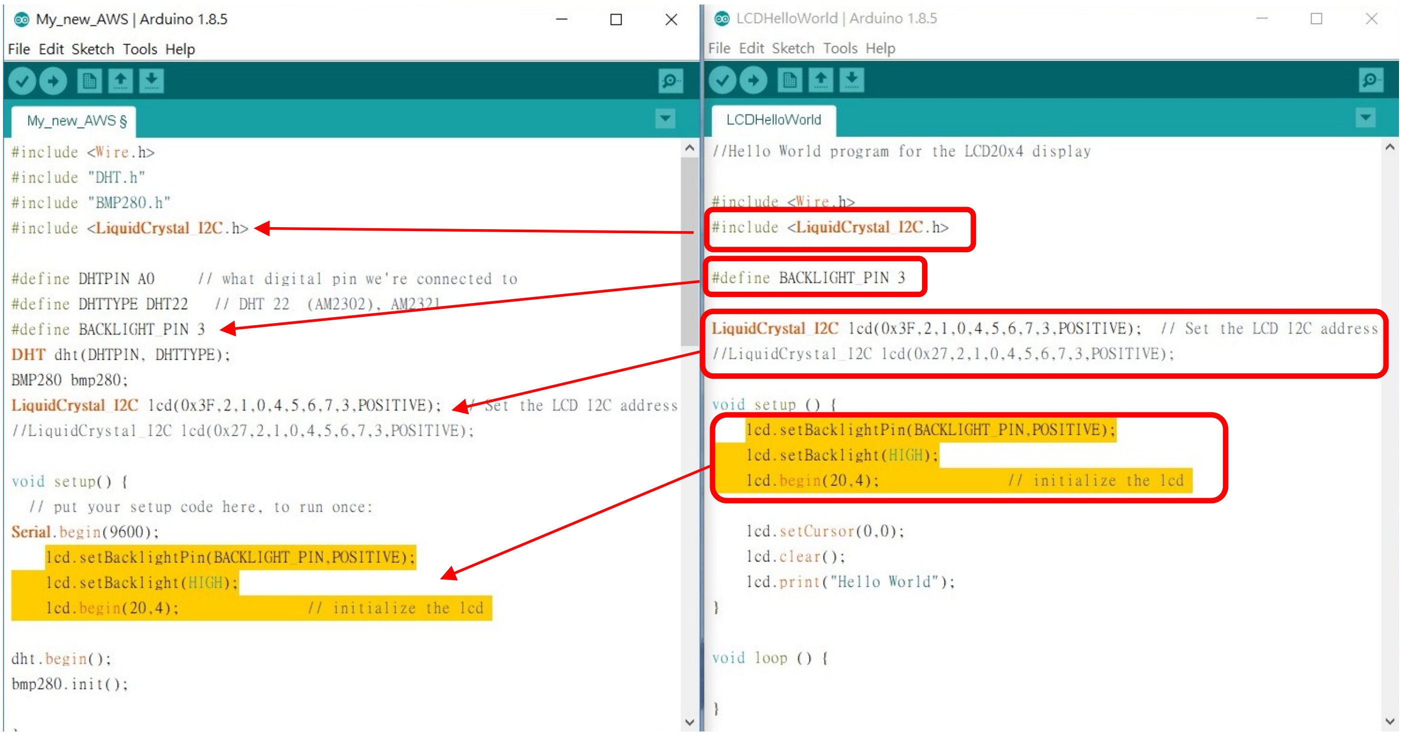
Done uploading.

