hw3 보고서

20기 인턴 송수민

앞 선 과제에 추가한 부분을 설명하도록 하겠습니다.

1. arm_pkg 메인 윈도우 파일

```
rclcpp::init(0, nullptr);
node_ = std::make_shared<rclcpp::Node>("robot_arm_gui_node");
sub_ = node_->create_subscription<std_msgs::msg::Float64MultiArray>(
    "arm_angles", 10,
    [this](const std_msgs::msg::Float64MultiArray::SharedPtr msg)
    {
        if (msg->data.size() >= 3) {
            double b = msg->data[0];
            double s = msg->data[1];
            double e = msg->data[2];
            QMetaObject::invokeMethod(this, "setArmAngles",
                Qt::QueuedConnection,
                Q_ARG(double, b), Q_ARG(double, s), Q_ARG(double, e));
        }
    });

rosThread_ = std::thread([this]() {
    rclcpp::Rate rate(30);
    while (rclcpp::ok() && rosRunning_) {
        rclcpp::spin_some(node_);
        rate.sleep();
    }
});
```

먼저 rclcpp::init(0, nullptr);로 ROS2를 초기화한다.
그 다음 "robot_arm_gui_node"라는 이름으로 노드를 생성하고 node_에 저장한다.

create_subscription으로 "arm_angles" 토픽을 구독한다.
구독 콜백에서는 수신한 메시지의 data 벡터가 3개 이상의 값이 있는지 확인하고,
첫 세 값을 각각 b, s, e에 저장한다.
그 후 QMetaObject::invokeMethod를 사용해 GUI 스레드 안전하게 setArmAngles 슬롯을 호출하고,
각 관절 각도를 전달한다.

마지막으로 별도의 스레드를 만들어 rosThread_에 저장한다.
이 스레드 안에서는 rclcpp::Rate rate(30);로 30Hz 주기를 설정하고 반복문에서
rclcpp::spin_some(node_);를 호출해 콜백을 처리한다.
반복이 끝나면 rate.sleep();으로 주기를 맞추며, rosRunning_이 false가 되거나 ROS2가

종료될 때까지 계속 반복한다.

```
// 슬롯: 받은 값으로 로봇팔 회전
void MainWindow::setArmAngles(double base, double shoulder, double elbow)
{
    baseArm->setRotation(base);
    shoulderArm->setRotation(shoulder);
    elbowArm->setRotation(elbow);

    ui->bass->setValue(base);
    ui->mani_1->setValue(shoulder);
    ui->horizontalSlider_4->setValue(elbow);
}
```

setArmAngles 슬롯은 ROS2로부터 받은 관절 각도를 기반으로 GUI와 로봇팔을 동시에 업데이트한다.
먼저 baseArm->setRotation(base)로 로봇팔의 베이스 관절을 회전시키고,
shoulderArm->setRotation(shoulder)로 어깨 관절을,
elbowArm->setRotation(elbow)로 팔꿈치 관절을 회전시킨다.

이와 동시에 슬라이더 UI도 갱신한다.
ui->bass->setValue(base)로 베이스 슬라이더를,
ui->mani_1->setValue(shoulder)로 어깨 슬라이더를,
ui->horizontalSlider_4->setValue(elbow)로 팔꿈치 슬라이더를 설정하여 사용자에게 현재 상태를 보여준다.

2. publisher_pkg 메인 윈도우 파일

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

    // ROS2 초기화
    rclcpp::init(0, nullptr);
    node_ = std::make_shared<rclcpp::Node>("arm_publisher_gui_node");
    pub_ = node_->create_publisher<std_msgs::msg::Float64MultiArray>("arm_angles", 10);

    rosThread_ = std::thread([this]() {
        rclcpp::Rate rate(30);
        while (rclcpp::ok() && rosRunning_) {
            rclcpp::spin_some(node_);
            rate.sleep();
        }
    });

    // 슬라이더가 바뀔 때 라벨 업데이트
    connect(ui->baseSlider, &QSlider::valueChanged, this, &MainWindow::updateLabels);
    connect(ui->shoulderSlider, &QSlider::valueChanged, this, &MainWindow::updateLabels);
    connect(ui->elbowSlider, &QSlider::valueChanged, this, &MainWindow::updateLabels);

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

    // 초기 라벨 세팅
    updateLabels();
}

MainWindow::~MainWindow()
{
    rosRunning_ = false;
    if (rosThread_.joinable()) rosThread_.join();
    rclcpp::shutdown();
    delete ui;
}
```

ROS2를 초기화하고, "arm_publisher_gui_node"라는 이름으로 노드를 만든다.
로봇팔 각도를 퍼블리시할 퍼블리셔를 "arm_angles" 토픽에 큐 사이즈 10으로 생성한다.

ROS 콜백을 처리하기 위해 별도의 쓰레드를 만든다.
쓰레드 안에서는 rclcpp::Rate rate(30);로 초당 30회 주기를 맞추고,
rclcpp::spin_some(node_);로 콜백을 실행한 뒤 rate.sleep();으로 대기한다.

슬라이더 값이 바뀔 때마다 라벨을 갱신하기 위해 updateLabels 슬롯과 연결한다.
퍼블리시 버튼 클릭 시에는 publishAngles 슬롯이 호출되어 현재 슬라이더 값으로 메시지를 발행한다.
마지막으로 생성자 안에서 updateLabels();를 호출해 초기 라벨을 세팅한다.

소멸자 ~MainWindow()에서는 먼저 rosRunning_ = false;로 ROS 쓰레드를 종료 신호를

보내고, 쓰레드가 실행 중이면 join()으로 안전하게 합류시킨다.
그 다음 rclcpp::shutdown();으로 ROS2를 종료하고, delete ui;로 UI 객체를 해제한다.

```
// 퍼블리시 버튼 클릭 시 호출
void MainWindow::publishAngles()
{
    auto msg = std_msgs::msg::Float64MultiArray();
    msg.data = {
        static_cast<double>(ui->baseSlider->value()),
        static_cast<double>(ui->shoulderSlider->value()),
        static_cast<double>(ui->elbowSlider->value())
    };
    pub_->publish(msg);
}

// 슬라이더 값 기반 라벨 갱신
void MainWindow::updateLabels()
{
    ui->baseLabel->setText(QString("Base: %1° ").arg(ui->baseSlider->value()));
    ui->shoulderLabel->setText(QString("Shoulder: %1° ").arg(ui->shoulderSlider->value()));
    ui->elbowLabel->setText(QString("Elbow: %1° ").arg(ui->elbowSlider->value()));
}
```

publishAngles() 함수는 퍼블리시 버튼이 클릭 될 때 호출된다.
슬라이더에서 현재 값을 읽어 data에 차례대로 넣는다.
그 후 퍼블리셔 pub_를 통해 메시지를 발행한다.

updateLabels() 함수는 슬라이더 값에 맞춰 라벨을 갱신한다.
ui->baseSlider->value() 등에서 값을 가져와 QString::arg로 문자열을 만들고, 각 라벨(baseLabel, shoulderLabel, elbowLabel)에 설정한다.
이렇게 하면 슬라이더를 움직일 때마다 UI에 각도 정보가 실시간으로 반영된다.