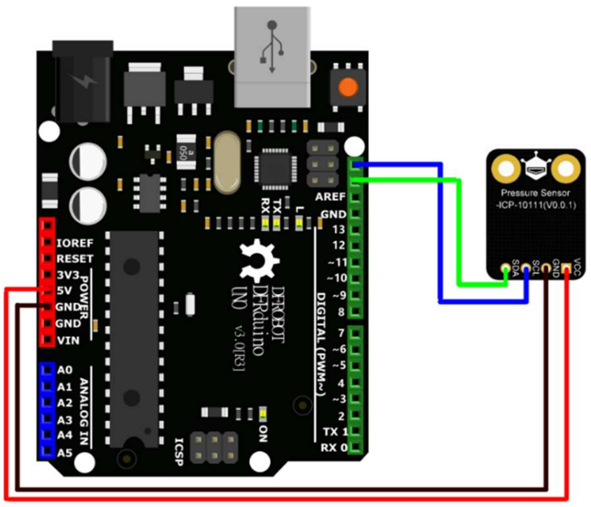
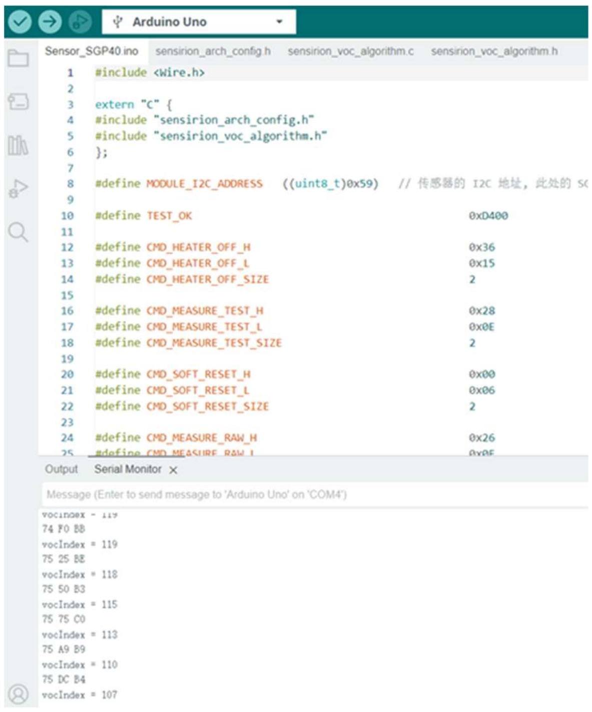
1. Test Sensor_ICP10111.ino

Connect the ICP10111 sensor to the Arduino by referring to the following wiring diagram, and upload the Sensor_ICP10111.ino from the current folder to the Arduino.



When the reading appears normally in the serial monitor, the sensor is normal.

The sensor can be connected to a Beacon for testing.



Arduino Uno
Sensor_SGP40.ino
sensirion_arch_config.h
sensirion_voc_algorithm.c
sensirion_voc_algorithm.h
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM4')

