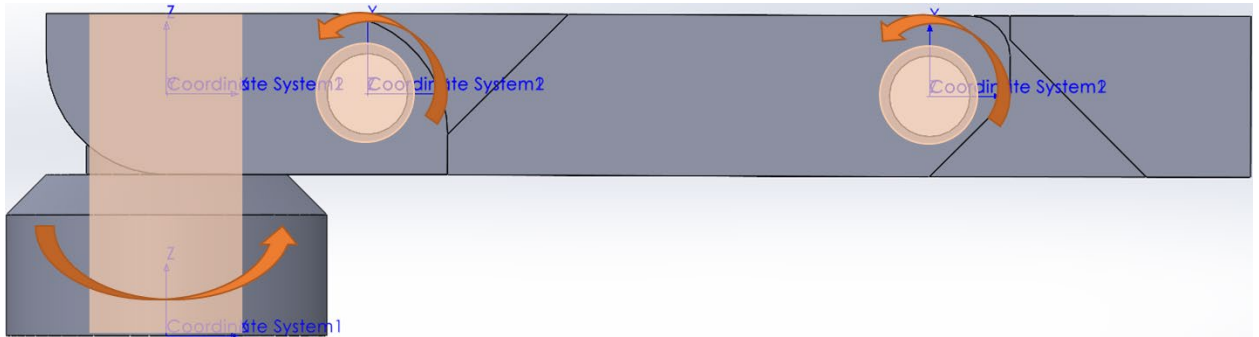


Lab_2 report: Robot Modeling & Kinematic

สรุปภาพรวม System Architecture

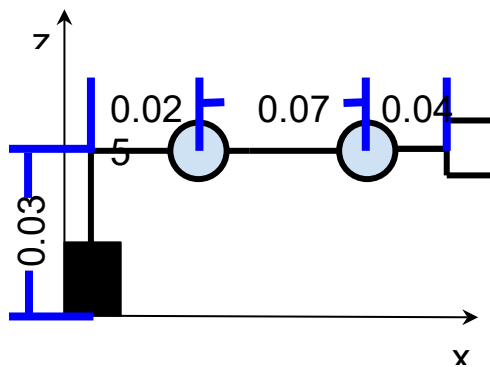
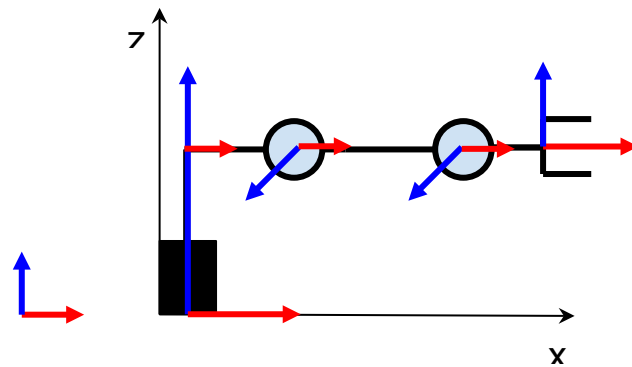
1. ออกแบบแขนกล 3 DOF

- ออกแบบแขนกลโดยโปรแกรม SolidWorks ซึ่งมีการออกแบบให้มี Revolute joint ทั้งหมด 3 ข้อต่อ



Forward Kinematic

:Configuration Variable



- สร้าง DH Parameter เพื่อหาความสัมพันธ์ในแต่ละก้านต่อ

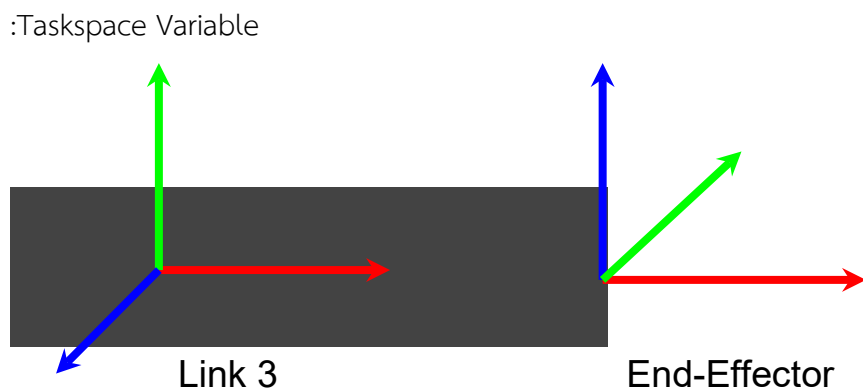
i-1	i	a	alpha	z	theta
0	1	0	0	h1	0
1	2	l1	pi/2	0	0
2	3	l2	0	0	0

Parameter

h1 = 0.030 m

l1 = 0.025 m

l2 = 0.070 m

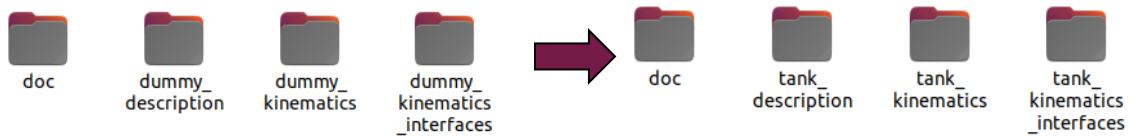


$$H_e^3 = \begin{bmatrix} 1 & 0 & 0 & 0.04 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

