

5. 키보드/게임패드로 로봇 팔 제어하기

로봇 팔 제어 관련 노드 제어 창(Rviz, Moveit2) 이 모두 닫혀 있어야 함.
로봇 팔 드라이버 노드 까지 닫혀 있다면, 재실행 필요함. (기존 강의 자료 [참고](#))

신규 터미널에서 아래 명령 실행 (rviz2 로봇 팔 모델 인터페이스 실행)

```
$ cd roarm_ws_em0  
$ ros2 launch moveit_servo demo.launch.py
```

5.1 로봇 팔의 키보드 제어

새로운 터미널에서 아래 명령 실행

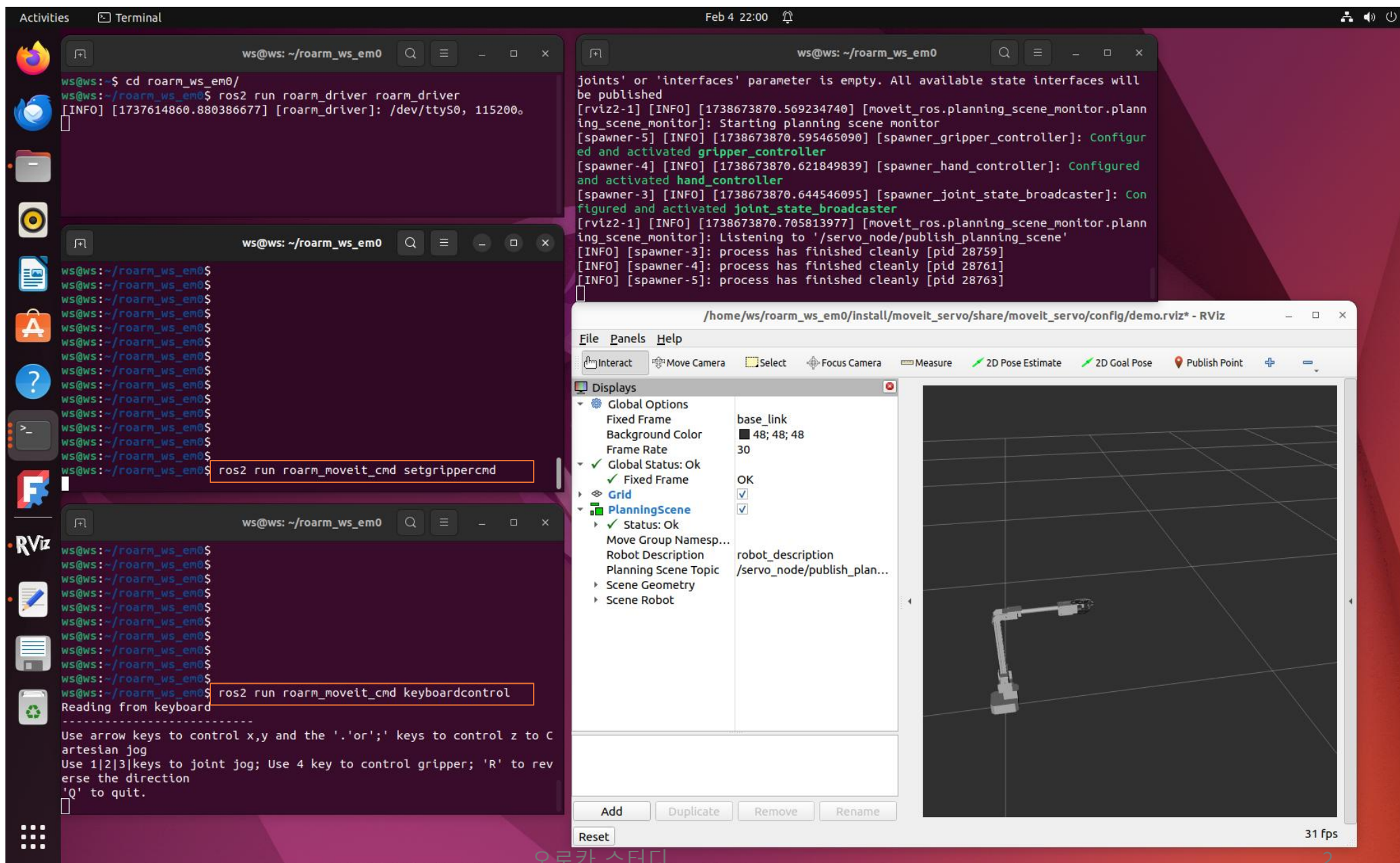
```
$ cd ~/roarm_ws_em0  
$ ros2 run roarm_moveit_cmd setgrippercmd
```

