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

Abstract:

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 web-server.

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!

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

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?

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.

So here is a real web-server on an atmega88 AVR microcontroller.

TCP is a state machine

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.

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

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:

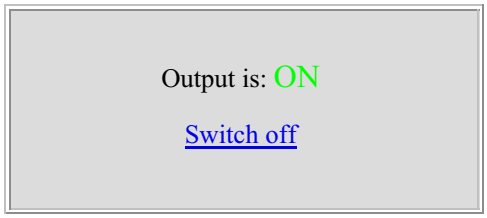

Figure 1: The web page needed for the Ethernet Remote Device circuit.

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!

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.

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.

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.

The Ethernet remote device with build-in web server: switching something on and off

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:

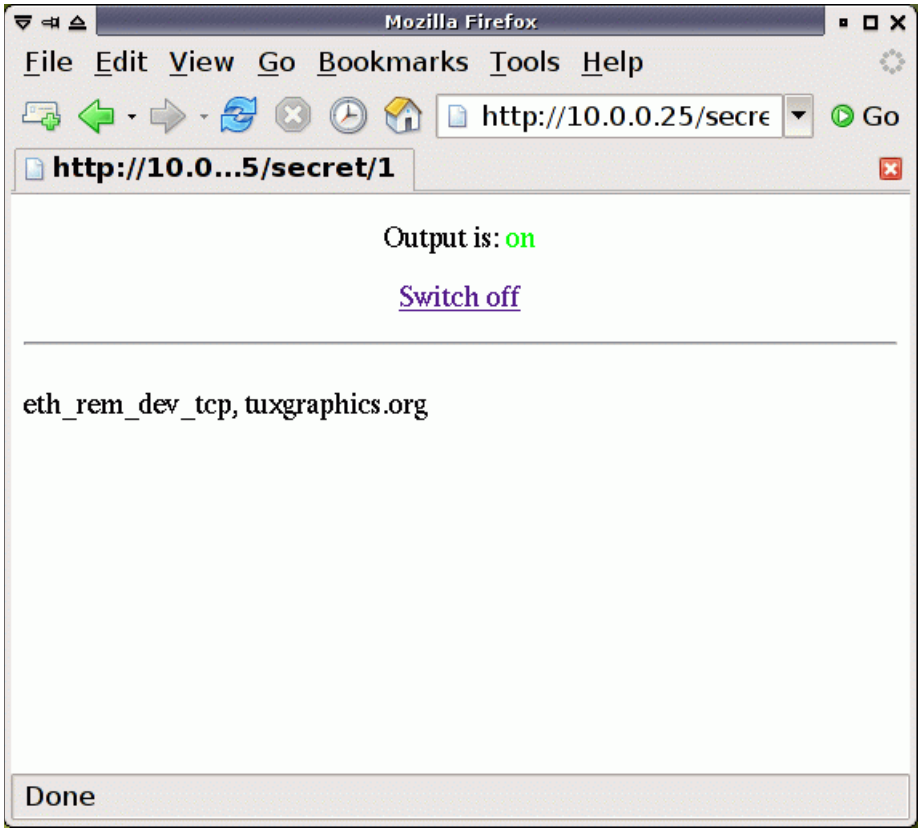

Figure 2: The atmega88 based web-server, screenshot with mozilla firefox

A Single Data Packet TCP/HTTP web-server on a microcontroller

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:

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

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.

The URL format

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.

