ROS2_3 일차_최범수

19 기 예비단원 최범수

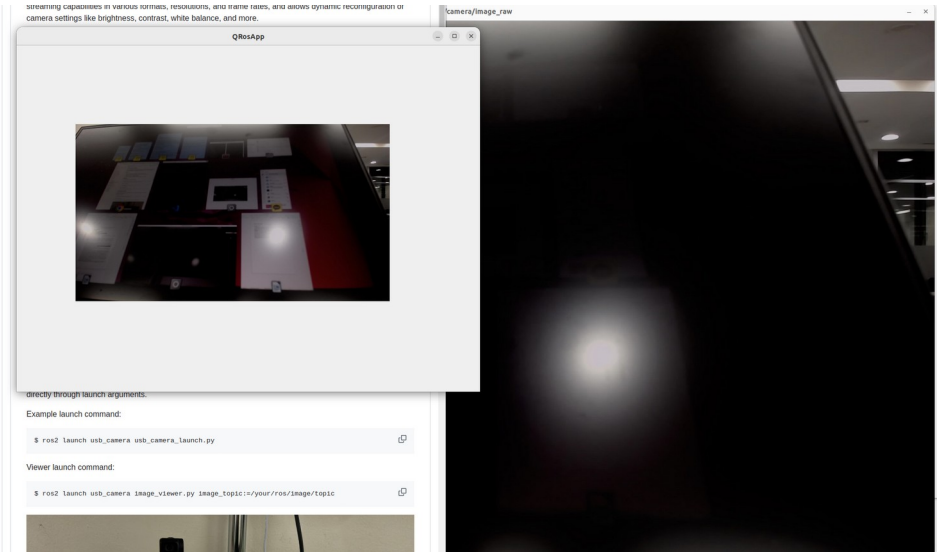
1. HW_001
main_window.hpp 이다. Q_OBJECT, UI 구성과 이미지를 업데이트할 함수들을 구성한 class 객체를 구성하였다. 생성자를 구현해 놓았고 private 멤버 변수내에서 포인터를 이용해 UI 요소들에 접근하능 기능이 있다. closeEvent 를 사용해 문제가 생길 부분들을 처리하였다. Public 슬롯 내에서는 이미지를 업데이트 하는 함수를 넣었고 const Qpixmap& pixmap 을 통해서 중간 다리 역할인 QPixmap 에서 전달받아 UI 에 이미지를 업데이트 하도록 하였다.
Qnode.hpp 에서는 subscribe 기능을 구현해 놓았고 imageCallback 함수를 private 멤버 변수로 구성했다. Q_SIGNALS 에서 새로운 이미지를 전달받았는지의 신호를 다룬다.
소스코드이다. main_window.cpp 에서는각각의 GUI 를 connect 해서 각각의 종속 함수들과 연결하였고 이미지를 출력하는 label 에 640, 480 의 규격을 가지도록 keepaspectRatio 를 이용해 Qlabel 에 이미지를 설정하였다. 그 뒤에는 closeEvent 를 통해 종료가 가능하다. Qnode.cpp 에서 기본 생성자가 존재하고 이미지를 subscribe 하도록 image_raw 의 토픽을 구독했다. Qnode::run 내부에서 loop_rate 를 20 으로 설정해 20Hz 로 설정하였고 spin_some(node)를 통해서 콜백함수를 호출했다. 그리고 호출된 함수는 Callback 함수로서 ROS 이미지 메시지를 OpenCV 이미지로 변환해서 cv::Mat 형식으로 변환하였다. 그 후 cv::Mat 의 이미지를 QT 의 Qimage 형식으로 변환하였고 QPixmap 객체를 생성해서 Q_EMIT 을 통해 newImage 신호로 GUI 에 전달하였다. 쉽게 말하면 imageCallback 함수를 통해 수신한 이미지를 OpenCV 와 QT 형식으로 변환해서 GUI 에 전달하였다.

```cpp
void QNode::imageCallback(const sensor_msgs::msg::Image::SharedPtr msg)
{
  try
  {
    // ROS 이미지 메시지를 OpenCV 이미지로 변환
    cv::Mat cv_image = cv_bridge::toCvCopy(msg, "bgr8")->image;

    // OpenCV 이미지를 QImage로 변환
    QImage qimage(cv_image.data, cv_image.cols, cv_image.rows, cv_image.step, QImage::Format_RG
    QPixmap pixmap = QPixmap::fromImage(qimage.rgbSwapped());  // RGB 형식 변경

    // newImage 신호로 MainWindow에 pixmap 전달
    Q_EMIT newImage(pixmap);
  }
  catch (cv_bridge::Exception& e)
  {
    RCLCPP_ERROR(node->get_logger(), "Error converting image: %s", e.what());
  }
}
```

```cpp
void MainWindow::updateImage(const QPixmap& pixmap)
{
  ui->label->setPixmap(pixmap.scaled(640, 480, Qt::KeepAspectRatio));  // QLabel에 이미지 설정
}
```



