Edge.Auto Getting Started Manual

TIER IV

2024 年 05 月 07 日

目次

1

ADLINK ROSCube RQX-58G 用 TIER IV カメラ スタート ガイド

注意: このドキュメントは ADLINK RQX-58G のユーザー向けです。

1.1 準備

1.1.1 機材

- ADLINK ROSCube-X RQX-58G
- JetPack4.5 (L4T 32.5.1) または JetPack4.6 (L4T 32.6.1)
- GMSL2 同軸ケーブル (FAKRA - mini FAKRA、1:4)
- TIER IV 車載 HDR カメラ C1 または C2

1.1.2 カメラの接続

1. まず、ROSCube-X の電源がオフになっていることを確認します。
2. カメラをケーブルの FAKRA コネクタ (シングル FAKRA コネクタ側) に接続します。次に、ケーブルのミニ FAKRA コネクタを ROSCube-X の GMSL2 ポートに挿入します。
3. FAKRA コネクタおよびミニ FAKRA コネクタのロックの方向が正しいことを確認してください (図 1 ～ 3)。

図 1.1　RosCube

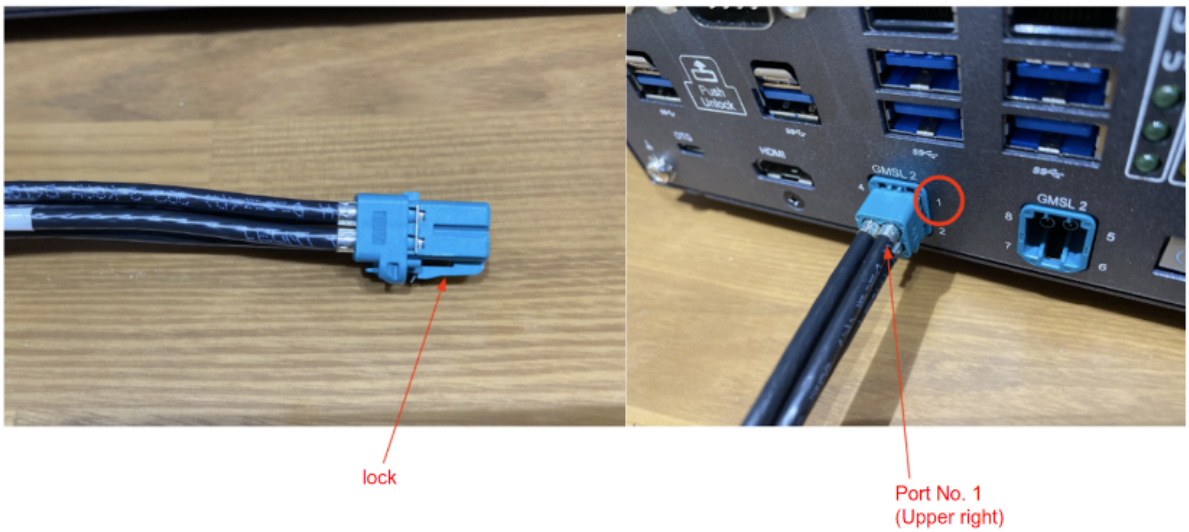

図 1.2　コネクタ

1.2 電源投入とカメラからの画像出力の確認

1.2.1 電源投入

ROSCube-X の電源を入れ、パワーオン LED が青色に点灯することを確認します。

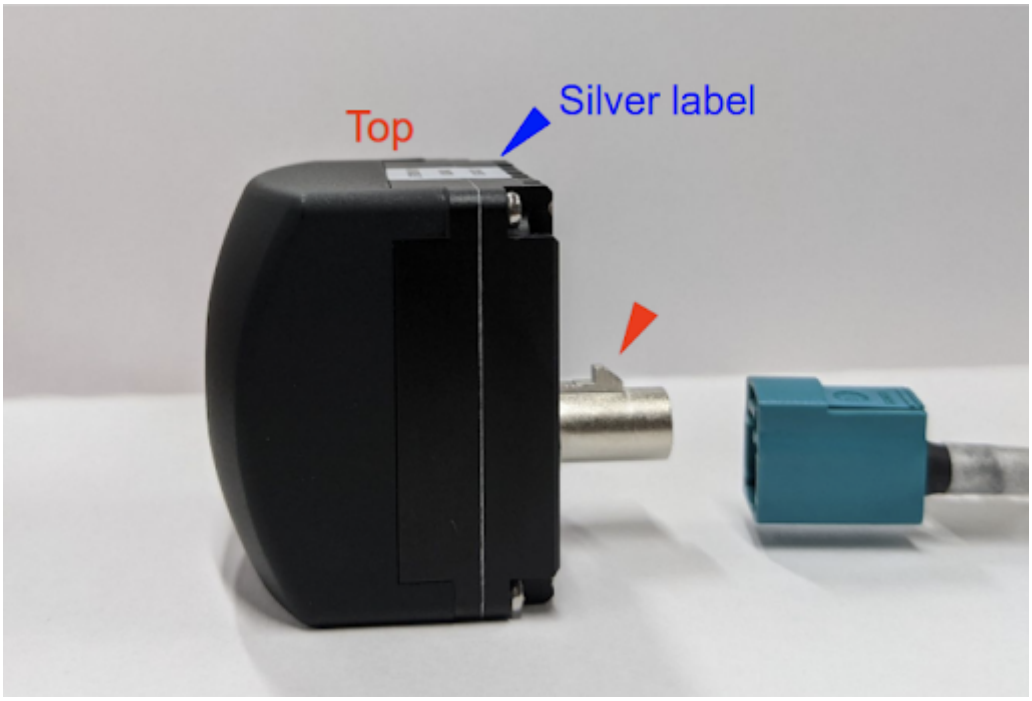

図 1.3　Fakra ケーブル挿入方向

1.2.2 ログイン

起動ウィンドウで、パスワードを入力してログインします。

デフォルトの設定は以下のとおりです。

ユーザー	ros
パスワード	adlinkros

1.2.3 カメラドライバーのインストール

注釈: C2 を使用する場合、カメラドライバ v1.4.1 以上が必須です。必要な場合ドライバをアップデートしてください。

必要なカメラドライバーがすでにインストールされている場合は、このセクションをスキップしてください。

Github から カメラドライバ deb パッケージ を入手します。最新リリースは RQX-58G で動作することが確認されています。

提供されたドライバー パッケージ ファイル (`tier4-isx021-gmsl_*.*.*_arm64.deb`) を ROSCube-X 内の任意のディレクトリ (例: `~/c1_driver`) にコピーします。次に、以下の操作をコマンドラインで行います。

1. apt update でパッケージを更新します。インターネット接続が必要です。次に、apt install コマンドを使用してドライバーをインストールします。*.dtbo ファイルが /boot に生成されていることを確認してください。

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

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

2. デバイスツリーオーバーレイのコマンドは、L4T バージョンによって異なります。L4T バージョンを確認するには、$ cat /etc/nv_tegra_release を実行して結果を確認します。たとえば、# R32 (release), REVISION: 5.1..., が返された場合、インストールされている L4T バージョンは「32.5.1」です。

3. インストールされている L4T バージョンに応じた手順を選択してください。また、各ポートにカメラを割り当てるには、カメラ ドライバーの README ページ を参照してください。実現したいカメラの割当に応じて適切なオーバーレイ コマンドを指定して下さい。

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 以降

```
# 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
```

/opt/nvidia/jetson-io/config-by-hardware.py -l を実行すると、利用可能なオーバーレイ オプションを確認することができます。

```
$ 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
```

たとえば、すべての GMSL ポートを C2 カメラに割り当てるには、overlay コマンドは次のようにして下さい。

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

C1 と C2 の両方に GMSL ポートを割り当てるには、overlay コマンドを次のようにして下さい。

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

この場合、ポート 1、2、5、および 6 は C1 に割り当てられ、ポート 3、4、7、および 8 は C2 に割り当てられます。

デバイスオーバーレイの詳細については、デバイスドライバの GitHub リポジトリを参照してください。

1.2.4 ROSCube-X がカメラを認識しているかの確認

ターミナルウィンドウを開き、次のコマンドを入力します。/dev/videoX が返された場合、カメラはビデオデバイスとして正しく認識されています。

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

1.2.5 GStreamer を使用したカメラ出力の視覚化

ターミナルウィンドウを開き、次のコマンドを入力します。Gstreamer が起動し、新しいウィンドウにカメラ画像ストリームが表示されます（図 4、図 5）

ケース 1: 1 台の C1 カメラが接続されている場合

```
# 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
```

ケース 2: 8 台の C1 カメラが接続されている場合

```
# 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 \
```

ケース 3: C2 カメラ 1 台が接続されている場合

```
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
```

C2 カメラに関する制限事項

現在、2 台の C2 カメラを 1 つのデシリアライザ ボードに接続することはできません。

例：

- ポート 1 および 2 ... NG
- ポート 3 と 4 ... NG
- ポート 1 とポート 3 ... OK
- ポート 1 と 5 ... OK

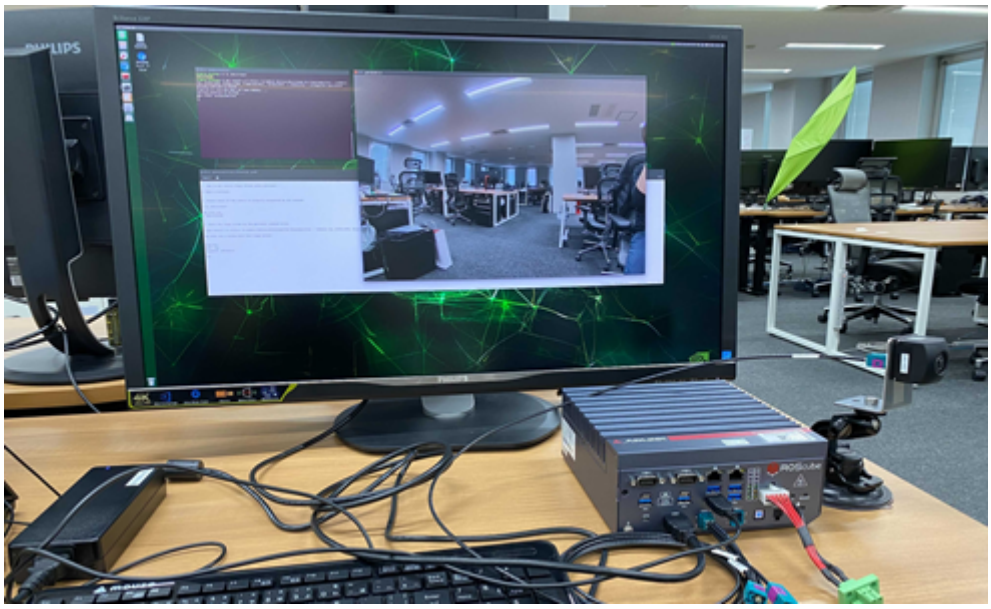

図 1.4　カメラ出力画像

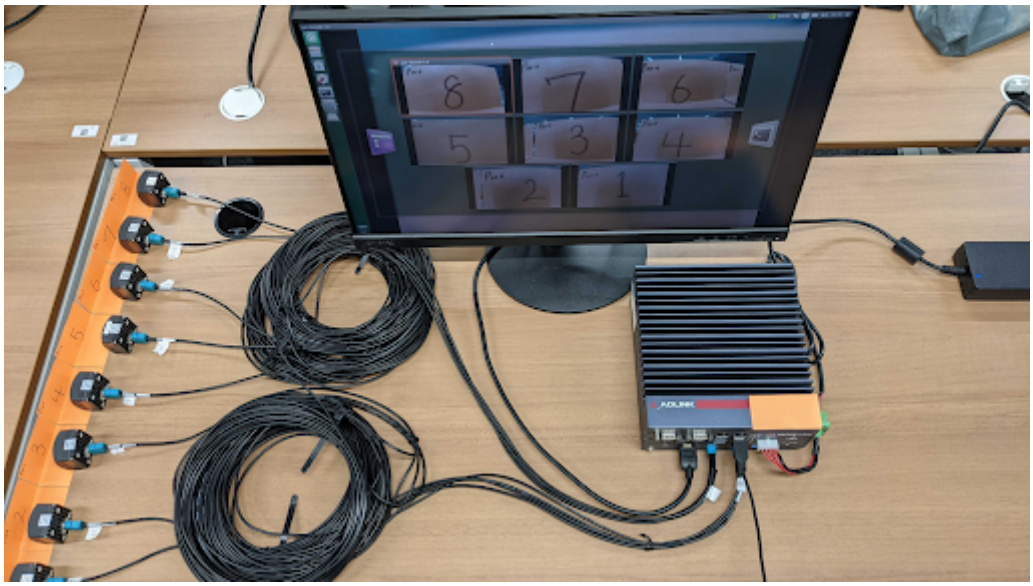

図 1.5　カメラ出力画像 (x8 カメラ)

