使用ATMega88实现HTTP/TCP（AVR Web服务器）

HTTP/TCP with an atmega88 microcontroller (AVR web server)

Abstract:

摘要

This is a continuation of the article An AVR microcontroller based Ethernet device. The hardware is still the same (ENC28J60 + atmega88). The software is now updated to provide also a a web-server.

本文是之前《基于AVR的以太网设备》的续篇。硬件还和之前一样（ENC28j60+ATMega88）。片内软件现在已经升级到能够支持做一个Web服务器了。

That is: instead of using a command line application and send UDP packets to the Ethernet device we can just point our web browser to it. .... and even better: we add only the web server and all the UDP based functionality is still there. Now you can use both!

升级的部分是：使用web浏览器取代之前的命令行发送UDR数据报的方式来访问我们的以太网设备……之前的UDP基本功能仍然保留。现在，这两种功能你都可以使用了!

The code is written in C and there is even a lot of space left on the atmega88 microcontroller.

即便代码是使用C编写的，（编译以后）ATMeg88存储器中仍然有相当的剩余。

All hardware components are available from **shop.tuxgraphics.org**.

The software and circuit diagrams are available for free (GPL V2 license).
代码和电路都是免费可用的。

Introduction

A UDP command interface is sufficient for most applications but an integrated web-server is much more universal and easier to use. How to build a web server into an atmega88 chip?

对于大部分应用场合，UDP命令行的方式已经足够了，但是，对于一个简单通用的综合性Web服务器来说，却远远不够。怎样在一个ATMega88中建立一个Web服务器呢？

Before starting this Ethernet project I did of course some prototyping and then I noticed already that UDP was not a problem with lots of space left on the atmega88. Therefore I was quite confident that TCP + HTTP will work. TCP/IP was invented more than 25 years ago. Todays microcontrollers provide almost the computing power a standard computer had at that time. No java or xml was used at that time. Things were done in smart and efficient ways.

在我开始做这个以太网项目的之前，我做了一些原型机，在这个过程中，我注意到，即便实现了UDP协议以后，ATMega88中仍然有大量的空间剩余。因此，我对于实现TCP+HTTP非常有信心。TCP/IP协议已经诞生了将近25年。当今的微控制器提供了一个几乎和从前标准计算机相同的计算能力。而从前，没有java和xml。今天，同样的事情通过一种更加小巧有效的方法完成。
So here is a real web-server on an atmega88 AVR microcontroller.

所以今天，我们将在这里通过ATMega88实现一个真正的Web网络服务器。

TCP is a state machine
TCP 是一个状态机

TCP is a protocol where one establishes a connection. For this a number of packets are first exchanged and both sides of the connection go through several states [see tcp state machine from rfc793]. Once the connection is established a number of data packets can be sent. More than one packet, large amounts of data can be sent. Counters and the state machine ensure that the actual user data arrives in correct order and without data loss.

TCP是一个建立网络连接的协议。通过该协议，在连接建立之前，连接双方通过交换一些IP

数据报实现在一些状态间进行切换[请参照RFC793标准中TCP状态机的相关内容]。一旦连接建立成功，包含数据的IP数据报将被传送——实现海量数据的传输。通过计数器和状态机我们可以确认数据被按照正确的顺序传送，并且没有发生丢失。

Large web pages will need to send many data packets. Small pages less. How many do we need to send???

大的网页需要发送大量的数据包，当然，小的网页需要传送的数据包就要少些（数据包就是包含实际数据的IP数据报），那么，我们究竟需要发送多少呢？

Let's take a look at the application introduced in the first article [June 2006, article060601]. In this first article we just switch on and off something. It can be done with a simple web page which might look like this:
让我们回顾一下本文的前篇中提到过的那个程序。在前文中，我们只是（用UDP协议实现）将一些开关打开（Switch on）或者关闭（Switch off），同样的功能可以通过下面的简单网页来实现：

Output is: ON

Switch off

Figure 1: The web page needed for the Ethernet Remote Device circuit.
Figure 1：使用网页来配置远处的以太网设备

Other applications might be measurement of temperature or air pressure. Those are all small web pages with is very little data. In other words we will send less than 100 bytes including all the html tags. How many IP packets with data will be sent for such a page?
Just one!
在其他的应用中，可能是（用UDP协议）来采集温度或者气压。这些都是小的网页，只需要很少少的信息量。换句话说，我们只需要发送不到100个字节的信息就可以包含所有的HTML网页信息了。那么，这种情况下究竟需要多少IP数据包来发送这样的页面呢？

The whole point of using TCP is that one can send more than one packet of data but we don't need that functionality. We need TCP only because HTTP is based on it and we want HTTP in order to

use our web browser.

