Handbook of
Marine HIL simulation laboratory
and
Marine cybernetics laboratory

NTNU – Trondheim
Norwegian University of
Science and Technology

Faculty of Engineering Science and Technology
Department of Marine Technology

Introduction

This handbook is a comprehensive reference for the marine hardware-in-the-loop (HIL) and marine cybernetics laboratories. The laboratories are used in teaching and research on development and real-time testing of marine control systems.

Structure

Part I explains the concepts and motivations for real-time and HIL testing.

Part II describes the laboratory facilities and equipment. A general overview of hardware and software.

Part III is a user guide intended for students of the course. Step-by-step instructions for development and deployment of programs to the real-time controller are given. Lower level details, intended for laboratory assistants and customized use, are given in Part V.

Part IV holds the exercise texts for TMR4243 Marine Control Systems II.

Contents

Nomenclature

BT	bow thruster
cRIO	compact reconfigurable input/output real-time embedded industrial controller by National Instruments
CSE1	Cybership Enterprise 1
CSS	Cybership Saucer
DP	dynamic positioning
ESC	electronic speed controller
FPGA	field-programmable gate array
HIL	hardware-in-the-loop
MC	marine cybernetics
PWM	pulse-width modulation
RPi	Raspberry Pi single-board computer
VSP	Voith Schneider propeller

Part I

Theory

Chapter 1

Realtime

Real-time testing involves using a real-time OS as part of a test system. The most common requirements driving the need for a real-time test system are to achieve greater reliability and performance than is possible using a general-purpose OS.

http://www.ni.com/white-paper/3938/en/

http://www.ni.com/white-paper/14238/en/

http://www.ni.com/white-paper/13068/en/

- Forklaring av hva realtime betyr i HW og SW

Chapter 2

HIL

In general, Hardware-In-the-Loop simulation is a method that can be used to test complex real-time control and monitoring systems. Such systems are becoming more complex and rely on more advanced integrated functionality of software-based real-time functions, and many separately designed control and monitoring systems need to cooperate on performing common tasks. Consequently, the control system software code becomes more complex, and may be hard to verify by running regular software simulations. With testing by HIL simulation, the control system is run on its intended hardware, but instead of controlling the real process, it is controlling a simulated process in a simulated environment. The control and monitoring system will, however, see no difference between the real process and the simulated process.

Through HIL testing, the Marine HIL-Lab aims for by students and researchers to qualify their experimental setups in other laboratories before their assigned laboratory time is started. This will aid in making experimental work more efficient by reducing debugging time, improve tuning of parameters and test scenarios, and thereby maximizing the outcome of the experimental work.

Smogeli, Fremtidens verifikasjon av kontrollsystemer for Skip og offshorefartøy

DNV, Hardware in the Loop Testing (HIL)

Johansen, Sørensen, Experiences with HIL Simulator Testing of Power Management Systems

Smogeli, Introduction to third- party HIL testing

Johansen, Fossen, Vik, Hardware-in-the-loop Testing of DP systems

Pivano, Experiences from seven years of DP software testing

DNV, Rules for Classification of Ships (Part 6, Ch 22)

Ambrosovskaya, Approach for Advanced Testing of DP Control System

Selvam, System Verification Helps Validate Complex Integrated Systems

A. Veksler prøveforelesning:

- Increased complexity marine vessels increases the need for testing and verification.
- A reasonably new approach to this is Hardware-In-the-Loop testing, or HIL.
- Widely used in the automotive industry
- Can be seen as something in between simulation testing and full scale testing
  - More realistic than a simulation, less realistic than a full-scale testing
  - Mathematical models of the systems that are not included as hardware.
- A real-time simulator, constructed by hardware and software, that is
  - configured for the control system under consideration
  - embedded in external hardware
  - and interfaced to the target system or component through appropriate I/O
- Advantages:
  - Another layer of independent verification
  - Allows testing emergency procedures that would be too dangerous on a real vessel
- Disadvantages:
  - Initial investment to set up a HIL simulator for a particular system. The resources could be spent on simulation testing or full scale testing.
  - Supplements, but does not replace, proper software design techniques
    - As with all software testing, it typically executes only a fraction of the control system code.

Asgeir

- HIL testing is accomplished by connecting a simulation PC in the system's communication network.
- Inputs to the equipment under test are simulated.
- The controllers respond as they would in a dynamic environment.
- Simulator responds to output from the controllers as the dynamic system would
- Software (core SW and/or configuration) errors are exposed.

Chapter 3

Real-time computing

3.1 SIL

3.2 Model-in-the-loop

3.2.1 Hybrid simulation

Another advantage of HIL testing is that scenarios that may be difficult to test experimentally (either due to the risk involved in the test, due to the need for a special state of the environment, or due to inadequate experimental facilities), can be tested thoroughly, without risk, if sufficient high-fidelity simulation software exists. This also makes it possible to use the Marine HIL-Lab for hybrid experimental test setups, where part of the experimental setup is real and part is simulated, and these parts are interconnected through real-time interfaces such as sensors, communications, and actuators.

Part II

Laboratory descriptions

Chapter 4

Marine HIL simulation laboratory

The laboratory is suitable for implementation, qualification and comprehensive testing of

- marine control algorithms
- communication interfaces,
- human-machine interfaces,
- experimental test scenarios, and
- experimental setups before proceeding to other laboratories (such as model scale testing).

4.1 Equipment

The laboratory consists of three equivalent portable setups, as illustrated in Figure 4.1, each including

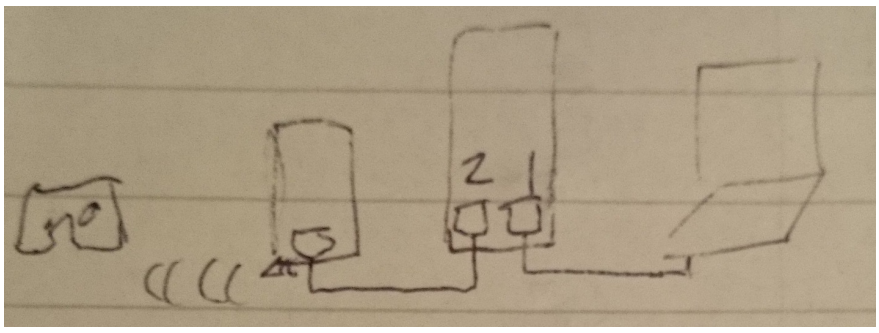


Figure 4.1: HIL setup

2	analog joysticks
1	pressure-sensitive directional pad
2	analog triggers (L2, R2)
6	pressure-sensitive buttons (△, ○, ×, □, L1, R1)
5	digital buttons (PS, L3, R3, Start, Select)
	motion sensing (6 degrees of freedom)

Table 4.1: Sixaxis input

- Sony Sixaxis wireless gamepad for PlayStation 3,
- Raspberry Pi (RPi) with Bluetooth dongle,
- National Instruments (NI) cRIO-9024 control and acquisition device, and
- Dell Latitude E6440 laptop.

4.1.1 Hardware

4.1.1.1 Sixaxis wireless gamepad

The Sixaxis is a widespread gamepad. It transmits a broad range of input, listed in Table 4.1, and is suitable for human operator input. The device communicates over Bluetooth.

4.1.1.2 Raspberry Pi

The RPi is a credit card-sized single-board computer including Ethernet and two USB connections. In the HIL laboratory setup, the unit is merely used to transmit the Sixaxis' data to the cRIO[1].

4.1.1.3 cRIO-9024

The cRIO is a compact reconfigurable input/output embedded real-time control and acquisition device, including two Ethernet ports. The unit also holds a field-programmable gate array (FPGA) chassis. In the HIL laboratory setup, an analog input module (NI 9215) and a digital output module (NI 9474) are mounted on this chassis.

4.1.1.4 Laptop

A generic personal computer is used to configure and upload applications to the cRIO. The same computer is used for graphical user interfacing during simulations.

[1]The cRIO USB port only supports common USB mass-storage devices, thus a Bluetooth dongle cannot be connected directly.

4.1.2 Software

4.1.2.1 Laptop

The main software used for the HIL laboratory are:

MathWorks Simulink graphical programming environment used for implementation and compilation of dynamic systems simulation models and control algorithms.

National Instruments VeriStand real-time testing environment used to configure testing applications to run on cRIO and provides a user interface for run-time monitoring and interaction with the applications.

Diverse utilities such as

National Instruments Measurement & Automation Explorer (NI MAX) hardware configuration tool

ping network utility to verify network connection

ftp network utility, or similarly WinSCP, to transfer data log files from cRIO to the laptop

Appendix A.3 covers installation and setup of the laptop software.

4.1.2.2 Sixaxis, Raspberry Pi and cRIO-9024

The Sixaxis, RPi and cRIO are configured for generic use according to Part III and the ordinary user may thus disregard the involved software.

The Sixaxis runs out-of-the-box firmware. Appendix A.1 covers the cRIO software and setup details. Appendix A.2 covers the RPi software and setup details.

4.1.3 Communication

Following Figure 4.1 from left to right:

Sixaxis transmits its information to the Bluetooth USB dongle with which it is previously paired[2].

RPi receives Sixaxis data through the USB dongle and forwards it through its TCP/IP[3] server over Ethernet to the cRIO.

cRIO reads Sixaxis data on Ethernet port 2 through its TCP/IP client. Simulation data and laptop input is transmitted and received on Ethernet port 1 by the VeriStand Engine.

Laptop reads simulation data and sends input to the cRIO over Ethernet.

[2]One-time pairing procedure described in Appendix A.2.2.2.
[3]All IP addresses are as given in Table D.1.

4.2 Simulation models

4.2.1 Cybership Enterprise 1

eta

4.2.1.1 Forces and moment input

CSE1_HIL_tau.out

Input:

4.2.1.2 Control input

CSE1_HIL_u.out

Chapter 5

Marine cybernetics laboratory

The laboratory is equipped for experimental testing of marine control systems and hydrodynamic tests. It consists of a wave basin with an advanced instrumentation package and a towing carriage. The basin, depicted in Figure 5.1, has dimensions 40m x 6.45m x 1.5m (LxBxD).

5.1 Equipment

5.1.1 Qualisys motion capture system

Qualisys provides 6 degrees of freedom data tracking. The system has millimeter precision, works in real time and is configured to 50Hz.

The positioning system consists of three Oqus high speed infrared cameras registering infrared reflectors placed on the vessel. Peer-to-peer (P2P) networking is used to transmit camera data to a dedicated computer running Qualisys Track Manager (QTM) software. QTM performs triangulation and broadcasts the vessel position over the HIL-lab network.

5.1.2 Towing carriage

The carriage runs at speeds up to 2m/s. It also has capability for precise movement of models in 6 degrees of freedom and is thus suitable for more specialized hydrodynamic tests.

