**STM Lesson 97. Connect the bluetooth module HC-05**

Posted on [November 27, 2017](http://narodstream.ru/stm-urok-97-podklyuchaem-bluetooth-modul-hc-05/)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-97-podklyuchaem-bluetooth-modul-hc-05/#respond)

[Beautiful strategy. Without downloadingSet up space, create a ship and conquer the game universe!To learn morexcraft.ru](https://an.yandex.ru/count/AdfZ3OtlFpa50ES1CVYxari00000ECgs7402I09Wl0Xe172yqvon0O01w8B8a0U80RBoavGqa07OyvlxBfW1rCNVfosW0Thpc_ikg06GmD-dBRW1iFVAp1d00GBO0UBzqXJW0Oo5Y1Je0HJu0G680WEW0j2jYWgv0axVGatajfDXo0ACdjtp-SZvfTu1y0A0uFoC0VW2We20W82G4u03kV_baGI80_sdujm8c0FIfWAe0nAm0mIu1Fy1w0I60lW4tvSHY0NVbn6G1UAY5w05bwe6g0MNgWQm1PUg1hW5ZSa8m0Mye0p81TV11D05rVi1u0K9y0K1c0QqZ8Br3Q06xWAe1ku2oGR6AvDQIO1r4z46nh_8qKc0THFP1W00BN1n0000gGVuKO06hdZ4IB07W82G3D070k07a0tu1m60207G2BgAW870a802u0ZebeK5W0e1mGgqOEWlvcB-FzWA0k0AW8bwY0i6gWiGoStdglXH00017_h-MdK50C0BWAC5o0k0r9C1sGluKO06hdZ4IEWBtvSHy0i6Y0puuTw-0UWC0fWDbOneyWq0-Wq0WWu0YGu00000003mFv0Em8Gzc0x6rypKbOdWkXEW3i24FR0E0Q4F00000000y3-e3_YZqUFIbPxYI000?stat-id=3&test-tag=84112668909569&format-type=24&banner-test-tags=eyI2MzAzMjE4OTI0IjoiODQxMTI2Mzk1NTc2MzIifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)18+

[Programming for beginnersDownload the free book and become a real programmer. Get it now:To learn moresheremetev.info](https://an.yandex.ru/count/MJHezaDWM1y50Dm1CVcxari00000ECgs7402I09Wl0Xe172WmvR10O01vE3CgmU80VY6tP0fa06YakID99W1Zhs-tYEW0VwIp8qag06M_w_U8xW1XgMPz1N00GBO0PQmhH3W0UQKqmxe0HRu0RQLthu1Y084e0BUkAe2kG9Etq9DvBQJOV02iQ-XoGNu0eA0W820a1E00vIAauqFY0EuWfgt0vW3YSK4i0C4k0J_0UW4tWNu1AYQ0eW5g9e2a0MPemAW1TbKg0NZHx05uqUu1OHAm0NmcGV81R3g0T05fY3W1Jxm1G6O1e3GhFCEe0Rk0gW6xWB91iOharf9W7KJqGR6lyZHIO1r4za60000S740002f1_XHCFBSUCH8i0U0W90Cq0S2u0U62lW70O080T08keg0WS2GW0BW2A-SbG602W712W0000000F0_s0e2u0g0YNhu2i3y5OWB2AeB47fL2m_uKG00uTa5_rfr1G302u2Z1SWBWDIJ0TaB-54myjnun4Ze2wYQ0l0B1eWCjfNUlW7e30AO3VozPl8D0FeD088E08aE00000000y3-G3i24FPWEnjVCr9M9uBeJe0x0X3sm3W6X3m0000000F0_g0_uez6vlfkUuaW0?stat-id=4&test-tag=84112668909569&format-type=24&banner-test-tags=eyI0ODQ2MjkwMDY1IjoiODQxMTI2Mzk1NTc2MzIifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)

It's time to take a little break from data transfer via the LAN interface.

