



I had a problem getting the P2 Pro working on Ubuntu 22.04 with a TOPDON TC001 already connected. After removing all other usb devices except for the keyboard and mouse, I ran the same code I had for the TOPDON TC001 and it worked. It's not liking any device numbers over 0. It works for negative device numbers, but that seems to just pull the same device.

Sources

[PDF] Ubuntu Linux 22.04 LTS Installation - Lenovo

Trackpad stopped working suddenly on Ubuntu and Windows after

Does AMD Ryzen 5 PRO 465

download.lenovo

answers.microsoft

community.acer

Answer

It seems you encountered an issue with USB device compatibility when using multiple devices on Ubuntu 22.04 with your Lenovo ThinkStation P2. This problem appears to be related to device enumeration and conflicts between different USB devices.

To address this issue, consider the following:

USB Device Management

1. **Device Enumeration:** Ubuntu may be assigning device numbers differently than expected, causing conflicts when multiple USB devices are connected.
2. **Device Priority:** The system might be prioritizing certain devices over others, leading to recognition issues for additional peripherals.

Troubleshooting Steps

1. **Limit Connected Devices:** As you discovered, removing all USB devices except for essential input devices (keyboard and mouse) allowed the TOPDON TC001 to function properly.
2. **Device Numbers:** The system seems to work with device number 0 and negative numbers, which likely correspond to the first recognized device.

