Hardware & Software interfacing for a compact rotary servo system

Hugues Garnier
Florian Collin

September 2023

Contents

Acknowledgements

The contents of this lab has been largely inspired and adapted from an initial version provided by Quanser Inc[1]. This is fully acknowledged.

[1] www.quanser.com

Part 1

QUBE-Servo 2 software and hardware interfacing

In this lab, we will mainly use the QUBE-Servo 2 with the inertia disc.
The work done during this 4h00 lab will be noted in a report and marked. You are asked to follow the instructions given below to write your report.

Instructions for writing your lab report

A lab report is a scientific document. It should be self-contained: that is, someone who has never seen the lab instructions should be able to understand the problems that you are solving and how you are solving them. All the choices you have made should be clearly motivated. Comment and explain all your plots so as to make it easy to follow your way of thinking.

Your lab report can be written in French or in English but **do not mix both languages**. It should be organized as follows:

- a general introduction specifying the objectives of the lab
- For each assignment or exercise:
    - ▷ a brief presentation of the expected outcomes
    - ▷ a description of the obtained results in graphical and or numerical form
    - ▷ a critical analysis of the results
    - ▷ a short conclusion
- a general conclusion explaining what has been understood during the lab and any difficulties encountered.

Sending your report to the tutor

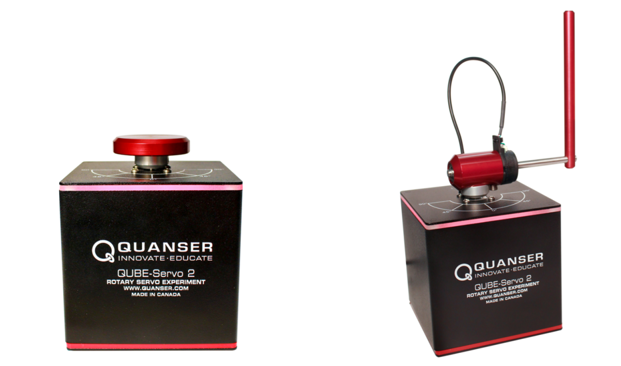
The report should be written in groups of two students and sent by email to the tutor in the form of a single pdf format file attached to the message by the deadline given by your instructor during the lab. Please indicate the following "subject" for your email when you send your report to the tutor: `Robotics_Lab_Names_LabGroupNumber"`

1.1 The QUBE-Servo 2 from Quanser

The Quanser QUBE-Servo 2, pictured in Figure 1.1, is a compact rotary servo system that can be used to perform a variety of classic servo control and inverted pendulum based experiments. The QUBE can be controlled by a computer via USB connection.
The system is driven using a direct-drive 18V brushed DC motor. The motor is powered by a built-in Pulse Width Modulation (PWM) amplifier with integrated current sense. Two add-on modules are supplied with the system:

- an inertia disc;
- a rotary pendulum.

The two modules can be easily attached or interchanged using magnets mounted on the QUBE-Servo 2 module connector.
Single-ended rotary encoders are used to measure the angular position of the DC motor and pendulum, while the angular velocity of the motor can be either estimated form the angular position encoder-based measurement or directly measured using an integrated software-based tachometer.



(a) QUBE-Servo 2 with Inertia Disc Module (b) QUBE-Servo 2 with Pendulum Module

Figure 1.1: QUBE-Servo 2 with the two different modules

Download of the QUBE-Servo 2 user manual and quick start guides
Download the zipped file `QUBE-Servo2_pdf_guides.zip` from the course website (in Section Robotique industrielle) and unzip it in the local disk folder `C:/temp/`.

1.2 Hardware and software interfacing

Topics covered

- Getting familiarized with the Quanser QUBE-Servo 2 hardware (sensor and actuator).
- Using Quanser QUARC software to interact with the QUBE-Servo 2 system.
- Encoder to measure angular position but also to estimate angular velocity.
- Tachometer to measure angular velocity.
- Sensor calibration.

Prerequisites

- Inertia disc load is on the QUBE-Servo 2.
- The QUBE-Servo 2 has been setup. If necessary, follow Step 2 to Step 4 as shown in the the QUBE-Servo 2 Quick Start Guide for details.
- You have the QUBE-Servo 2 User Manual. It can be useful for some of the experiments.
- You are familiar with the basics of Matlab and Simulink.
- If you come through some compiling errors in Simulink, refer to the Troubleshooting section at the end of this document.

1.2.1 Actuators and sensors

1.2.1.1 The actuator: the DC motor

Direct-Current (DC) motors are used in a variety of applications. As indicated in the QUBE-Servo 2 User Manual, the QUBE-Servo 2 has a brushed DC motor that is connected to a Pulse-Width Modulation (PWM) amplifier. See the QUBE-Servo 2 User Manual for details.

