Jetson Nano Peripheral Testing

1. Camera Testing

The Jetson Nano supports both USB cameras and MIPI-CSI cameras.

1.1. Test USB Camera

Prerequisites:

- A functional USB camera.
- The v4l-utils package installed on your Jetson Nano.
  - sudo apt update
  - sudo apt install v4l-utils

Steps:

1. **Connect the USB Camera:**
2. **List USB Devices to Verify Connection:**
   a. Open a terminal and run: lsusb
3. **Identify Camera Device Node:**
   Run the following command to list all video devices: ls /dev/video*

Output -

1.2. Test MIPI-CSI Camera (IMX219)

Purpose**:** To verify the functionality of a MIPI-CSI camera, commonly the Raspberry Pi Camera Module V2 (IMX219 sensor), which has default driver support on Jetson Nano.

Prerequisites**:**

- A MIPI-CSI camera (ex – IMX219)

- Dependency Check: The Jetson Nano's JetPack SDK includes the necessary drivers for the IMX219 sensor by default. No manual driver installation is typically required for this specific sensor.

Steps:

1. **Verify Camera Device Node:**
   a. Run: ls /dev/video*
2. **Test Live Video Stream with nvgstcapture-1.0:**
   This command defaults to the CSI camera if available:
   nvgstcapture-1.0

2. Display Output Testing

on Nano supports HDMI output. DisplayPort is generally not supported on the Jetson Nano Developer Kit.

1. Test HDMI Output

Purpose**:** To confirm that the Jetson Nano can successfully output video to an HDMI display.

Prerequisites:

- An HDMI cable.
- An HDMI-compatible monitor or TV.

Steps:

1. **Connect HDMI Cable:** Connect one end of the HDMI cable to the Jetson Nano HDMI port and the other end to your display.
2. **Power On/Reboot Jetson Nano:** If the Jetson Nano is already on
3. **Verify Display Output**

2. DisplayPort Display Check

Important Note: The **NVIDIA Jetson Nano Developer Kit does NOT support DisplayPort output.**

It only has an HDMI port for video output. If your display only has DisplayPort, you will need an active HDMI-to-DisplayPort adapter

3. USB Device Testing

This section covers testing general USB 2.0 peripherals and assessing USB drive speeds.

1. Test USB 2.0 Devices (Keyboard, Mouse)

Purpose**:** To ensure basic USB peripheral functionality.

Prerequisites**:**

- A USB keyboard.
- A USB mouse.

Steps:

1. **Connect USB Keyboard**
2. **Connect USB Mouse**
    a. **Expected Outcome:** The mouse cursor should appear on the

2. USB Drive Speed Test

Purpose: To measure the read and write speeds of a USB drive connected to the Jetson Nano's USB 3.0 ports.

While the Nano has USB 3.0 ports, the performance will also depend on the USB drive itself and whether it's a USB 2.0 or USB 3.0 device.

1. **Using lsusb -t (tree format, shows speed)**

    : lsusb -t

Output :

/: Bus 02. Port 1: Dev 1, Class=root_hub, Driver=tegra-xusb/4p, 5000M
|__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 5000M

/: Bus 01. Port 1: Dev 1, Class=root_hub, Driver= egra-xusb/5p, 480M

|__ Port 3: Dev 4, If 0, Class= Hub, Driver= hub/5p, 480M

- 5000M = **USB 3.0**
- 480M = **USB 2.0**

4. Storage Performance Testing

This section focuses on the performance of the microSD card, which is the primary storage for the Jetson Nano Developer Kit.

1. Test MicroSD Card Performance

Purpose: To assess the read and write speeds of the microSD card, which directly impacts system responsiveness and application loading times.

Prerequisites:

- A microSD card inserted into the Jetson Nano.

Steps:

1. **Identify MicroSD Card Device:**
    a. The root filesystem is typically on /dev/mmcblk0p1 (for microSD). You can confirm with lsblk.

5) GPIO Header Interfaces (40 pin)

1) GPIO - Blink LED Test

GPIO stands for General Purpose Input/Output
It is a flexible interface on microcontrollers and development boards (like Jetson Nano)
Use of controlling or reading digital signals from external hardware.