5.1.3 Wave generator

The single paddle wave generator is controlled by a dedicated computer. Available spectrum are first order Stoke, JONSWAP, Pierson-Moskowitz, Bretschnei-

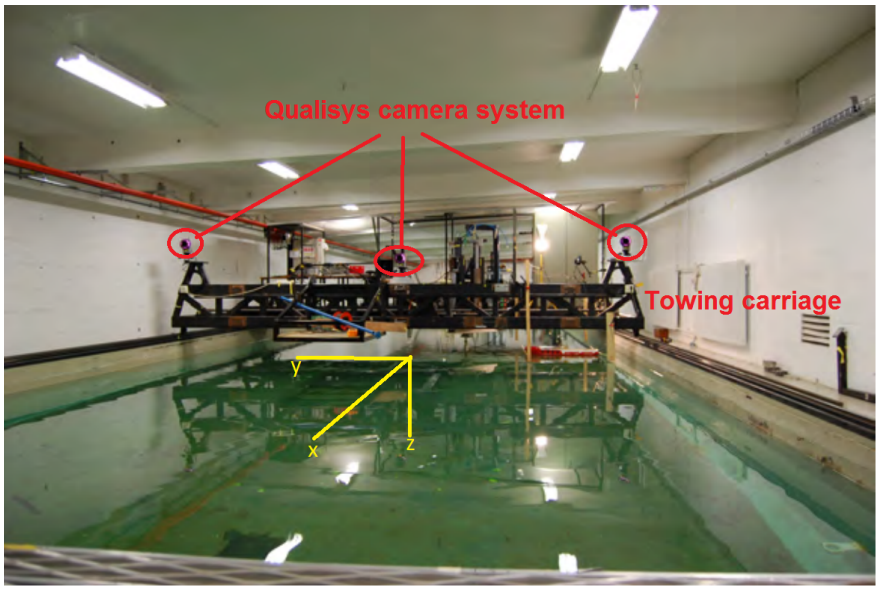


Figure 5.1: Marine cybernetics laboratory basin

	Height [m]	Period T [s]
Regular waves	$H < 0.25$	0.3 - 3.0
Irregular waves	$H_s < 0.15$	0.6 - 1.5

Table 5.1: Wave generator capacity

der, ISSC and ITTC. Table 5.1 summarizes the generation capacity.

Figure 5.2: CS Enterprise 1

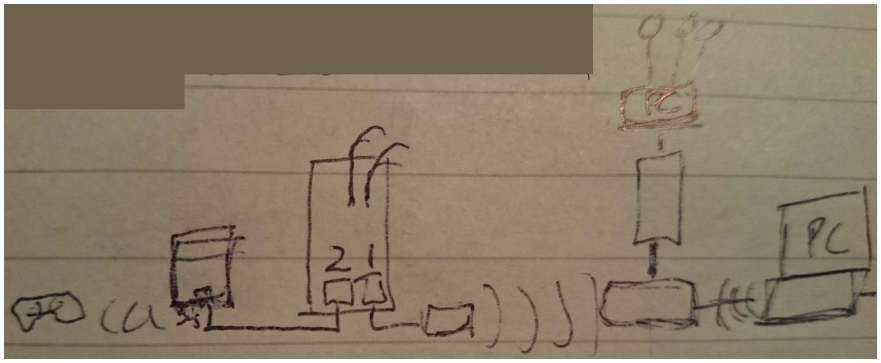


Figure 5.3: CSE1 communication

5.2 Vessels

5.2.1 Cybership Enterprise 1

CSE1, depicted in Figure 5.2, is a model ship fitted with two Voith Schneider propellers (VSP) astern and a bow thruster (BT). Its on-board control system consists of a RPi and a cRIO, as in the HIL laboratory setup described in Chapter 4, in addition to three electronic speed controllers (ESC) and four servos.

5.2.1.1 High-level communication

The communication is similar to the description of Section 4.1.3, with the exception of the wired Ethernet link between the cRIO and the laptop. For CSE1 this link instead passes through a Wi-Fi bridge to a wireless router to the laptop, as seen in Figure 5.3.

Broadcast QTM positioning data is retrieved through the same wireless network.

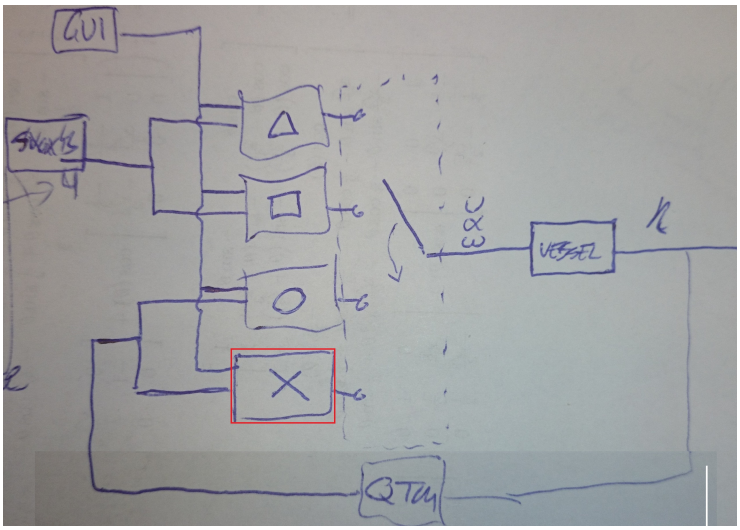


Figure 5.4: CSE1 generic control system

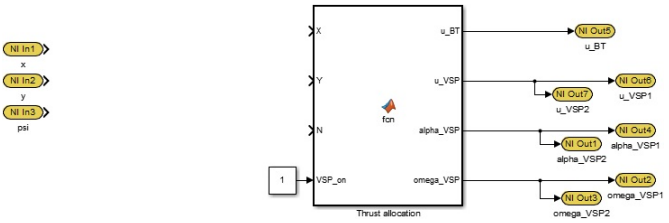


Figure 5.5: CSE1 `ctrl_student.slx` including thrust allocation

5.2.1.2 Low-level communication

The BT and VSP motor speeds are controlled by ESC. The ESC receive their setpoints as a pulse-width modulated (PWM) signals from the cRIO digital output module.

The VSP blade pitches are controlled by servos. The servos also receive their setpoint as PWM signals.

5.2.1.3 Control system

A diagram representation of CSE1's control system is given in Figure 5.4. The vessel can switch among four control modes, summarized in Table 5.2. The first three controllers are predefined, while users may implement their own controller in the fourth.

Implementation of the student controller is done in the `ctrl_student.slx` Simulink template, depicted in Figure 5.5. Detailed implementation steps are given in Section 7.2.

The generic control system consists of several Simulink modules, the details of

Sixaxis	Control mode
△	**Manual thruster control**
	VSP speed: directional pad up/down ±0.1
	Left joystick: VSP1 thrust
	Right joystick: VSP2 thrust
	L2/R2: BT thrust
□	**Manual forces and moment control**
	VSP speed: user interface button on/off
	Left joystick: surge and sway forces
	L2/R2: yaw moment
○	**Basic dynamic positioning (DP)**
	VSP speed: user interface button on/off
	Setpoint: user interface
	Gains: user interface
×	**Student controller**
	User implemented controller

Table 5.2: Generic control modes

which are given in Appendix B.2, a FPGA driver, described in Appendix A.1.3, and two custom device drivers.

5.2.2 Cybership 3

-

Part III

Laboratory user guide

Chapter 6

HIL simulation and testing

6.1 Simulink model adaptation and compilation

Complete the following steps to convert your model you created in Simulink into a compiled model that runs on RT targets.

Version compatibility is an issue for VeriStand-Simulink interaction. Mostly[1] Simulink code may be programmed in any version of the MATLAB, compilation, on the other hand, can only be done in version compatible with the intended VeriStand version. See Section A.3.

6.1.1 Modeling

6.1.1.1 Input and output

In order for the model to interact with VeriStand, special input and output blocks must be added to the block diagram[2]. These are found in the Simulink Library Browser under NI VeriStand Blocks.

6.1.1.2 Initial conditions

If the simulation is to be run with different initial conditions, one possible method is to allow external reset of the integrators. This is done right-click the integrator and selecting Block Parameters (Integrator) in the drop-down menu. Here, the reset condition is set. The initial condition source should be external, as in Figure 6.1.

[1] It has been experience that MATLAB function blocks are not compatible across versions. This results in build error message "invalid object ID". The MATLAB function block code must then be copied and pasted into a new MATLAB function block from the compatible version Simulink Library Browser.

[2] Ordinary input/source and output/sink blocks could be used at the diagram top level. However, subsystem ports are only available when using the VeriStand blocks.

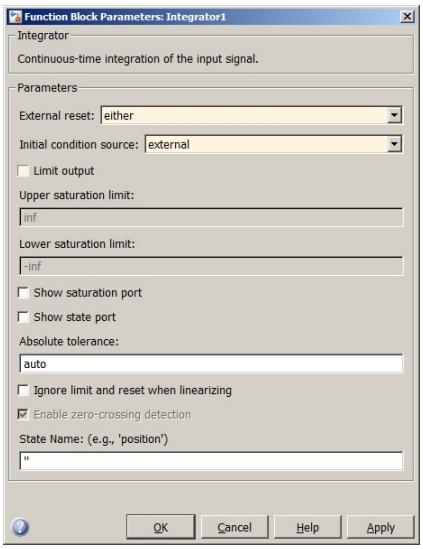


Figure 6.1: Integrator function block parameters

6.1.1.3 Real-time data logging

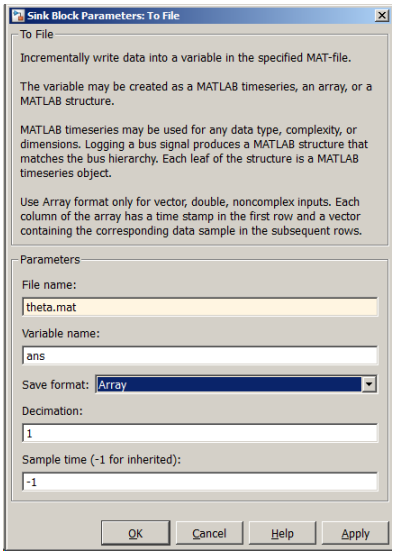


Figure 6.2: To File block parameters

Model output can be saved to the cRIO, for later retrieval through FTP, during simulation through a To File block. This block is found in the Simulink Library Browser under Sinks. The output file name is specified under the block parameters, as in Figure 6.2. The format should be set to Array, since the cRIO does not support the Timeseries format.

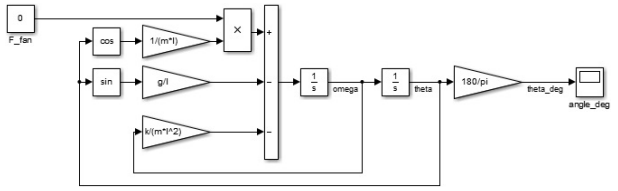


Figure 6.3: Simulink model for offline simulation

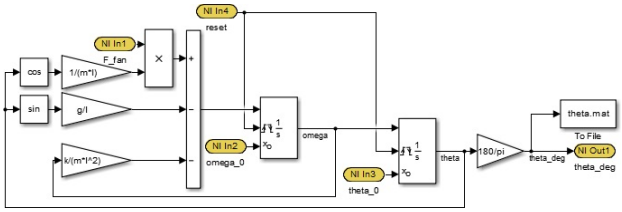


Figure 6.4: Simulink model for adjusted for compilation

Example: For a simple pendulum, $\dot{\omega} = -\frac{g}{l}\sin(\theta) - \frac{k}{ml^2}\omega + \frac{F_{\text{fan}}}{ml}\cos(\theta)$, the offline simulation block diagram could look as Figure 6.3. Figure 6.4 shows the same system adapted for VeriStand input, including reset and initial conditions, and output. The VeriStand blocks are yellow. omega_0 and theta_0 are ports corresponding to the initial conditions $(\omega(0), \theta(0))$. The integrators take these values whenever reset is rising or falling.

6.1.2 Model configuration

The code generation toolbox compiles the Simulink diagram to an output shared library in *.out format[3]. Model configuration parameters must be adjusted before generating, or building, the code.

The solver stop time should be `inf` (infinity) if the model is supposed to run until it is otherwise interrupted. The solver type must be fixed step. If your model only performs arithmetical operations, such as a mapping or transformation module would, the discrete solver should be used. If the model contains continuous states, i.e. if you have integrators, choose some differential equation solver such as ode3 or ode4. See Figure 6.5. Finally, the step size can be set: for a target running at 100 Hz, such as the cRIO-9024 default, a 0.01 step size results in the model running in simulating 1 second pr. second[4].

The correct target file should be selected depending on the target device. Select `NIVeriStand_VxWorks.tlc` for VxWorks targets[5], such as cRIO-9024, as in Figure 6.6.