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

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

```
See the current status and change nothing:
  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

The web-server hardware

The hardware is exactly the same as described in the previous article An AVR microcontroller based Ethernet device:

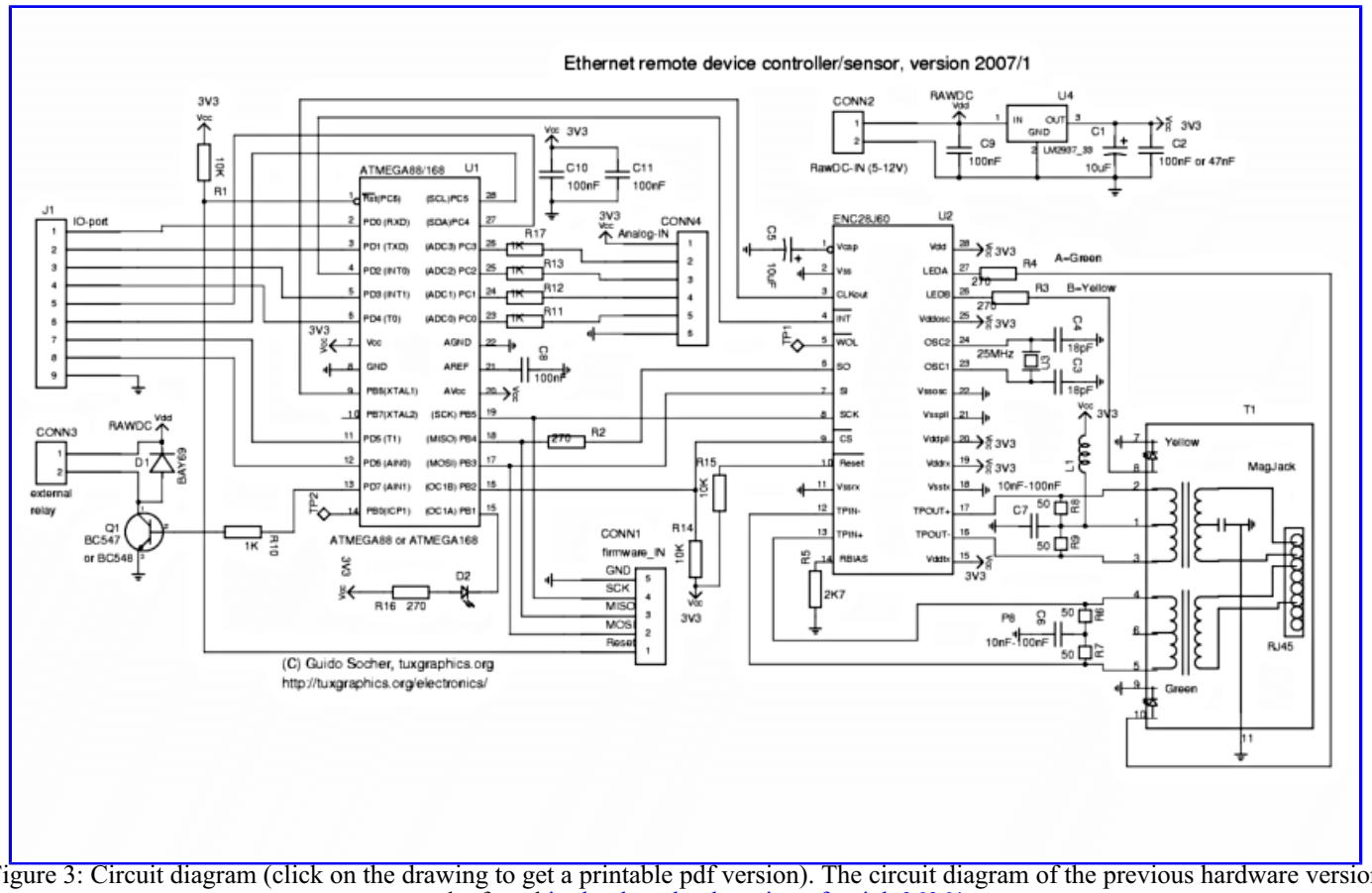

Figure 3: Circuit diagram (click on the drawing to get a printable pdf version). The circuit diagram of the previous hardware version can be found in the download section of article06061

You get the web-server just by uploading new firmware to the microcontroller.

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.

Next you need to set the IP address for your hardware. Edit the file main.c and change the 3 lines:

```
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.

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

```
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).

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 ;-)

Performance

Embedded systems are generally small in size. Not only physically small in size but also small in terms of memory and CPU speed.

All embedded TCP/IP stacks have therefore a rather low limit on the number of parallel users (e.g 2-5 parallel http connections).

The tuxgraphics stack takes a different approach. The amount of data that we want to display is generally small. Maybe you would want to display the values of attached sensors or you want to switch on/off something as in the above example. To display that on a web page requires just a few bytes of data. We limit the size of the web page to a few hundred bytes. With this limit in place we don't have to have a limit on the number of parallel connection.

The avr microcontroller is amazing CPU. Most operations are done in just one CPU clock cycle. With the The tuxgraphics TCP/IP stack on this CPU we can get really top performance out of this web server. There is no hard-coded limit to the number of users and it can serve hundreds of web pages per second. Some small PC based web servers would have trouble to keep up with it.

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.

2009-03-27, generated by tuxgrparser version 2.56