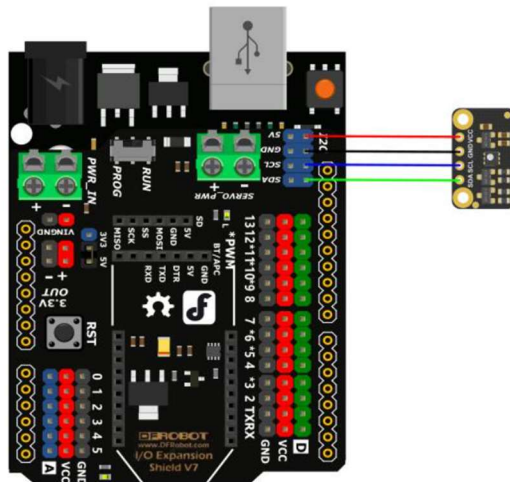


1. 测试 Sensor_SGP40.ino

将 SGP40 传感器参考如下连线图连接至 Arduino，并且上传当前文件夹中的 Sensor_SGP40.ino 至 Arduino



串口监视器中正常出现 VOC 指数，说明传感器正常。传感器可以被连接至 Beacon 测试。

```
Arduino Uno
Sensor_SGP40.ino  sensirion_arch_config.h  sensirion_voc_algorithm.c  sensirion_voc_algorithm.h

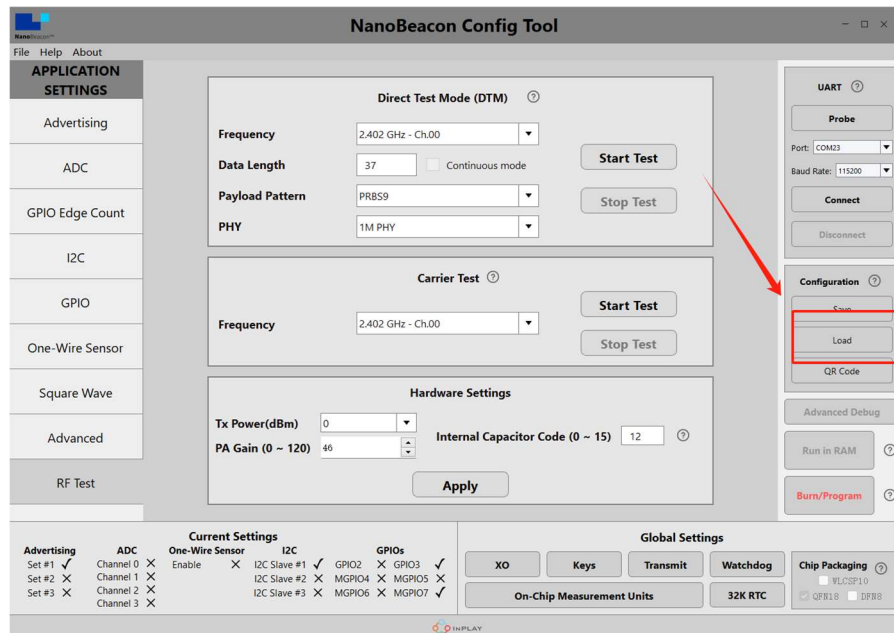
1  #include <Wire.h>
2
3  extern "C" {
4  #include "sensirion_arch_config.h"
5  #include "sensirion_voc_algorithm.h"
6  };
7
8  #define MODULE_I2C_ADDRESS ((uint8_t)0x59) // 传感器的 I2C 地址，此处的 59
9
10 #define TEST_OK 0xD400
11
12 #define CPD_HEATER_OFF_H 0x36
13 #define CPD_HEATER_OFF_L 0x15
14 #define CPD_HEATER_OFF_SIZE 2
15
16 #define CPD_MEASURE_TEST_H 0x28
17 #define CPD_MEASURE_TEST_L 0x0E
18 #define CPD_MEASURE_TEST_SIZE 2
19
20 #define CPD_SOFT_RESET_H 0x00
21 #define CPD_SOFT_RESET_L 0x06
22 #define CPD_SOFT_RESET_SIZE 2
23
24 #define CPD_MEASURE_RAW_H 0x26
25 #define CPD_MEASURE_RAW_L 0x0F

Output Serial Monitor x
Message (Enter to send message to 'Arduino Uno' on 'COM4')
vocIndex = 119
74 F0 B8
vocIndex = 119
75 25 8E
vocIndex = 118
75 50 83
vocIndex = 115
75 75 C0
vocIndex = 113
75 A9 B9
vocIndex = 110
75 DC B4
vocIndex = 107
```

