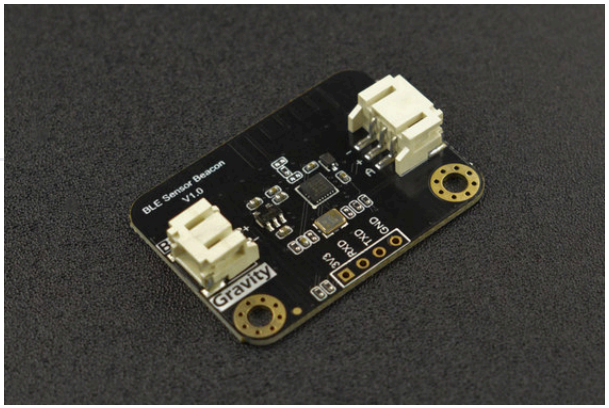
SKU:TEL0149 (https://www.dfrobot.com/product-2641.html)

(https://www.dfrobot.com/product-2641.html)



Introduction

This Gravity wireless BLE Sensor Beacon with built-in 11-bit ADC is capable of collecting data from digital and analog sensors and broadcasting via Bluetooth. And users can access the sensor data within broadcasting range on a Bluetooth-equipped device like a smartphone, ESP32, etc.
The BLE sensor beacon is integrated with low power BLE 5.3 technology, and its data format can be configured as iBeacon, Eddystone, custom format, etc. Besides, users can configure broadcast data format, content and time intervals on graphical interface as per their needs, which allows configuring a BLE beacon without any programming.
After configuration, it will run as a Bluetooth beacon when powered on, and collect and broadcast data automatically according to the settings. These sensor beacons can be used as IoT sensor nodes for data collection in many scenarios such as smart farms, offices, factories, and warehouses.

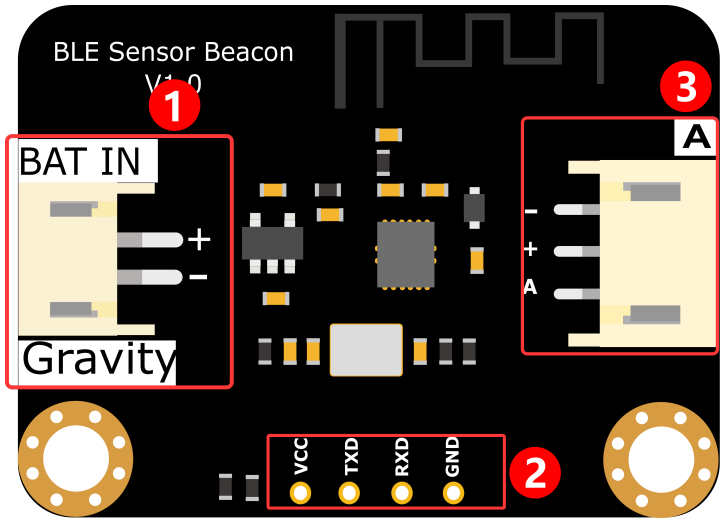
Note: Gravity: BLE beacon module needs to be configured with a 3.3V USB-TTL tool.

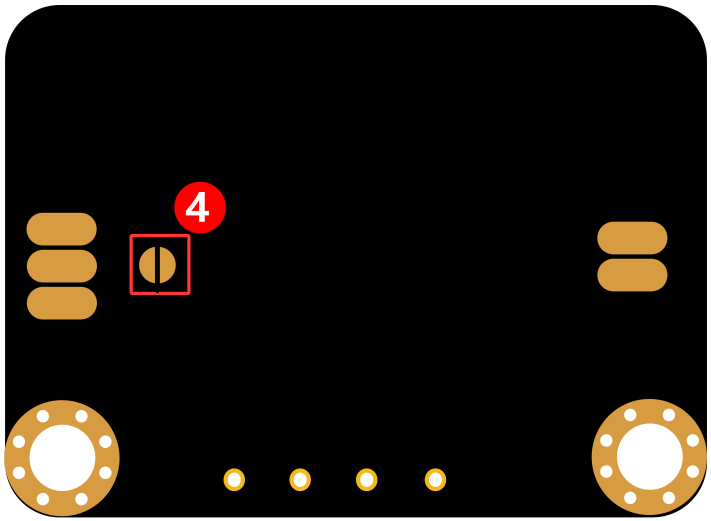
Specification

- Operating Voltage: 1.2-5.5V DC
- Operating Current: <2mA @Eddystone TLM
- Supported sensors: 1.2-3.3V digital/analog sensors
- Input Signal: digital/analog signals
- Operating Frequency Range: 2.4GHz ISM

- Modulation: GFSK
- Transmitting Power: +5.0dBm
- PCB Size: 27mm×33.5mm/1.06×1.32inch
- Mounting Hole Size: inner diameter of 3.1mm/outer diameter of 6mm

Pinout





NO.	Name	Description
1	Power Input	1.2-5.5V DC power input
2	Burning/Debugging	Used for module debugging and burning
3	Sensor Signal Input	"A": Sensor signal input "-": Sensor power supply GND "+": Sensor power supply VCC
4	Sensor VCC Select	Short circuit: 3.3V continuous power supply Disconnected (default): supply power only when broadcasting

Power supply description

When using a 1.2-3.3V power supply to power the beacon, the supply voltage at the sensor side follows the input voltage, for example, if a 1.5V AAA battery is used to power the beacon, the beacon will work normally and will provide 1.5V to the sensor. When using a 3.3-5.5V power supply to power the beacon, the supply voltage at the sensor side is a stable 3.3V.

Quick Start Guide

The guide demonstrates how to get sensor data by mobile app and ESP32 when the data is configured in custom format.

1. Requirements

- **Hardware**
  - TEL0149 Gravity: BLE Sensor Beacon x 1
  - 3.3V USB-TTL Tool x 1
    - Gravity: Analog LM35 Temperature Sensor (https://www.dfrobot.com/product-76.html) (or other analog sensors) x 1
  - Windows/Linux/Mac OS PC
    - ESP32
- **Software**
  - Recommended Mobile App: nRF Connect (iOS (https://apps.apple.com/us/app/nrf-connect-for-mobile/id1054362403), Android (https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp)), LightBlue (iOS

(https://apps.apple.com/us/app/lightblue/id557428110)), BLE Hero (iOS (https://apps.apple.com/us/app/ble-hero/id1013013325))

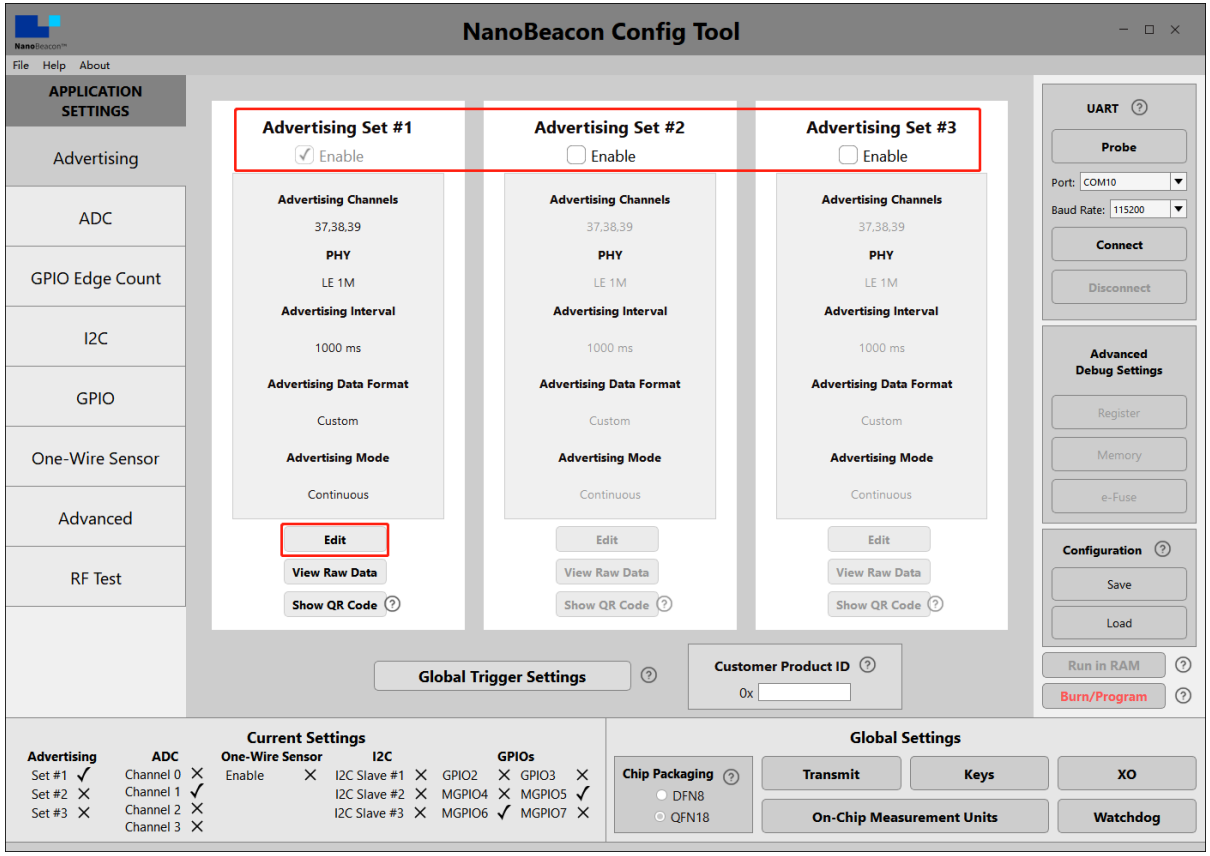
- nRF Connect APK (https://dfimg.dfrobot.com/nobody/wiki/30a8fac07e23db2da802e9647fb761ea.zip)
- Beacon Config Tool: NanoBeaconConfigTool_V3.2.11
- Arduino IDE & ESP32 Environment: How to use FireBeetle_ESP32_E for the first time? (https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654#target_6)

2. Configure Sensor Beacon

Note: The module can only be burned once, so don't click "Burn/Program" before confirming the configuration information. Test the module through "Run in RAM", which can be used infinitely before burning. The system will reset when powered off.

