

Crash course on ORCA software etc.

Aristotelis Sympetheros

RWR Teaching Assistant | ORCA Team

06.10.2025, Zurich



Goals and announcements:

- Understand the core software layout and what will be needed to use in task 1,2 and X.
- Hand check-up → All hands should be finished and properly working, calibrated etc.
- No tasks for next week → Use this time to plan and organize for task X.

Dynamixel Wizard 2.0 - Basics

- **Officially** Supported on x_86 architectures (not apple silicon devices)
- Changed ID or other various settings, check servo conditions, check errors (OL, OH, ES etc.).
- Can't have Dynamixel Wizard open **and** run a script to control servos (port is used by Wizard already).

Linux basic commands

Basic commands need to navigate through terminal, aws instance etc.

- **pwd** → Show current directory/path
- **mkdir dir** → Make directory dir
- **cd dir** → Change directory to dir/Enter that directory
- **cd ..** → Go up a directory
- **ls** → List files of current directory

Linux basic commands

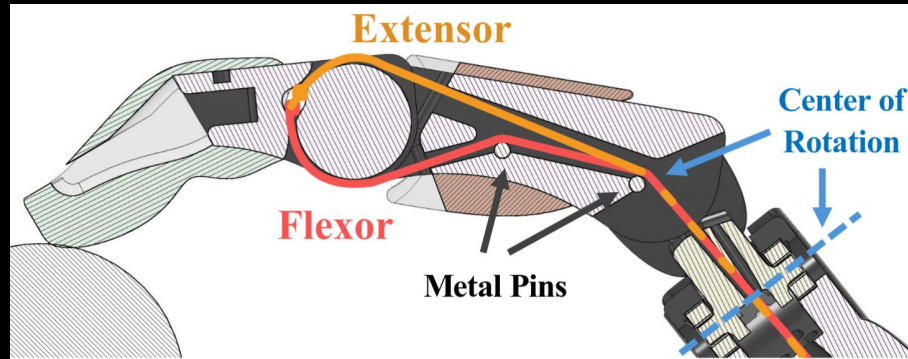
Basic commands need to navigate through terminal, aws instance etc.

- **chmod 775 file** → Change mode of file to 775 (read write execute permissions)
- **ls /dev/ttyUSB* /dev/ttyACM*** → Find available USB ports
- **sudo** → "superuser do". Run programs with root (administrator) privileges
- Many more → For assignment 2 you will learn about the needed ROS 2 commands such as colcon, source etc.

Find many more [here](#) and [here](#) or just googling them

ORCA Calibration

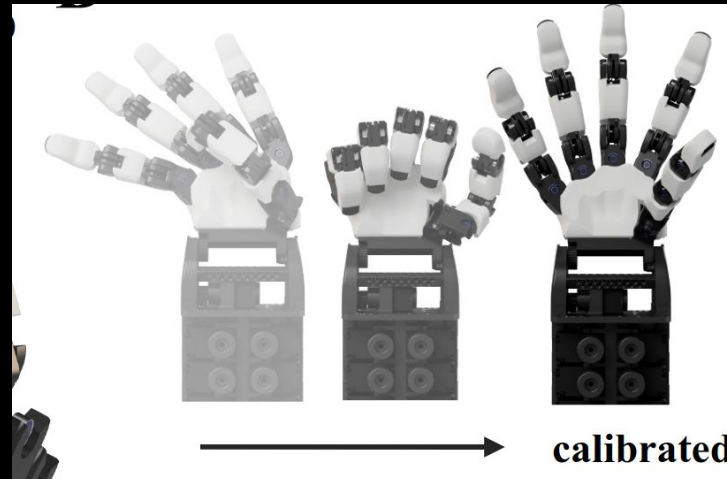
All joints have hard stops by design → Provide us a known Range of Motion (ROM)



Tendon routing is very close to center of rotation → No non-linearities → linear correlation of servo angle and joint angle

