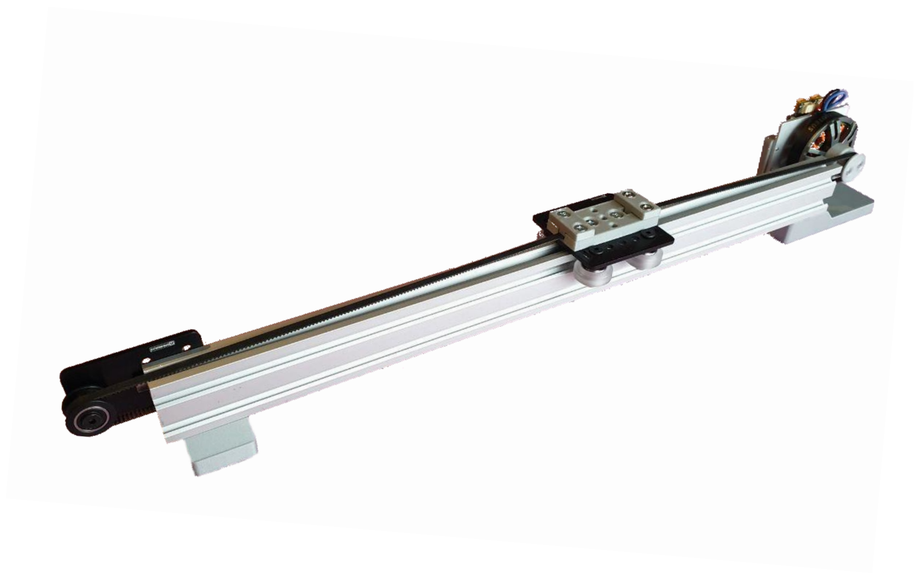
Robotics Practicals 2025 - Lab Worksheet

LiniX TP



Assistant

Furrer Nicolas

Peteut Nicolas

Lead of Rehassist

Bouri Mohamed

Students version from Monday 28th April, 2025 at 11:35

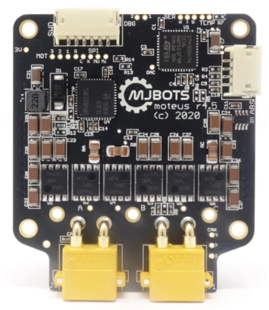
Contents

1 Lab Description

1.1 Aim

The objective of this lab is to have an introduction on transmission technologies used in industry. A deeper look will be done on the belt driven one using a brushless dc motor and a linear axis. The controller used is a moteus one from mjbots `https://mjbots.com/products/moteus-r4-5`, figure 1a. This is an open source project with great capabilities with the firmware available on github, `https://github.com/mjbots/moteus`.

At the end of the lab, the students will have done the assembly of a linear belt driven axis. They would better understand the brushless dc motors with PD/PID controllers and know the main parameters to have good performances on such a system.


(a) Moteus controller, revision 4.5.

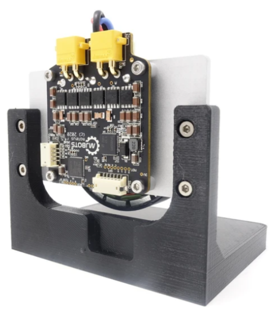

(b) Mjbots developper kit.

Figure 1: Moteus controller used in this TP.

1.2 Structure

The practical session, 4 hours, is organized as follow :

- Introduction to industry components.
- Linear axis assembly.
- Experimentation using :
  - moteus GUI : have a look on P / D factors effects, tune a PID controller and compute a gravity compensation feedforward.
  - python scripts : evaluate performances over load and different angle.

The grading of this practical is done through Moodle on the Quiz LiniX link, only the group leader have access to it (If there are any problem with the Quiz please inform the TAs during the practials). The quiz is separate in multiple part :

- Theoretical questions
- Questions about linear guide solutions
- Questions about the assembly process
- Questions about the tuning of the controller and the experiments

The quiz is design to be done alongside the practical. Each of the parts that directly refer to the quiz will be preceded by : 🗎 **MX** and fill in gray color(X is the question's number).

