**STM Lesson 109. NRF24L01. Transfer moisture and temperature from the sensor DHT22**

Posted on [February 23, 2018](http://narodstream.ru/stm-urok-109-nrf24l01-peredayom-vlazhnost-i-temperaturu-s-datchika-dht22/)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/)- [1 comment ↓](http://narodstream.ru/stm-urok-109-nrf24l01-peredayom-vlazhnost-i-temperaturu-s-datchika-dht22/#comments)

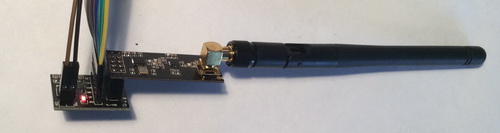
[Programming LessonsLearn all the in-house cooking **programming** training ! Get the book on the e-mail!To learn moresheremetev.info](https://an.yandex.ru/count/CtgNtI17XOG50Cq1CHM-ari00000ECgs7402I09Wl0Xe172WmvR10O01sE3CgmU80QE8tP0fa06cakID99W1eiM-tYEW0SAapOqag07iWR3U8xW1tAUPz1N00GBO0PQmhH3W0UQKqmxe0HRu0RQLthu1Y083e0BUkAe2kGAwH3FVpmT2TF02tucndWJu0eA0W820m1EO0wKVg0CIi0C4k0J_0UW4r0Fu19TyY0MNV905njq1e0MW9QW5oH-m1SaVk0MfEC05y9a7o0MmwW7G1PeEu0K_y0K1c0Q0qApp3g06xWAe1ku2oGR6AvDQIO1r4z46nh_8qKc0THFP1W00071n0000gGVuKTIHVld4IB07W82G3D070k07XWhu1m60207G2BgAW870a802u0Yld9K1W0e1mGe00000003mFzWA0k0AW8bw-0h0_1M82mQg2n2X7r1V-5400Fj1BXzRTGK0m0k0emN82u3Kam7P2_XHr95--SH8w0kNVF0B1eWCjfNUlW7e30AO3PMCQF8D0FeD088E08aE00000000y3-G3i24FPWEnjVCr9M9uBeJe0x0X3sm3W6X3m0000000F0_g0_uew68sw--uaW0?stat-id=3&test-tag=85761936351233&format-type=24&banner-test-tags=eyI0ODQ2MjkwMDY3IjoiODU3NjE5MDY5OTkyOTYifQ%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/KwOafnpTGRW50DW1CHQ-ari00000ECgs7402I09Wl0Xe1724kxc20O01nfYFzRESyguWY06-YR7-Av01j92OyYAO0UxNkFyYe06CvvxN0QW1gh7jhWAu0OYYjfC2m042s07gYUa2w04E-06s0OW21A02ykA82Ra2kaGptyy7GdJm0f-FjzK4-0A2W820WC0JW0EOdTdG0eW3WxVyxmgO0_7i0h031BW4_m7e19G3-0J-im681Vwp0P05dDi1e0N_DQW5uIsm1U4jk0NTDy05bu03o0NMOT05onJW1MBm1G6O1jxNq7IW1ku2g0Rk0ia6nYkJMac0THFH1iQ_oD59W7KJsGO0001mSG000907Aga7-578XB_vn4Ym1u20a0pG1mBW1uOA-0S1W0W1q0YwYe21m9200k08kR74180A0S4A00000000y3_O2WBW2e29UlWAmFmLY0i8gWiGLcO1SlXH002oVZuVMtK50C0BWAC5o0k0r9C1sGluKSY4l_d4IEWB_hC1y0i6Y0oiljw-0UWC0fWD_BrcyWq0-Wq0WWu0YGu00000003mFv0Em8Gzc0x6rypKbOdWkXEW3i24FR0E0Q4F00000000y3-e3_YZeP7hjxxYI000?stat-id=4&test-tag=85761936351233&format-type=24&banner-test-tags=eyI0NjgzMTM4MDc0IjoiODU3NjE5MDY5OTkyOTYifQ%3D%3D&)[Yandex.Direct](https://direct.yandex.ru/?partner)

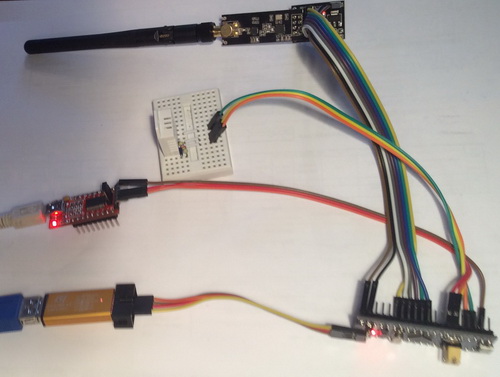
In [**lesson 105,**](http://narodstream.ru/stm-urok-105-nrf24l01-peredayom-dannye-chast-1/) we learned not only to read and write registers and buffers, but also to transfer data from the transmitter to the receiver, only this was the usual counter of some undefined ticks.

