<RO:BIT ROS2 - 3일차 과제1 보고서>

작성자: 박기홍

직책: 19기 예비 단원

소속: RO:BIT 지능 팀

작성일: 2024년 11월 4일

목차

1. 요약

이 보고서는 ROS2 환경에서 Qt GUI를 활용하여 USB 카메라로부터 수신된 영상을 실시간으로 표시하는 프로젝트의 개발 과정과 결과를 설명한다. 본 프로젝트의 주요 목표는 ROS2 토픽을 통해 전송된 이미지를 GUI 환경에서 직관적으로 볼 수 있도록 구현하는 것이다. 이를 위해 qt_usb_camera_viewer 패키지를 개발하였다.

본 패키지는 RO:BIT의 ROS2 - 3일차 과제1에 대한 패키지로 제시된 요구 사항을 충족하는 데 중점을 두었다. 이 보고서에서는 프로젝트의 개요, 관련 이론, 개발 환경, 개발 과정, 결과 등을 다루고 있다.

2. 프젝젝트 개요

본 프로젝트의 목적은 ROS2 환경에서 Qt GUI를 이용하여 USB 카메라의 사진을 사용자에게 직관적으로 제공하는 것이다. 구체적으로, 본 프로젝트는 ROS2의 퍼블리셔-서브스크라이버 모델을 기반으로 하여, USB 카메라에서 수신한 영상을 토픽을 통해 전송하고, 이를 qt_usb_camera_viewer 패키지에서 구독하여 실시간으로 표시하는 것이 목표이다.

3. 관련 이론 및 개념

3.1 관련 개념

Qt::KeepAspectRatio: keepAspectRatio는 이미지의 가로 세로 비율을 유지하면서 주어진 영역에 맞게 크기를 조정하도록 하는 옵션이다.

4. 개발 환경

이 프로젝트는 Ubuntu 22.04에서 ROS2 Humble을 기반으로 수행되었다. 하드웨어는 Intel i9 프로세서와 32GB RAM을 갖춘 Lenovo Legion 노트북을 사용하하였다. 또한, ROS2 패키지 개발을 위해 Visual Studio Code와 Git을 사용하여 코드 작성과 버전 관리를 하였다.

소프트웨어 환경:

운영체제: Ubuntu 22.04 LTS

ROS2 버전: Humble Hawksbill

IDE 및 VCD: Visual Studio Code, Git

컴파일러: GCC (GNU Compiler Collection) 11.4.0

CMake: 프로젝트 빌드 및 패키지 관리를 위해 CMake를 사용하였으며, ROS2 패키지의 빌드 시스템인 ament_cmake를 사용하여 프로젝트를 구성하였다.

필수 패키지:

ament_cmake: ROS2 빌드 시스템을 지원하는 패키지

rclcpp: ROS2 C++ 클라이언트 라이브러리

ensor_msgs: ROS2의 표준 메시지 패키지 중 하나

Qt5: GUI 구축을 위한 라이브러리

5. 패키지 분석

이 프로젝트에서 개발한 qt_usb_camera_viewer 패키지는 USB Camera로부터 영상 데이터를 받아 Qt로 개발한 GUI 앱에 출력하기 위해 설계 되었다.

qt_usb_camera_viewer 패키지의 주요 구성 요소는 다음과 같다.

GUI

- **QLabel**: QLabel에 이미지를 출력하고, 사이즈는 640*480으로 고정한다. QPixmap을 활용하여 이미지 데이터 해상도에 상관 없이 640*480 크기인 QLabel에 출력한다. 이미지의 원본 비율을 유지하는 것이 중요하다. 예) 1920*1080 -> 640*360 (16:9) QLabel의 사이즈는 반드시 고정한다.
- **keepaspectratio**를 사용하여 구현한다.

주요 기능

- image_recongnition의 usb_camera 패키지를 활용하여 카메라 데이터를 ROS2 토픽으로 서브스크라이브한다.

이 패키지는 Qt GUI와 ROS2 간의 원활한 상호작용을 위해 로빛 17기 주장님이신 이명진 선배님의 "**ros2_create_qt_pkg**" 스크립트를 사용하여 환경을 구성했으며, 주장님의 **image_projection** 중 **usb_camera** 패키지를 사용하여 개발을 진행했다.

