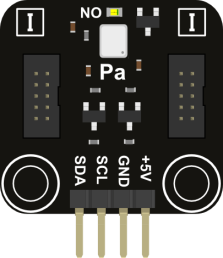
海盗船 ---气压和海拔检测

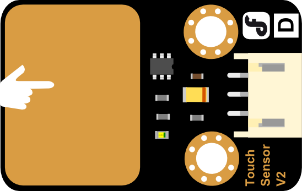
具有GPS功能的导航设备越来越多。借助全球卫星定位系统可以判定设备的经纬平面所处位置，但是无法得知目前的海拔高度。BMP180则可以做到，它是一款高精度的数字气压传感器。非常适合于应用各个领域，大气压强，海拔高度检测。

**硬件材料：**

* BMP180传感器 × 1



* 数字触摸开关 × 1



* I2C LCD1602液晶模块 × 1



* M3\*6尼龙柱、尼龙扎带 × 若干

**组装步骤**

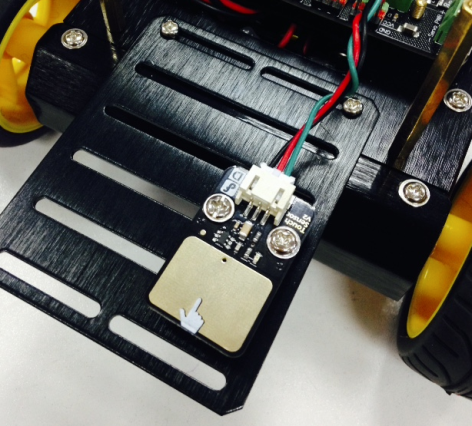
准备完以上这些材料，就开始组装了，组装不难，更着操作就行。

**STEP 1: 安装触摸传感器**

尼龙柱是用来固定传感器的，先将尼龙柱固定到触摸传感器的两个固定孔上。注意不要拧的过紧。

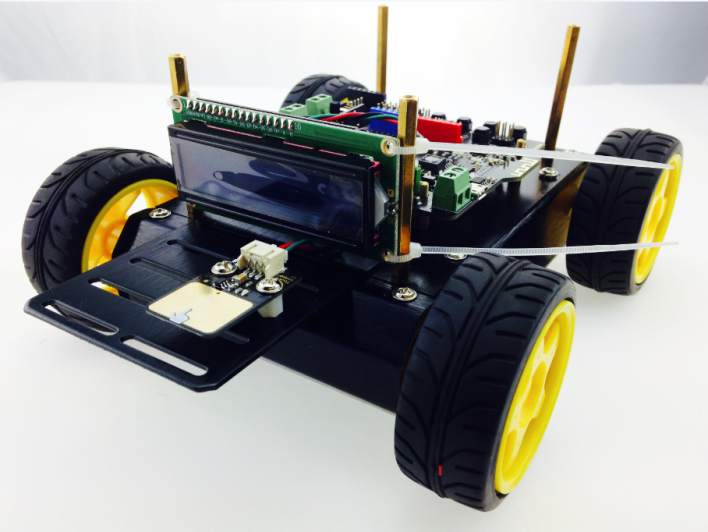


完成后，再将传感器固定到车身前端的传感器板上。下图为正反面。

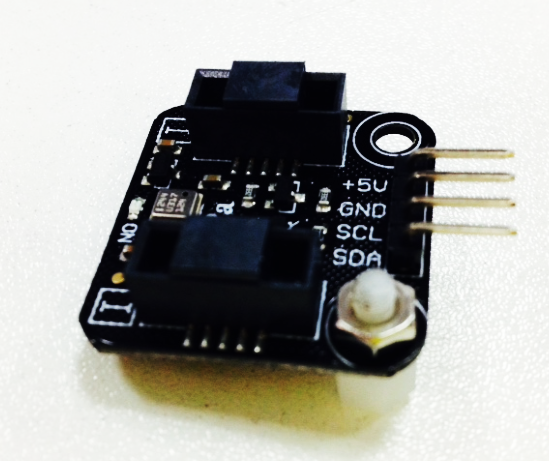
**STEP 2: 安装LCD屏**

取出4根尼龙扎带，依次将LCD屏上的4个固定孔和小车的支撑柱固定在一起。将多余的线减去即可。

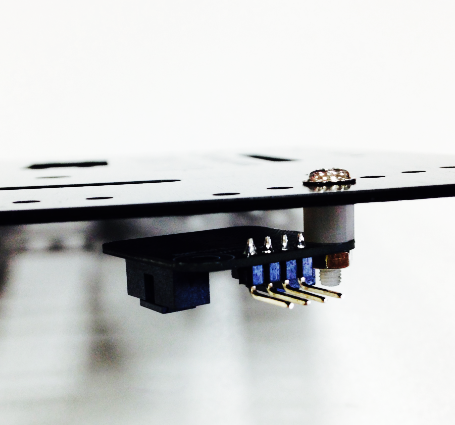
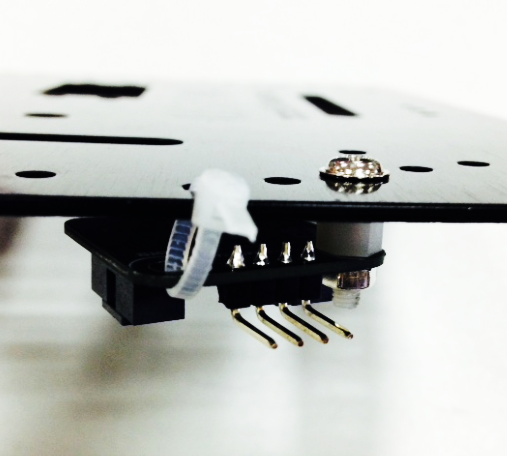


