Lab 5 Procedure:
Robot Manipulator

Preparation

List of components

- IBM PC with Linux
- Beaglebone set: Beaglebone embedded board, USB cable, 1 ethernet cable, 4(or More) GB microSD card (with debian Linux image in it)
- A 4 DOF manipulator
- DYNAMIXEL Starter Set
- Manipulator Control skeleton code

As the control of the manipulator is an extension of the motor control, please **review the Lab 3 material**, especially Development Environment Setup and the Articulated system class.

Problems

Problem 5A. Direct Teaching (week 1)

The first task is direct teaching which makes the manipulator follow three waypoints repetitively. The waypoint is defined as **joint angles** so its dimension is **four** which is the DOF of the manipulator. The first waypoint is the initial joint angles of the 4 DOF manipulator when the program runs, and the second and third waypoints are defined by the user. To perform the task, first, implement a code that saves two user-defined waypoints in "waypoints.csv" file. Next, using trajectory generation based on the first-order polynomial, compute the desired joint angle for each motor and design the joint space P position controller using velocity mode.

- Development setup
- Save two user-defined waypoints in joint space
- Generate trajectory
- Joint space P position control using velocity mode

Problem 5B. Teleoperation (week 2)

The second task is teleoperation which controls the end-effector position based on the keyboard input. To this end, first, get the task space command from the keyboard input. Next, compute the Jacobian matrix for the world linear velocity. Then, find the desired joint angles via inverse kinematics. Finally, design the joint space P position controller using velocity mode.

- Development setup
- Get task space command from the keyboard input
- Compute the Jacobian matrix
- Compute the desired joint angles using inverse kinematics
- Joint space P position control using velocity mode

Lab Procedure

Problem 5A. Direct Teaching (week 1)

<Development setup>

1. Turn on the development PC.
2. Move the skeleton code(**EE405_ManipulatorControl**) for manipulator control to your home directory on the development PC.
3. Turn on the BeagleBone Black using SD card.
4. Set up the NFS

---

1. Configure NFS on development PC

Remember IPs for PC and Bone, for example,

PC: 192.168.7.1

Beaglebone: 192.168.7.2

Edit /etc/exports to INCLUDE hosts allowed to connect (i.e., IP of Beaglebone).

$ sudo nano /etc/exports

```
# /etc/exports: the access control list for filesystems which may be exported
#         to NFS clients.         See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
```

```
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#

# nfs for Bone Ubuntu - Robot Manipulator
/home/<your computer's name>/EE405_ManipulatorControl/build 192.168.7.2(rw,sync,no_root_squash,no_subtree_check)
```

Whenever we modify /etc/exports, we must run 'exportfs' to make the changes effective afterwards.

$ sudo exportfs -a

2. Start PC NFS server

To start the NFS server, you can run the following command at a terminal prompt:

$ sudo /etc/init.d/nfs-kernel-server start

3. Make the mount point on BeagleBone Black

mkdir ~/nfs_client

Note. There should be no files or subdirectories in the ~/nfs_client directory.

4. Start Beaglebone nfs client

command: sudo mount {pc_ip}:{pc_directory} {beaglebone_directory}

cd ~
sudo mount 192.168.7.1:/home/<your computer's name>/EE405_ManipulatorControl/build ~/nfs_client

Check your PC IP!

---

5. (Install the Dynamixel SDK Shared Library On BeagleBone Black. **If the shared library of dynamixel SDK is already located in /usr/lib of the BeagleBone Black, you can skip it.**) Dynamixel SDK shared library has already been built for arm-based CPU, so we only need to copy the shared library of dynamixel SDK to /usr/lib of the BeaglBone Black.

