# Robotic Manipulation – ELEC60030

# Lab 1 – Dynamixel Actuator Interfacing and Control

This lab serves as a 'getting started' guide for using the Dynamixel Model-X smart servos, which are manufactured by Robotis. These actuators will be used for the remaining labs and coursework, so it is important that you understand how to use them.

Dynamixel Model-X servos are compact, high-performance actuators frequently used in contemporary robotics research. In addition to impressive torque and resolution, they have good control options, software support and documentation.

We will be using Matlab to interface with the Dynamixels during this course. Matlab also provides nice tools for performing the various calculations we will be using later in the course. To keep things simple, we will be starting with a single servo. Additional servos can be daisy-chained later.

---

## Task 1 – Setting up the Dynamixel SDK for Matlab on your laptop.

The Dynamixel SDK allows us to communicate efficiently with the Dynamixel actuators. These instructions will help you set it up on your computer.

Windows

1. Download the SDK (left column->Download->Repository->Option 2: Direct Download) and store it somewhere sensible (not a temp folder) https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/

2. Use the 'Set Path' button in Matlab and follow these instructions to import the libraries and references (don't do step 4.14.3.3 Run Example yet). **But instead of selecting '`[DynamixelSDK folder]/c/build/win32/output`' in one of the steps, we want to add the win64 version '`[DynamixelSDK folder]/c/build/win64/output`'** : *https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/library_setup/matlab_windows/#matlab-windows*

3. Install MinGW compiler if necessary, using the Matlab Add-Ons button

4. Matlab should now be configured – try to run the readwrite.m example (step 4.14.3.3 in the above instructions, **but open read_write.m in the 'protocol2.0' folder instead of the one in the 'protocol1.0' folder.** We will be using protocol 2.0 throughout the course). It should load the library but fail to open the port.

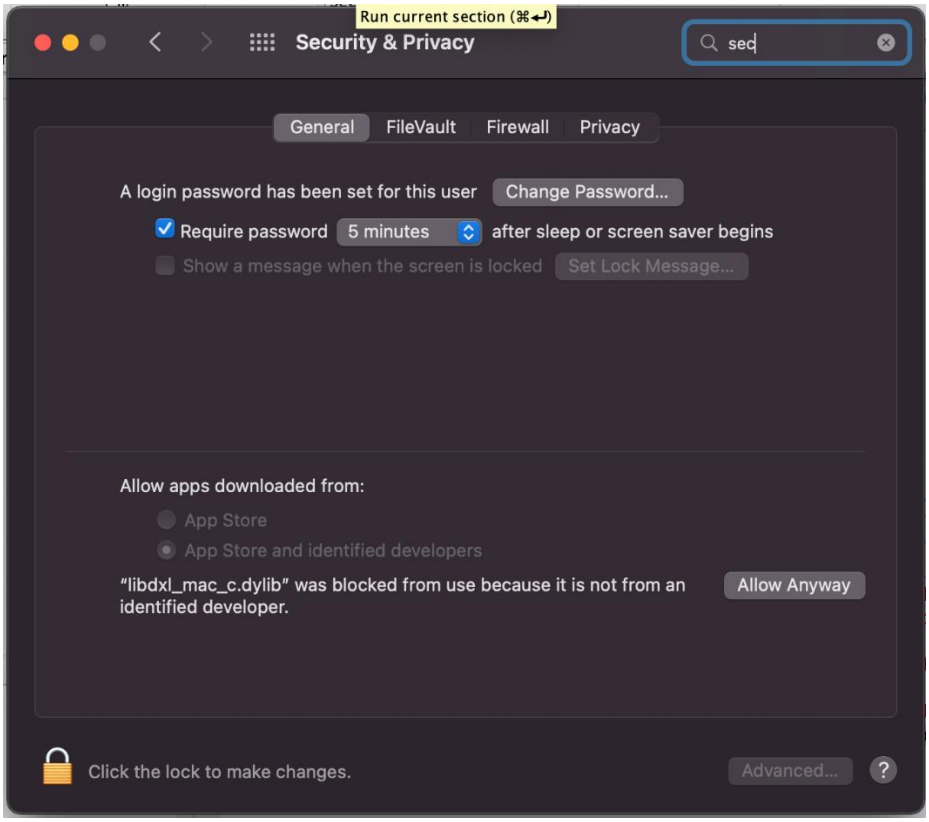
Mac

Dynamixel SDK does not work well with Mac (no performance issues, but trickier to set up). If you experience major difficulties, please complete the installation in your own time and continue with this lab on a lab mate's laptop.