Today we will try to transfer data from the microcontroller somewhere, and also back, using already wireless technologies, that is, technologies that make data transmission happen without using any wires.

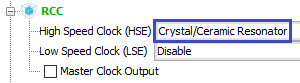
As a technology for wireless data transmission, we will use Bluetooth. And as a device for this we will take the module HC-05, the review and testing of which is [**here**](http://narodstream.ru/bluetooth_modul_hc_05/) . Be sure to review this review and perform all the tests described in it, also install the program on the smartphone that was presented in this review, and then begin to study the connection of the module to the controller. That is, the project of today's lesson will work normally on the configured module.

We will connect this module to the **STM32F103C8T6** controller , which is located on an inexpensive debugger board, which we already know well thanks to the many-sided lessons that we reviewed and read.

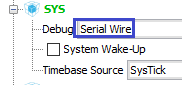
First let's create and configure the project.

Create a new project in the  **Cube MX** by selecting the **STM32F103C8Tx** controller  .

Let us set the clock from the quartz resonator



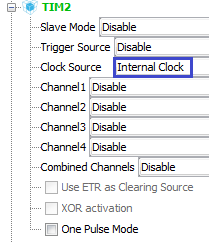
Turn on the debugger



Set up USART. We will use USART2 for the module, since the first one will be useful for us to connect the USB-TTL adapter for PC communication

image00

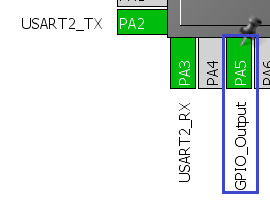
Turn on timer 2



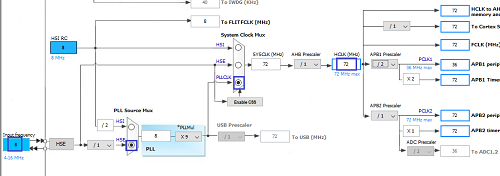
We plug in the leg of the PC13 port, which is responsible for the LED, to the output

Image04

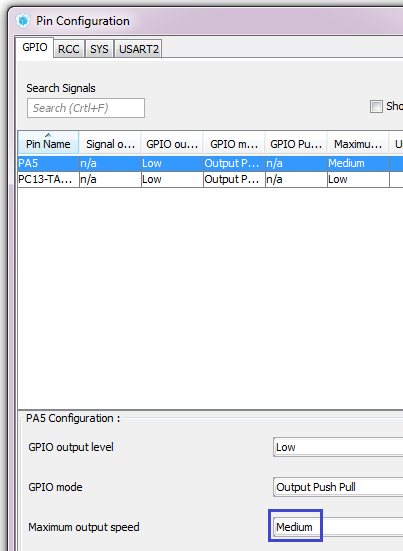
Also on the output we turn on the foot PA5, by means of which we will turn on the module support for AT commands, the rule is not yet in this lesson



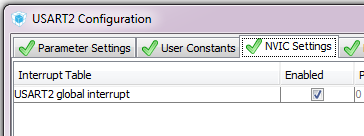
Let's adjust the frequencies in **System Configuration** (click on the picture to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2017/09/Image06.png)

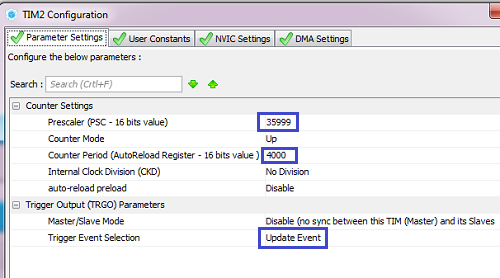
Now go to the **Configuration** section , USART will not be touched, we will leave the default settings (speed 115200 kbps), but the PA5 port foot will be adjusted slightly faster



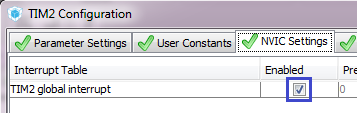
Nevertheless, in the USART settings we have to go for, in order to enable interrupts, we will also use it on the input



