Edge.Auto Getting Started Manual

TIER IV

May 07, 2024

Contents

1

TIER IV Cameras Getting Started Guide for ADLINK ROSCube RQX-58G

Attention: This document is for the users of ADLINK RQX-58G

1.1 Preparation

1.1.1 Required items

- ADLINK ROSCube-X RQX-58G
- JetPack4.5 (L4T 32.5.1) or JetPack4.6 (L4T 32.6.1) installed
- GMSL2 coaxial cable (FAKRA - mini FAKRA, 1:4)
- TIER IV Automotive HDR Camera C1 or C2

1.1.2 Camera connection

1. Firstly, make sure that the ROSCube-X is turned off.
2. Connect cameras to the FAKRA connectors of the cable(single FAKRA connector side). Then insert the mini FAKRA connectors of the cable to the ROSCube-X’ s GMSL2 ports.
3. Please make sure the direction of the lock clown on FAKRA connectors and mini FAKRA connectors is correct (Fig.1~3).

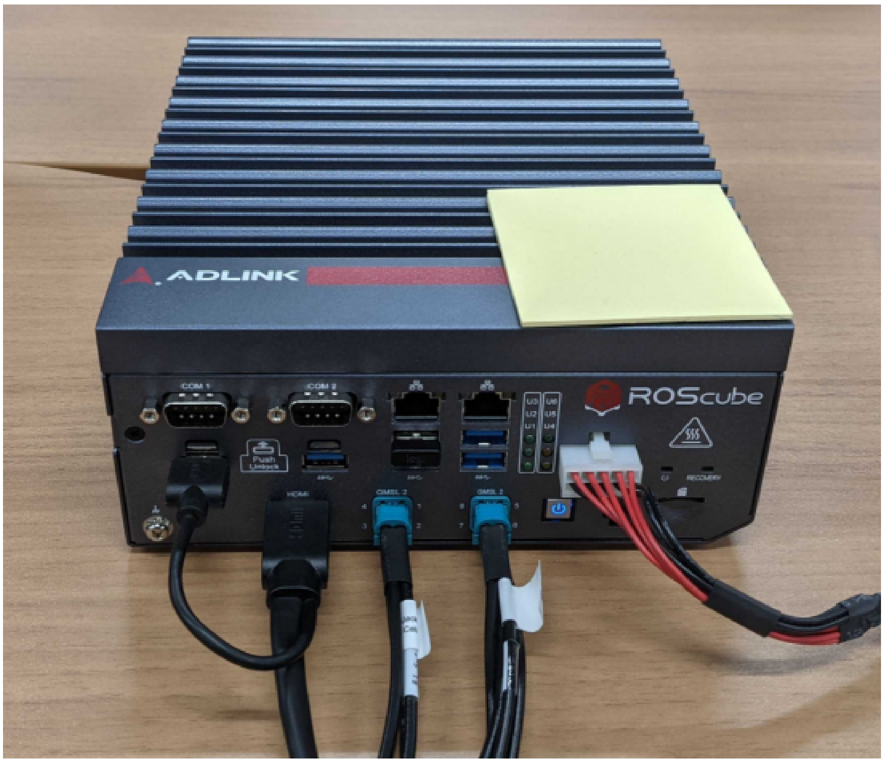

Fig. 1.1　RosCube

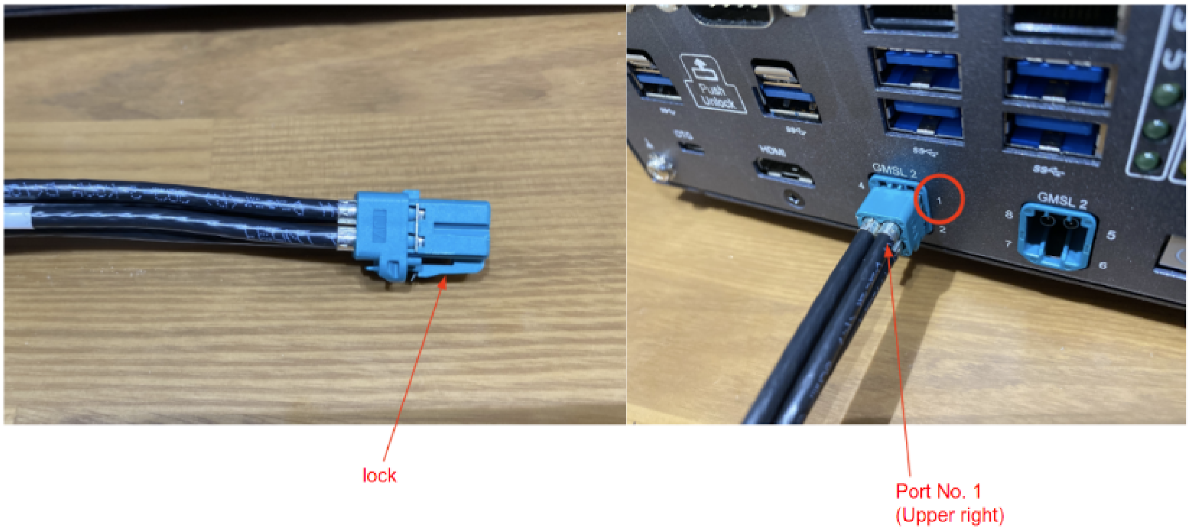

Fig. 1.2　Connector

1.2 Power on and check the camera image stream output

1.2.1 Power on

Power on the ROSCube-X and confirm that the power-on LED is turned on blue.

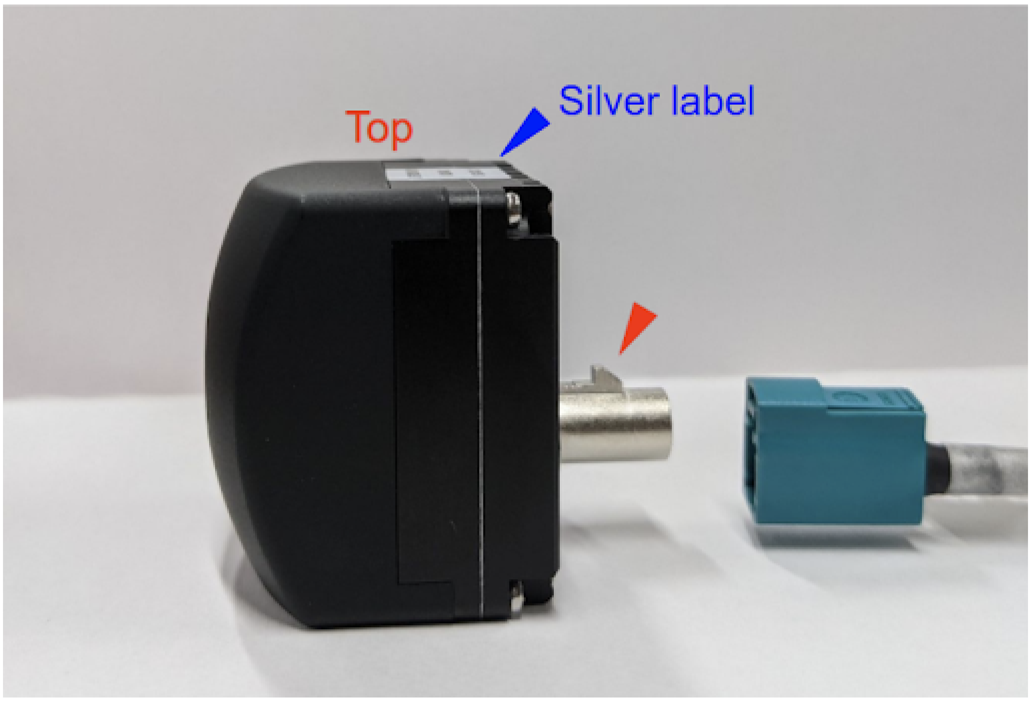

Fig. 1.3 Fakra cable insertion direction

1.2.2 Login

On the startup window, type the password to log in.

Default settings are shown below.

user	ros
password	adlinkros

1.2.3 Camera driver installation

Note: For C2, camera driver v.1.4.1 (or higher) is needed to set drive mode properly. Please upgrade the camera driver if needed.

If proper camera driver has already been installed, please skip this section.

Get the camera driver deb package from Github. The latest release has been confirmed to work with RQX-58G.

Copy the provided driver package file (`tier4-isx021-gmsl_*.*.*_arm64.deb`) to any directory in ROSCube-X (e.g. `~/c1_driver`). Then follow the command line instructions below:

1. Updating the apt package. It requires an internet connection. Then, install the driver by using the `apt install` command. Please make sure `*.dtbo` files are generated at `/boot`.

```
# Install
sudo apt update
sudo apt install make debhelper dkms
sudo apt install  ~/c1_driver/tier4-camera-gmsl_1.2.1_arm64.deb

```

(continues on next page)

(continued from previous page)

```
# Confirm  /boot/tier4-*.dtbo exists
ls /boot/*.dtbo
```

2. The command for the device tree overlay differs for the L4T version. To check the L4T version, execute `$ cat /etc/nv_tegra_release` and check the result. For example, If it returns `# R32 (release), REVISION: 5.1...`, the installed L4T version is `32.5.1`.

3. Choose appropriate instructions for the installed L4T version. Also, to assign the camera for each port, please refer to the camera driver’ s README page. The user needs to specify an appropriate overlay command for each user’ s configuration.

L4T 32.5.1

```
# For L4T 32.5.1
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n "TIERIV ISX021 GMSL2 Camera Device Tree Overlay"

# Confirm  /boot/kernel_tegra194-rqx-58g-tier4-isx021-gmsl2-camera-device-tree-overlay-roscube-r32_x.dtb has been generated
ls /boot/kernel_tegra194-rqx-58g-tier4-isx021-gmsl2-camera-device-tree-overlay*.dtb

# Then, shutdown the system
sudo shutdown -h now
```

L4T 32.6.1 or later

```
# For L4T 32.6.1
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 2="TIERIV ISX021 GMSL2 Camera Device Tree Overlay"

# Confirm  /boot/kernel_tegra194-rqx-58g-user-custom.dtb has been generated
ls /boot/kernel_tegra194-rqx-58g-user-custom.dtb

# Then, shutdown the system
sudo shutdown -h now
```

You can run `/opt/nvidia/jetson-io/config-by-hardware.py -l` to check available overlay options.

```
$ sudo /opt/nvidia/jetson-io/config-by-hardware.py -l
[sudo] password for ros:
Header 1 [default]: Jetson 40pin Header
No hardware configurations found!
Header 2: Jetson AGX Xavier CSI Connector
Available hardware modules:
1. TIERIV IMX490 GMSL2 Camera Device Tree Overlay
2. TIERIV ISX021 GMSL2 Camera Device Tree Overlay
3. TIERIV ISX021 IMX490 GMSL2 Camera Device Tree Overlay
```

For instance, to assign all GMSL ports to the C2 camera, the overlay command should be

```
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 2="TIERIV IMX490 GMSL2 Camera Device Tree Overlay"
```

To Assign GMSL ports for both C1 and C2, the overlay command should be

```
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 2="TIERIV ISX021 IMX490 GMSL2 Camera Device Tree Overlay"
```

In this case, ports 1, 2, 5, and 6 will be assigned to C1, and ports 3, 4, 7, and 8 will be assigned to C2. Please refer to the device driver GitHub repository for the details of the device overlay.

1.2.4 Check if the ROSCube-X recognizes cameras

Open a terminal window and type the following command. If it returns `/dev/videoX`, cameras are recognized properly as video devices.

```
ls /dev/video*
/dev/video0
/dev/video1
.
.
.
/dev/video7 # When 8 cameras are connected
```

1.2.5 Visualize camera output using GStreamer

Open a terminal window and type the following command. Gstreamer will launch and the camera image stream will appear on a new window (Fig. 4, Fig.5)

Case 1: 1 C1 camera is connected

```
# If cameras are running in slave mode, execute this i2cset command first
i2cset -f -y 2 0x66 0x04 0xff

# Start streaming
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! 'video/x-
↪raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! videoscale !↩
↪xvimagesink sync=false
```

Case 2: 8 C1 cameras are connected

```
# If cameras are running in slave mode, execute this i2cset command first
i2cset -f -y 2 0x66 0x04 0xff

# Start streaming
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false v4l2src io-mode=0 device=/dev/video1 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink↩
↪sync=false \
v4l2src io-mode=0 device=/dev/video2 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink↩
↪sync=false \
v4l2src io-mode=0 device=/dev/video3 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink↩
↪sync=false \
v4l2src io-mode=0 device=/dev/video4 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink↩
↪sync=false \
v4l2src io-mode=0 device=/dev/video5 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink↩
↪sync=false \
v4l2src io-mode=0 device=/dev/video6 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink↩
```

(continues on next page)

(continued from previous page)

```
↪sync=false \
v4l2src io-mode=0 device=/dev/video7 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
```

Case 3: 1 C2 camera is connected

```
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! 'video/x-
↪raw, width=2880, height=1860, framerate=30/1, format=UYVY' ! videoscale !
↪xvimagesink sync=false
```

Restriction regarding C2 camera

Currently, with ROSCube L4T 32.6.1, two C2 cameras cannot be connected to single deserializer board.

For example:

- Port 1 and 2 ··· NG
- Port 3 and 4 ··· NG
- Port 1 and port 3 ··· OK
- Port 1 and 5 ··· OK

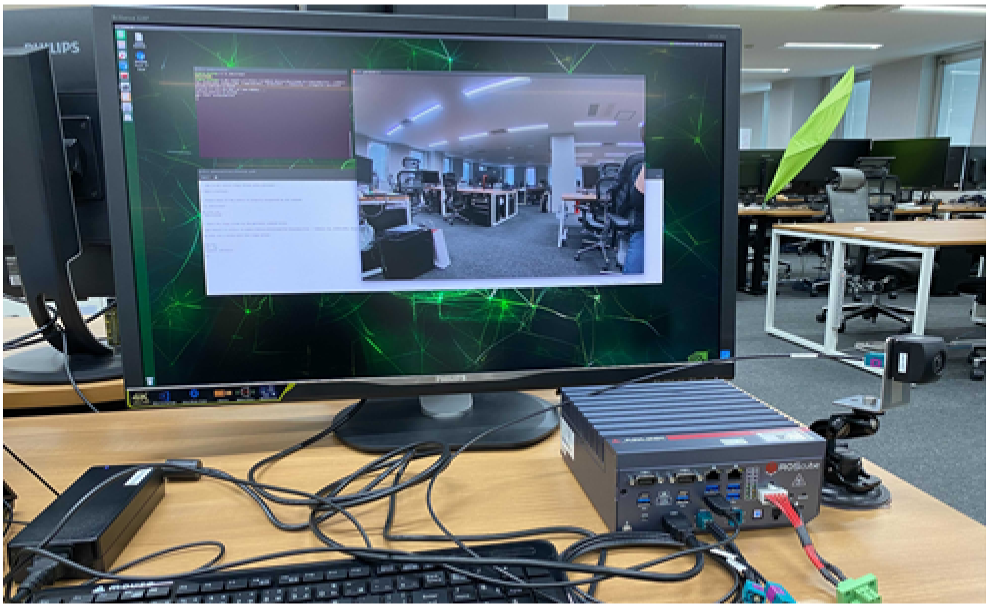

Fig. 1.4 Visualized camera output

Constraint on camera boot order (For driver version earlier than v1.1.1)

If you are using a driver version earlier than v1.1.1, please follow the specific instructions for booting. For driver versions v1.2.1 and later, there are no ordering or constraint requirements

To run multiple cameras, cameras need to be launched at once. For example, to run 2 cameras, the start-streaming command has to be a one liner

