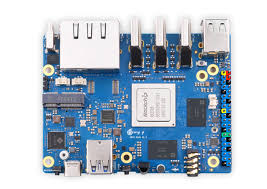
# Orange Pi 5 Plus Installation and usage Resources



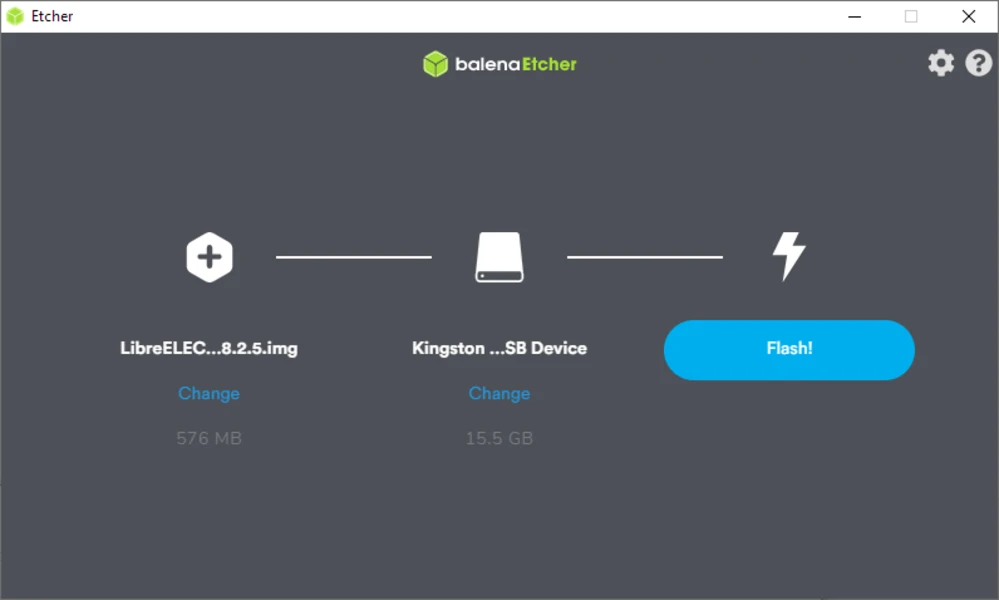
# OS Installation

## Step 1 : Downloading OS image :

Download the image from this link : <https://joshua-riek.github.io/ubuntu-rockchip-download/boards/orangepi-5-plus.html>

Step 2 : Flashing the image :

* Download Balena etcher <https://etcher.balena.io/#download-etcher>
* Insert SD card to SD card reader and plug it to your PC.
* Open Balena Etcher, select downloaded image , select target sd card and click on flash.



# ROS2 Installation :

* Set Locale UTF file type

locale # check for UTF-8

sudo apt update && sudo apt install locales

sudo locale-gen en\_US en\_US.UTF-8

sudo update-locale LC\_ALL=en\_US.UTF-8 LANG=en\_US.UTF-8

export LANG=en\_US.UTF-8

locale # verify settings

* Add Remote Repository

sudo apt install software-properties-common

sudo add-apt-repository universe

sudo apt update && sudo apt install curl -y

export ROS\_APT\_SOURCE\_VERSION=*$*(curl -s https:*//api.github.com/repos/ros-infrastructure/ros-apt-source/releases/latest | grep -F "tag\_name" | awk -F\" '{print $4}')*

curl -L -o /tmp/ros2-apt-source.deb "https://github.com/ros-infrastructure/ros-apt-source/releases/download/${ROS\_APT\_SOURCE\_VERSION}/ros2-apt-source\_${ROS\_APT\_SOURCE\_VERSION}.$(. /etc/os-release && echo $VERSION\_CODENAME)\_all.deb" # If using Ubuntu derivates use $UBUNTU\_CODENAME