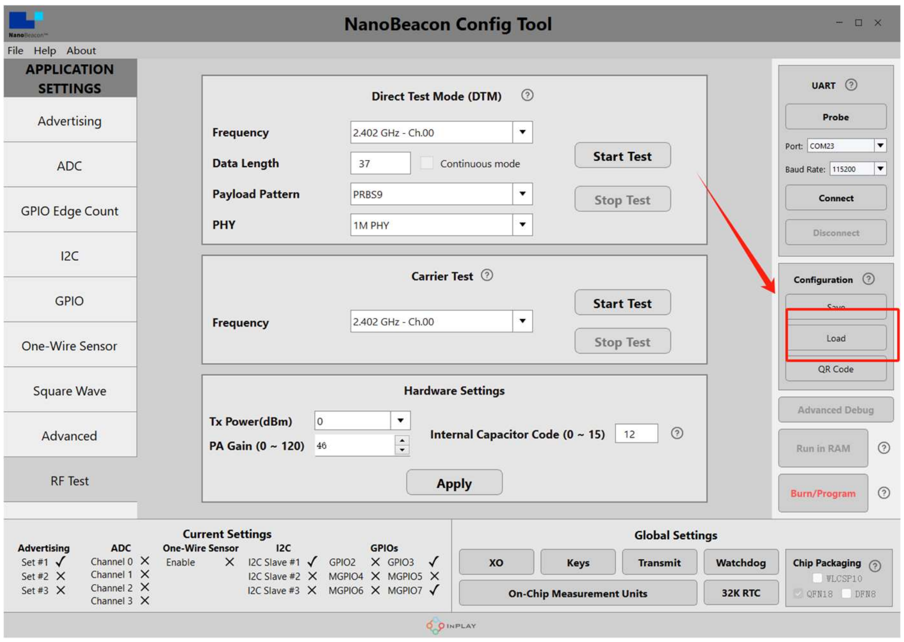
```
#include <Wire.h>

extern "C" {
#include "sensirion_arch_config.h"
#include "sensirion_voc_algorithm.h"
};

#define MODULE_I2C_ADDRESS   ((uint8_t)0x59)   // 传感器的 I2C 地址，此处的 SC

#define TEST_OK                                          0xD400

#define CMD_HEATER_OFF_H                                 0x36
#define CMD_HEATER_OFF_L                                 0x15
#define CMD_HEATER_OFF_SIZE                              2

#define CMD_MEASURE_TEST_H                               0x28
#define CMD_MEASURE_TEST_L                               0x0E
#define CMD_MEASURE_TEST_SIZE                            2

#define CMD_SOFT_RESET_H                                 0x00
#define CMD_SOFT_RESET_L                                 0x06
#define CMD_SOFT_RESET_SIZE                              2

#define CMD_MEASURE_RAW_H                                0x26
#define CMD_MEASURE_RAW_L                                0x0F
```

```
vocIndex = 119
74 F0 BB
vocIndex = 119
75 25 BE
vocIndex = 118
75 50 B3
vocIndex = 115
75 75 C0
vocIndex = 113
75 A9 B9
vocIndex = 110
75 DC B4
vocIndex = 107
```

2. Burning Beacon and Connecting the sensor

Please use a USB-TTL converter to burn the .cfg file into the Beacon.

NanoBeacon Config Tool can Load the ICP10111.cfg file in this folder.



Direct Test Mode (DTM)

Frequency: 2.402 GHz - Ch.00
Data Length: 37 Continuous mode
Payload Pattern: PRBS9
PHY: 1M PHY

Carrier Test

Frequency: 2.402 GHz - Ch.00

Hardware Settings

Tx Power(dBm): 0
PA Gain (0 ~ 120): 46
Internal Capacitor Code (0 ~ 15): 12

Port: COM23
Baud Rate: 115200

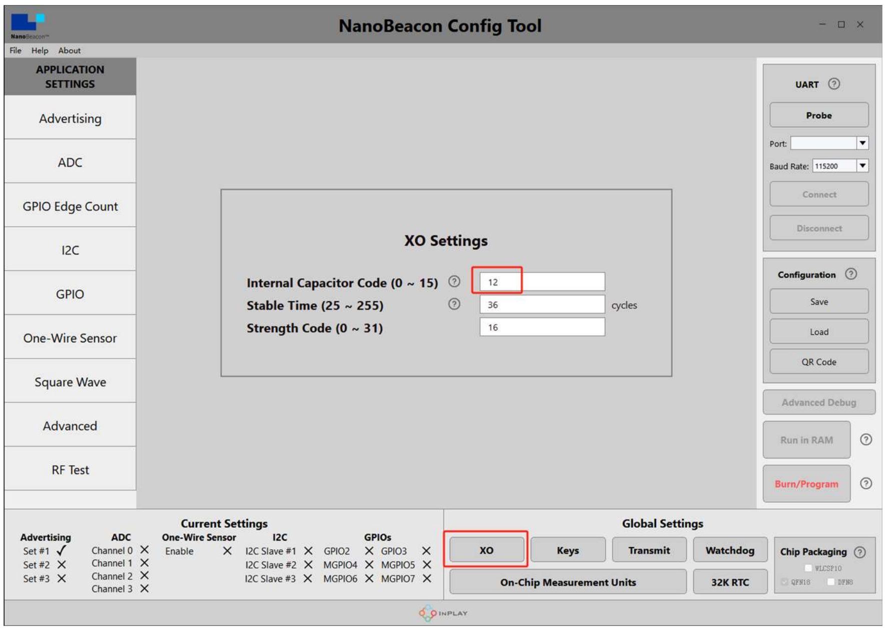
Current Settings

Advertising	ADC	One-Wire Sensor	I2C	GPIOs
Set #1 ✓	Channel 0 ✗	Enable ✗	I2C Slave #1 ✓	GPIO2 ✗ GPIO3 ✓
Set #2 ✗	Channel 1 ✗		I2C Slave #2 ✗	MGPIO4 ✗ MGPIO5 ✗
Set #3 ✗	Channel 2 ✗		I2C Slave #3 ✗	MGPIO6 ✗ MGPIO7 ✓
	Channel 3 ✗			

