<RO:BIT ROS2 - 2일차 과제2 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 사용하여 talker와 listener를 Qt 프로그램으로 구현한 과정과 결과를 설명한다. qt_talker_and_listener 패키지는 ROS2 환경에서 Qt GUI를 사용하여 메시지 송수신 기능을 구현한 패키지이다.

본 패키지는 RO:BIT의 ROS2 - 2일차 과제2에 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

이 프로젝트의 목적은 ROS2 환경에서 Qt GUI를 통해 메시지를 송수신하는 시스템을 구축하는 것이다. 즉, ROS2의 기본 패키지인 talker와 listener를 클론하여 구현한 chatter_cli 패키지를 Qt상으로도 구현하는 것이 이번 과제의 핵심 목적이다.

3. 관련 이론 및 개념

3.1 이론적 배경

이 프로젝트는 ROS2의 퍼블리셔-서브스크라이버 모델을 기반으로 하여 ROS2와 상호작용이 가능한 GUI 프로그램을 개발하는 것이다. ROS2(Robot Operating System 2)는 로봇 소프트웨어 개발을 지원하는 프레임워크로, 노드(Node), 토픽(Topic), 메시지(Message), 퍼블리셔(Publisher), 서브스크라이버(Subscriber)와 같은 개념을 통해 노드 간의 비동기식 데이터 교환을 제공한다. 이를 통해 분산 시스템 내에서 모듈 간 데이터를 효율적으로 송수신할 수 있다.

3.2 관련 개념

- **노드(Node)**: ROS2에서 독립적으로 실행되는 프로세스이다.
- **토픽(Topic)**: 토픽은 노드 간 비동기적으로 데이터를 주고받기 위한 채널이다.
- **서비스(Service)**: 서비스는 동기식 통신 메커니즘으로, 클라이언트가 서버에 요청을 보내고 응답을 받는 형태로 동작한다.
- **ROS2 메시지 타입**: ROS2에서는 노드 간 데이터를 주고받기 위해 다양한 메시지 타입을 제공한다.

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

std_msgs: ROS2의 표준 메시지 타입 패키지

turtlesim: ROS2에서 제공하는 시뮬레이션 패키지

Qt5: GUI 구축을 위한 라이브러리

5. 패키지 분석

이 프로젝트에서 개발한 qt_talker_and_listener 패키지는 Qt GUI를 통해 Talker와 Listener 노드를 구현하기 위해 설계 되었다. 이 패키지는 ROS2의 퍼블리셔-서브스크라이버 모델과 서비스 구조를 활용했으며, 사용자의 입력을 받아 내용물과 메시지 카운트를 퍼블리시하고, 서브스크라이브 받는 것을 목표로 개발했다.

qt_talker_and_listener 패키지의 주요 구성 요소는 다음과 같다.

GUI

- **QPushButton**: 입력된 내용을 퍼블리시한다.
- **QLabel**: 서브스크라이브 받은 내용과 메시지 카운트를 출력한다.

주요 기능

- **메시지 퍼블리시 및 서브스크라이브**: ROS2의 퍼블리셔-서브스크라이버 모델을 활용해서 사용자의 입력을 메시지로 송신하고, GUI에 수신하여 표시한다.

이 패키지는 Qt GUI와 ROS2 간의 원활한 상호작용을 위해 로빛 17기 주장님이신 이명진 선배님의 "**ros2_create_qt_pkg**" 스크립트를 사용하여 환경을 구성했다.

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. ROS2 노드간의 통신을 위해 MainWindow, QNode, Talker, Listener 클래스로 구성했다. MainWindow, QNode는 스크립트를 통해 자동으로 구성 되었으며, 추가한 클래스는 Talker와 Listener이다.

이번 패키지의 핵심 기능은 ROS2 1일차 과제3에서 구현한 chatter_cli를 UI로 표시하는 것이다. 입력부터 출력까지 UI에서 진행하고, 백엔드 쪽은 ROS2의 퍼블리셔-서브스크라이버로 작업하는 것으로 이해했다.

