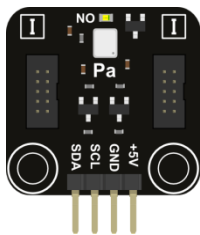


## Pirate Ship – Pressure&Altitude Monitoring

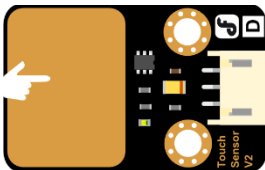
Nowadays, there are more and more navigation devices with GPS available in the market. Using global positioning system can determine equipment warp/weft location, but not the altitude. BMP180 Module can do this. This BMP180 module is a neo high precision digital sensor which is compatible with BMP085 module. It's very suitable for robot to detect the pressure and altitude.

### Hardware components

- BMP180 Sensor × 1



- Digital Touch Sensor x 1



- I2C LCD1602 LCD Module x 1



- M3\*6MM Nylon columns and tie-wraps



## Assembly Directions

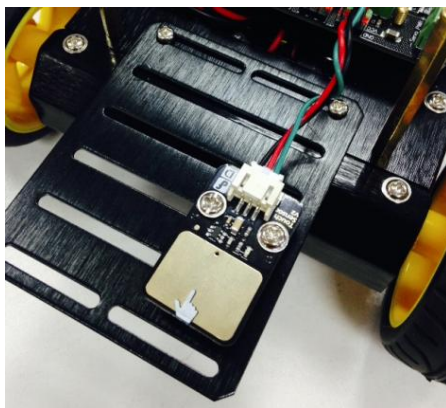
Now as we have all the components prepared, let's start assembling. It's quite easy. Just follow me and have fun.

### STEP 1: Add the touch sensor to the car

There are two holes for the Nylon columns in the touch sensor. Fix the Nylon columns on the sensor. And I know some of you just cannot help twisting the column like an OCD. I totally get that. But I would still like to remind you not to **over-twist those columns**.

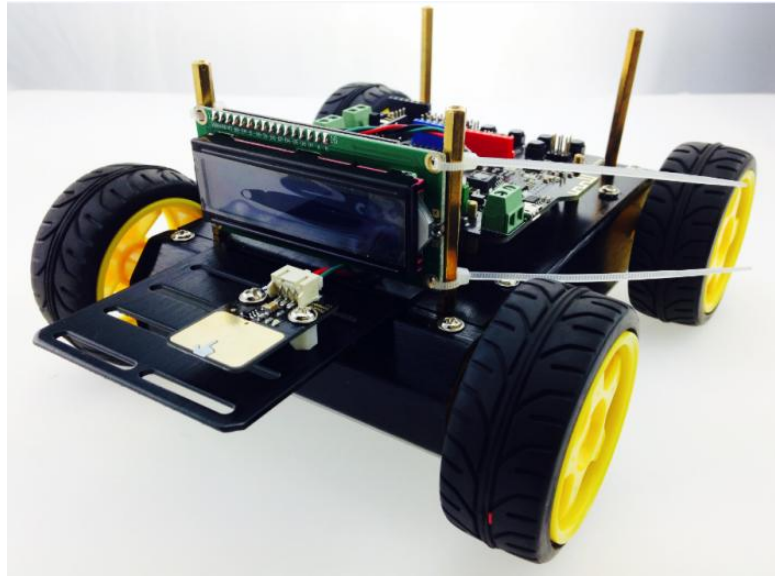


Once completed, attach the touch sensor to the sensor plate located at the fore end of the bodywork just as it is shown in the picture below.



