Camera preview (CSI)

Camera preview (CSI)

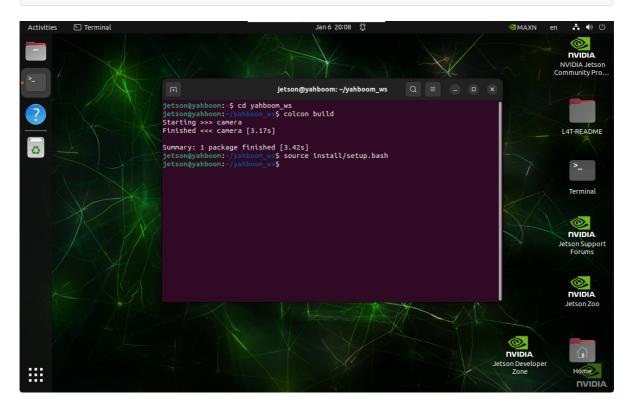
- 1. Compile the function package
- 2. Start the camera
- 3. Preview screen
- 4. Main code

1. Compile the function package

cd ~/yahboom_ws

colcon build

source install/setup.bash



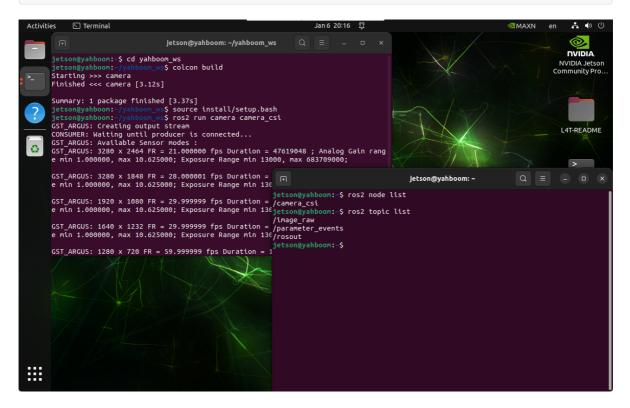
2. Start the camera

Start the camera

ros2 run camera camera_csi

View nodes and topics

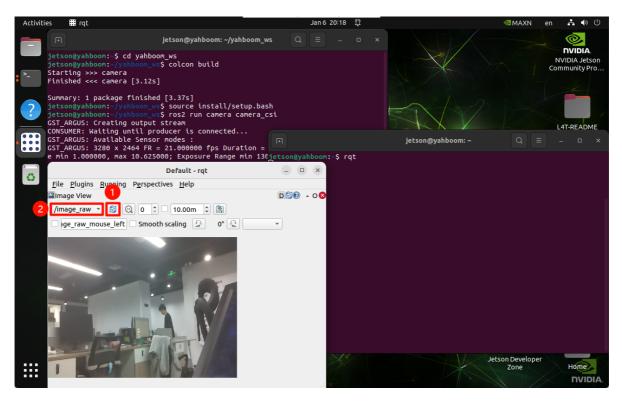
ros2 node list



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 NVIDIA Starting >>> camera
Finished <<< camera [3.12s] Summary: 1 package finished [3.37s]
jetson@yahboom:-/yahboom_ws source install/setup.bash
jetson@yahboom:-/yahboom_ws ros2 run camera camera_csi
GST_ARGUS: Creating output stream
CONSUMER: Waiting until producer is connected...
GST_ARGUS: AVAILable Sensor modes:
GST_ARGUS: 3280 x 2464 FR = 21.000000 fps Duration =
e min 1.000000, max 10.625000; Exposure Range min 136_jetsor L4T-README m:~\$ rqt 1 _ D X Default - rqt 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** Topics Image View Plot **iii** 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
from jetcam.csi_camera import CSICamera
class CameraNode(Node):
    def __init__(self):
        super().__init__('camera_csi')
        self.publisher = self.create_publisher(Image, 'image_raw', 10)
        self.bridge = CvBridge()
        self.cap = CSICamera(capture_device=0, width=640, height=480)
        self.timer = self.create_timer(0.05, self.timer_callback)
    def timer_callback(self):
        frame = self.cap.read()
        if frame is not None:
            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()
```