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

Arduino Nano, ATmega328P on COM5

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

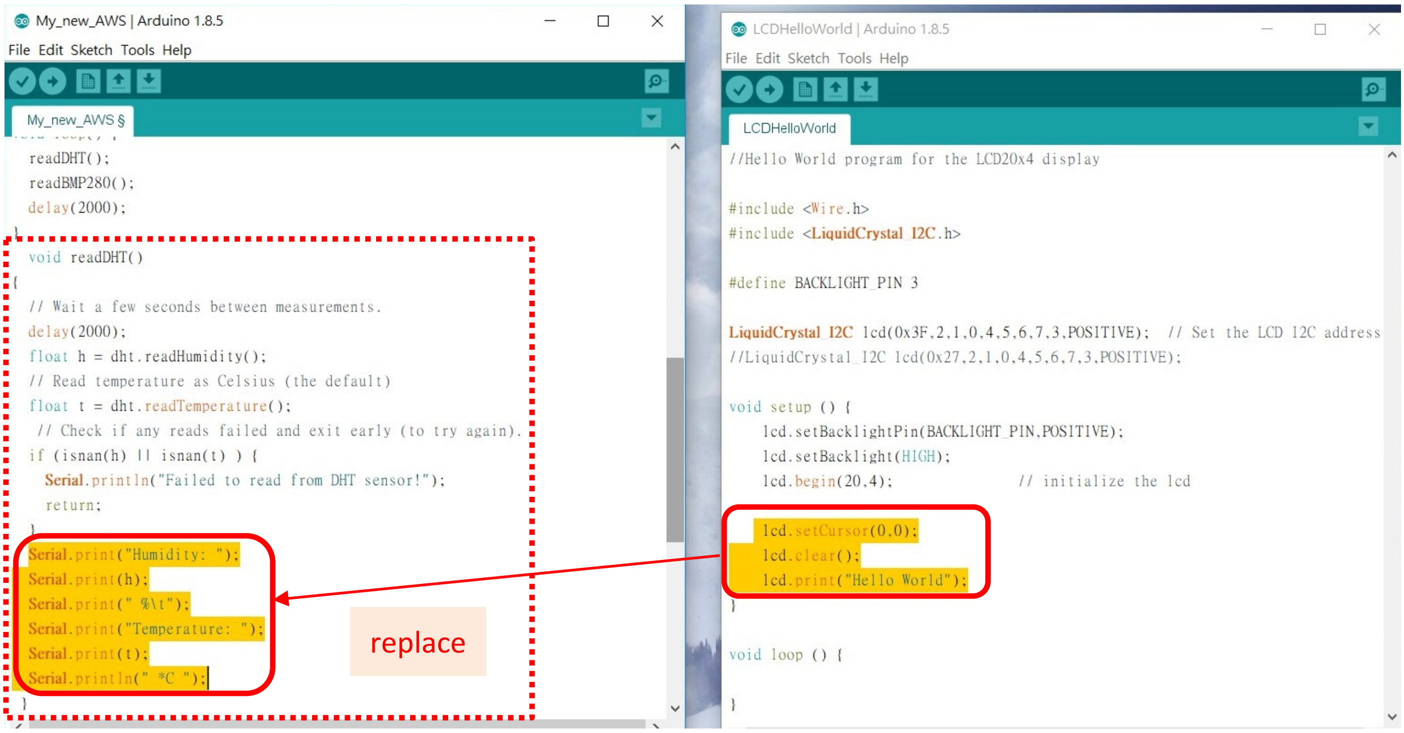
Copy the highlighted lines in the header and setup() sections from LCDHellowWorld to the My_new_AWS.


My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
LCDHelloWorld | Arduino 1.8.5
File Edit Sketch Tools Help
LCDHelloWorld

```
#include <Wire.h>
#include "DHT.h"
#include "BMP280.h"
#include <LiquidCrystal_I2C.h>

#define DHTPIN A0      // what digital pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
#define BACKLIGHT_PIN 3
DHT dht(DHTPIN, DHTTYPE);
BMP280 bmp280;
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() {
  // put your setup code here, to run once:
Serial.begin(9600);
    lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
    lcd.setBacklight(HIGH);
    lcd.begin(20,4);               // initialize the lcd

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

```
//Hello World program for the LCD20x4 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

