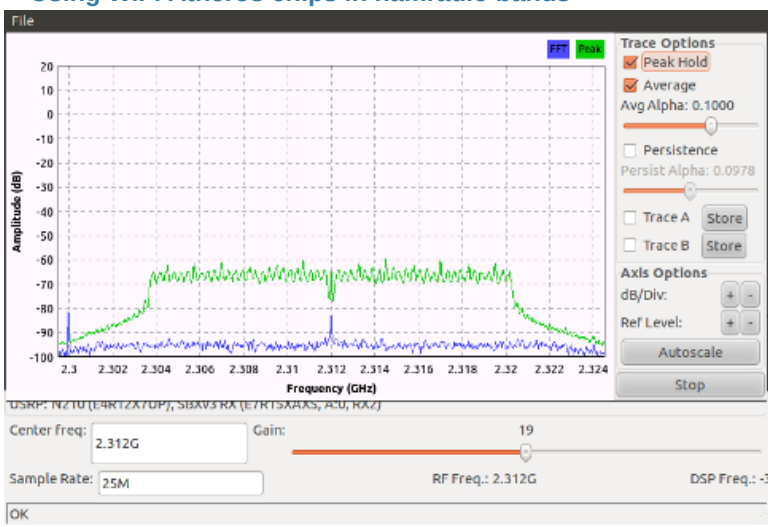
Radio Adventures

YO3IIU in hamradio

Using WiFi Atheros chips in hamradio bands



DVB-T implementation in GNUradio – part 3

Recent Posts

- Using WiFi Atheros chips in hamradio bands
- DVB-T implementation in GNUradio – part 3
- DVB-T implementation in GNUradio – part 2
- DVB-T implementation in GNUradio – part 1
- Running GPS software on USRP
- CDMA sensing with USRP
- Evaluating GSM A5/1 security on hopping channels
- A day with my personal GSM network
- Bang&Olufsen
- Running app_rpt on ASUS WL500
- Sending APRS message to Twitter
- Getting METAR weather through APRS
- Digital Coded Squelch codec part II
- Digital Coded Squelch codec part I
- Wideband transformers III (practical applications)

Archives

- November 2013
- July 2013
- June 2013
- February 2013
- April 2012
- September 2011
- February 2011
- November 2010
- October 2010
- August 2010
- July 2010
- June 2010
- May 2010

January 2014

M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

« Nov

NOV 13
30

Using WiFi Atheros chips in hamradio bands

Motivation:

I was aware for long time that Atheros (bought by Qualcom in May 2011) wifi chips can work in the below 2.4Ghz band and lately I saw an old post from here confirming that but I had never tried this.

So I thought it will be an interesting and not so complex experiment to try this out:

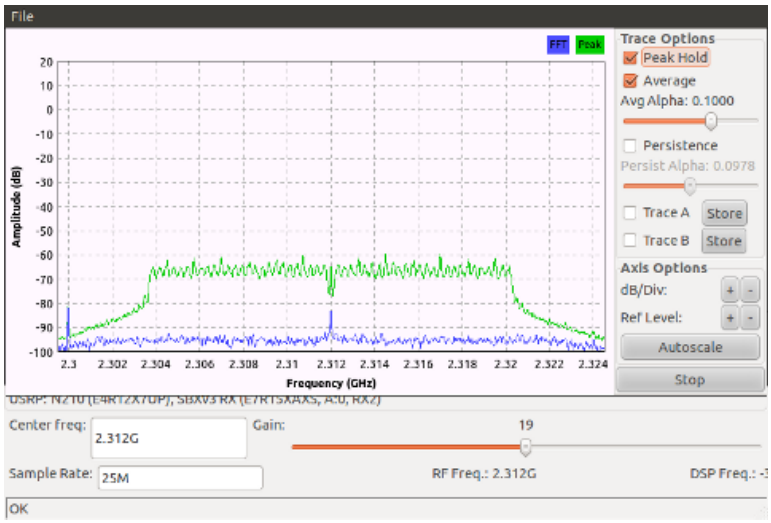
1. use wifi USB sticks in the 2300-2400Mhz band

2. increase the power level to more than the 20dBm which is the current limit

3. verify all of the above #1 and #2 in a real life environment

Note: check your country's legislation and your hamradio license to understand the conditions

under you can do this.



