Елена Бадло, Сергей Бадло
г. Запорожье
E-mail: **raxp@radioliga.com**

Еще не увидела свет последняя часть трилогии по работе с USB HID устройствами, а от нескольких читателей уже стали поступать вопросы и просьбы о том, возможно ли реализовать наблюдение за показаниями устройства через сеть Internet/Intranet. Отвечаем: конечно, возможно! И не только наблюдение, но и управление и диагностика. Сегодня мы с вами создадим такой интерактивный WEB сервер и встроим его в функционал нашей утилиты по измерению температуры из прошлой статьи [1].

USB термометр и дистанционка в одном флаконе. Часть 4 или… Интерактивный WEB сервер

Подобный подход не нов и был давно освоен не только производителями “умных” домов, но и серъезных SCADA систем, таких как TRACE MODE, Citect и многие другие. Как правило, WEB сервера обеспечивают удаленный доступ к информации реального времени через веб-браузер по сетям или по беспроводной сети (GSM, GPRS, Wi-Fi, Bluetooth и т.п.). При их помощи можно получить данные с любого АРМ предприятия, работающего под управлением любой операционной системы (Windows, Unix, Linux, QNX, Mac OS и т.д.), а при желании и с любого ПК в мире, подключенному к сети Интернет.

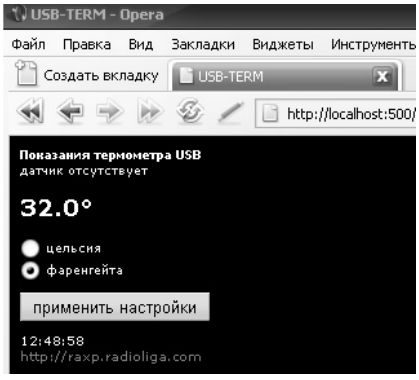

Рис. 1. “Управление USB термометром и наблюдение за его показаниями можно осуществлять как через локальную сеть, так и Интернет”

Краткий экскурс или… предпосылки реализации ПО

Возможность удаленного управления приложениями и устройствами при помощи WEB интерфейса становится все более востребованной, так как упрощает их сопровождение и администрирование как в условиях большой организации, так и малого предприятия, и обеспечивает выполнение следующих функций:

- предоставление графического интерфейса удаленным пользователям, в том числе и управление;
- предоставление графического интерфейса пользователям мобильных телефонов;
- дает доступ к документам и отчетам об аварийных ситуациях в удобном для восприятия виде.

WEB сервер можно реализовать программно, например на обычных сокетах или специальных компонентах, являющихся всего лишь надстройкой над ними. По запросу клиента сервер, получающий данные с устройства, должен сгенерить динамически страничку HTML, содержащую необходимую информацию и основные элементы управления. Основные настройки, как и само управление, можно задействовать через методы GET или POST, назначенные на кнопки и другие элементы в HTML. Для доступа к серверу можно свободно использовать порты выше 1024-го и с некоторыми ограничениями – меньшие его. Это обусловлено тем, что многие системные и стандартные приложения или сервисы уже используют определенные порты, например браузер по умолчанию работает через 80-й порт. Отсюда следует немаловажный вывод, что два приложения (WEB сервиса) на компьютере не могут одновременно использовать один и тот же порт. Это следует учитывать при разработке приложения…

Если у вас роутер и вы за NAT

Хочется обратить ваше внимание на особенность доступа из сети Интернет к нашему датчику в случае,

Рис. 2. Маршрутизатор WRT54GS

Рис. 3. Маршрутизатор DIR100

если между вами (ПК) и внешним миром установлен маршрутизатор, например беспроводной WRT54GS стандарта Wireless-G (802.11g) от Linksys или обычный DIR-100 от D-Link (см. **рис. 2** и **рис. 3**).

Как правило, при таком подключении WEB сервер будет недоступен из внешней сети. Для решения этой проблемы необходимо прописать в настройках роутера пути маршрутизации или, как говорят, “пробросить порты”. Для этого зайдите в таблицу перенаправления роутера и пропишите внутренний порт, по которому будет работать WEB сервер и внешний порт (порт перенаправления), доступный при HTTP запросе (см. **рис. 4**).