使用TCP协议的要点是：虽然发送者可以发送多个数据包，但我们并不需要这样能够的功能。我们是用TCP协议，只是因为我们网页浏览器使用的HTTP协议是建立在TCP基础之上的。

Under the assumption that you will never need to send more than one packet with data the whole TCP protocol and state handling can be simplified a lot. We can e.g send the FIN immediately together with the data. This makes the state handling for the closing of the connection very simple.

假设我们永远也不需要发送多于一个的数据包，那么整个TCP协议以及其状态处理可以大大的简化。例如，我们可以在发送完数据以后立即发送FIN（申请断开连接的请求信号）。这样，关闭连接的状态处理就变得非常简单了（因为状态少了）。

Under this assumption it possible to implement a web-server in a atmega88 and have just under 50% of the chip's memory still free for the actual application.

在这个假设下，就有可能在ATmega88中实现Web服务器，甚至我们还有将近50%的片内存储空间留下用来执行其他实际（功能）的应用程序。

The Ethernet remote device with build-in web server: switching something on and off
内建Web服务器的以太网远程设备：

开关的控制

The application that the eth_rem_dev_tcp-2.X software implements is a simple switch. You can switch on or off something. A simple password mechanism provides very basic protection to avoid that unauthorized users toggle the switch.
Here is a screen shot of the web-page that the eth_rem_dev_tcp-2.X displays:
代码：eth_rem_dev_tcp-2.X实现一个简单的开关控制。你可以通过这个实例来控制（远程以太网设备的某些）开关量。同时，通过一个简单的密码检测提供最基本的保护，防止未经授权的用户来攫取开关的控制权。
下面的图片是这个例程的一个截屏：