1. Download the SDK (left column->Download->Repository->Option 2: Direct Download) and store it somewhere sensible (not a temp folder)

https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/

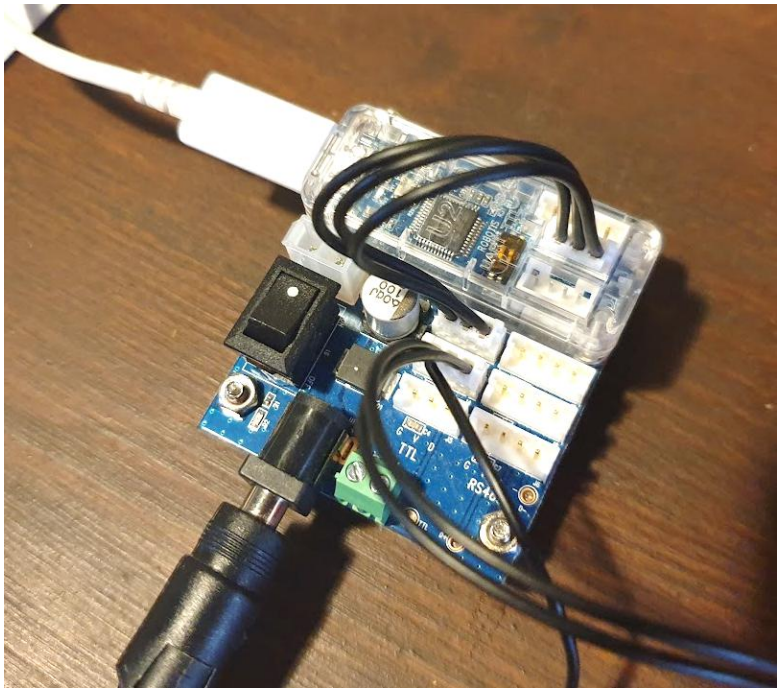
2. Copy and paste the provided 'libdxl_mac_c.dylib' file (teams channel files tab) into folder '`[DynamixelSDK folder]`/c/build/mac'
3. The Dynamixel SDK only supports matlab versions <=2023a. Install Matlab 2023a if your current Matlab version is higher.
4. Follow these instructions to add the library files to the Matlab path (When running the script, make sure to select the readwrite.m file in the 'protocol2.0' folder, which will be the protocol we use throughout the course):https://github.com/ROBOTIS-GIT/emanual/blob/master/docs/en/software/dynamixel/dynamixel_sdk/library_setup/matlab_macos.md
5. Upon running the first Matlab script, a pop-up warning might notify you that the library cannot be accessed because the developer cannot be identified. Go to Settings > Security & Privacy > General and press 'Allow Anyway'.




5. Matlab should now be configured – try to run the readwrite.m example again. It should load the library but fail to open the port.
6. If there are compiler issues, download XCode, which will prompt you to install 'developer tools' upon installation which are required for the Dynamixel SDK.

## Task 2 – 'Hello World' – Moving the Actuator via simple commands

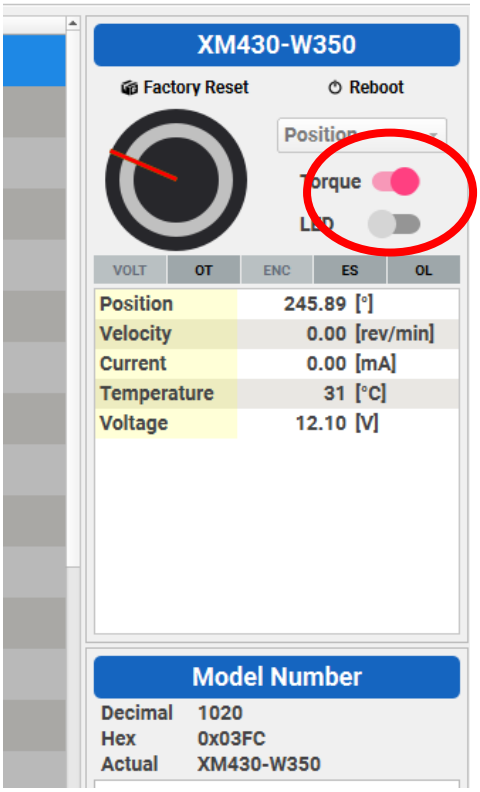
The Dynamixels use three cables for communication and power, which are provided by the U2D2 (which converts USB serial to TTL or RS485) and the U2D2 PHB set (which regulates power from an AC/DC convertor).

