D-LINK DWL-G122 USB WiFi dongle

under GNU/Linux

1. USB WiFi dongle

We decided to work with USB WiFi dongles instead of PCI cards just because the dongles have mobility to be moved from one computer to another and because they were not so expensive, besides a team member had previous experience with these devices.

The device chosen was the D-Link DWL-G122 [1], known to work under GNU/Linux.

We had two possibilities to get the dongle working under GNU/Linux:

- Windows driver emulation (ndiswrapper [2])
- Native driver

1.1. Windows driver emulation under ndiswrapper

As the device was reported to work with ndiswrapper [3] we decided to try at first with it because the installation was supposed to be very easy.

The first step to get ndiswrapper working is download it from the Internet [4], then uncompress it:

```
[root@workstation ndiswrapper]# tar -xzf ndiswrapper-1.16
```

and read the *README* and *INSTALL* files.

From the *INSTALL* file we obtain the information of how to install ndiswrapper:

```
[root@workstation ndiswrapper-1.16]# make uninstall
[root@workstation ndiswrapper-1.16]# make
[root@workstation ndiswrapper-1.16]# make install
```

Now we need the original Windows driver to be loaded by ndiswrapper. Usually this files are binary files for the firmware (.bin), and the system information (.sys & .inf), unfortunately in our CD Rom the files were, probably, inside ‘SETUP.EXE’, that we couldn’t open to extract the files we needed.

So we had to look on the Internet again and found the drivers for our revision [5].

We found the needed drivers inside the downloaded file and thanks to the help of the ndiswrapper *wiki* web page [3].

```
Driver: Dr71WU.inf and Dr71WU.sys from the CD
```

After creating a directory called ‘win_drivers’ inside the ndiswrapper directory, we moved four files called the same but with different file formats (inf, sys, bin, cat) and then the drivers had to be loaded by ndiswrapper.

```
[root@workstation ndiswrapper-1.16]# ndiswrapper -i  Dr71WU.inf
[root@workstation ndiswrapper-1.16]# ndiswrapper -l
Installed drivers:
dr71wu          driver installed, hardware present
```

So everything seem to be correct, time to load the kernel module:

```
[root@workstation ndiswrapper-1.16]# modprobe ndiswrapper
```

The dongle started working and we could make a normal *WiFi scanning* but somehow we could not change the mode of the device from *Ad-Hoc* to *Managed* which basically means that we could only create or join a 2 points device network, not join to a network managed by an Access Point.

```
[root@workstation ~]# iwconfig wlan0 mode managed
[root@workstation ~]# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

eth1      no wireless extensions.

sit0      no wireless extensions.

Warning: Driver for device wlan0 recommend version 18 of Wireless Extension,
but has been compiled with version 17, therefore some driver features
may not be available...

wlan0     IEEE 802.11g  ESSID:"netlab"  Nickname:"workstation.localdomain"
          Mode:Ad-Hoc  Frequency:2.462 GHz  Cell: 00:00:00:00:00:00
          Bit Rate=11 Mb/s   Tx-Power:20 dBm   Sensitivity=-121 dBm
          RTS thr=2347 B   Fragment thr=2346 B
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Also using the command `iwlist` with the option `scanning` a waring message alike the one above is shown:

```
[root@workstation ~]# iwlist wlan0 scanning
```

So we tried to find out why it wasn’t working. Again, at the ndiswrapper *wiki* page we found information on how to install it under Fedora Core and troubleshooting:

```
http://ndiswrapper.sourceforge.net/mediawiki/index.php/Fedora
```

The important feature was the 4k size kernel stack [6].

So we downloaded new kernels from the mentioned web pages [7, 8] and tried to make them work. We couldn’t make them work since the dependencies of the kernel packages

