**Lesson 45**

**Part 1**

**Connect the gyroscope LSM6DS0**

        Today we will once again work with a sensor that combines two functionalities - an accelerometer and a gyroscope - **LSM6DS0**. It was done using MEMS technology. Installed on the expansion board **X-NUCLEO-IKS01A1**, designed to work with the debug board 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.

The gyro in this sensor has the following technical characteristics:

        The reading range is ± 245 / ± 500 / ± 2000 dps;

The sensitivity is 8.75 - 70 mdps / LSb;

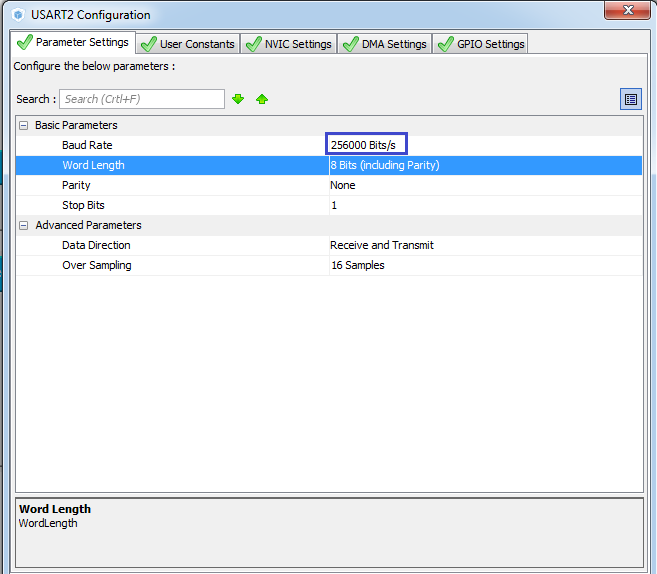
The deviation from zero is ± 30 dps when the range is set to 2000 dps.

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\_LSM6DS0, only we call this project now, respectively, Gyro\_LSM6DS0.

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 lsm6ds0.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 LIS3DSH\_H\_

#define LIS3DSH\_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 LSM6DS0\_ACC\_GYRO\_CTRL\_REG5\_XL 0X1F