[3]The *.out format is for targets running Wind River VxWorks real-time operating system (RTOS) such as cRIO-9024, while dynamic link libraries in *.dll format are for targets running IntervalZero Phar Lap ETS RTOS such as cRIO-9081.

[4]This can also be achieved by use of decimation, as described in Section 6.2.2.

[5]For PharLap targets, select `NIVeriStand.tlc`.

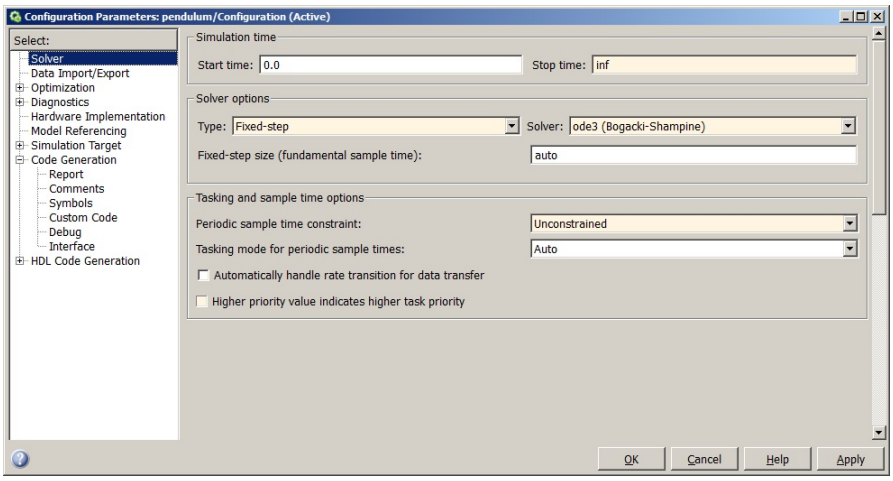


Figure 6.5: Simulink configuration parameters - solver

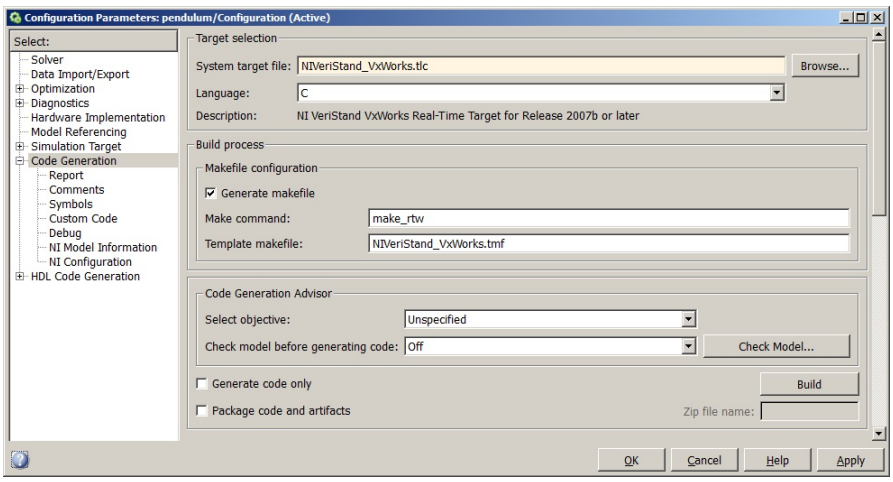


Figure 6.6: Simulink configuration parameters - target selection

The WindRiver GNU Toolchain must be present in the folder specified under NI Configuration, as in Figure 6.7.

6.1.3 Build

The build output is placed in a subfolder in the MATLAB Current Folder. The desired folder must therefore be active in the MATLAB main window, as in Figure 6.8, before compiling. The build subfolder name is [simulink model name]_niVeriStand_VxWorks_rtw.

The build is done in in Simulink, either with the Build button in the configuration window, by clicking the button, by the key combination CTRL+B, through the menu Code >C/C++ Code >Build model, or by pushing the icon

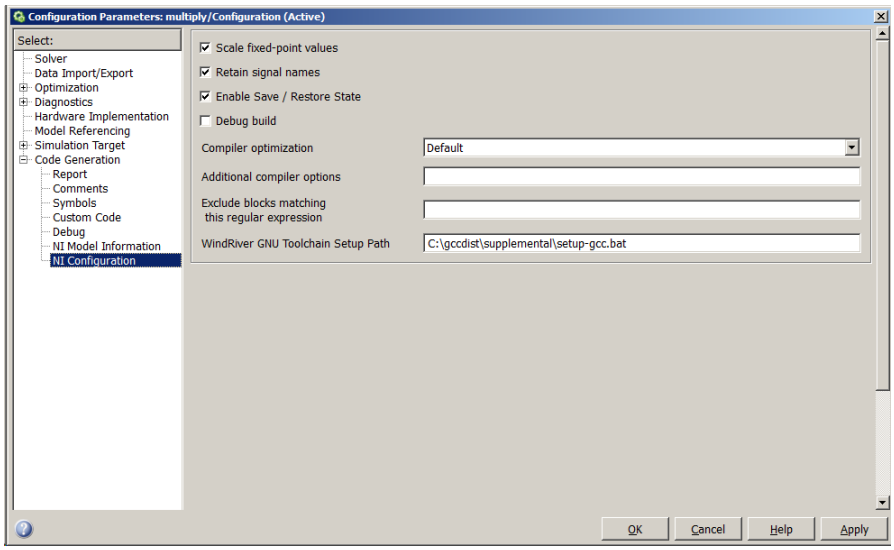


Figure 6.7: Simulink model configuration - NI configuration

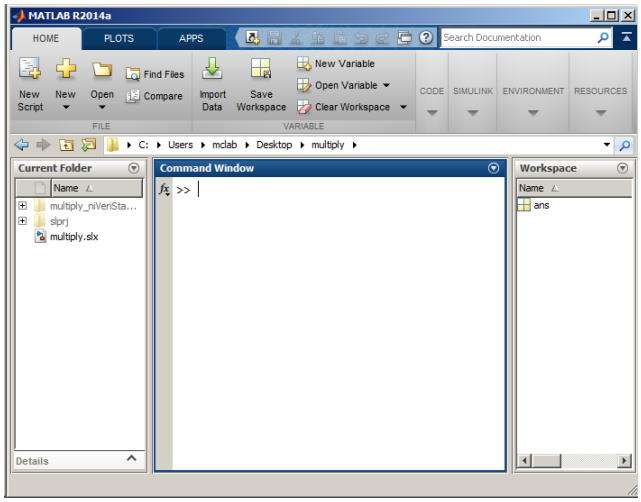


Figure 6.8: MATLAB console

button.

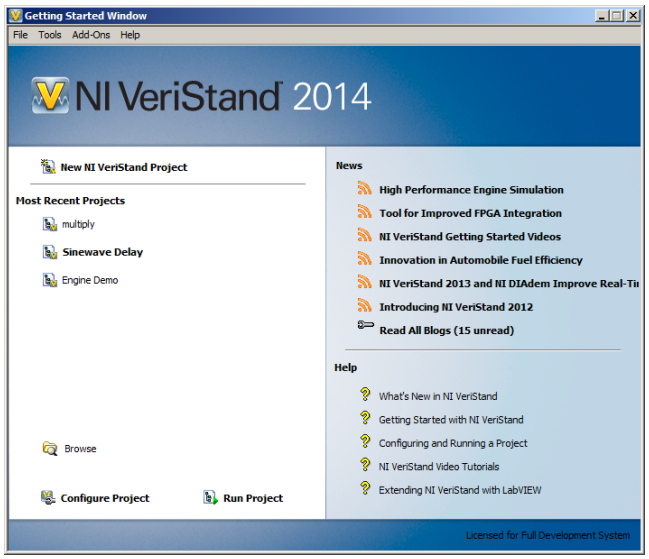


Figure 6.9: VeriStand start screen

6.2 Simulation configuration

Simulations are set up, deployed and interfaced through VeriStand. Figure 6.9 shows the start screen. Already configured projects can be run directly from here, or reconfigured.

6.2.1 Project creation

To deploy model for the first time, click New NI VeriStand Project. Give your new project a suitable name and location. Clicking OK creates the project files

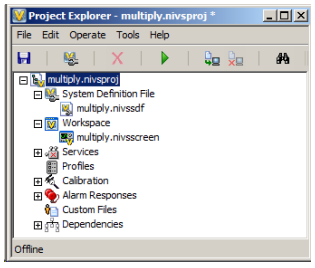


Figure 6.10: VeriStand Project Explorer

in given location and opens the Project Explorer, as in Figure 6.10. In this section, the example project name is multiply.

6.2.2 System setup

To configure the setup which will run on the cRIO, open the System Explorer by double-clicking the system definition file [project name].nivssdf.

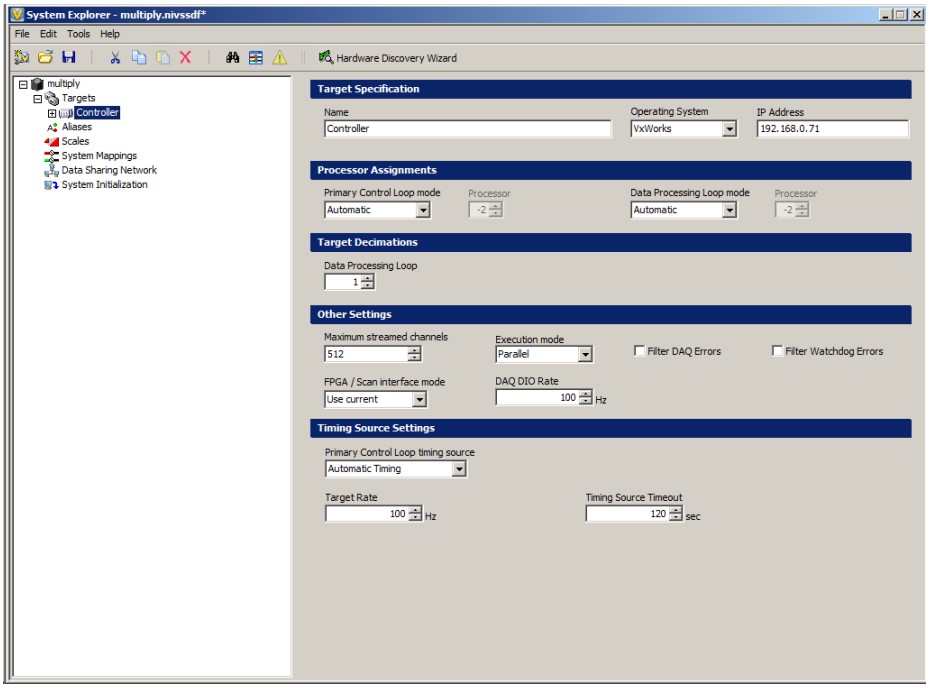


Figure 6.11: VeriStand - System Explorer - Controller

1. Set the correct controller operating system and IP address, as in Figure 6.11. All HIL and MC lab IP addresses are given in Table D.1. Also, note the target rate.

2. Click Add a Simulation Model, as seen at the top of Figure 6.12. Browse to the output of the Simulink compilation, as seen in Figure 6.13. Finally, click Auto Select Decimation to make sure the model runs at the intended rate.
   Repeat if several models should run simultaneously.

3. Add custom devices, such as network input, by right clicking the custom device pane and choosing the required device[6]. Figure 6.14 shows an example with the Sixaxis (WL_Joystick) device. Upon selection, a subfolder with the device name appears in the tree with signals listed inside it.

4. Configure mappings, by pushing the icon at the top of the window, to connect signals between custom devices, FPGA and models. Expand the trees to find the desired signals and click Connect, as in Figure 6.15.

5. Save and close to return to the Project Explorer.

[6]If the required device is not present, refer to the device driver installation instructions in Section A.1.4.

