SKU:SEN0374 (https://www.dfrobot.com/product-2142.html)

(https://www.dfrobot.com/product-2142.html)

Introduction

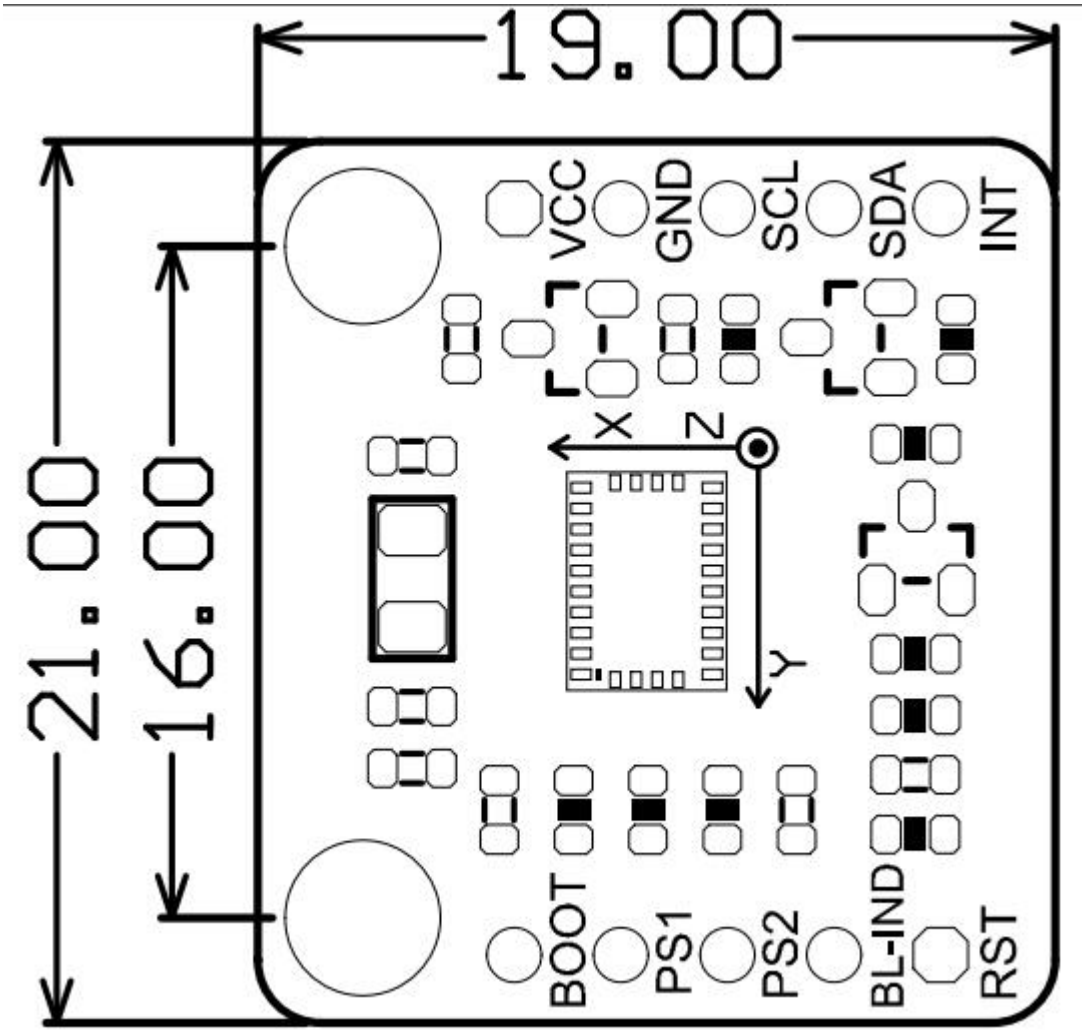
BNO055 is a new sensor IC for implementing an intelligent 9-axis Absolute Orientation Sensor. It is a System in Package, integrating a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope, a triaxial geomagnetic sensor and a 32-bit microcontroller.
With a size of 5.2 x 3.8 x 1.1mm, the BNO055 is significantly smaller than comparable discrete or system-on-board solutions and also is the sensor-hub product of the smallest size that supports Windows 8.1 at present.
BNO055 is able to provide not only single data of the three kinds of sensors (accelerometer/gyroscope/geomagnetic), but also integrated data, such as quaternions, Euler angles or vectors. Besides, the built-in MCU frees the users from the complexities of algorithm processing, which provides application support in many aspects for smart phone, wearable device and so on.

Features

- Outputs fused sensor data: quaternions, euler angles, rotation vector, linear acceleration, gravity, heading.
  - 3 sensors in one device:
    - 16-bit Gyroscope
    - 16-bit Accelerometer
    - Geomagnetic Sensor
  - Intelligent Power Management: normal, low power and suspend mode available

Specification

Specification



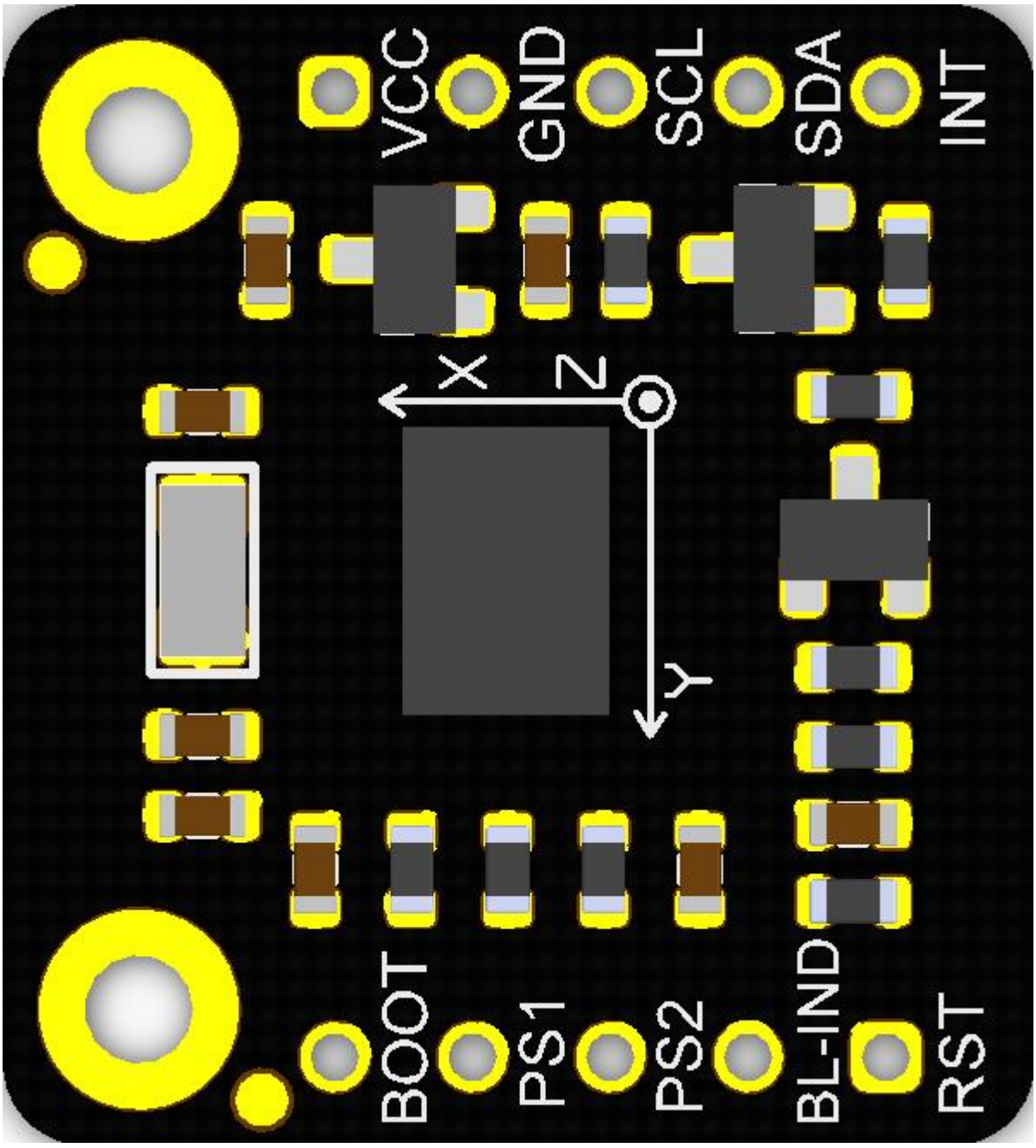
- Operating Voltage: 3.3V-5.5V
- Communication Interface: I2C (Support 5V) or SPI(Non-I2C ports only support 3.3V)
- Default I2C Address: 0x28
- BNO055 Accelerometer:
  - Acceleration ranges ±2g/±4g/±8g/±16
  - Low-pass filter bandwidths 1kHz~<8Hz
  - Operation modes: normal, suspend, low power, standby, deep suspend
- BNO055 Gyroscope:
  - Ranges switchable from ±125°/s~2000°/s
  - Low-pass filter bandwidths 523Hz~12Hz
  - Operation modes: normal, fast power up, deep suspend, suspend, advanced power save.
  - On-chip interrupt control: motion-triggered interrupt-signal
- BNO055 Geomagnetic:

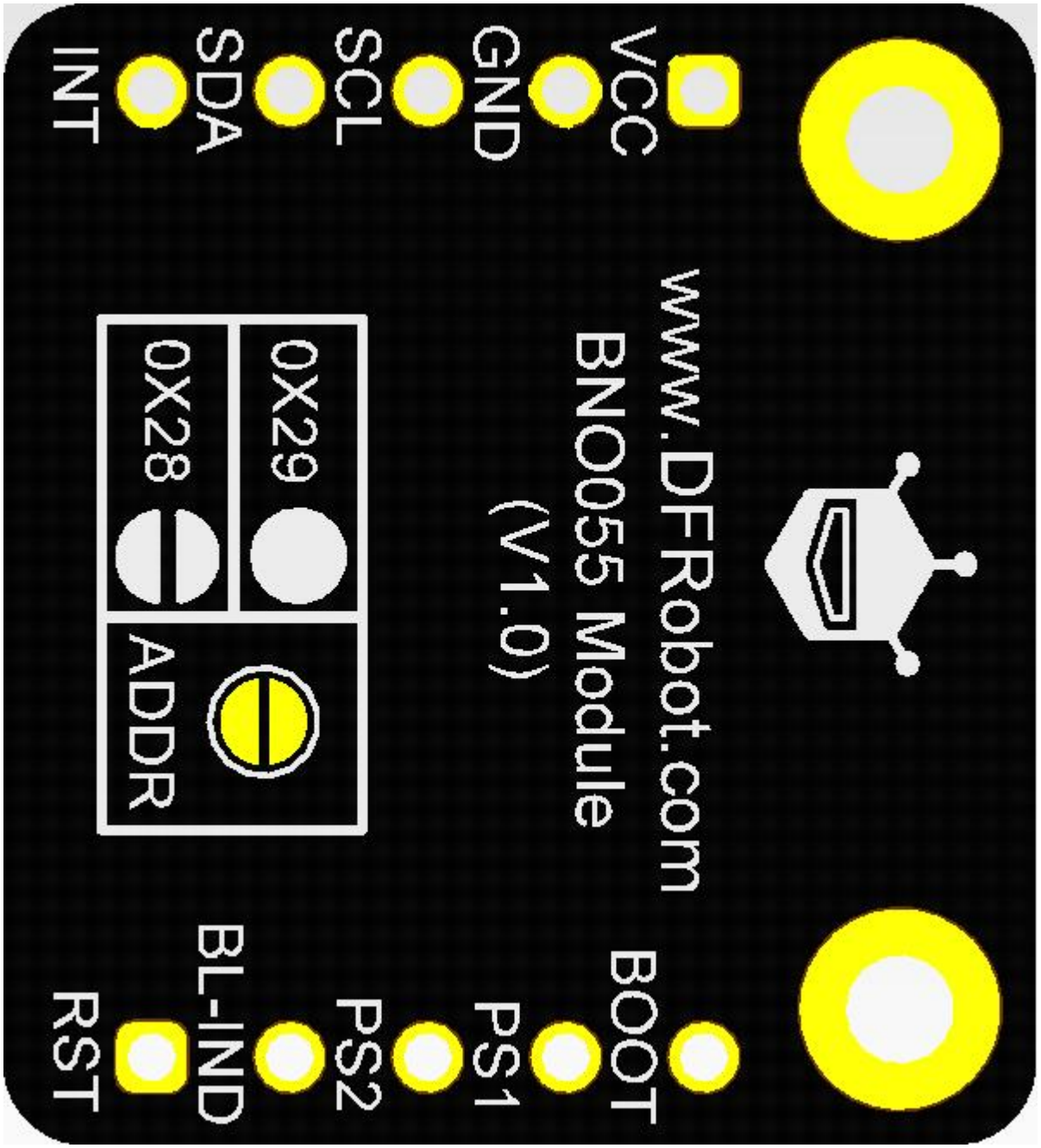
- Magnetic field range typical ±1300uT(x-,y-axis);±2500uT(z-axis)
- Magnetic field resolution: ~0.3
- Operating nodes: low power, regular, enhanced regular, high accuracy
- Power modes: normal, sleep, force

- Outline Dimension: 19 x 21mm/0.75 x 0.83”
- Mounting Hole Position: 16mm
- Mounting Hole Size: inner diameter 2mm/outer diameter 3.7mm

Note: the sensor default I2C address is 0X28.

Board Overview





Silkscreen	Function Description
VCC	+
GND	-
SCL	I2C Clock
SDA	I2C Data
INT	Interrupt Pin
BOOT	Lead Mode select pin

PS1	Protocol Select Pin 1
PS2	Protocol Select Pin 2

Silkscreen	Function Description
BL-IND	Lead Program Guide
RST	Reset Pin
ADDR	I2C Address Select

PS1	PS2	Function
0	0	Standard/Fast 12C Interface
0	1	HID OVER I2C
1	0	UART Interface
1	1	Reserved

Note: the PS1 and SP2 are default to be 0, 0.