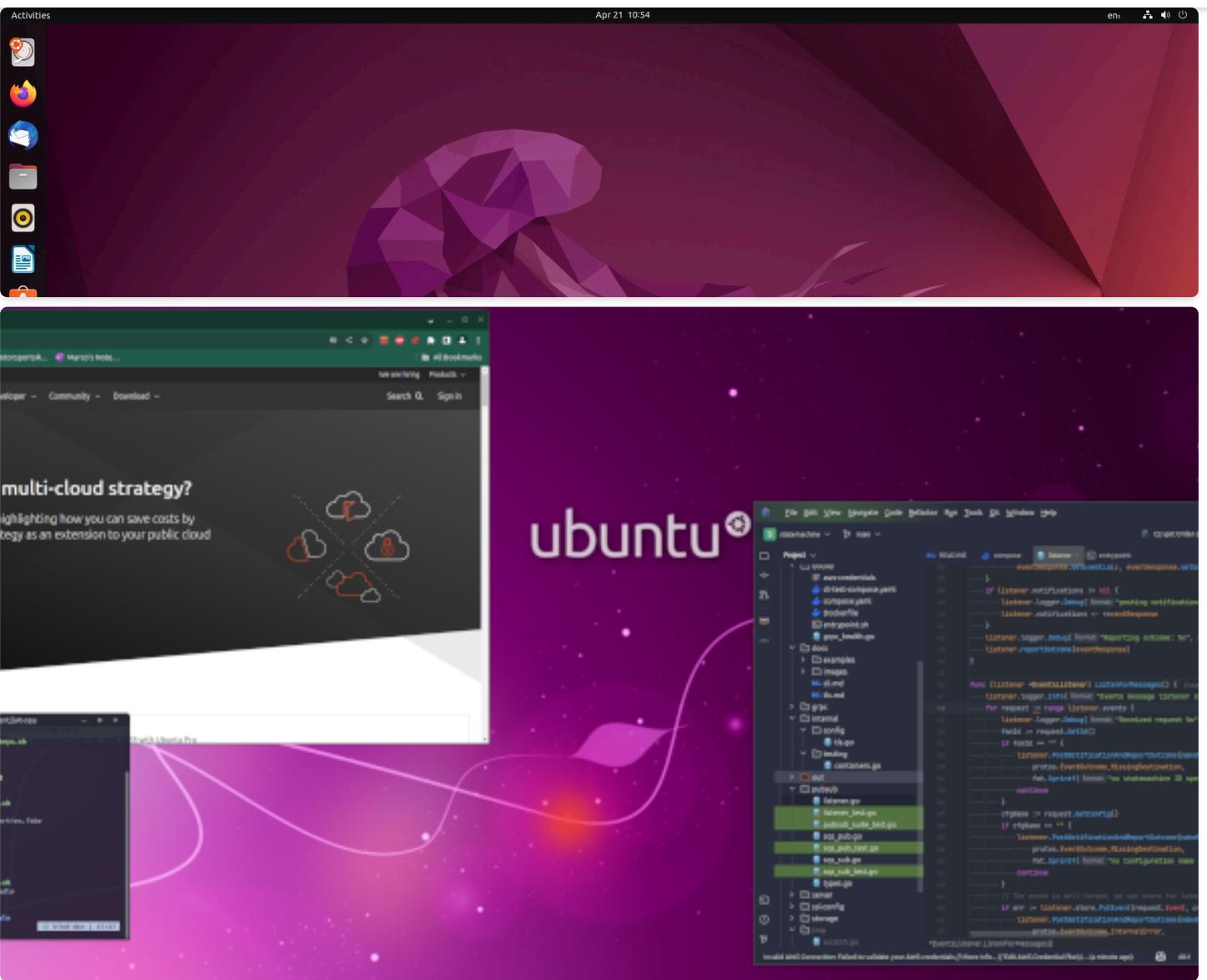
Potential Solutions

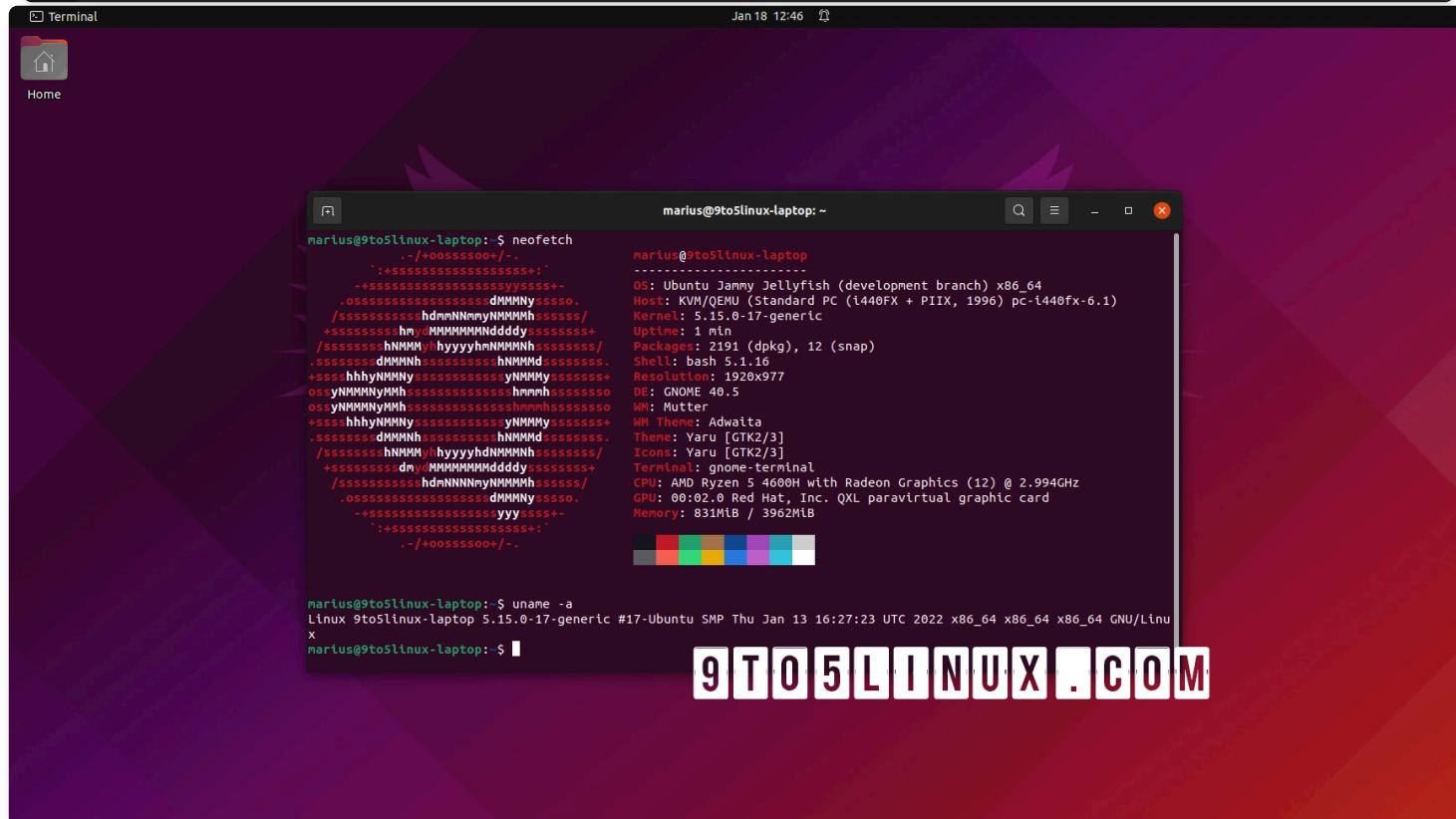
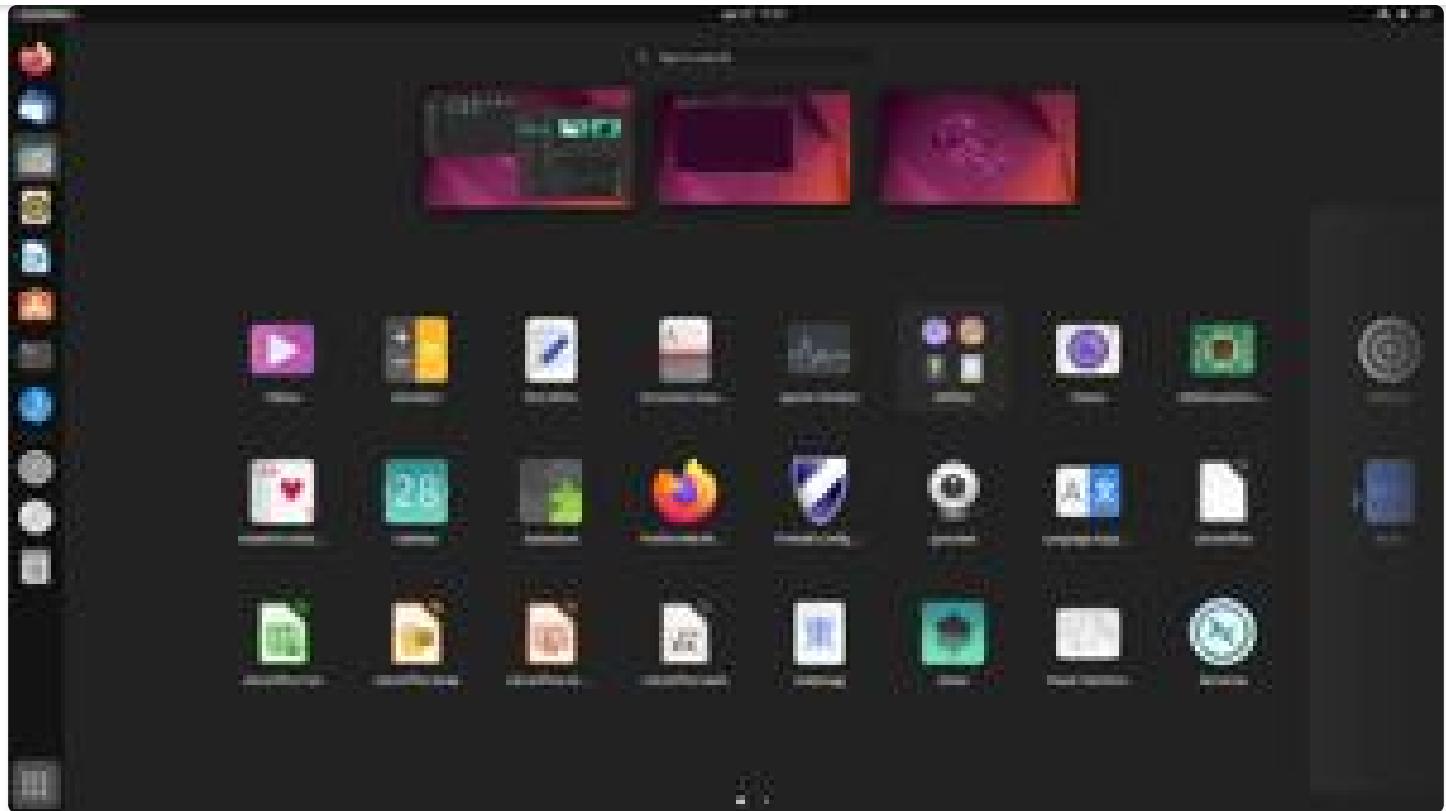
1. **USB Port Assignment:** Try connecting your devices to different USB ports on the ThinkStation P2. Some ports may have higher priority or be managed differently by the system.
2. **Update USB Drivers:** Ensure your system has the latest USB drivers and firmware updates for the ThinkStation P2 ¹.
3. **udev Rules:** Consider creating custom udev rules to assign specific device numbers or priorities to your USB devices.
4. **Kernel Parameters:** You may need to adjust kernel parameters related to USB device handling to improve compatibility with multiple devices.

Additional Considerations

- **BIOS Settings:** Check the BIOS settings of your ThinkStation P2 for any USB-related configurations that might affect device recognition ¹.
- **Ubuntu Version:** Ensure you're running the latest updates for Ubuntu 22.04 LTS, as newer versions may include fixes for USB device handling ⁹.

If the issue persists, you may want to consult Lenovo support or the Ubuntu community forums for more specific guidance related to the ThinkStation P2 and its USB compatibility with multiple devices.





9T05LINUX.COM



[View More](#)

When you do a device list with "v4l2-ctl --list-devices", can you tell from the output what device number you should use for each?

• Sources

[v4l2-ctl\(1\) — v4l-utils — Debian testing](#)

[Getting the v4l2 device number for a connected USB camera ...](#)

[How to check available webc](#)

manpages.debian

stackoverflow

askubuntu

Answer

Yes, when you use the command "v4l2-ctl --list-devices", you can determine the device numbers for each video device from the output. The command provides a list of video devices along with their associated device files [1](#) [5](#).

The output typically follows this format:

1. It lists the name or description of the video device.
2. Below each device name, it shows the corresponding device file(s) in the /dev directory.

For example, the output might look like this:

```
text
USB Camera (usb-0000:00:14.0-1):
/dev/video0
/dev/video1

Integrated Camera:
/dev/video2
```

- The Integrated Camera is associated with /dev/video2

To use a specific camera in your code or with other v4l2 commands, you would use the device number that appears after "/dev/video". So, for the USB Camera, you could use device numbers 0 or 1, and for the Integrated Camera, you would use device number 2^{4 5}.

It's important to note that some devices may have multiple entries, such as one for video and another for metadata. In recent Ubuntu versions (19.10 and later), you might see pairs of devices where even-numbered devices (e.g., /dev/video0, /dev/video2) represent actual cameras, while odd-numbered devices (e.g., /dev/video1, /dev/video3) contain metadata³.

When working with these devices programmatically, you can iterate through the available devices and use ioctl calls to get more information about each one, helping you identify the correct device for your needs⁵.

Share Rewrite

The command works once but then now it says: Cannot open device /dev/video0, exiting

Sources

v4l2-ctl - Cannot open device /dev/video0, exiting. - Reddit



V4l2-ctl command not being executed - Jetson Nano



unable to open control device
GitHub



Answer

Based on your experience, it seems that the v4l2-ctl command is encountering an issue with accessing the video device after initial use. This problem can occur due to several reasons:

1. Device Disconnection: The camera might have been physically disconnected or lost connection with the system¹.
2. Device Busy: Another process might be using the camera, preventing v4l2-ctl from accessing it³.
3. Permission Issues: The user running the command might lack the necessary permissions to access the device⁵.
4. Driver Problems: The camera driver might have encountered an error or been unloaded^{1 5}.

To troubleshoot this issue, try the following steps:

1. Verify the device is still connected and recognized:

text

```
ls /dev/video*
```



This should list your video devices⁵.

2. Check if the device is being used by another process:

text

```
sudo fuser /dev/video0
```



This will show if any process is using the device.

3. Ensure you have the correct permissions:

text



This grants read and write permissions to all users ⁵.

4. Reload the USB subsystem:

text

```
sudo modprobe -r uvcvideo  
sudo modprobe uvcvideo
```



This unloads and reloads the UVC driver.

5. Check kernel logs for any errors:

text



```
dmesg | grep -i camera
```



This might reveal any driver or hardware issues ^{1 2}.

If the problem persists, try rebooting your system, as this often resolves temporary device access issues. If none of these steps work, there might be a more serious issue with your camera driver or hardware that requires further investigation.

Share Rewrite

That's weird, the video folder doesn't exist in dev any more.

Answer

The disappearance of the /dev/video* devices is indeed unusual and indicates a more significant issue. This situation can occur due to several reasons:

1. Driver Unloading: The video driver might have been unloaded or crashed.
2. udev Rules: A udev rule might be preventing the creation of video device nodes.
3. Kernel Module Issues: The necessary kernel modules for video devices might not be loaded.
4. Hardware Disconnection: If all video devices were external (like USB cameras), they might have been disconnected or lost power.

To troubleshoot this issue, try the following steps:

1. Check if the video drivers are loaded:

text



```
lsmod | grep video
```

2. Try to manually load the video driver:

text



```
sudo modprobe uvcvideo
```

3. Check system logs for any errors:

text



```
dmesg | grep -i video
```

4. Verify USB devices are still connected:

```
text
```

```
lsusb
```



5. Restart the udev service:

```
text
```

```
sudo systemctl restart udev
```



6. If you're using a USB camera, try unplugging and replugging it.

7. As a last resort, reboot your system:

```
text
```

```
sudo reboot
```



If the problem persists after a reboot, there might be a more serious issue with your system's configuration or hardware. In that case, you may need to investigate further by checking your system's BIOS settings, updating your kernel, or consulting Ubuntu's support forums for more specific guidance.