カメラの起動順序の制約 (v1.1.1 以前のドライババージョンの場合)

v1.1.1 より前のバージョンのドライバを使用している場合は、起動のための特定の手順に従ってください。ドライババージョン v1.2.1 以降の場合、順序や制約の要件はありません。

複数のカメラを実行するには、カメラを一度に起動する必要があります。たとえば、2 台のカメラを実行するには、start-streaming コマンドをワンライナーにする必要があります。

```
# 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 \
```

これらの個別のコマンドは機能しません

```
# 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 映像データを rosbag（ROS 形式のログデータ）として記録する

1.3.1 ROS melodic と gscam (GStreamer 用の ROS カメラドライバー) のインストール

コマンドラインで次の入力を行い、ROS melodic と gscam をインストールして下さい。

> 注意: インストールコマンドは変更される可能性があります。次のコマンドが機能しない場合は、ROS 公式ドキュメント を参照してください。

```
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
```

(次のページに続く)

(前のページからの続き)

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
sudo apt install -y ros-melodic-gscam
```

1.3.2 gscam の実行と画像トピックの可視化

以下の手順で gscam を実行して下さい。

1. 新しいターミナルを開き、次のコマンドを実行して roscore を起動します

   ```
   roscore
   ```

2. 別のターミナルを開き、次のコマンドを実行します

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

3. トピック（ROS 形式のメッセージデータ）camera/image_raw が公開されたので、rqt_image_view でトピックを可視化します。rqt_image_view を起動するには、別のターミナルを開いて次のコマンドを実行します。

   ```
   rosrun rqt_image_view rqt_image_view
   ```

   このコマンドは、画像トピックを視覚化するためのウィンドウを開きます。ウィンドウ左上のプルダウンメニューをクリックして、camera/image_raw を選択します (図 6)。

複数のカメラで gscam を実行するには、次の XML メッセージを含むテキスト ファイルを作成し、~/2cameras.launch として保存します。この .launch ファイルは 2 台のカメラでのストリーミング用です。

```
<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_
```

(次のページに続く)

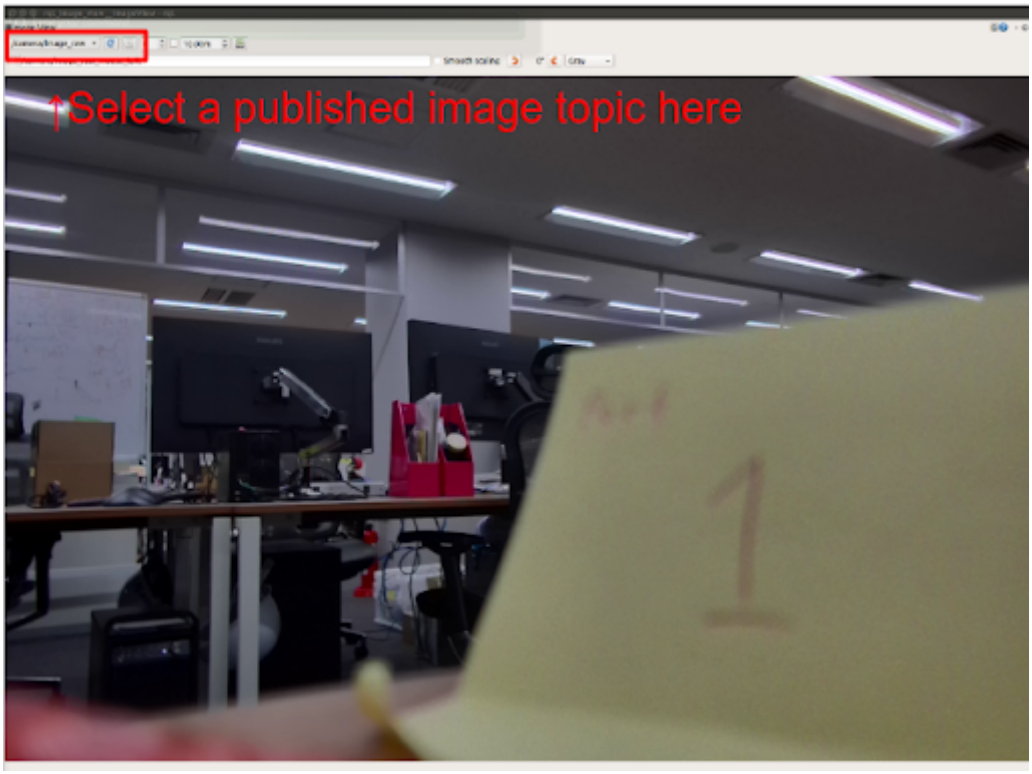

図 1.6　公開された画像トピックを選択してください

(前のページからの続き)

```
↪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>
```

注意: 上記は C1 の設定です。C2 の場合は、必要に応じて「width」、「height」、「framerate」を変更してください。

/home/ros/2cameras.launch を作成したとして、次のコマンドを実行して複数のカメラで gscam を実行します。

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

注釈: さらにカメラを追加するには、適切な node name、namespace(ns)、camera_name、frame_id、および gscam_config を使用してノードを追加します。

1.3.3 Rosbag を記録する

gscam の実行中に、次の方法で rosbag (ROS 形式のログ ファイル) を記録できます。

```
rosbag record -a
```

rosbag info で rosbag の情報を確認することができます。

```
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
```

1.4 設定変更

ドライバの設定を変更するには、設定ファイル /etc/modprobe.d/tier4-*.conf を編集します。(次のセクションを参照してください)。設定ファイルにはデフォルトで次の行が含まれています。(C1 の場合です)

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

設定を適用するには、編集後に ROSCube-X を再起動します。

1.4.1 駆動モードの切り替え

> 注意: カメラをスレーブモードで動作させるには、ROSCube-X からフレーム同期信号を入力する必要があります。カメラをスレーブ モードで駆動するには、Shutter triggering over GMSL2 をチェックしてください。

C1 の場合

設定ファイルを編集することで、カメラの駆動モードをマスター モードからスレーブ モード (またはその逆) に切り替えることができます。

- マスターモード: 30fps のフリーランモード (デフォルト)
- スレーブモード: シャッタートリガーモード。シャッタータイミング、フレームレートは FSYNC 信号周波数によって調整可能 (ROSCube-X で設定可能)

モードを切り替えるには、`/etc/modprobe.d/tier4-isx021.conf` の `trigger_mode` を編集します。(以下参照)

駆動モード	フレームレート	trigger_mode=
マスター	30fps	0
スレーブ	FSYNC 入力周波数に依存	1

C2 の場合

C2 の場合、/etc/modprobe.d/tier4-imx490.conf を編集することで以下のモードを選択することができます。次の表に、使用可能な設定を示します。

駆動モード	フレームレート	trigger_mode=
マスター	10fps	0
スレーブ	10fps	1
マスター	20fps	2
スレーブ	20fps	3
マスター	30fps	4
スレーブ	30fps	5

1.4.2 レンズ歪み補正 (LDC) の有効化/無効化

注釈: この設定は C1 と C2 に共通です。

LDC を有効にするには、フラグ `enable_distortion_correction=1` (デフォルト) を設定します。LDC を無効にするには、フラグ `enable_distortion_correction=0` を設定します。

1.4.3 オートエクスポージャーの有効化/無効化

注釈: この設定は C1 と C2 に共通です。

LDC を有効にするには、フラグ `enable_distortion_correction=1` (デフォルト) を設定します。LDC を無効にするには、フラグ `enable_distortion_correction=0` を設定します。

1.4.4 露光時間の固定

C1 の場合

C1 には 3 種類の露光時間があり、変数 `shutter_time_min`, `shutter_time_mid`, `shutter_time_max` を使用することで各々を設定することが可能です。実際の露光時間は、周囲の明るさに応じてこれら３つの３変数値及びそれらの線形補間値の間を遷移します。各変数値の単位はマイクロ秒です。異なる環境光条件下であっても露光時間が変化しないように固定するには、これらの変数に対してすべて同じ値を設定してください。例えば、`/etc/modprobe.d/tier4-isx021.conf` に対して以下の設定を記述することで露光時間を 11 ms に固定することができます。

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

C2 の場合

C1 と同様に、C2 では `shutter_time_min` 及び `shutter_time_max` の２種類の露光時間を設定することが可能です。各変数値の単位はマイクロ秒です。例えば、`/etc/modprobe.d/tier4-imx490.conf` に対して以下の設定を記述することで、露光時間を 11 ms に固定することができます。

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

1.5 GMSL2 経由のシャッタートリガー

スレーブ モードでのカメラ操作には、GMSL2 を介したトリガー信号入力 (FSYNC 入力) が必要です。FSYNC パラメータと GPIO 設定の構成方法については、ADLINK のドキュメント を参照してください。

1.5.1 スレーブモードでカメラを駆動するための準備

ターミナルで `i2cget -f -y 2 0x66 0x01` を実行して ROSCube-X の HW バージョンを確認し、以下の手順に従ってください。

i2cget -f -y 2 0x66 0x01 が 0x23 を返す場合

ストリーミングを開始する前に `i2cset -f -y 2 0x66 0x04 0xff` を実行してください。システムでは、起動後にこの準備が 1 回だけ必要になります。

i2cget -f -y 2 0x66 0x01 が 0x21 を返す場合

準備は必要ありません。

1.5.2 C1 の FSYNC 周波数

30fps より低い任意の周波数を入力可能です

1.5.3 C2 の FSYNC 周波数

C2 の場合、入力可能な FYNC の周波数は駆動モードに依存します。以下の表を参照して下さい。

- 30fps モード (`trigger_mode=5`): 15 < f <= 30 fps
- 20fps モード (`trigger_mode=3`): 10 < f <= 20 fps
- 10fps モード (`trigger_mode=1`): 5 < f <= 10 fps

1.5.4 フレームレートの確認

ビデオ上で フレームレート設定を確認するには、次のコマンドを実行してください。

- カメラがポート 1 にのみ接続されてる場合

```
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
```