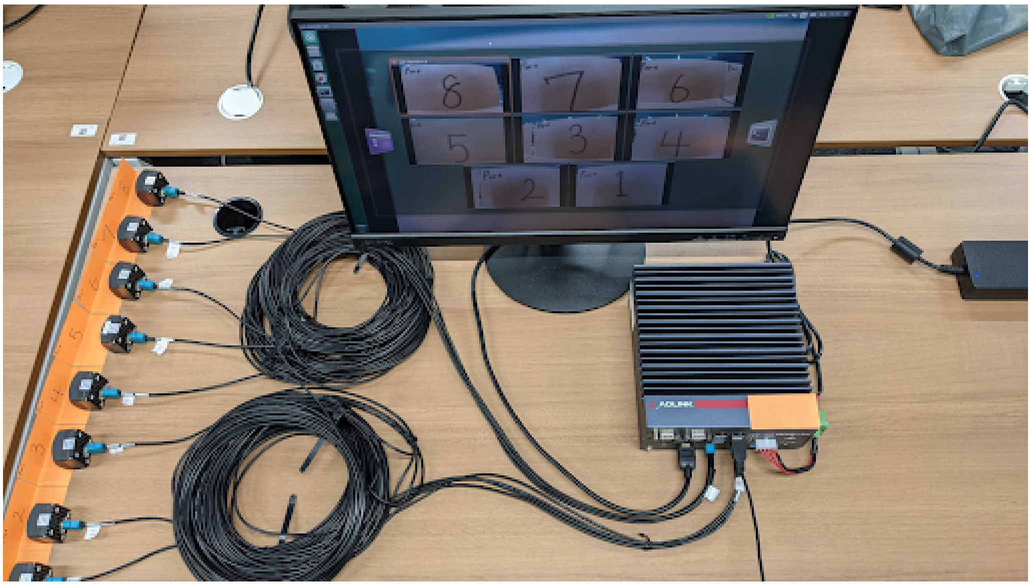

Fig. 1.5 Visualized camera output (x8 cameras)

```
# This one-liner command works
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false v4l2src io-mode=0 device=/dev/video1 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
```

These separate commands do not work

```
# These separated commands may not work

# On a terminal
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false

# On another terminal
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video1 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false
```

1.3 Record video data as rosbag (ROS format log data)

1.3.1 Install ROS melodic and gscam (ROS camera driver for GStreamer)

Install ROS melodic and gscam by following command line instructions.

Caution: The installation command may change. If the following commands do not work, please refer to the ROS official documentation.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
↪ /etc/apt/sources.list.d/ros-latest.list'
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-
↪key add -
sudo apt update
sudo apt install ros-melodic-desktop
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
sudo apt install -y ros-melodic-gscam
```

1.3.2 Run gscam and visualize the image topic

Run the gscam by following instructions.

1. Open a new terminal and execute the following command to start roscore

   ```
   roscore
   ```

2. Open another terminal and execute the following commands

   ```
   export GSCAM_CONFIG="v4l2src device=/dev/video0 ! video/x-raw,framerate=30/1 !
   ↪ videoconvert"
   rosrun gscam gscam
   ```

3. Now that a topic (ROS format message data) camera/image_raw is published, visualize the topic by rqt_image_view. To launch rqt_image_view, open another terminal and execute the following command.

   ```
   rosrun rqt_image_view rqt_image_view
   ```

   This command opens a window to visualize image topics. Select camera/image_raw by clicking the pull-down menu on the upper left of the window (Fig. 6)

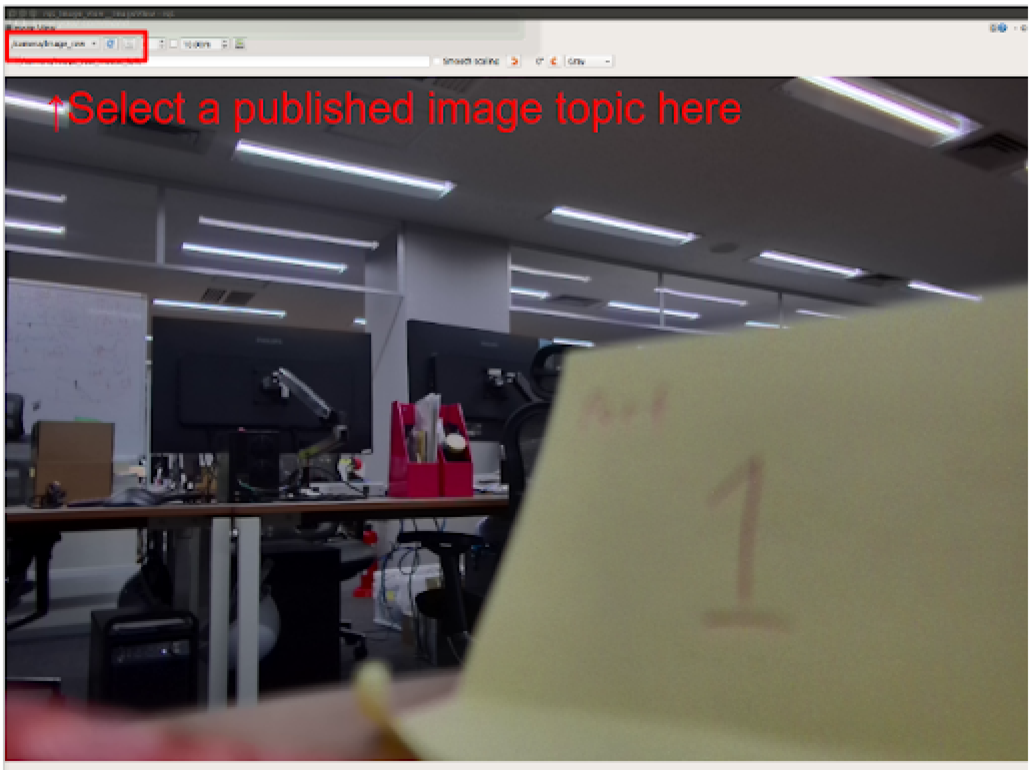

Fig. 1.6 Select a published image topic

To run gscam with multiple cameras, create a text file with the following XML message and save it as `~/2cameras.launch`. This .launch file is for streaming with 2 cameras.

```
<launch>
 <arg name="frame_rate" default="30"/>
 <node name="gscam_driver_v4l_port_1" pkg="gscam" type="gscam" output="screen" ns=
↪"port_1">
   <param name="camera_name" value="port_1"/>
   <param name="frame_id" value="port_1"/>
   <param name="camera_info_url" value="package://gscam/examples/uncalibrated_
↪parameters.ini"/>
   <param name="gscam_config" value="v4l2src device=/dev/video0 ! video/x-raw,
↪format=UYVY,width=1920,height=1280,framerate=30/1 ! videoconvert"/>
   <param name="sync_sink" value="true"/>
</node>
 <node name="gscam_driver_v4l_port_2" pkg="gscam" type="gscam" output="screen" ns=
↪"port_2">
   <param name="camera_name" value="port_2"/>
   <param name="frame_id" value="port_2"/>
   <param name="camera_info_url" value="package://gscam/examples/uncalibrated_
↪parameters.ini"/>
   <param name="gscam_config" value="v4l2src device=/dev/video1 ! video/x-raw,
↪format=UYVY,width=1920,height=1280,framerate=30/1 ! videoconvert"/>
   <param name="sync_sink" value="true"/>
</node>
</launch>
```

Caution: This is the setting for C1. For C2, please change the `width`, `height` and `framerate` if necessary.

Assuming that you have created `/home/ros/2cameras.launch`, execute the following command to run gscam with multiple cameras.

```
roslaunch /home/ros/2cameras.launch
```

Note: For more cameras, add nodes with appropriate node name, namespace(ns), camera_name, frame_id, and gscam_config.

1.3.3 Record a rosbag

While running gscam, you may record a rosbag (Log file in ROS format) by

```
rosbag record -a
```

Now you may also check the information of rosbag by rosbag info

```
rosbag info <recorded rosbag file name>
# example output
path:        2022-03-29-17-42-07.bag
version:     2.0
duration:    3.1s
start:       Mar 29 2022 17:42:07.97 (1648543327.97)
end:         Mar 29 2022 17:42:11.07 (1648543331.07)
size:        365.7 MB
messages:    113
compression: none [53/53 chunks]
```

(continues on next page)

(continued from previous page)

```
types:        rosgraph_msgs/Log       [acffd30cd6b6de30f120938c17c593fb]
              sensor_msgs/CameraInfo [c9a58c1b0b154e0e6da7578cb991d214]
              sensor_msgs/Image       [060021388200f6f0f447d0fcd9c64743]
topics:       /port_1/camera/camera_info   26 msgs    : sensor_msgs/CameraInfo
              /port_1/camera/image_raw     26 msgs    : sensor_msgs/Image
              /port_2/camera/camera_info   26 msgs    : sensor_msgs/CameraInfo
              /port_2/camera/image_raw     26 msgs    : sensor_msgs/Image
              /rosout                       8 msgs    : rosgraph_msgs/Log      (3␣
↪connections)
              /rosout_agg                   1 msg     : rosgraph_msgs/Log
```

1.4 Configuration

To change driver configuration, edit the configuration file /etc/modprobe.d/tier4-*.conf. (See following sections). The configuration file includes the line by default. (This is the case of C1)

```
options tier4_isx021 trigger_mode=0 enable_auto_exposure=1 enable_distortion_
↪correction=1
```

To apply the configurations, reboot ROSCube-X after editing.

1.4.1 Switch drive mode

Caution: To run the camera in slave mode, you need to input frame synchronization signal from ROSCube-X. please check the Shutter triggering over GMSL2 to drive cameras in slave mode

The case of C1

Camera drive mode can be switched from Master mode to Slave mode(and vice versa) by editing the configuration file.

- Master mode: Free running mode at 30fps (Default)
- Slave mode: Shutter triggering mode, Shutter timing, and the frame rate can be adjusted by the FSYNC signal frequency (configurable with ROSCube-X)

To switch modes, edit trigger_mode in /etc/modprobe.d/tier4-isx021.conf. (See following cases)

Drive mode	Frame rate	trigger_mode=
Master	30fps	0
Slave	Depends on the FSYNC input frequency	1

The case of C2

In the case of C2, the following modes can be selected by editing /etc/modprobe.d/tier4-imx490.conf. The following table shows the available settings.

Drive mode	Frame rate	trigger_mode=
Master	10fps	0
Slave	10fps	1
Master	20fps	2
Slave	20fps	3
Master	30fps	4
Slave	30fps	5

1.4.2 Enable/Disable Lens Distortion Correction (LDC)

Note: This setting is common for both C1 and C2.

To enable LDC, set the flag enable_distortion_correction=1 (default). To disable LDC, set the flag enable_distortion_correction=0.

1.4.3 Enable/Disable auto exposure (AE)

Note: This setting is common for both C1 and C2.

To enable AE, set the flag enable_auto_exposure=1 (default). To disable AE, set the flag enable_auto_exposure=0.

1.4.4 Setting exposure time

The case of C1

In the case of C1, users can specify three types of exposure time via variables named shutter_time_min, shutter_time_mid, and shutter_time_max. The actual exposure time of C1 transits these three (and linearly interpolated values) corresponding to the illuminance. The values for the variables should be specified in microseconds. Users who would like to fix exposure time under arbitral illuminance conditions can set the exact same value for these variables. For example, the following configuration in /etc/modprobe.d/tier4-isx021.conf fixes exposure time to 11 ms:

```
shutter_time_min=11000 shutter_time_mid=11000 shutter_time_max=11000
```

The case of C2

Likewise C1, C2 also has capability to set two types of exposure time via variables named `shutter_time_min` and `shutter_time_max`. The unit of the value for these variables is also microseconds. For example, the following configuration in `/etc/modprobe.d/tier4-imx490.conf` fixes exposure time to 11 ms:

```
shutter_time_min=11000 shutter_time_max=11000
```

1.5 Shutter triggering over GMSL2

Camera operation in slave mode requires triggering signals input (FSYNC input) over GMSL2. Please refer to ADLINK’ s documentation for how to configure the FSYNC parameters and GPIO settings.

1.5.1 Preparation to drive cameras in slave mode

Check the HW version of your ROSCube-X by executing `i2cget -f -y 2 0x66 0x01` on a terminal and follow the instructions below.

When i2cget -f -y 2 0x66 0x01 returns 0x23

Execute `i2cset -f -y 2 0x66 0x04 0xff` before start streaming. The system requires this preparation only once after its boot.

When i2cget -f -y 2 0x66 0x01 returns 0x21

No preparation is required.

1.5.2 Input FSYNC frequency for C1

You can input any frequency that is slower than 30fps.

1.5.3 Input FSYNC frequency for C2

For C2, the allowed FSYNC frequency depends on the drive modes. Please refer to the below list.

- In the case of 30fps mode (`trigger_mode=5`): 15 < f <= 30 fps
- In the case of 20fps mode (`trigger_mode=3`): 10 < f <= 20 fps
- In the case of 10fps mode (`trigger_mode=1`): 5 < f <= 10 fps

1.5.4 FPS check

To check the FPS setting with a visualized video, please execute the following commands.

- The camera is connected to only port 1

  ```
  gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true !
  ↪'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' !
  ↪fpsdisplaysink video-sink=xvimagesink sync=false
  ```

- Cameras are connected to ports 1 and 2