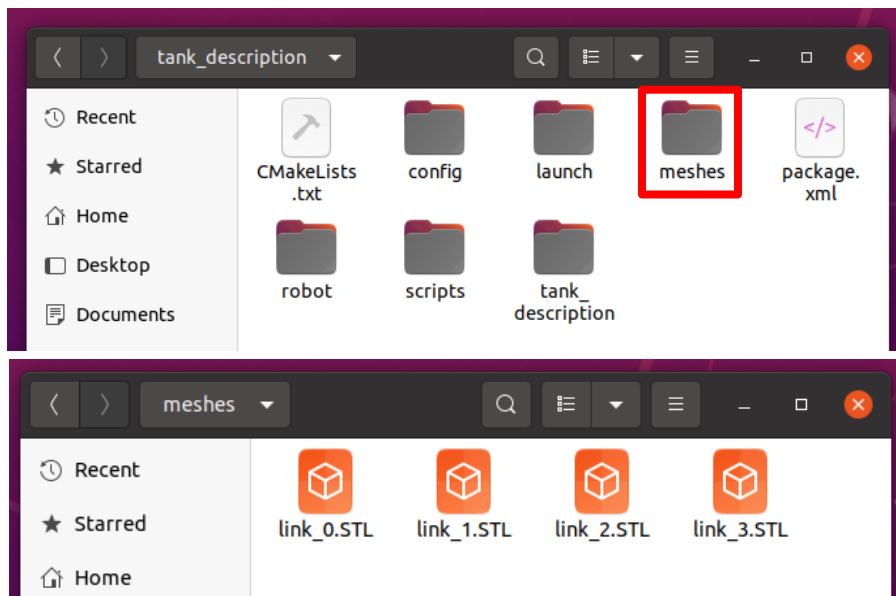
- โดย link ที่ได้ออกแบบทั้งหมดจะมีด้วยกัน 4 link ได้แก่ link_0, link_1, link_2, link_3
- export file เป็น .STL และตั้งค่าไฟล์ให้มีหน่วยเป็น Meter แล้วทำการเลือกเฟรมอ้างอิง

2. การสร้างแบบจำลองจลนศาสตร์

- Download file lab2 และแตกไฟล์
- ทำการเปลี่ยนชื่อ Package จากรูปด้านซ้ายเป็นดังรูปด้านขวา โดยให้ “tank” เป็นชื่อหุ่นยนต์



- เข้าไปใน Package: `tank_description` และทำการสร้าง Folder ชื่อ `meshes` เพื่อไว้เก็บไฟล์ `.STL`



- เปิดไฟล์ `.urdf` ใน Folder: `Robot` และเปลี่ยนชื่อไฟล์เป็น `tank.urdf`
- เปลี่ยนชื่อ robot name จาก `dummy` เป็น `tank`

```
1 <?xml version='1.0' encoding='utf-8'?>
2 <robot name="dummy">
3   <!-- add more codes -->
4 </robot>
```

- ทำการเพิ่ม link โดยกำหนดให้มี
 - base_link, link_0, link_1, link_2, link_3, end_effector
- และทำการเพิ่ม joint เพื่อเชื่อมต่อ link แต่ละก้านเข้าด้วยกัน โดยกำหนดให้มี
 - joint_0 (fixed joint เชื่อม base_link กับ link_0)
 - joint_1 (revolute joint เชื่อม link_0 กับ link_1)
 - joint_2 (revolute joint เชื่อม link_1 กับ link_2)
 - joint_3 (revolute joint เชื่อม link_2 กับ link_3)
 - joint_eff (fixed joint เชื่อม link_3 กับ end_effector)

```

16
17 <link name="base_link">
18 </link>
19
20 <joint name="joint_0" type="fixed">
21   <parent link="base_link"/>
22   <child link="link_0"/>
23   <origin rpy="0 0 0" xyz="0 0 0"/>
24   <axis xyz="0 0 0"/>
25 </joint>
26
27 <link name="link_0">
28   <visual>
29     <origin rpy="0 0 0" xyz="0 0 0"/>
30     <geometry>
31       <mesh filename="package://tank_description/meshes/link_0.STL"/>
32     </geometry>
33     <material name="Grey"/>
34   </visual>
35 </link>
36
37 > <joint name="joint_1" type="revolute">...
43 </joint>
44
45 > <link name="link_1">...
53 </link>
54
55 > <joint name="joint_2" type="revolute">...
61 </joint>
62
63 > <link name="link_2">...
71 </link>
72
73 > <joint name="joint_3" type="revolute">...
79 </joint>
80
81 > <link name="link_3">...
89 </link>
90
91 > <joint name="joint_eff" type="fixed">...
96 </joint>
97
98 <link name="end_effector">
99 </link>
100
101 </robot>
102

