

802.11 Monitor Mode on Windows

Putting a MediaTek **MT7612U** USB adapter (Alfa AWUS036ACM, or the generic MT7612U dongle) into monitor mode & packet injection — the clean, reliable way.

What you end up with

A small headless Linux VM on your Windows box that grabs the physical Wi-Fi adapter over **native USB passthrough**. Plug the adapter in, run one command, and you have a monitor-mode radio that both **captures** and **injects** — the executor radio for SignalLab's wireless tier.

wlan0mon · type monitor

injection working

auto-attaches on plug-in

no usbip · no WSL

Why a VM and not WSL? Windows itself has no 802.11 monitor mode — that is a Linux capability. WSL2 *can* run the driver, but getting a USB Wi-Fi adapter into WSL needs `usbip`, and `usbip` can't carry these adapters (the chip's USB control transfers time out). A VM takes the physical USB device **natively**, so the firmware loads and monitor mode comes up cleanly. That is the whole reason for this approach.

0. Before you start

- Windows 10/11 with hardware virtualization enabled (it is, by default, on modern machines).
- The MT7612U adapter (VID `0e8d` / PID `7612`).
- ~6 GB free disk and about 30 minutes (mostly an unattended Ubuntu install that runs itself).
- An **administrator** PowerShell for the VirtualBox install step.

A. The fast path **AUTOMATED**

One script does everything in Part B for you — installs VirtualBox + the USB Extension Pack, downloads the Ubuntu ISO, builds and installs the VM, adds the Wi-Fi tooling, and wires up USB passthrough + SSH.

1. Put `setup-rf-executor.ps1` in a folder (it ships next to this guide).
2. Right-click PowerShell → **Run as administrator**, then run:

```
powershell -ExecutionPolicy Bypass -File .\setup-rf-executor.ps1
```

3. Let it finish (the unattended Ubuntu install is the long part). It prints the login and the exact command to bring up monitor mode when it's done.

Re-running is safe. Every step checks whether it's already done and skips it, so you can run the script again any time without harm. If you'd rather understand each piece, Part B is the same steps by hand.

B. The manual path STEP BY STEP

B1 • Install VirtualBox + the Extension Pack

Install VirtualBox 7.2.x. Then install the **Extension Pack** — that is what enables USB passthrough (without it the VM can't see the adapter). Accept the license when prompted.

B2 • Get the Ubuntu Server ISO

Download `ubuntu-24.04.x-live-server-amd64.iso`. Server (not Desktop) keeps the VM small — it already ships the `mt76x2u` driver and MT7662 firmware.

B3 • Create the VM

From a normal PowerShell (using VirtualBox's `VBoxManage`):

```
$vbm = "$env:ProgramFiles\Oracle\VirtualBox\VBoxManage.exe"
& $vbm createvm --name asl-rf-executor --ostype Ubuntu_64 --register
& $vbm modifyvm asl-rf-executor --memory 2048 --cpus 2 --usbxhci on --nic1 nat
# DNS fix so apt works inside the VM on Hyper-V-backed hosts:
& $vbm modifyvm asl-rf-executor --natdnshostresolver1 on --natdnsproxy1 on
& $vbm createmedium disk --filename "$env:USERPROFILE\VirtualBox VMs\asl-rf-executor\disk.vdi"
& $vbm storagectl asl-rf-executor --name SATA --add sata --controller IntelAhci
& $vbm storageattach asl-rf-executor --storagectl SATA --port 0 --device 0 --type hdd `
    --medium "$env:USERPROFILE\VirtualBox VMs\asl-rf-executor\disk.vdi"
```

B4 • Install Ubuntu (unattended)

Let VirtualBox drive the whole install — no clicking through the installer. The hostname must be a dotted FQDN:

```
& $vbm unattended install asl-rf-executor `
  --iso="$env:USERPROFILE\Downloads\ubuntu-24.04.4-live-server-amd64.iso" `
  --user=asl --password=aslr123 --full-user-name="ASL RF" `
  --hostname="aslr1f.localdomain" --install-additions `
  --post-install-command="apt-get update && apt-get install -y aircrack-ng iw" `
  --start-vm=headless
```

This installs Ubuntu, creates the `asl` user, and installs the Wi-Fi tools — then powers off.