Share Rewrite



OK, I rebooted and it's available again. I did a command to list formats, "v4l2-ctl --list-formats-ext" and that yields:

ioctl: VIDIOC_ENUM_FMT

Type: Video Capture

[0]: 'YUYV' (YUYV 4:2:2)

Size: Discrete 256x192

Interval: Discrete 0.040s (25.000 fps)

Size: Discrete 256x384

Interval: Discrete 0.040s (25.000 fps)

If you were to use Python and cv2 to capture video, how would you change to the higher resolution format first?

❖ Sources

Python OpenCV access webcam maximum resolution - Stack Overflow

Why calling cv2.VideoCapture() changes the pixel format of a v4l ...

How to change video resolution
GeeksforGeeks

stackoverflow

stackoverflow

geeksforgeeks

✳ Answer

To capture video at the higher resolution format using Python and OpenCV (cv2), you can set the resolution before starting the video capture. Here's how you can do it:

```
python  
import cv2
```

```
# Create a VideoCapture object
```



Home

Discover

Spaces

Library

```
# Set the resolution to 256x384
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 256)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 384)

# Verify the settings
width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
print(f"Resolution set to: {width}x{height}")

# Capture and display video
while True:
    ret, frame = cap.read()
    if not ret:
        break

    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture object and close windows
cap.release()
cv2.destroyAllWindows()
```

This script does the following:

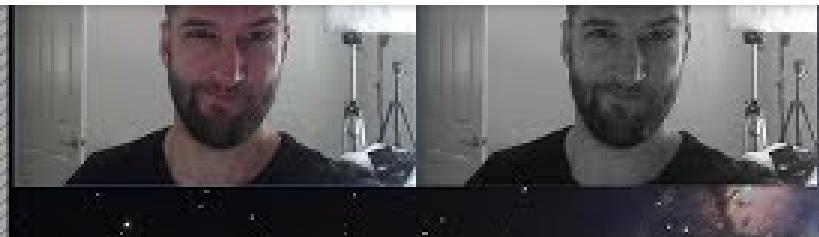
1. It creates a VideoCapture object for the default camera (index 0).
2. It sets the resolution to 256x384 using the `set()` method of the VideoCapture object ¹ ³.
3. It verifies the settings by getting the current width and height.
4. It then enters a loop to capture and display video frames.
5. The loop continues until the user presses 'q' to quit.

Note that while we attempt to set the resolution, some cameras may not support changing resolutions or may round to the nearest supported value. Always check the actual resolution after setting it to ensure it was applied correctly ¹ ³.

If you encounter any issues, you may need to check your camera's supported formats and resolutions using v4l2-ctl commands and adjust accordingly ⁴.

 Share  Rewrite





OpenCV & Python

Watch

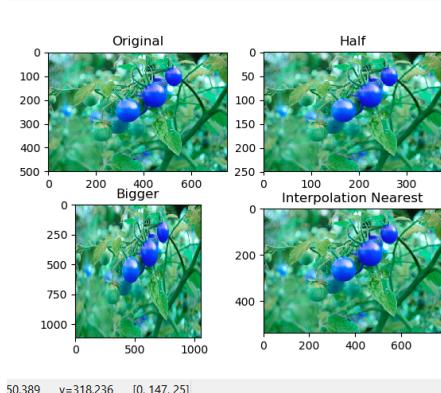
2



OpenCV & Python Scale & Res



Figure 1



50.389 y=318.236 [0, 147, 25]

```
python 3.7.1 Shell
Edit Shell Debug Options Window Help
python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 201
] on win32
'pe "help", "copyright", "credits" or "lic
>=====
RESTART: C:\Users\Saurabh\work\geek
'CAP_PROP_FRAME_WIDTH: '640.0'
'CAP_PROP_FRAME_HEIGHT : '480.0'
'AP_PROP_FPS : '30.0'
'AP_PROP_POS_MSEC : '0.0'
'AP_PROP_FRAME_COUNT : '-1.0'
'AP_PROP_BRIGHTNESS : '50.0'
'AP_PROP_CONTRAST : '50.0'
'AP_PROP_SATURATION : '50.0'
'AP_PROP_HUE : '50.0'
'AP_PROP_GAIN : '-1.0'
'AP_PROP_CONVERT_RGB : '1.0'
>|
```

View More

Camera Resolution



If I had ctrl-c'd out of that from the terminal instead of pressing 'q' in the window, would that cause a driver busy problem?

❖ Sources

[Signal handling: Catch Ctrl-C in Python - Code Maven](#)

[Control your Webcam with Python - OpenCV Beginners Tutorial #4](#)

[Contour Detection using OpenCV](#)



❖ Answer

Pressing Ctrl-C in the terminal to exit the Python script using OpenCV for webcam capture could potentially cause a driver busy problem, but it's not guaranteed. Here's why:

1. Signal Handling: When you press Ctrl-C in the terminal, it sends a SIGINT (interrupt signal) to the running process. By default, this signal terminates the program ¹.
2. Resource Release: Normally, when a Python script exits, it automatically releases system resources, including the webcam. However, abrupt termination might not always allow for proper cleanup.
3. OpenCV Behavior: OpenCV's `VideoCapture` object typically releases the camera when it's destroyed, which usually happens when the script ends. However, if the termination is too abrupt, this cleanup might not occur properly.
4. Driver State: If the camera driver doesn't receive the proper release command, it might remain in a "busy" state, thinking it's still in use.

To mitigate this issue, you can implement a signal handler in your Python script:

```
python
import signal
import cv2

def signal_handler(signum, frame):
    print("\nCtrl+C pressed. Releasing camera and exiting...")
    cap.release()
    cv2.destroyAllWindows()
    exit(0)

signal.signal(signal.SIGINT, signal_handler)

cap = cv2.VideoCapture(0)
```

```
# Rest of your video capture code here
```

This handler ensures that even if you press Ctrl-C, the script will properly release the camera before exiting ¹. This approach reduces the likelihood of leaving the camera driver in a busy state.

If you do encounter a busy driver problem after Ctrl-C, you may need to:

1. Unplug and replug the USB camera.
2. Restart the udev service: `sudo systemctl restart udev`
3. Reload the USB video driver: `sudo modprobe -r uvcvideo && sudo modprobe uvcvideo`

These steps can help reset the camera driver state if it becomes stuck.