sudo apt install /tmp/ros2-apt-source.deb

* Install ROS2

sudo apt update

sudo apt upgrade

sudo apt install ros-humble-desktop

* Add path to sytem

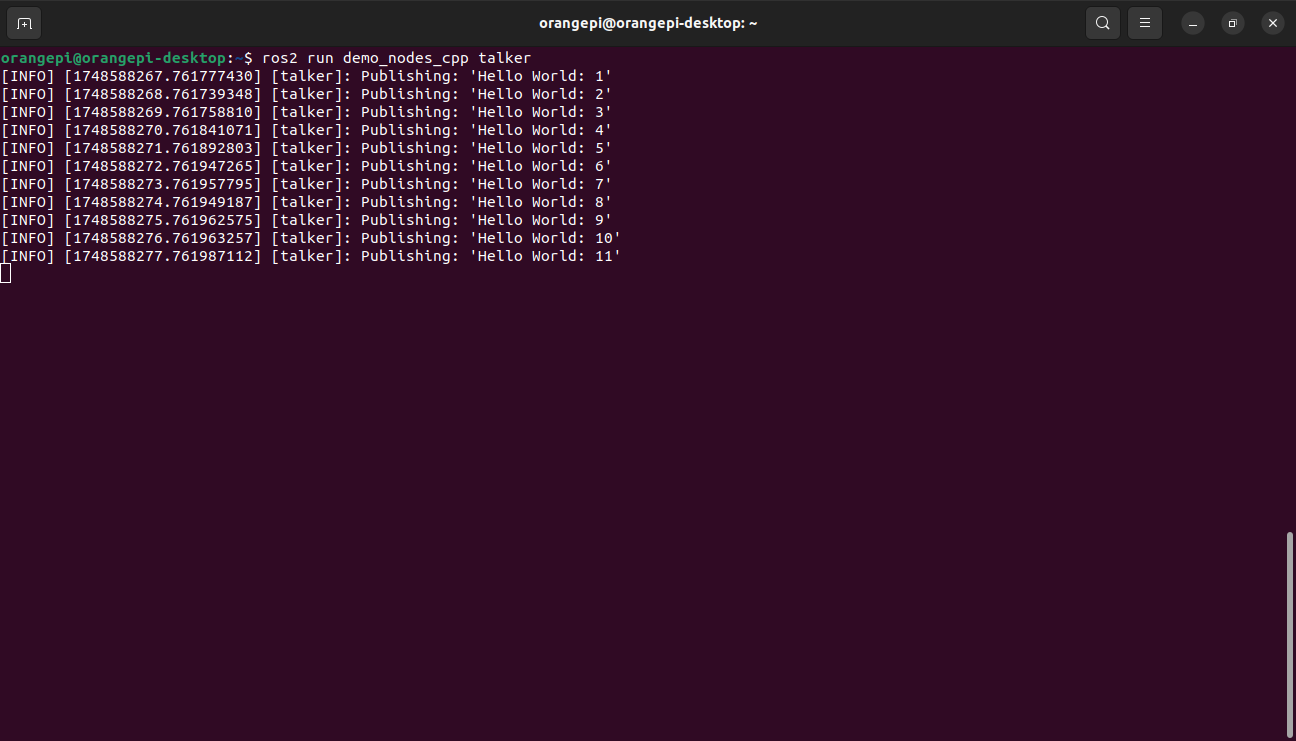
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc

source ~/.bashrc

Test Example nodes

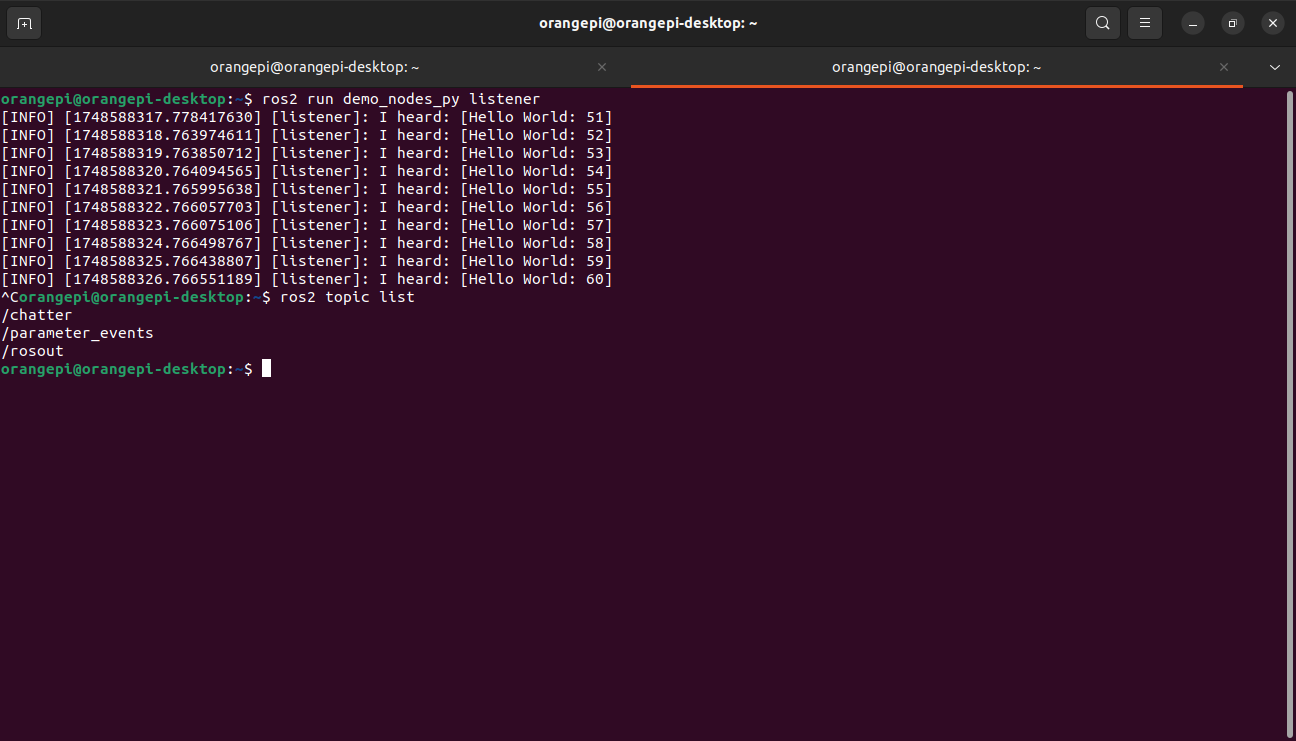
* Run demo Publisher /talker

*ros2* run demo\_nodes\_cpp talker



* Rune demo Subscriber/listener

*ros2* run demo\_nodes\_py listener



* Create a new workspace

*mkdir* -p ~/ros2\_ws/src

*cd* ~/ros2\_ws

# Install Robomaster ROS2 Package

* Install Dependencies

*sudo* apt install \

ros-humble-xacro \

ros-humble-launch-xml \

ros-humble-cv-bridge \

ros-humble-launch-testing-ament-cmake \

ros-humble-robot-state-publisher \

ros-humble-joint-state-publisher \

ros-humble-joint-state-publisher-gui \

ros-humble-joy \

ros-humble-joy-teleop \

ros-humble-joy-linux

* Install Robomaster SDK Python Drivers

*sudo* apt install libopus-dev python3-pip

*python3* -m pip install -U numpy numpy-quaternion pyyaml

*python3* -m pip install git+<https://github.com/jeguzzi/RoboMaster-SDK.git>

*python3* -m pip install git+https://github.com/jeguzzi/RoboMaster-SDK.git#"egg=libmedia\_codec&subdirectory=lib/libmedia\_codec"