1.2.1.2 The sensors: the encoders

To get a useful reminder about encoders, watch the first 15 mn of the excellent video from Marc Derumaux:
`youtu.be/SCR5YJZVyKg?si=42JR-znCino9vCTs`

Similar to rotary potentiometers, encoders can also be used to measure angular position. There are many types of encoders but one of the most common is the rotary incremental optical encoder, shown in Figure 1.2. Unlike potentiometers, encoders are relative. The encoder count is reset to 0 every time it is powered. The angle they measure depends on the last position and when it was last powered. It should be noted, however, that absolute encoders are available.

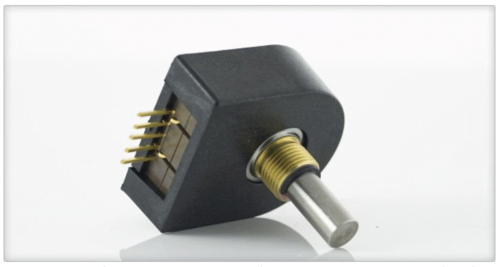


Figure 1.2: Digital incremental rotary optical shaft encoder

The encoder has a coded disc that is marked with a radial pattern. This disc is connected to the shaft of the DC motor. As the shaft rotates, a light from a LED shines through the pattern and is picked up by a photo sensor. This effectively generates the A and B signals shown in Figure 1.3. An index pulse is triggered once for every full rotation of the disc, which can be used for calibration or homing a system.

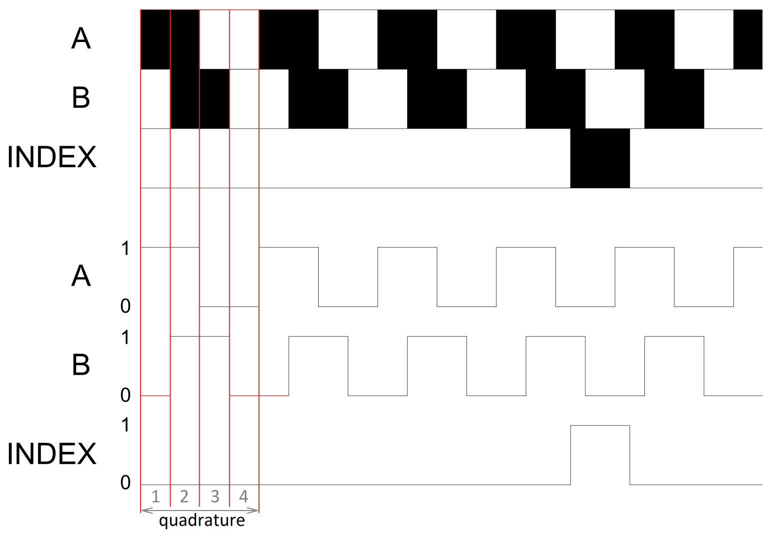


Figure 1.3: Optical incremental encoder signals

The A and B signals that are generated as the shaft rotates are used in a decoder algorithm to generate a count. The resolution of the encoder depends on the coding of the disc and the decoder. For example, a single encoder with 512 lines on the disc can generate a total of 512 counts for every rotation of the encoder shaft. However, in a quadrature decoder as depicted in Figure 1.3, the number of counts (and thus its resolution) quadruples for the same line patterns and generates 2048 counts per revolution. This can be explained by the offset between the A and B patterns: instead of a single strip being either on or off, now there are two strips that can go through a variety of on/off states before the cycle repeats. This offset also allows the encoder to detect the directionality of the rotation, as the sequence of on/off states differs for a clockwise and counter-clockwise rotation.

1.2.1.3 The sensors: the tachometers

A tachometer (revolution-counter, tach, rev-counter, RPM gauge) is an instrument measuring the rotation speed of a shaft or disk, as in a motor or other machine. The device usually displays the revolutions per minute (RPM) on a calibrated analogue dial, but digital displays are increasingly common.
Tachometers may be mechanical (analog) or electronic (digital). Mechanical tachometers use electrical and magnetic forces to measure rotation speed; though they employ principles of electricity, they do not need batteries. Electronic tachometers use electronic circuitry such as digital counters to determine RPMs. They use electric power, so they require batteries or other sources of current. An electronic tachometer uses a magnetic pickup positioned near a rotating engine part to produce electrical pulses at a frequency proportional to the engine speed. Circuitry in the meter converts the pulse frequency for the display of engine RPM.
In addition, tachometers may be contact or non-contact. A contact tachometer touches the rotating part you are measuring and is more precise. A non-contact tachometer uses light from a laser to illuminate a mark on the rotating equipment.

The interaction between the different hardware components of the QUBE-Servo 2 is shown in Figure 1.4. From the Figure, it can be observed that the sensor and actuator signals are interacting with the PC through the data acquisition (commonly abbreviated as DAQ) board where the signals are converted from the analog domain to the digital domain.

