ME 381

Experiment 6: Kinematics of a two-link serial planar manipulator

Aim:

Perform forward and inverse kinematic analyses of the two-link serial manipulators. Use inverse kinematics to draw 2D curves.

Preparation

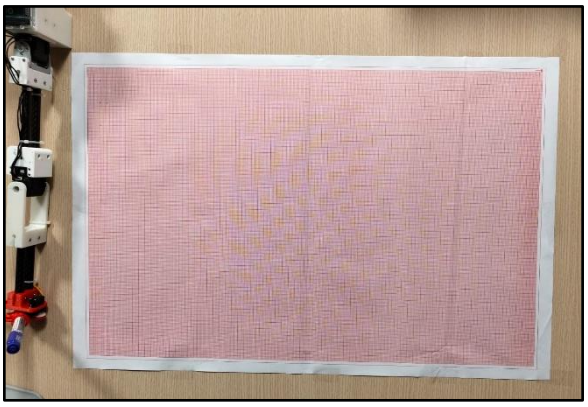
Software setup:

Basic knowledge of Python is necessary to operate the robot.

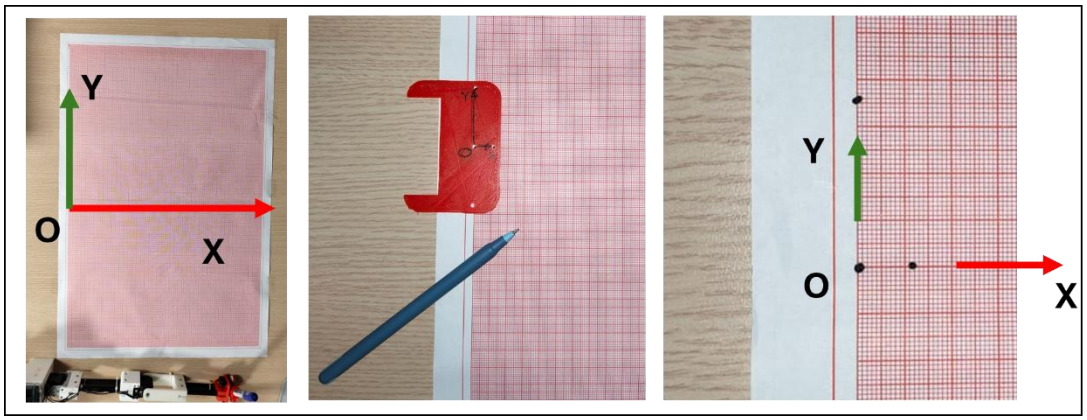
1. Follow the installation manual.
2. Open the robot API manual for command reference

Hardware setup:

1. Lay flat and stick the graph paper to the table using a cello tape as shown below.

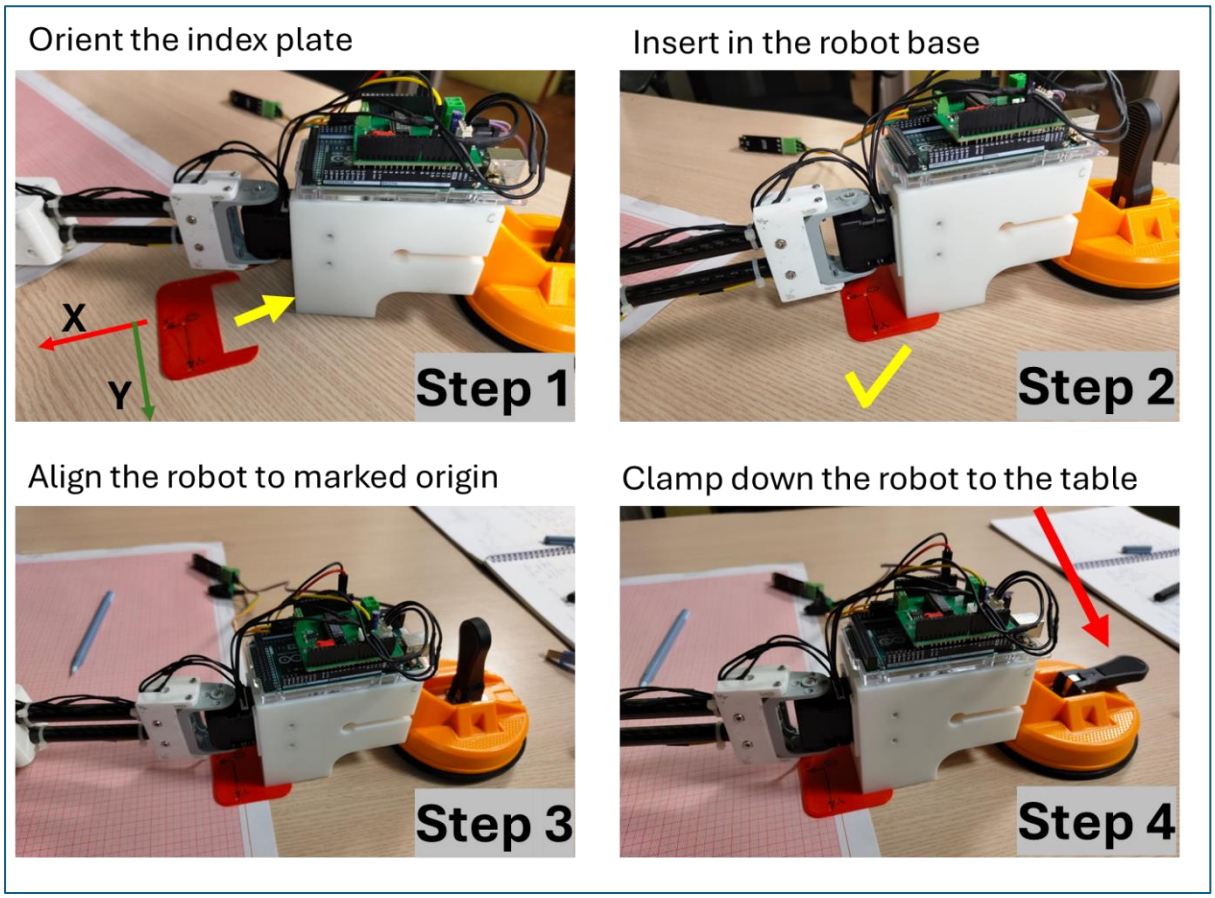


2. Mark the origin using pen by aligning the indexing plate to the grid as shown below



PTO

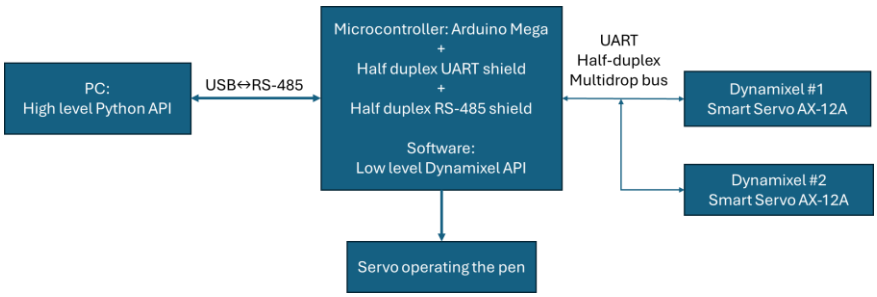
3. Align the robot to the grid, and clamp it to the table



4. Plug in the power supply and set it to 11 V.
5. **Turn off** the power supply, and connect it to the robot using the DC jack.
6. Turn on the power supply once the DC jack connection is secure.
7. Connect the computer to the robot through the USB-RS-485 converter using the USB cable provided. *Note: Do not connect to the Arduino MEGA port.*
8. ***Set the USB latency to 4 ms as described in Appendix A.***

Robot operation brief

Easy to use, high level Python API is developed for robot operation. The PC transmits commands (for the motor position, velocity, gain, etc. values) to Arduino Mega using serial communication. The Arduino MEGA microcontroller further converts these commands into low level commands for the smart servo motors connected to the robot joints. It also sends commands to operate the servo motor holding the end effector, i.e., the pen.