Hardware:

- 1x LED
- 1x 330Ω resistor (Storing data temporarily during calculations or data transfers).
- Connect **GPIO pin 12 (physical pin 32)** to the LED + resistor
- jumper wires (Male-to-Female or Male-to-Male) 2-3
- breadboard

Wiring:

- Jetson pin 32 →LED anode (-)
- LED cathode (+) → GND - pin 6

Test Script:

```
import Jetson.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)  # or GPIO.BCM if using GPIO
numbers directly

led_pin = 12  # Physical Pin 12 = GPIO18

GPIO.setup(led_pin, GPIO.OUT)

print("Blinking LED... Press Ctrl+C to stop.")
```

```
try:
    while True:
        GPIO.output(led_pin, GPIO.HIGH)  # LED ON
        time.sleep(1)
        GPIO.output(led_pin, GPIO.LOW)   # LED OFF
        time.sleep(1)
except KeyboardInterrupt:
    print("Stopping...")
finally:
    GPIO.cleanup()  # Reset GPIO state
```

- **Running the test**

1. **Install the GPIO library** (run once):

```
sudo apt update
sudo apt install python3-pip
sudo pip3 install Jetson.GPIO
```

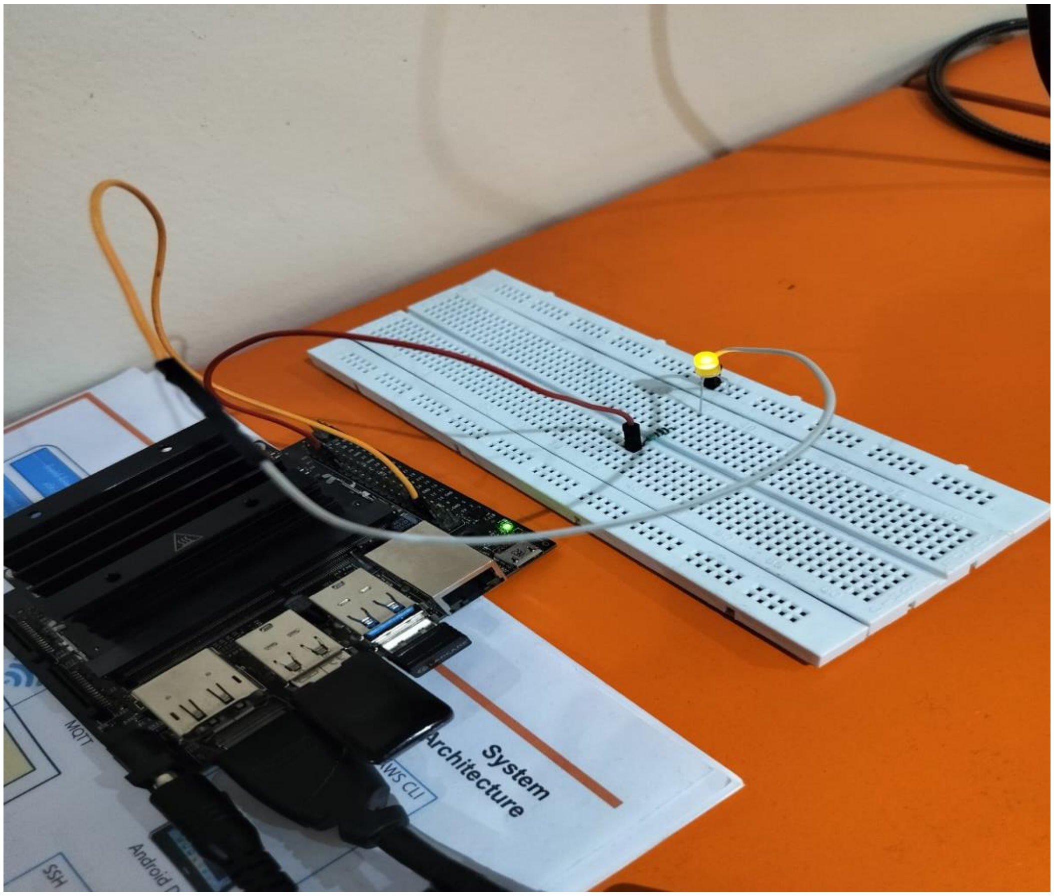
2. **Make the script executable**:

```
chmod +x test.py
```

