**STM Lesson 127. LAN8742A. LWIP. NETCONN. HTTP Server**

Posted on [August 7, 2018](http://narodstream.ru/stm-urok-127-lan8742a-lwip-netconn-http-server/)by [http://1.gravatar.com/avatar/4824b24065500834db4b9f331b608833?s=32&d=mm&r=gNarod Stream](http://narodstream.ru/author/admin/) Posted in [FreeRTOS](http://narodstream.ru/freertos/) , [LAN](http://narodstream.ru/lan/) , [Programming STM32](http://narodstream.ru/rub_stm32/)- [No Comments ↓](http://narodstream.ru/stm-urok-127-lan8742a-lwip-netconn-http-server/#respond)

[Automation in Excel to orderAutomate routine operations in Excel.**Programming** in **VBA** , Excel macrosTo learn moredipmasters.ru](https://an.yandex.ru/count/G3VPk4w3MKS50Da1CUx1ari00000ECgs7402I09Wl0Xe172yyRwI0u01lhpkc0A80SEhsgeja07mojRaAfW1fl_PkYYW0U2jvTCfg070vUYn9xW1f9dZlXV00GBO0TJsW1BW0QAnpmFe0J3u0PAIthu1Y083e0BesLUv0eeiYbom0YWTy0Bvp_Am1FW2We20W82W5803hjdA_WY80yFxqlS2c0FDemke0nAm0mJe19O5-0IhvWA81Qlc0f05reO4e0N0QwW5ZLkm1OrRk0NkWm701Rk03CW5q8O3q0N_AE05Ml050PW6wEM07w06xWAe1ku2oGR6AvDQIO1r4z46nh_8qKc0THFP1W00071n0000gGVuKK1rV-J5IB07W82G3D070k07XWhu1m60207G2BgAX860a802u0Zqx2k02W712W0000000F0_s0e2u0g0YNhu2i3y5OWB1geB4084fdpuKG00gNhkMLjr1G302u2Z1SWBWDIJ0TaB-550TN_anKZe2wlc0l0B1eWCafBUlW7e30AO3VYSPV8D0FeD088E08aE00000000y3-G3i24FPWEnjVCr9M9uBeJe0x0X3sm3W6X3m0000000F0_g0_uex7zw8_vuaW0?stat-id=3&test-tag=89060471234561&format-type=24&banner-test-tags=eyI1NzQyNzExNDgwIjoiODkwNjA0NDE4ODI2MjQifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)18+

[Help Desk from 79 000 rubles a year!No restrictions on the number of users. Test for free!To learn morevsdesk.ru](https://an.yandex.ru/count/KMufb4m_asu50Cy1CU_1ari00000ECgs7402I09Wl0Xe173asxZP3O01tnY80OwZtgKqa07ky_A3BvW1pDYClYsu0S22jz4Pm042s076ZjeKu06KhP8Dw06G0lW1tg3UlW680WIW0k3SaH6v0eeiYbom0YWTy0AZav7R0lW2We20W82W5803wjN3FOW3iUcUsmkO0_M-1R031BW4_m7e1Au1-0IXjGU81Q6r1v05oRGDe0NZd0Ae1Qln0R05g_41k0Mrj0F01R219CW5y9C9q0NCRE05Fl050PW6WD2iymwW1ku2g0Rk0ia6nYkJMac0THFH1iQ_oD59W7KJsGO0001mSG000Aa7-57ST9_anKYm1u20a0pG1mBW1uOA-0S1W0W1q0YQYe21m9200k08iEx8280A0S4A00000000y3_O2WBW2e29UlWAmFmLY0i8gWiGDgtsKFXH002nUlPPMtK50C0BWAC5o0k0r9C1sGluKTnqd-J5IEWBeRK7y0i6Y0pUeDw-0UWC0fWDoR9dyWq0-Wq0WWu0YGu00000003mFv0Em8Gzc0x6rypKbOdWkXEW3i24FR0E0Q4F00000000y3-e3_YZyUpea_dYI000?stat-id=4&test-tag=89060471234561&format-type=24&banner-test-tags=eyI2MzEyMzIwMjQ3IjoiODkwNjA0NDE4NDk4NTYifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)18+

We continue to work with the LWIP protocol stack, as well as with its NETCONN interface.

