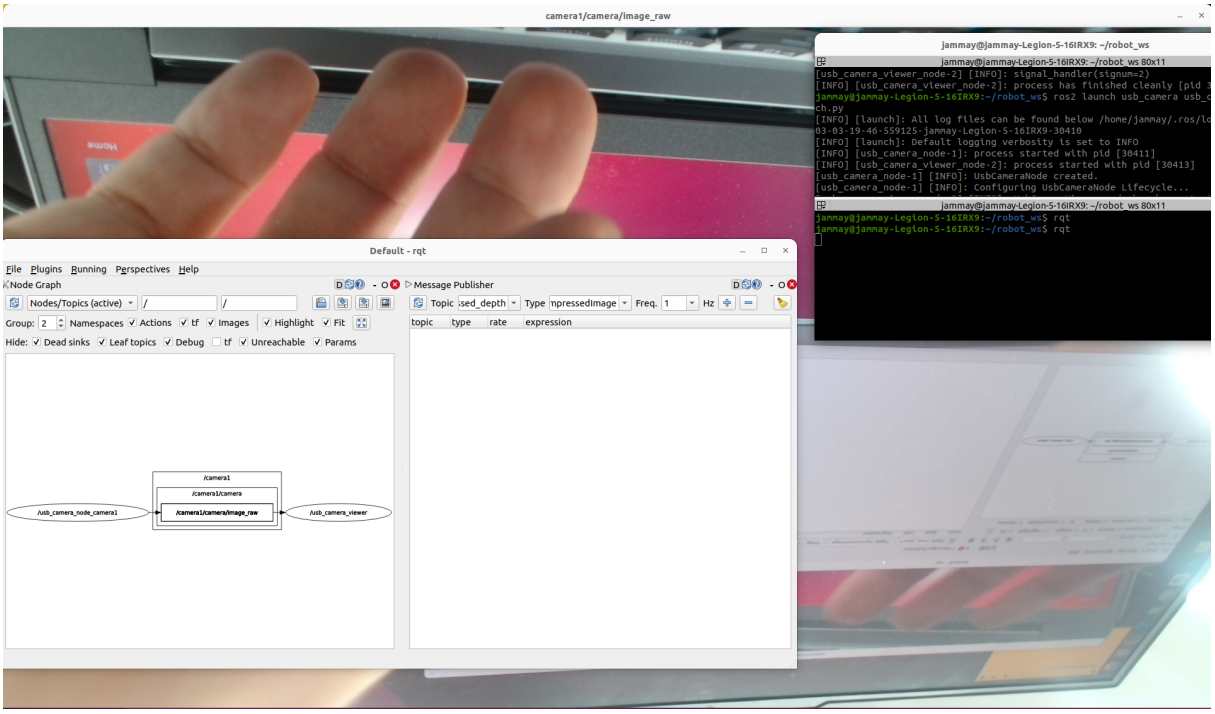
ROS2_day3

day3

먼저 ros2 launch usb_camera usb_camera_launch.py 실행 시켜서 통신 확인

usb_cam_node_cam1⇒usb_cam_viewer

토픽명 : /camera1/camera/image_raw을 sub 받기로 계획함



cv_bridge, sensor_msgs 의존송 추가

전체적 구성

sensor_msgs 형태의 카메라 데이터 sub ⇒ cv_bridge::tocvcopy로 ros2 메시지를 opencv 메시지로 변환⇒변환된 이미지를 main_window로 전달⇒main_window에서 QImage로 변환

초기 구성 QImage로 변환도 qnode에서 해주었다. 그렇게 하니 파일들의 구성의 의도(main_window는 qt gui 관련) 이라고 생각하여 cv::Mat 이미지를 받아와 QImage로 변환하여 작업