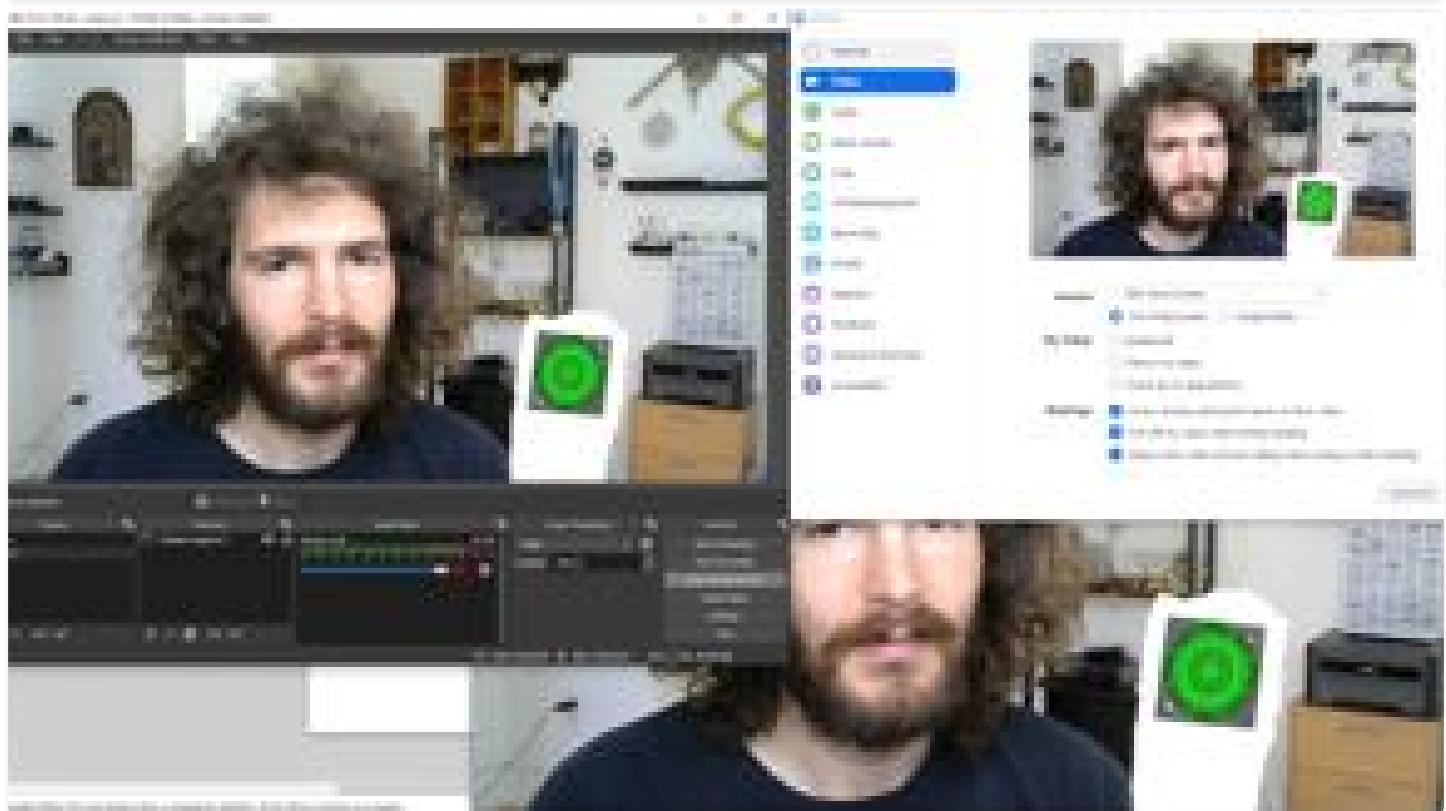
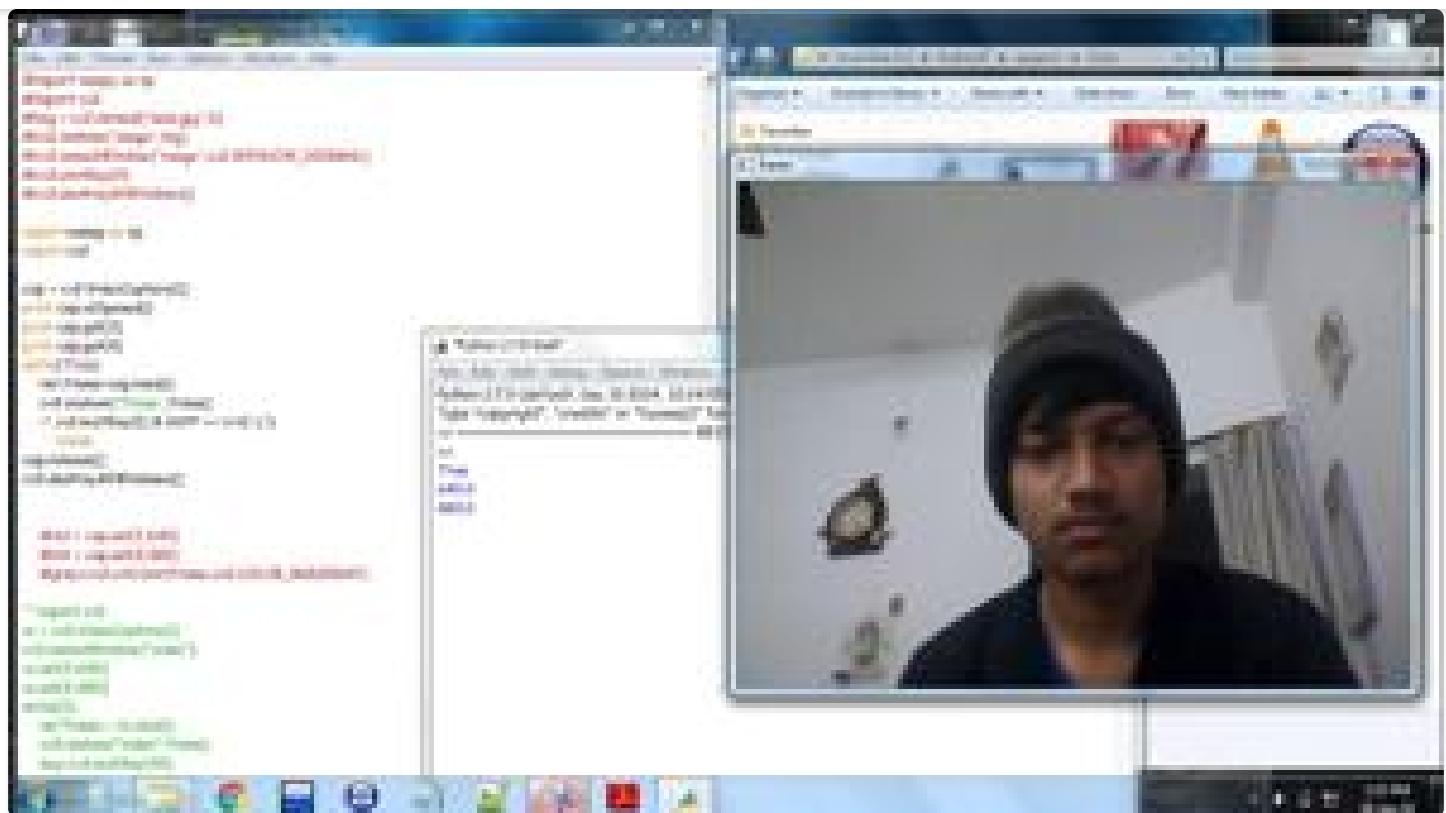
[Share](#) [Rewrite](#)

The screenshot shows a Jupyter Notebook cell with the following Python code:

```
1 #!/usr/bin/python
2 import cv2
3
4 img = cv2.imread('cat.jpg')
5 print(img.shape)
6 print(img[0, 0])
7 img = img * 2
8 cv2.imshow("Cat", img)
9 cv2.waitKey(0)
```

The code imports cv2, reads a 'cat.jpg' image, prints its shape and a specific pixel value, scales the image by 2, displays it with cv2.imshow, and waits for a key press. The Jupyter interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, and a 'VIDEOS AND WEB CAMS' section at the bottom.

Watch



```
Help      :<--> manpy - 2 - Visual Studio Code
ipy -->
ipy> import cv2

def mouse_click(event, x, y, flags, param):
    # to check if left mouse button was clicked
    if event == cv2.EVENT_LBUTTONDOWN:
        print("left click")
        cv2.imwrite("frame.jpg", param)

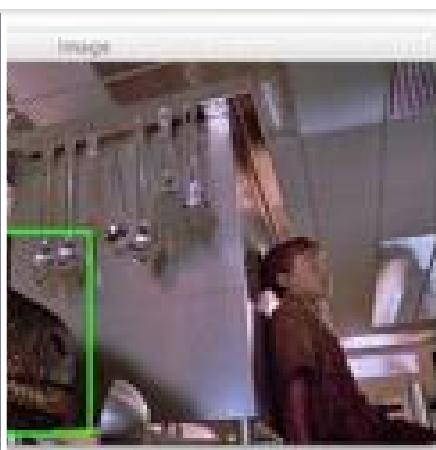
    # to check if right mouse button was clicked
    if event == cv2.EVENT_RBUTTONDOWN:
        print("right click")
        cv2.imshow("Current Frame", frame)

cap = cv2.VideoCapture("test.mp4")

if cap.isOpened() == False:
    # give error message
    print("Error in opening file.")
else:
    # proceed forward
    while(cap.isOpened()):
        ret, frame = cap.read()
        if ret == True:
            cv2.imshow("Frame", frame)
            cv2.setMouseCallback('Frame', mouse_click, param=frame)
            if cv2.waitKey(25) & 0xFF == ord('q'):
                break
        else:
            break

cap.release()
cv2.destroyAllWindows()


```



 View More

Is there any way for a script to pre-clear the driver busy condition without the separate signal handler?

Sources

How to check if webcam is busy edit - OpenCV Q&A Forum

Cannot turn off/on CameraCapture using Python/opencv: Device or

Camera remains busy after b



 Answer

While there isn't a guaranteed way to pre-clear the driver busy condition without a separate signal handler, there are a few approaches you can try to mitigate the issue:

- #### 1. Implement a retry mechanism:

```
python

import cv2
import time

def open_camera(max_attempts=5, delay=1):
    for attempt in range(max_attempts):
        cap = cv2.VideoCapture(0)
        if cap.isOpened():
            return cap
        cap.release()
        time.sleep(delay)
    return None
```



```
print("Failed to open camera after multiple attempts")
```

2. Use a context manager:

```
python   
import cv2  
from contextlib import contextmanager  
  
@contextmanager  
def video_capture(*args, **kwargs):  
    cap = cv2.VideoCapture(*args, **kwargs)  
    try:  
        yield cap  
    finally:  
        cap.release()  
  
with video_capture(0) as cap:  
    # Use the camera  
    ret, frame = cap.read()
```

3. Manually release resources:

```
python   
import cv2  
  
cap = cv2.VideoCapture(0)  
# Use the camera  
ret, frame = cap.read()  
  
# Explicitly release resources  
cap.release()  
cv2.destroyAllWindows()
```

These methods can help manage camera resources more effectively, potentially reducing the likelihood of a busy driver condition. However, they don't guarantee prevention of all possible issues, especially if the camera is being accessed by another process ¹ ².

It's important to note that the most reliable way to handle abrupt terminations and ensure proper resource cleanup is still to use a signal handler. If you're concerned about driver busy conditions, combining one of these approaches with a signal handler would provide the most robust solution ³.

 Share  Rewrite



What was the line again to use the '/dev/video0' device numbers with cv2.videocapture? Now that ffmpeg is installed where it wasn't before, I should try that again.