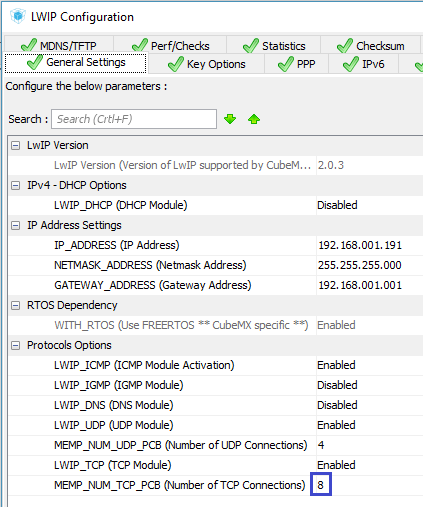
And today we will start working with the protocol already applied - HTTP. This protocol is higher than the TCP protocol, and the TCP protocol is for the HTTP protocol the transport layer protocol. We are already familiar with all this, since we have been working with the HTTP protocol for a long time and it makes no sense to delve into its features, as well as to how much it is claimed in the current age of the Internet.

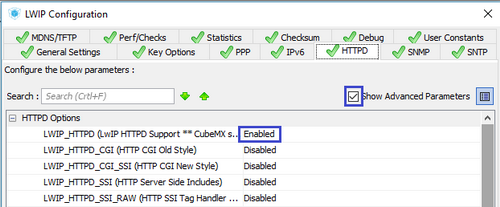
Therefore, we turn right to the point.

In studying the use of the HTTP protocol in the work plan with the NETCONN interface, we will also go in a consistent way. First we will work with HTML pages without using any additional technologies, and in the future lessons we will try to improve our project and try to transfer data to the browser from our server without reloading the entire page. What we will use for this technology - we learn in these lessons.

And now create a project from the [**lesson**](http://narodstream.ru/stm-urok-124-lan8742a-lwip-netconn-tcp-server/) project [**124**](http://narodstream.ru/stm-urok-124-lan8742a-lwip-netconn-tcp-server/) with the name **LAN8742\_TCP\_SERVER\_NETCONN** and assign it the name **LAN8742\_HTTP\_SERVER\_NETCONN**.

Open the project in the Cube MX, open the **LWIP** settings  , go there to the **General Settings** section and add a little the number of simultaneously open TCP connections

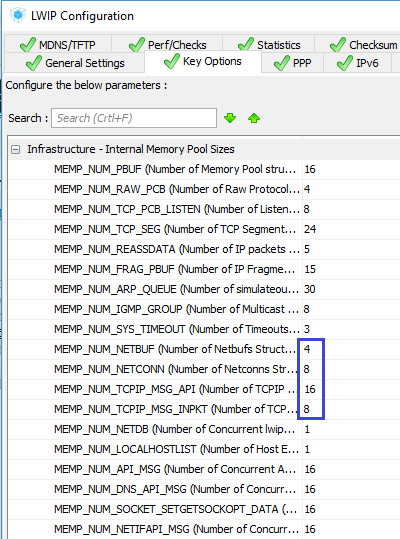


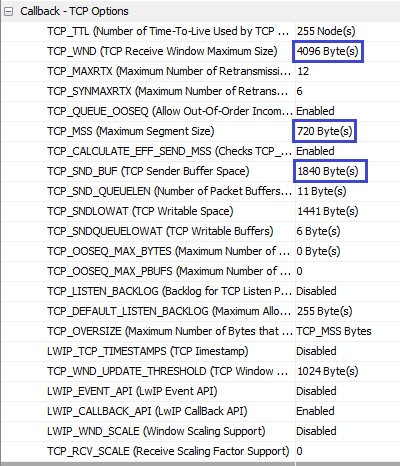


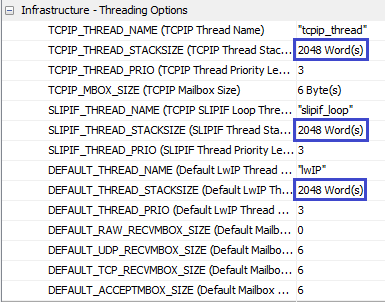
Make sure that the **Show Advanced Parameters** check box is checked.

Nothing in this section does not touch, we will leave everything by default.