```

*joint origin สามารถหาค่าได้จากการทำ DH Table

axis xyz คือ แกนที่ต้องการจะหมุนในที่นี้เราจะหมุนแค่แกน z เท่านั้น

*link mesh filename ให้ใส่ ตำแหน่งของไฟล์ที่จะมาใช้ใน Package นั้น

3. การแสดงผลของแบบจำลอง

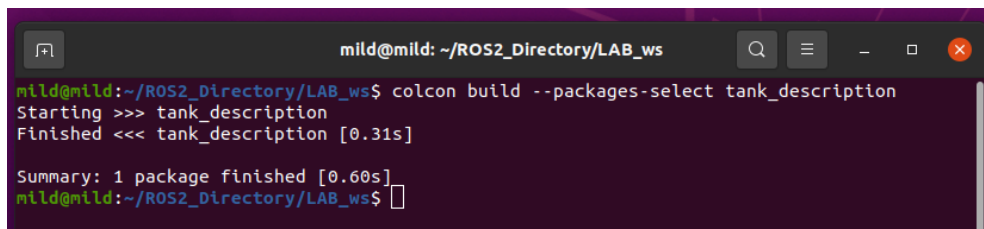
- เปิดไฟล์ display.launch.py ใน Folder: launch
- ทำการเปิดโหนด rviz 2 และ robot_state_publisher
- save file และเปิดไฟล์ CMakeList ของ Package: tank_description
- ทำการแก้ไข dummy_description ในบรรทัดที่ 2 เป็น tank_description

```
1 cmake_minimum_required(VERSION 3.5)
2 project(dummy_description)
```

- ใส่ชื่อ scripts และเพิ่ม directory ที่ต้องการใช้งานโดย
 - ใส่ scripts/ ตามด้วยชื่อไฟล์ scripts ที่ใช้ในบรรทัดที่ 30
 - สามารถเพิ่ม directory ตั้งแต่บรรทัดที่ 38 เสร็จแล้วกด save

```
28 # Install Python executables
29 install(PROGRAMS
30   scripts/tank_script.py
31   DESTINATION ${lib}/${PROJECT_NAME}
32 )
33
34
35 ##### INSTALL LAUNCH, ETC #####
36 install(DIRECTORY
37   # add directories here
38   config
39   launch
40   robot
41   meshes
42   DESTINATION share/${PROJECT_NAME})
```

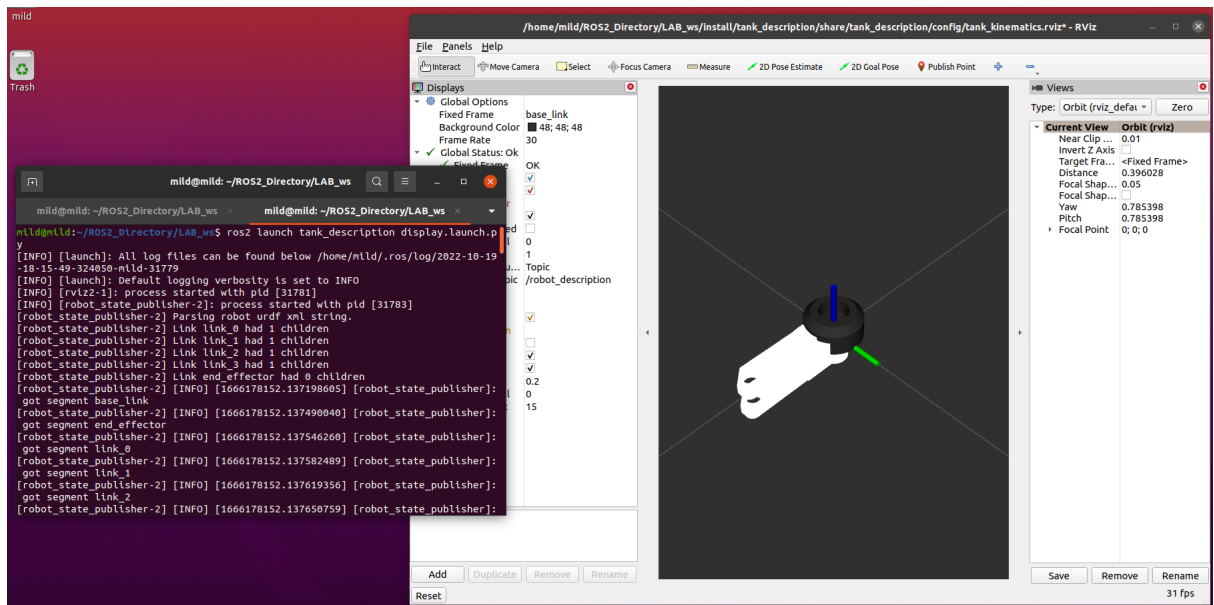
- ทำการ colcon build package tank_description