- カメラがポート 1 と 2 に接続されている場合

```
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

ADLINK ROSCube RQX-59G 用 TIER IV カメラスタートガイド

注意: このドキュメントは ADLINK RQX-59G のユーザー向けです

2.1 準備

2.1.1 必須項目

- ADLINK ROSCube-X RQX-59G
- BSP バージョン 1.4.2
- JetPack5.1.2 (L4T R35.4.1) がインストール済み
- GMSL2 同軸ケーブル (FAKRA - mini FAKRA、1:4)
- TIER IV 車載 HDR カメラ C1 または C2

2.1.2 カメラ接続

1. まず、RQX-59G の電源がオフになっていることを確認します。
2. カメラをケーブルの FAKRA コネクタ (シングル FAKRA コネクタ側) に接続します。次に、ケーブルのミニ FAKRA コネクタを RQX-59G の GMSL2 ポートに挿入します。
3. FAKRA コネクタおよびミニ FAKRA コネクタのロックピエロの方向が正しいことを確認してください (図 1 ～ 3)。

図 2.1　RosCube RQX-59G

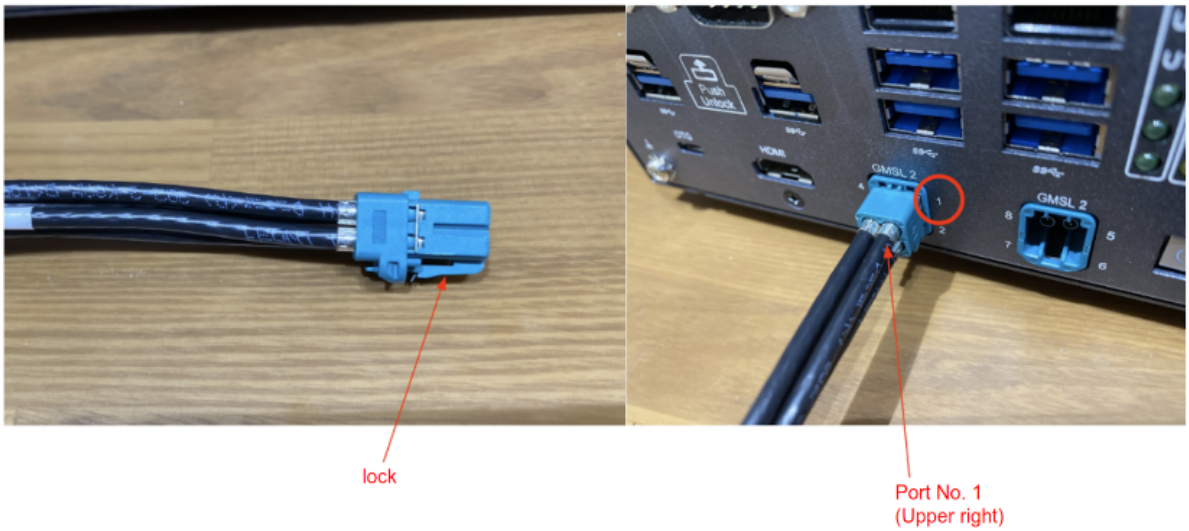

図 2.2　コネクタ

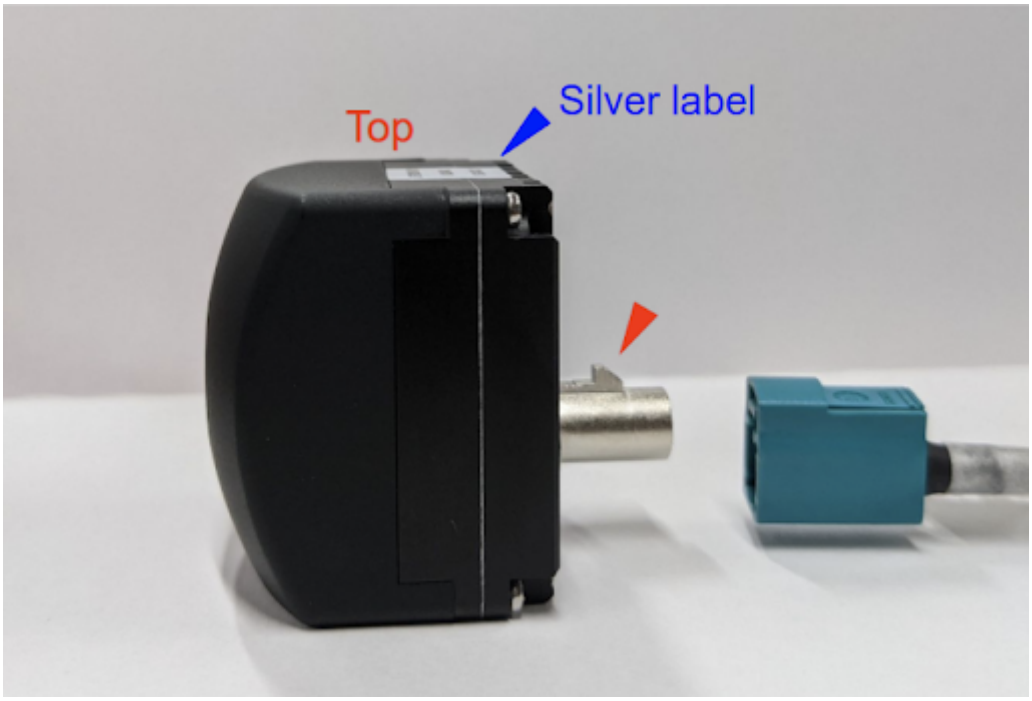

図 2.3　Fakra ケーブル挿入方向

2.2 電源を入れてカメラの画像ストリーム出力を確認します

2.2.1 電源オン

RQX-59G の電源を投入し、パワーオン LED が青色に点灯することを確認します。

2.2.2 ログイン

起動ウィンドウで、パスワードを入力してログインします。

デフォルトの設定は以下のとおりです。

ユーザー	ロス
パスワード	アドリンククロス

2.2.3 カメラドライバーのインストール

注釈: 2024-04-01 現在、RQX-59G を TIER IV カメラで使用するには、専用のドライバー deb パッケージが必要です。

Github からカメラ ドライバー deb パッケージを入手します。

提供されたドライバー パッケージ ファイル (tier4-camera-gmsl_1.4.2_59g_arm64.deb) を RQX-59G 内の任意のディレクトリ (例 ~/driver) にコピーします。次に、以下のコマンド ラインの指示に従います。

1. apt パッケージを更新しています。インターネット接続が必要です。次に、「apt install」コマンドを使用してドライバーをインストールします。*.dtbo ファイルが /boot に生成されていることを確認してください。

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

2. 各ポートにカメラを割り当てるには、configure-by-hardware.py スクリプトを使用できます。詳細については、カメラ ドライバーの README ページ を参照してください。ユーザーは、各ユーザーの設定に適切なオーバーレイ コマンドを指定する必要があります。

```
# 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
```

/opt/nvidia/jetson-io/config-by-hardware.py -l を実行して、利用可能なオーバーレイ オプションを確認できます。

```
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
```

たとえば、すべての GMSL ポートを C2 カメラに割り当てるには、overlay コマンドは次のようにする必要があります。

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

C1 と C2 の両方に GMSL ポートを割り当てるには、overlay コマンドを次のようにする必要があります。

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

この場合、ポート 1、2、5、6 は C1 に割り当てられ、ポート 3、4、7、8 は C2 に割り当てられます。

デバイス オーバーレイの詳細については、デバイス ドライバー GitHub リポジトリを参照してください。

2.2.4 RQX-59G がカメラを認識するか確認する

ターミナル ウィンドウを開き、次のコマンドを入力します。/dev/videoX が返された場合、カメラはビデオ デバイスとして正しく認識されています。

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

2.2.5 GStreamer を使用してカメラ出力を視覚化する

ターミナル ウィンドウを開き、次のコマンドを入力します。Gstreamer が起動し、新しいウィンドウにカメラ画像ストリームが表示されます（図 1)。4、図 5)。

さまざまな GStreamer の例については、GStreamer コマンドの例 も参照してください。

ケース 1: 1 台の C1 カメラが接続されている

```
# 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
```

ケース 2: 8 台の C1 カメラが接続されている

```
# 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 \
```

ケース 3: C2 カメラ 1 台が接続されている場合

```
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
```

複数の C2 カメラを接続する場合

RQX-59G では、8 つの GMSL ポートに対して 4 つのデシリアライザが実装され、2 つの GMSL ポートが 1 つ

のデシリアライザとその帯域幅を共有します。

2 つの C2 カメラが 1 つのデシリアライザーで動作できることを確認しましたが、特に最大 4 つのカメラを接続している場合は、デシリアライザーごとに 1 つのカメラだけを使用するのが最善です。

例えば：

- ポート 1 および 2 ... 推奨されません
- ポート 3 および 4 ... 推奨されません
- ポート 1 とポート 3 ... 推奨
- ポート 1 および 5 ... 推奨

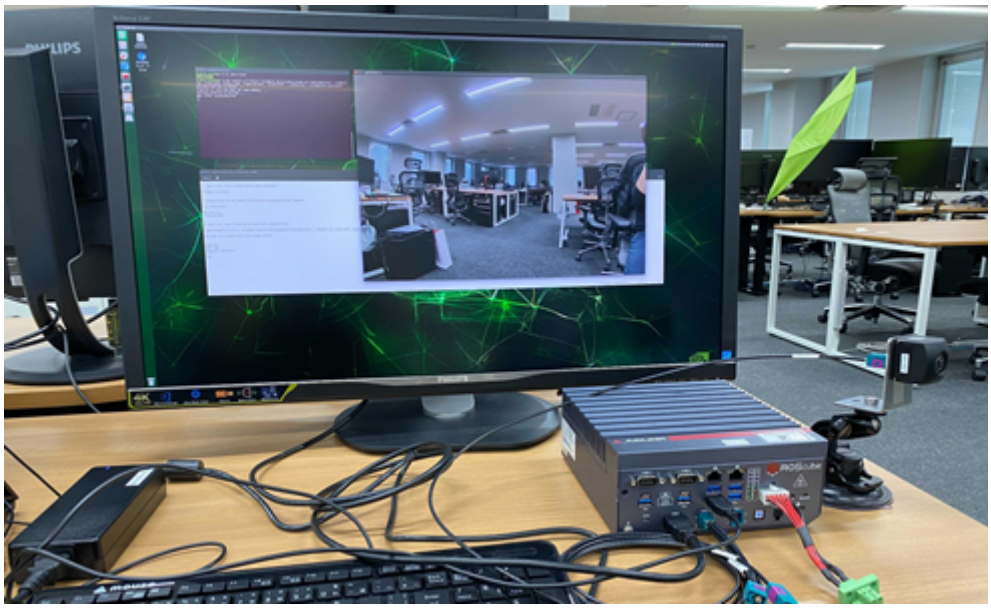

図 2.4　視覚化されたカメラ出力

2.3 映像データを rosbag（ROS 形式のログデータ）として記録

2.3.1 ROS melodic と gscam (GStreamer 用の ROS カメラドライバー) をインストールします。

コマンドラインの指示に従って、ROS melodic と gscam をインストールします。

> **注意：** インストールコマンドは変更される可能性があります。次のコマンドが機能しない場合は、ROS 公式ドキュメント を参照してください。

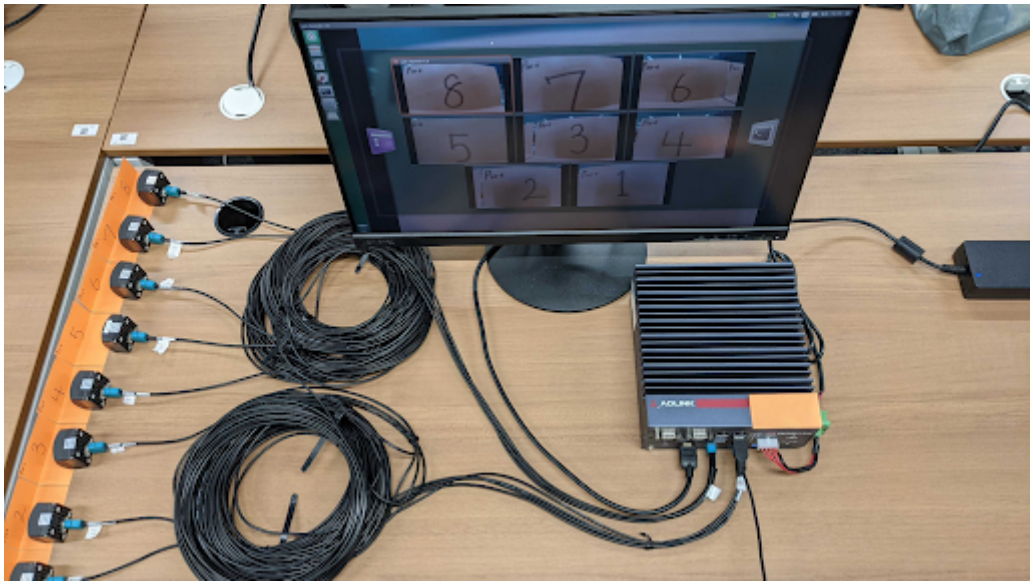

図 2.5　視覚化されたカメラ出力 (x8 カメラ)

```
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 gscam を実行し、画像トピックを視覚化します。

指示に従って gscam を実行します。

1. 新しいターミナルを開き、次のコマンドを実行して roscore を起動します

   ```
   roscore
   ```

2. 別のターミナルを開き、次のコマンドを実行します

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

3. トピック（ROS 形式のメッセージデータ）camera/image_raw が公開されたので、rqt_image_view でトピックを可視化します。rqt_image_view を起動するには、別のターミナルを開いて次のコマンドを実行します。

   ```
   rosrun rqt_image_view rqt_image_view
   ```

このコマンドは、画像トピックを視覚化するためのウィンドウを開きます。ウィンドウ左上のプルダウンメニューをクリックして、camera/image_raw を選択します (図 6)。