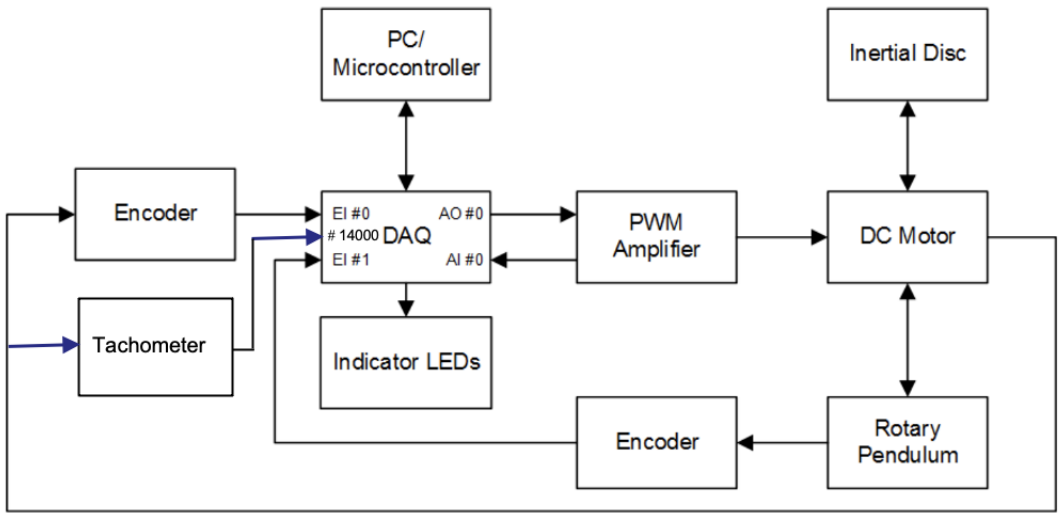


Figure 1.4: Interaction between the different hardware components of the QUBE-Servo 2.

1.2.2 QUARC software

The QUARC software is used to interact with the hardware of the QUBE-Servo 2 system. QUARC will be used to drive the DC motor and read the angular position and velocity of the inertia disc.
The basic steps to create a Simulink model with QUARC in order to interact with the QUBE-Servo 2 hardware through the DAQ are:

1. Make a Simulink model that interacts with the data acquisition board installed in QUBE-Servo 2 using blocks from the *QUARC Targets* library.
2. Build the real-time code.
3. Execute the code.

To know more about QUARC:

- follow the link given below to watch the first 15 mn of the video entitled *Getting Started with QUARC* where Peter from Quanser demonstrates how to interface with the QUBE-Servo 2 hardware, build a simple Simulink model and implement it on an actual system:
  `www.quanser.com/tutorial/quarc-essentials-hardware-interfacing/`

- Whenever necessary, you can also type `doc quarc` in Matlab to access QUARC documentation and demos.

1.3 Measuring the angular position from the encoder and driving the DC motor

The objective here is to build a Simulink model using QUARC blocks to drive the DC motor and then measure its corresponding angle, as shown in Figure 1.5.

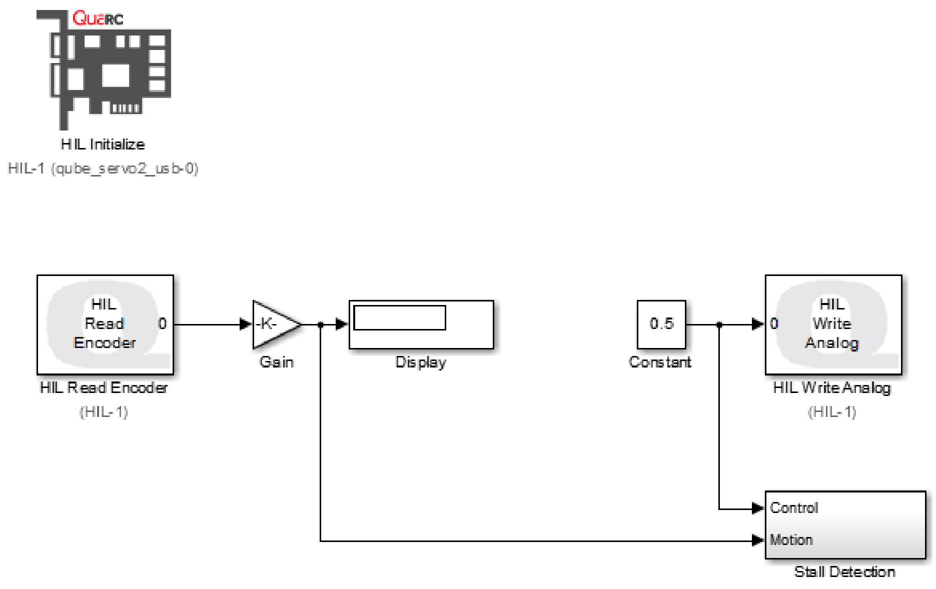


Figure 1.5: Simulink model used with QUARC to drive the DC motor and read angle on QUBE-Servo 2

1.3.1 Configuring a Simulink model for the QUBE-Servo 2

Follow the steps below to build a Simulink model that will interface to the QUBE-Servo 2 hardware using QUARC:

1. Open the the Quick Start Guide and follows the instruction to set up and connect the QUBE-Servo 2 to your PC USB port.