새로운 터미널에서 아래 명령 실행

```
$ cd ~/roarm_ws_em0  
$ ros2 run roarm_moveit_cmd keyboardcontrol
```

→ 이 실행창에 Focus 유지

실행화면



제어를 위한 키 설정 값

•좌표 제어

- 화살표 키 ↑ : X축 양의 방향으로 이동
- 화살표 키 ↓ : X축의 음수 방향으로 이동
- 화살표 키 ← : 양의 Y축 방향으로 이동
- 화살표 키 → : Y축의 음수 방향으로 이동
- ; 키 : 양의 Z축 방향으로 이동
- . 키 : Z축의 음의 방향으로 이동

•조인트 컨트롤

- 숫자 1 번 키 : 베이스 관절 운동
- 숫자 2번 키 : 어깨관절 운동
- 숫자 3번 키 : 팔꿈치 관절 운동
- 숫자 4 키 : 그리퍼 조인트 동작
- 문자 R 키: 위의 조인트 컨트롤의 방향을 전환합니다.

(주의사항)

Linux Terminal 창에서 키를 계속 누르고 있는 경우, 키 버퍼에 값이 누적되어 전송됨.

필요한 만큼만, 키를 눌러야 함.

Q 문자를 눌러 작업을 종료하세요.

5.2 로봇 팔을 위한 게임패드 제어

컨트롤러 수신기를 컴퓨터에 꽂고 "장치" → "USB" → Oracle VM VirtualBox 상단에 있는 "Xbox"가 있는 장치 이름을 클릭합니다. 장치 이름 앞에 체크 표시(✓)가 있으면 컨트롤러가 가상 머신에 연결되었음을 나타냅니다. 다음 게임 컨트롤러 버튼을 사용하여 로봇 팔을 제어할 수 있습니다.

•좌측 관절 제어:

- 좌우 버튼 : 기본 관절 이동
- 위아래 버튼: 어깨 관절 움직임

•오른쪽 관절 제어:

- X, B 버튼 : 팔꿈치 관절 동작
- Y, A 버튼 : 그리퍼 조인트 동작

•XY 축 좌표 제어:








- 왼쪽 스틱: X축 이동
- 오른쪽 스틱: Y축 이동

관련 코드는 `roam_moveit_cmd`에 있지 않고, `moveit_servo`에 존재함. 처리 방식은 `keyboardcontrol`과 유사함.

```
/home/ws/roarm_ws_em0/src/roarm_main/moveit\_servo/src/teleop_demo/  
joystick\_servo\_example.cpp 참조
```

소스코드 설명

/home/ws/roarm_ws_em0/src/roarm_main/[roarm_moveit_cmd](#)/src

Name	Size	Modified	
 getposecmd_moveit2.cpp	2.5 kB	20 Aug 2024	☆
 getposecmd_tf2.cpp	3.1 kB	20 Aug 2024	☆
 keyboardcontrol.cpp	9.1 kB	20:31	☆
 movecirclecmd.cpp	4.5 kB	20 Aug 2024	☆
 movepointcmd.cpp	2.4 kB	20 Aug 2024	☆
 setgrippercmd.cpp	4.0 kB	20 Aug 2024	☆
 webappcontrol.cpp	2.0 kB	20 Aug 2024	☆

→ Keyboard 입력을 받으면, Topic을 로봇제어 Node로 발송

→ Keyboard 입력에서 전달된 gripper 제어 Topic을 로봇제어 Node로 발송

Keyboardcontrol.cpp (1/3)

키보드 값 정의

moveit_servo에 정의된 Topic Node 지정

Gripper cmd 처리 topic

Topic Msg 발송시 필요한 ID 지정

키보드 입력 값 1개씩 읽는다

```
52 // Define used keys
53 #define KEYCODE_RIGHT 0x43
54 #define KEYCODE_LEFT 0x44
55 #define KEYCODE_UP 0x41
56 #define KEYCODE_DOWN 0x42
57 #define KEYCODE_PERIOD 0x2E
58 #define KEYCODE_SEMICOLON 0x3B
59 #define KEYCODE_1 0x31
60 #define KEYCODE_2 0x32
61 #define KEYCODE_3 0x33
62 #define KEYCODE_4 0x34
63 #define KEYCODE_Q 0x71
64 #define KEYCODE_R 0x72
65
66 // Some constants used in the Servo Teleop demo
67 const std::string TWIST_TOPIC = "/servo_node/delta_twist_cmds";
68 const std::string JOINT_TOPIC = "/servo_node/delta_joint_cmds";
69 const std::string GRIPPER_TOPIC = "/gripper_cmd";
70 const size_t ROS_QUEUE_SIZE = 10;
71 const std::string EEF_FRAME_ID = "hand_tcp";
72 const std::string BASE_FRAME_ID = "base_link";
73
74 // A class for reading the key inputs from the terminal
75 class KeyboardReader
76 {
77 public:
78     KeyboardReader() : kfd(0)
79     {
80         // get the console in raw mode
81         tcgetattr(kfd, &cooked);
82         struct termios raw;
83         memcpy(&raw, &cooked, sizeof(struct termios));
84         raw.c_lflag &= ~(ICANON | ECHO);
85         // Setting a new line, then end of file
86         raw.c_cc[VEOL] = 1;
87         raw.c_cc[VEOF] = 2;
88         tcsetattr(kfd, TCSANOW, &raw);
89     }
90     void readOne(char* c)
91     {
92         int rc = read(kfd, c, 1);
93         if (rc < 0)
94             throw std::runtime_error("read failed");
95     }
96 }
```

