APPLICATION NOTE

```
// Create an instant camera object with the firs
Camera_t camera( CTlFactory::GetInstance().Creat

// Register an image event handler that accesses
camera.RegisterImageEventHandler( new CSampleIma
Ownership_TakeOwnership);

// Open the camera.
camera.Open();
```

Interfacing Basler Cameras with ROS

Applicable to cameras only that allow images to be displayed by the Basler pylon Viewer

Document Number: AW001491
Version: 02   Language: 000 (English)
Release Date: 02 December 2019

BASLER
the power of sight

Contacting Basler Support Worldwide

Europe, Middle East, Africa

Basler AG
An der Strusbek 60–62
22926 Ahrensburg
Germany

Tel. +49 4102 463 515
Fax +49 4102 463 599

support.europe@baslerweb.com

The Americas

Basler, Inc.
855 Springdale Drive, Suite 203
Exton, PA 19341
USA

Tel. +1 610 280 0171
Fax +1 610 280 7608

support.usa@baslerweb.com

Asia-Pacific

Basler Asia Pte. Ltd.
35 Marsiling Industrial Estate Road 3
#05–06
Singapore 739257

Tel. +65 6367 1355
Fax +65 6367 1255

support.asia@baslerweb.com

www.baslerweb.com

Table of Contents

1 Introduction

Sensors and cameras are commonly used in robotics. The sensors are single-information and array detectors while cameras provide visual control. To interface cameras for robotics the Robot Operating System (ROS) user community continues to create camera driver wrappers and processing nodes.

ROS is an all open source framework of software libraries and tools. The framework supports the building of various robot applications. ROS provides the developing tools, algorithms and drivers for a variety of robotics platform projects.

ROS can run a large number of executables (nodes) in parallel and allows them to exchange data synchronously (service) or asynchronously (subscribed/published topics). In practice, the data are generally sensor queries whose result data are processed to cause robot actions.

This document illustrates how to interface Basler GigE Vision and USB3 Vision cameras with ROS using the pylon-ROS-camera driver package (expressed in code as **pylon_ros_camera**).

The procedures described in this document were evaluated with Basler pylon v. 5.2 installed and with the following Linux distribution and ROS software:

- Ubuntu 18.04.3 LTS (Bionic Beaver)
- ROS (Melodic Morenia)

Check pylon version compatibilities when creating or using further ROS nodes.

The document shows commands given in `orange` after the $ prompt. You can use them via copy-and-paste.

Legal Notice

Basler does not assume any liability for the functionality and suitability of any recommended open source products referenced in this application note. This is just a presentation of a sample use case. The readers of this application note are fully responsible to conduct their own testing procedures to assess the suitability of the mentioned open source products for their own applications.

2 Installing Software

In this section, the installation of the following software is described:

- Operating system
- Basler pylon Camera Software Suite for Linux x86
- ROS Robot Operating System
- pylon-ROS-camera driver package

2.1 Operating System Compatibilities

This document focuses on the ROS use with natively installed Linux x86 operating systems and assumes that you use or create a new operating system installation using a Linux ISO image.
In the present case an Ubuntu 18.04.3 Long Term Support (LTS) x64 installation has been used. Make sure you have an internet connection on your Linux machine available. In case of any difficulties check if any proxy server settings are necessary or must be adjusted. If the installations take place behind a proxy server, at least proper HTTPS and FTP settings including port access are mandatory.

Basler advises strongly against trying to use a Windows operating system with the pylon-ROS-camera driver package. Such constellations were never tried let alone tested.

2.2 Basler pylon Camera Software Suite for Linux x86 Installation

The pylon-ROS-camera driver package requires that the library of pylon version 5.2 or newer is installed. The following situations can apply:

- pylon is already installed and path variable PYLON_ROOT is set properly
- pylon is not yet installed but will now be manually installed and enabled to be applicable for ROS nodes
- pylon is not yet installed but will be automatically together with the Debian package during the ROS dependency install step

If you need to install a suitable pylon version, continue with this section. Otherwise, continue with the ROS Robot Operating System Installation section further below.

1. Got to http://www.baslerweb.com/ where two pylon Camera Software Suite for Linux x86 installer packages are available.
2. Download one of both packages, depending on applicability:
   - **tar.gz** (applicable to all Linux distributions)
   - **.deb** (applicable to Ubuntu and related Linux distributions)

3. Install the downloaded installer package:

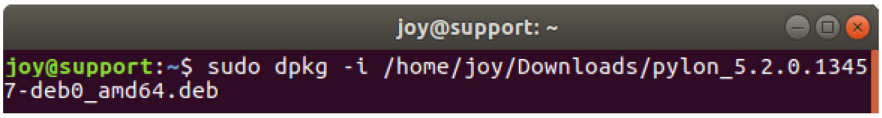
If you downloaded **tar.gz**

a) Install the pylon SDK from the **tar.gz** installer package. Details about installation and configuration are available from the included **INSTALL** and **README** files.

> NOTICE
>
> Make sure to carry out the necessary adjustments as described in the **INSTALL** file:
>
> 1. Run the **pylon-setup-env.sh** script to set the PYLON_ROOT environment variable.
> 2. If you want to use Basler USB3 Vision cameras, run the included **setup-usb.sh** script.

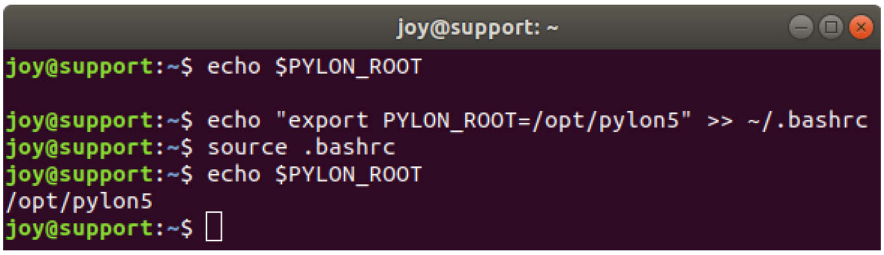
If you downloaded **.deb**

a. Install the Basler pylon Camera Software Suite for Linux on Debian and related Linux distributions (e.g. Ubuntu) from the **.deb** installer package that suits your platform. You must install from the **.deb** installer package using the dpkg command line tool:



```
joy@support:~$ sudo dpkg -i /home/joy/Downloads/pylon_5.2.0.1345
7-deb0_amd64.deb
```

b. Set the pylon root location environment variable and optionally make sure that it is persistent by adding variable creation to the **~/.bashrc** file.



```
joy@support:~$ echo $PYLON_ROOT

joy@support:~$ echo "export PYLON_ROOT=/opt/pylon5" >> ~/.bashrc
joy@support:~$ source .bashrc
joy@support:~$ echo $PYLON_ROOT
/opt/pylon5
joy@support:~$ 
```

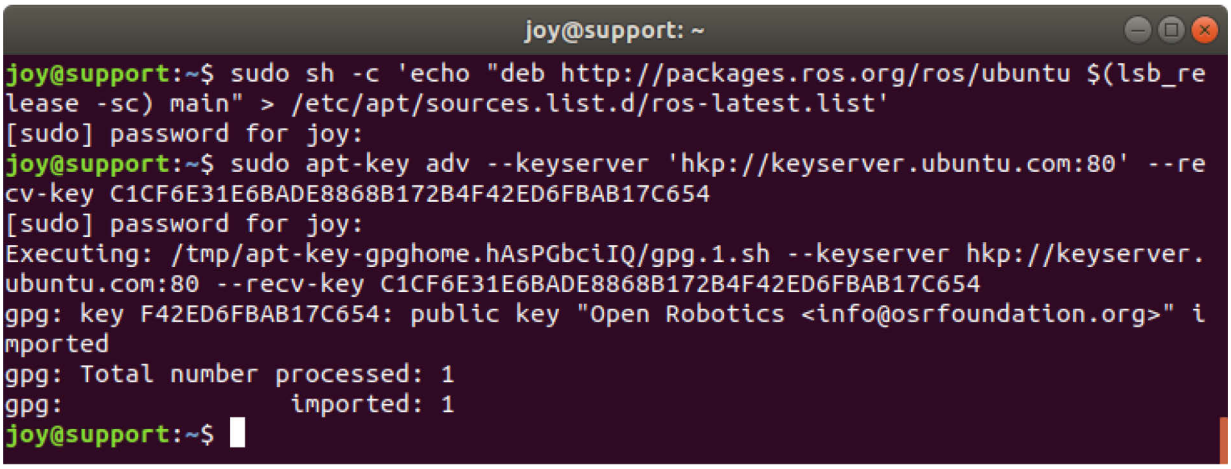
```
$ echo “export PYLON_ROOT=/opt/pylon5” >> ~/.bashrc
```

The PYLON_ROOT environment variable is necessary for pylon path identification related to development and pylon-ROS-camera driver package use. See below for more information about pylon-ROS-camera, designed for use with cameras supported by pylon.

2.3 ROS Robot Operating System Installation

The following installation steps are listed without detailed comment. For additional information, see the ROS wiki.

Preparatory Steps



```
joy@support:~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_re
lease -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
[sudo] password for joy:
joy@support:~$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --re
cv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
[sudo] password for joy:
Executing: /tmp/apt-key-gpghome.hAsPGbciIQ/gpg.1.sh --keyserver hkp://keyserver.
ubuntu.com:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: key F42ED6FBAB17C654: public key "Open Robotics <info@osrfoundation.org>" i
mported
gpg: Total number processed: 1
gpg:               imported: 1
joy@support:~$
```

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

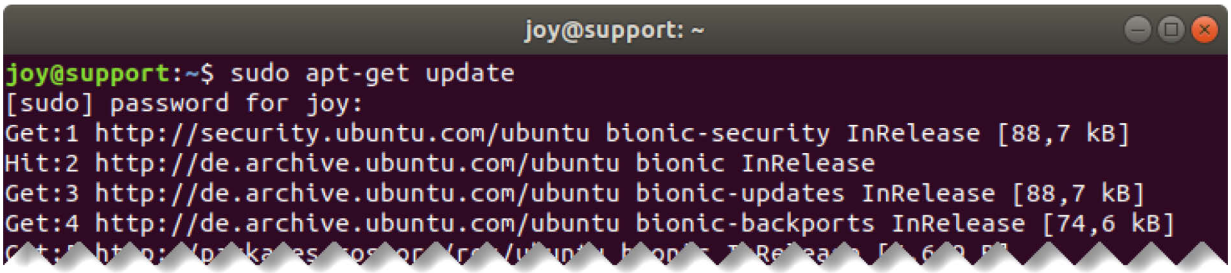
The ROS wiki installation site provides alternative ways for accessing the keyserver.