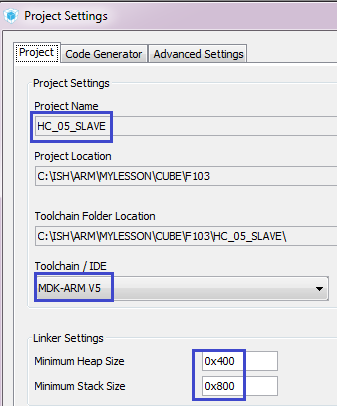
Set the timer for about 2 seconds



Also, enable the interrupt in the timer



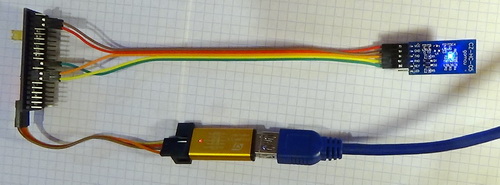
We will configure the project, increasing the size of the stack and heap in two (for the future) and selecting Keil as the development environment, and also call our project the corresponding topic of the occupation by the name



Apply the settings, collect the project, open it in **Keil** , configure the programmer for auto-cutting, and reduce the optimization level to level 1.

Let's try to assemble the project.

Let's look at the diagram of connecting the module to the controller (click on the picture to enlarge the image)

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

Let's return to the project and add to the **main.c** library for working with strings

/\* USER CODE BEGIN Includes \*/

**#include <string.h>**

/\* USER CODE END Includes \*/

We will also add a global string array

/\* USER CODE BEGIN PV \*/

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

**char str1[60]={0};**

/\* USER CODE END PV \*/

In **main () we** initiate the reception of data in USART, and also run our timer

/\* USER CODE BEGIN 2 \*/

**HAL\_UART\_Receive\_IT(&huart2,(uint8\_t\*)str1,1);**

**HAL\_TIM\_Base\_Start\_IT(&htim2);**

/\* USER CODE END 2 \*/

Add a global structure for USART properties, as well as a variable of its type

char str1[60]={0};

**typedef struct USART\_prop{**

**uint8\_t usart\_buf[60];**

**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;**

Also add a global variable for the counter output to the module strings, as well as an array of these same rows

USART\_prop\_ptr usartprop;

**uint8\_t i=0;**

**char \*str2[] =**

**{**

**"String1\r\n",**

**"String2\r\n",**

**"String3\r\n",**

**"String4\r\n",**

**"String5\r\n"**

**};**

We add also the function of parsing a string. While it will be almost empty. It will be filled in later studies, but in the meantime it will only change the state of the LED's glow, as well as the echo of the line to the source that sent it, that is, in our case, the smartphone by sending it to USART

/\* USER CODE BEGIN 0 \*/

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

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

**{**

**HAL\_UART\_Transmit(&huart2,(uint8\_t\*)buf\_str,strlen(buf\_str),0x1000);**

**HAL\_GPIO\_TogglePin(GPIOC, GPIO\_PIN\_13);**

**}**

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

/\* USER CODE END 0 \*/

Now add the handler for the character from USART

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

**void UART2\_RxCpltCallback(void)**

**{**

**uint8\_t b;**

**b = str1[0];**

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

**if (usartprop.usart\_cnt>59)**

**{**

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

**HAL\_UART\_Receive\_IT(&huart2,(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(&huart2,(uint8\_t\*)str1,1);**

**return;**

**}**

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

**HAL\_UART\_Receive\_IT(&huart2,(uint8\_t\*)str1,1);**

**}**

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

/\* USER CODE END 0 \*/

This handler takes a character from the USART port, writes it to the buffer at the address in the counter, checks for the newline, if any, then the reception stops, the string typed in the buffer is passed to the parse function and the counter is reset if another character , the counter is incremented. Also along with this, the function included a check code for exceeding the buffer size string.