#define LSM6DS0\_ACC\_GYRO\_CTRL\_REG6\_XL 0X20

#define LSM6DS0\_ACC\_GYRO\_CTRL\_REG8 0X22

#define LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G 0X10

#define LSM6DS0\_ACC\_GYRO\_CTRL\_REG2\_G 0X11

#define LSM6DS0\_ACC\_GYRO\_CTRL\_REG4 0X1E

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

#define LSM6DS0\_ACC\_GYRO\_BDU\_DISABLE 0x00

#define LSM6DS0\_ACC\_GYRO\_BDU\_ENABLE 0x40

#define LSM6DS0\_ACC\_GYRO\_BDU\_MASK 0x40

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

#define LSM6DS0\_ACC\_GYRO\_ODR\_XL\_POWER\_DOWN 0x00

#define LSM6DS0\_ACC\_GYRO\_ODR\_XL\_10Hz 0x20

#define LSM6DS0\_ACC\_GYRO\_ODR\_XL\_50Hz 0x40

#define LSM6DS0\_ACC\_GYRO\_ODR\_XL\_119Hz 0x60

#define LSM6DS0\_ACC\_GYRO\_ODR\_XL\_238Hz 0x80

#define LSM6DS0\_ACC\_GYRO\_ODR\_XL\_476Hz 0xA0

#define LSM6DS0\_ACC\_GYRO\_ODR\_XL\_952Hz 0xC0

#define LSM6DS0\_ACC\_GYRO\_ODR\_XL\_MASK 0xE0

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

#define LSM6DS0\_ACC\_GYRO\_ODR\_G\_POWER\_DOWN 0x00

#define LSM6DS0\_ACC\_GYRO\_ODR\_G\_15Hz 0x20

#define LSM6DS0\_ACC\_GYRO\_ODR\_G\_60Hz 0x40

#define LSM6DS0\_ACC\_GYRO\_ODR\_G\_119Hz 0x60

#define LSM6DS0\_ACC\_GYRO\_ODR\_G\_238Hz 0x80

#define LSM6DS0\_ACC\_GYRO\_ODR\_G\_476Hz 0xA0

#define LSM6DS0\_ACC\_GYRO\_ODR\_G\_952Hz 0xC0

#define LSM6DS0\_ACC\_GYRO\_ODR\_G\_MASK 0xE0

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

#define LSM6DS0\_ACC\_GYRO\_FS\_XL\_2g 0x00

#define LSM6DS0\_ACC\_GYRO\_FS\_XL\_16g 0x08

#define LSM6DS0\_ACC\_GYRO\_FS\_XL\_4g 0x10

#define LSM6DS0\_ACC\_GYRO\_FS\_XL\_8g 0x18

#define LSM6DS0\_ACC\_GYRO\_FS\_XL\_MASK 0x18

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

#define LSM6DS0\_ACC\_GYRO\_FS\_G\_245dps 0x00

#define LSM6DS0\_ACC\_GYRO\_FS\_G\_500dps 0x08

#define LSM6DS0\_ACC\_GYRO\_FS\_G\_1000dps 0x10

#define LSM6DS0\_ACC\_GYRO\_FS\_G\_2000dps 0x18

#define LSM6DS0\_ACC\_GYRO\_FS\_G\_MASK 0x18

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

#define LSM6DS0\_ACC\_GYRO\_OUT\_SEL\_BYPASS\_HPF\_AND\_LPF2 0x00

#define LSM6DS0\_ACC\_GYRO\_OUT\_SEL\_BYPASS\_LPF2 0x01

#define LSM6DS0\_ACC\_GYRO\_OUT\_SEL\_USE\_HPF\_AND\_LPF2 0x02

#define LSM6DS0\_ACC\_GYRO\_OUT\_SEL\_MASK 0x03

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

#define LSM6DS0\_ACC\_GYRO\_BW\_G\_LOW 0x00

#define LSM6DS0\_ACC\_GYRO\_BW\_G\_NORMAL 0x01

#define LSM6DS0\_ACC\_GYRO\_BW\_G\_HIGH 0x02

#define LSM6DS0\_ACC\_GYRO\_BW\_G\_ULTRA\_HIGH 0x03

#define LSM6DS0\_ACC\_GYRO\_BW\_G\_MASK 0x03

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

#define LSM6DS0\_ACC\_GYRO\_XEN\_XL\_ENABLE 0x08

#define LSM6DS0\_ACC\_GYRO\_YEN\_XL\_ENABLE 0x10

#define LSM6DS0\_ACC\_GYRO\_ZEN\_XL\_ENABLE 0x20

#define LSM6DS0\_ACC\_GYRO\_XEN\_XL\_MASK 0x08

#define LSM6DS0\_ACC\_GYRO\_YEN\_XL\_MASK 0x10

#define LSM6DS0\_ACC\_GYRO\_ZEN\_XL\_MASK 0x20

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

#define LSM6DS0\_ACC\_GYRO\_XEN\_G\_DISABLE 0x00

#define LSM6DS0\_ACC\_GYRO\_XEN\_G\_ENABLE 0x08

#define LSM6DS0\_ACC\_GYRO\_YEN\_G\_DISABLE 0x00

#define LSM6DS0\_ACC\_GYRO\_YEN\_G\_ENABLE 0x10

#define LSM6DS0\_ACC\_GYRO\_ZEN\_G\_DISABLE 0x00

#define LSM6DS0\_ACC\_GYRO\_ZEN\_G\_ENABLE 0x20

#define LSM6DS0\_ACC\_GYRO\_XEN\_G\_MASK 0x08

#define LSM6DS0\_ACC\_GYRO\_YEN\_G\_MASK 0x10

#define LSM6DS0\_ACC\_GYRO\_ZEN\_G\_MASK 0x20

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

#define LSM6DS0\_ACC\_GYRO\_OUT\_X\_L\_XL 0X28

#define LSM6DS0\_ACC\_GYRO\_OUT\_X\_H\_XL 0X29

#define LSM6DS0\_ACC\_GYRO\_OUT\_Y\_L\_XL 0X2A

#define LSM6DS0\_ACC\_GYRO\_OUT\_Y\_H\_XL 0X2B

#define LSM6DS0\_ACC\_GYRO\_OUT\_Z\_L\_XL 0X2C

#define LSM6DS0\_ACC\_GYRO\_OUT\_Z\_H\_XL 0X2D

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

#define LSM6DS0\_ACC\_GYRO\_OUT\_X\_L\_G 0X18

#define LSM6DS0\_ACC\_GYRO\_OUT\_X\_H\_G 0X19

#define LSM6DS0\_ACC\_GYRO\_OUT\_Y\_L\_G 0X1A

#define LSM6DS0\_ACC\_GYRO\_OUT\_Y\_H\_G 0X1B

#define LSM6DS0\_ACC\_GYRO\_OUT\_Z\_L\_G 0X1C

#define LSM6DS0\_ACC\_GYRO\_OUT\_Z\_H\_G 0X1D

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

void Accel\_Gyro\_Ini (void);

void AccelGyro\_Read (void);