ORCA Calibration

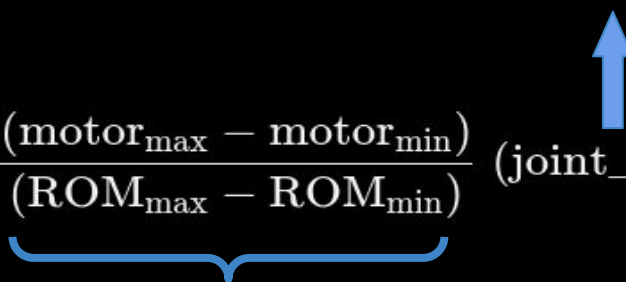
Calibration: Move each joint to its physical limits/hard-stops and record ROM of servo. Then we can interpolate: servo angle \rightleftharpoons joint angle



Interpolation:

$$\text{motor_pos} = \text{motor}_{\min} + \underbrace{\frac{(\text{motor}_{\max} - \text{motor}_{\min})}{(\text{ROM}_{\max} - \text{ROM}_{\min})}}_{\text{Slope/ratio found from calibration}} (\text{joint_pos} - \text{ROM}_{\min})$$

Desired Joint angle



- This creates a linear mapping between the physical joint range (Joint-ROM) and the Servo/Motor range (found from calibration).
- The same formula is inverted when converting motor positions back to joint angles.

