ROS2_day2 hw1 ROS2 && Qt
과제 보고서

로빛 20기 인턴 현창석

목차

- 코드 설명

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
#include <string>
#include <vector>
#include <thread>

class MyNode : public rclcpp::Node
{
public:
    MyNode();

private:
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscriber;
    void topic_callback(const std_msgs::msg::String::SharedPtr msg);
};

class MyNode1 : public rclcpp::Node
{
public:
    MyNode1();
    ~MyNode1();

private:
    void input_thread_func();
    rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
    std::thread input_thread_;
    std::string sendstring;
    std::vector<std::string> queue_v;
};
```

ROS2메시지 패키지 std_msgs 안의 String 메시지를 사용해 퍼블리시 , 서브스크라이버 하기 위해서 std_msgs/msg/string.hpp 를 인클루드해주어꼬 키 입력으로 문장을 사용자가 입력하고 벡터 형태로 문자열을 저장하여 퍼블리시 하기 위해서 벡터와 thread , string을 인클루드 해주었다.
클래스 mynode안에서 생성자, 섭스크라이버 를 선언해주었다. topic_callback 함수는 메세지를 서즈스크라이브 할 때 실행되는 함수이다 .

mynode1이라는 이름의 클래스를 보면 먼저 문장을 입력받아 벡터에 저장하기 위한 input_thread_func() 함수를 선언해주었고
std::thread input_thread_는 입력을 처리하기 위한 스레드 객체이다. nput_thread_를 쓰면,메인 루프는 계속 돌아가고, 입력은

이 스레드에서 관리한다 다.벡터에 저장된 문자열을 senstring에 담아 퍼블리시할 것이므로 sendstring을 정의해주었다.

```cpp
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
#include <chrono>
#include <memory>
#include <functional>
#include "std_msgs/msg/float32.hpp"
#include "my_node.hpp"

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

  subscriber = this->create_subscription<std_msgs::msg::String>("topicname",
      10, std::bind(&MyNode::topic_callback, this, std::placeholders::_1));

}

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

    RCLCPP_INFO(this->get_logger(), "Subscribed msg : '%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;
}
```

shared_ptr 같은 스마트 포인터 사용을 위해 memory인클루드, bind , placeholders 사용을 위해서 functional 인클루드 , 선언한 클래스에서 노드를 불러오기 위해 작성한 헤더파일을 불러왔다. 생성자에서는 mynode로 먼저 통신을 주고 받을 때의 이름을 설정하였다. topicnamed l 라는 토팍에서 string형태의 메세지를 서브스크라이브 하는 서브스크라이버를 선언하였다. 최근에 섭스크

라이브한 메세지를 최대 10개까지 저장 가능하다 . bind를 이용해 이를 topic_callback 함수와 연결해주었다. placeholders::_1 → 콜백 함수에 인자로 전달되는 메시지를 첫 번째 인자로 넘겨주겠다는 의미이다. 콜백함수에서는 RCLCPP_INFO 그리고 data.c_str()을 이용해 서브스크라이브한 메세지를 const char*로 변환하여 로그에 출력하였다.

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

#include <memory>
#include <functional>
#include <iostream>
#include <string>
#include <thread>

#include "my_node.hpp"

MyNode1::MyNode1() : Node("mynode1")
{
    publisher_ = this->create_publisher<std_msgs::msg::String>("topicname", 10);
    input_thread_ = std::thread(&MyNode1::input_thread_func, this);
    input_thread_.detach();
}

MyNode1::~MyNode1()
{

    if (input_thread_.joinable()) {
        input_thread_.join();
    }
}

void MyNode1::input_thread_func()
{
    while (1) {
        std::cout << "전송할 문장을 입력하세요: " << std::endl;
        std::getline(std::cin, sendstring);

        if (!sendstring.empty()) {
            queue_v.push_back(sendstring);

            auto msg = std_msgs::msg::String();
```

```
31            std::getline(std::cin, sendstring);
32
33            if (!sendstring.empty()) {
34                queue_v.push_back(sendstring);
35
36                auto msg = std_msgs::msg::String();
37                msg.data = sendstring;
38                publisher_->publish(msg);
39                RCLCPP_INFO(this->get_logger(), "Published message: '%s'", msg.data.c_str());
40            }
41        }
42    }
43
44    int main(int argc, char ** argv)
45    {
46        rclcpp::init(argc, argv);
47        auto node = std::make_shared<MyNode1>();
48        rclcpp::spin(node);
49        rclcpp::shutdown();
50        return 0;
51    }
```

MyNode1() 생성자에서는 topicname이라는 이름의 토픽으로 메세지를 퍼블리시하는 퍼블리셔를 선언해준다. 이 때10은 버퍼사이즈에 해당한다. 문자열을 입력받아 벡터에 저장하기 위한 nput_thread_func 함수를 스레드로 독립실행시키기 위해 input_thread로 스레드를 선언하고 detach(분리) 하였다.
input_thread_func 에서는 무한 반복문을 돌며 한 줄 입력을 블로킹읽어 sendstring에 사용자가 입력한 문장을 저장한다, 만약 문자열이 비어있지 않다면 std_msgs::msg::String()로 새로운 메세지를 만들고 이를 퍼블리시한다. (auto는 자동으로 타입을 정해줌)퍼블리시된 메세지는 RCLCPP_INFO로 로그에 출력된다