ROS2_day2 hw2 ROS2 && Qt
과제 보고서

로빛 20기 인턴 현창석

목차

Code Blame Raw

```
#ifndef ROS2_CREATE_QT_PKG_MAIN_WINDOW_HPP_
#define ROS2_CREATE_QT_PKG_MAIN_WINDOW_HPP_

#include <QMainWindow>
#include "qnode.hpp"
#include "ui_mainwindow.h"
#include <vector>
#include <string>


class MainWindow : public QMainWindow
{
  Q_OBJECT

public:
  MainWindow(std::shared_ptr<QNode> qnode, QWidget* parent = nullptr);
  ~MainWindow();


private slots:
  void on_pushButton_clicked();

private:
  Ui::MainWindowDesign* ui;
  std::shared_ptr<QNode> qnode_;
    QTextEdit *log;
  QVector< QString > queue_v;
};


```

```
5
6   #ifndef ROS2_CREATE_QT_PKG_QNODE_HPP_
7   #define ROS2_CREATE_QT_PKG_QNODE_HPP_
8
9   /*****************************************************************************
10  ** Includes
11  *****************************************************************************/
12  #ifndef Q_MOC_RUN
13  #include <rclcpp/rclcpp.hpp>
14  #include "std_msgs/msg/string.hpp"
15  #include <QThread>
16  #include <string>
17  #endif // Q_MOC_RUN
18
19  class MyNode : public rclcpp::Node
20  {
21  public:
22      MyNode();
23
24
25  private:
26      rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscriber;
27
28
29      void topic_callback(const std_msgs::msg::String::SharedPtr msg);
30
31
32  };
33
34
35
36  class QNode : public rclcpp::Node
37  {
38  public:
39    explicit QNode(const std::string& name);
40    void publishString(const std::string &text);
```

```
40      void publishString(const std::string &text);
41      rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
42    protected:
43      void run();
44
45    private:
46
47
48      std::shared_ptr<rclcpp::Node> node;
49
50
51
52      void topic_callback(const std_msgs::msg::String::SharedPtr msg);
53
54
55    Q_SIGNALS:
56      void rosShutDown();
57      void newMessageReceived(QString msg);
58    };
59
60
61    #endif // ROS2_CREATE_QT_PKG_QNODE_HPP_
```

```cpp
#include <QApplication>
#include <thread>
#include "../include/ros2_create_qt_pkg/main_window.hpp"
#include "../include/ros2_create_qt_pkg/qnode.hpp"

void spin_ros_node(std::shared_ptr<QNode> node)
{
  rclcpp::spin(node);
}

int main(int argc, char* argv[])
{
  rclcpp::init(argc, argv);
  auto qnode = std::make_shared<QNode>("ros2_create_qt_pkg_ui");

  QApplication a(argc, argv);
  MainWindow w(qnode);
  w.show();

  std::thread ros_thread(spin_ros_node, qnode);

  int result = a.exec();

  rclcpp::shutdown();
  if (ros_thread.joinable()) {
      ros_thread.join();
  }

  return result;
}
```

```cpp
#include "../include/ros2_create_qt_pkg/main_window.hpp"
#include "rclcpp/rclcpp.hpp"

MainWindow::MainWindow(std::shared_ptr<QNode> qnode, QWidget* parent)
  : QMainWindow(parent)
  , ui(new Ui::MainWindowDesign)
  , qnode_(qnode)
{
  ui->setupUi(this);
  log = ui->log;

}

MainWindow::~MainWindow()
{
  delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    static QVector<QString> queue_v;
    queue_v.prepend(ui->lineEdit->text());

    if (!queue_v.isEmpty()) {
        QString first_qstr = queue_v[0];

        std::string tostring = first_qstr.toStdString();

        std_msgs::msg::String msg;
        msg.data = tostring;

        qnode_->publisher_->publish(msg);

        ui->lineEdit->clear();
        ui->log->append(QString("[Publish] ") + QString::fromStdString(tostring));
    }
}
```

```cpp
#include "rclcpp/rclcpp.hpp"

#include <chrono>
#include <memory>
#include <functional>
#include "../include/ros2_create_qt_pkg/qnode.hpp"
#include "../include/ros2_create_qt_pkg/main_window.hpp"




MyNode::MyNode() : Node("mynode")
{

    subscriber = this->create_subscription<std_msgs::msg::String>("topicname", 10, std::bind(&MyNo


}
void MyNode::topic_callback(const std_msgs::msg::String::SharedPtr msg) {

    RCLCPP_INFO(this->get_logger(), "Subscribed to '%s'", msg ->data.c_str());

}




int main(int argc, char ** argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<MyNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

qnode.cpp에서

MyNode1() 생성자에서는 topicname이라는 이름의 토픽으로 메세지를 퍼블리시하는 퍼블리셔를 선언해준다. 이 때10은 버퍼사이즈에 해당한다.
publishString 함수에서 사용자가 입력한 문자열을 퍼블리시한다.

subs.cpp에서

생성자에서는 mynode로 먼저 통신을 주고 받을 때의 이름을 설정하였다. topicnamed l 라는 토팍에서 string형태의 메세지를 서브스크라이브 하는 서브스크라이버를 선언하였다. 최근에 섭스크라이브한 메세지를 최대 10개까지 저장 가능하다 . bind를 이용해 이를 topic_callback 함수와 연결해주었다.
topic_callbackg함수에서는 사용자가 입력한 문자열을 로그에 출력한다.

mainwindow cpp에서

ui에 입력 버튼 클릭하면 vector를 정적 변수로 설정하고 사용자가 QLinedit에 입력한 문자열을 가져와 prepend를 이용해 queue_v의 맨 앞에 추가한다.
그 후 벡터가 비어있지 않다면 첫 번째 원소를 가져와 string 타입으로 변경 후 퍼블리시 한다. 이 퍼블리시 한 문자열을 log 위젯에도 apppend하여 출력한다.