ARCH LINUX + KDE PLASMA

Laptop Optimization Guide

GPU drivers · Special keys · Battery · Touchpad · Touch screens
Energy management · Fingerprint readers · Sleep & Hibernation

HP · Apple MacBook M4 Pro · Lenovo · HP EliteBook 840 G8

OVERVIEW

Optimization Topics Covered

GPU Drivers

NVIDIA, AMD, Intel — correct driver stack for each model

Power Management

systemd targets, kernel params, CPU governors

Special Keys & Fn

ACPI, WMI modules, fn-lock, media keys, backlight

Fingerprint Readers

fprintd, libfprint, PAM integration

Battery Management

TLP, auto-cpufreq, charge thresholds per vendor

Sleep & Hibernate

suspend-to-RAM, swap hibernation, hybrid sleep

Touchpad & Touch

libinput config, multi-finger gestures, Wayland/X11

Display & Backlight

DDC/CI, ambient light, HiDPI scaling in Plasma

01

HP ProBook 465 G11

AMD Ryzen 7 · Radeon 780M iGPU · USB-C · Fingerprint

GPU Drivers & Special Keys

AMD GPU Drivers (Radeon 780M)

- Install mesa + vulkan-radeon + libva-mesa-driver
- Firmware: linux-firmware (includes AMD firmware)
- ROCm optional for compute: rocm-opencl-runtime
- Check: glxinfo | grep 'OpenGL renderer'

```
sudo pacman -S mesa vulkan-radeon libva-mesa-driver
sudo pacman -S xf86-video-amdgpu  # X11 only
# Wayland (KDE) uses mesa natively – no xf86 needed
```

Special Keys & Fn Buttons

- Load hid_generic + hp_wmi modules (usually autoloaded)
- Fn-lock: hold Fn+Esc — toggle in BIOS/UEFI
- Brightness: kernel handles via acpi_backlight=vendor
- Add to /etc/default/grub: GRUB_CMDLINE_LINUX="acpi_backlight=vendor"
- Microphone mute LED: may need hp-wmi-sensors AUR pkg

```
# /etc/modprobe.d/hp.conf
options hp_wmi force_tablet_mode=0
# Then: sudo grub-mkconfig -o /boot/grub/grub.cfg
```

Battery Management & Touchpad

Battery & Power Management

- Install TLP: sudo pacman -S tlp tlp-rdw
- Enable: sudo systemctl enable --now tlp
- HP charge thresholds via TLP (ACPI-based):
- Set START_CHARGE_THRESH_BAT0=20
- Set STOP_CHARGE_THRESH_BAT0=80
- Also install: auto-cpufreq (AUR) for CPU scaling
- powertop --auto-tune for extra savings

```
# /etc/tlp.conf
START_CHARGE_THRESH_BAT0=20
STOP_CHARGE_THRESH_BAT0=80
CPU_SCALING_GOVERNOR_ON_BAT=powersave
```

Touchpad (Synaptics / libinput)

- Arch uses libinput by default — no extra driver needed
- Enable natural scroll & tap-to-click in KDE Settings
- Touchpad: Settings → Input Devices → Touchpad
- Fine-tune: /etc/X11/xorg.conf.d/40-libinput.conf
- Wayland: settings apply via KWin directly
- Gestures: install libinput-gestures (AUR) for 3/4-finger

```
# /etc/X11/xorg.conf.d/40-libinput.conf
Option "Tapping" "on"
Option "NaturalScrolling" "true"
```

Fingerprint Reader & Sleep / Hibernation

Fingerprint Reader (Goodix / Synaptics)

- Check device: lsusb | grep -i 'finger'
- Install: sudo pacman -S fprintd libfprint
- Enroll: fprintd-enroll
- PAM config: add pam_fprintd.so to /etc/pam.d/
- KDE SDDM login with fingerprint: edit /etc/pam.d/sddm
- Verify: fprintd-verify

```
sudo pacman -S fprintd libfprint
fprintd-enroll  # scan 5 times
# /etc/pam.d/system-local-login — add:
auth sufficient pam_fprintd.so
```

Sleep & Hibernation

- Suspend-to-RAM: works out of box (S3 state)
- Hibernate: needs swap partition ≥ RAM size
- Add resume= kernel param pointing to swap UUID
- Enable: systemctl enable systemd-hibernate.service
- Hybrid sleep (safest): systemctl hybrid-sleep
- Lid close action: /etc/systemd/logind.conf → HandleLidSwitch=suspend

```
# /etc/default/grub — add resume=UUID=<swap-uuid>
sudo grub-mkconfig -o /boot/grub/grub.cfg
# Test suspend:
systemctl suspend
```

ARCH LINUX INSTALLATION GUIDE

02

Apple MacBook M4 Pro

M4

PRO

Step-by-step guide using Asahi Linux — the only path to Arch on Apple Silicon

ARM64	Asahi Kernel	Wayland	~45 min
Architecture	Required	Display Server	Install Time

⚠ **Standard Arch ISO will NOT boot on M4. You MUST use the Asahi Linux installer first, then switch to Arch packages.**

OVERVIEW

What is Asahi Linux & Why It's Required

Why Standard Arch Doesn't Work

- Apple Silicon uses custom ARM64 SoC — not standard UEFI
- Requires Apple-specific bootloader (m1n1)
- Custom kernel patches for GPU, PCIe, USB, NVMe
- Proprietary firmware must be extracted from macOS
- Standard linux kernel lacks all Apple-specific drivers

What Asahi Linux Provides

- m1n1 bootloader — bridges Apple boot to Linux
- U-Boot + GRUB chain for standard Linux boot experience
- Patched kernel with Apple Silicon device tree
- Firmware extraction script (runs from macOS)
- Asahi Arch variant: pacman + Arch userland on Asahi kernel

INSTALLATION FLOW

1. macOS Prep
2. Asahi Installer
3. Base Arch Setup
4. KDE Plasma
5. Drivers & Optimize