2. Open Matlab and then start Simulink by typing `Simulink` in the Command window or by clicking on its icon in the menu bar.

3. Make sure the QUBE-Servo 2 is connected to your PC USB port and the Power LED at the back of the QUBE-Servo 2 is lit green.

4. Create a new blank Simulink model (do not create a new blank QUARC model!) model by clicking on the icon in the Simulink Start page window or by going to `File` | `New` | `Simulink Model` item in the menu bar.

5. Go to `SIMULATION` item in the menu bar | Open the Library Browser window by clicking on its icon.

6. Expand the `QUARC Targets` item and go to the `Data Acquisition` | `Generic` | `Configuration` folder, as shown in Figure 1.6.

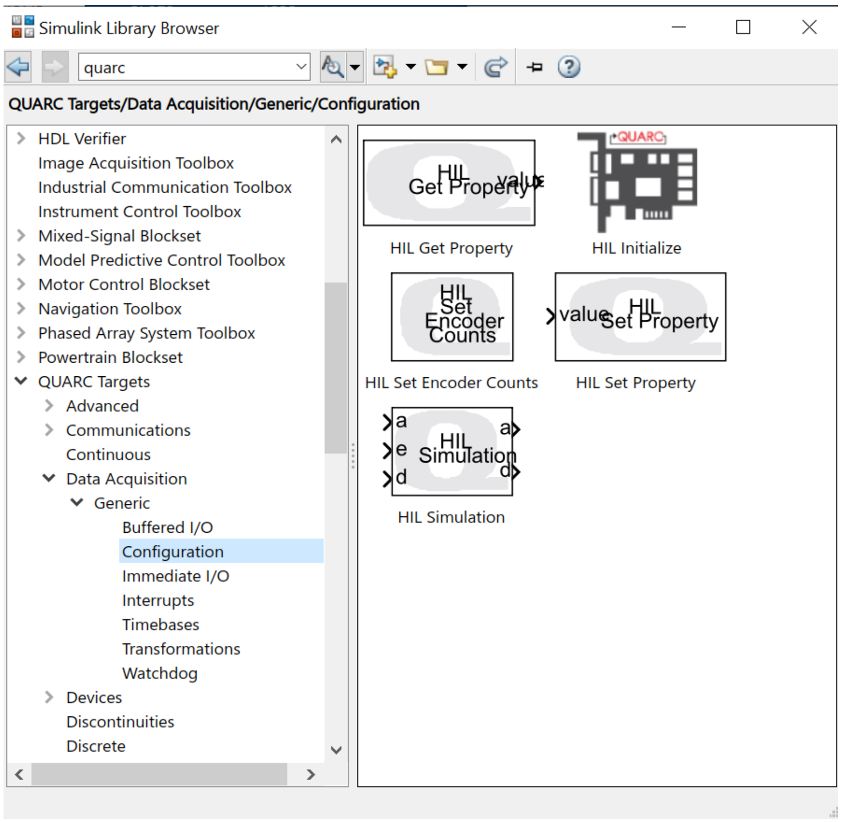


Figure 1.6: `QUARC Targets` in the Library Browser

7. Click-and-drag the `HIL Initialize` block from the library window into the blank model. This block is used to configure the data acquisition board.

8. Double-click on the `HIL Initialize` block.

9. In the `Board type` field, select `qube_servo2_usb`. Apply and close the window.

10. Go to `QUARC` item in the menu bar | `QUARC targets` item to set the correct `quarc_win64.tlc` for targets running on 64-bits Windows.

11. Save the file as `QUBE_iotest.slx` in the following folder `C:/temp/Lab_QUBE_Test`.

12. Important ! Go to the Matlab window. By clicking on the browse for folder icon, **change the current folder of Matlab so that it becomes your** `C:/temp/Lab_QUBE_Test` **folder** that contains the file you have just saved.

13. Go back to Simulink. Select the `HARDWARE` | `Monitor & Tune` item to build the code. Various lines in the Diagnosis Viewer Window (open the window) should be displayed as the model is being compiled. This creates a QUARC executable (.exe) file which we will commonly refer to as the **QUARC controller**.
Information about the compiling process appear at the bottom left corner of the Simulink window.
If the experimental procedure was followed correctly, no errors should be obtained in when running the QUARC controller. Time should flow in Simulink and the `LED` strip at the top of the QUBE-Servo 2 should turn from red to green.
Great, you have successfully set up the connection to the QUBE-Servo 2. Congratulations !

14. If you successfully ran the QUARC controller without any errors, then you can stop the code by clicking on the `Stop` button in the tool bar (or go to `QUARC` | `Stop`).

1.3.2 Reading the Encoder

Follow the steps below to read the encoder:

1. Using the Simulink model you configured in the previous section, add the `HIL Read Encoder` block by double clicking in the Simulink model and search for HIL Read Encoder. This block can also be found in the Library Browser from the `QUARC Targets` | `Data Acquisition` | `Generic` | `Immediate I/O` category.