API Functions

```
class DFRobot_BNO055 {

public:
  /**
   * @brief global axis declare (excepet eular and quaternion)
   */
  typedef enum {
    eAxisAcc,
    eAxisMag,
    eAxisGyr,
    eAxisLia,
    eAxisGrv
  } eAxis_t;

  /**
   * @brief global single axis declare
   */
  typedef enum {
    eSingleAxisX,
    eSingleAxisY,
    eSingleAxisZ
  } eSingleAxis_t;

  /**
   * @brief enum interrupt
   */
  typedef enum {
    eIntGyrAm = 0x04,
    eIntGyrHighRate = 0x08,
    eIntAccHighG = 0x20,
    eIntAccAm = 0x40,
    eIntAccNm = 0x80,
    eIntAll = 0xec
  } eInt_t;

  /**
   * @brief Operation mode enum
   */
  typedef enum {
    eOprModeConfig,
    eOprModeAccOnly,
    eOprModeMagOnly,
    eOprModeGyroOnly,
    eOprModeAccMag,
    eOprModeAccGyro,
    eOprModeMagGyro,
```

```
  eOprModeAMG,
  eOprModeImu,
  eOprModeCompass,
  eOprModeM4G,
  eOprModeNdofFmcOff,
  eOprModeNdof
} eOprMode_t;

/**
 * @brief Poewr mode enum
 */
typedef enum {
  ePowerModeNormal,
  ePowerModeLowPower,
  ePowerModeSuspend
} ePowerMode_t;

/**
 * @brief axis analog data struct
 */
typedef struct {
  float   x, y, z;
} sAxisAnalog_t;

/**
 * @brief eular analog data struct
 */
typedef struct {
  float   head, roll, pitch;
} sEulAnalog_t;

/**
 * @brief qua analog data struct
 */
typedef struct {
  float   w, x, y, z;
} sQuaAnalog_t;

/**
 * @brief enum accelerometer range, unit G
 */
typedef enum {
  eAccRange_2G,
  eAccRange_4G,
  eAccRange_8G,
  eAccRange_16G
} eAccRange_t;

/**
 * @brief enum accelerometer band width, unit HZ
 */
typedef enum {
```

```
  eAccBandWidth_7_81,    // 7.81HZ
  eAccBandWidth_15_63,   // 16.63HZ
  eAccBandWidth_31_25,
  eAccBandWidth_62_5,
  eAccBandWidth_125,
  eAccBandWidth_250,
  eAccBandWidth_500,
  eAccBandWidth_1000
} eAccBandWidth_t;

/**
 * @brief enum accelerometer power mode
 */
typedef enum {
  eAccPowerModeNormal,
  eAccPowerModeSuspend,
  eAccPowerModeLowPower1,
  eAccPowerModeStandby,
  eAccPowerModeLowPower2,
  eAccPowerModeDeepSuspend
} eAccPowerMode_t;

/**
 * @brief enum magnetometer data output rate, unit HZ
 */
typedef enum {
  eMagDataRate_2,
  eMagDataRate_6,
  eMagDataRate_8,
  eMagDataRate_10,
  eMagDataRate_15,
  eMagDataRate_20,
  eMagDataRate_25,
  eMagDataRate_30
} eMagDataRate_t;

/**
 * @brief enum magnetometer operation mode
 */
typedef enum {
  eMagOprModeLowPower,
  eMagOprModeRegular,
  eMagOprModeEnhancedRegular,
  eMagOprModeHighAccuracy
} eMagOprMode_t;

/**
 * @brief enum magnetometer power mode
 */
typedef enum {
  eMagPowerModeNormal,
  eMagPowerModeSleep,
```

```
  eMagPowerModeSuspend,
  eMagPowerModeForce
} eMagPowerMode_t;

/**
 * @brief enum gyroscope range, unit dps
 */
typedef enum {
  eGyrRange_2000,
  eGyrRange_1000,
  eGyrRange_500,
  eGyrRange_250,
  eGyrRange_125
} eGyrRange_t;

/**
 * @brief enum gyroscope band width, unit HZ
 */
typedef enum {
  eGyrBandWidth_523,
  eGyrBandWidth_230,
  eGyrBandWidth_116,
  eGyrBandWidth_47,
  eGyrBandWidth_23,
  eGyrBandWidth_12,
  eGyrBandWidth_64,
  eGyrBandWidth_32
} eGyrBandWidth_t;

/**
 * @brief enum gyroscope power mode
 */
typedef enum {
  eGyrPowerModeNormal,
  eGyrPowerModeFastPowerUp,
  eGyrPowerModeDeepSuspend,
  eGyrPowerModeSuspend,
  eGyrPowerModeAdvancedPowersave
} eGyrPowerMode_t;

/**
 * @brief Enum accelerometer interrupt settings
 */
typedef enum {
  eAccIntSetAmnmXAxis = (0x01 << 2),
  eAccIntSetAmnmYAxis = (0x01 << 3),
  eAccIntSetAmnmZAxis = (0x01 << 4),
  eAccIntSetHgXAxis = (0x01 << 5),
  eAccIntSetHgYAxis = (0x01 << 6),
  eAccIntSetHgZAxis = (0x01 << 7),
  eAccIntSetAll = 0xfc
} eAccIntSet_t;
```

```
/**
 * @brief Enum accelerometer slow motion mode or no motion mode
 */
typedef enum {
  eAccNmSmnmSm,  // slow motion mode
  eAccNmSmnmNm   // no motion mode
} eAccNmSmnm_t;

/**
 * @brief Enum gyroscope interrupt settings
 */
typedef enum {
  eGyrIntSetAmXAxis = (0x01 << 0),
  eGyrIntSetAmYAxis = (0x01 << 1),
  eGyrIntSetAmZAxis = (0x01 << 2),
  eGyrIntSetHrXAxis = (0x01 << 3),
  eGyrIntSetHrYAxis = (0x01 << 4),
  eGyrIntSetHrZAxis = (0x01 << 5),
  eGyrIntSetAmFilt = (0x01 << 6),
  eGyrIntSetHrFilt = (0x01 << 7),
  eGyrIntSetAll = 0x3f
} eGyrIntSet_t;

/**
 * @brief Declare sensor status
 */
typedef enum {
  eStatusOK,    // everything OK
  eStatusErr,   // unknow error
  eStatusErrDeviceNotDetect,    // device not detected
  eStatusErrDeviceReadyTimeOut, // device ready time out
  eStatusErrDeviceStatus,       // device internal status error
  eStatusErrParameter           // function parameter error
} eStatus_t;

/**
 * @brief begin Sensor begin
 * @return Sensor status
 */
eStatus_t   begin();

/**
 * @brief getAxisAnalog Get axis analog data
 * @param eAxis One axis type from eAxis_t
 * @return Struct sAxisAnalog_t, contains axis analog data, members unit d
 *                case eAxisAcc, unit mg
 *                case eAxisLia, unit mg
 *                case eAxisGrv, unit mg
 *                case eAxisMag, unit ut
 *                case eAxisGyr, unit dps
 */
```

```
sAxisAnalog_t getAxis(eAxis_t eAxis);

/**
 * @brief getEulAnalog Get euler analog data
 * @return Struct sEulAnalog_t, contains euler analog data
 */
sEulAnalog_t  getEul();

/**
 * @brief getQuaAnalog Get quaternion analog data
 * @return Struct sQuaAnalog_t, contains quaternion analog data
 */
sQuaAnalog_t  getQua();

/**
 * @brief setAccOffset Set axis offset data
 * @param eAxis One axis type from eAxis_t, only support accelerometer, ma
 * @param sOffset Struct sAxisAnalog_t, contains axis analog data, members
 *                case eAxisAcc, unit mg, members can't out of acc range
 *                case eAxisMag, unit ut, members can't out of mag range
 *                case eAxisGyr, unit dps, members can't out of gyr range
 */
void    setAxisOffset(eAxis_t eAxis, sAxisAnalog_t sOffset);

/**
 * @brief setOprMode Set operation mode
 * @param eOpr One operation mode from eOprMode_t
 */
void    setOprMode(eOprMode_t eMode);

/**
 * @brief setPowerMode Set power mode
 * @param eMode One power mode from ePowerMode_t
 */
void    setPowerMode(ePowerMode_t eMode);

/**
 * @brief Reset sensor
 */
void    reset();

/**
 * @brief setAccRange Set accelerometer measurement range, default value i
 * @param eRange One range enum from eAccRange_t
 */
void    setAccRange(eAccRange_t eRange);

/**
 * @brief setAccBandWidth Set accelerometer band width, default value is 6
 * @param eBand One band enum from eAccBandWidth_t
 */
void    setAccBandWidth(eAccBandWidth_t eBand);
```

```
/**