✓ M4 Pro support status (2025): GPU ✓ USB ✓ NVMe ✓ Wi-Fi ✓ Bluetooth ✓ Audio ✓ Touch ID ✗ Deep Sleep ✗

2a

macOS Preparation

Resize partition, disable SIP, download firmware

Resize Partition & Disable SIP

⚠ **Do this from macOS. Back up your data with Time Machine first. These steps are irreversible without reinstalling macOS.**

1. Check Available Space

- Open Disk Utility (Applications → Utilities)
- You need at least 30 GB free for Arch partition
- Recommended: 60+ GB for comfortable use
- Check: Apple Menu → About This Mac → Storage

```
# In macOS Terminal — check disk layout:
diskutil list
# Note the disk0 identifier and free space
# Asahi needs contiguous free space at end of disk
```

2. Resize macOS Partition

- Open Terminal in macOS
- Run: diskutil list (note your disk identifier)
- Use Disk Utility → Resize macOS partition
- OR: System Settings → General → Storage → Manage
- Leave unallocated space — Asahi installer will use it

3. Disable SIP (if needed)

- Reboot into Recovery Mode: hold Power on startup
- Select Options → Continue → open Terminal
- Run: csrutil disable
- Reboot back to macOS
- Note: Asahi installer handles this automatically on M4

2b

Asahi Installer

Download, run, choose Arch Linux (ALARM) variant

Running the Asahi Linux Installer from macOS

macOS Terminal

```
# Run this in macOS Terminal — downloads and launches the Asahi installer:
curl https://alx.sh | sh
```

Installer Steps

- 1. Installer downloads automatically
- 2. Enter macOS admin password
- 3. Choose: Resize an existing partition
- 4. Set partition size (min 30 GB)
- 5. Select: Arch Linux ARM
- (NOT 'macOS' or 'Fedora Asahi')
- 6. Wait for firmware extraction
- 7. Follow reboot instructions carefully
- 8. You'll boot into a setup environment
- 9. Complete initial Arch setup wizard

What Installer Does

- Extracts Apple firmware from macOS
- Creates EFI + Linux partitions
- Installs m1n1 bootloader into Apple NVRAM
- Installs U-Boot + GRUB
- Deploys Arch Linux ARM base system
- Configures kernel device tree for M4
- Sets up pacman with Asahi repo

Partition Layout Created

- EFI System Partition (~500 MB)
- m1n1 stage2 partition
- Linux root partition (your chosen size)
- macOS remains on original partition
- Boot selection: hold Power → pick OS
- Default OS: settable in Startup Disk
- GRUB loads Linux kernel with Apple DT

STEP 2 · FIRST BOOT

Initial Boot Setup & Base System Configuration

After installer: hold Power button → select your Linux boot entry → system boots to a minimal Arch environment

Initial Login & Setup

- Default user: alarm / password: alarm
- Root password: root
- First: sudo pacman-key --init
- Then: sudo pacman-key --populate archlinuxarm
- Update system: sudo pacman -Syu
- This may take 10–20 min on first run

Set Up Basic System

- Set hostname: sudo hostnamectl set-hostname mymacbook
- Set timezone: timedatectl set-timezone Europe/Warsaw
- Enable NTP: timedatectl set-ntp true
- Set locale: edit /etc/locale.gen → uncomment en_US.UTF-8
- Run: locale-gen
- Set LANG: echo 'LANG=en_US.UTF-8' > /etc/locale.conf

```
# First commands after login:
sudo pacman-key --init
sudo pacman-key --populate archlinuxarm
sudo pacman -Syu
# Reboot after updates:
sudo reboot
```

```
sudo hostnamectl set-hostname macbook-arch
sudo timedatectl set-timezone Europe/Warsaw
sudo timedatectl set-ntp true
sudo sed -i 's/#en_US.UTF-8/en_US.UTF-8/' /etc/locale.gen
sudo locale-gen
```

2c

User & Network Setup

Create user, configure sudo, connect to Wi-Fi

Create Admin User & Connect to Wi-Fi

Create user

```
# Create your user account:
useradd -m -G wheel,audio,video,storage -s /bin/bash yourname
passwd yourname
# Enable sudo for wheel group:
EDITOR=nano visudo
# → Uncomment: %wheel ALL=(ALL:ALL) ALL
```

sudo setup

```
# Install sudo if not present:
sudo pacman -S sudo
# Verify wheel group membership:
groups yourname
# Switch to your user:
su - yourname
```

Wi-Fi Setup (iwd)

- Asahi uses iwd (iNet Wireless Daemon) by default
- Start: systemctl start iwd
- Enable on boot: systemctl enable iwd
- Launch client: iwctl
- device list → note your device name (e.g. wlan0)
- station wlan0 scan
- station wlan0 get-networks
- station wlan0 connect YourSSID
- Exit iwctl, verify: ping -c3 archlinux.org

Wi-Fi via iwd

```
# Full iwd session:
systemctl enable --now iwd
iwctl
  > device list
  > station wlan0 scan
  > station wlan0 get-networks
  > station wlan0 connect "MyNetwork"
  > (enter password)
  > quit
ping -c3 archlinux.org
```

2d

Asahi Repo & Packages

Configure pacman, install Asahi kernel & GPU driver

Configure pacman with Asahi Repository

The Asahi repo contains M4-specific kernel, GPU driver (mesa-asahi-edge), and firmware packages not available in standard Arch repos.