Installation of ROS

Below, the installation of ROS Melodic Morenia is described. It is the LTS version until the year 2023. For more details and possible alternative installation steps visit the Ubuntu ROS Melodic Morenia Installation site.

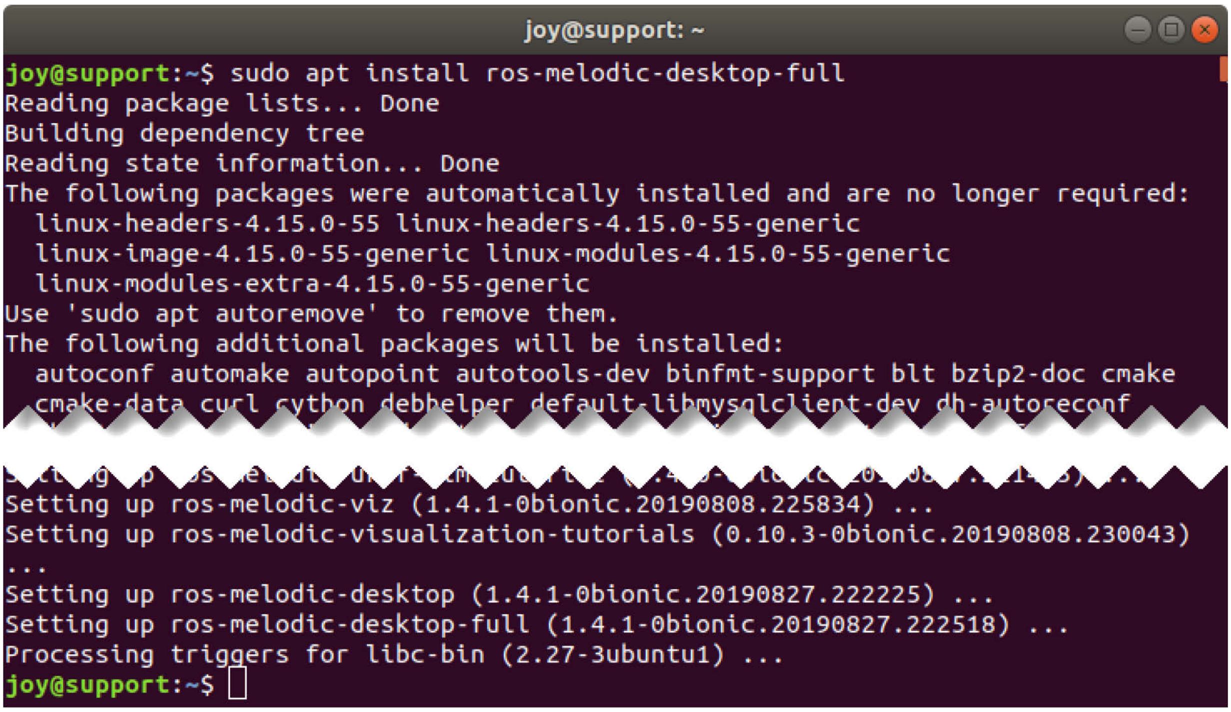
This application note may also apply to other ROS releases, with installations analogous to the installation of ROS Melodic Morenia. This, however, was not tested.

Install ROS Melodic Morenia:



```
joy@support:~$ sudo apt-get update
[sudo] password for joy:
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88,7 kB]
Hit:2 http://de.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://de.archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]
Get:4 http://de.archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]
```

```
$ sudo apt-get update
```

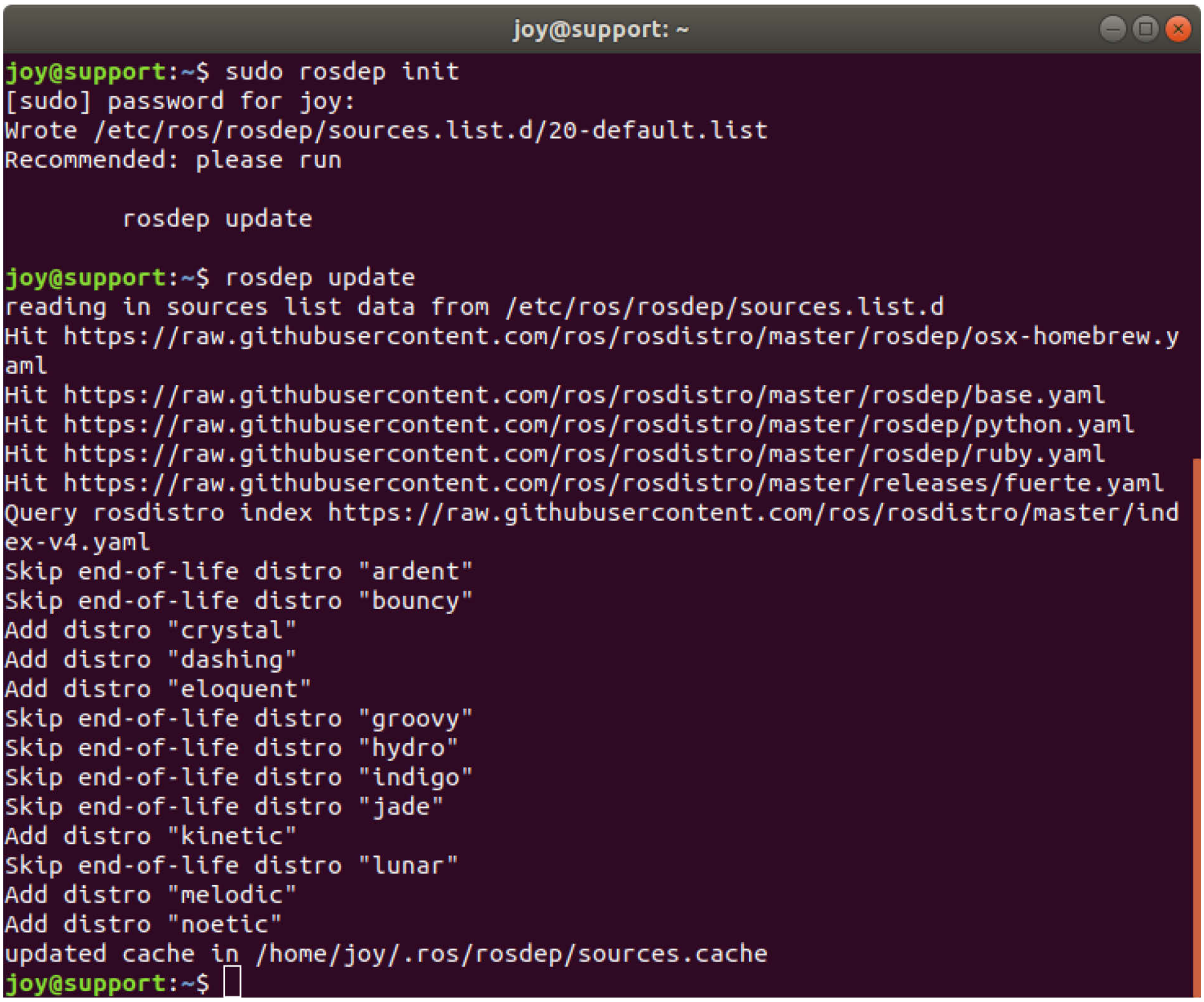


```
joy@support:~$ sudo apt install ros-melodic-desktop-full
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-4.15.0-55 linux-headers-4.15.0-55-generic
  linux-image-4.15.0-55-generic linux-modules-4.15.0-55-generic
  linux-modules-extra-4.15.0-55-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  autoconf automake autopoint autotools-dev binfmt-support blt bzip2-doc cmake
  cmake-data curl cython debhelper default-libmysqlclient-dev dh-autoreconf

Setting up ros-melodic-viz (1.4.1-0bionic.20190808.225834) ...
Setting up ros-melodic-visualization-tutorials (0.10.3-0bionic.20190808.230043)
...
Setting up ros-melodic-desktop (1.4.1-0bionic.20190827.222225) ...
Setting up ros-melodic-desktop-full (1.4.1-0bionic.20190827.222518) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
joy@support:~$
```

```
$ sudo apt-get install ros-melodic-desktop-full
```

Initialization of rosdep

Do not run a **rosdep update** with sudo. This would later result in permission errors.

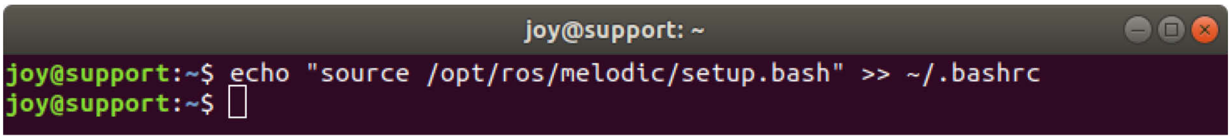


```
joy@support:~$ sudo rosdep init
[sudo] password for joy:
Wrote /etc/ros/rosdep/sources.list.d/20-default.list
Recommended: please run

        rosdep update

joy@support:~$ rosdep update
reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.y
aml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
Query rosdistro index https://raw.githubusercontent.com/ros/rosdistro/master/ind
ex-v4.yaml
Skip end-of-life distro "ardent"
Skip end-of-life distro "bouncy"
Add distro "crystal"
Add distro "dashing"
Add distro "eloquent"
Skip end-of-life distro "groovy"
Skip end-of-life distro "hydro"
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Add distro "kinetic"
Skip end-of-life distro "lunar"
Add distro "melodic"
Add distro "noetic"
updated cache in /home/joy/.ros/rosdep/sources.cache
joy@support:~$
```

$ `sudo rosdep init`

$ `rosdep update`

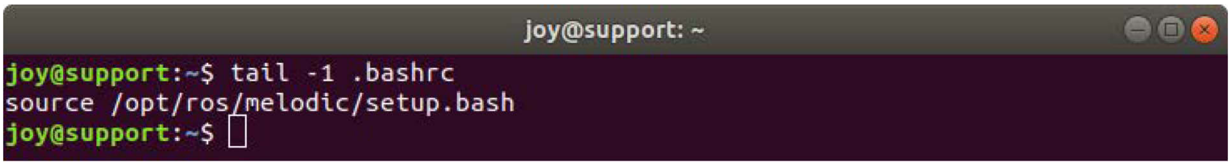
Establishing Environment Settings



```
joy@support:~$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
joy@support:~$
```