 * @brief setAccPowerMode Set accelerometer power mode, default value is e
 * @param eMode One mode enum from eAccPowerMode_t
 */
void    setAccPowerMode(eAccPowerMode_t eMode);

/**
 * @brief setMagDataRate Set magnetometer data output rate, default value
 * @param eRate One rate enum from eMagDataRate_t
 */
void    setMagDataRate(eMagDataRate_t eRate);

/**
 * @brief setMagOprMode Set magnetometer operation mode, default value is
 * @param eMode One mode enum from eMagOprMode_t
 */
void    setMagOprMode(eMagOprMode_t eMode);

/**
 * @brief setMagPowerMode Set magnetometer power mode, default value is eM
 * @param eMode One mode enum from eMagPowerMode_t
 */
void    setMagPowerMode(eMagPowerMode_t eMode);

/**
 * @brief setGyrRange Set gyroscope range, default value is 2000
 * @param eRange One range enum from eGyrRange_t
 */
void    setGyrRange(eGyrRange_t eRange);

/**
 * @brief setGyrBandWidth Set gyroscope band width, default value is 32HZ
 * @param eBandWidth One band width enum from eGyrBandWidth_t
 */
void    setGyrBandWidth(eGyrBandWidth_t eBandWidth);

/**
 * @brief setGyrPowerMode Set gyroscope power mode, default value is eGyrP
 * @param eMode One power mode enum from eGyrPowerMode_t
 */
void    setGyrPowerMode(eGyrPowerMode_t eMode);

/**
 * @brief getIntState Get interrupt state, interrupt auto clear after read
 * @return If result > 0, at least one interrupt triggered. Result & eIntX
 */
uint8_t   getIntState();

/**
 * @brief setIntMask Set interrupt mask enable, there will generate a inte
 * @param eInt One or more interrupt flags to set, input them through oper
```

```
 */
void    setIntMaskEnable(eInt_t eInt);

/**
 * @brief setIntMaskDisable Set corresponding interrupt mask disable
 * @param eInt One or more interrupt flags to set, input them through oper
 */
void    setIntMaskDisable(eInt_t eInt);

/**
 * @brief setIntEnEnable Set corresponding interrupt enable
 * @param eInt One or more interrupt flags to set, input them through oper
 */
void    setIntEnable(eInt_t eInt);

/**
 * @brief setIntEnDisable Set corresponding interrupt disable
 * @param eInt One or more interrupt flags to set, input them through oper
 */
void    setIntDisable(eInt_t eInt);

/**
 * @brief setAccAmThres Set accelerometer any motion threshold
 * @param thres Threshold to set, unit mg, value is dependent on accelerom
 *        case 2g, no more than 1991
 *        case 4g, no more than 3985
 *        case 8g, no more than 7968
 *        case 16g, no more than 15937
 *        Attenion: The set value will be slightly biased according to dat
 */
void    setAccAmThres(uint16_t thres);

/**
 * @brief setAccIntDur Set accelerometer interrupt duration,
 *        any motion interrupt triggers if duration (dur + 1) consecutive
 *        threshold define in any motion threshold
 * @param dur Duration to set, range form 1 to 4
 */
void    setAccIntAmDur(uint8_t dur);

/**
 * @brief setAccIntEnable Set accelerometer interrupt enable
 * @param eInt One or more interrupt flags to set, input them through oper
 */
void    setAccIntEnable(eAccIntSet_t eInt);

/**
 * @brief setAccIntDisable Set accelerometer interrupt disable
 * @param eInt One or more interrupt flags to set, input them through oper
 */
void    setAccIntDisable(eAccIntSet_t eInt);
```

```
/**
 * @brief setAccHighGDuration Set accelerometer high-g interrupt, the high
 * @param dur Duration from 2ms to 512ms
 */
void    setAccHighGDuration(uint16_t dur);

/**
 * @brief setAccHighGThres Set accelerometer high-g threshold
 * @param thres Threshold to set, unit mg, value is dependent on accelerom
 *        case 2g, no more than 1991
 *        case 4g, no more than 3985
 *        case 8g, no more than 7968
 *        case 16g, no more than 15937
 *        Attenion: The set value will be slightly biased according to dat
 */
void    setAccHighGThres(uint16_t thres);

/**
 * @brief setAccNmThres Set accelerometer no motion threshold
 * @param thres Threshold to set, unit mg, value is dependent on accelerom
 *        case 2g, no more than 1991
 *        case 4g, no more than 3985
 *        case 8g, no more than 7968
 *        case 16g, no more than 15937
 *        Attenion: The set value will be slightly biased according to dat
 */
void    setAccNmThres(uint16_t thres);

/**
 * @brief setAccNmSet Set accelerometer slow motion or no motion mode and
 * @param eSmnm Enum of eAccNmSmnm_t
 * @param dur Interrupt trigger delay (unit seconds), no more than 344.
 *            Attenion: The set value will be slightly biased according to
 */
void    setAccNmSet(eAccNmSmnm_t eSmnm, uint16_t dur);

/**
 * @brief setGyrIntEnable Set corresponding gyroscope interrupt enable
 * @param eInt One or more interrupt flags to set, input them through oper
 */
void    setGyrIntEnable(eGyrIntSet_t eInt);

/**
 * @brief setGyrIntDisable Set corresponding gyroscope interrupt disable
 * @param eInt One or more interrupt flags to set, input them through oper
 */
void    setGyrIntDisable(eGyrIntSet_t eInt);

/**
 * @brief setGyrHrSet Set gyroscope high rate settings
 * @param eSingleAxis Single axis to set
 * @param thres High rate threshold to set, unit degree/seconds, value is
```

```
    *        case 2000, no more than 1937
    *        case 1000, no more than 968
    *        case 500, no more than 484
    *        case 250, no more than 242
    *        case 125, no more than 121
    *        Attenion: The set value will be slightly biased according to dat
    * @param dur High rate duration to set, unit ms, duration from 2.5ms to 6
    *            Attenion: The set value will be slightly biased according to
    */
  void    setGyrHrSet(eSingleAxis_t eSingleAxis, uint16_t thres, uint16_t du

  /**
   * @brief setGyrAmThres Set gyroscope any motion threshold
   * @param thres Threshold to set, unit mg, value is dependent on accelerom
   *        case 2000, no more than 128
   *        case 1000, no more than 64
   *        case 500, no more than 32
   *        case 250, no more than 16
   *        case 125, no more than 8
   *        Attenion: The set value will be slightly biased according to dat
   */
  void    setGyrAmThres(uint8_t thres);

  /**
   * @brief lastOpreateStatus Show last operate status
   */
  eStatus_t   lastOpreateStatus;
};

class DFRobot_BNO055_IIC : public DFRobot_BNO055 {
public:

  /**
   * @brief DFRobot_BNO055_IIC class constructor
   * @param pWire select One TwoWire peripheral
   * @param addr Sensor address
   */
  DFRobot_BNO055_IIC(TwoWire *pWire, uint8_t addr);
};
```