2 Industrial Linear Guide Solutions

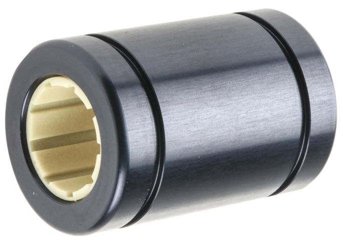
The role of a linear guide is to support and guide a load. Depending on the use, an ideal guide would have the following characteristics.

M4 From the introduction given by the teaching assistant, what are the ideal keys of such a Linear Guide Solution? *Select all the correct answers.*

- ☐ High reliability
- ☐ Low stiffness
- ☐ Good breakage resistance
- ☐ High friction
- ☐ High static load capacity
- ☐ No environment resistance
- ☐ Complex assembly
- ☐ Expensive

Since none of the existing linear guide has all the ideal characteristics, there exists planty of solutions. The best one has to be choose regard to the situation. Here are some existing ones.

2.1 Plain Bearing


(a) Plastic

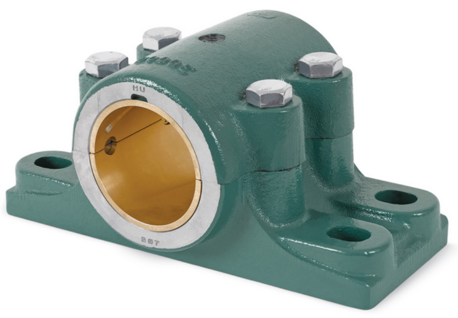

(b) Bronze

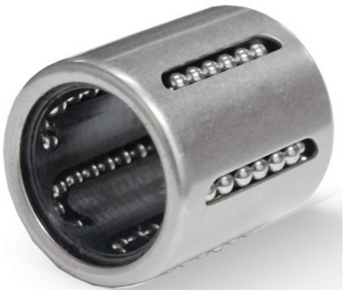
Figure 2: Different types of plain bearings.

This guide is the simplest one. It does not have any internal moving parts and it is only composed of a bearing surface. A good example is just a shaft rotating in a hole.

M5 Select all the benefits from the following features list :

- ☐ No wear
- ☐ Low noise
- ☐ High load-carrying capacity
- ☐ High accuracy
- ☐ Require friction resistance material
- ☐ Good in Contaminated environment

2.2 Linear Ball Bearing


(a) Linear ball bearing

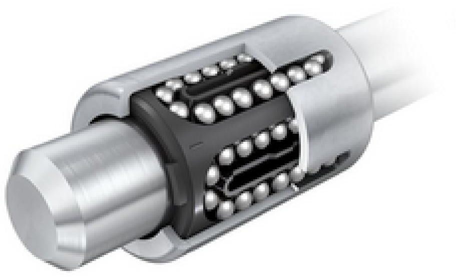

(b) Cut of linear ball bearing to see balls recirculating

Figure 3: Linear ball bearing examples.

These are mainly used when a high accuracy is needed. They can carry a smaller load than the roller linear guides but the efforts in industry tends to smaller the gap between them. Ball guides continue to be more popular than roller linear guides and consequently the prices tend to be lower.

M6 Select all the benefits from the following features list :

- ☐ High load capacity
- ☐ No noise
- ☐ High accuracy
- ☐ Work in Contaminated environment
- ☐ Need lubrication
- ☐ Low wear

2.3 Track Roller Guide

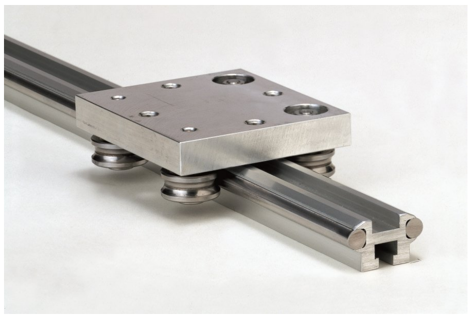
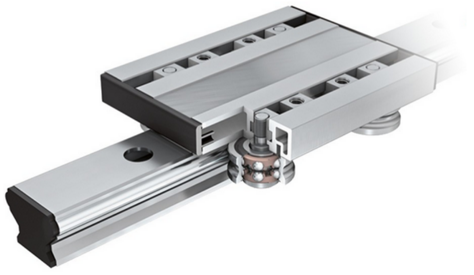



Figure 4: Track roller configuration examples.