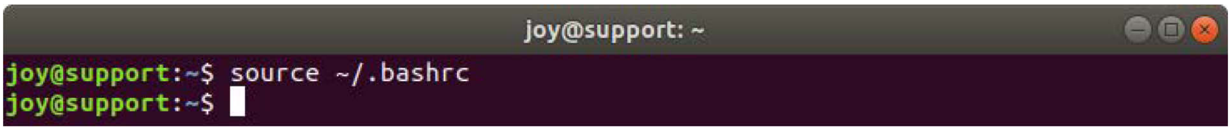
$ `echo “source /opt/ros/melodic/setup.bash” >> ~/.bashrc`

You can check the correct settings:



```
joy@support:~$ tail -1 .bashrc
source /opt/ros/melodic/setup.bash
joy@support:~$
```

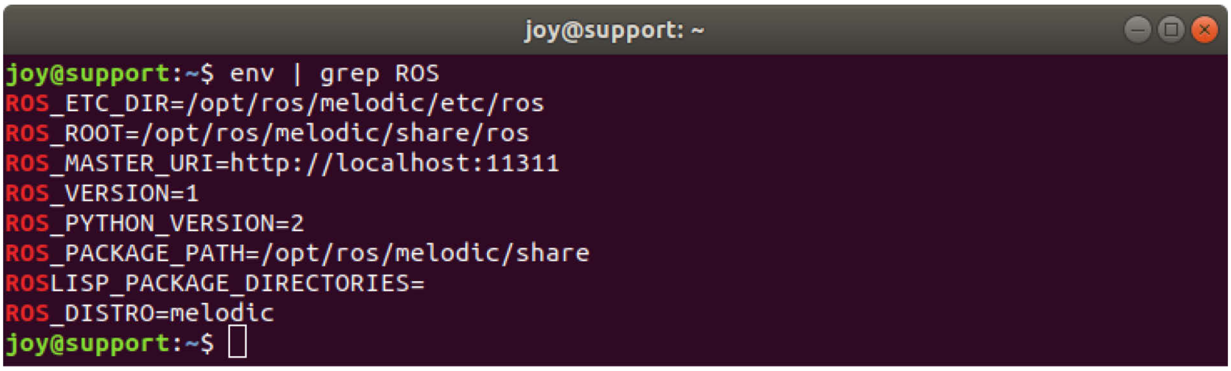
Source the **.bashrc** file to apply the modification:



```
joy@support:~$ source ~/.bashrc
joy@support:~$ 
```

$ source ~/.bashrc

You can check whether the ROS environment variables were successfully set.

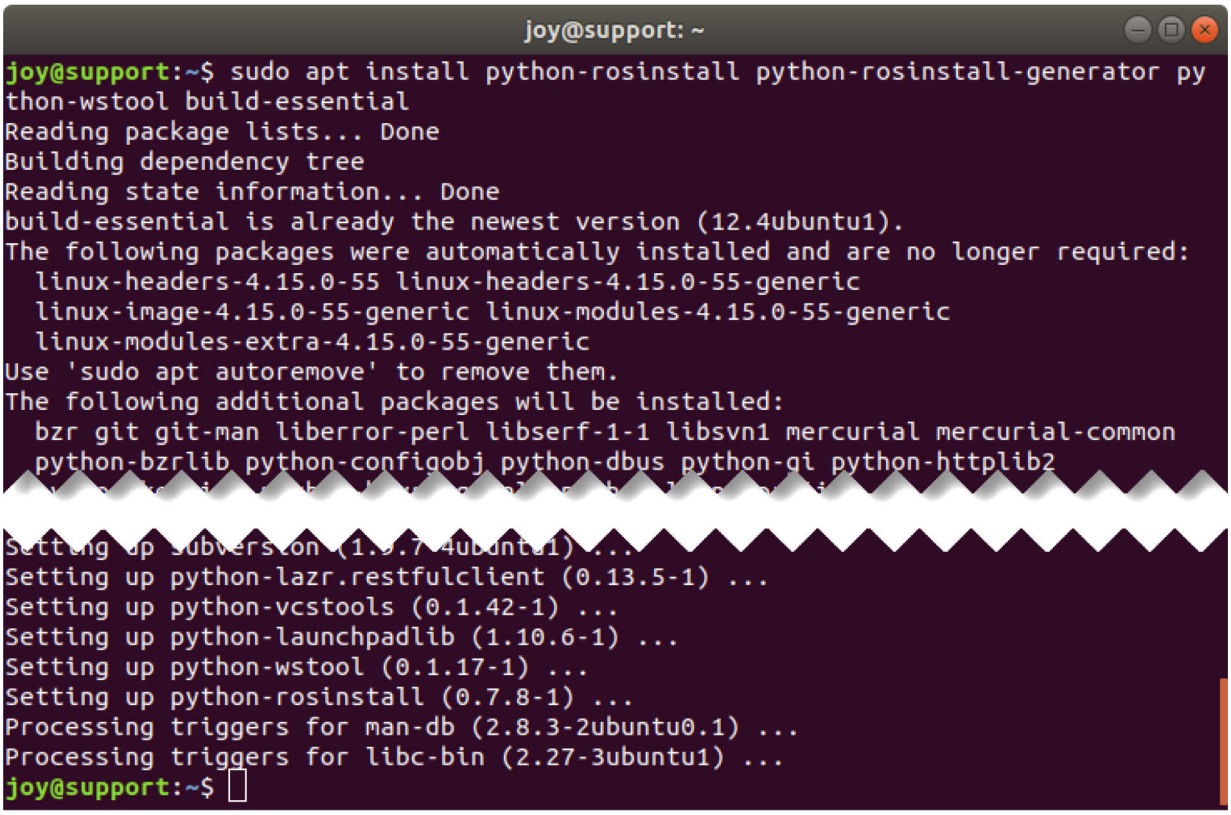


```
joy@support:~$ env | grep ROS
ROS_ETC_DIR=/opt/ros/melodic/etc/ros
ROS_ROOT=/opt/ros/melodic/share/ros
ROS_MASTER_URI=http://localhost:11311
ROS_VERSION=1
ROS_PYTHON_VERSION=2
ROS_PACKAGE_PATH=/opt/ros/melodic/share
ROSLISP_PACKAGE_DIRECTORIES=
ROS_DISTRO=melodic
joy@support:~$ 
```

Installation of Tools

After ROS installation, it is useful to add a couple of optional tools to create and manage your own ROS workspaces. Those bootstrap dependencies are not automatically supplied with ROS.

Meet the requirements and install useful tools for ROS package management.



```
joy@support:~$ sudo apt install python-rosinstall python-rosinstall-generator py
thon-wstool build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.4ubuntu1).
The following packages were automatically installed and are no longer required:
  linux-headers-4.15.0-55 linux-headers-4.15.0-55-generic
  linux-image-4.15.0-55-generic linux-modules-4.15.0-55-generic
  linux-modules-extra-4.15.0-55-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  bzr git git-man liberror-perl libserf-1-1 libsvn1 mercurial mercurial-common
  python-bzrlib python-configobj python-dbus python-gi python-httplib2
```

```
Setting up subversion (1.9.7-4ubuntu1) ...
Setting up python-lazr.restfulclient (0.13.5-1) ...
Setting up python-vcstools (0.1.42-1) ...
Setting up python-launchpadlib (1.10.6-1) ...
Setting up python-wstool (0.1.17-1) ...
Setting up python-rosinstall (0.7.8-1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
joy@support:~$ 
```

```
$ sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

2.4 Middleware Installation

The descriptions given so far do not consider the intermediary (“driver”) between the powerful pylon and ROS software structures. Such driver is usually created by the ROS-oriented developers community.

The installation of a driver is illustrated here using the pylon-ROS-camera driver package as the driver. The installation assumes that operating system and ROS Robot Operating System are already installed, as described above.

2.4.1 Details About the pylon-ROS-camera Driver Package

The pylon-ROS-camera driver package is the official pylon ROS driver for all recent Basler GigE Vision and USB3 Vision cameras. You can download the driver package using this URL: https://github.com/basler/pylon-ros-camera

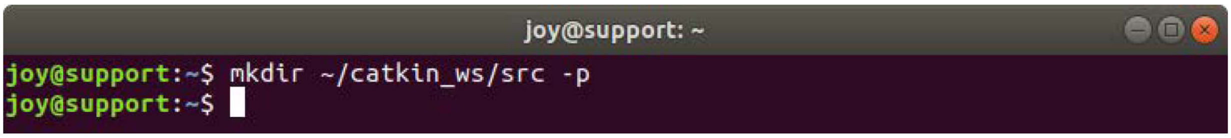
The driver package provides a range of the pylon API features that allow interactive camera operation. Images are published into ROS. The package is designed to meet certain application tasks and is therefore not a complete wrapper for all pylon API methods. However, adhering to the open source concept, pylon-ROS-camera can be studied, copied or modified, observing the related Copyright and the BSD license model.

For further information about pylon-ROS-camera, go to its GitHub: https://github.com/basler/pylon-ros-camera

2.4.2 Preparation of a ROS Build Workspace

When ROS is installed, catkin is included. It is a workspace build system and provides low level build system macros and infrastructure. The catkin system is necessary to build code projects like pylon-ROS-camera, for example.

A workspace must be set up where single or multiple packages can be built. In the following, the folder **catkin_ws** and its subfolder **src** are created, unless they are present already.