   [On the Development PC]

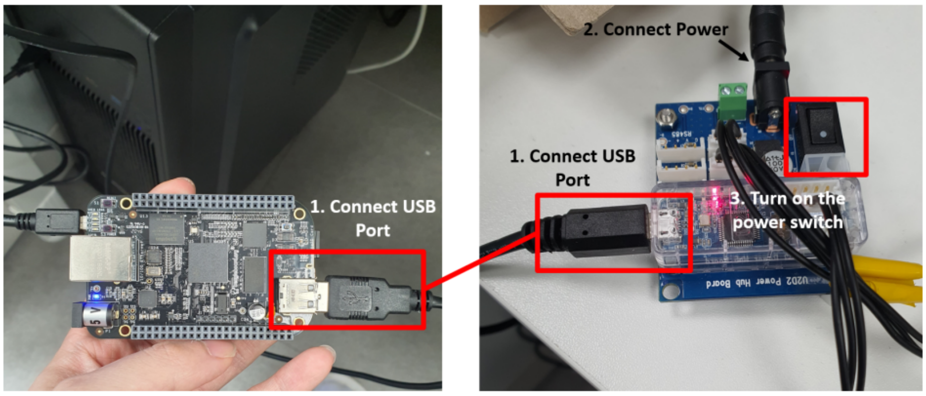
   $ cd ~/EE405_ManipulatorControl/lib/dynamixel_sdk

   $ sudo scp libdxl_sbc_cpp.so debian@192.168.7.2:/home/debian

   [On BeagleBone Black]

   debian@beaglebone$ sudo cp ~/libdxl_sbc_cpp.so /usr/lib

6. Connect BeagleBone Black with dynamixel U2D2. Supply power to U2D2.



7. Change the latency timer from 16ms to 1ms on BeagleBone Black.

debian@beaglebone:~$ echo 1 | sudo tee /sys/bus/usb-serial/devices/ttyUSB0/latency_timer

[Note!] Whenever you reconnect the Dynamixel U2D2 and BeagleBone Black with the serial communication cable, you must enter this command to set the latency timer.

8. Open the vscode on development PC.

   $ cd ~/EE405_ManipulatorControl

   $ code .

<Save two user-defined waypoints in joint space>

9. Open the "**save_waypoints.cpp**" file in EE405_ManipulatorControl/**src** directory.
10. Check the **39th** line. The manipulator for our lab has 4 DOF. Hence, **dof=4.**
11. Implement the code on **the 72nd,** and **79th lines** to save two user-defined waypoints in "waypoints.csv" file. We will use getch() function to get the keyboard input and obtain the joint angle of each motor using **GetJointAngle()** which is defined in "ArticulatedSystem.hpp". The following instruction explains the keyboard input in detail.
    - Change the configuration of the manipulator to define the first user-defined waypoint, and press '1' to save this waypoint. Similarly, change the configuration of the manipulator to define the second user-defined waypoint, and press '2' to save this waypoint. To modify the waypoints freely, the user can redefine the waypoints by typing in '1' or '2' repetitively. To finish saving the waypoints, press 'e'.

12. Build the project using vs code button.

- Set the compiler as **GCC arm-linux-gnueabihf.**
- Set the compile mode as **Release mode. (Do not set Debug mode.)**

Now, you can build the project by either clicking on the build button or pressing the F7 shortcut key.

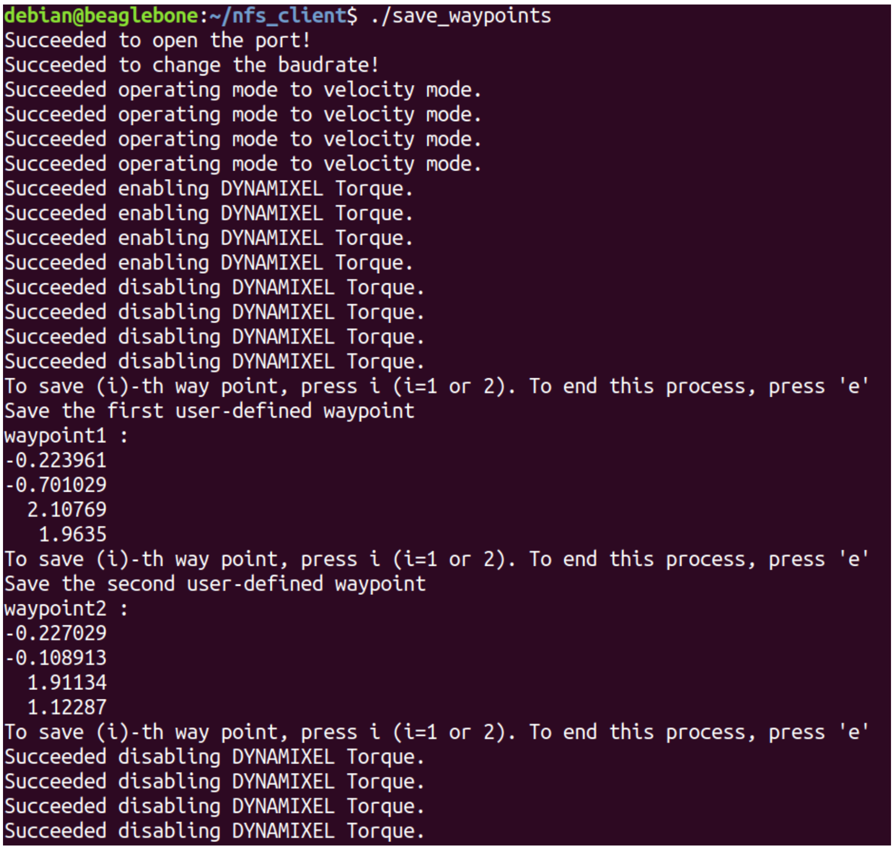
13. Run this code on the BeagleBone.