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();

  // 버튼 클릭 시 메시지 퍼블리시
  connect(ui->btn_Publish, &QPushButton::clicked, this,
&MainWindow::onPublishButtonClicked);

  // Listener의 새로운 메시지를 라벨에 업데이트
```

```
  connect(qnode, &QNode::newMessageReceived, this,
&MainWindow::updateLabel);

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

connect()를 사용하여 버튼이 클릭 되었을 때, onPublishButtonClicked() 메서드가 실행되도록 하였다. 또, 수신 받은 메시지가 있으면 updateLabel() 메서드가 실행되도록 구성하였다.

connect(발신자, SIGNAL(신호), 수신자, SLOT(슬롯));

발신자: 이벤트를 발생시키는 객체

신호: 발신자가 발생키는 이벤트

수신자: 이벤트 처리 객체

슬롯: 이벤트가 발생되었을 때 처리되는 함수

6.2.1.2. onPublishButtonClicked() 메서드

```
// Publish 버튼 눌렀을 때 퍼블리쉬하는 메서드
void MainWindow::onPublishButtonClicked()
{
    // 입력된 텍스트를 QNode를 통해 퍼블리쉬
    QString text = ui->lineEdit_Text->text();
    qnode->publishMessage(text.toStdString());
}
```

onPublishButtonClicked() 메서드는 Qt UI 상의 Publish 버튼이 클릭되면 실행되는 슬롯 함수이다. UI 상에 있는 텍스트를 text라는 변수에 저장하고, QNode의 publishMessage 메서드를 호출해서 퍼블리시한다.

6.2.1.3. updateLabel 메서드

```
// label 업데이트 메서드
void MainWindow::updateLabel(const QString &message, int count)
```

```
{
  ui->labelSubscribe->setText(QString("\"%1\",
\"%2\"").arg(message).arg(count)); // 메시지와 카운트를 동시에 출력함
}
```

updateLabel() 메서드는 Listener 노드가 새 메시지를 수신했을 때 호출되는 슬롯 함수이다. 수신 받은 메시지와 카운트를 동시에 출력한다. 형식은 "{메시지 내용}", "{카운트}" 이다.

6.2.2. QNode

QNode 클래스는 ROS2와 Qt GUI 간의 중계하는 클래스이고, MainWindow와 Talker, Listener 노드를 연결해서 데이터의 송수신을 관리한다.

6.2.2.1. QNode() 생성자

QNode 클래스는 ROS2와 Qt GUI 간의 중계하는 클래스이고, MainWindow와 Talker, Listener 노드를 연결해서 데이터의 송수신을 관리한다.

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

  // Talker와 Listener 초기화
  talker = std::make_shared<Talker>();
  listener = std::make_shared<Listener>();

  // Listener에 메시지 처리 콜백 설정

listener->setChatterCallback(std::bind(&QNode::onChatterReceived,
this, std::placeholders::_1));
```

```
  this->start();
}
```

QNode() 생성자에서 talker와 listener를 각각 Talker와 Listener의 클래스 객체로 초기화한다.

listener 객체에 setChatterCallback() 메서드를 사용해서 메시지를 처리하는 콜백 함수를 설정했다. 이 콜백은 Listener가 새로운 메시지를 수신했을 때 실행 되고, QNode::onChatterReceived() 메서드를 호출한다.

6.2.2.2. publishMessage() 메서드

```
void QNode::publishMessage(const std::string &message)
{
  talker->publishMessage(message);
}
```

publishMessage() 메서드는 UI에 입력된 텍스트를 Talker 노드를 통해 퍼블리시 하는 메서드이다. const로 파라미터를 받아 값 변경을 방지한다. std::string을 참조로 전달하면 문자열 데이터를 복사하지 않고, 메모리 주소만 전달한다는 것을 배울 수 있었다.

6.2.2.3. publishMessage() 메서드