Tutorial

Visit the I2C address of BNO055 via I2C interface to get the related position data.

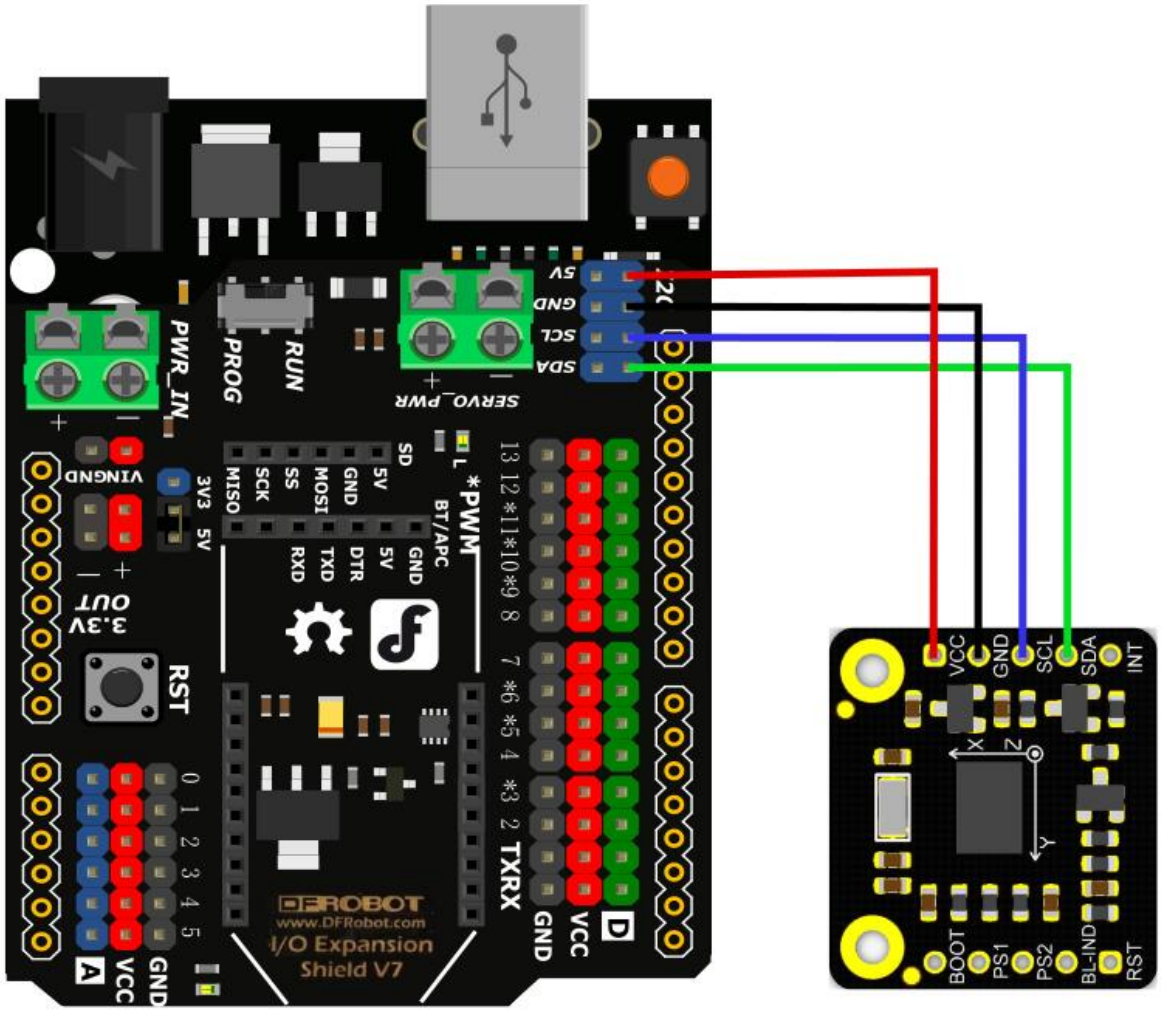
Requirements

- **Hardware**
  - DFRduino UNO R3 (https://www.dfrobot.com/product-838.html) (or similar) x 1
  - BNO055 Intelligent 9-axis Sensor Module x1

- ~~BNO055 Intelligent 9-axis Sensor Module x1~~
- Jumper wires

- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)
  - Download and install the **BMX160 Library** (https://github.com/DFRobot/DFRobot_BNO055) (About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))

Connection Diagram



BMX160 Tutorial

Function: read the pitch angle, roll angle and yaw angle of BNO055 sensor via I2C interface, and print out the data through the serial port. Using this demo with a small visual software Euler angle visual tool.exe (https://github.com/DFRobot/Wiki/raw/master/Resource/Euler%20angle%20visual%20tool%5BV1.0%5D.rar) we specifically designed, you can directly observe the attitude variation of 10DOF. As shown below.

```
/*!
  * imu_show.ino
  *
  * Download this demo to show attitude on [imu_show](https://github.com/DFR
  * Attitude will show on imu_show
  *
  * Product: http://www.dfrobot.com.cn/goods-1860.html
  * Copyright   [DFRobot](http://www.dfrobot.com), 2016
  * Copyright   GNU Lesser General Public License
  *
  * version  V1.0
  * date  07/03/2019
  */

#include "DFRobot_BNO055.h"
#include "Wire.h"

typedef DFRobot_BNO055_IIC    BNO;    // ******** use abbreviations instead

BNO   bno(&Wire, 0x28);    // input TwoWire interface and IIC address

// show last sensor operate status
void printLastOperateStatus(BNO::eStatus_t eStatus)
{
  switch(eStatus) {
  case BNO::eStatusOK:    Serial.println("everything ok"); break;
  case BNO::eStatusErr:   Serial.println("unknow error"); break;
  case BNO::eStatusErrDeviceNotDetect:    Serial.println("device not detecte
  case BNO::eStatusErrDeviceReadyTimeOut: Serial.println("device ready time
  case BNO::eStatusErrDeviceStatus:       Serial.println("device internal st
  default: Serial.println("unknow status"); break;
  }
}

void setup()
{
  Serial.begin(115200);
  bno.reset();
  while(bno.begin() != BNO::eStatusOK) {
    Serial.println("bno begin faild");
    printLastOperateStatus(bno.lastOperateStatus);
    delay(2000);
  }
  Serial.println("bno begin success");
}
```

```
void loop()
{
  BNO::sEulAnalog_t   sEul;
  sEul = bno.getEul();
  Serial.print("pitch:");
  Serial.print(sEul.pitch, 3);
  Serial.print(" ");
  Serial.print("roll:");
  Serial.print(sEul.roll, 3);
  Serial.print(" ");
  Serial.print("yaw:");
  Serial.print(sEul.head, 3);
  Serial.println(" ");
  delay(80);
}
```

