# This one.

**1️⃣ Check your kernel version**

uname -r

Make sure you have the matching linux-headers package for your current kernel.

sudo apt install linux-headers-$(uname -r)

**2️⃣ Remove and rebuild v4l2loopback**

Sometimes dkms builds it for a different kernel version.

sudo apt remove v4l2loopback-dkms --purge

sudo apt install v4l2loopback-dkms

This will rebuild it for your current kernel.

**3️⃣ Load with parameters (required for some kernels)**

If you just run sudo modprobe v4l2loopback, some kernels reject it without parameters.  
Try:

sudo modprobe v4l2loopback devices=1 video\_nr=10 card\_label="VirtualCam" exclusive\_caps=1

**4️⃣ Check if it loaded**

ls /dev/video\*

You should see something like /dev/video10.

**5️⃣ Make it persistent (optional)**

If it works, create a config file:

echo "options v4l2loopback devices=1 video\_nr=10 card\_label='VirtualCam' exclusive\_caps=1" | sudo tee /etc/modprobe.d/v4l2loopback.conf

**6️⃣ Use with camera streaming**

Once v4l2loopback works, you can send video from your webcam, OBS, or ffmpeg to the virtual device, then multiple programs (browser + software) can read from it.

Example:

ffmpeg -f v4l2 -i /dev/video0 -f v4l2 /dev/video10

If you want, I can also show you how to **share your real webcam to multiple apps at once** in Kali using v4l2loopback + OBS without breaking your current camera feed.

# Do you want me to prepare that setup?

Alright, let’s go step-by-step from the very beginning so we can fix your **"Invalid argument"** issue and then get **v4l2loopback** + **OBS** working in Kali without breaking your main webcam feed.

**1️⃣ First – Remove any loaded broken module**

Sometimes a previous attempt leaves a partially loaded module in memory.

sudo modprobe -r v4l2loopback

If it says "Module not found," that's fine.

**2️⃣ Check Kernel Version**

Since v4l2loopback is a kernel module, the DKMS build must match your **running kernel**.

uname -r

If the kernel version here doesn’t match the modules in /lib/modules/$(uname -r), you’ll get this error.

To make sure DKMS rebuilt for your current kernel:

sudo dkms autoinstall

**3️⃣ Load the Module with Correct Parameters**

Invalid argument usually happens when the options are wrong.  
Let’s load it clean with a working config:

sudo modprobe v4l2loopback devices=1 video\_nr=10 card\_label="VirtualCam" exclusive\_caps=1

* devices=1 → only one virtual camera
* video\_nr=10 → creates /dev/video10 (avoids clashing with your real /dev/video0)
* card\_label="VirtualCam" → what apps will see as its name
* exclusive\_caps=1 → allows OBS virtual cam to work with apps like Zoom/Meet

**4️⃣ Verify It Loaded**

ls /dev/video\*

You should see something like:

/dev/video0 # your real webcam

/dev/video10 # your virtual webcam

Also check:

v4l2-ctl --list-devices

**5️⃣ Set Up in OBS**

1. Open **OBS**
2. **Tools → VirtualCam → Start**
3. In v4l2sink (Linux), set it to /dev/video10  
   If you don't see VirtualCam in OBS:
4. sudo apt install obs-studio v4l2loopback-dkms v4l-utils

**6️⃣ Use in Multiple Apps Without Breaking Real Webcam**

You can now select:

* **Real webcam** (/dev/video0) in OBS
* Output to **VirtualCam** (/dev/video10)
* Other apps (Zoom, Meet, Discord) use /dev/video10 instead of /dev/video0  
  This way multiple apps see the same feed without “camera already in use” errors.

💡 **Common reason for "Invalid argument"** in your case:  
You were probably loading it without required parameters or with wrong defaults that your kernel build rejects.  
The exclusive\_caps=1 and unique video\_nr usually solve this.