/etc/pacman.conf

```
# /etc/pacman.conf — add these sections at the bottom:
[asahi]
Server = https://cdn.asahilinux.org/$arch/$repo

[alarm]
Server = http://mirror.archlinuxarm.org/$arch/$repo

# Then update keyring:
sudo pacman -Sy archlinux-keyring
sudo pacman -Sy asahi-keyring
```

Core packages

```
# Install core Asahi packages:
sudo pacman -S linux-asahi linux-asahi-headers
sudo pacman -S mesa-asahi-edge
sudo pacman -S linux-firmware-asahi
sudo pacman -S uboot-asahi
# Rebuild initramfs:
sudo mkinitcpio -P
```

Essential Base Packages

- base base-devel — core Arch tools
- networkmanager — for GUI network management later
- git curl wget — fetch tools
- vim nano — text editors
- man-db man-pages — documentation
- htop neofetch — system monitoring

```
sudo pacman -S base-devel git curl wget
sudo pacman -S networkmanager
sudo pacman -S vim nano man-db man-pages
sudo systemctl enable NetworkManager
# Disable iwd if switching to NetworkManager:
sudo systemctl disable iwd
```

2e

KDE Plasma Installation

Display server, Plasma desktop, SDDM login manager

STEP 5 · KDE PLASMA

Install KDE Plasma 6 + SDDM on Wayland

Use Wayland exclusively on Apple Silicon — X11 has limited support. KDE Plasma 6 on Wayland is stable and fully recommended.

KDE Plasma

```
# Install KDE Plasma + SDDM:
sudo pacman -S plasma-meta
sudo pacman -S sddm
sudo systemctl enable sddm
# Optional full KDE apps:
sudo pacman -S kde-applications-meta
```

Minimal KDE

```
# Minimal KDE (lighter install):
sudo pacman -S plasma-desktop plasma-pa plasma-nm
sudo pacman -S konsole dolphin kate gwenview
sudo pacman -S kscreen powerdevil bluedevil
sudo pacman -S sddm sddm-kcm
sudo systemctl enable sddm
```

Wayland Configuration

- SDDM auto-selects Wayland on Asahi
- Force Wayland session in SDDM:
- Edit /etc/sddm.conf.d/wayland.conf
- [General]
- DisplayServer=wayland
- KDE session type: Plasma (Wayland)
- Verify: echo $XDG_SESSION_TYPE

GPU Driver for KDE

- mesa-asahi-edge already installed
- Set env var for Mesa Asahi driver:
- Add to /etc/environment:
- MESA_LOADER_DRIVER_OVERRIDE=asahi
- KDE hardware acceleration: auto-detected
- Check: kinfo → GPU shows Asahi Mesa

Audio Setup

- PipeWire: recommended (replaces PulseAudio)
- sudo pacman -S pipewire pipewire-pulse
- sudo pacman -S wireplumber
- systemctl --user enable pipewire
- systemctl --user enable wireplumber
- KDE: System Settings → Audio works after reboot

2f

AUR Helper & Fonts

Install yay, configure AUR, install fonts & nerd fonts

AUR Helper (yay) and Font Configuration

Install yay

```
# Install yay AUR helper (as your user, not root):
cd /tmp
git clone https://aur.archlinux.org/yay.git
cd yay
makepkg -si
# Verify:
yay --version
```

Fonts

```
# Install essential fonts:
sudo pacman -S noto-fonts noto-fonts-emoji
sudo pacman -S ttf-liberation ttf-dejavu
sudo pacman -S ttf-jetbrains-mono
# From AUR (nerd fonts):
yay -S ttf-meslo-nerd
yay -S ttf-hack-nerd
```

Useful AUR Packages for M4

- yay -S auto-cpufreq — smart CPU frequency scaling
- yay -S power-profiles-daemon — KDE battery integration
- yay -S macbook-key-fix — keyboard layout tweaks
- yay -S asahi-battery — battery charge limit control
- yay -S touchegg — touchpad gesture enhancement
- yay -S pamac-aur — graphical package manager
- yay -S downgrade — rollback package versions

Font Rendering Optimization

- Install fontconfig: sudo pacman -S fontconfig
- Enable subpixel rendering in /etc/fonts/local.conf
- KDE: System Settings → Fonts → Anti-aliasing: Enabled
- Sub-pixel rendering: RGB (most M4 displays)
- Hinting: Slight
- Force DPI: 96 (or 192 for native HiDPI)
- Apply: fc-cache -fv

2g

Power & Battery Setup

auto-cpufreq, charge limits, sleep configuration

Battery, Sleep & Energy Optimization on M4

Do NOT install TLP on Apple Silicon — it conflicts with Asahi power management. Use power-profiles-daemon + auto-cpufreq instead.

Power setup

```
# Power management stack for M4:
sudo pacman -S power-profiles-daemon
sudo systemctl enable --now power-profiles-daemon
yay -S auto-cpufreq
sudo systemctl enable --now auto-cpufreq
# Check profile:
powerprofilesctl
```

Battery limit

```
# Battery charge limit (requires asahi-battery):
yay -S asahi-battery
# Set 80% charge limit:
echo 80 | sudo tee /sys/class/power_supply/macsmc-
battery/charge_control_end_threshold
# Make permanent: create udev rule or systemd service
```

Sleep / Suspend Configuration

- S3 deep sleep: NOT available on Apple Silicon
- Available: s2idle (freeze, uses some power)
- Set default: add to /etc/default/grub kernel params:
- mem_sleep_default=s2idle
- Lid close: /etc/systemd/logind.conf:
- HandleLidSwitch=suspend
- Test: systemctl suspend
- Wake: any key press or lid open

auto-cpufreq Config

- Config file: /etc/auto-cpufreq.conf
- [battery] section:
- governor = powersave
- turbo = auto
- [charger] section:
- governor = performance
- turbo = auto
- Monitor: auto-cpufreq --stats

2h

Hardware Features

Wi-Fi, Bluetooth, keyboard, touchpad, webcam

Keyboard, Touchpad, Webcam & Bluetooth

Keyboard & Fn Keys

- Module: hid-apple (auto-loaded)
- Fn mode config: /etc/modprobe.d/hid_apple.conf
- options hid_apple fnmode=2
- (2 = media keys default, Fn+F1 = F1)
- (1 = F1-F12 default, Fn = media)
- Brightness keys: work via Asahi kernel
- Keyboard backlight: set in KDE → Input Devices
- Language/layout: localectl set-x11-keymap us

