Basim Ali Yoshiguchi Follow

Jan 18, 2021 · 5 min read · Listen

Save

Edimax 7822 wifi USB dongle on Raspberry Pi 4

It may be the case that you want more throughput for your homemade NAS or your sweet setup to play your PC games on your TV with the Raspberry. All of these things, in many cases, require a fast and stable wifi connection (unless you go for ethernet). I personally needed one because my Raspberry was on the edge of my network. It was performing inadequately, and I wanted to stream my PC games to my tv using Parsec.

So I bought the cheap but powerful “Edimax 7822” USB wifi dongle. I didn’t think twice. I just bought it, plugged it in, and expected the drivers to exist in Linux, just like my old trusty TP-link TL-WN722N (that has the tried and tested Atheros chip). Turns out that’s not the case. I had to do some digging and working things out to get it to run.

Dude! Enough rambling, get to it already!

Ok, sorry, I guess social isolation has affected me more than I thought. I'll stop oversharing now… Let's get to it!

Ok, so what we need to do to get it working is grab a modified version of a driver with the same chipset but another brand and model of a dongle. Cilinx on GitHub has been so kind as to modify this for us. So assuming you are using RaspiOS with a 32bit kernel, here is what to do.

So firstly, **remove the USB dongle from the RPI4, this is important or else it could make the installation fail!**

Go ahead and open a terminal and update, upgrade and install prerequisites.

```
# Update all packages and upgrade
sudo apt update
sudo apt upgrade

# Install prerequisites
sudo apt install git dnsmasq hostapd bc build-essential dkms
raspberrypi-kernel-headers

# Reboot (not strictly necessary but still recommended)
sudo reboot

# Update and Upgrade your system
sudo apt update
sudo apt upgrade

# Install prerequisites
sudo apt install git dnsmasq hostapd bc build-essential dkms
raspberrypi-kernel-headers

# Reboot to refresh the system (not necessary but recommende
sudo reboot
```

Clone the driver repo with these commands. I cloned it inside a folder called

```
git clone https://github.com/cilynx/rtl88x2bu && cd rtl88x2bu/
```

Let's compile and install the kernel module (driver). This enables support for our Wi-Fi dongle. Since this driver is outside the default kernel source tree, we will use the Dynamic Kernel Module Support (DKMS) CLI tool to configure, build, and install the kernel module. This is a framework that will recompile the driver automatically for you in case you upgrade the kernel. There will be no need to go back and do it manually in the future.

Copy-paste the following commands inside the rtl88x2bu folder that you should already be standing in. We start preparing for the compilation.

First, we need to tell the Makefile that this is a Raspberry pi and that it has an ARM architecture.

```
sed -i 's/ARM_RPI = n/ARM_RPI = y/' Makefile
```

We grab the package version from the dkms.conf and assign it to a variable called "VER"

```
VER=$(sed -n 's/\PACKAGE_VERSION="\(.*\)"/\1/p' dkms.conf)
```

We then copy all the necessary files to a folder in /usr/src (where your kernel source files reside).

```
sudo rsync -rvhP ./ /usr/src/rtl88x2bu-${VER}
```