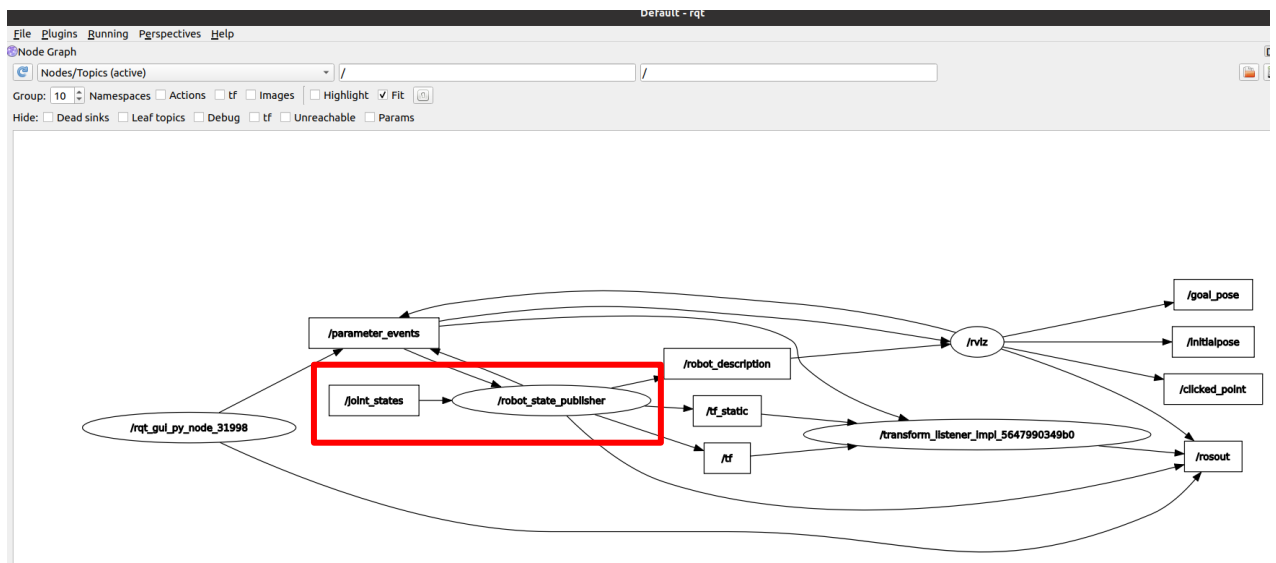
```
mild@mild: ~/ROS2_Directory/LAB_ws
mild@mild:~/ROS2_Directory/LAB_ws$ colcon build --packages-select tank_description
Starting >>> tank_description
Finished <<< tank_description [0.31s]

Summary: 1 package finished [0.60s]
mild@mild:~/ROS2_Directory/LAB_ws$
```

- เปิด terminal และ run ไฟล์ display.launch.py
- ros2 launch tank_description display.launch.py

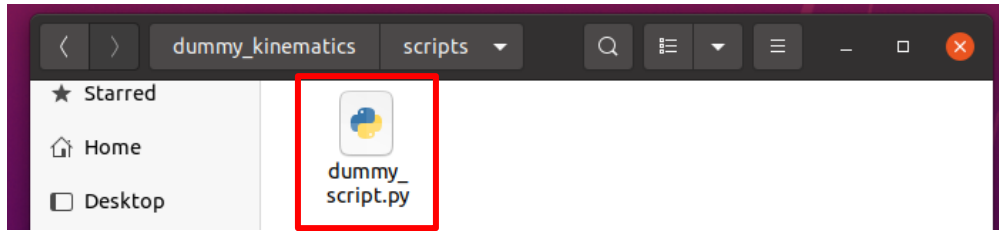


- เปิด rqt ใน terminal ใหม่



4. พัฒนาโหนด kinematics server

- เข้าไปใน Folder: script ที่ Package: tank_kinematics
- และเปลี่ยนชื่อไฟล์ dummy_script.py เป็น kinematics_server.py



- เปิดไฟล์ kinematics_server.py
- จากนั้นใส่ from sensor_msgs.msg import JointState เพื่อนำเข้ามาใช้ใน scripts
- เปิดไฟล์ CMakeList ของ Package: tank_kinematics และเพิ่มบรรทัดที่ 38, 45 ตามรูปภาพ เสร็จแล้ว กด save

```
34
35 find_package(rosidl_default_generators REQUIRED)
36 find_package(geometry_msgs)
37 find_package(std_msgs)
38 find_package(sensor_msgs)
39
40 rosidl_generate_interfaces(${PROJECT_NAME}
41   "srv/SolveIK.srv"
42   "srv/GetPosition.srv"
43   DEPENDENCIES geometry_msgs
44   DEPENDENCIES std_msgs
45   DEPENDENCIES sensor_msgs
46 )
```

- กลับมาใน kinematics_server.py ที่ function __init__ ให้ทำการ create publisher โดยให้ Type: JointState, Topic: /joint_states, Queue size: 10
- และทำการ create timer เพื่อส่งข้อความทุกๆ 10 Hz และทำการเรียก function timer_callback

```
class DummyNode(Node):
    def __init__(self):
        super().__init__('dummy_node')
        self.Publisher = self.create_publisher(JointState, '/joint_states', 10)
        self.rate = 10 #Hz
        timer_period = 1/self.rate #Sec
        self.timer = self.create_timer(timer_period, self.timer_callback)
```

```
def timer_callback(self): #sent name, position, timer
    msg = JointState()
    now = self.get_clock().now()
    msg.header.stamp = now.to_msg()
    msg.name = ['joint_1', 'joint_2', 'joint_3']
    msg.position = self.q
    self.Publisher.publish(msg)
```

