HP WI-FI DIRECT MOUSE ON LINUX

In my last post I took apart an HP Wi-Fi Direct mouse based on the OZMO2000 controller from Ozmo Devices. OZMO based devices are officially supported on Windows 7 platform only, rendering them completely useless as nobody uses Windows anymore… right? Well, it turns out most of the code to use them in Linux is already in place, just waiting to be enabled!

Wi-Fi direct support in Linux is quite young and still considered as experimental, so read on if you dare to try!

OZMO Devices: Theory of Operation

OZMO controller uses Wi-Fi Direct (a.k.a. Wi-Fi P2P) for communication with the host system, so a Wi-Fi Direct Linux supported card is the only hardware requirement.

Wi-Fi Direct itself is a relatively recent extension of the 802.11 standards to allow Wi-Fi devices to associate and communicate between themselves (point-to-point) without requiring a dedicated common AP but also without the need to disable the existing traditional Wi-Fi connection. This is implemented using an additional “virtual interface” that acts as a software access point (P2P-GO) with WPS push-button client support.

Once the P2P-GO interface is up and running, the mouse can be connected using the WPS push-button scheme: press the soft-button on the host and the “connect” button on the mouse that will automatically associate with the first available AP.

As the device is connected, the real fun begins: as the mouse starts transmitting data to the host using a specific Ozmo’s in-house developed USB over Ethernet protocol, registered to ethertype `0x892e`.

Quoting IEEE’s eth.txt:

```
892E         Ozmo Devices
```

```
A protocol to enable USB type data and USB type commands to be sent

between two devices connected by an IEEE802 link, typically an

IEEE802.11 link. The protocol allows encoding of all information

that would normally be present in a wired USB request, such as

request_id and rcode along with payload data. Defined header

structures allow correct delivery of the required fields. The

informaion provided through the protocol enables USB like behaviour

to be implemented between wirelessly connected

devices. www.ozmodevices.com
```

Sounds complicated? This is actually quite ingenious, as it allows a gadget designer to implement an USB Class protocol, so that once the device is associated and the “virtual USB hub” is installed the actual device-specific protocol can be handled by the already existing USB device-class driver.
Thus, the only OZMO specific code to be installed by the user is the actual virtual USB hub – backed on the Wi-Fi Direct interface. The good news is that the Linux driver is already here, silently laying in the `staging/ozwpan`directory of modern kernels… Ozmo itself made it, well done Ozmo!
So, how do we use it?

Kernel

First thing we need is to make sure that our Wi-Fi device supports Wi-Fi Direct, which you can tell by running the `iw list` command and searching for P2P entries between supported interface modes, as in:

```
# iw list

Wiphy phy0

[...]

        Supported interface modes:

                 * IBSS

                 * managed

                 * AP

                 * AP/VLAN

                 * monitor
```

```
                 * P2P-client

                 * P2P-GO

[...]
```

In this case I'm using an Intel's Advanced-N 6205 mini-PCIe card. If you are using an `iwlwifi` based card, make sure your kernel has the `CONFIG_IWLWIFI_P2P` option turned on, as that enables the experimental P2P support for the driver.
Check with:

```
# zcat /proc/config.gz |grep P2P

CONFIG_IWLWIFI_P2P=y
```

The other thing you need from the kernel side is the `ozwpan` driver, built as a module. Just verify that the module exists, but don't load it yet, as you need to bring up the Wi-Fi direct interface before:

```
# modinfo  ozwpan

filename:       /lib/modules/[...]/ozwpan/ozwpan.ko

license:        GPL

version:        1.0.13

description:    Ozmo Devices USB over WiFi hcd driver

author:         Chris Kelly

parm:           g_net_dev:charp
```

Userspace
Once your kernel is good to go, you need a version of `wpa_supplicant` with P2P and WPS support. Again, this is quite recent so check if it's already there with:

```
$ wpa_cli help p2p

Selected interface 'wlan0'

commands:

  p2p_find [timeout] [type=*] = find P2P Devices for up-to timeout seconds

  p2p_stop_find = stop P2P Devices search

  p2p_connect <addr> <"pbc"|PIN> [ht40] = connect to a P2P Device
```

```
  p2p_listen [timeout] = listen for P2P Devices for up-to timeout seconds

[...]
```

If `wpa_cli` does not list any supported commands, you want to get a new wpa_supplicant version, so just grab the latest one from `git://w1.fi/srv/git/hostap.git` and build it using `defconfig` as a base. Make sure to turn on these options beforehand:

```
CONFIG_WPS=y

CONFIG_WPS2=y

CONFIG_READLINE=y

CONFIG_AP=y

CONFIG_P2P=y
```

Once you are done, reconnect your main connection with the new `wpa_supplicant` using the `nl80211` driver and cross your fingers. Still online? Go ahead then!