```
joy@support:~$ mkdir ~/catkin_ws/src -p
joy@support:~$ 
```

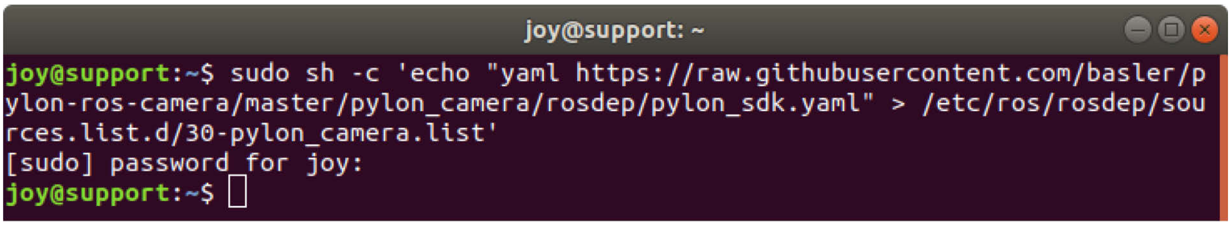
```
$ mkdir ~/catkin_ws/src
```

Later on in the process, the ROS packages are cloned into the **src** folder for building.

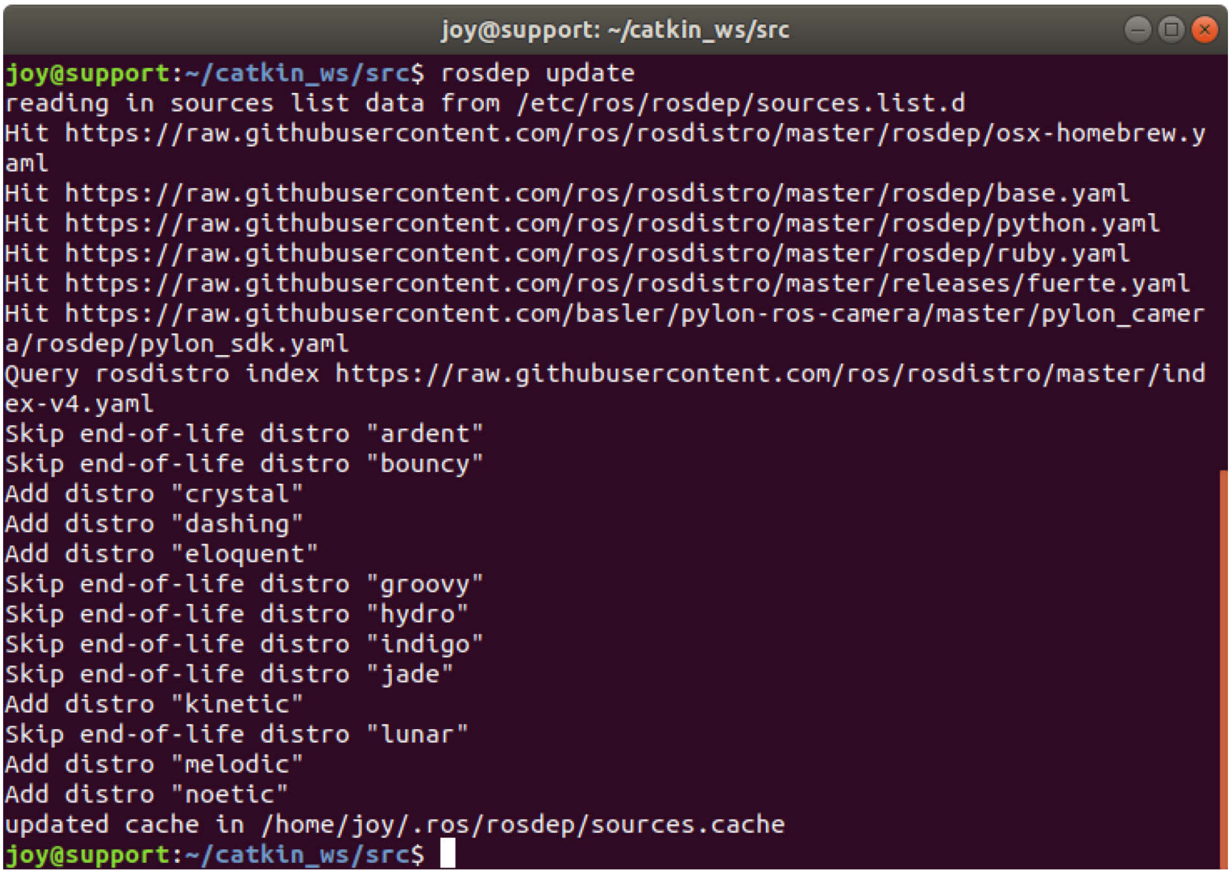
2.4.3 The Driver Employment

If the pylon Camera Software Suite for Linux version is already installed as described above, make sure the PYLON_ROOT environment variable is properly set. If pylon is not installed yet you need not worry because the installation will automatically be performed from an external repository and ROS dependencies are updated.

Now, it is just necessary to configure rosdep, the ROS command-line tool for adding system dependencies. This creates a **30-pylon_ros_camera.list** file. The file is scanned with all current files in the same folder during the following **rosdep update**.



```
joy@support:~$ sudo sh -c 'echo "yaml https://raw.githubusercontent.com/basler/p
ylon-ros-camera/master/pylon_camera/rosdep/pylon_sdk.yaml" > /etc/ros/rosdep/sou
rces.list.d/30-pylon_camera.list'
[sudo] password for joy:
joy@support:~$ 
```



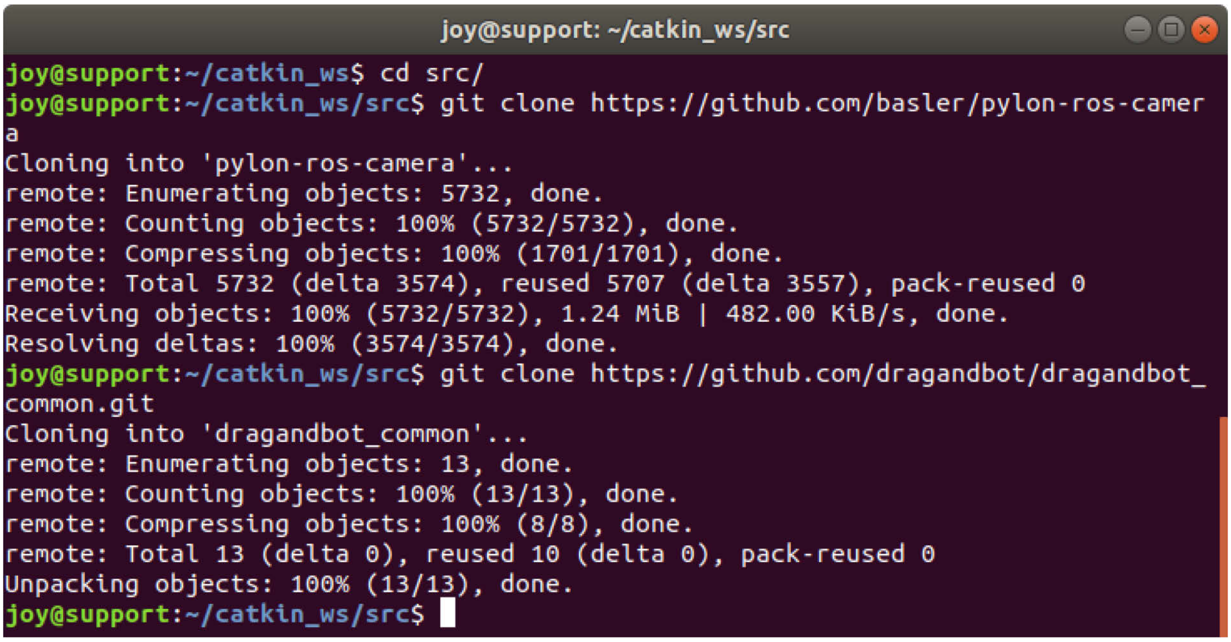
```
joy@support:~/catkin_ws/src$ rosdep update
reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.y
aml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
Hit https://raw.githubusercontent.com/basler/pylon-ros-camera/master/pylon_camer
a/rosdep/pylon_sdk.yaml
Query rosdistro index https://raw.githubusercontent.com/ros/rosdistro/master/ind
ex-v4.yaml
Skip end-of-life distro "ardent"
Skip end-of-life distro "bouncy"
Add distro "crystal"
Add distro "dashing"
Add distro "eloquent"
Skip end-of-life distro "groovy"
Skip end-of-life distro "hydro"
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Add distro "kinetic"
Skip end-of-life distro "lunar"
Add distro "melodic"
Add distro "noetic"
updated cache in /home/joy/.ros/rosdep/sources.cache
joy@support:~/catkin_ws/src$ 
```

```
$ sudo sh -c 'echo "yaml
https://raw.githubusercontent.com/basler/pylon_ros_camera/master/pylon_came
ra/rosdep/pylon_sdk.yaml " > /etc/ros/rosdep/sources.list.d/30-
plyon_camera.list'
```

```
$ rosdep update
```

Clone the necessary driver packages from GitHub to the catkin build system workspace **src** folder.

Go to the workspace folder **src.**

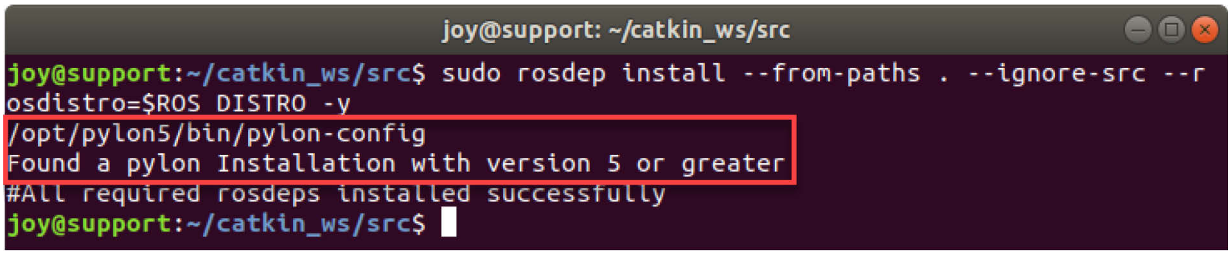


```
joy@support:~/catkin_ws$ cd src/
joy@support:~/catkin_ws/src$ git clone https://github.com/basler/pylon-ros-camer
a
Cloning into 'pylon-ros-camera'...
remote: Enumerating objects: 5732, done.
remote: Counting objects: 100% (5732/5732), done.
remote: Compressing objects: 100% (1701/1701), done.
remote: Total 5732 (delta 3574), reused 5707 (delta 3557), pack-reused 0
Receiving objects: 100% (5732/5732), 1.24 MiB | 482.00 KiB/s, done.
Resolving deltas: 100% (3574/3574), done.
joy@support:~/catkin_ws/src$ git clone https://github.com/dragandbot/dragandbot_
common.git
Cloning into 'dragandbot_common'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 13 (delta 0), reused 10 (delta 0), pack-reused 0
Unpacking objects: 100% (13/13), done.
joy@support:~/catkin_ws/src$
```

