**Lesson 47**

**Part 1**

**Connect the gyroscope LSM6DS3**

Today we will continue to work with one more sensor, which combines two functionalities - the accelerometer and the gyroscope - **LSM6DS3**. It is also implemented using MEMS technology. Installed on the expansion board **STEVAL-MKI160V1**, which in turn is inserted into the expansion board **X-NUCLEO-IKS01A1**, designed to work with the debug card Nucleo. We will connect this evaluation board to the Nucleo STM32F401RE board.

This accelerometer-gyroscope also can, along with the I2C interface, connect using the SPI interface. But we will use the connection specifically for I2C, since it is this connection that takes place in the evaluation board X-NUCLEO-IKS01A.

Also, we will use this sensor as a gyro in this lesson, since we already connected it as an accelerometer.

This gyro in this sensor has the following technical characteristics in comparison with the previous one:

The reading range is ± 125 / ± 245 / ± 500 / ± 2000 dps (another smallest limit appeared);

The sensitivity of 4.375 - 70 mdps / LSb (respectively, the lower limit became better);

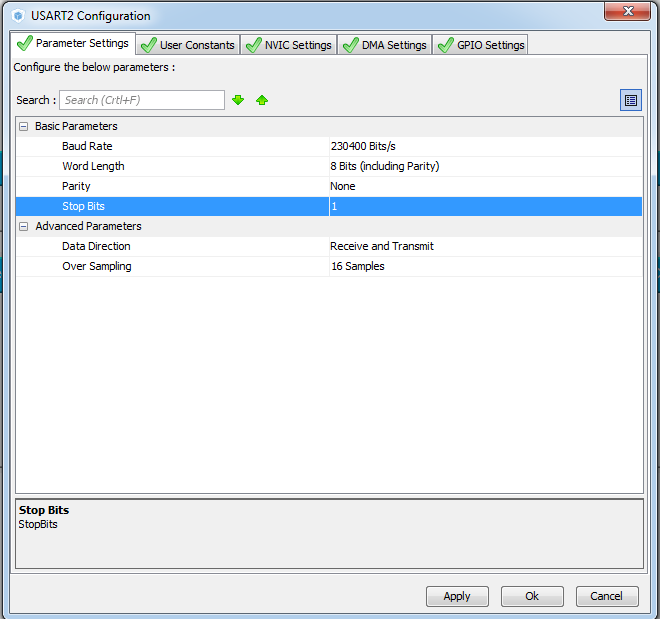
The deviation from zero is ± 10 dps when setting the range to 2000 dps (which is three times better than the sensor considered earlier).

The measurement frequency is 14.9 - 952 Hz.

With some other indicators, registers, values ​​and other details of the gyroscope, we will get acquainted in the course of its programming.

We will create the project from the finished project, in which we worked with the accelerometer of this sensor - from the project Accel\_LSM6DS3, only we call this project now, respectively, Gyro\_LSM6DS3.

Run the Cube MX project. We will change here only the speed of USART.



Generate the project, open it. Let's set up the programmer for auto-cutting. Add the file lsm6ds3.c. We will compile the project.

In an infinite loop, while commenting out the code for calling the data reading function and sending them to the USART

  / \* USER CODE BEGIN 3 \* /

                // Accel\_ReadAcc ();

  }

For the universality of the project, since, perhaps later, we will combine work with the accelerometer and gyroscope into one project, rename the function Accel\_Ini in the file lsm6ds0.c to Accel\_Gyro\_Ini. The same will be done with the prototype and with the call of this function in main ().

Comment in the function Accel\_Gyro\_Ini here this line

        LD2\_OFF;

**// AccInit (ctrl);**

        LD2\_ON;

Add the initialization function of the gyroscope by the similarity of the accelerometer initialization function

// ---------------

**void GyroInit (uint16\_t InitStruct)**

**{**

**uint8\_t value = 0;**

**}**

// ---------------

Call it in the function of general initialization

// AccInit (ctrl);

**GyroInit (ctrl);**

        LD2\_ON;

In the file lsm6ds0.h we will add in advance several macros necessary for working with the gyro. The code for this file will look like this after all the changes:

#ifndef LSM6DS3\_H\_

#define LSM6DS3\_H\_

#include "stm32f4xx\_hal.h"