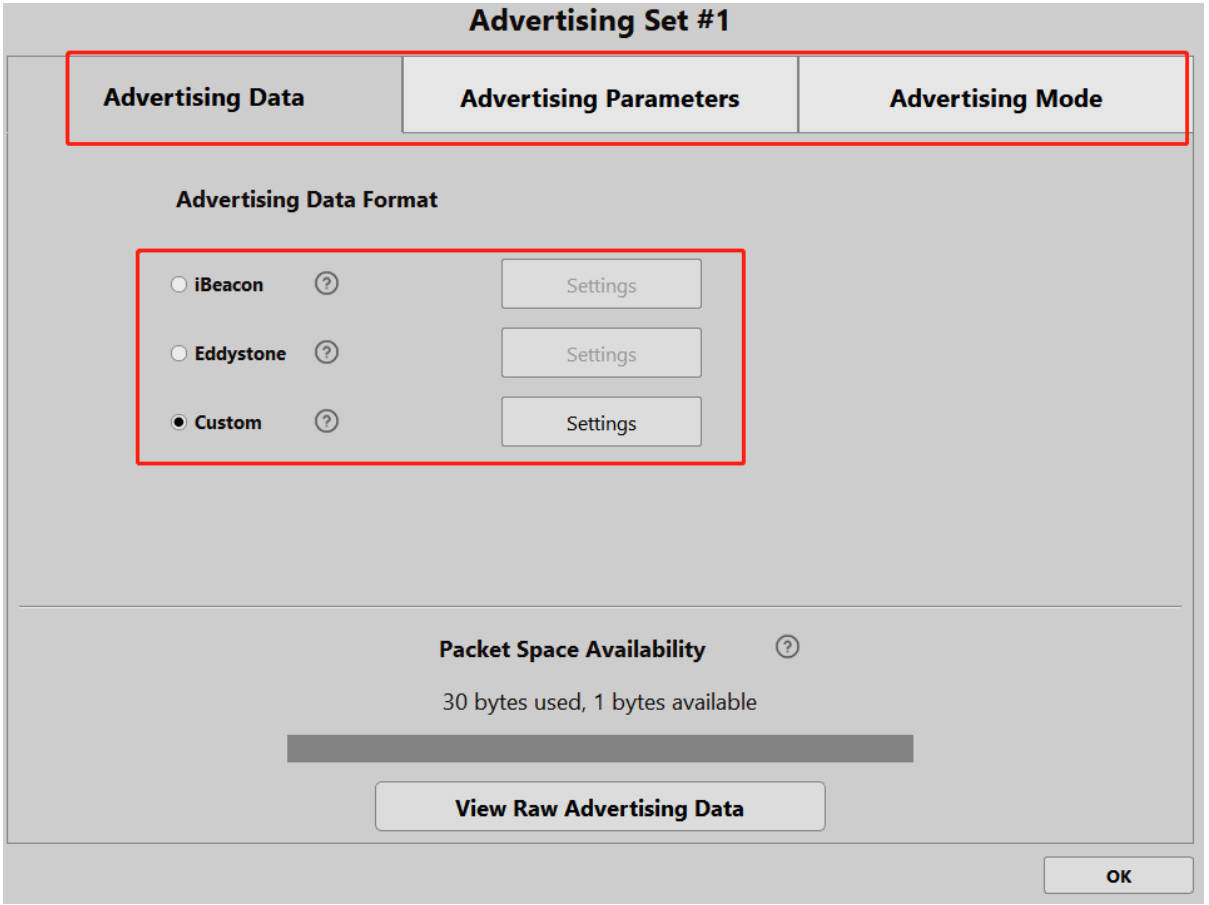
- **1. Download NanoBeaconConfigTool_V3.2.11 and run NanoBeaconConfig.exe.**
- **2. Advertising**

This Gravity: BLE sensor beacon supports three advertising sets. Tick to Enable it. One of them is enabled by default; click **Edit** to enter the config page.



- **3. Advertising Set#1 - Edit - Advertising Data**

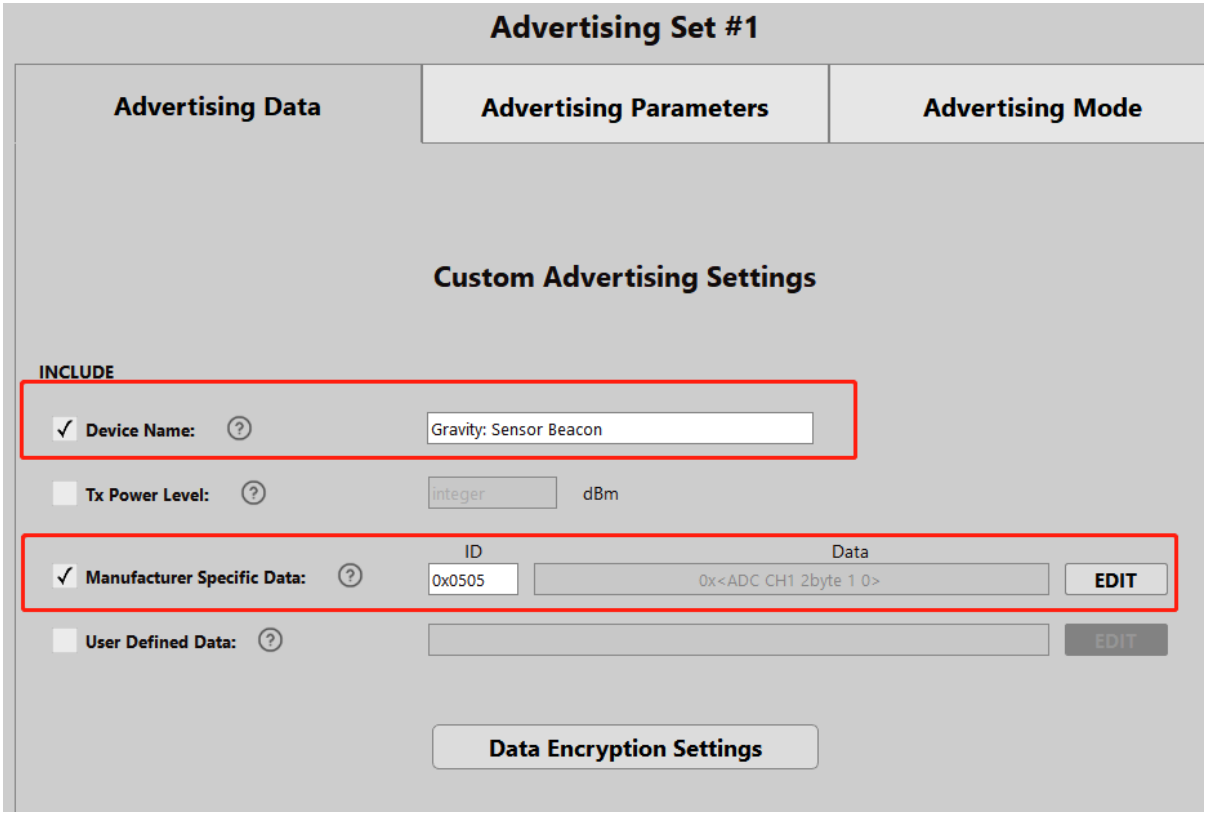
Three data formats are supported: iBeacon, Eddystone and Custom. This tutorial mainly uses custom data format.



- **4. Advertising Set#1 - Edit - Advertising Data - Custom Settings**

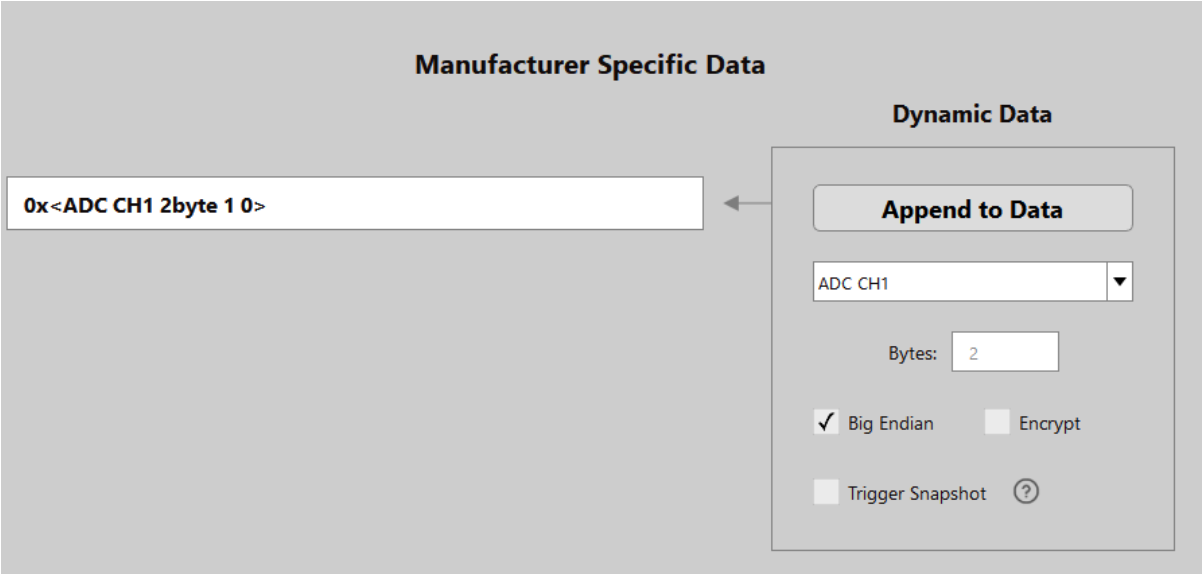
Check "Device Name", and enter "Gravity: Sensor Beacon" or other names. So later it will be easy to scan and find the device on the mobile phone or ESP32 by name.

Check "Manufacturer Specific Data", and click "EDIT" to configure data.



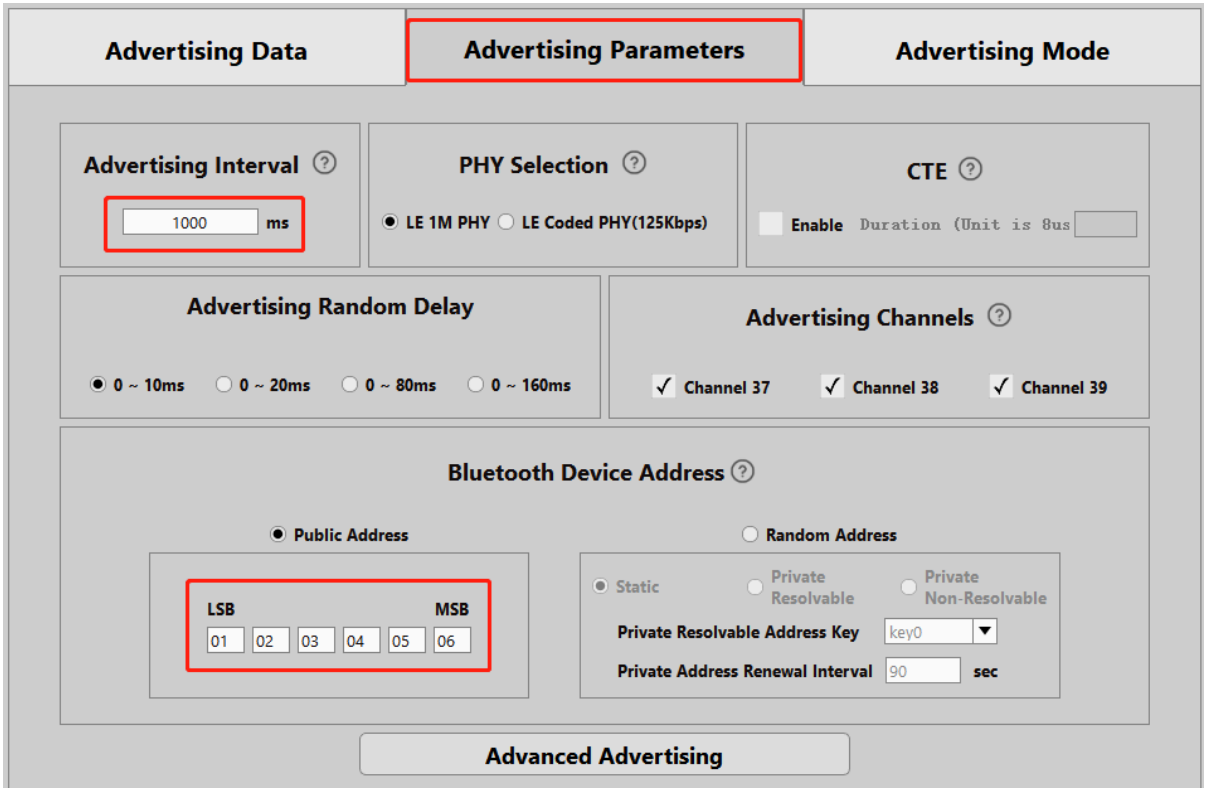
- **5. Advertising Set#1 - Edit - Advertising Data - Custom Settings - EDIT**

Only one analog data is configured here. Select "ADC CH1" in the drop-down menu, check "Big Endian", click "Append to Data", and then "0x<ADC CH1 2byte 1 0>" appears in the window. Click OK to exit.