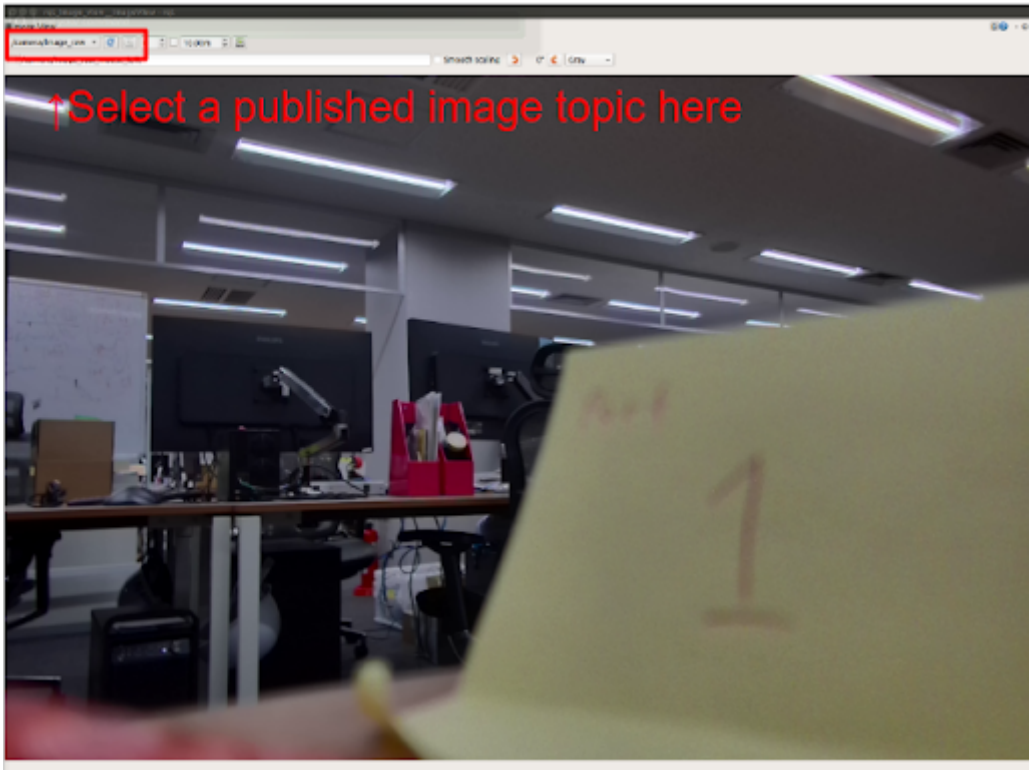

図 2.6　公開された画像トピックを選択してください

複数のカメラで gscam を実行するには、次の XML メッセージを含むテキスト ファイルを作成し、~/2cameras.launch として保存します。この .launch ファイルは 2 台のカメラでのストリーミング用です。

```
<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"/>
```

(次のページに続く)

(前のページからの続き)

```
   <param name="sync_sink" value="true"/>
</node>
</launch>
```

注意: C1 の設定です。C2 の場合は、必要に応じて「width」、「height」、「framerate」を変更してください。

/home/ros/2cameras.launch を作成したと仮定して、次のコマンドを実行して複数のカメラで gscam を実行します。

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

注釈: さらにカメラを追加するには、適切なノード名、namespace(ns)、camera_name、frame_id、および gscam_config を使用してノードを追加します。

2.3.3 ロスバッグを記録する

gscam の実行中に、次の方法で rosbag (ROS 形式のログ ファイル) を記録できます。

```
rosbag record -a
```

rosbag info で rosbag の情報を確認することもできます。

```
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 構成

ドライバー構成を変更するには、構成ファイル /etc/modprobe.d/tier4-*.conf を編集します。(次のセクションを参照してください)。設定ファイルにはデフォルトで次の行が含まれています。(C1 の場合です)

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

設定を適用するには、編集後に RQX-59G を再起動します。

2.4.1 ドライブモードを切り替える

> 注意: カメラをスレーブモードで動作させるには、RQX-59G からフレーム同期信号を入力する必要があります。カメラをスレーブ モードで駆動するには、Shutter triggering over GMSL2 をチェックしてください。
>
> RQX-59G からトリガー信号を生成する方法については、トリガー入力に関する ADLINK のドキュメントを参照してください。

C1 の場合

設定ファイルを編集することで、カメラの駆動モードをマスター モードからスレーブ モード (またはその逆) に切り替えることができます。

- マスターモード: 30fps のフリーランニングモード (デフォルト)
- スレーブモード：シャッタートリガーモード、シャッタータイミング、フレームレートは FSYNC 信号周波数によって調整可能（RQX-59G で設定可能）

モードを切り替えるには、「/etc/modprobe.d/tier4-isx021.conf」の trigger_mode を編集します。(以下の事例を参照)

ドライブモード	フレームレート	トリガーモード=
マスター	30fps	0
スレーブ	FSYNC 入力周波数に依存	1

C2 の場合

C2 の場合、/etc/modprobe.d/tier4-imx490.conf を編集することで以下のモードを選択することができます。次の表に、使用可能な設定を示します。

ドライブモード	フレームレート	トリガーモード=
マスター	10fps	0
スレーブ	10fps	1
マスター	20fps	2
スレーブ	20fps	3
マスター	30fps	4
スレーブ	30fps	5

2.4.2 レンズ歪み補正 (LDC) を有効/無効にする

注釈: この設定は C1 と C2 に共通です。

LDC を有効にするには、フラグ `enable_distortion_correction=1` (デフォルト) を設定します。LDC を無効にするには、フラグ「enable_distortion_correction=0」を設定します。

2.4.3 自動露出 (AE) を有効/無効にする

注釈: この設定は C1 と C2 に共通です。

AE を有効にするには、フラグ `enable_auto_exposure=1` (デフォルト) を設定します。AE を無効にするには、フラグ「enable_auto_exposure=0」を設定します。

2.4.4 露光時間の設定

C1 の場合

C1 の場合、ユーザーは `shutter_time_min`、`shutter_time_mid`、および `shutter_time_max` という名前の変数を介して 3 種類の露光時間を指定できます。C1 の実際の露光時間は、照度に応じてこれら 3 つの値 (および線形補間値) を推移します。変数の値はマイクロ秒単位で指定する必要があります。任意の照度条件下で露光時間を固定したいユーザーは、これらの変数にまったく同じ値を設定できます。たとえば、「/etc/modprobe.d/tier4-isx021.conf」の次の設定では、露出時間が 11 ミリ秒に固定されます。

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

C2 の場合

同様に、C1、C2 にも、shutter_time_min および shutter_time_max という名前の変数を介して 2 種類の露光時間を設定する機能があります。これらの変数の値の単位もマイクロ秒です。たとえば、「/etc/modprobe.d/tier4-imx490.conf」の次の設定では、露出時間が 11 ミリ秒に固定されます。

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

2.5 GMSL2 経由のシャッタートリガー

スレーブ モードでのカメラ操作には、GMSL2 を介したトリガー信号入力 (FSYNC 入力) が必要です。FSYNC パラメータと GPIO 設定の構成方法については、ADLINK のドキュメント を参照してください。

2.5.1 スレーブモードでカメラを駆動するための準備

端末で「i2cget -f -y 2 0x66 0x01」を実行して RQX-59G の HW バージョンを確認し、以下の手順に従ってください。

返される値が少なくとも 0x24 であることを確認してください。

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

2.5.2 C1 の FSYNC 周波数を入力

30fps より遅い任意の周波数を入力できます。

2.5.3 C2 の入力 FSYNC 周波数

C2 の場合、許可される FSYNC 周波数はドライブ モードによって異なります。以下のリストを参照してください。

- 30fps モード (trigger_mode=5) の場合: 15 < f <= 30 fps
- 20fps モード (trigger_mode=3) の場合: 10 < f <= 20 fps
- 10fps モード (trigger_mode=1) の場合: 5 < f <= 10 fps

2.5.4 FPS チェック

視覚化されたビデオで FPS 設定を確認するには、次のコマンドを実行してください。

- カメラはポート 1 にのみ接続されています

```
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
```

- カメラはポート 1 と 2 に接続されています

```
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 参照

- ADLINK のドキュメント

3

Nvidia Jetson AGX Orin/Xavier 開発者キット用 TIER IV カメラ スタートガイド

注意: このドキュメントは NVIDIA Jetson 開発キット Xavier/Orin のユーザーを対象としています。

3.1 必要な機材のリスト

- Nvidia Jetson AGX Orin 開発者キットまたは Nvidia Jetson AGX Xavier 開発者キット x 1
- デシリアライザキット
  - LI-JXAV-MIPI-ADPT-4CAM × 1
  - LI-GMSL2-IPX-DESER x 1〜4 (カメラ 2 台につき 1 個)
  - FAW-1233-03 x 1〜4 (カメラ 2 台につき 1 個)
- GMSL2 ケーブル (FAKRA C-FAKRA C) x 1〜8 (カメラ 1 台につき 1 本)
- TIER IV 車載 HDR カメラ C1 x 1~6 または C2

3.2 ハードウェアの接続

Jetson AGX Orin/Xavier Developer Kit（以下、Devkit）の電源が切れていることを確認してください。確認後、以下のようにデシリアライザーキットとカメラを取り付けます。次の手順を参照し、各コンポーネントを適切に接続してください。

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

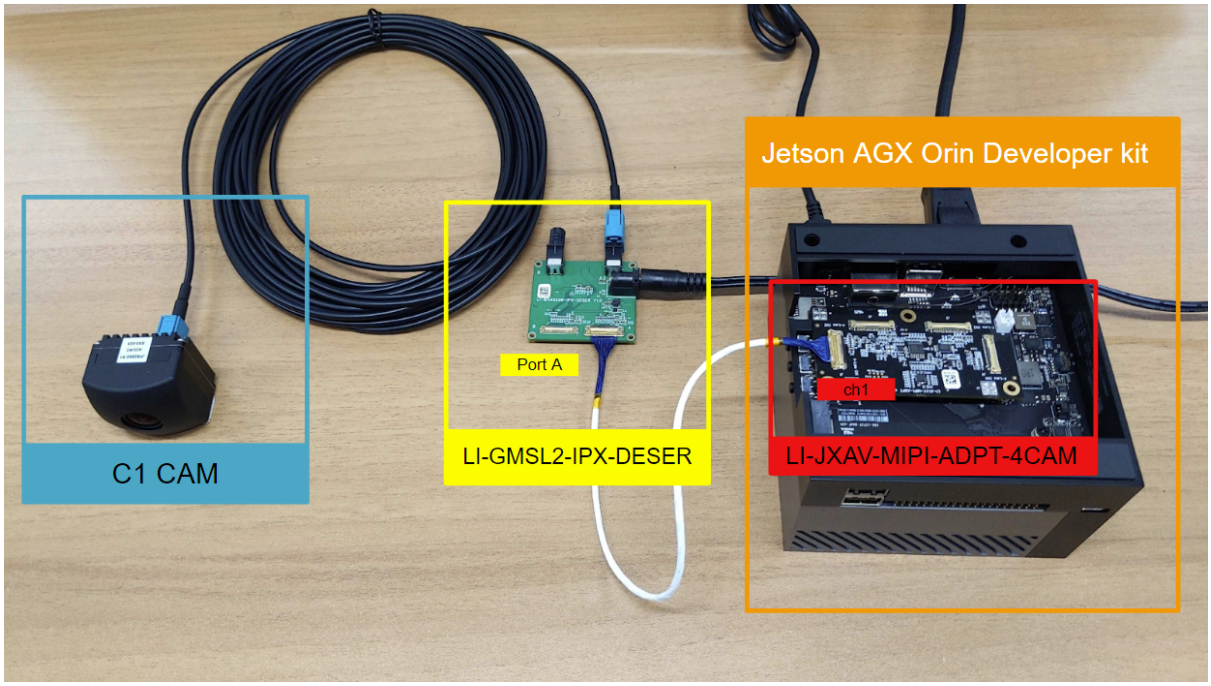

図 3.1　各コンポーネントの接続例 (Orin Devkit、カメラ 1 台)

3.2.1 デシリアライザキットとカメラの接続

LI-JXAV-MIPI-ADPT-4CAM を開発キットの背面にあるコネクタに接続します。

注釈: LI-JXAV-MIPI-ADPT-4CAM は Jetson AGX Xavier 開発者キット用に設計されているため、Jetson AGX Orin 開発者キットに接続するとエンクロージャとわずかに干渉する可能性があります。慎重に接続するか、問題を防ぐために リンクに記載されているアダプター の使用を検討してください。