- ในที่นี้เราจะส่งค่าตำแหน่งเริ่มต้นเป็นค่า q

```
self.q = [0.,0.,0.] #initial position
```

- ทำการสร้าง service server ใน function `__init__` โดยให้ Type: `GetPosition` , Topic: `/set_joint` และทำเรียกใช้ function `set_join_callback`

```
self.srv_FK = self.create_service(GetPosition,'/set_joints',self.set_join_callback)
```

- ใน function `set_join_callback` เป็นการแปลงค่าตำแหน่งข้อต่อ เป็น ค่าข้อต่อ

```
def set_join_callback(self,request:GetPosition.Request,response:GetPosition.Response):
    self.q = request.joint.position #request joint position
    type_joint = ([1,1,1]) #0: prismatic joint | 1 :revolute joint
    n = 3 #DOF
    H = np.identity(4) #identity 4x4
    DH = np.array([[0,0,0.03,0], [0.025,np.pi/2,0,0],[0.07,0,0,0]]) #DH Table
    H3e = np.array([[0,1,0,0.04], [0,0,1,0],[1,0,0,0],[0,0,0,1]])
    for i in range (n):
        if type_joint[i] == 1: #check type joint if it is revolute joint
            Hj = self.rz(self.q[i]) # let Hj is rotation matrix about z axis
        else: #check type joint if it is prismatic joint
            Hj = self.tz(self.q[i]) # let Hj is translation matrix about z axis
        H = H.dot(self.tx(DH[i][0])).dot(self.rx(DH[i][1])).dot(self.tz(DH[i][2])).dot(self.rz(DH[i][3])).dot(Hj)
    H0e = H.dot(H3e) #pose of end-effector
    response.position.x = H0e[0][3] #position x
    response.position.y = H0e[1][3] #position y
    response.position.z = H0e[2][3] #position z
    return response
```

- เข้าไปใน Folder package `tank_kinematics_interfaces` และเปิดไฟล์ `CMakeLists.txt`
- ใส่ชื่อ “srv/ ตามด้วยไฟล์ srv ที่ใช้” ลงในบรรทัดที่ 41
- จากนั้นทำการเพิ่ม `DEPENDENCIES sensor_msgs` เสร็จแล้วกด save

```
39
40 rosidl_generate_interfaces(${PROJECT_NAME}
41   "srv/SolveIK.srv"
42   "srv/GetPosition.srv"
43   DEPENDENCIES geometry_msgs
44   DEPENDENCIES std_msgs
45   DEPENDENCIES sensor_msgs
46 )
47
```

- เปิดไฟล์ `package.xml` และทำการเปลี่ยนชื่อ ในบรรทัดที่ 4 จาก `dummy_kinematics_interfaces` เป็น `tank_kinematics_interfaces` เสร็จแล้วกด save

```
2 <?xml-model href="http://download.ros.org/schema/package_format_3.xsd" type="application/xml"?>
3 <package format="3">
4   <name>dummy_kinematics_interfaces</name>
5   <version>0.0.0</version>
```


- จากนั้นเปิดไฟล์ GetPosition.srv เพื่อทำการใส่ request และ response ของ service
- request (Input คือ บรรทัดก่อน - - -)

ประกาศตัวแปร joint โดยให้ Type: sensor_msgs/JointState

- response (Output คือ บรรทัดหลัง - - -)

ประกาศตัวแปร position โดยให้ Type: geometry_msgs/Point

```
1 # add request
2 sensor_msgs/JointState joint
3 ---
4 geometry_msgs/Point position
5 # add response
```

- ต่อมาเปิดไฟล์ display_with_kinematics_srv.launch.py
- ทำการ launch file display.launch.py และ run kinematics_server.py

```
1 #!/usr/bin/env python3
2 from launch import LaunchDescription
3 from launch_ros.actions import Node
4 from launch.substitutions import FindPackageShare
5 from launch.actions import IncludeLaunchDescription
6 from launch.substitutions import PathJoinSubstitution
7 from launch.launch_description_sources import PythonLaunchDescriptionSource
8
9 import sys
10
11
12 def generate_launch_description():
13
14     ### How to launch another launch file ###
15     #
16     # You must specify the package, folder, and the name
17     #
18     display = IncludeLaunchDescription(
19         PythonLaunchDescriptionSource([
20             PathJoinSubstitution([
21                 FindPackageShare('tank_description'),
22                 'launch',
23                 'display.launch.py'
24             ])
25         ])
26     )
27     kinematics_server = Node(
28         package='tank_kinematics',
29         executable='kinematics_server.py',
30         name='tank_kinematics'
31         # output='screen'
32     )
33
34     # Launch Description
35     launch_description = LaunchDescription()
36     launch_description.add_action(display)
37     launch_description.add_action(kinematics_server)
38
39     return launch_description
40
41 def main(args=None):
42     try:
43         generate_launch_description()
44     except KeyboardInterrupt:
45         # quit
46         sys.exit()
```

