Связь по сети Ethernet: как использовать платформу Arduino в сетях IoT и IIoT

Вот уже несколько лет как создание разветвленных компьютерных сетей перестало служить только для соединения компьютеров. Падение цен и увеличение вычислительной мощности небольших микроконтроллеров положили начало бурному процессу подключения к локальным сетям Ethenet или даже глобальной сети Интернет маломощных устройств, в основном выполняющих функции контроля, управления и измерения. Более того, эти решения стали появляться и в профессиональных промышленных сетях, постепенно вытесняя старые системы на основе RS-232 и его производных.

Таким образом, в начале XXI века наступила эра «Интернета вещей» (Internet of Things — IoT). Хотя в настоящее время на рынке IoT преобладают устройства, коммуницирующие между собой с помощью беспроводных сетей и стандартов Wi-Fi, ZigBee, BLE или Z-Wave, во многих аппаратных решениях (в основном из сегмента IIoT — Industrial Internet of Things), требующих надежной передачи и безопасности данных, одним из самых популярных вариантов по-прежнему остается сеть Ethernet. Создатели платформы Arduino не оставили без ответа пожелания разработчиков IIoT-устройств, и в стандартном предложении модулей Arduino появились платы расширения типа Ethernet Shield 2, адресованные отдельным пользователям, или Arduino MKR ETH SHIELD для профессиональных решений на базе контроллеров WIZnet W5100/W5200/W5500 и интегральные схемы MAC и PHY в одной интегральной схеме. Довольно быстро независимые производители дополнили это предложение новыми и гораздо более дешевыми модулями на основе популярных схем ENC28J60. В статье кратко охарактеризованы оба решения: официальное, основанное на схемах серии W5x00, и разрабатываемое главным образом сообществом Open Source/Open Hardware решение на основе модулей ENC28J60.

Связь с использованием модулей WIZnet W5x00 и библиотеки Arduino Ethernet

Несомненным преимуществом официальных модулей на основе схем серии W5x00 (в том числе их аппаратных аналогов, например плат расширения OKYSTAR OKY2102 или DFROBOT DFR0125, рис. 1) является полная программная поддержка в виде библиотеки Ethernet, встроенной в стек Arduino. Таким образом, пользователь может приступить к созданию программы сразу после запуска Arduino IDE, не устанавливая дополнительные программные пакеты.

В зависимости от схемы WIZnet и объема доступной памяти RAM библиотека Ethernet поддерживает максимум четыре (для схемы W5100 и памяти RAM ≤ 2 кбайт) или восемь (схемы W5200 и W5500) параллельных выходящих/входящих соединений. Программный интерфейс библиотеки разделен на пять классов, сгруппированных по отдельным функциональностям. Для нужд IP-адресации создан класс IP Address. Чтобы запустить простое серверное приложение на стороне Arduino, необходимо использовать класс Ethernet Server для выполнения чтения и записи данных со всех подключенных устройств. Взаимодополняющим классом является класс Ethernet Client, который позволяет с помощью нескольких простых вызовов подготовить функционального сетевого клиента, выполняющего операции записи и чтения данных с сервера. Для связи по протоколу UDP библиотека Ethernet предоставляет класс Ethernet UDP. Полное описание классов и методов представлено на сайте [1].

Характерным для платформы Arduino способом все сложные программные операции реализованы непосредственно в предоставляемой библиотеке. В распоряжении программиста имеется ограниченный, но очень функциональный набор API, благодаря чему процесс разработки приложения проходит быстро и не требует детального знания сетевых стеков. Поэтому проанализируем строение простейшего серверного приложения, поставляемого с библиотекой Ethernet, задача которого состоит в прослушивании входящих соединений от клиента протокола Telnet.

