

STM32 and Ethernet. Part 5. Transport level. UDP protocol.



Aveal MT

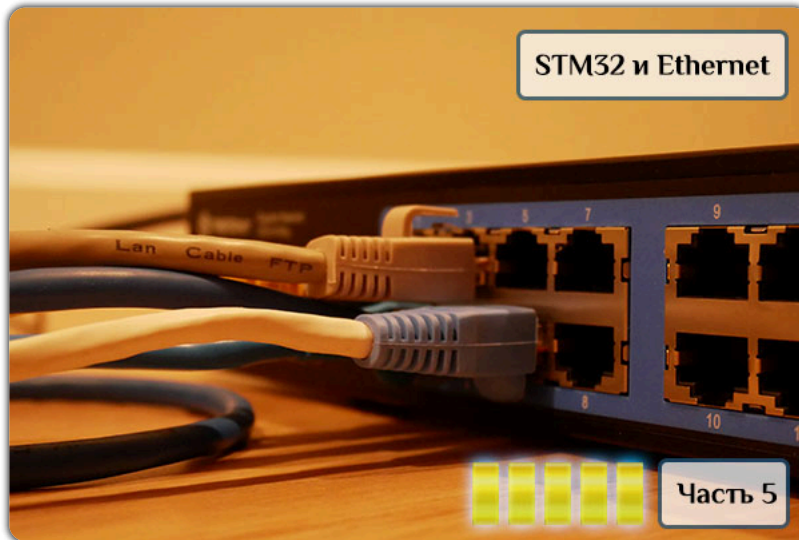
14/05/2023



13



STM32, STM32CubeMx



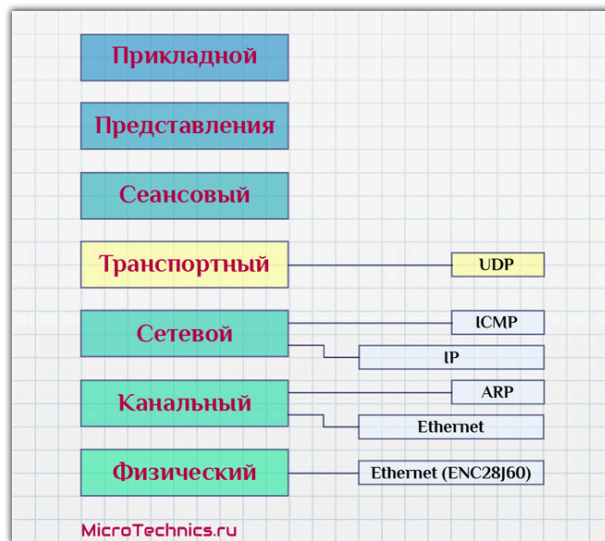
Unexpectedly, thanks [topic on the forum](#), there was a return to one of the previous topics, namely the STM32 and ENC28J60 connection and Ethernet. And in this fifth part, we process the received UDP packet and send a response to it. Moreover, unlike the previous parts, I will not make an extensive insert with the theory regarding UDP, maybe later, in a separate article, but it is unlikely (Simply and briefly add the necessary functionality to the previously created project and look at the result.

But first, let's refresh our memory of the previous parts:

- [Ethernet technology. Overview, description, frame format.](#)
- [STM32 and Ethernet. Part 1. Connecting and configuring ENC28J60.](#)
- [STM32 and Ethernet. Part 2. ENC28J60. Reception and transfer of personnel.](#)
- [STM32 and Ethernet. Part 3. Channel level. ARP protocol.](#)
- [STM32 and Ethernet. Part 4. Network layer. IP and ICMP protocols.](#)

So, the UDP protocol belongs to the transport layer, which means we are again moving up the OSI model:

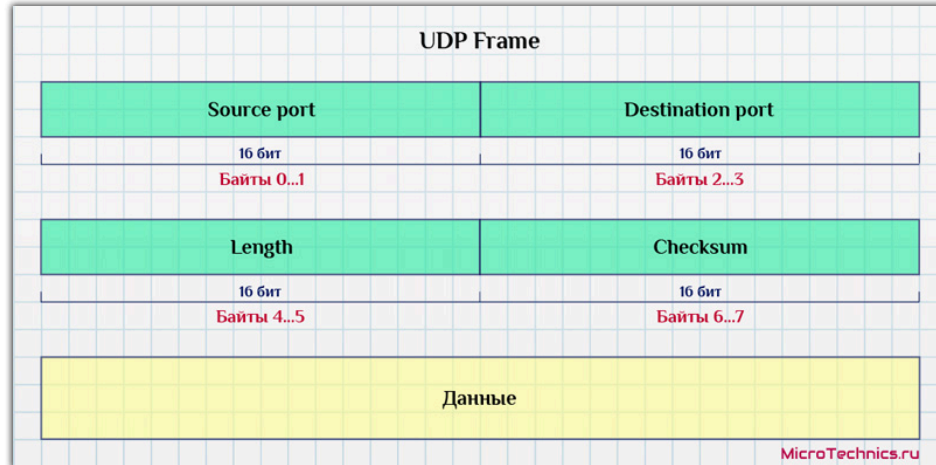




A distinctive feature of UDP is that devices do not require any prior preparation to exchange datagram messages. That is, no installation of connections, handshakes, opening of communication channels and the like. Data can easily get lost along the way, the order of datagrams can be disrupted, all this is not controlled. A reasonable question arises - why is this necessary then? And the report is quite simple - for example, for very time-sensitive systems, real-time systems. In this case, additional checks and controls are sacrificed, which is why we gain time. If packet loss, errors, and duplications are critical, then, for example, TCP comes into play, but that's not what we're talking about today.

To exchange data over UDP, you need to know the ip address and port of the exchange participants. The port is an integer from 0 to 65535 (16-bit value). Actually, in a practical example we will see this mechanism clearly.

Each datagram includes a header and data itself:



Let's take a look at the header fields:

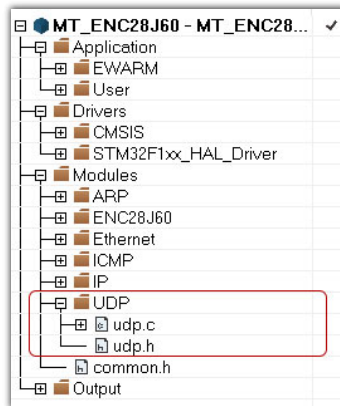
- *Source port* - sender port
- *Destination port* - recipient port
- *Length* The length of the datagram includes both the header and the data (!)
- *Checksum* - checksum

However, for IPv4, the checksum and sender port are optional fields. We found out everything we needed in practice, adding two

13 new files to the project:



- *udp.c*
- *udp.h*



First of all, let's define the structure for UDP datagrams in *udp.h*, it looks like this:

```
1. typedef struct UDP_Frame
2. {
3.     uint16_t srcPort;
4.     uint16_t destPort;
5.     uint16_t len;
6.     uint16_t checksum;
7.     uint8_t data data[];
8. } UDP_Frame;
```

Great, here we'll set the port number that we use for testing:

```
1. #define UDP_DEMO_PORT 33333
```

For example, we implement a simple mechanism: we increment all bytes in the received datagram and send them back as modified. It all materializes into the following code:

```
1. /*-----*/
2. uint16_t UDP_Process(UDP_Frame* udpFrame, uint16_t frameLen)
3. {
4.     uint16_t newFrameLen = 0;
5.
6.     uint16_t destPort = ntohs(udpFrame->destPort);
7.     uint16_t len = ntohs(udpFrame->len);
8.     uint16_t dataLen = len - sizeof(UDP_Frame);
9.
10.    if(destPort == UDP_DEMO_PORT)
11.    {
12.        for(uint16_t i = 0; i < dataLen - 1; i++)
13.        {
14.            udpFrame->data data[i]++;
15.        }
16.    }
17.
18.    uint16_t swapPort = udpFrame->destPort;
19.    udpFrame->destPort = udpFrame->srcPort;
20.    udpFrame->srcPort = swapPort;
21.
22.    udpFrame->checksum = 0;
23.    newFrameLen = len;
24.
25.    return newFrameLen;
26. }
27.
28.
29.
30. /*-----*/
```