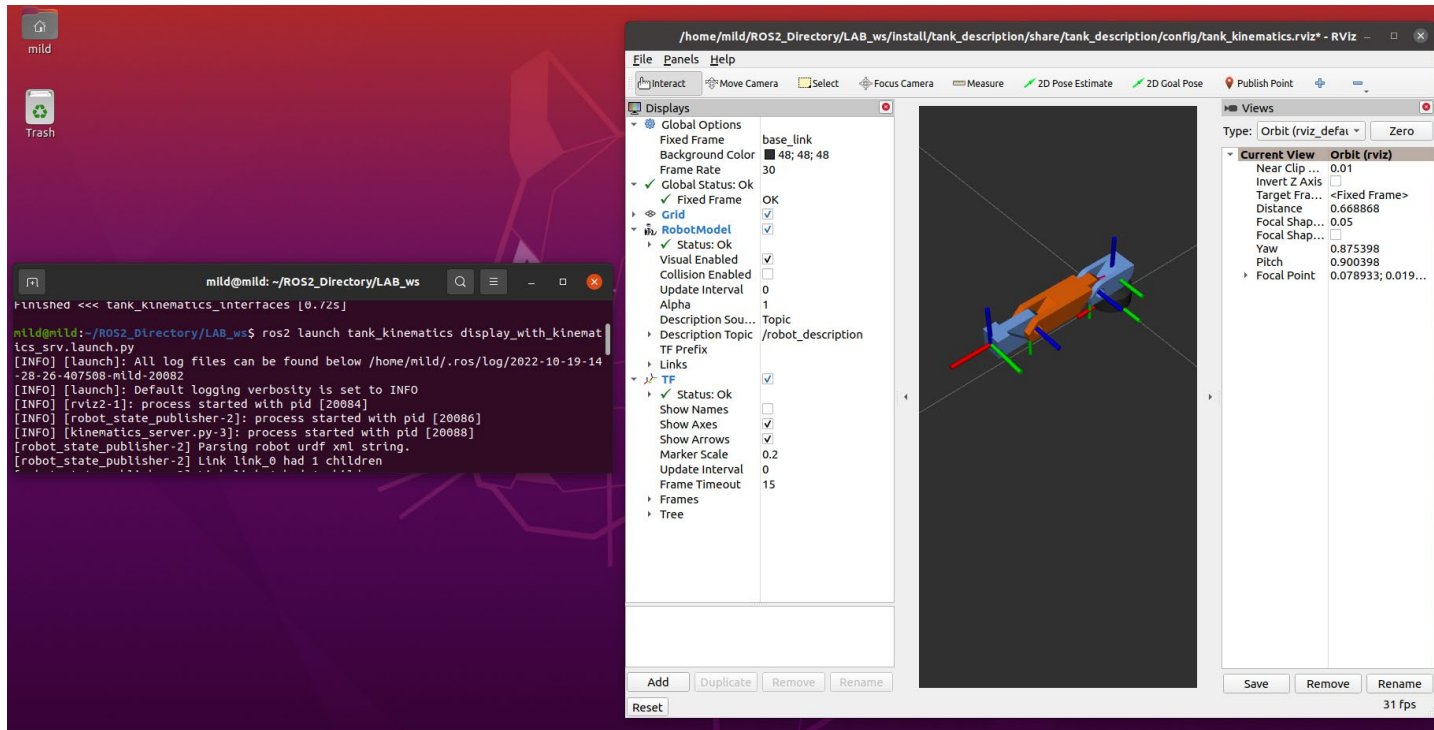
- เสร็จแล้วเปิด Terminal ทำการ colcon build ที่ workspace

```
mild@mild: ~/ROS2_Directory/LAB_ws
mild@mild:~/ROS2_Directory/LAB_ws$ colcon build
Starting >>> tank_description
Starting >>> tank_kinematics
Starting >>> tank_kinematics_interfaces
Finished <<< tank_description [0.34s]

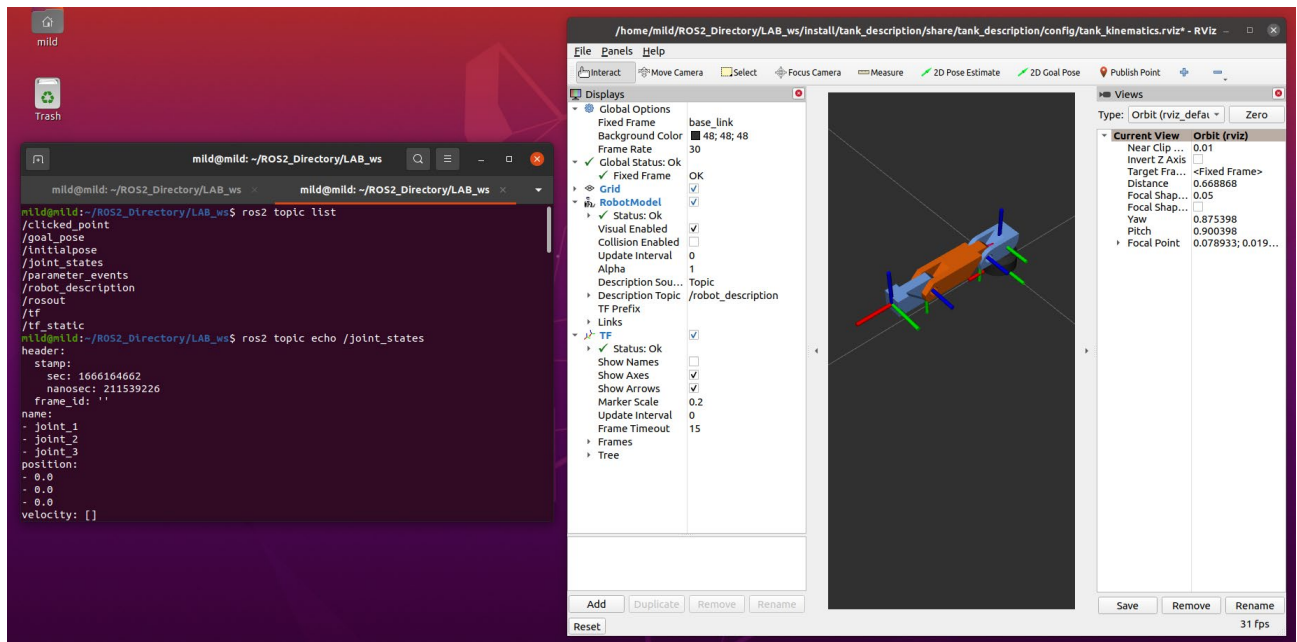
Finished <<< tank_kinematics [0.34s]
Finished <<< tank_kinematics_interfaces [0.72s]

Summary: 3 packages finished [1.02s]
mild@mild:~/ROS2_Directory/LAB_ws$ source install/setup.bash
mild@mild:~/ROS2_Directory/LAB_ws$
```