#include <string.h>

// ----------------

#define ABS (x) (x <0)? (-x): x

// ----------------

#define LD2\_Pin GPIO\_PIN\_5

#define LD2\_GPIO\_Port GPIOA

#define LD2\_ON HAL\_GPIO\_WritePin (GPIOA, GPIO\_PIN\_5, GPIO\_PIN\_SET) // GREEN

#define LD2\_OFF HAL\_GPIO\_WritePin (GPIOA, GPIO\_PIN\_5, GPIO\_PIN\_RESET)

// ----------------

#define LSM6DS3\_ACC\_GYRO\_CTRL1\_XL 0X10

#define LSM6DS3\_ACC\_GYRO\_CTRL2\_G 0X11

#define LSM6DS3\_ACC\_GYRO\_CTRL3\_C 0X12

#define LSM6DS3\_ACC\_GYRO\_FIFO\_CTRL5 0X0A

#define LSM6DS3\_ACC\_GYRO\_CTRL9\_XL 0X18

#define LSM6DS3\_ACC\_GYRO\_CTRL10\_C 0X19

// ----------------

#define LSM6DS3\_ACC\_GYRO\_IF\_INC\_DISABLED 0x00

#define LSM6DS3\_ACC\_GYRO\_IF\_INC\_ENABLED 0x04

#define LSM6DS3\_ACC\_GYRO\_IF\_INC\_MASK 0x04

// ----------------

#define LSM6DS3\_ACC\_GYRO\_BDU\_CONTINUOS 0x00

#define LSM6DS3\_ACC\_GYRO\_BDU\_BLOCK\_UPDATE 0x40

#define LSM6DS3\_ACC\_GYRO\_BDU\_MASK 0x40

// ----------------

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_BYPASS 0x00

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_FIFO 0x01

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_STREAM 0x02

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_STF 0x03

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_BTS 0x04

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_DYN\_STREAM 0x05

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_DYN\_STREAM\_2 0x06

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_BTF 0x07

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_MASK 0x07

// ----------------

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_POWER\_DOWN 0x00

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_13Hz 0x10

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_26Hz 0x20

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_52Hz 0x30

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_104Hz 0x40

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_208Hz 0x50

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_416Hz 0x60

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_833Hz 0x70

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_1660Hz 0x80

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_3330Hz 0x90

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_6660Hz 0xA0

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_13330Hz 0xB0

#define LSM6DS3\_ACC\_GYRO\_ODR\_XL\_MASK 0xF0

// ----------------

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_POWER\_DOWN 0x00

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_13Hz 0x10

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_26Hz 0x20

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_52Hz 0x30

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_104Hz 0x40

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_208Hz 0x50

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_416Hz 0x60

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_833Hz 0x70

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_1660Hz 0x80

#define LSM6DS3\_ACC\_GYRO\_ODR\_G\_MASK 0xF0

// ----------------

#define LSM6DS3\_ACC\_GYRO\_FS\_XL\_2g 0x00

#define LSM6DS3\_ACC\_GYRO\_FS\_XL\_16g 0x04

#define LSM6DS3\_ACC\_GYRO\_FS\_XL\_4g 0x08

#define LSM6DS3\_ACC\_GYRO\_FS\_XL\_8g 0x0C

#define LSM6DS3\_ACC\_GYRO\_FS\_XL\_MASK 0x0C

// ----------------

#define LSM6DS3\_ACC\_GYRO\_FS\_G\_245dps 0x00