```
$ cd ~/catkin_ws/src/ && git clone https://github.com/basler/pylon-ros-camera

$ git clone https://github.com/dragandbot/dragandbot_common.git
```

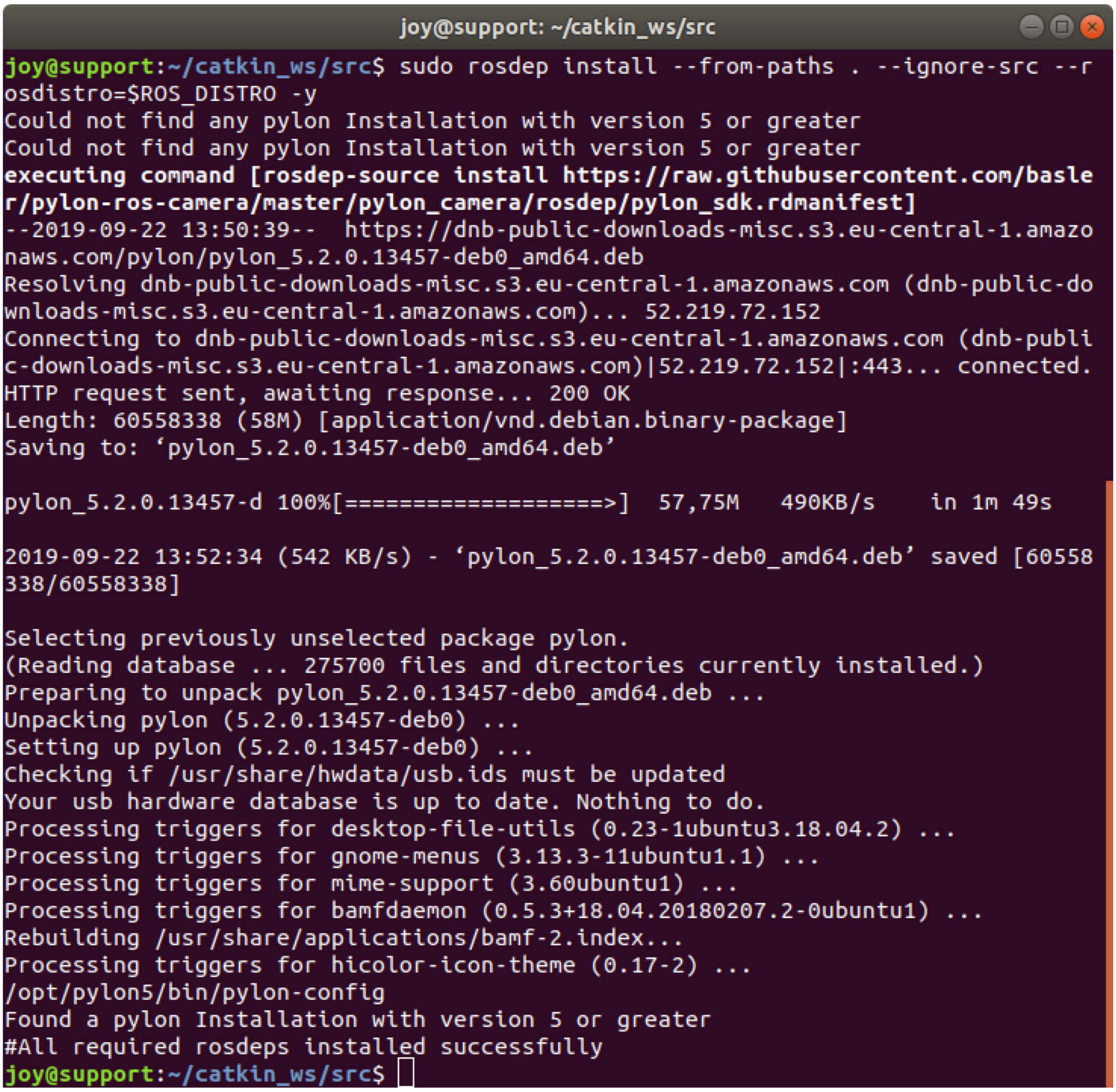
Install mandatory dependencies.

If the pylon SDK API is installed already, it will be recognized and the installation is skipped.



```
joy@support:~/catkin_ws/src$ sudo rosdep install --from-paths . --ignore-src --r
osdistro=$ROS_DISTRO -y
/opt/pylon5/bin/pylon-config
Found a pylon Installation with version 5 or greater
#All required rosdeps installed successfully
joy@support:~/catkin_ws/src$
```

Otherwise, the pylon API SDK is automatically installed through rosdep installation from external repository. In this case, the output of dependencies installation looks as follows:



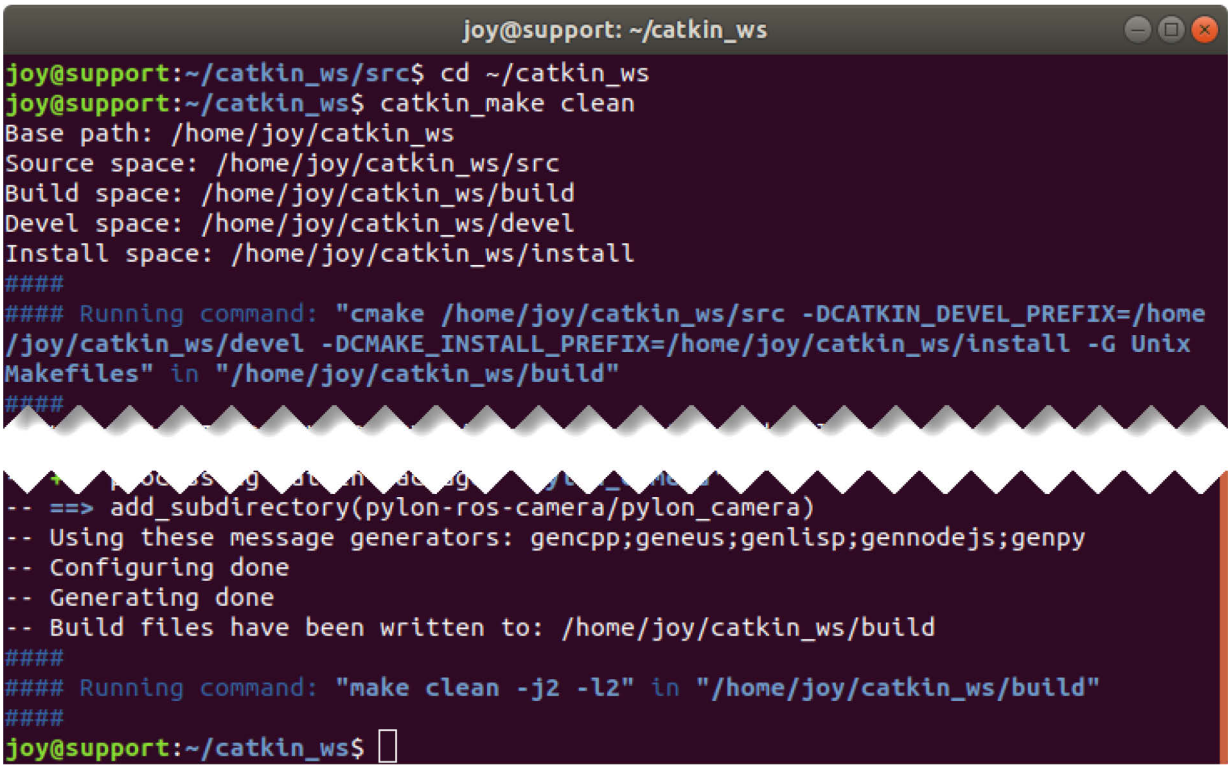
```
joy@support:~/catkin_ws/src$ sudo rosdep install --from-paths . --ignore-src --r
osdistro=$ROS_DISTRO -y
Could not find any pylon Installation with version 5 or greater
Could not find any pylon Installation with version 5 or greater
executing command [rosdep-source install https://raw.githubusercontent.com/basle
r/pylon-ros-camera/master/pylon_camera/rosdep/pylon_sdk.rdmanifest]
--2019-09-22 13:50:39--  https://dnb-public-downloads-misc.s3.eu-central-1.amazo
naws.com/pylon/pylon_5.2.0.13457-deb0_amd64.deb
Resolving dnb-public-downloads-misc.s3.eu-central-1.amazonaws.com (dnb-public-do
wnloads-misc.s3.eu-central-1.amazonaws.com)... 52.219.72.152
Connecting to dnb-public-downloads-misc.s3.eu-central-1.amazonaws.com (dnb-publi
c-downloads-misc.s3.eu-central-1.amazonaws.com)|52.219.72.152|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 60558338 (58M) [application/vnd.debian.binary-package]
Saving to: ‘pylon_5.2.0.13457-deb0_amd64.deb’

pylon_5.2.0.13457-d 100%[===================>]  57,75M   490KB/s    in 1m 49s

2019-09-22 13:52:34 (542 KB/s) - ‘pylon_5.2.0.13457-deb0_amd64.deb’ saved [60558
338/60558338]

Selecting previously unselected package pylon.
(Reading database ... 275700 files and directories currently installed.)
Preparing to unpack pylon_5.2.0.13457-deb0_amd64.deb ...
Unpacking pylon (5.2.0.13457-deb0) ...
Setting up pylon (5.2.0.13457-deb0) ...
Checking if /usr/share/hwdata/usb.ids must be updated
Your usb hardware database is up to date. Nothing to do.
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for bamfdaemon (0.5.3+18.04.20180207.2-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for hicolor-icon-theme (0.17-2) ...
/opt/pylon5/bin/pylon-config
Found a pylon Installation with version 5 or greater
#All required rosdeps installed successfully
joy@support:~/catkin_ws/src$
```

