**STM Lesson 99. HC-05. Master. We connect two MCs. Part 1**

Posted on [December 4, 2017](http://narodstream.ru/stm-urok-99-hc-05-master-soedinyaem-dva-mk-chast-1/)by [http://1.gravatar.com/avatar/4824b24065500834db4b9f331b608833?s=32&d=mm&r=gNarod Stream](http://narodstream.ru/author/admin/) Published in [Programming STM32](http://narodstream.ru/rub_stm32/)- [No Comments ↓](http://narodstream.ru/stm-urok-99-hc-05-master-soedinyaem-dva-mk-chast-1/#respond)

[Programming LessonsLearn all the in-house cooking **programming** training ! Get the book on the e-mail!To learn moresheremetev.info](https://an.yandex.ru/count/UlKIP9AbcQW50Dm1CLQyari00000ECgs7402I09Wl0Xe172WmvR10O01sE3CgmU80QE8tP0fa06cakID99W1eiM-tYEW0SAapOqag07iWR3U8xW1tAUPz1N00GBO0PQmhH3W0UQKqmxe0HRu0RQLthu1Y083e0BUkAe2kG8fKeJSb6qon_02kyIhoGRu0eA0W820e1600-RYgP81Y0EIhhwt29W3ky06g0CIi0C4k0J_0UW4gmRu1FfUY0NwNf05tkq2e0M-7AW5aXYm1P8Ok0NzNC05y9a7o0MmwW7G1UuAu0KCy0K1c0Q0qApp3g06xWAe1ku2oGR6AvDQIO1r4z46nh_8qKc0THFP1W00071n0000gGVuKVZX1e_4IB07W82G3D070k07XWhu1m60207G2BgAW870a802u0Yld9K1W0e1mGe00000003mFzWA0k0AW8bw-0h0_1M82mQg2n1w_M51-5400DXLa0HRTGK0m0k0emN82u3Kam7P2_XH-E46ZyH8w0lwNl0B1eWCjfNUlW7e30AO3PMCQF8D0FeD088E08aE00000000y3-G3i24FPWEnjVCr9M9uBeJe0x0X3sm3W6X3m0000000F0_g0_uey7nuE2ZuaW0?stat-id=3&test-tag=75866331701249&format-type=24&banner-test-tags=eyI0ODQ2MjkwMDY3IjoiNzU4NjYzMDIzNDkzMTIifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)

[Dating and chatting on web camerasRegistration and communication on the site are free. There is a guest entrance.To learn morehitlove.ru](https://an.yandex.ru/count/2MlJFi2Ubtu50DW1CLUyari00000ECgs7402I09Wl0Xe1724kxc20O01nfYFzRESyguWY06-YR7-Av01j92OyYAO0UxNkFyYe06CvvxN0QW1gh7jhWAu0OYYjfC2m042s07gYUa2w04E-06s0OW21A02ykA82Ra2ALA4t9HjCiVm0eUrqyS6-0A2W820WA0HW0EOdTdG0eW3WxVyxmgO0_7i0h031BW4_m7e19i3-0J6um681SRZ0P05ele1e0MYHAW5xJcm1Uqvk0NGFy05ee03o0NPOT05X1hW1MBm1G6O1lBBgdQW1ku2g0Rk0ia6nYkJMac0THFH1iQ_oD59W7KJsGO0001mSG000907Aga7-54Su2QFn4Ym1u20a0pG1mBW1uOA-0S1W0W1q0YwYe21m9200k08kR74180A0S4A00000000y3_O2WBW2e29UlWAmFmLY0i8gWiGe0EYiFXH0022LPW4MtK50C0BWAC5o0k0r9C1sGluKHpW9e_4IEWBnkC1y0i6Y0oiljw-0UWC0fWD_BrcyWq0-Wq0WWu0YGu00000003mFv0Em8Gzc0x6rypKbOdWkXEW3i24FR0E0Q4F00000000y3-e3_YZqSRYvAFYI000?stat-id=4&test-tag=75866331701249&format-type=24&banner-test-tags=eyI0NjgzMTM4MDc0IjoiNzU4NjYzMDIzNDkzMTIifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)

In [**lesson 97,**](http://narodstream.ru/stm-urok-97-podklyuchaem-bluetooth-modul-hc-05/) we learned how to connect the HC-05 module in the slave mode to a master device in the form of a smartphone. Also we were able to transfer data in both directions in the form of lines. Data is transmitted very confidently, so we can safely continue to move forward on this topic.