次に、FAW-1233-03 ケーブルを LI-MAX9296-IPX-DESER の CH1 に挿入します。

前に接続したケーブルのもう一方の端を LI-MAX9296-IPX-DESER のポート A に挿入します。カメラ１つの評

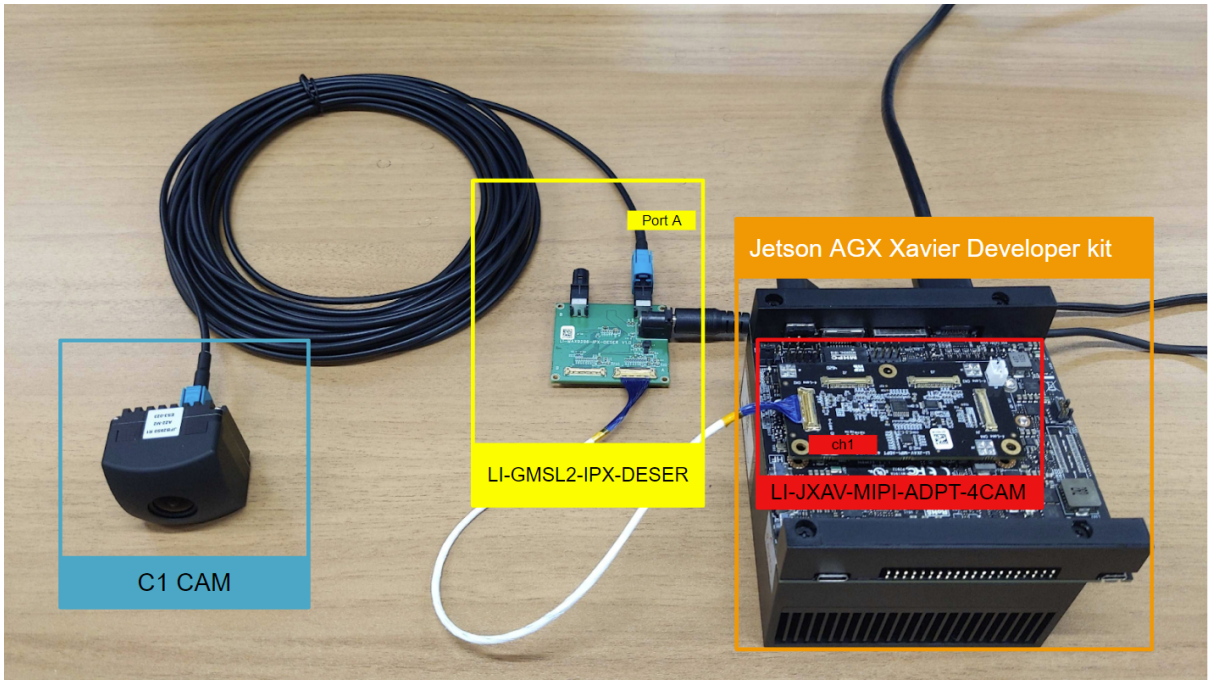

図 3.2　各コンポーネントの接続例（Xavier Devkit、カメラ 1 台）

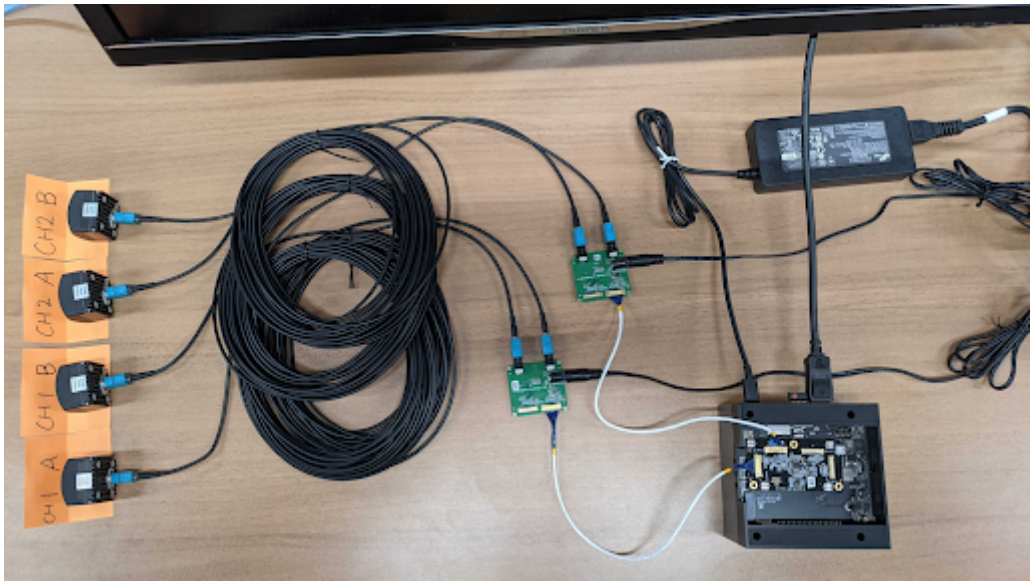

図 3.3　ケーブル接続例（カメラ 4 台）

価の場合、GMSL2 ケーブルの FAKRA コネクタを LI-MAX9296-IPX-DESER のポート A に挿入します。2 台のカメラを接続するには、FAKRA コネクタをポート A とポート B に接続します。