Here we check in advance that this is our package, that is, the Destination port in the header corresponds to the one we chose for ourselves - `UDP_DEMO_PORT`. We will place a call to this function in the processor of received packets at the network level, that is, at `IP_Process()`:

```
1. /*-----*/
2. uint16_t IP_Process(IP_Frame* ipFrame, uint16_t frameLen)
3. {
4.     uint16_t newFrameLen = 0;
5.
6.     if(memcmp(ipFrame->destIpAddr, ipAddr, IP_ADDRESS_BYTES_NUM) == 0)
```

```

7.  {
8.      uint16_t rxChecksum = ipFrame->checksum;
9.      ipFrame->checksum = 0;
10.     uint16_t calcChecksum = IP_CalcChecksum((uint8_t*)ipFrame, sizeof(IP_Frame));
11.
12.     if(rxChecksum == calcChecksum)
13.     {
14.         uint16_t dataLen = frameLen - sizeof(IP_Frame);
15.         uint16_t newDataLen = 0;
16.
17.         if(ipFrame->protocol == IP_FRAME_PROTOCOL_ICMP)
18.         {
19.             newDataLen = ICMP_Process((ICMP_EchoFrame*)ipFrame->data, dataLen);
20.         }
21.
22.         if(ipFrame->protocol == IP_FRAME_PROTOCOL_UDP)
23.         {
24.             newDataLen = UDP_Process((UDP_Frame*)ipFrame->data, dataLen);
25.         }
26.
27.         newFrameLen = newDataLen + sizeof(IP_Frame);
28.         ipFrame->len = htons(newFrameLen);
29.
30.         ipFrame->fragId = 0;
31.         ipFrame->fragoffset = 0;
32.
33.         memcpy(ipFrame->destipAddr, ipFrame->srcipAddr, IP_ADDRESS_BYTES_NUM);
34.         memcpy(ipFrame->srcipAddr, ipAddr, IP_ADDRESS_BYTES_NUM);
35.
36.         ipFrame->checksum = IP_CalcChecksum((uint8_t*)ipFrame, sizeof(IP_Frame));
37.     }
38. }
39.
40. return newFrameLen;
41. }
42.
43.
44.
45. /*-----*/

```

IP_FRAME_PROTOCOL_UDP we define accordingly in *ip.h*:

```

1.  /* Declarations and definitions -----*/
2.
3.  #define IP_FRAME_PROTOCOL_ICMP 1
4.  #define IP_FRAME_PROTOCOL_UDP 17

```

Overall, everything fits perfectly into the original architecture that we planned. It seems like I haven't forgotten anything, we compile it, flash it, and we can move on to checking it.

For testing, I use the ncat utility and run the following in the command line:

```
1.  .\ncat.exe -u 169.254.191.22333 33
```

The -u key signals that UDP will be used, followed by the board's IP address and the recipient's port number. The recipients in this case are STM32 and ENC28J60. After entering the command, we can send the data itself:

```

PS C:\Program Files (x86)\Nmap> ping 169.254.191.22

Обмен пакетами с 169.254.191.22 по 32 байтами данных:
Ответ от 169.254.191.22: число байт=32 время=1мс TTL=128
Ответ от 169.254.191.22: число байт=32 время=1мс TTL=128
Ответ от 169.254.191.22: число байт=32 время=1мс TTL=128
Ответ от 169.254.191.22: число байт=32 время=1мс TTL=128

Статистика Ping для 169.254.191.22:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
              (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 1мсек, Максимальное = 1 мсек, Среднее = 1 мсек
PS C:\Program Files (x86)\Nmap> .\ncat.exe -u 169.254.191.22 33333
libsock ssl_init_helper(): OpenSSL legacy provider failed to load.

26648472551236132
37759583662347243

```



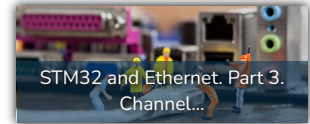
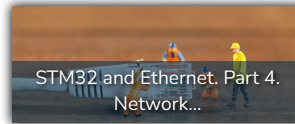
As a result, we see that it works as intended - the sent data is incremented and sent back (the bottom two lines, the first sent, the second accepted). We will end today's post with this positive moment, thank you everyone for your attention, stay in touch 🙏

+ udp.c file

+ File udp.h

Project reference - [MT_ENC28J60_Part_5](#).

Useful information in the forum thread - [link](#).