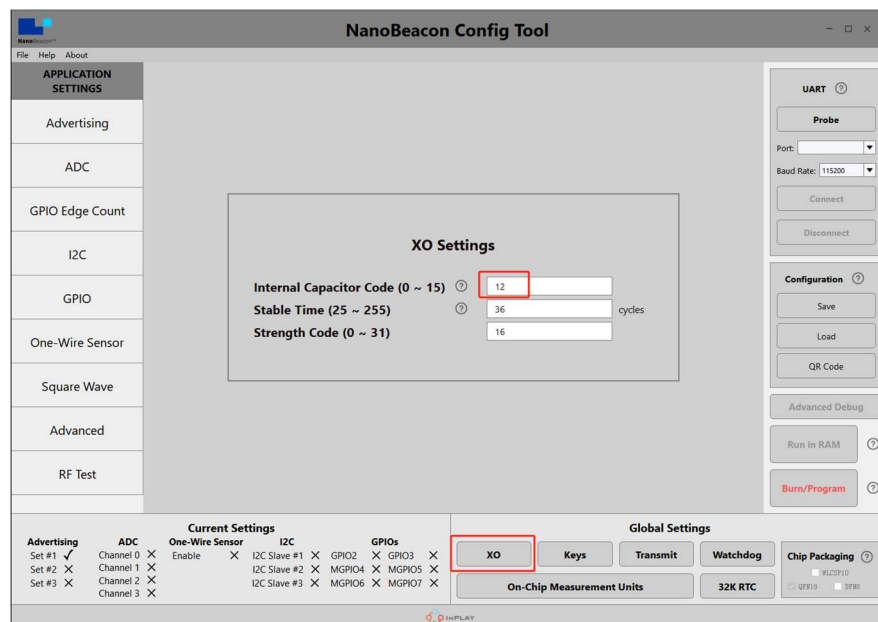
2. 烧录 Beacon 并且连接传感器

请您使用 USB-TTL 转换器将.cfg 文件烧录进 Beacon。

NanoBeacon Config Tool 中可以 Load 本文件夹中的 SGP40.cfg 文件。



检查 XO 电容配置是否为 12

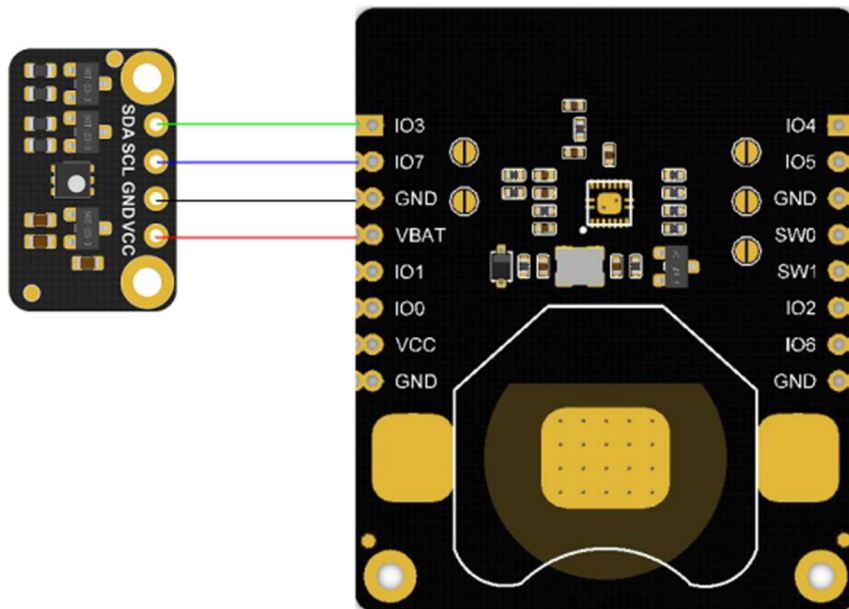


烧录流程请参考 Beacon 的 wiki:

https://wiki.dfrobot.com.cn/_SKU_TEL0168_Fermion_BLE_%E4%BC%A0%E6%84%9F%E5%99%A8%E4%BF%A1%E6%A0%87#target_4

