english

TMR4243 - Marine Control Systems II

Case study A: Thrust allocation and joystick control

The assignment introduces the use of National Instruments VeriStand, the cRIO and the toolbox around C/S (Cybership) Enterprise, which is a model scale tug boat that you will be using for your lab assignments. It is fitted with two Voith Schneider propellers (VSP) and a bow thruster (BT).

You will develop your own code, which interprets inputs from a joystick controller and assigns correct outputs to the thrusters. You will learn how to upload your code to the cRIO inside C/S Enterprise.

The assignment contains three parts:

1. A theoretical part,
2. A simulation part, and
3. A scale model test part.

You complete all three parts as a group. The theoretical and simulation part can be done whenever you want and you can borrow the necessary equipment. An instruction on how to borrow the simulation equipment can be found on blackboard. The model scale test will take place as scheduled in the laboratory plan.

1 Progress-report

After finishing the HIL-part you are to hand in computer written answers to the theory questions (no more than 3 pages) together with a short report (no more than 2 pages) which report on your graphical interface and as well as task 10. These are (pass/no-pass) and is not used to evaluate your final grade. They are meant to show that you are progressing after schedule. At any rate, a good effort at this stage is useful when you design your final report.

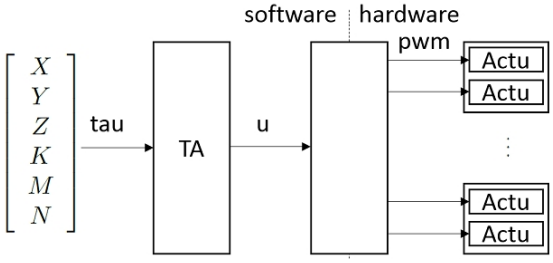

Figure 1: Generalized force control

Part I

Theory

You can prepare the theory in your group whenever you want, whereever you want. You have to include the relevant aspects in your final report, which will be marked. Tasks marked with an asterix (*) are not required to be included in your report.

2 Task: Reduced control model of CS Enterprise

Assume a situation in which the vessel is only operated at low speeds in calm waters.

*Task **2.1**: Establish a simple 3 degrees-of-freedom (DOF) control model for the horizontal motion of C/S Enterprise. You find help in Chapter 3 of the CS Enterprise handbook.*

3 Task: Thrust allocation

*Task **3.1**: Explain with the help of Figure 1 in a few sentences the principle of thrust allocation.*

Inputs u_i to the thrusters' PWM modules is a normalised force vector with $u_i = [-1, 1]$.

The angle input to the VSP thrusters is a vector $\alpha_i = [-\pi, \pi]$

For six degrees of freedom (6 DOF) control, the thrust load vector is

$$\tau = \begin{bmatrix} \mathcal{F} \\ \mathcal{M} \end{bmatrix}, \qquad \mathcal{F} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} N, \qquad \mathcal{M} = \begin{bmatrix} K \\ M \\ N \end{bmatrix} [Nm] \tag{1}$$

The thruster configuration for a 6DOF vessel is in general given by a set of lever arms from the vessel origin (VO) to each individual thruster. For a single thruster, let the thrust vector it produces in the body frame, and its location in the body frame, be

$$f = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \qquad l = \begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix}, \tag{2}$$

respectively. Then the corresponding thrust loads from this thruster become

$$\tau = \begin{bmatrix} f \\ l \times f \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_z \\ l_y f_z - l_z f_y \\ l_z f_x - l_x f_z \\ l_x f_y - l_y f_x \end{bmatrix}. \tag{3}$$

For a thruster producing thrust F in pure surge direction becomes $f = col(F, 0, 0)$, pure sway becomes $f = col(0, F, 0)$, and pure heave becomes $f = col(0, 0, F)$. For an azimuth thruster producing thrust in surge-sway at an angle α becomes $f = col(F\cos\alpha, F\sin\alpha, 0)$. For m thrusters, each with an orientation α_i, we get the total generalized thrust loads

$$\tau = \sum_{i=1}^{m} \tau_i = \sum_{i=1}^{m} \begin{bmatrix} f_i \\ l_i \times f_i \end{bmatrix} = B(\alpha)Ku \tag{4}$$

where $B(\alpha)$ is the thruster configuration matrix with $\alpha = col(\alpha_1, \ldots, \alpha_m)$, $K = diag(k_1, \ldots, k_m)$ is a matrix of scaling gains, and $u = col(u_1, \ldots, u_m)$ is the individual normalized thruster forces (i.e., $F_i = k_i u_i$).

