**Lesson 41**

**Part 1**

**We connect the accelerometer LIS3DSH**

The theme of our today's lesson is to connect a newer accelerometer than the one we used to study in[**lesson 39**](http://narodstream.ru/stm-urok-39-podklyuchaem-akselerometr-lsm303dlhc-chast-1/) . This accelerometer is also an accelerometer made using MEMS technology - **LIS3DSH**.

        First, this accelerometer, along with the I2C interface, can also be connected using the SPI interface, which makes the data transfer and use more reliable. Secondly, this accelerometer is installed on the STM32F4 Discovery board, which we have been working with for a very long time, and it has already become like a native. And this I consider to be an important motivation. And thirdly, it has better technical characteristics:

        The reading range is ± 2g / ± 4g / ± 6g / ± 8g / ± 16g;

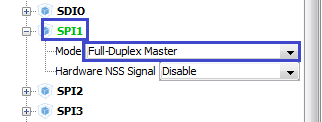
The sensitivity is 0.06-0.73 mg / digit;

Deviation from zero ± 60 mg.

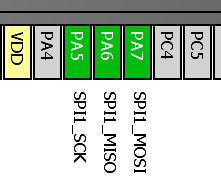
Well, we will get acquainted with other characteristics, subtleties, registers and other pitfalls of the accelerometer during its programming.

We will create a project for the Cube MX from one of the past projects USB\_OTG\_CDC, because instead of USART to transfer the accelerometer readings to the PC, we will try to use the USB CDC Device, since it is more convenient to work with it because of the need for some intermediate adapters. Call the project ACCEL407.

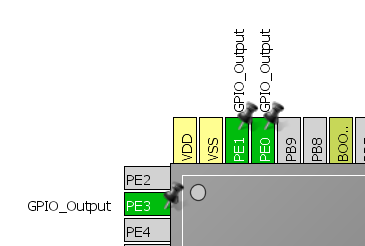
        Run the project in Cube MX, disable I2C, enable SPI1 in Full-Duplex Master mode



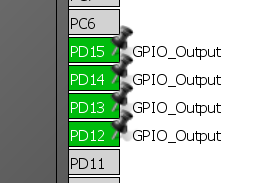
        SPI leaves the default and does not redirect anywhere



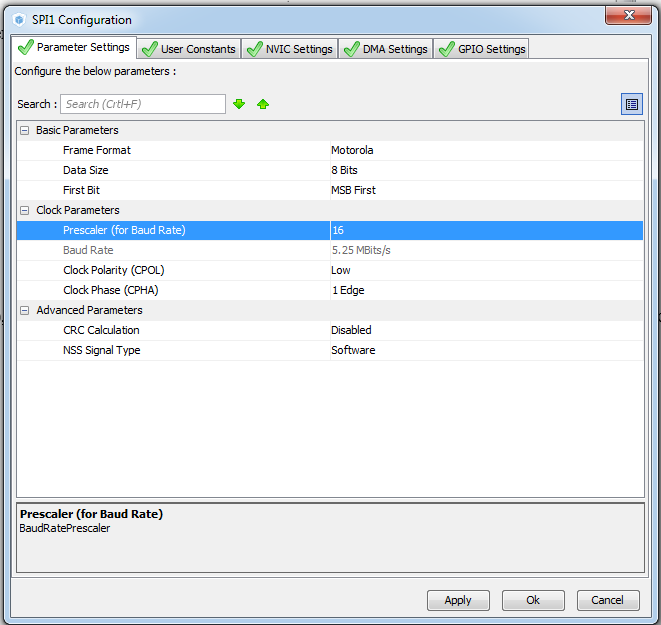
We will include 3 more legs. One for chip selection (CS), the other two for interrupt detection. Perhaps the latter we do not need, but for the sake of inclusion, so that by mistake they do not subsequently be involved in something else.