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

#endif / \* LIS3DSH\_H\_ \* /

After copying from another function, add the following code to the GyroInit function:

        uint8\_t value = 0;

**// set the BDU bit**

**value = Accel\_IO\_Read (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG8);**

**value & = ~ LSM6DS0\_ACC\_GYRO\_BDU\_MASK;**

**value | = LSM6DS0\_ACC\_GYRO\_BDU\_ENABLE;**

**Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG8, value);**

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

Now add the following code here:

        Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG8, value);

**// while we turn off the sensor (ODR\_G = 000)**

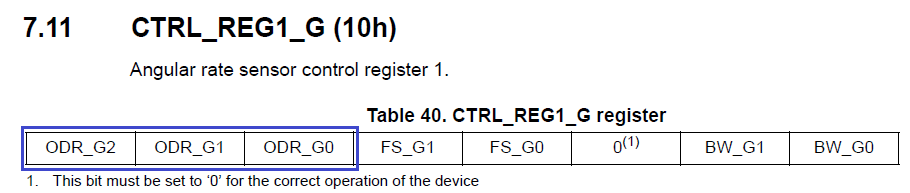
**value = Accel\_IO\_Read (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G);**

**value & = ~ LSM6DS0\_ACC\_GYRO\_ODR\_G\_MASK;**

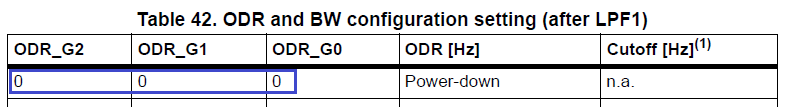
**value | = LSM6DS0\_ACC\_GYRO\_ODR\_G\_POWER\_DOWN;**

**Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G, value);**

In this code, we work with the register CTRL\_REG1\_G  (address 0X10), 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 (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G, value);

**// Full scale selection 500 dps**

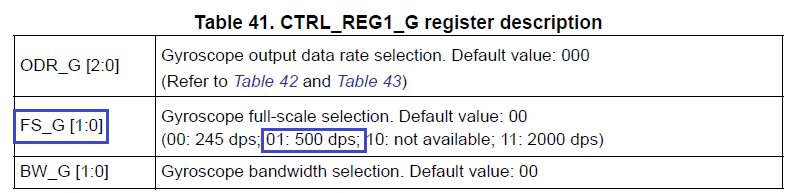
**value = Accel\_IO\_Read (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G);**

**value & = ~ LSM6DS0\_ACC\_GYRO\_FS\_G\_MASK;**

**value | = LSM6DS0\_ACC\_GYRO\_FS\_G\_500dps;**

**Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_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 (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G, value);

**// Turn on the axes**

**value = Accel\_IO\_Read (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG4);**

**value & = ~ (LSM6DS0\_ACC\_GYRO\_XEN\_G\_MASK | \**

**LSM6DS0\_ACC\_GYRO\_YEN\_G\_MASK | \**

**LSM6DS0\_ACC\_GYRO\_ZEN\_G\_MASK);**

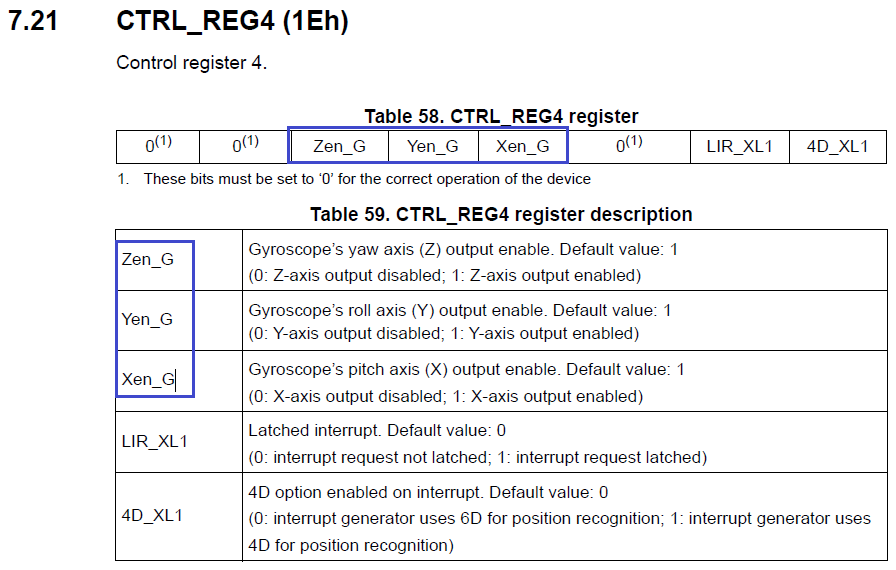
**value | = (LSM6DS0\_ACC\_GYRO\_XEN\_G\_ENABLE | \**

