FIRST OF ALL (before explanation) execute SHELL in both sides (RPi3 and AsusZEN w/ SSD Crucial)

PC-Asus con RaspiOS: para arranque de USBIP-Server y del Freenove\_Client hay que ejecutar la Shell: **sh Str\_USB.sh**

RPi3 de Frenove-4WD: para arranque de USBip-Client y abrir el SERVER y ejecutar Mi-Aplicación (My\_DSW.py) existe la Shell: **sh Str\_DSW.sh**

Y luego en RPi3-Frenove4WD arrancar el SERVER: Client: **sh Str\_SER.sh**

Para finalmente, en PC-Asus y ya conectada la RPi3-Frenove4WD, la de USBIP-Client: **sh Str\_CLI.sh**

Como consecuencia del uso de un volante (o Joystick o manillar o Gamepad) de juegos de PC que se conectan al PC (o a la Raspberry)

(ver este interesantísimo link de donde salió como hacerlo con las RPi3 y Python

<https://diyprojects.io/python-library-evdev-raspberry-pi-use-gamepad-diy-projects-servomotor-games/>)

Se miro de hacerlo desde una conexión a un USB remoto de forma que la RPi y el Frenove-4WD no tengan que “depender” de esa conexión que lo hace inviable (a modo de control remoto, pero haciéndolo por Wi-Fi) se miró a ver como se puede compartir (unir o “bind”) los puertos PC de otro computar y por ende los dispositivos conectados a esto y… et voilà (or… and there you go)

**Proyecto USB/IP (para Linux) de “USB-over-IP server” (quien comparte su dispositivo) y client (quien lo usa como si estuviera conectado a uno de sus USBs)**

Viendo muchos de ellos, destacar los que parecieron más adecuados para la aplicación sobre la Raspberry Pi:

<https://www.flexihub.com/usb-over-ip-raspberry-pi/>

<https://www.youtube.com/watch?v=gBCNLs_5pwM> (este video está bien)

<https://jon.sprig.gs/blog/post/657>

El que más se parecía: <https://www.raspberrypi.org/forums/viewtopic.php?p=121691>

De todos ellos (y de mucha prueba/error) salió todo lo siguiente:

**HOST: PC ASUS running RaspiOS con IP: 192.168.1.136**

(Se señala en ROJO si se quiere además cargar también al CLIENT)

INSTALL (# Install usbip ..//..):

Se instaló “usbip”:

sudo apt-get install usbip

Luego se hizo (..//.. and setup the kernel module ..//..)

sudo modprobe usbip\_host #Load kernel module for SERVER

sudo modprobe vhci-hcd #Load kernel module for CLIENT

Para ponerlo en fichero /etc/modules (..//.. to load at startup.)

echo 'usbip\_host' >> /etc/modules

echo 'vhci-hcd' >> /etc/modules

Ver con

cat /etc/modules

Para ver los dispositivos conectados

lsusb

Para ver los ID de cada dispositivo conectado

usbip list -l

**CLIENT: RasPi con IP: 192.168.1.141**

(Se señala en ROJO si se quiere además cargar también al SERVER)

INSTALL:

Se instaló “usbip”:

sudo apt-get install usbip

Luego se hizo

sudo modprobe usbip\_host

sudo modprobe vhci-hcd

Para ponerlo en fichero /etc/modules

echo 'usbip\_host' >> /etc/modules

echo 'vhci-hcd' >> /etc/modules

Ver con

cat /etc/modules

Para ver los dispositivos conectados

lsusb

Para ver los ID de cada dispositivo conectado

usbip list -l

usbip list – remote **192.169.1.136**

lsusb

Se hizo un “reset” de ambos dispositivos pues no funcionaba, pero seguramente se debió a que se hicieron muchas pruebas para llegar a entender de aquí y de allí como se hacía correctamente pues ninguno lo hizo como yo quería.

**Arranque y ejecución:**

En Host:

lsusb #ver todos los USB disponibles

usbip list -l #ver los “busid” de estos

**sudo usbip bind -b 1-2** #(o: **sudo usbip bind –-busid 1-2)**

**sudo usbipd**

Con este último comando (que arranca el “Daemon”) se queda a la espera de que se conecte para ir mostrando que acontece.

En Client:

lsusb # se verá que no aparece el USB del SERVER todavía

**sudo usbip attach -r 192.168.1.136 -b 1-2**

(o más largo:

sudo usbip attach --remote 192.168.1.136 --busid 1-2 )

usbip list -r **192.168.1.136**

(o más largo:

usbip list --remote 192.168.1.136 )

lsusb # se verá que YA aparece el USB del SERVER.

Con este último ya se ve al “remote” estando disponible como si fuera uno más.

NOTAS:

En otro link (<https://askubuntu.com/questions/1077323/setup-usbip-on-ubuntu-18-04-server>) vi 2 comandos interesantes:

1. Para el cliente: usbip --debug list -r 192.168.3.182 [I've seen this kind of error when the Pi server simply isn't responding and needs to be rebooted. I've also seen it in laptops where usbip broke suspend, and the laptop needs to be rebooted. The --debug output usually helps to narrow the cause of the error.]
2. También para el Cliente, donde hace 2 “modprobe” y no solo uno (que es el segundo):

modprobe usbip-core

modprobe vhci-hcd

**usbip SERVICE para Servidor y Cliente empleando “systemd”**:

Para establecer como SERVICEs a ambos, Server & Client, hay que seguir este “guión”

<https://community.home-assistant.io/t/rpi-as-z-wave-zigbee-over-ip-server-for-hass/23006>

(que corrobora lo que se apuntaba también en el que fue ya señalado anteriormente

<https://www.flexihub.com/usb-over-ip-raspberry-pi/> )

Y si se quiere también certificar con preguntas en un FORUM sobre el tema, ver

[**https://unix.stackexchange.com/questions/528769/usbip-startup-with-systemd**](https://unix.stackexchange.com/questions/528769/usbip-startup-with-systemd)

NOTA:

Por el tema de abrir el **“Cliente” en el PC-RaspiOS** y conectarlo con la Raspberry Pi:

Al directori **. \Freenove\_4WD\_Smart\_Car\_Kit\_for\_Raspberry\_Pi-master\Code\Client** es te que obrir amb : **sudo nano Client\_Ui.py**

Canviar la “IP Address” por “192.168.1.144” en **line 558**

~~Line 558 self.IP.SetText(\_translate(“Client”, “IP Address”))~~

Line 558 self.IP.SetText(\_translate(“192.168.1.143”))