Lab Procedure for Python
Inverse Kinematics

Setup

1. It is recommended that you review Lab 2 - Application Guide before starting this lab.
2. Hardware Preparation:
    a. Ensure that the QArm Mini is securely attached to the base.
    b. Verify that the manipulator is in the rest position.
    c. Confirm that the QArm Mini is connected to the PC and turn it ON (the light in the switch should be red).
    d. Check and update the latency setting as shown in Figure 1:
        i. Navigate to Device Manager > Ports
        ii. Select the appropriate device - USB Serial Port (COMx) Make a note of the COM port Number.
        iii. Go to Port Settings > Advanced > Latency
        iv. Set the latency to 2 ms

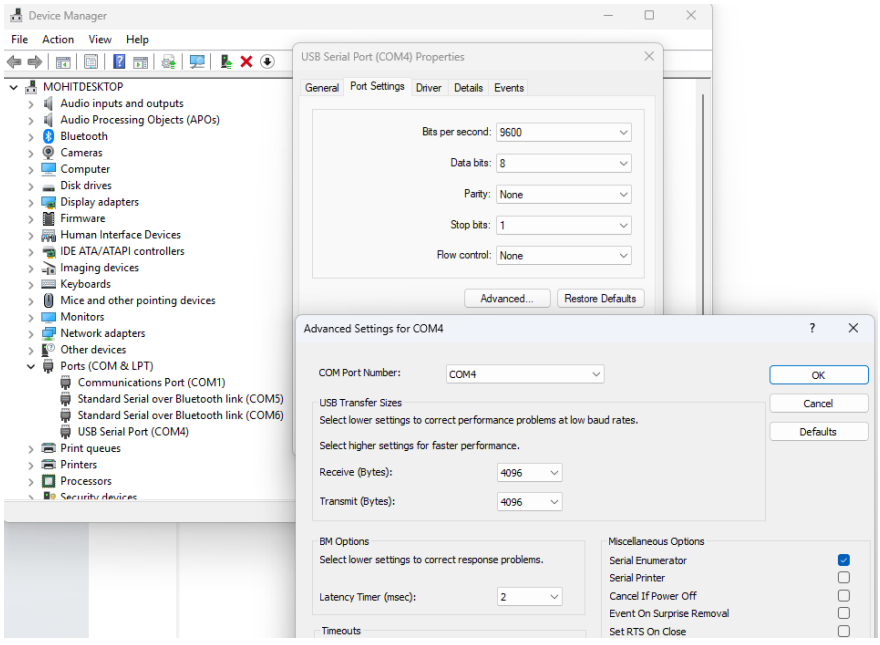

Figure 1. Latency Settings

Inverse Kinematics

1. Launch **Visual Studio Code** and browse to the lab directory via the **File > Open Folder** menu. Once in the correct directory, open teach_pendant_learn.py.
    a. Update the **id** parameter in line 20 to match the COM port you noted during setup.
2. This section of the lab explores inverse kinematics and allows you control the QArm Mini within the cartesian workspace.
3. Ensure the space around the QArm Mini is clear of any objects.
4. Run the script using the ▷ button in the top-right corner.
5. The manipulator starts in the home position. Note the following actions you can take.
    a. Tap any of the **X, Y, Z,** keys once to select the corresponding axis along which the end-effector will move. The **G** key controls the end effector angle.
    b. Press and hold the UP or DOWN arrow keys to incrementally increase or decrease the end-effector's position along the selected axis.
    c. At any point, you can tab SPACE BAR to record the current (x, y, z) coordinates of the end effector in 3d space.
6. The terminal displays the current (x, y, z) coordinates of the end-effector in meters. At the home position, these coordinates are approximately [0.256, 0, 0.256]. Tap the X key to select the X-axis, then press and hold the UP arrow key to move the end-effector approximately 5 cm in the positive X direction. The new position should read close to [0.306, 0, 0.256]. Observe the motion of the manipulator and verify that it moves as expected along the X-axis.
7. When you are satisfied tap CTRL-C until the script outputs **Received user terminate command**.
8. Next, choose a point within the valid workspace—optionally, place a reference object at that point. Using a ruler or tape measure, measure the distance from the front of the gripper (home position) to the selected point in the x, y, and z directions (in meters). Estimate the desired end-effector angle by tapping G. Using the X, Y and Z keys move the end effector to the reference point of object and verify the measurements you took approximately match the position printed out in the terminal.
9. Keeping in mind, joint limits identified from Workspace Identification lab, what do you observe if you attempt to drive the arm beyond its reachable workspace? How does the manipulator respond when commanded outside its valid range?
10. Terminate the script, tap CTRL-C until the script outputs **Received user terminate command**.
11. Proceed to the Teach Pendant section or turn off the arm and gently move it to its resting position (refer User Manual) if not proceeding immediately

Teach Pendant (Learn)