Sources

[How to run OpenCV VideoCapture without root privilege on linux](#)

[Create openCV VideoCapture from interface name instead of ...](#)

[How to specify exact camera ...](#)

Answer

To use the '/dev/video0' device with cv2.VideoCapture, you can use the following line:

```
python
cap = cv2.VideoCapture('/dev/video0')
```

Now that ffmpeg is installed, this method should work better than before. The '/dev/video0' typically represents the first video device on your system ² ⁴.

If you encounter any issues, you can also try using the device number directly:

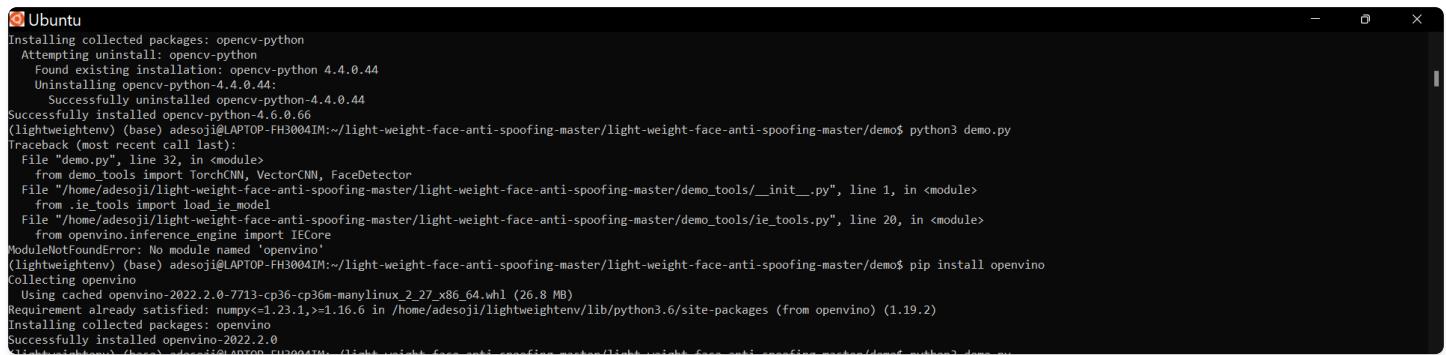
```
python
cap = cv2.VideoCapture(0)
```

Where 0 corresponds to the first video device (/dev/video0) ².

If you're still having trouble, you might want to check the device permissions and ensure your user has the necessary access rights to the video device ¹.

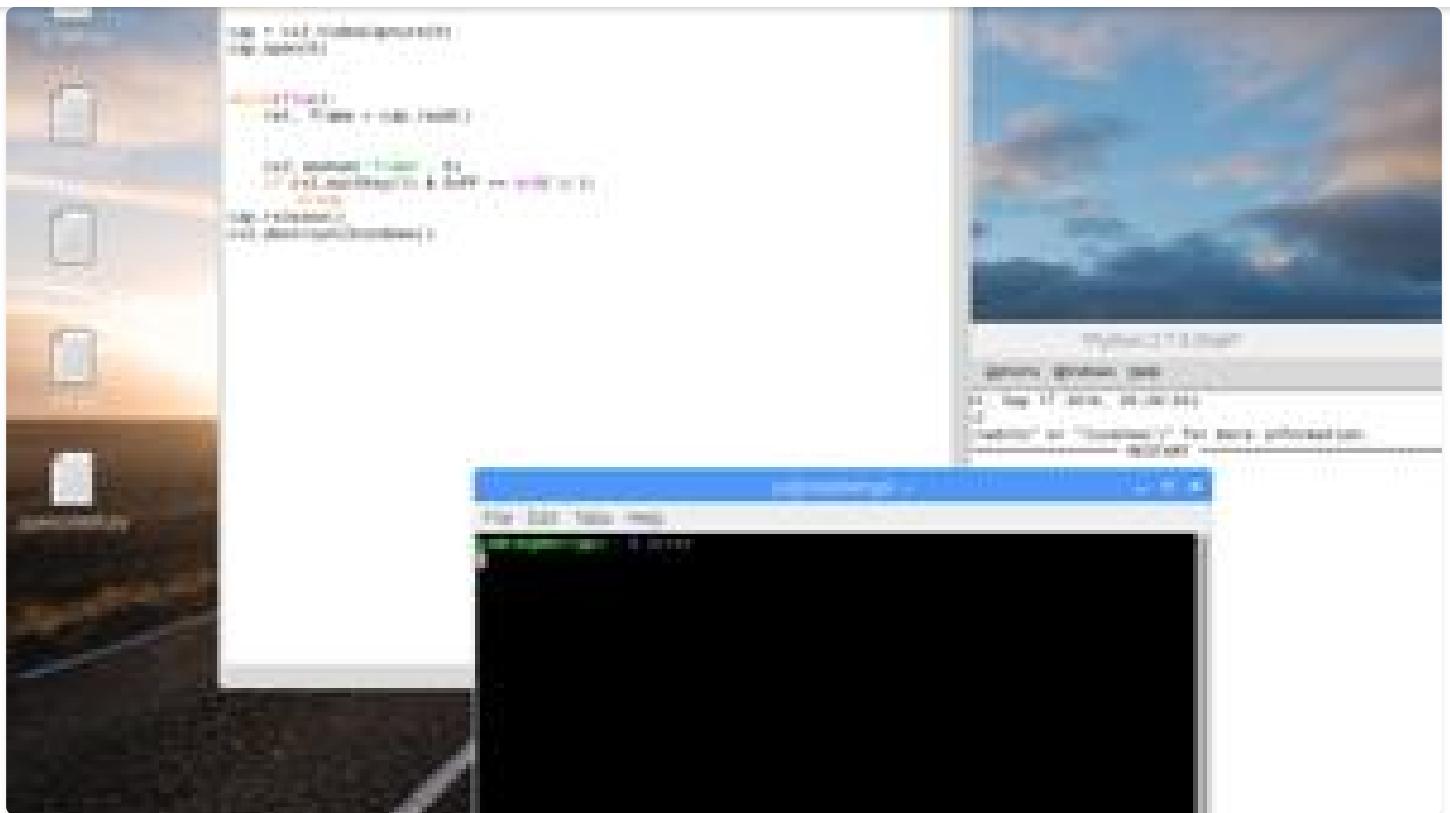
Share Rewrite

Like Comment Save ...



The screenshot shows a terminal window titled "Ubuntu" running on an Ubuntu system. The terminal output is as follows:

```
Installing collected packages: opencv-python
Attempting uninstall: opencv-python
  Found existing installation: opencv-python 4.4.0.44
  Uninstalling opencv-python-4.4.0.44:
    Successfully uninstalled opencv-python-4.4.0.44
Successfully installed opencv-python-4.6.0.66
(lighweightenv) (base) adesoji@LAPTOP-FH30041M:~/light-weight-face-anti-spoofing-master/light-weight-face-anti-spoofing-master/demo$ python3 demo.py
Traceback (most recent call last):
  File "demo.py", line 32, in <module>
    from demo_tools import TorchCNN, VectorCNN, FaceDetector
  File "/home/adesoji/light-weight-face-anti-spoofing-master/light-weight-face-anti-spoofing-master/demo_tools/_init_.py", line 1, in <module>
    from .ie_tools import load_ie_model
  File "/home/adesoji/light-weight-face-anti-spoofing-master/light-weight-face-anti-spoofing-master/demo_tools/ie_tools.py", line 20, in <module>
    from openvino.inference_engine import IECore
ModuleNotFoundError: No module named 'openvino'
(lighweightenv) (base) adesoji@LAPTOP-FH30041M:~/light-weight-face-anti-spoofing-master/light-weight-face-anti-spoofing-master/demo$ pip install openvino
Collecting openvino
  Using cached openvino-2022.2.0-7713-cp36-manylinux_2_27_x86_64.whl (26.8 MB)
Requirement already satisfied: numpy<=1.23.1,>=1.16.6 in /home/adesoji/lighweightenv/lib/python3.6/site-packages (from openvino) (1.19.2)
Installing collected packages: openvino
Successfully installed openvino-2022.2.0
(lighweightenv) (base) adesoji@LAPTOP-FH30041M:~/light-weight-face-anti-spoofing-master/light-weight-face-anti-spoofing-master/demo$
```