Where dirt is an issue, the track roller are clearly the best solution. They offer low coefficients of friction and there are different systems to suit a wide range of loads.

M7 Select all the benefits from the following features list :

- ☐ No wear
- ☐ High noise
- ☐ High load-carring capacity
- ☐ Good in contaminated environment
- ☐ Need lubrication
- ☐ No accuracy

2.4 Summary

To summarize all of these informations, considering the right linear guide for your motion is essential to have a good design longevity. There is a lot of different parameters to take into account such as price, material to used, load, environment, precision and friction. For example, the friction can be multiplied by almost 500 between two different guide solutions[1].

3 Assembly

To well understand the overall mechanics, let's begin the assembly. All the instructions are available in a given video.

M8 From the assembly video, what are the three main aspects to take care on this design? *explain why*. **Then ask an assistant to check your assembly!**

4 Experiments

In this section, some experiments will allow you to take charge of the linear axis system.

4.1 Installation

To make the practical works you need the following programs:

- Python 3.X
- *moteus_gui* library (*pip install moteus_gui* in a terminal)
- An IDE to edit the *linix_tp.py* python script

To do these installation on the PC of the room do the following steps:

- Install python form the Microsoft store.
- Install VSCode form the Microsoft store.
- Back in the terminal type *python -m pip install moteus_gui*.
- In VSCode install the python extension

If you have any issue installing the TP on the room computer please ask an assistant.

4.2 GUI PID Explorations

In this part the GUI will be used to easily tune and understand PID parameters. Let's open the GUI and explore the physical behavior behind P and D gains.

Hints

In general to produce a nice plot with the GUI :

[1] https://euro-bearings.com/blog/a-quick-guide-to-choosing-the-best-linear-motion-product/

1. Plot the metric you want.
2. Put axis and plot title *icon on the left of save one*
3. Do a stop command to reset fault if needed.
4. By hand, push the carriage or turn the motor shaft to put the motor in a suitable initial position.
5. Do a position command.
6. Pause the graph and use the save button *or do a screenshot*.

4.3 GUI Opening

To power the controller follow :

1. Put the carriage at the middle of the axis.
2. Plug the fd can usb adapter on the controller and on your laptop.
3. Plug in the power adapter on the controller first.
4. Plug in the power adapter on the wall plug.

Note

The different connectors are shown in figure 5 and both are present twice. It does not matter the one you will use.

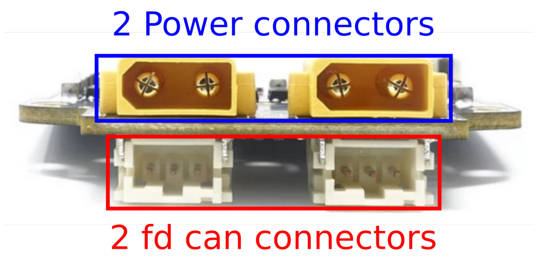


Figure 5: Moteus controller connectors.

To open the moteus GUI windows, run the following command in a terminal :

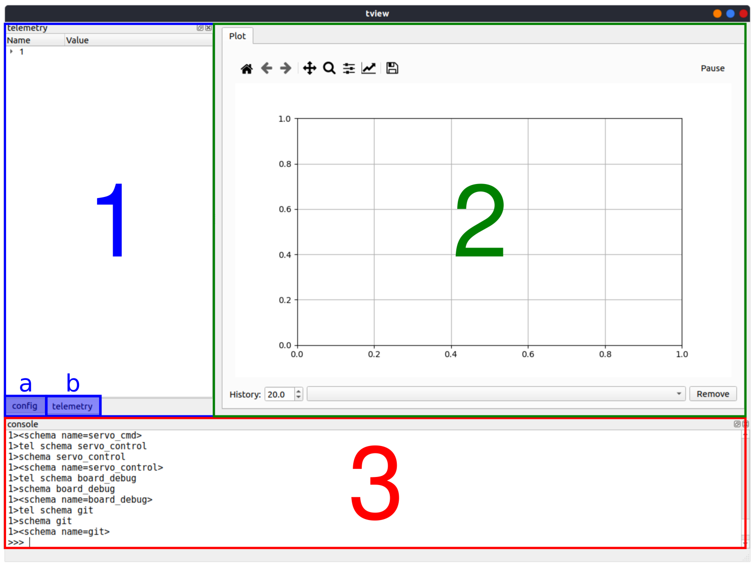
```
    python -m moteus_gui.tview --devices=1
```

If the installation is working well, a GUI, such as the one shown on figure 6 is opening. The GUI is splitted in different parts as follow.