2. Connect the `HIL Read Encoder` to a `Gain` and `Display` block similar to Figure 1.5 (without the `HIL Write Analog` block. It will added later). Double click in the Simulink model and search for the `Gain` and `Display` blocks. In the Library Browser, you can find the `Display` block from the `Simulink` | `Sinks` and the `Gain` block from `Simulink` | `Math Operations`.

3. Select the `HARDWARE` | `Monitor & Tune` item to build the code. The code needs to be re-generated again because you have modified the Simulink model. No errors should be obtained and the QUARC controller should be running.

4. Rotate the disc by hand, back and forth. The `Display` block should show the number of counts measured by the encoder. Remind that the encoder counts are proportional to the angle (angular position) of disc.

5. In the User Manual of the QUBE-Servo 2, find out how many count per revolution the DC motor shaft encoder has ?

6. Stop the controller, rotate the disc by 90 degrees clockwise, and re-start the controller. What do you notice about the encoder measurement when the controller is re-started? Is-it reset to zero or not ? Is the encoder relative or absolute ?

7. Measure how many counts the encoder outputs for a full rotation. To do so, stop the controller and move the disc to the 0 degree position marked on the QUBE-Servo 2. Start the controller and rotate the disc one full rotation. Is-it in-line with the value you found in the User Manual ?

8. Now we want to display the disc angle in degrees, not in counts. Modify the value of the `Gain` block to the value that converts counts to degrees. This is called the `sensor gain`. To confirm that the sensor gain is correct, start the controller with the disc at the 0 degree position marked on the QUBE-Servo 2. Rotate it one full rotation, and verify that the `Display` block reads 360.

9. Ultimately we want to display the disc angle in degrees, not in counts. Modify the `Gain` block to the value that converts counts to radians. To confirm that the new sensor gain is correct, start the controller with the disc at the 0 position marked on the QUBE-Servo 2. Rotate it one full rotation, and verify that the `Display` block reads $2\pi \approx 6.28$.

1.3.3 Driving the DC motor

1. Add the `HIL Write Analog` block. This block is used to output a signal from analog output channel #0 on the data acquisition device. This is connected to the on-board PWM amplifier which drives the DC motor.

2. Add a `Constant` block. Connect the `Constant` and `HIL Write Analog` blocks together, as shown in Figure 1.5.

3. Set the `Constant` block to 0.5. This applies 0.5 V to the DC motor of the QUBE-Servo 2.

4. Click on `Monitor & Tune` to build and run the QUARC controller.

5. Confirm that you are obtaining a *positive measurement when a positive signal is applied.* This convention is important, especially in control systems when the design assumes the measurement goes up positively when a positive input is applied. Finally, in what direction does the disc rotate (clockwise or counter-clockwise) when a positive input is applied?

6. Keep the program running and modify the `Constant` block to -0.5. Verify that the disc rotates in the counter-clockwise direction.

7. Stop the QUARC controller.

1.4 Measuring and estimating angular velocity

Topics covered

- Using an encoder to estimate angular velocity.
- Low-pass and high-pass filters.
- Using a tachometer to measure angular velocity.

1.4.1 Low-pass and high-pass filtering

A low-pass filter can be used to block out the high-frequency components of a signal. The Laplace transfer function of a first-order low-pass analog filter has the form

$$H_{LP}(s) = \frac{\omega_f}{s + \omega_f} \tag{1.1}$$

where s is the Laplace transform and ω_f is the cut-off frequency of the filter in (rad/s).
All higher frequency components of the signal will be attenuated by at least -3 dB ($\approx 70\%$ by amplitude).
A high-pass filter can be used to approximate the time-derivative of a signal. The Laplace transfer function of a first-order high-pass analog filter has the form

$$H_{HP}(s) = \frac{\hat{\Omega}_f(s)}{\Theta(s)} = \frac{\omega_f\, s}{s + \omega_f} \tag{1.2}$$

Note that the high-pass filter can be seen as the cascade of a pure derivate (which takes the form of s in the Laplace domain) and a low-pass filter as shown in Figure 1.7.

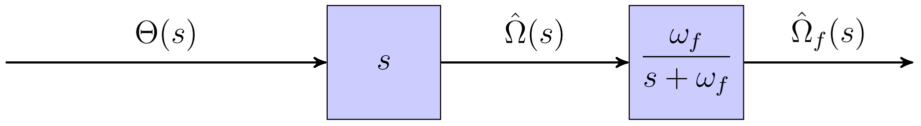


Figure 1.7: High-pass filter seen as a cascaded derivative and low-pass blocks

This high-pass filter can be very useful to provide a filtered estimate of the angular velocity $\dot{\hat{\theta}}_f(t) = \hat{\omega}_f(t)$ from a measured angular position $\theta(t)$.

1.4.2 Estimating the angular velocity from the encoder via high-pass filtering

Based on the Simulink model developed in the previous section, the goal is now to build a Simulink model that estimates the angular velocity (or speed) of the motor shaft using the angular position measure provided by the encoder for a constant (or step) motor input as shown in Figure 1.8.

