Use Zydas based WLAN USB adaptor with ADNP/9200 (2.6 Kernel)

This is a short howto that describes the necessary steps to make the ADNP/9200 platform ready for wireless networks using a commercial USB WLAN adaptor. In this case we're using a QuickWLAN USB 54 adaptor manufactured by Yakumo. The device uses a chipset from Zydas (ZD1211b), which is very common, and you may find many other adaptors on the market using this chipset. So let's start!

Check your environment and sources

First it is assumed that you already have a working development environment on your host PC, which will mean you have a working cross compiler and the necessary 2.6 kernel source files to generate working kernel images for the ADNP/9200 device. You will find everything you need on the Starter-Kit CDROM. Follow the given instructions to set up your environment on your host.

You will also need some sources from the internet to compile the driver for Zydas chipset and a wireless toolset to configure your WLAN adaptor (iwconfig, iwlist, iwpriv etc.).
The source code (version 2.15.0) for the ZD1211B can be found at the Atheros website at:
http://www.atheros.com/RD/ZyDAS/web_driver/ZD1211B/Linux/ZD1211LnxDrv_2_15_0_0.tar.gz
The wireless toolset (version 28) can be downloaded at:
http://pcmcia-cs.sourceforge.net/ftp/contrib/wireless_tools.28.tar.gz

Prepare the kernel

The first step to make the ADNP/9200 use WLAN, is to enhance the kernel with the wireless extension. You may call ***make xconfig*** or ***make menuconfig*** to do so.
Next go through the kernel configuration and activate the following item in the section:
->Device Drivers
->Network device support
->Wireless LAN (non hamradio)
->Wireless LAN drivers (non hamradio) & Wireless Extensions

Please activate this setting (Wireless LAN drivers (non hamradio) & Wireless Extensions) for the wireless extension and save your configuration on exit. Now you're ready to compile the kernel image by typing:
make Image
at the console prompt on your host. After a successful compilation you can find your new kernel in:
/directory_to_your_kernel_source/linux-2.6.16.20-at91-ssv2/arch/arm/boot

The 2.6.16.20 kernel for the AT91 uses version 19 of the wireless extension API. Keep this in mind, because we will also use this version information for compiling the wireless tools for the ADNP/9200.

To get the new Kernel running on the ADNP/9200, copy it into a directory we're you put the tools to generate ROM-images for the platform. These tools can also be found on the CDROM.
The main tool is a script called mkimage.sh. Use this script to put the kernel and the root filesystem together to one file, that can be loaded by the ADNP/9200's bootloader.
Please type ***sh mkimage.sh*** at your console prompt. The result will be: image-dnp9200.bin

Download this image to your platform using the tftp download feature of the bootloader. You may test the new image immediately by typing ***bootm 0x2100000*** at the ADNP/9200 bootloader console prompt. If your linux system boots up without any errors everything went fine and you may store this new image to the ADNP/9200 onboard flash. Please refer to the ssv-embedded systems website how to do this.

Cross-compile the driver

The following instructions will guide you through the necessary steps to compile the driver for the Zydas chipset:

1. Change the Makefile of the ZD1211 (2.15.0) driver source code
   - Insert your arm-linux cross-compiler
   - Modifiy the path entry to point at your AT91 kernel sources
   - Modifiy the path entry to point at the module directory for AT91 modules
   - Configure entry to use flags for 2.6 Kernel
2. Now your ready to cross-compile the driver!!
   - Type ***make*** in your working directory
   - The driver can be found in your AT91 module directory

Cross-compile the wireless toolset

The following instructions will guide you through the necessary steps to compile the wireless toolset for ARM:

1. Change the Makefile of the wireless tools source code
   - Insert your arm-linux cross-compiler
   - Configure entry to compile static
   - Configure entry not to use libm
2. Modify header file to compile for wireless extension version 19:
   - Modify line 134 in the file iwlib.h from ***WE_VERSION=20*** to ***WE_VERSION=19***
3. Now your ready to cross-compile the wireless toolset!!
   - Type ***make*** in your working directory

Connect wireless

Now we're ready to run our new enhancement. For this first test you will need a wireless accesspoint which is configured with WEP security (64-Bit key in this example, 128Bit should also work).

1. copy the kernel driver (zd1211b.ko) and the wireless tools (iwconfig, iwlist, iwpriv) to ADNP/9200
2. connect USB WLAN adaptor to the ADNP/9200 starter-kit
3. load the kernel driver from within the console:

```
[root@emblinux modules]$insmod zd1211b.ko


 _____      ____    _    ____
|__  /    _|  _ \  / \  / ___|
  / / | | | | | |/ _ \ \___ \
 / /| |_| | |_| / ___ \ ___) |
/____\__, |____/_/   \_\____/
     |___/
ZD1211B - version 2.15.0.0
vendor_id = 0ace
product_id = 1215
USB 1.1 Host
Release Ver = 4810
EEPORM Ver = 4810
Finsih download Firmware. Ready to reboot
PA type: 0
PHYNEWLayout = 1
Airoha AL2230S_RF
AllowedChannel = 000107ff
Region:48
usbcore: registered new driver zd1211b
[root@emblinux modules]$
```

4. configure fixed IP-address:

```
[root@emblinux wireless]$ifconfig eth2 192.168.100.110
[root@emblinux wireless]$ifconfig eth2
eth2      Link encap:Ethernet  HWaddr 00:02:72:57:26:88
          inet addr:192.168.100.110  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

5. configure eth2 with your wireless network SSID, WEP-key (HEX-values) and activate the key for the adaptor:

```
[root@emblinux wireless]$./iwconfig eth2 essid "my_wlan"
[root@emblinux wireless]$./iwconfig eth2 key A1B2C3D4E5 [1]
[root@emblinux wireless]$./iwconfig eth2 key [1] open
```

6. check your settings

```
[root@emblinux wireless]$./iwconfig eth2
eth2      802.11b/g NIC  ESSID:"my_wlan"
          Mode:Managed  Frequency=2.422 GHz  Access Point: 00:0C:41:9D:09:DB
          Bit Rate:11 Mb/s
          Retry:off   RTS thr=9999 B   Fragment thr:off
          Encryption key:****-****-** [1]   Security mode:open
          Power Management:off
          Link Quality:98/100  Signal level:72/100  Noise level:0/100
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:293   Missed beacon:0
```

7. check the connection with a ping to your wireless accesspoint

```
[root@emblinux wireless]$ping 192.168.100.1
PING 192.168.100.1 (192.168.100.1): 56 data bytes
64 bytes from 192.168.100.1: icmp_seq=0 ttl=64 time=6.3 ms
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=3.5 ms
64 bytes from 192.168.100.1: icmp_seq=2 ttl=64 time=2.7 ms
64 bytes from 192.168.100.1: icmp_seq=3 ttl=64 time=2.8 ms
64 bytes from 192.168.100.1: icmp_seq=4 ttl=64 time=2.9 ms
64 bytes from 192.168.100.1: icmp_seq=5 ttl=64 time=4.1 ms
64 bytes from 192.168.100.1: icmp_seq=6 ttl=64 time=3.2 ms
64 bytes from 192.168.100.1: icmp_seq=7 ttl=64 time=2.4 ms

--- 192.168.100.1 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 2.4/3.4/6.3 ms
[root@emblinux wireless]$
```

It’s done! Welcome to the wireless world on your ADNP/9200 embedded device.
Now you can access the ADNP/9200 onboard webserver and browse the platform via ftp wireless. To make it more comfortable to connect to the internet you have to add some settings on the ADNP/9200 (e.g. nameserver, gateway).