Global Settings

Chip Packaging: WLCSP10 [✓ QFN18] DFN8

Check that the XO capacitor configuration is 12



XO Settings

Internal Capacitor Code (0 ~ 15): 12
Stable Time (25 ~ 255): 36 cycles
Strength Code (0 ~ 31): 16

Port:
Baud Rate: 115200

Current Settings

Advertising	ADC	One-Wire Sensor	I2C	GPIOs
Set #1 ✓	Channel 0 ✗	Enable ✗	I2C Slave #1 ✗	GPIO2 ✗ GPIO3 ✗
Set #2 ✗	Channel 1 ✗		I2C Slave #2 ✗	MGPIO4 ✗ MGPIO5 ✗
Set #3 ✗	Channel 2 ✗		I2C Slave #3 ✗	MGPIO6 ✗ MGPIO7 ✗
	Channel 3 ✗			

Global Settings

Chip Packaging: WLCSP10 QFN18 DFN8

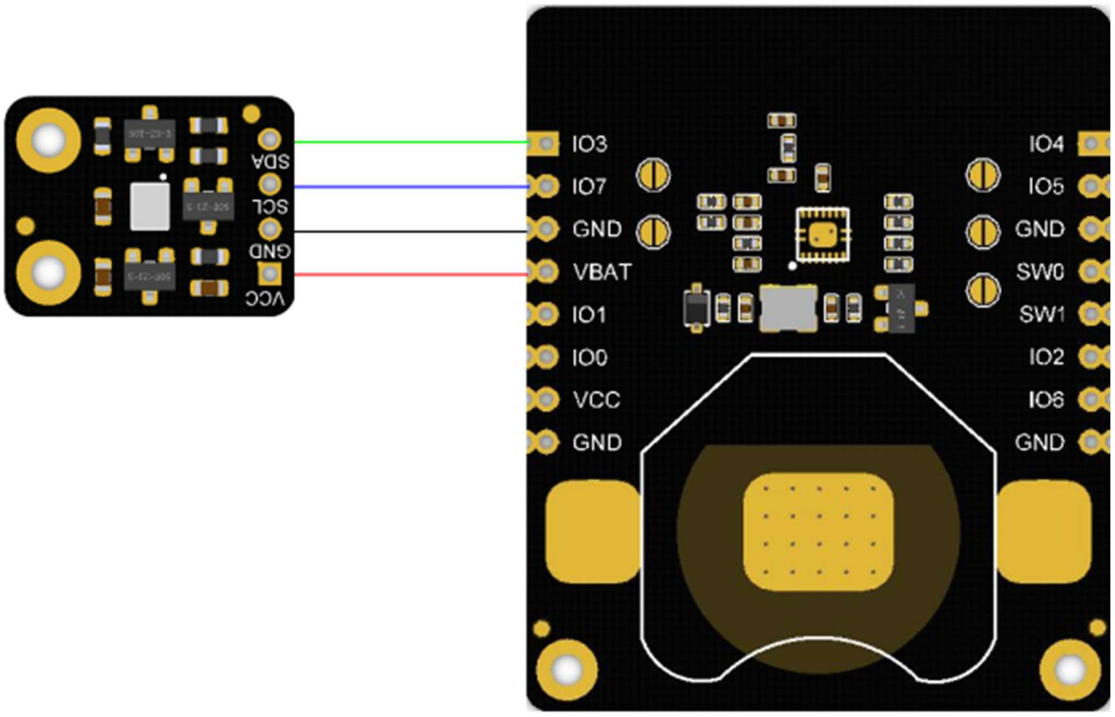
Please refer to Beacon's wiki for the burn-in process:

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

After the burn-in is complete, refer to the following diagram to connect the Beacon and the sensors.

Note: Our .cfg example file defaults SCL->GPIO7, SDA->GPIO3.