If you want, I can give you a **single bash script** so you just run ./start-virtualcam.sh and it sets up v4l2loopback + starts OBS VirtualCam instantly.  
Would you like me to make that script?

Gotcha — you want your webcam to be available **at the same time** to both:

1. **Desktop software** (like OBS, Zoom, etc.)
2. **Web browsers** (Google Meet, Jitsi, etc.)

**The problem**

Most webcams can only be accessed by **one application at a time** in Linux.  
If OBS is using it, Chrome/Firefox can’t, and vice versa.  
We solve this by **splitting/cloning** the video stream using v4l2loopback.

**The solution: OBS ➡ Virtual Camera ➡ All Apps**

Here’s how you do it in Kali Linux:

**1️⃣ Install required packages**

sudo apt update

sudo apt install v4l2loopback-dkms obs-studio

**2️⃣ Load v4l2loopback module**

We create a **virtual camera device** that OBS can output to, and browsers can read from.

sudo modprobe v4l2loopback devices=1 video\_nr=10 card\_label="OBS-VirtualCam" exclusive\_caps=0

**Explanation:**

* devices=1 → create 1 virtual camera
* video\_nr=10 → it appears as /dev/video10
* card\_label="OBS-VirtualCam" → name in apps
* exclusive\_caps=0 → makes it usable in browsers like Chrome & Firefox

**3️⃣ Start OBS and enable virtual camera**

1. Open **OBS Studio**
2. Add your **real webcam** as a source
3. Click **Start Virtual Camera** (bottom right)
4. Select /dev/video10 (OBS-VirtualCam) in other apps

**4️⃣ Use in browser**

* Open **Chrome** or **Firefox**
* Go to **Google Meet** or **Jitsi**
* Select **OBS-VirtualCam** as your camera

**5️⃣ Optional: Keep your real webcam free for other software**

If you want **both OBS-VirtualCam and your real camera** available simultaneously:

* In OBS, don’t “take over” the camera exclusively —  
  use **V4L2 Source** → Device → your real webcam,  
  but leave **exclusive\_caps=0** when loading v4l2loopback.

✅ **Now both your browser and desktop apps can use the camera at the same time** — browser sees the OBS virtual cam, and OBS gets your real webcam feed.

If you want, I can also show you **how to stream the same OBS camera to multiple virtual devices** so you can join two Google Meets at once. That’s a fun hack.

# On obs setting up

Alright — here’s your **one-command startup script** for OBS Virtual Camera that works in Chrome (webcamtests.com, StudyStream, Meet, etc.) and other apps **at the same time**.

Save it as:  
start-obs-virtualcam.sh

#!/bin/bash

# start-obs-virtualcam.sh

# Starts OBS Virtual Camera for Chrome + other apps

VIRTUAL\_CAM\_NR=10

VIRTUAL\_CAM\_LABEL="OBS-VirtualCam"

echo "[1/3] Installing required packages..."

sudo apt update

sudo apt install -y obs-studio v4l2loopback-dkms v4l-utils

echo "[2/3] Loading v4l2loopback module..."

sudo modprobe -r v4l2loopback 2>/dev/null

sudo modprobe v4l2loopback devices=1 video\_nr=${VIRTUAL\_CAM\_NR} card\_label="${VIRTUAL\_CAM\_LABEL}" exclusive\_caps=0

if [ $? -ne 0 ]; then

echo "❌ Failed to load v4l2loopback. Please check your kernel and module installation."

exit 1

fi

echo "[3/3] Starting OBS..."

echo "✅ Virtual camera created at /dev/video${VIRTUAL\_CAM\_NR} (${VIRTUAL\_CAM\_LABEL})"

echo "➡ In OBS: Add your real webcam as a source, then click 'Start Virtual Camera'."

echo "➡ In Chrome/Firefox: Select '${VIRTUAL\_CAM\_LABEL}' as your camera."

obs &

**How to use**

chmod +x start-obs-virtualcam.sh

./start-obs-virtualcam.sh

1. Script will:
   * Install required packages (only first time)
   * Create /dev/video10 as **OBS-VirtualCam**
   * Launch OBS
2. In OBS:
   * Add your **real webcam** as a Video Capture Device
   * Click **Start Virtual Camera**
3. In Chrome / webcamtests.com / StudyStream:
   * Select **OBS-VirtualCam** as the camera

⚡ With this setup:

* OBS uses your **real webcam** once.
* All other apps see **OBS-VirtualCam** — no “camera in use” error.