* **Install ROS2 Package for robomaster.**

***cd* ~/ros2\_ws/src**

***git* clone https://github.com/jeguzzi/robomaster\_ros.git**

***cd* ~/ros2\_ws/**

***source* /opt/ros/humble/setup.bash**

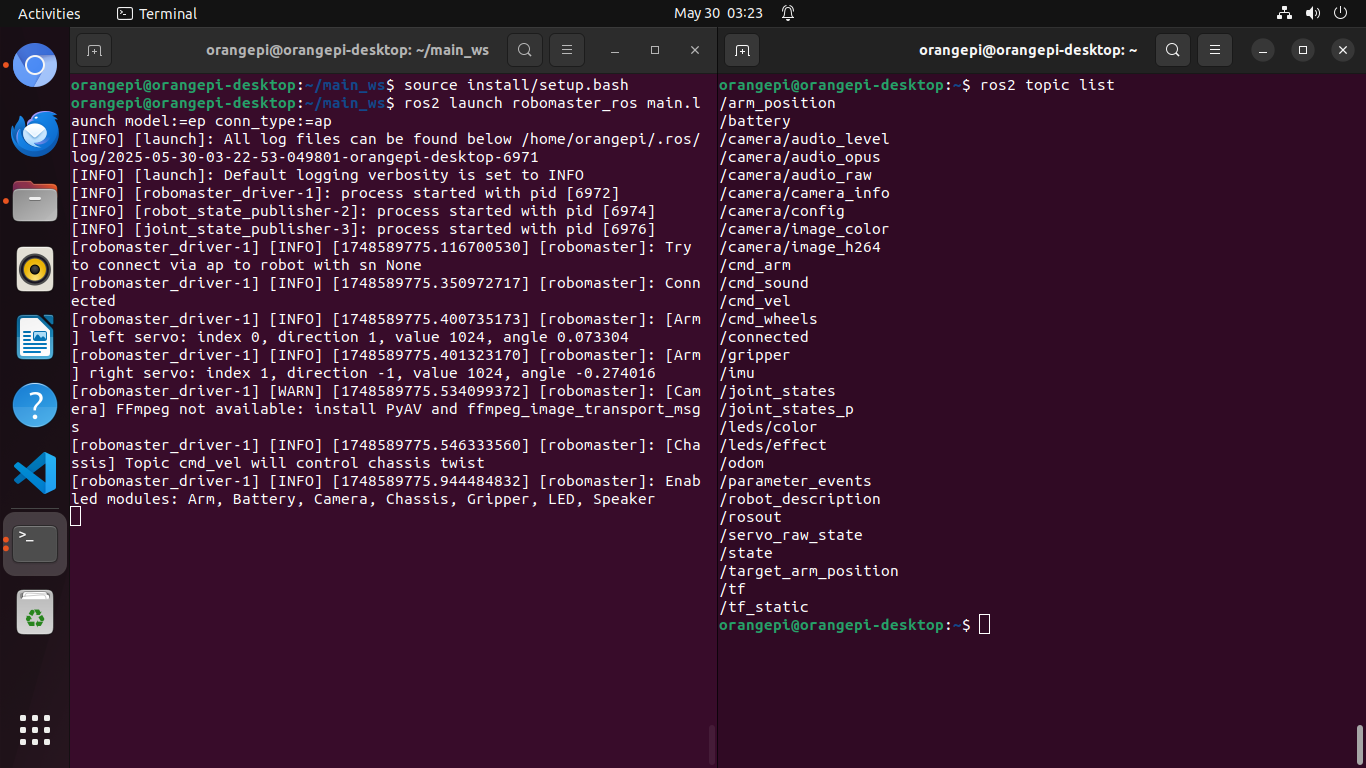
***colcon* build**

* **Connect PC/jetson wifi to robomaster wifi**
* **Launch Robomaster Node**

***source ./install/setup.bash***

***ros2* launch robomaster\_ros main.launch model:=ep conn\_type:=ap**

**For USB cable, use conn\_type:NDIS**

****

# Install RP lidar ROS2 package and Drivers

* Install Package

*cd* ~/ros2\_ws/src

*git* clone https://github.com/Slamtec/rplidar\_ros.git

*cd* ~/ros2\_ws/

*source* /opt/ros/humble/setup.bash

*colcon* build --symlink-install

*echo* "source ros2\_ws/install/setup.bash" >> ~/.bashrc

*source* ~/.bashrc

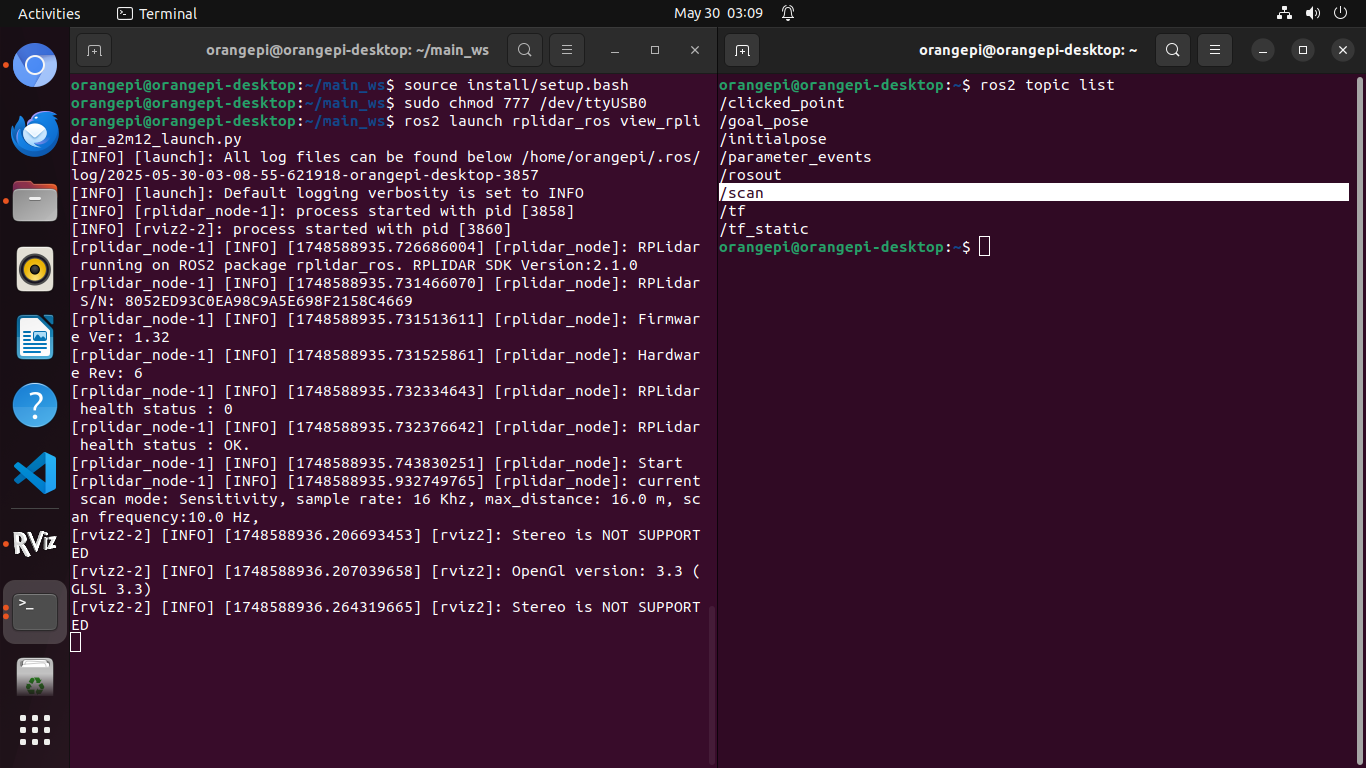
* Launch the node

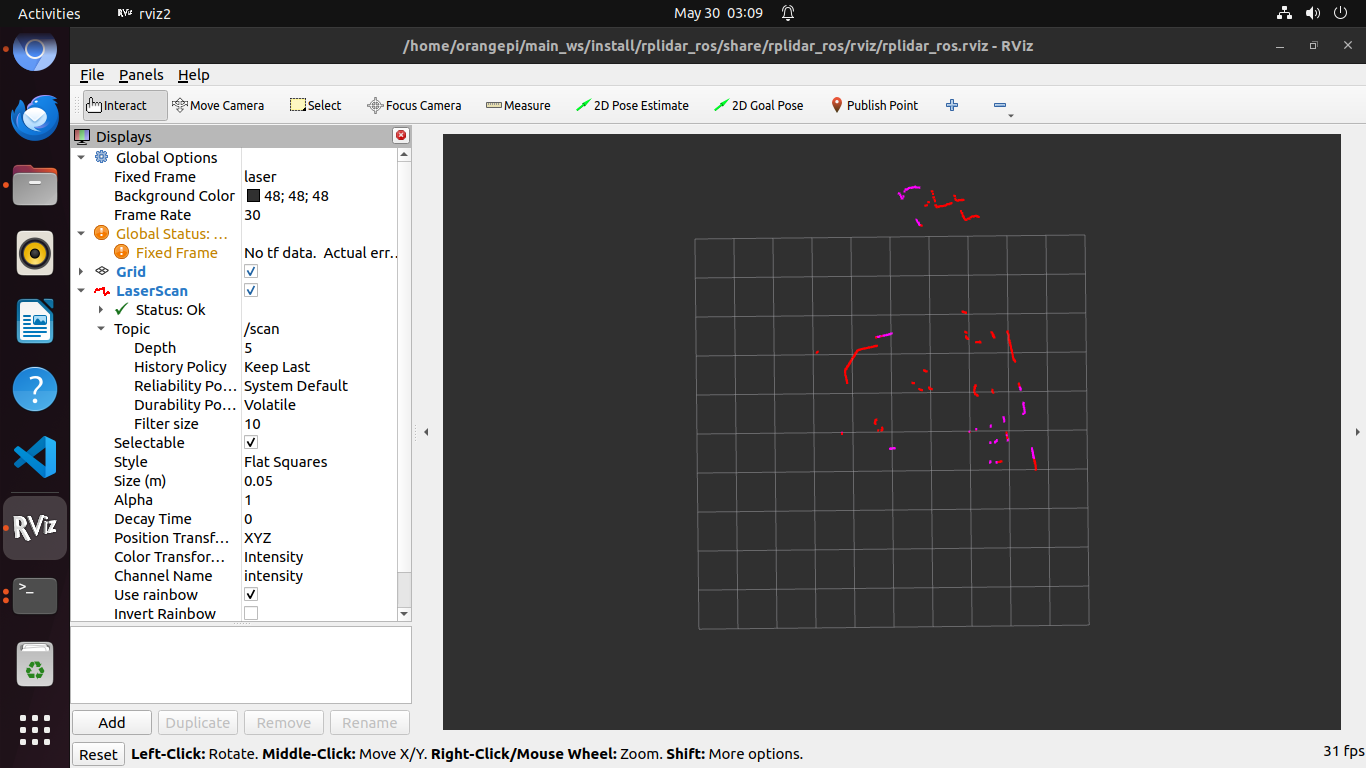
*sudo* chmod 777 /dev/ttyUSB0

*cd* src/rpldiar\_ros/

*source* scripts/create\_udev\_rules.sh

*ros2 launch rplidar\_ros view\_rplidar\_a2m12\_launch.py*





# Install realsense drivers and package

* Install librealsense2 (realsense drivers)

*sudo* apt-get install libssl-dev libusb-1.0-0-dev libudev-dev pkg-config libgtk-3-dev

*sudo* apt-get install libglfw3-dev libgl1-mesa-dev libglu1-mesa-dev at

*sudo* apt-get install git wget cmake build-essential

*git* clone https://github.com/IntelRealSense/librealsense.git

*cd* librealsense

*sudo* cp config/99-realsense-libusb.rules /etc/udev/rules.d/

*sudo* udevadm control --reload-rules && *sudo* udevadm trigger

*mkdir* build && *cd* build

*cmake* ../ -DFORCE\_RSUSB\_BACKEND=true -DCMAKE\_BUILD\_TYPE=release -DBUILD\_EXAMPLES=true -DBUILD\_GRAPHICAL\_EXAMPLES=true

*sudo* make uninstall && *make* clean && *make* -j5 && *sudo* make install

Note This will take some time (1-2 hours)