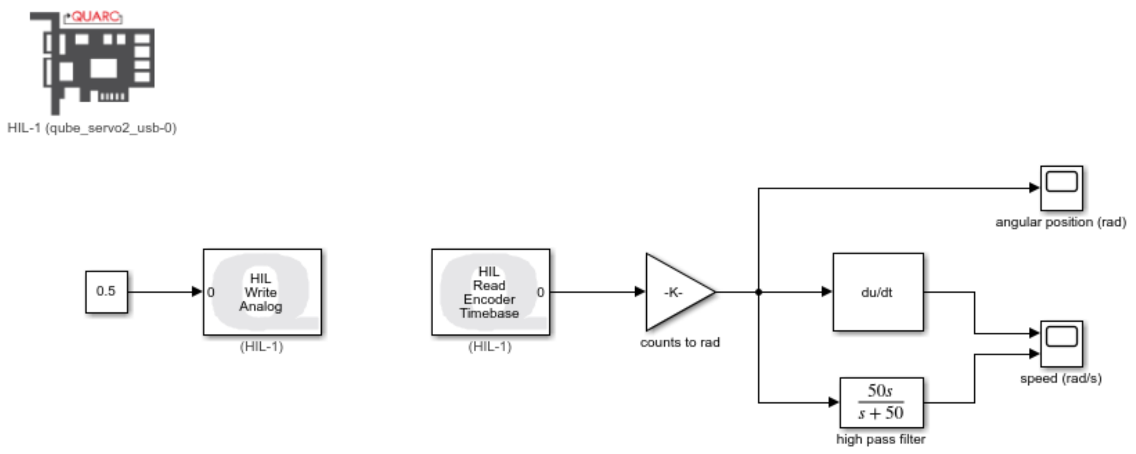


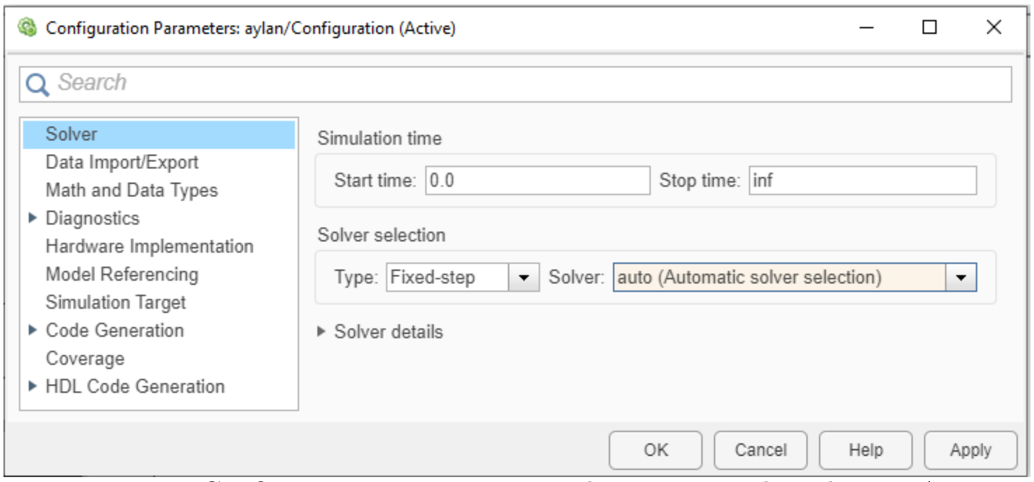
Figure 1.8: Estimating angular speed from the angular position measure provided by the encoder when the motor input is a constant value (similar to a step since the constant starts at t=0)

1. Open the model you developed in the previous section. Change the encoder calibration gain to measure the angular position in radians (instead of degrees). Change the `HIL Read Encoder` by an `HIL Read Encoder Timebase`. Double-click in the Simulink white and look for `HIL Read Encoder Timebase`.

2. Build the Simulink model as shown in Figure 1.8. Add a `Switch derivative for linearization` $\frac{du}{dt}$ block (look for a `Switched derivative for linearization` block) to the encoder calibration gain output to estimate the angular speed using the encoder (in rad/s). Connect the output of the $\frac{du}{dt}$ block to a `Scope`. For now, do not include the high-pass `Transfer Fcn` filter block $\frac{50s}{s+50}$, it will be added later.

3. Setup the source block a constant value.

4. Select the `HARDWARE | Configuration parameters` item to set the solver option to `Auto` as shown in Figure 1.9.



Simulation time

Start time: 0.0 Stop time: inf

Solver selection

Type: Fixed-step Solver: auto (Automatic solver selection)

Figure 1.9: Configuration parameters window to setup the solver to Auto

5. Select the `HARDWARE | Monitor & Tune` item to build and execute the code. Examine the angular position and estimated velocity responses.