As part of today's lesson, we will try to configure this module as a master device, connect it to the same module acting as a slave device, leaving it connected to the STM32F103 controller. Then we already have a module configured as a master device, we connect to a similar microcontroller and as a result, the two controllers will connect wirelessly to us wirelessly via Bluetooth. Also to this controller we connect a character display with a resolution of 20 × 4 through the adapter to the I2C bus in order to track the data that will come to us from the slave device.

The task at first glance seems complicated, but it is not so. We will certainly cope and you will understand. that there is nothing absolutely terrible in this.

First, let's take another HC-05 module and also, as in the [**review**](http://narodstream.ru/bluetooth_modul_hc_05/) , connect it to the USB-TTL adapter, which in turn will be connected to the PC and run the terminal program CoolTerm also configured, as in [**the same review**](http://narodstream.ru/bluetooth_modul_hc_05/) . and, before connecting it to the controller, configure it in the master mode.

First of all, we will move our module to the mode of working with AT commands. Then we will check it by sending the command

http://narodstream.ru/wp-content/uploads/2017/10/Image11-1.png

We do not need the address and name of this module in principle. We need to know the address of the slave device to which we will connect the master.

Further we go on points:

1) First, reset all settings with the command " **AT + ORGL** ".

2) We will reset all paired devices by entering the **AT + RMAAD** command in the terminal .

3) Set the module mode of the master device - " **AT + ROLE = 1** ".

4) You can set the desired USART speed, for example 115200 - " **AT + UART = 115200,0,0** ". And it is not necessary that the speed was the same on the modules, they will agree with each other.

5) Restart the module with the command " **AT + RESET** ". and again we translate into the reception mode of AT commands.

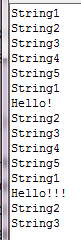
6) Set the password of the slave device " **AT + PSWD = 1234** ".

7) Recall the address of the slave from the survey class and connect the master with the slave at the command " **AT + BIND = 98d3,31,4058cf** ". If you do not remember or do not know the address of the slave, it can be recognized by also connecting to the USART-adapter and terminal program and then entering the universal command. But if we can not do this, for example, the slave device is somewhere soldered to us and we can not connect and test it using AT commands, that is, alternative ways of recognizing the address.

8) Once again we reboot " **AT + RESET** " and we can not switch the device to the command receiving mode, but just organize the support by connecting the KEY leg to the power leg for a short time. After that, do not forget to change the speed in the terminal program to the one that we installed in step 4.

9) Add the pair " **AT + PAIR = 98d3,31,4058cf, 5** " (the terminal in this case most likely does not answer OK) and just in case again reboot.

Now the controller to which the slave is connected can be turned on by supplying power to it. Devices should be connected and due to the program that we wrote in lesson 97, the corresponding data from the slave controller will come to the terminal program to which the master controller is connected via the adapter. Also we can now remove the echo from the port setting in the terminal program and try to transfer remotely lines from it, since we, as you remember, have an echo in the slave controller's program. And as a result, we will see the following picture



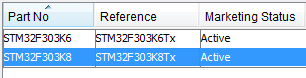
Excellent!

We connected the master module to the slave.

The next task: connect it to the controller, then connect the two controllers. For this we will take a similar small debug board. I have not found one at the moment, there are a couple of NUCLEO-F303K8 boards, one of which we'll take. The controller, mounted on it only 32 legs, but we do not need much. Although this card is not expensive, but it is very convenient, since it does not require a programmer that already exists in it and does not require USART adapters either, since this port is virtualized there directly via ST-Link. While we immediately connect the module HC-05 to this board will not, and we connect to it the display 20x4 through the I2C adapter. With this display we did a lot of work, so we know it. To which legs to connect the display, we will know when we will create a project in Cube MX, to which we,

You, of course, can apply any boards and any controllers that you have available, not necessarily those like me.