```
void QNode::onChatterReceived(const std::string &message)
{
    int message_count = talker->getMessageCount(); // Talker의
message_count_ 값 가져오기
    emit newMessageReceived(QString::fromStdString(message),
message_count); // 메시지랑 카운트 보내기
}
```

onChatterReceived() 메서드는 Listener 노드가 수신한 메시지와 메시지 카운트를 GUI에 전달하는 콜백 함수이다. Talker의 getMessageCount()를 호출해서 값을 변수에 저장하고, newMessageReceived() 시그널을 통해 수신 데이터를 표시한다.

*emit: 시그널을 호출하고, 연결된 슬롯 함수로 데이터를 전달하는 역할을 한다.

6.2.3. Listener 노드

Listener 클래스는 수신 노드로, Talker가 퍼블리시한 메시지와 메시지 카운트를 서브스크라이브하고 GUI에 표시한다.

6.2.3.1. Listener() 생성자

```
Listener::Listener() : Node("listener")
{
    // 문자열 메시지 서브스크라이버 초기화
    subscription_ =
this->create_subscription<std_msgs::msg::String>("/chatter", 10,
std::bind(&Listener::chatterCallback, this, std::placeholders::_1));

    // 카운트 메시지 서브스크라이버 초기화
    count_subscription_ =
this->create_subscription<std_msgs::msg::Int64>("/chatter_count", 10,
std::bind(&Listener::countCallback, this, std::placeholders::_1));

    RCLCPP_INFO(this->get_logger(), "Listener Node 초기화 완료");
}
```

Listener() 클래스는 ROS2의 서브스크라이버를 초기화 하고, /chatter 토픽과 /chatter_count 토픽을 각각 서브스크라이브한다.

create_subscription() 메서드를 통해 각각의 토픽에 대해 콜백 함수를 등록한다. /chatter 토픽에서 메시지를 수신하면, chatterCallback 메서드가 실행되고, /chatter_count 토픽에서 카운트를 수신하면 countCallback 함수가 실행되도록 설계했다.

6.2.3.2. setChatterCallback() 메서드

```
void Listener::setChatterCallback(std::function<void(const std::string
&)> callback)
{
    chatter_callback_ = callback;
}
```

setChatterCallback() 메서드는 외부에서 전달된 콜백 함수를 등록하고, 수신된 chatter 메시지가 있을 때 콜백을 호출한다.

*콜백 함수: 콜백 함수는 이벤트나 작업 완료 시 자동으로 호출되는 함수이다. (이벤트 발생 시 특정 작업을 수행하도록 미리 등록해 두는 함수)

6.2.3.3. chatterCallback() 메서드

```
// 서브스크라이브 받아 메시지를 콜백으로 전달함
void Listener::chatterCallback(const std_msgs::msg::String::SharedPtr
msg)
{
    chatter_callback_(msg->data);
}
```

chatterCallback() 메서드는 /chatter 토픽에서 문자열을 수신했을 때 실행되는 콜백 함수이다. QNode로 전달하고, GUI에 표시하게 된다.

6.2.3.4. countCallback() 메서드