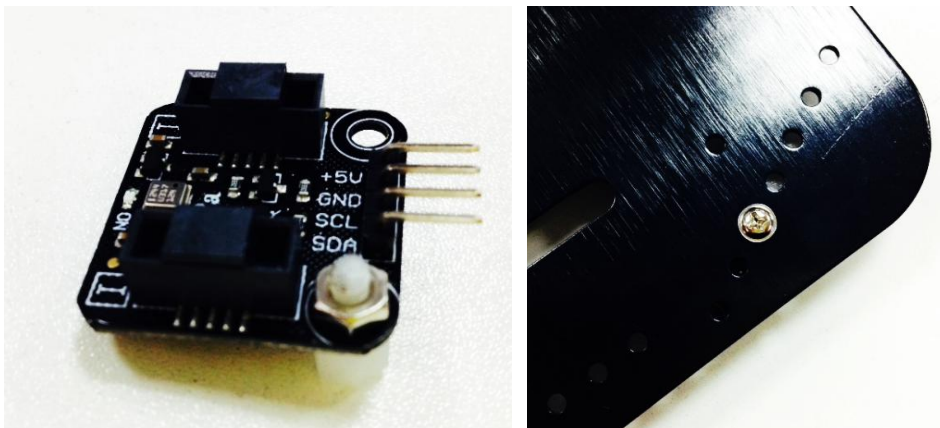
### STEP2: Add the LCD Screen

Slide the shores into the four holes in the LCD screen and fix them on the car with four tie-wraps. Then cut the remaining part of the tie-wraps.

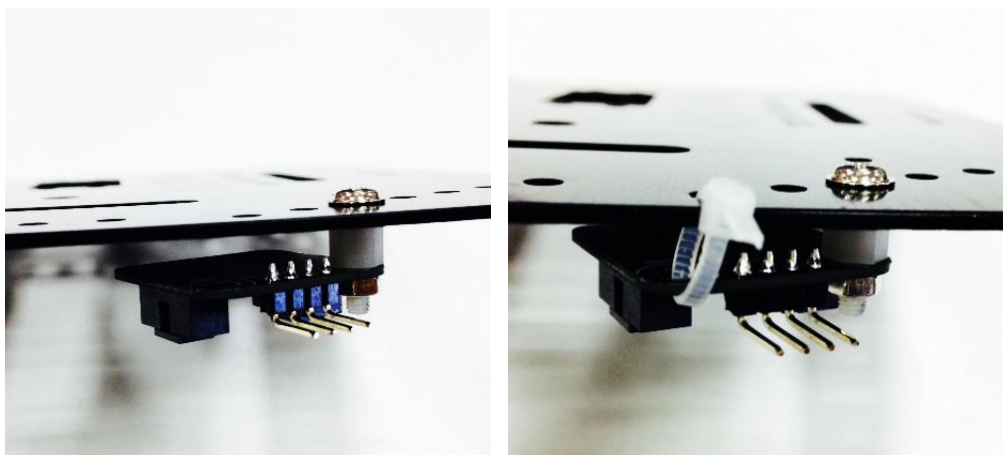


### STEP3: Assemble the BMP180 Sensor

Fix the Nylon columns on the BMP180 Sensor just as it's shown in the picture below.

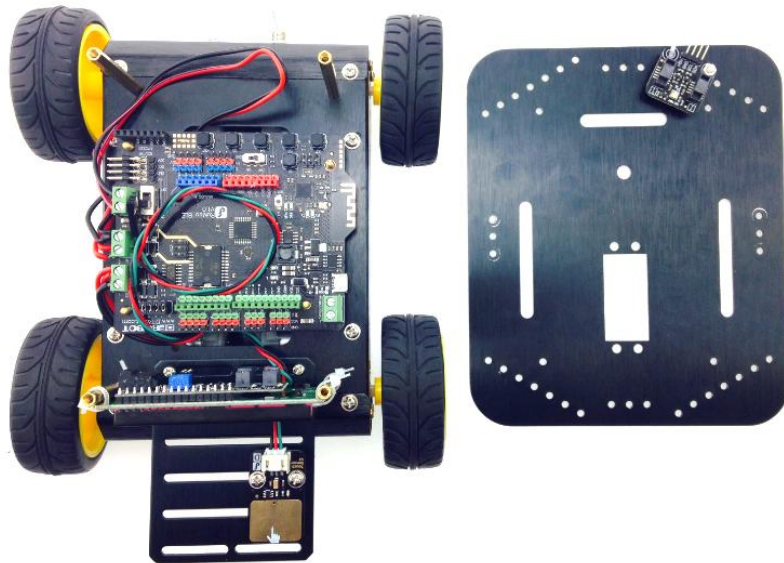


Use nut to fix the board on the back of the Pirate's roof. Because of the fixing holes, you only need to twist the nut on the one side. The holes on the other side is fixed by Nylon cable.