#define LSM6DS3\_ACC\_GYRO\_FS\_G\_500dps 0x04

#define LSM6DS3\_ACC\_GYRO\_FS\_G\_1000dps 0x08

#define LSM6DS3\_ACC\_GYRO\_FS\_G\_2000dps 0x0C

#define LSM6DS3\_ACC\_GYRO\_FS\_G\_MASK 0x0C

// ----------------

#define LSM6DS3\_ACC\_GYRO\_XEN\_XL\_MASK 0x08

#define LSM6DS3\_ACC\_GYRO\_YEN\_XL\_MASK 0x10

#define LSM6DS3\_ACC\_GYRO\_ZEN\_XL\_MASK 0x20

#define LSM6DS3\_ACC\_GYRO\_XEN\_XL\_ENABLED 0x08

#define LSM6DS3\_ACC\_GYRO\_YEN\_XL\_ENABLED 0x10

#define LSM6DS3\_ACC\_GYRO\_ZEN\_XL\_ENABLED 0x20

// ----------------

#define LSM6DS3\_ACC\_GYRO\_XEN\_G\_DISABLED 0x00

#define LSM6DS3\_ACC\_GYRO\_XEN\_G\_ENABLED 0x08

#define LSM6DS3\_ACC\_GYRO\_YEN\_G\_DISABLED 0x00

#define LSM6DS3\_ACC\_GYRO\_YEN\_G\_ENABLED 0x10

#define LSM6DS3\_ACC\_GYRO\_ZEN\_G\_DISABLED 0x00

#define LSM6DS3\_ACC\_GYRO\_ZEN\_G\_ENABLED 0x20

#define LSM6DS3\_ACC\_GYRO\_XEN\_G\_MASK 0x08

#define LSM6DS3\_ACC\_GYRO\_YEN\_G\_MASK 0x10

#define LSM6DS3\_ACC\_GYRO\_ZEN\_G\_MASK 0x20

// ----------------

#define LSM6DS3\_ACC\_GYRO\_OUTX\_L\_XL 0X28

#define LSM6DS3\_ACC\_GYRO\_OUTX\_H\_XL 0X29

#define LSM6DS3\_ACC\_GYRO\_OUTY\_L\_XL 0X2A

#define LSM6DS3\_ACC\_GYRO\_OUTY\_H\_XL 0X2B

#define LSM6DS3\_ACC\_GYRO\_OUTZ\_L\_XL 0X2C

#define LSM6DS3\_ACC\_GYRO\_OUTZ\_H\_XL 0X2D

// ----------------

#define LSM6DS3\_ACC\_GYRO\_OUTX\_L\_G 0X22

#define LSM6DS3\_ACC\_GYRO\_OUTX\_H\_G 0X23

#define LSM6DS3\_ACC\_GYRO\_OUTY\_L\_G 0X24

#define LSM6DS3\_ACC\_GYRO\_OUTY\_H\_G 0X25

#define LSM6DS3\_ACC\_GYRO\_OUTZ\_L\_G 0X26

#define LSM6DS3\_ACC\_GYRO\_OUTZ\_H\_G 0X27

// ----------------

void Accel\_Gyro\_Ini (void);

void AccelGyro\_Read (void);

// ----------------

#endif / \* LSM6DS3\_H\_ \* /

After copying from another function, add the following code to the GyroInit function (since nothing changes here):

        uint8\_t value = 0;

**// auto-increment the address of the register**

**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C);**

**value & = ~ LSM6DS3\_ACC\_GYRO\_IF\_INC\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_IF\_INC\_ENABLED;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C, value);**

**// set the BDU bit**

**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C);**

**value & = ~ LSM6DS3\_ACC\_GYRO\_BDU\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_BDU\_BLOCK\_UPDATE;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C, value);**

**// select FIFO mode**

**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_FIFO\_CTRL5);**

**value & = ~ LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_BYPASS;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_FIFO\_CTRL5, value);**

Since nothing has changed in the code, an explanation is not required.

Now add the following code here:

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_FIFO\_CTRL5, value);

**// while we turn off the sensor (ODR\_XL = 0000)**

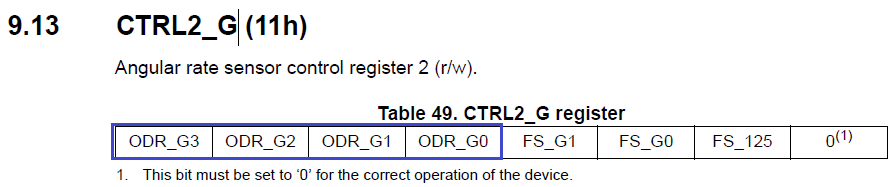
**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL2\_G);**

**value & = ~ LSM6DS3\_ACC\_GYRO\_ODR\_G\_MASK;**

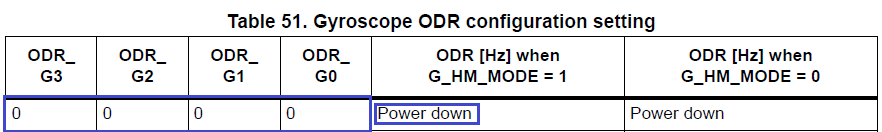
**value | = LSM6DS3\_ACC\_GYRO\_ODR\_G\_POWER\_DOWN;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL2\_G, value);**

In this code, we work with register **CTRL2\_G** (address 0X11), which is intended only for work with only the gyro, with its bits responsible for the frequency of data removal from the gyro



While we here disconnect the sensor (all bits are set to 0).