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

void loop () {


}
```

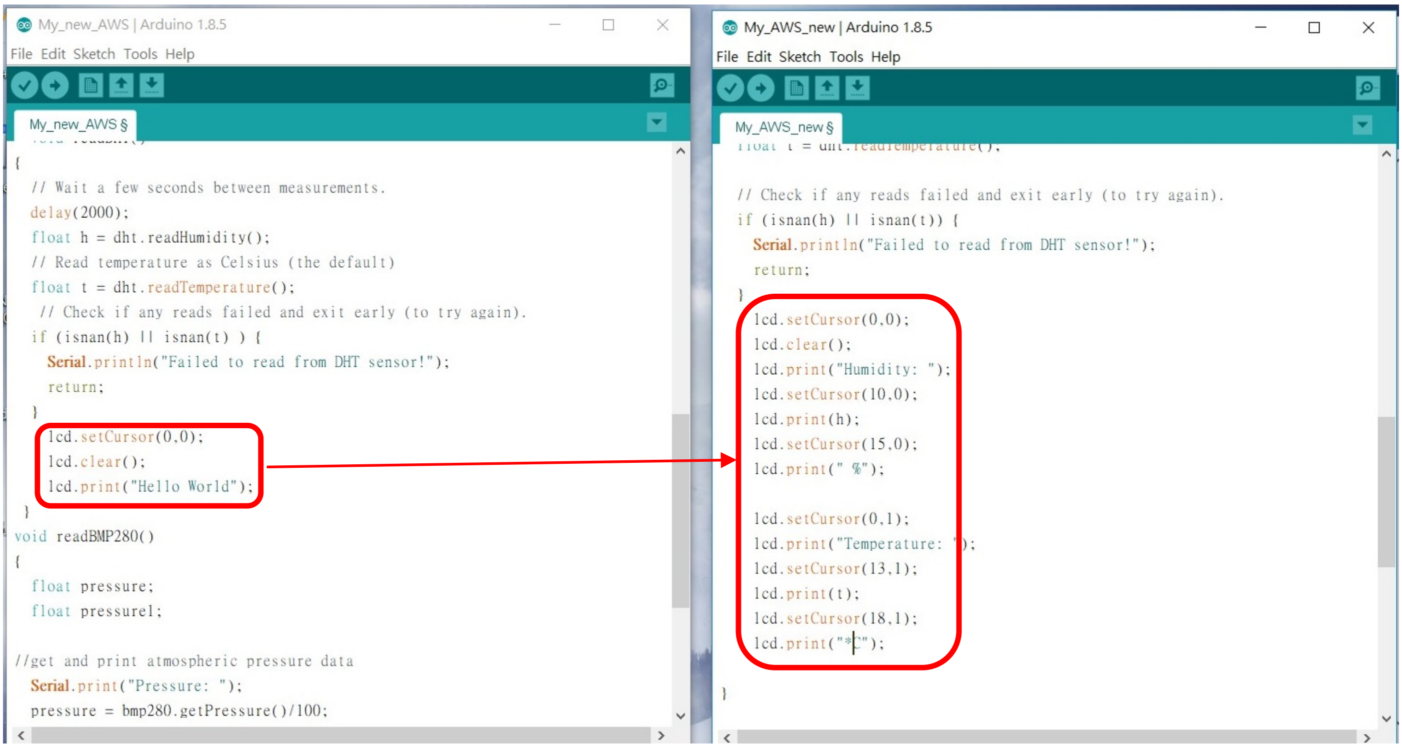
Copy the following 3 highlighted lines from LCDHellowWorld to the readDHT() subroutine.


My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
replace
LCDHelloWorld | Arduino 1.8.5
LCDHelloWorld

```
  readDHT();
  readBMP280();
  delay(2000);
}
  void readDHT()
{
  // Wait a few seconds between measurements.
  delay(2000);
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
   // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");
}
```

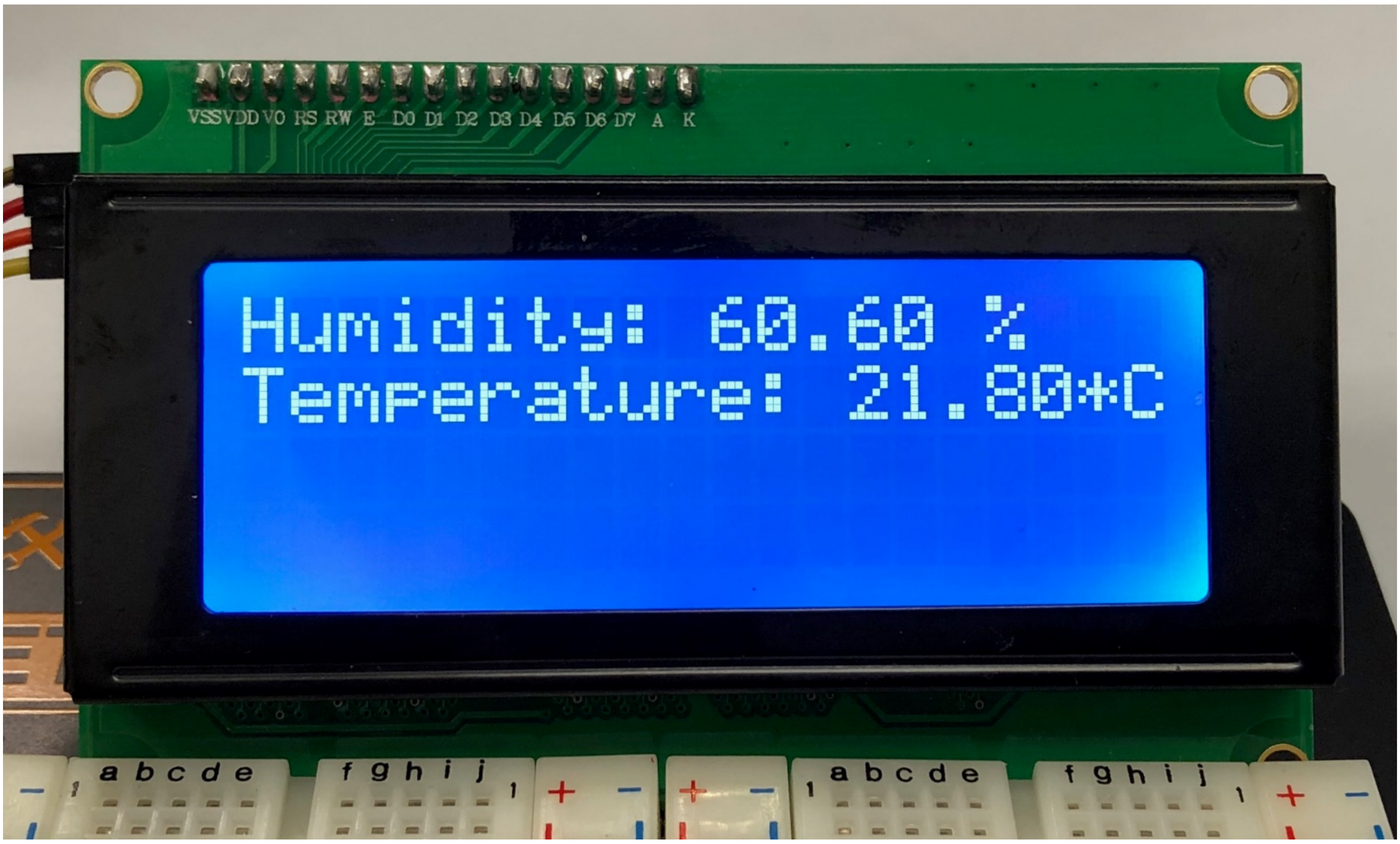
```
//Hello World program for the LCD20x4 display