3. **Run it:**

```
python3 test.py
```

- **Output** -



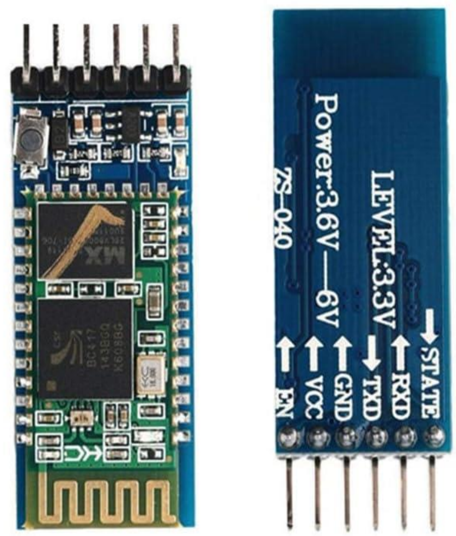
2) UART Loopback Test (Bluetooth)

Universal Asynchronous Receiver/Transmitter

A) **<u>Bluetooth -</u>**

- Jetson Nano
- **HC-05 Bluetooth Module** (3.3V logic recommended)

- Jumper wires



HC-05 Bluetooth Module

- **Wiring HC-05 to Jetson**

HC-05Pin	Jetson Nano Pin
TX	RX (Pin 10)
RX	TX (Pin 8)
GND	GND (Pin 6)
VCC	5V (Pin 2 or 4)

Step 1: Enable UART

sudo /opt/nvidia/jetson-io/jetson-io.py

Step 2: Test Bluetooth Serial

After pairing HC-05 with a phone/PC:

1. Open a serial terminal on your phone
2. Pair with **HC-05** (default PIN: 1234 or 0000)
3. Connect to the serial and send text.

On Jetson side:

```
import serial
bt = serial.Serial('/dev/ttyTHS1', 9600, timeout=1)

print("Listening for Bluetooth messages...")
try:
    while True:
        data = bt.readline().decode(errors='ignore').strip()
        if data:
            print("Received:", data)
            bt.write(f"Echo: {data}\n".encode())
except KeyboardInterrupt:
    bt.close()
```

6) Networking -

1) Test Wi-Fi (USB or dongle)

Step 1. Check if Wi-Fi dongle is detected

Plug in the USB Wi-Fi dongle, then run: lsusb

You should see an entry like:

Bus 001 Device 004: ID 0bda:8179 Realtek Semiconductor Corp.
RTL8188EUS 802.11n Wireless Network Adapter

Step 2. Check network interface

Run: iwconfig

You should see:
wlan0 IEEE 802.11 ESSID:off/any
Mode:Managed Access Point: Not Associated

2) Test Ethernet -

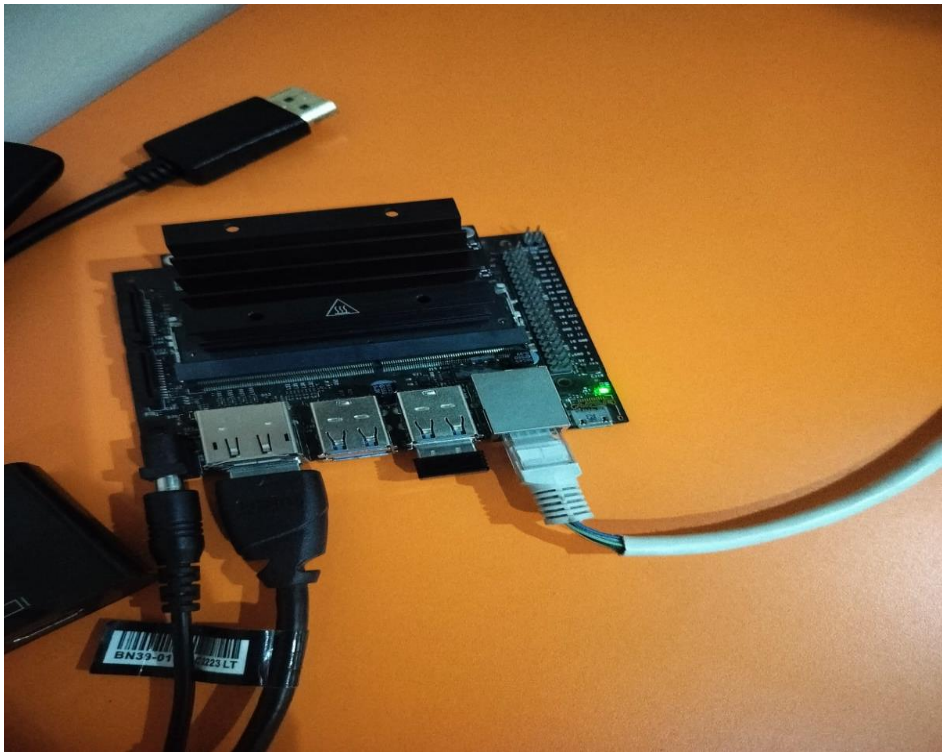
- **Hardware Setup**
  Connect Jetson Nano to your **router/switch/modem** using a standard **RJ45 Ethernet cable**.