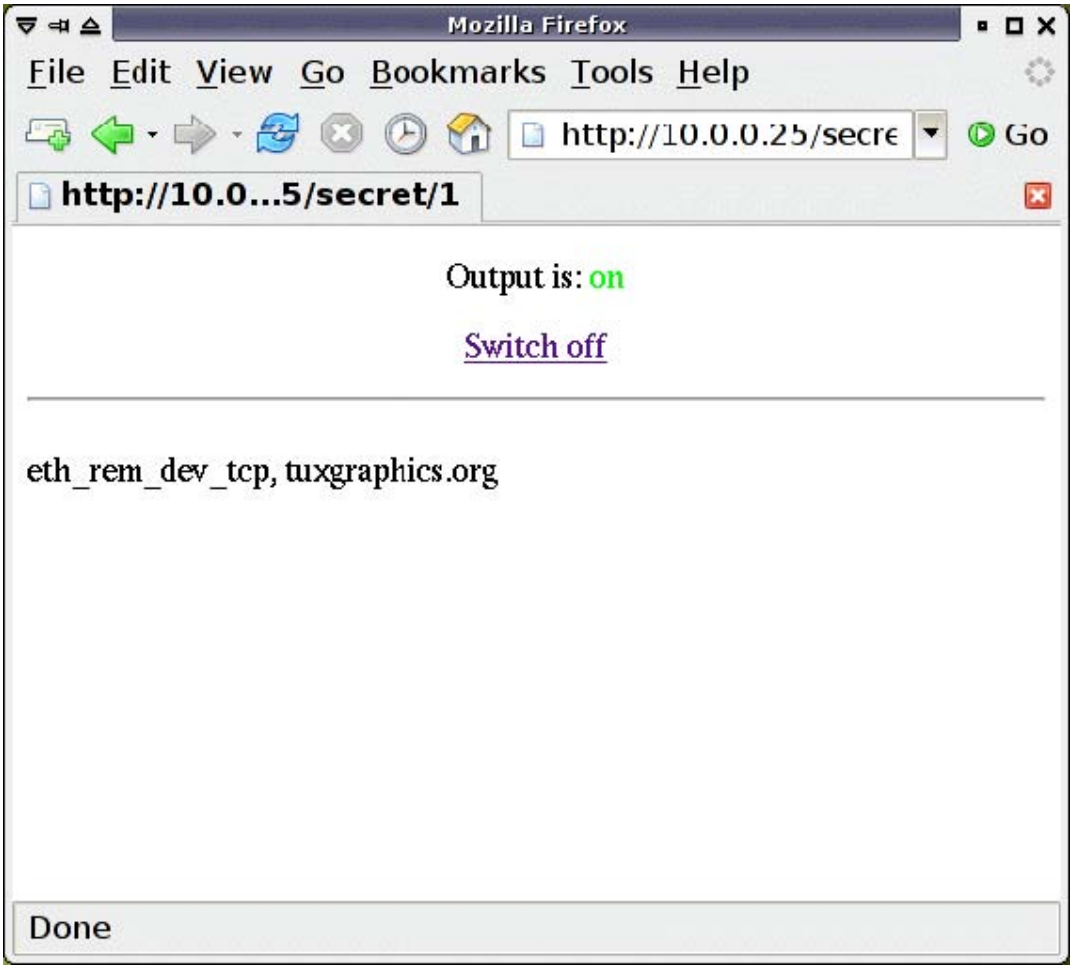

Figure 2: The atmega88 based web-server, screenshot with mozilla firefox
Figure 2:基于ATMega88的Web服务器的网页截图

A Single Data Packet TCP/HTTP web-server on a microcontroller
使用微控制器实现的单数据包的TCP/HTTP网页服务器

The code is available at the end of the article and I will explain it a bit. That way you will hopefully be able to modify it and adapt it also to other applications. The Single Data Packet web-server will go through the following TCP states:
文章结尾处的代码是可以直接使用的，当然我会适当作一些解释。这样你就可以通过修改代码来适应其他应用的要求。单数据包Web服务器将经过以下（简化过的）状态（实现一次数据传输）：

1 receive SYN
2 send SYN,ACK
3 receive ACK (the connection is now established)
4 receive ACK with HTTP GET command
5 send ACK
6 send FIN,ACK with HTTP data (e.g 200 OK)
7 receive FIN,ACK
8 send ACK

1 接收“从对方到己方”的连接请求信号（SYN）
2 发送“从己方到对方”的连接请求（SYN），并应答前面的对方的连接请求（ACK）
3 接收对方回应的应答信号（现在，已经建立了一个连接）
4 接收HTTP GET 指令的应答信号（ACK）
5 发送应答信号
6 发送HTTP数据包（比方说，200个字节的数据），同时请求终止“从己方到对方”连接（FIN），并直接应答（ACK）
7 接收对方请求终止“从对方到己方”连接的信号（FIN）以及对刚才发送出去的终止请求的应答信号（ACK）
8 发送应答信号，告知对方它终止连接的请求已经被响应（ACK）

As you can see this is a quite simple command and action sequence. I have implemented the needed functions for this the file ip_arp_udp_tcp.c The file main.c is where the receive loop for the data is implemented. main.c has in this loop a number of if statements in order to decide what action to take. Here you will also see that the code branches between udp and tcp with port 80 (=web-server). If you want to implement your own application (e.g read temperatures, air pressure, whatever...) the you just need to modify the code above the call to the function print_webpage and modify the function print_webpage in order to print your own webpage. All this is in the file main.c The file enc28j60.c implements the driver to the Ethernet chip. You don’t have to worry about the enc28j60.c.
就像你上面看到的那样，这一过程非常简单。我已经在ip_arp_udp_tcp.c文件中完成了所需的基本函数。主文件main.c中有一个查询式的数据接收循环。上面描述的状态以及状态的处理程序都在这个循环中得到实现。例程中，你会发现用于分别处理UDP和TCP（Web服务器通常使用80端口）的程序分支。如果你想实现一个你自己的功能（比方说，读取温度、气压或者其他什么东东……）你只需要修改所生成网页的内容以及之前的一些代码就可以了。所有的这些都在main.c中，而文件enc28j60.c是以太网芯片的驱动函数库，你可以不必理它。

The URL format
URL格式

In order to build an interactive web page the HTML code provides "<a href=" for links and HTML Forms for more complicated dialogs. The problem with forms is however that HTML Forms are code intensive and difficult to decode. A much easier solution is to implement virtual folders and files. The password can e.g be one folder. In other words you have to type http://IP_or_HOST/password . Behind this url we can implement a virtual file which is the command. In our case switch on (=1) or swith off (=0). The full URL would then e.g look like this.

构建一个交互式的网页，通过HTML提供的代码”<a href=”，我们可以在网页中添加我们需要的连接，同时HTML还允许我们实现其他复杂的窗体效果。关键问题是，描述HTML窗体结构的代码相当的复杂，难以解读。一个非常简单的解决方法是实现一个虚拟文件夹和文件。而密码实际上就是我们需要访问的文件夹。也就是说，我们需要输入：
http://IP_or_HOST/Password.通过这个超级连接，我们实现一个虚拟的文件，而这个文件就是我们需要的那条指令。在先前的实例中，我们打开开关（=1）或者关闭开关（=0）。完整URL地址的通常为：

```
Switch on:
   http://IP_or_HOST/password/1
```