Touchpad

- Driver: apple-touchpad-ng (Asahi kernel built-in)
- libinput handles configuration
- Enable tap-to-click in KDE: System Settings
- Natural scroll: enabled by default on Asahi
- Multi-finger gestures: work on Wayland
- 3-finger swipe: virtual desktop switching
- Pinch to zoom: supported in most KDE apps
- Palm rejection: auto-configured
- Fine-tune: libinput list-devices

Webcam, BT & Speakers

- FaceTime HD camera: works (apple-isp driver)
- Check: v4l2-ctl --list-devices
- Bluetooth:
- sudo pacman -S bluez bluez-utils
- sudo systemctl enable --now bluetooth
- bluetoothctl → power on → scan on
- KDE: System Settings → Bluetooth
- Speakers / microphone:
- PipeWire handles Apple audio codec
- May need: asahi-audio (AUR) for speaker tuning
- yay -S asahi-audio

Post-Install Tweaks

HiDPI, Rosetta alternative, known issues & workarounds

HiDPI Scaling, Performance & Known Issues

HiDPI / Retina Display Setup

- M4 Pro screen is 3024×1964 — very high DPI
- KDE: System Settings → Display → Global Scale
- Recommended scale: 200% (logical 1512×982)
- Or 175% for more screen real estate
- Per-app Wayland scaling: works in KDE Plasma 6
- Set DPI in /etc/environment:
- QT_SCALE_FACTOR=2
- GDK_SCALE=2 (for GTK apps)

x86 / Rosetta Alternative

- No Rosetta on Linux — no x86 emulation by default
- Option: FEX-Emu (AUR) — x86/x86_64 emulator on ARM
- yay -S fex-emu-bin
- For most use cases — native ARM64 apps preferred
- Flatpak: many apps have native ARM64 builds
- sudo pacman -S flatpak
- flatpak remote-add --if-not-exists flathub ...
- Check: flatpak search <app> → look for aarch64

KNOWN LIMITATIONS ON M4

Feature	Status	Workaround
Touch ID	Not supported	Use password / PIN in KDE
Deep Sleep (S3)	Not available	s2idle works, more battery drain
Hibernate	Unstable / experimental	Avoid for daily use
Thunderbolt eGPU	Not supported	Internal GPU only
ProMotion 120Hz	120Hz may not work	60Hz is stable
Charge Threshold	Needs udev rule	asahi-battery AUR package

System Updates & Asahi-Specific Maintenance

Asahi is a rolling release — update frequently. Kernel updates may require bootloader update. Always check asahilinux.org/news before major updates.

System updates

```
# Full system update (Arch + Asahi repos):
sudo pacman -Syu
# Update AUR packages:
yay -Syu
# Check for Asahi-specific announcements:
# https://asahilinux.org/news
# After kernel update — reboot required!
```

Asahi-specific

```
# Bootloader update (after linux-asahi update):
sudo update-m1n1
# Check current kernel:
uname -r
# Should show: X.X.X-asahi or similar
# Check Asahi package versions:
pacman -Q | grep asahi
```

Useful System Commands

- pacman -Qs asahi — list all Asahi packages
- journalctl -b -p err — errors from last boot
- dmesg | grep -i apple — Apple driver messages
- lspci — list PCI devices (GPU, Wi-Fi, etc.)
- lsusb — USB devices
- inxi -Faz — full system info (sudo pacman -S inxi)
- sensors — CPU temperature (lm_sensors)

Rollback & Recovery

- Boot to macOS: hold Power → select macOS
- Rollback package: sudo downgrade <package>
- Arch wiki: wiki.archlinux.org
- Asahi wiki: github.com/AsahiLinux/docs/wiki
- Asahi Discord: chat with the community
- If Plasma broken: boot to TTY → Ctrl+Alt+F2
- Reinstall display: sudo pacman -S plasma-meta

SUMMARY

Complete Installation Checklist

1. macOS: Free up space (30 GB+ unallocated), back up data
2. macOS: Run curl https://alx.sh | sh in Terminal
3. Installer: Select 'Arch Linux ARM', set partition size, wait for firmware extraction
4. First boot: pacman-key --init, pacman-key --populate, pacman -Syu
5. Add Asahi repo to /etc/pacman.conf, install linux-asahi + mesa-asahi-edge
6. Create user with sudo, configure locale, timezone, hostname
7. Connect Wi-Fi via iwd or NetworkManager
8. Install KDE Plasma 6 + SDDM, enable sddm.service
9. Install PipeWire for audio, bluez for Bluetooth
10. Set HiDPI scale to 200% in KDE Display Settings
11. Install power-profiles-daemon + auto-cpufreq (NOT TLP)
12. Configure Fn keys: /etc/modprobe.d/hid_apple.conf
13. Install yay AUR helper, install asahi-battery + asahi-audio
14. Set mem_sleep_default=s2idle in GRUB, configure lid close action

ARCH LINUX ON APPLE M4 PRO

Installation Complete

asahilinux.org

Official Asahi Linux project

wiki.archlinux.org

Arch Linux Wiki

github.com/AsahiLinux

Asahi source & issue tracker

discord.gg/asahilinux

Asahi Linux community Discord

Keep the Asahi kernel updated regularly — M4 support is actively developed.

03

HP Victus 16-s0044nt

AMD Ryzen 5 7535HS · Radeon 660M + RTX 2050 (hybrid)

Hybrid GPU — AMD iGPU + NVIDIA dGPU

This laptop uses AMD+NVIDIA hybrid graphics (PRIME). Correct setup is critical — wrong config causes black screens or heavy battery drain.