Considering the thruster configuration of C/S Enterprise, where only the horizontal motions are considered:

- *Task **3.2**: Which DOFs can be directly manipulated, and which DOFs do we neglect? Reduce the thrust load vector accordingly.*
- *Task **3.3**: Given the VO of C/S Enterprise, make a drawing of the ship, its body-fixed reference frame with positive directions indicated, and the locations of the thrusters. (Hint: See CSE1 Handbook)*
- *Task **3.4**: Make an accompanying table of thruster parameters in your report, that include the (l_x, l_y) coordinates of each of the three thrusters of C/S Enterprise (note that the sign should be included, i.e. l_x and l_y can take both positive and negative values).*

3.1 Allocation with thrust vectors in polar coordinates

Let each of the three thrusters produce a thrust F_i in a direction α_i, for $i = 1, 2, 3$, that is, in polar coordinates. This gives

$$f_i = \begin{bmatrix} f_{i,x} \\ f_{i,y} \end{bmatrix} = \begin{bmatrix} F_i \cos\alpha_i \\ F_i \sin\alpha_i \end{bmatrix}$$

Task: The angle α_i can be fixed or varying. What are the possible angles that the three thrusters can take?

For the VSPs and the BT, the following maximum thrusts are given, corresponding to $u_i = 1$:

- $F_{max,VSP} = 1.03[N]$
- $F_{max,BT} = 2.629[N]$

(Values as of 2019)

Note that the input to the PWM module for bau thruster is in the interval $u_i = [-1, 1]$ whilst the VSP thrusters is in the intervall $u_i = [0, 1]$. (The VSP thrusters cannot produce negative force in a given thrust configuration. For negative force we need to flip the thrust-angle by 180 degrees. The C/S Enterprise is equipped with 2(!) VSPs and that $F_{mnax,VSP}$ above is the maximum force of one of these.

*Task **3.5**: Calculate the gains k_1, k_2, k_3 assuming the linear map $F_i = k_i u_i$ for $i = 1, 2, 3$.*

*Task **3.6**: Using (3) for the relevant DOFs, show the details of the thrust map*

$$\tau = B(\alpha)Ku$$

where $\alpha = col(\alpha_1, \alpha_2, \alpha_3)$, $u = col(u_1, u_2, u_3)$, $B(\alpha)$ is the thruster configuration matrix, and K is the gain matrix. Express $B(\alpha)$ and K in your report.

*Task **3.7**: Suppose you fix the azimuth angles α_i in constant directions, so that $B(\alpha)$ becomes a constant matrix. This is called "Fixed azimuth mode". For a given, commanded $\tau = \tau_{cmd}$, propose a mapping that calculates the corresponding individual thuster commands $u_{i,cmd}$, that is, if $\tau_{cmd} = g(u_{cmd})$ then calculate $u_{cmd} = g^{-1}(\tau_{cmd})$. Then you have done your first thrust allocation.*

*Task: **3.8** Is there some set of fixed angles α that makes $B(\alpha)$ a singular matrix? If so, then state such singular configurations in your report.*

3.2 Allocation with thrust vectors in rectangular coordinates

Suppose you now keep

$$f_i = \begin{bmatrix} f_{i,x} \\ f_{i,y} \end{bmatrix}$$

as the thrust input for each thruster, that is, in rectangular coordinates.

Task: If you are given $(f_{i,x}, f_{i,y})$ for a thruster, show how you can calculate the corresponding values for (α_i, u_i) for the thruster.

Since the bow thruster is only producing thrust in the sway direction, we will have $f_3 = f_{3,y}$. Let then the total thrust vector be

$$f = \begin{bmatrix} f_{1,x} \\ f_{1,y} \\ f_{2,x} \\ f_{2,y} \\ f_3 \end{bmatrix}.$$

*Task **3.9** Show that the thrust mapping can now be written*

$$\tau = Bf$$

(where B is not the same matrix as used in Ch 3.1)

*Task **3.10**: Explain the thruster configuration matrix B. What do the rows represent? What do the columns represent?. Show that this is an underdetermined linear set of equations*

Suppose from your joystick controller, you output the 3DOF command

$$\tau_{cmd} = \begin{bmatrix} X_{cmd} \\ Y_{cmd} \\ N_{cmd} \end{bmatrix}.$$

Hence, we now want to allocate these 3DOFs to the thrusters. An *underdetermined system of equations* has infinitely many solutions, and we will therefore solve the thrust allocation for f_{cmd} by finding its minimum norm solution. This is given by the minimization problem

$$\begin{array}{ll} min & f_{cmd}^\top f_{cmd} = |f_{cmd}|^2 \\ s.t. & Bf_{cmd} = \tau_{cmd} \end{array}$$

*Task **3.11**: Suggest a method for finding the optimal solution for f_{cmd}?*

*Task **3.12**: Does this solution take into account constraints, like thruster saturation, forbidden sectors, etc.?*

*Task **3.13**: Let two thrust commands be*

$$\tau_{cmd,1} = \begin{bmatrix} 1 \\ 1 \\ 0.5 \end{bmatrix} \quad and \quad \tau_{cmd,2} = \begin{bmatrix} 2 \\ 0 \\ 0.5 \end{bmatrix}.$$

Calculate the corresponding f_{cmd} for each of these, and the corresponding sets (α_i, u_i) for the three thrusters.