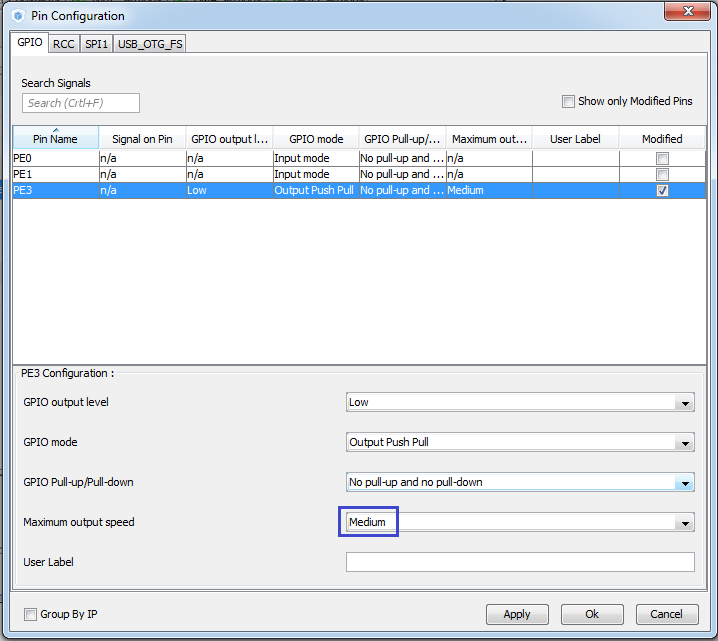
Also, we'll turn on the output of the ports of the ports to control the multi-colored LEDs on the board



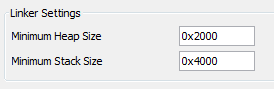
In the SPI settings, only the transfer rate will be changed



Also, change the settings in the GPIO at the foot of the PE3 port, setting it to Medium.



We go into Project -> Settings and change the stack and heap values ​​there so that the USB device does not give us an error after installing the driver (Code 10).



Generate the project and open it in Keil 5. Also, by tradition, configure the programmer for auto-cutting.We will collect the project.

We remove everything that concerns the character display, since we probably will not need it. Remove the library connection from main.h:

#include "stm32f4xx\_hal.h"

~~#include "lcd.h"~~

Here we leave only the transfer to USB to check if it works.

  / \* USER CODE BEGIN 2 \* /

        sprintf (str\_tx, "USB Transmitrn");

  / \* USER CODE END 2 \* /

In the endless loop, we also leave only what concerns the transfer to USB

  while (1)

  {

        CDC\_Transmit\_FS ((uint8\_t \*) str\_tx, strlen (str\_tx));

        HAL\_Delay (500);

  / \* USER CODE END WHILE \* /

Let's run the controller to make sure the USB port is working and delete the whole transfer, the line array and the line preparation:

/ \* Private variables ------------------- \* /

~~char str\_rx [21];~~

/ \* USER CODE END PV \* /

-

  / \* USER CODE BEGIN 2 \* /

~~sprintf (str\_tx, "USB Transmitrn");~~

  / \* USER CODE END 2 \* /

-

  while (1)

  {

~~CDC\_Transmit\_FS ((uint8\_t \*) str\_tx, strlen (str\_tx));~~

~~HAL\_Delay (500);~~

  / \* USER CODE END WHILE \* /

-

~~char str\_tx [21];~~

  / \* USER CODE END 1 \* /

Also delete the following lines from the file usbd\_cdc\_if.c, and uncomment it one line:

/ \* USER CODE BEGIN PRIVATE\_VARIABLES \* /

~~extern char str\_rx [21];~~

/ \* USER CODE END PRIVATE\_VARIABLES \* /

-

static int8\_t CDC\_Receive\_FS (uint8\_t \* Buf, uint32\_t \* Len)

{

  / \* USER CODE BEGIN 6 \* /

**USBD\_CDC\_SetRxBuffer (& hUsbDeviceFS, & Buf [0]);**

~~strncpy (str\_rx, (char \*) Buf, \* Len);~~

~~str\_rx [\* Len] = 0;~~

  USBD\_CDC\_ReceivePacket (& hUsbDeviceFS);

  return (USBD\_OK);

  / \* USER CODE END 6 \* /

We rewrite the project just in case, to make sure that we do not have any errors.

Two files to work with the accelerometer can be used from the last lesson, renaming them properly, connecting to the project and making some changes.