Это значит, что когда вы запрашиваете внешний адрес IP маршрутизатора из любого компьютера в сети, Интернет определяет местонахождение вашего маршрутизатора, он в свою очередь перенаправляет ваш запрос к локальному адресу IP, который вы назначили для сервера.

Если у вас нет выделенного адреса и IP динамический

И в этом случае расстраиваться не стоит. Выход из положения есть… Достаточно посетить, например, сайт [2], с которого можно скачать небольшую программку (~3Mb). После чего, запустив ее, вы сможете создать глобальный линк (переадресацию) на свой компьютер. Ссылка в Интернете будет выглядеть следующим образом: **http://ваше название.dynip.com** и любой, кто наберет в браузере этот адрес, попадет к Вам на компьютер...

Таким образом, уже можем сформировать основные требования к нашему интерактивному WEB серверу (термометру):

- отсутствие дополнительных драйверов;
- работоспособность во всех Windows системах;
- диагностика наличия-отсутствия подключенного HID устройства;
- трансляция показаний температуры с датчика в локальную сеть и Интернет;
- возможность управления режимами индикации в градусах Цельсия или Фаренгейта из браузера;
- автообновление страницы с данными;
- открытые исходники.

Теперь перейдем непосредственно к коду...

Итак, приступим к основной задаче. Для работы нам понадобиться следующее [3]:

- среда Borland Delphi 5-7 (компиляция и отладка тестового проекта USB термометра)
- собственно наш USB модуль температуры (см. **рис. 5**)
- наличие Интернета для реальных испытаний

Ввиду ограниченности места в журнале, рассмотрим только основные моменты реализации интерактивного WEB сервера. Так как предполагается лишь вывод показаний в трее и их трансляция по TCP/IP, то для упрощения разработку осуществим на WinAPI. Для упрощения алгоритма и возможности работы в консоли создадим свой класс, в котором осуществим инициализацию и пропишем все необходимые методы компонентов по доступу к HID устройствам TjvHidDeviceController и самого сервера на TserverSocket, впрочем, и TidHTTPServer тоже подойдет (см. **листинг 1**).

Поиск HID устройств и получение данных осуществим следующим образом (см. **листинг 2**).

И собственно то, ради чего все задумывалось – получение данных с датчика температуры в браузере. Как это сделать? Все достаточно просто. Воспользуемся событием – onClientRead() компонента TServerSocket. В этом событии будем осуществлять обработку входящих запросов и выдачу клиенту соответствующей информации по протоколу HTTP. При этом, обмен состоит из нескольких шагов:

- установка соединения по определенному порту (производится по инициативе клиента);
- запрос клиента, содержащий ID ресурса и версию протокола;
- ответ сервера;
- разрыв соединения с клиентом (инициатором может служить как сервер, так и сам клиент).