And today we will try to transmit any conscious data via wireless communication with the NRF24L01 modules. But since we have already transmitted the temperature from the DS18B20 sensor with Bluetooth, it will not be so interesting for us, let's give both humidity and temperature from the DHT22 sensor, with which we met in [**lesson 107**](http://narodstream.ru/stm-urok-107-datchik-vlazhnosti-i-temperatury-dht22-chast-1/) .

We will leave the diagrams the same as in the lesson 105, only first we will try to connect the NRF module on the transmitter not through the stabilizer that we had under the heat shrinkage, but through a specially designed adapter for our modules with a stabilizer (click on the picture for increase the image)

[](http://narodstream.ru/wp-content/uploads/2018/02/stm109img00.jpeg)

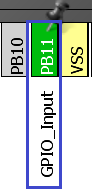
And also the difference from the scheme of the above lesson will be that we have to connect the sensor DHT22 to the circuit. After that, the transmitter circuit will take the following form (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2018/02/stm109img02.jpeg)

We connected the sensor in exactly the same way as in lesson 107 - the information input to the leg **PB11**.

First, let's deal with the transmitter. We will take the project from [**lesson 105**](http://narodstream.ru/stm-urok-105-nrf24l01-peredayom-dannye-chast-1/) intended for the transmitter, **NRF24\_TX** and create on its basis a project named **NRF24\_TX\_DHT22** .

Open our project in Cube MX and include the foot **PB11** in it



Generate the project for Keil, open it, install the programmer on atoret, enable the optimization level in **1**, connect the file **NRF24.c**  and try to assemble the project. If everything is going well, we'll deal with the temperature sensor. To do this, take from the project **lesson 107 DHT22** files **dht22.h** and **dht22.c** and copy them to the appropriate folders of our project. We connect the file **dht22.c** to the project tree and connect our library for the sensor in the main.c file

#include "NRF24.h"

**#include "dht22.h"**

Initialize the sensor in **main ()**

/\* USER CODE BEGIN 2 \*/

**port\_init();**

**dht22\_init();**

NRF24\_ini();

We will not send values ​​of temperature and humidity. We will send raw data from the buffer, into which the sensor will give its values ​​when reading

Therefore, we will do this in an infinite loop, where we will remove all the code in advance. Also we here let's deduce our raw values ​​read from the sensor into the terminal program. After all corrective procedures, the code in an infinite loop will take the following form

/\* USER CODE BEGIN 3 \*/

**if(dht22\_GetData(buf1))**

**{**

**sprintf(str1,"dt: %02X:%02X:%02X:%02X:%02X\r\n",buf1[4],buf1[3],buf1[2],buf1[1],buf1[0]);**

**HAL\_UART\_Transmit(&huart1,(uint8\_t\*)str1,strlen(str1),0x1000);**

**NRF24L01\_Send(buf1);**

**}**

**HAL\_Delay(2000);**

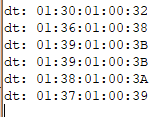
/\*

Also, we remove all our local variables from the **main ()** function , they will not be needed now

~~uint8\_t retr\_cnt, dt;~~

~~uint16\_t i=1,retr\_cnt\_full;~~

We will collect the code, we will tell the controller and see the result of our work while in the terminal program, since it will be unclear in the receiving indicator that we have not yet corrected the code of the receiver



Excellent! The sensor is working. Now let's get on with the receiver. To do this, we power the transmitter controller from an autonomous source, and the receiver board, on the contrary, connect to the PC.

The project for the receiver is also used from [lesson 105](http://narodstream.ru/stm-urok-105-nrf24l01-peredayom-dannye-chast-1/)**NRF24\_RX\_DHT22** . **Let's** call it **NRF24\_RX\_DHT22** .

Only files **max7219.c** and **max7219.h** in it will be **replaced by**  copying their project **DHT22**[**lesson 107**](http://narodstream.ru/stm-urok-107-datchik-vlazhnosti-i-temperatury-dht22-chast-1/), since there we added functions for displaying floating-point values ​​on the indicator.

