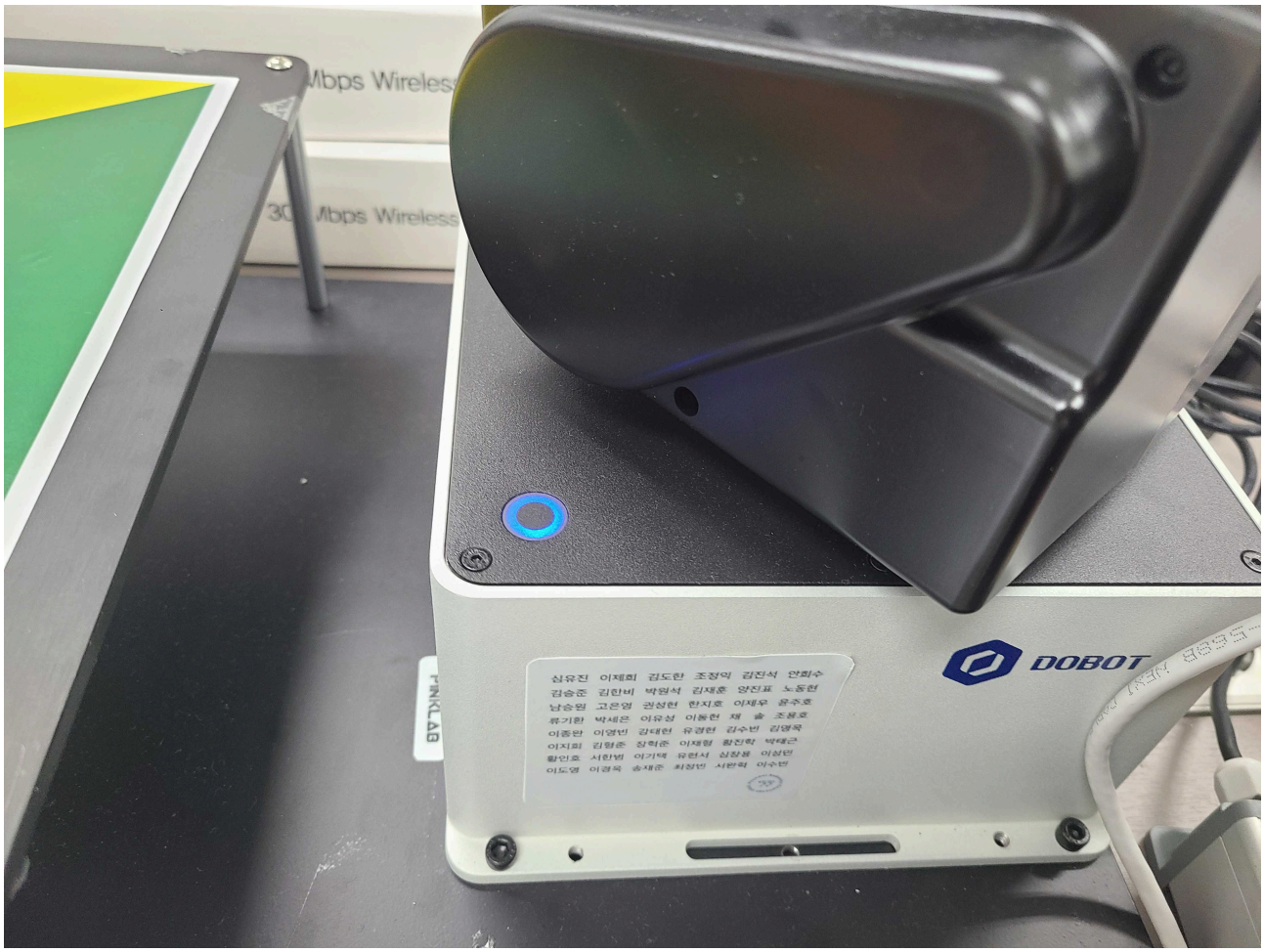
두봇 패키지 매뉴얼

ai-contents

Exported on 01/06/2024

Table of Contents

1 먼저 두봇 전원을 켜서 파란불이 들어올때까지 대기한다



2 그리고 두봇 dashboard server노드를 실행하고 연결성공이 뜨면 서비스 명령어를 받을 수 있다

```
mkh@mkh:~/d435_depallet$ ros2 run dobot_ros2 dashboard_server
[INFO] [1704531197.945099696] [dashboard]: 연결 설정 중...
[INFO] [1704531197.949615780] [dashboard]: 연결 성공!!
```