The names, respectively, will change to lis3dsh.h and lis3dsh.c. At the same time, we will remove the files lcd.h and lcd.c. from Inc and Src. We connect these files by adding lis3dsh.c in the project to the Application / User group, and connecting lis3dsh.h to main.h. Also we'll add macros, like those we made in the project under 3 discovery for convenient control of the LEDs.

#include "stm32f4xx\_hal.h"

**#include "lis3dsh.h"**

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

**#define LD\_PORT GPIOD**

**#define LD3 GPIO\_PIN\_13 // ORANGE**

**#define LD4 GPIO\_PIN\_12 // GREEN**

**#define LD5 GPIO\_PIN\_14 // RED**

**#define LD6 GPIO\_PIN\_15 // BLUE**

**#define LD3\_ON HAL\_GPIO\_WritePin (LD\_PORT, LD3, GPIO\_PIN\_SET) // ORANGE**

**#define LD4\_ON HAL\_GPIO\_WritePin (LD\_PORT, LD4, GPIO\_PIN\_SET) // GREEN**

**#define LD5\_ON HAL\_GPIO\_WritePin (LD\_PORT, LD5, GPIO\_PIN\_SET) // RED**

**#define LD6\_ON HAL\_GPIO\_WritePin (LD\_PORT, LD6, GPIO\_PIN\_SET) // BLUE**

**#define LD3\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD3, GPIO\_PIN\_RESET) // ORANGE**

**#define LD4\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD4, GPIO\_PIN\_RESET) // GREEN**

**#define LD5\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD5, GPIO\_PIN\_RESET) // RED**

**#define LD6\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD6, GPIO\_PIN\_RESET) // BLUE**

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

Composition of these files after making changes.

The contents of lis3dsh.h:

**#ifndef LIS3DSH\_H\_**

**#define LIS3DSH\_H\_**

**#include "stm32f4xx\_hal.h"**

**#include <string.h>**

**// ----------------**

**#define ABS (x) (x <0)? (-x): x**

**// ----------------**

**#define LD\_PORT GPIOD**

**#define LD3 GPIO\_PIN\_13 // ORANGE**

**#define LD4 GPIO\_PIN\_12 // GREEN**

**#define LD5 GPIO\_PIN\_14 // RED**

**#define LD6 GPIO\_PIN\_15 // BLUE**

**#define LD3\_ON HAL\_GPIO\_WritePin (LD\_PORT, LD3, GPIO\_PIN\_SET) // ORANGE**

**#define LD4\_ON HAL\_GPIO\_WritePin (LD\_PORT, LD4, GPIO\_PIN\_SET) // GREEN**

**#define LD5\_ON HAL\_GPIO\_WritePin (LD\_PORT, LD5, GPIO\_PIN\_SET) // RED**

**#define LD6\_ON HAL\_GPIO\_WritePin (LD\_PORT, LD6, GPIO\_PIN\_SET) // BLUE**

**#define LD3\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD3, GPIO\_PIN\_RESET) // ORANGE**

**#define LD4\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD4, GPIO\_PIN\_RESET) // GREEN**

**#define LD5\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD5, GPIO\_PIN\_RESET) // RED**

**#define LD6\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD6, GPIO\_PIN\_RESET) // BLUE**

**// ----------------**

**void Accel\_Ini (void);**

**void Accel\_ReadAcc (void);**

**// ----------------**

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

The contents of the file lis3dsh.c:

**#include "lis3dsh.h"**

**// ------------**

**static void Error (void)**

**{**

**}**

**// -------------**

**uint8\_t Accel\_IO\_Read (uint16\_t DeviceAddr, uint8\_t RegisterAddr)**

**{**

**return 0;**

**}**

**// -------------**

**void Accel\_IO\_Write (uint16\_t DeviceAddr, uint8\_t RegisterAddr, uint8\_t Value)**

**{**

**}**

**// -------------**

**uint8\_t Accel\_ReadID (void)**

**{**

**uint8\_t ctrl = 0x00;**

**return ctrl;**

**}**

**// -------------**

**void Accel\_AccFilterConfig (uint8\_t FilterStruct)**

**{**

**}**

**// -------------**

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

**{**

**}**

**// -------------**

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

**{**

**}**

**// -------------**

**void Accel\_ReadAcc (void)**

**{**

**}**

**// -------------**

**void Accel\_Ini (void)**

**{**

**}**

**// -------------**

Call the sensor initialization in main ();