If we compare 10DOF to an airplane whose nose points at due east, the positive direction of X axis will be the direction of the nose, the positive direction of Y axis will be the direction of the left wing, which is due north. Z axis is perpendicular to the plane XOY that formed by X and Y axes. When the 10 DOF's direction of X, Y, and Z totally coincides with the above-mentioned direction, the values of the pitch, roll and yaw angle are 0°. Here we define: pitch is the angle between the nose and XOY when the airplane noses up or down along the Y axis, and nose up is positive while nose down is negative; roll is the angle between the body and XOY when the airplane rolls along the X axis; yaw is the angle between the nose and XOZ when the airplane moves along the Z axis.

⚠ **Please note that you need to close the serial port occupied by the printer when using the test software to observe the sensor's movement posture.**

Sample Code

Function: get the acceleration data of the sensor's movement on X, Z and Y, and print it out through the serial port.

```
/*!
  * read_data.ino
  *
  * Download this demo to test read data from bno055
  * Data will print on your serial monitor
  *
  * Product: http://www.dfrobot.com.cn/goods-1860.html
  * Copyright   [DFRobot](http://www.dfrobot.com), 2016
  * Copyright   GNU Lesser General Public License
  *
  * version  V1.0
  * date  07/03/2019
  */

#include "DFRobot_BNO055.h"
#include "Wire.h"

typedef DFRobot_BNO055_IIC    BNO;    // ******** use abbreviations instead

BNO   bno(&Wire, 0x28);    // input TwoWire interface and IIC address

// show last sensor operate status
void printLastOperateStatus(BNO::eStatus_t eStatus)
{
  switch(eStatus) {
  case BNO::eStatusOK:   Serial.println("everything ok"); break;
  case BNO::eStatusErr:  Serial.println("unknow error"); break;
  case BNO::eStatusErrDeviceNotDetect:   Serial.println("device not detected
  case BNO::eStatusErrDeviceReadyTimeOut:    Serial.println("device ready ti
  case BNO::eStatusErrDeviceStatus:    Serial.println("device internal statu
  default: Serial.println("unknow status"); break;
  }
}

void setup()
{
  Serial.begin(115200);
  bno.reset();
  while(bno.begin() != BNO::eStatusOK) {
    Serial.println("bno begin faild");
    printLastOperateStatus(bno.lastOperateStatus);
    delay(2000);
  }
  Serial.println("bno begin success");
}
```

```
#define printAxisData(sAxis) \
  Serial.print(" x: "); \
  Serial.print(sAxis.x); \
  Serial.print(" y: "); \
  Serial.print(sAxis.y); \
  Serial.print(" z: "); \
  Serial.println(sAxis.z)

void loop()
{
  BNO::sAxisAnalog_t   sAccAnalog, sMagAnalog, sGyrAnalog, sLiaAnalog, sGrvA
  BNO::sEulAnalog_t    sEulAnalog;
  BNO::sQuaAnalog_t    sQuaAnalog;
  sAccAnalog = bno.getAxis(BNO::eAxisAcc);    // read acceleration
  sMagAnalog = bno.getAxis(BNO::eAxisMag);    // read geomagnetic
  sGyrAnalog = bno.getAxis(BNO::eAxisGyr);    // read gyroscope
  sLiaAnalog = bno.getAxis(BNO::eAxisLia);    // read linear acceleration
  sGrvAnalog = bno.getAxis(BNO::eAxisGrv);    // read gravity vector
  sEulAnalog = bno.getEul();                  // read euler angle
  sQuaAnalog = bno.getQua();                  // read quaternion
  Serial.println();
  Serial.println("======== analog data print start ========");
  Serial.print("acc analog: (unit mg)       "); printAxisData(sAccAnalog);
  Serial.print("mag analog: (unit ut)       "); printAxisData(sMagAnalog);
  Serial.print("gyr analog: (unit dps)      "); printAxisData(sGyrAnalog);
  Serial.print("lia analog: (unit mg)       "); printAxisData(sLiaAnalog);
  Serial.print("grv analog: (unit mg)       "); printAxisData(sGrvAnalog);
  Serial.print("eul analog: (unit degree)   "); Serial.print(" head: "); Ser
  Serial.print("qua analog: (no unit)       "); Serial.print(" w: "); Serial
  Serial.println("========  analog data print end  ========");

  delay(1000);
}
```