Do you want me to also make an **auto-stop script** so when you’re done, it stops OBS, unloads the virtual cam, and frees system resources? That way you won’t have /dev/video10 hanging around.

# From start –

Perfect — Kali with systemd and XFCE is straightforward to set up.

Here’s the plan:

**1️⃣ Updated obs-vcam-auto.sh (permanent hide until manual unlock)**

Save this as ~/bin/obs-vcam-auto.sh (make sure executable):

#!/usr/bin/env bash

set -euo pipefail

REAL\_DEV="/dev/video0"

VDEV="/dev/video10"

MOD\_OPTS="devices=1 video\_nr=10 card\_label=OBS-VirtualCam exclusive\_caps=0"

OBS\_CMD="obs"

OBS\_WEBSOCKET\_PORT=4455

OBS\_WEBSOCKET\_PASS="changeme"

kill\_users() {

echo "[\*] Killing processes using $REAL\_DEV..."

PIDS=$(lsof -t "$REAL\_DEV" 2>/dev/null || true)

if [ -n "$PIDS" ]; then

echo "Found: $PIDS"

sudo kill $PIDS || true

sleep 1

PIDS\_LEFT=$(lsof -t "$REAL\_DEV" 2>/dev/null || true)

if [ -n "$PIDS\_LEFT" ]; then

echo "Force killing: $PIDS\_LEFT"

sudo kill -9 $PIDS\_LEFT || true

fi

else

echo "No processes using $REAL\_DEV."

fi

}

hide\_real\_cam() {

echo "[\*] Hiding $REAL\_DEV from other apps..."

sudo chmod 000 "$REAL\_DEV" || true

}

reload\_loopback() {

echo "[\*] Reloading v4l2loopback..."

sudo modprobe -r v4l2loopback 2>/dev/null || true

sleep 0.5

sudo modprobe v4l2loopback $MOD\_OPTS

sleep 0.5

if [ -e "$VDEV" ]; then

echo "[OK] Virtual cam ready at $VDEV"

else

echo "[!] Virtual cam failed to load"

fi

}

start\_obs\_background() {

if pgrep -x obs >/dev/null 2>&1; then

echo "[\*] OBS already running."

else

echo "[\*] Starting OBS in background..."

nohup $OBS\_CMD >/dev/null 2>&1 &

sleep 5

fi

}

wait\_for\_obs\_websocket() {

echo "[\*] Waiting for OBS WebSocket on port $OBS\_WEBSOCKET\_PORT..."

for i in {1..20}; do

if timeout 1 bash -c "echo > /dev/tcp/127.0.0.1/$OBS\_WEBSOCKET\_PORT" 2>/dev/null; then

echo "[OK] OBS WebSocket is ready."

return 0

fi

sleep 1

done

echo "[!] Timed out waiting for OBS WebSocket."

return 1

}

start\_virtual\_camera\_obs() {

wait\_for\_obs\_websocket

echo "[\*] Sending StartVirtualCam command to OBS..."

curl -s \

-H "Content-Type: application/json" \

-u ":$OBS\_WEBSOCKET\_PASS" \

-d '{"op":6,"d":{"requestType":"StartVirtualCam","requestId":"1"}}' \

http://localhost:$OBS\_WEBSOCKET\_PORT > /dev/null

echo "[OK] Virtual camera started automatically."