*Task **3.14**: Given the azimuth angles α_i that you calculated above, set the thrusters in fixed mode with these angles, and then use the polar coordinate allocation in the previous section to calculate the u_i. Is there a difference? Are any of the thruster constraints violated?.*

Part II

Simulink and Processor-in-the-loop-simulation

4 Task*: Download the CS Enterprise Simulink package from GitHub

On GitHub, you find a repository containing all required Simulink-Files for the lab.

Task: Start your computer, open GitHub, go to

- `https://github.com/NTNU-MCS/CS_EnterpriseI_archive/blob/master/TMR4243Lab19/TMR4243Lab_material_2019.zip`

and download the TMR423 lab folder.

5 Simulink

Simulink is great for development, rapid prototyping and testing. We advise for initial development to be performed in the Simulink-model before HIL-testing. (i.e, for those tasks where it is possible, test your system using Simulink model before HIL-testing).

The Simulink model *"CSE_HIL-full.slx"* is set up in a similarly as the Veristand HIL Project (where you are developing by updating and compiling the file *"custom_ctrl.slx"*). Therefore, to keep transitioning to HIL-testing simple, it is advised that while working in *"CSE_HIL-full.slx"*, you mainly edit *"CSE_HIL-full/Controller/CtrlCustom_SLX"*. (It may be smart to "scope" signals elsewhere in the model for verification).

Note:

Since you in the Simulink model can see how $(u \rightarrow force)$ *are mapped , you could in theory develop a very accurate mapping for* $(force \rightarrow u)$*. However, in the model-scale testing, we expect modeling errors to result in a deviation between desired and achieved generalized force. Therefore, you are supposed to keep the linear mapping* $(force \rightarrow u)$ *developed in Section 3 rather than developing a more complex mapping. It is expected to be minor deviations between desired and achieved force, even with a correctly implemented thrust-allocation system. These deviation can be seen as modeling errors, and is something that the control systems you later design should be capable of handling.*

Tip: *You are working in groups, so some resources should be used to get familiar with the HIL-setup early on. (i.e at times where your group have a HIL-box, then most of the time, one person should always work with the HIL-computer, at-least until you are familiar with it)*

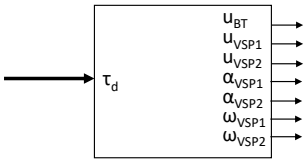

Figure 2: Thrust allocation in Simulink model

6 HIL-testing

In HIL testing you will perform all your work in the *"custom_ctrl.slx"* file, which is the block that is to be compiled and run on the HIL computer. If you in the Simulink part worked in subsystem *"CSE_HIL-full.slx/Controller/CtrlCustom_SLX"*, it should be relatively simple to transfer your previous work over to *"custom_ctrl.slx". Remember that the HIL-computer have Simulink 2016. So if you are working on a newer version locally, then you need to export this to 2016 to open it on the HIL-computer*

We refer to the Appendix for getting started with HIL-testing.

Task: Use the getting started part of the Appendix to become familiar with the HIL setup (APPENDIX 1.1 - .1.4) This include:

1. Compile a non-modified version of the project file, and ensure that the graphical interface react to the Sixais controller.
2. See that you can edit the graphical interface
3. Verify that you can log data.
4. Add new input and output- ports to the Simulink model and verify that they also appear in Veristand.

Unless otherwise specified, all tasks are hereafter to be tested using the HIL-boxes.

7 Task: Implement the polar coordinate thrust allocation

*Task **7.1**: Implement the polar coordinate thrust allocation you developed in subsection 2.1 in, where you specify fixed azimuth angles and then calculate the u_i commands. Include this in the ctrl_custom.slx file. See figure 1.*

Hint: The angular input to the thrusters VSP1 and VSP2 is in the interval $\alpha_i = [-\pi, \pi]$, whilst you cannot give negative input to the VSP thrusters.