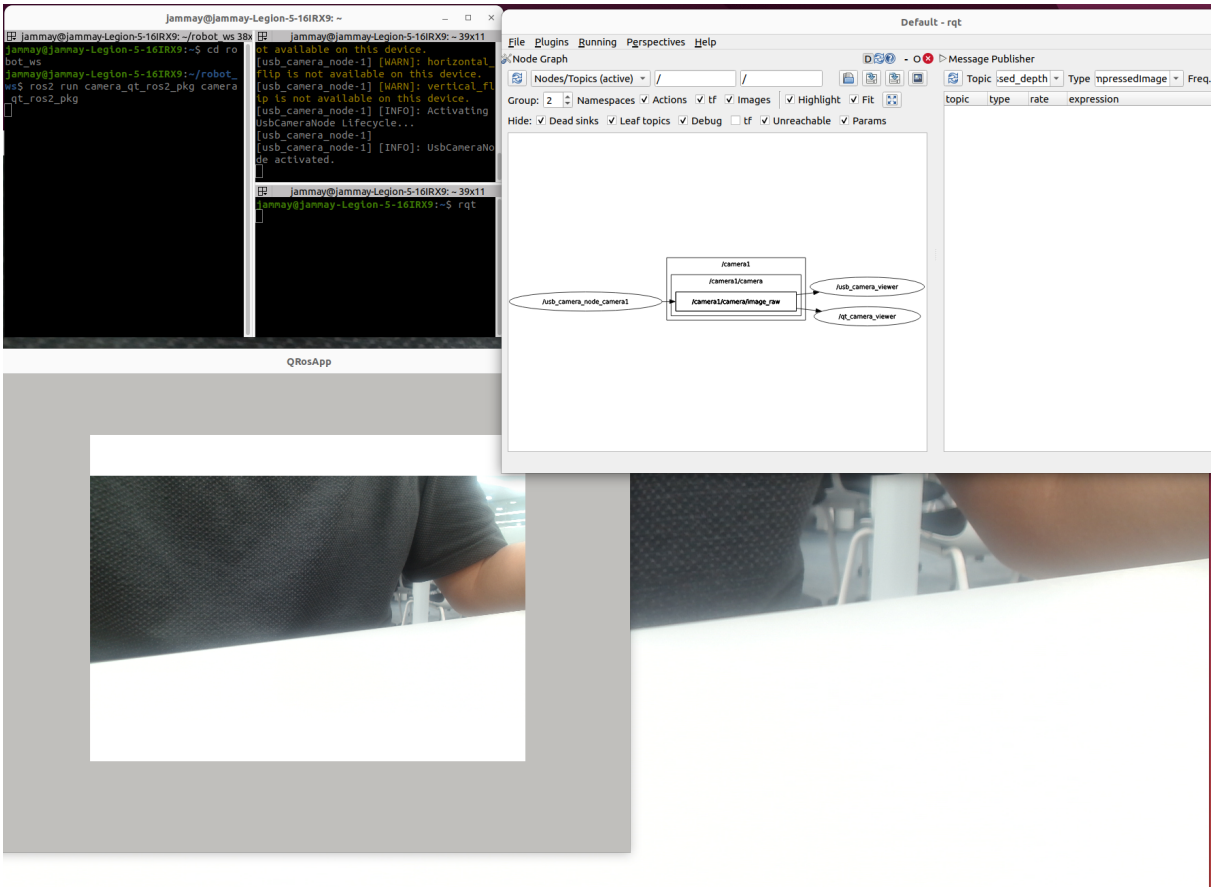
초기 구성으로 했으면 Qimgae로 변환을 qnode에서 해주면 cv::Mat의 데이터를 QImage가 직접 참조할 때, 해당 데이터가 비동기적으로 변경되거나 삭제될 가능성이 있기 때문, 이를 방지하기 위해 데이터를 복사하여 안정성을 확보하는 것이 좋다. 그래서 처음에 copy()함수를 써줘서 cv::Mat의 데이터를 복사하여 새로운 메모리 영역에 저장하여 main_window에 넘기는 구성이였는데 그냥 Qimgae로 변환하는 걸 main_window에 함으로써 copy()도 안쓰고 싶어서 수정하여 진행함.

정리하자면

초기 : qnode(sensor_msgs)⇒qnode(convert_cv_image)⇒qnode(connect.cv_image를 QImage로 convert)⇒main_window(Qimage 설정)

현재 : qnode(sensor_msgs)⇒qnode(convert_cv_image)⇒main_window(connect.cv_image를 QImage로 convert)

```
void MainWindow::displayImage(const cv::Mat &image)
{
    QImage qimage(image.data, image.cols, image.rows, image.step
    QPixmap pixmap = QPixmap::fromImage(qimage);
    ui->label->setPixmap(pixmap.scaled(ui->label->size(), Qt::Ke
    ui->label->setAlignment(Qt::AlignCenter);
}
```



cv::Mat 객체 image
객체 image.data, cols, rows, step, format_BGR888

QPixmap pixmap = QPixmap::fromImage(qimage);
QPicmap은 QT에서 이미지를 화면에 표시하는데 최적화딘 클래스, QImgae보다 더 효율적
변환된 qimgae 객체를 QPixmap객체로 변환

label→setPixmap
: QPixmap 객체를 QLabel에 표시, QLabel은 GUI에서 이미지나 텍스트를 표시

pixmap_scaled(...)
: pixmap 의 크기 조정(640)에 맞추는 것

QT::KeepAspectRatio
: 이미지의 원래 비율을 유지하면서 QLabel 크기에 맞게 조정
Qt::SmoothTransFormation
: 크기 변경시 부드럽게해줌

ui→ label→ setAlignCenter

: label안에 이미지를 중앙에 정렬(라벨의 크기가 이미지보다 큰 경우 이미지가 중에 정렬)