6. 개발 과정

6.1. 패키지 설계 및 구조

패키지와 노드 구조를 설계 할 때 노드를 독립적으로 구현하는 것에 초점을 맞추어 개발을 진행하는 것이 중요하다. 이번 프로젝트에서는 카메라를 GUI상에 출력만 하면 되기에 기본 패키지 구성 요소에서 작업을 진행하였으며, 기능은 QNode에서 구현하였다.

6.2. 코드 및 주요 기능 설명

ROS2와 Qt를 통합하여 패키지를 개발하게 되면 코드의 양이 길어질 수 밖에 없다. 구성에 대한 내용은 주석 처리를 하였으며, 이번 보고서에서는 주요 핵심 기능에 대해서만 다루도록 하겠다.

6.2.1. MainWindow

6.2.1.1. MainWindow() 생성자

```
MainWindow::MainWindow(QWidget* parent) : QMainWindow(parent), ui(new
Ui::MainWindowDesign)
{
  ui->setupUi(this);

  QIcon icon("://ros-icon.png");
  this->setWindowIcon(icon);

  qnode = new QNode();

  // QNode의 시그널과 MainWindow 슬롯 연결
  QObject::connect(qnode, &QNode::imageReceived, this,
&MainWindow::updateImage);

  QObject::connect(qnode, SIGNAL(rosShutDown()), this, SLOT(close()));
}
```

MainWindow() 생성자에서 시그널과 슬롯 연결 처리를 진행했다. connect()로 작성해도 되며, 이번 패키지에서는 QObject::를 명시하여 작성했다. 코드가 더러워 보이긴 하지만, 아직 코딩 스타일을 정확하게 이해하고 적용한 게 아니기에 이번 프로젝트에서는 QObject::를 명시해 개발해 보려고

했다. 명시해서 적은 코드가 있는 패키지와 명시하지 않고 적은 코드가 있는 패키지 중 어느 패키지와 코드가 더 깔끔하고 보기 좋은지 직접 비교하려고 한다.

6.2.1.2. updateImage) 메서드

```
// 이미지 업데이트 메서드
void MainWindow::updateImage(const QPixmap& pixmap)
{
  /* KeepAspectRatio: 이미지 원본 비율 유지
  */

ui->labelDisplayUSBCamera->setPixmap(pixmap.scaled(ui->labelDisplayUSBCa
mera->size(), Qt::KeepAspectRatio)); // QLabel에 QPixmap을 설정
}
```

updateImage() 메서드는 QNode에서 수신한 이미지를 QLabel에 표시하는 메서드이다. setPixmap 메서드를 사용해서 QLabel에 QPixmap 이미지를 설정할 수 있다. 그리고 인자로 카메라의 사이즈와 비율 유지를 위한 Qt::KeepAspectRatio를 넘겨 주어 원본 비율을 유지하면서 QLabel 크기에 맞춰 이미지를 조정하도록 하였다. KeepAspectRatio에 대한 개념이 부족해서 찾아보고, 어떻게 적용할 수 있는지 레퍼런스를 확인했다. StackOverflow에 관련 문서를 확인할 수 있었다.

*__setPixmap()__: 객체를 상수 참조로 받아서 QLabel에 설정한다.

*__scaled()__의 구조: QPixmap QPixmap::scaled(const QSize &size, Qt::AspectRatioMode aspectRatioMode) const;

6.2.2. QNode

QNode는 USB 카메라의 이미지 데이터를 서브스크라이브하고, 수신한 이미지를 MainWindow에 전달하는 노드이다.