3 서비스 리스트 /dashboard_server 서비스에 명령어를 보내어 두봇을 설정한다

```
mkh@mkh:~/d435_depallet$ ros2 service list
/dashboard/describe_parameters
/dashboard/get_parameter_types
/dashboard/get_parameters
/dashboard/list_parameters
/dashboard/set_parameters
/dashboard/set_parameters_atomically
/dashboard_server
/find_roi/describe_parameters
/find_roi/get_parameter_types
/find_roi/get_parameters
/find_roi/list_parameters
/find_roi/set_parameters
/find_roi/set_parameters_atomically
```

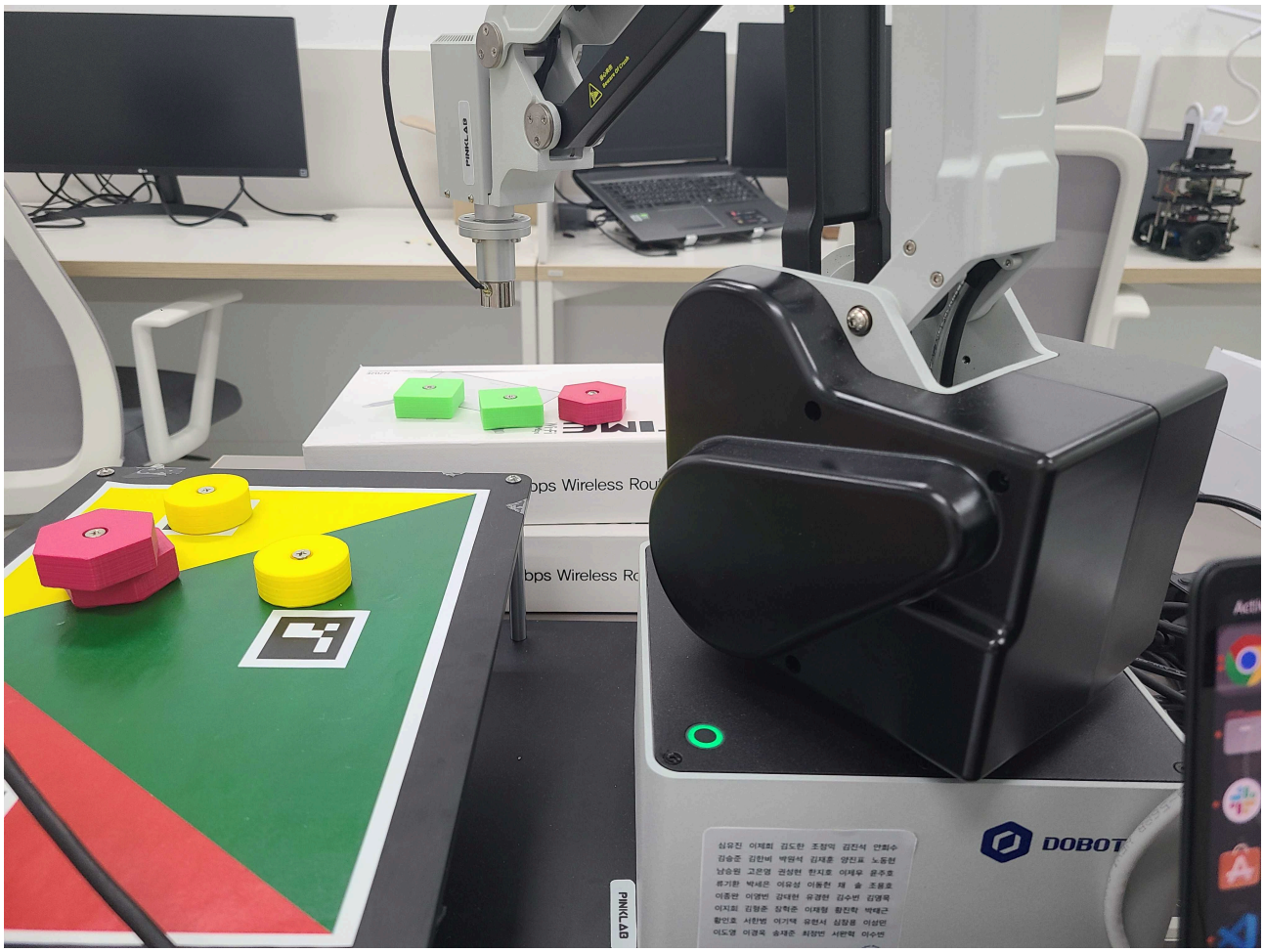
4 /dashboard_server 의 서비스 인터페이스

```
mkh@mkh:~/d435_depallet$ ros2 interface show dobot_ros2_interface/srv/Dash
board
string req_str
---
string res_str
mkh@mkh:~/d435_depallet$
```

5 먼저 power on을 보내어 로봇의 전원을 킨다

```
mkh@mkh:~/d435_depallet$ ros2 service call /dashboard_server dobot_ros2_in
terface/srv/Dashboard "{req_str: power on}"
requester: making request: dobot_ros2_interface.srv.Dashboard_Request(req_
str='power on')

response:
dobot_ros2_interface.srv.Dashboard_Response(res_str='power on')

mkh@mkh:~/d435_depallet$
```

6 초록불이 들어오면 성공!