要打开开关，访问这个地址：
http://IP_or HOST/Password/1

```
Switch off:
   http://IP_or_HOST/password/0
```

要关闭开关，访问这个地址：
http://IP_or_HOST/Password/0

```
See the current status and change nothing:
   http://IP_or_HOST/password
```

只是查询当前的开关状态，直接访问：
http://IP_or_HOST/password

This is very easy to understand and easy to decode in the microcontroller. If you want to implement just a thermometer or present some other readings without password protection then you can just implement the "root" folder: http://IP_or_HOST and delete the /password/command code in main.c

这非常好理解，对于微控制器来说也非常容易解码。如果你只是想实现一个温度计或者其他信息读取的功能，并且不需要密码的保护，你甚至可以直接在根目录上实现所需的功能：
http://IP_or_HOST 只需要删除main.c中/password/command相关的代码。

The web-server hardware
Web服务器的硬件电路

The hardware is exactly the same as described in the previous article An AVR microcontroller based Ethernet device:
硬件电路和之前《基于AVR的以太网设备》中描述的一样：

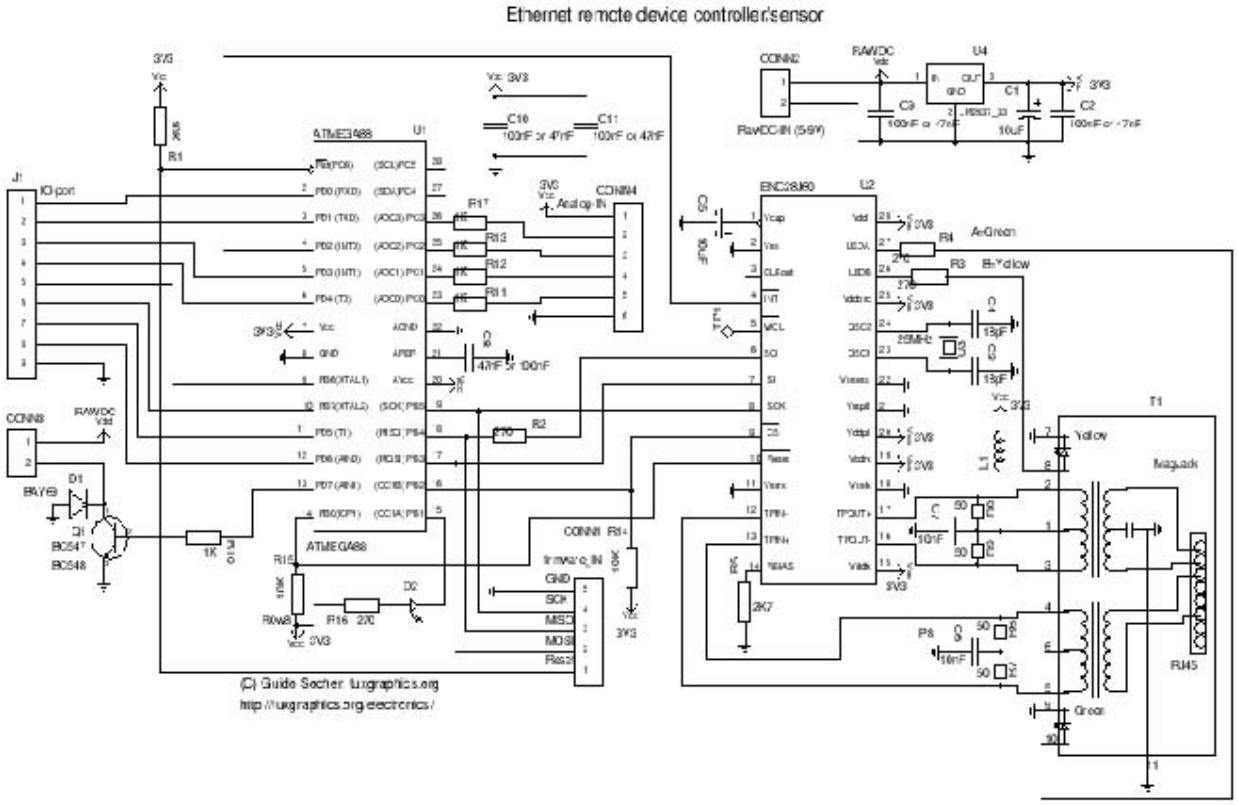

Figure 3: Circuit diagram (click for a printable pdf version).