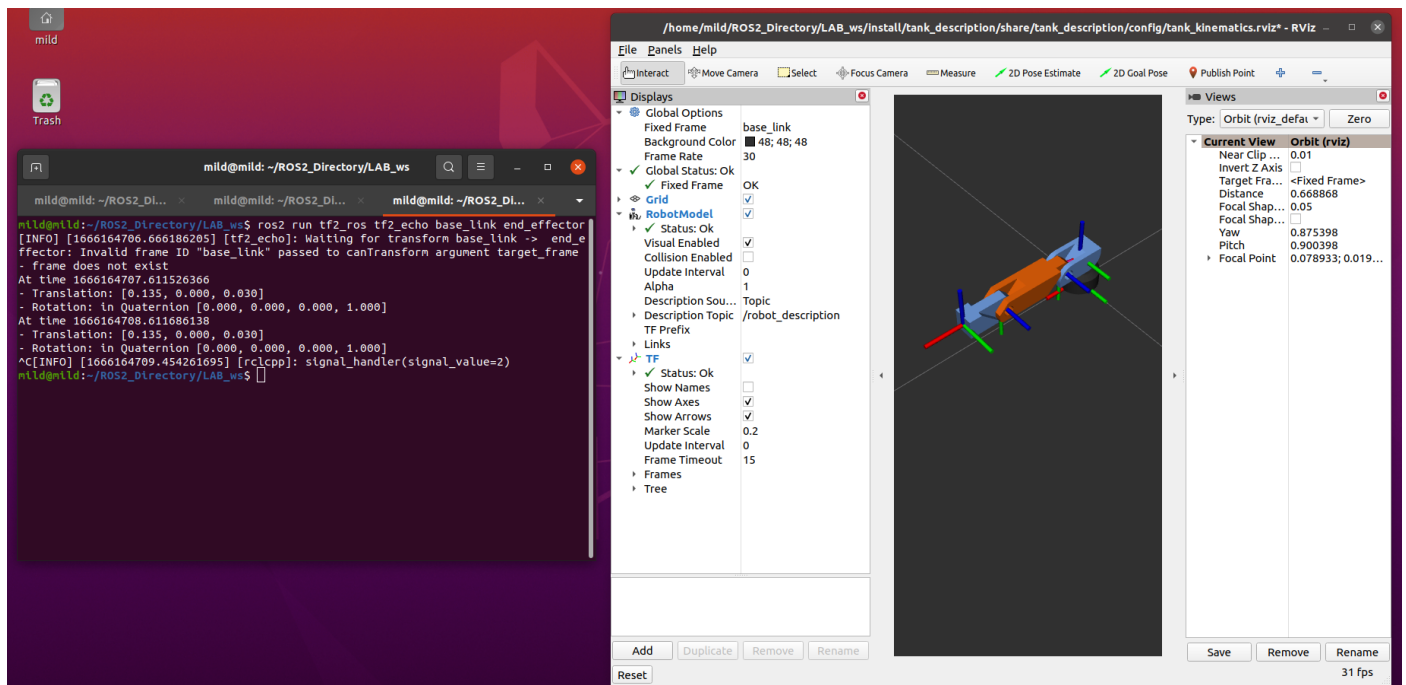
- จากนั้นทำการ launch file display_with_kinematics_srv.launch.py
- ros2 launch tank_kinematics display_with_kinematics_srv.launch.py