Sample Code

Function: moitor the sensor interrupts, including high/low speed interrupt, and fast tilt interrupt.

```
/*!
  * interrupt.ino
  *
  * Download this demo to test bno055 interrupt
  * Connect bno055 int pin to arduino pin 2
  * If there occurs interrupt, it will printr on you serial monitor, more de
  *
  * Product: http://www.dfrobot.com.cn/goods-1860.html
  * Copyright   [DFRobot](http://www.dfrobot.com), 2016
  * Copyright   GNU Lesser General Public License
  *
  * version  V1.0
  * date  07/03/2019
  */

#include "DFRobot_BNO055.h"
#include "Wire.h"

typedef DFRobot_BNO055_IIC    BNO;    // ******** use abbreviations instead

BNO   bno(&Wire, 0x28);    // input TwoWire interface and IIC address

// show last sensor operate status
void printLastOperateStatus(BNO::eStatus_t eStatus)
{
  switch(eStatus) {
  case BNO::eStatusOK:    Serial.println("everything ok"); break;
  case BNO::eStatusErr:   Serial.println("unknow error"); break;
  case BNO::eStatusErrDeviceNotDetect:    Serial.println("device not detecte
  case BNO::eStatusErrDeviceReadyTimeOut: Serial.println("device ready time
  case BNO::eStatusErrDeviceStatus:       Serial.println("device internal st
  default: Serial.println("unknow status"); break;
  }
}

bool intFlag = false;

void intHandle()
{
  intFlag = true;
}

void setup()
{
  Serial.begin(115200);
  bno.reset();
```

```
while(bno.begin() != BNO::eStatusOK) {
  Serial.println("bno begin faild");
  printLastOperateStatus(bno.lastOperateStatus);
  delay(2000);
}
Serial.println("bno begin success");

bno.setOprMode(BNO::eOprModeConfig);    // set to config mode

bno.setIntMaskEnable(BNO::eIntAll);    // set interrupt mask enable, signa
// bno.setIntMaskDisable(BNO::eIntAccAm | BNO::eIntAccNm);    // set inter

bno.setIntEnable(BNO::eIntAll);   // set interrupt enable
// bno.setIntDisable(BNO::eIntAccAm | BNO::eIntAccNm);    // set interrupt

bno.setAccIntEnable(BNO::eAccIntSetAll);    // set accelerometer interrupt
// bno.setAccIntDisable(BNO::eAccIntSetAmnmXAxis | BNO::eAccIntSetHgXAxis)

/* accelerometer any motion threshold to set, unit mg, value is dependent
 * case 2g, no more than 1991
 * case 4g, no more than 3985
 * case 8g, no more than 7968
 * case 16g, no more than 15937
 * attenion: The set value will be slightly biased according to datasheet
 * tips: default accelerometer range is 4g
 */
// how to trig this: still --> fast move
bno.setAccAmThres(200);
// any motion interrupt triggers if duration consecutive data points are a
// threshold define in any motion threshold
bno.setAccIntAmDur(1);
// set high-g duration, value from 2ms to 512ms
bno.setAccHighGDuration(80);
/*
 * accelerometer high-g threshold to set, unit mg, value is dependent on a
 * case 2g, no more than 1991
 * case 4g, no more than 3985
 * case 8g, no more than 7968
 * case 16g, no more than 15937
 * Attenion: The set value will be slightly biased according to datasheet
 */
// how to trig this: still --> (very) fast move
bno.setAccHighGThres(900);
// accelerometer (no motion) / (slow motion) settings, 2nd parameter unit
bno.setAccNmSet(BNO::eAccNmSmnmNm, 4);
/*
 * accelerometer no motion threshold to set, unit mg, value is dependent o
 * case 2g, no more than 1991
 * case 4g, no more than 3985
 * case 8g, no more than 7968
 * case 16g, no more than 15937
 * Attenion: The set value will be slightly biased according to datasheet
```

```
   */
  // hot to trig this: any motion --> still --> still
  bno.setAccNmThres(100);

  bno.setGyrIntEnable((BNO::eGyrIntSet_t) (BNO::eGyrIntSetHrXAxis | BNO::eGy
  // bno.setGyrIntEnable(BNO::eGyrIntSetAmYAxis | BNO::eGyrIntSetAmYAxis | B
  // bno.setGyrIntDisable(BNO::eGyrIntSetHrXAxis | BNO::eGyrIntSetAmXAxis);

  /*
   * 2nd parameter, high rate threshold to set, unit degree/seconds, value i
   * case 2000, no more than 1937
   * case 1000, no more than 968
   * case 500, no more than 484
   * case 250, no more than 242
   * case 125, no more than 121
   * Attenion: The set value will be slightly biased according to datasheet
   * 3rd parameter, high rate duration to set, unit ms, duration from 2.5ms
   * Attenion: The set value will be slightly biased according to datasheet
   */
  // how to trigger this: still --> fast tilt
  bno.setGyrHrSet(BNO::eSingleAxisX, 300, 80);
  bno.setGyrHrSet(BNO::eSingleAxisY, 300, 80);
  bno.setGyrHrSet(BNO::eSingleAxisZ, 300, 80);
  /*
   * gyroscope any motion threshold to set, unit mg, value is dependent on a
   * case 2000, no more than 128
   * case 1000, no more than 64
   * case 500, no more than 32
   * case 250, no more than 16
   * case 125, no more than 8
   * Attenion: The set value will be slightly biased according to datasheet
   * tips: default range is 2000
   */
  // how to trigger this: still --> fast tilt
  bno.setGyrAmThres(20);

  bno.setOprMode(BNO::eOprModeNdof);    // configure done

  attachInterrupt(0, intHandle, RISING);   // attach interrupt
  bno.getIntState();    // clear unexpected interrupt
  intFlag = false;
}

void loop()
{
  if(intFlag) {
    intFlag = false;
    uint8_t   intSta = bno.getIntState();   // interrupt auto clear after re

    Serial.println("interrupt detected");
    if(intSta & BNO::eIntAccAm)
      Serial.println("accelerometer any motion detected");
```

```
    if(intSta & BNO::eIntAccNm)
      Serial.println("accelerometer no motion detected");
    if(intSta & BNO::eIntAccHighG)
      Serial.println("acceleromter high-g detected");
    if(intSta & BNO::eIntGyrHighRate)
      Serial.println("gyroscope high rate detected");
    if(intSta & BNO::eIntGyrAm)
      Serial.println("gyroscope any motion detected");
  }
}
```