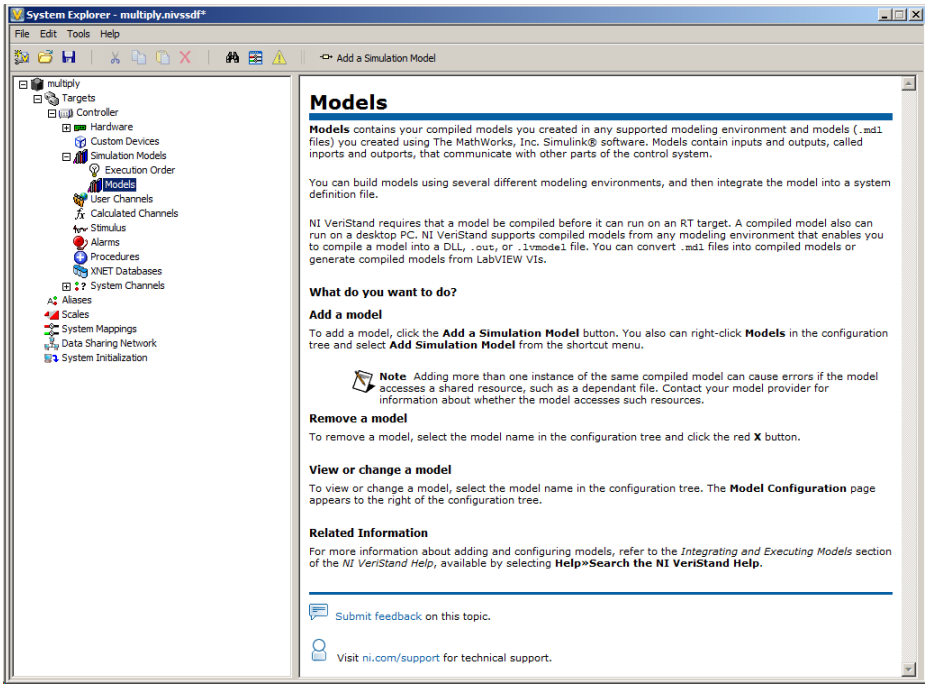


Figure 6.12: VeriStand - System Explorer - Models

6.2.3 Create computer interface

To configure the computer interface, open the Workspace editor by double-clicking the workspace file [project name].nivsscreen. The blank workspace pops up.

1. Enter Edit mode by Ctrl+M or Screen >Edit Mode.
2. Click the Workspace Control pane on the left side to access indicators, controls and such.
3. Drag and drop the desired item to the desired position in the workspace. Select the corresponding signal in the pop-up dialog.
4. Close the Workspace editor.

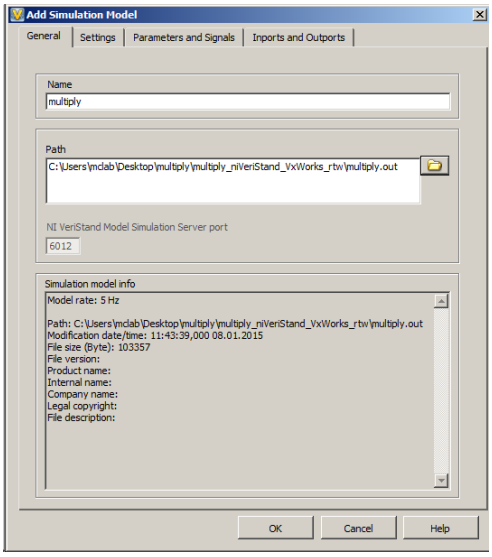


Figure 6.13: VeriStand - System Explorer Model

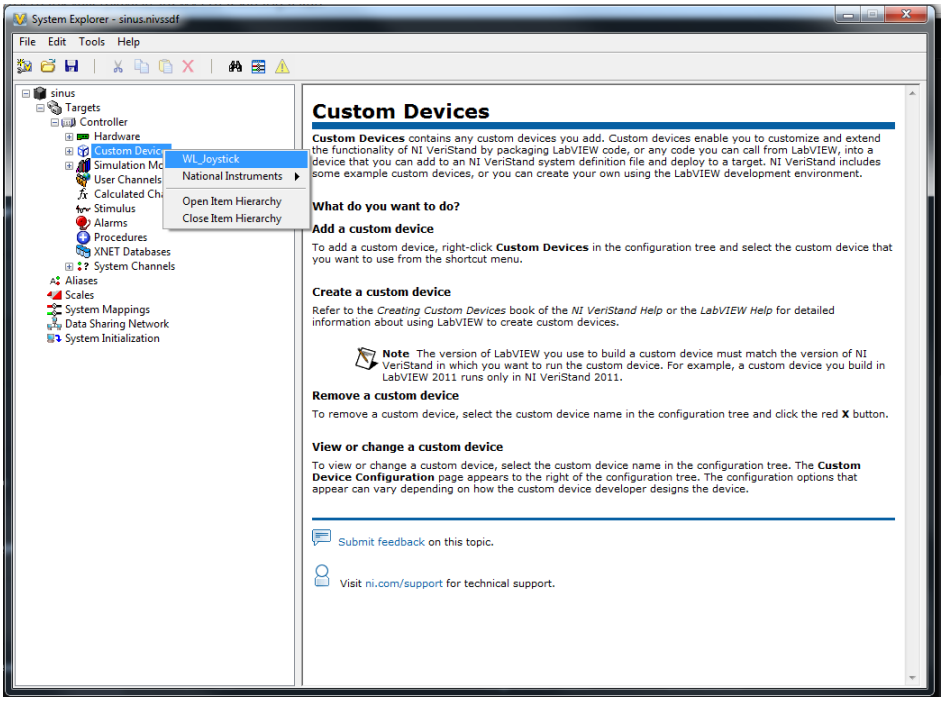


Figure 6.14: Custom device selection

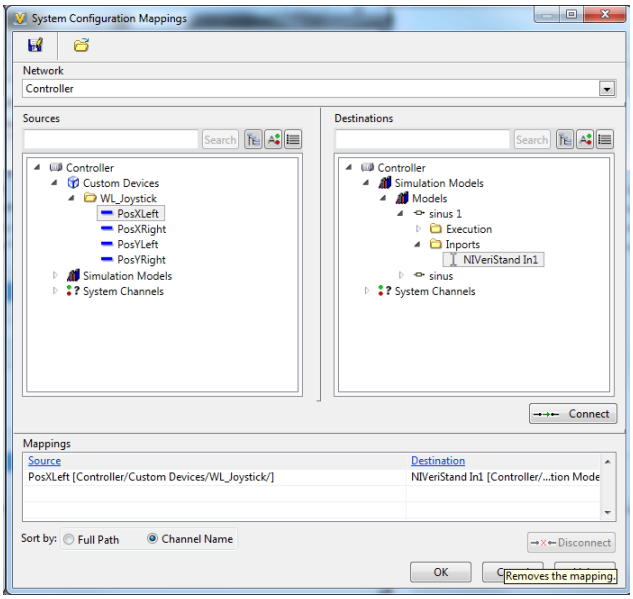


Figure 6.15: VeriStand System Configuration Mappings

6.3 Deployment and simulation

6.3.1 Run

Deploy by tapping the F6 key, or button, or Operate >Deploy. A dialog box appears. Upon successful deployment, the workspace pops up.

6.3.2 User interface side data logging

For reliability, it is recommended to log data directly on the cRIO during simulation, as described in Section 6.1.1.3. It is also possible to log via the laptop user interface.

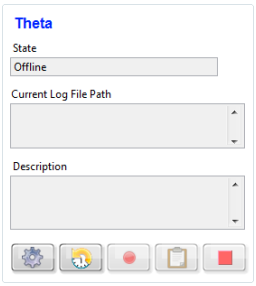


Figure 6.16: Logging Control

A Logging Control, as seen in Figure 6.16, must be added to the workspace to export data from the simulation. The control is added as described in Section 6.2.3.

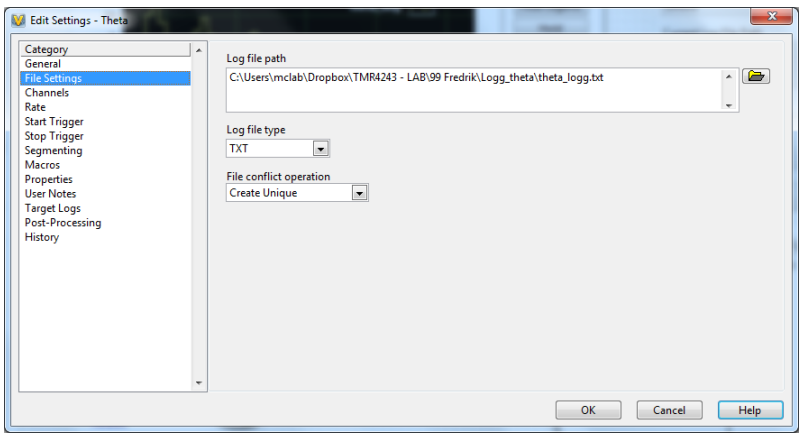


Figure 6.17: Logging Control file settings

Once the control is added, a pop-up window allows to edit the settings. The log file path is specified under File Settings, see Figure 6.17. Under Channels, the desired channels can be selected and added, as in Figure 6.18.

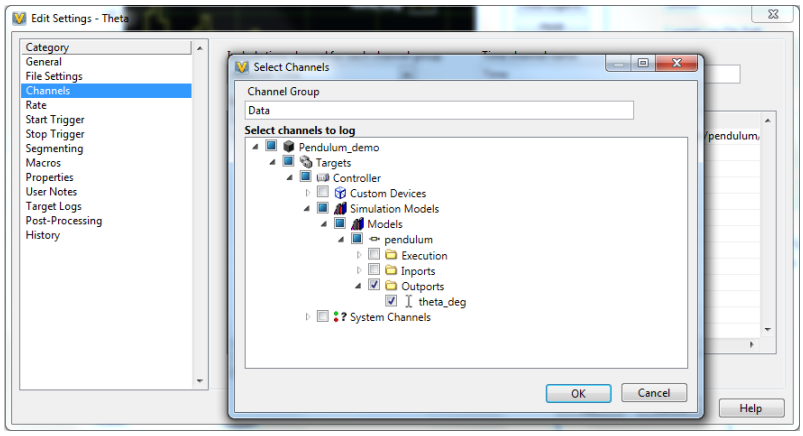


Figure 6.18: Logging Control add channel

6.3.3 Stop

button

6.3.4 FTP data retrieval

Data logged on the cRIO through To File blocks can be retrieved after simulation over FTP with software such as WinSCP.

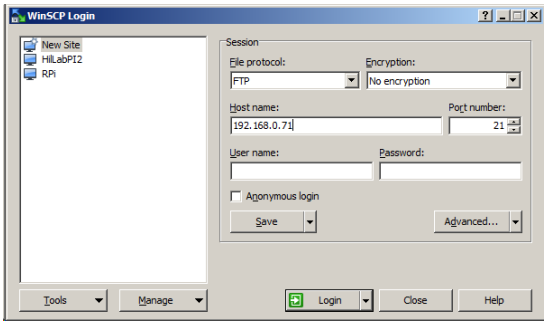


Figure 6.19: WinSCP login

To connect to the cRIO, the correct IP must be specified, as in Figure 6.19. For the standard HIL setup, the user name and password are blank.

Logged data with file names corresponding to the To File block names are located on the cRIO root, as seen in the right pane of Figure 6.20. Data is transferred to the laptop by drag and drop to the desired location in the left pane.