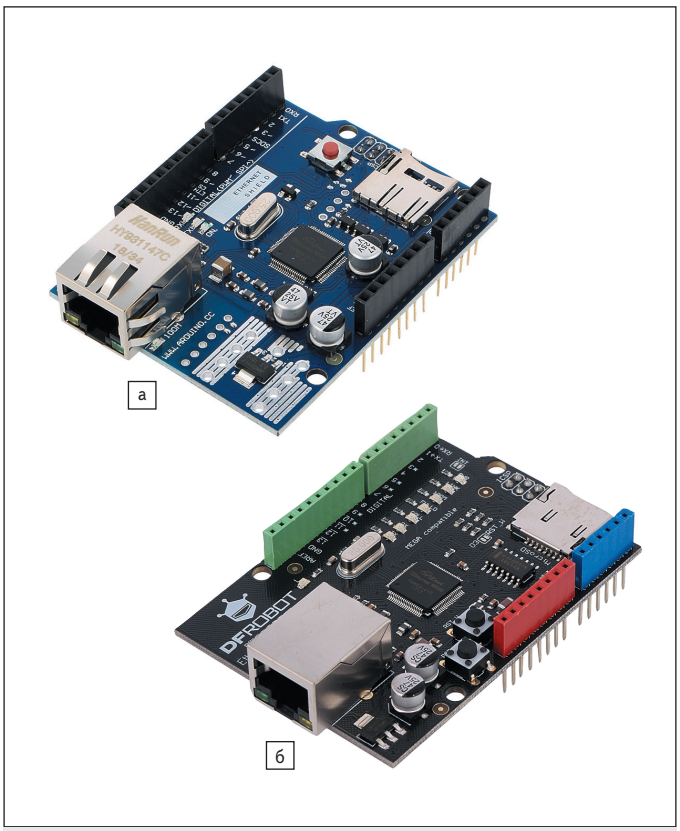

Рис. 1. Модули, оснащенные контроллером WIZnet W5100: а) OKY2102; б) DFR0125

Код серверного приложения начинает добавлять файлы заголовков, необходимые для установления связи SPI (модули WIZnet обмениваются данными с микроконтроллером, используя этот протокол), а также файлы заголовков библиотеки Ethernet:

```
#include <SPI.h>
#include <Ethernet.h>
```

Следующий шаг — настройка сетевых параметров (MAC-адрес контроллера, IP-адрес шлюза доступа и маски подсети), а также создание сервера для прослушивания порта № 23 (порт по умолчанию для протокола Telnet):

```
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

IPAddress ip(192,168,1, 177);
IPAddress gateway(192,168,1, 1);
IPAddress subnet(255, 255, 0, 0);

EthernetServer server(23);
```

В теле функции ***setup()*** необходимо инициализировать библиотеку Ethernet и запустить процесс прослушивания. Дополнительно помещена конфигурация последовательного порта, на котором будут отображаться сообщения об адресе сервера, подключении нового клиента и данных, полученных во время установленного сеанса:

```
void setup() {

    Ethernet.begin(mac, ip, gateway, subnet);
    server.begin();

    Serial.begin(9600);
        while (!Serial) {
    }

    Serial.print("Chat server address:");
    Serial.println(Ethernet.localIP());
}
```