Then add the following code

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL2\_G, value);

**// Full scale selection 500 dps**

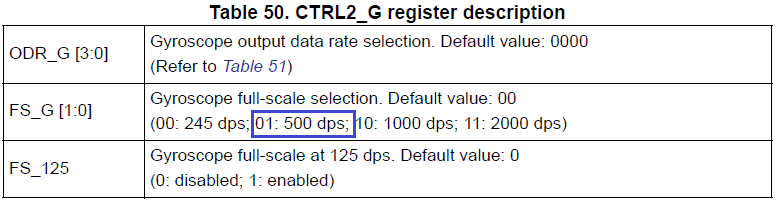
**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL2\_G);**

**value & = ~ LSM6DS3\_ACC\_GYRO\_FS\_G\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_FS\_G\_500dps;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL2\_G, value);**

Next we work with the same register, only with other bits, responsible for the maximum angular velocity measured by the sensor relative to the axis. I think we are enough 500 degrees per second, faster we will not be dispersed.



Write the code further

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL2\_G, value);

**// Turn on the axes**

**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL10\_C);**

**value & = ~ (LSM6DS3\_ACC\_GYRO\_XEN\_G\_ENABLED | \**

**LSM6DS3\_ACC\_GYRO\_YEN\_G\_ENABLED | \**

**LSM6DS3\_ACC\_GYRO\_ZEN\_G\_ENABLED);**

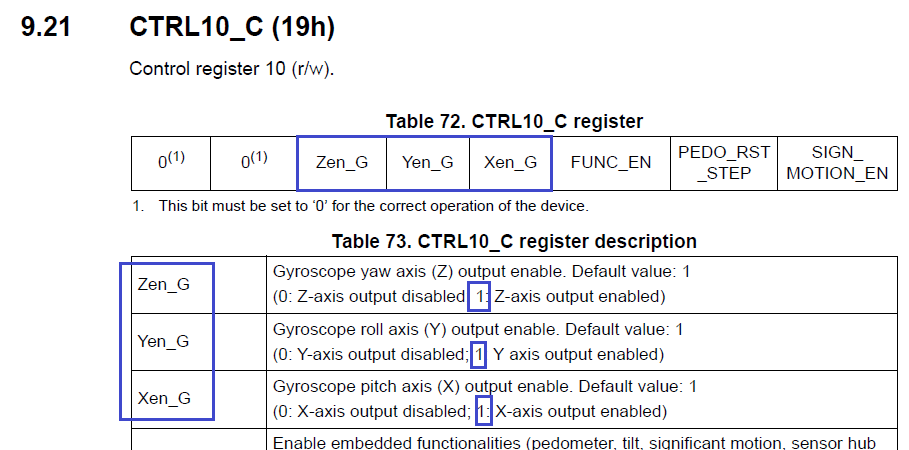
**value | = (LSM6DS3\_ACC\_GYRO\_XEN\_G\_MASK | \**

**LSM6DS3\_ACC\_GYRO\_YEN\_G\_MASK | \**

**LSM6DS3\_ACC\_GYRO\_ZEN\_G\_MASK);**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL10\_C, value);**

As indicated in the commentary, here we include the gyro axes. Turn on all 3 axes



We continue to write the source code

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL10\_C, value);

        // Enable Data Rate 833 Hz

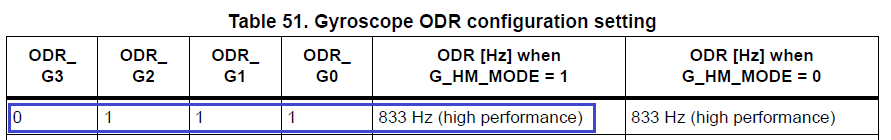
**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL2\_G);**

**value & = ~ LSM6DS3\_ACC\_GYRO\_ODR\_G\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_ODR\_G\_833Hz;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL2\_G, value);**

Here we turned on the frequency of reading 833 Hz. We also work with the above-mentioned 2 registers.



We do not use filters today. in this sensor and so is organized a good filtration

This initialization can be considered complete.

We will collect the code, we will sew the controller and make sure that the green LED is on.