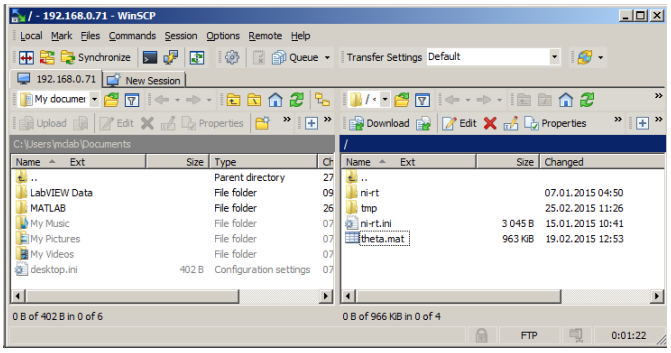


Figure 6.20: WinSCP

Chapter 7

CSE1 model scale testing

7.1 Safety - hazards and measures

7.1.1 Personnel injury

7.1.1.1 Drowning

It is required to have two or more persons present when using the basin.

7.1.1.2 Electric shock

The towing catenary should not be approached or touched.

7.1.1.3 Carriage collision

It is forbidden to run the towing carriage when there are people alongside the basin.

7.1.1.4 Thruster blade cuts

CSE1 must stay in the water as long as actuators are active. Before removing the vessel from the water, the control system must be stopped and the VeriStand project undeployed.

7.1.2 Material damage

7.1.2.1 Cybership Enterprise 1

Water damage CSE1 is not waterproof and has excessive thrust capability which can inflict large roll angles. The risk of water on deck is reduced through

thrust limitation and HIL testing before application of new control algorithms.

Propeller dry running BT must only be run in water. Before removing the vessel from the water, the control system must be stopped and the VeriStand project undeployed.

Loss of laptop control Wireless network instability may result in loss of connection between the laptop user interface and the cRIO. In this event, fall back to manual thruster control, by pushing △ on the Sixaxis.

Loss of position measurement -

Total loss of control Pull the vessel with a boat hook. Keep the CSE1 in water while disconnecting batteries.

7.1.2.2 Towing carriage

Stop before automatic stop at high speeds.

7.2 Student controller implementation

1. Unzip the CSE1 Veristand Project `CSE1.zip` to `C:\CSE1\`.[1]
2. Simulink implementation and compilation
   (a) Update `ctrl_student.slx` according to your controller design. Additional input and output, resets and data logging may be added, as described in Section 6.1.1.
   Do not alter the predefined input and output: x, y, psi, u_BT, u_VSP1, u_VSP2, alpha_VSP1, alpha_VSP2, omega_VSP1 and omega_VSP2.
   (b) Select a suitable solver, as described in Section 6.1.2.
   The remaining configuration, such as target selection is preselected in the file.
   (c) Compile the model as described in Section 6.1.3. The MATLAB current folder should be `C:\CSE1\`, in order to ensure that the resulting `.out` file is created in `C:\CSE1\ctrl_student_niVeriStand_VxWorks_rtw`.
3. CSE1 Veristand Project configuration
   (a) Open `CSE1.nivsproj`[2] to access the project.
   (b) Update `ctrl_student.out`:
      i. Open the System Explorer by double-clicking the system definition file `CSE1.nivssdf`.
      ii. Browse the left pane tree, as seen in Figure 7.1, and select ctrl_student. Refresh by pushing the icon.
      iii. If necessary, add mappings.
      Do not change the existing mappings. Position input (x, y, psi) and controller output (u_BT, u_VSP1, u_VSP2, alpha_VSP1, alpha_VSP2, omega_VSP1, omega_VSP2) are already mapped as necessary.
      iv. Save and close to return to the Project Explorer.
   (c) Implement a suitable workspace, as described in Section 6.2.3, for your controller in control screen 4: ctrl_student. Figure 7.2 shows control screen 4 before customization.

[1] Other paths are possible but require changing all VeriStand project paths.
[2] Not `CSE1.nivssdf`, since not only the system definition should be altered.

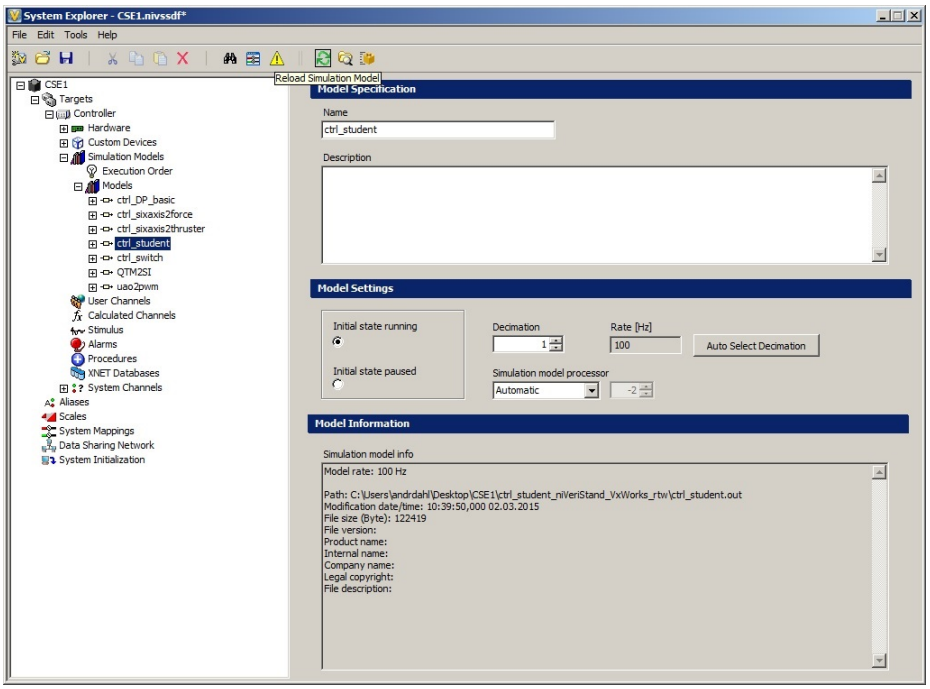


Figure 7.1: CSE1 Veristand Project simulation models

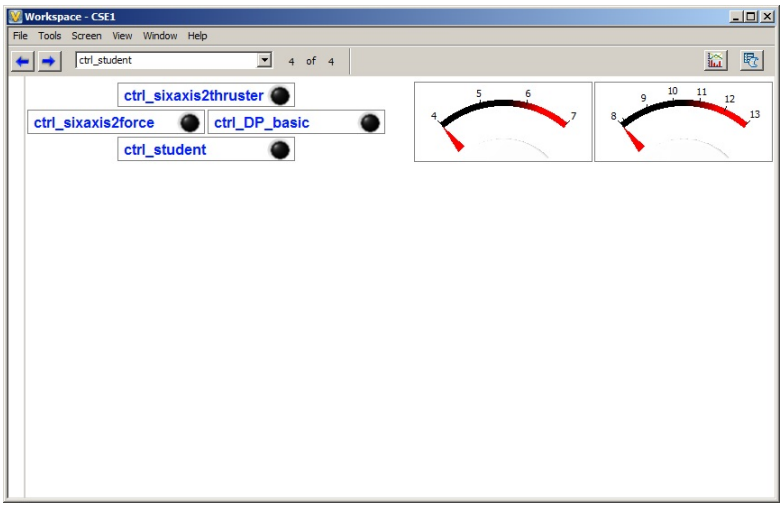


Figure 7.2: CSE1 Veristand Project ctrl_student workspace

7.3 Ship launching procedure - before sailing

7.3.1 Power up and connection

1. Place the batteries adjacent to the watertight box: main battery battery (12 V) astern and secondary battery (6 V) in the bow.
2. Connect main battery: first the red wire to the red/positive pole, then the black wire to the black/negative pole[3].
   cRIO LED nr.1 (power) will light up green.
3. Wait for cRIO and RPi start up.
   When complete, the Bluetooth dongle blue LED blinks evenly at approximately 1 Hz.
4. Turn on Sixaxis by pushing the PS3 button.
   When succesfully connected, the Bluetooth dongle blue LED is almost constantly lit and the Sixaxis' red LEDs 1, 2, 3, and 4 blink at approximately 2 Hz.
5. Connect the seconday battery: red wire to the red/positive pole, then the black wire to the black/negative pole.
   The Wifi bridge Power LED will light up green.
6. Wait for WiFi connection to HILlab network.
   When connected, the Wifi bridge WLAN green LED turns on.

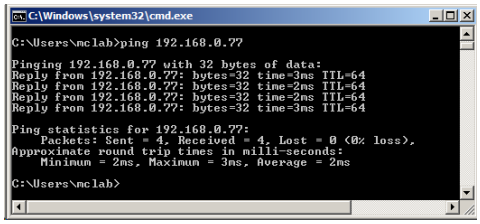


Figure 7.3: Ping, successful access to CSE1

7. Verify laptop access: ping the CSE1 IP in the command promt, as in Figure 7.3. While the round trip times may vary, it is essential to have 0% loss.
8. Gently place the vessel in the basin, avoiding any water splashes.

7.3.2 Positioning system

[3]The connection order of the wires should not matter. However, experiences favor the this order of connection.

7.4 Deploy control system

Veristand osv

7.5 Ship docking procedure - after sailing

Undeploy the running project to disable all actuators.

Lift out of water avoiding water on rail.

Put CSE1 in its stand. The vessel should not be left on the water for extensive periods, i.e. overnight.

Remove and put used batteries to charge. Load fresh batteries in vessel.

Connect the Sixaxis gamepad to the laptop for charging.

Part IV

TMR4243 exercises and expected results

General report requirements.

Chapter 8

Pendulum lab

Adjust Simulink model for VeriStand.

Deploy

Create suitable workspace

Actuate fan manually.

Simulate to get same results as Simulink.

Export data

Try with handed out model (surprises).

Chapter 9

Internal dynamics lab

Chapter 10

Estimation lab

10.1 Objective

Teste konseptet, tau inputt,

10.2 Theory and simulation

Pure sway and surge dampings

$$d_{\text{pure surge}} = d_{11} u \tag{10.1}$$

$$d_{\text{pure sway}} = d_{22} v \tag{10.2}$$

Common DP model

$$d_{11}(u) = -X_u - X_{|u|u}|u| - X_{uuu} u^2 \tag{10.3}$$

$$d_{22}(v, r) = -Y_v - Y_{|v|v}|v| - Y_{vvv} v^2 - Y_{|r|v}|r| \tag{10.4}$$

however, Fitting the drag force to experiemtal results yields the best fit for

$$d_{11}(u) = -X_u - X_{uu} u - X_{uuu} u^2 \tag{10.5}$$

$$d_{22}(v) = -Y_v - Y_{vv} v - Y_{vvv} v^2 \tag{10.6}$$

with parameters as listed in Table 10.1.

10.2.1 Tasks

1. Use (10.5) to find the thurst required to achieve 0.6 m/s and -0.6 m/s in surge velocity. Equivalently with (10.6) and velocities 0.3 m/s and -0.3 m/s in sway.

Surge		Sway	
X_u	-0.6555	Y_v	-1.330
X_{uu}	0.3545	Y_{vv}	-2.776
X_{uuu}	-3.7870	Y_{vvv}	-64.910

Table 10.1: CSE1 hydrodynamic damping parameters

$$X = -X_u u - X_{uu} u^2 - X_{uuu} u^3$$
$$Y = -Y_v v - Y_{vv} v^2 - Y_{vvv} v^3$$

2. Motion control

   switche mellom sway og surge fixed, alltid rotasjon fixed

3. Surge hydrodynamics $\begin{bmatrix} X_{\dot{u}} \\ X_u \\ X_{|u|u} \end{bmatrix}$

   design reasonable towing input

4. Sway hydrodynamics $\begin{bmatrix} Y_{\dot{v}} \\ Y_v \\ Y_{|v|v} \end{bmatrix}$

5. Real surge towing data $\begin{bmatrix} X_u \\ X_{|u|u} \end{bmatrix}$

10.2.2 Specific report requirements

10.3 Hardware-in-the-loop simulation

10.3.1 Tasks

1. Motion control

   discrete and switcing controller

2. Real-time application configuration

   (a) Create a VeriStand simulation project including the compiled control system ctrl_student.out and the CSE1 HIL model. i. Map control signals and measurements according to Tables 1 and 2, respectively.