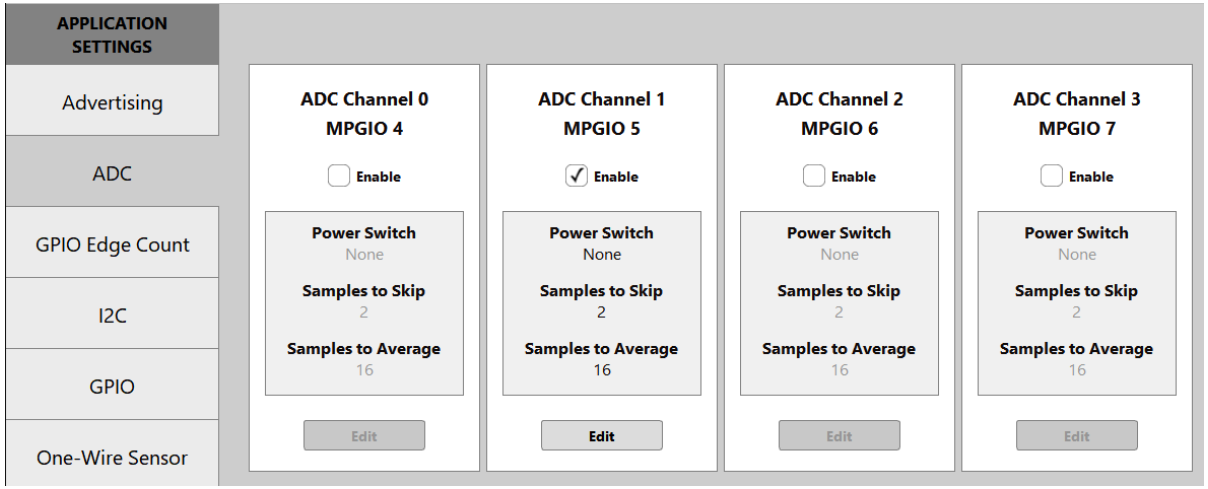
- **6. Advertising Set#1 - Edit - Advertising Parameters**

The advertising interval and address are set here. Make changes as required, and click OK to exit when done. Now the advertising data format is configured, and the module will broadcast data once every 1s.



- **7. ADC**

Next, configure ADC. The Gravity: BLE sensor beacon uses IO5 for analog acquisition, so enable "ADC Channel 1 MPGIO 5" in the ADC config page, and click Edit to set.



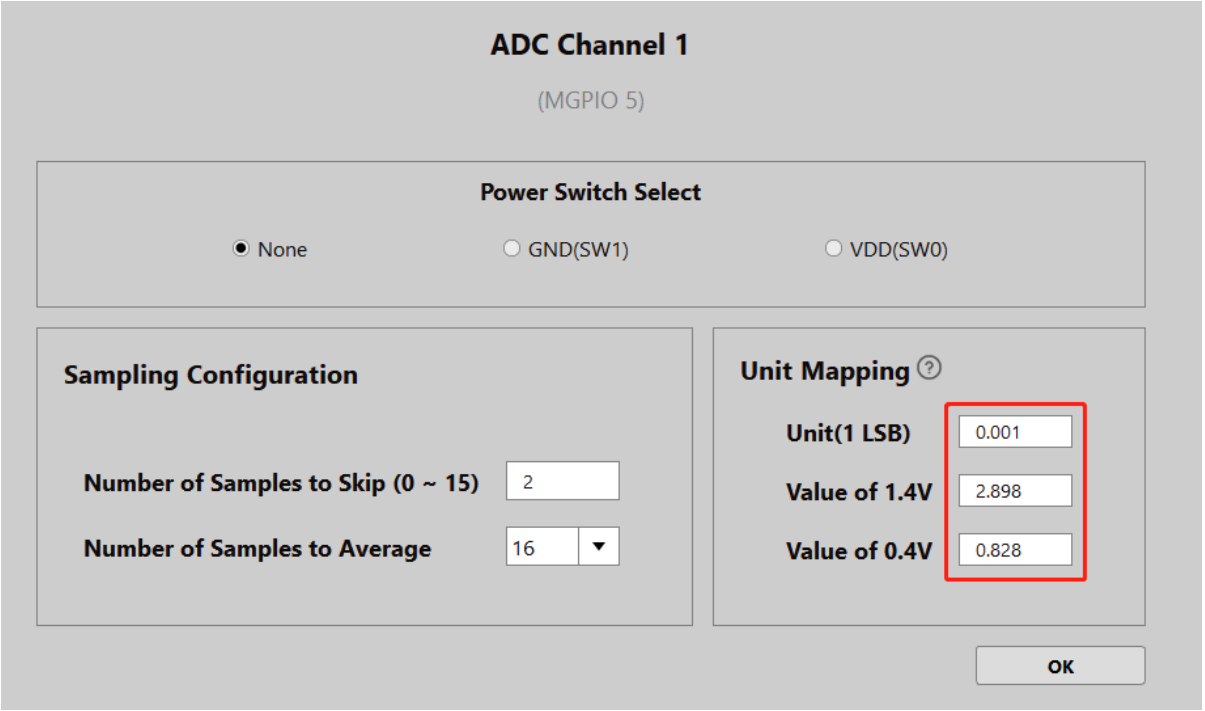
- **8. ADC - ADC Channel 1 MPGIO 5 - Edit**

Change the unit to 0.001 for easy calculation, which has little effect on the accuracy. But if high accuracy is required, leave it alone. Since the analog input voltage is divided (2.06) in the circuit, it is necessary to remap the divided voltage value.

Change Value of 1.4V to 2.898

Change Value of 0.4V to 0.828

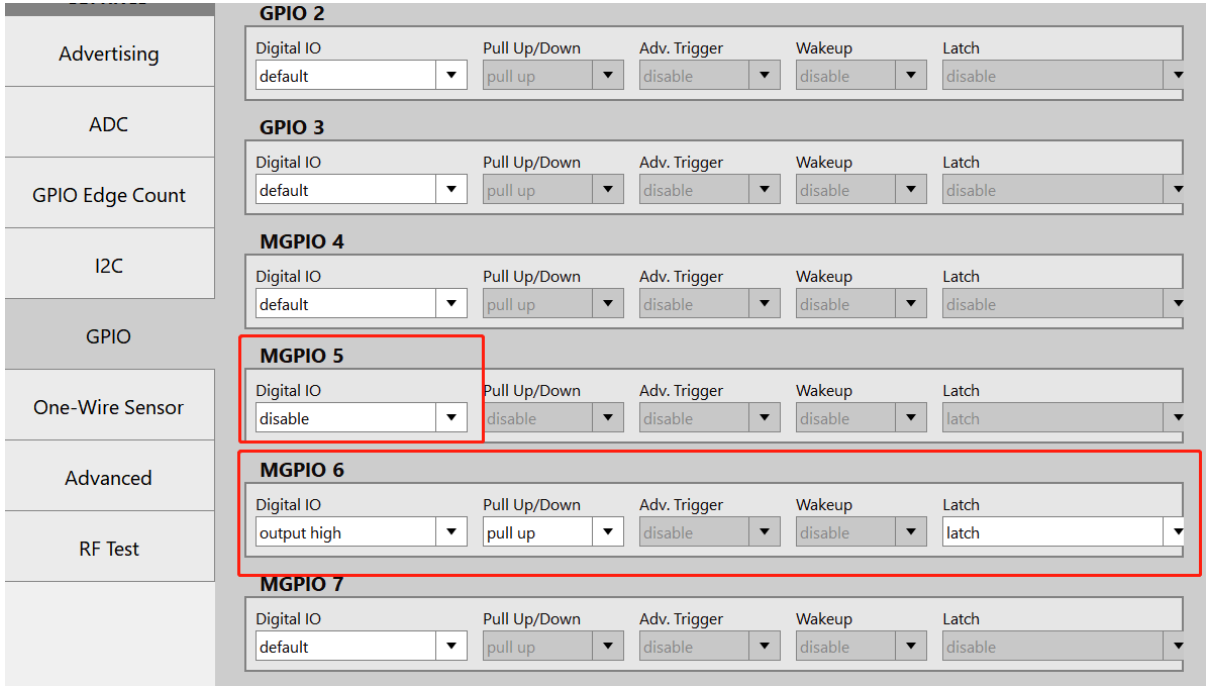
Now the ADC sampling config is completed, and the data broadcasted by the beacon will be the voltage of the "sensor signal input"; unit is mV.



- **9. GPIO**

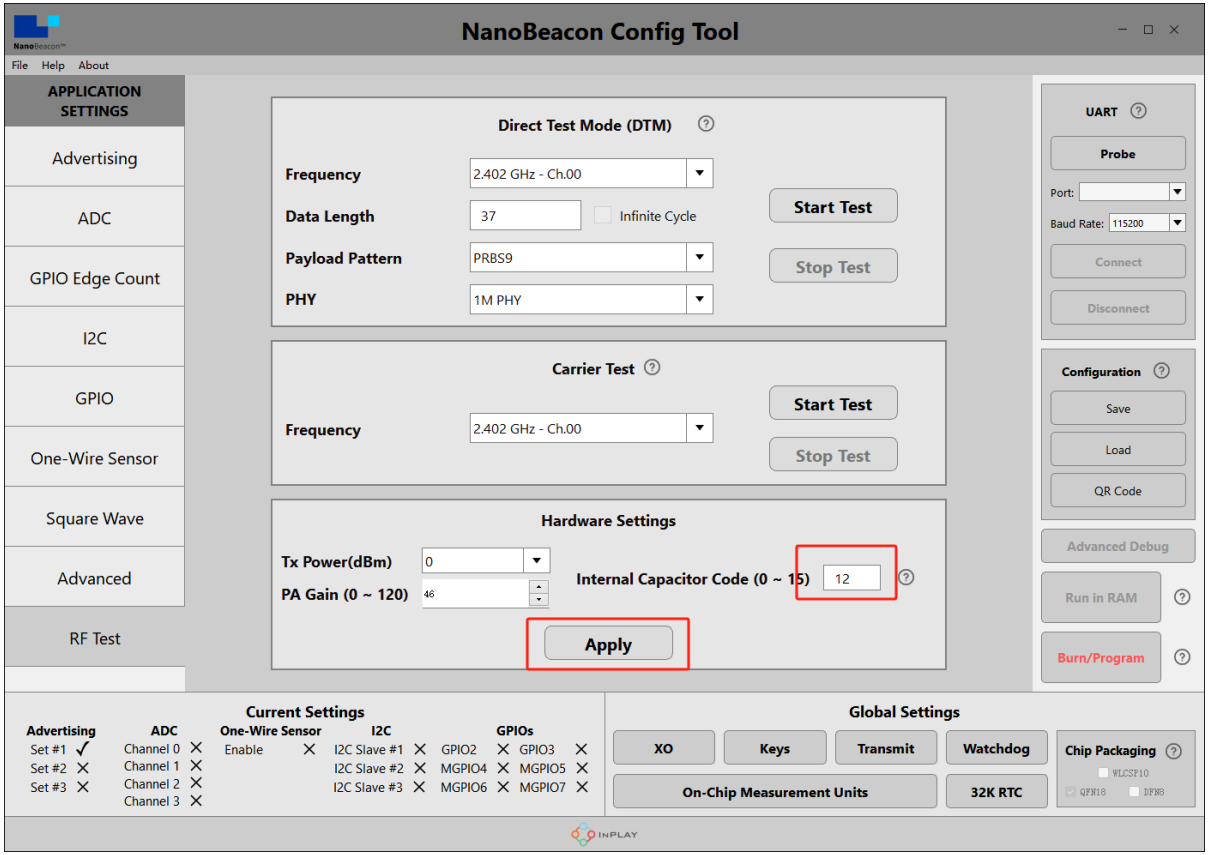
Since MGPIO 5 serves as ADC input, it needs to be configured as "disable".