Open the project in Cube MX and, without touching it, generate the project for Keil, open it, configure the programmer for auto-cutting, set the optimization level to **1** , connect the files **NRF24.c** and **max7219.c**to the project tree.

Let's try to assemble the project. Only it is unlikely to be collected, because we connected the eight-digit indicator to another controller. So go to the file **max7219.h** and fix the library connection there

#include "stm32f**4**xx\_hal.h"

Now go to the file  **max7219.c** and fix the connection of the SPI bus

extern SPI\_HandleTypeDef hspi**3**;

Also, we fix this in **Send\_7219**

HAL\_SPI\_Transmit (&hspi**3**, (uint8\_t\*)aTxBuf, 1, 5000);

aTxBuf[0]=dt;

HAL\_SPI\_Transmit (&hspi**3**, (uint8\_t\*)aTxBuf, 1, 5000);

Do not forget also about the Chip Select leg. It is also different here. If we do not fix it, then the project will be assembled, but the indicator will not work

#define cs\_set() HAL\_GPIO\_WritePin(GPIO**A**, GPIO\_PIN\_1**5**, GPIO\_PIN\_RESET)

#define cs\_reset() HAL\_GPIO\_WritePin(GPIO**A**, GPIO\_PIN\_1**5**, GPIO\_PIN\_SET)

Now the project will have to be assembled. If everything goes well, we go to the **NRF24.c** file and make some corrections to the function of receiving the **NRF24L01\_Receive** package , because here we will need to convert the raw bytes of the packet that came to the values ​​of temperature and humidity. We already did this in the lesson on the sensor DHT22, therefore, I think it will not be special.

We remove from this function the declaration of this local variable

~~uint16\_t dt=0;~~

Add to other local variables for humidity and temperature

uint8\_t status=0x01;

**float temper, hum;**

Remove from the function these lines

~~dt = \*(int16\_t\*)RX\_BUF;~~

~~Clear\_7219();~~

~~Number\_7219(dt);~~

~~dt = \*(int16\_t\*)(RX\_BUF+2);~~

~~NumberL\_7219(dt);~~

Instead, they will now have this code

**temper = (float)((\*(uint16\_t\*)(RX\_BUF+1)) & 0x7FFF) / 10;**

**if((\*(uint16\_t\*)(RX\_BUF+1)) & 0x8000) temper \*= -1.0;**

**hum = (float)(\*(int16\_t\*)(RX\_BUF+3)) / 10;**

**Clear\_7219();**

**NumberLF\_7219(hum);**

**NumberF\_7219(temper);**

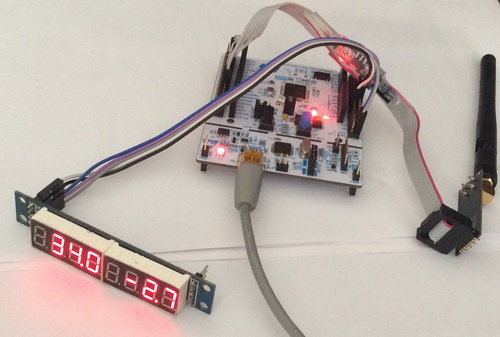
The code almost coincides with the code from the infinite loop of the main () function of the sensor tutorial, so it does not need an explanation.

We will collect the code, we will sew the controller and we will look at the result on the indicator of the receiver

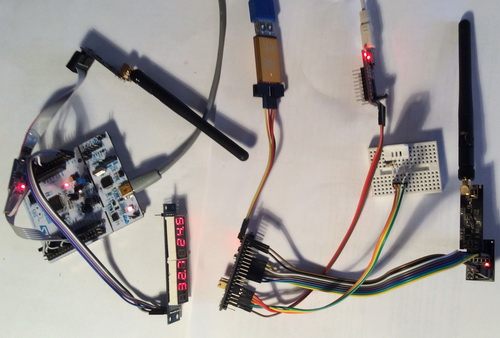


Excellent! Everything is transmitted and displayed correctly.

And of course I could not resist, took the sensor of the receiver to the street and measured the negative temperature (click on the picture to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2018/02/stm109img01.jpeg)

And here is the whole of our scheme now completely looks (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2018/02/stm109img03_1.jpeg)

Thus, in this lesson we learned how to transmit, by means of **NRF24L01** wireless data communication **modules,** not just lines or some non-useful data, but rather conscious data that may be required in practice - in our case, the humidity and temperature from the sensor DHT22 connected to the receiver circuit.

Thank you all for attention!