- **Check Ethernet Connection Status**
  Check via Terminal: Ifconfig

- **Test Network Connectivity**
  Ping Gateway or Internet: ping -c 4 8.8.8.8

- **Get Interface Name and Details**

nmcli device status



- **To transfer a file using Ethernet between two devices (like Jetson Nano to Laptop/PC) - Using Python Socket**

- **<u>Client.py (laptop/pc)</u>**

import socket import os

file_path = './recording.py'

print("Current directory:", os.getcwd()) print("File exists:", os.path.isfile(file_path)) print("Readable?", os.access(file_path, os.R_OK)) jetson_ip = '192.168.1.92' # Use correct Jetson IP

try: with open(file_path, 'rb') as f: print(f"Sending {file_path}...") s = socket.socket() s.connect((jetson_ip, 5001))

```
    while True:
        data = f.read(1024)
        if not data:
            break
        s.send(data)

    s.close()
    print("File sent successfully.")
```

except Exception as e: print(f"ERROR: {e}")

- **Server.py (jetson nano)**

import socket import os

save_path = './received_recording.py'

print("Will save to:", os.path.abspath(save_path))

s = socket.socket() s.bind(('0.0.0.0', 5001))

s.listen(1) print("Waiting for connection...")

conn, addr = s.accept() print(f"Connected by {addr}")

with open(save_path, 'wb') as f:

```
    print("Receiving file...")

    while True: data = conn.recv(1024)
```

if not data:

break

f.write(data)

conn.close() print(f"File received and saved to {save_path}")

7) Software APIs

1) **OpenCV camera stream (Capture Recording, Snapshot)**

An OpenCV camera stream refers to real-time video capture using a camera via OpenCV's cv2.VideoCapture() interface.

a) Capture Recording -

```
'''
Controls:
- Press R (uppercase only) to start recording (saves to a new timestamped folder)
- Press S to stop recording
- Press q or ESC to quit
'''



import cv2 #camera access and video writing.
import os #Used for path operations
import pathlib #file path
import datetime #Current timestamp
import time

# ------------------ Configuration ------------------ #
CAM_ID = 0 # Camera index Usually /dev/video0
FRAME_SIZE = (1280, 720) # Width x Height
DEFAULT_FPS = 20 #30 # Default fallback FPS (Frame Per Second)
ROOT_SAVE_DIR = pathlib.Path("/home/jetson/DMS/Recording/videos")
FOURCC = cv2.VideoWriter_fourcc(*"mp4v") # Output code
```

```
# ------------------------------------------------------- #

# Use V4L2 backend to avoid GStreamer issues
cap = cv2.VideoCapture(CAM_ID, cv2.CAP_V4L2)

# Set camera properties
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG')) #Sets the camera to MJPG format
cap.set(cv2.CAP_PROP_FRAME_WIDTH , FRAME_SIZE[0]) #Sets width, height, and FPS.
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FRAME_SIZE[1])
cap.set(cv2.CAP_PROP_FPS , DEFAULT_FPS)

# Get actual FPS from camera
actual_fps = cap.get(cv2.CAP_PROP_FPS)
if actual_fps <= 1.0:
print("Warning: Unable to detect camera FPS, falling back to default:", DEFAULT_FPS)
actual_fps = DEFAULT_FPS
else:
print("Detected camera FPS:", actual_fps)

# Verify camera is open, check camera access
if not cap.isOpened():
print("Error: Cannot open USB camera.")
exit()

recording = False
writer = None
last_frame_time = time.time()

print("Controls: R=start recording S=stop recording q/ESC=quit")

while True:
ok, frame = cap.read()
if not ok:
print("Warning: Failed to grab frame")
break

cv2.imshow("Live USB Camera (R=start, S=stop, q=quit)", frame)

key = cv2.waitKey(1) & 0xFF
if key in (ord('q'), 27): # Quit on 'q' or ESC
break
```

