Wifi Direct Raspberry Pi:

What you need to use Wifi Direct on the Raspberry Pi:

First of all you need the raspberry pi with at least 2 GB SD Card and a Power supply. The power supply should deliver a minimal power of 1000 mAh, otherwise the USB and the Pi in general would shut down because of power loss. The Prototype system is a Debian Linux (Raspbian). The installation process will be described later. For Wifi direct usage you need a compatible Wifi USB Stick.

Setup the System:

The official Raspberry Pi Website offers many available Linux versions, which work with the raspberry pi. For this prototype we chose Raspbian Linux. It is possible to load Linux versions with the so called NOOBS image, which will be installed on the SD Card and inserted into the raspberry pi. NOOBS guides the User through the installation process and from there you can choose your preferred Linux Version. For this project I used the ready-to-install Raspbian image. The image will be loaded onto the SD Card. All data which was saved onto the card will be erased in this process. An installation guide will be found on the Raspberry website. (<http://www.raspberrypi.org/documentation/installation/installing-images/>)

After installing the image onto the SD Card, it can be plugged into the raspberry pi. If the power supply will be plugged into the raspberry pi, the raspberry will be boot up. After some installation instructions, the raspberry is ready. You will need an external monitor and keyboard plugged into the raspberry pi for using it. Another possibility is ssh. Raspbian will be installed with a ssh server running. You only need command line or putty (on windows machines) for connecting to the raspberry pi. The user name is “pi” and the password is “raspberry”. You will need an Ethernet cable to connect to the pi. Furthermore you have to identify the raspberry pi’s ip-address on the network for connection.

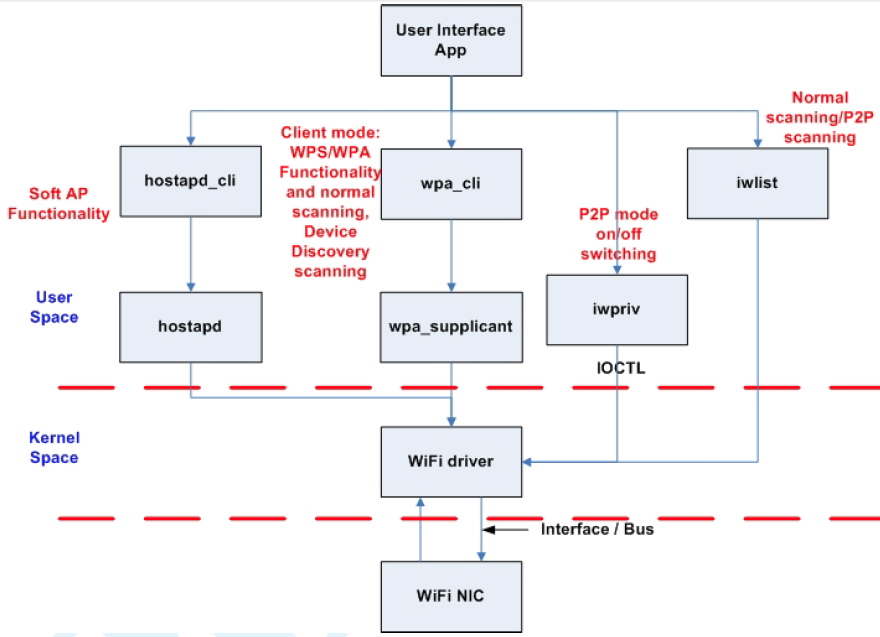
Install Wifi Stick driver:

If you have luck and the driver for the Wifi Stick is preinstalled and supported in the Kernel of Raspbian, than the stick will be recognized automatically. Otherwise you have to search for an arm build driver on the internet. You can either compile the driver for your kernel version yourself, or you can search on the web. However not every Wifi Stick is capable of using Wifi direct. In this project a TP-Link TL-WN725N Wifi stick is used. It is a so-called Nano USB Adapter. It needs not much space and is therefore perfect for our needs. The chipset is a Realtek 8188EU. Out of the box Wifi direct is not supported. You have to enable this feature on the chipset. This can be done with commands or a little command line program from Realtek called P2P\_UI. P2P\_UI is helpful, especially for visual people. An installation guide for P2P\_UI and the wlan adapter we used, can be found on youtube (http://www.youtube.com/watch?v=6GPv8TfZqe4).

Wifi direct establish connection:

(following is found in the realtek wifi direct programming guide)

Wifi direct was developed by the Wi-Fi Alliance. It should establish a peer to peer connection between clients. Following diagram shows the Linux Wi-Fi Direct Software architecture:

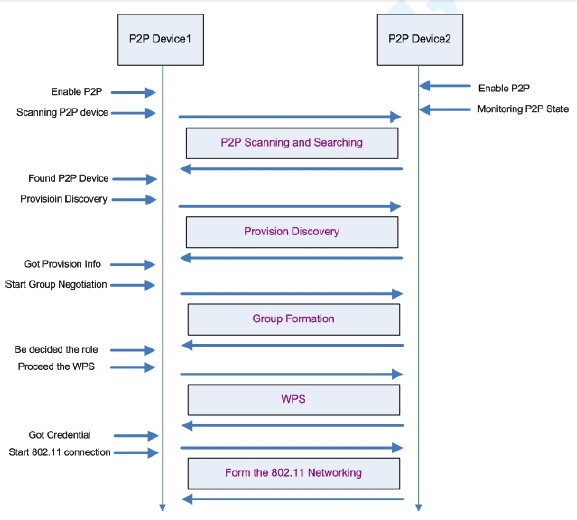


The iwpriv application is responsible for the communication to the Wi-Fi driver. With this application you can enable the p2p functionality of Wi-Fi direct.