```
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true !
↪'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' !
↪xvimagesink sync=false v4l2src io-mode=0 device=/dev/video1 do-
↪timestamp=true ! 'video/x-raw, width=1920, height=1280, framerate=30/1,
↪format=UYVY' ! fpsdisplaysink video-sink=xvimagesink sync=false
```

2

TIER IV Cameras Getting Started Guide for ADLINK ROSCube RQX-59G

Attention: This document is for the users of ADLINK RQX-59G

2.1 Preparation

2.1.1 Required items

- ADLINK ROSCube-X RQX-59G
- BSP version 1.4.2
- JetPack5.1.2 (L4T R35.4.1)installed
- GMSL2 coaxial cable (FAKRA - mini FAKRA, 1:4)
- TIER IV Automotive HDR Camera C1 or C2

2.1.2 Camera connection

1. Firstly, make sure that the RQX-59G is turned off.
2. Connect cameras to the FAKRA connectors of the cable(single FAKRA connector side). Then insert the mini FAKRA connectors of the cable to the RQX-59G’ s GMSL2 ports.
3. Please make sure the direction of the lock clown on FAKRA connectors and mini FAKRA connectors is correct (Fig.1~3).

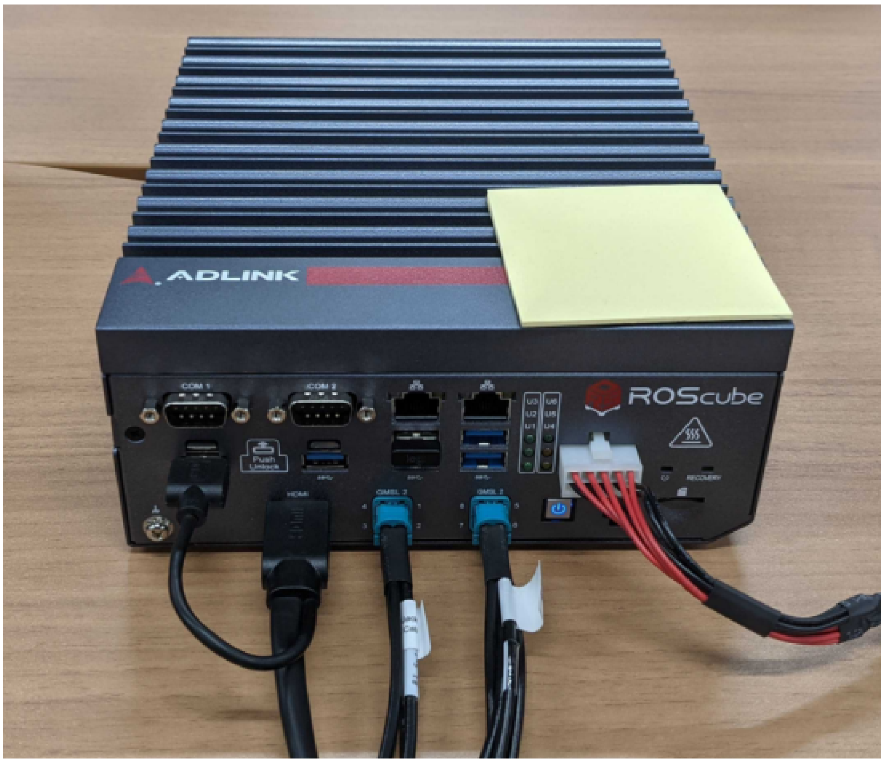

Fig. 2.1 RosCube RQX-59G

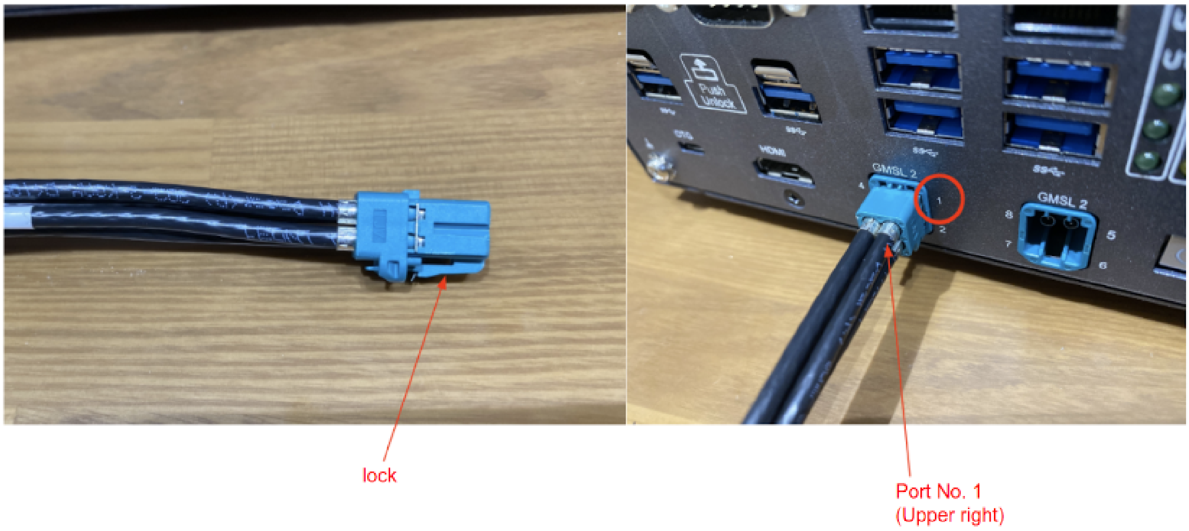

Fig. 2.2 Connector

2.2 Power on and check the camera image stream output

2.2.1 Power on

Power on the RQX-59G and confirm that the power-on LED is turned on blue.

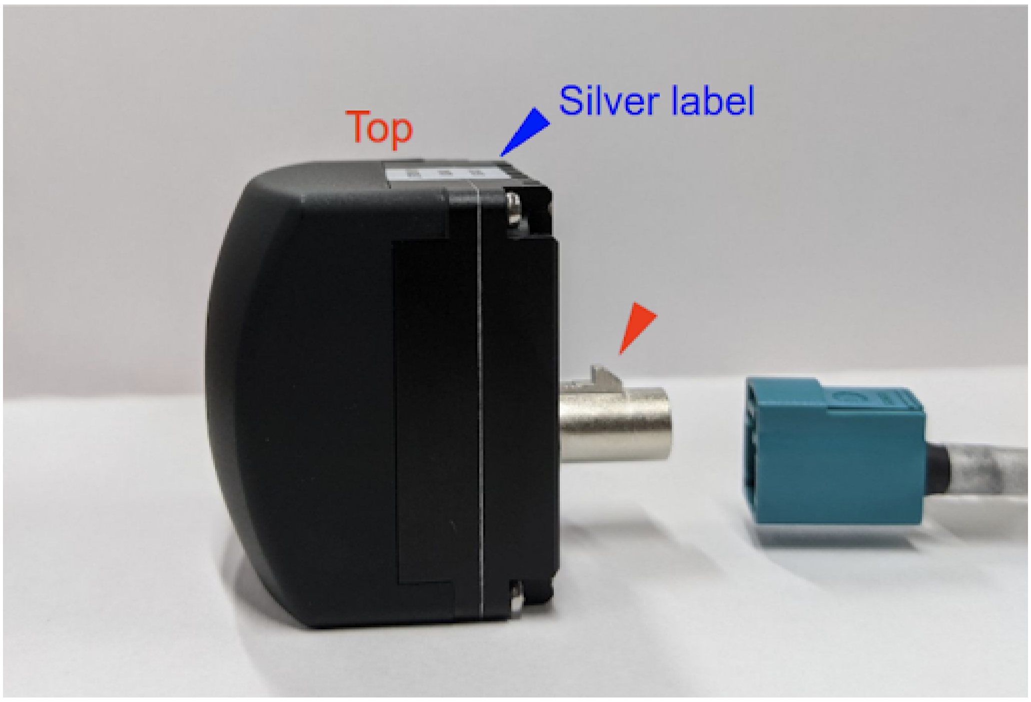

Fig. 2.3 Fakra cable insertion direction

2.2.2 Login

On the startup window, type the password to log in.

Default settings are shown below.

user	ros
password	adlinkros

2.2.3 Camera driver installation

Note: As of 2024-04-01, to use RQX-59G with TIER IV cameras, dedicated driver deb package is needed.

Get the camera driver deb package from Github.

Copy the provided driver package file (`tier4-camera-gmsl_1.4.2_59g_arm64.deb`) to any directory in RQX-59G (e.g. `~/driver`). Then follow the command line instructions below:

1. Updating the apt package. It requires an internet connection. Then, install the driver by using the `apt install` command. Please make sure `*.dtbo` files are generated at `/boot`.

```
# Install
sudo apt update
sudo apt install make debhelper dkms
sudo apt install  ~/driver/tier4-camera-gmsl_1.2.1_arm64.deb

# Confirm  /boot/tier4-*.dtbo exists
ls /boot/*.dtbo
/boot/tier4-imx490-gmsl-device-tree-overlay-roscube-orin-r354.dtbo
```

(continues on next page)

(continued from previous page)

```
/boot/tier4-isx021-gmsl-device-tree-overlay-roscube-orin-r354.dtbo
/boot/tier4-isx021-imx490-gmsl-device-tree-overlay-roscube-orin-r354.dtbo
```

2. To assign the camera for each port, you can use `configure-by-hardware.py` script. Please refer to the camera driver’ s README page for the details. The user needs to specify an appropriate overlay command for each user’ s configuration.

```
# For RQX-59G
# Please be noted that you need to specify "1=" (not "2=")
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 1="TIERIV ISX021 GMSL2⊠
↪Camera Device Tree Overlay"

# Confirm  kernel_tegra234-p3701-0004-rqx590-nosuspend-user-custom.dtb has⊠
↪been generated
ls /boot/*.dtb
/boot/kernel_tegra234-p3701-0004-rqx590-nosuspend.dtb
/boot/kernel_tegra234-p3701-0004-rqx590-nosuspend-user-custom.dtb

# Then, shutdown the system
sudo shutdown -h now
```

You can run `/opt/nvidia/jetson-io/config-by-hardware.py -l` to check available overlay options.

```
sudo /opt/nvidia/jetson-io/config-by-hardware.py -l
[sudo] password for ros:
Header 1 [default]: Jetson AGX CSI Connector
  Available hardware modules:
  1. TIERIV IMX490 GMSL2 Camera Device Tree Overlay
  2. TIERIV ISX021 GMSL2 Camera Device Tree Overlay
  3. TIERIV ISX021 IMX490 GMSL2 Camera Device Tree Overlay
```

For instance, to assign all GMSL ports to the C2 camera, the overlay command should be

```
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 1="TIERIV IMX490 GMSL2⊠
↪Camera Device Tree Overlay"
```

To Assign GMSL ports for both C1 and C2, the overlay command should be

```
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 1="TIERIV ISX021 IMX490⊠
↪GMSL2 Camera Device Tree Overlay"
```

In this case, ports 1, 2, 5, and 6 will be assigned to C1, and ports 3, 4, 7, and 8 will be assigned to C2.

Please refer to the device driver GitHub repository for the details of the device overlay.

2.2.4 Check if the RQX-59G recognizes cameras

Open a terminal window and type the following command. If it returns `/dev/videoX`, cameras are recognized properly as video devices.

```
ls /dev/video*
/dev/video0
/dev/video1
.
.
.
/dev/video7 # When 8 cameras are connected
```

2.2.5 Visualize camera output using GStreamer

Open a terminal window and type the following command. Gstreamer will launch and the camera image stream will appear on a new window （Fig. 4, Fig.5）.

Please also refer to GStreamer command examples for various GStreamer examples.

Case 1: 1 C1 camera is connected

```
# If cameras are running in slave mode, execute this i2cset command first
i2cset -f -y 2 0x66 0x04 0xff

# Start streaming
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! 'video/x-
↪raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! videoscale !
↪xvimagesink sync=false
```

Case 2: 8 C1 cameras are connected

```
# If cameras are running in slave mode, execute this i2cset command first
i2cset -f -y 2 0x66 0x04 0xff

# Start streaming
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false v4l2src io-mode=0 device=/dev/video1 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
v4l2src io-mode=0 device=/dev/video2 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
v4l2src io-mode=0 device=/dev/video3 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
v4l2src io-mode=0 device=/dev/video4 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
v4l2src io-mode=0 device=/dev/video5 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
v4l2src io-mode=0 device=/dev/video6 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
v4l2src io-mode=0 device=/dev/video7 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
```

Case 3: 1 C2 camera is connected

```
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! 'video/x-
↪raw, width=2880, height=1860, framerate=30/1, format=UYVY' ! videoscale !
↪xvimagesink sync=false
```

When you connect multiple C2 cameras

At RQX-59G, four deserializer are implemented for eight GMSL ports, and two GMSL port shares single deserializer and its bandwidth.

Even though we’ ve checked that two C2 cameras can work with one deserializer, it’ s best to use just one camera per deserializer, especially if you’ re connecting up to four cameras.

For example:

- Port 1 and 2 ··· Not recommended
- Port 3 and 4 ··· Not recommended
- Port 1 and port 3 ··· Recommended
- Port 1 and 5 ··· Recommended

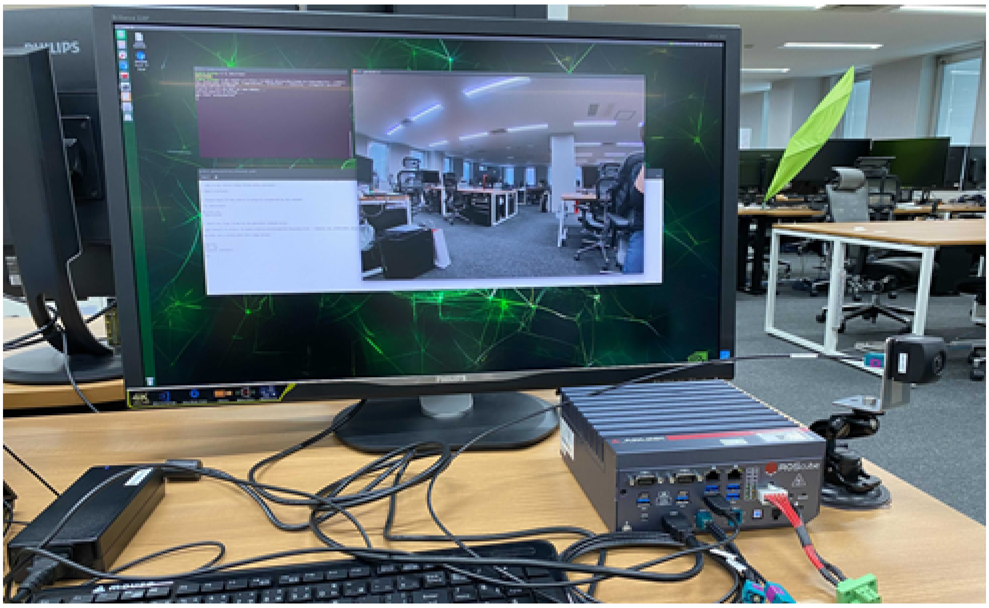

Fig. 2.4 Visualized camera output

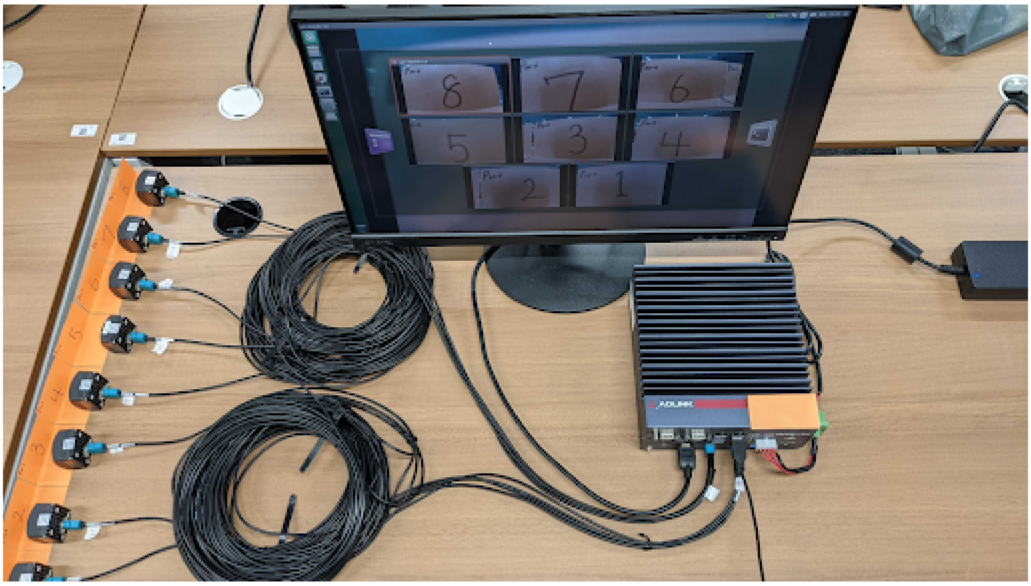

Fig. 2.5 Visualized camera output (x8 cameras)

2.3 Record video data as rosbag (ROS format log data)

2.3.1 Install ROS melodic and gscam (ROS camera driver for GStreamer)

Install ROS melodic and gscam by following command line instructions.

> **Caution:** The installation command may change. If the following commands do not work, please refer to the ROS official documentation.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
↪ /etc/apt/sources.list.d/ros-latest.list'
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-
↪key add -
sudo apt update
sudo apt install ros-melodic-desktop
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
sudo apt install -y ros-melodic-gscam
```

2.3.2 Run gscam and visualize the image topic

Run the gscam by following instructions.

1. Open a new terminal and execute the following command to start roscore

   ```
   roscore
   ```

2. Open another terminal and execute the following commands

   ```
   export GSCAM_CONFIG="v4l2src device=/dev/video0 ! video/x-raw,framerate=30/1 !
   ↪ videoconvert"
   rosrun gscam gscam
   ```

3. Now that a topic (ROS format message data) camera/image_raw is published, visualize the topic by rqt_image_view. To launch rqt_image_view, open another terminal and execute the following command.