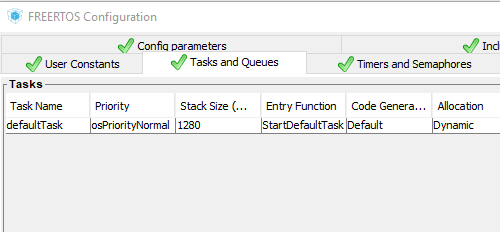
Next we go to the Key Options section and we will correct something here.







Go to the settings **FREERTOS** in the **Tasks and Queues** section and remove the second task from there, we still do not use it. There will be only one default task



Generate the project for  **System Workbench**  and open it there. Set the optimization level to  **1** , remove it with debugging settings and comment out the lines unknown to the compiler in the **main.c** file  .

Let's fix the header in  **main ()**



|  |  |
| --- | --- |
| 1 | TFT\_DisplayString(0, 10, (uint8\_t \*)"[php theme="twilight"]HTTP[/php] Server", CENTER\_MODE); |

PHP



|  |  |
| --- | --- |
| 1 | your code |



|  |  |
| --- | --- |
| 1 | your code |

We will try to assemble the project, only it will not be collected from us and we will get an error on the absence of the file **fsdata.c** . From [**lesson 102,**](http://narodstream.ru/stm-urok-102-lan8720-lwip-http-raw-chast-1/) we remember how to deal with this, so with the help of **makefsdata we will**generate this file. But, before we generate it, we will have to put something in the folder **fs** , that is, prepare the page file.

You can put any page there, only it is desirable that the name of its file be **index.html** . I created the page myself, as usual. I took the technical documentation of our debug board and copied from it some of the text in the CMS WordPress editor on the virtual site. The size is about 15 kilobytes, then what you need. The file is not too large or small, it will take more than one TCP packet to transmit it, so we will check how our server fragmentes the packets.

We will also create a file that we will give to the client, if we do not find the requested document. Let's call it 404.html. This file will have the following contents

XHTML



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>    <head>      <title>STM32F746G-DISCO</title>    </head>    <body style="color: black; background-color: white;">      <h2>404 - Page not found</h2>      <p><span style="font-family: Times New Roman,Times,serif;"> Sorry,      the page you are requesting was not found on this server.</span> </p>  </body></html> |

By default, the **makefsdata** utility  generates a header in the HTTP 1.0 format. In order to get HTTP 1.1, we need to use the special command-line parameter of the utility.

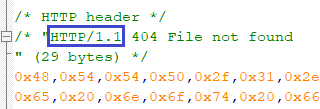
And in order to prevent us from entering the file name and parameters all the time, create a batch file in the folder with the utility named **makefsdata.cmd** next

**makefsdata.exe -11**

Save and run the file  **makefsdata.cmd** and get the file **fsdata.c** .

Now when changing the contents of the folder, we will run the file **makefsdata.cmd** .

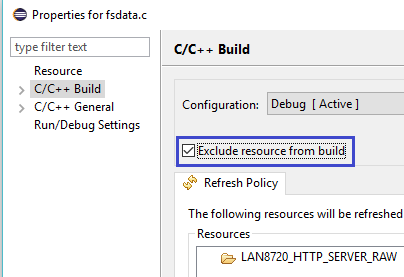
If we look at the contents of the file fsdata.c, then we see that in the header we have the HTTP 1.1 protocol



We copy the file fsdata.c, as well as the utility along with the command file and the fs folder with all the contents and, if necessary, the file  **msvcr100d.dll**  in the project folder under the path **"** Project **folder // Middlewares / Third\_Party / LwIP / src / apps / httpd / "** . Now we will collect the files there so as not to copy the file every time.

Let's return to the project, we'll do **Refresh** .

Then find the fsdata.c file in the project tree, call the context menu on it, right-click on it and enter its properties. Select  **C / C ++ Build**  and check the box next to " **Exclude resource from build** "



Now the project is going to be fine.