```
[root@workstation Downloads]# unzip kernel-2.6.16-
1.2108_FC4.stk16.i686.rpm.zip
'Archive:  kernel-2.6.16-1.2108_FC4.stk16.i686.rpm.zip
  inflating: kernel-2.6.16-1.2108_FC4.stk16.i686.rpm
[root@workstation Downloads]# rpm -Uhv kernel-2.6.16-
1.2108_FC4.stk16.i686.rpm
error: Failed dependencies:
        kernel = 2.6.11-1.1369_FC4 is needed by (installed) cman-kernel-
2.6.11.5-20050601.152643.FC4.2.i686
        kernel = 2.6.11-1.1369_FC4 is needed by (installed) dlm-kernel-
2.6.11.5-20050601.152643.FC4.2.i686
        kernel = 2.6.11-1.1369_FC4 is needed by (installed) GFS-kernel-
2.6.11.8-20050601.152643.FC4.2.i686
        kernel = 2.6.11-1.1369_FC4 is needed by (installed) gnbd-kernel-
2.6.11.2-20050420.133124.FC4.35.i686
        /lib/modules/2.6.11-1.1369_FC4 is needed by (installed) cman-kernel-
2.6.11.5-20050601.152643.FC4.2.i686
        /lib/modules/2.6.11-1.1369_FC4 is needed by (installed) dlm-kernel-
2.6.11.5-20050601.152643.FC4.2.i686
        /lib/modules/2.6.11-1.1369_FC4 is needed by (installed) GFS-kernel-
2.6.11.8-20050601.152643.FC4.2.i686
        /lib/modules/2.6.11-1.1369_FC4 is needed by (installed) gnbd-kernel-
2.6.11.2-20050420.133124.FC4.35.i686
```

We decided not to continue trying to get ndiswrapper working since it made no sense writing a section at the «How-To» about WiFi which needs a kernel replacing, something not at all for beginners.

1.2. Native driver

Having any kind of device working with it's own native driver is always much better than running the device with software emulation, despite we tried first with ndiswrapper because there was no comment on the ndiswrapper *wiki* about the native driver for the chipset our dongle is using.

The installation of the driver is just a bit more complicated that the ndiswrapper. First of all we have to download the last driver for our dongle from the website of the manufacturer [9], as we already knew, from the ndiswrapper *wiki,* our driver is RT73 (RT2571W) [10].

Then it has to be uncompressed into a directory, usually inside `/usr/src/`:

```
[root@workstation ~]# cd /usr/src/
[root@workstation src]# mkdir rt73
[root@workstation rt73]# cd rt73
[root@workstation rt73]# cp
/root/Desktop/Downloads/RT73_Linux_STA_Drv1.0.3.0.tar.gz ./
[root@workstation rt73]# tar zvxf RT73_Linux_STA_Drv1.0.3.0.tar.gz
```

jump into the `Module` directory to see which important files should be read (*REAMDE*, *INSTALL*, ...) . There is a *README* file that explains very well how to compile the driver, the configuration file, the firmware and some other things it needs before inserting it into the kernel.

Following the *README* file, first, as our Fedora Core 4 are running 2.6.X kernel series, we copy the file Makefile.6 to Makefile:

```
[root@workstation Module]# cp Makefile.6 ./Makefile
```

Then the *Configure* file must have execution permit:

```
[root@workstation Module]# chmod 755 Configure
[root@workstation Module]# ./Configure
```

Running the *Configure* file we have to insert the path where our kernel is:

```
/usr/src/kernels/2.6.11-1.1369_FC4-i686
```

Before continuing with the *README* instructions we'll patch a header file, due a missing part inside the file ***`rtmp_def.h`*** our dongle will not be recognized by the driver [11].
In the *VENDOR* section -line 800- two new lines must be added:

```
#define DLVID1 0x07d1
#define DLPID1 0x3c03
```

And below that section, where the USB devices are defined:

#define RT73_USB_DEVICES { \

this new line must be added:

```
{USB_DEVICE(DLVID1,DLPID1)}, \
```

Once this is done we continued following the *README* file.