**LSM6DS0\_ACC\_GYRO\_YEN\_G\_ENABLE | \**

**LSM6DS0\_ACC\_GYRO\_ZEN\_G\_ENABLE);**

**Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG4, 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 (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG4, value);

**// Enable HPF and LPF2 (filtering)**

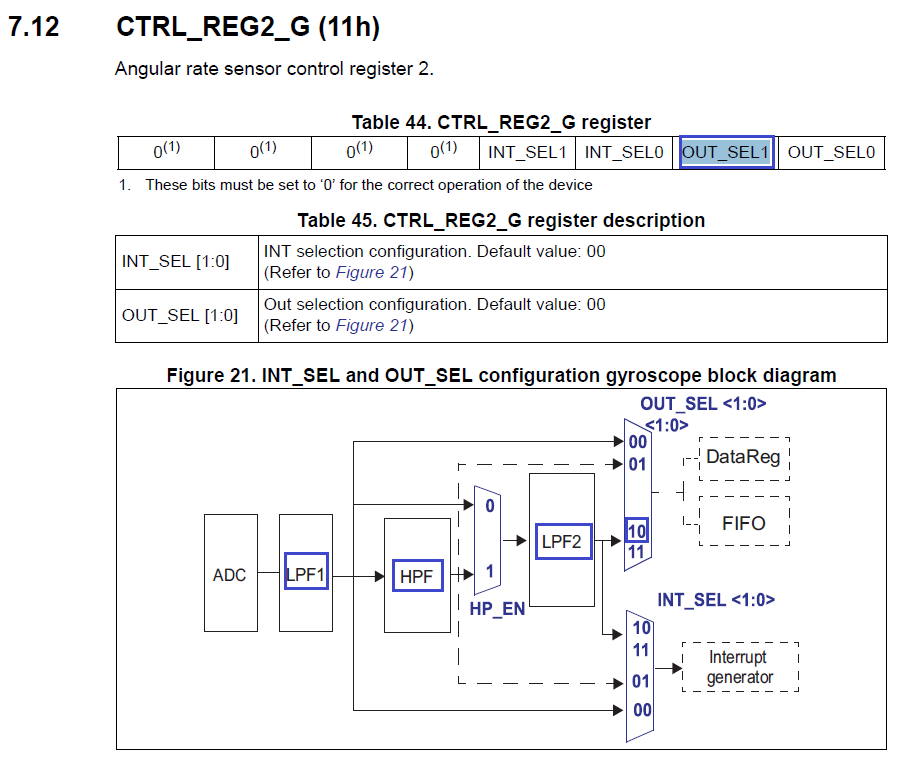
**value = Accel\_IO\_Read (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG2\_G);**

**value & = ~ LSM6DS0\_ACC\_GYRO\_OUT\_SEL\_MASK;**

**value | = LSM6DS0\_ACC\_GYRO\_OUT\_SEL\_USE\_HPF\_AND\_LPF2;**

**Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG2\_G, value);**

In register 2 (address 0x11), turn on the bit OUT\_SEL1, thereby including all the filters



Next include the following:

        Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG2\_G, value);

**// Enable BW (bandwidth). At 952 Hz settings there will be a throughput of 100 Hz**

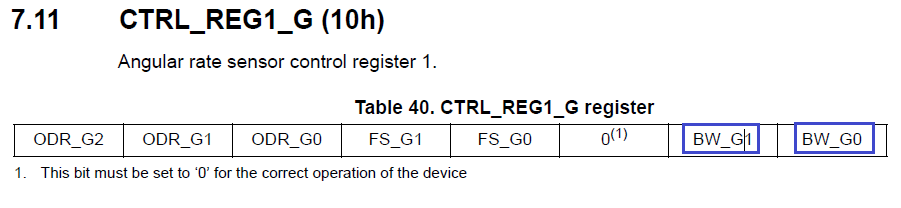
**value = Accel\_IO\_Read (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G);**