Warning

Note that the controller will be detected, and all the config / metrics found, only if it is connected and power on.

1. In this first sector there is two different tabs. Click on the arrow on the left of the id, here 1, to expand the values :
   - 1a - config : which is listing all the different parameters that can be changed. All the values actually set can be checked in this tab (like PID gains, min and max position and so on ...).
   - 1b - telemetry : which is listing all the different metrics that can be plotted.
2. This is the plotting area. Each telemetry metrics can be plotted by right clicking on its name and choose plot left / right depending on where you would place the axis scale. You can pause it, remove some plotted values, save the plot and so on ...
3. The last part is the console one where the different command will be send.



```
1><schema name=servo_cmd>
1>tel schema servo_control
1>schema servo_control
1><schema name=servo_control>
1>tel schema board_debug
1>schema board_debug
1><schema name=board_debug>
1>tel schema git
1>schema git
1><schema name=git>
>>>
```

Figure 6: Presentation of the moteus GUI.

The complete documentation can be found on the mjbots' github. I suggest you to go read the part on "Theory of Operation" to well understand how the following commands works. And since we will use only a few commands, here are the informations about them.

To stop the controller and reset default status :

```
    d stop
```

To set :

```
        d pos <pos>  <stop> <max_torque> [options...]
```

- <pos> : The desired position in revolutions
- <stop> : Value to put at 0 to make the motor stop at the value set in <pos>
- <max_torque> : Never use more than this amount of torque when controlling

Each optional element consists of a prefix character followed by a value. Permissible options are:

- v - Velocity limit : configure the maximum velocity of the motor the value is given in [Hz]
- a - Acceleration limit: configure the maximum acceleration of the motor the value is given in [Hz^2]
- f - feedforward torque: configure the value of the feedforward torque in [Nm]

Here is an example of command to move the motor to the position -0.5 with a torque maximum of 1.2 Nm and a maximum velocity of 5 Hz:

```
        d pos -0.5 0 1.2 v5.0
```

Hints

There is as software security on the controller. If the motor is out of its min / max position boundaries it will not move *by default they are set to -1.5 [turn] and 1.5 [turn]*. It means that it will have a fault if it begins out of these boundaries or will be stopped if it reaches them. This fault is shown in the metric servo_stats.fault :

```
fault.39 : 39
```

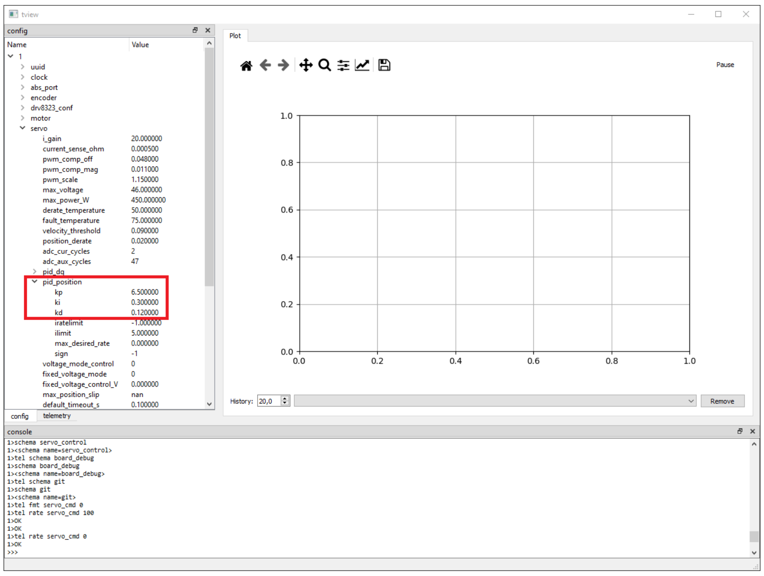
Do a stop command to reset it.

```
    d stop
```

4.4 Default value and calibration

Before sending command to the motor, it is important to verify the default value of the controller. This manipulation is done through the GUI in the config tab (see 7). Set the following default value :