root@ubuntu:~\$

```
: unable to open display ":0"
rm --runtime nvidia --network host ncr.io/nvidia/l4t-pytorch:r35.1.6-pth1.12-py3
Unknown runtime specified nvidia.

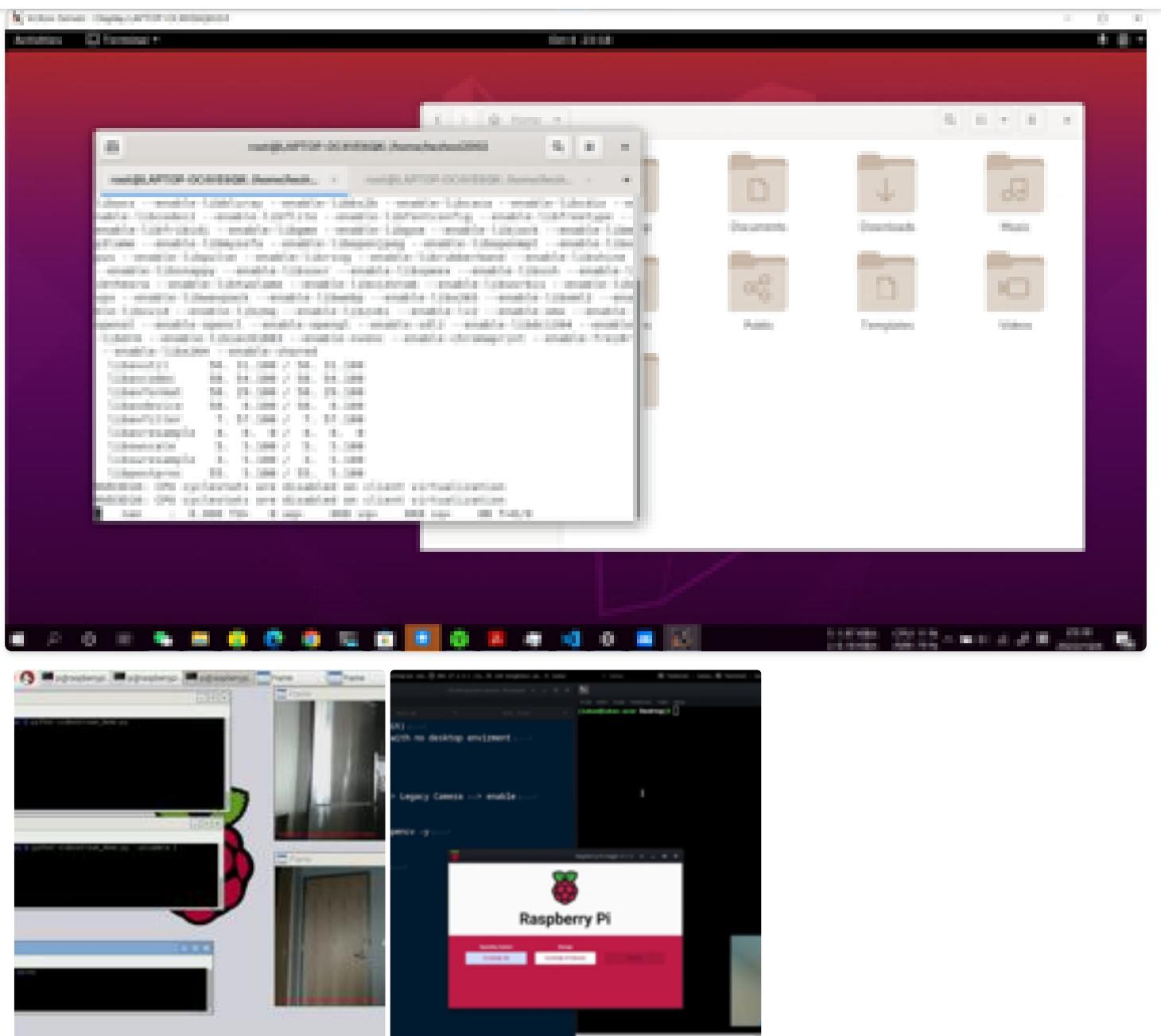
t docker.service
rm --runtime nvidia --network host ncr.io/nvidia/l4t-pytorch:r35.1.6-pth1.12-py3

, 20:18:18)
or "license" for more information.

int8)

nnect: Connection refused

e>
v/modules/highgui/src/window_gtk.cpp:624: error: (-2:Unspecified error) Can't initialize GTK backend tr
```



[View More](#)

I plugged in the second camera again and I'm seeing everything from:

v4l2-ctl --list-devices

USB Camera: USB Camera (usb-0000:00:14.0-5):

/dev/video2

/dev/video3

USB Camera: USB Camera (usb-0000:00:14.0-6):

```
/dev/video0  
/dev/video1  
/dev/media0
```

But this line doesn't change the device pulled from, iterating from 0 to -6:

```
cap = cv2.VideoCapture(-6)
```

❖ Sources

How do I choose which camera I am accessing? edit

 answers.opencv

V4L2 with opencv in python - Jetson TX2 - NVIDIA Developer Forums

 forums.developer.nvidia

How to specify exact camera
...

 forum.opencv

✳ Answer

Based on the output from v4l2-ctl --list-devices, you have two USB cameras connected to your system. To use a specific camera with OpenCV, you should use the device file path directly instead of numerical indices. Here's how you can access each camera:

For the first USB Camera:

```
python  
cap = cv2.VideoCapture('/dev/video0')
```

or

```
python  
cap = cv2.VideoCapture('/dev/video1')
```

For the second USB Camera:

```
python  
cap = cv2.VideoCapture('/dev/video2')
```

or

```
python  
cap = cv2.VideoCapture('/dev/video3')
```

Using negative indices like -6 with cv2.VideoCapture() is not a standard method and may not work as expected ¹. Instead, stick to using the direct device file paths.

If you're still having issues, try the following:

1. Ensure you have the necessary permissions to access the video devices.
2. Try setting the pixel format to MJPG, which can sometimes resolve capture issues:

```
python  
cap = cv2.VideoCapture('/dev/video0')  
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'))
```

3. Check if the camera is opened successfully:

```
python
cap = cv2.VideoCapture('/dev/video0')
if not cap.isOpened():
    print("Error: Could not open camera.")
    exit()
```

4. If problems persist, try using v4l2capture library as an alternative to OpenCV for camera access ². Remember to release the camera when you're done:

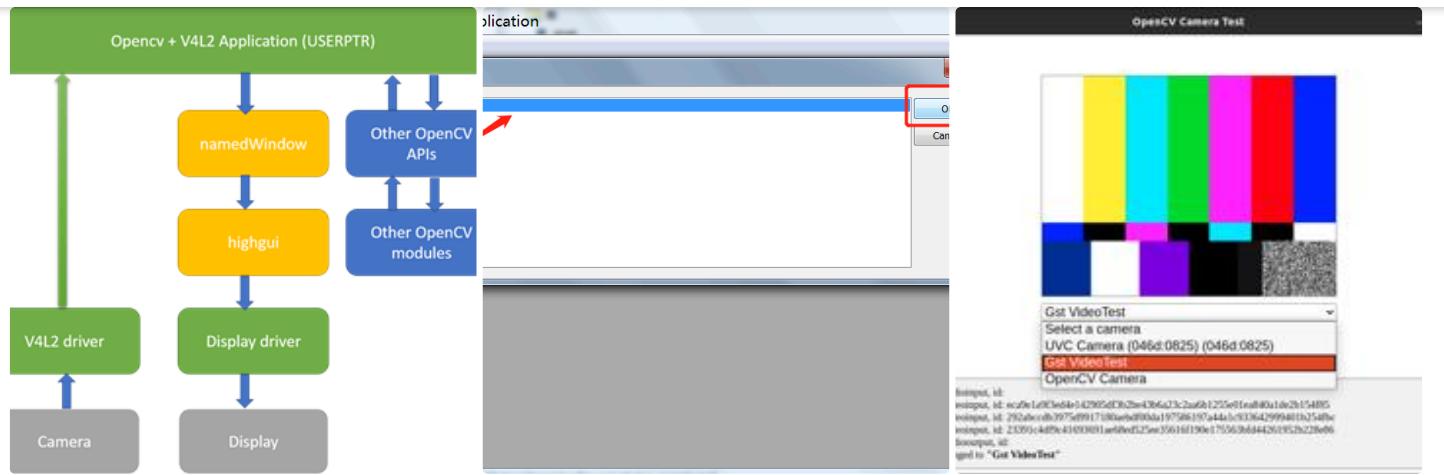
```
python
cap.release()
```

Share Rewrite



Resolution	4,208 × 3,120 pixels	HFOV	72.9°
Aspect Ratio	1.34:1	Resolution	640 × 480 pixels
Focus Range	8 cm → ∞	Aspect Ratio	1.33:1
Lens Size	1/3.1 inch	Focus Range	6.5 cm → ∞ (AT=11 cm)
FL	3.37 mm	Lens Size	1/2.8 inch
F-number	2.2 ± 5%	FL	1.8 mm
Distortion	<1.0%	F-number	2.2
		Distortion	<1.0%





[View More](#)

Could you give me an argparse setup for taking in one integer and then parsing that out to that line for the last character in /dev/videoX?

• Sources

[Convert to integer - Code Maven](#)

[Get the Last Character of a String in Python - Flexiple](#)

[CheckRange class for the Py](#)

slides

flexiple

gist.github

Answer

Certainly! Here's an argparse setup that takes in one integer and uses it to construct the device path for the camera:

```
python
import argparse
import cv2

parser = argparse.ArgumentParser(description='Capture video from a specific camera device.')
parser.add_argument('device_number', type=int, help='The camera device number (e.g., 0 for /dev/video0)')

args = parser.parse_args()

device_path = f'/dev/video{args.device_number}'

cap = cv2.VideoCapture(device_path)

if not cap.isOpened():
```

```
# Your video capture code here
# For example:
ret, frame = cap.read()
if ret:
    cv2.imshow('Frame', frame)
    cv2.waitKey(0)

cap.release()
cv2.destroyAllWindows()
```

This script does the following:

1. It sets up an ArgumentParser to accept one integer argument, `device_number`.
2. It uses the parsed integer to construct the device path string (`/dev/videoX`).
3. It then uses this device path to open the camera using OpenCV.