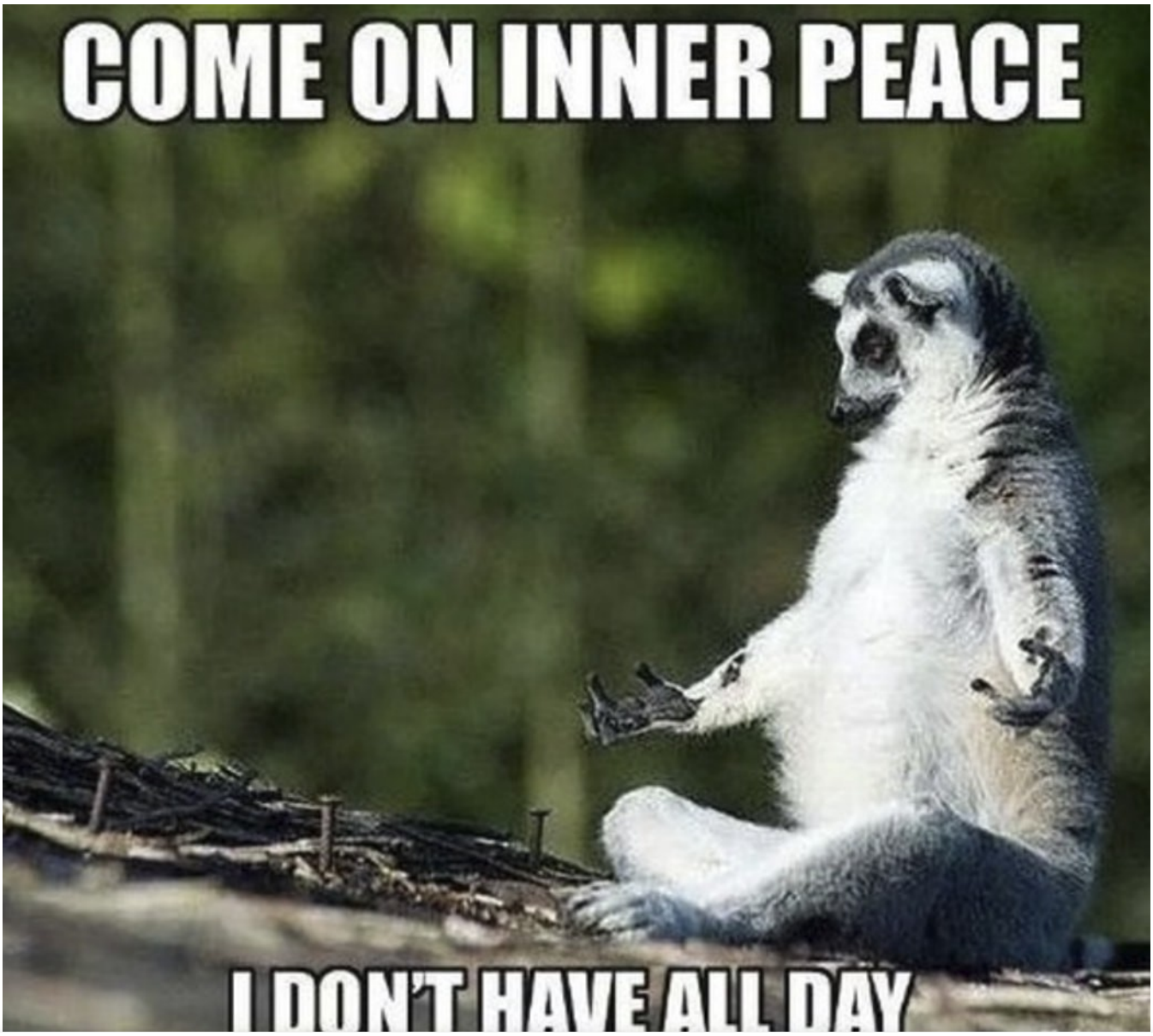
We then tell dkms that "Listen, dude, this module is of importance to me, please use this and recompile it when necessary" In other words… We add it to the system as a kernel module.


Let's build the binaries

```
sudo dkms build -m rtl88x2bu -v ${VER}
```

Please note that the module cannot be built without the prerequisites. So if you run into any problems in this step, please make sure that you have followed the previous steps correctly.

If you haven't run into any problems and the module is building, great! This will take a couple of minutes. It's a perfect time for some contemplation and zoning out a bit.



```
❯ sudo dkms add -m rtl88x2bu -v ${VER}

Creating symlink /var/lib/dkms/rtl88x2bu/5.6.1/source -
                 /usr/src/rtl88x2bu-5.6.1

DKMS: add completed.
❯ sudo dkms build -m rtl88x2bu -v ${VER}

Kernel preparation unnecessary for this kernel.  Skipp

Building module:
cleaning build area...
make -j4 KERNELRELEASE=5.4.79-v7l+ KVER=5.4.79-v7l+ src
.....................................
cleaning build area...

DKMS: build completed.
~/git_repos/rtl88x2bu 5.6.1_30362.…0180928-6a6a !1
❯
```

The last step will be to install the module to the kernel

```
sudo dkms install -m rtl88x2bu -v ${VER}
```

The result of this, if it's successful, should look like this.

```
❯ sudo dkms install -m rtl88x2bu -v ${VER}

88x2bu.ko:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/5.4.79-v7l+/kernel/drivers/net/

depmod...

DKMS: install completed.
~/git_repos/rtl88x2bu 5.6.1_30362.…0180928-6a6a !1          4s
❯
```

Now you can plug in your dongle and type this command.

```
ip a
```

This will show you your IP address. You can also see you have a new network interface, “wlan1”. That should be your Edimax dongle. Congratulations! You can now use your wifi dongle!

Optional

I like to set priorities for my interfaces, so if I, for example, have several connections to my router (wlan0, wlan1, and eth0 in my case). I want to route my traffic through the interface I know has the highest throughput. I need the ethernet to be used if it’s plugged in, then the Edimax dongle (wlan1), and lastly, the built-in wifi (wlan0) as the last fallback.

If you type:

```
sudo route -n
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    300    0        0 wlan1
0.0.0.0         192.168.1.1     0.0.0.0         UG    303    0        0 wlan0
169.254.0.0     0.0.0.0         255.255.0.0     U     202    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     300    0        0 wlan1
192.168.1.0     0.0.0.0         255.255.255.0   U     303    0        0 wlan0
```

The metrics here specify your priority. The lower the number the higher the priority. f you are using dhcpcd, which is used by default in the latest RaspiOS at the time of writing, you modify the metrics by editing /etc/dhcpcd.conf

```
sudo nano /etc/dhcpcd.conf
```

Add these lines to the bottom of that file then save and close.

```
interface wlan1
metric 300

interface wlan0
metric 303
```

Metrics for wired interfaces are usually in the 200 range and 300 for wireless interfaces. Mine was 202 by default, so I did not modify it. I simply just choose two numbers in the 300-range for wlan1 and wlan0. Wlan1 got the lower number since I want that to have a higher priority than the built-in wifi.

Thanks for reading! I hope this helped you out!