- kp set to 0.0
- ki set to 0.0
- kd set to 0.0



Name	Value
1	
uuid	
clock	
abs_port	
encoder	
drv8323_conf	
motor	
servo	
i_gain	20.000000
current_sense_ohm	0.000500
pwm_comp_off	0.048000
pwm_comp_mag	0.011000
pwm_scale	1.150000
max_voltage	46.000000
max_power_W	450.000000
derate_temperature	50.000000
fault_temperature	75.000000
velocity_threshold	0.090000
position_derate	0.020000
adc_cur_cycles	2
adc_aux_cycles	47
pid_dq	
pid_position	
kp	6.500000
ki	0.300000
kd	0.120000
iratelimit	-1.000000
ilimit	5.000000
max_desired_rate	0.000000
sign	-1
voltage_mode_control	0
fixed_voltage_mode	0
fixed_voltage_control_V	0.000000
max_position_slip	nan
default_timeout_s	0.100000

```
1>schema servo_control
1><schema name=servo_control>
1>tel schema board_debug
1>schema board_debug
1><schema name=board_debug>
1>tel schema git
1>schema git
1><schema name=git>
1>tel fmt servo_cmd 0
1>tel rate servo_cmd 100
1>OK
1>OK
1>tel rate servo_cmd 0
1>OK
>>>
```

Figure 7: Location of the default values of the PID controler.

To be sure that the motor will correctly work in the assembly a calibration is necessary. Before doing this calibration, make sure to remove the belt from the motor and make sure that the motor can rotate freely. Then open a terminal and run the following command :

```
        python -m moteus.moteus_tool --target 1 --calibrate
```

4.5 P Gain

By setting P gain only, the motor will act as a mechanical springs and makes it oscillate around the desired position with the following behavior :

- Converge to the desired position if Kp's value is under the stable limit.
- Continuously oscillate around the desired value if Kp's value is equal to the stable limit.
- Diverge from the desired position if the Kp's value is bigger than the stable limit.

To see theses behavior, try to set different values of Kp in the config tab (Figure 7). After the Kp is set, you can run the following command to move the carriage.

```
    d pos <pos>  0 1.5
```

where

<pos> $\in [-1, 1]$	Target position

To better see the spring behavior, When the carriage is stabilized, you can try to manually move it and check the behavior. Increasing Kp will increase the spring stiffness.

Warning

You can test the effect of the Kp without the load. During theses experiments, be ready to input the d stop command if the motor diverges

M9 Explain why there is this huge behavior difference between the two settup *with and without belt fixed on the motor.*

M10 Plot the behavior of this system with Kp values such that :

- it converges
- it continuously oscillates
- it diverges ***take care*** *to be able to quickly run "d stop" to prevent breaking something*

Hint : it will highly depend on the velocity and torque limitations. If you cannot manage to make it diverge, explain the reason why.

4.6 D Gain

By setting D gain only, it will act as a mechanical damping on the motor. To inspect this behavior, the experiment is about the same as the previous one.

The goal here is to set a small Kd value to see the damping effect. Set the Kp, and Ki to 0.0 and then increase the kd (In the config tab Figure 7), then run the following command in the GUI to make the carriage hold the 0 position:

```
    d pos 0.0 0 1.5
```

Without any Kp value, the carriage will not move by itself. But moving it by hand will make you feel the damping effect. It can be increased by increasing the Kd.

M11 Plot the behavior of such a system using at least velocity and torque. Does it behave as expected?

4.7 PID Tuning

By default, there is only PD gains set. Try this configuration with a step position command with :

- Start position around -1.0 [turn], *set by hand*
- Target position around 0.0 [turn]
- Velocity at least 15.0 [turns / s]
- Maximum torque = 1.5 [Nm] *The motor has a peak torque of 1.7 Nm.*

Now tune your PID values, you can try using the following manual tuning and trials. The modification of those parameters is done through the GUI in the config tab. Do not spend more than 30mn on that part.

1. Set Kd and Ki to zero

2. Increase Kp until it critically oscillates.
3. Set Kp to around half of the critical values find in step 2.
4. Increase Ki until the steady state error is canceled. Increasing it too much will cause instability.
5. Finally increase Kd to minimize the overshoot.