}

unhide\_real\_cam() {

echo "[\*] Restoring $REAL\_DEV permissions..."

sudo chmod 666 "$REAL\_DEV" || true

}

status\_info() {

echo

echo "================ STATUS ================"

v4l2-ctl --list-devices || true

echo "OBS PID: $(pgrep -x obs || echo 'Not running')"

echo "Real cam permissions: $(ls -l $REAL\_DEV 2>/dev/null || echo 'not found')"

echo "========================================"

}

if [[ "${1:-}" == "--unlock" ]]; then

unhide\_real\_cam

exit 0

fi

echo "[1/5] Killing any process using webcam..."

kill\_users

echo "[2/5] Hiding real webcam from other apps..."

hide\_real\_cam

echo "[3/5] Reloading v4l2loopback..."

reload\_loopback

echo "[4/5] Starting OBS..."

start\_obs\_background

echo "[5/5] Starting Virtual Camera via OBS WebSocket..."

start\_virtual\_camera\_obs

status\_info

echo

echo "✅ Setup complete!"

echo "➡ In Discord & Studystream, select: OBS-VirtualCam"

echo "➡ Real webcam ($REAL\_DEV) is hidden from all apps except OBS."

echo "➡ Run './obs-vcam-auto.sh --unlock' to restore webcam permissions."

**2️⃣ Create a simple unlock script (optional shortcut)**

Save this as ~/bin/obs-vcam-unlock.sh:

#!/usr/bin/env bash

~/bin/obs-vcam-auto.sh --unlock

echo "Webcam permissions restored."

Make it executable:

chmod +x ~/bin/obs-vcam-unlock.sh

**3️⃣ Create a systemd user service to run at login**

Save this as ~/.config/systemd/user/obs-vcam.service:

[Unit]

Description=Start OBS VirtualCam and hide real webcam

After=graphical-session.target

[Service]

Type=oneshot

ExecStart=/home/YOUR\_USERNAME/bin/obs-vcam-auto.sh

RemainAfterExit=yes

Environment=DISPLAY=:0

[Install]

WantedBy=default.target

**Replace YOUR\_USERNAME with your actual username** (check with whoami).

**4️⃣ Enable the systemd service**

Run:

systemctl --user daemon-reload

systemctl --user enable obs-vcam.service

systemctl --user start obs-vcam.service

**5️⃣ How to use**

* On login, the script runs automatically:
  + Kills any webcam users
  + Hides real webcam (/dev/video0) from apps
  + Loads virtual cam
  + Starts OBS + virtual cam auto start
* If you ever need the real webcam for something else:

~/bin/obs-vcam-auto.sh --unlock

or

~/bin/obs-vcam-unlock.sh

**Final notes**

* Make sure OBS WebSocket is enabled on port 4455 with password changeme (or update script accordingly).
* DISPLAY=:0 works for most X11 XFCE setups, adjust if you use Wayland or different display.
* You can check service logs with:

journalctl --user -u obs-vcam.service

If you want, I can help write a little .desktop file for XFCE that runs the unlock script easily from the menu or shortcuts.  
Would that be helpful?

# obs-vcam-auto.sh

#!/usr/bin/env bash

set -euo pipefail

REAL\_DEV="/dev/video1" # Updated to your real webcam device

VDEV="/dev/video10"

MOD\_OPTS="devices=1 video\_nr=10 card\_label=OBS-VirtualCam exclusive\_caps=0"

OBS\_CMD="/usr/bin/obs"

OBS\_WEBSOCKET\_PORT=4455

OBS\_WEBSOCKET\_PASS="changeme"

kill\_users() {

echo "[\*] Killing processes using $REAL\_DEV..."