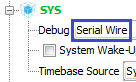
Run Cube MX, create a new project and select the controller



Turn on the quartz resonator

http://narodstream.ru/wp-content/uploads/2017/11/image02-4.png

Then enable debugging via SWD



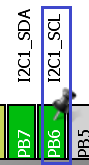
Turn on USART1

http://narodstream.ru/wp-content/uploads/2017/11/image04-3.png

Turn on the I2C for the display adapter

http://narodstream.ru/wp-content/uploads/2017/11/image05-3.png

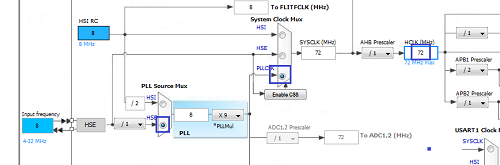
And the SCL foot is redefined in it, since the foot is not output by default and is used for the port through ST-Link



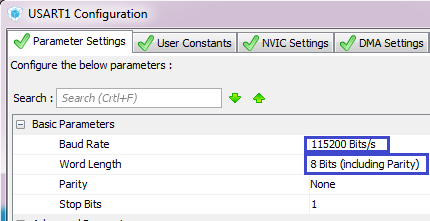
The foot of the PB3 is set to the output, since on it we have an LED

http://narodstream.ru/wp-content/uploads/2017/11/image07-4.png

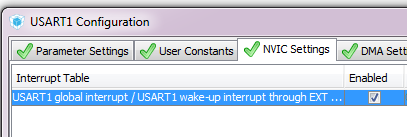
**Clock Configuration is** configured as follows (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2017/11/image08-3.png)

Go to **Configuration** and start by configuring USART



Enable interrupts

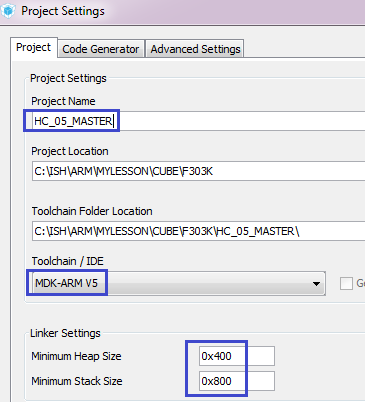


I do not touch anything in I2C.

But for USART in the NVIC section we will change the priority, otherwise for some reason, as soon as the interrupts processing from USART started, I2C stopped working

http://narodstream.ru/wp-content/uploads/2017/11/image11-3.png

We will configure the project, adding a little stack and heaps, we will also specify the path and come up with a project name, well, let's choose the programming environment



Generate the project, open it in Keil, configure the overload of the programmer, and also enable optimization level 1 and try to assemble our project.

Next, we **'ll** take the files **lcd.h** and **lcd.c** from the [**lesson 22**](http://narodstream.ru/stm-urok-22-hal-i2c-i2c-to-lcd2004/) project **I2CLCD80** and copy them to the corresponding folders of our project.

Then connect the file **lcd.c** to the project .

Open the file **lcd.h** and replace the name of the **HAL** library , and connect the library to work with the strings

#include "stm32f**3**xx\_hal.h"

**#include <string.h>**

At the same time, we will correct the errors in the comments a little

#define setled() LCD\_WriteByteI2CLCD(portlcd|=0x08) //**включение бита подсветки**

#define setwrite() LCD\_WriteByteI2CLCD(portlcd&=~0x02) //**установка линии RW в 0**

In the lcd.c file, we remove the declaration of the global string array

extern I2C\_HandleTypeDef hi2c1;

~~cha~~

Connect to the **main.c** library LCD

/\* USER CODE BEGIN Includes \*/

**#include "lcd.h"**

/\* USER CODE END Includes \*/

Let's add a global string array

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

**char str1[21];**

/\* USER CODE END PV \*/

Also, we initialize and clear the display in the **main ()** function and start receiving the byte in the USART bus

/\* USER CODE BEGIN 2 \*/

**LCD\_ini();**

**LCD\_Clear();**

**HAL\_UART\_Receive\_IT(&huart1,(uint8\_t\*)str1,1);**

Well, then we'll check our display

HAL\_UART\_Receive\_IT(&huart1,(uint8\_t\*)str1,1);