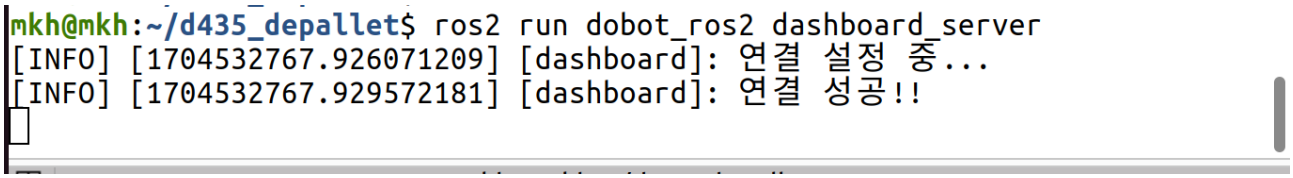
7 로봇이 켜진 상태에서 메시지를 보내면 다양한 제어를 할 수 있다

8 현재 구현된 메시지는 로봇이나 그리퍼를 키고 끄는 기능이 구현되어있다

```
def callback_service(self, request, response):
    if request.req_str == 'power on':
        self.dashboard.EnableRobot()
        self.robot_speed(self.dashboard, self.speed_value)
        time.sleep(0.1)
        response.res_str = 'power on'

    elif request.req_str == 'power off':
        self.dashboard.DisableRobot()
        time.sleep(0.1)
        response.res_str = 'power off'

    elif request.req_str == 'clear error':
        self.dashboard.DisableRobot()
        time.sleep(0.3)
        self.robot_clear(self.dashboard)
        time.sleep(0.3)
        self.dashboard.EnableRobot()
        self.robot_speed(self.dashboard, self.speed_value)
        time.sleep(0.3)
        response.res_str = 'clear error'

    #gripper

    elif request.req_str == 'gripper on':
        self.gripper_DO(self.dashboard, self.gripper_p, 1)
        time.sleep(0.1)
        response.res_str = 'gripper on'

    elif request.req_str == 'gripper off':
        self.gripper_DO(self.dashboard, self.gripper_p, 0)
        time.sleep(0.1)
        response.res_str = 'gripper off'
```

9 그리고 만약에 로봇팔을 제어하다가 빨간불이 들어오면 로봇을 제어할수없다

10 그러면 dashboard server를 다시 껏다가 키고



11 clear error 메시지를 이용해 로봇을 다시 제어 할수 있게한다

```
mkh@mkh:~/d435_depallet$ ros2 service call /dashboard_server dobot_ros2_in
terface/srv/Dashboard "{req_str: clear error}"
requester: making request: dobot_ros2_interface.srv.Dashboard_Request(req_
str='clear error')

response:
dobot_ros2_interface.srv.Dashboard_Response(res_str='clear error')

mkh@mkh:~/d435_depallet$
```

12 로봇팔의 속도 설정은 ros2 파라미터 설정으로 설정할 수 있다

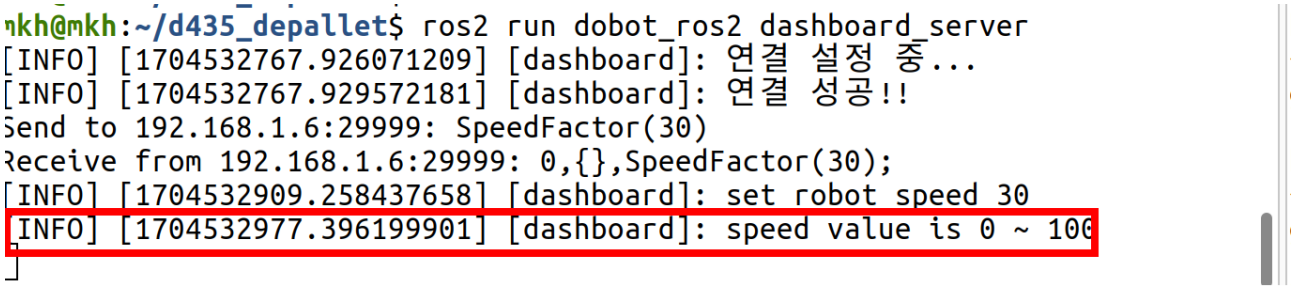
```
mkh@mkh:~/d435_depallet$ ros2 param list
/dashboard:
  speed_value
  use_sim_time
mkh@mkh:~/d435_depallet$
```