  / \* USER CODE BEGIN 2 \* /

**Accel\_Ini ();**

  / \* USER CODE END 2 \* /

In the function-error handler, turn on the red LED

static void Error (void)

{

**LD5\_ON;**

}

Add the handle of our SPI:

#include "lis3dsh.h"

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

**extern SPI\_HandleTypeDef hspi1;**

In the [**next part of the**](http://narodstream.ru/stm-urok-41-podklyuchaem-akselerometr-lis3dsh-chast-2/) lesson, we will attempt to read the data identifier, thereby making sure that the correct tire settings are set and that we are working with this sensor.   
Also write the function of writing data to the accelerometer registers.

**Lesson 41**

**Part 2**

# ****We connect the accelerometer LIS3DSH****

In the [**previous part of**](http://narodstream.ru/stm-urok-41-podklyuchaem-akselerometr-lis3dsh-chast-1/) our lesson, we briefly studied the documentation for the sensor, created a project for it, added some macros and set up library files.

Let's now write a function for receiving and transmitting over SPI:

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

**static uint8\_t SPIx\_WriteRead (uint8\_t Byte)**

**{**

**uint8\_t receivedbyte = 0;**

**if (HAL\_SPI\_TransmitReceive (& hspi1, (uint8\_t \*) & Byte, (uint8\_t \*) & receivedbyte, 1, 0x1000)! = HAL\_OK)**

**{**

**Error ();**

**}**

**return receivedbyte;**

**}**

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

Add some more macros to the header file lis3dsh.h:

#define LD6\_OFF HAL\_GPIO\_WritePin (LD\_PORT, LD6, GPIO\_PIN\_RESET) // BLUE

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

**#define CS\_GPIO\_PORT GPIOE**

**#define CS\_PIN GPIO\_PIN\_3**

**#define CS\_ON HAL\_GPIO\_WritePin (CS\_GPIO\_PORT, CS\_PIN, GPIO\_PIN\_RESET)**

**#define CS\_OFF HAL\_GPIO\_WritePin (CS\_GPIO\_PORT, CS\_PIN, GPIO\_PIN\_SET)**

**#define DUMMY\_BYTE ((uint8\_t) 0x00)**

**// ----------------**

**#define LIS3DSH\_WHO\_AM\_I\_ADDR 0x0F**

**// ----------------**

**#define READWRITE\_CMD ((uint8\_t) 0x80)**

**#define MULTIPLEBYTE\_CMD ((uint8\_t) 0x40)**

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

Add the code to the **Accel\_IO\_Read**function , changing also the arguments and the input and return values:

**void** Accel\_IO\_Read ( **uint8\_t \* pBuffer, uint8\_t ReadAddr, uint16\_t NumByteToRead**)

{

**if (NumByteToRead> 0x01)**

**{**

**ReadAddr | = (uint8\_t) (READWRITE\_CMD | MULTIPLEBYTE\_CMD);**

**}**

**else**

**{**

**ReadAddr | = (uint8\_t) READWRITE\_CMD;**

**}**

**CS\_ON;**

**SPIx\_WriteRead (ReadAddr);**

**while (NumByteToRead> 0x00)**

**{**

**/ \* Send dummy byte (0x00) to generate the SPI clock to ACCELEROMETER (Slave device) \* /**

**\* pBuffer = SPIx\_WriteRead (DUMMY\_BYTE);**

**NumByteToRead-;**

**pBuffer ++;**

**}**

**CS\_OFF;**

}

Add a line to the **Accel\_ReadID**function :

uint8\_t Accel\_ReadID (void)

{

  uint8\_t ctrl = **0**;

**Accel\_IO\_Read (& ctrl, LIS3DSH\_WHO\_AM\_I\_ADDR, 1);**

  return ctrl;

}

Well, also add the code to the initialization function **Accel\_Ini**:

void Accel\_Ini (void)

{

**uint16\_t ctrl = 0x0000;**

**HAL\_Delay (1000);**

**if (Accel\_ReadID () == 0x3F) LD4\_ON;**

**else Error ();**

}

We will collect the code and tell the controller and see if we have a green LED



At us the light-emitting diode shines, means we on a correct way.