```
// 서브스크라이브 받아 퍼블리시된 횟수를 출력하는 메서드
void Listener::countCallback(const std_msgs::msg::Int64::SharedPtr msg)
{
    RCLCPP_INFO(this->get_logger(), "퍼블리시된 횟수: %ld", msg->data);
}
```

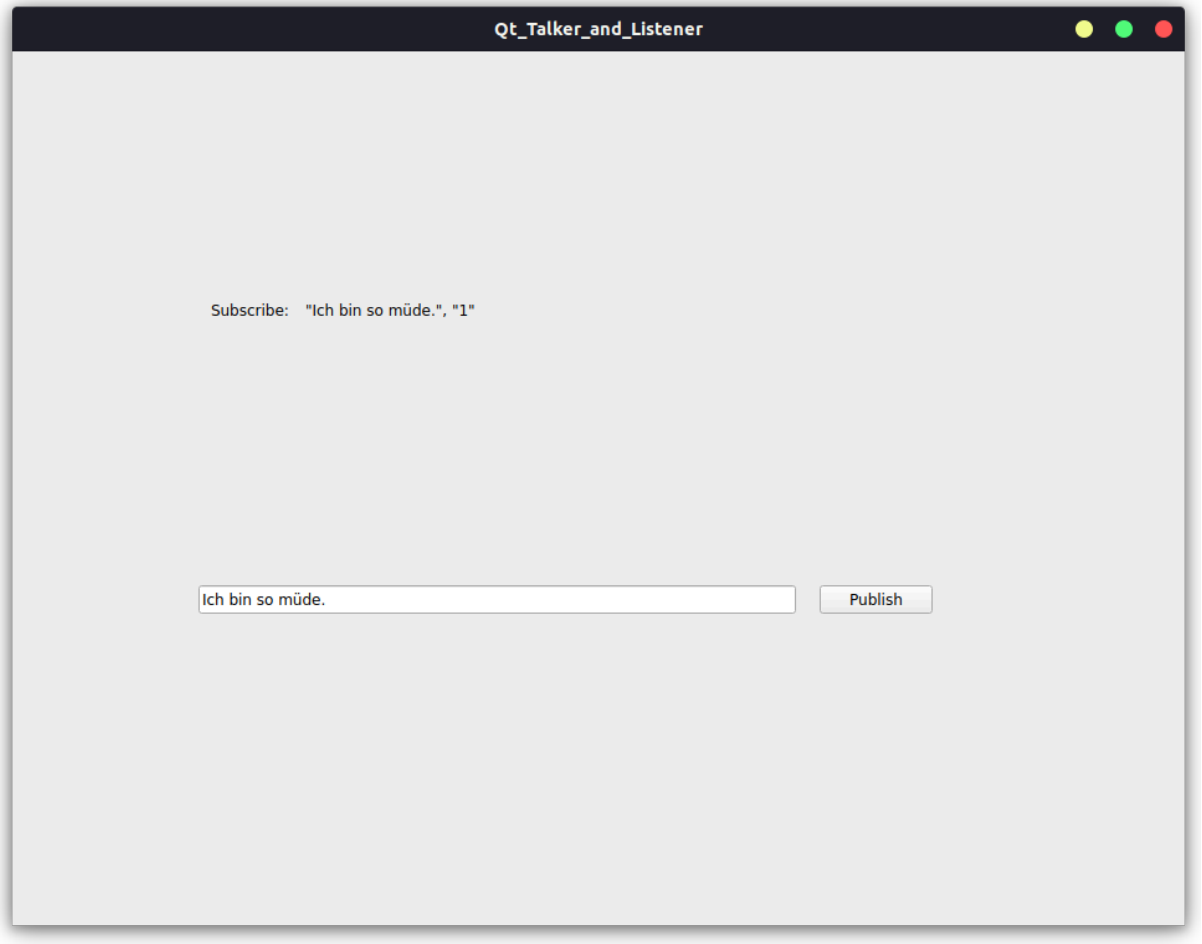
countCallback() 메서드는 /chatter_count 토픽에서 메시지 카운트를 수신했을 때 실행되는 콜백 함수이다. 디버깅용으로 콘솔에 출력하게 구현했다.

6.3.실패 사례와 시도한 방법

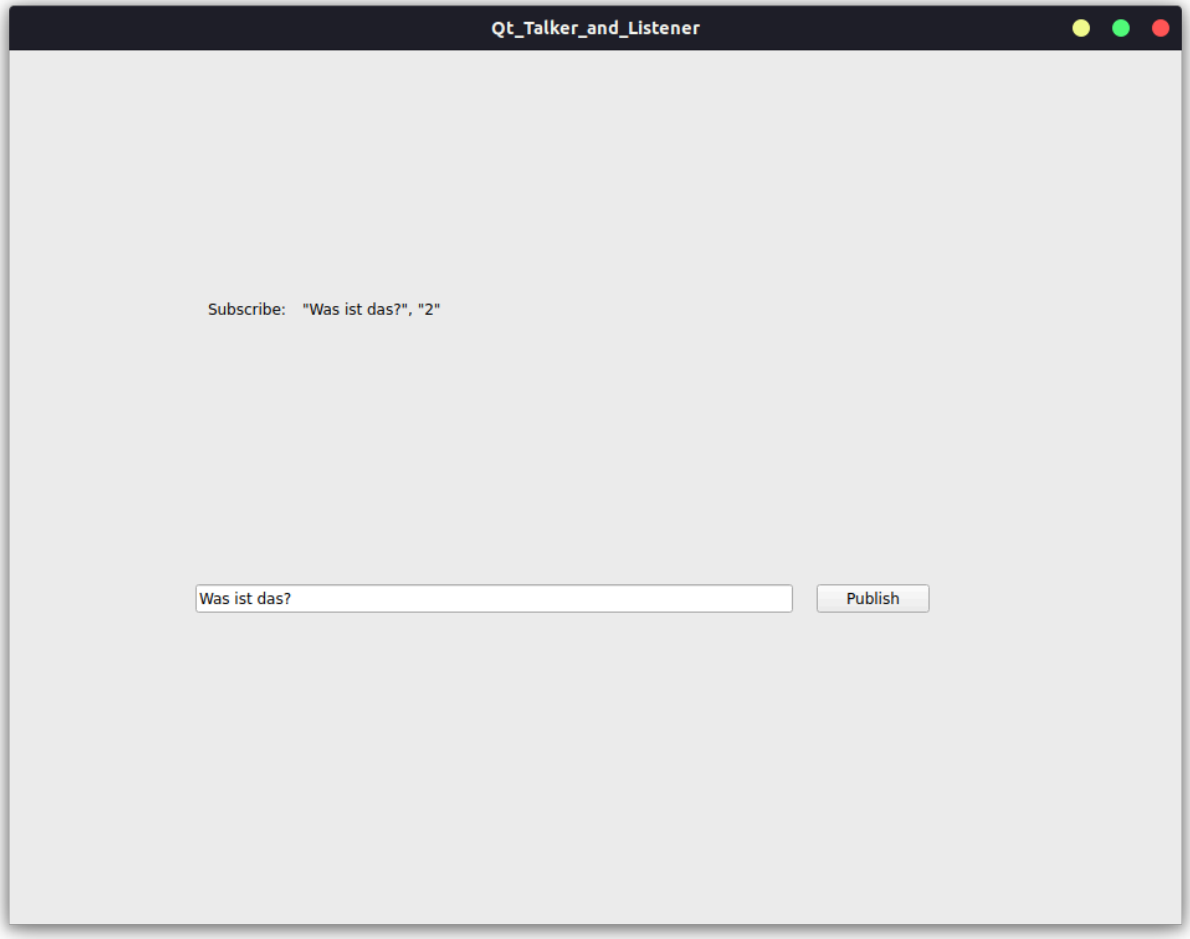
처음에는 퍼블리셔 UI 하나랑 서브스크라이버 UI 하나 이렇게 총 두 개의 UI를 구성해야 하는 줄 알았다. 그러나, 노드만 두 개로 처리하고, 하나의 UI에서 관리하면 된다는 것을 알게 되어 개발을 진행했다.

