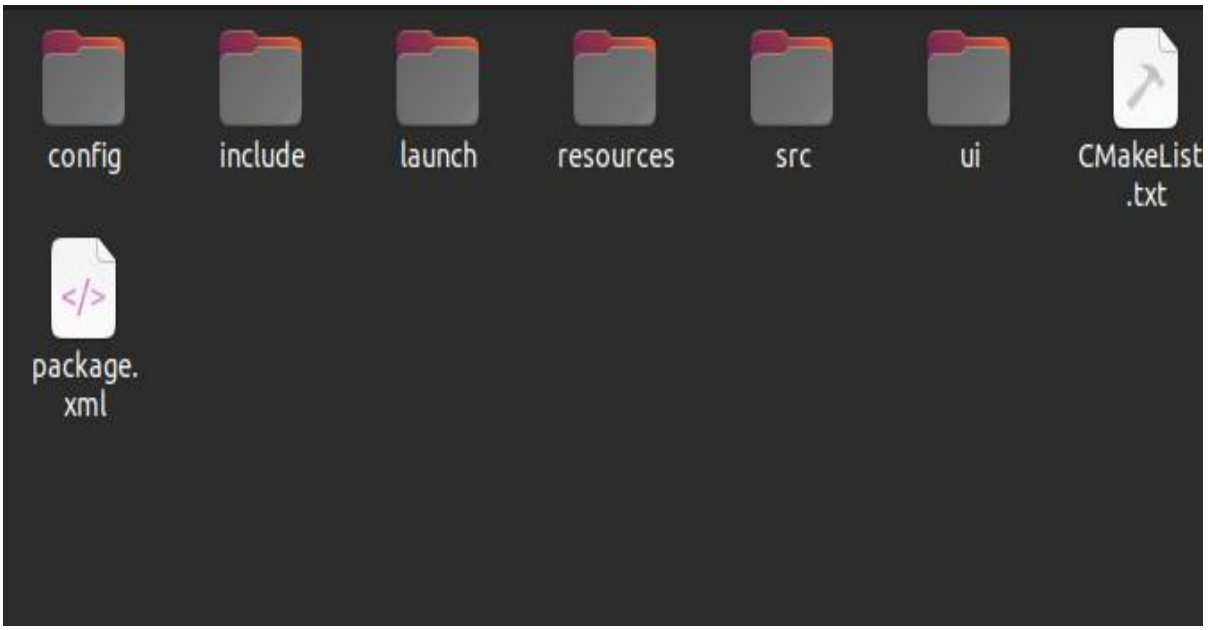
ROS DAY3
과제 2번 보고서

2023741024
로봇학부 박건후

목차

(1) 전체적인 폴더 구성



hw4_pkg패키지의 내부입니다. 기본적인 include, src 등등이 있고 그 상태에서 config 폴더와 launch 폴더를 만든 것입니다. config내에는 yaml파일을 작성하였고 launch 파일에서는 py파일을 작성하였습니다.

(2) 런치 파일 설명

이 런치 파일을 파이썬으로 작성하기 위해서는 LaunchDescription이 필요합니다. 그러기 위해서 그 부분을 import 해왔습니다. 그리고 Node단위로 프로그램을 실행시키기 때문에 동시에 여러 프로그램을 실행시키기 위해서는 여러 Node를 구성해야 합니다. 그래서 Node 또한 import해서 가져왔습니다.

```
def generate_launch_description():
```

Node를 구성하는 코드를 이 함수에서 작성해야 합니다. 그리고 이 함수의 반환값은 LaunchDescription으로 묶어서 반환하며 이 Node()들이 여러 개일 경우 리스트를 사용하여 여러 Node()들을 반환합니다.

Node 내부에는 package, executable, name, 더 구분하기 위해서는 namespace를 사용하는 등의 형식 혹은 변수를 사용합니다. 저는 구동시킬 패키지, 실행 파일을 작성하는

executable, 노드 이름을 의미하는 name을 작성하였습니다. 그리고 추가적으로 매개변수를 받는 노드는 parameters라는 변수를 사용하여 yaml파일의 절대경로를 적어줬습니다.

(3) yaml 파일 설명

yaml 파일에서는 파라미터들을 입력해줘야 합니다. 이번 과제에서는 3축 로봇팔에 필요한 3개의 링크, 3개의 관절의 각도를 각각 매개변수로 설정했습니다. 우선 파일을 작성할 때는 노드명을 작성하고 ros_parameters:라는 문구를 작성한 후 매개변수를 작성하였습니다.