**LCD\_SetPos(0,0);**

**sprintf(str1,"String 1");**

**LCD\_String(str1);**

**LCD\_SetPos(4,1);**

**sprintf(str1,"String 2");**

**LCD\_String(str1);**

**LCD\_SetPos(8,2);**

**sprintf(str1,"String 3");**

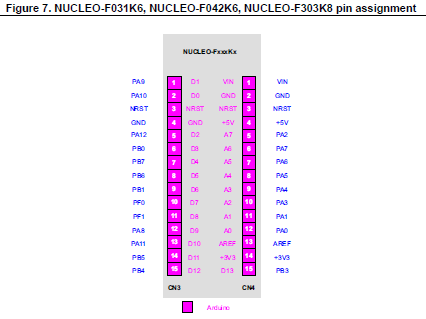
**LCD\_String(str1);**

**LCD\_SetPos(12,3);**

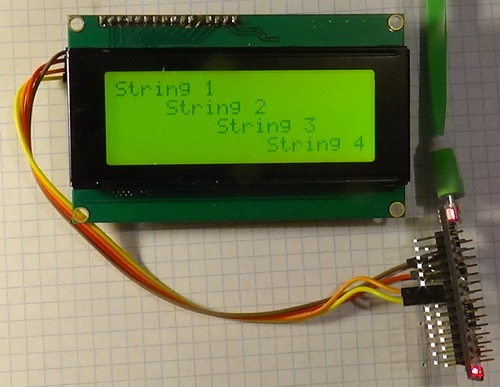
**sprintf(str1,"String 4");**

**LCD\_String(str1);**

Let's see the location of the legs on our debug board, since obviously the port legs are not specified - everything is indicated as in Arduino



Guided by this image, as well as pinout in the Cube MX, connect our display to the board, collect our code, we'll riddle the controller and see its performance



As we can see, everything works for us.

In the [**next part of**](http://narodstream.ru/stm-urok-99-hc-05-master-soedinyaem-dva-mk-chast-2/) our lesson we will write some functions on receiving a line from the module and sending them to the display and testing our code in practice.

**STM Lesson 99. HC-05. Master. We connect two MCs. Part 2**

Posted on [December 6, 2017](http://narodstream.ru/stm-urok-99-hc-05-master-soedinyaem-dva-mk-chast-2/)by [http://1.gravatar.com/avatar/4824b24065500834db4b9f331b608833?s=32&d=mm&r=gNarod Stream](http://narodstream.ru/author/admin/) Published in [Programming STM32](http://narodstream.ru/rub_stm32/)- [No Comments ↓](http://narodstream.ru/stm-urok-99-hc-05-master-soedinyaem-dva-mk-chast-2/#respond)

[Do you want to become a programmer from scratch?Read the mini-book of Konstantin Sheremetyev "The Way to the programmers". Save time!To learn moresheremetev.info](https://an.yandex.ru/count/Fupr2tIvKby50E01CMQyari00000ECgs7402I09Wl0Xe172yjVlW2801oVR8wjN_iy7O0OW1rDsVY3EG0PYatQmkc07YheoABA01eAJTh2we0SIkZ8eik07WXAYZ6S010jW1XCQR5E01vfJJ3kW1YW7u0Uhpuhu1Y083e0BUkAe2kG9gYXYJd1bcil02g8dCdGBu0eA0W820e1I00uEkzxu5Y0EQoAEZ2AW34h031BW4_m7e1Em3-0IblG681QMz0P05sVi2e0NbEAW5a32m1P0mk0N4OC05ku0Co0NGXWFG1TCLu0K-y0K1c0Q0qApp3g06xWAe1ku2oGR6AvDQIO1r4z46nh_8qKc0THFP1W00071n0000gGVuKUJFqvB4IB07W82G3D070k07XWhu1m60207G2BgAW870a802u0Yld9K1W0e1mGe00000003mFzWA0k0AW8bw-0h0_1M82mQg2n1qh7-T-54007-HWmLRTGK0m0k0emN82u3Kam7P2_XHvC_JaiH8w0kblG7m2mQ83Ehpuhu1w0m2c0sLZ6Zo3G3w3G223W293W0000000F0_a0x0X3sO3iRNpDILYU2w4w0Em8Gzi0u1eGy00000003mFwWF-AE1wQ_QfE98?stat-id=3&test-tag=89060471234561&format-type=24&banner-test-tags=eyI2MjIwOTE3MDA0IjoiODkwNjA0NDE4ODI2MjQifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)18+