Now, to further initialize the accelerometer, we need a function to write data to the registers of our sensor

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

**void Accel\_IO\_Write (uint8\_t \* pBuffer, uint8\_t WriteAddr, uint16\_t NumByteToWrite)**

**{**

**CS\_OFF;**

**if (NumByteToWrite> 0x01)**

**{**

**WriteAddr | = (uint8\_t) MULTIPLEBYTE\_CMD;**

**}**

**CS\_ON;**

**SPIx\_WriteRead (WriteAddr);**

**while (NumByteToWrite> = 0x01)**

**{**

**SPIx\_WriteRead (\* pBuffer);**

**NumByteToWrite-;**

**pBuffer ++;**

**}**

**CS\_OFF;**

**}**

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

In the [**next part of the**](http://narodstream.ru/stm-urok-41-podklyuchaem-akselerometr-lis3dsh-chast-3/) lesson, we finish writing the initialization of the accelerometer, and also write a function to read the data from the sensor.

**Lesson 41**

**Part 3**

# ****We connect the accelerometer LIS3DSH****

In the [**previous part of**](http://narodstream.ru/stm-urok-41-podklyuchaem-akselerometr-lis3dsh-chast-2/) our lesson, we considered the data identifier, thus making sure the correct settings of the bus and the fact that we are working with this sensor, and also wrote the function of writing data to the accelerometer registers.

Now write write the code in the function of initializing the settings

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

**{**

**uint8\_t ctrl = 0x00;**

**ctrl = (uint8\_t) (InitStruct);**

**Accel\_IO\_Write (& ctrl, LIS3DSH\_CTRL\_REG4\_ADDR, 1);**

**ctrl = (uint8\_t) (InitStruct >> 8);**

**Accel\_IO\_Write (& ctrl, LIS3DSH\_CTRL\_REG5\_ADDR, 1);**

**}**

The purpose of the registers 4 and 5 of the sensor will be discussed below when we enter the settings data there.

Now we will continue to write the code in the main initialization function of our accelerometer

        else Error ();

**/ \* Configure MEMS: power mode (ODR) and axes enable \* /**

**ctrl = (uint16\_t) (LIS3DSH\_DATARATE\_100 | LIS3DSH\_XYZ\_ENABLE);**

**/ \* Configure MEMS: full scale and self test \* /**

**ctrl | = (uint16\_t) ((LIS3DSH\_SERIALINTERFACE\_4WIRE |**

**LIS3DSH\_SELFTEST\_NORMAL |**

**LIS3DSH\_FULLSCALE\_2 |**

**LIS3DSH\_FILTER\_BW\_800) << 8);**

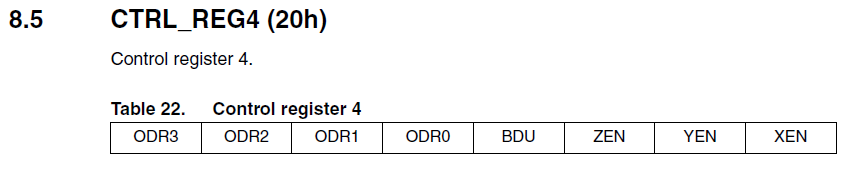
**AccInit (ctrl);**

**LD6\_ON;**

}

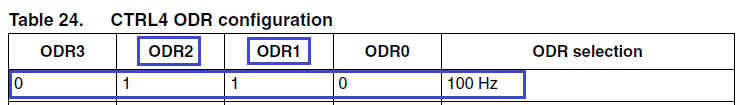
And now we'll try to figure out what and where we're putting it.