```
$ sudo rosdep install --from-paths . --ignore-src --rosdistro=$ROS_DISTRO -y
```

Build **pylon_ros_camera** through **catkin_make** after changing to the workspace folder:

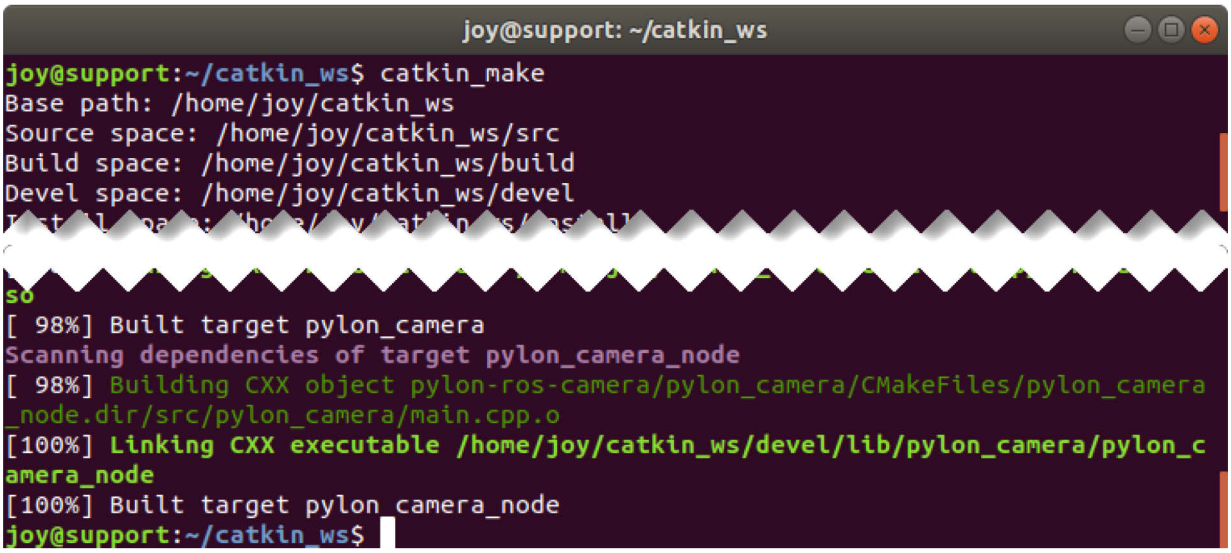
Change to the workspace folder.

Build **pylon_ros_camera** using **catkin_make**.



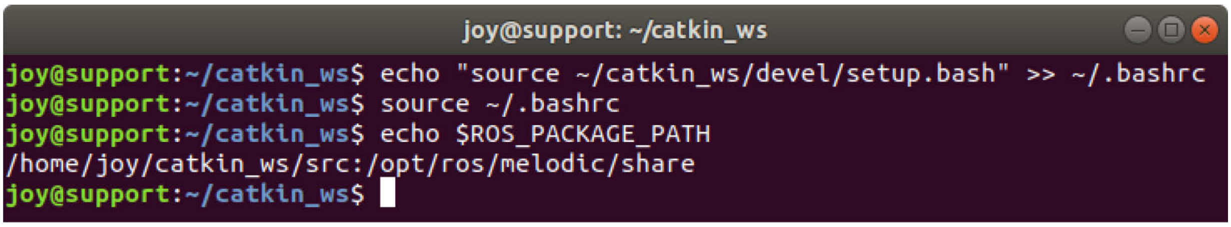
```
joy@support:~/catkin_ws/src$ cd ~/catkin_ws
joy@support:~/catkin_ws$ catkin_make clean
Base path: /home/joy/catkin_ws
Source space: /home/joy/catkin_ws/src
Build space: /home/joy/catkin_ws/build
Devel space: /home/joy/catkin_ws/devel
Install space: /home/joy/catkin_ws/install
####
#### Running command: "cmake /home/joy/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/home
/joy/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/joy/catkin_ws/install -G Unix
Makefiles" in "/home/joy/catkin_ws/build"
####
-- ==> add_subdirectory(pylon-ros-camera/pylon_camera)
-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy
-- Configuring done
-- Generating done
-- Build files have been written to: /home/joy/catkin_ws/build
####
#### Running command: "make clean -j2 -l2" in "/home/joy/catkin_ws/build"
####
joy@support:~/catkin_ws$
```

$ cd ~/catkin_ws && catkin_make clean



```
joy@support:~/catkin_ws$ catkin_make
Base path: /home/joy/catkin_ws
Source space: /home/joy/catkin_ws/src
Build space: /home/joy/catkin_ws/build
Devel space: /home/joy/catkin_ws/devel
so
[ 98%] Built target pylon_camera
Scanning dependencies of target pylon_camera_node
[ 98%] Building CXX object pylon-ros-camera/pylon_camera/CMakeFiles/pylon_camera
_node.dir/src/pylon_camera/main.cpp.o
[100%] Linking CXX executable /home/joy/catkin_ws/devel/lib/pylon_camera/pylon_c
amera_node
[100%] Built target pylon_camera_node
joy@support:~/catkin_ws$
```

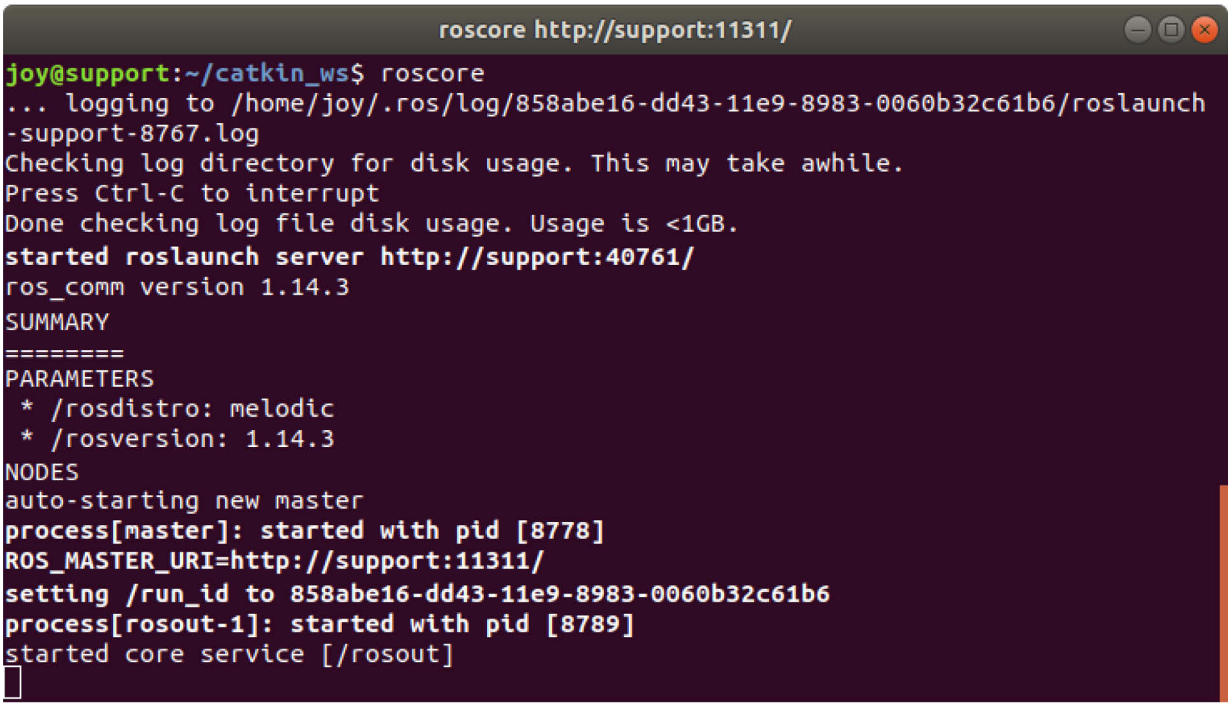
$ catkin_make



```
joy@support:~/catkin_ws$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
joy@support:~/catkin_ws$ source ~/.bashrc
joy@support:~/catkin_ws$ echo $ROS_PACKAGE_PATH
/home/joy/catkin_ws/src:/opt/ros/melodic/share
joy@support:~/catkin_ws$
```

$ echo “source ~/catkin_ws/devel/setup.bash” >> ~/.bashrc

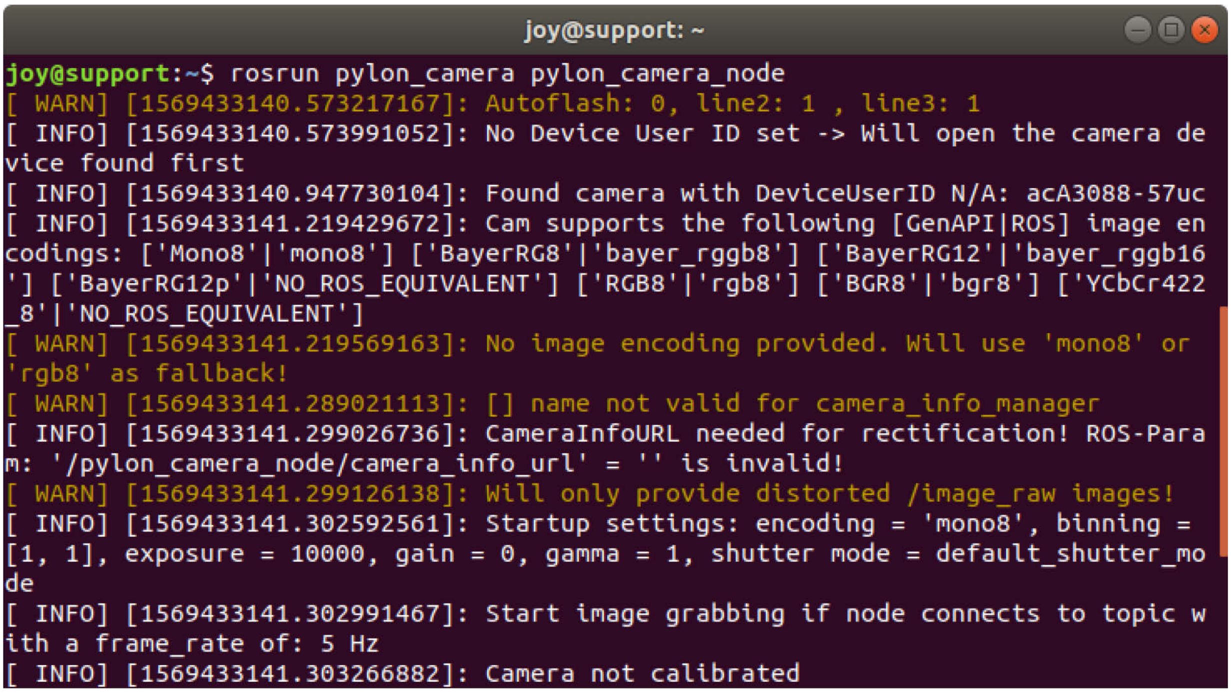
Run roscore as a prerequisite for ROS node communication.