6.2.2.1. QNode() 생성자

```
QNode::QNode()
{
  int argc = 0;
  char** argv = NULL;
  rclcpp::init(argc, argv);
  node = rclcpp::Node::make_shared("qt_usb_camera_viewer");

  /* 이미지 서브스크라이브 설정
  * 레퍼런스:
https://github.com/mjlee111/image_recognition/blob/master/image_pr
ojection/usb_camera/src/usb_camera_viewer.cpp
  * 리뷰: 오픈소스를 보고 필요한 요소를 적재적소 가져와야 하는 능력이
중요하다는 것을 몸소 느낌 -> 제대로 분석하고 파악만 한다면 패키지 개발
단축 시간 감소
  * rqt로 리버싱 느낌 내면서 하는 것도 중요함. 물로 이 과정이 선행으로
진행 되어야 어떤 토픽이 패키지 내부에 있는지 알 수 있음
  * 그래도 못 찾겠다 싶으면 오픈소스 repo로 들어가서 토픽이랑 로직이
어떻게 구성 되어있는지 찾으면 됨
  */
  image_subscription_ =
node->create_subscription<sensor_msgs::msg::Image>("/camera1/camer
a/image_raw", 10, std::bind(&QNode::imageCallback, this,
std::placeholders::_1));

  this->start();
}
```

/camera1/camera/image_raw 토픽을 서브스크라이브하고, 콜백 함수로 imageCallback을 설정하고, 변수 image_subscription_에 할당했다.

6.2.2.2. imageCallback() 메서드

```
// 이미지 콜백 메서드
void QNode::imageCallback(const sensor_msgs::msg::Image::SharedPtr
msg)
```

```
{
  // ROS2 이미지 메시지를 QImage로 변환
  QImage qImage(msg->data.data(), msg->width, msg->height,
QImage::Format_RGB888);
  QPixmap pixmap = QPixmap::fromImage(qImage);

  emit imageReceived(pixmap);  // QLabel 표시 시그널
}
```

imageCallback() 메서드는 sensor_msgs::msg::Image 메시지를 수신할 때 호출되는 메서드로, 이 메시지를 QImage로 변환한 후 QPixmap 형식으로 변경하여 GUI에 전달한다.

"QImage qImage(msg->data.data(), msg->width, msg->height, QImage::Format_RGB888);"를 통해 이미지 데이터를 QImage 형식으로 변환한다는 것을 알게 되었다. 이미지의 너비, 높이, 형식를 메시지에서 추출하여 사용하는 방법이라는 것을 공부할 수 있었다. 인자로 많은 값을 받으니 구조가 복잡하게 느껴졌다.

"QPixmap pixmap = QPixmap::fromImage(qImage);"를 통해 QImage를 QPixmap으로 변환하여 GUI에 표시 가능한 형식으로 만들 수 있다는 것을 알게 되었다.
Image 처리 관련해서는 백그라운드가 없다보니 제로부터 배우는 느낌으로 공부했다.
그동안은 그냥 Label에 할당해도 정적 이미지 처리가 가능했었는데 말이다.

이후, emit 키워드를 통해 imageReceived 시그널을 발행하고 수신된 이미지가 QLabel에 표시된다.