You can run this script from the command line like this:

```
text
python script_name.py 0
```

This would open `/dev/video0`. To use a different camera, just change the number:

```
text
python script_name.py 2
```

This would open `/dev/video2`¹ .

[Share](#) [Rewrite](#)



```
C:\Users\Uanshi\Desktop>python prog.py -h
usage: prog.py [-h] N [N ...]

sort some integers.

positional arguments:
  N                  an integer for the accumulator
  sum
```

```
import argparse

# Define and parse command-line arguments
parser = argparse.ArgumentParser(
    prog = 'Model training',
    description = 'Train a random forest classifier on the iris dataset'
)
# Add positional argument
parser.add_argument(
    'train_data', help='training data (.csv format)'
)
# Add optional argument
parser.add_argument(
    '--n_estimators', type=int, default=100,
    help='number of trees in the forest (default: 100)'
)
# Parse arguments
args = parser.parse_args()
```

```
Set up argument parser
parser = argparse.ArgumentParser(description="Command-line calculator.")
subparsers = parser.add_subparsers()

Add subcommands
add_parser = subparsers.add_parser("add", help="Add two numbers.")
add_parser.add_argument('x', type=float, help="First number.")
add_parser.add_argument('y', type=float, help="Second number.")
add_parser.set_defaults(func=add)

subtract_parser = subparsers.add_parser("subtract", help="Subtract two numbers.")
subtract_parser.add_argument('x', type=float, help="First number.")
subtract_parser.add_argument('y', type=float, help="Second number.")
subtract_parser.set_defaults(func=subtract)

multiply_parser = subparsers.add_parser("multiply", help="Multiply two numbers.")
multiply_parser.add_argument('x', type=float, help="First number.")
multiply_parser.add_argument('y', type=float, help="Second number.")
multiply_parser.set_defaults(func=multiply)
```

```
[hyunyoubg2-desktop:~] in ~ [14:59:37]
[JSON]... [FILE]...
[JSON]... --files=FILE
, word, and byte counts for each FILE, and a total
FILE is specified. A word is a non-zero-length se
limited by white space.

or when FILE is -, read standard input.

--allow may be used to select which counts are printed
order: newline, word, character, byte, maximum lin
    print the byte counts
    print the character counts
    print the newline counts
--from=F   read input from the files specified by
        NUL-terminated names in file F;
        If F is - then read names from standa
line-length  print the maximum display width
    print the word counts
    display this help and exit
--version  output version information and exit

online help: <http://www.gnu.org/software/coreutil
ation at: <http://www.gnu.org/software/coreutils/wc
locally via: info '(coreutils) wc invocation'
```

```
c:\Windows\system32\cmd.exe
c:\Windows\system32>cd \Desktop\internship\geeks\argparse>odd_even.py 2 4
c:\Windows\system32>cd \Desktop\internship\geeks\argparse>odd_even.py
argumentError(ref_arg1, "Argument 1 Can't be even n
or: argument odd: Argument 1 Can't be even number!
c:\Windows\system32>cd \Desktop\internship\geeks\argparse>odd_even.py 3 6
c:\Windows\system32>cd \Desktop\internship\geeks\argparse>
```

View More

That's weird, everything else works, but I get an error on one of the listed devices:

Error: Could not open camera at /dev/video1

❖ Sources

[OpenCV2 VideoCapture issues - Jetson Nano](#)

[OpenCv error can't open camera through video capture](#)

[Unable to take pictures with i](#)

forums.developer.nvidia

stackoverflow

forum.opencv

✳ Answer

The error you're encountering with /dev/video1 while other devices work is not uncommon and can occur due to various reasons. Here are some potential causes and solutions:

1. Device Availability: The /dev/video1 device might not be available or accessible at the moment. This can happen if the device was recently unplugged or if another process is using it.
2. Permission Issues: Ensure you have the correct permissions to access /dev/video1. You can check this by running:

```
text
ls -l /dev/video1
```



If you don't have the necessary permissions, you may need to add your user to the 'video' group ².

3. Driver Support: Some cameras may not support memory mapping, which can cause issues with OpenCV. You might see an error message like "/dev/video1 does not support memory mapping" ³.
4. Multiple Initializations: Ensure you're not creating multiple instances of VideoCapture for the same device elsewhere in your code.

5. Device Indexing: Sometimes, the device indexing in OpenCV doesn't match the system's device numbering. Try using a different index or the full device path ¹.

To troubleshoot:

1. Try opening the camera with a different index, like cv2.VideoCapture(0) or cv2.VideoCapture(-1) ².
2. Use the full device path: cv2.VideoCapture("/dev/video1") ¹.
3. Check if the camera works with other applications or tools like v4l2-ctl.
4. Ensure the camera is properly connected and recognized by your system.
5. If the issue persists, try rebooting your system or reconnecting the camera ².

If none of these solutions work, the problem might be related to your specific camera model or system configuration, and you may need to investigate further or try alternative capture methods.

Share Rewrite



The screenshot shows a browser window with two tabs. The top tab is a Jupyter Notebook cell containing Python code for object detection. The bottom tab is a Streamlit app titled "test2" which has encountered an error. The Streamlit error message is redacted, but the full error details are visible in the browser's developer console.

```
[1]: #https://towardsdatascience.com/how-to-detect-objects-in-real-time-using-opencv-and-python-c1ba0c2c69c0
[5]: pip install opencv-python
Requirement already satisfied: opencv-python in ./local/lib/python3.8/site-packages (4.5.4.60)
Requirement already satisfied: numpy>=1.17.3 in ./local/lib/python3.8/site-packages (from opencv-python) (1.21.2)
Note: you may need to restart the kernel to use updated packages.

[7]: import cv2
[14]: #Since we want to detect the objects in real-time, we will be using the webcam feed. Use the below code to initiate the webcam.
# Enable we
# at the default to run without webcam
```

2024-02-26 02:30:08.990 uncaught app exception

```
Traceback (most recent call last):
  File "/home/adminuser/venv/lib/python3.9/site-packages/streamlit/runtime/script_exec(code, module.__dict__)
  File "/mount/src/mlproject/cvstream.py", line 20, in <module>
    cv2.destroyAllWindows()
cv2.error: OpenCV(4.9.0) /io/opencv/modules/highgui/src/window.cpp:1266: error: ([ WARN:1095.947] global cap_v4l.cpp:997 open VIDEOIO(V4L2:/dev/video0): can't open [ERROR:1095.948] global obsensor_uvc_stream_channel.cpp:159 getStreamChannelGroup
2024-02-26 02:36:56.077 Uncaught app exception
```

```
Traceback (most recent call last):
  File "/home/adminuser/venv/lib/python3.9/site-packages/streamlit/runtime/script_exec(code, module.__dict__)
  File "/mount/src/mlproject/cvstream.py", line 20, in <module>
    cv2.destroyAllWindows()
cv2.error: OpenCV(4.9.0) /io/opencv/modules/highgui/src/window.cpp:1266: error: ([ WARN:2096.403] global cap_v4l.cpp:997 open VIDEOIO(V4L2:/dev/video0): can't open [ERROR:2096.403] global obsensor_uvc_stream_channel.cpp:159 getStreamChannelGroup
2024-02-26 02:36:56.533 Uncaught app exception
```

```
Traceback (most recent call last):
  File "/home/adminuser/venv/lib/python3.9/site-packages/streamlit/runtime/script_exec(code, module.__dict__)
  File "/mount/src/mlproject/cvstream.py", line 20, in <module>
    cv2.destroyAllWindows()
cv2.error: OpenCV(4.9.0) /io/opencv/modules/highgui/src/window.cpp:1266: error: ([ WARN:30284.448] global cap_v4l.cpp:997 open VIDEOIO(V4L2:/dev/video0): can't open [ERROR:30284.448] global obsensor_uvc_stream_channel.cpp:159 getStreamChannelGroup
2024-02-26 02:40:04.577 Uncaught app exception
```

```
Traceback (most recent call last):
  File "/home/adminuser/venv/lib/python3.9/site-packages/streamlit/runtime/script_exec(code, module.__dict__)
  File "/mount/src/mlproject/cvstream.py", line 20, in <module>
    cv2.destroyAllWindows()
cv2.error: OpenCV(4.9.0) /io/opencv/modules/highgui/src/window.cpp:1266: error: ([
```

main affanmohammed/mlproject/main/cvstream.py

```
root@tegra-ubuntu:~# v4lctl --list-devices
VIDIOC_QUERYCAP: failed: Inappropriate ioctl for device
VIDIOC_QUERYCAP: failed: Inappropriate ioctl for device
vi-output, ov5693 2-0036 (platform:15700000.vi12):
    /dev/video8
```



```
root@tegra-ubuntu:~# v4lctl --list-devices
VIDIOC_QUERYCAP: failed: Inappropriate ioctl for device
VIDIOC_QUERYCAP: failed: Inappropriate ioctl for device
vi-output, ov5693 2-0036 (platform:15700000.vi12):
    /dev/video8

USB Camera 04.09.24.1 (usb-3530000.usb1-2.1):
    /dev/video1
    /dev/v4l-subdev1
    /dev/v4l-subdev0

root@tegra-ubuntu:~# cheese -d /dev/v4l-subdev1
Segmentation fault (core dumped)
root@tegra-ubuntu:~# cheese -d /dev/v4l-subdev0
Segmentation fault (core dumped)
root@tegra-ubuntu:~# cheese -d /dev/video1
Segmentation fault (core dumped)
root@tegra-ubuntu:~#
```