AMD iGPU (Radeon 660M)

- Driver: mesa + vulkan-radeon (same as ProBook)
- Default GPU for display output and desktop
- Used for: browser, office, low-power tasks
- Wayland: works natively with KDE Plasma
- VA-API decode: libva-mesa-driver

NVIDIA RTX 2050 (dGPU)

- Install: nvidia nvidia-utils lib32-nvidia-utils
- DO NOT install xf86-video-nouveau — conflicts
- PRIME offload: run apps on dGPU on demand
- prime-run glxgears — launches on NVIDIA
- Or: set env DRI_PRIME=1 per app
- nvidia-settings for fan/clock control

```
sudo pacman -S mesa vulkan-radeon libva-mesa-driver
sudo pacman -S nvidia nvidia-utils lib32-nvidia-utils
# Reboot after nvidia install — mkinitcpio runs automatically
```

```
# Run any app on NVIDIA GPU:
prime-run %command%
# Or in Steam launch options:
DRI_PRIME=1 %command%
# Check active GPU: glxinfo | grep renderer
```

Battery, Touchpad & Sleep

Battery Management

- TLP: sudo pacman -S tlp tlp-rdw
- systemctl enable --now tlp
- CRITICAL: disable NVIDIA dGPU when idle
- Install: envycontrol (AUR)
- envycontrol --switch integrated
- Reboot — NVIDIA fully powered off
- TLP charge thresholds (HP ACPI):
- START=20, STOP=80 in /etc/tlp.conf

Touchpad

- Controller: ELAN / Synaptics via libinput
- No extra drivers needed on Arch
- Enable in KDE: System Settings → Input Devices
- Gestures (3-finger swipe for virtual desktops):
- Install libinput-gestures (AUR)
- libinput-gestures-setup start
- libinput-gestures-setup autostart
- Config: ~/.config/libinput-gestures.conf

Sleep & Hibernate

- S3 sleep: fully supported
- Check: cat /sys/power/mem_sleep → deep
- Hibernate: needs swap ≥ RAM
- Critical: NVIDIA must be suspended too
- Add to /etc/modprobe.d/nvidia-pm.conf:
- options nvidia NVreg_PreserveVideoMemoryAllocations=1
- systemctl enable nvidia-suspend.service
- systemctl enable nvidia-hibernate.service

04

HP EliteBook 840 G8

Intel Core i5/i7 11th Gen · Intel Iris Xe · MIL-SPEC · 5G option

Intel GPU & Special Keys

Intel Iris Xe GPU

- Driver: mesa + vulkan-intel (ANV driver)
- Install: intel-media-driver for VA-API
- DO NOT install xf86-video-intel (deprecated, buggy)
- KDE Wayland: iris driver is default in Mesa
- GuC/HuC firmware: load for better power mgmt
- Add to modprobe: options i915 enable_guc=3
- Verify: intel_gpu_top (package: intel-gpu-tools)

```
sudo pacman -S mesa vulkan-intel intel-media-driver
sudo pacman -S intel-gpu-tools
# /etc/modprobe.d/i915.conf:
options i915 enable_guc=3 enable_fbc=1
```

Special Keys & HP Sure Features

- HP WMI: load hp-wmi module
- Privacy Camera key: works via hp-wmi
- HP Sure View (privacy screen): toggle via Fn key
- Backlit keyboard: controlled by KDE/ACPI automatically
- Volume/mute OSD: works natively in KDE Plasma
- HP Command Center equivalent: tlp + thermald
- 5G modem (if equipped): ModemManager + nmcli

```
# HP WMI sensor support:
sudo pacman -S lm_sensors
sudo sensors-detect
# For 5G modem:
sudo pacman -S modemmanager && systemctl enable ModemManager
```

Battery, Fingerprint & Hibernate

Battery & Power (EliteBook)

- TLP: best choice — full ACPI support
- Intel-specific: enable_pcie_aspm=1 in TLP
- thermald: sudo pacman -S thermald
- systemctl enable --now thermald
- Charge thresholds via TLP (HP ACPI):
- START_CHARGE_THRESH_BAT0=20
- STOP_CHARGE_THRESH_BAT0=80
- powertop --calibrate (run once on AC)

Fingerprint Reader

- Sensor: Synaptics / Validity (common on G8)
- Install: fprintd + libfprint-tod-vfs0090 (AUR)
- Check: lsusb | grep -i '138a\|06cb'
- Some sensors need: python-validity (AUR)
- Enroll: fprintd-enroll -f right-index-finger
- PAM: edit /etc/pam.d/system-local-login
- KDE lock screen: add pam_fprintd to kde PAM config

```
# Synaptics fingerprint (common on G8):
yay -S python-validity open-fprintd-next
sudo systemctl enable --now open-fprintd
fprintd-enroll
```

```
# /etc/pam.d/system-local-login — add BEFORE password:
auth sufficient pam_fprintd.so
# Verify enrollment:
fprintd-verify -f right-index-finger
```

Sleep, Hibernate & Suspend Config

Hibernate Setup (Intel)

- 1. Create or verify swap partition ≥ RAM size
- OR create swapfile: fallocate -l 32G /swapfile
- 2. chmod 600 /swapfile && mkswap /swapfile
- 3. Add to /etc/fstab: /swapfile none swap sw 0 0
- 4. Get UUID: blkid /dev/sdXY (swap partition)
- OR for swapfile: findmnt -no UUID -T /swapfile
- 5. Get offset: filefrag -v /swapfile | head -4
- 6. Kernel params: resume=UUID=xxx resume_offset=yyy
- 7. sudo mkinitcpio -P && grub-mkconfig ...
- 8. Add 'resume' to HOOKS in /etc/mkinitcpio.conf
- 9. Test: systemctl hibernate

Suspend Configuration

- S3 (deep) sleep: verified working on G8
- Check modes: cat /sys/power/mem_sleep
- Set default: add mem_sleep_default=deep to grub
- Lid close: /etc/systemd/logind.conf
- HandleLidSwitch=suspend
- HandleLidSwitchExternalPower=suspend

Hybrid Sleep (Recommended)

- Safest option: saves RAM to disk AND suspends
- On power loss during sleep — no data loss
- Enable: systemctl enable systemd-hybrid-sleep
- Set in logind.conf: HandleLidSwitch=hybrid-sleep
- Or trigger: systemctl hybrid-sleep