   ```
   rosrun rqt_image_view rqt_image_view
   ```

   This command opens a window to visualize image topics. Select camera/image_raw by clicking the pull-down menu on the upper left of the window (Fig. 6)

To run gscam with multiple cameras, create a text file with the following XML message and save it as `~/2cameras.launch`. This .launch file is for streaming with 2 cameras.

```
<launch>
 <arg name="frame_rate" default="30"/>
 <node name="gscam_driver_v4l_port_1" pkg="gscam" type="gscam" output="screen" ns=
↪"port_1">
   <param name="camera_name" value="port_1"/>
   <param name="frame_id" value="port_1"/>
   <param name="camera_info_url" value="package://gscam/examples/uncalibrated_
↪parameters.ini"/>
   <param name="gscam_config" value="v4l2src device=/dev/video0 ! video/x-raw,
↪format=UYVY,width=1920,height=1280,framerate=30/1 ! videoconvert"/>
   <param name="sync_sink" value="true"/>
</node>
 <node name="gscam_driver_v4l_port_2" pkg="gscam" type="gscam" output="screen" ns=
```

(continues on next page)

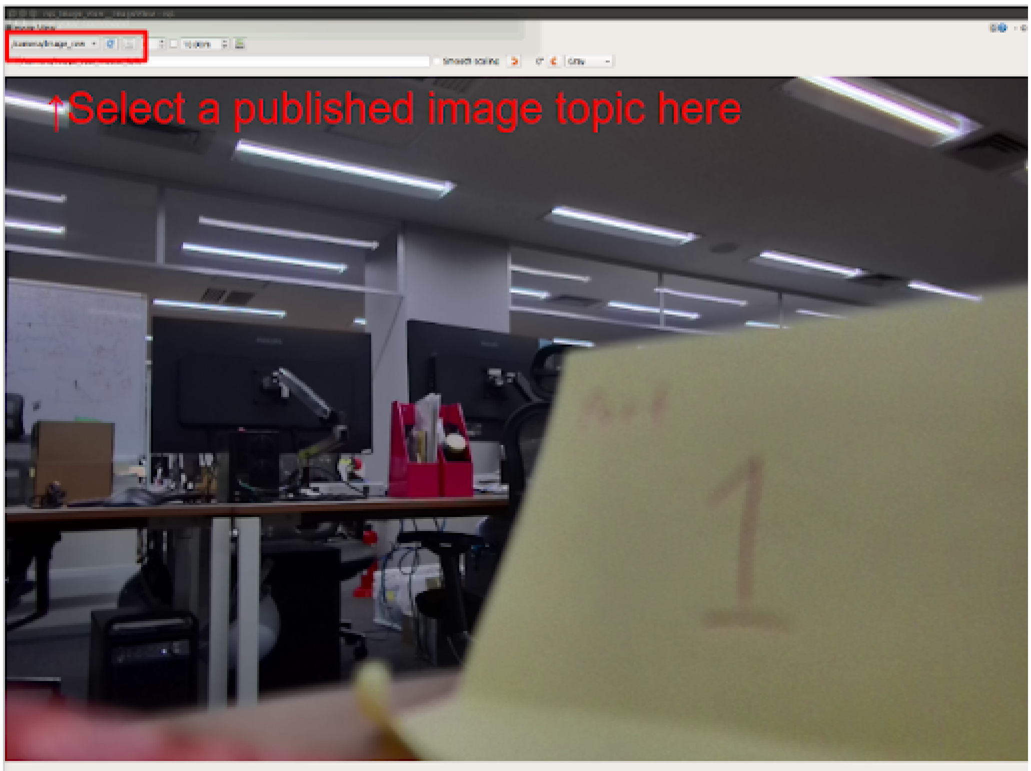

Fig. 2.6 Select a published image topic

(continued from previous page)

```
↪"port_2">
   <param name="camera_name" value="port_2"/>
   <param name="frame_id" value="port_2"/>
   <param name="camera_info_url" value="package://gscam/examples/uncalibrated_
↪parameters.ini"/>
   <param name="gscam_config" value="v4l2src device=/dev/video1 ! video/x-raw,
↪format=UYVY,width=1920,height=1280,framerate=30/1 ! videoconvert"/>
   <param name="sync_sink" value="true"/>
</node>
</launch>
```

Caution: This is the setting for C1. For C2, please change the `width`, `height` and `framerate` if necessary.

Assuming that you have created `/home/ros/2cameras.launch`, execute the following command to run gscam with multiple cameras.

```
roslaunch /home/ros/2cameras.launch
```

Note: For more cameras, add nodes with appropriate node name, namespace(ns), camera_name, frame_id, and gscam_config.

2.3.3 Record a rosbag

While running gscam, you may record a rosbag (Log file in ROS format) by

```
rosbag record -a
```

Now you may also check the information of rosbag by rosbag info

```
rosbag info <recorded rosbag file name>
# example output
path:        2022-03-29-17-42-07.bag
version:     2.0
duration:    3.1s
start:       Mar 29 2022 17:42:07.97 (1648543327.97)
end:         Mar 29 2022 17:42:11.07 (1648543331.07)
size:        365.7 MB
messages:    113
compression: none [53/53 chunks]
types:       rosgraph_msgs/Log      [acffd30cd6b6de30f120938c17c593fb]
             sensor_msgs/CameraInfo [c9a58c1b0b154e0e6da7578cb991d214]
             sensor_msgs/Image      [060021388200f6f0f447d0fcd9c64743]
topics:      /port_1/camera/camera_info   26 msgs    : sensor_msgs/CameraInfo
             /port_1/camera/image_raw     26 msgs    : sensor_msgs/Image
             /port_2/camera/camera_info   26 msgs    : sensor_msgs/CameraInfo
             /port_2/camera/image_raw     26 msgs    : sensor_msgs/Image
             /rosout                       8 msgs    : rosgraph_msgs/Log      (3
↪connections)
             /rosout_agg                   1 msg     : rosgraph_msgs/Log
```

2.4 Configuration

To change driver configuration, edit the configuration file /etc/modprobe.d/tier4-*.conf. (See following sections). The configuration file includes the line by default. (This is the case of C1)

```
options tier4_isx021 trigger_mode=0 enable_auto_exposure=1 enable_distortion_
↪correction=1
```

To apply the configurations, reboot RQX-59G after editing.

2.4.1 Switch drive mode

Caution: To run the camera in slave mode, you need to input frame synchronization signal from RQX-59G. please check the Shutter triggering over GMSL2 to drive cameras in slave mode.

Please refer to the ADLINK’ s documentation regarding trigger input how to generate the trigger signal from RQX-59G.

The case of C1

Camera drive mode can be switched from Master mode to Slave mode(and vice versa) by editing the configuration file.

- Master mode: Free running mode at 30fps (Default)
- Slave mode: Shutter triggering mode, Shutter timing, and the frame rate can be adjusted by the FSYNC signal frequency (configurable with RQX-59G)

To switch modes, edit trigger_mode in `/etc/modprobe.d/tier4-isx021.conf`. (See following cases)

Drive mode	Frame rate	trigger_mode=
Master	30fps	0
Slave	Depends on the FSYNC input frequency	1

The case of C2

In the case of C2, the following modes can be selected by editing `/etc/modprobe.d/tier4-imx490.conf`. The following table shows the available settings.

Drive mode	Frame rate	trigger_mode=
Master	10fps	0
Slave	10fps	1
Master	20fps	2
Slave	20fps	3
Master	30fps	4
Slave	30fps	5

2.4.2 Enable/Disable Lens Distortion Correction (LDC)

Note: This setting is common for both C1 and C2.

To enable LDC, set the flag `enable_distortion_correction=1` (default). To disable LDC, set the flag `enable_distortion_correction=0`.

2.4.3 Enable/Disable auto exposure (AE)

Note: This setting is common for both C1 and C2.

To enable AE, set the flag `enable_auto_exposure=1` (default). To disable AE, set the flag `enable_auto_exposure=0`.

2.4.4 Setting exposure time

The case of C1

In the case of C1, users can specify three types of exposure time via variables named `shutter_time_min`, `shutter_time_mid`, and `shutter_time_max`. The actual exposure time of C1 transits these three (and linearly interpolated values) corresponding to the illuminance. The values for the variables should be specified in microseconds. Users who would like to fix exposure time under arbitral illuminance conditions can set the exact same value for these variables. For example, the following configuration in `/etc/modprobe.d/tier4-isx021.conf` fixes exposure time to 11 ms:

```
shutter_time_min=11000 shutter_time_mid=11000 shutter_time_max=11000
```

The case of C2

Likewise C1, C2 also has capability to set two types of exposure time via variables named `shutter_time_min` and `shutter_time_max`. The unit of the value for these variables is also microseconds. For example, the following configuration in `/etc/modprobe.d/tier4-imx490.conf` fixes exposure time to 11 ms:

```
shutter_time_min=11000 shutter_time_max=11000
```

2.5 Shutter triggering over GMSL2

Camera operation in slave mode requires triggering signals input (FSYNC input) over GMSL2. Please refer to ADLINK’ s documentation for how to configure the FSYNC parameters and GPIO settings.

2.5.1 Preparation to drive cameras in slave mode

Check the HW version of your RQX-59G by executing `i2cget -f -y 2 0x66 0x01` on a terminal and follow the instructions below.

Please confirm that returned value is at least `0x24`.

```
i2cget -f -y 2 0x66 0x01
0x24
```

2.5.2 Input FSYNC frequency for C1

You can input any frequency that is slower than 30fps.

2.5.3 Input FSYNC frequency for C2

For C2, the allowed FSYNC frequency depends on the drive modes. Please refer to the below list.

- In the case of 30fps mode (`trigger_mode=5`): 15 < f <= 30 fps
- In the case of 20fps mode (`trigger_mode=3`): 10 < f <= 20 fps
- In the case of 10fps mode (`trigger_mode=1`): 5 < f <= 10 fps

2.5.4 FPS check

To check the FPS setting with a visualized video, please execute the following commands.

- The camera is connected to only port 1

```
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true !
↪'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' !
↪fpsdisplaysink video-sink=xvimagesink sync=false
```

- Cameras are connected to ports 1 and 2

```
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true !
↪'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' !
↪xvimagesink sync=false v4l2src io-mode=0 device=/dev/video1 do-
↪timestamp=true ! 'video/x-raw, width=1920, height=1280, framerate=30/1,
↪format=UYVY' ! fpsdisplaysink video-sink=xvimagesink sync=false
```

2.6 Reference

- ADLINK’ s documentation

3

TIER IV Cameras Getting Started Guide for Nvidia Jetson AGX Orin/Xavier Developer Kit

Attention: This document is for the users of NVIDIA Jetson devkit Xavier/Orin

3.1 List of Required Equipment

- Nvidia Jetson AGX Orin Developer Kit or Nvidia Jetson AGX Xavier Developer Kit x 1
- Deserializer kit
  - LI-JXAV-MIPI-ADPT-4CAM x 1
  - LI-GMSL2-IPX-DESER x 1~4 (1 per 2cameras)
  - FAW-1233-03 x 1~4 (1 per 2cameras)
- GMSL2 Cable(FAKRA C-FAKRA C) x 1~8 (1 per 1 cameras)
- TIER IV AUTOMOTIVE HDR CAMERA C1 x 1~6 or C2

3.2 Hardware connection

Make sure the power of the Jetson AGX Orin Developer Kit (hereinafter referred to as Devkit) is turned off. After confirming, attach the Deserializer Kit and cameras as shown below. Refer to the following steps and connect each component accordingly.

```
C1 Camera <== GMSL2 cable(FAKRA Connector) ==> LI-GMSL2-IPX-DESER <== FAW-1233-03 ==> MIPI-ADPT-4CAM <=> Jetson
AGX Orin Developer Kit
```

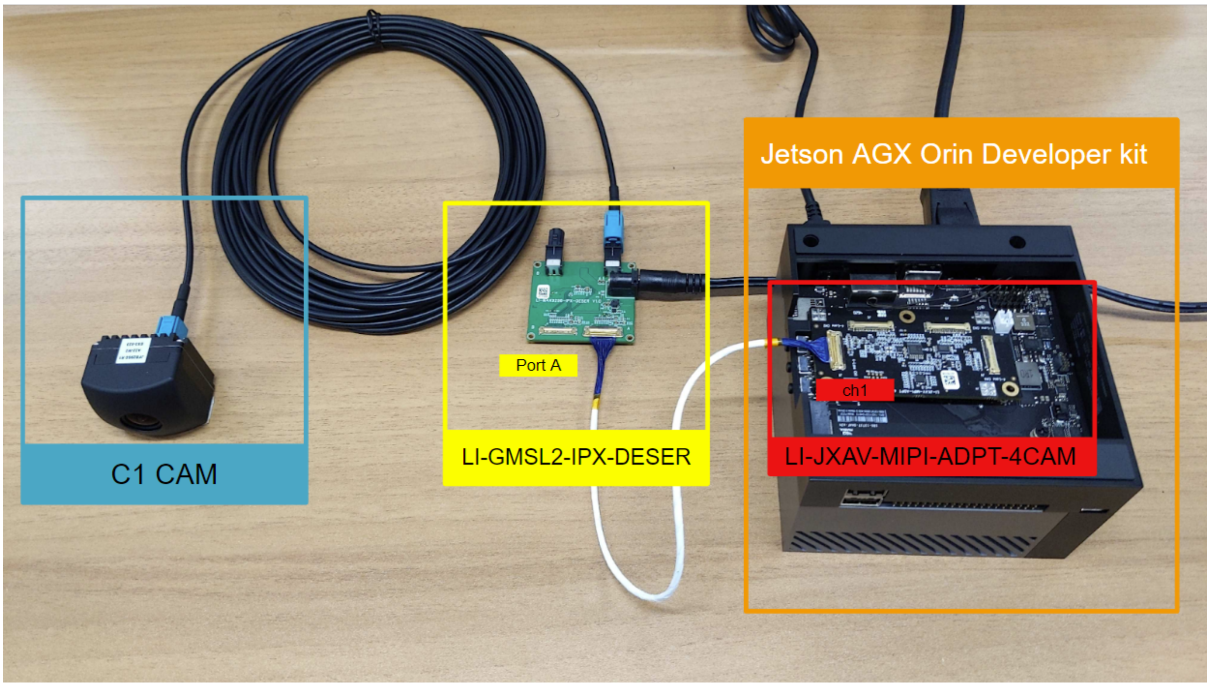

Fig. 3.1 Connection Examples for Each Component (Orin Devkit, 1 Camera)