1. Launch **Visual Studio Code,** navigate the correct directory like step 1 in Inverse Kinematics. Once in the correct directory, open teach_pendant_learn.py.
    a. Update the **id** parameter in line 20 to match the correct COM port you noted during setup.
2. This script lets you to "teach" the QArm Mini a series of waypoints that simulate a robotic assembly process, emulating the learning mode of a real teach pendant.
3. There are five predefined waypoints that form the edges of the rectangle-the path the manipulator will follow:
    a. Home Position: [0.256; 0; 0.256; 0]
    b. Point 1: [0.12; -0.08; 0.05; -pi/2]
    c. Point 2: [0.12; 0.08; 0.05; -pi/2]
    d. Point 3: [0.18; 0.08; 0.05; -pi/2]
    e. Point 4: [0.18; -0.08; 0.05; -pi/2]

    Note: x, y, z positions are in meters, relative to the base and γ (gamma), is the end-effector angle.
4. Ensure the workspace is clear of any objects.
5. Run the script using the ▷ button in the top-right corner.
6. **Recording the Motion:**
    a. Tap any of the **X, Y, Z** keys once to select the corresponding axis along which the end-effector will move. Use the **G** key to adjust the end-effector's orientation (γ).
    b. Press and hold the UP or DOWN arrow keys to incrementally increase or decrease the end-effector's position along the selected axis.
    c. While monitoring the terminal output, move the end-effector as close as possible to each vertex of the rectangular path, aligning it with the target point.
    d. When aligned with a point, press the SPACE BAR to record the current coordinate.
    e. Repeat steps a – d until you have gone through all the points.

    Note: If you accidentally add a waypoint or need to restart, tap CTRL-C to stop and rerun the script to record new points. The saved points are stored in a 'end_effector_data.csv' file.
7. Once you add all waypoints, stop the script. Bring the manipulator back to rest position.

Teach Pendant (Follow)

1. Launch **Visual Studio Code** navigate the correct directory like step 1 in Inverse Kinematics. Once in the correct directory, open teach_pendant_follow.py. You will use this script to read the waypoints that were recorded in the Teach Pendant (Learn) section and recreate a robotic assembly process automatically.
    a. Update the **id** parameter in line 19 to match the correct COM port you noted during setup.
2. The variable **threshold**, in Line 22 determines the duration between each waypoint. Once the script is running, it reads a new position from the 'end_effector_data.csv' file every 3 seconds.
3. Ensure the workspace is clear of any objects. Verify that the manipulator is in the rest position.
4. Run the script using the ▷ button in the top-right corner.
5. Does the manipulator follow the rectangular path as expected? Record your observations.
6. Stop the script, tap CTRL-C until the script outputs **Received user terminate command**. Once the script is stopped, a 3D plot will be generated showing the commanded trajectory alongside the actual path followed by the manipulator. You can click and drag with your mouse to rotate and explore the plot from different angles.
7. Take screenshots of the plot.
8. Close the 3D plot to complete the script.
9. Turn off the arm, gently move it back to its resting position.

Trajectory Generation

1. Launch **Visual Studio Code** navigate the correct directory like step 1 in Inverse Kinematics. Once in the correct directory, open trajectory_generation.py. This script follows a rectangular path using predefined waypoints, like the Teach Pendant method but with automated trajectory generation.
    a. Update the **id** parameter in line 20 to match the correct COM port you noted during setup.
2. Lines 26 to 34 in the script define the different waypoint value:
    a. home = [0.256; 0; 0.256; 0]
    b. A = [0.12; -0.08; 0.05; -pi/2]
    c. B = [0.12; 0.08; 0.05; -pi/2]
    d. C = [0.18; 0.08; 0.05; -pi/2]

e. D = [0.18; -0.08; 0.05; -pi/2]

3. The `waypoint_navigator` function determines the current and next waypoint based on the elapsed time. It returns the starting pose (Pi), the target pose (Pf), and a time vector used for trajectory generation. The function cycles through a predefined list of waypoints at regular time intervals (duration), for trajectory generation.

4. The `cubic_spline` function calculates the coefficients of a smooth cubic trajectory between two points (Pi and Pf) over a specified time duration T. It returns a 3×4 matrix of coefficients used to interpolate the X, Y, and Z positions of the end-effector, ensuring smooth motion between waypoints without abrupt changes in velocity.

5. Ensure the workspace is clear of any objects.

6. Run the script using the ▷ button in the top-right corner.

7. Observe whether the end effector moves as the same way as it did in the Teach Pendant experiment. Note any differences in smoothness and accuracy when moving between waypoints (e.g., A to B, B to C).

8. Stop the script. Once the script is stopped, a 3D plot will be generated showing the commanded trajectory alongside the actual path followed by the manipulator. Do you notice any differences?

9. Try increasing the duration parameter in Line 44, from 3.0 to 6.0. Rerun the script. How does this affect the motion between waypoints? Take screenshots of the 3D plot generated.

10. Repeat the experiment but try decreasing the duration parameter from 3s to 1s. how does this affect the motion between waypoints? Take screenshots of the 3D plot generated.

11. Close the 3D plot to complete the script.

12. Turn off the arm, gently move it back to its resting position.