* Install ROS2 Package

*mkdir* -p ~/realsense\_ws/src

*git* clone https://github.com/IntelRealSense/realsense-ros.git -b ros2-master

*cd* ~/realsense\_ws

*sudo* apt-get install python3-rosdep -y

*sudo* rosdep init *# "sudo rosdep init --include-eol-distros" for Foxy and earlier*

*rosdep* update *# "sudo rosdep update --include-eol-distros" for Foxy and earlier*

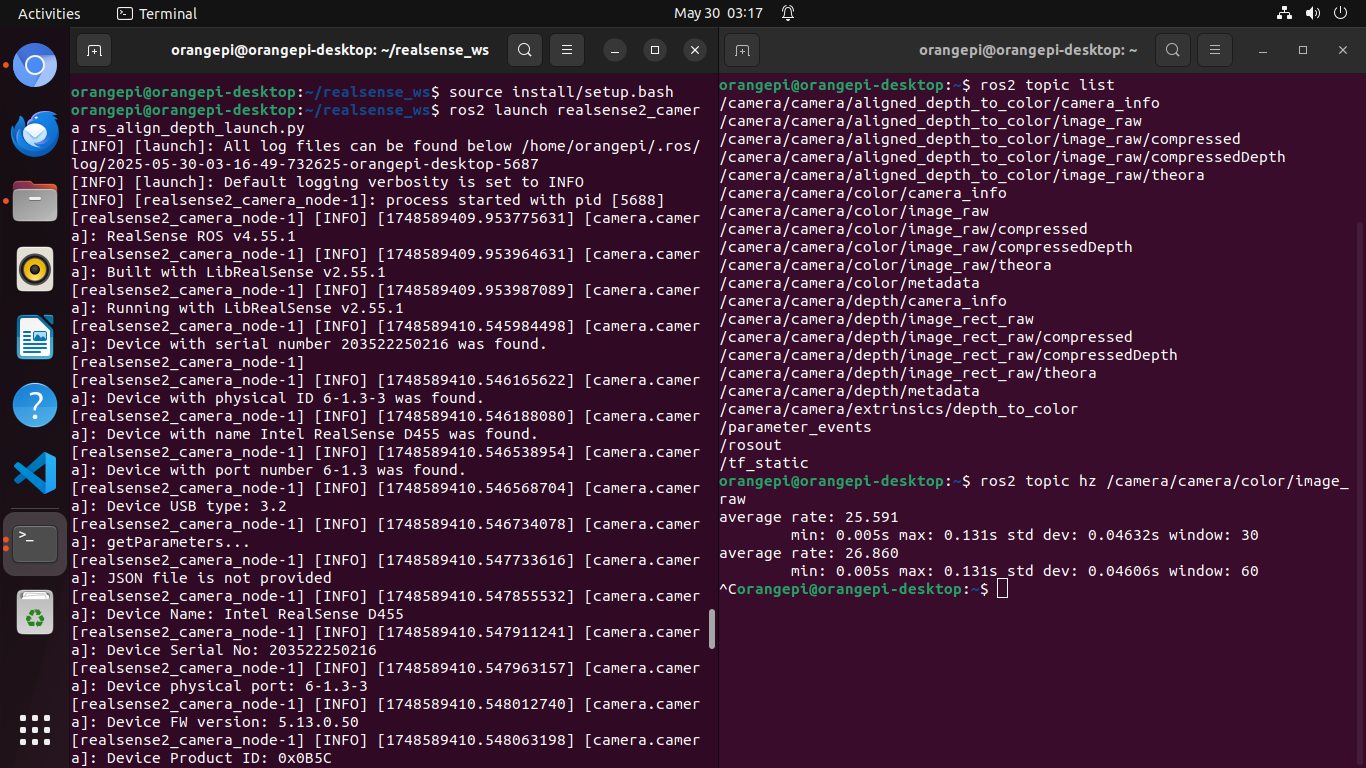
*rosdep* install -i --from-path src --rosdistro $ROS\_DISTRO --skip-keys=librealsense2 -y