debian@beaglebone:~$ cd ~/nfs_client

debian@beaglebone:~/nfs_client$ ./save_waypoints

14. Save two user-defined waypoints by referring to the following instructions.

    - Change the configuration of the manipulator to define the first user-defined waypoint, and press '1' to save this waypoint. Similarly, change the configuration of the manipulator to define the second user-defined waypoint, and press '2' to save this waypoint. To modify the waypoints freely, the user can redefine the waypoints by typing in '1' or '2' repetitively. To finish saving the waypoints, press 'e'.
    - The following figure shows an example of the result.



```
debian@beaglebone:~/nfs_client$ ./save_waypoints
Succeeded to open the port!
Succeeded to change the baudrate!
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
To save (i)-th way point, press i (i=1 or 2). To end this process, press 'e'
Save the first user-defined waypoint
waypoint1 :
-0.223961
-0.701029
  2.10769
   1.9635
To save (i)-th way point, press i (i=1 or 2). To end this process, press 'e'
Save the second user-defined waypoint
waypoint2 :
-0.227029
-0.108913
  1.91134
  1.12287
To save (i)-th way point, press i (i=1 or 2). To end this process, press 'e'
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
```

15. After saving the two user-defined waypoints, check "waypoints.csv" file in the build directory. If there is no "waypoints.csv" file in the build directory, run the following commands and repeat 13~14.

    [On the Development PC]

    $ cd ~/EE405_ManipulatorControl

    $ sudo chmod 777 build

16. The following is an example of "waypoints.csv" file. Here, the first user-defined waypoint (=joint angle) is [-0.223961195031304, -0.701029220063738, 2.10768960255487, 1.96349540849362], and the second user-defined waypoint (=joint angle) is [-0.227029156607075, -0.10891263593988, 1.91134006170551, 1.12287393673229].

    waypoints.csv

    build > waypoints.csv

```
-0.223961195031304, -0.701029220063738, 2.10768960255487, 1.96349540849362
-0.227029156607075, -0.10891263593988, 1.91134006170551, 1.12287393673229
```

<Generate trajectory>

17. Open the "**manipulator_control_week1.cpp**" file in EE405_ManipulatorControl/**src** directory
18. Check the **59th, 62nd, and 80th** lines. The manipulator for our lab has 4 DOF and the control frequency is 100Hz. To control the joint angles of the manipulator, we will utilize velocity mode. Since DOF is 4, **the dimensions of targetQ (89th line) and targetQdot (90th line) are 4** in the code.
19. Open the "**DirectTeaching.cpp**" file in **EE405_ManipulatorControl/include/Common** directory.
20. To compute the desired joint angles for direct teaching, **initialization(current_q)** function and **TrajectoryGeneration()** function are needed. The roles of two functions are described as follows.
    - **initialization(current_q)** : Define and print stack_waypoints where each row represents a waypoint for direct teaching in joint space. **The first waypoint for direct teaching is the initial joint angle when the program runs. The second and third waypoint are two user-defined waypoints which are defined in "waypoints.csv" file**. The dimension of stack_waypoints is 3 by 4.
    - **TrajectoryGeneration()** : Using the first-order polynomial, generate the trajectory

which passes **'the first waypoint'** → **'the second waypoint'** → **'the third waypoint'** → **'the first waypoint'** → **'the second waypoint'** → **...** repetitively. The number of nodes between two waypoints is defined as NumNodes. This function returns the desired joint angles of the four motors.