The HW:

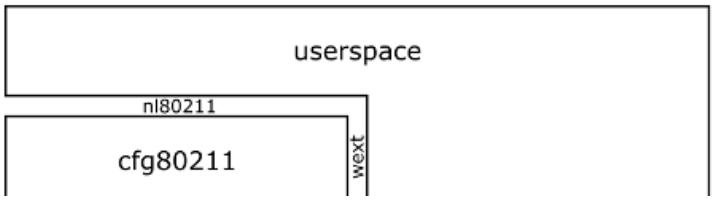
I bought two TL-WN722N WiFi USB sticks, they are under 10EUR each. The dongles have the AR9271 chips:

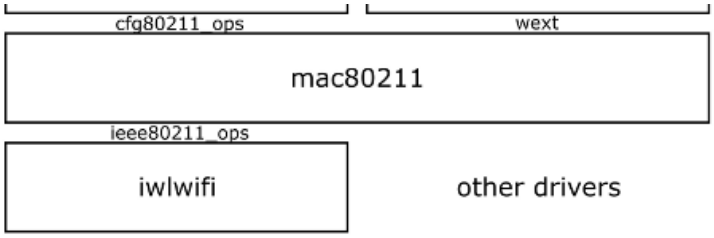
Having a connector for an external antenna makes room for a lot of experiments, including using of high gain point-to-point antennas.

The SW:

A bit about Wireless stack on Linux:

The AR9271 chip is using a kernel driver and a SoftMAC driver which is the software implementation of the MLME (MAC Layer Management Entity). However, other chips implement this sublayer into HW. The wireless stack in Linux kernel is presented below and a good overview is presented here.





Linux kernel HW driver changes:

The AR9271 chip actually does not know about wireless channels and that seems to be the solution to add more frequencies in the hamradio bands. The driver that knows AR9271 is ath9k_htc. I added a number of channels below channel 1 of 2.4Ghz by modifying the channel table in *drivers/net/wireless/ath/ath9k/htc_drv_init.c* like in the following snippet:

…

```
#define CHAN2G(_freq, _idx)  { \
.center_freq = (_freq), \
.hw_value = (_idx), \
- .max_power = 20, \
+ .max_power = 30, \
}
```

…

```
CHAN2G(2397, 16), /* Channel -2 */
CHAN2G(2402, 15), /* Channel -1 */
CHAN2G(2407, 14), /* Channel 0 */
CHAN2G(2412, 0), /* Channel 1 */
CHAN2G(2417, 1), /* Channel 2 */
```

…

Another change to do is in *drivers/net/wireless/ath/ath9k/regd.c*:

```
-#define ATH9K_2GHZ_CH01_11 REG_RULE(2412-10, 2462+10, 40, 0, 20, 0)
+#define ATH9K_2GHZ_CH01_11 REG_RULE(2312-10, 2462+10, 40, 0, 30, 0)
```

The new channels will appear as negative channels and for that there is another change to do, this time in *net/wireless/util.c*, *ieee80211_channel_to_frequency()* function:

```
} else { /* IEEE80211_BAND_2GHZ */
+
+    chan = (int)(char)chan;
+
```

This is because the channel is seen internally as unsigned number and it need to be transformed in a negative number so that the returned frequency will be the right one. The channels in 80211 are separated by 5Mhz and how the new ones are seen in user space is shown below.

Recompiling the kernel and installing the new drivers should give the following capabilities of the wireless device:

>iw list

```
Wiphy phy0
Band 1:
Capabilities: 0×116e
HT20/HT40
SM Power Save disabled
RX HT20 SGI
RX HT40 SGI
RX STBC 1-stream
Max AMSDU length: 7935 bytes
DSSS/CCK HT40
Maximum RX AMPDU length 65535 bytes (exponent: 0×003)
Minimum RX AMPDU time spacing: 8 usec (0×06)
HT TX/RX MCS rate indexes supported: 0-7
Frequencies:
* 2312 MHz [-19] (30.0 dBm)
* 2317 MHz [-18] (30.0 dBm)
* 2322 MHz [-17] (30.0 dBm)
* 2327 MHz [-16] (30.0 dBm)
* 2332 MHz [-15] (30.0 dBm)
* 2337 MHz [-14] (30.0 dBm)
* 2342 MHz [-13] (30.0 dBm)
* 2347 MHz [-12] (30.0 dBm)
* 2352 MHz [-11] (30.0 dBm)
* 2357 MHz [-10] (30.0 dBm)
* 2362 MHz [-9] (30.0 dBm)
* 2367 MHz [-8] (30.0 dBm)
```

* 2372 MHz [-7] (30.0 dBm)
* 2377 MHz [-6] (30.0 dBm)
* 2382 MHz [-5] (30.0 dBm)
* 2387 MHz [-4] (30.0 dBm)
* 2392 MHz [-3] (30.0 dBm)
* 2397 MHz [-2] (30.0 dBm)
* 2402 MHz [-1] (30.0 dBm)
* 2407 MHz [0] (30.0 dBm)
* 2412 MHz [1] (30.0 dBm)
* 2417 MHz [2] (30.0 dBm)

…

Regulatory Database changes:

The regulatory database (regdb) is the Linux solution to achieve legal regulation for each country the machine is used in. It is basically a user space database that lists specifics like allowed bands, channels enabled, power levels, antenna gain, etc for each country. The WiFi chips usually have an EEPROM to store various similar details and those will be intersected with the details from the regdb set by the system/user.