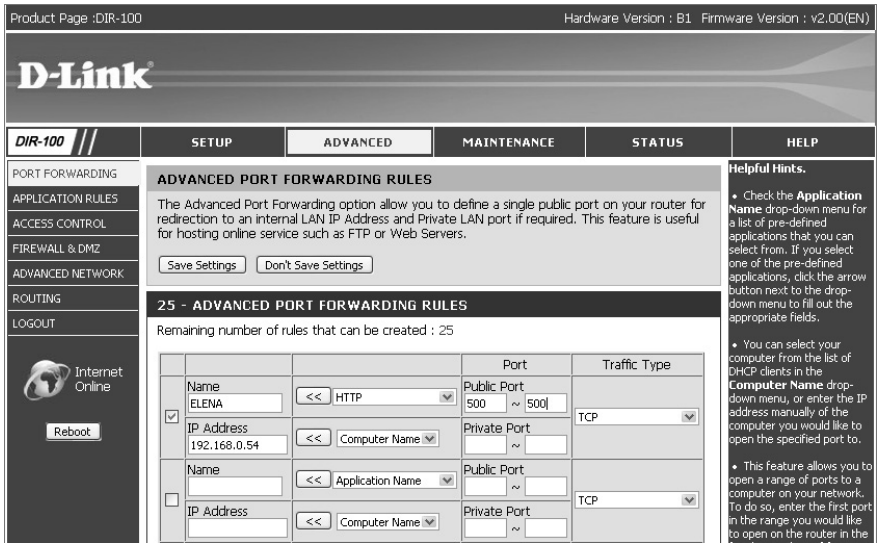

Рис. 4. Таблица маршрутизации

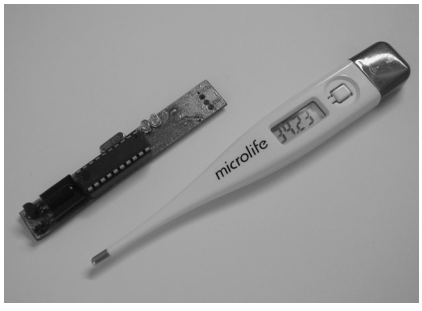

Рис. 5. USB.HID модуль термометра

Ответ сервера всегда будет состоять из строки с версией протокола HTTP, пробела и трехзначного кода статуса, за которым через пробел может следовать его расшифровка. После этого передаются символы перевода строки 0x0D и 0x0A, затем идет необязательная информационная часть в формате “параметр=значение” и завершается все опять парой символов перевода строки. Уже после этого идет запрошенная информация. Типичный пример ответа:

“HTTP/1.0 200 OK”.

Рассмотрим самый простой по структуре HTML код (см. **листинг 3**).

Как видите, код содержит тег <META HTTP-EQUIV=“Refresh” CONTENT=“4;URL=”> обновления странички, которая будет раз в 4 секунды обновлять сама себя в браузере, т.е. посылать запрос серверу на новые данные. Также страничка содержит данные о наличии датчика, его температуре, текущем времени и компоненты обратной связи <input>, выполняющие GET запрос* по нажатию кнопки “применить настройки”.

Если просмотреть эти запросы HTTP снифферо, например с помощью Commview, то что же мы увидим? А увидим мы типичный набор данных, отличающихся лишь тегами (кодами запроса) “mode=1 и 2” (см. **листинг 4**).

* Метод GET является самым часто используемым и предназначен для получения данных с сервера. В качестве таких данных может выступать файл или результаты выполнения какого-либо процесса, например CGI. В основном, данный метод используется при передаче небольших объемов информации в виде параметров.

листинг 1

```
создаем класс для работы в консоли
...
type
  HIDC = class
    hid: TJvHidDeviceController;
    procedure hidArrival(HidDev: TJvHidDevice);
    procedure hidRemoval(HidDev: TJvHidDevice);
    procedure hidDeviceUnplug(HidDev: TJvHidDevice);
    function hidEnumerate(HidDev: TJvHidDevice; const Idx: Integer): Boolean;
    procedure hidDeviceChange(Sender: TObject);
    procedure hidDeviceData(HidDev: TJvHidDevice; ReportID: Byte; const Data: Pointer; Size: Word);

    procedure srv_read(Sender: TObject; Socket: TCustomWinSocket);    // обработка запроса
    procedure srv_error(Sender: TObject; Socket: TCustomWinSocket;    // перехват ошибок
                        ErrorEvent: TErrorEvent; var ErrorCode: Integer);
public
    constructor Create;
    destructor Destroy; override;
    end;

constructor hidc.Create;
Begin
 inherited Create;
 // класс доступа к HID устройствам
 hid1                        := TJvHidDeviceController.Create(nil);
 hid1.onEnumerate    := hidEnumerate;
 hid1.onDeviceChange := hidDeviceChange;
 hid1.onDeviceData   := hidDeviceData;
 hid1.OnArrival              := hidArrival;
 hid1.onRemoval      := hidRemoval;
 hid1.onDeviceUnplug := hidDeviceUnplug;

 // класс web- сервера
 srv1                        := TServerSocket.create(nil);
 srv1.port           := 500;          // задаем порт 500
 srv1.onClientError  := srv_error;    // назначаем события
 srv1.onClientRead   := srv_read;
 srv1.active         := true          // активируем сервер
End;
destructor hidc.Destroy;
Begin
 hid1.Free;
 srv1.Free;
 inherited Destroy
End;
...
```