13 param set으로 설정한다

```
mkh@mkh:~/d435_depallet$ ros2 param set /dashboard speed_value 30
Set parameter successful
mkh@mkh:~/d435_depallet$
```

```
mkh@mkh:~/d435_depallet$ ros2 run dobot_ros2 dashboard_server
[INFO] [1704532767.926071209] [dashboard]: 연결 설정 중...
[INFO] [1704532767.929572181] [dashboard]: 연결 성공!!
Send to 192.168.1.6:29999: SpeedFactor(30)
Receive from 192.168.1.6:29999: 0,{},SpeedFactor(30);
[INFO] [1704532909.258437658] [dashboard]: set robot speed 30
```

14 로봇팔의 속도 범위는 0 ~ 100이기 때문에 범위를 벗어나는 숫자이면 설정이 안된다



15 로봇팔의 위치 제어는 action으로 할 수 있다

```
mkh@mkh:~/d435_depallet$ ros2 action list
/move_goal_server
mkh@mkh:~/d435_depallet$
```

16 /move_goal_server의 action 인터페이스

```
mkh@mkh:~/d435_depallet$ ros2 interface show dobot_ros2_interface/action/M
ove
geometry_msgs/Pose goal_pose
        Point position
                float64 x
                float64 y
                float64 z
        Quaternion orientation
                float64 x 0
                float64 y 0
                float64 z 0
                float64 w 1
---
string result_str
---
geometry_msgs/Pose current_pose
        Point position
                float64 x
                float64 y
                float64 z
        Quaternion orientation
                float64 x 0
                float64 y 0
                float64 z 0
                float64 w 1
mkh@mkh:~/d435_depallet$
```

17 goal좌표를 설정해서 send_goal을 하면 로봇팔이 goal좌표로 이동하게 된다

```
ros2 action send_goal --feedback /move_goal_server dobot_ros2_interface/action/Move
"{goal_pose: {position: {x: 300, y: 19, z: -56.5}, orientation: {w: 1}}}"
```

```
mkh@mkh:~/d435_depallet$ ros2 action send_goal --feedback /move_goal_serve
r dobot_ros2_interface/action/Move "{goal_pose: {position: {x: 300, y: 19,
 z: -56.5}, orientation: {w: 1}}}"
Waiting for an action server to become available...
Sending goal:
     goal_pose:
  position:
    x: 300.0
    y: 19.0
    z: -56.5
  orientation:
    x: 0.0
    y: 0.0
    z: 0.0
    w: 1.0

Goal accepted with ID: 31dfc399eb7a4069b929c7088643eca3
```

18 goal좌표가 로봇팔이 움직일 수 없는 범위로 설정되면 로봇팔이 빨간불이 뜬다

19 이는 위의 빨간불 해결방법으로 해결한다

20 그리고 위에 설명한 터미널 방법으로 매번 입력하기 힘들다

21 그래서 client 노드를 이용해 arg를 받아 실행 할 수있게 한다

22 dashboard_client 노드를 이용해 위에 터미널을 입력했던것 처럼 로봇을 (on-off)설정할 수 있고

```
mkh@mkh:~/d435_depallet$ ros2 run dobot_ros2 dashboard_client "power on"
[INFO] [1704534407.771520963] [dashboard_client]: Result : power on
mkh@mkh:~/d435_depallet$ ros2 run dobot_ros2 dashboard_client "power off"
[INFO] [1704534414.940112360] [dashboard_client]: Result : power off
mkh@mkh:~/d435_depallet$ ros2 run dobot_ros2 dashboard_client "gripper on"
[INFO] [1704534425.121469753] [dashboard_client]: Result : gripper on
mkh@mkh:~/d435_depallet$ ros2 run dobot_ros2 dashboard_client "power on"
[INFO] [1704534439.776514661] [dashboard_client]: Result : power on
mkh@mkh:~/d435_depallet$ ros2 run dobot_ros2 dashboard_client "gripper on"
[INFO] [1704534441.684141129] [dashboard_client]: Result : gripper on
mkh@mkh:~/d435_depallet$
```

23 move_goal_cllinet 노드를 이용해 로봇팔을 제어할 수 있다

```
mkh@mkh:~/d435_depallet$ ros2 run dobot_ros2 move_goal_client 300 10 -19
mkh@mkh:~/d435_depallet$
[INFO] [1704533657.618943959] [dashboard]: Move to goal[300.0, 10.0, -19.0
, 0.0]
Send to 192.168.1.6:30003: MovL(300.000000,10.000000,-19.000000,0.000000)
Receive from 192.168.1.6:30003: 0,{},MovL(300.000000,10.000000,-19.000000,
0.000000);
[INFO] [1704533657.726527905] [dashboard]: current_pose is [300.00192941
19.00010658 -56.49992371   1.00090337   0.
   0.        ]
[INFO] [1704533658.727505790] [dashboard]: current_pose is [300.02970443
18.9922284  -56.48145676   1.00181723   0.
   0.        ]
[INFO] [1704533658.727872720] [dashboard]: goal succeed
```