개발을 진행하면서 콜백 함수에 대한 개념을 다시 보게 되었으며, 더 공부하게 되었다. 퍼블리셔와 서브스크라이버에 대한 개념, ROS2의 기본 개념에 대해 슬슬 감이 잡히기 시작했다. 머릿속에 엉켜 있지만, 어떨 때 사용하고 그 개념에 대해 이제는 따라가고 있다고 생각한다. 그래도, 정리할 시간이 필요하다.

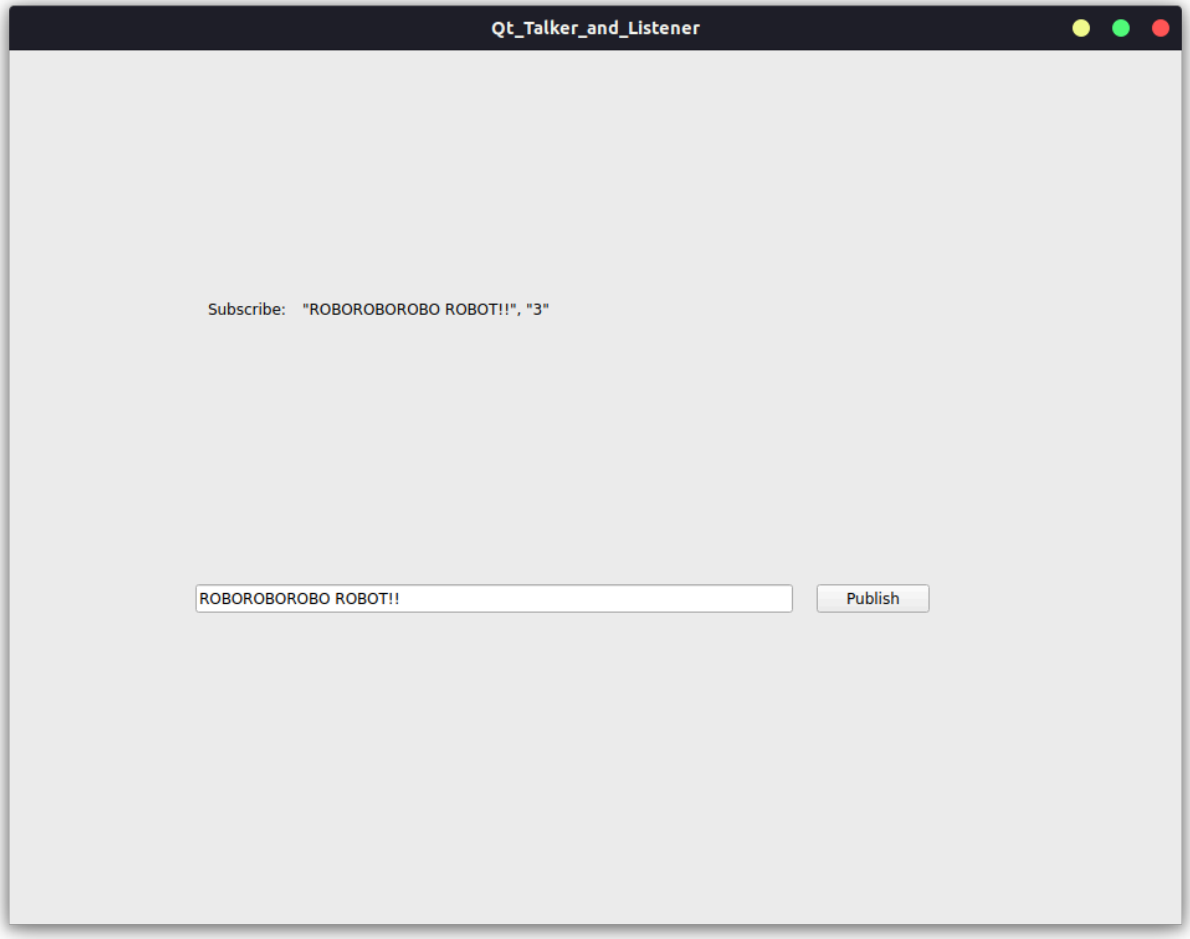
7. 결과



<qt_turtlesim_controller 실행 결과 1>



<qt_turtlesim_controller 실행 결과 2>



<qt_turtlesim_controller 실행 결과 3>

8. 결론 및 향후 발전 방향

이번 프로젝트에서는 ROS2 환경에서 Qt GUI를 활용하여 qt_talker_and_listener 패키지를 개발하였다. 이 패키지는 1일차 과제3의 내용의 연장선이며, CLI로 구현한 것을 GUI로 구현하는 것에 초점을 두었다.

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

10. 부록

10.1 레파지토리 링크

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

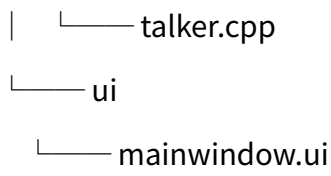
10.2 과제 Wiki

ROS2_HW_DAY_2: https://github.com/kinesis19/ROBIT_ROS2_HW/wiki/ROS2_HW_DAY2

10.3 패키지 구조

패키지의 구조는 다음과 같다:

```
qt_talker_and_listener
├── CMakeLists.txt
├── include
│   └── qt_talker_and_listener
│       ├── listener.hpp
│       ├── main_window.hpp
│       ├── qnode.hpp
│       └── talker.hpp
├── package.xml
├── resources
│   ├── images
│   │   └── icon.png
│   └── images.qrc
├── src
│   ├── listener.cpp
│   ├── main.cpp
│   ├── main_window.cpp
│   ├── qnode.cpp
```


talker.cpp
ui
mainwindow.ui