листинг 2

```
поиск и получение данных с HID устройств
…
// поиск и наполнение динамического списка HID устройств
function hidc.hidEnumerate(HidDev: TJvHidDevice; const Idx: Integer): Boolean;
begin
 setlength(dev1,length(dev1)+1);
 hid1.CheckOutByIndex(dev, Idx);
 Dev.NumInputBuffers := 128;
 Dev.NumOverlappedBuffers    := 128;
 dev1[length(dev1)-1]        := dev;
 Result:= True
end;

// обновление списка подключенных устройств
procedure hidc.hidDeviceChange(Sender: TObject);
var i: Integer;
begin
 try
  for i:= 0 to length(dev1)-1 do begin // очищаем
   dev:= TJvHidDevice(dev1[i]);
   dev.Free
  end;
  setlength(dev1,0);

  hid1.CheckOut(Dev);
  setlength(dev1, length(dev1)+1);
  Dev.NumInputBuffers         := 128;
  Dev.NumOverlappedBuffers    := 128;
  dev1[length(dev1)-1]        := dev
 except end
end;

// получение данных
procedure hidc.hidDeviceData(HidDev: TJvHidDevice; ReportID: Byte; const Data: Pointer; Size: Word);
const koef= -3;
var i: integer;
     s: string;
     k: longword;
begin
 k:= 0;
 //
 for i:= Size-1 downto 0 do
  k:= (k shl 8) or Cardinal(PChar(Data)[i]);
 gl_temp:= k - 273 + koef
end;
...
```

Следовательно, получив эти данные и пропарсив их на наличие тегов управления, WEB сервер может сформировать динамически нужную информацию, согласно алгоритма управления, и выдать обратно браузеру. Для этого достаточно воспользоваться функцией поиска POS(). Реализация подобного подхода представлена в (см. **листинг 5**).

Обратите внимание на строку “Content-Type” content=“text/html; Charset=windows-1251”, которая определяет описание последующего содержимого, а именно кода HTML и кодировки страницы. Без этих тегов браузер клиента может некорректно отображать нашу страничку.

После компиляции проекта запустим приложение на выполнение. При этом, набрав в строке адреса любого браузера путь “http://localhost:500” уже можем наблюдать работу сервера (см. **рис. 6**).

Остается проверить управление через браузер. Для чего на страничке выберем единицу измерения в Цельсиях или Фаренгейтах (пощелкаем птичку) и нажмем кнопочку “применить настройки”. После обновления сессии получим новые данные от сервера. Отлично, теперь мы управляем устройством!

листинг 3

```
контент выдаваемый клиенту
...
<html><head><title>USB-TERM</title>
<META HTTP-EQUIV=»Refresh» CONTENT=»4; URL=»>

</head><body>
<b>Показания термометра USB</b><br>датчик отсутствует<h1>0°</h1>

<FORM ACTION=»»>
  <input type=»radio» name=»mode» value=»1" checked> цельсия<br>
  <input type=»radio» name=»mode» value=»2"> фаренгейта<p>
  <INPUT TYPE=SUBMIT VALUE=»применить настройки»>
</FORM>

<p>07:48:48<br></p></body></html>
...
```

листинг 4

```
типичные запросы от браузера к WEB – серверу при выборе клиентом режимов отображения
...
GET /?mode=1 HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-
flash, application/vnd.ms-excel, application/msword, */*
Referer: http://localhost:500/
Accept-Language: ru
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: localhost:500
Connection: Keep-Alive

GET /?mode=2 HTTP/1.1
...
```

