과제1

1. 과제의 목표

로봇 팔의 링크와 관절 각도를 제어하고, 사용자가 입력하는 값에 따라 실시간으로 로봇 팔의 움직임을 시각화하는 시스템을 구현하는 것을 목표로 한다.

2. 구현 과정

Qt Designer를 이용해 사용자 인터페이스를 설계하였다. 사용자로부터 링크의 길이와 관절의 각도를 입력받기 위해 QSlider와 QDoubleSpinBox 위젯을 사용하였으며, QLCDNumber를 통해 각도를 실시간으로 시각적으로 표시하도록 구성하였다.

각 링크의 길이와 관절의 각도를 기반으로 로봇 팔의 위치를 계산하였다. 링크의 끝점 좌표는 삼각함수와 각도 변환을 통해 계산되었다. Qt의 QGraphicsScene을 활용하여 2D 공간에서 로봇 팔을 시각적으로 그려냈다.

각 관절의 각도는 슬라이더(QSlider)의 값으로 조정되며, 이 값은 QLCDNumber에 표시된다. 또한, 슬라이더 값의 변화에 따라 로봇 팔의 그래픽도 즉시 업데이트되도록 구현하였다.

```
connect(ui->joint1_slider, &QSlider::valueChanged, this, &MainWindow::updateJointAngles);
connect(ui->joint2_slider, &QSlider::valueChanged, this, &MainWindow::updateJointAngles);
connect(ui->joint3_slider, &QSlider::valueChanged, this, &MainWindow::updateJointAngles);
connect(ui->link1_spinBox, QOverload<double>::of(&QDoubleSpinBox::valueChanged), this, &MainWindow::updateJointAngles);
connect(ui->link2_spinBox, QOverload<double>::of(&QDoubleSpinBox::valueChanged), this, &MainWindow::updateJointAngles);
connect(ui->link3_spinBox, QOverload<double>::of(&QDoubleSpinBox::valueChanged), this, &MainWindow::updateJointAngles);
```

"Reset" 버튼을 통해 링크의 길이와 각도를 초기화할 수 있는 기능을 추가하였다. 이로 인해 사용자는 초기 상태로 돌아가 새로운 값을 입력할 수 있다.

주요 구성 요소

- MainWindow: 주 윈도우 클래스이며, UI 및 로봇 팔의 그래픽을 처리하는 핵심 클래스.

- scene: QGraphicsScene 객체로, 로봇 팔을 시각적으로 표시하기 위해 사용.

- fixlength1, fixlength2, fixlength3: 초기화된 각 링크의 길이.

- length1, length2, length3: 실시간으로 변경되는 각 링크의 길이.

```
}

double fixlength1, fixlength2, fixlength3;
double length1, length2, length3;
void MainWindow::on_makeArm_btn_clicked()
{
    fixlength1 = length1;
    fixlength2 = length2;
    fixlength3 = length3;

    drawArm();
}
```

- on_makeArm_btn_clicked(): 사용자가 설정한 링크의 길이를 고정값으로 저장하고, 로봇 팔을 그린다.

- updateJointAngles(): 슬라이더와 스핀박스의 값이 변경될 때마다 각도를 업데이트하고, LCD에 각도를 표시하며, 팔을 다시 그리는 메서드.

- drawArm(): 삼각함수를 이용해 각 링크의 끝점 좌표를 계산하고, 링크를 QGraphicsScene 위에 그린다.

```
}
void MainWindow::drawArm()
{
    //현재 링크 길이와 각도 가져오기
    length1 = ui->link1_spinBox->value();
    length2 = ui->link2_spinBox->value();
    length3 = ui->link3_spinBox->value();

    double angle1 = ui->joint1_slider->value();
    double angle2 = ui->joint2_slider->value();
    double angle3 = ui->joint3_slider->value();

    //각도는 라디안으로
    double angle1Rad = qDegreesToRadians(angle1);
    double angle2Rad = qDegreesToRadians(angle2);
    double angle3Rad = qDegreesToRadians(angle3);

    //각 링크의 끝점 계산
    QPointF p1(fixlength1 * cos(angle1Rad), fixlength1 * sin(angle1Rad));
    QPointF p2 = p1 + QPointF(fixlength2 * cos(angle1Rad + angle2Rad), fixlength2 * sin(angle1Rad + angle2Rad));
    QPointF p3 = p2 + QPointF(fixlength3 * cos(angle1Rad + angle2Rad + angle3Rad), fixlength3 * sin(angle1Rad + angle2Rad + angle3Rad));

    scene->clear();

    //각 링크 그리기
    lineItem* link1 =(new lineItem(fixlength1, angle1));
    link1->setPos(0, 0);
    scene->addItem(link1);

    lineItem* link2 =(new lineItem(fixlength2, angle1 + angle2));
    link2->setPos(p1);
    scene->addItem(link2);

    lineItem* link3 =(new lineItem(fixlength3, angle1 + angle2 + angle3));
    link3->setPos(p2);
    scene->addItem(link3);
}
```

결론

-본 프로젝트에서는 사용자가 입력하는 값을 실시간으로 반영하여 로봇 팔의 움직임을 시각화하는 시스템을 성공적으로 구현하였다. 링크 길이와 관절 각도를 조정하여 팔의 다양한 움직임을 표현할 수 있었다.

-향후 3D 공간에서의 시각화 및 다중 관절을 가진 로봇 팔 제어 시스템 구현을 계획 중이다.