#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

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

void loop () {

}
```

In the readDHT() subroutine, change the code in My_new_AWS as shown below.

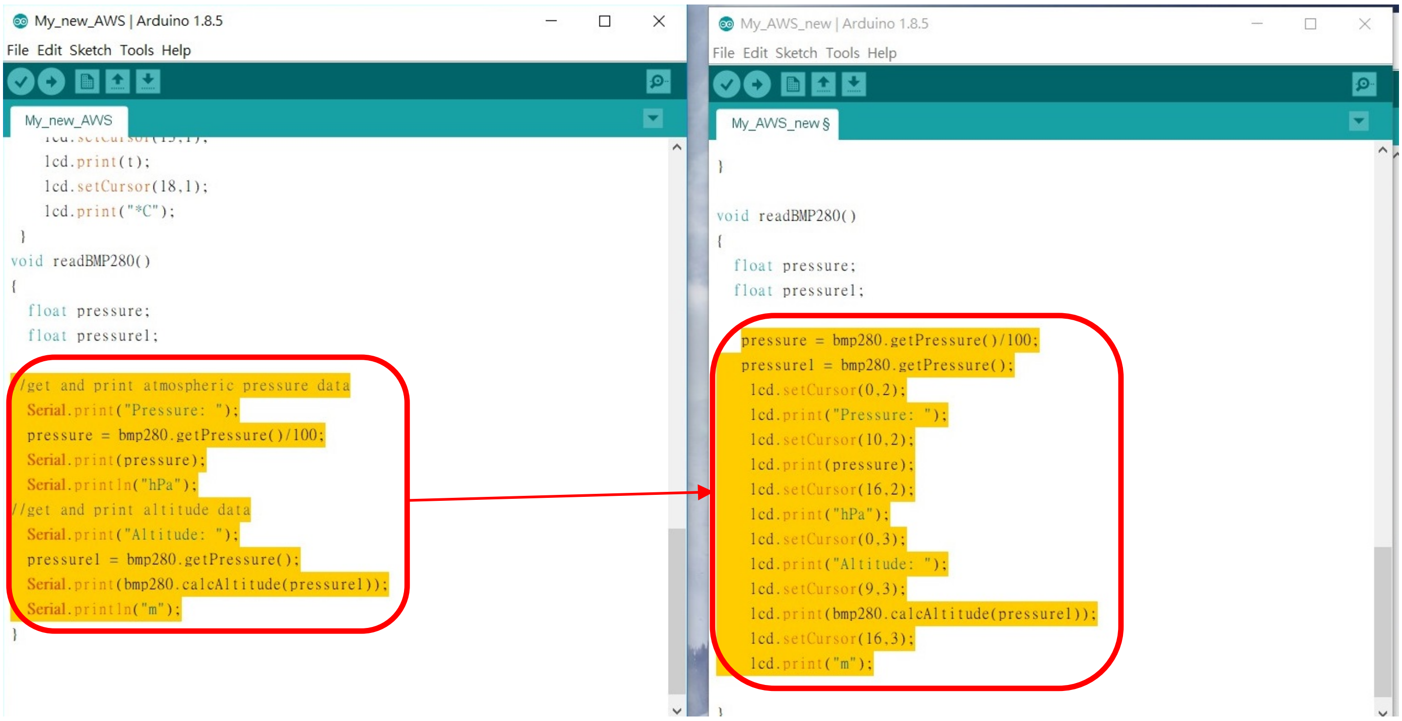

My_new_AWS | Arduino 1.8.5
File Edit Sketch Tools Help
My_new_AWS §
My_AWS_new | Arduino 1.8.5
My_AWS_new §

```
{
  // Wait a few seconds between measurements.
  delay(2000);
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
   // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
    lcd.setCursor(0,0);
    lcd.clear();
    lcd.print("Hello World");
 }
void readBMP280()
{
  float pressure;
  float pressure1;

//get and print atmospheric pressure data
  Serial.print("Pressure: ");
  pressure = bmp280.getPressure()/100;
```

```
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
    lcd.setCursor(0,0);
    lcd.clear();
    lcd.print("Humidity: ");
    lcd.setCursor(10,0);
    lcd.print(h);
    lcd.setCursor(15,0);
    lcd.print(" %");

    lcd.setCursor(0,1);
    lcd.print("Temperature: ");
    lcd.setCursor(13,1);
    lcd.print(t);
    lcd.setCursor(18,1);
    lcd.print("*C");

}
```



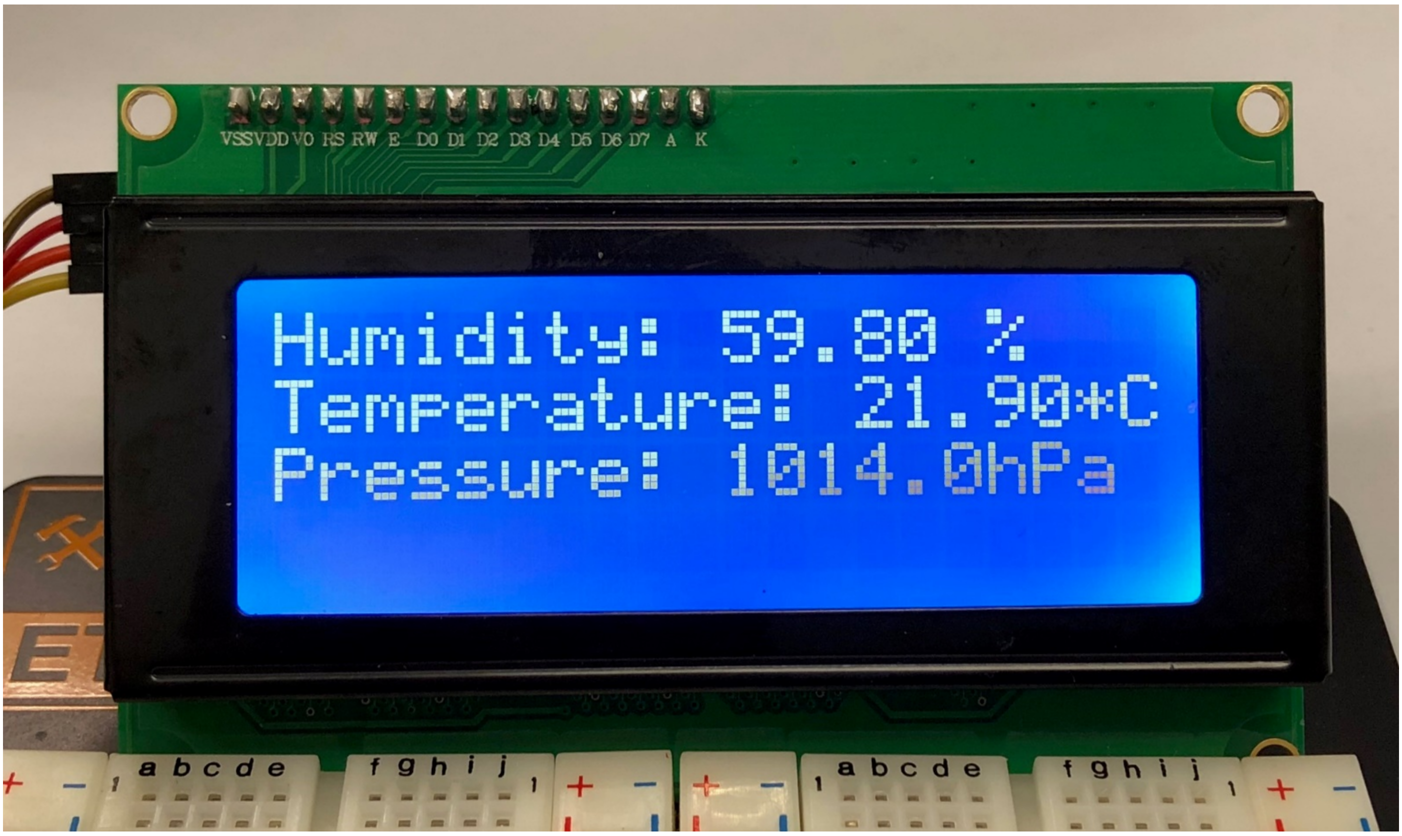
In the readBMP280() subroutine, change the code in My_new_AWS as shown below.



```
    lcd.setCursor(13,1);
    lcd.print(t);
    lcd.setCursor(18,1);
    lcd.print("*C");
  }
void readBMP280()
{
  float pressure;
  float pressure1;

 /get and print atmospheric pressure data
  Serial.print("Pressure: ");
  pressure = bmp280.getPressure()/100;
  Serial.print(pressure);
  Serial.println("hPa");
//get and print altitude data
  Serial.print("Altitude: ");
  pressure1 = bmp280.getPressure();
  Serial.print(bmp280.calcAltitude(pressure1));
  Serial.println("m");
}
```

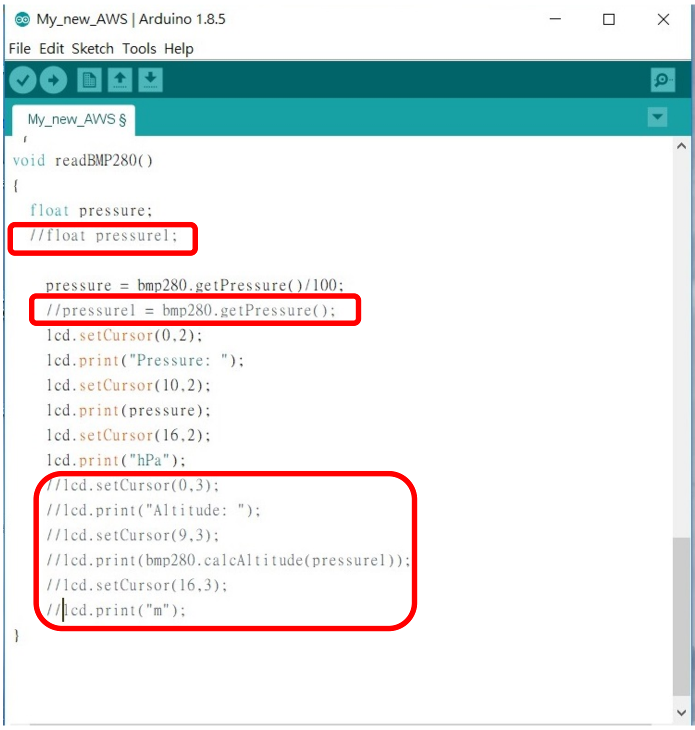
```
}

void readBMP280()
{
  float pressure;
  float pressure1;

  pressure = bmp280.getPressure()/100;
  pressure1 = bmp280.getPressure();
   lcd.setCursor(0,2);
   lcd.print("Pressure: ");
   lcd.setCursor(10,2);
   lcd.print(pressure);
   lcd.setCursor(16,2);
   lcd.print("hPa");
   lcd.setCursor(0,3);
   lcd.print("Altitude: ");
   lcd.setCursor(9,3);
   lcd.print(bmp280.calcAltitude(pressure1));
   lcd.setCursor(16,3);
   lcd.print("m");

}
```

Compile and upload the program to Arduino Nano and see the LCD display.

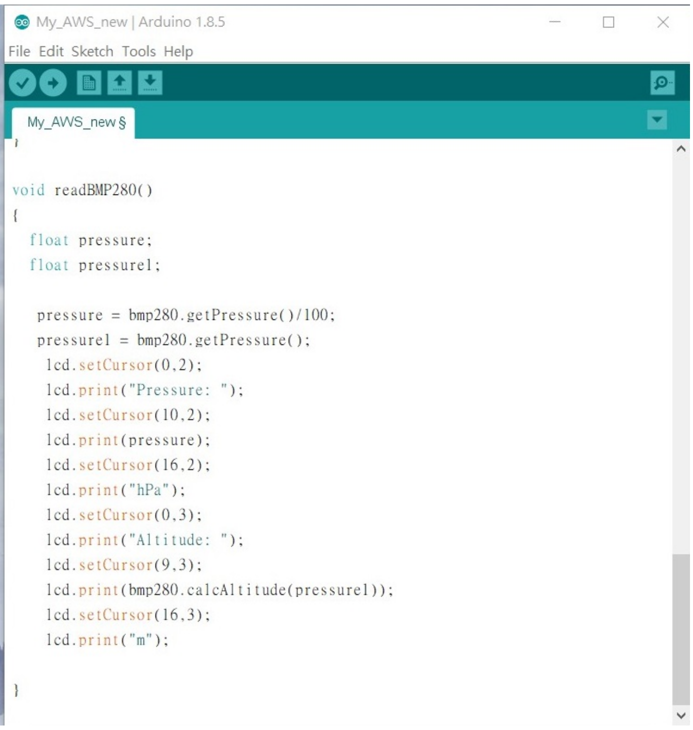


```
Humidity: 59.80 %
Temperature: 21.90*C
Pressure: 1014.0hPa
```

Try adding "//" for lines on the left screen and compare the effects on the LCD display.

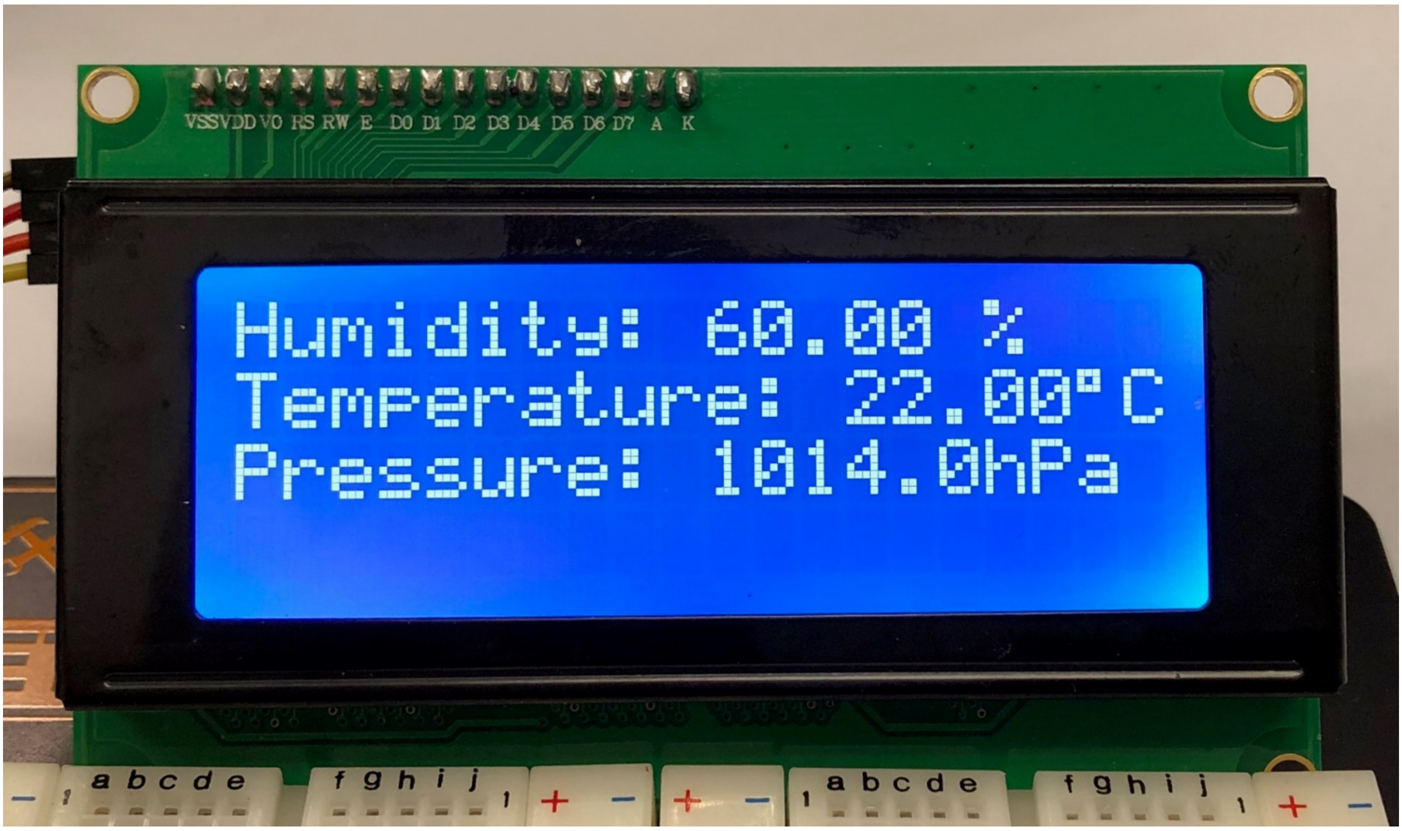


```
void readBMP280()
{
  float pressure;
  //float pressure1;