(4) 소스 파일에서 추가로 작성한 부분

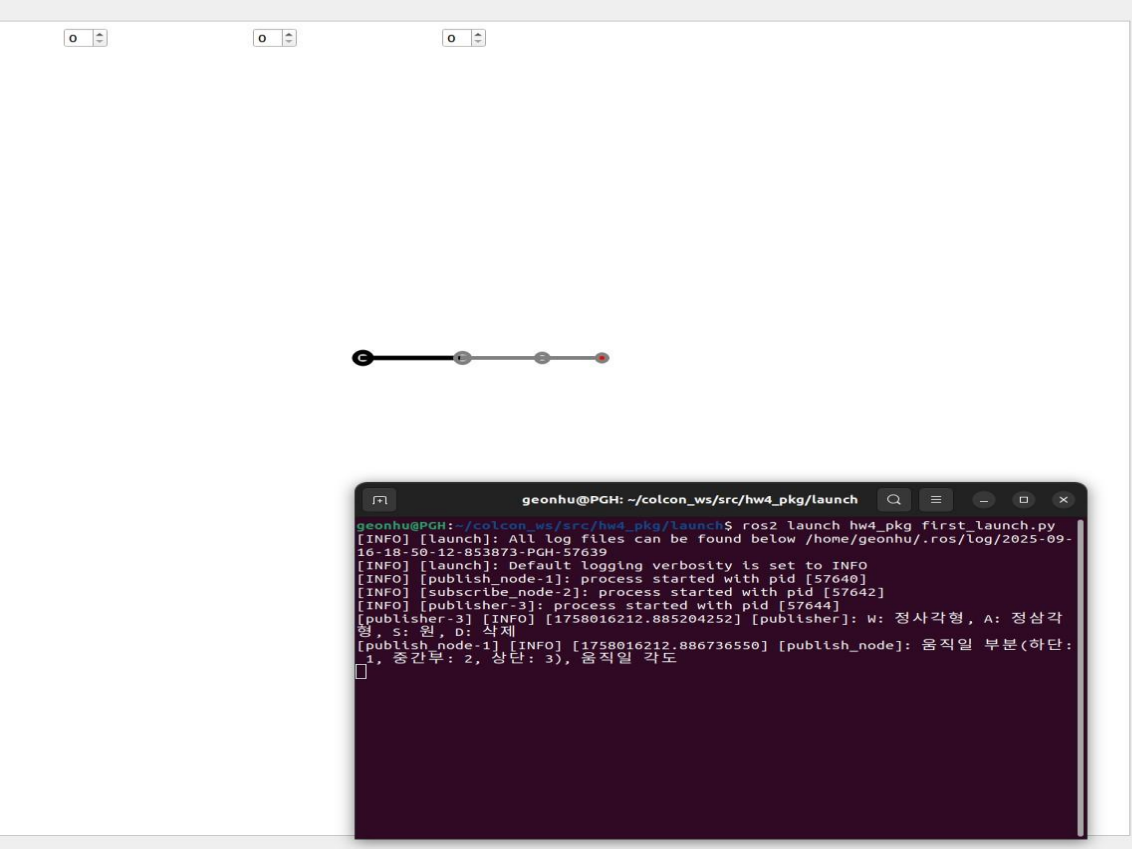
1번 과제에서 파라미터를 작성하는 메서드에 대해서 배웠습니다. 이는 Node클래스에 정의되어 있는 메서드이고 이 메서드를 작성하는 경우에는 Node클래스를 상속받았거나 이 노드 객체를 멤버로 하는 클래스에서 작성해야 합니다. 그래서 저는 subscribe_node라는 클래스의 생성자에 그 코드를 작성했습니다. 그래서 아래 사진과 같이 매개변수 이름을 적고 디폴트값을 적어주었습니다.

```
this->declare_parameter<double>("link_bottom", 0.0);
```

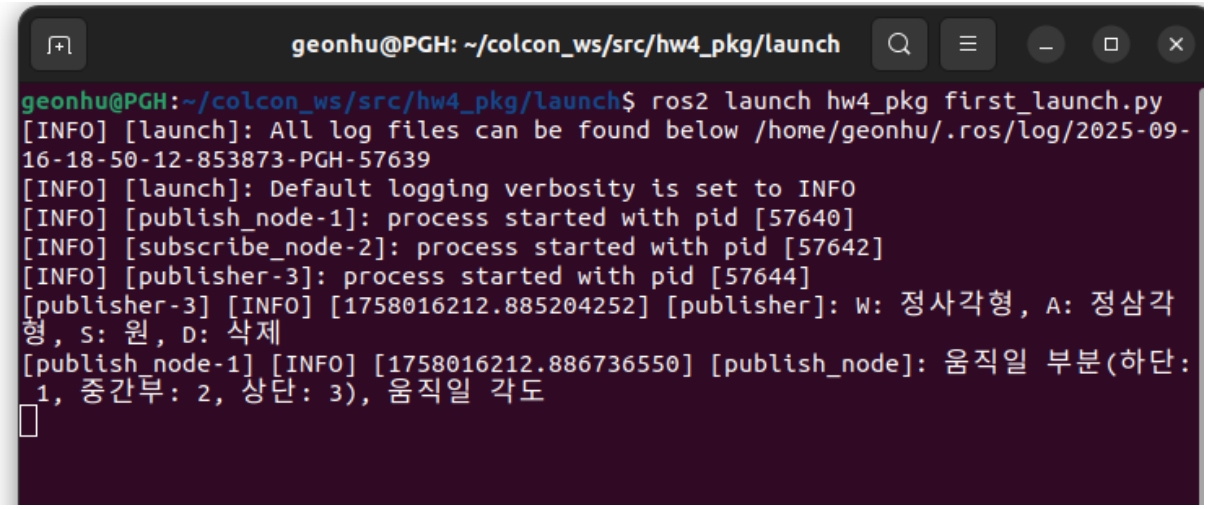
그리고 그 다음 아래 사진은 선언을 한 후에 get_parameter를 통해 yaml파일에 있는 값을 가져오는 코드입니다. 그래서 이 값을 변수에 저장한 후에 시그널-슬롯의 매개변수를 통해 MainWindow 쪽으로 그 정보를 보내고자 했습니다.

```
double L1 = this->get_parameter("link_bottom").as_double();
```