21. Implement the code on **the 51st line**. Recall that the result (=C) of the first-order polynomial interpolation between point A and point B can be represented as follows; **C=(B-A)/NumNodes*NodeIndex+A** where NumNodes is the number of nodes between two waypoints (= A and B) and the range of NodeIndex is 0 ~ (NumNodes-1). To complete the code, use the following variables (NumNodes, NodeIndex, StartPointIndex, EndPointIndex).

```
include > Common > DirectTeaching.cpp
DirectTeaching::DirectTeaching(int dof_args)
{
    NumWaypoints=3; // the number of waypoints
    dof=dof_args; // DOF of the manipulator
    stack_waypoints.resize(NumWaypoints, dof); // Each row in the stack_waypoints matrix represents a waypoint for direct teaching

    NumNodes=400; // the number of nodes
    NodeIndex=0; // node index. The range of NodeIndex is 0 ~ (NumNodes-1)
    StartPointIndex=0; // Waypoint index of start point in trajectory generation. The range of StartPointIndex is 0 ~ (NumWaypoints-1)
    EndPointIndex=1; // Waypoint index of end point in trajectory generation. The range of EndPointIndex is 0 ~ (Numwaypoints-1)

}
```

22. Implement the code on **the 62nd, 65th, and 68th lines** using the NodeIndex, StartPointIndex, and EndPointIndex ranges.

<Joint space P position control using velocity mode>

23. Open the "**manipulator_control_week1.cpp**" file in EE405_ManipulatorControl/**src** directory
24. Check the **110th line.** The stack_waypoints matrix is defined by the **initialization** function.
25. Check the **128th line**. The targetQ is computed by the **TrajectoryGeneration** function.
26. Now we will implement a joint space P position controller using velocity mode. This controller can be represented as follows; **targetQdot=-Kp*(current_q-target_q)**. Here, current_q is the current joint angles, target_q is the target joint angles, and Kp is the P gain. Implement this on the **136th line**.
27. Uncommand the **25th and 39 ~ 42nd lines** in CMakeLists.txt file
28. Uncommand the **111th line** in "**ArticulatedSystem.cpp**" file in **EE405_ManipulatorControl/include/Controller** directory.
29. Build the project using vs code button.
30. Now, you can check the executable file "maniulator_control_week1" in the build directory. If you execute this file, direct teaching will start. The manipulator will pass the following trajectory repetitively; **'the first waypoint'** → **'the second waypoint'** → **'the third**

waypoint' → **'the first waypoint'** → **'the second waypoint'** → **...** . Since the number of nodes between waypoints is 400, it takes 4 (=400/100) seconds to move from one waypoint to the next waypoint. Therefore, please make sure that the waypoints are not far from each other. Note that you can modify the waypoints by executing the save_waypoints file. If you want to increase the time it takes to move between two waypoints, increase **NumNodes**.

31. Run "manipulator_control_week1" on the BeagleBone. (If you want to terminate the process, press **Ctrl + C** once.)

debian@beaglebone:~$ cd ~/nfs_client

debian@beaglebone:~/nfs_client$ ./manipulator_control_week1

32. Check that direct teaching works well. The following shows an example of the result.

```
debian@beaglebone:~/nfs_client$ ./manipulator_control_week1
Succeeded to open the port!
Succeeded to change the baudrate!
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Each row represents a waypoint for direct teaching in joint space:
-0.220893  -1.32383    1.1919     0.948
-0.223961 -0.701029   2.10769    1.9635
-0.227029 -0.108913   1.91134   1.12287
^CInterrupt signal (2) received.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
```

Problem 5B. Teleoperation (week 2)

<Development setup>

33. Turn on the development PC.
34. Move the skeleton code(**EE405_ManipulatorControl**) for manipulator control to your home directory on the development PC.
35. Turn on the BeagleBone Black using SD card.
36. Set up the NFS

---

1. Configure NFS on development PC

Remember IPs for PC and Bone, for example,

PC: 192.168.7.1

Beaglebone: 192.168.7.2

Edit /etc/exports to INCLUDE hosts allowed to connect (i.e., IP of Beaglebone).