**STEP 3: 安装BMP180传感器**

还是先将尼龙柱固定到BMP180传感器上，如下图所示。只需固定一边。

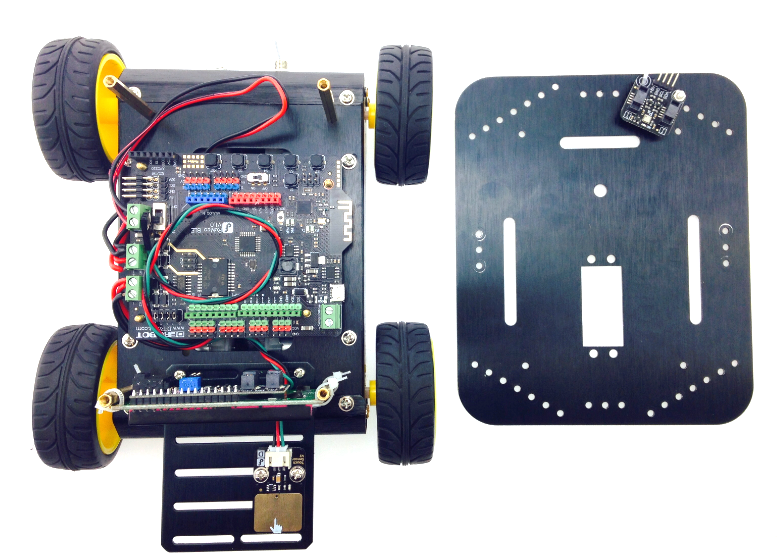
 

再用螺母将其固定到小车的顶板的背部，因为固定孔的原因，也只需要拧上一个螺母即可，另一边用尼龙扎带固定如下图所示。

**STEP 4: 完整效果图、**

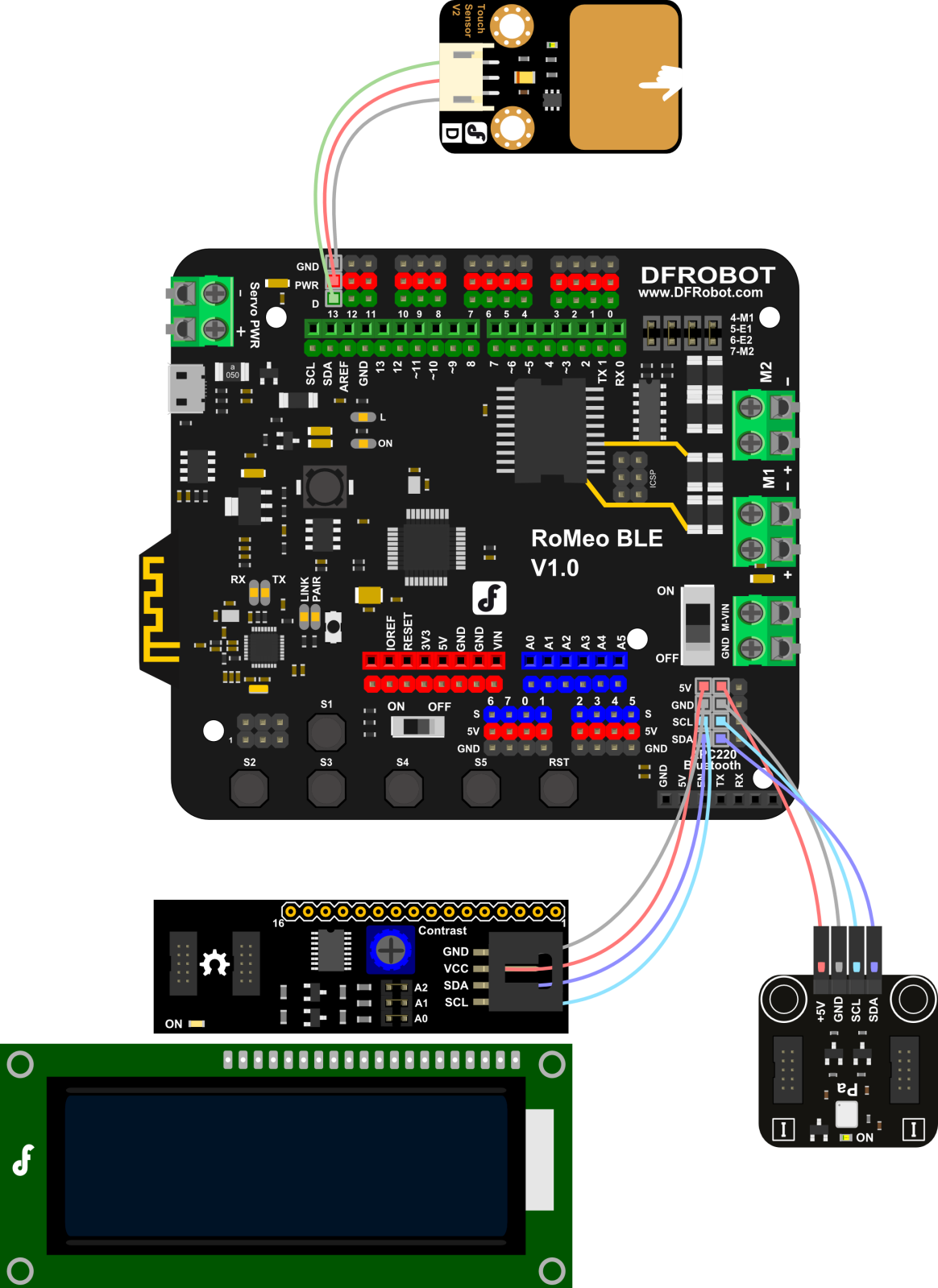
建议先不要将小车上层的顶板固定上去，因为我们下一步就需要硬件电路连接了。



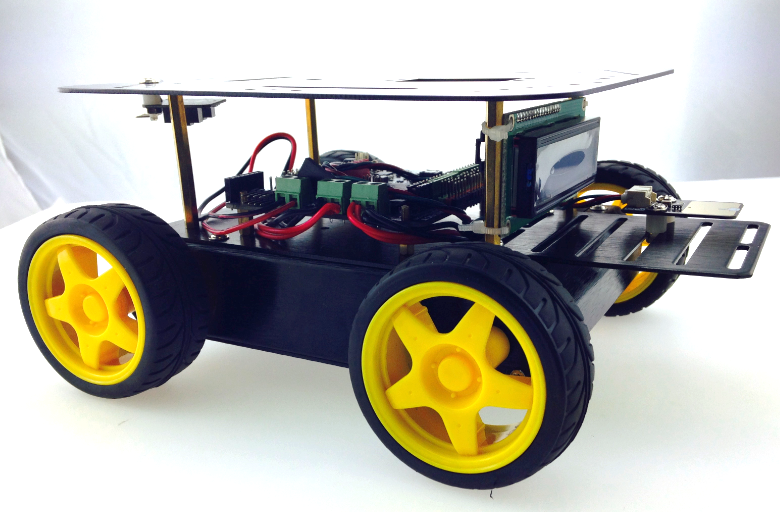
**硬件连接**

安装完成后，我们就要开始电路连接了，参照我们下图的硬件连接图。只需将传感器的3Pin引脚直接连接到Romeo BLE上即可。

注意线序：传感器的绿色为信号线，红色为VCC，黑色为GND。LCD显示屏依次接VCC，GND，SCL，SDA。