The idea of regdb is to allow a roaming user's machine to accomodate the settings of wifi while going to other countries than the base country.

Regdb is presented in a signed binary form compiled from a text in a user readable form. One needs to modify the database in text format and then compile it together with the necessary keys. I will not explain this as it is better explained on wireless kernel pages.

Wireless-regdb

There are basically two packages one need to build and install after one another. The first one is wireless-regdb that generates the binary form of regdb from it's text form. My wif USB stick came with CN as the country set in EEPROM so the CN will be modified into db.txt:

country CN:

(2402 – 2482 @ 20), (N/A, 20)

(2302 – 2482 @ 20), (N/A, 30)

Central Regulatory Domain Agent (CRDA)

CRDA is the user space app that kernel uses to interrogate the regdb database. It is mandatory to be rebuild and installed together with the keys generated in the wireless-regdb build process.

Hostapd changes (for v2.0):

One of the goals of this project is to run an Access point (AP) to which a client (STA) will connect and communicate. Scanning, association, sending, receiving data should work.

One of quick solution for running an AP is hostapd package because it implements MLME in user space. One need to change hostapd to accept the negative channels we have just been added.

In *src/ap/hw_features.c:hostapd_select_hw_mode()*

- if (chan->chan == iface->conf->channel) {

+if ((0xff & chan->chan) == iface->conf->channel) {

In *src/ap/hw_features.c:*hostapd_hw_get_freq()

-if (ch->chan == chan)

+if ((0xff & ch->chan) == chan)

In *hostapd.conf*:

channel=-19 or whatever the card shows that it supports

Run hostapd after it is built:

./hostapd ./hostapd.conf

And now we have the AP running on the channel -19 (freq 2312Mhz):

>iwconfig

wlan5 IEEE 802.11bgn Mode:Master Frequency:2.312 GHz Tx-Power=30 dBm
Retry long limit:7 RTS thr:off Fragment thr:off
Power Management:off

mon.wlan5 IEEE 802.11bgn Mode:Monitor Tx-Power=30 dBm
Retry long limit:7 RTS thr:off Fragment thr:off
Power Management:off

Real life tests:

We now need to connect a client (STA) to the AP so this is done on another Linux machine with the same changes done on the kernel.

Scanning:

>iw scan

BSS 10:fe:ed:14:2f:bd (on wlan0)
TSF: 399462784 usec (0d, 00:06:39)
freq: 2312
beacon interval: 100
capability: ESS ShortSlotTime (0×0401)
signal: -49.00 dBm
last seen: 168 ms ago
Information elements from Probe Response frame:
SSID: test
Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
DS Parameter set: channel 237
ERP: Barker_Preamble_Mode
Extended supported rates: 24.0 36.0 48.0 54.0
WMM: * Parameter version 1
* BE: CW 15-1023, AIFSN 3
* BK: CW 15-1023, AIFSN 7
* VI: CW 7-15, AIFSN 2, TXOP 3008 usec
* VO: CW 3-7, AIFSN 2, TXOP 1504 usec

Connect:

This can be done using regular UI tools on Linux like nm-applet. If the dhcp is not available on AP then a static IP address is to be given:

On AP: ifconfig wlan0 192.168.30.1

On STA: ifconfig wlan0 192.168.30.2

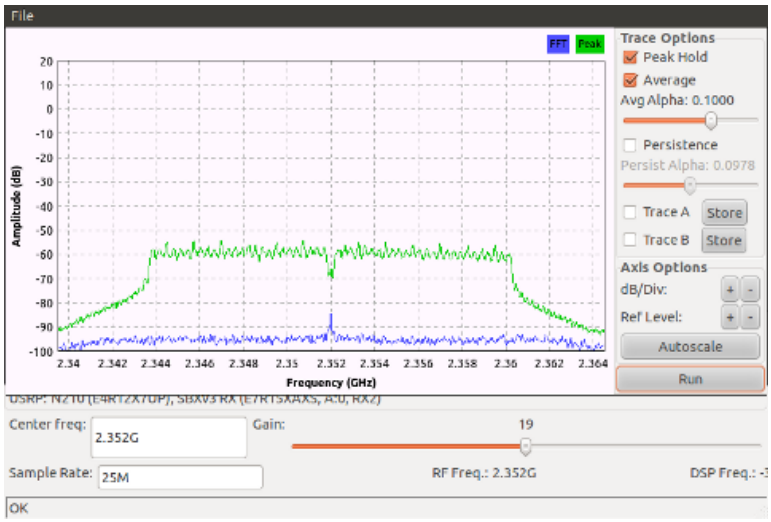
Show the connection:

>iwconfig

wlan0 IEEE 802.11bgn ESSID:"test"
Mode:Managed Frequency:2.312 GHz Access Point: 10:FE:ED:14:2F:BD
Bit Rate=54 Mb/s Tx-Power=30 dBm
Retry long limit:7 RTS thr:off Fragment thr:off
Power Management:off
Link Quality=70/70 Signal level=-24 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:7 Missed beacon:0

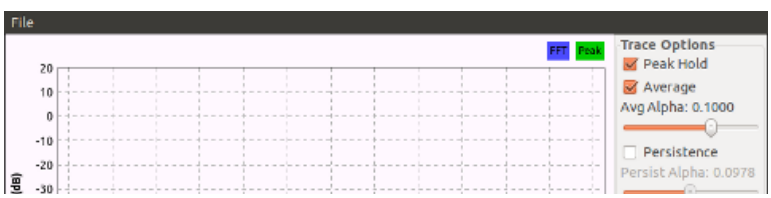
Check the communication is on selected channel:

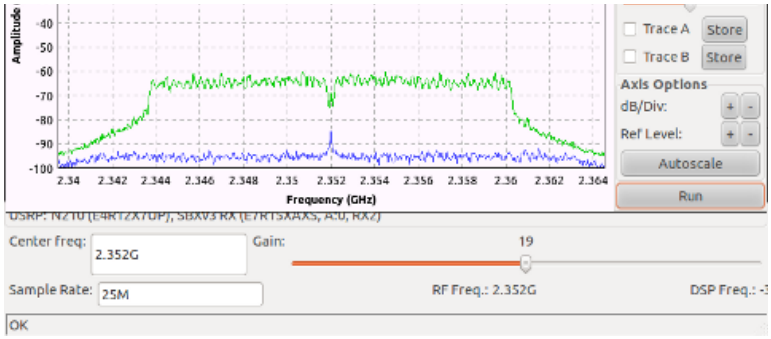
I choose another channel -11 <=> 2352Mhz to see if I can see a change in the frequency domain. It is important to have an ongoing data transfer between the AP and STA in order to see the shape of the waveform and for that I used NC (Network Cat):

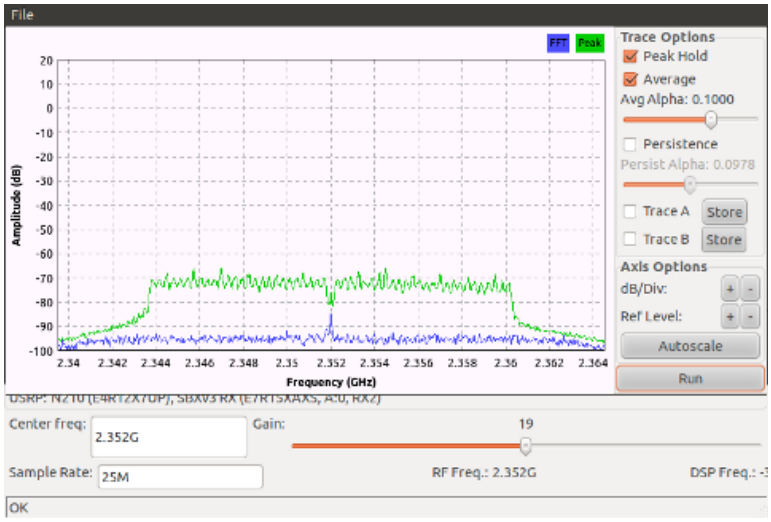


Check the change in the power level:

The above picture is for the 30dBm power limit. The next one is for the 20dBm and one with 10dBm transmit power. There is a visible difference between the three images, however probably a more precise measurement needs to be done.







Source code:

Patches for the kernel code and hostapd are here: atheros_hambands.tgz

Conclusions:

- it was possible to add 20 channels from 2312Mhz to 2407Mhz and the connection between similar wifi devices is possible.

- there seems to be no difference in the transmit power for any channel between 2312Mhz (channel -19) to 2472Mhz (channel 13).

- there is an increase of power level when changing above 20dBm although this needs to be checked with finer tools and also the thermal regime of the devices needs to be taken into consideration.

- the same traffic speed is obtained for each channel compared with regular channels (1-14).

Next steps:

- to create a point-to-point link between two wifi devices using high gain antennas. An excel file calculating the theoretical results is here: Radio_Link_Budget_Calc.xls.

- play around with spectral scan

Posted in Uncategorized by admin. No Comments

JUL 13
16

DVB-T implementation in GNUradio – part 3