Sample Code

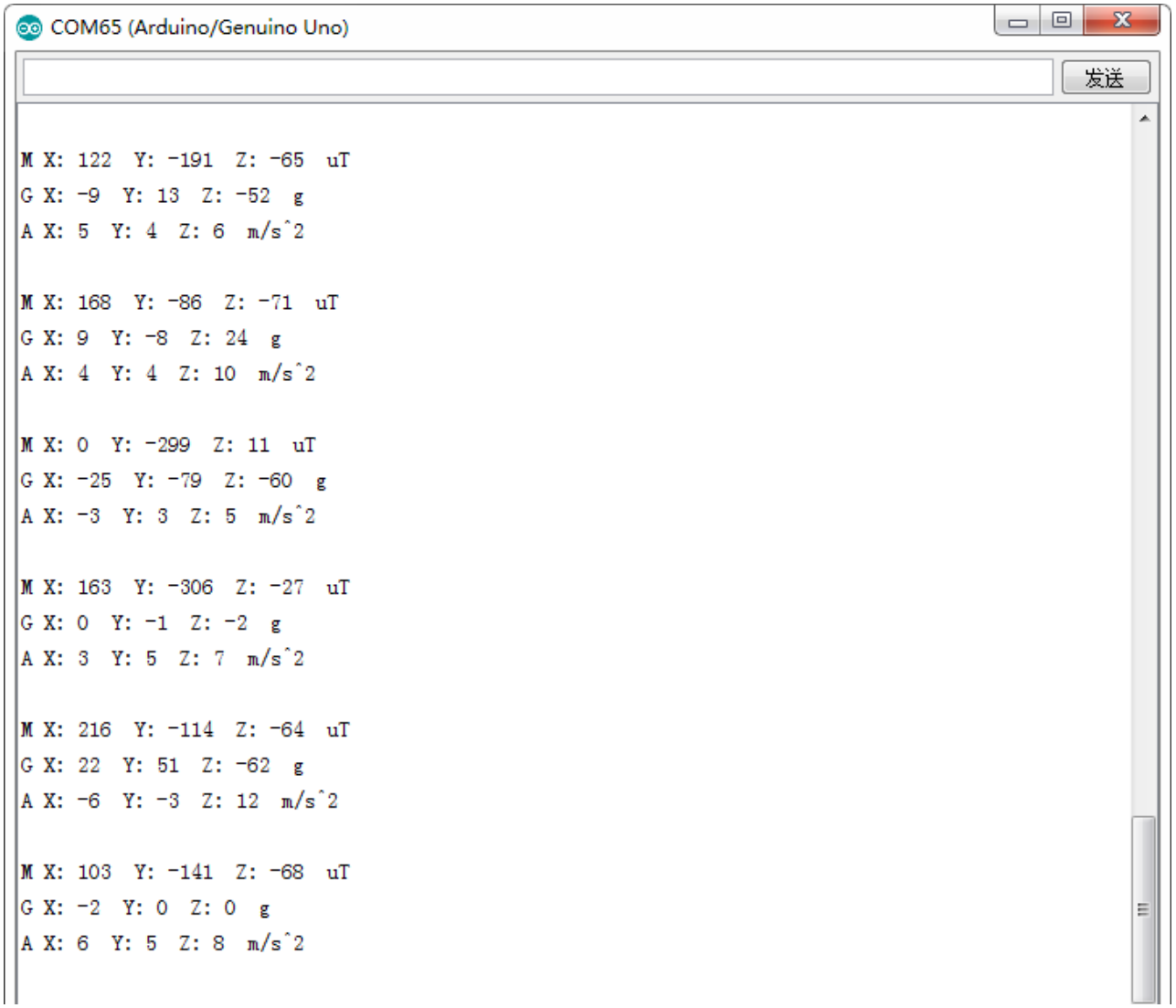
Function: sensor configuration.

```
/*!
  * config.ino
  *
  * Download this demo to test config to bno055
  * Data will print on your serial monitor
  *
  * Product: http://www.dfrobot.com.cn/goods-1860.html
  * Copyright   [DFRobot](http://www.dfrobot.com), 2016
  * Copyright   GNU Lesser General Public License
  *
  * version  V1.0
  * date  07/03/2019
  */

#include "DFRobot_BNO055.h"
#include "Wire.h"

typedef DFRobot_BNO055_IIC    BNO;    // ******** use abbreviations instead

BNO   bno(&Wire, 0x28);    // input TwoWire interface and IIC address

// show last sensor operate status
void printLastOperateStatus(BNO::eStatus_t eStatus)
{
  switch(eStatus) {
  case BNO::eStatusOK:    Serial.println("everything ok"); break;
  case BNO::eStatusErr:   Serial.println("unknow error"); break;
  case BNO::eStatusErrDeviceNotDetect:    Serial.println("device not detecte
  case BNO::eStatusErrDeviceReadyTimeOut: Serial.println("device ready time
  case BNO::eStatusErrDeviceStatus:       Serial.println("device internal st
  default: Serial.println("unknow status"); break;
  }
}

void setup()
{
  Serial.begin(115200);
  bno.reset();
  while(bno.begin() != BNO::eStatusOK) {
    Serial.println("bno begin faild");
    printLastOperateStatus(bno.lastOperateStatus);
    delay(2000);
  }
  Serial.println("bno begin success");

  bno.setPowerMode(BNO::ePowerModeNormal);    // set to normal power mode
```

```
  bno.setOprMode(BNO::eOprModeConfig);    // must set sensor to config-mode
  bno.setAccPowerMode(BNO::eAccPowerModeNormal);    // set acc to normal pow
  bno.setGyrPowerMode(BNO::eGyrPowerModeNormal);    // set gyr to normal pow
  bno.setMagPowerMode(BNO::eMagPowerModeForce);     // set mag to force powe

  // accelerometer normal configure
  bno.setAccRange(BNO::eAccRange_4G);   // set range to 4g
  bno.setAccBandWidth(BNO::eAccBandWidth_62_5);   // set band width 62.5HZ
  bno.setAccPowerMode(BNO::eAccPowerModeNormal);  // set accelerometer power

  // magnetometer normal configure
  bno.setMagDataRate(BNO::eMagDataRate_20);   // set output data rate 20HZ
  bno.setMagPowerMode(BNO::eMagPowerModeForce);   // set power mode
  bno.setMagOprMode(BNO::eMagOprModeRegular); // set operate mode

  // gyroscope normal configure
  bno.setGyrRange(BNO::eGyrRange_2000);   // set range
  bno.setGyrBandWidth(BNO::eGyrBandWidth_32);   // set band width
  bno.setGyrPowerMode(BNO::eGyrPowerModeNormal);    // set power mode

  BNO::sAxisAnalog_t    sOffsetAcc;   // unit mg, members can't out of acc r
  BNO::sAxisAnalog_t    sOffsetMag;   // unit ut, members can't out of mag r
  BNO::sAxisAnalog_t    sOffsetGyr;   // unit dps, members can't out of gyr
  sOffsetAcc.x = 1;
  sOffsetAcc.y = 1;
  sOffsetAcc.z = 1;
  sOffsetMag.x = 1;
  sOffsetMag.y = 1;
  sOffsetMag.z = 1;
  sOffsetGyr.x = 1;
  sOffsetGyr.y = 1;
  sOffsetGyr.z = 1;
  bno.setAxisOffset(BNO::eAxisAcc, sOffsetAcc);   // set offset
  bno.setAxisOffset(BNO::eAxisMag, sOffsetMag);
  bno.setAxisOffset(BNO::eAxisGyr, sOffsetGyr);

  bno.setOprMode(BNO::eOprModeNdof);   // shift to other operate mode, refer
}

#define printAxisData(sAxis) \
  Serial.print(" x: "); \
  Serial.print(sAxis.x); \
  Serial.print(" y: "); \
  Serial.print(sAxis.y); \
  Serial.print(" z: "); \
  Serial.println(sAxis.z)

void loop()
{
  BNO::sAxisAnalog_t   sAccAnalog, sMagAnalog, sGyrAnalog, sLiaAnalog, sGrvA
  BNO::sEulAnalog_t    sEulAnalog;
  BNO::sQuaAnalog_t    sQuaAnalog;
```

```
    sAccAnalog = bno.getAxis(BNO::eAxisAcc);
    sMagAnalog = bno.getAxis(BNO::eAxisMag);
    sGyrAnalog = bno.getAxis(BNO::eAxisGyr);
    sLiaAnalog = bno.getAxis(BNO::eAxisLia);
    sGrvAnalog = bno.getAxis(BNO::eAxisGrv);
    sEulAnalog = bno.getEul();
    sQuaAnalog = bno.getQua();
    Serial.println();
    Serial.println("======== analog data print start ========");
    Serial.print("acc analog: (unit mg)       "); printAxisData(sAccAnalog);
    Serial.print("mag analog: (unit ut)       "); printAxisData(sMagAnalog);
    Serial.print("gyr analog: (unit dps)      "); printAxisData(sGyrAnalog);
    Serial.print("lia analog: (unit mg)       "); printAxisData(sLiaAnalog);
    Serial.print("grv analog: (unit mg)       "); printAxisData(sGrvAnalog);
    Serial.print("eul analog: (unit degree)   "); Serial.print(" head: "); Ser
    Serial.print("qua analog: (no unit)       "); Serial.print(" w: "); Serial
    Serial.println("========  analog data print end  ========");

    delay(1000);
}
```

Expected Results



FAQ

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

More Documents

- SEN0374 Schematics.pdf (https://dfimg.dfrobot.com/nobody/wiki/bee5425b1af453621180d84de73ba298.pdf)
- BNO055 Datasheet (https://github.com/Strictus/DFRobot/raw/master/SEN0253/BST_BNO055_DS000_14.pdf)