ROS2_day2 hw2 ROS2 && Qt
과제 보고서

로빛 20기 인턴 현창석

목차

```cpp
#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(QNode* qnode, QWidget* parent = nullptr);
  ~MainWindow();


public slots:
  void updateJointAngles(const std::vector<double>& positions, const std::vector<std::string>& name
  void updateJointAngle(const QString& name, int angle);


protected:
  void paintEvent(QPaintEvent* event) override;

private:
  Ui::MainWindowDesign* ui;
  QNode* qnode_;

  int angle1 = 0;
  int angle2 = 0;
  int angle3 = 0;
};

#endif  // ROS2_CREATE_QT_PKG_MAIN_WINDOW_HPP_
```

```
#ifndef Q_MOC_RUN
#include <rclcpp/rclcpp.hpp>
#include "std_msgs/msg/string.hpp"
#include "sensor_msgs/msg/joint_state.hpp"
#include <QThread>
#include <QString>
#include <string>
#include <vector>
#endif // Q_MOC_RUN

class QNode : public QThread
{
  Q_OBJECT
public:
  QNode(int argc, char** argv);
  virtual ~QNode();
  bool init();
  void run();

private:
  rclcpp::Subscription<sensor_msgs::msg::JointState>::SharedPtr joint_state_subscriber;
  rclcpp::Subscription<std_msgs::msg::String>::SharedPtr command_subscriber_;

  int init_argc;
  char** init_argv;
  std::shared_ptr<rclcpp::Node> node_;

signals:
  void rosShutDown();
  void jointStateReceived(const std::vector<double>&, const std::vector<std::string>&);
  void commandReceived(const QString&, int);
};

#endif // ROS2_CREATE_QT_PKG_QNODE_HPP_
```

```cpp
#include <QApplication>
#include "../include/hw4/main_window.hpp"
#include "../include/hw4/qnode.hpp"

int main(int argc, char* argv[])
{
  QApplication a(argc, argv);

  QNode qnode(argc, argv);
  qnode.init();

  MainWindow w(&qnode);
  w.show();

  int result = a.exec();

  return result;
}
```

```cpp
#include "../include/hw4/main_window.hpp"
#include <QPainter>
#include <cmath> // For M_PI

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


   connect(qnode_, &QNode::jointStateReceived,
           this, &MainWindow::updateJointAngles);
   connect(qnode_, &QNode::commandReceived,
           this, &MainWindow::updateJointAngle);
}

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

void MainWindow::updateJointAngles(const std::vector<double>& positions,
const std::vector<std::string>& names)
{

    for (size_t i = 0; i < names.size(); ++i)
    {

        int degree = static_cast<int>(positions[i] * 180.0 / M_PI);

        if (names[i] == "joint1")
        {
            angle1 = qBound(0, degree, 360);
        }
        else if (names[i] == "joint2")
        {
            angle2 = qBound(0, degree, 360);
```

```
38              {
39                  angle2 = qBound(0, degree, 360);
40              }
41              else if (names[i] == "joint3")
42              {
43                  angle3 = qBound(0, degree, 360);
44              }
45          }
46          update();
47      }
48
49      void MainWindow::paintEvent(QPaintEvent* event)
50      {
51          Q_UNUSED(event);
52          QPainter painter(this);
53          painter.setRenderHint(QPainter::Antialiasing);
54
55          painter.save();
56          painter.translate(300, 300);  //팔1 시작점
57          painter.rotate(angle1);
58          painter.drawLine(0, 0, 100, 0);  // 팔 1
59
60          painter.translate(100, 0);  // 팔 1끝으로 이동
61          painter.rotate(angle2);
62          painter.drawLine(0, 0, 80, 0);   // 팔 2
63
64          painter.translate(80, 0);  // 팔 2끝으로 이동
65          painter.rotate(angle3);
66          painter.drawLine(0, 0, 60, 0);   // 팔 3
67
68          painter.restore();
69      }
70
71      void MainWindow::updateJointAngle(const QString& name, int angle)
72      {
73          if (name == "joint1")
```

```
71  void MainWindow::updateJointAngle(const QString& name, int angle)
72  {
73      if (name == "joint1")
74      {
75          angle1 = qBound(0, angle, 360);
76      }
77      else if (name == "joint2")
78      {
79          angle2 = qBound(0, angle, 360);
80      }
81      else if (name == "joint3")
82      {
83          angle3 = qBound(0, angle, 360);
84      }
85      update();
```