In the [**next part of the**](http://narodstream.ru/stm-urok-47-podklyuchaem-giroskop-lsm6ds3-chast-2/) lesson we will write all the functions for data collection and sending them to the USART port and check this functionality in practice.

**Lesson 47**

**Part 2**

# ****Connect the gyroscope LSM6DS3****

In the [previous part of](http://narodstream.ru/stm-urok-47-podklyuchaem-giroskop-lsm6ds3-chast-1/) our lesson, we wrote all the macros that bring convenience in reading and writing code, wrote the initialization function and tested it in practice.

Add one more function Gyro\_GetXYZ, designed to poll the gyro axes. We'll make it completely based on the same for the accelerometer (Accel\_GetXYZ), copying the entire code into the function as well. Let's correct the code, using in the survey the registers intended for reading the gyro axes

// ---------------

void Gyro\_GetXYZ (int16\_t \* pData)

{

        uint8\_t buffer [6];

        uint8\_t i = 0;

        buffer [0] = Accel\_IO\_Read (0xD4, **LSM6DS3\_ACC\_GYRO\_OUTX\_L\_G**);

        buffer [1] = Accel\_IO\_Read (0xD4, **LSM6DS3\_ACC\_GYRO\_OUTX\_H\_G**);

        buffer [2] = Accel\_IO\_Read (0xD4, **LSM6DS3\_ACC\_GYRO\_OUTY\_L\_G**);

        buffer [3] = Accel\_IO\_Read (0xD4, **LSM6DS3\_ACC\_GYRO\_OUTY\_H\_G**);

        buffer [4] = Accel\_IO\_Read (0xD4, **LSM6DS3\_ACC\_GYRO\_OUTZ\_L\_G**);

        buffer [5] = Accel\_IO\_Read (0xD4, **LSM6DS3\_ACC\_GYRO\_OUTZ\_H\_G**);

        for (i = 0; i <3; i ++)

        {

                pData [i] = ((int16\_t) ((uint16\_t) buffer [2 \* i + 1] << 8) + buffer [2 \* i]);

        }

}

// ---------------

The function Accel\_ReadAcc also for the order will be renamed to the more universal - AccelGyro\_Read.The same will be done with the prototype of this function

// ---------------

void **AccelGyro\_Read**(void)

{

        int16\_t buffer [3] = {0};

Let's fix the line in it

        int16\_t xval, yval, zval;

**Gyro\_GetXYZ**(buffer);

Uncomment the lines of code responsible for the output of the information read from the axes into textual form, and in the graphical comment

        sprintf (str1, "X:% 06d Y:% 06d Z:% 06drn", xval, yval, zval);

        HAL\_UART\_Transmit\_DMA (& huart2, (uint8\_t \*) str1, strlen (str1));

// buf2 [0] = 0x11;

// buf2 [1] = 0x55;

// buf2 [2] = (uint8\_t) (xval >> 8);

// buf2 [3] = (uint8\_t) xval;

// buf2 [4] = (uint8\_t) (yval >> 8);

// buf2 [5] = (uint8\_t) yval;

// buf2 [6] = (uint8\_t) (zval >> 8);

// buf2 [7] = (uint8\_t) zval;

// HAL\_UART\_Transmit\_DMA (& huart2, buf2,8);

Here, too, a little bit of code

        if ( **z**val> **500**)

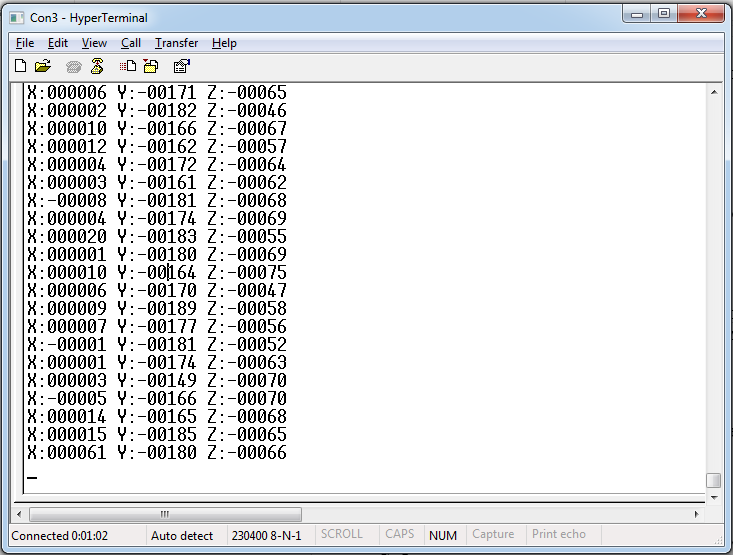
        {

In an infinite loop, in the main () function, uncomment and correct the function call

  / \* USER CODE BEGIN 3 \* /

                AccelGyro\_Read ();

Let's assemble the code, we'll sew the controller and see: when the board rotates counterclockwise relative to the vertical axis, the green LED should light up. Then we check the data in the Hyper Terminal program. Should be such a result.



