Practical Class 1a - ENGM300

The robots that you will be exploring in this week's class are from Trossen Robotics. They are designed more for educational and hobby activities than with a particular industrial or medical setting in mind. Control of the robots can be achieved using many different tools and applications including Matlab, python and ROS - the Robot Operating System.

Today the class is designed to:

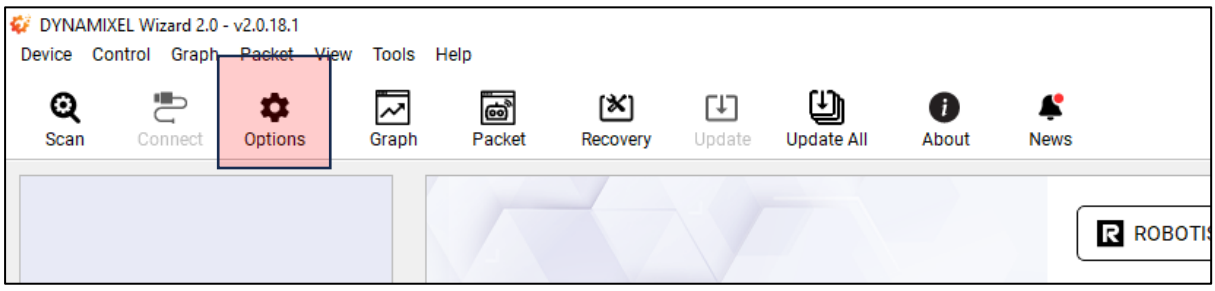
- introduce the robots;
- explore their functionality;
- and to link the robots with of the concepts that have been covered in the lectures so far.

Before using the robots, you will need to download the software for operating their motors. This is a standalone application and there are versions for running on a Windows, Linux and Mac operating system. The software is available here:

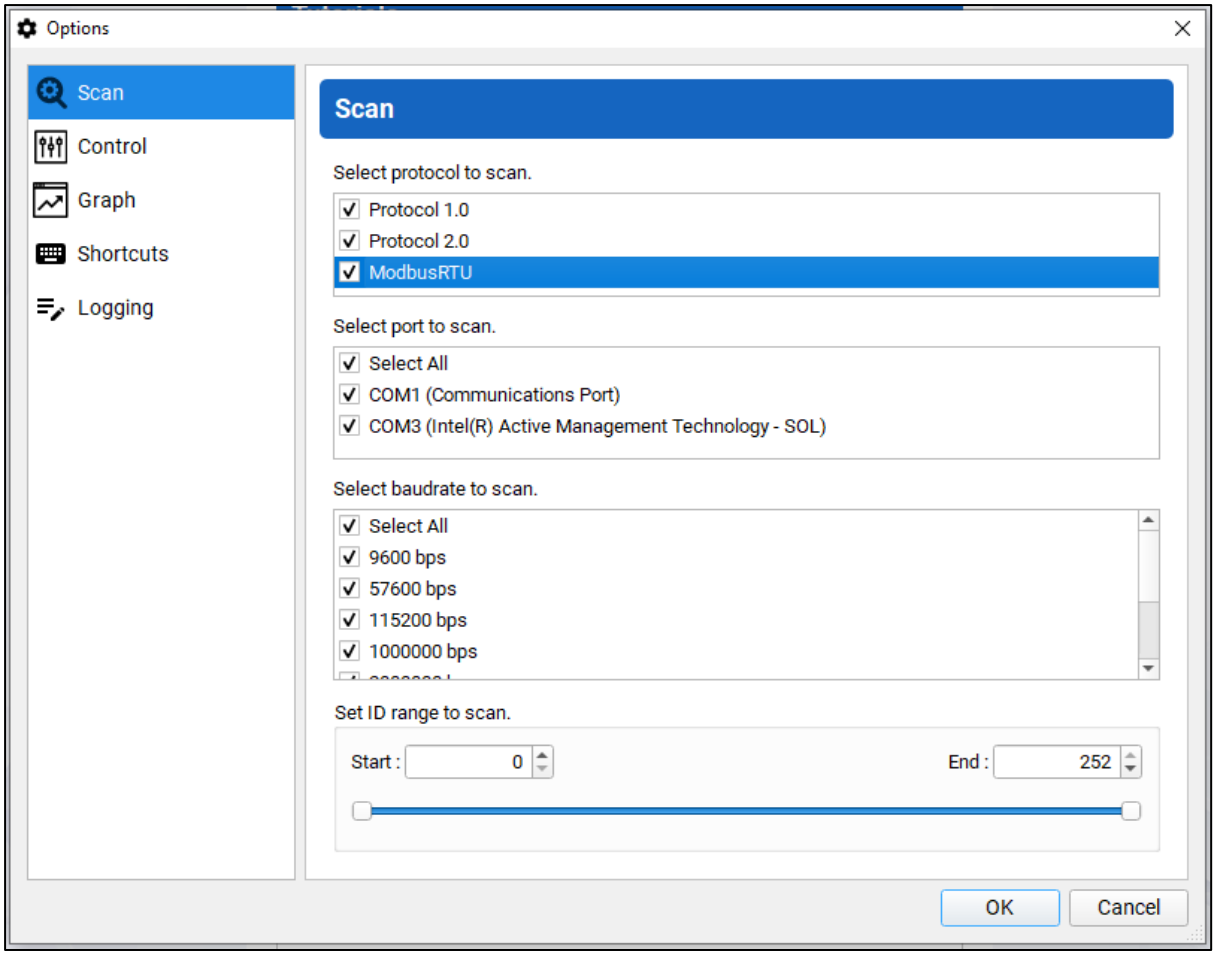
- https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/