이렇게 소스코드를 수정하고 launch 파일을 만들고, config폴더를 만들었습니다. 하지만 동시에 노드들을 실행시키는 것은 가능했지만 무슨 이유인지 모르겠지만 파라미터 값을 가져오는 것이 되지 않았습니다. 그렇게 되면 디폴트 값이라도 가져와야 하는데 디폴트 값까지 받아오지 못했습니다. 과제 2번의 2번째 조건을 충족시키지 못해 죄송합니다.



이는 3개의 노드를 동시에 실행시킨 사진입니다.

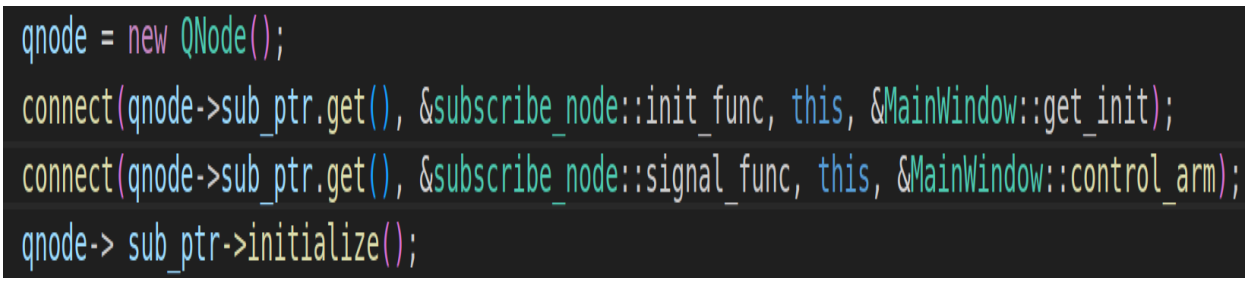


위의 사진을 보시면 publisher노드로써 정사각형, 정삼각형, 원을 그리는 것에 대한 입력을 받는 창과, publish_node라는 노드로써 로봇팔을 움직이는 것에 대해 어느 부분을 어느 각도로 움직이게 할 것이지에 대한 입력을 받는 것이므로 qt팔을 움직이는 윈도우 창과 연결되며 메시지를 받는 subscribe_node 노드까지 3개의 노드가 동시에 호출되었음을 보실 수 있습니다.

2025.09.20 수정한 부분:

파라미터 값을 받아오지 못했기에 차후에 문제점을 찾고 보완하였습니다. 문제점은 매개변수를 받아오고 이를 시그널을 발생시키는 타이밍에 있었습니다.

위의 사진은 main_window.cpp파일의 MainWindow의 생성자 구현부분입니다. 여기서 링크 길이, 팔 각도 등의 상태를 저장하며 팔을 그리는 메서드를 정의한 클래스입니다. 생성자 내부에서 이 클래스를 동적할당하는 모습입니다. 그 이후의 코드가



이와 같습니다. 위 사진에서는 QNode를 동적할당하시는 것을 보실 수 있고 QNode안에는 subscribe_node에 대한 shared_ptr이 있기에 QNode를 동적할당한다는 의미는 곧 subscribe_node 객체 또한 할당된다는 것입니다. 그리고 예전에 저는 yaml 파일로부터 매개변수를 받아오는 코드를 subscribe_node의 생성자에서 받아오고 생성자의 마지막에 시그널 메서드를 발생시키는 코드를 짰었습니다. 하지만 그렇게 되면 아래의 connect라는 코드가 실행되기도 전에 시그널을 발생시키는 것이 됩니다. 그래서 subscribe_node 생성자에 있던 파라미터 받는 코드를 subscribe_node에 initialize라는 메서드를 하나 만들어서 거기에서 매개변수를 받아오는 것으로 수정하였습니다. 그리하여 subscribe_node생성자가 호출되고, connect 구문이 실행된 후에 그 밑에 initialize를 호출하게 하여 문제를 해결하였습니다.

마지막으로, 실행 영상을 매개변수까지 완벽하게 받는 것으로 업데이트하였습니다. 감사합니다.