Workshop “Interfacing & Controlling your Robotic Hand”

Prerequisites:

- Install the *Dynamixel Wizard* to assign unique IDs to your Dynamixel motors: https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/

- Update or install *Python3* on your computer using your preferred way.
  On **Linux**, you can use *apt* and run the following commands from the terminal:
  ```
  sudo apt update
  sudo apt install python3
  ```

- Install the dynamixel SDK either from the Robotis Github (https://github.com/ROBOTIS-GIT/DynamixelSDK/tree/master/python) or using a *pip* package. On **Linux**, run the following commands from the terminal:
  ```
  sudo apt install python3-pip
  sudo pip3 install dynamixel-sdk
  ```

Setting up the Dynamixel Servos:

To communicate with the dynamixel servos, each motor must have a unique ID. As the factory settings are all identical, we must first manually adjust the IDs of each motor. Therefore, we need the Dynamixel Wizard.

- Connect the *U2D2* to your computer using the USB cable. Connect the *U2D2* to the *U2D2 Power Hub Board* (PHB) using a three-wire cable. Connect one dynamixel to the *PHB* using another three-wire cable. Provide Power to the *PHB* using the 12V Power supply and flick the power switch, so the red LED lights up.

- Start the *Dynamixel Wizard*. You will now need to connect one motor at a time to set the ID. If multiple motors with the same ID are connected, the *Wizard* will not be able to communicate with any of them.
  After startup, click on *Options* to set some filters for the scanning of the motors. In the *Scan* tab, select *Protocol 2.0*, the USB port you connected the *U2D2* to, *57600 bps & 3Mbps* and the ID range of 0 to 12. Click on OK and hit the *Scan* button. The software scans for available Dynamixel motors and it should detect and connect to your connected motor.
  In the table, select the row indicating the ID (Address 7) and adjust the ID in the bottom right corner. Make sure to label the motor ID physically on the motor. This way you know

which motors are connected to which fingers later.
Also select the row indicating the baud rate and change the baud rate of the motor to 3 Mbps.
After changing the IDs of a motor, you can connect your next motor either directly to the *PHB* or by daisy chaining it to the previously connected motors.

Your motors should now all have a unique ID which can be checked by connecting all of them in a daisy chain and using the *Dynamixel Wizard* to scan and connect to all of them. If all motors are detected by the software, you are good to continue.

Running the example code

To run the example code, connect two dynamixels with the IDs 1 and 2 to the *U2D2* and run the *example.py* script. You will get prompted to move the fingers (in this case just the motors) to the initial position and hit enter. This will calibrate the motors to the initial position. The motors will then move to a defined position, wait for one second and move back to initial position.
Check the code to see how the framework is set up in general. You will most likely not have to deal with every part of these files but it might be helpful to have an overview.

Adjusting the framework to your desires

To adjust the code framework to your hand specifically, certain parts of the code have to be rewritten or created from scratch. Feel free to also adjust other parts of the code to your likings.

gripper_defs.yaml

Here you define the structure of your hand by defining each finger with its according joints, tendons and motors.
finger1 shows an example of a finger with two joints and one antagonistic motors connected to each of the joints. *finger2* defines a finger with three joints and three antagonistic motors.

finger_kinematics.py

This code defines the kinematics of your fingers and how the desired tendon length is calculated for each tendon based on the joint angles of your finger. You will have to adjust the mathematical formulas based on your joint design and the calculation of the total tendon length, based on your chosen tendon routing.

gripper_controller.py

The two most important changes you will have to make are the calculation of the motor positions based on the desired joint positions and the calibration process.

The calculation of the motor positions is performed in the function *pose2motors()* where you will have to extend the calculations for each of your fingers, or for each finger that has different kinematics.
The calibration process is defined in the function *init_joints()*. With the current implementation, a user can move the fingers or the motors to the initial position and the motor positions of this pose are then read. There are more efficient ways to initialize the hand that do not require someone to move the fingers themselves. Can you figure out a clever way to move the motors to the same, absolute position repeatedly?

dynamixel_client.py

You will not have to adjust this file.

cal.yaml

This calibration file is written automatically. Feel free to also store other calibration information in this same file.

example.py

This file shows you a very simple example of how you can use the code framework. Copy, edit and extend this example to create the applications you require. Control defined grasping motions, follow trajectories or create a graphical user interface to control your hand - there's no limits.