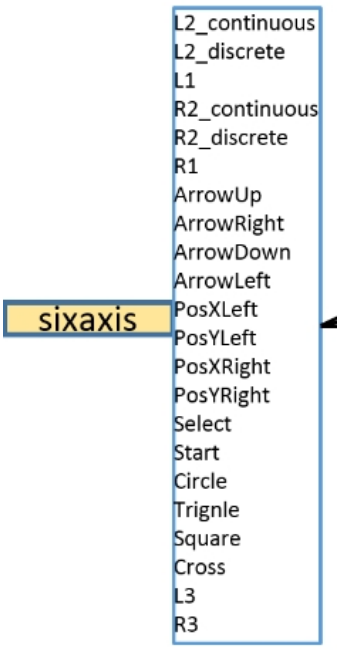

Figure 3: Available joystick inputs in Simulink

8 Task: Implement the rectangular coordinate thrust allocation

*Task **8.1**: Implement the thrust allocation you developed in subsection 2.2 and include it in the ctrl_custom.slx file. This should for each thrust command provide a desired u_i and α_i to the thrusters (α_i only to VSPs). The structure is the same as shown in Figure 1, but the allocation is slightly more complicated.*

Hint: Use the *atan2* function to find the force angles α_{VSP}.

9 Task: Implement a joystick to force conversion block

C/S Enterprise is controlled by the user with a controller borrowed from a popular home entertainment system. The controller has two joysticks (PosXRight, PosYRight, PosXLeft, PosYLeft), several buttons and two continous triggers (R2_continuous) at the front. A full list of available inputs from the joystick is given in Figure 3. The buttons Circle, Triangle, Square and Cross are not available, because they will be used later.

- *Task: By experimenting and using the graphical interface, find out what the input ranges for the two joysticks, the continous triggers and the other buttons (except Circle, Triangle, Square and Cross) are. Prepare a list with the input ranges, this can be very useful in later lab experiments*
- *Task: Before continuing, make a block, that checks that you do not violate the limits for*

the inputs into the VSP- and BT-system. The limits are $u_{VSP} = [0, 1]$ *and* $u_{BT} = [-1, 1]$. If you violate the limit, set $u = 0$.

- *Task: Make a conversion block, that maps the input of the joystick to your simple thrust allocation. With the forward-and-backwards direction of the joystick, you control the surge force and with the sideways direction of the joystick (left/right), you control the yaw-moment of the vessel.*
- *Task: Make a conversion block, that maps the input of the joystick to your full thrust allocation. With the joystick, you control the surge- and sway force and with the continuous trigger (R2), you control the yaw-moment acting on the vessel.*
- *Task: Make a function, that allows you to change between the two conversion blocks by pressing a button of your choice (that is not the Circle, Triangle, Square or Cross button ;)).*
- *Task* ***9.1****: Make a graphical user interface for the ctrl_custom environment . (See Appendix and Read section 5.1 of the Cybership Enterprise I manual). Include the graphical interface (and explain the elements you have added) in the report*

10 Task: Processor-in-the-loop test

Task ***10.1****: Make suitable tests with the simulated CS Enterprise in the cRIO (this is called a processor-in-the-loop test). Prepare plots that show the inputs of the joystick and the corresponding movements of the ship.*

Part III

Scale model test

In March/April 2019, you will have access to the model scale ship. It is equipped with the same CRIO as you have been using in the processor-in-the-loop test before. However instead of a simulated vessel inside the controller, this time the controller sends the command signals to the actuators of the model ship.

11 Scale model test

Task 11.1*: Perform the exact same test as you did in Section 7-9 again on the model scale CS Enterprise. Do you notice any differences?*

Task 11.2*: (Repeat task 10) Make suitable tests with the simulated CS Enterprise in the cRIO. Present relevant data in your report.*

Part IV

Appendix

This Appendix is a Draft under development

A *Simulink, Veristand, HIL and Model-Scale Testing*

These are some initial instructions to get started. More detailed instructions is found in MC-Lab Handbook (Particularly in Ch-2)

A.1 HIL-Testing

A.1.1 HIL Equipment

For HIL testing, three boxes with HIL equipment are available (See Figure 4). These contains the following:

1. A real-time computer (Compact Rio) which is the simulation target in HIL testing. This is the same computer that will be installed on the ship in the model-scale testing. (i.e., we are using the same interface in HIL testing as in the model-scale testing).
2. A laboratory-computer installed with Virtual-Machine, where correct software, compatible with the rest of the equipment is installed.
3. A RaspberryPi used for communication with the Sixais Controller

4. A Sixais controller for remote control of system. **Important note**: Do not allow this to empty of battery as we then may need to resynchronize with the Raspberry Pi, which is kind of a hassle
5. Various cables and chargers.