листинг 5

```
формирование динамической HTML страницы в зависимости от запроса и выдача клиенту
...
procedure html3(s, ms: string; Socket: TCustomWinSocket);
var temp: string;
begin
 if pos(“mode=1”, s) > 0 then p := false; // поиск команд во входящем запросе-
 if pos(“mode=2”, s) > 0 then p := true;
 if (p) then temp:= format(“%.1f”, [gl_temp*1.8 + 32])       // °F – выдача показаний
  else temp:= format(“%.1d”, [gl_temp]);                     // °C

 Socket.SendText(“HTTP/1.0 200 OK” + #$0D + #$0A);
 Socket.SendText(“Server: USB-TERM” + #$0A);
 Socket.SendText(#$0D+#$0A);

 //формирование HTML-
 Socket.SendText(“<html><head><title>USB-TERM</title>” +
                 “<STYLE TYPE=»text/css»><!–BODY {background-color: black; font-family: Verdana; color: white; font-size: 9px} –> </STYLE>’ +
                 “<style>a{color:#668791;text-decoration: none; font:10px verdana} a:hover {color:lime}</style>” +
                 “<meta http-equiv=»Content-Type» content=»text/html; Charset=windows-1251”>’ +
                 “<META HTTP-EQUIV=»Refresh» CONTENT=»4;URL=»>’ +
                 “</head><body><b>Показания термометра USB</b><br>” + status +’<h1>’ +
                 temp +
                 “°</h1><FORM ACTION=»»>’);

 if (p) then
 Socket.SendText(“<input type=»radio» name=»mode» value=»1"> цельсия<br>’ +
                 “<input type=»radio» name=»mode» value=»2" checked> фаренгейта’)
 else
 Socket.SendText(“<input type=»radio» name=»mode» value=»1" checked> цельсия<br>’ +
                 “<input type=»radio» name=»mode» value=»2"> фаренгейта’);

 Socket.SendText(“<P><INPUT TYPE=SUBMIT VALUE=»применить настройки»></FORM>’ +

                 “<p>” + formatdatetime(“hh:nn:ss”,time) +
                 “<br><a href=http://raxp.radioliga.com>http://raxp.radioliga.com</a></p></body></html>”);

 socket.Close
end;

// событие приема запроса-
procedure hidc.srv_read(Sender: TObject; Socket: TCustomWinSocket);
var s: string;
begin
 s:= Socket.ReceiveText;
 html3(s, “500”, Socket) // формируем динамический ответ любому клиенту по порту 500
end;
// END WEB.Server
...
```

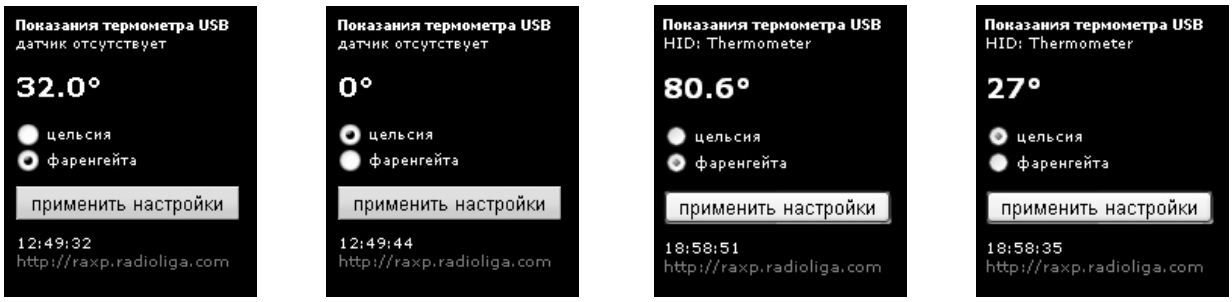

Рис. 6. Просмотр показаний температуры в браузере при наличии и отсутствии датчика

Далее, подключите USB-HID термометр к вашему USB порту. После инициализации модуля наберите реальный адрес, по которому WEB сервер будет доступен из Интернета, и наслаждайтесь удаленным контролем и просмотром показаний датчика температуры (см. **рис. 7** и **рис. 8**).