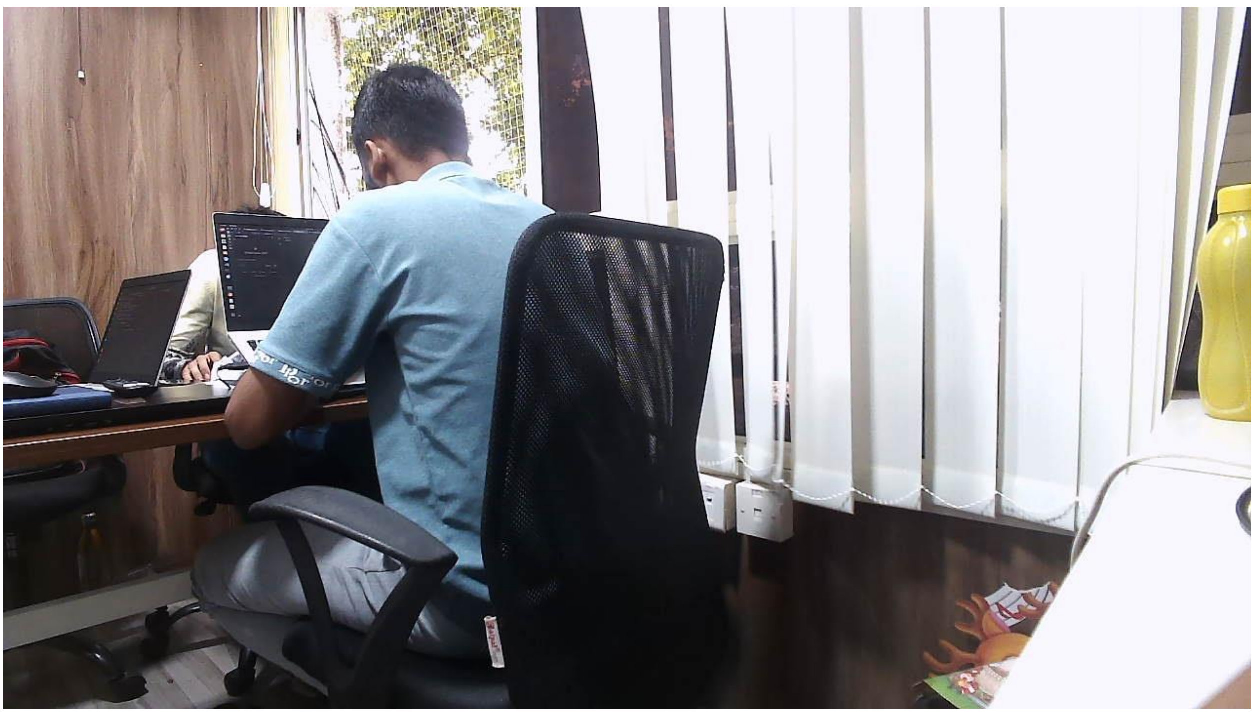
```
# Start recording
if key == ord('R') and not recording:
now = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
folder = ROOT_SAVE_DIR / now
folder.mkdir(parents=True, exist_ok=True)
filepath = folder / f"capture_{now}.mp4"

writer = cv2.VideoWriter(str(filepath), FOURCC, actual_fps, FRAME_SIZE)
if not writer.isOpened():
print("Error: Failed to open video writer.")
writer = None
else:
recording = True
last_frame_time = time.time()
print(f"Recording started -> {filepath}")

# Stop recording
if key == ord('S') and recording:
recording = False
writer.release()
writer = None
print("Recording stopped.")

# Save frame if recording with correct FPS pacing
if recording and writer is not None:
current_time = time.time()
if current_time - last_frame_time >= (1.0 / actual_fps):
writer.write(frame)
last_frame_time = current_time

# Cleanup
if writer is not None:
writer.release()
cap.release()
cv2.destroyAllWindows()
print("Program finished.")
```