> 注意：デシリアライザには 12V の電源を外部 AC アダプタから別に供給する必要があります。

カメラを GMSL2 ケーブルのもう一方の FAKRA コネクタに接続します。銀色のラベルが付いている面が上を向くようにしてください。

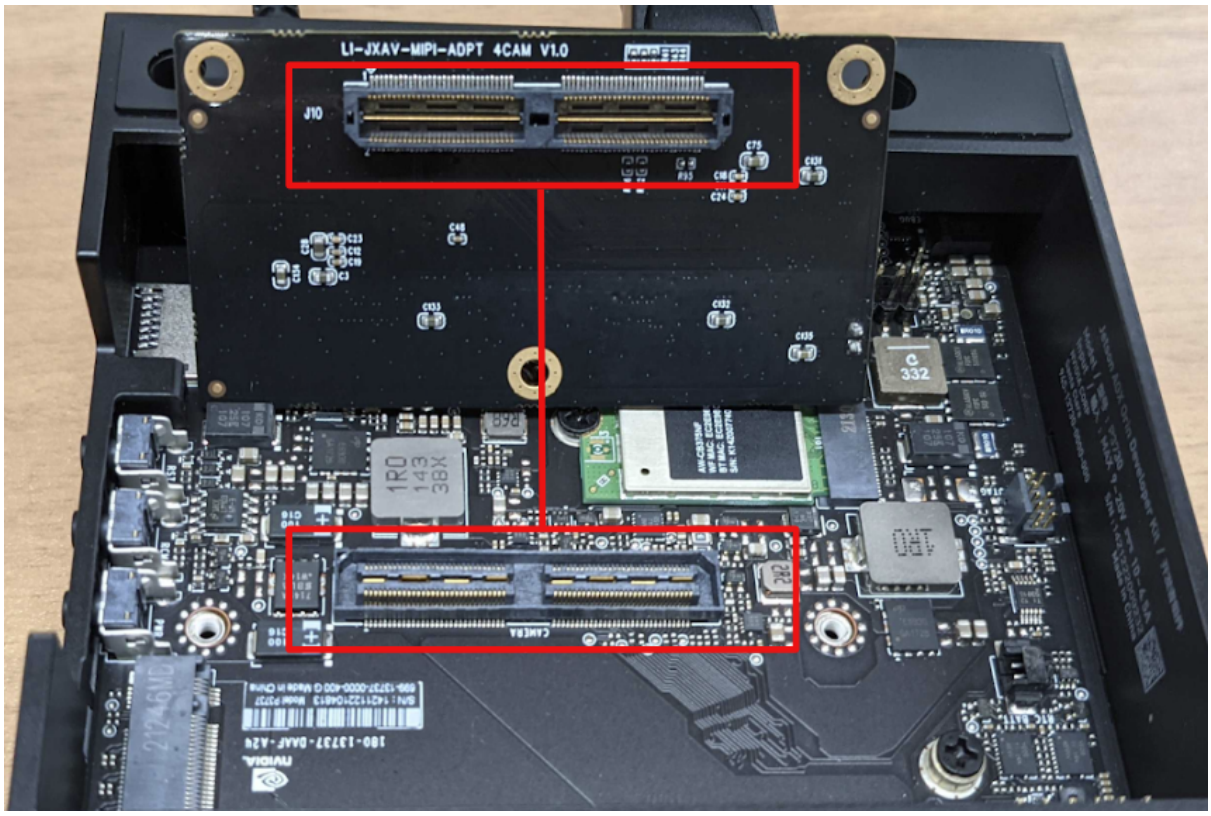

図 3.4　LI-JXAV-MIPI-ADPT-4CAM コネクタ

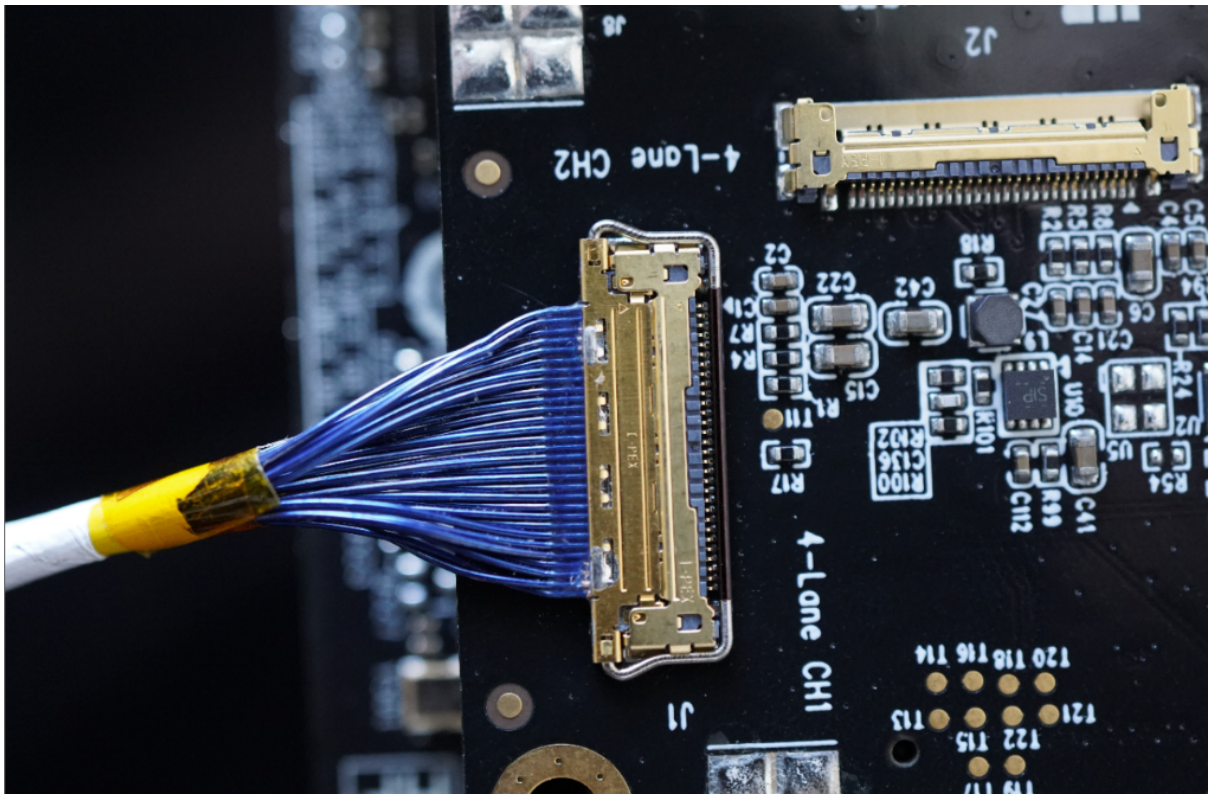

図 3.5　FAW-1233-03 のケーブル接続例

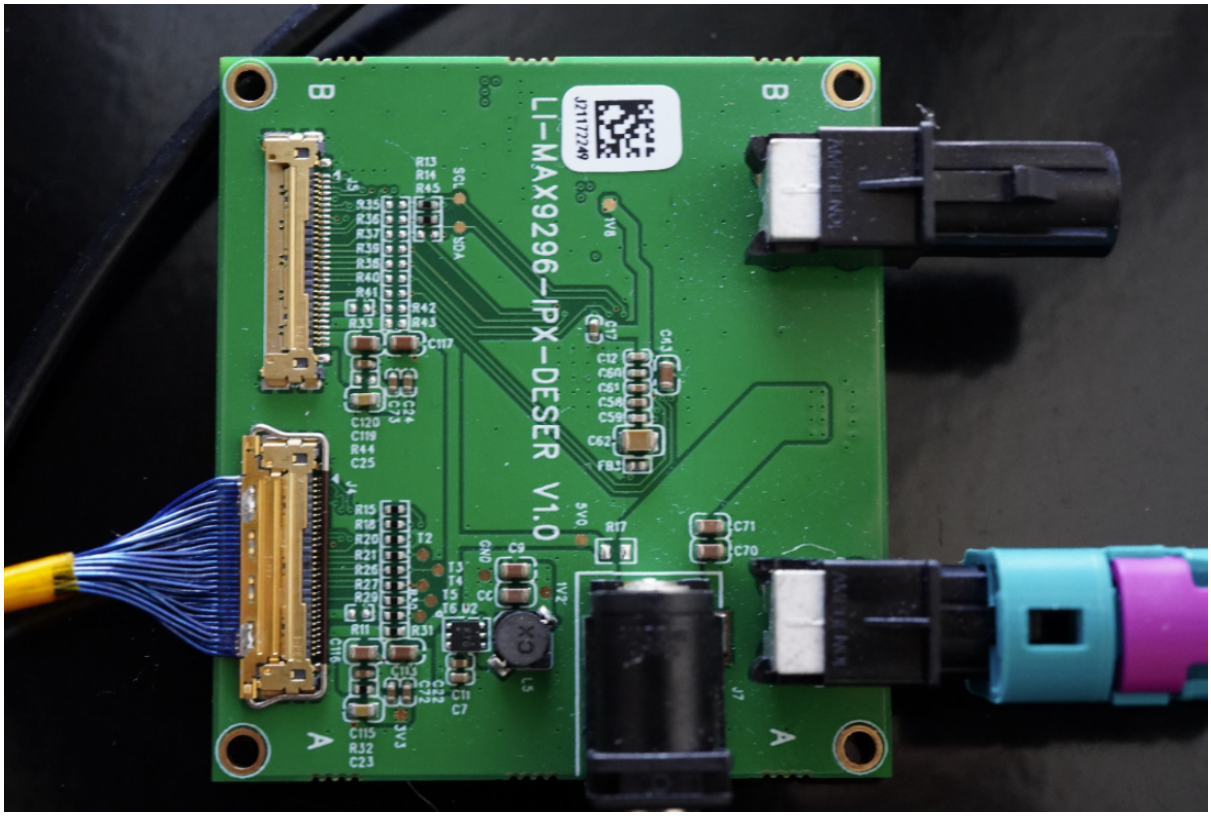

図 3.6　LI-MAX9296-IPX-DESER のケーブル接続例

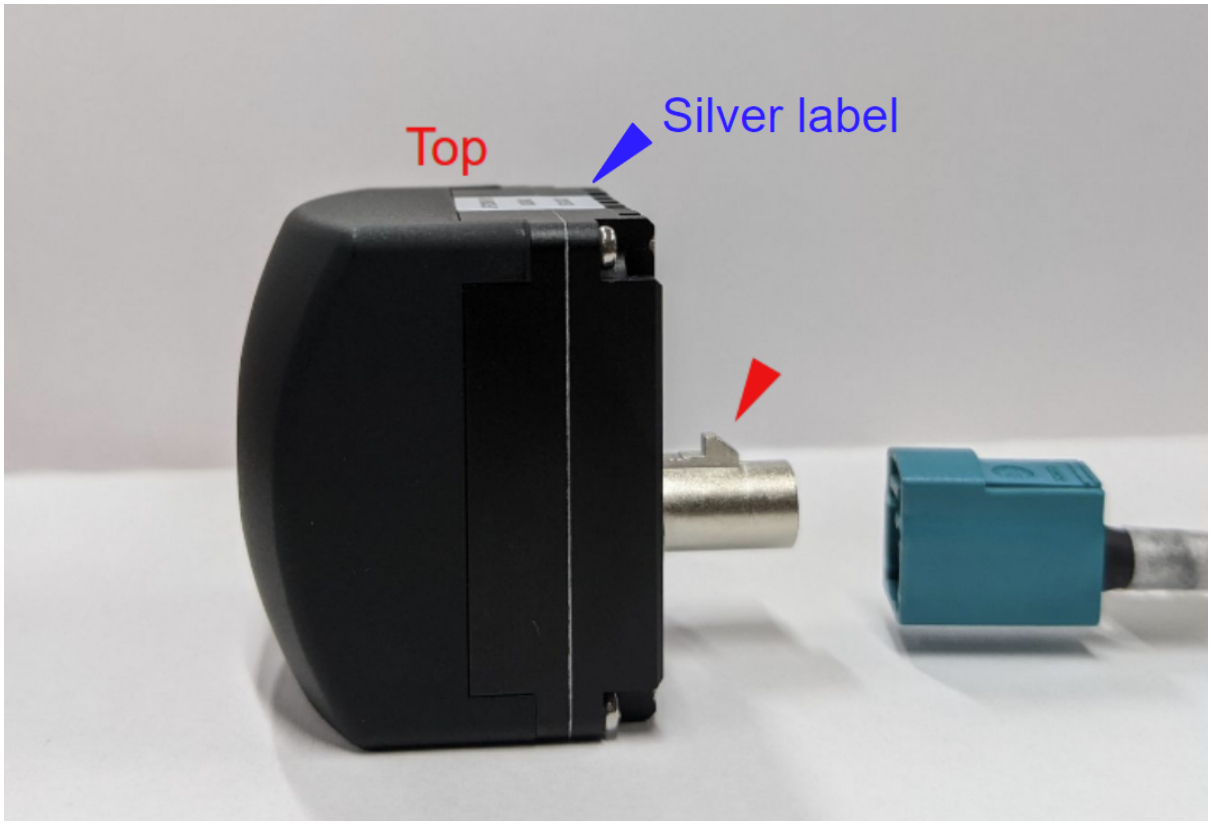

図 3.7　カメラの向きとケーブルの挿入方向 (銀ラベル側を上)

3.2.2 その他のハードウェア接続

必要に応じて、マウス、キーボード、HDMI ケーブルなどを Devkit に接続します。各種作業に必要な機器は別途ご用意ください。

3.3 ソフトウェアのセットアップ

3.3.1 ソフトウェア要件

開発キットが次のドライバーのソフトウェア要件を満たしているかどうかを確認してください。

ソフトウェア	バージョン	ダウンロードリンク
JetPack	5.1.1	Nvidia Jetson quickstart guide
tier4-gmsl-camera	1.2.1 以降	TIER IV Camera driver repository

注意: C2 カメラを使用する場合はドライバを v1.4.1 以上にアップデートしてください。

ターゲットの Devkit がソフトウェア要件を満たしていない場合は、JetPack のインストールに進んでください。JetPack のインストールには 2 つの方法があります。

1. 手動インストール: 提供されたリンクから JetPack データをダウンロードし、手動でインストールを実行します。この場合、L4T 35.2.1 をインストールして下さい。
2. Nvidia SDK マネージャ: インストールには Nvidia SDK マネージャを使用します。

潜在的なインストールの問題を回避するために、インストールには SDK マネージャー (方法 2) を使用することをお勧めします。

注釈: Ubuntu22.04 をホストとして使用している場合、JetPack をインストールするのに SDK マネージャーは使用できません。

3.3.2 ドライバーパッケージのインストール

この手順は、Devkit で実行する必要があります。

ドライバー パッケージ ファイル (提供されたリンクから最新バージョンを入手してください) を任意のディレクトリにコピーします。コピーが完了したら、以下のコマンドを入力します。※ xxx はドライバーのバージョンに置き換えてください。ダウンロードしたドライバーのバージョンを指定します。

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

ドライバーをインストールした後、カスタムの dtb ファイルを生成して、GMSL ポートごとにカメラの割り当てを定義する必要があります。詳細はカメラドライバーのインストールを参照してください。コマンドのオーバーレイの例については、ADLINK ROSCube のスタート ガイド を参照してください。

```
# 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
```

完了したら、Devkit を再起動してください。

ドライバーをインストールした後、i2c 書き込みエラーまたは同様の問題に関連するエラー メッセージがカーネル メッセージ (syslog または dmesg コマンド出力) に表示される場合があります。これらのエラーは通常、カメラが接続されていないポートに関連しており、問題なくカメラ画像の取得を続行できます。

エラーメッセージの例:

```
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_
```

(次のページに続く)

(前のページからの続き)

```
↪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
```

NVMe ストレージ（M.2 SSD）使用時

NVMe ストレージにオペレーティング システムをインストールしている場合、システムはオンボード ストレージから構成を読み取っている可能性があることに注意してください。再起動後もカメラ デバイスが表示されない場合は、前述のセットアップ手順に加えて、オンボード ストレージをマウントし、次のファイルをオンボード ストレージの /boot ディレクトリにコピーする必要がある場合があります。

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

3.4 カメラ画像の取得

3.4.1 カメラデバイスを確認する

ターミナルを開き、次のコマンドを実行して、カメラが v4l2 デバイスとして認識されているかどうかを確認します。ビデオデバイスの数と接続されているカメラの数が一致していれば、カメラの認識チェックは完了です。

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

3.4.2 カメラ画像データ取得

ターミナルを開いて以下のコマンドを実行すると、Gstreamer を使用した画像取得が開始されます。新しいウィンドウが開き、カメラ画像が表示されます。

例 1：C1 カメラによる評価

```
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
```

図 3.8　カメラ画像データ取得例

例 2: 4 台の C1 カメラによる評価

```
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
```

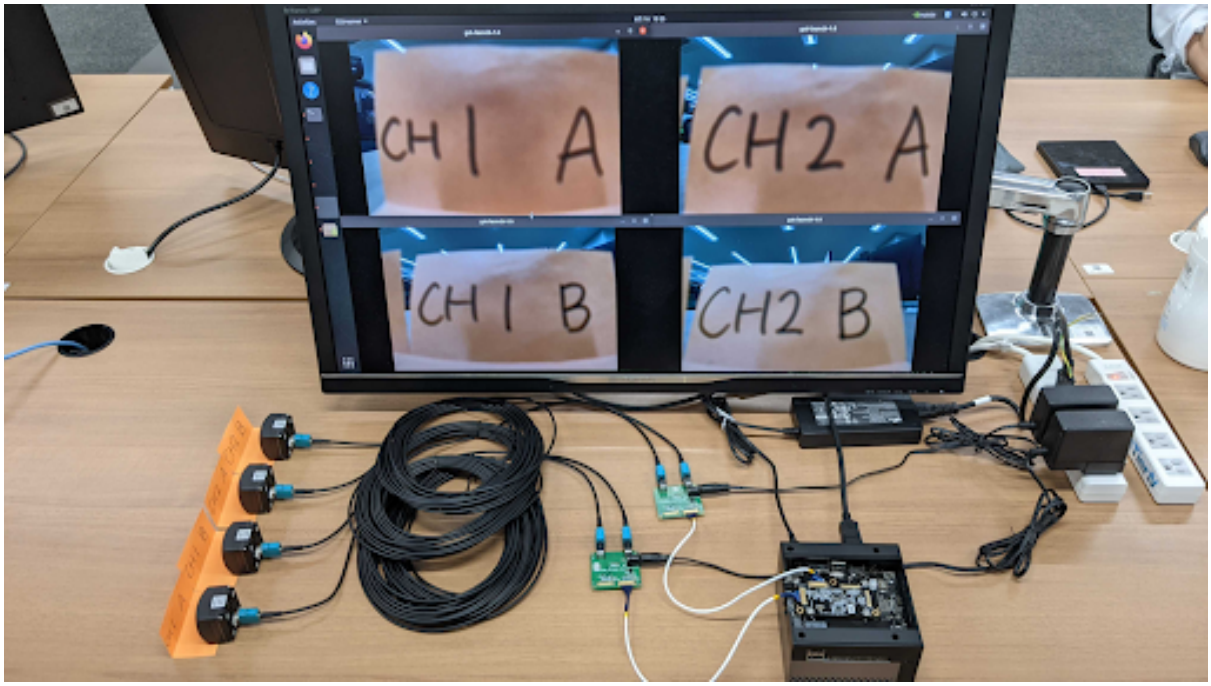

図 3.9　カメラ画像データ取得例（カメラ 4 台）

例 3：C2 カメラによる評価

```
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
```

注釈: gstreamer 起動スクリプト も使用できます。

注釈: 「no element "v4l2src"」というエラー メッセージが表示された場合は、ディレクトリ ~/.cache/gstreamer-1.0 を削除して、コマンドを再試行してください。

カメラの起動順序の制約 (v1.1.1 より前のドライバー バージョンの場合)

v1.1.1 より前のバージョンのドライバーを使用している場合は、起動のための特定の手順に従ってください。ドライバー バージョン v1.2.1 以降の場合、順序や制約の要件はありません。

複数のカメラを使用する場合、すべてのカメラを同時に起動する必要があります。

たとえば、2 台のカメラを使用している場合は、次のワンライナー コマンドを使用して画像の取得を開始する必要があります。

```
```
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 \
```
```

これらの分割されたコマンドは機能しない可能性があります。

```
```
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 画像データの記録

Gstreamer を使用すると、画像データを H.265 でエンコードしながら記録できます。必要に応じてビットレート、ファイル名、その他のパラメータを変更してください。

例 1: C1 カメラでの録画

```
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
```

例 2: 2 台の C2 カメラで録画する

```
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
```

(次のページに続く)

(前のページからの続き)

```
↪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 ROS2 でカメラ画像データを利用する

3.6.1 ROS2 のインストール

Jetpack が提供する最新のディストリビューションは Ubuntu 20.04 です。Ubuntu 20.04 と互換性のある ROS2 ディストリビューションは、Galaxy です。ただし、Galaxy はサポート終了 (EOL) に達しているため、ソースからビルドするか、Docker を使用して Humble をインストールしてください。

- ソースから Humble をインストールする場合
- Docker を使う場合 (推奨)

Jetson Orin (Ubuntu 20.04) 上のソースから ROS2 Humble をビルドすると、OS の互換性がないために失敗する可能性があります。Docker の使用をお勧めします。

3.6.2 ノードのインストール (未インストールの場合)

v4l2_camera ノードの使用をお勧めします。インストール手順については、Github リポジトリ を参照してください。

3.6.3 v4l2_camera ノードの起動と画像データの視覚化のトピック

ターミナルを開き、次のコマンドで v4l2_camera ノードを起動します。

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

注釈: C2 の場合は image_size の設定を [2880,1860] に変更してください。

画像トピック (ROS 形式) /image_raw が公開されます。rqt_image_view を使用して視覚化します。新しいターミナルを開き、次のコマンドを実行して rqt_image_view を起動します。