You get the web-server just by uploading new firmware to the microcontroller.
你可以只需要给微控制器更新程序就可以实现Web服务器

Building and loading the software

Unpack the eth_rem_dev_tcp-2.X package (command tar -zxvf eth_rem_dev_tcp-2.X to unpack, software download at the end of this article). Take a look at the included README file it contains detailed instructions.
解压 eth_rem_dev_tcp-2.X压缩包（通过本文后面的地址下载文件以后，使用命令 tar -zxvf eth_rem_dev_tcp-2.X 解压缩文件）请仔细阅读包中的readme中关于细节的描述。

Next you need to set the IP address for your hardware. Edit the file main.c and change the 3 lines:
接下来，你需要设定硬件的IP地址。打开main.c，修改下面三行：

```
static uint8_t mymac[6] = {0x54,0x55,0x58,0x10,0x00,0x24};
 static uint8_t myip[4] = {10,0,0,24};
 static char baseurl[]="http://10.0.0.24/";
```

For the first device you build you will not need to change the mymac line. But you will probably need to change the IP address (myip). It must be a free address from the address range in your

network.

作为你制作的第一个网络器件，你不需要修改mymac的内容（配置MAC地址，每一个网络设备都有一个唯一的网络MAC地址）。但是，你很可能需要修改一下IP地址（myip）。这个IP一定要是一个你所在的局域网络中没有使用的地址。

There is a range of private addresses (not routed on the public Internet) which you can use:

这里有一些关于地址设置的范围信息（这些地址和公共的Internet地址是无关的）：

```
Netmask Network Addresses

255.0.0.0 10.0.0.0 - 10.255.255.255

255.255.0.0 172.16.0.0 - 172.31.255.255

255.255.255.0 192.168.0.0 - 192.168.255.255
```

Example: your WIFI router might have 192.168.1.1, your PC might have 192.168.1.2. This means you could e.g use 192.168.1.10 and leave some room for more PCs. If you use DHCP from your router then make sure that the address it not double allocated (exclude it from the DHCP range).

例如：假设你的局域网路由是192.168.1.1，你的电脑IP地址是192.168.1.2。这就是说，你可以使用192.168.1.10作为我们的Web服务器的IP地址（留下一定的IP空间给其他电脑）。如果你的路由配置使用了IP动态主机配置协议，那么你需要确信你分配给设备的地址不会与路由分配的其他IP地址发生冲突。
Now compile the software with the command "make". Load the eth_rem_dev_tcp.hex file into the microcontroller. Open a web browser and point it to http://Ip.Addr.you.assigned/secret

Easy ;-)

现在，通过”make”指令来编译我们的程序，将eth_rem_dev_tcp.hex下载到微控制器中。打开网页浏览器连接到：“http://Ip.Addr.you.assigned/secret”。

Internet Security
网络安全

As you can imagine this web-server code is written for a friendly environment. Don’t put it on the open internet. The code uses e.g the incoming packet to build from it the outgoing packet. An attacker who sends a mal formed IP packet might be able to confuse the TCP/IP stack. The code is verified with a lot of different browsers and a number of operating systems. I have tested Linux, BSD Unix, Win 98, Win XP, Mac OS-X with a number of different web browsers. It works very well but I did not do any attack or break and destroy tests with invalid data.

正像你想象的那样，这个Web服务器被设计工作在一个“友好”的环境中的。千万不要把它真的方到互联网上。代码使用来进行连接的数据包构建发送出去的数据包。攻击者通过简单的发送不规则的IP数据包就能使我们简单的TCP/IP协议栈陷入混乱。当然，这个代码经过了

很多不同平台下的不同浏览器的测试。我曾经测试过Linux平台、BSD Unix平台、Win98、WinXP、Mac OS-X 下的很多不同的浏览器。服务器工作的很好，但是任何携带无效数据的攻击都可以破坏服务器的正常工作。

The system can however still be used on the internet if you put some additional security measures in-between. I will show in a coming article a "proxy cgi script" which forwards and filters the requests at a firewall or a server connected to the public internet. It is save to use the eth_rem_dev_tcp in combination with this "proxy cgi script" on the public internet.

这个系统在你添加了一些安全措施以后，还是可以直接在互联网上使用的。我将在下一篇文章《CGI Script代理》中展示如何通过防火墙过滤互联网上的连接请求。该文将仍然使用eth_rem_dev_tcp中的内容，来保证与本文的一致性。

Download and links

Download page for this article: the eth_rem_dev_tcp software, diagrams, software updates
The previous article with the description of the hardware: An AVR microcontroller based Ethernet device.

- The tuxgraphics shop: shop.tuxgraphics.org Here you can get all the components needed to build this small web-server.

<--, tuxgraphics Go to the index Home of this section

2006-11-27, generated by tuxgrparser version 2.54
2006-12-4 傻孩子 翻译