If you want to establish a Wi-Fi direct connection, you need to go through 4 stages:

1. Device Discovery
2. Provision Discovery
3. Group Formation
4. Provisioning

In the following picture we see the workflow of a Wi-Fi direct connection process, including the 4 stages.



Let us go deeper into the different stages of this diagram:

# Enable P2P

In this stage both devices must be Wi-Fi direct compatible. With the iwpriv command, we can enable the Wi-Fi direct functionality on the Realtek driver. Following command is used:

iwpriv <interface name> p2p\_set enable=<mode>

The interface name defines your wlan interface name from your raspberry. In most cases it will be wlan0. Wi-Fi direct can be enabled in different modes:

* enable=0: Wi-Fi direct will be disabled.
* enable=1: Wi-Fi direct will be enabled in Device mode.
* enable=2: Wi-Fi direct will be enabled in Client mode.
* enable=3: Wi-Fi direct will be enabled in Group owner mode.

# Scanning Devices

After enabling Wi-Fi direct, the device has to search for other Wi-Fi compatible devices. This can be done with wpa\_supplicant/wpa\_cli or iwlist command. We used iwlist, because wpa\_supplicant/wpa\_cli did not work on the raspberry pi. The iwlist command will be used as following:

iwlist <interface name> scan

The command will list Wi-Fi direct enabled devices only, if you wish to scan for 802.11 network devices like access points, you have to disable Wi-Fi direct functionality on the driver.

# Provisioning Discovery

The Provisioning Discovery is used to get the WPS Pin Code from the other device. There are several configurations which can be used. The command looks like this:

iwpriv <interface name> p2p\_set prov\_disc=<peer device mac address>\_<mode>

The interface name is, as we mentioned before, wlan0. You also need the Mac address from the device, with which you want to establish a connection. The mac address is listed by doing a iwlist scan. The Provisioning Discovery offers 4 modes:

* display: The peer device will display its pin code, which the local device can use for pairing.
* keypad: This mode is the opposite of the display mode. The pin code will be displayed on the local device.
* pbc: In this mode, the devices are using the WPS push button procedure.
* label: In this mode the user should read the pin code from the label of the peer device and key-in this pin code of this label on the local device.

After the Provisioning Discovery a network connection between the two devices should be established.

To inform the driver from the successful connection process, the got\_wpsinfo command is used.

iwpriv <interface name> p2p\_set got\_wpsinfo=<mode>

Mode definition:

* 1: Pin code was sent from peer device by showing on display or label.
* 2: Pin code was sent from local device to the peer device.
* 3: WPS PBC was used.

On the peer side, the device can check with the status command, if the connection was established. Status=08 means that connecting was successful. After the successful status the req\_cm command should be used, to determine which WPS method was used. The result of this command will be returned as followed:

* CM=dis: Display
* CM=lab: Label
* CM=pbc: PBC
* CM=pad: Keypad

# Start group negotiation

In this stage will be determined which device should be group owner, a soft Access Point, and which should be the client which will connect to the access point.

Therefore the intent command is used:

Iwpriv <interface name> p2p\_set intent=<intent value>

The intent value goes from 0 to 15.  If one peer has intent 15, it forces to be the group owner. Is the intent from the other peer lower, it will be the group client and will connect to the group owner.

Before the negotiation process can be started, a ssid should be sent to the driver if the device will become group owner in the future.

 iwpriv <interface name> p2p\_set ssid=<ssid string>

After setting the ssid, the group negotiation command can be started.

iwpriv <interface name> p2p\_set nego=<peer mac address>

The peer device should now monitor the negotiation with the status and role command. These enums define the return values of the commands:

enum P2P\_STATE {

P2P\_STATE\_NONE = 0, // P2P disable

P2P\_STATE\_IDLE = 1, // P2P had enabled and do nothing

P2P\_STATE\_LISTEN = 2, // In pure listen state

P2P\_STATE\_SCAN = 3, // In scan phase

P2P\_STATE\_FIND\_PHASE\_LISTEN = 4, // In the listen state of find phase

P2P\_STATE\_FIND\_PHASE\_SEARCH = 5, // In the search state of find phase

P2P\_STATE\_TX\_PROVISION\_DIS\_REQ = 6,// In P2P provisioning discovery

P2P\_STATE\_RX\_PROVISION\_DIS\_RSP = 7,

P2P\_STATE\_RX\_PROVISION\_DIS\_REQ = 8,

P2P\_STATE\_GONEGO\_ING = 9, // Doing the group owner negoitation handshake

P2P\_STATE\_GONEGO\_OK = 10, // finish the group negoitation handshake with success

P2P\_STATE\_GONEGO\_FAIL = 11,// finish the group negoitation handshake with failure

P2P\_STATE\_RECV\_INVITE\_REQ = 12,// receiving the P2P Inviation request

P2P\_STATE\_PROVISIONING\_ING = 13, // Doing the P2P WPS

P2P\_STATE\_PROVISIONING\_DONE = 14, // Finish the P2P WPS

};

enum P2P\_ROLE {

P2P\_ROLE\_DISABLE = 0,

P2P\_ROLE\_DEVICE = 1,

P2P\_ROLE\_CLIENT = 2,

P2P\_ROLE\_GO = 3

};

# Proceed the WPS

Depending on the confirmed roles, the device with role command response Role=02 should launch the wpa\_supplicant in background and the wpa\_cli with pin code or PBC to perform the WPS procedure. The Device with role command response Role=3, should use the hostapd in background and the hostapd\_cli with pin code or PBC to perform the WPS procedure.

# DHCP

The device which is the group owner should also provide a DHCP-Server for IP Address exchange.