$ sudo nano /etc/exports

```
# /etc/exports: the access control list for filesystems which may be exported
#        to NFS clients.        See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#

# nfs for Bone Ubuntu - Robot Manipulator
/home/<your computer's name>/EE405_ManipulatorControl/build 192.168.7.2(rw,sync,no_root_squash,no_subtree_check)
```

Whenever we modify /etc/exports, we must run 'exportfs' to make the changes effective afterwards.

$ sudo exportfs -a

2. Start PC NFS server

To start the NFS server, you can run the following command at a terminal prompt:

$ sudo /etc/init.d/nfs-kernel-server start

3. Make the mount point on BeagleBone Black

mkdir ~/nfs_client

Note. There should be no files or subdirectories in the ~/nfs_client directory.

4. Start Beaglebone nfs client

command: sudo mount {pc_ip}:{pc_directory} {beaglebone_directory}

cd ~
sudo mount 192.168.7.1:/home/<your computer’s name>/EE405_ManipulatorControl/build ~/nfs_client

Check your PC IP!

---

37. (Install the Dynamixel SDK Shared Library On BeagleBone Black. **If the shared library of dynamixel SDK is already located in /usr/lib of the BeagleBone Black, you can skip it.**) Dynamixel SDK shared library has already been built for arm-based CPU, so we only need to copy the shared library of dynamixel SDK to /usr/lib of the BeaglBone Black.

    [On the Development PC]

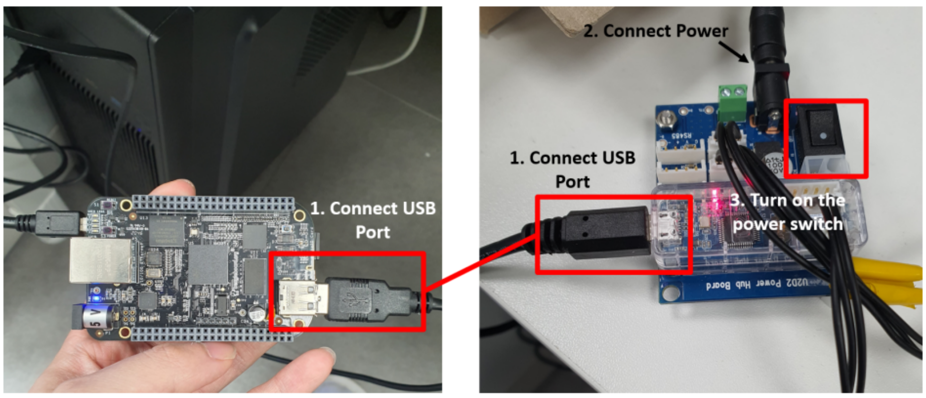
    $ cd ~/EE405_ManipulatorControl/lib/dynamixel_sdk

    $ sudo scp libdxl_sbc_cpp.so debian@192.168.7.2:/home/debian

    [On BeagleBone Black]

    debian@beaglebone$ sudo cp ~/libdxl_sbc_cpp.so /usr/lib

38. Connect BeagleBone Black with dynamixel U2D2. Supply power to U2D2.



39. Change the latency timer from 16ms to 1ms on BeagleBone Black.

debian@beaglebone:~$ echo 1 | sudo tee /sys/bus/usb-serial/devices/ttyUSB0/latency_timer

[Note!] Whenever you reconnect the Dynamixel U2D2 and BeagleBone Black with the

serial communication cable, you must enter this command to set the latency timer.

40. Open the vscode on development PC.

    $ cd ~/EE405_ManipulatorControl

    $ code .