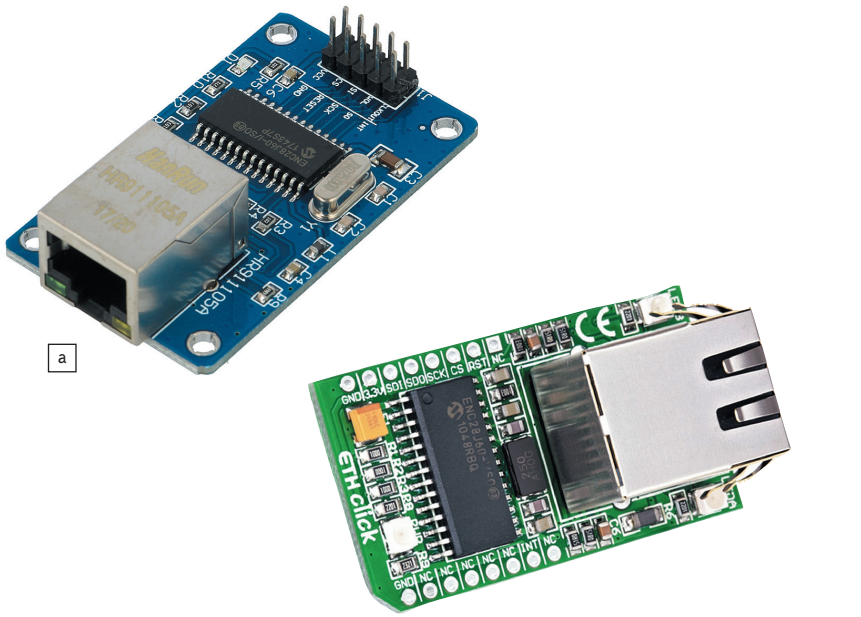

Рис. 2. Модули, оснащенные контроллером ENC28J60: а) OKY3486; б) ETH CLICK

Основной цикл ***loop()*** ожидает соединения от клиента и проверяет наличие данных для считывания. При получении данных он отправляет их без изменений клиенту, выполняя таким образом простую функцию ***echo***:

```
void loop() {

 EthernetClient client = server.available();

   if (client) {
      if (!alreadyConnected) {
         client.flush();
         Serial.println("We have a new client");
         client.println("Hello, client!");
         alreadyConnected = true;
      }

      if (client.available() > 0) {

         char thisChar = client.read();

         server.write(thisChar);
         Serial.write(thisChar);
      }
   }
}
```

Правильность работы этого приложения можно протестировать, используя произвольного клиента протокола Telnet (например, программу **Putty** в системе Windows или команду **telnet** в системе Linux) или еще один комплект Arduino и классы Ethernet Client.

Связь с использованием модулей ENC28J60 и внешних библиотек

Альтернативным решением для официально поддерживаемых схем WIZnet W5x00 являются модули на базе контроллера ENC28J60 (например, OKYSTAR OKY3486 или ETH CLICK, рис. 2). Благодаря более низкой цене и простоте ручного монтажа корпуса (в отличие от схем W5x00, содержащихся в 80-контактных корпусах LQFP, контроллер ENC28J60 предлагается в 28-контактных корпусах типа SSOP, SOIC, QFN, а также в предназначенном для сквозного монтажа корпусе SPDIP) эта схема очень популярна среди электронщиков-любителей.

Несмотря на отсутствие официальной поддержки со стороны Arduino, в распоряжение программистов передано много библиотек типа open source, обеспечивающих быструю интеграцию схем ENC28J60 с программным обеспечением. Особое внимание следует уделить библиотеке UIP Ethernet, а также предоставляемой по лицензии GPLv2 библиотеке Ether Card. Несомненным преимуществом первого из упомянутых проектов является совместимость интерфейса API с официальной библиотекой Arduino Ethernet, таким образом процесс разработки приложений становится независимым от выбора, сделанного между схемами W5x00 и схемой ENC28J60 на аппаратном уровне. Второй из проектов, Ether Card, реализует независимый программный интерфейс, который, в соответствии с предпочтениями программиста, может оказаться интересной альтернативой. Как и в случае с библиотекой Arduino Ethernet, реализация довольно сложной функциональности (например, реализация DHCP-клиента) может быть выполнена в несколько строк кода:

```
#include <EtherCard.h>

static byte mymac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

byte Ethernet::buffer[700];

void setup () {

   Serial.begin(57600);
   Serial.println(F("
[testDHCP]"));

   if (ether.begin(sizeof Ethernet::buffer, mymac, SS) == 0)
      Serial.println(F("Failed to access Ethernet controller"));

   Serial.println(F("Setting up DHCP"));
   if (!ether.dhcpSetup())
      Serial.println(F("DHCP failed"));

   ether.printIp("My IP: ", ether.myip);
   ether.printIp("Netmask: ", ether.netmask);
   ether.printIp("GW IP: ", ether.gwip);
   ether.printIp("DNS IP: ", ether.dnsip);
}

void loop () {
   ether.packetLoop(ether.packetReceive());
}
```

Литература

1. www.arduino.cc/en/Reference/Ethernet