1. If your set is not assembled yet, assemble it and connect the Dynamixel to the U2D2 as shown in the above image. Get the GTAs to check the connections before you power up the system.
2. When you activate the switch on the U2D2 PHB, you should see the red LED blink once on the Dynamixel. Make sure the switch is on before carrying out the next steps.

The **Dynamixel Wizard** – is some simple software used to interface with Dynamixel. It allows you to scan a network of actuators to determine ID numbers and communication settings. You can also manually access the various functions of each Dynamixel. This can be very useful for debugging.

3. Download the Dynamixel Wizard from https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/
4. Run the Dynamixel Wizard, press the Options button (gear button on top), and set your serial (COM) port to the one you have connected to the U2D2. Only keep protocol2.0 selected, have all baudrates ticked, and change the ID range to scan from 0 to 15. These settings will decrease the scanning time from a few minutes to a few seconds.
5. Scan for connected servos. You should see disco lights on the U2D2 interface. If you do not see disco lights then you are probably not connected to the correct COM port.
6. The system should find a servo – write down the baud rate, ID and COM port.




7. Using the interface in the top right of the window, Put the actuator in position mode and move the servo by hand. You should see the encoder value changing.
8. Let go of the servo horn and activate the torque. Now click in the circle interface to move the servo.
9. Try getting the servo to move in velocity control mode. You should see that it can now spin continuously.
10. Don't forget to disconnect the connection in the Wizard software when you are done (or you won't be able to complete the next step).

## Task 3 – Communicating with the Actuator via Matlab

Now let's try and talk to the Dynamixel via Matlab, using the SDK we just installed. This will unlock the potential for automated control of the actuator, and automation is a key part of robotics. As you finish each section, show a GTA your working system. They will tick off this task for your group.

TIP: If your code opens the port and then encounters an error before the port is closed, Matlab will fail to connect to the actuator when run again. Simply unplug and plug-back-in your USB cable to fix this.

**Read Encoder Position**

We are going to start as in Task 2, by reading the Actuator's position.

1. Run the code 'Lab1_Read_Dyn_Position.m'. (Change the variables *DXL_ID, BAUDRATE* and *DEVICENAME* to the values you wrote down in Task 2) Try turning the servo while observing the output.
2. Reading the encoder position is something that cannot be done with regular servos. Why is this? Why is it important to be able to do this?

Look at the 'Control Table Address' part of the code (lines 28-30) – notice how each command name corresponds to a number. These numbers are variables of the actuator that are stored in local memory (within the actuator). You can find the complete control table on the online documentation for the XM430-W350 actuator. Note that the control table varies for different Dynamixel models (as they have different features).
https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/#control-table

**Step Demands**

3. Using the online documentation, look up the control codes for activating the torque on the Dynamixel and making it move to a position.

4. Save the code example with a new file name and modify it to make the Dynamixel move to 0deg and then 180deg. Use the *pause(1)* command after calling each motion command to make sure the actuator has time to get to the target.

5. Figure out how to change the moving speed. Hint: You only need to change one single local variable to achieve this.

6. Use matlab's tic and toc functions to measure the execution time of a read function and write function (just the single line). Write a loop to average 100 sample measurements. What operating frequency does a system have if it reads and writes in every loop iteration? What frequency would a system have if it uses 5 servo motors?

7. Change the baudrate from the Wizard and observe whether the execution times change. Try at least 9600, 57600, 115200. Do not increase the baudrate beyond 3000000, as that is the hard limit for mac users. **3000000 will be the maximum allowed baudrate for this course.**

**Data Logging with Step Demand**

8. Now we're going to make use of the encoder data from step 1. You need to write some code that will read the encoder position of the actuator as it is commanded to move between 0

deg and 180deg (or more steps if you like). There are various ways to achieve this. You can trigger the step demands in an open loop fashion (send a position command, wait some time, then send another command) or you can use the encoder position to trigger the next target (keep reading the position and send the next command if the current position is close enough to your desired target).

9. Set the moving speed to maximum (setting it to 0). Plot the logged data and save the image. Discuss the actuator response to the step demand within your group. Why is it also not a step response? Why is the position data not a completely straight line between targets?

**Trajectory Tracking [Bonus]**

10. If you have time before the end of the session, see if you can write some code to get the Dynamixel to track a sine wave position demand. This means that if you track the servo moving repeatedly between 0deg and 180deg, the resulting plot should resemble a sine wave. This will mean updating the goal position frequently.