After downloading and installing the software on your machines you should first identify make sure that the robots can be 'seen' by your computer. To do this, you need to open the Dynamixel Wizard V2.0 application. The application must scan for the robot's motors using different protocols, ports and baud rates.

- In Dynamixel, open the *Options* menu.

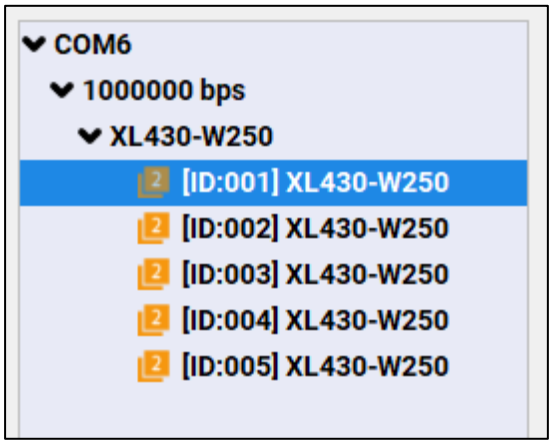


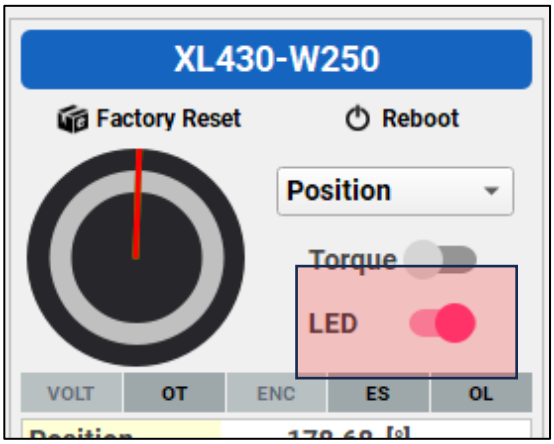
- Under *Scan* [see next page], make sure that all of the protocols, ports and baud rates are selected.



- In the main workspace, select *Scan* and wait for the operation to finish. If you want to reduce the time for the scanning process, try all of the ports but with only Protocol 2.0 and 1000000 bps. Five motors should then be visible on the left-hand side of the workspace.