05

General Configuration

Common optimizations applicable to all laptops

GPU Driver Matrix — All Brands

Laptop	GPU	Packages	Notes
HP ProBook 465 G11	AMD Radeon 780M	mesa, vulkan-radeon, libva-mesa-driver	Pure AMD — no xf86-video-amdgpu on Wayland
MacBook M4 Pro	Apple AGX G14	mesa-asahi-edge (Asahi repo)	Asahi kernel mandatory — X11 not supported
HP Victus 16-s0044nt	AMD 660M + RTX 2050	mesa + nvidia + envycontrol	Hybrid PRIME — use envycontrol for switching
HP EliteBook 840 G8	Intel Iris Xe	mesa, vulkan-intel, intel-media-driver	GuC/HuC via i915 modprobe — no xf86-video-intel
Lenovo ThinkPad (Intel)	Intel UHD/Iris	mesa, vulkan-intel, intel-media-driver	ThinkPad ACPI: thinkpad_acpi module
Dell XPS (NVIDIA)	Intel + NVIDIA	mesa + nvidia + optimus-manager	optimus-manager or prime-run for GPU switch
ASUS ROG/TUF	AMD/NVIDIA	asus-wmi-sensors, asusctl (AUR)	supergfxctl for GPU switching on ASUS

Universal Battery & Power Management

TLP Configuration

- Install: sudo pacman -S tlp tlp-rdw
- Enable: systemctl enable --now tlp
- Disable conflicting: power-profiles-daemon
- Config file: /etc/tlp.conf
- Charge thresholds (most HP/Lenovo/Dell):
- START_CHARGE_THRESH_BAT0=20
- STOP_CHARGE_THRESH_BAT0=80
- USB autosuspend: USB_AUTOSUSPEND=1
- Check status: tlp-stat -b

auto-cpufreq

- Smarter than TLP for CPU scaling
- Install from AUR: yay -S auto-cpufreq
- Enable: systemctl enable --now auto-cpufreq
- Works alongside TLP (disable TLP CPU section)
- Set in /etc/auto-cpufreq.conf:
- [battery] → governor = powersave
- [charger] → governor = performance
- Monitor: auto-cpufreq --stats

Power Profiles Daemon

- Alternative to TLP (simpler, KDE-integrated)
- sudo pacman -S power-profiles-daemon
- systemctl enable --now power-profiles-daemon
- KDE uses it automatically in taskbar
- Profiles: power-saver, balanced, performance
- CLI: powerprofilesctl set power-saver
- Note: do NOT run TLP + PPD simultaneously
- Apple Silicon: use PPD (TLP limited there)

Fingerprint Reader — Universal Setup

Installation & Enrollment

- sudo pacman -S fprintd libfprint
- List devices: fprintd-list
- Enroll finger: fprintd-enroll -f right-index-finger
- Scan 5x as prompted
- Verify: fprintd-verify
- For unsupported sensors: check AUR for libfprint-tod
- Validity/Synaptics on HP: python-validity (AUR)

PAM Integration (KDE/SDDM)

- Edit /etc/pam.d/system-local-login:
- auth sufficient pam_fprintd.so
- For SDDM (KDE login): /etc/pam.d/sddm
- For sudo: /etc/pam.d/sudo
- For KDE lock screen: /etc/pam.d/kde
- Place pam_fprintd BEFORE pam_unix for fallback
- Test: lock screen → touch sensor

```
sudo pacman -S fprintd libfprint
fprintd-enroll -f right-index-finger
fprintd-verify -f right-index-finger
# List enrolled:
fprintd-list "$USER"
```

```
# /etc/pam.d/system-local-login
auth     sufficient  pam_fprintd.so
auth     include     system-auth
# /etc/pam.d/sddm — add same line at top
```

Sleep & Hibernation — Universal Guide

Suspend (S3/S2Idle)

- Test: systemctl suspend
- Check modes: cat /sys/power/mem_sleep
- Set default (grub):
- mem_sleep_default=deep (S3)
- mem_sleep_default=s2idle (for AMD/Apple)
- Lid close: /etc/systemd/logind.conf
- HandleLidSwitch=suspend
- After edit: systemctl restart systemd-logind

Hibernate Setup

- 1. Swap must be ≥ RAM (partition or file)
- 2. Swapfile: fallocate -l 32G /swapfile
- chmod 600 /swapfile; mkswap /swapfile
- 3. /etc/fstab: /swapfile none swap sw 0 0
- 4. Get offset: filefrag -v /swapfile
- 5. /etc/default/grub add:
- resume=UUID=X resume_offset=Y
- 6. /etc/mkinitcpio.conf HOOKS: add 'resume'
- 7. sudo mkinitcpio -P
- 8. sudo grub-mkconfig -o /boot/grub/grub.cfg

Hybrid Sleep

- Best of both worlds: RAM + disk snapshot
- Test: systemctl hybrid-sleep
- Set default in logind.conf:
- HandleLidSwitch=hybrid-sleep
- NVIDIA hibernate services:
- nvidia-suspend.service
- nvidia-hibernate.service
- nvidia-resume.service
- Enable all 3 with systemctl enable
- NVreg_PreserveVideoMemoryAllocations=1

Touchpad & Touchscreen Configuration

libinput — All Touchpads

- Default on Arch — no extra packages
- KDE: System Settings → Input Devices → Touchpad
- Enable: Tap to click, Natural Scroll, Two-finger scroll
- Disable touchpad while typing (palmDetect)
- Wayland: settings go direct to KWin
- X11 config: /etc/X11/xorg.conf.d/40-libinput.conf
- Debug: libinput debug-events

```
# /etc/X11/xorg.conf.d/40-libinput.conf
Section "InputClass"
  Identifier "touchpad"
  Option "Tapping" "on"
  Option "NaturalScrolling" "true"
  Option "DisableWhileTyping" "true"
EndSection
```

Touchscreen (HP ProBook / EliteBook)