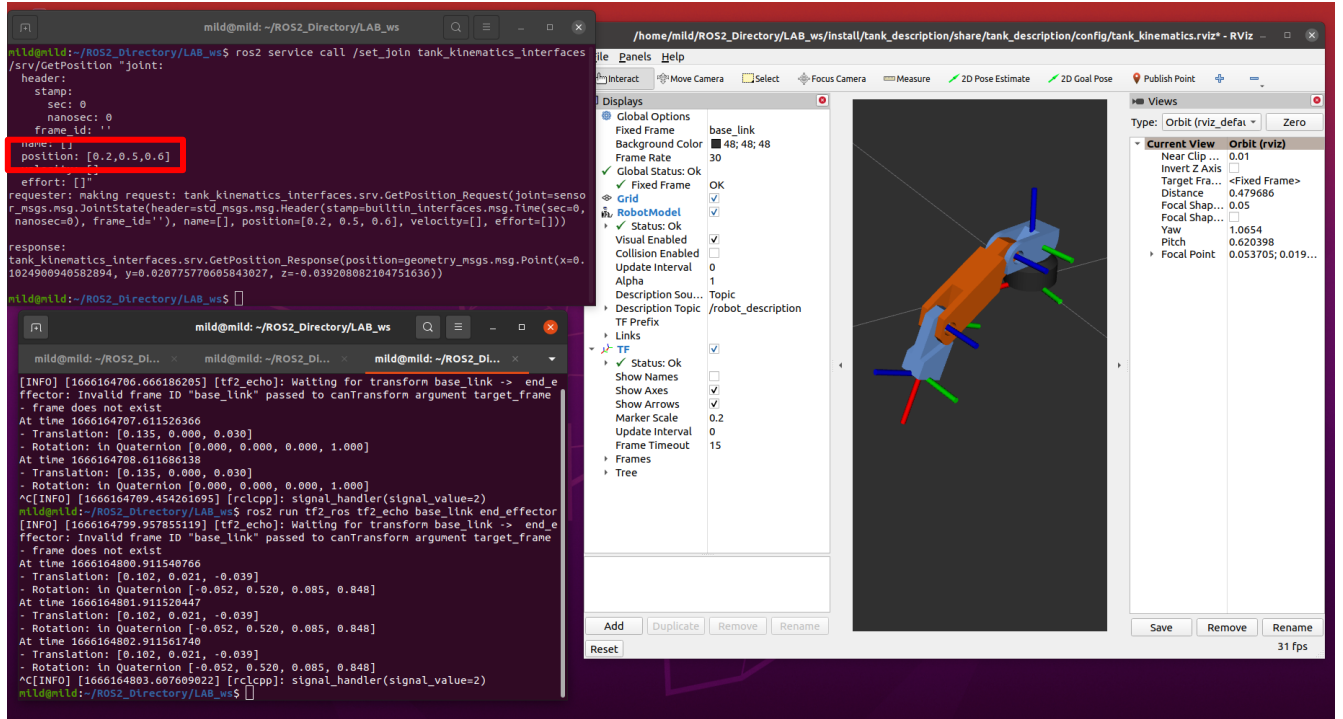
- และทำการเปิด Terminal เพิ่มอีก Tab เพื่อเอาไว้ดู topic /joint_states
- ros2 topic echo /joint_states



- และทำการเปิด Terminal เพิ่มอีก Tab เพื่อเอาไว้ดู ตำแหน่ง end-effector เทียบกับ base link
- ros2 run tf2_ros tf2_echo base_link end_effector (Translation)

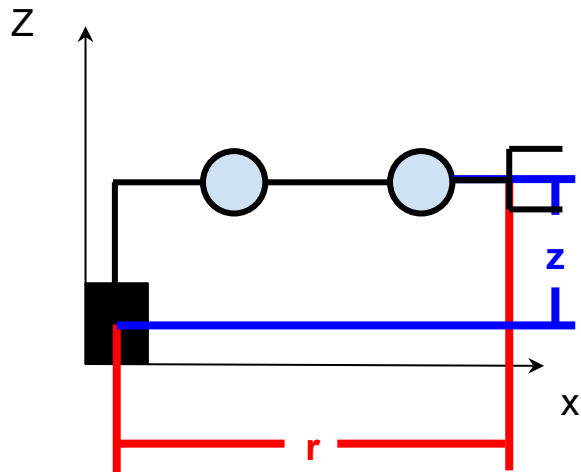


- และทำการเปิด Terminal เพิ่มอีก Tab เพื่อ Call Service
- ros2 service call /set_join tank_kinematics_interfaces/srv/GetPosition "joint:"
- จากนั้นใส่ค่าตำแหน่งของ end-effector ที่ต้องการจะไป ที่ position นั้น



Inverse Kinematic

:Home Configuration v



Find : $q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$

Taskspace : $\chi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$