For more information on the Dynamixel smart servo motors visit:
https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

Common errors and troubleshooting:

1. Err: Timeout
    a. USB cable from PC must be connected to the USB-RS485 converter, not Arduino Mega.
    b. Ensure that the red DIP switch on the Arduino MEGA shield is in down position
    c. Ensure firm connection of 3-pin Dynamixel motor connector (white) on the Arduino MEGA shield
    d. Reset the Arduino MEGA using the RESET button
    e. Disconnect and reattach the USB cable to the PC
    f. Adjust the USB communication latency as mentioned in Appendix A.
2. Pen servo not moving
    a. Check the servo wires connected to Arduino pins 9, 5V, and GND
3. Curve not traced properly
    a. Adjust the pen by expanding the clamp when the servo is OFF
    b. Load the pen lightly while the robot is tracing the trajectory

Tasks and report

The experiment has five parts

1. Get to know the robot operations (20 min)
2. Forward kinematics (15 min)
3. Teach and toggle (15 min)
4. Inverse kinematics (20 min)
5. Trajectory tracking (10 min)

1. Getting to know the robot operations

Keep the API manual handy and learn how to work with the following functions by tweaking the '**trials_2R.py**' file.

Show the output to the TAs, there is no report for this section.

Sr.	Get to know	Trial code snippet / TODO

1	Program initialisation	`from RobotAPI_class import RobotAPI`
		`import time`
		`from time import sleep`
		
		`BAUDRATE = 1000000 # PC-Arduino communication rate`
		`DEVICENAME = "COM9" # COM port number may vary`
		`ROBOT = "2R" # Robot name (used in some functions)`
		`BASE_MOTOR = 1 # Predefined ID of the base motor`
		`ELBOW_MOTOR = 2 # Predefined ID of the elbow motor`
		
		`# Make an instance of RobotAPI class`
		`my2R = RobotAPI(DEVICENAME, BAUDRATE, ROBOT)`

2	penDown()
	penUp()
		`for i in range(5):`
		`  my2R.penDown()`
		`  sleep(1)`
		`  my2R.penUp()`
		`  sleep(1)`
		
		Puts the pen down and lifts it up five times

3	setTorqueON()
	setTorqueOFF()
		`my2R.setTorqueOFF(BASE_MOTOR)`
		`sleep(0.1)`
		`my2R.setTorqueOFF(ELBOW_MOTOR)`
		`sleep(10)`
		`my2R.setTorqueON(BASE_MOTOR)`
		`sleep(0.1)`
		`my2R.setTorqueON(ELBOW_MOTOR)`
		
		Try moving gently both the robot links while the torque is OFF for 10 seconds

4	goHome()	`my2R.goHome()`
		`sleep(2)`
		
		The robot moves to the home position

5	Have a look at the API manual and try out any other functions if you wish to!	

PTO

2. Forward kinematics

The forward kinematic analysis of a robot is defined as computing the end effector (pen in this case) pose given the joint angles.

Tweak the same '**trials_2R.py**' file for the following tasks.

Report (homework): Complete the equations of the forward kinematics of 2R:

$$X = f(\theta_1, \theta_2)$$

$$Y = g(\theta_1, \theta_2)$$

Notation:
θ_1: base motor angle, θ_2: elbow motor angle, l_1: proximal (base) link, l_2: distal(second) link

The robot link lengths are: $l_1 = 200, l_2 = 200$ (mm)

Sr.	Function	Trial code snippet / TODO	**Report**

	setJointAngle()	`# Command the robot to move the base motor by <angle>.`
		`my2R.setJointAngle(BASE_MOTOR,-30, unit: "deg")`
		`my2R.setJointAngle(ELBOW_MOTOR, angle: 30, unit: "deg")`
		`# Allow the robot to move to the said position.`
		`sleep(2)`
		`# make a mark`
		`my2R.penDown()`
		`sleep(1)`
		`my2R.penUp()`
		Repeat the experiment for the following angle values:
			Fill in the table below
			
			Measure and report all the coordinates in mm
			
			(HW) List reasons for the deviation in the calculated and measured coordinates, if any.