Let's try to adjust the reading a little, because if you do not twist the board, then the readings will be slightly different from 0.

xval = buffer [0];

        yval = buffer [1] +161;

        zval = buffer [2] +80;

I got such numbers, you, maybe there will be others. While I did not find another way to calibrate the sensor.

We'll collect the project and see the testimony again.

Now, on the contrary, uncomment the code for visualization, and comment out the text output. Also, change the x and y axes with each other, rotate one axis in the buffer due to the perpendicular arrangement of the small board, so it will be more convenient to twist the cube.

        zval = buffer [2];

// sprintf (str1, "X:% 06d Y:% 06d Z:% 06drn", xval, yval, zval);

// HAL\_UART\_Transmit\_DMA (& huart2, (uint8\_t \*) str1, strlen (str1));

        xval = -xval;

        buf2 [0] = 0x11;

        buf2 [1] = 0x55;

        buf2 [2] = (uint8\_t) (xval >> 8);

        buf2 [3] = (uint8\_t) xval;

        buf2 [4] = (uint8\_t) (yval >> 8);

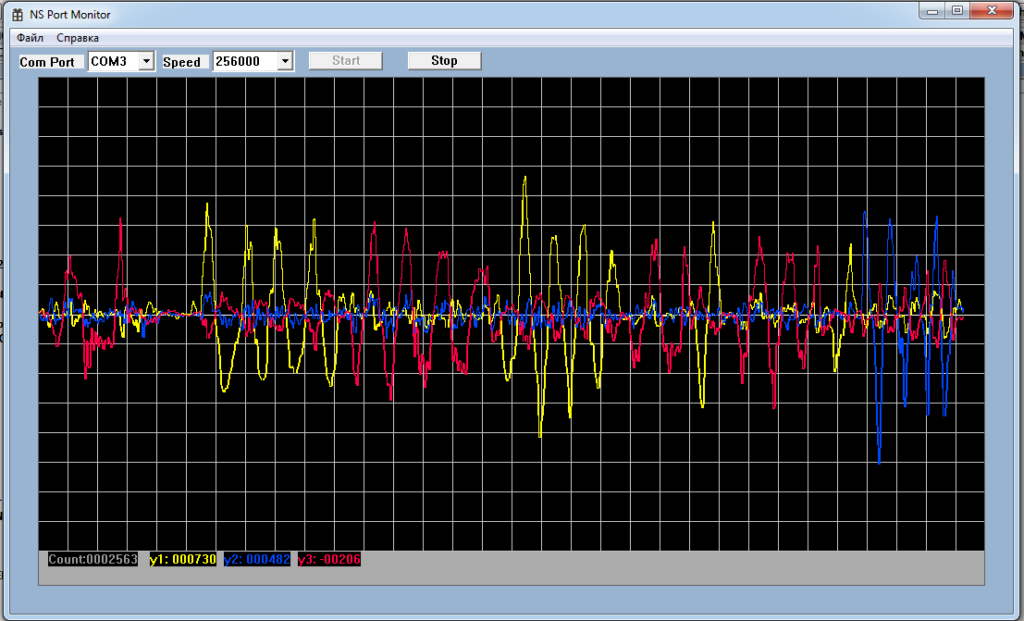
        buf2 [5] = (uint8\_t) yval;

        buf2 [6] = (uint8\_t) (zval >> 8);

        buf2 [7] = (uint8\_t) zval;

        HAL\_UART\_Transmit\_DMA (& huart2, buf2.8);

We will collect the code and we will sew the controller. Start the NS Port Monitor program first. We twist the fee. The result should be as follows (to increase the size, click on the picture):

[](http://narodstream.ru/wp-content/uploads/2016/11/image03_1107.png)

Now start another program, NS Port Visual. Also turn the board. The result should be as follows:

