4.17 스터디 송주훈

목표: ROS2 패키지 설치하기

패키지명: ROS2 USB Camera Node

주소: https://github.com/klintan/ros2_usb_camera

README Apache-2.0 license

ROS2 USB Camera Node

Its based on both the image_tools cam2image demos for ROS2 as well as the libuvc and usb_cam project for ROS1.

Features

- CameraInfo available
- CompressedImage topic (see compressed images for republishing using image_transport)
- Image topic
- Select camera (running the node for each camera connected)

There might be major changes to the code as it is a WIP. This is a simple camera driver using OpenCV. With this comes less flexibility for custom camera settings etc but simple to setup and use. If you have more complex requirements I've listed some alternate packages in the Alternate packages section.

Topics

- `/camera_info` - topic for camera info
- `/image_raw` - topic for raw image data

Installation

Make sure to run setup.bash and local_setup.bash for all dependencies or symlink them into the repo.

Run

```
colcon build
```

Usage

`ros2 run usb_camera_driver usb_camera_driver_node __ns:=/<your namespace> __params:=config.yaml`

Available parameters:

- `frame_id` -> transform frame_id of the camera, defaults to "camera"
- `image_width` -> image width (1280 default)
- `image_height` -> image height (720 default)
- `fps` -> video fps (10 default)
- `camera_id` -> id of camera to capture (0 - 100, 0 default)

Calibration files

To use the camera info functionality you need to load a file from the camera_calibration (https://github.com/ros-perception/image_pipeline/tree/ros2/camera_calibration) library and put it in/name it `file:///Users/<youruser>/.ros/camera_info/camera.yaml`

Compressed images

To get compressed images (works seamlessly with web streaming) republish the topic using image_transport which is available for ROS2.

`ros2 run image_transport republish raw in:=image_raw compressed out:=image_raw_compressed`

Make sure to link/install https://github.com/ros-perception/image_transport_plugins/tree/ros2 before to enable compressed image republishing using image_transport since its not included in the base package. More information here http://wiki.ros.org/image_transport, here http://wiki.ros.org/compressed_image_transport and here https://answers.ros.org/question/35183/compressed-image-to-image/

README 파일을 통해 ROS2에서 USB 카메라를 사용할 수 있는 노드임을 확인할 수 있었다.

주요 토픽으로는

/camera_info: 카메라 정보가 담긴 토픽

/image_raw: 원시 이미지 데이터를 담은 토픽

으로 이루어져있다.

설치과정

```
hbk@hbk-IdeaPad-Slim-3-16ABR8:~/ros2_example_ws$ colcon build --packages-select
usb_camera_driver
Starting >>> usb_camera_driver
--- stderr: usb_camera_driver
CMake Error at CMakeLists.txt:21 (find_package):
  By not providing "Findcamera_info_manager.cmake" in CMAKE_MODULE_PATH this
  project has asked CMake to find a package configuration file provided by
  "camera_info_manager", but CMake did not find one.

  Could not find a package configuration file provided by
  "camera_info_manager" with any of the following names:

    camera_info_managerConfig.cmake
    camera_info_manager-config.cmake

  Add the installation prefix of "camera_info_manager" to CMAKE_PREFIX_PATH
  or set "camera_info_manager_DIR" to a directory containing one of the above
  files.  If "camera_info_manager" provides a separate development package or
  SDK, be sure it has been installed.


---
Failed   <<< usb_camera_driver [0.98s, exited with code 1]

Summary: 0 packages finished [1.21s]
  1 package failed: usb_camera_driver
  1 package had stderr output: usb_camera_driver
```

해당 오류를 해결하기 위해

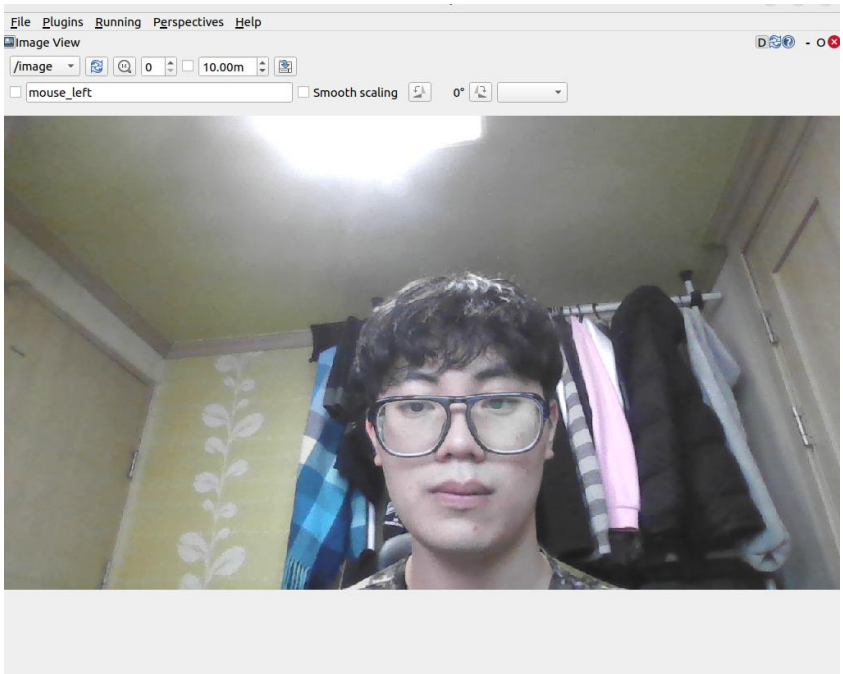
```
hbk@hbk-IdeaPad-Slim-3-16ABR8:~$ sudo apt install ros-humble-camera-info-manager
```

다음 명령어를 실행하고 다시 빌드를 진행하였다.

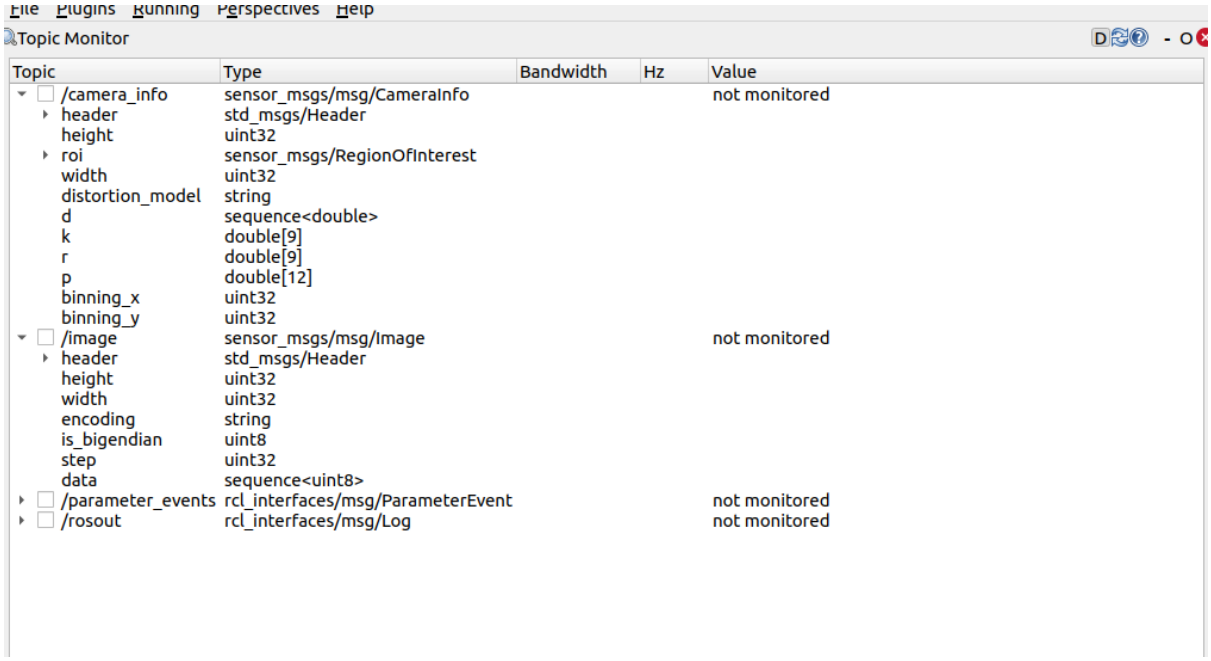
```
hbk@hbk-IdeaPad-Slim-3-16ABR8:~/ros2_example_ws$ colcon build --packages-select
usb_camera_driver
Starting >>> usb_camera_driver
Finished <<< usb_camera_driver [7.34s]

Summary: 1 package finished [7.54s]
```

```
--prefix                usb_camera_driver_node
hbk@hbk-IdeaPad-Slim-3-16ABR8:~/ros2_example_ws$ ros2 run usb_camera_driver usb_
camera_driver_node
[INFO] [1744871384.131527939] [usb_camera_driver]: camera calibration URL: file:
//config/camera.yaml
[ERROR] [1744871384.131625797] [usb_camera_driver]: Invalid camera calibration U
RL: file://config/camera.yaml
[ WARN:0] global ./modules/videoio/src/cap_gstreamer.cpp (1100) open OpenCV | GS
treamer warning: Cannot query video position: status=0, value=-1, duration=-1
```

실행시 calibration 을 위한 yaml 파일이 존재하지 않아 error가 발생하는 것을 확인했지만 토픽의 발행은 정상적으로 이루어진다.

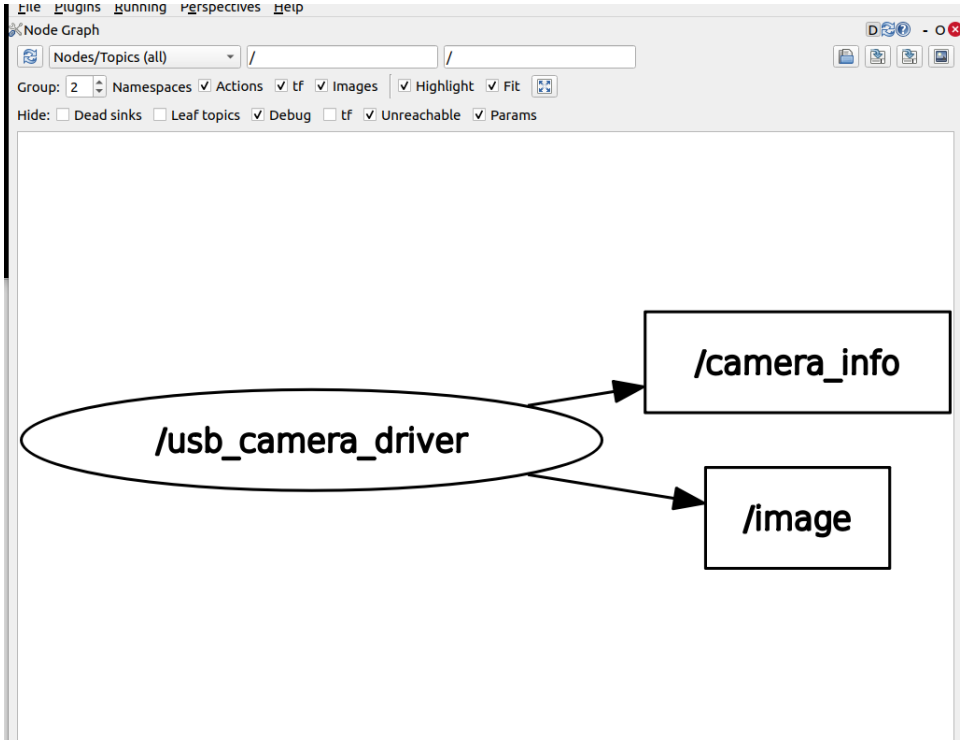


노트북의 웹캠을 통한 이미지를 rqt의 image view를 통해 볼 수 있었다.