```
ros2 run rqt_image_view rqt_image_view
```

3.6.4 Rosbag の記録

v4l2_camera ノードの実行中に、次のコマンドを使用して rosbag (ROS 形式のログ ファイル) を記録できます。

```
ros2 bag record -a
```

3.7 設定変更

カメラドライバーの設定を変更するには、/etc/modprobe.d/tier4-*.conf ファイルを編集します。設定ファイルには次の行が含まれています (tier4-isx021.conf の場合)。

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

これらのフラグを編集した後、変更を有効にするためにシステムを再起動します。

3.7.1 駆動モードの切り替え（フリーラン/トリガー）

> **警告:** Jetson Orin/Xavier 開発キットには FSYNC の入力機能がないため、Jetson Orin/Xavier 開発キットではトリガーモードはサポートされていません。フリーランモードからの変更は意図しない動作をする可能性がありますので、行わないでください。

C1 の場合

駆動モードは 30fps のフリーランモードに固定されます。他のドライブモードはサポートされていません。

C2 の場合

ドライブモード	フレームレート	trigger_mode=
フリーラン	10fps	0
フリーラン	20fps	2
フリーラン	30fps	4

3.7.2 レンズ歪み補正 (LDC) 機能の有効化/無効化

enable_distortion_correction フラグの値を変更することで、レンズ歪み補正 (LDC) 機能を有効または無効にできます。

レンズ歪み補正を有効にするには、enable_distortion_correction=1 を設定します。レンズ歪み補正を無効にするには、enable_distortion_correction=0 を設定します。

3.7.3 露光時間の固定

C1 の場合

C1 には 3 種類の露光時間があり、変数 shutter_time_min, shutter_time_mid, shutter_time_max を使用することで各々を設定することが可能です。実際の露光時間は、周囲の明るさに応じてこれら３つの３変数値及びそれらの線形補間値の間を遷移します。各変数値の単位はマイクロ秒です。異なる環境光条件下であっても露光時間が変化しないように固定するには、これらの変数に対してすべて同じ値を設定してください。例えば、/etc/modprobe.d/tier4-isx021.conf に対して以下の設定を記述することで露光時間を 11 ms に固定することができます。

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

C2 の場合

C1 と同様に、C2 では shutter_time_min 及び shutter_time_max の２種類の露光時間を設定することが可能です。各変数値の単位はマイクロ秒です。例えば、/etc/modprobe.d/tier4-imx490.conf に対して以下の設定を記述することで、露光時間を 11 ms に固定することができます。

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

3.7.4 その他のパラメータ調整について

T4cam-ctrl ユーザーマニュアル を参照してください。

4

Vecow EAC-5000 用 TIER IV カメラ スタート ガイド

注意: この文書は Vecow EAC-5000 のユーザーを対象としています。

4.1 準備

4.1.1 必要アイテム

- Vecow EAC-5000
  - Jetpack 5.1.1 (Linux for Tegra 35.3.1) がインストールされていること
- GMSL2 同軸ケーブル
- TIER IV 車載 HDR カメラ C1 または C2

4.1.2 カメラの接続

1. まず、EAC-5000 の電源が切れていることを確認してください。
2. FAKRA ケーブルを使用してカメラを EAC-5000 に接続します。

4.2 電源投入とログイン

4.2.1 電源投入

EAC-5000 の電源を入れ、パワーオン LED が青色に点灯することを確認します。

4.2.2 ログイン

起動ウィンドウで、パスワードを入力してログインします。

デフォルトの設定は以下のとおりです。

ユーザー	ubuntu
パスワード	ubuntu

4.3 カメラドライバーのインストール

注釈: 他の ECU とはインストール手順が異なりますので注意してください

現在、EAC-5000 には、Out-of-tree モジュールのビルドとロードに関して問題があります。

複雑なカーネルの処理を避けるために、TIER IV は予めにビルドされたドライバーとインストール スクリプトを提供します。

GitHub リリース ページ からプリビルド ドライバーをダウンロードしてください。

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. インストール スクリプトは次の手順を実行します。
   - カメラドライバーの設定ファイルのインストール
   - 事前に構築されたカメラドライバーのインストール
   - 事前に構築されたデバイス ツリー BLOB のインストール
   - Create a new boot entry in the `/boot/extlinux/extlinux.conf`
3. インストール スクリプトを実行した後、`/opt/nvidia/jetson-io/configure-by-hardware.py` を実行できます。次のプロンプトが表示されることを確認してください。

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

4. configure-by-hardware.py を使用して GMSL ポートの割り当てを選択します。詳細については、GitHub README を参照してください。

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

このコマンドは以下を出力します。

```
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
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!
```

次に、GMSL ポートの割り当てを選択してください。

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

上記のコマンドは、すべての GMSL ポートに C1 カメラを割り当てます。他の割当を設定したい場合は 2= 以降を修正して下さい。

5. 再起動

```
sudo reboot now
```

6. カメラデバイスファイル (/dev/video*) が作成されていることを確認してください

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

インストール スクリプトが完了した後 extlinux.conf が適切な形式で有効であることを確認する必要があります。この手順では手動での編集が必要になる場合があります。

4.4 GStreamer を使用してカメラ出力を表示する

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

GStreamer コマンド例も参照してください。

4.5 制約事項

> **注意:** EAC-5000 はトリガ信号 (FSYNC) の入力を通常構成でサポートしていません。TIER IV カメラのフレーム同期機能は EAC-5000 との組み合わせでは使用することができません。

5

Connect Tech Anvil 用 TIER IV カメラスタートガイド

注意: このドキュメントは Connect Tech Anvil のユーザー向けです

5.1 準備

5.1.1 必須項目

- Connect Tech Anvil
  - BSP: AGX Orin L4T r35.4.1
- Fakra 同軸ケーブル
- TIER IV 車載 HDR カメラ C1 または C2

5.1.2 カメラの接続

1. まず、アンビルがオフになっていることを確認します。
2. FAKRA ケーブルを使用してカメラを Anvil に接続します。

5.2 電源投入とログイン

5.2.1 電源投入

Anvil の電源をオンにします。

5.2.2 ログイン

起動ウィンドウで、パスワードを入力してログインします。

デフォルトの設定は以下のとおりです。

ユーザー	nvidia
パスワード	nvidia

5.3 カメラドライバーのインストール

注釈: インストール手順は他の ECU とは異なります

現在、Anvil ではツリー外のモジュールのビルドとロードに問題があります。

複雑なカーネルの処理を避けるために、TIER IV は予めにビルドされたドライバーとインストール スクリプトを提供します。

弊社の GitHub リリースページからプレビルドドライバーをダウンロードしてください。

1. tarball を抽出し、`./install.sh` を実行してカメラ ドライバーをインストールします。

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

2. インストール スクリプトは次の手順を実行します。
   - カメラドライバーの設定ファイルのインストール
   - あらかじめ構築されたカメラドライバーのインストール
   - 事前に構築されたデバイスツリー BLOB のインストール
   - `/boot/extlinux/extlinux.conf` に新しいブートエントリを作成
3. インストール スクリプトを実行した後、`/opt/nvidia/jetson-io/configure-by-hardware.py` を実行してください。次のプロンプトが表示されます。

```
[*] 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. configure-by-hardware.py を使用して GMSL ポート割り当てを選択します。詳細については、GitHub README を参照してください。

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

このコマンドは以下を出力します。

```
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
```

次に、GMSL ポートの割り当てを選択します。たとえば、すべての GMSL ポートを C1 に割り当てる場合は、次のように設定します。

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

上記のコマンドは、すべての GMSL ポートに C1 カメラを割り当てます。設定に応じて 1= の後を変更してください。configure-by-hardware.py が正常に終了した場合は、次の出力が表示されます。

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

5. 再起動

```
sudo reboot now
```

6. カメラデバイスファイル（/dev/video*）が作成されていることを確認してください

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

インストール スクリプトが完了したら、`extlinux.conf` が適切に形成され、有効であることを確認する必要があります。この手順では、手動での編集が必要になる場合があります。

5.4 GStreamer を使用したカメラ出力の視覚化

GStreamer を使用して画像を出力する手順は他の ECU と同じです。マニュアルを参照してください。

GStreamer コマンドの例も参照してください。

5.5 制限

> **注意:** Anvil との組み合わせにおけるカメラ同期機能は、現時点でサポートでされませんが、今後サポートされます。

5.6 参考資料

- Anvil ユーザーガイド
- Anvil リリースノート

6

センサー フュージョン開発キット スタート ガイド

6.1 ハードウェアのセットアップ

最初のステップとして、センサーと ECU を含むハードウェアを準備します。

6.1.1 ハードウェア構成例

このチュートリアルでは、次のハードウェア構成が使用されます。

- ECU セットアップ
  - x86 ベースの ECU: ADLINK AVA-3510
  - Jetson ベースの ECU: ADLINK RQX-58G
- センサーのセットアップ
  - 構成例 1
    - カメラ: TIER IV Automotive HDR カメラ C1 (x2)
    - LiDAR: HESAI AT128 (x1)
  - 構成例 2
    - カメラ: TIER IV Automotive HDR カメラ C1 (x2)
    - LiDAR: HESAI Pandar XT32 (x1)

接続図

以下の図は、このチュートリアルのセンサーと ECU 間の接続を示しています。特定のインターフェイスへの IP アドレスの適用を含むこのネットワークの構成は、インストール ページの手順中に自動的に行われます。

インストールにはインターネット接続が必要です。

次の インストール ステップでは、git clone と ML モデルをダウンロードのため、インターネット接続が必要です。インターネット接続には、下図に示すポートにイーサネットケーブルを接続してください。

注釈: ディスプレイに何も表示されない場合は、別のディスプレイ ポートを試してください。

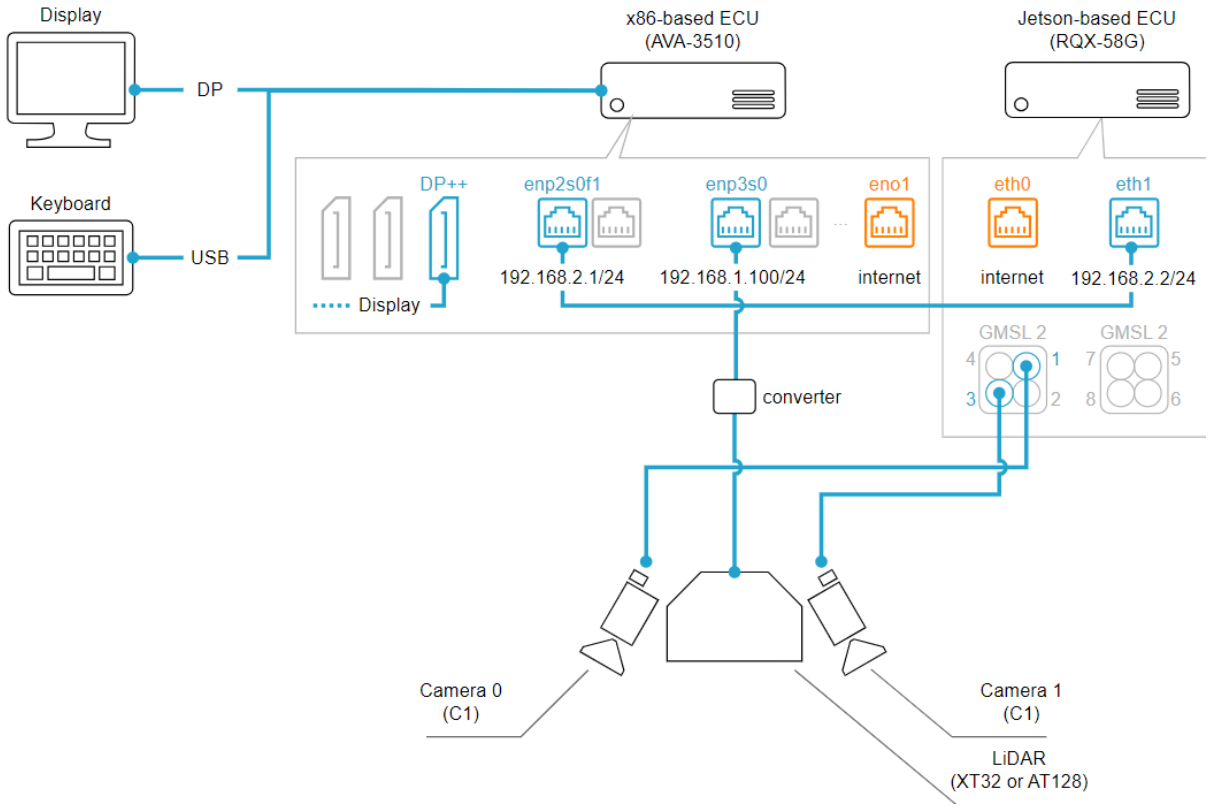

図 6.1　サンプルシステムの接続図

図 6.2　サンプルシステムのハードウェアセットアップ

センサードライバ

Edge.Auto は、さまざまなタイプのセンサーをサポートしています。これらのセンサーを ROS2 環境で利用できるようにするために、次のリポジトリを使用します。詳細については、各リポジトリを参照してください。