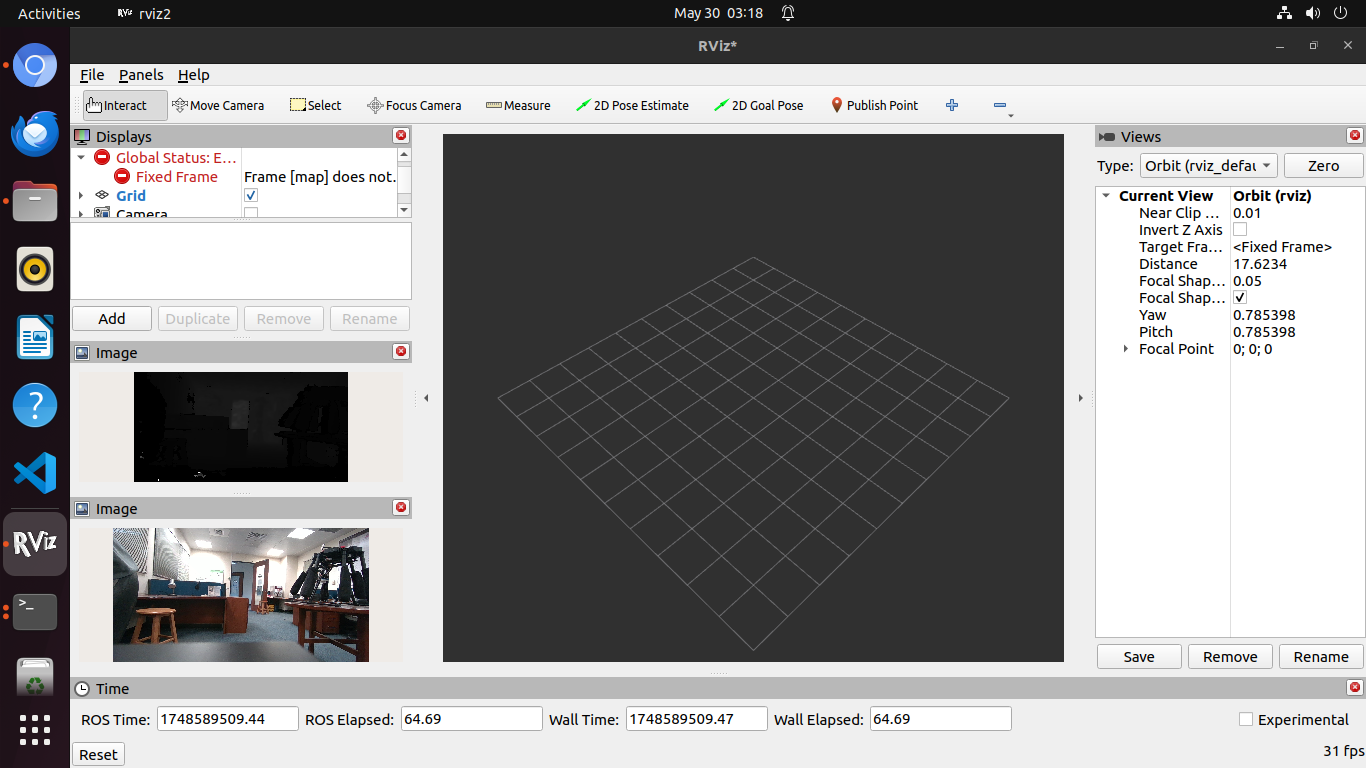
*colcon* build

* Lunch Node :

*source install/setup.bash*

*ros2 launch realsense2\_camera rs\_align\_depth\_launch.py*





# Deploying Pytorch Model on Orange Pi NPU

There is 2 Stage Workflow to deploy model on orange pi . First you convert your model from pytorch to rknn format for orange pi NPU on your Intel x86 PC/laptop

* **Stage 1 PC/laptop Setup (Model Conversion)**

**Complete Flow is Pytorch -> ONNX -> RKNN**

* Convert to ONNX format

*python3* -m pip install onnx

* Convert Model from Pytorch to onnx

*import* torch

*import* torchvision

*# Load the ResNet18 model from torchvision*

*model* = torchvision.models.resnet18(pretrained=True)

*# Set the model to evaluation mode*

*model.eval*()

*# Define the input shape (batch size = 1, channels = 3, height = 224, width = 224)*

*input\_shape* = (1, 3, 224, 224)

*# Create a dummy input tensor*

*dummy\_input* = torch.randn(\*input\_shape)

*# Export the model to ONNX format*

*torch.onnx.export(model,*

*dummy\_input,*

*"resnet18.onnx",*

export\_params=True,

opset\_version=11, *# ONNX opset version*

*input\_names* = ['input'], *# input name*

*output\_names* = ['output']) *# dynamic batch size for output*

*print*(*"ResNet18 model exported to resnet18.onnx"*)

* Install RKNN toolkit to convert onnx model to .rknn format

*python3* -m pip install onnx

*sudo* apt-get install libxslt1-dev zlib1g-dev libglib2.0 libsm6 libgl1-mesa-glx libprotobuf-dev gcc

*git* clone --depth=1 https://github.com/airockchip/rknn-toolkit2

*cd* rknn-toolkit2

PYTHON\_VERSION=$(*python3* -c 'import sys; print(f"cp{sys.version\_info.major}{sys.version\_info.minor}")')

*python3* -m pip install -r rknn-toolkit2/packages/x86\_64/requirements\_${PYTHON\_VERSION}-2.3.2.txt

PYTHON\_VERSION=$(*python3* -c 'import sys; print(f"cp{sys.version\_info.major}{sys.version\_info.minor}-cp{sys.version\_info.major}{sys.version\_info.minor}")')