- < [How to add a library to Arduino IDE.](#)
- > [Where is the error in the diagram?](#)

✉ Subscribe ▼



Присоединиться к обсуждению

B I U {} [+]



13 COMMENTS

Old ones ▼



Shen

🕒 1 year ago

Cool guide, thank you. It would be nice if you posted not a github of the fifth (final) version of the project, otherwise there is only the fourth. I had to manually correct several inconsistencies, but that's okay 😊

↩ Reply



Aveal MT

Administrator

Author

🗨 Reply to comment [Shen](#)

🕒 1 year ago

Still won't get around to it... I'll post)

↩ Reply



Vladimir

🕒 1 year ago

Спасибо за курс полезных статей! Но я так и не смог допереть, как вы задаете данные, которые собираетесь отправлять помимо всех хедеров и служебных байтов? Через ethFrame, которая задается непосредственно в Eth_Process или есть иной способ?

Просто у вас в коде вы берете UDP_PProcess и конкретно в ней по идее инкрементируете все полученные данные, но как конкретно данные, которые я хочу отправить, задаются в коде?

13



[Ответить](#)**Aveal MT**

Администратор

Автор

Ответ на комментарий [Владимир](#)
1 год назад

Доброго времени суток!

Да, тут для демонстрации максимально просто сделал, а так в целом в зависимости от задачи - если нужно ответный фрейм со своими данными сформировать, то тут же, в UDP_Process() формируем фрейм и помещаем в него нужные данные.

[Ответить](#)**Владимир**

Ответ на комментарий [Aveal MT](#)
1 год назад

спасибо большое, получилось!

[Ответить](#)**Aveal MT**

Администратор

Автор

Ответ на комментарий [Владимир](#)
1 год назад

Отлично!

[Ответить](#)**Владимир**

Ответ на комментарий [Aveal MT](#)
1 год назад

Добрый день, опять появилась проблема) : при отправке через nmap запроса -sU -p 33333 и IP я в ответ получаю лишь 40 байт информации поля data. Где можно явно задать число передаваемых байтов?

[Ответить](#)**Aveal MT**

Администратор

Автор

Ответ на комментарий [Владимир](#)
1 год назад

А UDP_Process() сколько возвращает (newDataLen)?

[Ответить](#)**E****Erop**

1 год назад

Лучшие статьи с красивым и понятным кодом. Не ожидается финалочка для TCP? Пытаюсь уже неделю на этой основе установить хотя бы соединение, но с TCP всё сложнее, нежели с UDP. SYN получаю, а вот ACK+SYN уже не принимается. Контрольная там вроде как считается ещё от куска IpFrame. В общем...продолжаем. Как бы на lwIP не пришлось бы уходить.

[Ответить](#)**Aveal MT**

Администратор

Автор

Ответ на комментарий [Erop](#)
1 год назад

Yes, there are a lot of plans for new articles, but they still won't get around to it, probably for 5 years now...

[Reply](#)**Ivan**

Reply to comment [Aveal MT](#)
10 months ago

13



I agree with Egor. really the best articles! I haven't found a more understandable explanation throughout the network!
My question is: when connecting to network connections, an unidentified network is written, Can the network name be specified somewhere in the code?

↪ Reply

 **Ivan**
🕒 10 months ago

I agree with Egor. the best articles on this topic. Thank you very much to the author for this wonderful work! My question is: "unidentified network" is written in network connections, is it possible to specify the network name in the code?

↪ Reply

 **Aveal MT** Administrator Author

💬 Reply to comment [Ivan](#)
🕒 10 months ago

Thanks for the great review!

The network name can be set directly in windows.

↪ Reply

 MicroTechnics | STM32, Raspberry Pi...

4,562 followers



Follow on VK

The entrance

13

Registration



Search for...



Categories

Arduino

Qt

QML

Raspberry Pi

Algorithms

Sensors

Microcontrollers

AVR

MSP430

STM32

Neural networks

Miscellaneous

Rebuses, tasks, charades

Training courses

STM32 from scratch

STM32CubeMx

Fundamentals of electronics

Arduino lessons

Electronics

Community

Arduino

C++

Sensors

Microcontrollers

STM32

Developments

Electronics

13



Fresh entries

NextPCB's DFM Tool.

Where is the error in the diagram?

STM32 and Ethernet. Part 5. Transport level. UDP protocol.

How to add a library to Arduino IDE.

How to accelerate Raspberry Pi 4 Model B.

© 2013 - 2026 MicroTechnics.ru | Privacy Policy | Site rules | Images and materials on the site are protected by copyright

13