First, **let's look at the LIS3DSH\_CTRL\_REG4\_ADDR**register **(address 0x20)**

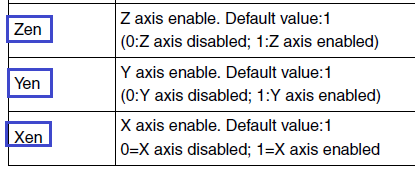


In it we will configure the following bits:

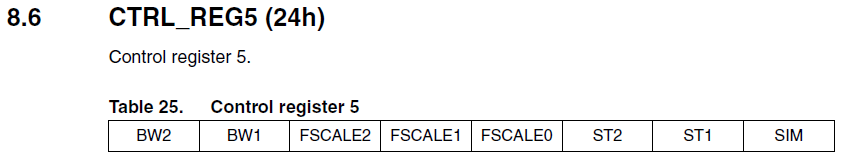
**LIS3DSH\_DATARATE\_100**: value **0x60**: with this value we will include the bits ODR1 and ODR2, thereby adjust the data rate of 100 hertz.



**LIS3DSH\_XYZ\_ENABLE**: value **0x07: enable the**bits of all axes, thus let the sensor know that it reads the data of all three axes (x, y and z)

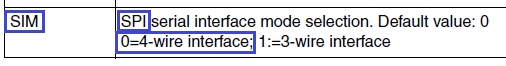


We do not include anything else in this register. Now let's **look**at the following case:**LIS3DSH\_CTRL\_REG5\_ADDR (address 0x24).**

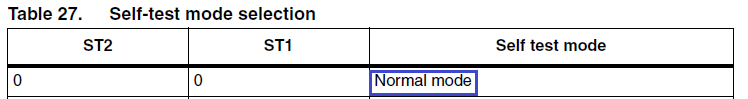


Here are the settings:

**LIS3DSH\_SERIALINTERFACE\_4WIRE:**value **0x00:**this bit is responsible for enabling the interface bus mode. Leave 0, so we turn on the 4-wire SPI.



**LIS3DSH\_SELFTEST\_NORMAL:**value **0x00:**here we will denote that we will not include bits 1 and 2 (ST1 and ST2), thereby enabling the initial self-test mode (normal mode).



**LIS3DSH\_FULLSCALE\_2:**value **0x00:**we also do not turn on the FSCALE bits, let's say the accelerometer so that it measures the readings on all axes ranging from -2G to + 2G.

image14

**LIS3DSH\_FILTER\_BW\_800:**value **0x00:**bits of the bandwidth of the anti-aliasing filter. We do not install them, so we will have a bandwidth of 800 hertz.

image15

Well, in order to get information that we have nothing moved, we'll turn on the blue LED at the end of the function.

Now let's move on to the more interesting part of our lesson - getting the measured readings from the sensor. We will enter a certain code into the function **Accel\_GetXYZ.**

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

{

**int8\_t buffer [6];**

**uint8\_t crtl, i = 0x00;**

**float sensitivity = LIS3DSH\_SENSITIVITY\_0\_06G;**

**float valueinfloat = 0;**

}

In this code, adding to the variable function, we are interested in the sensitivity value**LIS3DSH\_SENSITIVITY\_0\_06G.**Here we set up that in a certain register we will insert later such data that will allow us to adjust the sensitivity of the sensor to 0.06 mg.

We **count the**data from the register **LIS3DSH\_CTRL\_REG5\_ADDR (address 0x24)**, the assignment of bits of which we analyzed above, to the variable ctrl.

float valueinfloat = 0;

**Accel\_IO\_Read (& crtl, LIS3DSH\_CTRL\_REG5\_ADDR, 1);**

}

Further from the corresponding sensor registers we consider the lower and higher bytes of the 16-bit readings measured for each of the three axes

  Accel\_IO\_Read (& crtl, LIS3DSH\_CTRL\_REG5\_ADDR, 1);