   (b) Develop a suitable graphical user interface.

3. Simulate to estimate surge, sway and yaw hydrodynamic parameters separately with the discrete controller as control input. Suitable velocity thresholds are u = 0; 25, v = 0; 20 and r = 0; 30. Keep CSE1 within the line f(x; y) j1 x 7:5; y = 0:25g to ensure position measurements.

10.3.2 Specific report requirements

10.4 Model scale testing

HIL

Chapter 11

The maneuvering lab

Part V

Equipment setup and configuration

Appendix A

HIL lab setup

A.1 cRIO

The cRIO runs Wind River VxWorks real-time operating system.

A.1.1 Ethernet ports

The cRIO has two Ethernet ports the primary communicates with the PC and the secondary with the Raspberry PI.

A.1.1.1 Primary

Set fixed IP, set fixed IP on HIL-computers

A.1.1.2 Enabling the secondary ethernet port

1. Start *NI MAX*
2. In the left pane tree, select the cRIO under *Remote Systems*
3. Open the *Network Settings* tab (located at the bottom of the window)
4. Set *Adapter Mode* to *TCP/IP Network*
5. Set *Configure IPv4 Address* to *Static*

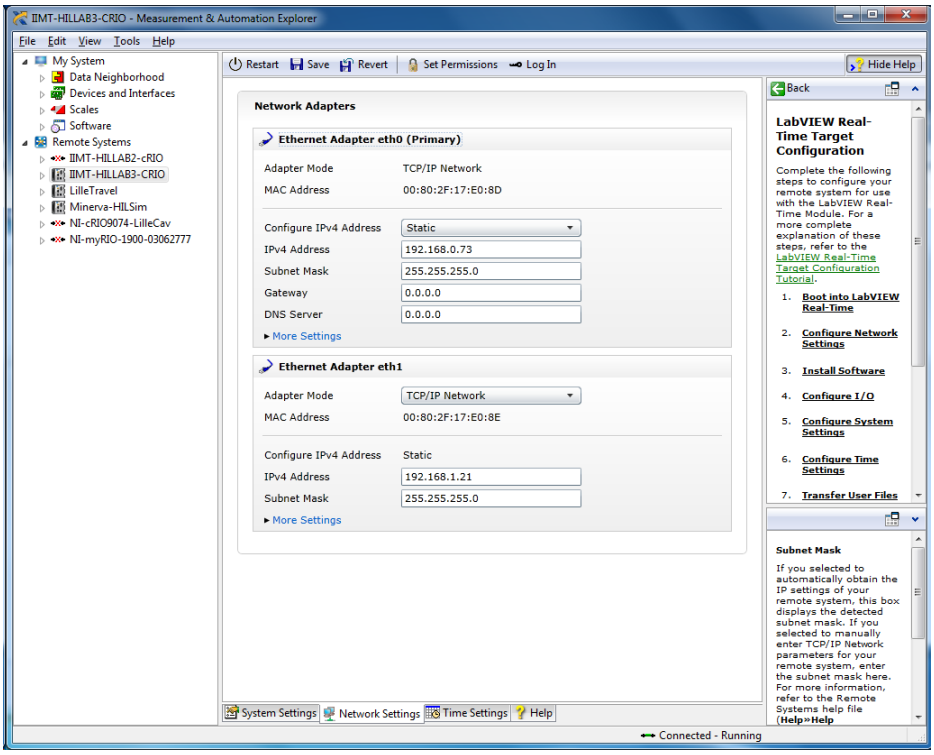


Figure A.1: NI MAX - Network Settings

A.1.2 Update cRIO software

To be able to run the models on the cRIO, the software version on the cRIO and PC must match. In addition you must install the NI Veristand Engine. Software changes on the cRIO is handled in NI Max.

A.1.2.1 Update

1. Open NI Max
2. Find your cRIO on the left hand side and click it
3. Click Software, and then Add/Rem ove Software located on the top pane, see Figure A.2
4. A new window will now open. Choose the option that matches your LabVIEW/Veristand edition (in our case 14.0 or 14.0.1) under LabVIEW Real-Time 14.0.0 and click next. See Figure A.3
5. Click next without making any changesA.4
6. Click next without making any changesA.5
7. Wait for the installtion to finish and the cRIO to reboot

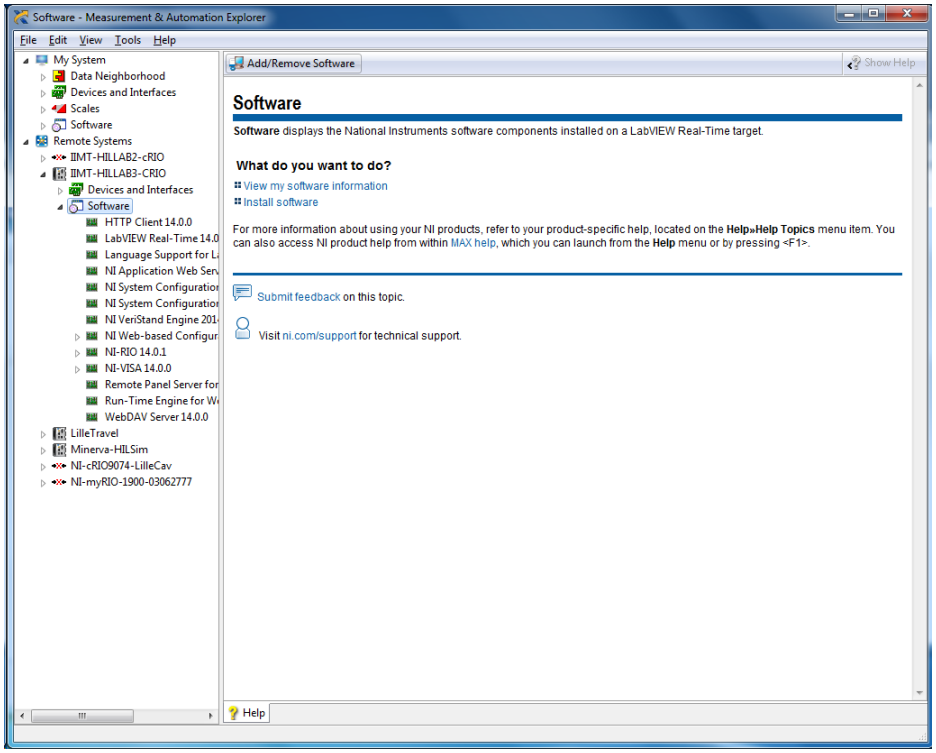


Figure A.2: NI MAX - Software Update 1

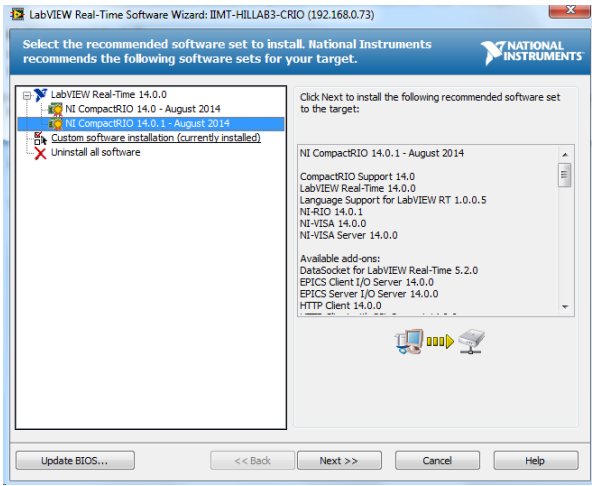


Figure A.3: NI MAX - Software Update 1

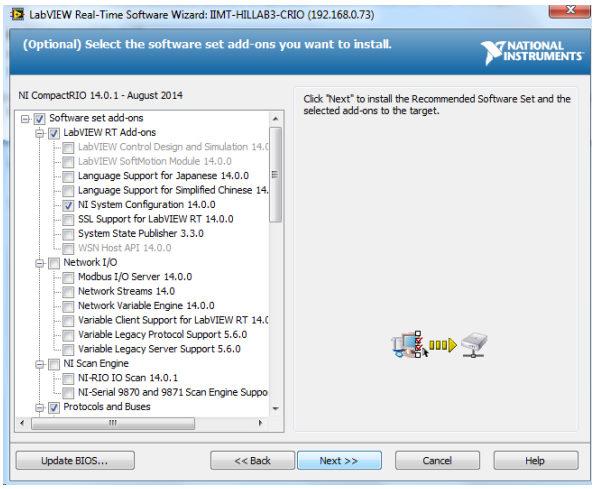


Figure A.4: NI MAX - Software Update 3

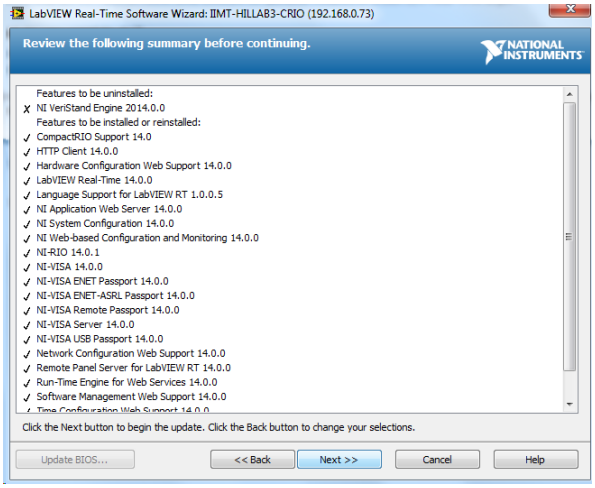


Figure A.5: NI MAX - Software Update 4

A.1.2.2 NI Veristand Engine

1. Repeat step 1-3 from the previous guide
2. Now you choose Custom Software installation in the menu, see Figure A.6
3. Ignore the warning, See Figure A.7
4. Locate NI Veristand Engine 2014 0.0 and click install feature. See Figure A.8
5. Click your way through the rest of the installation and let the cRIO reboot.