B5 · Wire up USB passthrough + SSH

A USB **filter** makes the adapter attach to the VM automatically whenever it's plugged in. Forward a host port so you can SSH in:

```
& $vbm usbfilter add 0 --target asl-rf-executor --name mt7612u --vendorid 0e8d --productid 761
& $vbm modifyvm asl-rf-executor --natpf1 "ssh,tcp,127.0.0.1,2222,,22"
```

C. Bring up monitor mode

1. Start the VM (headless — no window needed):

```
& $vbm startvm asl-rf-executor --type headless
```

2. **Plug in the adapter.** The USB filter hands it straight to the VM. (On the Windows side it will "disappear" — that's correct; VirtualBox has captured it for the VM.)
3. SSH in and start monitor mode:

```
ssh asl@localhost -p 2222      # password: aslr123
sudo airmon-ng start wlan0     # creates wlan0mon
```

4. Confirm it:

```
iw dev                        # wlan0mon ... type monitor
sudo aireplay-ng --test wlan0mon # "Injection is working!"
```

That's the finish line. `type monitor` plus "Injection is working!" means the radio is both capturing and transmitting — ready for capture, and for SignalLab's active tier.

Golden rule: start the VM first, then attach the adapter — and unplug it before any

reboot. A USB Wi-Fi adapter attached *during* boot can stall the VM's startup while the kernel enumerates it (especially a cheap/flaky dongle). Hot-plugging into an already-running VM is the safe path.

D. The Windows dashboard OPTIONAL

A small local dashboard lets you drive the adapter from a browser window on Windows — see whether it's attached and in monitor mode, start/stop monitor mode, and run live scans that list the access points and client devices the adapter hears. It connects to the VM over **SSH** behind the scenes (no VBox guest channel, so nothing to jam).

D1 · Requirements

- **Node.js** (nodejs.org) — runs the little bridge server.
- **PuTTY** — the dashboard reaches the VM through PuTTY's `plink.exe` . It auto-discovers the VM's SSH host key, so there is nothing to configure.

D2 · Run it

```
cd rf-monitor-guide
.\start-dashboard.ps1           # or:  node alfa-dashboard\alfa-bridge.js
# then open http://127.0.0.1:8790
```

Close the Node window to stop it.

D3 · Two capture engines (your choice)

The scan has an engine selector next to the Scan button:

- **tcpdump · reliable** (default) — a plain packet capture that hops channels and lists BSSID, SSID, channel, signal and encryption, plus client devices seen in probe requests. Rock-solid when run headless.
- **airodump · richer** — aircrack-ng's scanner: more detail (cipher/auth, client↔AP associations).
Caveat: it does not capture when driven headless (it can't hop channels without an interactive terminal), and on a flaky dongle it can wedge the adapter — so the dashboard shows an honest note if it returns nothing. Use it by hand from an SSH terminal for its richer output, ideally with a solid radio like the Alfa. **On the dashboard, tcpdump is the one to use.**

D4 · Using the interface

- **Status pills** (top) — VM reachable, adapter attached, and monitor-mode state. They're read from `sysfs`, never the driver, so a wedged adapter can't hang them or turn the dashboard red.
- **Buttons are state-aware** — *Start monitor mode* is active only when an adapter is attached and you're not already in monitor mode; *Stop* and *Scan* are active only while in monitor mode (you can't capture without it).
- **Radio strip** — once monitor mode is on, a labeled row appears showing the monitor interface, PHY, driver, and chipset.
- **Scan** — set the duration, pick the engine, click Scan. The tables fill with access points (signal, BSSID, SSID, channel, encryption) and client devices seen in probe requests.

E. Quick reference

VM name	<code>asl-rf-executor</code>
Login	user <code>asl</code> · password <code>aslr123</code>
SSH	<code>ssh asl@localhost -p 2222</code>
Dashboard	<code>.\start-dashboard.ps1</code> → <code>http://127.0.0.1:8790</code> (needs Node.js + PuTTY)
Start / stop VM	<code>VBoxManage startvm asl-rf-executor --type headless</code> · <code>VBoxManage controlvm asl-rf-executor acpipowerbutton</code>
Monitor on / off	<code>sudo airmon-ng start wlan0</code> · <code>sudo airmon-ng stop wlan0mon</code>
Live scan	<code>sudo airodump-ng wlan0mon</code>
Adapter ID	MediaTek MT7612U — VID <code>0e8d</code> / PID <code>7612</code>
Same VM, the Alfa	The Alfa AWUS036ACM is the same chipset — plug it into this VM and it works identically.

ASL · SignalLab — RF Executor setup guide. Native USB passthrough via VirtualBox; no usbip, no WSL kernel required. The adapter runs entirely inside the Linux VM; Windows only hosts it.