streaming capabilities in various formats, resolutions, and frame rates, and allows dynamic reconfiguration of camera settings like brightness, contrast, white balance, and more.

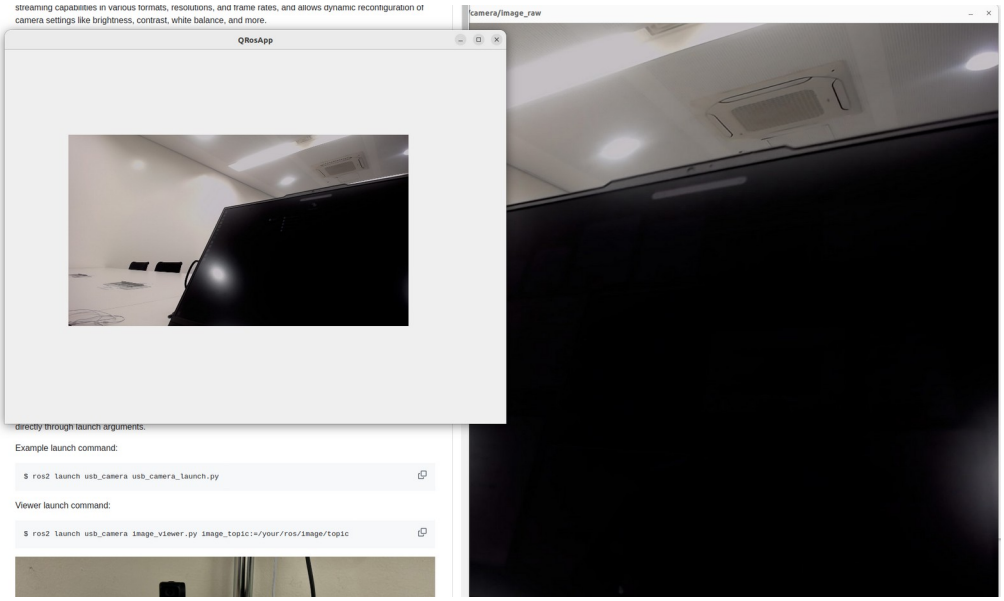
directly through launch arguments.

Example launch command:

```
$ ros2 launch usb_camera usb_camera_launch.py
```

Viewer launch command:

```
$ ros2 launch usb_camera image_viewer.py image_topic:=/your/ros/image/topic
```



streaming capabilities in various formats, resolutions, and frame rates, and allows dynamic reconfiguration of camera settings like brightness, contrast, white balance, and more.

directly through launch arguments.

Example launch command:

```
$ ros2 launch usb_camera usb_camera_launch.py
```

Viewer launch command:

```
$ ros2 launch usb_camera image_viewer.py image_topic:=/your/ros/image/topic
```