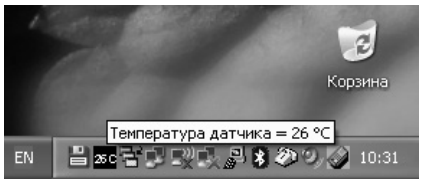

Рис. 7. Подключаем USB-HID термометр

Заключение

Как видите, ничего сложного. Единственно, хочется обратить ваше внимание на отсутствие возможности назначить альтернативный порт для работы WEB сервера, но при желании освоить работу с INI файлами вдумчивый читатель сможет с легкостью проделать это и сам. Будем считать это домашним заданием…

В следующих наших статьях мы научим USB термометр озвучивать показания датчика, для чего разработаем плагин синтеза речи, а также узнаем, как интегрировать GSM модем в любое приложение, в том числе и SCADA системы.

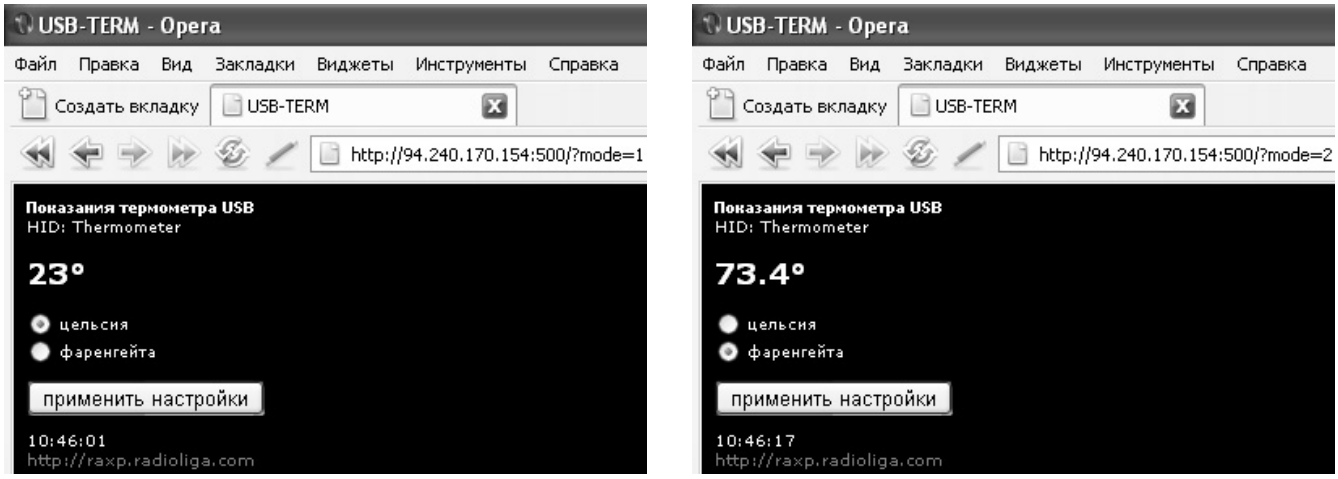

Рис. 8. Просмотр показаний датчика температуры и управление через Интерент (по реальному адресу)

Полные исходные тексты и компиляцию проекта интерактивного WEB сервера USB термометра (файл *usb4.zip*) вы можете загрузить с сайта нашего журнала:

http://www.radioliga.com (раздел “Программы”)

а также с сайта автора:

http://raxp.radioliga.com

Если тема представляет для вас интерес – пишите, задавайте вопросы на форуме: **http://raxp.radioliga.com/forum**

Литература, ресурсы

1. Е.Бадло, С.Бадло. USB термометр и дистанционка в одном флаконе. Часть 3. - Радиолюбитель, 2010, №2, с. 46-51.

2. http://www.dynip.com

3. Ресурсы тестового проекта и компиляция - http://raxp.radioliga.com/cnt/s.php?p=usb4.zip