Repeat steps 2 to 5 until you are happy with your tuning. This method can be found on robotsforroboticists[2] website. Finally plot your tuning.

M12 Plot the result of your PID tuning and write the values of Kp Ki and Kd.

4.8 Performances Benchmark

For the next experiments a python script *linix_tp.py* will be used in place of the moteus GUI. The goal will be to have a look on the effect of frictions over the mechanism's performances.

The python script will let you set the previous PID parameters that you found in the function *set_params, line 266*.

```python
        async def set_params(stream: moteus.Stream, save: bool = True
):
        """ The goal of this function is to reset the configuration
values.

        Args :
        stream : moteus stream connected to a specific motor
        save : if needed, save parameters in permanent memory
        """
        params = {}
        params['servopos.position_min'] = -1.5  # default : -1.5
        params['servopos.position_max'] = 1.5  # default : 1.5

        params['servo.pid_position.kp'] = 0.0
        params['servo.pid_position.ki'] = 0.0
        params['servo.pid_position.kd'] = 0.0

        await modify_params(stream, save=save, **params)
```

The different parameters to choose are in the main call *from line 284 to 314*. If you have any doubt on the parameter meaning, do not hesitate to call a teaching assistant.

```python
        if __name__ == "__main__":
        # Set the different parameters to send to the main
        settings = {}

```

[2] http://robotsforroboticists.com/pid-control/

```
        # min : 0.25 (frequency = 1 / period_time => freq_max = 4Hz)
        settings['period_time'] = 2.0
        settings['nb_periods'] = 4
        settings['amplitude'] = 1.0
        settings['command_type'] = 'position'  # 'position' or '
    velocity'
        settings['mode'] = 'run'  # 'stop', 'run', 'reset'
        settings['feedforeward'] = 0.0

        # to plot the position in degree instead of turn
        settings['translate'] = True

        # Type of signal : 'rectangular', 'triangular', 'trapezoidal
    ', 'sine'
        settings['signal_type'] = 'rectangular'

        # To show and save plots
        settings['plot'] = True
        settings['filter'] = True  # show filter plot
        settings['save'] = True
        settings['fig_options'] = {
                'signal': {
                        'name': 'default_name_signal_plot',
                        'extension': 'png'
                },
                'result': {
                        'name': 'default_name_result_plot',
                        'extension': 'png'
                }
        }
```

4.8.1 Test of the PID controller

M13 With the help of the script test the system with the rectangular position profile with your PID parameters.
Plot the results and conclude on them.

4.9 Feedforward : Gravity compensation

A feedforward control is uses to improve the system performance by taking preemptive action baed on a known disturbances rather than letting the system react to it. In the case of this practical, Feedforward control will be use to compensate the gravity in the system.

To be able to test the feedforward, first, modify the setup to have an angle of about 35 degree with the help of the two 35 degree attachment (ask an assistant to measure the angle of your setup).

Then with the help of the Figure 8, answer the question M13 and M14.

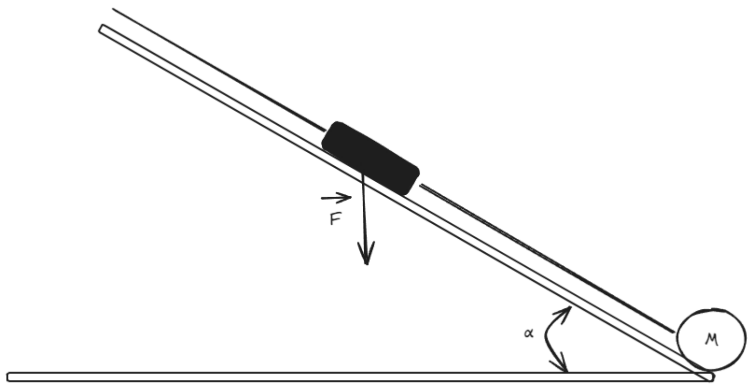


Figure 8: Simple force diagram

M14 Find algebric formula for feedforward torque in function of the angle α, the masse of the carriage m and the raduis of the pulley r.

M15 Compute the feedforward torque needed to compensate the gravity with $\alpha = 35^{\circ}$, $m = 523g$ and $r = 10mm$.

