PX4 SITL + Gazebo Sim → ROS 2 Humble Camera Bridge (Single Camera)

Ubuntu 22.04 workflow using the provided zip repo (px4-gz-ros2-camera-bridge_repo.zip). Generated: 2025-12-14

What this gives you

A ready-to-run launch file that bridges a single Gazebo Sim camera (gz.msgs.Image + CameraInfo) into ROS 2 topics (**/camera/image_raw** and **/camera/camera_info**). This is intended for the PX4 SITL model **gz_x500_mono_cam**.

- No colcon build required if you run the launch file by path (ros2 launch ./launch/...).
- Works with ROS 2 Humble and ros_gz packages (ros_gz_bridge, ros_gz_image).
- Keeps the pipeline single-camera only (no stereo pipeline, no camera merging).

Prerequisites

- Ubuntu 22.04 + ROS 2 Humble installed (source /opt/ros/humble/setup.bash works).
- PX4-Autopilot already built and able to run: **make px4_sitl gz_x500_mono_cam**.
- ros_gz packages available: **ros_gz_bridge** and **ros_gz_image** (you already have them).

Step 0 — Unzip the repo

Place **px4-gz-ros2-camera-bridge_repo.zip** in **~/Downloads**, then:

```
cd ~/Downloads
unzip -o px4-gz-ros2-camera-bridge_repo.zip
cd px4-gz-ros2-camera-bridge
```

Step 1 — Terminal 1 (start PX4 SITL + Gazebo Sim)

```
cd ~/0Test/PX4-Autopilot
make px4_sitl gz_x500_mono_cam
```

Leave this terminal running. Gazebo Sim should open and you should see the **x500_mono_cam** vehicle in the world.

Step 2 — Terminal 2 (bridge camera topics to ROS 2)

Recommended: use the provided launch file (uses ros_gz_image image_bridge for images).

```
source /opt/ros/humble/setup.bash
cd ~/Downloads/px4-gz-ros2-camera-bridge

# Optional: if your PX4/Gazebo terminal prints or exports a partition, match it here:
# export GZ_PARTITION=<same_value_as_terminal_1>

ros2 launch ./launch/bridge_single_camera.launch.py \
  gz_image_topic:=/world/default/model/x500_mono_cam_0/link/camera_link/sensor/camera/image \
  gz_info_topic:=/world/default/model/x500_mono_cam_0/link/camera_link/sensor/camera/camera_info
```

```
  ros_image_topic:=/camera/image_raw \
  ros_info_topic:=/camera/camera_info \
  mode:=image_bridge
```

Optional quick checks (in another shell):

```
source /opt/ros/humble/setup.bash
ros2 topic list | egrep "camera|image"
ros2 topic hz /camera/image_raw
ros2 topic echo /camera/camera_info --once
```

Minimal alternative (no launch file)

If you prefer the simplest explicit commands, bridge CameraInfo with parameter_bridge and Image with image_bridge.

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

# CameraInfo (gz -> ROS2)
ros2 run ros_gz_bridge parameter_bridge \
  "/world/default/model/x500_mono_cam_0/link/camera_link/sensor/camera/camera_info@sensor_msgs/ms

# Image (gz -> ROS2)
ros2 run ros_gz_image image_bridge \
  /world/default/model/x500_mono_cam_0/link/camera_link/sensor/camera/image \
  --ros-args -r /world/default/model/x500_mono_cam_0/link/camera_link/sensor/camera/image:=/camer
```

Step 3 — Terminal 3 (your Python vision script)

Run your subscriber / viewer (example based on your setup). It should subscribe to **/camera/image_raw**.

```
cd ~/0Test/Drone/Cpp/t02
source .venv/bin/activate
python vision.py
```

Step 4 — Terminal 4 (your MAVSDK / control app)

```
cd ~/0Test/Drone/Cpp/t02
./run.sh
```

At this point you should have: Gazebo Sim running, ROS 2 publishing /camera/image_raw, a live viewer from vision.py, and your control program running.

How to adapt this for gz_x500 (no mono_cam)

The default **gz_x500** model may not include a camera sensor. Before trying to bridge, confirm whether camera topics exist.

1) Start PX4 SITL:

```
cd ~/0Test/PX4-Autopilot
make px4_sitl gz_x500
```

2) In a separate terminal, list Gazebo topics and look for camera/image topics:

```
# Either use gz directly:
gz topic -l | egrep -i "camera|image|camera_info"

# Or use the helper script from this repo:
cd ~/Downloads/px4-gz-ros2-camera-bridge
./scripts/find_gz_camera_topics.sh
```

If you do **not** see any camera/image topics, the model has no camera—use **gz_x500_mono_cam** or add a camera sensor to the model. If you **do** see topics, copy the exact **image** and **camera_info** topic paths and substitute them into the launch command in Step 2.

Troubleshooting (keep it simple)

- If **“Unknown message type [9]”** almost always means the *parameter_bridge* mapping string is malformed. Put the mapping in quotes exactly like the examples, and do not add a closing bracket.
- If **/camera/image_raw** exists but your viewer shows nothing: confirm your viewer subscribes to **/camera/image_raw**, and check a rate with **ros2 topic hz /camera/image_raw**.
- If **no topics show up** in ROS 2: ensure Terminal 1 is still running, and (if used) set **GZ_PARTITION** in every terminal consistently.

Notes

A “ROS 2 launch file” (.launch.py) is simply a Python script that starts one or more ROS 2 nodes with arguments. Here it starts the bridge node(s) with the correct topic mappings so you do not have to type long commands every time.