PIDS=$(lsof -t "$REAL\_DEV" 2>/dev/null || true)

if [ -n "$PIDS" ]; then

echo "Found: $PIDS"

sudo kill $PIDS || true

sleep 1

PIDS\_LEFT=$(lsof -t "$REAL\_DEV" 2>/dev/null || true)

if [ -n "$PIDS\_LEFT" ]; then

echo "Force killing: $PIDS\_LEFT"

sudo kill -9 $PIDS\_LEFT || true

fi

else

echo "No processes using $REAL\_DEV."

fi

}

hide\_real\_cam() {

echo "[\*] Hiding $REAL\_DEV from other apps..."

sudo chmod 000 "$REAL\_DEV" || true

}

reload\_loopback() {

echo "[\*] Reloading v4l2loopback..."

sudo modprobe -r v4l2loopback 2>/dev/null || true

sleep 0.5

sudo modprobe v4l2loopback $MOD\_OPTS

sleep 0.5

if [ -e "$VDEV" ]; then

echo "[OK] Virtual cam ready at $VDEV"

else

echo "[!] Virtual cam failed to load"

fi

}

start\_obs\_background() {

if pgrep -x obs >/dev/null 2>&1; then

echo "[\*] OBS already running."

else

echo "[\*] Starting OBS in background..."

nohup $OBS\_CMD >/dev/null 2>&1 &

sleep 5

fi

}

wait\_for\_obs\_websocket() {

echo "[\*] Waiting for OBS WebSocket on port $OBS\_WEBSOCKET\_PORT..."

for i in {1..20}; do

if timeout 1 bash -c "echo > /dev/tcp/127.0.0.1/$OBS\_WEBSOCKET\_PORT" 2>/dev/null; then

echo "[OK] OBS WebSocket is ready."

return 0

fi

sleep 1

done

echo "[!] Timed out waiting for OBS WebSocket."

return 1

}

start\_virtual\_camera\_obs() {

wait\_for\_obs\_websocket

echo "[\*] Sending StartVirtualCam command to OBS..."

curl -s \

-H "Content-Type: application/json" \

-u ":$OBS\_WEBSOCKET\_PASS" \

-d '{"op":6,"d":{"requestType":"StartVirtualCam","requestId":"1"}}' \

http://localhost:$OBS\_WEBSOCKET\_PORT > /dev/null

echo "[OK] Virtual camera started automatically."

}

unhide\_real\_cam() {

echo "[\*] Restoring $REAL\_DEV permissions..."

sudo chmod 666 "$REAL\_DEV" || true

}

status\_info() {

echo

echo "================ STATUS ================"

v4l2-ctl --list-devices || true

echo "OBS PID: $(pgrep -x obs || echo 'Not running')"

echo "Real cam permissions: $(ls -l $REAL\_DEV 2>/dev/null || echo 'not found')"

echo "========================================"

}

if [[ "${1:-}" == "--unlock" ]]; then

unhide\_real\_cam

exit 0

fi

echo "[1/5] Killing any process using webcam..."

kill\_users

echo "[2/5] Hiding real webcam from other apps..."

hide\_real\_cam

echo "[3/5] Reloading v4l2loopback..."

reload\_loopback

echo "[4/5] Starting OBS..."

start\_obs\_background

echo "[5/5] Starting Virtual Camera via OBS WebSocket..."

start\_virtual\_camera\_obs

status\_info

echo

echo "✅ Setup complete!"

echo "➡ In Discord & Studystream, select: OBS-VirtualCam"

echo "➡ Real webcam ($REAL\_DEV) is hidden from all apps except OBS."

echo "➡ Run './obs-vcam-auto.sh --unlock' to restore webcam permissions."

# To check: frame

└─$ v4l2-ctl --all -d /dev/video10

v4l2-ctl --all -d /dev/video11

# script: obs

#!/usr/bin/env bash

set -euo pipefail

# === CONFIG ===

REAL\_DEV="/dev/video1" # Your physical webcam

VDEV1="/dev/video10" # Main OBS virtual cam

VDEV2="/dev/video11" # Duplicate cam

MOD\_OPTS="devices=2 video\_nr=10,11 card\_label=OBS-Discord,OBS-Study exclusive\_caps=0"

OBS\_CMD="/usr/bin/obs"

OBS\_WEBSOCKET\_PORT=4455

OBS\_WEBSOCKET\_PASS="changeme"

# === COLORS ===

RED="\033[1;31m"

GREEN="\033[1;32m"

NC="\033[0m" # No Color

kill\_users() {

echo "[\*] Killing processes using $REAL\_DEV..."