#### STEP 4: What It Looks Like When Finished Assembling

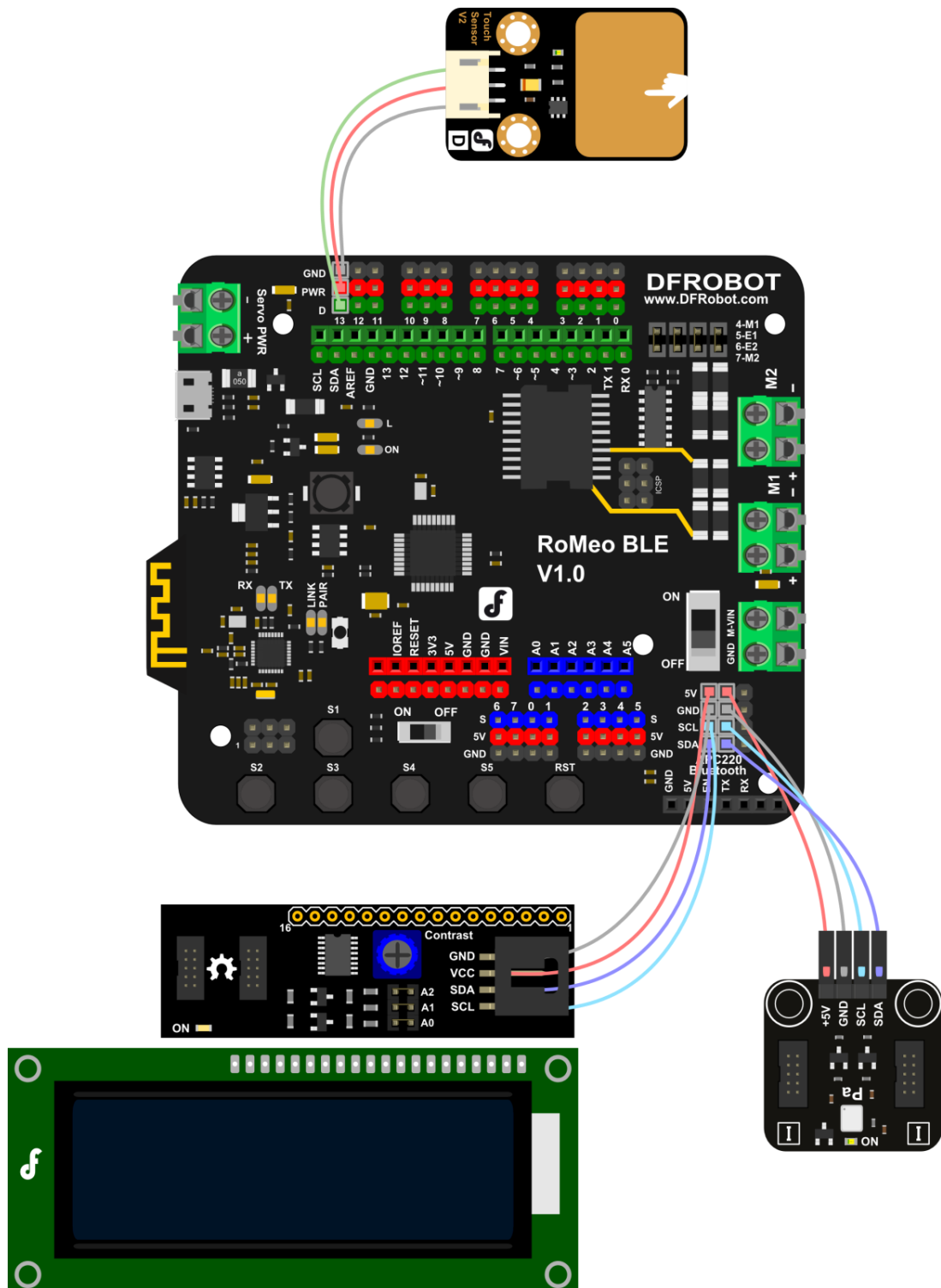
But now it's still too early for fixing the upper plate onto the car as we need to work on the electric circuit connection.



#### Connect the hardware

Now it's time for electric circuit connection. All you have to do is to connect the 3 pins on the sensor to the Romeo BLE.