```
roscore http://support:11311/
joy@support:~/catkin_ws$ roscore
... logging to /home/joy/.ros/log/858abe16-dd43-11e9-8983-0060b32c61b6/roslaunch
-support-8767.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://support:40761/
ros_comm version 1.14.3
SUMMARY
========
PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.3
NODES
auto-starting new master
process[master]: started with pid [8778]
ROS_MASTER_URI=http://support:11311/
setting /run_id to 858abe16-dd43-11e9-8983-0060b32c61b6
process[rosout-1]: started with pid [8789]
started core service [/rosout]
```

$ roscore

Open another terminal instance and run the pylon-ROS-camera driver package as a node. It will occupy this newly opened terminal as well.



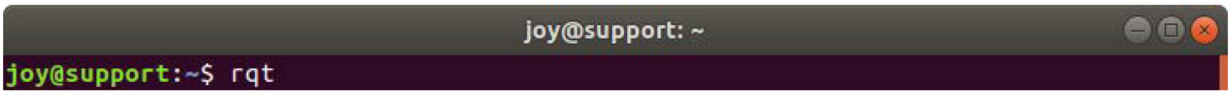
```
joy@support: ~
joy@support:~$ rosrun pylon_camera pylon_camera_node
[ WARN] [1569433140.573217167]: Autoflash: 0, line2: 1 , line3: 1
[ INFO] [1569433140.573991052]: No Device User ID set -> Will open the camera de
vice found first
[ INFO] [1569433140.947730104]: Found camera with DeviceUserID N/A: acA3088-57uc
[ INFO] [1569433141.219429672]: Cam supports the following [GenAPI|ROS] image en
codings: ['Mono8'|'mono8'] ['BayerRG8'|'bayer_rggb8'] ['BayerRG12'|'bayer_rggb16
'] ['BayerRG12p'|'NO_ROS_EQUIVALENT'] ['RGB8'|'rgb8'] ['BGR8'|'bgr8'] ['YCbCr422
_8'|'NO_ROS_EQUIVALENT']
[ WARN] [1569433141.219569163]: No image encoding provided. Will use 'mono8' or
'rgb8' as fallback!
[ WARN] [1569433141.289021113]: [] name not valid for camera_info_manager
[ INFO] [1569433141.299026736]: CameraInfoURL needed for rectification! ROS-Para
m: '/pylon_camera_node/camera_info_url' = '' is invalid!
[ WARN] [1569433141.299126138]: Will only provide distorted /image_raw images!
[ INFO] [1569433141.302592561]: Startup settings: encoding = 'mono8', binning =
[1, 1], exposure = 10000, gain = 0, gamma = 1, shutter mode = default_shutter_mo
de
[ INFO] [1569433141.302991467]: Start image grabbing if node connects to topic w
ith a frame_rate of: 5 Hz
[ INFO] [1569433141.303266882]: Camera not calibrated
```

$ rosrun pylon_camera pylon_camera_node

This node is now operating with the camera and provides received images via the topic channel.

To merely view the images you can use the **image_view** node of the **image_pipeline** node stack. This node subscribes to the provided image topics. However, because of the more extended functionalities of image display and manipulation (see below) Basler recommends to use the GUI - based **rqt** framework.

Open a third terminal and execute the following command line.



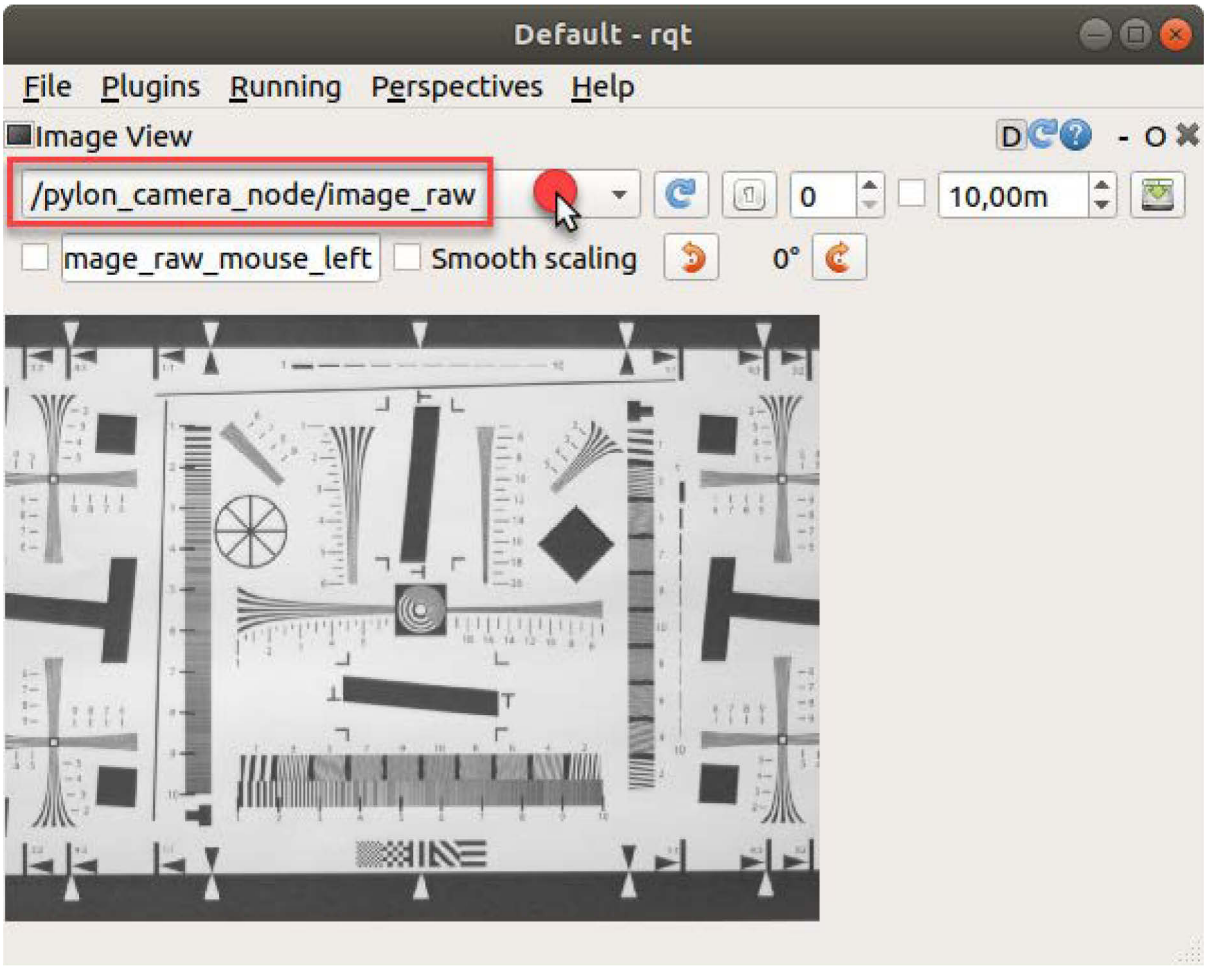
```
joy@support:~$ rqt
```

```
$ rqt
```

The framework GUI opens.

If not yet done, open the **Plugins:Visualization** menu and select **Image View**. This enables permanent image display.

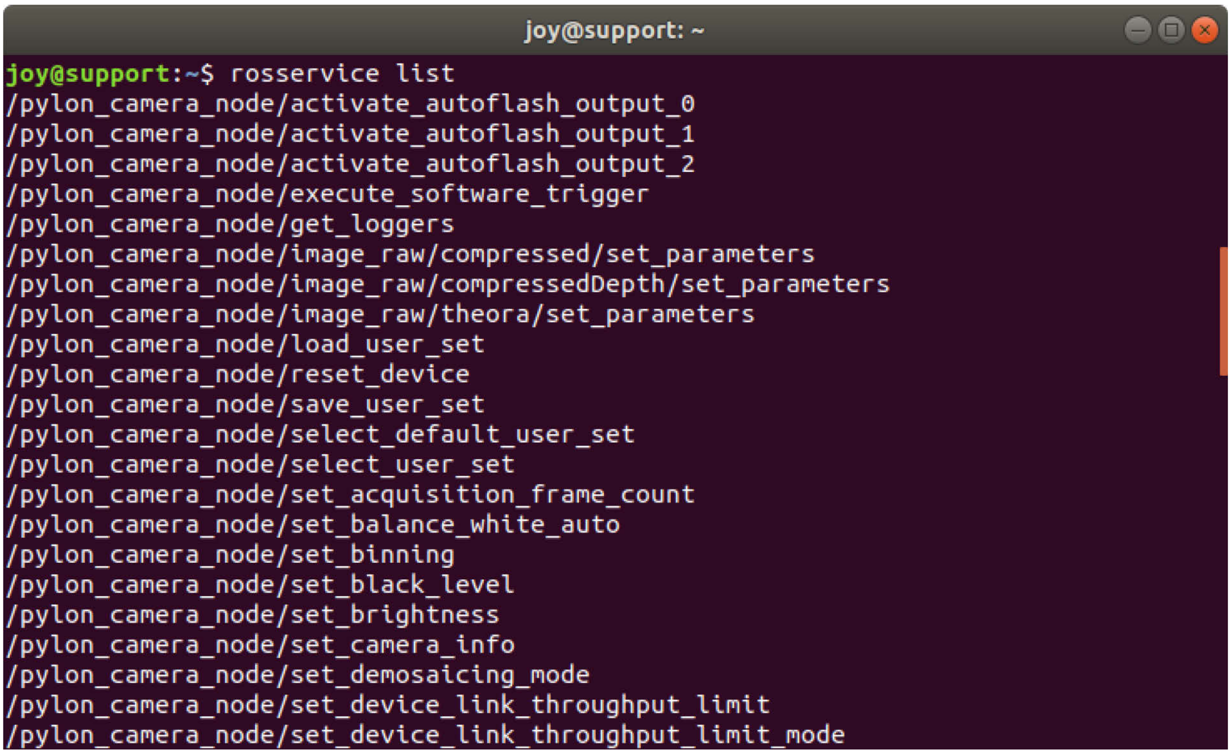
An image viewer control opens where the camera’s live images can be seen, zoomed, and saved. Apply the **/pylon_camera_node/image_raw** topic.



The camera interfacing is complete.

3 Camera Control

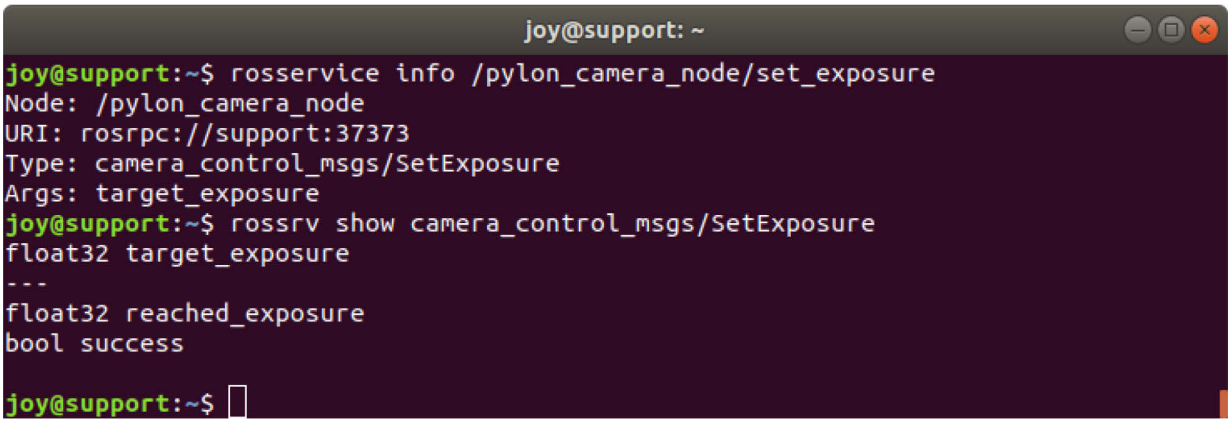
To control the cameras by setting camera parameters, so-called services are used. Contrary to single message topics, the services are able to handle request Reply communication. Therefore, a pair of messages defines them. The abilities of the **pylon_camera** node can be seen by issuing ROS commands like **rosservice list**, **rosservice info**, and **rossrv show**. The execution is realized by **rosservice call**.