PIDS=$(lsof -t "$REAL\_DEV" 2>/dev/null || true)

if [ -n "$PIDS" ]; then

echo "Found: $PIDS"

sudo kill $PIDS || true

sleep 1

PIDS\_LEFT=$(lsof -t "$REAL\_DEV" 2>/dev/null || true)

if [ -n "$PIDS\_LEFT" ]; then

echo "Force killing: $PIDS\_LEFT"

sudo kill -9 $PIDS\_LEFT || true

fi

else

echo "No processes using $REAL\_DEV."

fi

}

hide\_real\_cam() {

echo "[\*] Hiding $REAL\_DEV from other apps..."

sudo chmod 000 "$REAL\_DEV" || true

}

reload\_loopback() {

echo "[\*] Reloading v4l2loopback..."

sudo modprobe -r v4l2loopback 2>/dev/null || true

sleep 0.5

sudo modprobe v4l2loopback $MOD\_OPTS

sleep 0.5

if [ -e "$VDEV1" ] && [ -e "$VDEV2" ]; then

echo "[OK] Virtual cams ready: $VDEV1 (OBS Discord), $VDEV2 (OBS Study)"

else

echo "[!] One or both virtual cams failed to load"

fi

}

start\_obs\_background() {

if pgrep -x obs >/dev/null 2>&1; then

echo "[\*] OBS already running."

else

echo "[\*] Starting OBS in background..."

nohup $OBS\_CMD >/dev/null 2>&1 &

sleep 5

fi

}

wait\_for\_obs\_websocket() {

echo "[\*] Waiting for OBS WebSocket on port $OBS\_WEBSOCKET\_PORT..."

for i in {1..30}; do

if timeout 1 bash -c "echo > /dev/tcp/127.0.0.1/$OBS\_WEBSOCKET\_PORT" 2>/dev/null; then

echo "[OK] OBS WebSocket is ready."

return 0

fi

sleep 1

done

echo "[!] Timed out waiting for OBS WebSocket."

return 1

}

start\_virtual\_camera\_obs() {

wait\_for\_obs\_websocket

echo "[\*] Sending StartVirtualCam command to OBS..."

curl -s \

-H "Content-Type: application/json" \

-u ":$OBS\_WEBSOCKET\_PASS" \

-d '{"op":6,"d":{"requestType":"StartVirtualCam","requestId":"1"}}' \

http://localhost:$OBS\_WEBSOCKET\_PORT > /dev/null

echo "[OK] Virtual camera started in OBS."

}

wait\_for\_video\_signal() {

echo "[\*] Waiting for $VDEV1 to produce video..."

while true; do

if ffmpeg -f v4l2 -i "$VDEV1" -vframes 1 -f null - 2>/dev/null; then

echo "[OK] $VDEV1 is active."

break

fi

sleep 1

done

}

start\_ffmpeg\_duplication() {

echo "[\*] Starting ffmpeg to duplicate $VDEV1 -> $VDEV2..."

nohup ffmpeg -f v4l2 -i "$VDEV1" -f v4l2 "$VDEV2" >/dev/null 2>&1 &

sleep 1

}

monitor\_ffmpeg\_and\_signal() {

while true; do

# Restart ffmpeg if it's not running

if ! pgrep -x ffmpeg >/dev/null; then

echo "[!] ffmpeg stopped! Restarting..."

start\_ffmpeg\_duplication

fi

# Check signal

if ffmpeg -f v4l2 -i "$VDEV1" -vframes 1 -f null - 2>/dev/null; then

echo -e "Heartbeat: ${GREEN}OK${NC} — signal is live"

else

echo -e "Heartbeat: ${RED}NO SIGNAL${NC}"

fi

sleep 10

done

}

unhide\_real\_cam() {

echo "[\*] Restoring $REAL\_DEV permissions..."