Videocapture file reading from using backend...

posted on June 22, 2018

```
v
capture[2]
Opened():
    nnot open camera")

me
= cap.read()

Can't receive frame (stream end?). Exiting

ame
capture", frame)
ey(1) & 0xFF == ord('q'):

ndows()
```

View More

I gave this:

ls -l /dev/video1

crw-rw----+ 1 root video 81,1 Dec 23 21:29 /dev/video1

• Sources

[dev/video permissions Arch Linux \(Remmina using FreeRDP\)](#)

[Access webcam without sudo edit - OpenCV Q&A Forum](#)

[22.04 Webcam permissions](#)

unix.stackexchange

answers.opencv

askubuntu

Answer

The output you've provided shows the permissions and ownership of the /dev/video1 device file. Let's break it down:

- The first character 'c' indicates that this is a character device file.
- The permissions are set to 'rw-rw----', which means:
 - The owner (root) has read and write permissions
 - The group (video) has read and write permissions
 - Others have no permissions
- The '+' at the end indicates there are additional access control lists (ACLs) applied to this file.
- The file is owned by the user 'root' and belongs to the group 'video'.
- The device numbers are 81 and 1.
- The file was last modified on December 23 at 21:29.

To access this device without using sudo, you can try the following:

1. Add your user to the 'video' group:

text

```
sudo usermod -a -G video $USER
```



You'll need to log out and back in for this change to take effect.

2. If adding yourself to the video group doesn't work, you can temporarily change the permissions to allow all users to read and write:

text

```
sudo chmod 666 /dev/video1
```



However, this is less secure and the changes may not persist after a reboot.

3. Check for any ACLs using:

text

```
getfacl /dev/video1
```

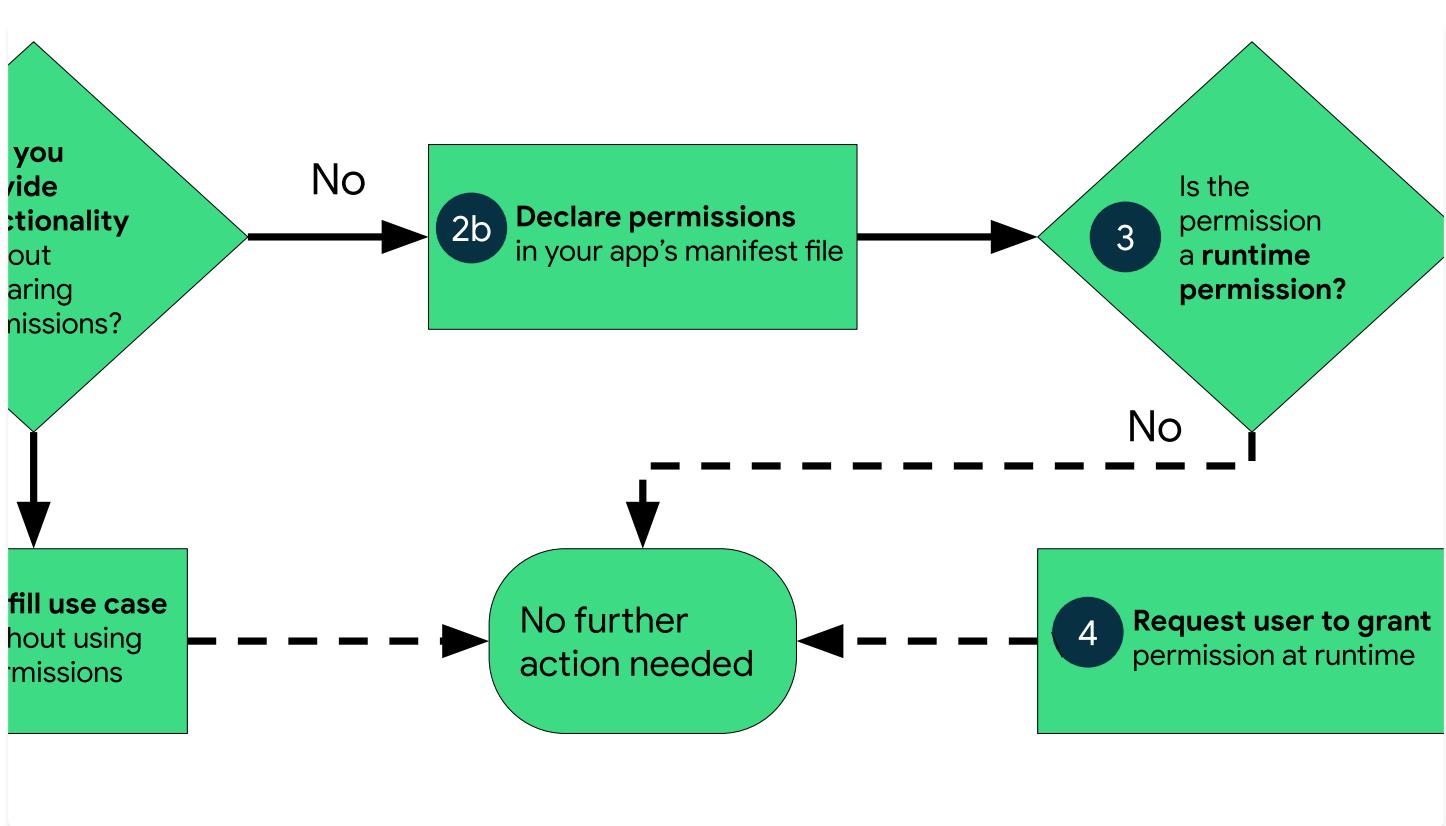


This might reveal additional permissions or restrictions.

If you're still unable to access the device after these steps, you may need to investigate further into your system's udev rules or any security policies that might be restricting access [1](#) [3](#).

[Share](#) [Rewrite](#)

[Like](#) [Comment](#) [Save](#) [...](#)



```
Welcome to Ubuntu 20.04.4 LTS  
Monday, 2 May 2022, 06:51:41 AM  
Up time: 0 days, 00:08:15  
Free memory: 6204676 / 7644824 kB  
IP: 192.168.1.96
```

1 device has a firmware upgrade available. **account for this issue.**
Run `fwupdmgr get-upgrades` for more information.

0 updates can be applied immediately.

Typical kids will severely suffer for this, and while solving this kinds of issue, it will promote anger toward Hardkernel.

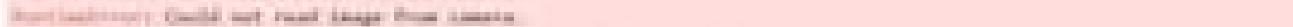
```
[!] Please visit 'https://launchpad.net/odroid-image' to report a bug  
Last login: Mon May 2 06:45:31 2022 from 192.168.1.53  
/usr/bin/xauth: file /home/odroid/.Xauthority does not exist  
odroid@gnome-desktop:~$ ls /dev/video*  
/dev/video0 /dev/video1  
odroid@gnome-desktop:~$ ls /dev/video* -asl  
0 crw-rw----+ 1 root video 81, 0 May 2 06:43 /dev/video0  
0 crw-rw----+ 1 root video 81, 1 May 2 06:43 /dev/video1
```

Create the camera object

First, create a camera object by importing the `CV2Camera` class from the library by executing the following Python code-cell. Please note that `camera` instance.

```
#!/usr/bin/python3  
# Import the required libraries  
from jetcam.usb_camera import CV2Camera
```

```
camera = CV2Camera(resolution=(640, 480), brightness=50, exposure_mode='auto') # capture the capture from camera
```

Result: 

Result: 

```
camera = CV2Camera(resolution=(640, 480), brightness=50, exposure_mode='auto') # capture the capture from camera
```

```
Result: 
```

```
camera = CV2Camera(resolution=(640, 480), brightness=50, exposure_mode='auto') # capture the capture from camera
```

```
Result: 
```

```
-s ls  
i_run.sh examples.desktop  
-s echo "sudo docker run  
k ha  
vdli-data:/nvdli-nano/di
```



Meaning for Folders	Meaning for Files
mits viewing and listing of files and subfolders	Permits viewing or a file's contents
mits adding of files and subfolders	Permits writing to a
mits viewing and listing of files and subfolders as well as executing of files; inherited by files and	Permits viewing and file's contents as well