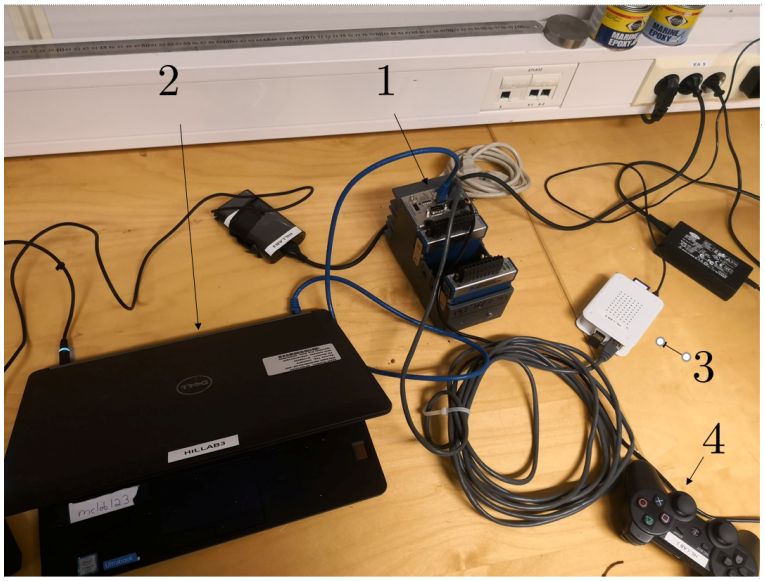

Figure 4

A.1.2 First Deployment

1. Set up and connect the equipment as shown in Figure 4. It is adviced that you first turn on Compact rio, then Raspberry Pi, then wait with turning the controller on until the Raspberry Pi has loaded.
2. At this stage, number 1 of the controller is indicated with a red light, and the bluetooth dongle is blinking blue. You should be able to ping your CRIO IP from the laboratory-computer.
3. On laboratory-computer, open Virtual Machine.

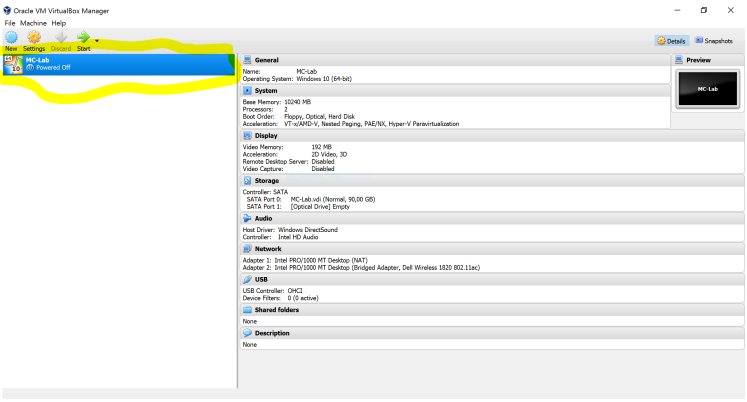

Figure 5

4. Insert the memory stick containing your working folder (i.e., the one originally downloaded from the Git-hub folder), or send yourself the folder as a zipped file over email if it is difficult to access the USB stick in Virtual Machine. You should also be able to "drag-and-drop" between the computer and the virtual machine.

5. Open Veristand (this is the software we are using to connect MATLAB and the CRIo), and open the project CSE1-HIL from the Git-hub folder. Press "configure".

6. Ensure that the Target-IP found in the project file Veristand/Controller is correct. (See Figure 6)

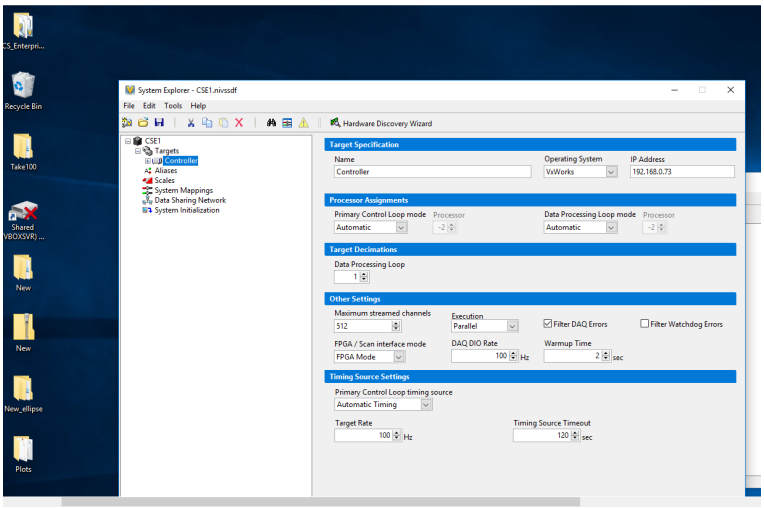

Figure 6

7. If you have connected correctly, with the Sixais controller on, the project should be deployable. Test it by pressing deploy (A).