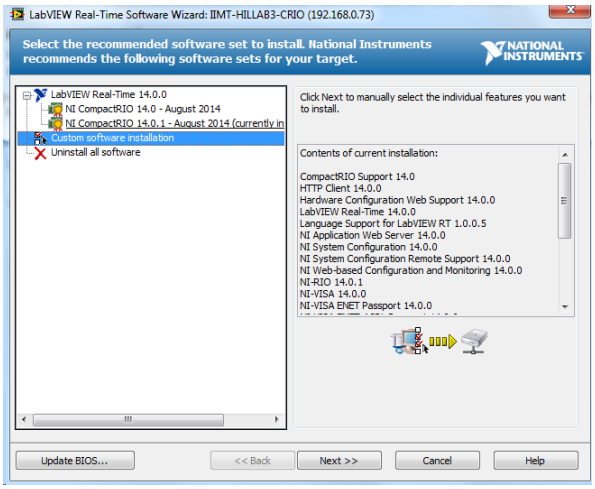


Figure A.6: NI MAX - NI Veristand Engine installation 1

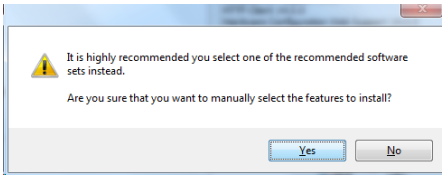


Figure A.7: NI MAX - NI Veristand Engine installation 1

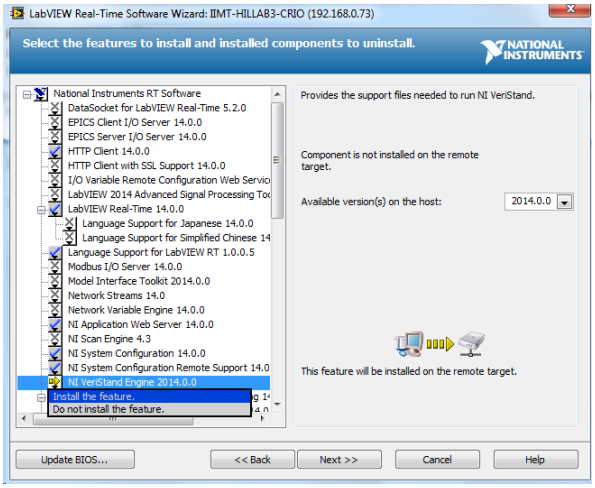


Figure A.8: NI MAX - NI Veristand Engine installation 1

A.1.3 Create FPGA target and XML

If you do not haav a Veristand FPGA target at your disposal, follow the steps below. If you have a target available and just need to install it in NI Veristand, please jump to the next subsection

1. Open LabVIEW and create new project. We have done this in LabVIEW 2013 because of some instalation issues with LabVIEW 2014, but the procedure should be the same for both editions.
2. Choose NI Veristand FPGA Project in project templates and proceed.
3. Choose CompactRIO Reconfiguarble Embedded System and click next.
4. You will now get the choice between letting LabVIEW detect your cRIO system or configure it yourself. If you are connected to the cRIO and it has all of the I/O ports connected, the option "Discover existing system" is simpler and therefore recommneded. If you do not have your cRIO connected choose "Create new system", this is the version that will be worked through here.
5. Select your controller, in our case cRIO-9024.
6. Select your FPGA target, in our case cRIO-9113.
7. Then you select your I/O modules to the correct slots. In our case NI 9215 in slot 1 and NI 9474 in slot 4.
8. You are now finished with configuring your project. Press next.
9. The project menu will now appear and should look something like Figure A.18. Select the LabVIEW VI as demonstrated our is called Custom Personality FPGA.vi
10. The UI window will now present itself, select window and show block diagram.
11. You should now see a block diagram similar to Figure A.19. You will now have to redesign this to look like Figure A.20. This will be valid for our system, if you have different I/O modules the block diagram need to reflect this.
12. Now, return to the Project explorer and select Build Specifications and Custom Personality FPGA
13. A new window will open. Check that the name and project path is correct and press build.
14. Select your preferred compile server. The compilation process will take quite some time (approx 15-30 min).
15. When the compilation process is finished, the last step is to edit the automatically generated XML file. You will now have to find you project directory in Windows. Here there will be a folder called bitfiles which contains the files you compiled in the last step, there will also be a .XML file. The point of editing this file is to match the actually compiled VI, meaning the packets must match the connected I/O. The recommended

way to edit the file is to copy our XML file from: Dropbox\TMR4243 - LAB\04 cRIO software\FPGA IO. You will have to make sure that the name of your bitfile matches the name in the XML file as seen in Figure A.25, also make sure the I/O modules matches your setup.

16. Copy the bitfiles from the bitfile folder to the level above so that the bitfile aand the XML file is in the same folder.

Documentation: `https://decibel.ni.com/content/docs/DOC-13815`

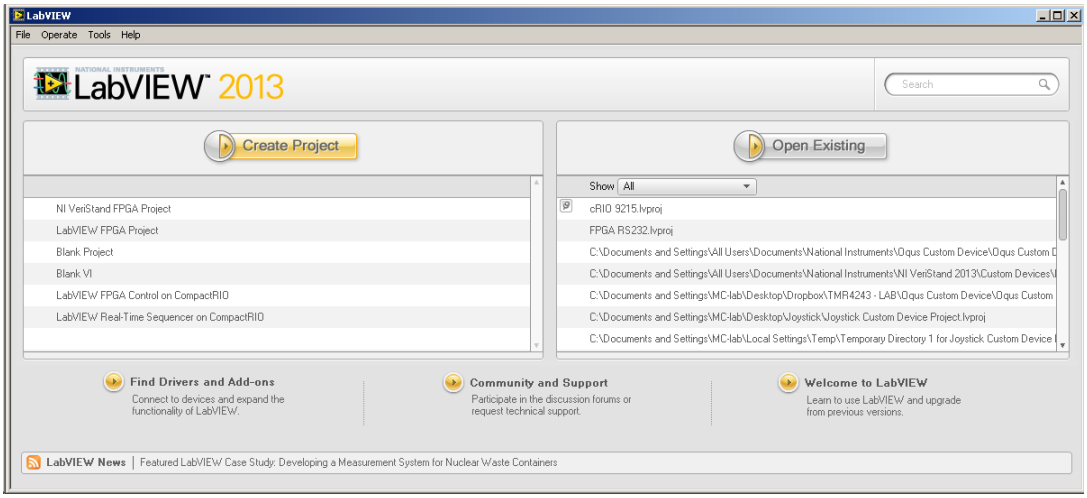


Figure A.9: Create Labview FPGA target and XML - 1

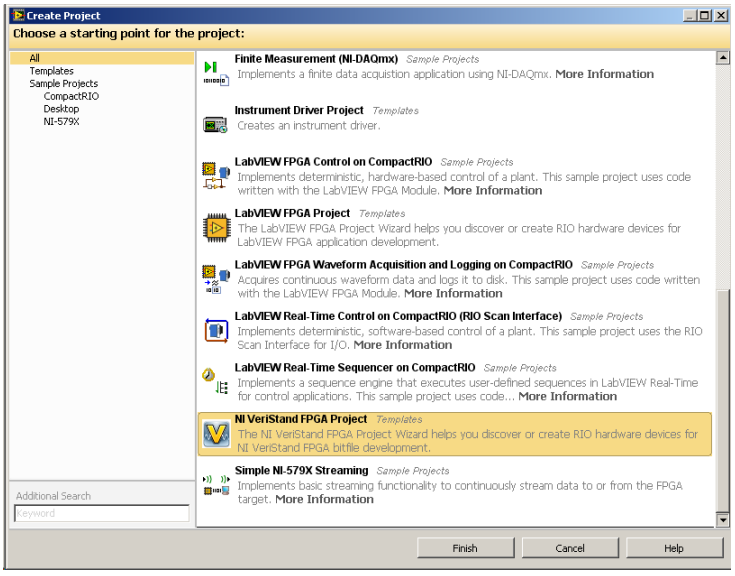


Figure A.10: Create Labview FPGA target and XML - 2

Veristand FPGA programmering

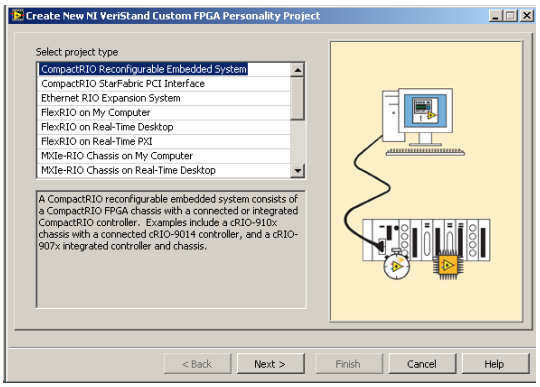


Figure A.11: Create Labview FPGA target and XML - 3

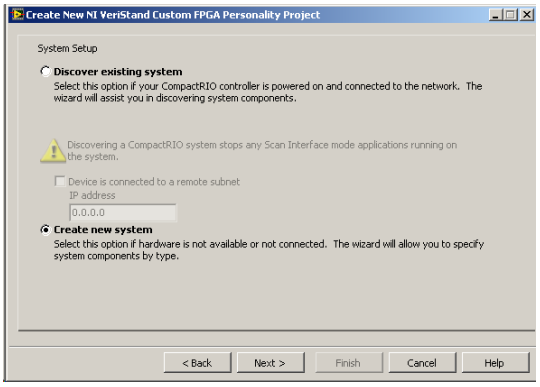


Figure A.12: Create Labview FPGA target and XML -4

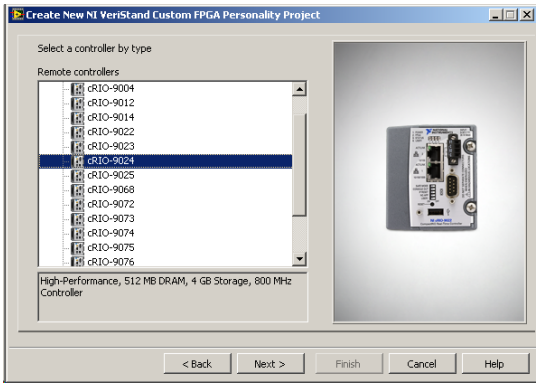


Figure A.13: Create Labview FPGA target and XML - 5

In order to access the analogue and digital I/O modules on our cRIO from Veristand, it is necessary to create a FPGA target in Labview with Labview and you will have to write a custom XML file.

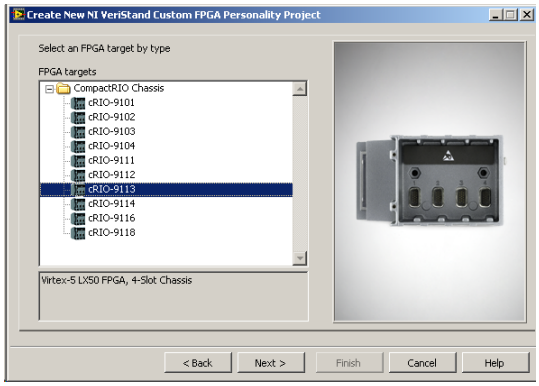


Figure A.14: Create Labview FPGA target and XML - 6

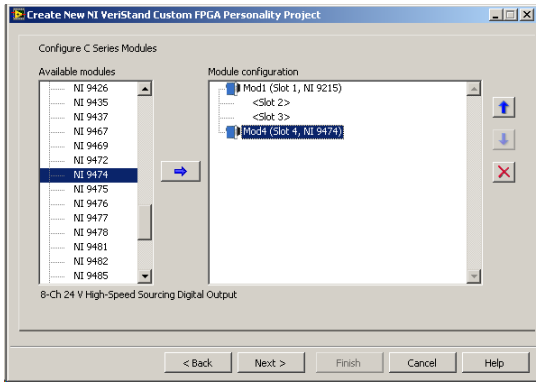


Figure A.15: Create Labview FPGA target and XML - 7

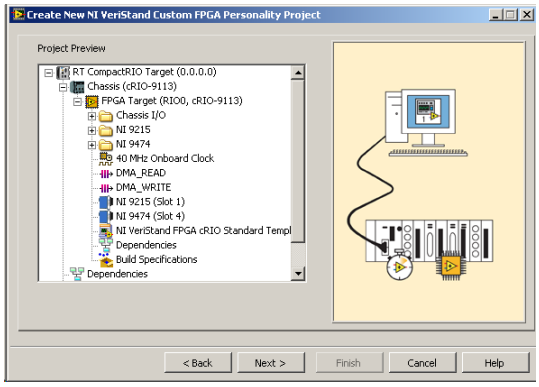


Figure A.16: Create Labview FPGA target and XML - 8

A.1.3.1 Install in veristand

The Veristand software does not recognize the physical I/O components of the cRIO. It is necessarry to write a specific FPGA mapping for the specific setup. This results in a XML file that maps the ports.