MGPIO 6 will be used as a power supply for the sensor, so configure it as "output high" "pull up" and "latch" to keep it outputting a high 3.3V for powering sensor.

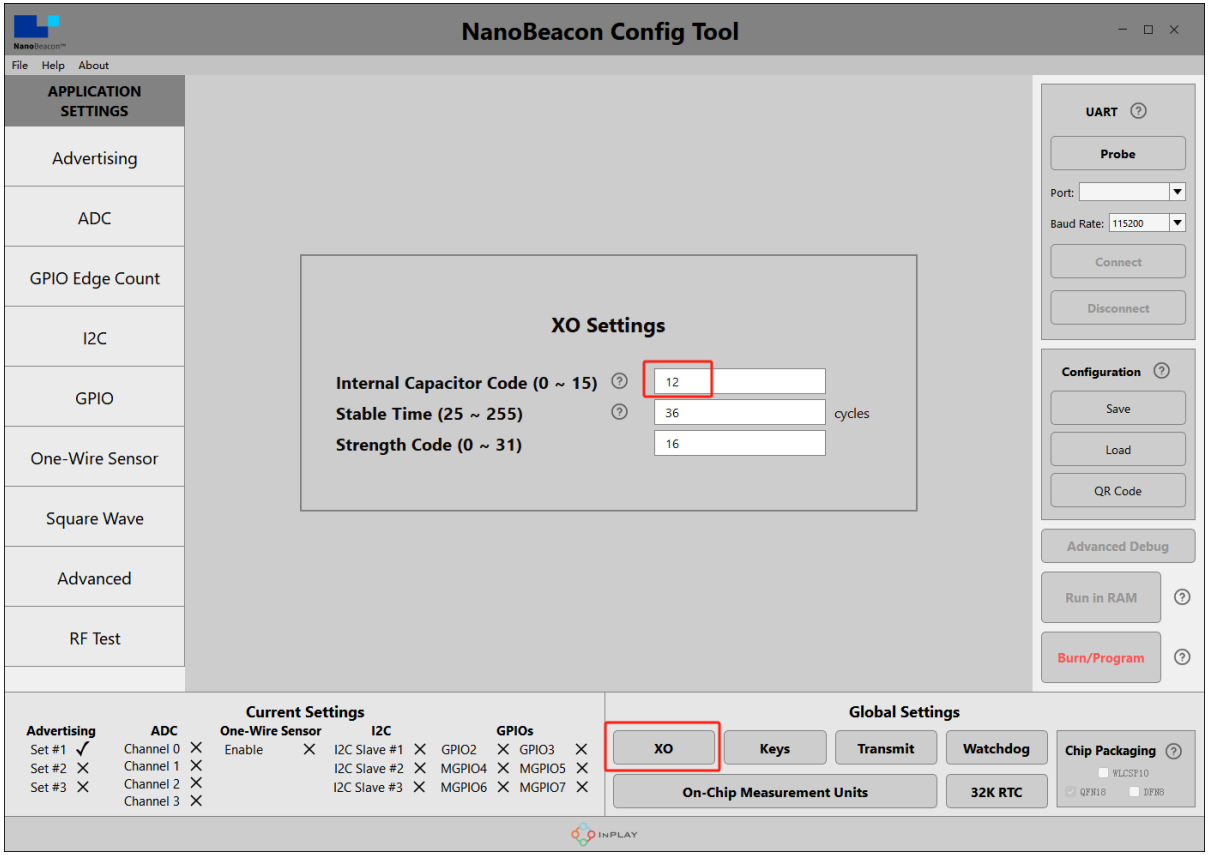


- **10.Crystal Capacitance Matching**

  The NanoBeaconConfig Tool can be set to match the crystal capacitance, and in conjunction with our circuit, in order to keep the frequency bias at an optimal level, we recommend that you change the following two parameters to 12.



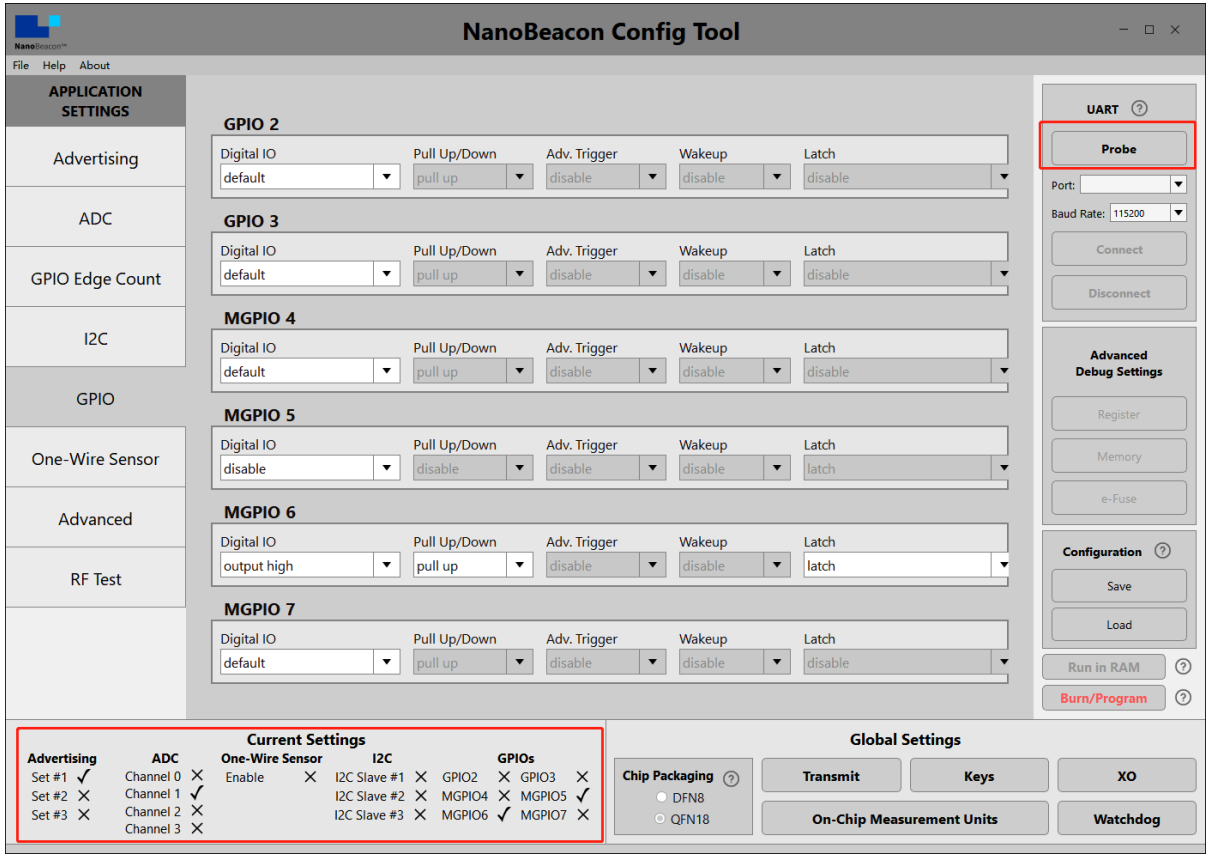
(https://img.dfrobot.com.cn/wiki/62b2fb5caa613609f271523c/6b45f4a6dbe685ae2392096d4813002d.png)



(https://img.dfrobot.com.cn/wiki/62b2fb5caa613609f271523c/602428247045a63ea5ac83d26b44f4d0.png)

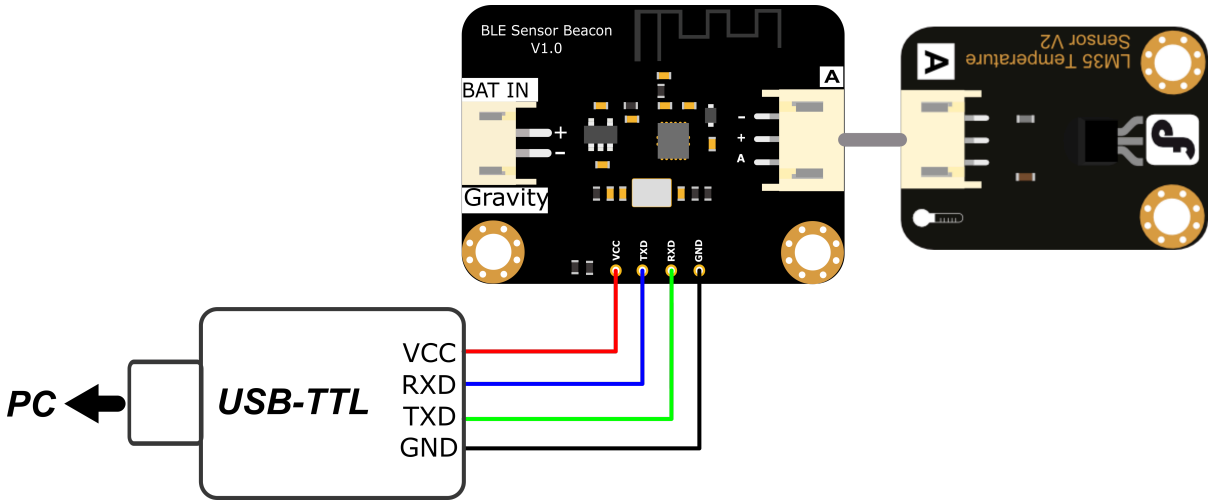
- **11. Check Config**

As shown at the lower left corner of the page, Set #1, ADC Channel 1, MGPIO5 and MGPIO6 are enabled.



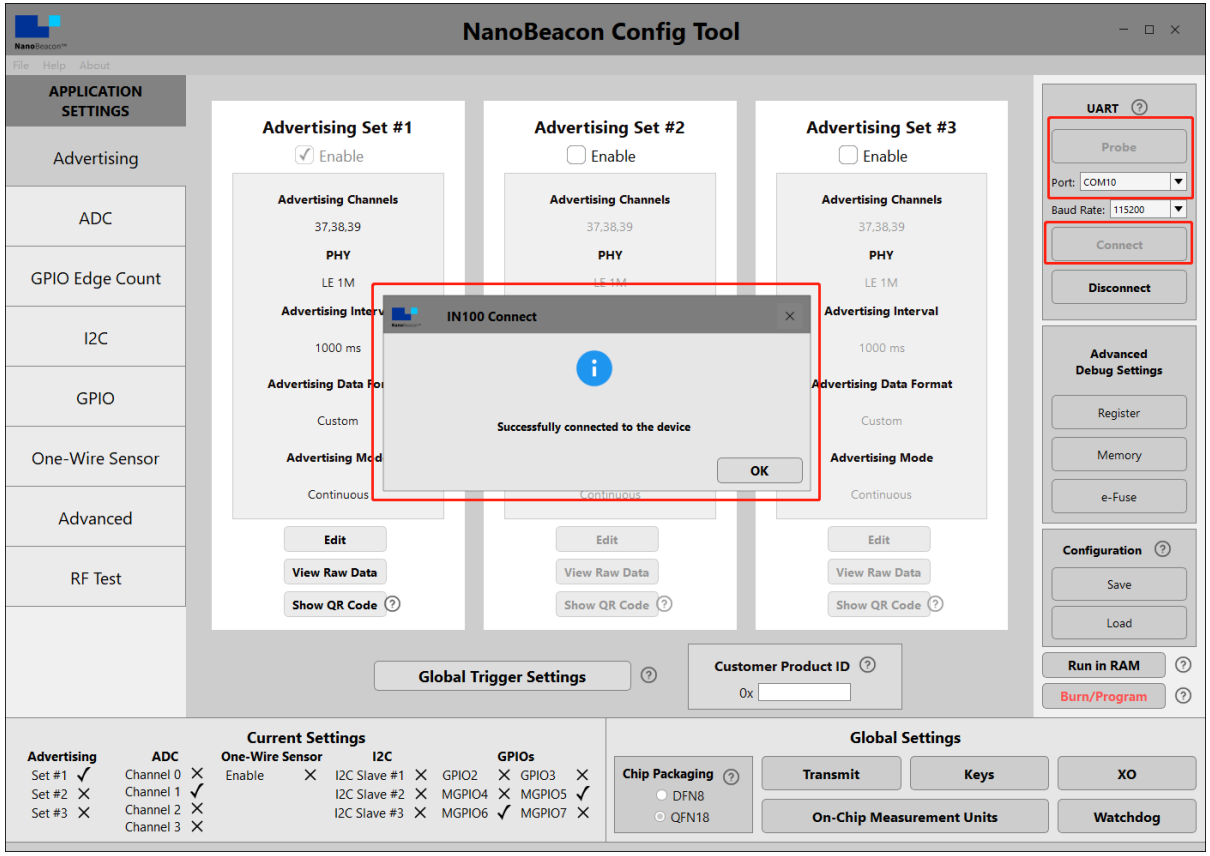
- **12. Hardware connection**

Connect hardware according to the connection diagram.