**Accel\_IO\_Read ((uint8\_t \*) & buffer [0], LIS3DSH\_OUT\_X\_L\_ADDR, 1);**

**Accel\_IO\_Read ((uint8\_t \*) & buffer [1], LIS3DSH\_OUT\_X\_H\_ADDR, 1);**

**Accel\_IO\_Read ((uint8\_t \*) & buffer [2], LIS3DSH\_OUT\_Y\_L\_ADDR, 1);**

**Accel\_IO\_Read ((uint8\_t \*) & buffer [3], LIS3DSH\_OUT\_Y\_H\_ADDR, 1);**

**Accel\_IO\_Read ((uint8\_t \*) & buffer [4], LIS3DSH\_OUT\_Z\_L\_ADDR, 1);**

**Accel\_IO\_Read ((uint8\_t \*) & buffer [5], LIS3DSH\_OUT\_Z\_H\_ADDR, 1);**

Next, we check the few bits from the 5 register. Since we are only interested in 1,2 and 5 bits, the rest will be reset by the mask (LIS3DSH\_\_FULLSCALE\_SELECTION = 0x38)

  Accel\_IO\_Read ((uint8\_t \*) & buffer [5], LIS3DSH\_OUT\_Z\_H\_ADDR, 1);

**switch (crtl & LIS3DSH\_\_FULLSCALE\_SELECTION)**

**{**

**}**

Depending on the values ​​of these bits, we make a value in the sensitivity variable, that is, we will thereby adjust the sensitivity of the sensor depending on the measurement limits.

  switch (crtl & LIS3DSH\_\_FULLSCALE\_SELECTION)

  {

**case LIS3DSH\_FULLSCALE\_2:**

**sensitivity = LIS3DSH\_SENSITIVITY\_0\_06G;**

**break;**

**case LIS3DSH\_FULLSCALE\_4:**

**sensitivity = LIS3DSH\_SENSITIVITY\_0\_12G;**

**break;**

**case LIS3DSH\_FULLSCALE\_6:**

**sensitivity = LIS3DSH\_SENSITIVITY\_0\_18G;**

**break;**

**case LIS3DSH\_FULLSCALE\_8:**

**sensitivity = LIS3DSH\_SENSITIVITY\_0\_24G;**

**break;**

**case LIS3DSH\_FULLSCALE\_16:**

**sensitivity = LIS3DSH\_SENSITIVITY\_0\_73G;**

**break;**

**default:**

**break;**

}

Well, in accordance with the sensitivity, we transform the measured readings a little and insert them into the buffer for further use in the work of our program.

   break;

  }

**for (i = 0; i <3; i ++)**

**{**

**valueinfloat = ((buffer [2 \* i + 1] << 8) + buffer [2 \* i]) \* sensitivity;**

**pData [i] = (int16\_t) valueinfloat;**

**}**

}

Next, we'll write the code in the Accel\_ReadAcc function.

void Accel\_ReadAcc (void)

{

**int16\_t buffer [3] = {0};**

**int16\_t xval, yval , zval = 0x00;**

**Accel\_GetXYZ (buffer);**

**HAL\_Delay (20);**

}

Readings from the sensor are read in an infinite loop in the main () function.

  / \* USER CODE BEGIN 3 \* /

**Accel\_ReadAcc ();**

  }

  / \* USER CODE END 3 \* /

In the [**next part of the**](http://narodstream.ru/stm-urok-41-podklyuchaem-akselerometr-lis3dsh-chast-4/) lesson, we will complete the work with the accelerometer-accelerometer LIS3DSH, finish writing the function of collecting values ​​from the axes of the accelerometer and monitoring these values ​​using LEDs, a terminal and a visualization program that I wrote myself.

# ****We connect the accelerometer LIS3DSH****

In the [**previous part of**](http://narodstream.ru/stm-urok-41-podklyuchaem-akselerometr-lis3dsh-chast-3/) our lesson, we finished writing the initialization of the accelerometer, and also wrote a function to read the data from the sensor.

We continue the function Accel\_ReadAcc. Enter the sensor readings into the buffer:

  Accel\_GetXYZ (buffer);

**xval = buffer [0];**

**yval = buffer [1];**

**zval = buffer [2];**

  HAL\_Delay (20);

Now we process these indications and, depending on their values, light certain LEDs on the board.

  zval = buffer [2];

**if ((ABS (xval))> (ABS (yval)))**

**{**

**if (xval> 200)**

**{**

**LD5\_ON;**

**}**

**else if (xval <-200)**

**{**

**LD4\_ON;**

**}**

**}**

**else**

**{**

**if (yval> 200)**

**{**

**LD3\_ON;**

**}**

**else if (yval <-200)**

**{**

**LD6\_ON;**

**}**

**}**

  HAL\_Delay (20);

And, accordingly, at the end of the function, we all extinguish the LEDs.

