DayServiceRBの運動プログラムで制御

　目標は、デイサービスで働くロボット。rvizで表示したDayServiceRBをプログラムで制御する。

構成

・T8$　　Client PC CF-T8　　Debian10
・noetic3$　　T8 Docker Container

目次

python スクリプト

当初のプログラム

noetic3 yo@T8:~$ vi ~/catkin_ws/src/my_robot/scripts/sin_cos_joint_control.py

```
#!/usr/bin/env python3
import rospy
from sensor_msgs.msg import JointState
```

```
import math
import time

def main():
    # ROSノード初期化
    rospy.init_node("sin_cos_joint_control")

    # パブリッシャ作成
    joint_pub = rospy.Publisher("/joint_states", JointState,
queue_size=10)

    # ジョイントステートメッセージの初期設定
    joint_state = JointState()
    joint_state.name = ["joint1", "joint2"]  # ジョイント名
    joint_state.position = [0.0, 0.0]        # 初期角度 [joint1, joint2]

    # 周期 (0.5Hz)
    frequency = 0.5  # Hz
    period = 1.0 / frequency  # 秒
    rate = rospy.Rate(50)  # 50Hzでループ実行

    start_time = time.time()  # 開始時刻

    while not rospy.is_shutdown():
        # 経過時間を取得
        elapsed_time = time.time() - start_time

        # 角度をsin/cos波で更新 (周期: 0.5Hz)
        joint_state.header.stamp = rospy.Time.now()  # タイムスタンプを
更新
        joint_state.position[0] = math.sin(2 * math.pi * frequency *
elapsed_time)  # joint1: sin波
        joint_state.position[1] = math.cos(2 * math.pi * frequency *
elapsed_time)  # joint2: cos波

        # ジョイント状態をパブリッシュ
        joint_pub.publish(joint_state)
```

```
        # 指定したレートでスリープ
        rate.sleep()

if __name__ == "__main__":
    try:
        main()
    except rospy.ROSInterruptException:
        pass
```

修正プログラム

noetic3 yo@T8:~$ vi ~/catkin_ws/src/day_service_rb/scripts/sin_cos_joint_control.py

```
#!/usr/bin/env python3
import rospy
from sensor_msgs.msg import JointState
import math
import time

def main():
    # ROSノード初期化
    rospy.init_node("sin_cos_joint_control")

    # パブリッシャ作成
    joint_pub = rospy.Publisher("/joint_states", JointState,
queue_size=10)

    # ジョイントステートメッセージの初期設定
    joint_state = JointState()
    joint_state.name = ["joint1", "joint2" , "joint3", "joint4"]  # ジョイント
名
    joint_state.position = [0.0, 0.0, 0.0, 0.0]        # 初期角度 [joint1,
joint2]

    # 周期 (0.5Hz)
    frequency = 0.5  # Hz
    period = 1.0 / frequency  # 秒
    rate = rospy.Rate(50)  # 50Hzでループ実行
```

```
    start_time = time.time()  # 開始時刻

    while not rospy.is_shutdown():
        # 経過時間を取得
        elapsed_time = time.time() - start_time

        # 角度をsin/cos波で更新 (周期: 0.5Hz)
        joint_state.header.stamp = rospy.Time.now()  # タイムスタンプを
更新
        joint_state.position[0] = math.sin(2 * math.pi * frequency *
elapsed_time)  # joint1: sin波
        joint_state.position[1] = math.cos(2 * math.pi * frequency *
elapsed_time)  # joint2: cos波
        joint_state.position[2] = math.sin(2 * math.pi * frequency *
elapsed_time)  # joint1: sin波
        joint_state.position[3] = math.cos(2 * math.pi * frequency *
elapsed_time)  # joint2: cos波

        # ジョイント状態をパブリッシュ
        joint_pub.publish(joint_state)

        # 指定したレートでスリープ
        rate.sleep()

if __name__ == "__main__":
    try:
        main()
    except rospy.ROSInterruptException:
        pass
```

lanchファイルの作成

noetic3 yo@T8:~$ vi ~/catkin_ws/src/day_service_rb/launch/display_program.launch

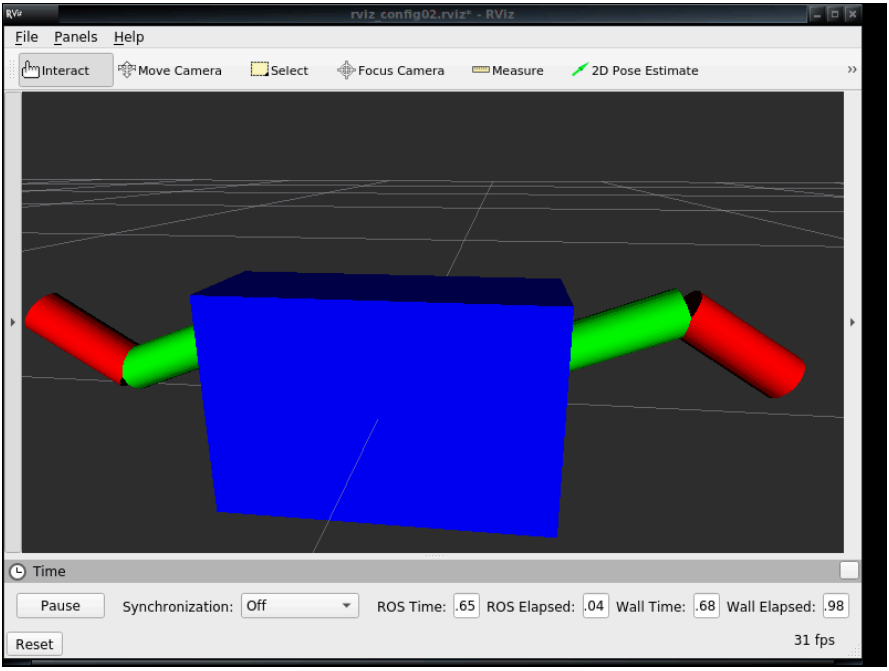
```
<launch>
  <!-- ロボットのURDFをロード -->
```

```
  <param name="robot_description" command="cat $(find
day_service_rb)/urdf/day_service_rb.urdf" />

  <!-- ジョイントステートパブリッシャー -->
   <node name="sin_cos_joint_control" pkg="day_service_rb"
type="sin_cos_joint_control.py" />

  <!-- ロボットモデルを表示 -->
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher" />

  <!-- RViz起動 -->
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
day_service_rb)/config/rviz_config02.rviz" />
  <!-- node name="rviz" pkg="rviz" type="rviz"  / -->
</launch>
```

実行手順

launchファイルの実行

noetic3 yo@T8:~$ roslaunch day_service_rb display_program.launch



今後の予定

7. servoを使った機体のデザイン決定
8. day_service_rbの動きを作る
9. 実際の機体の作成
10. ros noeticで構成されるプログラムの起動をroslaunchで纏める
・全てを一つのlanchファイルで構成するのではなく、各PC毎にlaunchファイルを作成する。
11. 改善点多数、全体の流れを再考する
・ggmlの起動について：反応しすぎるのはどうかと。「えっと」を削除する。

参考

20250129noeticでrvizに表示したRBをプログラムで制御