To try your value of feedforward torque :

- Mount the two additional masses on the carriage (Ask a teaching assistant for the masses)
- Set Kp, Ki and Kd to 0.
- Use the following code where X is the value of the feedforward torque in [Nm]

```
        d pos 0.0 0 1.5 fX
```

M16 Plot the position and the torque of the system at different position set by hand. Please comment.

4.9.1 Feedforward with rectangular position profile

M17 In this subsection, test the feedforward control with a rectangular position profile. For the PID parameters please use $kp = 2$, $ki = 20$ and $kd = 0.07$. Test the system with the following conditions :

- Without the masses with the rectangular position profile
- With the masses with the rectangular position profile, with $ki = 20$ and $ki = 0$.
- With the masses and the feedforward active with the rectangular position profile with $ki = 20$ and $ki = 0$.

Plot the results and conclude on them.
Hint: If the system cannot follow the position command try to change the *default_velocity_limit* from *math.nan* to *15*.

Appendices

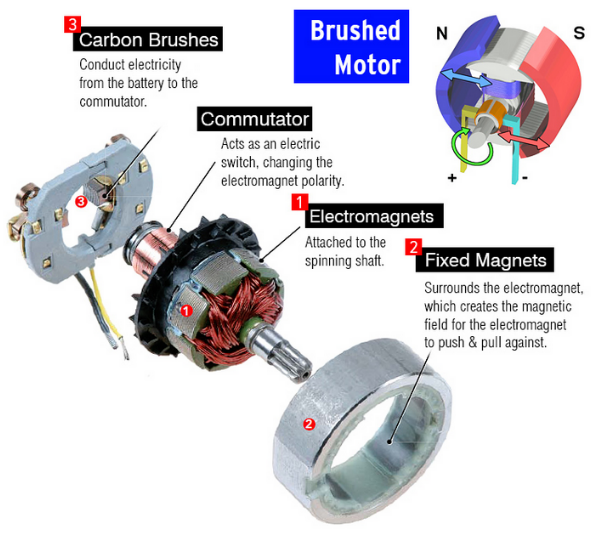
Appendix A Brushless DC Motors

A.1 Actuators and Sensors

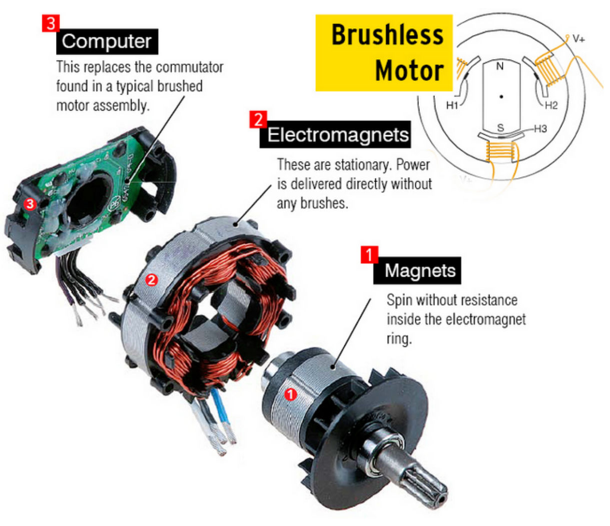
A brushless DC motors (BLDC) is used for linear axis. Before looking at the working principle of a brushless motor, let's see how the "brushed motors" are working.

The **brushed motors** derive their name from having brushes within the internal structure. In a brushed motor a stator with magnets surrounds a turning rotor. When connected to a current source, opposite polarities create a magnetic field torque. Because of this, the rotor starts turning around its axis. In other words, the rotor turns under the influence of the magnetic field such that its north pole will move toward the south pole of the stator (Figure 9a). Brushes are used to switch the polarity of the rotor magnetas the rotor rotates, so that the opposite poles on the rotor and stator are always close to each other (and therefore creating torque). This constant switching of the polarity is called "commutation". You can refer to this video for a very easy to understand explanation.

The **brushless motor** is so named since the magnetic field is provided by rotating magnets while the armature is stationary, thus eliminating the need for brush commutation. The flow of the electric current between the rotor and the stationary part of the machine is switched externally. As the rotor keeps turning, the commutator reverses the direction of the electric current (and thus the magnetic field) to create a one-way torque to keep the motor turning.