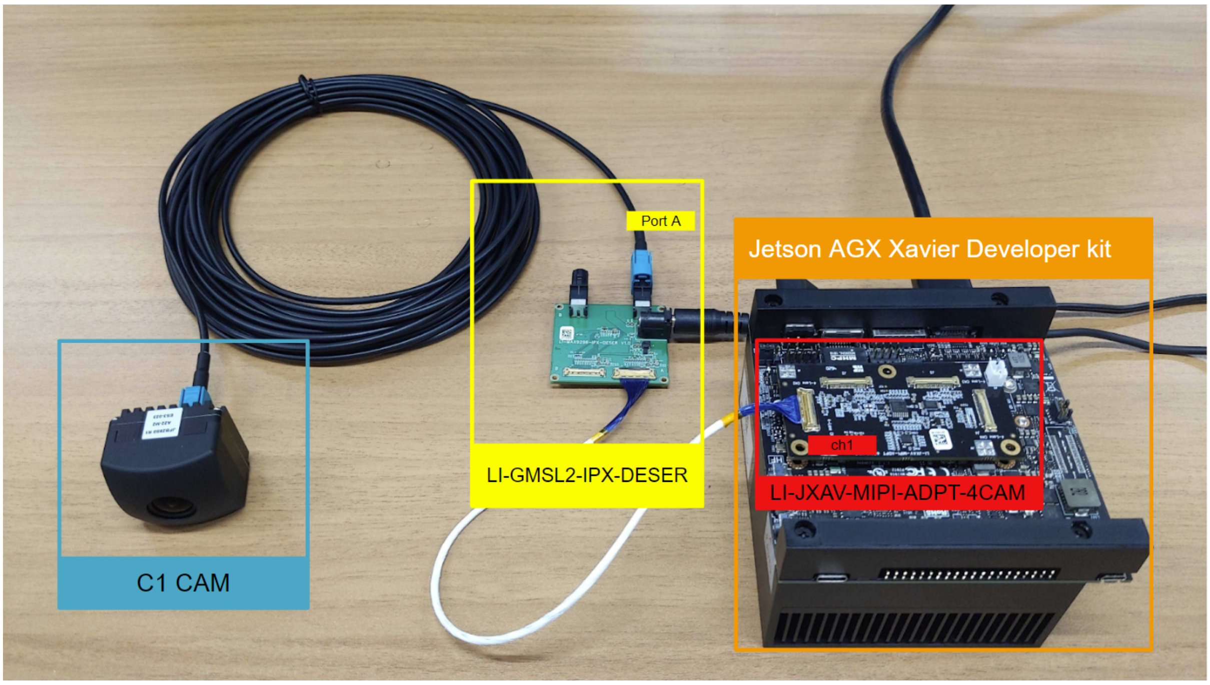

Fig. 3.2 Connection Examples for Each Component (Xavier Devkit, 1 Camera)

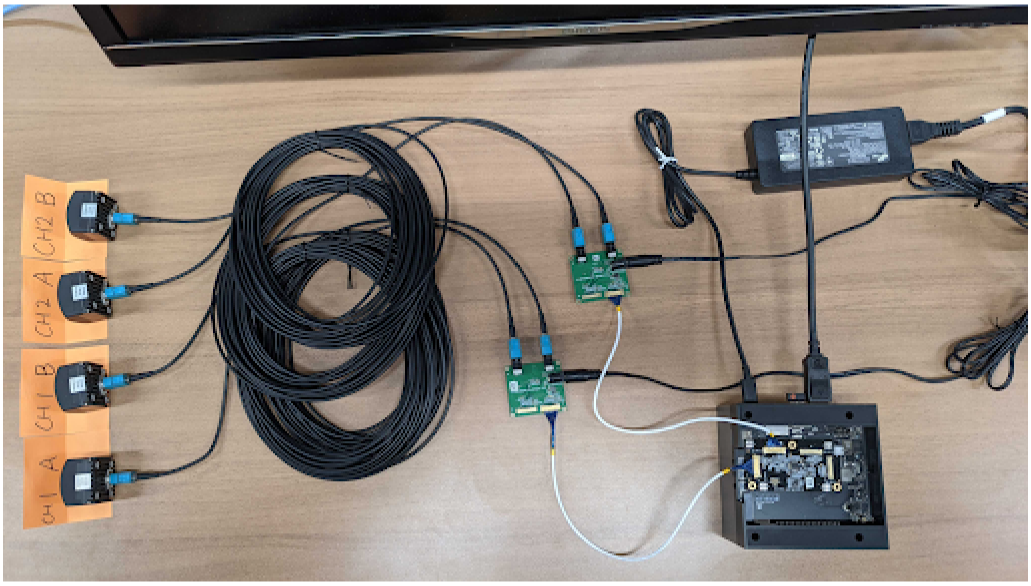

Fig. 3.3 Example of the cable connection (4 cameras)

3.2.1 Connection of Deserializer Kit and Camera

Connect the LI-JXAV-MIPI-ADPT-4CAM to the connector on the rear side of the Devkit.

Note: The LI-JXAV-MIPI-ADPT-4CAM is designed for the Jetson AGX Xavier Developer Kit, so there may be slight interference with the enclosure when connecting it to the Jetson AGX Orin Developer Kit. Please use it with caution, or consider using the adapter mentioned in the link to prevent any issues

Next, insert the FAW-1233-03 cable into CH1 of the LI-MAX9296-IPX-DESER.

Insert the other end of the previously connected cable into Port A of the LI-MAX9296-IPX-DESER. For single-camera evaluation, insert the FAKRA connector of the GMSL2 cable into Port A of the LI-MAX9296-IPX-DESER. For connecting two cameras, connect the FAKRA connector to Port A and Port B.

Attention: Please be noticed that you need to supply 12V for deserializer from external AC adapter, separately.

Connect the camera to the other FAKRA connector of the GMSL2 cable. Ensure that the side with the silver label is facing upwards.

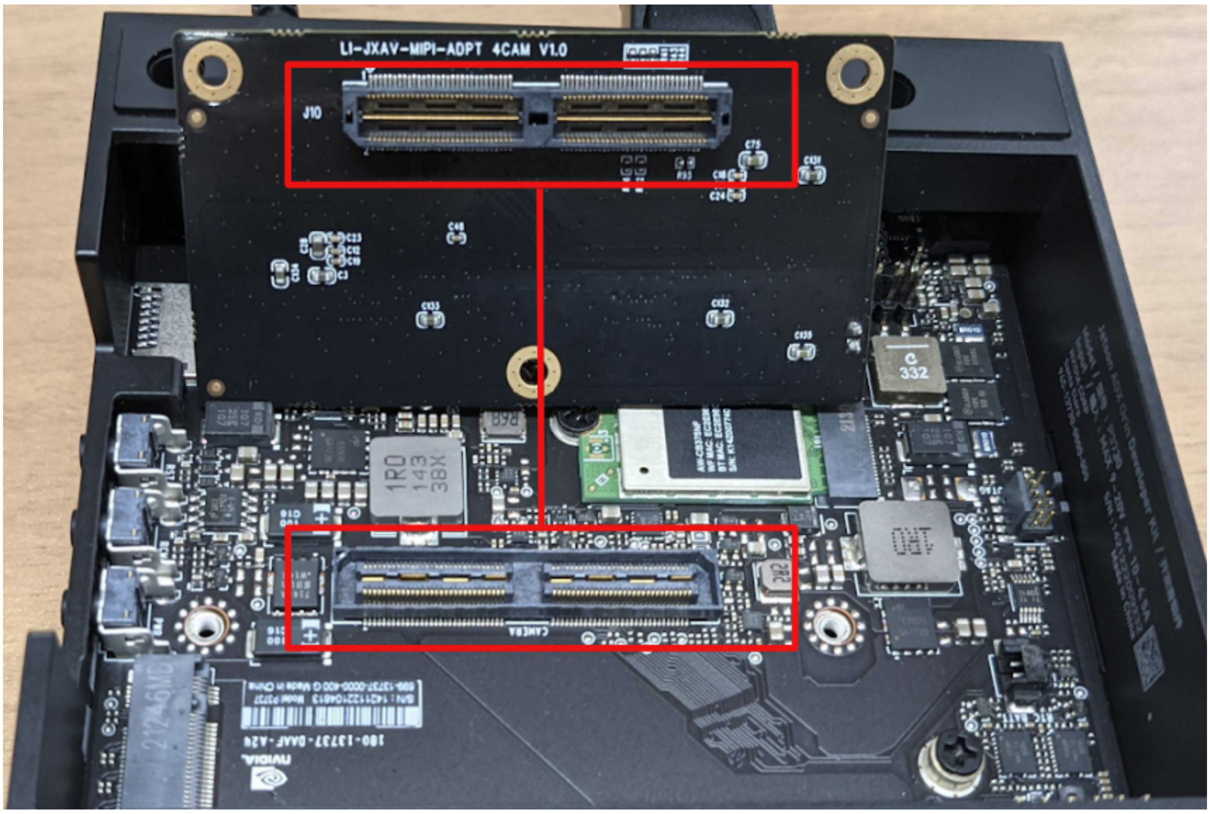

Fig. 3.4 LI-JXAV-MIPI-ADPT-4CAM connector

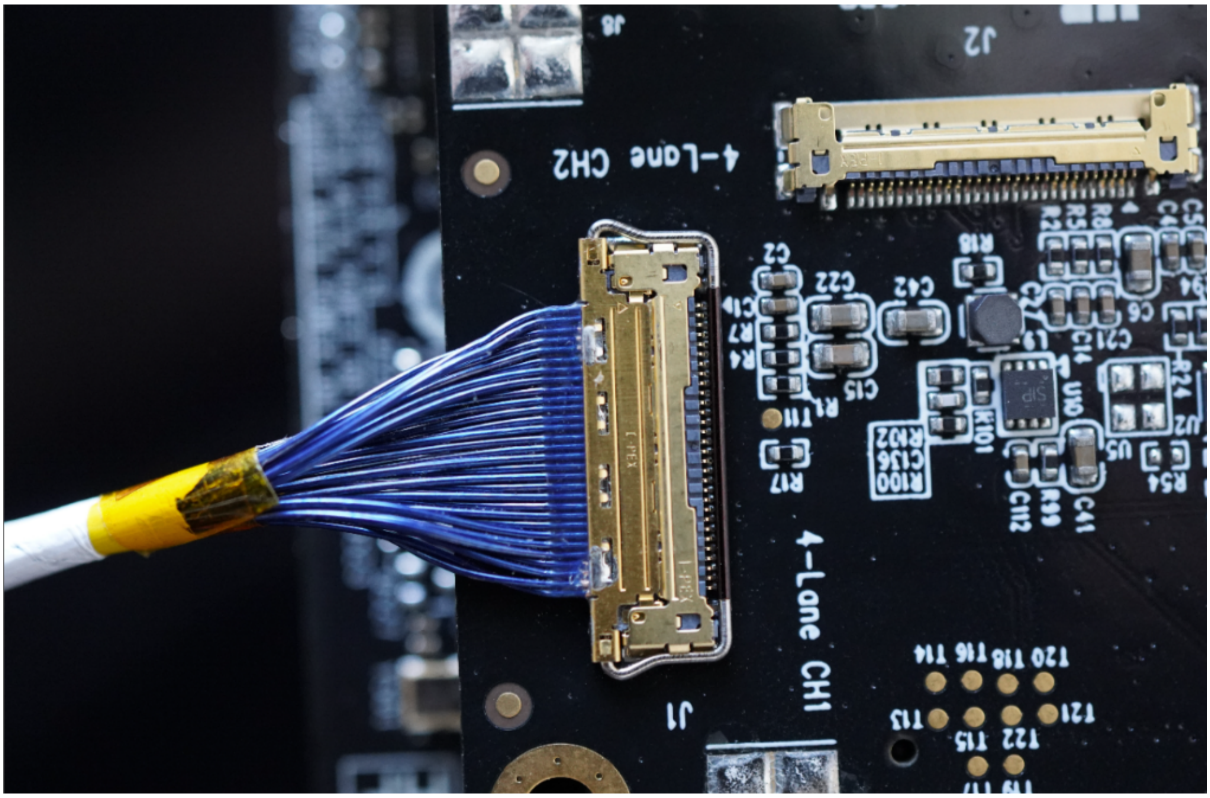

Fig. 3.5 Example of FAW-1233-03 Cable Connection

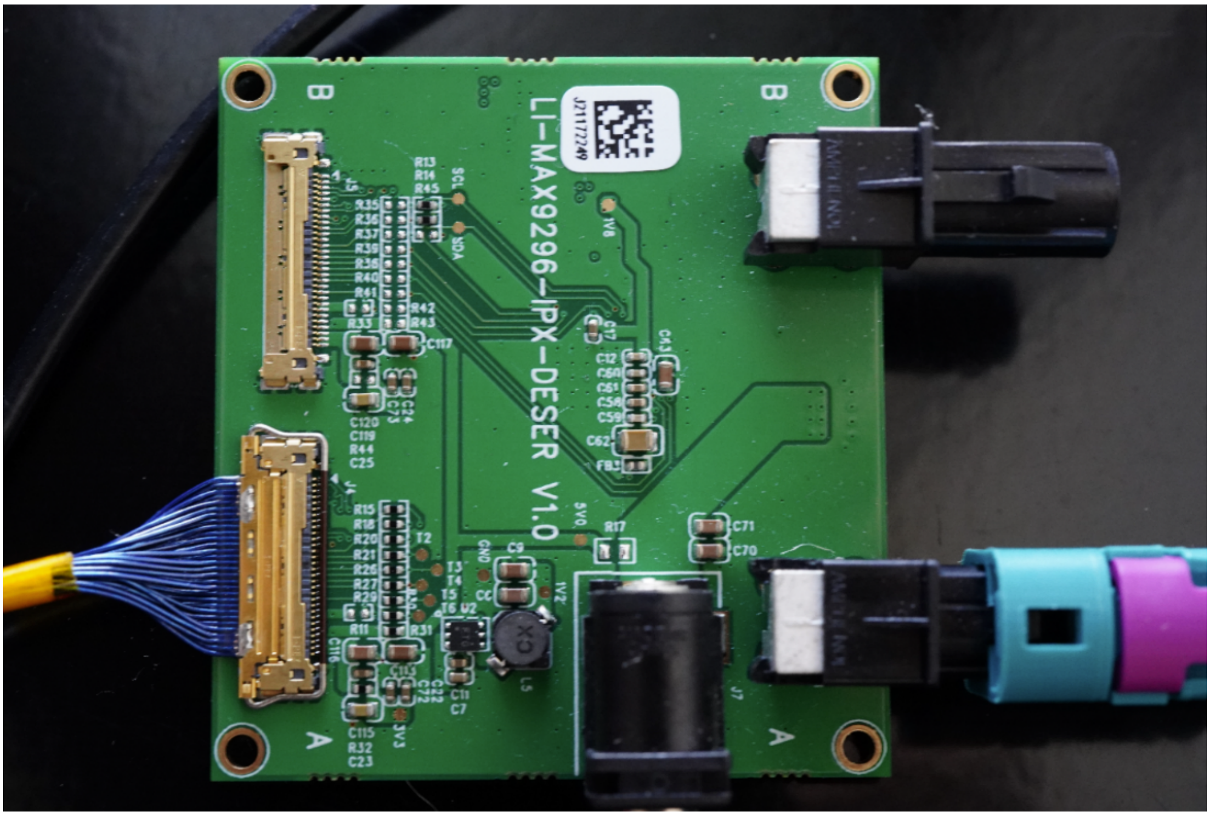

Fig. 3.6 Example of LI-MAX9296-IPX-DESER Cable Connection

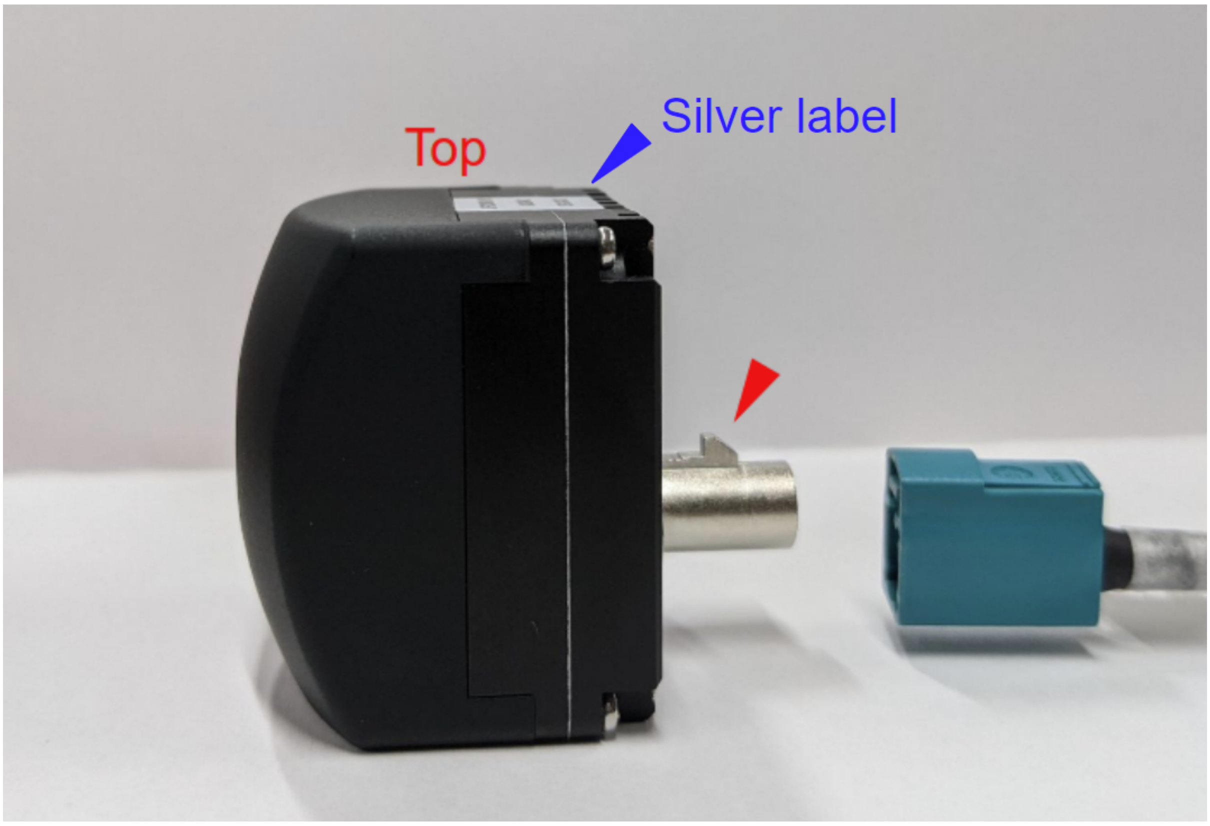

Fig. 3.7 Camera Orientation and Cable Insertion Direction (Silver Label Side Up)