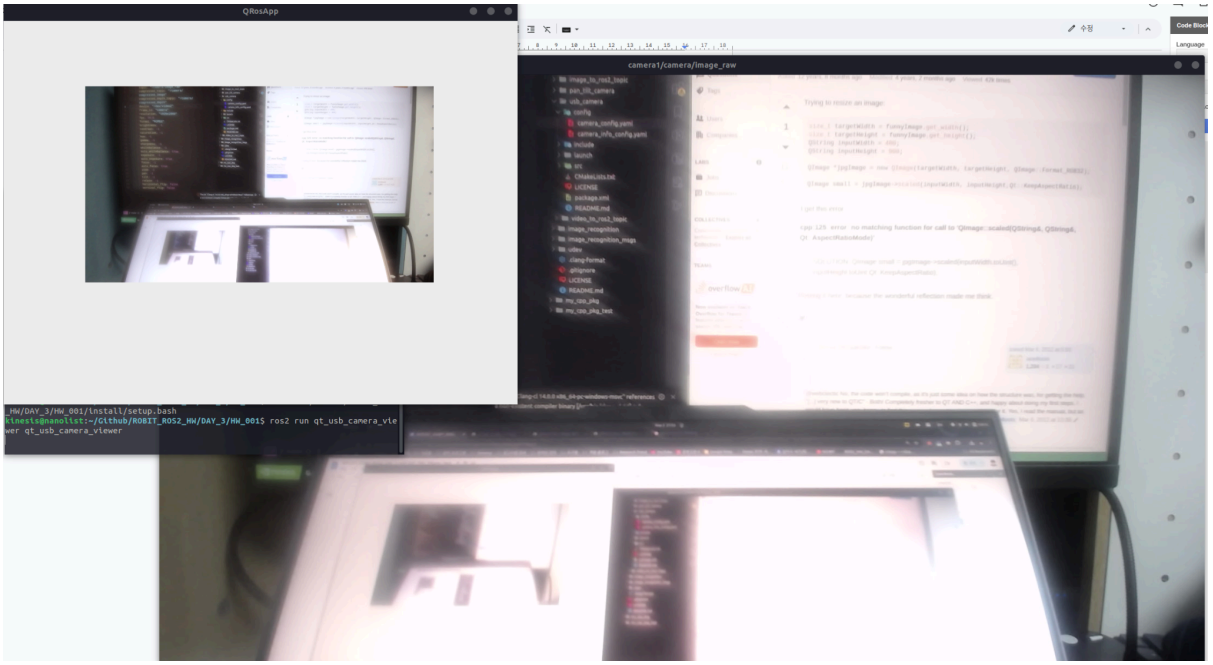
6.3.실패 사례와 시도한 방법

usb_camera 패키지를 사용하여 토픽을 알아내야 하는데, rqt를 사용해도 관련 토픽을 찾기 어려웠다. 토픽 하나씩 Call하면서 값들을 봤는데, 이미지에 대한 토픽은 보이지 않았다. 어쩌면 못

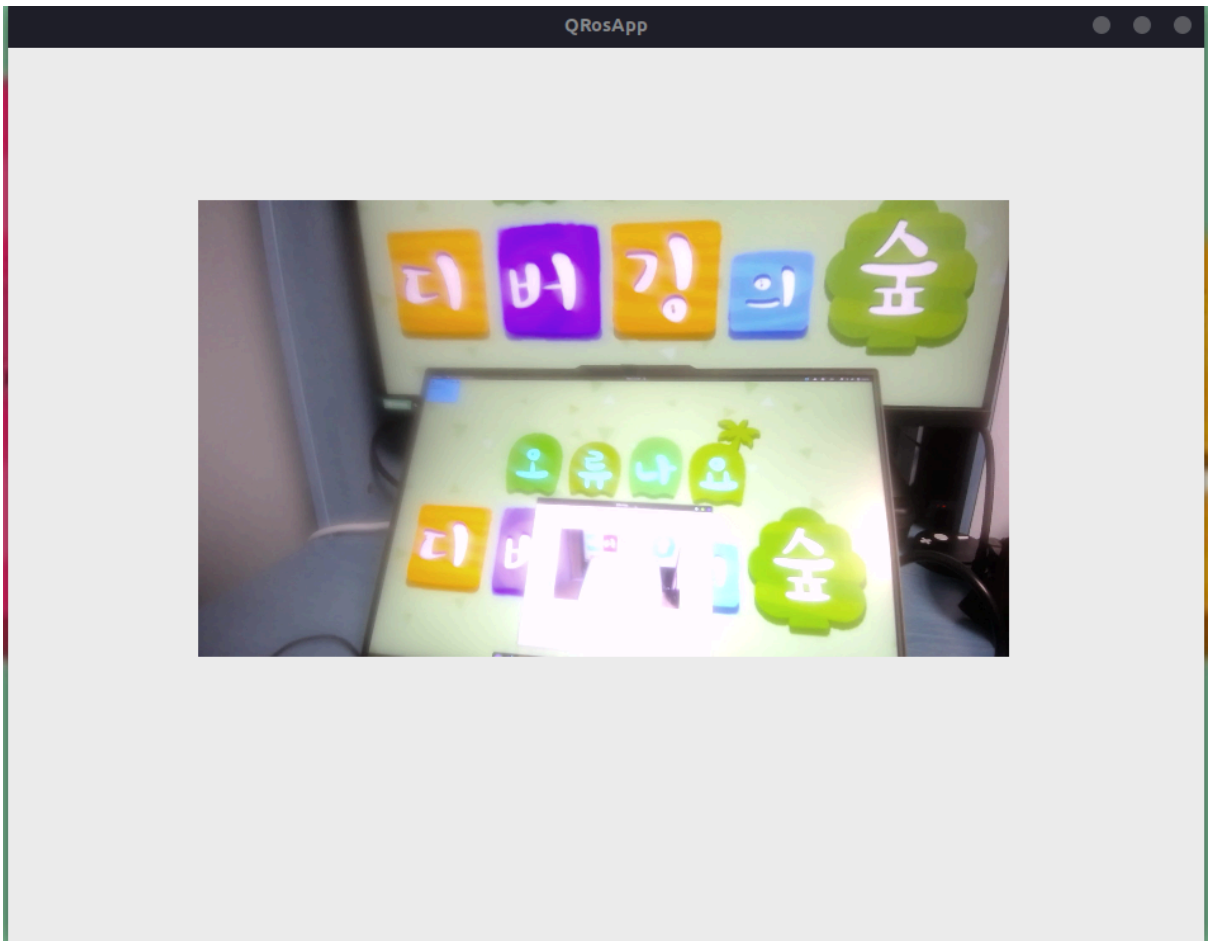
찾은 것일 수도 있다. 그래서 어떻게 할까 고민하던 중, usb_camera 패키지는 USB Camera로부터 이미지를 입력 받아 Qt에 출력하는 패키지라는 것이 떠올랐다. 그래서 usb_camera 패키지를 github에서 코드를 보면서 어떤 토픽을 사용하면 되는지 알 수 있었다.
"usb_camera_node.cpp"의 361 라인에 topic이 나와 있었다. 그래서 이 토픽을 가지고 개발하면 될 것이라 생각하고 개발을 진행했다.