6. Examine the encoder angular position measurement using a new Scope. Zoom up on the position response. Is the observed signal continuous or piecewise constant?
   From your answer, explain why the encoder-based angular speed measurement is noisy.

7. One way to remove some of the high-frequency components is adding a low-pass filter of the form $\dfrac{50}{s+50}$ in cascade with the derivative component $\dfrac{du}{dt}$ which corresponds to an s transfer function in the Laplace domain. Note that it is better to implement the two cascaded blocks in one single high-pass transfer function block $\dfrac{50s}{s+50}$.
   Add a `Transfer Fcn` block and connect it to the `Scope`. Set the `Transfer Fcn` block to $50s/(s+50)$, as illustrated in Figure 1.8.

8.

9. Select the `HARDWARE | Monitor & Tune` item to build and execute the code. Look at the filtered encoder-based angular velocity response and the motor voltage. Has it improved?

10. What is the cutoff frequency of the low-pass filter $50/(s+50)$? Give your answer in both rad/s and Hz.

11. Vary the cutoff frequency, ω_f, between 10 to 200 rad/s (or 1.6 to 32 Hz). What effect does it have on the filtered response? Consider the benefit and the trade-off of lowering and increasing this parameter.

12. Stop the QUARC controller.

1.4.3 Observing the estimated angular velocity for a square wave input

Based on the Simulink model developed in the previous section, the goal is now to modify the Simulink model to observe the estimated speed of the motor shaft when a square wave input is sent to the DC motor as shown in Figure 1.10.

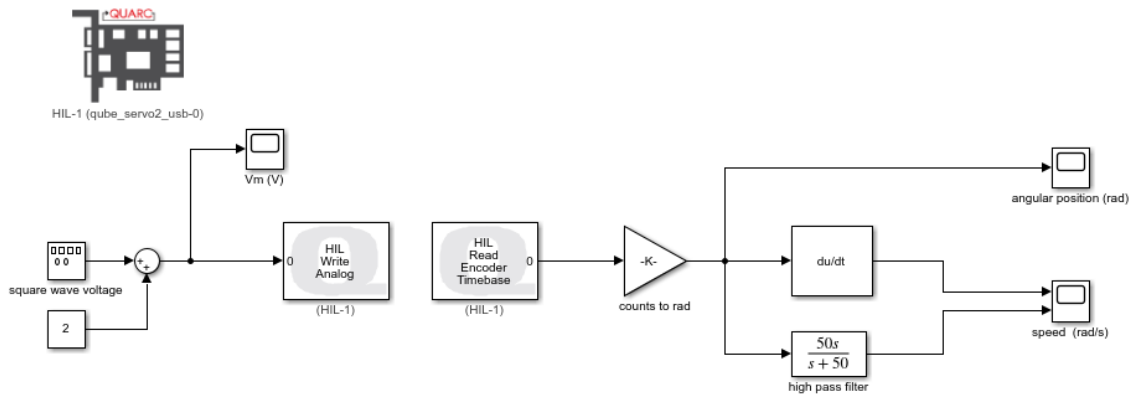


Figure 1.10: Estimating angular speed from the angular position measure provided by the encoder when the motor input is a square wave

1. Open the model you developed in the previous section.
2. Setup the source block as a `Signal generator + Constant` to output a *square* wave voltage that goes from 1 V to 3 V at 0.4 Hz.
3. Select the `HARDWARE` | `Monitor & Tune` item to build and execute the code. Examine the angular position and velocity responses.
4. Vary the cutoff frequency, ω_f, between 10 to 200 rad/s (or 1.6 to 32 Hz). What effect does it have on the filtered response? Consider the benefit and the trade-off of lowering and increasing this parameter.
5. Give your recommended value of the cutoff frequency to get the best estimate of the angular velocity from the encoder-based angular position measure by using this high-pass filtering method.
6. Stop the QUARC controller.

1.4.4 Reading the angular velocity from the tachometer

Based on the Simulink model developed in the previous section, the goal is now to modify the Simulink model to add the measure of the angular velocity of the motor shaft obtained by the tachometer sensor and compare it to the estimated angular velocity computed from the encoder. The input sent to the DC motor remains a square wave input. Build the Simulink model as shown in Figure 1.11.