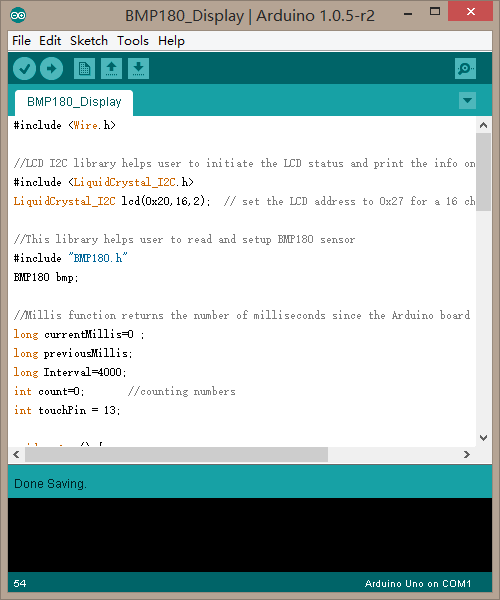
连接完成后，可将上层顶板用螺丝固定到小车上了。

****

连接完成后，我们下一步就可以烧写代码了。

**输入代码**

插上USB线，下载代码。由于代码较长，所以这里就不全部显示出来，你可在软件包中找到BMP180\_Display.ino的代码，下载进去后即可。不要忘记加载LiquidCrystal\_I2C和BMP180的库。

****

如果一切都顺利的话，下载成功后，用手轻触两下触摸传感器，你就可以在LCD屏上看到实时显示的气压和海拔值了。触摸传感器在这里起到两个作用。

第一个作用： LCD背光灯的开关作用，轻按触摸传感器，背光灯会打开，长时间不按，又会自己关闭，节约电源。

第二个作用：屏幕切换作用，当你用到多个环境传感器的时候，触摸传感器就能起到很好的屏幕切换的作用。通过切换可查看多个环境监测数据。

**代码回顾**

代码开始还是一些库函数的声明，库很重要，工程师已经将一些很复杂的数据处理部分写在库中了，你没有必要理解这么深奥的理论知识，只需要会使用就可以了。所以，如果没有库，也就没有看上去那么容易理解的代码了。

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x20,16,2);

#include "BMP180.h"

BMP180 bmp;

下面这串是一些时间变量的声明，下面会有说明。touchPin对应的就是触摸传感器，13号数字引脚。

long currentMillis=0;

long previousMillis;

long Interval=4000;

int count=0; //counting numbers

int touchPin = 13;

进入setup()函数，是一些初始化设置

pinMode(touchPin,INPUT);