*python3* -m pip install rknn-toolkit2/packages/x86\_64/rknn\_toolkit2-2.3.2-${PYTHON\_VERSION}-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl

* Convert to rknn
  + Prepare dataset.txt : Create text Document with 1 or 2 images paths for conversion (These should be the images the model was trained for

*/path/to/image1.jpg*

*/path/to/image2.jpg*

*...*

* Convert Model

For float16

*import* numpy as np

*from* rknn.api import RKNN

*import* uuid

*ONNX\_MODEL* = 'resnet18.onnx'

*RKNN\_MODEL* = 'resnet18.rknn'

*INPUT\_SHAPE* = [1, 3, 224, 224]

*TARGET\_PLATFORM* = 'rk3588'

*rknn* = RKNN()

*# Confihgure input tensor normalization , these should be configured deending upon your dataset pre-processing*

*rknn.config(*

mean\_values=[[0, *0,* 0]], *# Default mean values for normalization*

std\_values=[[255, *255,* 255]], *# Default std values for normalization*

target\_platform=target\_platform

)

*ret* = rknn.load\_onnx(model=ONNX\_MODEL, input\_size\_list=[INPUT\_SHAPE])

*ret* = rknn.build(do\_quantization=False)

*ret* = rknn.export\_rknn(*f'{RKNN\_MODEL}\_float.rknn'*)

* For Quantization int8

For quantization we will use dataset.txt

*import* numpy as np

*from* rknn.api import RKNN

*import* uuid

*ONNX\_MODEL* = 'resnet18.onnx'

*RKNN\_MODEL* = 'resnet18.rknn'

*INPUT\_SHAPE* = [1, 3, 224, 224]

*DATASET\_PATH* = './dataset.txt' *# Path to dataset file for quantization (optional)*

*TARGET\_PLATFORM* = 'rk3588'

*rknn* = RKNN()

*# Confihgure input tensor normalization , these should be configured deending upon your dataset pre-processing*

*rknn.config(*

mean\_values=[[0, *0,* 0]], *# Default mean values for normalization*

std\_values=[[255, *255,* 255]], *# Default std values for normalization*

target\_platform=target\_platform

)

*ret* = rknn.load\_onnx(model=ONNX\_MODEL, input\_size\_list=[INPUT\_SHAPE])

*ret* = rknn.build(do\_quantization=True, dataset=DATASET\_PATH)

*ret* = rknn.export\_rknn(*f'{RKNN\_MODEL}\_int8.rknn'*)

* Step 2 : Mode Deployment
* Install RKNN toolkit on Orange Pi

git clone --depth=1 https://github.com/Pelochus/ezrknn-toolkit2

cd ezrknn-toolkit2/

pip3 install opencv-python

pip3 install -r rknn-toolkit2/packages/arm64/arm64\_requirements\_cp310.txt

pip install rknn-toolkit-lite2/packages/rknn\_toolkit\_lite2-2.3.0-cp310-cp310-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl

pip install rknn-toolkit2/packages/arm64/rknn\_toolkit2-2.3.0-cp310-cp310-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl

* Run inference

Demo Inference example

import numpy as np

from rknn.api import RKNN

import cv2

import time

model\_path = '/home/orangepi/Downloads/resnet18\_fp16.rknn'

image\_path = '/home/orangepi/Downloads/n01491361\_tiger\_shark.JPEG'

target\_platform = 'rk3588'

class\_labels = [f"class\_{i}" for i in range(1000)]

rknn = RKNN()

print('--> Configuring model')

if rknn.config(target\_platform=target\_platform) != 0:

print('Failed to configure RKNN model')

exit(1)

print('Configuration done')

print('--> Loading RKNN model')

if rknn.load\_rknn(model\_path) != 0:

print(f'Failed to load RKNN model: {model\_path}')

exit(1)

print('Model loaded successfully')

print('--> Initializing runtime environment')

if rknn.init\_runtime(target\_platform) != 0:

print('Failed to initialize runtime environment')

exit(1)

print('Runtime environment initialized')

img = cv2.imread(image\_path)

if img is None:

print(f'Failed to load image: {image\_path}')

exit(1)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

img = cv2.resize(img, (224, 224))

img = np.expand\_dims(img, axis=0)

print('--> Running inference')

t1 = time.time()

outputs = rknn.inference(inputs=[img])

t2 = time.time()

print(f"Executed in {(t2-t1)\*1000:.2f} ms")

if outputs is not None:

outputs = [np.array(out) for out in outputs]

print('Inference completed successfully')

for i, output in enumerate(outputs):

print(f'Output {i} shape: {output.shape}')

output = outputs[0].flatten()

pred\_class\_idx = np.argmax(output)

confidence = output[pred\_class\_idx]

class\_name = class\_labels[pred\_class\_idx]

print(f'Predicted class: {class\_name}, Confidence: {confidence:.4f}')

else:

print('Inference failed')

rknn.release()

print('RKNN resources released')