# **Camera Preview (USB)**

#### **Camera Preview (USB)**

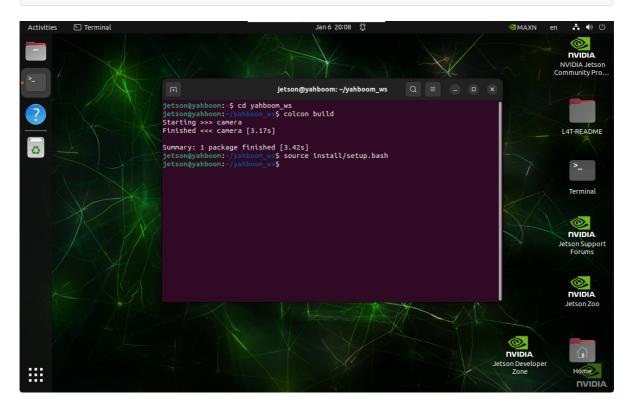
- 1. Compile the function package
- 2. Start the camera
- 3. Preview screen
- 4. Main code

## 1. Compile the function package

cd ~/yahboom\_ws

colcon build

source install/setup.bash



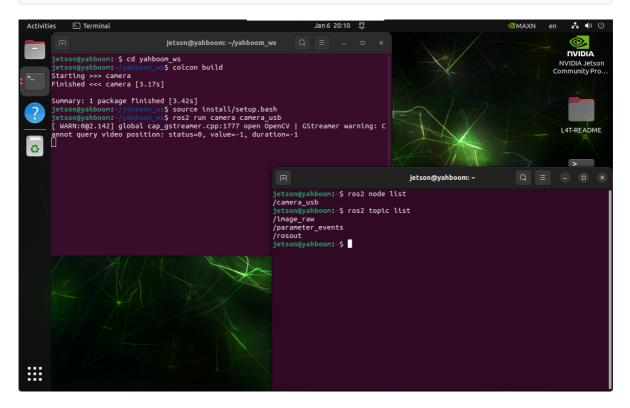
### 2. Start the camera

Start the camera

ros2 run camera camera\_usb

View nodes and topics

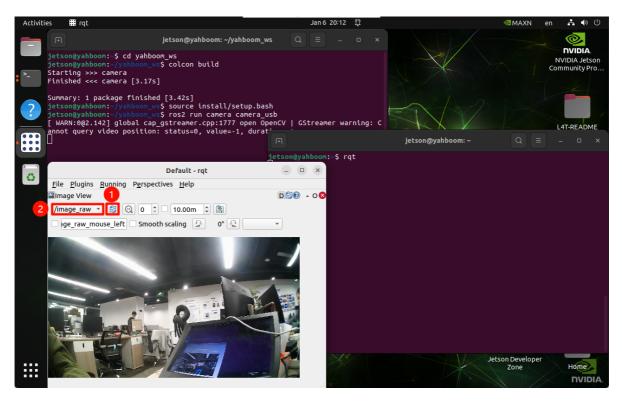
ros2 node list



#### 3. Preview screen

Use rqt to view the screen corresponding to the camera: rqt  $\to$  Plugins  $\to$  Visualization  $\to$  Image View

rqt jetson@yahboom: ~/yahboom\_ws etson@yahboom:-\$ cd yahboom\_ws
cyshboom:-/yahboom\_ws\$ colcon build NVIDIA Starting >>> camera
Finished <<< camera [3.17s] Summary: 1 package finished [3.42s]
jetson@yahboom:-/yahboom\_ws\$ source install/setup.bash
jetson@yahboom:-/yahboom\_ws\$ ros2 run camera camera\_usb
[ WARN:09c1.142] global cap\_gstreamer.cpp:1777 open OpenCV | GStreamer warning: C
annot query video position: status=0, value=-1, durat\* m:~\$ rqt 1 Default - rqt \_ D X 0 File Plugins Running Perspectives Help
Container **Actions** nat is able to load various Configuration windows. There are introspection ected. To add plug-ins, **Logging** Jains menu. miscellaneous Tools icular arrangement of Services using the **Perspectives** Image View Plot **:::** DVIDIA Plugins related to visualization



### 4. Main code

```
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2
class CameraNode(Node):
    def __init__(self):
        super().__init__('camera_usb')
        self.publisher = self.create_publisher(Image, 'image_raw', 10)
        self.bridge = CvBridge()
        self.cap = cv2.VideoCapture(0)
        if not self.cap.isOpened():
            self.get_logger().error('Unable to open camera')
            return
        self.timer = self.create_timer(0.05, self.timer_callback)
    def timer_callback(self):
        ret, frame = self.cap.read()
            image_msg = self.bridge.cv2_to_imgmsg(frame, encoding="bgr8")
            self.publisher.publish(image_msg)
        else:
            self.get_logger().warn('Failed to capture image')
def main(args=None):
    rclpy.init(args=args)
    node = CameraNode()
    rclpy.spin(node)
    node.cap.release()
```

```
rclpy.shutdown()

if __name__ == '__main__':
    main()
```