<Get task space command from the keyboard input>

41. Open the “**manipulator_control_week2.cpp**” file in EE405_ManipulatorControl/**src** directory
42. Check the **83rd, 86th, and 104th** lines. The manipulator for our lab has 4 DOF and the control frequency is 100Hz. To control the joint angles of the manipulator, we will utilize velocity mode.
43. The **GetKeyboardInput() function** computes the task command from the keyboard input for teleoperation. The following keyboard inputs determine the task command. The position resolution for teleoperation is set to 1cm. You can change this by modifying the **position_resolution** variable. When the new keyboard input is received, is_key_upated is set to true.
    - r : move 1cm in the world z-axis
    - f : move -1cm in the world z-axis
    - w : move 1cm in the world y-axis
    - s : move -1cm in the world y-axis
    - d : move 1cm in the world x-axis
    - a : move -1cm in the world x-axis
    - i : move to the initial position
    - e : end the teleoperation
44. Implement the code on the **62nd, 65th, 68th, and 71st** lines based on the keyboard input.
45. Check the **80th, and 208th** lines. We create thread t1 to control the end effector position while receiving keyboard input.

<Compute the Jacobian matrix>

46. Open the “**Kinematics.cpp**” file in **EE405_ManipulatorControl/include/Common** directory.
47. To compute the Jacobian matrix for the world linear velocity, **GetRotationMatrix(r, p, y)** and **GetJacobianMatrix(current_q)** functions are needed. **GetRotationMatrix function** returns the following successive rotation; rotation of y (rad) along the body z-axis -> rotation

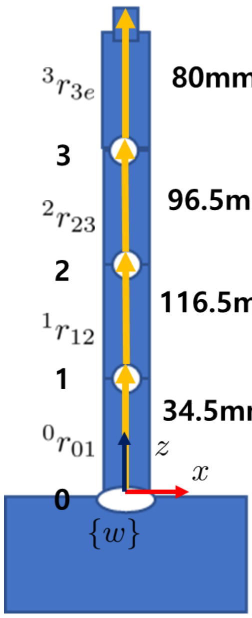
of p (rad) along the body y-axis -> rotation of r (rad) along the body x-axis. **GetJacobianMatrix function** returns the Jacobian matrix for the world linear velocity. **Note that the dimension of this Jacobian matrix is 3 by 4.**

48. Complete **GetJacobianMatrix**() function using the following equations. The cross product of two vectors (= a × b) can be computed by a.cross(b) in the Eigen library.

$$
\begin{aligned}
{}^{w}r_{3e} &= {}^{w}R_3\,{}^{3}r_{3e} \\
{}^{w}r_{2e} &= {}^{w}r_{3e} + {}^{w}R_2\,{}^{2}r_{23} \\
{}^{w}r_{1e} &= {}^{w}r_{2e} + {}^{w}R_1\,{}^{1}r_{12} \\
{}^{w}r_{0e} &= {}^{w}r_{1e} + {}^{w}R_0\,{}^{0}r_{01}
\end{aligned}
$$

$$
\begin{aligned}
{}^{w}V_e &= {}^{w}\Omega_0 \times {}^{w}r_{0e} + {}^{w}\Omega_1 \times {}^{w}r_{1e} + {}^{w}\Omega_2 \times {}^{w}r_{2e} + {}^{w}\Omega_3 \times {}^{w}r_{3e} \\
&= \left[{}^{w}P_0 \times {}^{w}r_{0e} \quad {}^{w}P_1 \times {}^{w}r_{1e} \quad {}^{w}P_2 \times {}^{w}r_{2e} \quad {}^{w}P_3 \times {}^{w}r_{3e}\right] \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \\
&= J(q)\dot{q}
\end{aligned}
$$

In this configuration, the joint angles of the manipulator are [0, 0, 0, 0].