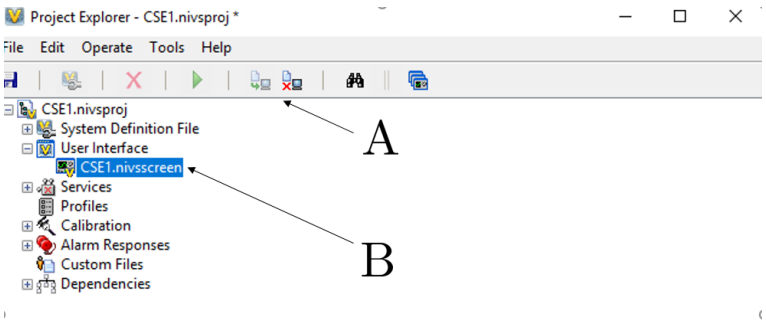

Figure 7

8. With successful deployment, the graphical interface should be functioning. Press CSE1.nivsscreen (B) to open it. Ensure that the graphical interface respond to the buttons of the Sixais controller. (See CH-5 in CSE1-Handbook)

At this point, you have deployed a Veristand-model. If you used the orignal folder from GitHub, the custom control-component "(custom_ctrl)" block is empty. You, as students will work using this block to implement the necessary control components, according to task specifications.

A.1.3 Compiling a Simulink Model, then uploading it to the Veristand

1. For compilations, you are to work with *"custom_ctrl.slx"*. This already is set up with correct build settings, such that you can compile it simply by pressing *"build model"* (See Figure 8-A. As long as you work in the same folder, the Veristand will find your compiled code (ctrl_custom.out) automatically.

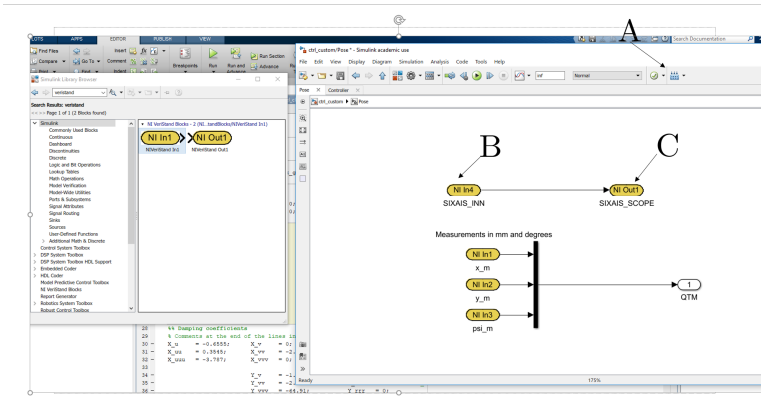

Figure 8

2. Before deploying you newly compiled block, you need to update the Veristand project as follows (See Figure 9). A) Locate ctrl_custom, B) Press Update, C) Ensure that the modification date and time has been updated to the time that you compiled your model.
3. You should now be able to deploy your new code.

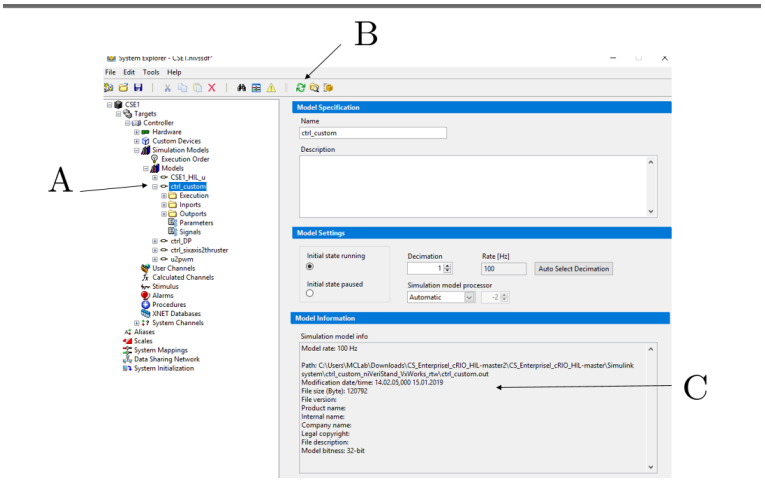

Figure 9

A.1.4 Map a new input/output ports to your Simulink block

You may want to add additional inputs or scopes to your *custom_ctrl* block. This is done as follows.

1. Add new inputs (B) and outputs ports (C) before building in Simulink (See Figure 8). Outputs can be accessed by editing the graphical interface in Veristand. New inputs are typically sent from either from graphical interface or the Sixais controller.