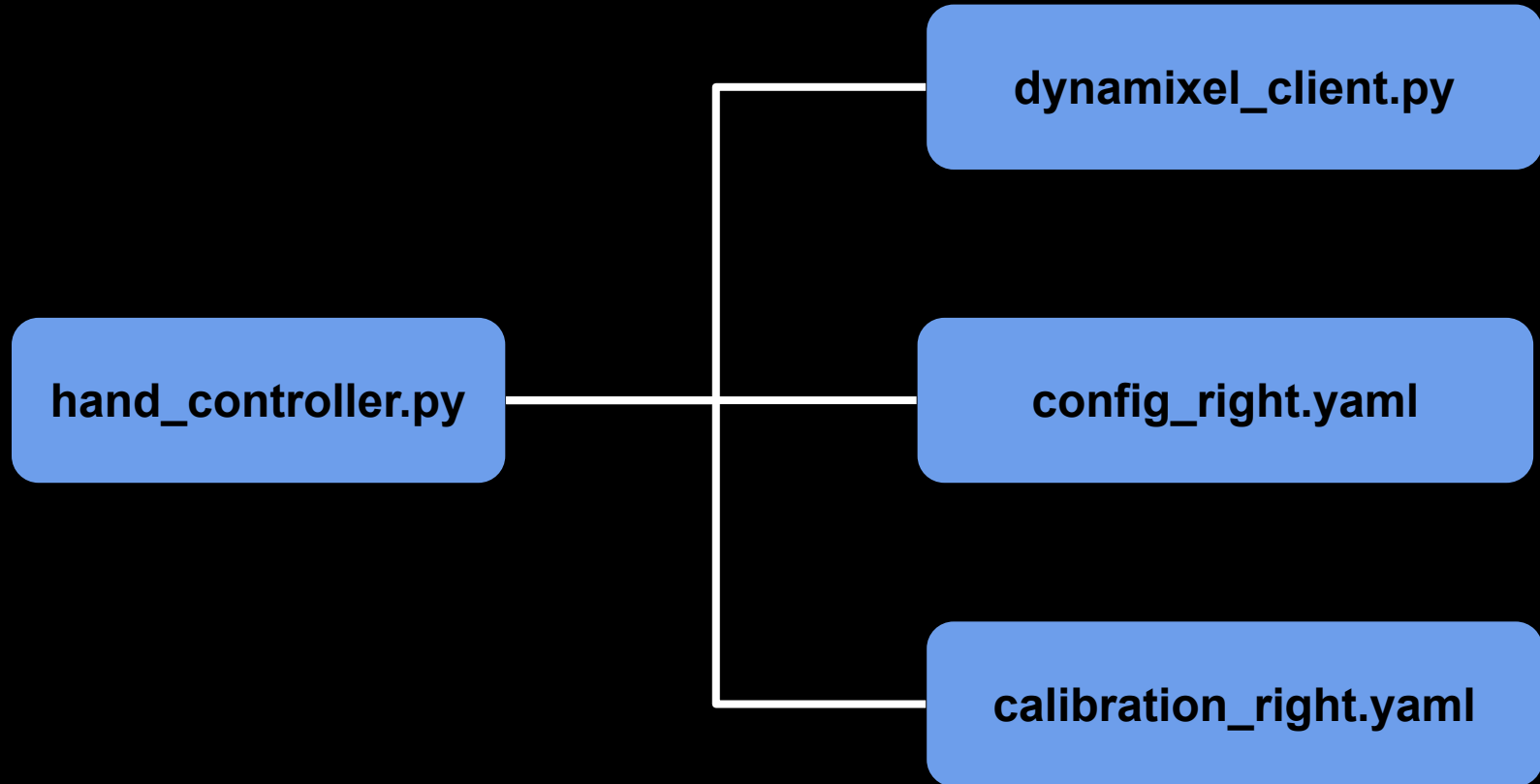
ORCA Calibration

Calibrate + tension x 2-3 times → Until calibration runs and no tensioning is needed.

Calibration should be runned after some hours of use or by feel (depends on how aggressive teleop/policy is).

Do not tension too much → Increases friction and creates “springiness” on fingers.

ORCA Software (SRL Repo) → Very high level



ORCA Software (SRL Repo)

config_right.yaml : Contains necessary info about setting up

- Port U2D2 board is connected
- Servo - hand joints mapping
- Current and P gains of servos
- Calibration order and parameters

If assembly guide followed properly, except from changing the port, no other changes should be required.

ORCA Software (SRL Repo)

calibration_right.yaml: Saves the linear mapping calculated from calibration of servo angle to joint angle.

```
1 calibrated: true
2 joint_to_motor_ratios:
3   1: 0.01855343317650255
4   2: 0.014906202642814073
5   3: 0.014869716992246292
6   4: 0.016551652701286067
7   5: 0.01580255874986858
8   6: 0.01697698373792709
9   7: 0.01463724467800479
10  8: 0.015567832050028619
11  9: 0.01519152306936079
12 10: 0.016289597650022268
13 11: 0.016786863088761865
14 12: 0.01266156034108851
15 13: 0.015912859273187977
```

```
14 12: 0.01266156034108851
15 13: 0.015912859273187977
16 14: 0.01739813119060114
17 15: 0.015350172613909705
18 16: 0.017640779060684873
19 17: 0.0338923575426778
20 motor_limits:
21 1:
22 - 1.1857671490356005
23 - 3.393625697039404
24 2:
25 - -0.8084078752157329
26 - 1.4275225212063778
27 3:
28 - 2.8455343615278643
29 - 3.7674568150471344
30 4:
```

ORCA Software (SRL Repo)

dynamixel_client.py : Provides low-level setup and communication with dynamixel motors. Typically should not be used directly.

- Manages motor state reading and command writing
- Runs a background update loop at fixed frequency (useful for RL)
- Handles basic configuration & safety (IDs, modes, homing)

ORCA Software (SRL Repo)

hand_controller.py : OrcaHand class is used to abstract hardware control of the hand with simple high level control methods in joint space.

High-level joint I/O (should mostly use these):

- **get_joint_pos()** → {joint_name: pos} | Get current joint angles.
- **set_joint_pos(joint_pos: dict)** → Command joints in degrees; auto clips to ROM, handles inversion & ratios.

ORCA Software (SRL Repo)

Motor Information:

- `get_motor_pos()` → Raw motor positions (rad).
- `get_motor_current()` → Motor currents.
- `get_motor_temp()` → Motor temperatures.

Demo some code on VS Code

Try out ORCA core software to get familiar with it!