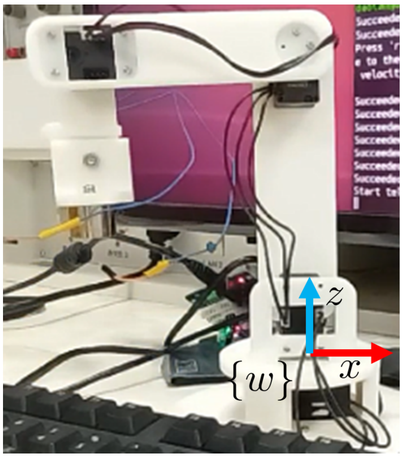


```
Eigen::Vector3d r3e_w=/* implement a vector pointing the end-effector frame from the ID 3 motor frame represented in the world frame */
Eigen::Vector3d r2e_w=/* implement a vector pointing the end-effector frame from the ID 2 motor frame represented in the world frame */
Eigen::Vector3d r1e_w=/* implement a vector pointing the end-effector frame from the ID 1 motor frame represented in the world frame */
Eigen::Vector3d r0e_w=/* implement a vector pointing the end-effector frame from the ID 0 motor frame represented in the world frame */

Eigen::MatrixXd JacobianTranslationEE(3, 4); JacobianTranslationEE.setZero();

JacobianTranslationEE=/* implement the Jacobian matrix for world linear velocity */

return JacobianTranslationEE;
```

<Compute the desired joint angles using inverse kinematics>

49. When computing the desired joint angles using inverse kinematics, we use the right-inverse of the Jacobian matrix. Hence, the end-effector position should be far from singularities. To this end, the initial joint position for teleoperation is [0, 0, PI/2, -PI/2]. This configuraiton is far from singularities. Therefore, teleoperation should be started after moving from the initial joint angles, when the program runs, to [0, 0, Pi/2, -PI/2]. The following code makes the manipulator moves from the initial joint angles to [0, 0, Pi/2, -Pi/2] for 5 seconds.

```
if(!is_ready_for_teleoperation){ // Move the end-effector to initial_position before starting teleoperation
    targetQ = (initial_position - initialQ)/NodeNum*NodeIndex+initialQ; // trajectory generation using the first-order polynomial
    NodeIndex++;
    if(NodeIndex>NodeNum) {
        is_ready_for_teleoperation=true; // Now, it is ready for teleopeartion
        std::cout<<"Start teleoperation!"<<std::endl;
    }
}
```



This is the initial joint position for teleoperation (= [0, 0, PI/2, -PI/2]).

50. When teleoperation mode starts, targetQ is defined based on the keyboard input. If the keyboard input is 'i', targetQ is [0, 0, PI/2, -PI/2] where the initial joint position for teleoperation. If the keyboard input is 'r', 'f', 'w', 's', 'd' or 'a', targetQ is computed via inverse kinematics; $q_d = q + J^+(q)(x_d - x)$. Implement the code on the **176th** line to define

targetQ. Here, J^{+} is the right inverse of the Jacobian matrix, $J^{+} = J^{T}(JJ^{T})^{-1}$ and $x_{d} - x$ is task_command defined in **GetKeyboardInput() function** (refer to 43). In the Eigen library, A^{T} and A^{-1} are computed by A.transpose() and A.inverse(), respectively.

51. If the keyboard input is 'e', end the control loop. If the wrong keyboard input is received or there is no keyboard input, targetQ is set to the previous targetQ. This is implemented in the following code. is_key_updated is used to check if the new keyboard input is received.

```
else{
    ///////////////////////////////////// Teleoperation mode Start /////////////////////////////////////
    if(is_key_updated){ // Perform teleoperation only when a new keyboard input is received (is_key_upated=true)
        if(key=='i') // Move to the initial position for teleoperation
        {
            targetQ=initial_position;
            prev_targetQ=targetQ; // Update prev_targetQ
        }
        else if(key=='r' || key=='f' || key=='w' || key=='s' || key=='d' || key=='a') // Move the end-effector based on the task command
        {
            Eigen::MatrixXd J=Kinematics::GetJacobianMatrix(currentQ); // J is the Jacobian matrix for linear velocity represented in the world frame
            targetQ=/*implement targetQ using inverse kinematics*/
            prev_targetQ=targetQ; // Update prev_targetQ
        }
        else if(key=='e'){ // End teleoperation
            break;
        }
        else // When the wrong keyboard input is received, the targetQ is the privious targetQ value
        {
            targetQ=prev_targetQ;
        }

        is_key_updated=false; // Set is_key_updated to false in order to check if the new keyboard input is received
    }
    else{ // Before receiving the new keyboard input, the targetQ is the privious targetQ value
        targetQ=prev_targetQ;
    }
    ///////////////////////////////////// Teleoperation mode END /////////////////////////////////////
}
```

<Joint space P position control using velocity mode>

52. Implement the code on **the 200th line** for the joint space P position controller using velocity mode.
53. Uncommand the **26th and 46 ~ 57th lines** in CMakeLists.txt file
54. Build the project using vs code button.