화면 비율 유지 같은 경우에는 stack overflow에 비슷한 레퍼런스가 있어서 참고했다.

7. 결과



<qt_usb_camera_viewer 실행 결과 1>



<qt_usb_camera_viewer 실행 결과 2>

8. 결론 및 향후 발전 방향

이번 프로젝트에서는 ROS2 환경에서 Qt GUI를 활용하여 qt_usb_camera_viewer 패키지를 개발하였다.

9. 참고 문헌

- 표윤석. (2020.08.18.). "000 로봇 운영체제 ROS 강좌 목차". https://cafe.naver.com/openrt/24070
- 표윤석. (2024.09.05.). "로봇 운영체제 ROS 2 특강 발표 자료", https://docs.google.com/presentation/d/1QvUHj8iZCWIGiWRCgHFG-DLAmoUK6khO9xBXCzjPTTs/edit
- i_robo_u. (n.d.). "인공지능 로봇 개발". Velog. https://velog.io/@i_robo_u/series/%EC%9D%B8%EA%B3%B5%EC%A7%80%EB%8A%A5%EB%A1%9C%EB%B4%87%EA%B0%9C%EB%B0%9C
- mjlee111. usb_camera_node.cpp Github Repository, https://github.com/mjlee111/image_recognition/blob/master/image_projection/usb_camera/src/usb_camera_node.cpp.
- "QT resizing a QImage with scaled." Stack Overflow, https://stackoverflow.com/questions/9578578/qt-resizing-a-qimage-with-scaled.

10. 부록

10.1 레파지토리 링크

ROBIT_ROS2_HW: https://github.com/kinesis19/ROBIT_ROS2_HW

10.2 과제 Wiki

ROS2_HW_DAY_3: https://github.com/kinesis19/ROBIT_ROS2_HW/wiki/ROS2_HW_DAY3

10.3 패키지 구조

패키지의 구조는 다음과 같다:

```
qt_usb_camera_viewer
├── CMakeLists.txt
├── include
│   └── qt_usb_camera_viewer
│       ├── main_window.hpp
│       └── qnode.hpp
├── package.xml
├── resources
│   ├── images
│   │   └── icon.png
│   └── images.qrc
├── src
│   ├── main.cpp
│   ├── main_window.cpp
│   └── qnode.cpp
└── ui
    └── mainwindow.ui
```