        HAL\_Delay (20);

**LD3\_OFF;**

**LD4\_OFF;**

**LD5\_OFF;**

**LD6\_OFF;**

}

We will sew the controller and see the result.

If everything is fine, we will try to transfer these results to a PC via USB CDC. Let's take some code from the last lesson.

Connect file for USB CDC

#include "lis3dsh.h"

**#include "usbd\_cdc\_if.h"**

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

Add a global variable to the file lis3dsh.c

extern SPI\_HandleTypeDef hspi1;

**uint8\_t buf2 [8] = {0};**

**char str1 [30] = {0};**

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

Insert the code into the Accel\_ReadAcc function

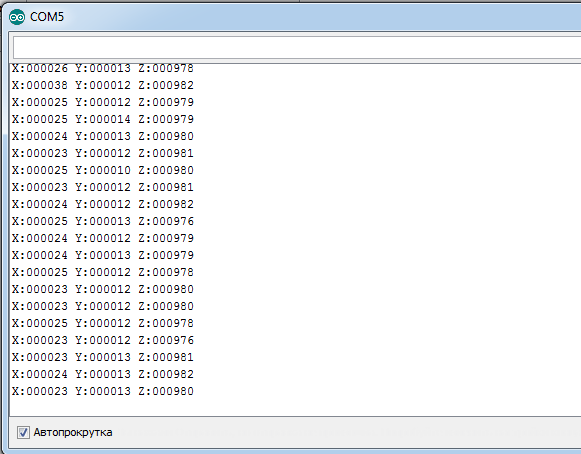
  zval = buffer [2];

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

**CDC\_Transmit\_FS ((uint8\_t \*) str1, strlen (str1));**

  if ((ABS (xval))> (ABS (yval)))

We'll collect the code, we'll sew the controller and see the result with the help of, for example, the monitor of the port built into the software for Arduino, as it slows down less than some similar utilities.



Excellent!

        Now let's try to see the indications with the help of some visualization utility. But since I did not find any utility to work correctly with USB, I had to write it myself. To use this utility, fill the buffer appropriately by commenting out the character transfer code in USB, and call the transfer function:

  zval = buffer [2];

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

// CDC\_Transmit\_FS ((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;**

**CDC\_Transmit\_FS (buf2, 8);**

        if ((ABS (xval))> (ABS (yval)))

Also we will remove while the multiplier in function of information gathering

  for (i = 0; i <3; i ++)

  {

    valueinfloat = ((buffer [2 \* i + 1] << 8) + buffer [2 \* i]) ~~\* sensitivity~~;

    pData [i] = (int16\_t) valueinfloat;

And instead of 200 in the LED control code, we put 2000, adding zero to 200, so that the LEDs are less sensitive to acceleration

        if ((ABS (xval))> (ABS (yval)))

  {

                if (xval> 200 **0**)

                {

                        LD5\_ON;

                }

                else if (xval <-200 **0**)

                {

                        LD4\_ON;

                }

        }

        else

        {

                if (yval> 200 **0**)

                {

                        LD3\_ON;

                }

                else if (yval <-200 **0**)

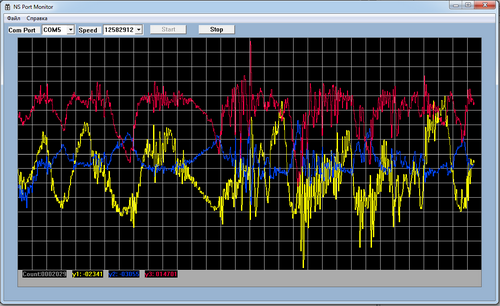
                {

                        LD6\_ON;

                }

        }

Run the utility and see the change in the axis chart in time, depending on the position of the debug board, and hence the sensor accelerometer (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2016/11/image00_1103.png)