55. Before executing teleoperation, you should check that the GetJacobianMatrix(current_q) function returns the correct Jacobian matrix. To this end, run "testJacobian" on the BeagleBone. This file is in the build directory. When you enter the joint angles as shown in the following figure (e.g. "0 0 0 0 \n"), the corresponding Jacobian matrix will be printed to the terminal. This Jacobian matrix must be the same as when you run "getJacobian" on the BeagleBone. This executing file is also in the build directory and it prints the correct Jacobian matrix. **Therefore, please check that the two Jacobian matrices are the same when the joint angles are identical.** If you cannot run "getJacobian", please type the following command.

    [On the Development PC]

    $ cd ~/EE405_ManipulatorControl/build

    $ sudo chmod 777 getJacobian

    ```
    debian@beaglebone:~/nfs_client$ ./testJacobian
    Enter the joint angles
    0 0 0 0
    Jacobian matrix for the world linear velocity
               0        0.293      -0.1765          0.08
               0            0   2.1615e-17 -1.95943e-17
               0            0            0            0
    debian@beaglebone:~/nfs_client$ ./getJacobian
    Enter the joint angles
    0 0 0 0
    The correct Jacobian matrix for the world linear velocity
               0        0.293      -0.1765          0.08
               0            0   2.1615e-17 -1.95943e-17
               0            0            0            0
    ```

56. Now, check the executable file "maniulator_control_week2" in the build directory. If you execute this file, the manipulator moves from the initial joint angles to [0, 0, Pi/2, -Pi/2] for 5 seconds. Please make sure that the initial joint angles are not far from the initial position for teleoperation (=[0, 0, Pi/2, -Pi/2]). When teleoperation starts, then you can control the end-effector position based on the keyboard input. The following keyboard inputs determine the task command. If the end-effector is close to singular positions, the teleoperation may not work well. In this case, press 'i' to move to the initial joint position for teleoperation.
    - r : move 1cm in the world z-axis
    - f : move -1cm in the world z-axis
    - w : move 1cm in the world y-axis
    - s : move -1cm in the world y-axis
    - d : move 1cm in the world x-axis
    - a : move -1cm in the world x-axis
    - i : move to the initial position
    - e : end the teleoperation

57. Run “manipulator_control_week2” on the BeagleBone. (If you want to terminate the teleoperation, press ‘e’.)

debian@beaglebone:~$ cd ~/nfs_client

debian@beaglebone:~/nfs_client$ ./manipulator_control_week2

58. Check that teleoperation works well. The following shows an example of the result.

```
debian@beaglebone:~/nfs_client$ ./manipulator_control_week2
Succeeded to open the port!
Succeeded to change the baudrate!
Press 'r', 'f', 'w', 's', 'd', or 'a' to move the end-effector. Press 'i' to mov
e to the initial position. Press 'e' to end this mode
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded operating mode to velocity mode.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Succeeded enabling DYNAMIXEL Torque.
Start teleoperation!
Press 'r', 'f', 'w', 's', 'd', or 'a' to move the end-effector. Press 'i' to mov
e to the initial position. Press 'e' to end this mode
Press 'r', 'f', 'w', 's', 'd', or 'a' to move the end-effector. Press 'i' to mov
e to the initial position. Press 'e' to end this mode
Press 'r', 'f', 'w', 's', 'd', or 'a' to move the end-effector. Press 'i' to mov
e to the initial position. Press 'e' to end this mode
Press 'r', 'f', 'w', 's', 'd', or 'a' to move the end-effector. Press 'i' to mov
e to the initial position. Press 'e' to end this mode
Press 'r', 'f', 'w', 's', 'd', or 'a' to move the end-effector. Press 'i' to mov
e to the initial position. Press 'e' to end this mode
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
Succeeded disabling DYNAMIXEL Torque.
```