- **13. Connect module to PC**

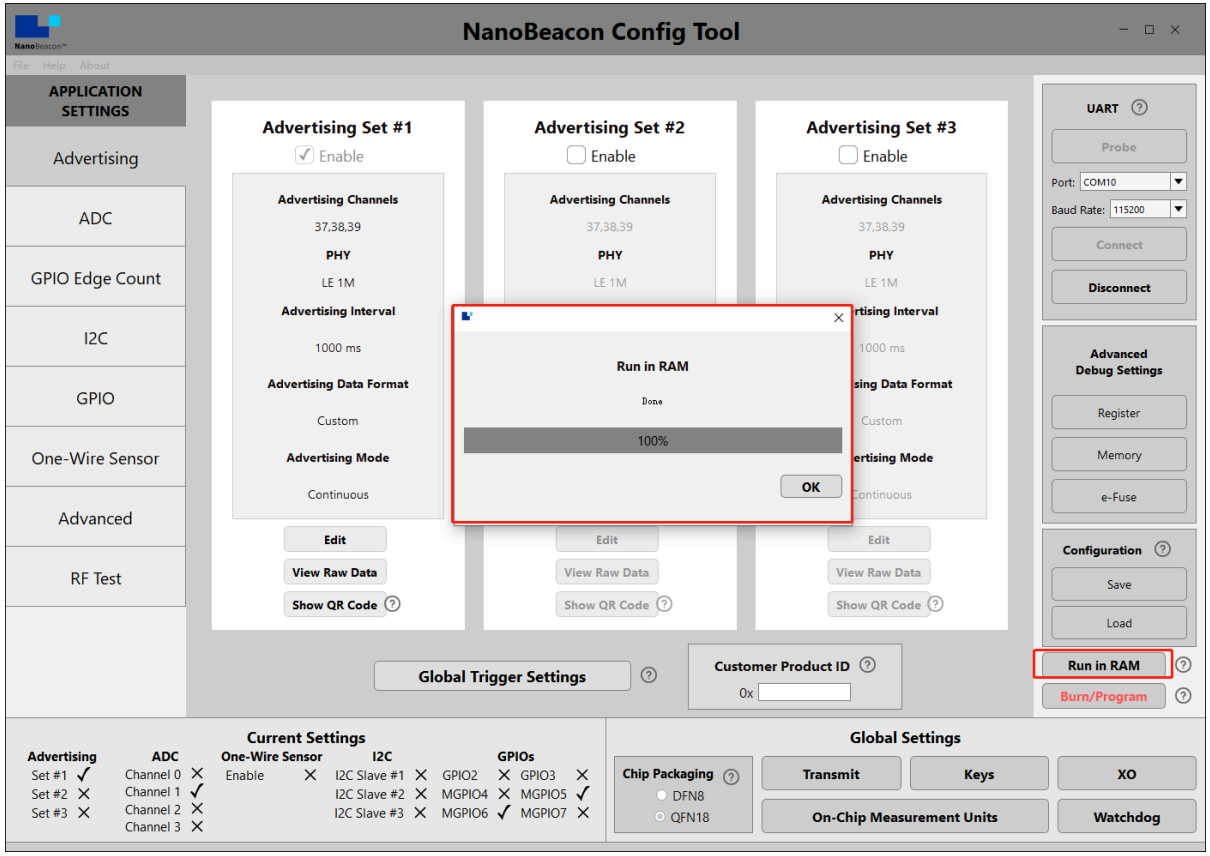
Click "Probe" at the upper right corner to refresh the port, then select the corresponding port, and click "Connect". A pop-up window will appear after a successful connection.



- **14. Run Test**

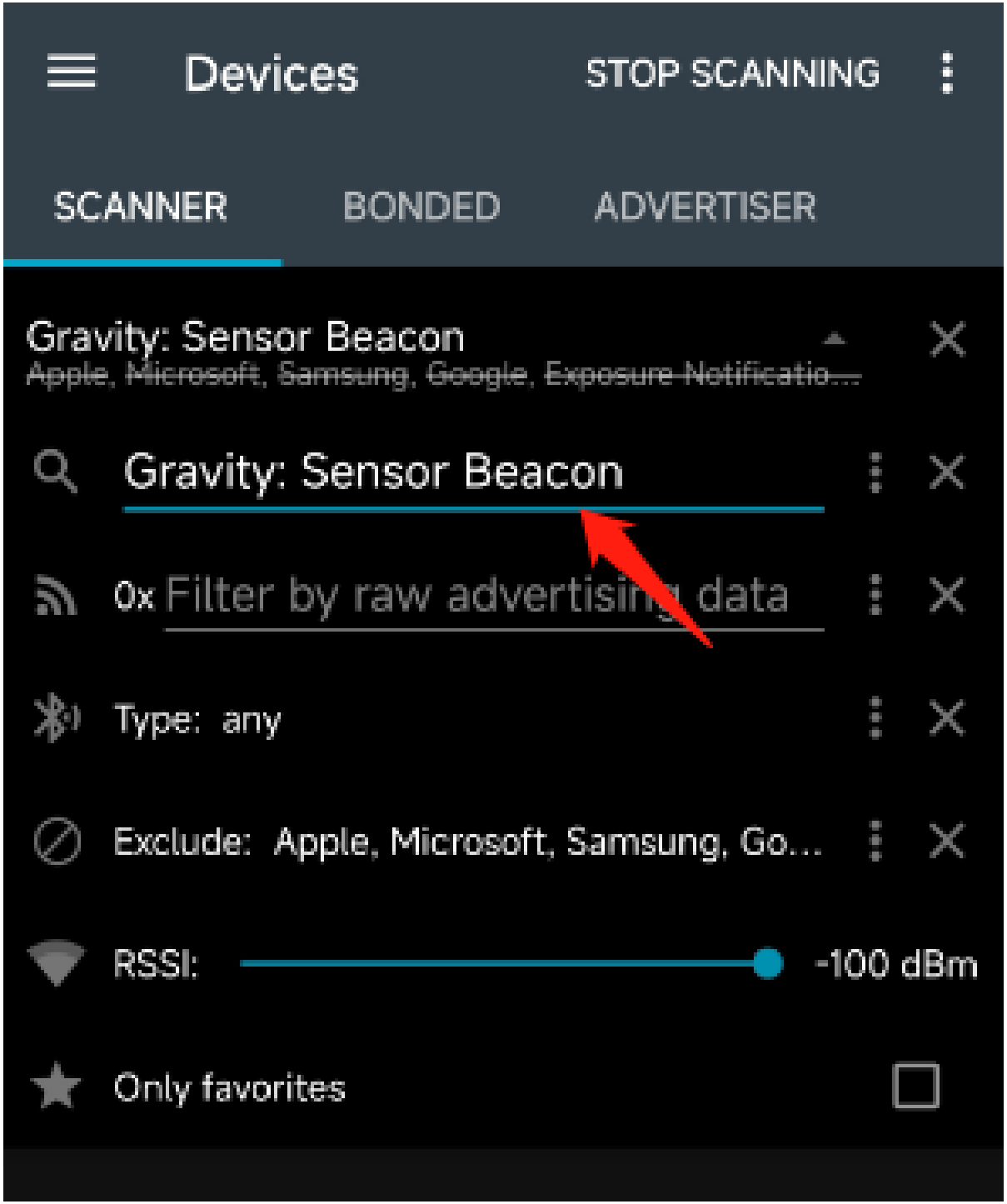
Click "Run in RAM", and a pop-up window will appear when it's done.

Note: The module can only be burned once, so don't click "Burn/Program" before confirming the configuration information. The module can be tested through "Run in RAM", which can be used infinitely before burning. The system will reset when powered off.



3. Get Data via Mobile App

- i. Take Android phone as an example, install and open nRF Connect.apk (https://dfimg.dfrobot.com/nobody/wiki/61fabd11c754c46a02685bf36a6f83ea.zip).
- ii. If there are too many other beacon devices nearby, find the device by entering the device name of the beacon in the filter. In the tutorial step 4, beacon config, the device has been named as "Gravity: Sensor Beacon".



- iii. Only "Gravity: Sensor Beacon" is kept in the menu; click to see the details.
- iv. Data Interpretation

"Gravity: Sensor Beacon" is the Device Name set in step 4 of the tutorial for beacon config;

"06:05:04:03:02:01" is the address set in step 6;

"0X00E3" is the ADC-sampled data set in step 5.

- v. Sensor Data Calculation

The known sensor data sampled by the beacon is "0X00E3", equalling 227 when converted to a decimal number, which means the voltage value sampled by the beacon is 227mV.

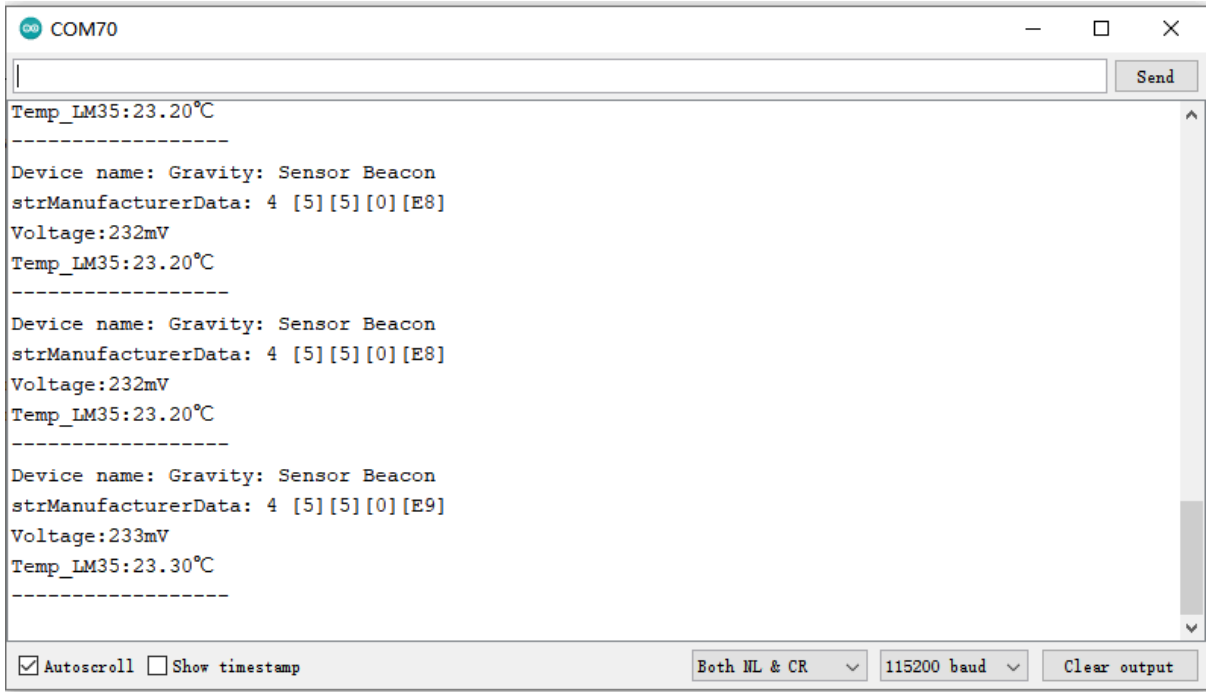
The sensor connected is LM35 temperature sensor. And LM35 wiki shows the relationship between its output voltage and temperature: 10mV for one degree Celsius, which means the sensor temperature data broadcasted by the beacon is 22.7°C.