- カメラドライバ
  - tier4/tier4_automotive_hdr_camera: Video4Linux2 インターフェイスで TIER IV カメラを使用するためのカーネル ドライバ
  - tier4/ros2_v4l2_camera: Video4Linux2 を使用するカメラドライバー用の ROS2 パッケージ
- LIDAR ドライバ
  - tier4/nebula: イーサネットベースの統合 LiDAR ドライバの ROS2 パッケージ
- センサー同期
  - tier4/sensor_trigger: センサー トリガー信号を生成するための ROS2 パッケージ

センサー/ECU 同期

本サンプルシステムでは、センサーと ECU 間のクロック同期やタイミング同期を実現し、高精度なセンサーフュージョンを実現します。以下の図は、このサンプル システムにおけるセンサーと ECU 間の同期設計を示しています。

詳細については、tier4/sensor_trigger リポジトリを参照してください。

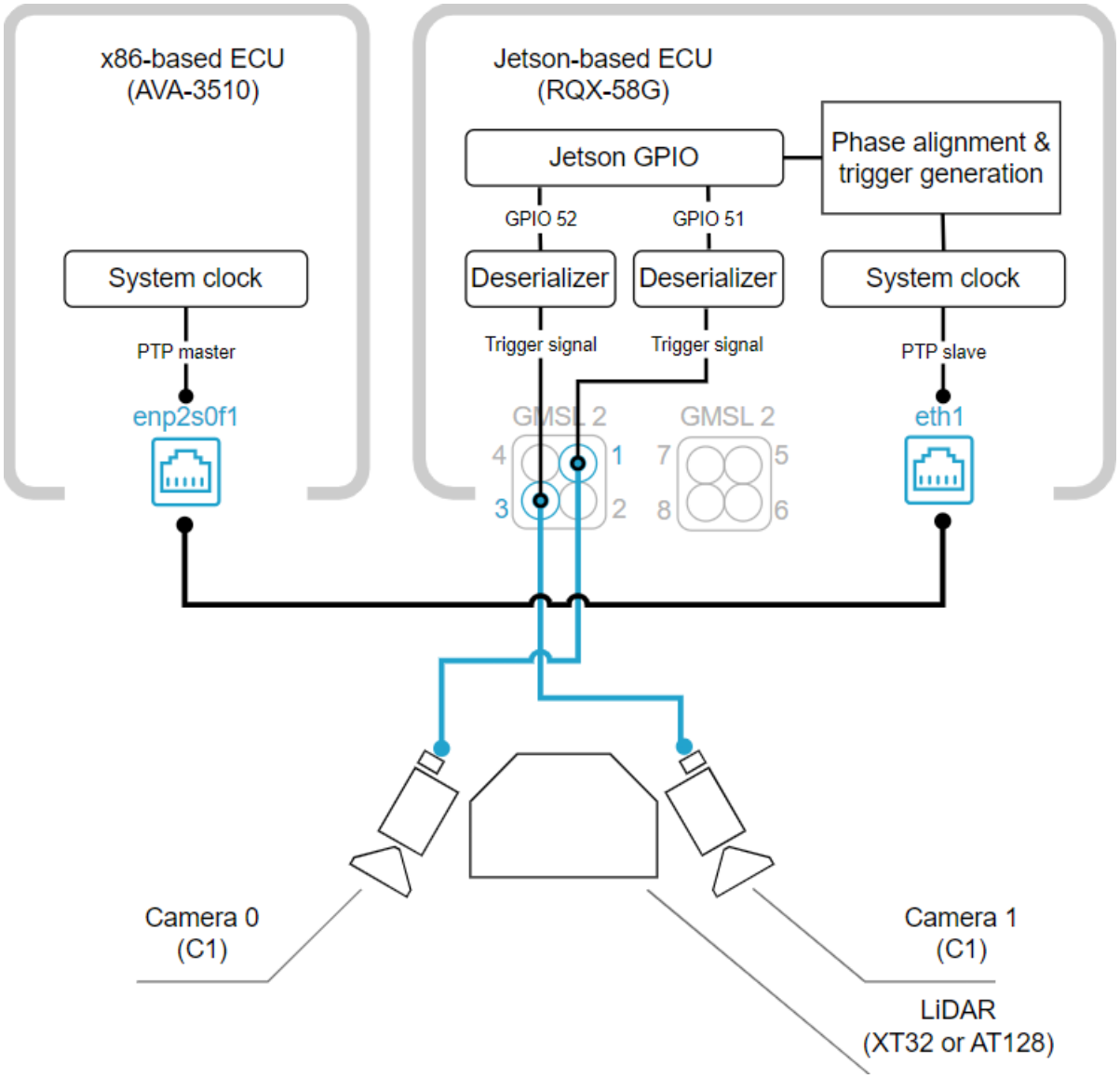

図 6.3　サンプルシステムの同期設計

6.1.2 x86 ベースの ECU

2. インストール の手順に進む前に、x86 ベースの ECU に Ubuntu 22.04 をインストールします。

6.1.3 Jetson ベースの ECU

2. インストール の手順に進む前に、NVIDIA L4T R32.6.1 (Ubuntu 18.04 を含む) を Jetson ベースの ECU にインストールします。

ADLINK RQX-58G 用 BSP のインストール

RQX-58G は、ADLINK Technology, Inc. の公式クイック スタート ガイドに従って適切に設定する必要があります。公式ドキュメント を参照してください。

BSP イメージをダウンロードするには、ADLINK 公式ページ にアクセスしてください。(初めてサイトにアクセスする場合は、アカウントの作成を求められます。)

TIER IV カメラ ドライバー (tier4/tier4_automotive_hdr_camera) は RQX-58G BSP 公式イメージに含まれていますが、次のセットアップ プロセス中に更新することもできます。

6.2 インストール

x86 ベースと Jetson ベースの両方の ECU のセットアップを行います。

注釈: このステップにはインターネット接続が必要です。

6.2.1 x86 ベース ECU

警告: このステップを行うと、ネットワーク設定が自動的に更新されます。

この手順では、`netplan` を使用して、いくつかのネットワーク インターフェイスに IP アドレスが割り当てられます (詳細については、ハードウェア セットアップ の接続図を参照してください)。これらのインターフェイスを介して ECU にリモートでアクセスしている場合、この動作により予期しない切断が発生する可能性があります。

割り当てるネットワークインターフェースや IP アドレスを変更したい場合は、`setup-dev-env.sh` を実行する前に `edge-auto/ansible/playbooks/vars/edge_auto.yaml` を編集してください。

リポジトリをダウンロードして環境をセットアップする

最初のステップとして、`tier4/edge-auto` のクローンを作成し、そのディレクトリに移動します。

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

提供された ansible スクリプトを使用して依存関係をインストールできます。

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

最後に、システムを再起動して、インストールされた依存関係と権限の設定を有効にしてください。

```
sudo reboot
```

edge-auto のビルド

vcstool を使用して ROS ワークスペースを作成し、リポジトリのクローンを作成します。

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

ros パッケージの依存関係をインストールし、ROS ワークスペースを構築します。

```
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 ベースの ECU

次の手順は、**x86** ベースの **ECU** から **ssh** 経由で実行できます

リポジトリをダウンロードして環境をセットアップする

最初のステップとして、tier4/edge-auto-jetson のクローンを作成し、そのディレクトリに移動します。

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

提供された ansible スクリプトを使用して依存関係をインストールできます。インストール プロセス中に、TIER IV カメラ ドライバーをインストールするかどうかを尋ねられます。ドライバーがすでにインストールされており、この手順をスキップしたい場合は、「N」を入力して続行してください。

> **注意:**「setup-dev-env.sh」スクリプトには数時間かかる場合があります。

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

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

最後に、システムを再起動して、インストールされた依存関係と権限の設定を有効にしてください。

```
sudo reboot
```

Edge-auto-jetson ワークスペースを構築する

vcstool を使用して ROS ワークスペースを作成し、リポジトリのクローンを作成します。

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

ROS ワークスペースを構築します。

```
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 ワークスペースを更新する

クローン作成されたリポジトリを更新する場合は、次のコマンドを使用します。

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

6.2.4 カメラの露出タイミングを変更する

注釈: ハードウェアセットアップの手順で導入したサンプルシステムでは、これを変更する必要はありません。

センサー同期のためにカメラの露光時間を変更したい場合は、以下のファイルを変更してください。

```
edge-auto-jetson/src/individual_params/individual_params/config/
└── default
    ├── camera0
    │   ├── trigger.param.yaml
    ├── camera1
    │   ├── trigger.param.yaml
```

詳細については、tier4/sensor_trigger リポジトリを参照してください。

6.3 センサーのキャリブレーション

tier4/calibration_tools を使用して、センサー システムの固有/外部パラメーターを推定します。

注釈: x86 ベースの ECU で次のタスクを実行します。

6.3.1 カメラの固有パラメータを計算する

まず、calibration_intrinsic を起動して、カメラの固有パラメータを推定します。ツールの詳細な操作については、このドキュメント を参照してください。

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

ros2 launch edge_auto_launch calibration_intrinsic_sample.launch.xml
```

システムを構成するすべてのカメラの組み込みパラメータ (たとえば、このチュートリアルの camera0 と camera1) は、individual_params に保存する必要があります。ファイルを取得したら、**Jetson** ベースの **ECU** 上の適切なフォルダーに配置し、camera_info トピックとして読み込まれて公開されるようにします。

```
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 のみ) LiDAR から補正ファイルを取得

HESAI AT128 には、内部に保存されている独自の補正ファイルにアクセスする機能があります。より良い結果を得るには、個々の LiDAR から補正ファイルをダウンロードし、x86 ベースの ECU 上の適切なフォルダーに保存することをお勧めします。

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

6.3.3 LiDAR とカメラの間の外部パラメータを計算する

最後に、LiDAR 設定に一致する `calibration_extrinsic` を起動して、LIDAR とカメラの間の外部パラメータを推定します。ツールの詳細な操作については、このドキュメント を参照してください。

```
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
```

センサー フュージョンを実行するには、すべてのフューズされたセンサー間の姿勢関係 (すなわち外部パラメーター) を「TF」に登録する必要があります。これは、関係を表現するための ROS 形式です。

注釈: これらのサンプルでは、すべてのセンサーが固定されており、それらの相対位置が変化しないと仮定しています。

外部パラメータを計算した後、結果を x86 ベースの ECU 上の適切なファイルに保存します。

```
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 アプリケーションの起動

autoware.universe に実装されている認識アプリケーションを起動します。

6.4.1 Jetson ベースの ECU

注釈: 次の手順は、x86 ベースの ECU から ssh 経由で実行できます。

次のサンプルでは、2 台のカメラで個別に実行される画像ベースのオブジェクト検出を起動します。

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

ros2 launch edge_auto_jetson_launch edge_auto_jetson.launch.xml
```

注釈: 最初の実行時に結果が利用可能になるまでに約 15 分かかる場合があります。これは、ONNX モデルを TensorRT エンジンに変換することによって発生します。変換結果はディスクにキャッシュされるため、2 回

目の起動以降は結果がすぐに利用可能になります。

このサンプルは、autoware.universe に実装されている tensorrt_yolox および bytetrack を利用します。詳細については、これらのパッケージの README を参照してください。

6.4.2 x86 ベースの ECU

次のサンプルは、LiDAR ベースのオブジェクト検出と、2D オブジェクト検出と 3D オブジェクト検出の間のバウンディングボックス レベルのフュージョン (すなわち レイトフュージョン) を起動します。LiDAR 設定に一致する起動ファイルを起動します。

```
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
```

このサンプルは主に autoware.universe の pointcloud_preprocessor、centerpoint および image_projection_based_fusion を利用します。詳細については、これらのパッケージの README を参照してください。

このサンプルでは、パーセプションスタックに加えて、ユーザーが認識結果を視覚的に確認できるようにビューアも起動します。

6.5 トラブルシューティング