Topic	Type	Bandwidth	Hz	Value
/camera_info	sensor_msgs/msg/CameraInfo			not monitored
header	std_msgs/Header			
height	uint32			
roi	sensor_msgs/RegionOfInterest			
width	uint32			
distortion_model	string			
d	sequence<double>			
k	double[9]			
r	double[9]			
p	double[12]			
binning_x	uint32			
binning_y	uint32			
/image	sensor_msgs/msg/Image			not monitored
header	std_msgs/Header			
height	uint32			
width	uint32			
encoding	string			
is_bigendian	uint8			
step	uint32			
data	sequence<uint8>			
/parameter_events	rcl_interfaces/msg/ParameterEvent			not monitored
/rosout	rcl_interfaces/msg/Log			not monitored

앞서 언급한 /camera_info와 /image를 확인할 수 있었다.



usb_camera_driver 노드와 발행하는 토픽을 확인할 수 있다.

```xml
package.xml
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>usb_camera_driver</name>
  <version>0.1.1</version>
  <description>Package containing a simple web camera setup</description>
  <maintainer email="ankl@kth.se">Andreas Klintberg</maintainer>
  <license>Apache License 2.0</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <depend>libopencv-dev</depend>
  <depend>rclcpp</depend>
  <depend>rclcpp_components</depend>
  <depend>std_msgs</depend>
  <depend>sensor_msgs</depend>
  <depend>image_transport</depend>
  <depend>camera_calibration_parsers</depend>
  <depend>camera_info_manager</depend>
  <depend>launch_ros</depend>

  <test_depend>ament_lint_auto</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>

</package>
```

package.xml을 통해 의존성 파일을 확인할 수 있었다. 해당 파일은 cmake로 빌드 되어 ament_cmake 를 확인할 수 있고 의존성 패키지로 std_msgs와 sensor_msgs를 볼 수 있다.

```cmake


find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(rclcpp_components REQUIRED)
find_package(std_msgs REQUIRED)
find_package(sensor_msgs REQUIRED)
find_package(camera_info_manager REQUIRED)
find_package(image_transport REQUIRED)
find_package(camera_calibration_parsers REQUIRED)
find_package(OpenCV 4 REQUIRED)

include_directories(include ${cv_bridge_INCLUDE_DIRS})


add_library(usb_camera_driver SHARED
  src/usb_camera_driver.cpp)
target_compile_definitions(usb_camera_driver
  PRIVATE "COMPOSITION_BUILDING_DLL")

ament_target_dependencies(usb_camera_driver
    "rclcpp"
    "rclcpp_components"
    "sensor_msgs"
    "std_msgs"
    "camera_info_manager"
    "image_transport"
    "camera_calibration_parsers"
    "OpenCV")
```

해당 CMakeLists.txt를 통해서도 필요한 패키지들을 확인할 수 있었다.

usb_camera_driver.hpp

include > usb_camera_driver.hpp

```cpp
#include <stdio.h>
#include <iostream>
#include "rclcpp/rclcpp.hpp"

#include "std_msgs/msg/string.hpp"
#include "sensor_msgs/msg/image.hpp"
#include "sensor_msgs/msg/camera_info.hpp"

#include "sensor_msgs/image_encodings.hpp"

#include <camera_info_manager/camera_info_manager.hpp>
#include <image_transport/image_transport.hpp>

#include "opencv2/highgui/highgui.hpp"
#include <opencv2/imgproc.hpp>

namespace usb_camera_driver
{

class CameraDriver : public rclcpp::Node {
public:
    explicit CameraDriver(const rclcpp::NodeOptions&);
    ~CameraDriver() {};

private:
    rclcpp::TimerBase::SharedPtr timer_;
    cv::Mat frame;
    cv::Mat flipped_frame;
    cv::VideoCapture cap;

    bool is_flipped;

    std::string frame_id_;
    int image_height_;
    int image_width_;
    double fps_;
    int camera_id;

    std::chrono::steady_clock::time_point last_frame_;

    std::shared_ptr<camera_info_manager::CameraInfoManager> cinfo_manager_;
    image_transport::CameraPublisher camera_info_pub_;

    std::shared_ptr<sensor_msgs::msg::Image> image_msg_;

    std::shared_ptr<sensor_msgs::msg::Image> ConvertFrameToMessage(cv::Mat & frame);

    void ImageCallback();


};
} // namespace usb_camera_driver
#endif //USB_CAMERA_DRIVER__CAMERA_DRIVER_HPP_
```

c++로 작성된 헤더 파일을 통해 sensor_msgs/msg의 image, camera_info를 확인할 수 있었고 노드가 어떻게 구성되는지 확인할 수 있었다.