b) Capture Snapshot-

```
import cv2
import os
import pathlib
import datetime

# ------------------- Configuration ------------------- #
CAM_ID = 0
FRAME_SIZE = (1280, 720)
SNAPSHOT_SAVE_DIR = pathlib.Path("/home/jetson/DMS/Recording/snapshots")
# ----------------------------------------------------- #

cap = cv2.VideoCapture(CAM_ID, cv2.CAP_V4L2)

# Set camera properties
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
cap.set(cv2.CAP_PROP_FRAME_WIDTH , FRAME_SIZE[0])
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FRAME_SIZE[1])

if not cap.isOpened():
print("Error: Cannot open USB camera.")
exit()

print("Controls: S = snapshot q / ESC = quit")

while True:
ret, frame = cap.read()
if not ret:
print("Warning: Failed to grab frame")
break

cv2.imshow("Live USB Camera (S=snapshot, q=quit)", frame)

key = cv2.waitKey(1) & 0xFF
if key in (ord('q'), 27): # Quit
break
```

```
if key == ord('S'):
now = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
folder = SNAPSHOT_SAVE_DIR / now
folder.mkdir(parents=True, exist_ok=True)
filepath = folder / f"snapshot_{now}.jpg"
cv2.imwrite(str(filepath), frame)
print(f"Snapshot saved to {filepath}")

cap.release()
cv2.destroyAllWindows()
print("Snapshot program finished.")
```

Output -



2) <u>Jetson-IO for enabling SPI/UART</u>

SPI – Serial Peripheral Interface
UART – Universal Asynchronous Receiver-Transmitter

The **Jetson-IO tool** configures the pinmux (alternate functions of GPIO pins) on the 40-pin header for peripherals like **UART, SPI, I2C**, etc.

- **<u>Steps - Enable SPI or UART via Jetson-IO</u>**

<u>Step 1: Launch Jetson-IO Tool</u>

Open a terminal and run: sudo /opt/nvidia/jetson-io/jetson-io.py

<u>Step 2: Configure 40-pin Header</u>

In the Jetson-IO UI:

1. Select Configure Jetson 40-pin Header
2. You will see a list of available interfaces, like:
    a. UART1, UART2, etc.
    b. SPI1, SPI2
    c. I2C1, I2C2, etc.

<u>Step 3: Enable UART or SPI</u>

- Use the keyboard to select UART (UART2) or SPI1.
- Press Enter to toggle the interface ON.
- After making changes, select Save and Exit.

<u>Step 4: Reboot to Apply Changes</u>

Sudo reboot

Step 5: Check if SPI/UART is Enabled

For UART: ls /dev/ttyTHS*

For SPI: ls /dev/spidev*

3) V4L2 camera interface test

To test a V4L2 (Video4Linux2) camera interface on Linux

Step 1: Plug in Your Camera

Ensure the USB camera is connected properly.

Step 2: Check If V4L2 Camera is Detected

Run: v4l2-ctl --list-devices

Output -

jetson@nano:~$ v4l2-ctl --list-devices

NVIDIA Tegra Video Input Device (platform:vi):

/dev/media0

Lenovo FHD Webcam Audio (usb-70090000.xusb-2.2):

/dev/video0

/dev/media1

Step 3: Get Detailed Info -

Run: v4l2-ctl --all -d /dev/video0

Output:

jetson@nano:~$ v4l2-ctl --all -d /dev/video0

Driver Info:

Driver name : uvcvideo

Card type : Lenovo FHD Webcam Audio

Bus info : usb-70090000.xusb-2.2

Driver version : 4.9.253

Capabilities : 0x84200001

Video Capture

Streaming

Extended Pix Format

Device Capabilities

Device Caps : 0x04200001

Video Capture

Streaming

Extended Pix Format