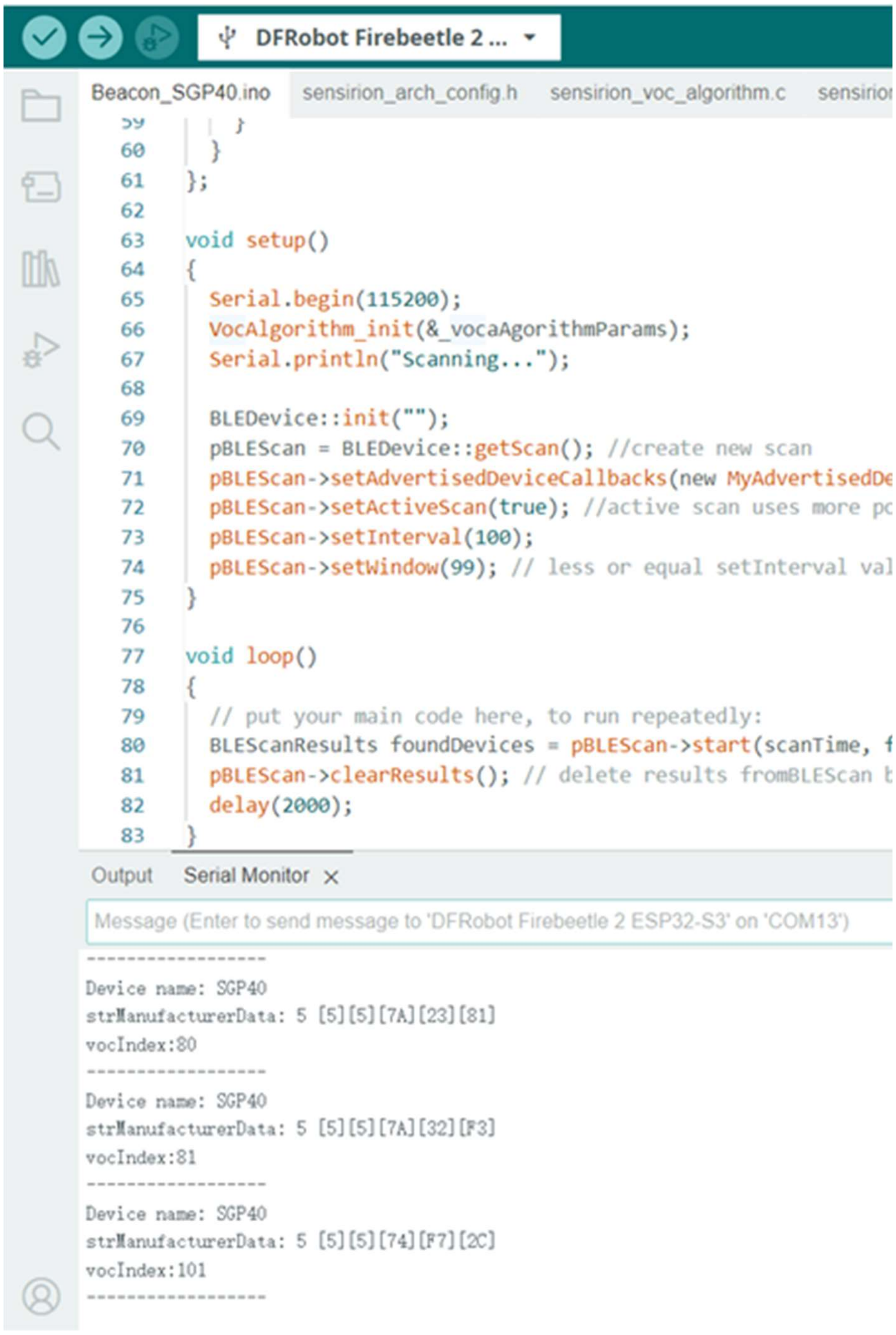


3. Upload ESP32 code and get readings

Upload the Beacon_SGP40.ino in the same directory to the ESP32.

And power up the Beacon and sensors, power supply can be selected from CR2032 coin cell battery, or VCC and GND input 3.3V.

You will see the relevant data printed in the serial monitor.



DFRobot Firebeetle 2 ...
Beacon_SGP40.ino
sensirion_arch_config.h
sensirion_voc_algorithm.c
sensirion
Output
Serial Monitor
Message (Enter to send message to 'DFRobot Firebeetle 2 ESP32-S3' on 'COM13')

```
      }
    }
};

void setup()
{
  Serial.begin(115200);
  VocAlgorithm_init(&_vocaAgorithmParams);
  Serial.println("Scanning...");

  BLEDevice::init("");
  pBLEScan = BLEDevice::getScan(); //create new scan
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDe
  pBLEScan->setActiveScan(true); //active scan uses more pc
  pBLEScan->setInterval(100);
  pBLEScan->setWindow(99); // less or equal setInterval val
}

void loop()
{
  // put your main code here, to run repeatedly:
  BLEScanResults foundDevices = pBLEScan->start(scanTime, f
  pBLEScan->clearResults(); // delete results fromBLEScan b
  delay(2000);
}
```

```
-----------------
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
-----------------
```