sudo chmod 666 "$REAL\_DEV" || true

}

status\_info() {

echo

echo "================ STATUS ================"

v4l2-ctl --list-devices || true

echo "OBS PID: $(pgrep -x obs || echo 'Not running')"

echo "ffmpeg PID: $(pgrep -x ffmpeg || echo 'Not running')"

echo "Real cam permissions: $(ls -l $REAL\_DEV 2>/dev/null || echo 'not found')"

echo "========================================"

}

cleanup() {

echo

echo "[\*] Caught exit signal — cleaning up..."

pkill -x ffmpeg 2>/dev/null || true

unhide\_real\_cam

echo "[\*] Cleanup done."

}

trap cleanup EXIT

if [[ "${1:-}" == "--unlock" ]]; then

unhide\_real\_cam

exit 0

fi

echo "[1/6] Killing any process using webcam..."

kill\_users

echo "[2/6] Hiding real webcam from other apps..."

hide\_real\_cam

echo "[3/6] Reloading v4l2loopback..."

reload\_loopback

echo "[4/6] Starting OBS..."

start\_obs\_background

echo "[5/6] Starting Virtual Camera via OBS WebSocket..."

start\_virtual\_camera\_obs

echo "[6/6] Waiting for OBS feed..."

wait\_for\_video\_signal

start\_ffmpeg\_duplication

status\_info

echo

echo "✅ Setup complete!"

echo "➡ Discord: select 'OBS-Discord'"

echo "➡ StudyStream: select 'OBS-Study'"

echo "➡ Run './obs-vcam-auto.sh --unlock' to restore webcam permissions."

# Keep monitoring forever

monitor\_ffmpeg\_and\_signal

# without obs setting up two virtual camera.

#!/bin/bash

set -e

# === CONFIG ===

NUM\_VCAMS=2 # Change to 1 if you only want one virtual cam

# === FUNCTIONS ===

find\_real\_cam() {

for dev in /dev/video\*; do

if v4l2-ctl --device="$dev" --all &>/dev/null; then

if ! v4l2-ctl --device="$dev" --all | grep -q "v4l2 loopback"; then

echo "$dev"

return

fi

fi

done

}

kill\_using\_video() {

echo "[\*] Killing processes using /dev/video\*..."

sudo fuser -k /dev/video\* 2>/dev/null || true

}

load\_v4l2loopback() {

echo "[\*] Loading v4l2loopback..."

sudo modprobe -r v4l2loopback 2>/dev/null || true

sudo modprobe v4l2loopback devices=$NUM\_VCAMS exclusive\_caps=1

}

find\_vcam\_devices() {

for dev in /dev/video\*; do

if v4l2-ctl --device="$dev" --all &>/dev/null; then

if v4l2-ctl --device="$dev" --all | grep -q "v4l2 loopback"; then

echo "$dev"

fi

fi

done

}

start\_ffmpeg() {

echo "[\*] Starting ffmpeg loop..."

local outputs=()

for vdev in "${VCAM\_DEVICES[@]}"; do

outputs+=(-f v4l2 -pix\_fmt yuv420p "$vdev")

done

ffmpeg -f v4l2 -i "$REAL\_CAM" \

-vf format=yuv420p \

"${outputs[@]}"

}

# === MAIN SCRIPT ===

kill\_using\_video

REAL\_CAM=$(find\_real\_cam)

echo "[OK] Real camera found: $REAL\_CAM"

load\_v4l2loopback

VCAM\_DEVICES=($(find\_vcam\_devices))

echo "[OK] Virtual cams: ${VCAM\_DEVICES[\*]}"

start\_ffmpeg

## device name:

dev1: virtualcam (v412loopback-virtualcam

dev2: dummy video device 0\*0000

# s