Jetson Nano Complete Setup: Chromium, RealSense, ROS, YOLOv8

1. Flashing the SD Card (Host Computer Required)

1.1. Download the Jetson Nano Developer Kit SD Card Image (JetPack OS) from the NVIDIA website. 1.2. Download and install a disk imaging tool like balenaEtcher on your host computer. 1.3. Insert your microSD card (32GB or larger, UHS-I speed recommended) into your host computer. 1.4. Use Etcher to select the downloaded image file, select the microSD card drive, and start the flashing process. 1.5. Safely eject the microSD card when flashing is complete.

2. Connect to Monitor and Power-On (Jetson Nano)

2.1. Insert the flashed microSD card into the slot on the underside of the Jetson Nano board. 2.2. Connect a USB Keyboard and USB Mouse. 2.3. Connect a monitor using an HDMI or DisplayPort cable. 2.4. Connect the Intel RealSense Depth Camera to a USB 3.0 (blue) port. 2.5. Connect the 5V/4A DC power supply to the barrel jack or a 5V/2A Micro-USB power supply. The Nano powers on automatically. 2.6. Follow the on-screen prompts for initial OS setup, creating your username and password.

3. Connect to Wi-Fi and Install Chromium

3.1. Connect to Wi-Fi: Click the Network Icon, select your network, and enter the password. 3.2. Open Terminal: Press Ctrl + Alt + T to open the Terminal application. 3.3. System Update and Chromium Installation (Terminal Commands):

```
sudo apt update
sudo apt upgrade -y
sudo apt install chromium-browser -y
```

4. Intel RealSense Depth Camera Setup (librealsense SDK)

4.1. Install Dependencies (Terminal Commands):

```
sudo apt install build-essential git libssl-dev libusb-1.0-0-dev pkg-config libgtk-3-dev
```

4.2. Clone and Setup SDK (Terminal Commands):

```
cd ~
git clone https://github.com/IntelRealSense/librealsense.git
cd librealsense
git checkout v2.55.1
./scripts/setup_udev_rules.sh
```

4.3. Critical Action: Unplug the RealSense camera and plug it back into the USB 3.0 port now to apply the new access rules.

4.4. Compile and Install SDK (Terminal Commands): This process is time-consuming (1 to 2 hours).

```
mkdir build
cd build
cmake ../ -DCMAKE_BUILD_TYPE=Release -DFORCE_RSUSB_BACKEND=ON -DBUILD_EXAMPLES=true -DBU
make -j4
sudo make install
```

4.5. Verification Command (Terminal Command):

```
realsense-viewer
```

5. Robot Operating System (ROS) Noetic Setup

5.1. Setup ROS Repository (Terminal Commands): Assuming Ubuntu 20.04 (Focal Fossa) compatibility.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu focal main" > /etc/apt/sources.
sudo apt install curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key ad
sudo apt update
```

5.2. Install ROS Noetic (Terminal Commands): Installing the base environment.

```
sudo apt install ros-noetic-desktop
sudo rosdep init
rosdep update
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt install python3-catkin-tools
```

6. YOLOv8 and OpenCV Test Environment Setup

YOLOv8 requires PyTorch and a compatible Python version (3.8+ recommended). Due to Jetson's specific architecture, PyTorch must be installed via a pre-built wheel.

6.1. Install PyTorch and Dependencies (Terminal Commands): The specific wheel URL may change; verify the latest for your JetPack version. Example for JetPack 4.6 (Python 3.6).

```
sudo apt install python3-pip libopenblas-base libopenmpi-dev libomp-dev
# Download a compatible PyTorch wheel for your JetPack/Python version
# Example URL for a known older version (adjust for your JetPack/Python):
# wget https://nvidia.box.com/shared/static/p57jwntv436lfrd78inwl7iml6p13fzh.whl -O torch
# sudo pip3 install numpy torch-1.8.0-cp36-cp36m-linux_aarch64.whl
# rm torch-1.8.0-cp36-cp36m-linux_aarch64.whl

# Build and Install Torchvision (specific version compatible with PyTorch)
sudo apt install libjpeg-dev zlib1g-dev libpython3-dev
# Adjust branch to match your PyTorch version (e.g., release/0.9)
# git clone --branch release/0.9 https://github.com/pytorch/vision torchvision
# cd torchvision
# sudo python3 setup.py install
# cd ..

# Install Ultralytics YOLOv8
sudo pip3 install ultralytics
```

6.2. Test YOLOv8 Inference (Terminal Command): This tests if PyTorch/YOLOv8 and the camera (typically /dev/video0 for USB cameras) are functioning together.

```
# Run YOLOv8 on the camera stream (use '0' for the first detected camera)
yolo task=detect mode=predict model=yolov8n.pt source=0 show=True
```

6.3. OpenCV Test Script (Terminal Commands): This confirms OpenCV is installed and can access the RealSense's color stream via the standard V4L2 device node (which the SDK exposes).

```
# Install dependencies for OpenCV test
sudo apt install python3-opencv

# Create a simple Python test file
echo '
import cv2
import sys

# Try opening the camera device node (0 is usually the first USB camera)
```

```
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Error: Could not open camera.")
    sys.exit()

while True:
    ret, frame = cap.read()
    if not ret:
        break

    cv2.imshow("Camera Test", frame)

    # Exit loop on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
' > opencv_test.py

# Run the test
python3 opencv_test.py
```