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

#include "../include/hw4/qnode.hpp"

QNode::QNode(const std::string& name) : Node(name)
{
    publisher_ = this->create_publisher<std_msgs::msg::String>("/joint_states", 10);
}

void QNode::publishString(const std::string &text)
{
    std::string message = std_msgs::msg::String();
    message.data = text;
    publisher_->publish(message);
}


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

```cpp
#include "../include/hw4/qnode.hpp"
#include <QDebug>
#include <QStringList>

QNode::QNode(int argc, char** argv) :
  init_argc(argc),
  init_argv(argv)
{}

QNode::~QNode() {
    if(rclcpp::ok()) {
        rclcpp::shutdown();
    }
    wait();
}

bool QNode::init() {
    rclcpp::init(init_argc, init_argv);
    node_ = std::make_shared<rclcpp::Node>("hw4_node");


    auto joint_state_callback =
      [this](const sensor_msgs::msg::JointState::SharedPtr msg) -> void
      {
        if (msg->position.size() > 0 && msg->name.size() > 0)
        {
            Q_EMIT jointStateReceived(msg->position, msg->name);
        }
      };

    auto subscription_options = rclcpp::SubscriptionOptions();
    rclcpp::QoS qos(rclcpp::KeepLast(10));

    joint_state_subscriber = node_->create_subscription<sensor_msgs::msg::JointState>(
      "/joint_states", qos, joint_state_callback, subscription_options);
```

Code Blame Raw

```cpp
#include "../include/hw4/qnode.hpp"
#include <QDebug>
#include <QStringList>

QNode::QNode(int argc, char** argv) :
  init_argc(argc),
  init_argv(argv)
{}

QNode::~QNode() {
    if(rclcpp::ok()) {
        rclcpp::shutdown();
    }
    wait();
}

bool QNode::init() {
    rclcpp::init(init_argc, init_argv);
    node_ = std::make_shared<rclcpp::Node>("hw4_node");

    auto command_callback =
      [this](const std_msgs::msg::String::SharedPtr msg) -> void
      {
        QString q_msg = QString::fromStdString(msg->data);
        QStringList parts = q_msg.split(" ");
        if (parts.length() == 2) {
            QString name = parts[0];
            bool ok;
            int angle = parts[1].toInt(&ok);
            if (ok) {
                Q_EMIT commandReceived(name, angle);
            }
        }
      };

    command_subscriber_ = node_->create_subscription<std_msgs::msg::String>(
      "/gui_command", qos, command_callback, subscription_options);
```

main_window.cpp에서는
QNode에서 commandReceived 가 발생하면 updateJointAngle 함수가 실행되어 각 로봇팔이 움직이도록 연결하였다. 메인화면에 로봇팔을 그리기 위해
ex (300,300) 각 기준 점 좌표로 이동한 후 angle 만큼 회전하고 끝 으로 이동 , 이 과정을 반복해 3축 로봇팔을 만들었다. updateJointAngle는 팔 이름 각도 를 퍼블리시 한 값을 섭스크라이브해서 각 팔의 각도를 제어할 수 있다.

Qnode.cpp에서는
로봇팔 명령을 cmd로 입력할 때 문자열을 공백으로 나누기 위해 StringList 헤더파일을 사용하였다 init() 함수에서 ROS2 노드 초기화 후 hw4_node라는 이름의 노드를 생성한다. joint_state_callback 은 /joint_states 라는 이름의 토픽을 섭스크라이브 할 함수이다. 인터넷을 찾아보았는데 이를 람다라고 부르는 듯 하다 . 데이터를 섭스크라이브하면 msg를 보내 jointStateReceived 를 실행시킨다.

그 아래 람다에 대해서도 설명하겠다. 사용자가 입력한 명령 문자열을 공백 기준으로 나누어 관절 이름과 각도 로 변환한다. 변환에 성공하면 commandReceived(name, angle); 함수를 실행시킨다.