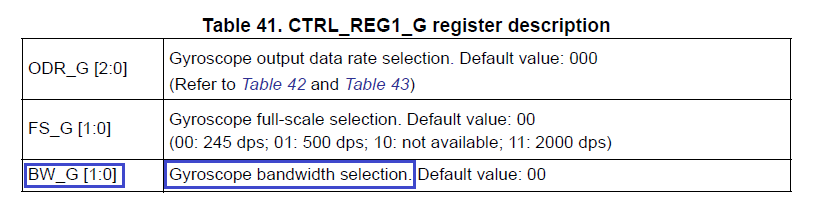
**value & = ~ LSM6DS0\_ACC\_GYRO\_BW\_G\_MASK;**

**value | = LSM6DS0\_ACC\_GYRO\_BW\_G\_ULTRA\_HIGH;**

**Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G, value);**

Here we in register 1 included both bits of BW\_G, thereby setting the bandwidth that will allow all filters to work.





Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG4, value);

**// Enable Data Rate 952 Hz**

**value = Accel\_IO\_Read (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G);**

**value & = ~ LSM6DS0\_ACC\_GYRO\_ODR\_G\_MASK;**

**value | = LSM6DS0\_ACC\_GYRO\_ODR\_G\_952Hz;**

**Accel\_IO\_Write (0xD6, LSM6DS0\_ACC\_GYRO\_CTRL\_REG1\_G, value);**

In this code, we will turn on the data polling frequency from the axes of 952 Hz. The actual frequency with all possible filters and throughput will be 100 Hz.

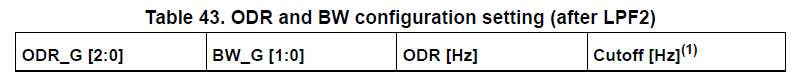


image04

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**](http://narodstream.ru/stm-urok-45-podklyuchaem-giroskop-lsm6ds0-chast-2/) our lesson, we will write the functions responsible for removing the data from the sensor and displaying them in various visualization programs, we will see in practice the operation of the gyroscope.

**Lesson 45**

**Part 2**

# ****Connect the gyroscope LSM6DS0****

In the [**previous part of**](http://narodstream.ru/stm-urok-45-podklyuchaem-giroskop-lsm6ds0-chast-1/) this lesson, we wrote all the macros that brought 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 (0xD6, **LSM6DS0\_ACC\_GYRO\_OUT\_X\_L\_G**);

        buffer [1] = Accel\_IO\_Read (0xD6, **LSM6DS0\_ACC\_GYRO\_OUT\_X\_H\_G**);

        buffer [2] = Accel\_IO\_Read (0xD6, **LSM6DS0\_ACC\_GYRO\_OUT\_Y\_L\_G**);

        buffer [3] = Accel\_IO\_Read (0xD6, **LSM6DS0\_ACC\_GYRO\_OUT\_Y\_H\_G**);

        buffer [4] = Accel\_IO\_Read (0xD6, **LSM6DS0\_ACC\_GYRO\_OUT\_Z\_L\_G**);

        buffer [5] = Accel\_IO\_Read (0xD6, **LSM6DS0\_ACC\_GYRO\_OUT\_Z\_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**)

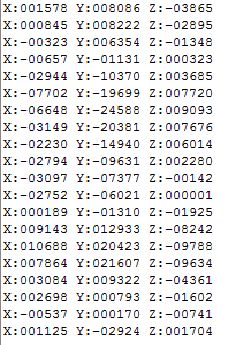
        {

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 Arduino port monitor. 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] -103;

        yval = buffer [1] -47;

        zval = buffer [2] -41;

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.

        zval = buffer [2];

// 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);

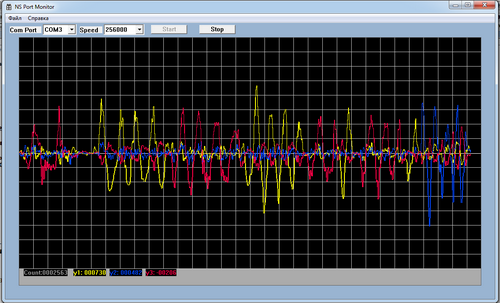
        buf2 [5] = (uint8\_t) yval;

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

        buf2 [7] = (uint8\_t) zval;

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

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 like this (click on the image to enlarge the image):

[](http://narodstream.ru/wp-content/uploads/2016/11/image08_1107.png)

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