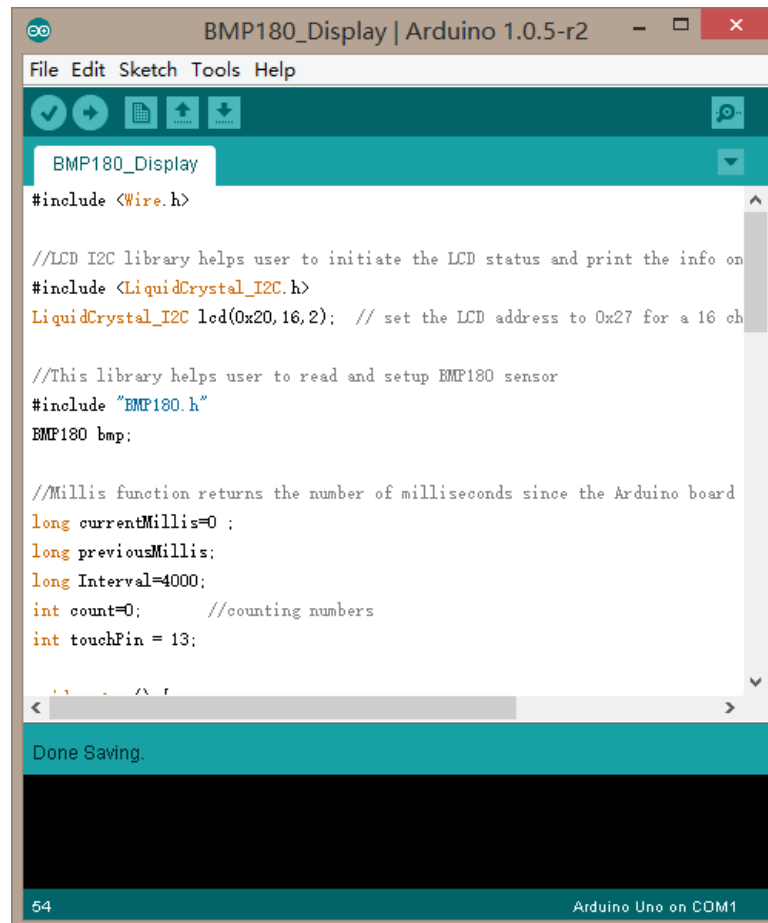
Keeping cables in order is quite important. The green cable represents the signal one; the red one stands for VCC and the black one represents GND. The LCD monitor shall be connected to VCC, GND, SCL and SDA in that particular order.





## Coding

Download the code first. The following picture only shows a part of the coding. You can find codes named *BMP180\_Display.ino* and then download. And don't forget the library for *LiquidCrystal\_I2C* and *BMP180*.

A screenshot of the Arduino IDE interface. The title bar reads "BMP180\_Display | Arduino 1.0.5-r2". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening, saving, and running. The main text area shows the following code:

```
#include <Wire.h>

//LCD I2C library helps user to initiate the LCD status and print the info on
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x20,16,2); // set the LCD address to 0x27 for a 16 ch

//This library helps user to read and setup BMP180 sensor
#include "BMP180.h"
BMP180 bmp;

//Millis function returns the number of milliseconds since the Arduino board
long currentMillis=0 ;
long previousMillis;
long Interval=4000;
int count=0; //counting numbers
int touchPin = 13;
```

The status bar at the bottom shows "54" on the left and "Arduino Uno on COM1" on the right.

If everything goes smoothly and you've successfully downloaded the coding, real-time pressure and altitude will be shown on the LCD screen. Hereby the touch sensor functions in two ways.

First, it functions switch of the LCD backlight lamp. If you don't touch the sensor for a long period, it will turn off itself automatically, which is energy-conservative.

Secondly, when you have several environment sensors, touch switch can help you quickly cut among different functions where you may see different environment monitoring data.

## Code Synopsis

We need to call library at the beginning of the code. Library is quite important and software engineers have already written some functions to process complicated data in the library. It's unnecessary for you to understand such abstruse theory as long as you know how to use it. Thus, it won't be easy to understand code without library.

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x20,16,2);

#include "BMP180.h"
BMP180 bmp;
```

Here you need to know about the variable *bmp*, which is used to declare the BMP180 sensor.

The followings are some declarations for the time variables. TouchPin represents touch sensor while 13 stands for digital pin.