将触摸传感器设置为输入模式，具体可参看Arduino语法手册pinMode()函数介绍。

接下来，初始化LCD屏，并且打开一次LCD灯的背光灯，表示LCD上电开始运作。

lcd.init();

lcd.backlight();

delay(100);

lcd.setBacklight(0);

这里需要对BMP180做一些初始化设置：

Wire.begin();

bmp.init(2);

接着进入loop()函数中，首先需要读取触摸传感器的值，用touchState这个变量来存储读到的值。

int touchState = digitalRead(touchPin);

紧接着就是判断，是不是有手触碰传感器，如果有手触碰传感器，也就是说控制器会读到HIGH这个信号。那么count就会加1（可查看自加++用法）。

if (touchState == HIGH){

count++;

previousMillis= millis();

}

这里count是用来记录按下的次数的。并且，每按下一次，就用millis()函数记录下时间。通过if语句，不断和前一次按下的时间做比较。Interval，在这里就是设置的时间点，4000ms(也就是4s)。如果4秒内有手接触触摸传感器，那么执行些什么动作。超过4s又执行些什么动作。

if(currentMillis - previousMillis < Interval) {

//4秒内，执行些什么

else{

//超过4秒，执行些什么

}

我们先看超过4秒的，比较容易理解：

lcd.setBacklight(0);

setBacklight()这个函数是用来关闭LCD背光灯的。

超过4秒，我们知道可以关闭LCD背光灯。

那么不超过4秒呢？执行些什么动作？

if (count==1){

// 按下一次 LCD屏幕不做任何反应

}

else if (count==2){

// 连按二次 LCD屏幕才显示BMP180传感器的值

}

如果4秒内，传感器被按下一次，那么屏幕仍然保持关闭状态。只有当被连按两次的时候，LCD背光灯才被开启，显示BMP180传感器的值。

这里有一点需要注意，再最后一次按下后，count要被清零，那么下次监测的时候才能又从0开始计数了。

count=0;

所以对应的完整代码就是：

if (count==1){

lcd.setBacklight(0);

}

else if (count==2){

lcd.backlight();

BMP180Show();

count=0;

}

这里数据读取完之后，需要记录下当前的时间(currentMillis)，可与前面的时间(previousMillis)做个比较，这一点也很重要。

currentMillis = millis();

这里还涉及一个函数BMP180Show()，可以从函数的名字上猜出，这个函数是用来显示BMP180传感器的数值的。依次来看下：

bmp.calculate();

calculate()函数是用来读取并处理BMP180传感器数值。

下面这部分全部是LCD屏的一些相关用法：

lcd.setCursor(0,0);

lcd.setCursor(0,1);

setCursor(column,row)函数是用来设置光标显示的行与列的位置，从0开始计算。

比如说(0,1)也就是说，从第一列，第二行开始显示数据。

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

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

print()函数可直接在屏幕上显示对应的气压和海拔的数值。

**多传感器结合**

可能你在选购的时候，用到了多个传感器环境监测的传感器，那该如何将它们全部结合起来呢？

不用担心，我们提供给你一个多传感器检测的代码模板，你可参照模板来进行修改。其实原理和单个传感器是一样的，只是中间LCD屏幕切换部分多增加了几个环节。

如下所示的代码，红色标注的部分就是需要修改的部分。在前面代码回顾中讲到，count是用来手指触摸传感器的次数的，所以count=2表示按下两次，让它显示第一个传感器的数值。那么如果增加一个传感器，我们就记count=3，被按下第三次的时候，让它显示第二个传感器的数值。依次类推……不要忘记再最后一个传感器显示结束之后，需要将count清零。

样例代码:

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;

}

}

当然，传感器的一些初始化设置不能少，也就是代码开始的变量声明，以及独立的Show()函数。否则控制器无法读取到传感器的数值。

如果还不是很清楚如何修改你的代码，可查看我们提供的**WeatherStation.ino**的样例代码。