4. Get Data with ESP32

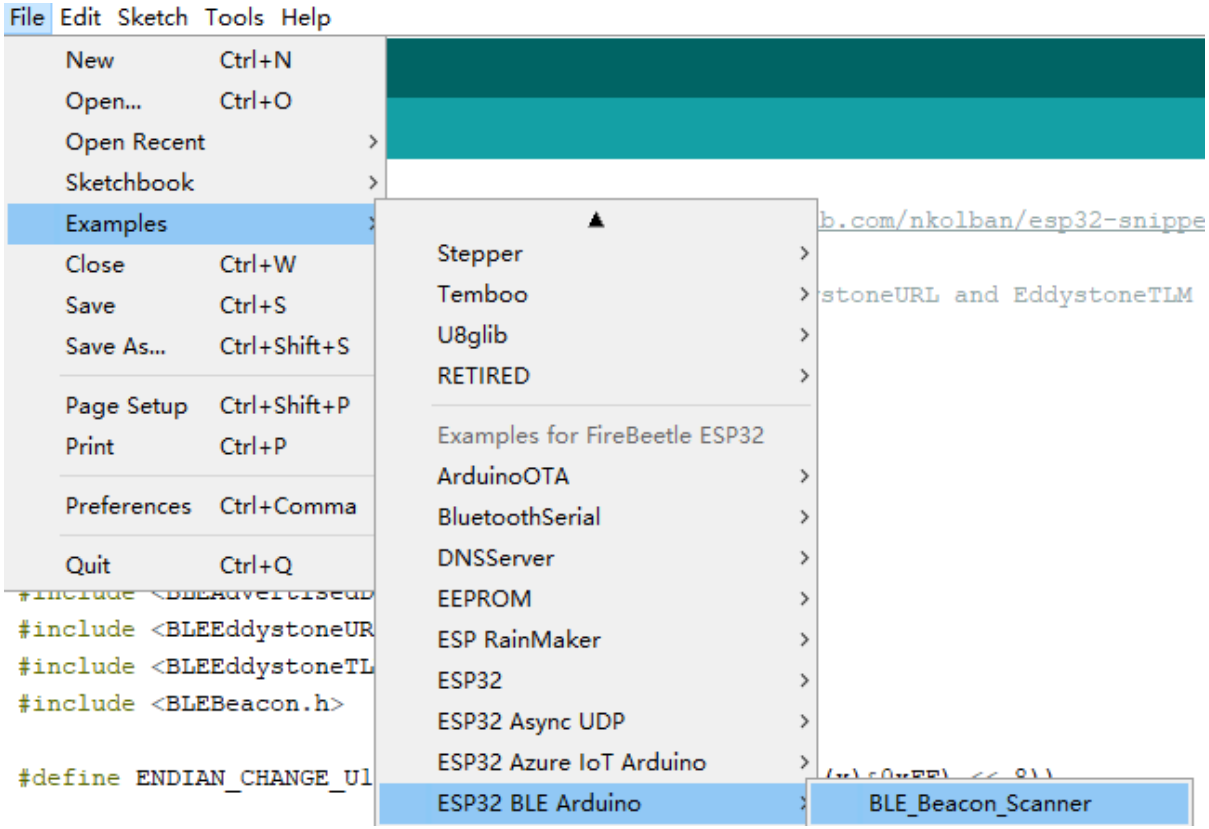
- Prepare Arduino IDE & ESP32 Environment: How to use FireBeetle_ESP32_E for the first time? (https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654#target_6)
- Burn codes below for ESP32

```
/*
   Based on Neil Kolban example for IDF: https://github.com/nkolban/esp32-snippets/blob/maste
   Ported to Arduino ESP32 by Evandro Copercini
   Changed to a beacon scanner to report iBeacon, EddystoneURL and EddystoneTLM beacons by be
*/

#include <Arduino.h>
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>
#include <BLEEddystoneURL.h>
#include <BLEEddystoneTLM.h>
#include <BLEBeacon.h>
#define ENDIAN_CHANGE_U16(x) ((((x)&0xFF00) >> 8) + (((x)&0xFF) << 8))

float Sensor_Data;
int scanTime = 5; //In seconds
BLEScan *pBLEScan;

class MyAdvertisedDeviceCallbacks : public BLEAdvertisedDeviceCallbacks
{
    void onResult(BLEAdvertisedDevice advertisedDevice)
    {
      if (advertisedDevice.haveName())
      {
        if(String(advertisedDevice.getName().c_str()) == "Gravity: Sensor Beacon"){
          Serial.print("Device name: ");
          Serial.println(advertisedDevice.getName().c_str());
          std::string strManufacturerData = advertisedDevice.getManufacturerData();
          uint8_t cManufacturerData[100];
          strManufacturerData.copy((char *)cManufacturerData, strManufacturerData.length(), 0
          Serial.printf("strManufacturerData: %d ", strManufacturerData.length());
            for (int i = 0; i < strManufacturerData.length(); i++)
            {
```

```
            Serial.printf("[%X]", cManufacturerData[i]);
          }
          Sensor_Data = int(cManufacturerData[2]<<8 | cManufacturerData[3]);
          Serial.println();
          Serial.print("Voltage:");Serial.print(int(Sensor_Data));Serial.println("mV");
          Serial.print("Temp_LM35:");Serial.print(Sensor_Data/10);Serial.println("℃");
          Serial.println("------------------");
        }
      }
    }
};
void setup()
{
  Serial.begin(115200);
  Serial.println("Scanning...");

  BLEDevice::init("");
  pBLEScan = BLEDevice::getScan(); //create new scan
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setActiveScan(true); //active scan uses more power, but get results faster
  pBLEScan->setInterval(100);
  pBLEScan->setWindow(99); // less or equal setInterval value
}
void loop()
{
  // put your main code here, to run repeatedly:
  BLEScanResults foundDevices = pBLEScan->start(scanTime, false);
  pBLEScan->clearResults(); // delete results fromBLEScan buffer to release memory
  delay(2000);
}
```



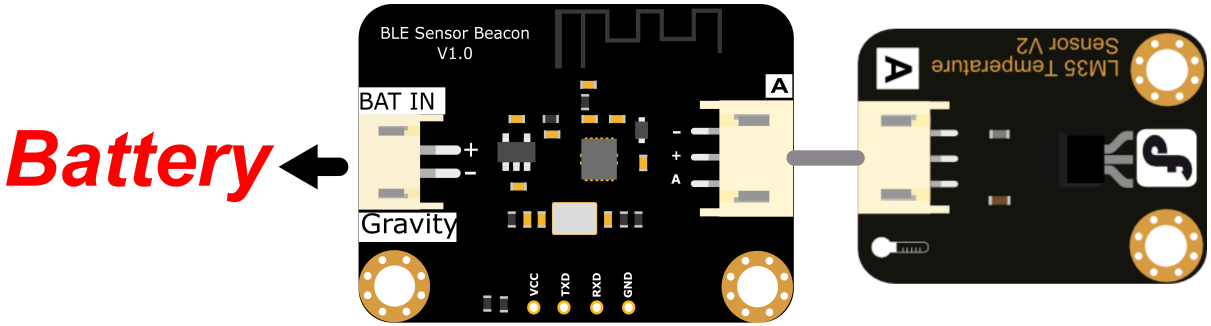
- The codes are modified based on the built-in BLE_Beacon_Scanner on ESP32. Please make changes when necessary.



5. Confirm Data & Burn

- **Note: The module can only be burned once, skip the step if you're just for function test.**
- The data above broadcast in the custom format. For configuring other data formats, please refer to software specifications for details.
- Check and confirm data, then burn it into the chip.
- Click the "Burn/Program" button at the lower right to burn codes; the corresponding pop-up window will appear when done.

- After burning, you can disconnect the module from the programming device and power the module with batteries.



Advanced Application

1. Broadcast Eddystone TLM Data

The Eddystone TLM comes with temperature data obtained from the temperature sensor inside the beacon chip. But it may be affected by the heat generated by the chip.

When configuring Eddystone TLM, select "Eddystone" in "Advertising Data Format", and "TLM Frame" in "Eddystone - Settings". Keep parameters like broadcast interval and address as default or make changes when necessary.

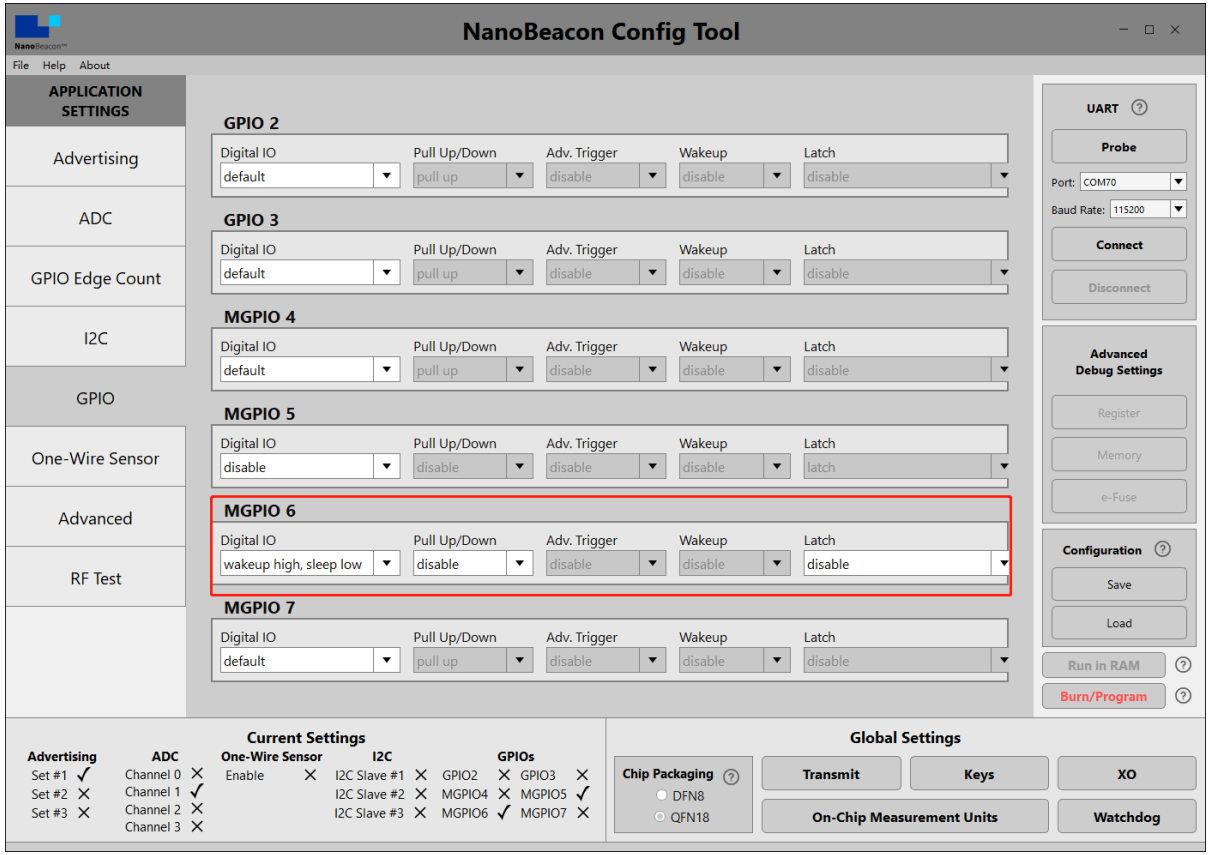
2. Power Supply for High-power Sensor

When using MGPIO6 as power output, it may fail to drive some high-power sensors due to weak loading capacity. So there are jumper pads designed on the back of the beacon board, which allows to power sensor through onboard LDO by short-circuiting, thus providing stabler 3.3V voltage with stronger electrical load capacity.