[Buy robot constructorTeach your child to program. 6900 for a set of shares. OrderTo learn moredorado-uno.ru](https://an.yandex.ru/count/B3DPTxeQwCC50D41CMUyari00000ECgs7402I09Wl0Xe172AffoG1e01xCtW6OW1rBMagJMG0VZAnCelc07gZ-3LBg01zih4oY-e0UYEuDKkk07st_Rr6S010jW1ieMB5U01qBto4UW1OFW1elhUlW680WIW0g2-sHQv0cgA69ES6MQoy0ATlwU51_W2We20W82W5803xuV0-WM80vs3l_qAi0C4w0J70lW4XQq1Y0M5hG6G1RNn0w05zJEe1Vyhi0N_AxW5ktx01R3x3CW5yAK3q0Nb4-05Ml050PW6_f3f6Q06xWAe1ku2oGR6AvDQIO1r4z46nh_8qKc0THFP1W00071n0000gGVuKR0P49F4IB07W82G3D070k07XWhu1m60207G2BgAW870a802u0YGtwmBW0e1mGe00000003mFzWA0k0AW8bw-0h0_1M82mYg2n1cw2OD-5400DYZaWLRTGK0m0k0emN82u3Kam7P2_XHi1aGayH8w0k5hG7m2mQ83ABwthu1w0m2c0tylMRo3G3w3G223W293W0000000F0_a0x0X3sO3iRNpDILYU2w4w0Em8Gzi0u1eGy00000003mFwWF-AEnYlJXfE98?stat-id=4&test-tag=89060471234561&format-type=24&banner-test-tags=eyI2Mzg2Mzg5NjkyIjoiODkwNjA0NDE4ODI2MjQifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)

In the **previous part of** our lesson, we set up the HC-05 module to work in the master mode, and also configured the controller, to which we will connect this module, to work with the display.

Until we connect the module, and configure our display. We will draw the output of the lines that come to us from the module into the bottom line of the display, while the upper lines will be shifted. At first I had a plan to read the memory from the display and then shift the lines. but I did not manage to find this memory, unfortunately, through the adapter, so I decided to store all the three lower lines in the array for 60 characters. I think there is not much memory for memory, and it will work much faster in memory than reading it via I2C from the display. And the top line to us to store to what, as it all the same will replace the bottom. So let's go to the **lcd.c** file and add this array

uint8\_t portlcd; //ячейка для хранения данных порта микросхемы расширения

**char str2[60]={0};**

Fill it with spaces in the initialization function **LCD\_ini**

setwrite();//запись

**//Запишем пробелы в строковый массив**

**memset((void\*)str2,(uint8\_t)' ',60);**

Above the initialization function, we add the function of outputting a line down the display

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

**void LCD\_StrBottom(char\* st)**

**{**

**uint8\_t i=0,j=0,str\_cnt=0;**

**}**

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

And start writing her body. Since we agreed that we store the bottom three lines, we will now derive them in the upper three rows

uint8\_t i=0,j=0,str\_cnt=0;

//Выведем ранее заготовленные верхние три строки на дисплей

for(j=0;j<3;j++)

**{**

**LCD\_SetPos(0,j);**

**for(i=0;i<20;i++)**

**{**

**sendbyte(str2[j\*20+i],1);**

**}**

**}**

Next, we'll output the bottom line from the incoming array

    sendbyte(str2[j\*20+i],1);

  }

}

**//выведем последнюю строку**

**LCD\_SetPos(0,3);**

**while((st[str\_cnt]!=0)&&(st[str\_cnt]!=0x0D)&&(st[str\_cnt]!=0x0A))**

**{**

**sendbyte(st[str\_cnt],1);**

**str\_cnt++;**

**}**

The rest of the line in the display will be filled with spaces, since there may remain characters from the line that was already there

  str\_cnt++;

}

**//остальное - пробелы**

**for(i=str\_cnt;i<20;i++)**

**{**

**sendbyte((uint8\_t)' ',1);**

**}**

Next, we must store the bottom three rows in the array the next time. First, in the array, move the bottom two lines up

  sendbyte((uint8\_t)' ',1);

**}**

**//сдвигаем 2 нижние строки вверх в массиве**

**for(i=0;i<2;i++){**

**memcpy((void\*)(str2+i\*20),(void\*)(str2+i\*20+20),20);**

**}**

And then we copy the remaining line to the third (lower) part of the array, and also fill the remaining bytes with spaces

    memcpy((void\*)(str2+i\*20),(void\*)(str2+i\*20+20),20);