- Kernel: usbhid / hid_multitouch handles it automatically
- Wayland (KDE): touchscreen works out of box
- Calibration: xinput_calibrator (X11) or touchegg (Wayland)
- Gestures: Install touchegg (AUR) for tap/swipe/pinch
- systemctl enable --now touchegg
- KDE Plasma 6: native touchscreen gestures built in
- On-screen keyboard: maliit-keyboard (AUR)

```
# Touchscreen gestures (Wayland):
yay -S touchegg
systemctl enable --now touchegg
# Verify input device:
libinput list-devices | grep -A3 -i touch
```

SUMMARY

Quick Reference — Package Checklist

GPU (AMD)

```
mesa vulkan-radeon
libva-mesa-driver
xf86-video-amdgpu (X11 only)
```

GPU (Intel)

```
mesa vulkan-intel
intel-media-driver
intel-gpu-tools
```

GPU (NVIDIA)

```
nvidia nvidia-utils
lib32-nvidia-utils
nvidia-settings
```

NVIDIA Hybrid

```
envycontrol (AUR)
optimus-manager (AUR)
prime-run (script)
```

Battery

```
tlp tlp-rdw
auto-cpufreq (AUR)
thermald powertop
```

Touchpad Gestures

```
libinput-gestures (AUR)
touchegg (AUR)
xdotool
```

Fingerprint

```
fprintd libfprint
python-validity (AUR)
libfprint-tod (AUR)
```

Sleep/Hibernate

```
systemd (built-in)
nvidia-suspend.service
resume hook in mkinitcpio
```

HP Specific

```
hp-wmi (kernel built-in)
hp-wmi-sensors (AUR)
lm_sensors
```

KDE Extras

```
power-profiles-daemon
plasma-systemmonitor
kde-gtk-config
```

pacman — Essential Commands

Install, Upgrade & Remove

- pacman -S <pkg> — install package
- pacman -Syu — full system upgrade
- pacman -Syy — force refresh package DB
- pacman -Runcs <pkg> — remove with deps & config
- pacman -Runcs $(pacman -Qdtq) — remove orphans

Search & Info

- pacman -Ss <term> — search remote repos
- pacman -Qs <term> — search locally installed
- pacman -Si <pkg> — remote package info
- pacman -Qi <pkg> — installed package info
- pacman -Qe — list explicitly installed
- pacman -Qd — list installed as deps

Extra tips

```
# Enable parallel downloads (faster installs):
sed -i 's/^#ParallelDownloads/ParallelDownloads/g' /etc/pacman.conf

# Enable colored output:
sed -i 's/^#Color/Color/g' /etc/pacman.conf

# Enable verbose package list + Pac-Man progress bar easter egg:
sed -i 's/^#VerbosePkgLists/VerbosePkgLists/' /etc/pacman.conf
sed -i 's/VerbosePkgLists/VerbosePkgLists\nILoveCandy/' /etc/pacman.conf
```

AUR — Manual git Install & paru Helper

Manual AUR Install (git)

- Install git first: pacman -S git
- Create a workspace: mkdir ~/git && cd ~/git
- Clone AUR package: git clone <aur-url>
- cd into folder, then run: makepkg -si
- makepkg deps: pacman -S debugedit fakeroot sudo patch
- Update AUR pkg: git pull && makepkg -si

paru — AUR Helper

- paru wraps pacman and handles AUR automatically
- Install paru itself via git (bootstrap once):
- git clone https://aur.archlinux.org/paru.git
- cd paru && makepkg -si
- paru -S <pkg> — install from AUR or repos
- paru -Syu — upgrade system + AUR packages
- paru -Ss <term> — search AUR

```
# Bootstrap paru AUR helper:
pacman -S git base-devel
cd /tmp
git clone https://aur.archlinux.org/paru.git
cd paru && makepkg -si

# Example AUR installs with paru:
paru -S teams-for-linux-bin brave-bin wps-office
```

```
# Manual AUR example (WPS Office):
mkdir ~/git && cd ~/git
git clone https://aur.archlinux.org/wps-office.git
cd wps-office && makepkg -si
# WPS PDF export requires libtiff5 from AUR:
paru -S libtiff5
```

firejail — App Sandboxing & Network Isolation

What is firejail?

- Sandboxes applications using Linux namespaces
- Restricts file system, network, and system call access
- Useful for: office suites, browsers, untrusted apps
- No root required to run sandboxed apps
- Install: pacman -S firejail
- Profiles in: /etc/firejail/*.profile

Block Internet Access for an App

- Edit or create app profile: nano /etc/firejail/wps.profile
- Uncomment or add line: net none
- Rename original binary: mv /usr/bin/wps /usr/bin/wps_orig
- Create wrapper script at /usr/bin/wps:
- #!/bin/bash
- exec firejail --noprofile "/usr/bin/wps_orig" "$@"
- chmod +x /usr/bin/wps — make wrapper executable

```
# Install firejail:
pacman -S firejail

# Create wrapper to launch WPS Office sandboxed (no internet):
mv /usr/bin/wps /usr/bin/wps_orig
mv /usr/bin/wpspdf /usr/bin/wpspdf_orig

cat > /usr/bin/wps << 'EOF'
#!/bin/bash
exec firejail --noprofile "/usr/bin/wps_orig" "$@"
EOF

cat > /usr/bin/wpspdf << 'EOF'
#!/bin/bash
exec firejail --noprofile "/usr/bin/wpspdf_orig" "$@"
EOF

chmod +x /usr/bin/wps /usr/bin/wpspdf
```

Wi-Fi Access Point — Setup & DHCP (Part 1/3)

Required Packages

- pacman -S dnsmasq hostapd fail2ban iptables
- dnsmasq — simple DHCP and DNS server
- hostapd — Wi-Fi access point daemon
- fail2ban — brute-force attack blocker
- iptables — packet routing and NAT firewall
- Tip: supports dual-radio (two Wi-Fi networks)