3. Dynamic Power Supply Control

Dynamic power supply control means only supplying power to the sensor during broadcast, and stopping powering when the broadcast ends. For this purpose, you need to set MGPIO6 as "wakeup high, sleep low" and Latch as "disable".

The sample config file can be loaded directly after downloading: DynamicPower.zip (https://dfimg.dfrobot.com/nobody/wiki/b9cecef20db8edf19d9b0b2dbb8a8e39.zip)

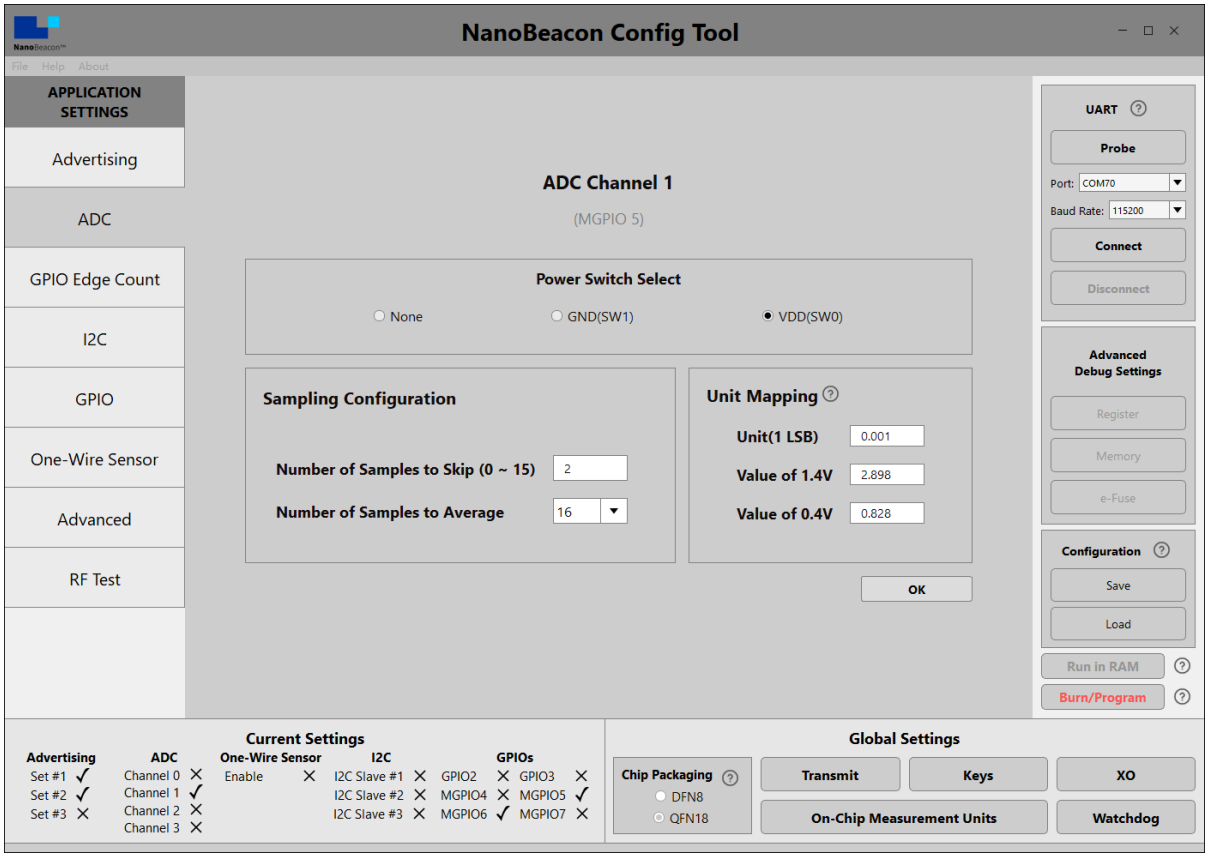


4. Avoid Packet Loss During Long Interval

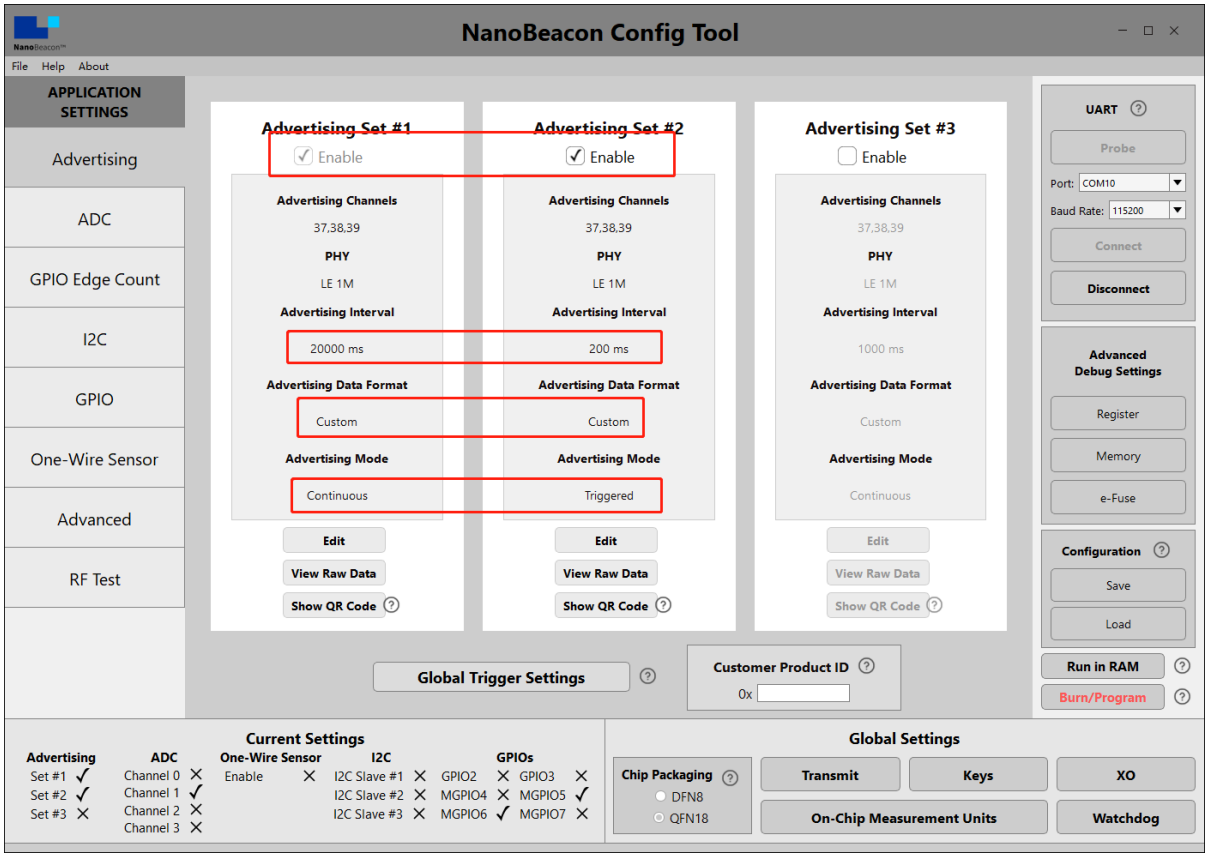
When the advertising interval is set to 10s or even longer, if the receiver fails to receive, it has to wait for the next broadcast from the beacon after 10s, in which the failure risk may still exist. In this case, it is recommended to do multiple broadcasts after the interval.

The steps are shown below:

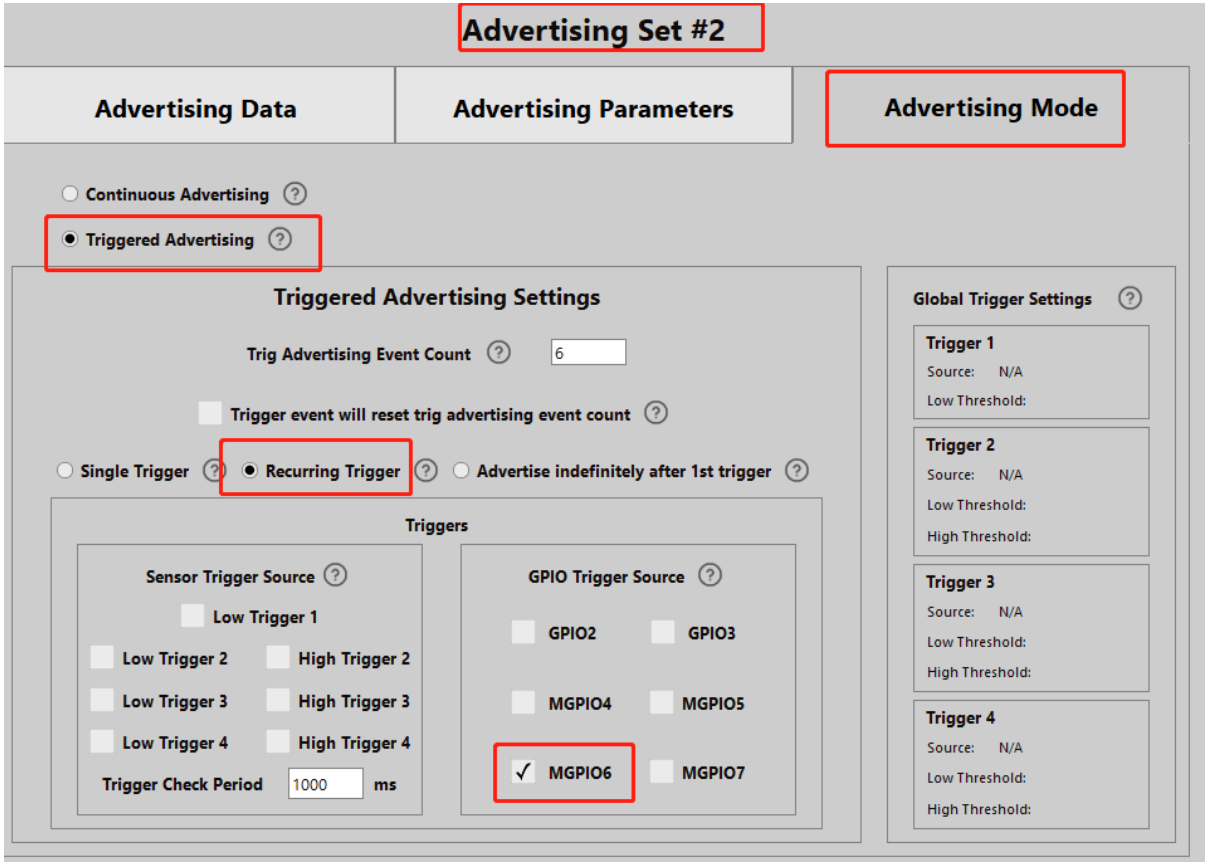
- Enable SW0, the dynamic power supply control port of the beacon chip. It outputs high every time the broadcast is enabled.