Let's add now official reception handler from USART, in which we will call our "self-made" handler

/\* USER CODE BEGIN 4 \*/

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

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

**{**

**if(huart==&huart2)**

**{**

**UART2\_RxCpltCallback();**

**}**

**}**

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

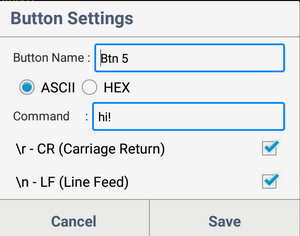
/\* USER CODE END 4 \*/

We will collect the code, we will sew the controller and check the code in the terminal program on the smartphone.

In order not to suffer and constantly not to enter lines, the program has five buttons, in the handlers of which you can add any lines. These buttons

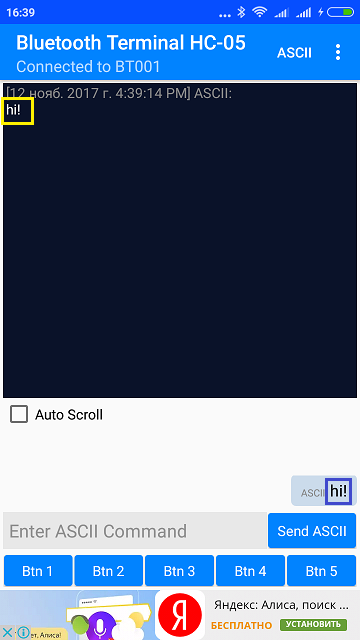
image09

Let's assign a string to the fifth button, for example, it will be sent to bluetooth by pressing. We can enter the settings by clicking on the button



We can configure several parameters here. Change the name of the button (the name displayed on the button itself, as well as the data type, the line itself, as well as whether the line ends with carriage return codes and line feeds.) We'll set everything up as in the screenshot and save the settings.Now by clicking on our button the line whose text we are added to the Command field, it will be sent to the port, of course, there will be a logical question: "How do I go back into the settings and reconfigure the button properties?" Answer: "by long pressing the button (long tap)"

Click on the button and the line from the button will go to the chat room, and also get into the chat room. And in the chat room, it will get thanks to the echo in the function of parsing the line. Thus, we checked that the line is sent to the microcontroller and sent back from the microcontroller to the smartphone via the connected module HC-05



Now let's add an event handler from the timer, thanks to which rows from the array will be sent to the smartphone

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

**void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)**

**{**

**if(htim==&htim2)**

**{**

**HAL\_GPIO\_TogglePin(GPIOC, GPIO\_PIN\_13);**

**HAL\_UART\_Transmit(&huart2,(uint8\_t\*)str2[i],strlen(str2[i]),0x1000);**

**i++;**

**if(i>4)**

**{**

**i=0;**

**}**

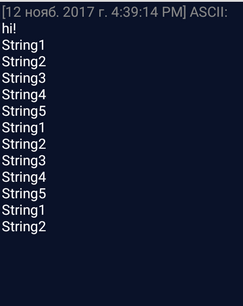
**}**

**}**

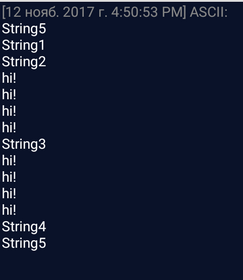
**//-----------------------------------------------**

/\* USER CODE END 4 \*/

We will collect the code once again and sew the controller. After this line, approximately once every 2 seconds from our controller via the bluetooth interface will come to the window of the terminal program on the smartphone



Excellent! Lines are coming, and for a period of 2 seconds the controller can perform any operations due to the use of an interrupt from the timer. check this by clicking on our button with a string in the terminal program of the smartphone. If everything is normal, then the lines sent by the button should echo almost instantly, without waiting for the output of the next line from the MK



So it is, we can send as many data as you like and they do not go to any queue, the work is instant.

Thus, for such a short lesson we learned how to connect the **HC-05** bluetooth module to the microcontroller, to receive and send data from the controller and to the controller to other devices supporting bluetooth, in particular, a smartphone.

Thank you all for your attention!