3.2.2 Other Hardware Connections

If necessary, connect a mouse, keyboard, HDMI cable, etc., to the Devkit.

3.3 Software Setup

3.3.1 Software Requirements

Please verify if your Devkit meets the software requirements for the following drivers:

Software	Version	Download Link
JetPack	5.1.1	Nvidia Jetson quickstart guide
tier4-gmsl-camera	1.2.1 or higher	TIER IV Camera driver repository

Caution: For C2, you need to update the camera driver to 1.4.1 (or higher)

If your target Devkit does not meet the software requirements, please proceed with the installation of JetPack. There are two methods available for JetPack installation:

1. Manual Installation: Download the JetPack data from the provided link and perform the installation manually. In this case, you should install L4T 35.2.1.
2. Nvidia SDK Manager: Use the Nvidia SDK Manager for installation.

We recommend using the SDK Manager (method 2) for installation to ensure a smoother installation process and avoid potential installation problems.

Note: Nvidia SDK Manager cannot be used to install JetPack if you have Ubuntu 22.04.

3.3.2 Driver Package Installation

This step should be performed on the Devkit.

Copy the driver package file (obtain the latest version from the provided link) to any directory of your choice. After the copy is complete, enter the following command: ※ Replace x.x.x with the driver version. Specify the version of the downloaded driver.

```
cd <Directory of your choice>
sudo apt install ./tier4-camera-gmsl_x.x.x_arm64.deb
```

After installing the driver, you need to generate a custom dtb file to define the assignment of cameras for each GMSL port. Please refer camera driver installation for the details. You also can refer to the getting started guide for ADLINK ROSCube for some examples of overlay the commands.

```
# This is the case that all GMSL ports are assigned as C1
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 2="TIERIV ISX021 GMSL2 Camera⊠
↪Device Tree Overlay"
sudo shutdown -h now # Power off
```

After completing the process, please restart the Devkit.

After installing the drivers, you may encounter error messages related to i2c write errors or similar issues in the kernel messages (syslog or dmesg command output). These errors are typically related to ports where cameras are not connected, and you can proceed with camera image acquisition without any problems.

Example error message:

```
Jun 29 17:13:53 jetson-desktop kernel: [   11.114217] tier4_max9295 30-0062:⊠
↪[tier4_max9295_write_reg] : Max9295 I2C write failed.    Reg Address = 0x0000 ⊠
↪Data= 0xC0.
Jun 29 17:13:53 jetson-desktop kernel: [   11.115551] tier4_max9295 30-0060:⊠
↪[tier4_max9295_write_reg] : Max9295 I2C write failed.    Reg Address = 0x0010 ⊠
↪Data= 0x22.
Jun 29 17:13:53 jetson-desktop kernel: [   11.116391] tier4_max9295 30-0060:⊠
↪[tier4_max9295_setup_control]: Ser device not found
Jun 29 17:13:53 jetson-desktop kernel: [   11.116617] tier4_isx021 30-001c: [tier4_
↪isx021_gmsl_serdes_setup] : Failed to setup GMSL serializer.
Jun 29 17:13:53 jetson-desktop kernel: [   11.116873] tier4_isx021 30-001c: [tier4_
↪isx021_probe] : Failed GMSL Serdes setup.
Jun 29 17:13:53 jetson-desktop kernel: [   11.117134] tier4_isx021: probe of 30-
↪001c failed with error -121
Jun 29 17:13:53 jetson-desktop kernel: [   11.117340] tier4_isx021 30-001b: [tier4_
↪isx021_probe] : Probing V4L2 Sensor.
Jun 29 17:13:53 jetson-desktop kernel: [   11.117666] debugfs: Directory 'isx021_a
↪' with parent '/' already present!
Jun 29 17:13:53 jetson-desktop kernel: [   11.117871] tier4_isx021 30-001b:⊠
↪tegracam sensor driver:isx021_v2.0.6
Jun 29 17:13:53 jetson-desktop kernel: [   11.226016] tier4_max9295 30-0062:⊠
↪[tier4_max9295_write_reg] : Max9295 I2C write failed.    Reg Address = 0x0000 ⊠
↪Data= 0x84.
Jun 29 17:13:53 jetson-desktop kernel: [   11.227182] tier4_max9295 30-0042:⊠
↪[tier4_max9295_write_reg] : Max9295 I2C write failed.    Reg Address = 0x0010 ⊠
↪Data= 0x21.
Jun 29 17:13:53 jetson-desktop kernel: [   11.228005] tier4_max9295 30-0042:⊠
↪[tier4_max9295_setup_control]: Ser device not found
Jun 29 17:13:53 jetson-desktop kernel: [   11.228218] tier4_isx021 30-001b: [tier4_
↪isx021_gmsl_serdes_setup] : Failed to setup GMSL serializer.
Jun 29 17:13:53 jetson-desktop kernel: [   11.228465] tier4_isx021 30-001b: [tier4_
↪isx021_probe] : Failed GMSL Serdes setup.
Jun 29 17:13:53 jetson-desktop kernel: [   11.228717] tier4_isx021: probe of 30-
↪001b failed with error -121
```

When using NVMe storage (M.2 SSD)

Please be aware that if you have installed the operating system on the NVMe storage, the system may be reading the configuration from the onboard storage. If the camera devices are not visible even after restarting, in addition to the setup instructions provided earlier, you may need to mount the onboard storage and copy the following files to the /boot directory on the onboard storage:

```
- /boot/extlinux/extlinux.conf
- /boot/kernel_tegra234-p3701-0000-p3737-0000-user-custom.dtb
```

3.4 Camera Image Acquisition

3.4.1 Confirm the Camera Devices

Open a terminal and run the following command to verify if the cameras are recognized as v4l2 devices. If the number of video devices matches the number of connected cameras, the camera recognition check is complete.

```
ls /dev/video*
/dev/video0
/dev/video1
.
.
```

3.4.2 Camera Image Data Acquisition

Open a terminal and execute the following command to start image acquisition using Gstreamer. A new window will open, displaying the camera image.

Example 1: Evaluation with a C1 camera

```
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! 'video/x-
↪raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! videoscale !
↪xvimagesink sync=false
```

Example 2: Evaluation with four C1 cameras

```
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false v4l2src io-mode=0 device=/dev/video1 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
v4l2src io-mode=0 device=/dev/video2 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
v4l2src io-mode=0 device=/dev/video3 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false
```

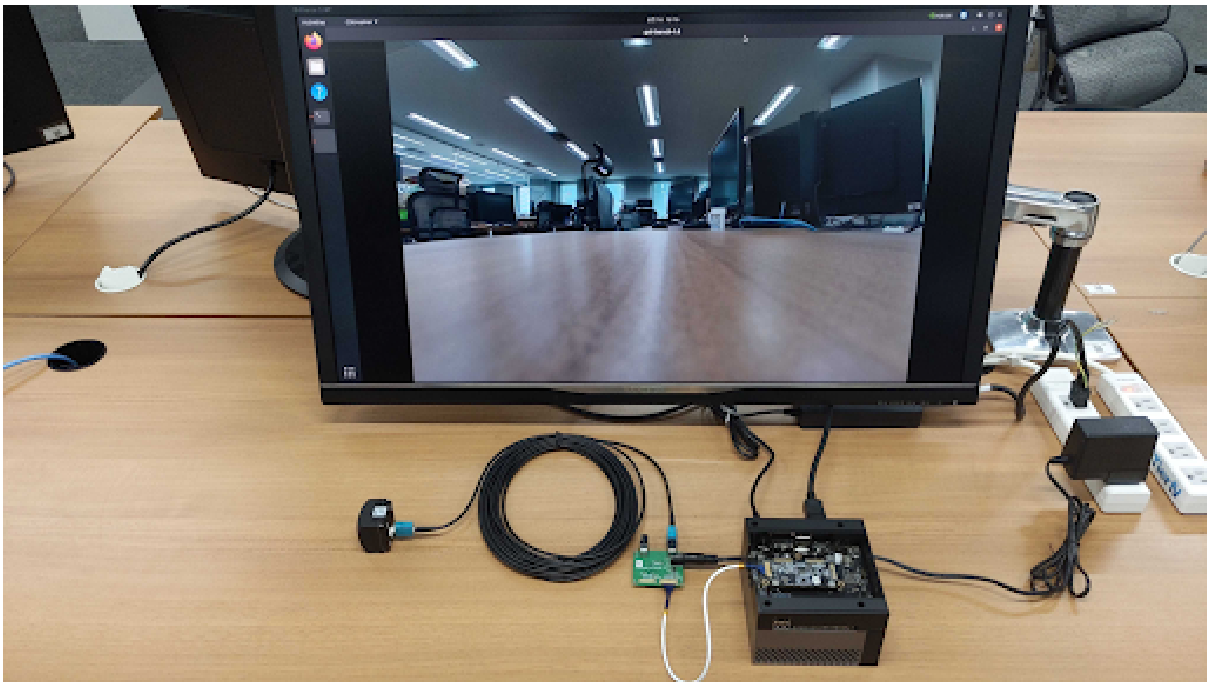

Fig. 3.8 Example of Camera Image Data Acquisition

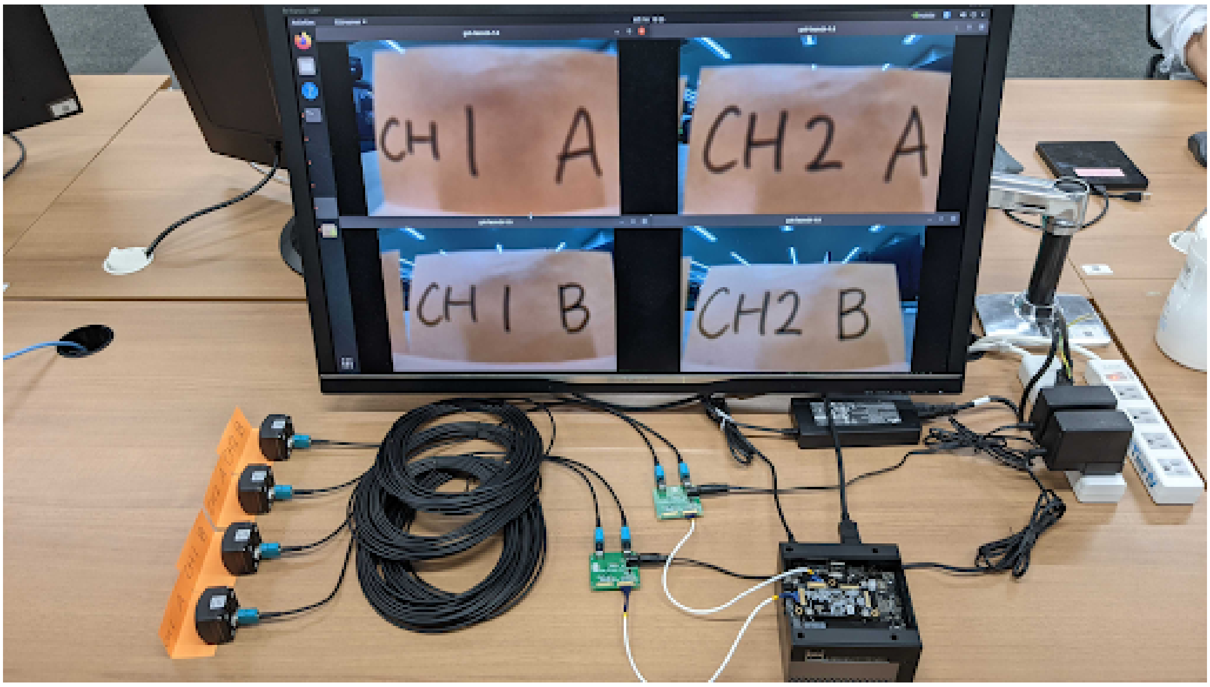

Fig. 3.9 Example of Camera Image Data Acquisition (Four Cameras)

Example 3: Evaluation with a C2 camera

```
gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! 'video/x-
↪raw, width=2880, height=1860, framerate=30/1, format=UYVY' ! videoscale !
↪xvimagesink sync=false
```

Note: You can also utilize the gstreamer launch script.

Note: If you encounter the error message 'no element "v4l2src"' , please delete the directory ~/.cache/gstreamer-1.0 and try the command again.

Constraint on camera boot order (For driver version earlier than v1.1.1)

If you are using a driver version earlier than v1.1.1, please follow the specific instructions for booting. For driver versions v1.2.1 and later, there are no ordering or constraint requirements

When using multiple cameras, all the cameras need to be started simultaneously.

For example, if you are using two cameras, you need to start image acquisition with the following one-liner command:

~~~
```
# This one-liner command works
tier4@jetson:~$ gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-
↪timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false v4l2src io-mode=0 device=/dev/video1 do-timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! xvimagesink
↪sync=false \
```
~~~

These separated commands may not work:

~~~
```
# These separated commands may not work

# On a terminal
tier4@jetson:~$ gst-launch-1.0 v4l2src io-mode=0 device=/dev/video0 do-
↪timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false

# On another terminal
tier4@jetson:~$ gst-launch-1.0 v4l2src io-mode=0 device=/dev/video1 do-
↪timestamp=true ! \
'video/x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! \
xvimagesink sync=false

```
~~~

3.5 Recording Image Data

You can use Gstreamer to record the image data while encoding it in H.265. Please modify the bitrate, file name, and other parameters according to your needs.