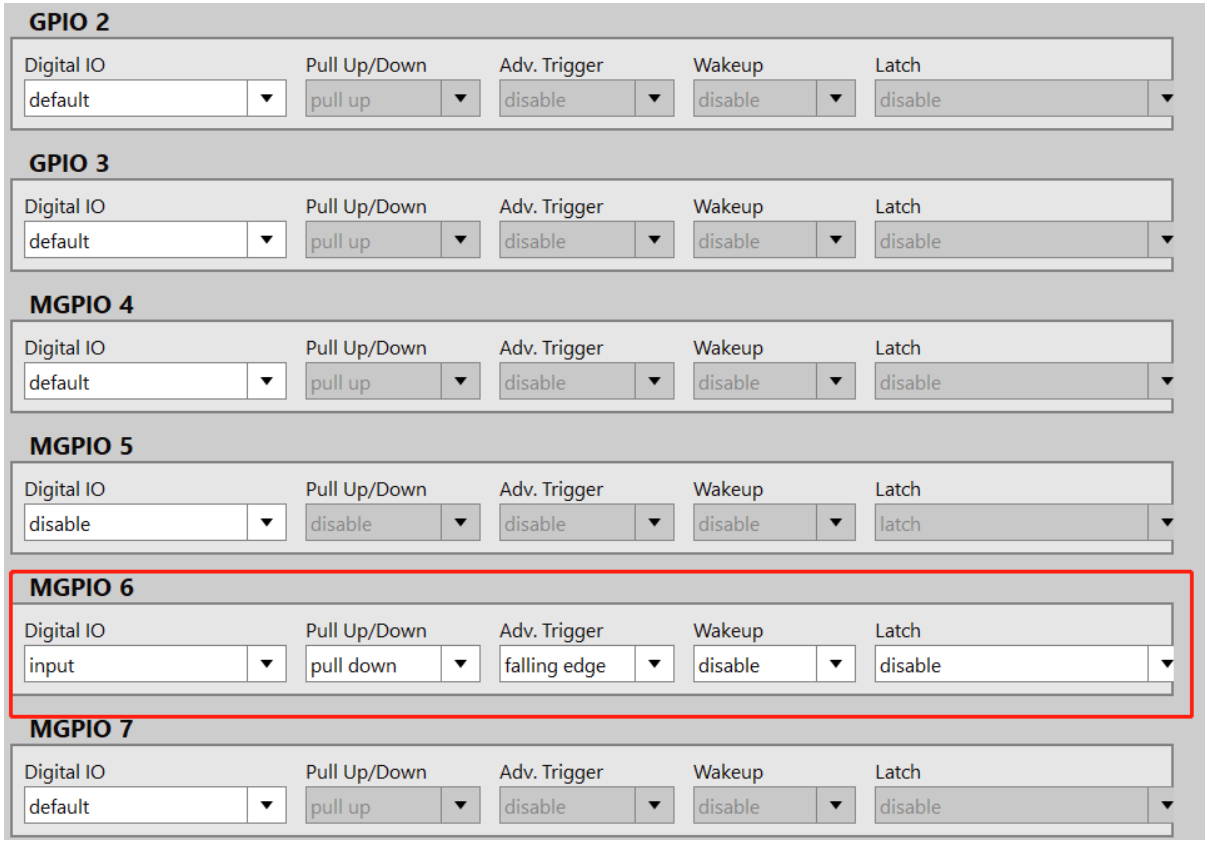
- Enable two advertising sets, set one as continuous advertising of 20s interval, the other as triggered advertising of 200ms interval. And set the same data format for them.



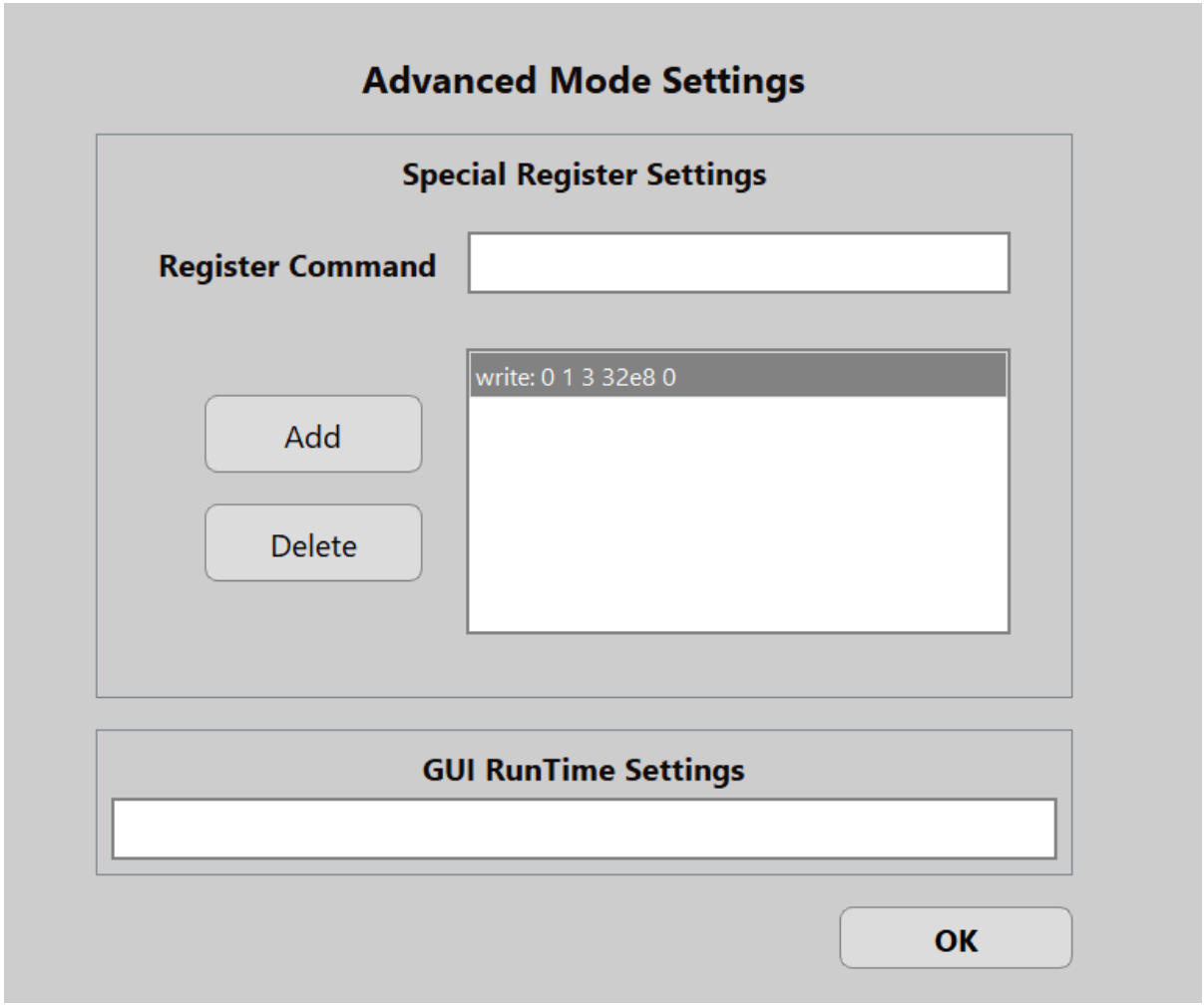
- Continuous advertising will not be talked about further here. For triggered advertising, select MGPIO6 as it is connected to SW0 in the circuit and the level jump of SW0 can be detected through it. There will be 6 data broadcasts every time the SW0 level jump is detected. (In triggered advertising mode, the system will re-enter the detection state only when completing the set times of broadcasts.)



- Set MGPIO6 as input mode, and enable edge detection.



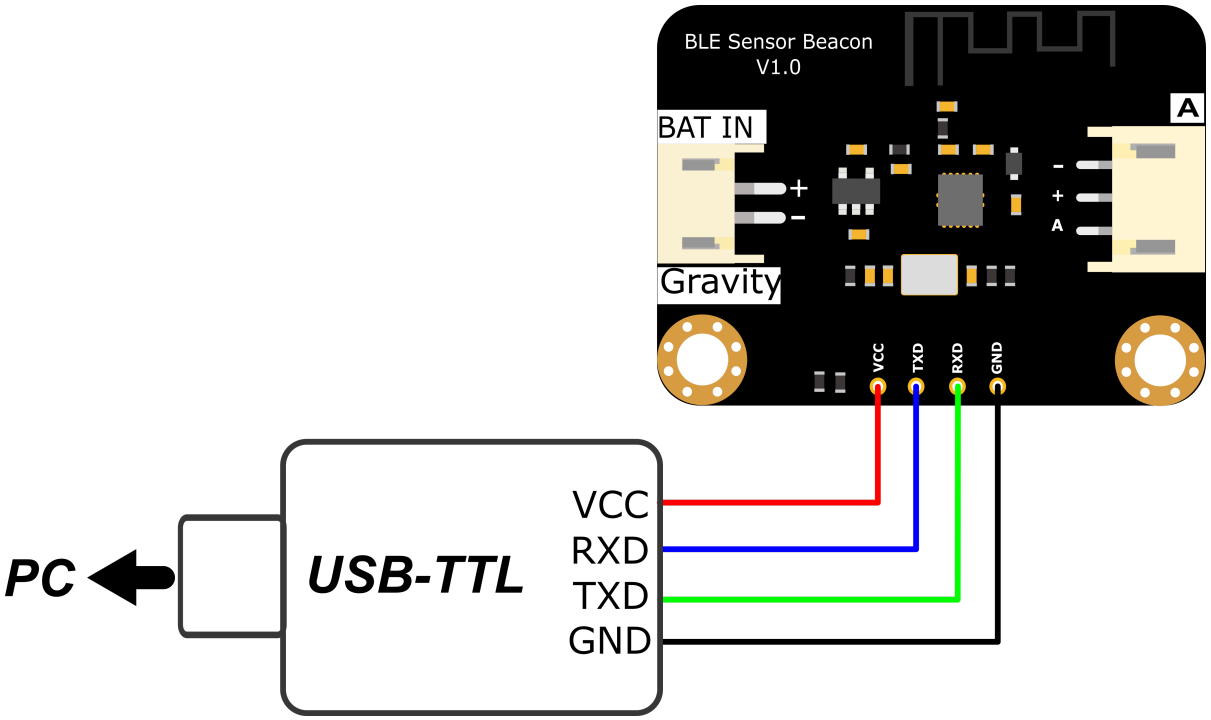
- In the Advanced Mode Settings, enter the command "register write: 0 1 3 32e8 0" to disable the anti-shake feature.



- After completing the operations above, you can do a test by Run in RAM. And the config file above can be loaded directly after downloading: trigger.zip (https://dfimg.dfrobot.com/nobody/wiki/78a418d7b6e323b162d2fe9f199e37a7.zip)

Instructions for NanoBeacon Config Tool

For more information on the use of the NanoBeacon Config Tool, see the software user guide: NanoBeacon Config Tool User Guide EN.pdf (https://dfimg.dfrobot.com/nobody/wiki/5d9b79a87f78ef9c0fe3c98077f89809.pdf) The user guide uses the "Beacon development kit" and when using the Gravity: BLE sensor beacons, the 3.3V USB-TTL tool can be used directly.



FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (https://www.dfrobot.com/forum/).

More Documents

- IN100 Datasheet (https://dfimg.dfrobot.com/nobody/wiki/f30e2bb6d7af5262fe616285a26e4573.pdf)
- TEL0149 Schematic.pdf (https://dfimg.dfrobot.com/nobody/wiki/95c8d95b965c098d5a382fe7cac1ef74.pdf)
- TEL0149 Dimensional.pdf (https://dfimg.dfrobot.com/nobody/wiki/89247fbc6ddc464baa05437b134346a3.pdf)

Turn to the Top