Keyboardcontrol.cpp (2/3)

```
198 // // Create the messages we might publish
199 auto twist_msg = std::make_unique<geometry_msgs::msg::TwistStamped>();
200 auto joint_msg = std::make_unique<control_msgs::msg::JointJog>();
201 auto gripper_msg = std::make_unique<std_msgs::msg::Float32>();
202
203 // Use read key-press
204 switch (c)
205 {
206     case KEYCODE_LEFT:
207         RCLCPP_DEBUG(nh_ ->get_logger(), "LEFT");
208         twist_msg->twist.linear.y = -0.5; //-1.0;
209         publish_twist = true;
210         break;
211     case KEYCODE_RIGHT:
212         RCLCPP_DEBUG(nh_ ->get_logger(), "RIGHT");
213         twist_msg->twist.linear.y = 0.5; //1.0;
214         publish_twist = true;
215         break;
216 }
```

```
255     case KEYCODE_4:
256         RCLCPP_DEBUG(nh_ ->get_logger(), "4");
257         if (joint_vel_cmd_ > 0)
258         {
259             gripper_value_ += 0.01;
260         }
261         else if (joint_vel_cmd_ < 0)
262         {
263             gripper_value_ -= 0.01;
264         }
265         if (gripper_value_ > 1.5)
266         {
267             gripper_value_ = 1.5;
268             puts("MAX 1.5");
269         }
270         else if (gripper_value_ < 0.0)
271         {
272             gripper_value_ = 0.0;
273             puts("MIN 0,0");
274         }
275         gripper_msg->data = gripper_value_;
276         publish_gripper = true;
277         break;
278     case KEYCODE_R:
279         RCLCPP_DEBUG(nh_ ->get_logger(), "R");
280         joint_vel_cmd_ *= -1;
281         break;
282     case KEYCODE_Q:
283         RCLCPP_DEBUG(nh_ ->get_logger(), "quit");
284         return 0;
285 }
```

각 키에 따라 정의 된 동작을 위해서,
Topic에 값을 설정하거나
제어 값을 변경한다.

Keyboardcontrol.cpp (3/3)

```
287 // If a key requiring a publish was pressed, publish the message now
288 if (publish_twist)
289 {
290     twist_msg->header.stamp = nh_->now();
291     twist_msg->header.frame_id = frame_to_publish_;
292     twist_pub_->publish(std::move(twist_msg));
293     publish_twist = false;
294 }
295 else if (publish_joint)
296 {
297     joint_msg->header.stamp = nh_->now();
298     joint_msg->header.frame_id = BASE_FRAME_ID;
299     joint_pub_->publish(std::move(joint_msg));
300     publish_joint = false;
301 }else if (publish_gripper)
302 {
303     gripper_pub_->publish(std::move(gripper_msg));
304     publish_gripper = false;
305 }
306
307 // adding for other thread works by techne.sc
308 usleep(10000);
309 }
310
311 return 0;
312 }
313
```

좌표제어 topic msg 발송

조인트 제어 topic msg 발송

그ripper 제어 topic msg 발송

<코드 수정 후 빌드시 명령>

\$ cd roarm_ws_em0

\$ colcon build --packages-select roarm_moveit_cmd

setgrippercmd.cpp (1/2)

```
8 public:
9     using GripperCommand = control_msgs::action::GripperCommand;
10    using GoalHandleGripperCommand = rclcpp_action::ClientGoalHandle<GripperCommand>;
11
12    GripperActionClient()
13    : Node("gripper_action_client")
14    {
15        // 创建 Action 客户端
16        this->action_client_ = rclcpp_action::create_client<GripperCommand>(this, "/gripper_controller/gripper_cmd");
17
18        // 创建订阅者, 订阅 /gripper_cmd 话题
19        this->subscription_ = this->create_subscription<std_msgs::msg::Float32>(
20            "/gripper_cmd", 10,
21            std::bind(&GripperActionClient::listener_callback, this, std::placeholders::_1));
22    }
23
24 private:
25    rclcpp_action::Client<GripperCommand>::SharedPtr action_client_;
26    rclcpp::Subscription<std_msgs::msg::Float32>::SharedPtr subscription_;
27
28    void listener_callback(const std_msgs::msg::Float32::SharedPtr msg)
29    {
30        // 将接收到的 int32 值转换为浮点数 position
31        double position = static_cast<double>(msg->data);
32        RCLCPP_INFO(this->get_logger(), "Received position: %f", position);
33
34        // 发送目标
35        send_goal(position);
36    }
```