Use Case

- Turn laptop/desktop into a wireless router
- Route one Wi-Fi through normal internet (wlan0)
- Route second Wi-Fi through TOR network (wlan1)
- Useful for: shared internet, privacy research, labs
- Check Wi-Fi interfaces: ip link show
- Note: verify your interface names (may be wlp3s0 etc.)

```
# /etc/dhcpcd.conf – static IPs for both Wi-Fi interfaces:
echo "interface wlan0
    static ip_address=192.168.1.1/24
    nohook wpa_supplicant
interface wlan1
    static ip_address=192.168.2.1/24
    nohook wpa_supplicant" >> /etc/dhcpcd.conf

# /etc/dnsmasq.conf – DHCP ranges:
mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
echo "interface=wlan0
dhcp-range=192.168.1.20,192.168.1.200,255.255.255.0,24h
interface=wlan1
dhcp-range=192.168.2.20,192.168.2.200,255.255.255.0,24h" > /etc/dnsmasq.conf
```

Wi-Fi Access Point — hostapd & Routing (Part 2/3)

```
# /etc/hostapd/wlan0.conf — regular Wi-Fi:
echo "interface=wlan0
country_code=PL
ssid=MyNetwork_WiFi
hw_mode=g
channel=11
wmm_enabled=0
auth_algs=1
wpa=2
wpa_passphrase=YourPassword1
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP" > /etc/hostapd/wlan0.conf

# /etc/hostapd/wlan1.conf — TOR-routed Wi-Fi:
echo "interface=wlan1
country_code=PL
ssid=TOR_Network
hw_mode=g
channel=6
wmm_enabled=0
auth_algs=1
wpa=2
wpa_passphrase=YourPassword2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP" > /etc/hostapd/wlan1.conf
```

```
# Enable IP forwarding:
echo 'net.ipv4.ip_forward=1' >> /etc/sysctl.d/99-sysctl.conf
sysctl -p

# iptables NAT routing (replace enp0s3 with your uplink iface):
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
iptables -A FORWARD -i enp0s3 -o wlan0 -m state \
  --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i wlan0 -o enp0s3 -j ACCEPT
iptables -A FORWARD -i enp0s3 -o wlan1 -m state \
  --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i wlan1 -o enp0s3 -j ACCEPT

# Save rules and enable all services:
iptables-save > /etc/iptables/iptables.rules
rfkill unblock wlan
systemctl enable --now dhcpcd dnsmasq \
  hostapd@wlan0 hostapd@wlan1 fail2ban
```

Wi-Fi Access Point — TOR Transparent Proxy (Part 3/3)

How TOR Routing Works

- wlan1 clients get all traffic routed through TOR
- iptables redirects TCP to TOR TransPort (9040)
- DNS queries redirected to TOR DNSPort (53)
- wlan0 clients use normal internet
- TOR hides source IP for wlan1 devices
- Note: TOR service needs restart after reboot

```
# Install TOR:
pacman -S tor torsocks
cp /etc/tor/torrc /etc/tor/torrc.bak
# Configure TOR transparent proxy on wlan1 interface (192.168.2.1):
echo "Log notice file /var/log/tor/notices.log
VirtualAddrNetwork 10.192.0.0/10
AutomapHostsSuffixes .onion,.exit
AutomapHostsOnResolve 1
TransListenAddress 192.168.2.1
TransPort 192.168.2.1:9040
DNSPort 192.168.2.1:53
DataDirectory /var/lib/tor" > /etc/tor/torrc
```

```
# iptables rules to redirect wlan1 traffic through TOR:
iptables -t nat -A PREROUTING -i wlan1 -p tcp --dport 22 -j REDIRECT --to-ports 22
iptables -t nat -A PREROUTING -i wlan1 -p udp --dport 53 -j REDIRECT --to-ports 53
iptables -t nat -A PREROUTING -i wlan1 -p tcp --syn -j REDIRECT --to-ports 9040
iptables-save > /etc/iptables/iptables.rules

# Create TOR log directory and enable service:
mkdir -p /var/log/tor && chmod 777 /var/log/tor
touch /var/log/tor/notices.log && chmod 666 /var/log/tor/notices.log
systemctl enable --now tor
# Note: if TOR doesn't connect after reboot, run: systemctl restart tor
```

SSH Passwordless Login & MiniDLNA Media Server

SSH — Passwordless Login

- Generate key pair (if not already done): ssh-keygen
- Copy public key to remote host:
- ssh-copy-id user@remote_host_ip
- Keys stored in: ~/.ssh/authorized_keys on remote
- Test: ssh user@remote_host_ip — no password prompt
- Config in: ~/.ssh/config for per-host settings

```
# SSH key setup:
ssh-keygen -t ed25519 -C "mykey"
ssh-copy-id username@192.168.1.100
# Test:
ssh username@192.168.1.100

# Optional: ~/.ssh/config
Host myserver
  HostName 192.168.1.100
  User username
  IdentityFile ~/.ssh/id_ed25519
```

MiniDLNA — Media Server

- Streams media (video, music, photos) on local network
- Clients: smart TVs, phones, media players via DLNA/UPnP
- Install: pacman -S minidlna
- Create media folder: mkdir /media_share
- Edit config: /etc/minidlna.conf
- Enable service: systemctl enable --now minidlna

```
# /etc/minidlna.conf:
echo "media_dir=/media_share
network_interface=wlan0
port=8200
friendly_name=NET_PLAYER
model_name=Windows Media Connect compatible (MiniDLNA)
inotify=yes
inotify_interval=900
album_art_names=Cover.jpg/cover.jpg/Folder.jpg/folder.jpg" > /etc/minidlna.conf

mkdir /media_share
systemctl enable --now minidlna
```

ARCH LINUX + KDE PLASMA

Ready to Optimize Your Laptop

Refer to the Arch Wiki for latest driver updates.
All AUR packages: install via yay or paru.

wiki.archlinux.org · aur.archlinux.org · asahilinux.org