Pairing

To connect the Wi-Fi Direct device without disrupting the existing connection, we want to create a new interface in GO mode:

```
# wpa_cli -i wlan0



Interactive mode



> p2p_group_add

OK

<3>P2P-GROUP-STARTED p2p-wlan0-0 GO ssid="DIRECT-3G" freq=2437

   passphrase="sDv0YD4F" go_dev_addr=8c:70:5a:xx:xx:xx

> quit

# iw dev

phy#0

        Interface p2p-wlan0-0
```

```
                ifindex 19
                type P2P-GO
                channel 6 (2437 MHz) NO HT
        Interface wlan0
                ifindex 4
                type managed
                channel 6 (2437 MHz) NO HT
```

Finally, start `wpa_cli` on the new interface, run the `wps_pbc` (for Push Button Client) command, and press the "connect" button on the Wi-Fi Mouse. It should eventually go like this:

```
# wpa_cli -i p2p-wlan0-0


Interactive mode


> wps_pbc
OK [press the button on the mouse for ~1s now]
<3>CTRL-EVENT-EAP-STARTED 38:0d:d4:xx:xx:xx
<3>CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
<3>CTRL-EVENT-EAP-STARTED 38:0d:d4:xx:xx:xx
<3>CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
<3>CTRL-EVENT-EAP-STARTED 38:0d:d4:xx:xx:xx
<3>CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
<3>CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=254
<3>WPS-REG-SUCCESS 38:0d:d4:xx:xx:xx f53e5a2c-604c-483f-9256-380dd401eecb
<3>WPS-SUCCESS
<3>CTRL-EVENT-EAP-FAILURE 38:0d:d4:xx:xx:xx
```

```
<3>AP-STA-CONNECTED 38:0d:d4:xx:xx:xx

<3>AP-STA-DISCONNECTED 38:0d:d4:xx:xx:xx

<3>AP-STA-CONNECTED 38:0d:d4:xx:xx:xx

<3>AP-STA-DISCONNECTED 38:0d:d4:xx:xx:xx

<3>AP-STA-CONNECTED 38:0d:d4:xx:xx:xx

<3>AP-STA-DISCONNECTED 38:0d:d4:xx:xx:xx

[...]
```

The mouse is now continuously reconnecting to our host searching for its USB-over-Ethernet driver. To make it happy just load the `ozwpan` module, specifying the name of the new interface:

```
# modprobe ozwpan g_net_dev=p2p-wlan0-0

# dmesg

[...]

[23398.236275] ozwpan: module is from the staging directory, the quality is unknown,
you have been warned.

[23398.237235] ozwpan ozwpan: Ozmo Devices WPAN

[23398.237494] ozwpan ozwpan: new USB bus registered, assigned bus number 5

[23398.237537] usb usb5: New USB device found, idVendor=1d6b, idProduct=0001

[23398.237544] usb usb5: New USB device strings: Mfr=3, Product=2, SerialNumber=1

[23398.237550] usb usb5: Product: Ozmo Devices WPAN

[23398.237556] usb usb5: Manufacturer: Linux 3.8.0-rc3-00668-g55706d3 Ozmo WPAN

[23398.237561] usb usb5: SerialNumber: ozwpan

[23398.237725] hub 5-0:1.0: USB hub found

[23398.237731] hub 5-0:1.0: 8 ports detected

[23398.569110] usb 5-1: new full-speed USB device number 2 using ozwpan

[23398.779642] usb 5-1: New USB device found, idVendor=03f0, idProduct=a907
```

```
[23398.779650] usb 5-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3

[23398.779654] usb 5-1: Product: Wi-Fi

[23398.779657] usb 5-1: Manufacturer: HP

[23398.779660] usb 5-1: SerialNumber: 00001

[23398.881283] input: HP Wi-Fi  as /devices/platform/ozwpan/usb5/5-1/5-
1:1.0/input/input18

[23398.881802] hid-generic 0003:03F0:A907.0006: input,hidraw0: USB HID v1.10 Mouse [HP
Wi-Fi ] on usb-ozwpan-1/input0
```

That’s it! The mouse is connected and registered with the standard HID driver, just like a standard wired USB device. Congratulations, you’ve made it!
You can also check the existing connection with `wpa_cli`:

```
$ wpa_cli -i p2p-wlan0-0 all_sta

38:0d:d4:xx:xx:xx

dot11RSNAStatsSTAAddress=38:0d:d4:xx:xx:xx

dot11RSNAStatsVersion=1

dot11RSNAStatsSelectedPairwiseCipher=00-1e-ca-8

dot11RSNAStatsTKIPLocalMICFailures=0

dot11RSNAStatsTKIPRemoteMICFailures=0

hostapdWPAPTKState=11

hostapdWPAPTKGroupState=0

wpsPrimaryDeviceType=2-0050F204-0

wpsDeviceName=PANdev

wpsManufacturer=Ozmo

wpsModelName=HP Wi-Fi Mobile

wpsModelNumber=OZ2000

connected_time=551
```

That’s more or less how the basic thing works. The process is still complex for the casual user but all the pieces are in place.

Source : http://fabiobaltieri.com/2013/02/14/hp-wi-fi-direct-mouse-on-linux/