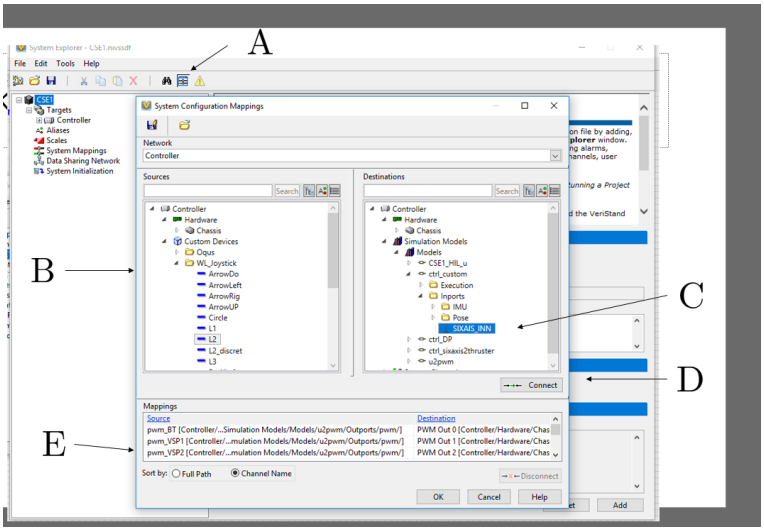

Figure 10

2. After compiling a Simulink model new input ports, do as follow to map a signal to the new input. (See Figure 10)

   A Open mapping window

   B Locate the desired input signal. (In this case L2 from the Sixais controller)

   C Locate the name of the new Simulink Input-port (in this case SIXAIS_INN)

   D Click Connect

   E Your new mapping should be in the bottom of the mapping list.

A.1.5 Editing Graphical Interface

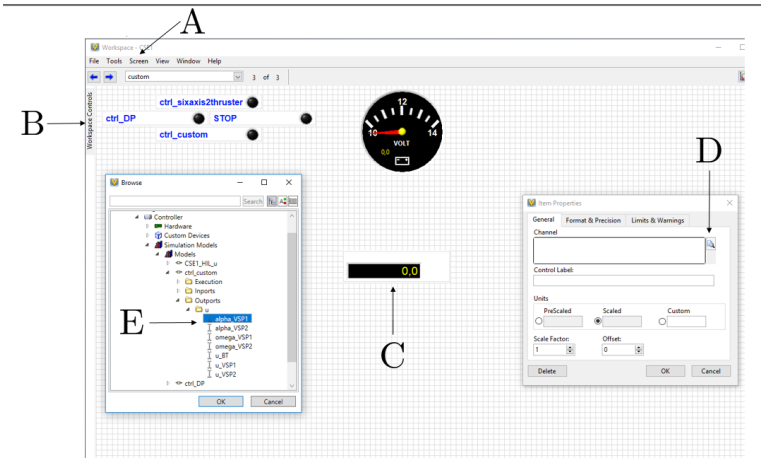

Figure 11

0 Go into the graphical interface (CSE1.nivsscreen)

A Go into Edit Mode

B Go to workspace control and add a workspace control (ex: Numerical Indicator/Medium). You can also add graphs etc..

C Right-click on the newly created workspace control

D Click on add channel

E Add the channel you want to display (or control)

A.2 Logging Data

1. Locate *"Logging/Logging Control"*, found in Workspace Controls (See editing Workspace).
2. Add a Data Logging Control in your Workspace window, set the desired path for the file and add the channels/parameters of interest. See Figure 12.
3. In File-Settings, set "Log Type File" as .csv, which are easy to work with in MATLAB.