  }

**//скопируем выводимую строку в последнюю (третью строку массива)**

**memcpy((void\*)(str2+40),(void\*)st,str\_cnt);**

**//оставшуюся часть строки заполним пробелами**

**memset((void\*)(str2+40+str\_cnt),(uint8\_t)' ',20-str\_cnt);**

}

Let's create a prototype for this function in the header file and test our function in **main ()** , Changing there the corresponding code

HAL\_UART\_Receive\_IT(&huart1,(uint8\_t\*)str1,1);

sprintf(str1,"String 1");

**LCD\_StrBottom**(str1);

sprintf(str1,"String 2");

**LCD\_StrBottom**(str1);

sprintf(str1,"String 3");

**LCD\_StrBottom**(str1);

sprintf(str1,"String 4");

**LCD\_StrBottom**(str1);

/\* USER CODE END 2 \*/

As we see on the display screen - our code works



Now we connect our module to the debugger board, also looking at the drawing from the board's manual and in the Cube MX (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2017/11/image15.jpg)

The code we added for the test in main () can be deleted. Now let's go to the USART to catch the lines that come to us from the slave device. Add a global structure and a variable of its type

char str1[21];

**typedef struct USART\_prop{**

**uint8\_t usart\_buf[23];**

**uint8\_t usart\_cnt;**

**uint8\_t is\_tcp\_connect;//статус попытки создать соединение TCP с сервером**

**uint8\_t is\_text;//статус попытки передать текст серверу**

**} USART\_prop\_ptr;**

**USART\_prop\_ptr usartprop;**

We add the function of parsing a line, in which today we simply send the incoming string to the display

/\* USER CODE BEGIN 0 \*/

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

**void string\_parse(char\* buf\_str)**

**{**

**LCD\_StrBottom(buf\_str);**

**}**

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

/\* USER CODE END 0 \*/

And we will also add the familiar function of processing bytes that came by USART

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

**void UART1\_RxCpltCallback(void)**

**{**

**uint8\_t b;**

**b = str1[0];**

**//если вдруг случайно превысим длину буфера**

**if (usartprop.usart\_cnt>22)**

**{**

**usartprop.usart\_cnt=0;**

**HAL\_UART\_Receive\_IT(&huart1,(uint8\_t\*)str1,1);**

**return;**

**}**

**usartprop.usart\_buf[usartprop.usart\_cnt] = b;**

**if(b==0x0A)**

**{**

**usartprop.usart\_buf[usartprop.usart\_cnt+1]=0;**

**string\_parse((char\*)usartprop.usart\_buf);**

**usartprop.usart\_cnt=0;**

**HAL\_UART\_Receive\_IT(&huart1,(uint8\_t\*)str1,1);**

**return;**

**}**

**usartprop.usart\_cnt++;**

**HAL\_UART\_Receive\_IT(&huart1,(uint8\_t\*)str1,1);**

**}**

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

And also add an interrupt handler from USART

/\* USER CODE BEGIN 4 \*/

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

**void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart)**

**{**

**if(huart==&huart1)**

**{**

**UART1\_RxCpltCallback();**

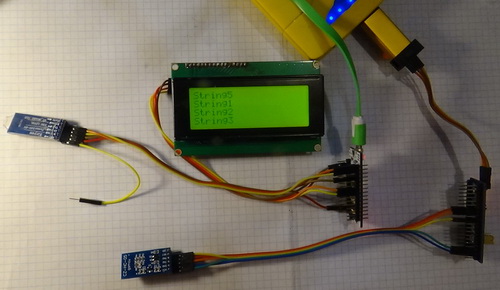
**}**

**}**

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

/\* USER CODE END 4 \*/

We turn on our scheme, we will also turn on the circuit with the slave module. We will collect the code. We will sew the controller. And, as soon as the modules are connected, we will begin to receive the lines here in this way (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2017/11/image16.jpg)

Thus, in this lesson we managed to configure the bluetooth module HC-05 in the master mode, also connect it to the controller, connect two modules connected by air to each controller, and transmit data via bluetooth using the data of the modules from one microcontroller to another and display them on the display. Now we can safely connect to the slave controller, for example, any sensors and transmit via bluetooth data to the slave controller.

Thank you all for your attention! Wait for the next lessons!