Example 1: Recording with a C1 camera

```
gst-launch-1.0 -e v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! 'video/
↪x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! nvvidconv !
↪nvv4l2h265enc bitrate=3000000 ! h265parse ! qtmux ! filesink location=out.mp4
```

Example 2: Recording with two C2 cameras

```
gst-launch-1.0 -e v4l2src io-mode=0 device=/dev/video0 do-timestamp=true ! 'video/
↪x-raw, width=1920, height=1280, framerate=30/1, format=UYVY' ! tee name=camera1
↪v4l2src io-mode=0 device=/dev/video1 do-timestamp=true ! 'video/x-raw,
↪width=1920, height=1280, framerate=30/1, format=UYVY' ! tee name=camera2 camera1.
↪ ! nvvidconv ! nvv4l2h265enc bitrate=3000000 ! h265parse !  qtmux ! filesink
↪location=out_1.mp4 camera2. ! nvvidconv ! nvv4l2h265enc bitrate=3000000 !
↪h265parse !  qtmux ! filesink location=out_2.mp4
```

In the case of the C2 camera, please change the `width`, `height`, and `framerate` settings appropriately. `width=2880` and `height=1860`.

3.6 Using Camera Image Data with ROS2

3.6.1 ROS2 Installation

The latest distribution provided by Jetpack is Ubuntu 20.04. The ROS2 distribution compatible with Ubuntu 20.04 is Galactic. However, Galactic has reached its end of life (EOL), so please install Humble either by building from the source or using Docker. Please refer to the below links for the details of installation.

- In the case of building Humble from the source
- In the case of using Docker (**recommended**)

We recommend using Docker because building ROS2 Humble from source on Jetson Orin (Ubuntu 20.04) may fail due to OS incompatibility.

3.6.2 Node Installation (if not installed yet)

We recommend using the v4l2_camera node. Please refer to the Github repository for installation instructions.

3.6.3 Start v4l2_camera Node and Visualize Image Data Topics

Open a terminal and start the v4l2_camera node with the following command:

```
ros2 run v4l2_camera v4l2_camera_node --ros-args \
  -p video_device:=/dev/video0 \
  -p pixel_format:=UYVY \
  -p image_size:=[1920,1280]
```

Note: In the case of C2, please change the configuration of image_size to [2880,1860].

The image topic (in ROS format) /image_raw will be published. Use rqt_image_view to visualize it. Open a new terminal and execute the following command to launch rqt_image_view:

```
ros2 run rqt_image_view rqt_image_view
```

3.6.4 Recording rosbag

While the v4l2_camera node is running, you can record rosbag (ROS format log files) using the following command:

```
ros2 bag record -a
```

3.7 Configuration

To modify camera driver settings, edit the /etc/modprobe.d/tier4-*.conf file. The configuration file contains the following lines (the case of tier4-isx021.conf):

```
# /etc/modprobe.d/tier4-isx021.conf
options tier4_isx021 trigger_mode=0 enable_auto_exposure=1 enable_distortion_
↪correction=1
```

After editing these flags, restart the system for the changes to take effect.

3.7.1 Switching Drive Modes (Freerun/Trigger)

Warning: Please note that drive mode switching is not supported on Jetson Orin/Xavier Development Kit. as there is no FSYNC input function on Jetson Orin/Xavier devkit. Changing from the default settings may cause unintended behavior, so please refrain from doing so.

The case of C1

Drive mode is fixed to 30fps freerun mode. Other drive mode is not supported.

The case of C2

Drive mode	Frame rate	trigger_mode=
Master	10fps	0
Master	20fps	2
Master	30fps	4

3.7.2 Enabling/Disabling Lens Distortion Correction (LDC) Feature

You can enable or disable the Lens Distortion Correction (LDC) feature by modifying the value of the enable_distortion_correction flag.

To enable Lens Distortion Correction, set enable_distortion_correction=1. To disable Lens Distortion Correction, set enable_distortion_correction=0.

3.7.3 Setting exposure time

The case of C1

In the case of C1, users can specify three types of exposure time via variables named `shutter_time_min`, `shutter_time_mid`, and `shutter_time_max`. The actual exposure time of C1 transits these three (and linearly interpolated values) corresponding to the illuminance. The values for the variables should be specified in microseconds. Users who would like to fix exposure time under arbitral illuminance conditions can set the exact same value for these variables. For example, the following configuration in `/etc/modprobe.d/tier4-isx021.conf` fixes exposure time to 11 ms:

```
shutter_time_min=11000 shutter_time_mid=11000 shutter_time_max=11000
```

The case of C2

Likewise C1, C2 also has capability to set two types of exposure time via variables named `shutter_time_min` and `shutter_time_max`. The unit of the value for these variables is also microseconds. For example, the following configuration in `/etc/modprobe.d/tier4-imx490.conf` fixes exposure time to 11 ms:

```
shutter_time_min=11000 shutter_time_max=11000
```

3.7.4 For other parameter adjustments

Please refer to the T4cam-ctrl user manual

4

TIER IV Cameras Getting Started Guide for Vecow EAC-5000

Attention: This document is for the users of Vecow EAC-5000

4.1 Preparation

4.1.1 Required items

- Vecow EAC-5000
  - Jetpack 5.1.1 (Linux for Tegra 35.3.1) installed
- GMSL2 coaxial cable
- TIER IV Automotive HDR Camera C1 or C2

4.1.2 Camera connection

1. Firstly, make sure that the EAC-5000 is turned off.
2. Connect cameras to EAC-5000 using the FAKRA cables.

4.2 Power on and log in

4.2.1 Power on

Power on the EAC-5000 and confirm that the power-on LED is turned on blue.

4.2.2 Login

On the startup window, type the password to log in.

Default settings are shown below.

user	ubuntu
password	ubuntu

4.3 Camera driver installation

Note: Installation process is different from the other ECUs

Currently, EAC-5000 has issues with building and loading the out-of-tree modules.

We provide the pre-built drivers and the installation script to avoid dealing with the complicated kernel stuff.

Please download the pre-build driver from our GitHub release page

1. Extract the tarball and run `./install.sh` to install the camera drivers.

   ```
   tar -xzvf tier4-drivers-for-eac-5000.tar.gz
   cd tier4-drivers-for-eac-5000/
   ./install.sh
   ```

2. The installation script does the following steps for you:

   - Install the camera drivers' configuration files
   - Install the pre-built camera drivers
   - Install the pre-built device-tree blob
   - Create a new boot entry in the `/boot/extlinux/extlinux.conf`

3. After running the installation script, you can run `/opt/nvidia/jetson-io/configure-by-hardware.py`. The following prompt should appear.

   ```
   [*] Now you can run config-by-hardware.py
   ```

4. Select the GMSL port assignment by using `configure-by-hardware.py`. For the details, please refer to GitHub README.

   ```
   sudo /opt/nvidia/jetson-io/config-by-hardware.py -l
   ```

   This command will output the following:

   ```
   Header 1 [default]: Jetson 40pin Header
   Available hardware modules:
   1. Adafruit SPH0645LM4H
   2. Adafruit UDA1334A
   3. FE-PI Audio V1 and Z V2
   4. ReSpeaker 4 Mic Array
   5. ReSpeaker 4 Mic Linear Array
   Header 2: Jetson AGX CSI Connector
   Available hardware modules:
   1. Jetson Camera Dual-IMX274
   2. Jetson Camera E3331 module
   3. Jetson Camera E3333 module
   ```

(continues on next page)

(continued from previous page)

```
4. Jetson Camera Hawk-Owl p3762 module
5. Jetson Camera IMX185
6. Jetson Camera IMX390
7. Jetson Camera e3653-dual-Hawk module
8. TIERIV IMX490 GMSL2 Camera Device Tree Overlay
9. TIERIV ISX021 GMSL2 Camera Device Tree Overlay
10. TIERIV ISX021 IMX490 GMSL2 Camera Device Tree Overlay
Header 3: Jetson M.2 Key E Slot
No hardware configurations found!
```

Then please select the GMSL port assignment.

```
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 2='TIERIV ISX021 GMSL2 Camera Device Tree Overlay'
```

The command above assigns the C1 camera for all GMSL ports. Please modify after `2=` according to your preference.

5. Reboot

```
sudo reboot now
```

6. Make sure that the camera device file (`/dev/video*`) is created

```
ls -al /dev/video*
```

You need to make sure `extlinux.conf` is well-formed and valid after the installation script is finished. This step may require your manual editing.

4.4 Visualize the camera output using GStreamer

The procedure to output the image using Gstreamer is identical to the other ECUs. Please refer to the manual.

Please also refer to the GStreamer command examples.

4.5 Restrictions

Caution: EAC-5000 does not supports Camera trigger (FSYNC) function as a default configuration. TIER IV camera' s frame synchronization function is not supported with combination with EAC-5000.

5

TIER IV Cameras Getting Started Guide for Connect Tech Anvil

Attention: This document is for the users of Connect Tech Anvil

5.1 Preparation

5.1.1 Required items

- Connect Tech Anvil
  - BSP: AGX Orin L4T r35.4.1
- Fakra coaxial cable
- TIER IV Automotive HDR Camera C1 or C2

5.1.2 Camera connection

1. Firstly, make sure that the Anvil is turned off.
2. Connect cameras to Anvil using the FAKRA cables.

5.2 Power on and log in

5.2.1 Power on

Power on the Anvil.

5.2.2 Login

On the startup window, type the password to log in.

Default settings are shown below.

user	nvidia
password	nvidia

5.3 Camera driver installation

Note: Installation process is different from the other ECUs

Currently, Anvil has issues with building and loading the out-of-tree modules.

We provide the pre-built drivers and the installation script to avoid dealing with the complicated kernel stuff.

Please download the pre-build driver from our GitHub release page

1. Extract the tarball and run `./install.sh` to install the camera drivers.

```
tar -xzvf tier4-camera-drivers-for-anvil.tar.gz
cd tier4-camera-drivers-for-anvil
sudo ./install.sh
```

2. The installation script does the following steps for you:
   - Install the camera drivers' configuration files
   - Install the pre-built camera drivers
   - Install the pre-built device-tree blob
   - Create a new boot entry in the `/boot/extlinux/extlinux.conf`

3. After running the installation script, you can run `/opt/nvidia/jetson-io/configure-by-hardware.py`. The following prompt should appear.

```
[*] Installing configuration files...
[*] Enabling the C1 firmware...
Adding firmware to initrd...
42009 blocks
42009 blocks
INITRD Updated with Tier4 firmware.
[*] Installing the camera drivers...
[*] Renaming the existing base devicetree blobs: /boot/dtb/tegra234-orin-agx-
↪cti-AGX201-JCB002-base.dtb
[*] Copying the new devicetree blob
[*] Copying the new kernel image with VI patches applied
[*] Adding an entry to /boot/extlinux/extlinux.conf
[*] Making the entry the default one
[*] You can run config-by-hardware.sh to enable TIER IV cameras after
↪rebooting
[!] Please make sure extlinux.conf is well-formed and valid before rebooting.
```

4. Select the GMSL port assignment by using `configure-by-hardware.py`. For the details, please refer to GitHub README.

```
sudo /opt/nvidia/jetson-io/config-by-hardware.py -l
```

This command will output the following:

```
Header 1 [default]: Jetson AGX CSI Connector
Available hardware modules:
1. TIERIV GMSL2 Camera Device Tree Overlay: C1x4 C2x4
2. TIERIV GMSL2 Camera Device Tree Overlay: C1x8
3. TIERIV GMSL2 Camera Device Tree Overlay: C2x8
```

Then please select the GMSL port assignment. For instance, if you assign all GMSL ports to the C1, you set:

```
sudo /opt/nvidia/jetson-io/config-by-hardware.py -n 1='TIERIV GMSL2 Camera⊠
↪Device Tree Overlay: C1x8'
```

The command above assigns the C1 camera for all GMSL ports. Please modify after `1=` according to your preference. The following output should appear in case `configure-by-hardware.py` finishes successfully.

```
Configuration saved to /boot/tegra234-orin-agx-cti-AGX201-JCB002-base-user-
↪custom.dtb.
Reboot system to reconfigure.
```

5. Reboot

   ```
   sudo reboot now
   ```

6. Make sure that the camera device file (`/dev/video*`) is created

   ```
   ls -al /dev/video*
   ```

You need to make sure `extlinux.conf` is well-formed and valid after the installation script is finished. This step may require your manual editing.

5.4 Visualize the camera output using GStreamer

The procedure to output the image using GStreamer is identical to the other ECUs. Please refer to the manual.

Please also refer to the GStreamer command examples.

5.5 Restrictions

Caution: Camera synchronization function will be supported in case of the combination with Anvil.

5.6 Reference document

- Anvil user guide
- Anvil release note

6

Sensor fusion development kit getting started guide

6.1 Hardware setup

As a first step, prepare your hardware including Sensor and ECU.

6.1.1 Sample hardware configuration

This following hardware configuration is used throughout this tutorial.

- ECU setup
  - x86-based ECU: ADLINK AVA-3510
  - Jetson-based ECU: ADLINK RQX-58G
- Sensor setup
  - Sample configuration 1
    - Camera: TIER IV Automotive HDR Camera C1 (x2)
    - LiDAR: HESAI AT128 (x1)
  - Sample configuration 2
    - Camera: TIER IV Automotive HDR Camera C1 (x2)
    - LiDAR: HESAI Pandar XT32 (x1)

Connection diagram

The figure below depicts the connection diagram between sensors and ECUs for this tutorial. This network configuration, including applying the IP addresses to the specific interface, will be automatically done during the steps in Installation page.

Internet connection is required for Installation step.

The next Installation step requires the internet connection for git clone and ML model download. Please connect an Ethernet cable to the port indicated in the figure below for the internet connection.

Note: Try another display port if nothing is shown on the display.

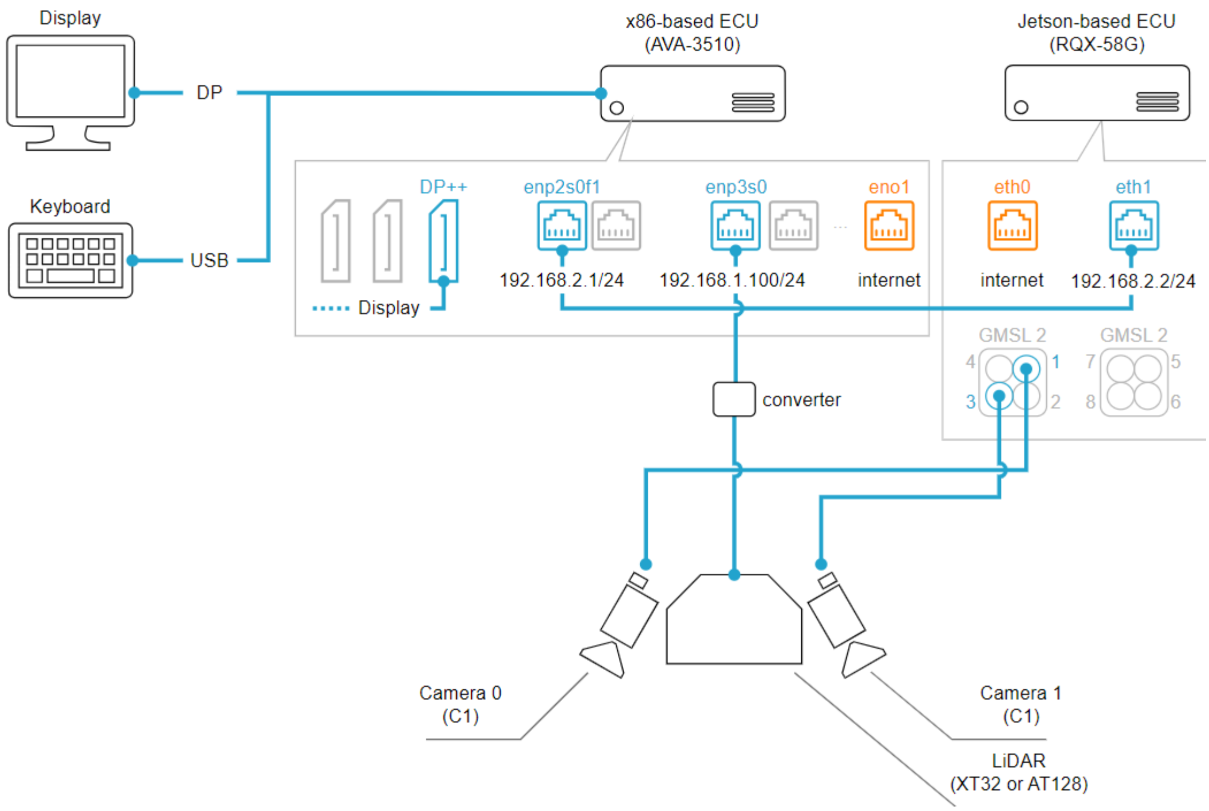

Fig. 6.1 connection diagram of the sample system

Sensor driver

Edge.Auto supports a variety of sensor types. The following repositories are used to make those sensors available in your ROS2 environment. Please refer to the each repositories for more details.