```
[root@workstation Module]# make all
```

Now we have to move the configuration and the firmware files to a specified folder -that we'll have to create as well-, but before the configuration file must be in UNIX format:

```
[root@workstation Module]# dos2unix rt73sta.dat
[root@workstation Module]# cd /etc/
[root@workstation etc]# mkdir Wireless
[root@workstation etc]# cd Wireless
[root@workstation Wireless]# mkdir RT73STA
[root@workstation Wireless]# cd RT73STA
[root@workstation RT73STA]# cp
/usr/src/rt73/RT73_Linux_STA_Drv1.0.3.0/Module/rt73.bin ./
[root@workstation RT73STA]# cp
/usr/src/rt73/RT73_Linux_STA_Drv1.0.3.0/Module/rt73sta.dat ./
```

NOTE: the `rt73sta.dat` configuration file should be edited following the *README* file in order to get it work automatically during the bootup process.

We can also create a file in order to get the dongle configurated on bootup. For our Fedora Core the file *`ifcfg-rausb0`* will be located into:

`/etc/sysconfig/network-scripts/`

The basic configuration is:

```
   DEVICE='rausb0'
   ONBOOT='yes'
```

We can also set auto configuration with DHCP:

```
   BOOTPROTO=dhcp
```

Or set a static configuration with other details:

```
   BOOTPROTO=static
   IPADDR=192.168.0.169
   NETMASK=255.255.255.0
   GATEWAY=194.182.170.1
```

NOTE2: under debian you can set all this in the file '*`interfaces`*' located into the folder *`/etc/network/`*.

Now we must set an alias for the 'rt73' to be recognized as 'rausb0', as we configured the device with that name before.
In the file *`/etc/modprobe.conf`*-for our Fedora Core 4- add the following:

```
alias rausb0 rt73
```

NOTE3: under other distributions the file should be '*modules.conf*' instead of '*modprobe.conf*'.

The next step is to insert the driver into the kernel:

```
[root@workstation RT73STA]# cd
/usr/src/rt73/RT73_Linux_STA_Drv1.0.3.0/Module/
[root@workstation Module]# insmod rt73.ko
```

And now executing ‘*iwconfig*’ we’ll see if our card is recognized by our GNU/Linux box:

```
[root@workstation ~]# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

eth1      no wireless extensions.

sit0      no wireless extensions.

rausb0    RT73 WLAN  ESSID:""
          Mode:Auto  Frequency=11 MHz  Access Point: 00:03:47:15:BC:CC
          Bit Rate=11 Mb/s
          RTS thr:off   Fragment thr:off
          Encryption key:off
          Link Quality=91/100  Signal level:-48 dBm  Noise level:-79 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Now if we configured correctly the files *rt73sta.dat* and *ifcfg-rausb0* mentioned before, using the command ***ifup rausb0*** the device should be set up, for example:

```
[root@workstation ~]# ifup rausb0
[root@workstation ~]# ifconfig rausb0
rausb0    Link encap:Ethernet  HWaddr 00:15:E9:A3:3D:8F
          inet addr:192.168.0.169  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::215:e9ff:fea3:3d8f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:314 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:114392501 (109.0 MiB)  TX bytes:2874704 (2.7 MiB)
```

2. Referrals

[1]: http://www.dlink.com/products/?pid=334

[2]: http://ndiswrapper.sourceforge.net/

[3]: http://ndiswrapper.sourceforge.net/mediawiki/index.php/List#D

[4]: http://sourceforge.net/project/showfiles.php?group_id=93482

[5]: ftp://ftp.dlink.it/pub/Wireless Extreme G/DWL-G122/Driver/HW_C1/

-> DWL-G122_C1_V3.00.zip

[6]: http://ndiswrapper.sourceforge.net/mediawiki/index.php/Fedora

-> #4k_kernel_stack_size.2Ffreezing_issue

[7]: http://www.linuxant.com/driverloader/wlan/full/downloads-fc4-kernel-i686.php

[8]: http://www.linuxant.com/driverloader/wlan/full/downloads-fc4-kernel-i586.php

[9]: http://www.ralinktech.com/supp-1.htm

[10]: http://www.ralinktech.com/drivers/Linux/RT73_Linux_STA_Drv1.0.3.0.tar.gz

[11]: http://61.222.76.235/phpbb2/viewtopic.php?t=2197