    pressure = bmp280.getPressure()/100;
    //pressure1 = bmp280.getPressure();
    lcd.setCursor(0,2);
    lcd.print("Pressure: ");
    lcd.setCursor(10,2);
    lcd.print(pressure);
    lcd.setCursor(16,2);
    lcd.print("hPa");
    //lcd.setCursor(0,3);
    //lcd.print("Altitude: ");
    //lcd.setCursor(9,3);
    //lcd.print(bmp280.calcAltitude(pressure1));
    //lcd.setCursor(16,3);
    //lcd.print("m");
}
```



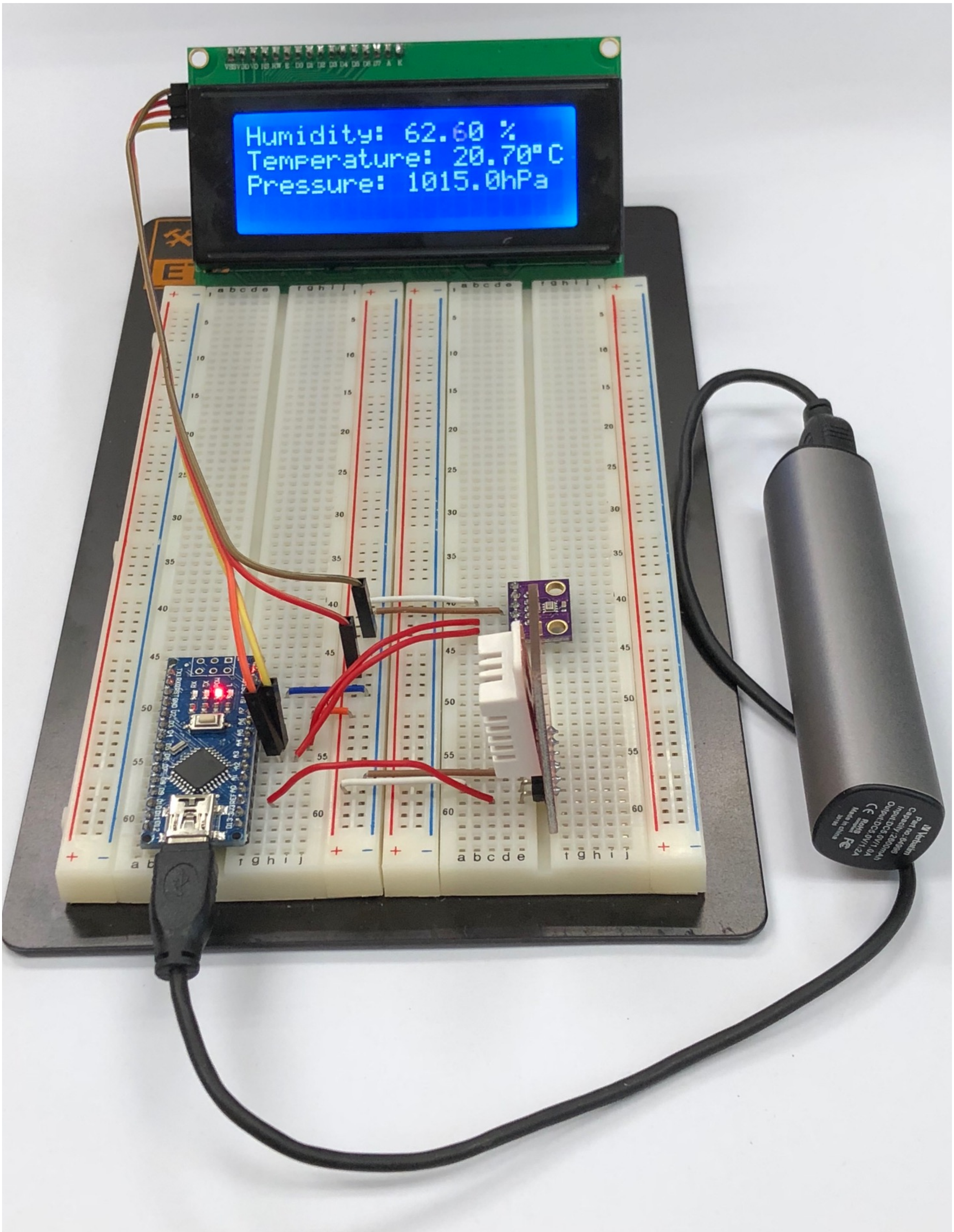
```
void readBMP280()
{
  float pressure;
  float pressure1;

    pressure = bmp280.getPressure()/100;
    pressure1 = bmp280.getPressure();
     lcd.setCursor(0,2);
     lcd.print("Pressure: ");
     lcd.setCursor(10,2);
     lcd.print(pressure);
     lcd.setCursor(16,2);
     lcd.print("hPa");
     lcd.setCursor(0,3);
     lcd.print("Altitude: ");
     lcd.setCursor(9,3);
     lcd.print(bmp280.calcAltitude(pressure1));
     lcd.setCursor(16,3);
     lcd.print("m");

}
```

If you want to output the degree symbol °C, try to change the command lcd.print(“C”); to lcd.print(“\337C”);



```
Humidity: 60.00 %
Temperature: 22.00°C
Pressure: 1014.0hPa
```

Congratulation! You have now completed your My_new_AWS project!



```
Humidity: 62.60 %
Temperature: 20.70°C
Pressure: 1015.0hPa
```