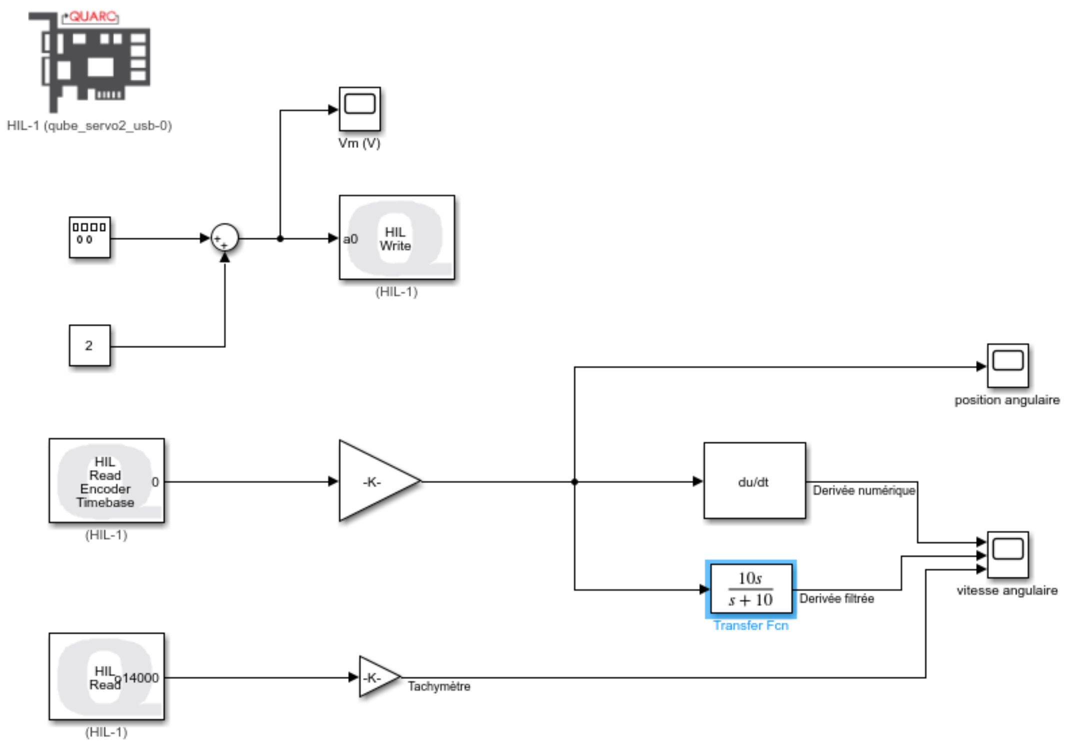
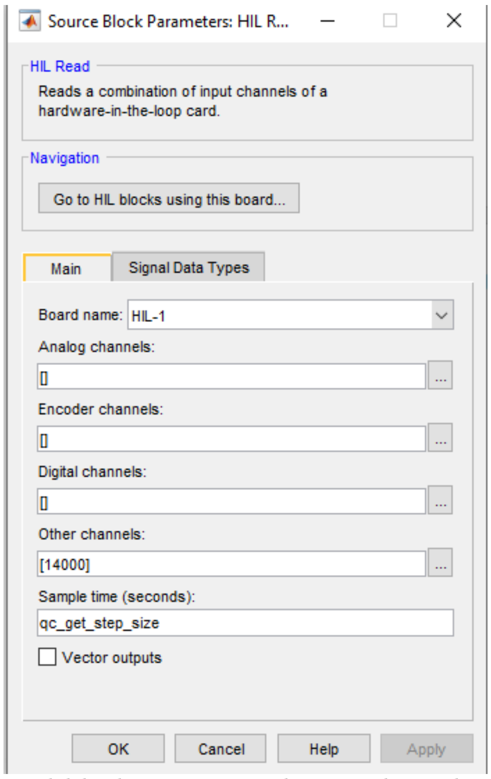


Figure 1.11: Measuring the angular velocity from the tachometer and comparing it to the filtered estimates computed from the encoder when the motor input is a square wave

1. Open the model you developed in the previous section.
2. Add a HIL read block. Double click on it and setup it to output the tachometer measure as shown in Figure 1.12.


Source Block Parameters: HIL R...

HIL Read

Reads a combination of input channels of a hardware-in-the-loop card.

Navigation

Go to HIL blocks using this board...

Main | Signal Data Types

Board name: HIL-1

Analog channels: []

Encoder channels: []

Digital channels: []

Other channels: [14000]

Sample time (seconds): qc_get_step_size

☐ Vector outputs

OK | Cancel | Help | Apply

Figure 1.12: Setup of the HIL read block to output the angular velocity from the tachometer sensor

3. Add a (calibration) gain with the same numerical value than for the encoder. Connect the gain output to the scope.
4. Select the `HARDWARE` | `Monitor & Tune` item to build and execute the code. Examine the high-pass filtered encoder and tachometer-based velocity responses. Do they look similar ?
5. Stop the QUARC controller.

1.5 Troubleshooting

How to fix the following QUARC license issue that may appear:

- If the following error message appears when using Simulink: "Error occurred while executing External Mode MEX-file 'quarc_comm': One of the arguments is invalid."

  You will need to re-activate the QUARC License on your computer. Follow the steps below:

  ▷ Log out.

  ▷ Log in with the following login: `.\admin_TPauto`

  ▷ Ask the teacher to enter the password.

  ▷ Go to the `C:\temp\` directory.

  ▷ Double-click on the licence file: `Quarc essentials Polytech Nancy`.

  ▷ Tick both options and save the file.

  ▷ Unplug and re-plug the QUBE-Servo 2 to reset the micro-controller.

  ▷ Log out the admin account.

  ▷ Log in with your student account.

  ▷ Start Matlab and Simulink again.