3 Carbon Brushes
Conduct electricity from the battery to the commutator.
Brushed Motor
N
S
Commutator
Acts as an electric switch, changing the electromagnet polarity.
+
-
1 Electromagnets
Attached to the spinning shaft.
2 Fixed Magnets
Surrounds the electromagnet, which creates the magnetic field for the electromagnet to push & pull against.

(a) Brushed DC motor's parts.


3 Computer
This replaces the commutator found in a typical brushed motor assembly.
Brushless Motor
V+
N
S
H1
H2
H3
2 Electromagnets
These are stationary. Power is delivered directly without any brushes.
1 Magnets
Spin without resistance inside the electromagnet ring.

(b) Brushelless DC motor's parts

Figure 9: Working principles of brushless and brushed DC motors[3].

[3] https://www.chainsawjournal.com/what-is-a-brushless-motor-and-how-does-it-work/

The difference between a brushed motor and a brushless motor is thus in how the commutation is done :

Brushed motors the commutation is done mechanically in the motor.

Brushless motor the commutation needs to be done by an external electrical circuit, and thus we need sensing.

A Hall-effect sensor which is a solid-state, magnetic field sensor is used to sense the rotation angle and thus control the current supply to the stator such that the magnetic field is in the right direction. In other words, the Hall-effect sensor can be used to measure the rotation speed, and the angle of the rotor, and then decide when the current should be applied to the motor coils to make the magnets rotate at the right orientation. The Hall-effect sensor employs the principle that when a conductor with current flowing through it is placed in a magnetic field, the magnetic field induces a transverse force on the charge carriers. This forcepushes them to the sides of the conductor-negative to one side and positive to the other side. This creates charge on the sides of the conductor induces a voltage.

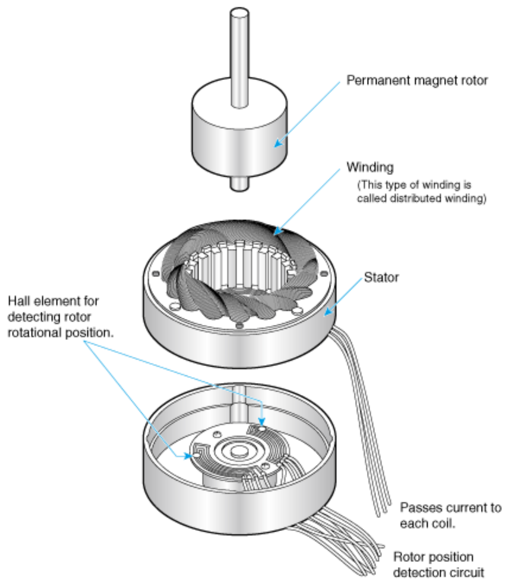

Permanent magnet rotor
Winding
(This type of winding is called distributed winding)
Stator
Hall element for detecting rotor rotational position.
Passes current to each coil.
Rotor position detection circuit

Figure 10: Structure of brushless DC motors[4].

Compared to other types of motors (DC brushed, stepper motors...), these ones have many advantages such as higher efficiency, high torque to weight ratio, compact size, increased reliability, low vibration and reduced noise. Thanks to these advantages, BLDC motors are often used in modern devices where low noise and low heat are required especially if devices run continuously. These primary distinctions are relevant to haptic interface design.

A.2 Encoder

To accurately determine the motor's shaft angle at any point in time, a position sensors is attached to one end of the motor. For such position sensors, several technologies exist such as optical encoders or magnetic sensing (using Hall Effect sensors) like the ones we are using.

[4] https://www.nidec.com/en/technology/motor/basic/00018/

Moteus controller uses an AS5147P [5]. It is a high-resolution, 14 bits, rotary position sensor for high speed (up to 28krpm) angle measurement over a full 360 degree range. When the motor is running, the magnetic sensor detects the position and direction the motor is spinning.

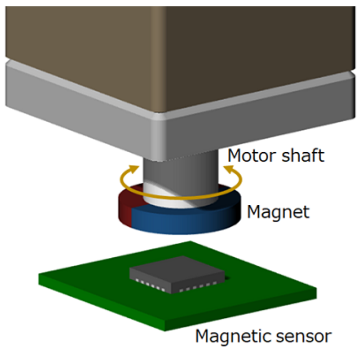


Figure 11: Magnetic encore assembly[6].