```
joy@support:~$ rosservice list
/pylon_camera_node/activate_autoflash_output_0
/pylon_camera_node/activate_autoflash_output_1
/pylon_camera_node/activate_autoflash_output_2
/pylon_camera_node/execute_software_trigger
/pylon_camera_node/get_loggers
/pylon_camera_node/image_raw/compressed/set_parameters
/pylon_camera_node/image_raw/compressedDepth/set_parameters
/pylon_camera_node/image_raw/theora/set_parameters
/pylon_camera_node/load_user_set
/pylon_camera_node/reset_device
/pylon_camera_node/save_user_set
/pylon_camera_node/select_default_user_set
/pylon_camera_node/select_user_set
/pylon_camera_node/set_acquisition_frame_count
/pylon_camera_node/set_balance_white_auto
/pylon_camera_node/set_binning
/pylon_camera_node/set_black_level
/pylon_camera_node/set_brightness
/pylon_camera_node/set_camera_info
/pylon_camera_node/set_demosaicing_mode
/pylon_camera_node/set_device_link_throughput_limit
/pylon_camera_node/set_device_link_throughput_limit_mode
```

```
$ rosservice list
```

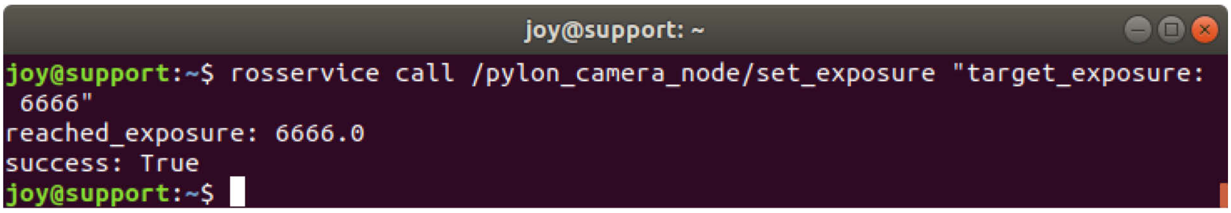
As an example, the exposure time target can be set after finding the argument **target_exposure**.



```
joy@support:~$ rosservice info /pylon_camera_node/set_exposure
Node: /pylon_camera_node
URI: rosrpc://support:37373
Type: camera_control_msgs/SetExposure
Args: target_exposure
joy@support:~$ rossrv show camera_control_msgs/SetExposure
float32 target_exposure
---
float32 reached_exposure
bool success

joy@support:~$ 
```

```
$ rosservice info /pylon_camera_node/set_exposure
```

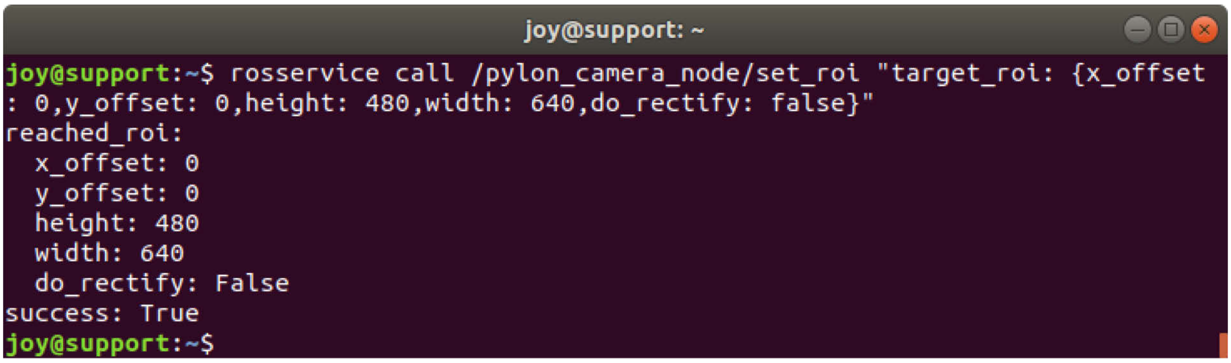
```
$ rossrv show camera_control_msgs/SetExposure
```



```
joy@support:~$ rosservice call /pylon_camera_node/set_exposure "target_exposure:
 6666"
reached_exposure: 6666.0
success: True
joy@support:~$
```

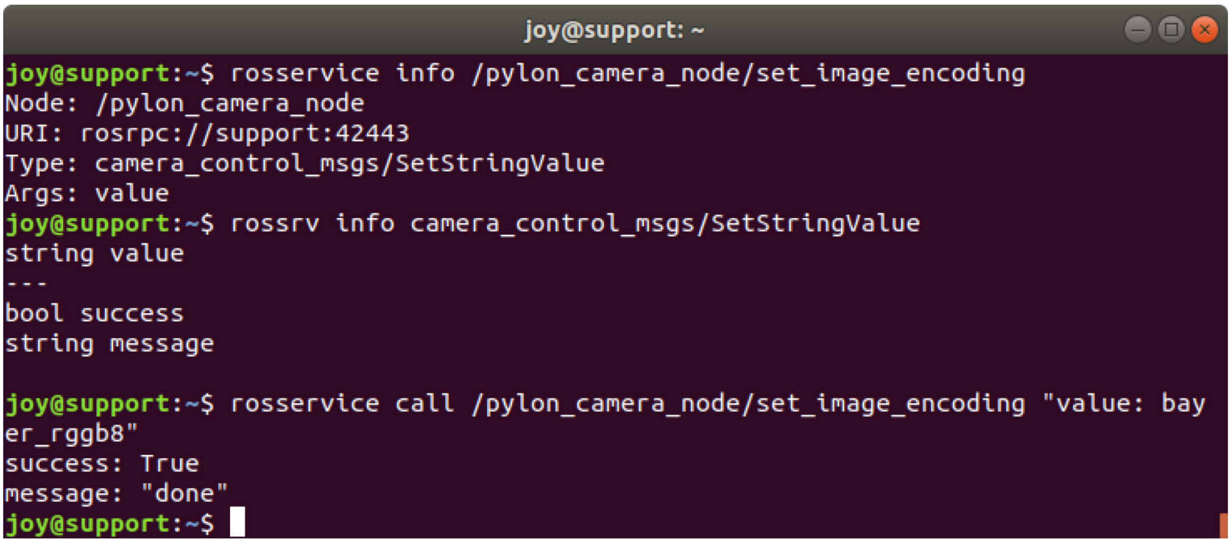
```
$ rosservice call /pylon_camera_node/set_exposure “target_exposure: 6666”
```

Some specific service calls concern e.g. the definition of a ROI setup and the selection of a pixel format. See the related sample code:



```
joy@support:~$ rosservice call /pylon_camera_node/set_roi "target_roi: {x_offset
: 0,y_offset: 0,height: 480,width: 640,do_rectify: false}"
reached_roi:
  x_offset: 0
  y_offset: 0
  height: 480
  width: 640
  do_rectify: False
success: True
joy@support:~$
```

```
$ rosservice call /pylon_camera_node/set_roi “target_roi: {x_offset: 0,
y_offset: 0, height: 480, width: 640, do_rectify: false}”
```



```
joy@support:~$ rosservice info /pylon_camera_node/set_image_encoding
Node: /pylon_camera_node
URI: rosrpc://support:42443
Type: camera_control_msgs/SetStringValue
Args: value
joy@support:~$ rossrv info camera_control_msgs/SetStringValue
string value
---
bool success
string message

joy@support:~$ rosservice call /pylon_camera_node/set_image_encoding "value: bay
er_rggb8"
success: True
message: "done"
joy@support:~$
```

```
$ rosservice call /pylon_camera_node/set_image_encoding “value: bayer_rggb8”
```

4 Driver Adjustment

ROS packages are open source projects. The ROS driver package, presented in this document serves as an example. You can, however, program your own ROS driver package according to your needs.

To get informed about latest developments of the pylon-ROS-camera driver packet, access the issue tracker on the GitHub for pylon-ROS-camera.

Revision History

Document Number	Date	Changes
AW00149101000	17 May 2018	Initial release version of this document.
AW00149102000	02 Dec 2019	Considered new ROS camera driver software, specifically modified for use with Basler cameras (with **pylon_ros_camera** (pylon-ROS-camera driver packet) replacing **pylon_camera**; for more information, see section 2.4).