```
long currentMillis=0;
long previousMillis;
long Interval=4000;
int count=0;          //counting numbers
int touchPin = 13;
```

Bring in the function of `setup()`, which is a setting-up for the initiation.

```
pinMode(touchPin,INPUT);
```

Then keep the touch sensor with a type-in mode. For specific information, you may check the Arduino Reference in Arduino Website([www.arduino.cc](http://www.arduino.cc)), which has introductof the function of `pinMode()`.

Next, you need to initialize the LCD screen and turn on the LCD light, which show that LCD screen is ready.

```
lcd.init();
lcd.backlight();
delay(100);
lcd.setBacklight(0);
```

Here, we need to set the initialization for BMP180 Sensor.

```
Wire.begin();  
bmp.init(2);
```

Now it's turn for the function of loop(). First we need to read the value from the touch sensor and then store those data with one variable touchState.

```
int touchState = digitalRead(touchPin);
```

Then check if the controller will receive a signal of HIGH once you touch the touch sensor with you fingers, 1 shall be added to the *count*.

```
if (touchState == HIGH) {  
    count++;  
    previousMillis= millis();  
}
```

Hereby *count* means how many times you have touched the screen. But if you only touch the sensor one time, then the amount of time for each touch will be included in the function of millis().

We change the length of touch time with a sub sentence initiating with if. Interval here means the period for touch we set up. Thus, we know what action shall be taken within four-second of touch and more than four seconds' touch respectively.

```
if(currentMillis - previousMillis < Interval) {  
    //do something in 4 second  
} else{  
    //do something more than 4 seconds  
}
```

What action shall be taken when we touch the sensor for more than four seconds.

```
lcd.setBacklight(0);
```

The function of setBacklight() is used to turn off the LCD backlight lamp.

If we touch the sensor for more than four seconds, we know that the LCD backlight lamp can be turned off.

what action shall be taken within four-second of touch.

```
if (count==1){  
    // One touch, the LCD screen won't show any difference
```



```

}
else if (count==2){
    // Touch twice, value will be shown on the LCD screen
}

```

Press the touch sensor one more time within four seconds; the screen would still be off. Only if you touches it twice at the same time, will the LCD backlight be on and figures of dust density be shown.

Please remember to keep the count as zero after you touch the sensor for the last time.

```
count=0;
```

Thus the complete code shall be:

```

if (count==1){
    lcd.setBacklight(0);
}
else if (count==2){
    lcd.backlight();
    BMP180Show();
    count=0;
}

```

Then we need to keep a track of the current time as we can compare it with *previousMillis*. This point is very important.

```
currentMillis = millis();
```

The function of `BMP180Show()` is used to show data of dust density. And The function of `calculate()` will be used to calculate the data from the BMP180 sensor.

```
bmp.calculate();
```

Here is how we would used the function related to LCD screen.

```

lcd.setCursor(0,0);
lcd.setCursor(0,1);

```

The function of `setCursor(column,row)` is used to demonstrate which column and row the cursor is shown, starting from zero within the brackets.

```

lcd.print(bmp.ReadPressure(),1);
lcd.print(bmp.ReadAltitude(),0);

```

`print()` means this figure can be shown on the screen directly.

## A Combination of Multiple Sensors

How can you combine multiple environmental sensors once you have bought some kind of sensors?

Don't worry. We will offer you guys a coding template for the testing of multiple sensors. You can make adjustments of the combination by referring to the mentioned template. In fact, the theory is the same as single sensor except that there are for steps for the changes of LCD screen.

The coding in red below needs to be modified. We mentioned before that count refers to how many times fingers touch the sensor. Thus, count=2 means that we have pressed twice and it shows the figures for the first sensor. Keep going! Please bear in your mind that you shall keep the count zero again.

Sample Code:

```
if(currentMillis - previousMillis < Interval) {  
  if (count==1){  
    lcd.setBacklight(0);  
  }  
  else if (count==2){      //No.1 Sensor  
    Sensor1Show();  
    lcd.backlight();  
  }  
  else if(count==3){      //No.2 Sensor  
    Sensor2Show();  
    lcd.backlight();  
    count = 0;  
  }  
}
```

Of course, initiation set-up, declaration of variables at the beginning, for the sensor is important.

You can check the sample code named *WeatherStation.ino* for reference if you still have no idea how to modify your codes.