Connect the main.c file to the virtual file system

C



|  |  |
| --- | --- |
| 1  2 | #include "lwip/api.h"  #include "lwip/apps/fs.h" |

In the default task function **StartDefaultTask,** we can comment out the creation of another task. We have enough and one



|  |  |
| --- | --- |
| 1  2 | sys\_thread\_new("tcp\_thread1", tcp\_thread, (void\*)&sock01, DEFAULT\_THREAD\_STACKSIZE, osPriorityNormal );  //sys\_thread\_new("tcp\_thread2", tcp\_thread, (void\*)&sock02, DEFAULT\_THREAD\_STACKSIZE, osPriorityNormal ); |

We delete in this function also the initialization of the altitude parameter, and in the other a little vertical coordinate will be raised

sock01.y\_pos = **45**;

~~sock02.y\_pos = 180;~~

Initialization of the second parameter is also deleted

~~sock02.conn = conn;~~

Let's **go** to the TCP connection **task tcp\_thread** and declare there a variable of this type for the file structure

char\* buf;

**struct fs\_file file;**

After creating a connection with the client in the condition, we delete the entire body. After that, in an infinite task cycle, we will have only this code

for(;;)

{

  err = netconn\_accept(conn, &newconn);

  if (err == *ERR\_OK*)

  {

  }

  else

  {

    osDelay(1);

  }

}

If the connection with the client passed normally, then we will try to accept the package

err = netconn\_accept(conn, &newconn);

if (err == *ERR\_OK*)

{

**recv\_err = netconn\_recv(newconn, &inbuf);**

If the packet is normally received, then we will write its contents into the buffer, and then free the memory of the buffer structure, also the connection structure, and close the connection with the client

C



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | recv\_err = netconn\_recv(newconn, &inbuf);  if (recv\_err == ERR\_OK)  {    if (netconn\_err(newconn) == ERR\_OK)    {      netbuf\_data(inbuf, (void\*\*)&buf, &buflen);    }  }  netconn\_close(newconn);  netbuf\_delete(inbuf);  netconn\_delete(newconn); |

Now let's go into the body of the condition and work there with the accepted line.

If the string is a HTTP protocol GET request, then clear the position 40 pixels below our line. We will find out why it is needed later. And also write "Connect" at the very bottom of the display

netbuf\_data(inbuf, (void\*\*)&buf, &buflen);

**if ((buflen >=5) && (strncmp(buf, "GET /", 5) == 0))**

**{**

**qstruct = osMailAlloc(strout\_Queue, osWaitForever);**

**qstruct->y\_pos = 250;**

**sprintf(qstruct->str,"%-20s", "Connect");**

**osMailPut(strout\_Queue, qstruct);**

**osDelay(1);**

**qstruct->y\_pos = arg\_sock->y\_pos + 40;**

**sprintf(qstruct->str,"%-20s", " ");**

**osMailPut(strout\_Queue, qstruct);**

**osDelay(1);**

**}**

If the client has addressed only to the address to the server, or he explicitly specified the main page file, then we will give it to him

osDelay(1);

**if ((strncmp((char const \*)buf,"GET / ",6)==0)||(strncmp((char const \*)buf,"GET /index.html",15)==0))**

**{**

**fs\_open(&file, "/index.html");**

**netconn\_write(newconn, (const unsigned char\*)(file.data), (size\_t)file.len, NETCONN\_NOCOPY);**

**fs\_close(&file);**

**}**

And if you asked for something else, which we do not have on the server, then we will give him the page 404.html with an error, and also write a corresponding message on the display screen 40 pixels below the query line

  fs\_close(&file);

}

**else**

**{**

**/\* Load Error page \*/**

**fs\_open(&file, "/404.html");**

**netconn\_write(newconn, (const unsigned char\*)(file.data), (size\_t)file.len, NETCONN\_NOCOPY);**

**fs\_close(&file);**

**qstruct->y\_pos = arg\_sock->y\_pos + 40;**

**sprintf(qstruct->str,"%-20s", "file not found");**

**osMailPut(strout\_Queue, qstruct);**

**osDelay(1);**

**}**

We display the HTTP request line on the display, just in case a little cropping, if it is more than 20 characters, and also free the memory of the structure of the queue

  osDelay(1);

}

**if ((buflen >=20)) buf[20] = 0;**

**else buf[buflen] = 0;**

**qstruct->y\_pos = arg\_sock->y\_pos;**

**sprintf(qstruct->str,"%-20s", buf);**

**osMailPut(strout\_Queue, qstruct);**

**osMailFree(strout\_Queue, qstruct);**

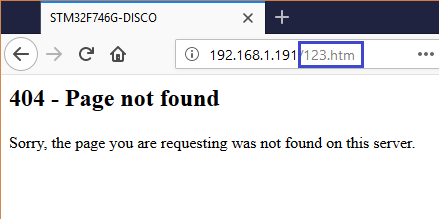
**osDelay(2);**

We will collect the project, we will impose the controller and we will try to enter in the browser the IP-address of the server



A similar picture will be obtained if we type 192.168.1.191/index.html in the address bar of the browser.