Try out eight/nine values based on the number of students in the group. Use angles of your choice wherever left blank. **Each student to mention all the values in their report.**

Sr. no.	Base motor (degrees)	Elbow motor (degrees)	****Calculated coordinates (X,Y) (units: mm)**	**Measured coordinates (X,Y) (units: mm)**
1	0	0		
2	90	-90		
3	-90	90		
4				
5				
6				
7				
8				
9				

** fill in the calculated coordinates as homework.

PTO

3. Teach and toggle

This is an example of a practical use case of the robot wherein the user moves the end-effector manually to two different positions, records those, and then the robot toggles between them.

1. Open the file '**teach_and_toggle.py**'
2. Run the program
3. Move the robot manually to the first desired position, and press 'R' to record the pose
4. Move the robot to the second desired position, and record the same
5. Record the third desired position
6. Press 'Enter' and watch!

Report:

1. Take two photos of the taught (recorded) positions
2. Take two photos when the robot reaches those positions (achieved positions)

PTO

4. Inverse kinematics

The inverse kinematic analysis of a robot is defined as the computation of (all the) solution(s) of the joint angles given the end effector pose.

Use the file '**example_IK_2R.py**' for the following. The inverse kinematics function has been written already in '**IK_functions.py**' file.

1. Edit the **x** and **y** values and run the program. The program will show both branches of the inverse kinematic solutions. An angle is positive according to the right-hand rule with the thumb pointing upwards.

Each student must report all the values

	X (m)	Y(m)	Solution 1: (θ_1, θ_2) in degrees	Solution 2: (ϕ_1, ϕ_2) in degrees	Remarks (if any)
1	0.2	0.2			
2	0.3	0.0			
3	0.2	-0.2			
4	0.3	-0.3			
5					
6					
7					
8					
9					

PTO

5. Inverse kinematics: trajectory tracking

Run the two programs:

1. Code 1: '**discrete_trajectory_ellipse_2R.py**'
2. Code 2: '**discrete_trajectory_line_2R.py**'

Report:

1. Connect the dots!
2. Take the photos of the plotted trajectories

Report (homework)

1. What curves are being tracked in code 1 and code 2?
2. Give the specifics of the curves by reading the code.

REPORT

File name: Exp6_<Name>_<RollNo>_<A6> (batch and group no.)

<Your details>

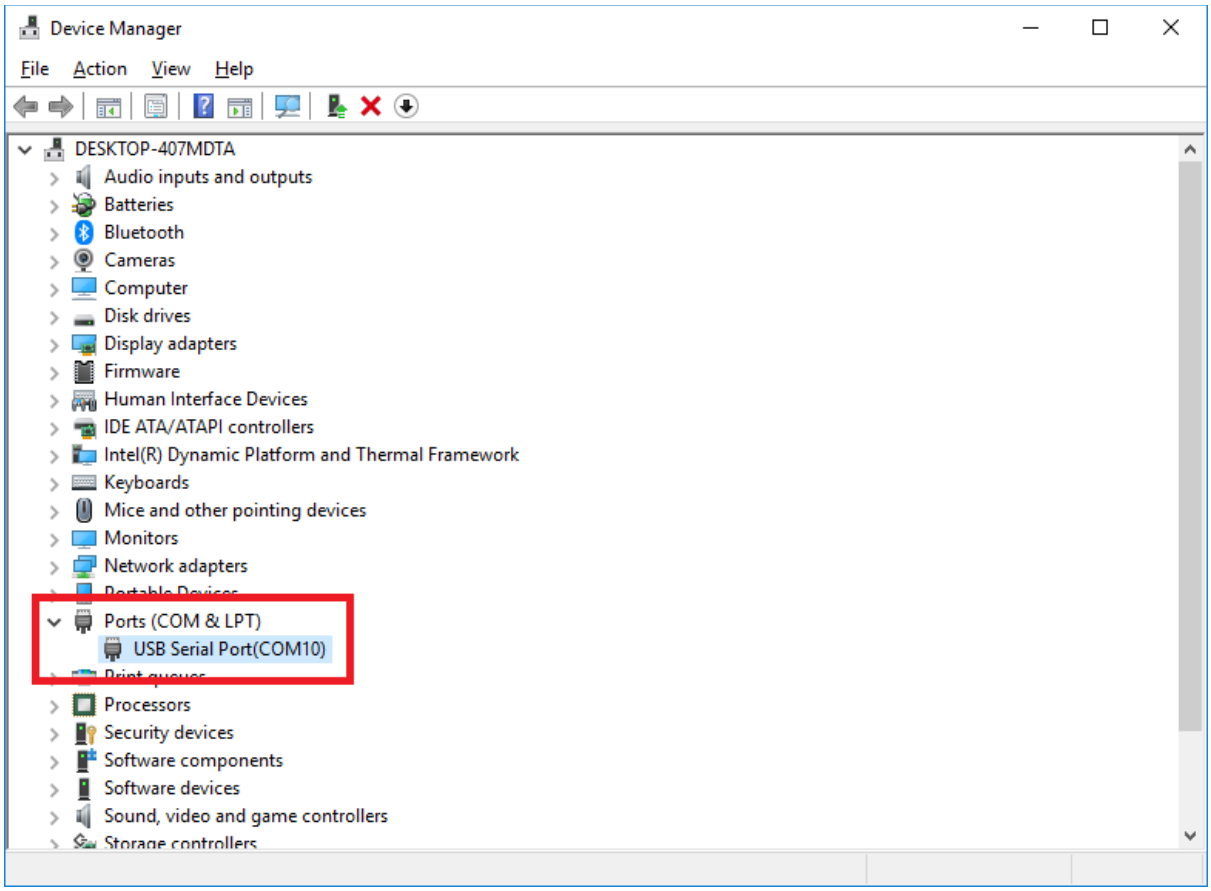
	Tasks	Report
1	Get to know the robot operations	No report
2	Forward kinematics	Table with 8/9 values
3	Teach and toggle (images and answers)	Photographs as mentioned
4	Inverse kinematics	Table with remarks (point outside workspace/inside workspace)
5	Trajectory tracking	Answers