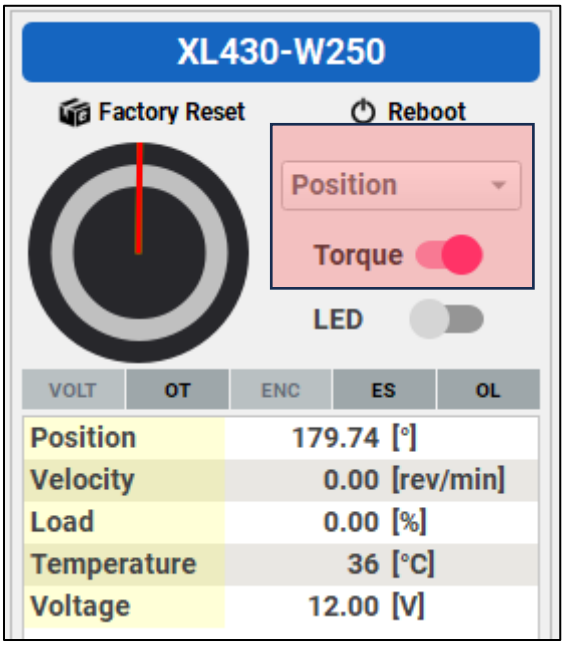
There are very many things that one can try with the motors. It is helpful to identify all of the motors before trying to make them move. This can be done by selecting them on the left-hand side of the workspace and then toggling their light on the right-hand side of the workspace [you may have to look quite hard for some of the lights when the robot is in its powered-down configuration!]:





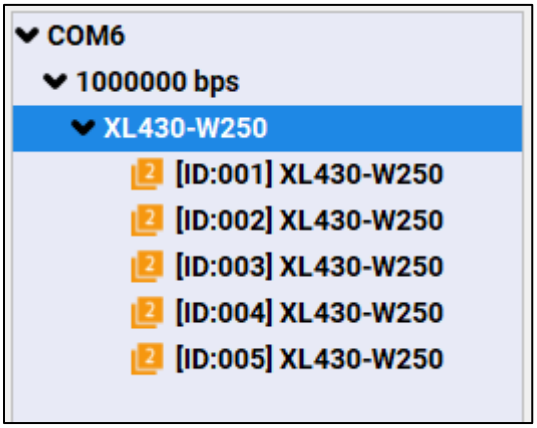
The motors will not move unless the *Torque* mode is switched on. To switch the T*orque* mode on, the *Torque* toggle must be changed. For most of today's work, the dropdown menu can be left in *Position* mode. Later, you may want to see what happens if you switch to other modes like *Velocity* and *PWM*. What does *PWM* mean?

To begin working with the robot to work, set the *Torque* mode on for *Position* control:



You may want to turn the *Torque* mode off and on for a particular motor and **gently** see whether you can move the arms in each case.

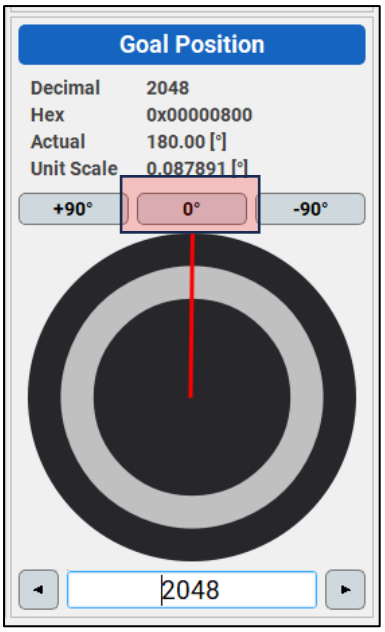
To make the manipulator arm move, one must specify the robot's position [when in *Position* mode]. Once again, make sure that all of the robot motors are set to have *Torque* mode on. This can be done by selecting all of the motors simultaneously:



Scroll down to Address 116 in the central table, which should be *Goal Position*:

108	Profile Acceleration	0	0x00000000	0.00 [rev/min²]
112	Profile Velocity	0	0x00000000	0.00 [rev/min]
116	Goal Position	2048	0x00000800	180.00 [°]
❯ Indirect Address 1~28				

When this row is selected, the bottom right-hand side of the screen should show you tools for controlling the *Goal Position*:

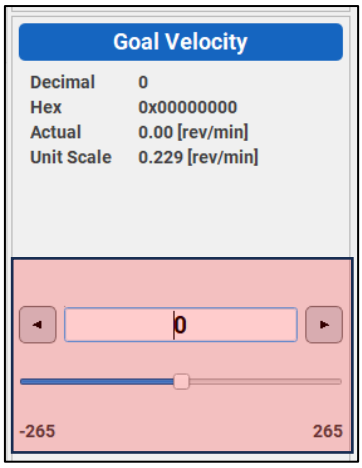


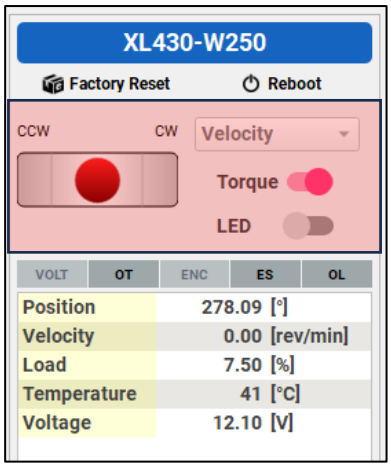
Make sure that you are not putting any load on the robot and press 0°. What has happened?

Now select motor 4 from the list on the left-hand side. Toggle *Torque* off. Gently, move the penultimate link in the chain by hand to an arbitrary position. Turn *Torque* back on and select again 0°. Repeat the process of turning *Torque* off, repositioning arbitrarily and gently by hand, turning *Torque* back on and selecting a *Goal Position* of 0°. What does that tell you about the way that the motors are being controlled?

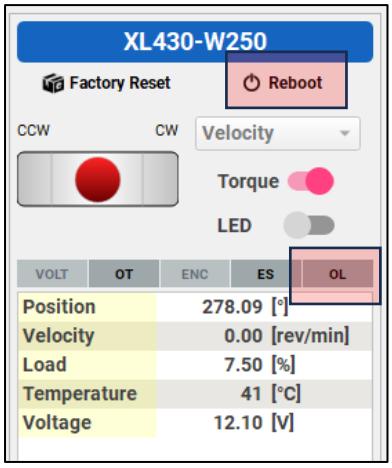
Take some time to explore the motor control too. You may want to see what happens with individual motors when *Goal Velocity* is selected. When the manipulator arm is moving beyond its safe limits it will switch off to protect itself. To minimise this risk, try to avoid making big adjustments to the position or velocity. The fine tuning and velocity controller can both help in this regard:

104	Goal Velocity	0	0x00000000	0.00 [rev/min]
108	Profile Acceleration	0	0x00000000	0.00 [rev/min²]
112	Profile Velocity	0	0x00000000	0.00 [rev/min]
116	Goal Position	2043	0x000007FB	179.56 [°]





N.B. If the *OL* tab is highlighted then the motor needs to be *Rebooted*:



You may also want to try plotting graphs before or while attempting to answer the questions on the following page. A live view can be achieved while the manipulator is in motion. Detailed instructions for using the wizard can also be found at:

- https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/

Questions to Explore

Try to answer the following questions with as much detail as possible. These are formative [not assessed] and should help to link the physical robot with things that have been covered during lectures. **In answering the questions, you are encouraged to experiment with the Dynamixel interface as much as possible.** Some answers may require internet searches.

The questions are somewhat divided into categories and aim to cover aspects of the robot kinematics, operation, and strengths and weaknesses of the device.

Robot kinematics

- How many joints can you identify?
- What type of joints are they?
- Where is the {*B*} and where is {*W*}?
- What are the differences between the robot you are using and the Puma robots that have been covered during the lectures?
- What are the similarities in the kinematics of the robot you are using and the Puma robots?
- In the Dynamixel interface, what type of workspace(s) are being used to control the robot arm?
- Looking at the robot's tool, how would you define this? Is the cartesian, joint or actuator space most appropriate?

A practical task!

- Try to make the robot pick something up from the desk, transport it to somewhere else and then release it (at desk level)?

Movement and its control

- What can you find out about the motors and the possible languages used to control them?
- How is position data being read? Is it absolute or relative data?
- When experimenting with the robot motion, what happens when the robot moves too far or too fast?

An overview of the robot design

- What are the limitations of the robot in its conceptual design and when operating it using the Dynamixel interface?
- Is the Dynamixel software controlling the robot? This seems an odd question that is open to a variety of interpretations – what may some of those different interpretations be?