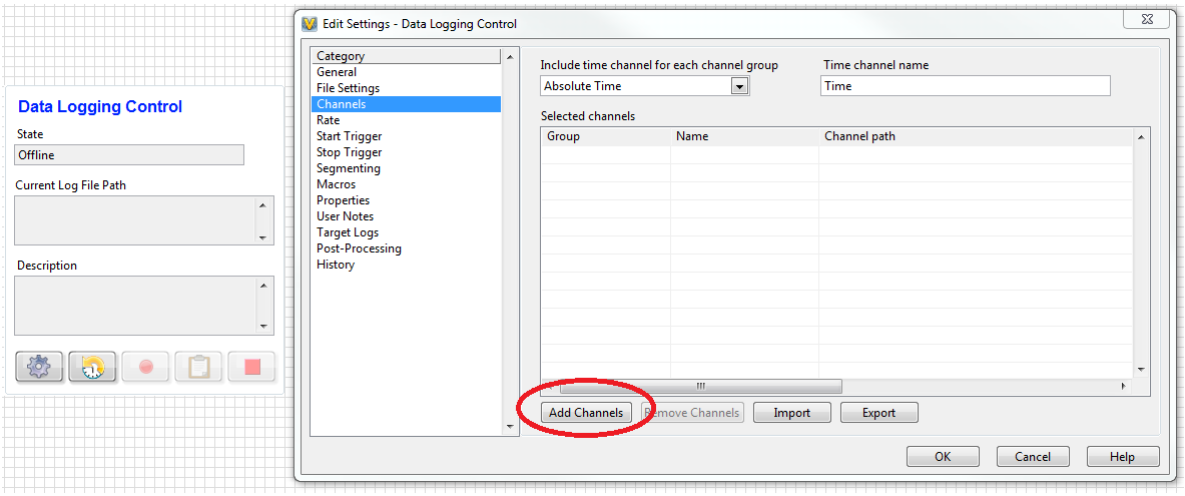

Figure 12: Data logging in Workspace

A.3 Handing over HIL boxes to new students

- The HIL-boxes are to be transferred between students according to schedule set by student-assistants/vitass.
- When you hand the HIL-boxes over to a new group everything should be as when you received it. If the system has been degraded, with any troubles or hardware/software failures , please notify Student-assistants.
- Remove your files from the HIL-computer and onto a memory stick.

A.4 From HIL To Model-Scale Testing

In HIL work, we have used the Veristand project-file CSEHIL. The Veristand project-file CSE-MOD are setup for use with the ship, and is identical as the former,with the exception that the simulation model is replaced with the real ship. In practice this means that the output PWM signals are forwarded to the motor-controller (instead of to the simulation model), whilst the position (and possibly IMU-data) are output from the Qualisys-track manager (instead of from the simulation model).

Although an effort has been made at making the HIL model similar to the model-scale vessel (June 2017, See MCLAB handbook), there is far from an exact correspondence between them. In particular expect errors in the mapping between u and force. Importantly, the interface (output-input signal types) between the two systems correspond, with similar trend being achieved (tested, 2019). The discrepancy is something that you as control engineers should be trained at handling, with readjusting your system once you enter on-line testing.

Due to their similarity, it should be simple to transfer your HIL setup to the Laboratory testing. However, do not expect everything that functioned in HIL- testing to function in the laboratory. To account for discrepancies, it may be smart to consider robustness to modeling errors in your simulations. Also expect that variables need to be re-tuned after transferring from HIL to

laboratory. Therefore, it may be smart to have control gains as input in the graphical interface when you start your testing, such that it is simple to re-tune on-line.

B Trouble-Shooting

This guide is in progress and will be expanded as new problems occur. Please record trouble-shooting solutions and hand them over to student-assistants such that this can be updated.

B.1 Trouble-Shooting: HIL and Veristand

Not able to deploy Veristand project:

- Possibly the most efficient method of troubleshooting is as follows: If something unexplainable error **restart all components.**
- In general, some unexpected unexplainable error happens when deploying, the first rule is to restart all components (including Virtual Machine).
- Check that you have contact with the CRIO. Open "cmd" and write the command "ping 192.168.0.XX", where XX is 70-80, depending on the IP written on your HIL box. If you are connected, you should get replies.
- Check that your ethernet cables is connected correctly (in correct port (see image in getting started)) Check that all cables are properly connected.
- In the Sixais controller the 1 led light should be red.
- Check that the Sixais controller is connected to the bluethooth dongle.
- If you are promoted with password for CRIO try "Marin33". If this does not work, try rewrite admin in usr, and Marin33 a couple of times.
- Check that the Target-IP found in the project file Veristand/Controller is correct.
- Update all Veristand model files. (See Figure 9)
- If you are not able to connect the controller to the bluethooth dongle, it may be that they are no longer synchronized. Torgeir Wahl will need to resynchronize them. Ask student assistant for help.
- Rumour has it that in the previous year, sometimes the automatically compiled Simulink folders (slprj and ctrl_sixaxis2thruster_niVeriStand_VxWorks_rtw) had to be deleted before recompiling new Simulink versions.
- After relevant troubleshooting steps: Ask student assistants.'

Please go through relevant points before asking student-assistants for help.

If you are not able to resolve your problem, please start with the original project (ensure correct IP) and see if it can compile. If it can, then there are probably some errors in your code (or a bug induced at a later point). Try a little more before asking student assistants.

If you are stuck you should of-course ask student-assistants for help.

Cannot compile Simulink

- Cannot compile + Unexplainable weird build error → Restart virtual machine and or Simulink
- Google is your friend.
- After using some time trying for yourself: Ask student assistant