- Camera driver
  - tier4/tier4_automotive_hdr_camera: Kernel driver for using TIER IV cameras with Video4Linux2 interface.
  - tier4/ros2_v4l2_camera: ROS2 package for camera driver using Video4Linux2.
- LIDAR driver
  - tier4/nebula: ROS2 package for unified ethernet-based LiDAR driver.

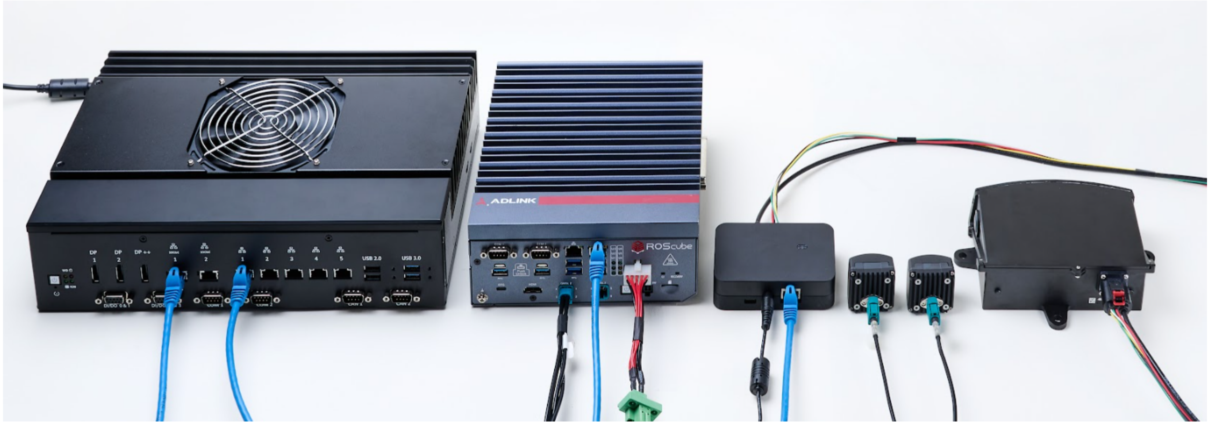

Fig. 6.2 hardware setup of sample system

- Sensor synchronization
  - tier4/sensor_trigger: ROS2 package for generating sensor trigger signals.

Sensor/ECU synchronization

In this sample system, clock synchronization and timing synchronization between sensors and ECUs are realized to achieve highly accurate sensor fusion. The figure below depicts the synchronization design between sensors and ECUs in this sample system.

For more details, please refer to the tier4/sensor_trigger repository.

6.1.2 x86-based ECU

Before proceeding with 2.Installation step, install Ubuntu 22.04 to your x86-based ECU.

6.1.3 Jetson-based ECU

Before proceeding with 2.Installation step, install NVIDIA L4T R32.6.1 (including Ubuntu 18.04) to your Jetson-based ECU.

BSP installation for ADLINK RQX-58G

RQX-58G needs to be properly configured according to the official quick start guide from ADLINK Technology, Inc. Please see the official document in detail.

To download the BSP image, please visit the ADLINK official page. (If you are accessing the site for the first time, you will be prompted to create an account.)

While the TIER IV camera driver (tier4/tier4_automotive_hdr_camera) is included in the RQX-58G BSP official image, you can also update it during the following setup process.

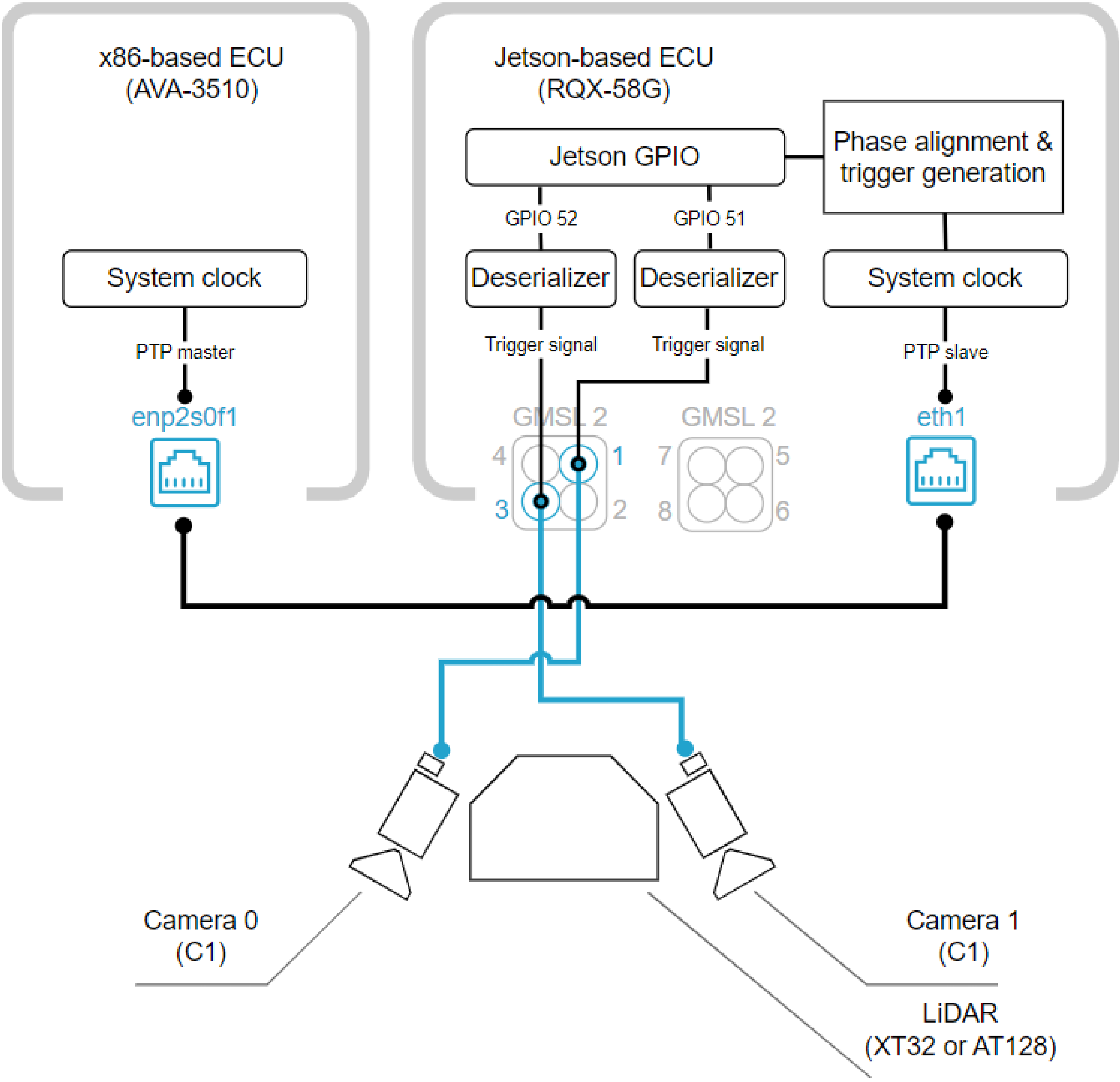

Fig. 6.3 Synchronization design of sample system

6.2 Installation

Setup for both x86-based and Jetson-based ECUs.

Note: Internet connection is required in this step.

6.2.1 x86-based ECU

Warning: Network settings are automatically updated.

During this procedure, IP addresses are assigned to some network interfaces (refer to the connection diagram on Hardware setup for more detail) using `netplan`. This behavior may cause unexpected disconnection, in case you are accessing the ECU remotely via those interfaces.

If you would like to change network interfaces or IP addresses to be assigned, edit `edge-auto/ansible/playbooks/vars/edge_auto.yaml` before running `setup-dev-env.sh`

Download the repository and setup your environment

As a first step, clone `tier4/edge-auto` and move to the directory.

```
git clone https://github.com/tier4/edge-auto.git
cd edge-auto
```

You can install the dependencies using the provided ansible script.

```
./setup-dev-env.sh
```

Finally, please reboot the system to make the installed dependencies and permission settings effective.

```
sudo reboot
```

Build edge-auto

Create your ROS workspace and clone repositories using vcstool.

```
cd edge-auto
mkdir src
vcs import src < autoware.repos
```

Install ros package dependencies and build your ROS workspace.

```
rosdep install -y -r --from-paths src --ignore-src --rosdistro $ROS_DISTRO

colcon build \
  --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release \
  --packages-up-to edge_auto_launch
```

6.2.2 Jetson-based ECU

This following steps can be performed from your x86-based ECU via ssh

Download the repository and setup the environment

As a first step, clone `tier4/edge-auto-jetson` and move to the directory.

```
git clone https://github.com/tier4/edge-auto-jetson.git
cd edge-auto-jetson
```

You can install the dependencies using the provided ansible script. During the installation process, you will be asked if you want to install the TIER IV camera driver. If you already have the driver installed and want to skip this step, please type `N` to continue.

> **Caution:** `setup-dev-env.sh` script may take several hours.

```
./setup-dev-env.sh

[Warning] Do you want to install/update the TIER IV camera driver? [y/N]:
```

Finally, please reboot the system to make the installed dependencies and permission settings effective.

```
sudo reboot
```

Build edge-auto-jetson workspace

Create your ROS workspace and clone repositories using vcstool.

```
cd edge-auto-jetson
mkdir src
vcs import src < autoware.repos
```

Build your ROS workspace.

```
colcon build \
  --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release \
  -DPython3_EXECUTABLE=$(which python3.6) -DCMAKE_CUDA_STANDARD=14 \
  --packages-up-to edge_auto_jetson_launch
```

6.2.3 Update your workspace

If you want to update cloned repositories, use the following command.

```
vcs import src < autoware.repos
vcs pull src
```

6.2.4 Modify camera exposure timing

> **Note:** On the sample system introduced in Hardware setup step, this does not need to be changed.

If you want to change the exposure time of cameras for sensor synchronization, please modify the following files.

```
edge-auto-jetson/src/individual_params/individual_params/config/
└── default
```

(continues on next page)

(continued from previous page)

```
    ├── camera0
    │   ├── trigger.param.yaml
    ├── camera1
    │   ├── trigger.param.yaml
```

For more details, please refer to the tier4/sensor_trigger repository.

6.3 Sensor Calibration

Estimate your intrinsic/extrinsic parameters on your sensor system using tier4/calibration_tools.

Note: Perform the following tasks on the x86-based ECU.

6.3.1 Calculate intrinsic parameters for cameras

First, launch `calibration_intrinsic` to estimate the intrinsic parameters of your cameras. See this document for detailed operation on the tool.

```
cd edge-auto
source install/setup.bash

ros2 launch edge_auto_launch calibration_intrinsic_sample.launch.xml
```

Intrinsic parameters for all cameras consisting of your system (e.g., `camera0` and `camera1` in this tutorial) should be stored on `individual_params`. After acquiring the files, put them in the appropriate folder on your **Jetson-based ECU** so that they are loaded and published as `camera_info` topics:

```
edge-auto-jetson/src/individual_params/individual_params/config/
└── default # <- this will be an identifier, which refered to as the value of⌧
↪`VEHICLE_ID` environment variable, of your system
    ├── camera0
    │   ├── camera_info.yaml  # <- replace this file with your calculated results
    │   ├── trigger.param.yaml
    │   └── v4l2_camera.param.yaml
    ├── camera1
    │   ├── camera_info.yaml  # <- replace this file with your calculated results
    │   ├── trigger.param.yaml
    │   └── v4l2_camera.param.yaml
```

6.3.2 (HESAI AT128 only) Get a correction file from LiDAR

HESAI AT128 have the capability to access their own correction file stored inside them. To get better results, you are encouraged to download the correction file from individual LiDAR, and store it in the appropriate folder on the x86-based ECU:

```
edge-auto/src/individual_params/individual_params/config/
└── default
    └── lidar
        └── at128_default.dat # <- replace this file with your downloaded dat file
```

6.3.3 Calculate extrinsic parameters between LiDAR and cameras

Finally, launch `calibration_extrinsic` that matches your LiDAR setup to estimate your extrinsic parameters between your lidar and cameras. See this document for detailed operation on the tool.

```
cd edge-auto
source install/setup.bash

ros2 launch edge_auto_launch calibration_extrinsic_at128_sample.launch.xml
## or
ros2 launch edge_auto_launch calibration_extrinsic_xt32_sample.launch.xml
```

To perform sensor fusion, pose relationships (i.e., extrinsic parameters) between all fused sensors need to be registered in `TF`, which is the ROS-fashion to represent the relationships.

Note: In these samples, there is the assumption that all sensors are fixed and their relative positions do not change.

After calculating extrinsic parameters, put the result in the appropriate files on the x86-based ECU:

```
edge-auto/src/individual_params/individual_params/config/
└── default
    ├── at128_to_camera0.json # <- replace this file with your calculated results
    └── at128_to_camera1.json # <- replace this file with your calculated results
```

6.4 Launch applications

Launch the perception application implemented in autoware.universe.

6.4.1 Jetson-based ECU

Note: This following steps can be performed from your x86-based ECU via ssh.

The following sample launches image-based object detection performed on two cameras individually.

```
cd edge-auto-jetson
source install/setup.bash

ros2 launch edge_auto_jetson_launch edge_auto_jetson.launch.xml
```

Note: It may take about 15 minutes before turning the results available at the first execution, which is caused by converting the ONNX model into a TensorRT engine. From the second launch, the results should be available immediately since conversion results are cached on the disk.

This sample utilizes tensorrt_yolox and bytetrack packages implemented in autoware.universe. See the READMEs of these packages for more detail.

6.4.2 x86-based ECU

The following sample launches LiDAR-based object detection and bounding-box-level fusion (i.e. late fusion) between 2D and 3D object detection. Launch the launch file that matches your LiDAR setup.

```
cd edge-auto
source install/setup.bash

ros2 launch edge_auto_launch perception_at128_sample.launch.xml
## or
ros2 launch edge_auto_launch perception_xt32_sample.launch.xml
```

This sample mainly leverages pointcloud_preprocessor, centerpoint, and image_projection_based_fusion packages in autoware.universe. See the READMEs of these packages for more detail.

In addition to the perception stack, this sample also launches viewers so that users can check perception results visually.

6.5 Trouble shooting