Now try to enter the name of a nonexistent document



In this case, we will receive an error notification using the page 404.html.

And on the display screen is this



Excellent! The server is running. Now let's check how the page with pictures will be displayed. That is, this is the case when additional GET requests arrive from the page.

For this we will illustrate our page a little. Adding pictures to it. To do this, we will create an additional **IMG** folder in the folder **fs** for convenience and copy some pictures there.

The file **index.html will be** duplicated with a different name, for example **index1.html** , so as not to spoil the original without pictures.

We will insert in a file of a picture approximately such here a method

**<p style="text-align: center;">**

**<img src="IMG/img01.jpg">**

**</p>**

We re-arrange the file fsdata.c, update the project tree, and also add a reaction to the request for the file **index1.html** in our task . About processing requests for pictures also do not forget

if ((strncmp((char const \*)buf,"GET / ",6)==0)||(strncmp((char const \*)buf,"GET /index.html",15)==0))

{

  ...

}

**else if (strncmp((char const \*)buf,"GET /index1.html",16)==0)**

**{**

**fs\_open(&file, "/index1.html");**

**netconn\_write(newconn, (const unsigned char\*)(file.data), (size\_t)file.len, NETCONN\_NOCOPY);**

**fs\_close(&file);**

**}**

**else if (strncmp((char const \*)buf,"GET /IMG/img01.jpg",18)==0)**

**{**

**fs\_open(&file, "/IMG/img01.jpg");**

**netconn\_write(newconn, (const unsigned char\*)(file.data), (size\_t)file.len, NETCONN\_NOCOPY);**

**fs\_close(&file);**

**}**

**else if (strncmp((char const \*)buf,"GET /IMG/img02.jpg",18)==0)**

**{**

**fs\_open(&file, "/IMG/img02.jpg");**

**netconn\_write(newconn, (const unsigned char\*)(file.data), (size\_t)file.len, NETCONN\_NOCOPY);**

**fs\_close(&file);**

**}**

**else if (strncmp((char const \*)buf,"GET /IMG/img03.jpg",18)==0)**

**{**

**fs\_open(&file, "/IMG/img03.jpg");**

**netconn\_write(newconn, (const unsigned char\*)(file.data), (size\_t)file.len, NETCONN\_NOCOPY);**

**fs\_close(&file);**

**}**

**else if (strncmp((char const \*)buf,"GET /IMG/img04.jpg",18)==0)**

**{**

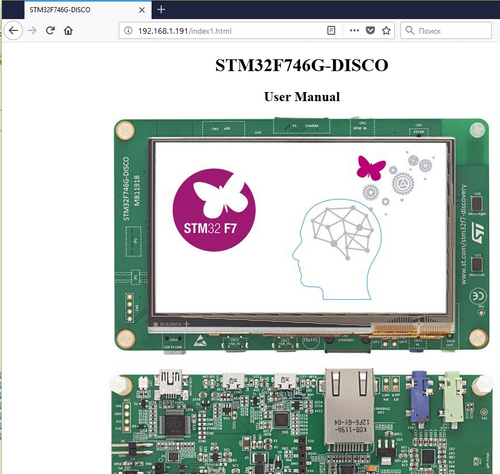
**fs\_open(&file, "/IMG/img04.jpg");**

**netconn\_write(newconn, (const unsigned char\*)(file.data), (size\_t)file.len, NETCONN\_NOCOPY);**

**fs\_close(&file);**

**}**

Now we will compile the code, we will tell the controller and try in the browser to request the file **index1.html**



All the pictures on the page are displayed.

So everything works great. I think that there is no need to analyze network traffic in the analyzer program. If you have such a desire, then you will do it yourself, as we constantly use this program and we do not need to teach it.

So, in this lesson we created a simple but fully functioning **HTTP server** using the **NETCONN** interface of the **LWIP** protocol **stack** .

Thank you all for attention!