moveit_servo의
gripper action
client 생성

Keyboardcontrol에
서 발송되는 topic
수신

수신된 topic 처리

setgrippercmd.cpp (2/2)

```
38 void send_goal(double position)
39 {
40     // 构建 GripperCommand 的 Goal 消息
41     auto goal_msg = GripperCommand::Goal();
42     goal_msg.command.position = position;
43     goal_msg.command.max_effort = 10.0; // 固定 max_effort
44
45     // 等待 Action 服务启动
46     if (!this->action_client_->wait_for_action_server()) {
47         RCLCPP_ERROR(this->get_logger(), "Action server not available");
48         return;
49     }
50
51     // 异步发送 Goal
52     auto send_goal_options = rclcpp_action::Client<GripperCommand>::SendGoalOptions();
53     send_goal_options.goal_response_callback =
54         std::bind(&GripperActionClient::goal_response_callback, this, std::placeholders::_1);
55     send_goal_options.result_callback =
56         std::bind(&GripperActionClient::result_callback, this, std::placeholders::_1);
57     send_goal_options.feedback_callback =
58         std::bind(&GripperActionClient::feedback_callback, this, std::placeholders::_1, std::placeholders::_2);
59
60     this->action_client_->async_send_goal(goal_msg, send_goal_options);
61 }
62
63 void goal_response_callback(GoalHandleGripperCommand::SharedPtr goal_handle)
64 {
65     if (!goal_handle) {
66         RCLCPP_INFO(this->get_logger(), "Goal was rejected by server");
67     } else {
68         RCLCPP_INFO(this->get_logger(), "Goal accepted by server, waiting for result");
69     }
70 }
71
72 void result_callback(const GoalHandleGripperCommand::WrappedResult & result)
73 {
74     switch (result.code) {
75         case rclcpp_action::ResultCode::SUCCEEDED:
76             RCLCPP_INFO(this->get_logger(), "Goal succeeded");
77             break;
```

Action msg 값 설정

Action msg 전송

<ROS2 명령으로 확인한 Node/Topic/action 목록>

roamdriver, keyboardcontrol, grippercmd,
moveit_servo(Rviz) launch까지 모두 실행 된 상태

```
ws@ws:~/roarm_ws_em0$ ros2 node list
/controller_manager
/controller_to_servo_node
/gripper_controller
/hand_controller
/joint_state_broadcaster
/joy_node
/launch_ros_34079
/moveit_servo_demo_container
/roarm_driver
/robot_state_publisher
/rviz2
/rviz2_private_137329768319904
/servo_node
/servo_node_private_110430895027520
/transform_listener_impl_646fb2452050
/transform_listener_impl_64ed3a9e8f30
/transform_listener_impl_7ce694009eb0
ws@ws:~/roarm_ws_em0$ ros2 action list
/gripper_controller/gripper_cmd
/hand_controller/follow_joint_trajectory

ws@ws:~/roarm_ws_em0$ ros2 topic list
/attached_collision_object
/clicked_point
/dynamic_joint_states
/goal_pose
/gripper_cmd
/gripper_controller/transition_event
/hand_controller/controller_state
/hand_controller/joint_trajectory
/hand_controller/state
/hand_controller/transition_event
/hand_pose
/initialpose
/joint_state_broadcaster/transition_event
/joint_states
/joy
/joy/set_feedback
/parameter_events
/planning_scene
/robot_description
/rosout
/servo_node/collision_velocity_scale
/servo_node/delta_joint_cmds
/servo_node/delta_twist_cmds
/servo_node/publish_planning_scene
/servo_node/status
/tf
/tf_static
```

roamdriver, keyboardcontrol,
grippercmd 창 까지만 실행된
상태

```
ws@ws:~/roarm_ws_em0$ ros2 node list
/gripper_action_client
/roarm_driver
/servo_keyboard_input
ws@ws:~/roarm_ws_em0$ ros2 topic list
/gripper_cmd
/hand_pose
/joint_states
/parameter_events
/rosout
/servo_node/delta_joint_cmds
/servo_node/delta_twist_cmds
ws@ws:~/roarm_ws_em0$ ros2 action list
/gripper_controller/gripper_cmd
```