Appendix A

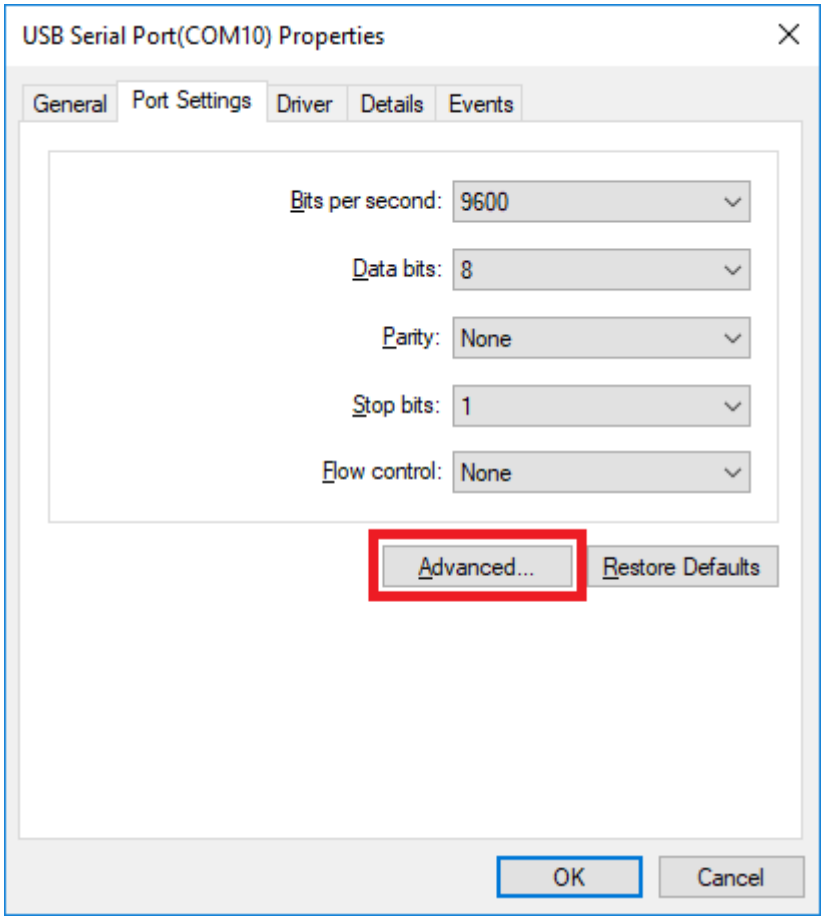
USB Latency Setting

Windows

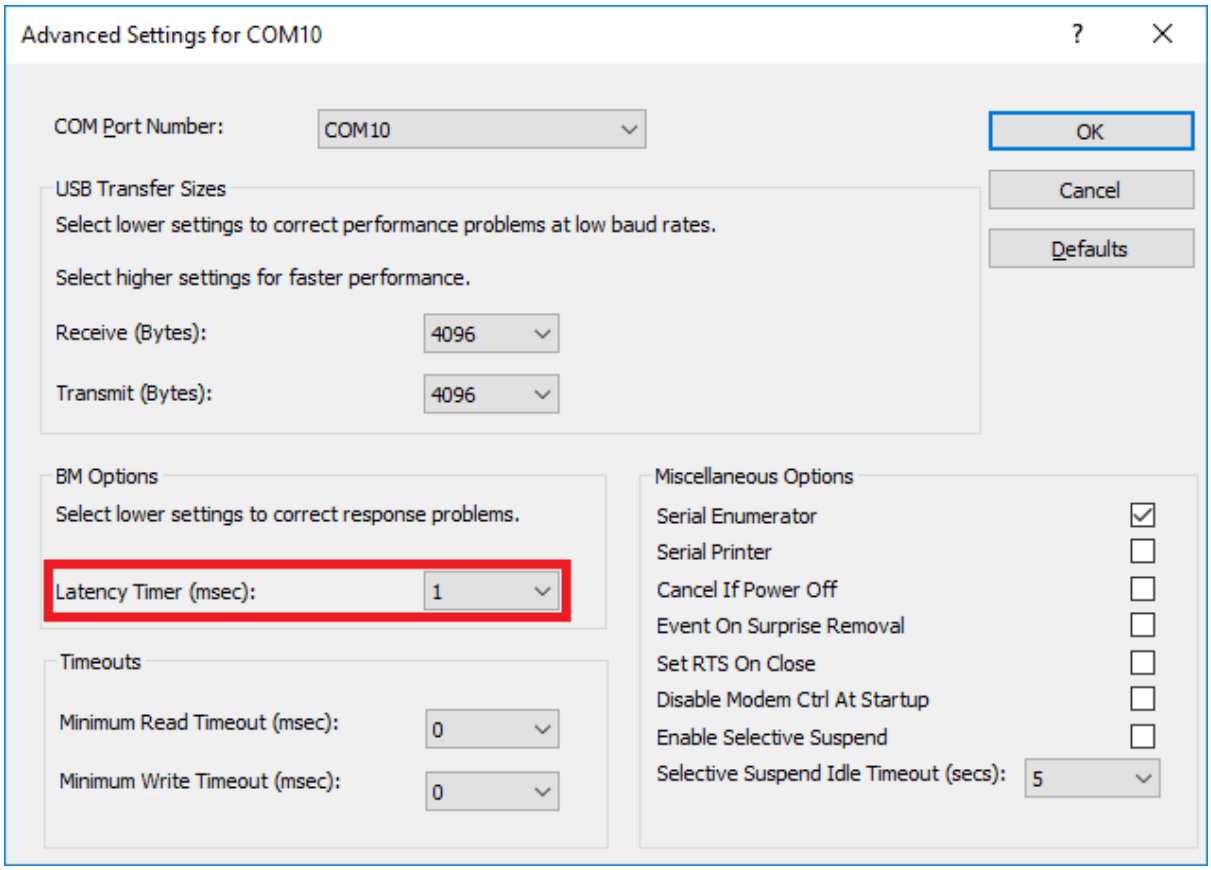
1. Open Device Manager. Go to Ports item and right click on the relative serial port to select Properties.



2. In the Properties window, go to Port Settings tab and click Advanced button.



3. Set the Latency Timer (msec) to 1-4 ms and click OK to confirm the change.



6. 8. 1. 2. Linux

1. Execute below commands to configure the `latency_timer` to `1ms`.

```
# cat /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
16
# echo 1 > /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
# cat /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
1
```