在烧录完成后，参考下图连接 Beacon 和传感器。

注：我们的.cfg 示例文件默认 SCL->GPIO7, SDA->GPIO3。



3. 上传 ESP32 代码并获取读数

将同目录下的 Beacon_SGP40.ino 上传至 ESP32 主板。

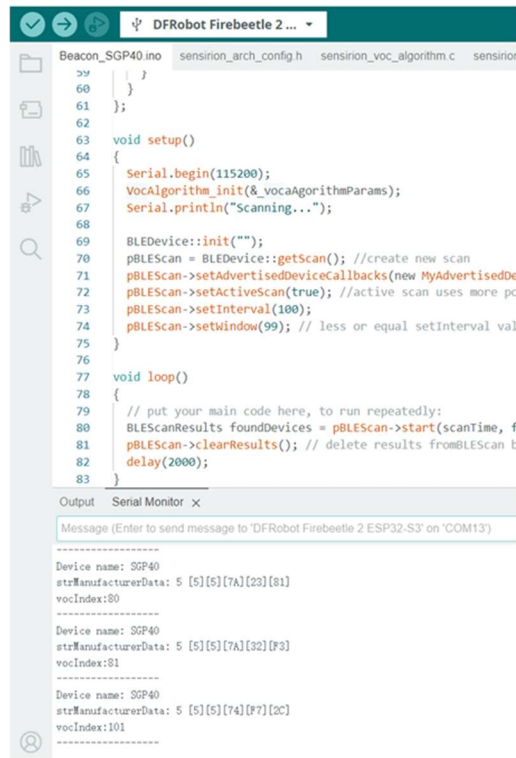
并且将 Beacon 和传感器供电，供电方式可选 CR2032 纽扣电池，或者 VCC 和 GND 输入 3.3V。

您将会看到串口监视器中打印相关数据。

注：由于 SGP40 的 VOC 算法需要采集大量基础数据进行计算，并且

Beacon_SGP40.ino 设定 5 秒获取一次。所以在给 Beacon 和 SGP40 供电后。您将

需要等待 5-10 分钟后才能看到 VOC 数据。



The screenshot displays the Arduino IDE interface. The top pane shows the code for Beacon_SGP40.ino, which includes initialization of the SGP40 sensor and BLE module, and a loop that scans for devices every 2000ms. The bottom pane shows the Serial Monitor output, which displays the device name 'SGP40', the manufacturer data as a hex string, and the VOC index.

```
Beacon_SGP40.ino  sensirion_arch_config.h  sensirion_voc_algorithm.c  sensirion_voc_algorithm.h

59   }
60   }
61   };
62
63   void setup()
64   {
65     Serial.begin(115200);
66     VocAlgorithm_init(&vocaAlgorithmParams);
67     Serial.println("Scanning...");
68
69     BLEDevice::init("");
70     pBLEScan = BLEDevice::getScan(); //create new scan
71     pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallback());
72     pBLEScan->setActiveScan(true); //active scan uses more power
73     pBLEScan->setInterval(100);
74     pBLEScan->setWindow(99); // less or equal setInterval value
75   }
76
77   void loop()
78   {
79     // put your main code here, to run repeatedly:
80     BLEScanResults foundDevices = pBLEScan->start(scanTime, false);
81     pBLEScan->clearResults(); // delete results from BLEScan object
82     delay(2000);
83   }
```

Output Serial Monitor x

Message (Enter to send message to 'DFRobot Firebee2 2 ESP32-S3' on 'COM13')

Device name: SGP40
strManufacturerData: 5 [5][5][7A][23][81]
vocIndex:80

Device name: SGP40
strManufacturerData: 5 [5][5][7A][32][F3]
vocIndex:81

Device name: SGP40
strManufacturerData: 5 [5][5][74][F7][2C]
vocIndex:101