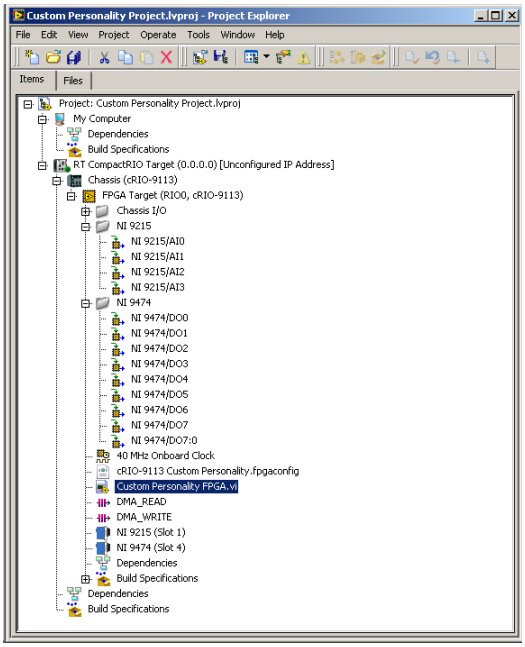


Figure A.17: Create Labview FPGA target and XML -9

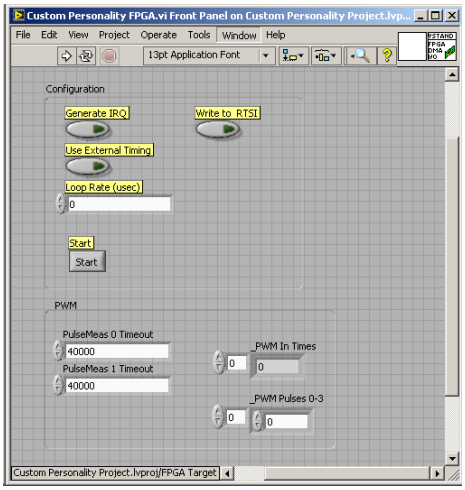


Figure A.18: Create Labview FPGA target and XML - 10

To add this file to your Veristand project, enter the system explorer and find the FPGA pane under *targets\controller\hardware\chassis*, as seen in figure A.26.

The next step is to find your XML file. In this case called cRIO-9113 Ex, it is very important that the XML file is placed on level above the FPGA bitfile folder in the directory system, as the files are really being used are the FPGA bitfiles.

The menu in should now look something like Figure A.27, here you can see the

analogue input signals and the digital output PWM signals. These can again be linked to other signals as seen in FigureA.33.

PWM

Ticks og sånt tick = FPGA clock pulse

$$\text{tick in seconds} = \frac{1}{\text{frequency}} = \frac{1}{40MHz} = \frac{1}{40 * 10^6} = 25 * 10^{-}9 = 25ns$$

output at 50 Hz demands output every

$$\frac{40MHz}{50Hz} = \frac{40 * 10^6}{50} = 800000tick$$

VeriStand FPGA programming LabView -¿ Create project -¿ All -¿ NI VeriStand FPGA project -¿ Compact RIO -¿ Discover existing system -¿ Velge eget utstyr -¿ Vente på discovering -¿ I Project explorer *.vi (er bitfilen) *.fpgaconfig (egentlig XML) Endre på *.vi Fjerne overføldige pakker Oppdatere antall pakker i XML-filen og fjerne pakker som ikke er aktuelle, oppdatere tall på beholdte pakker. Kompiler

Kopier bit-file ut i samme mappe som *.fpgaconfig I System exlporer, FPGA -¿ Add FPGA target -¿ Finne *.fpgaconfig

Analog input

Eirik: FPGA-greier osv

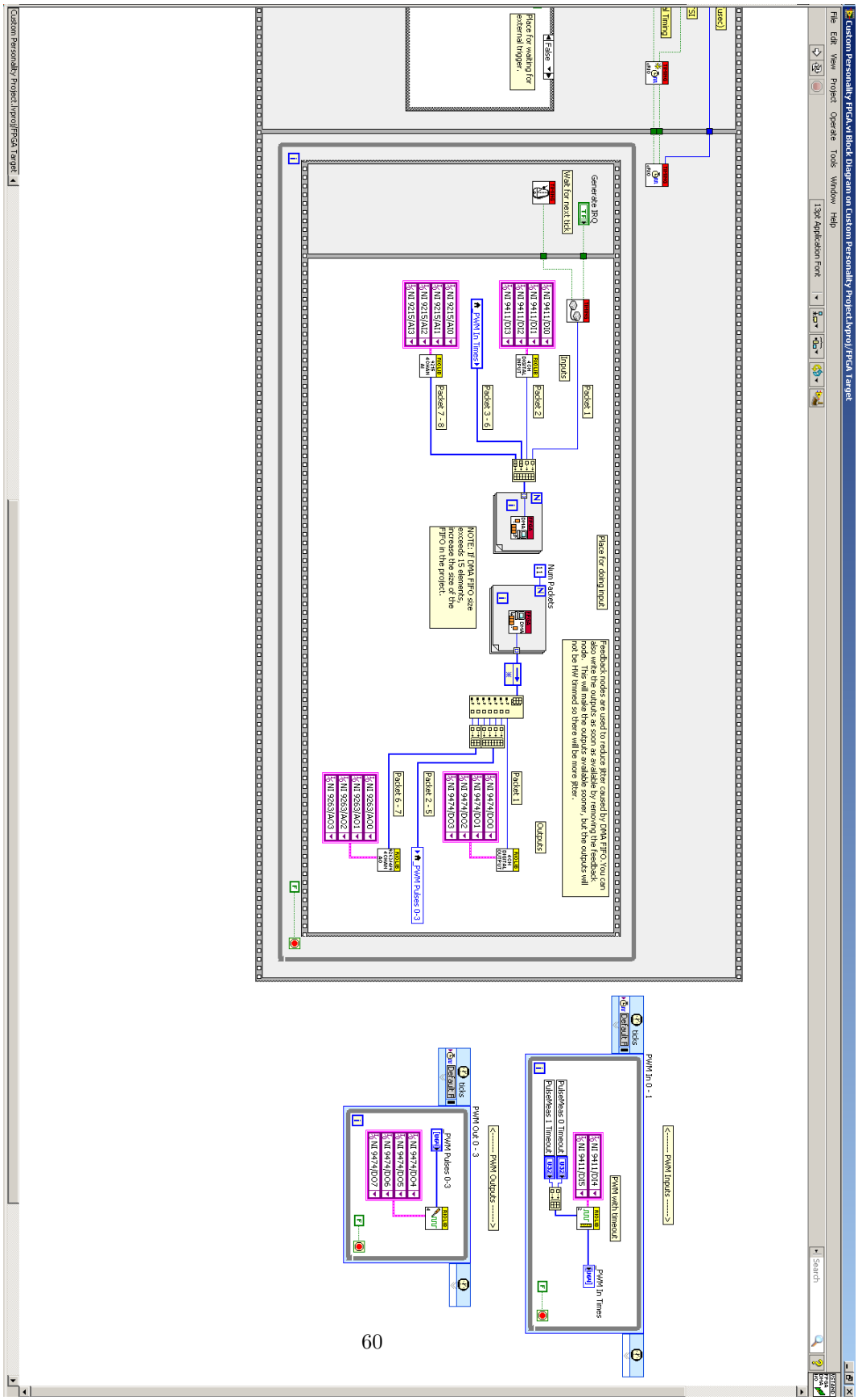


Figure A.19: Create Labview FPGA target and XML - 11

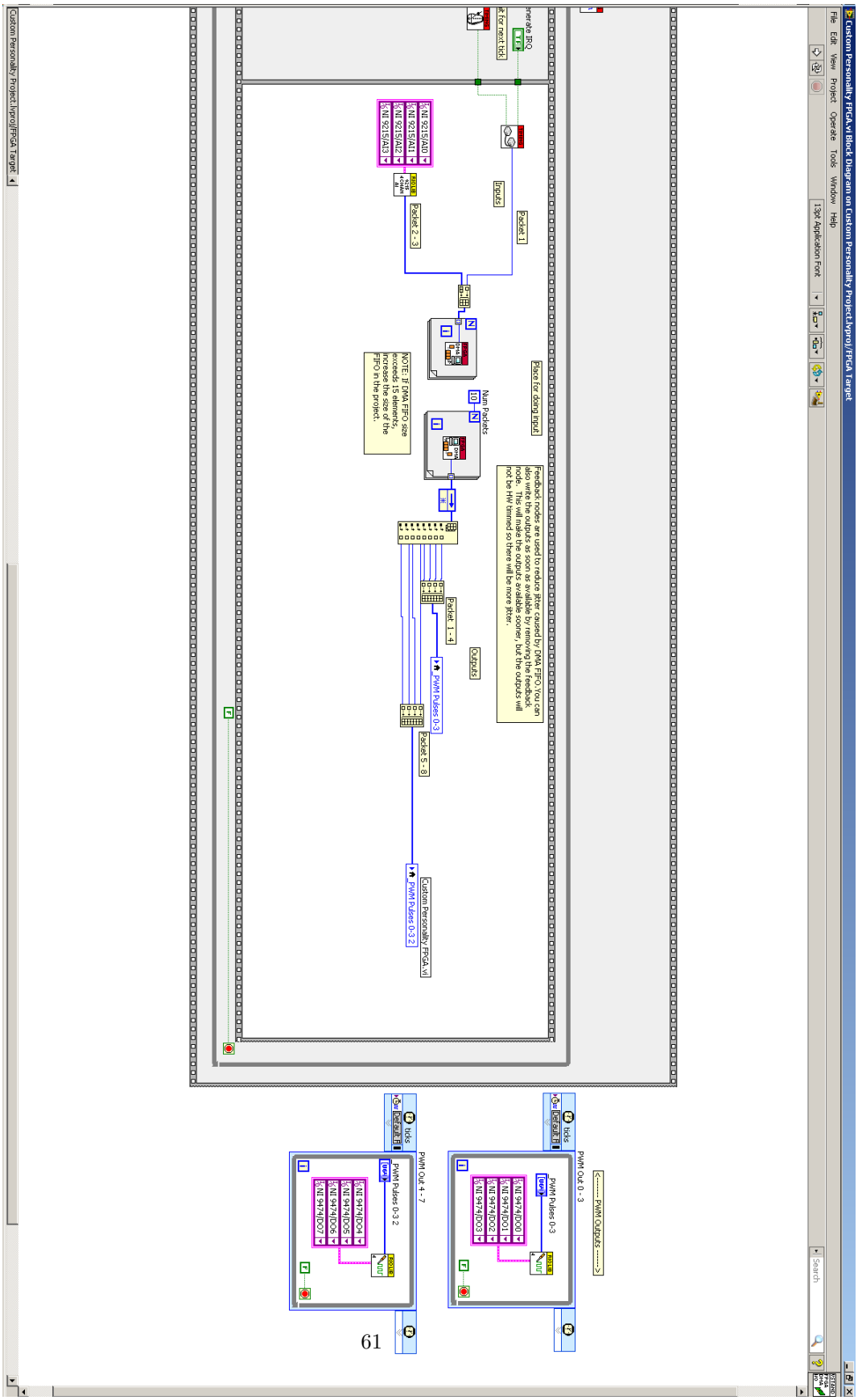


Figure A.20: Create Labview FPGA target and XML - 12

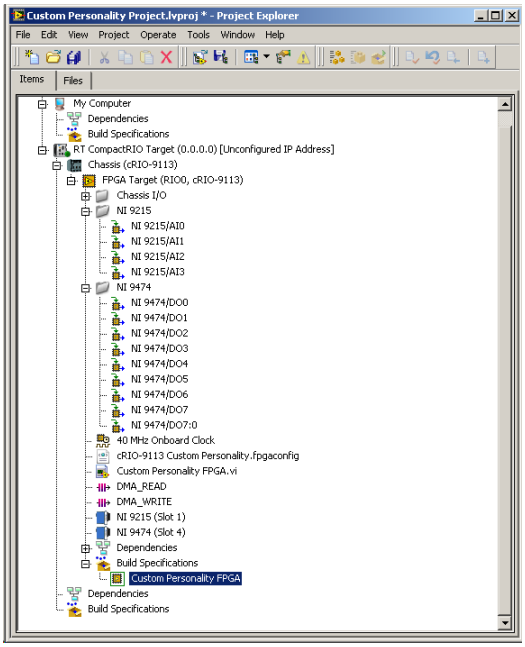


Figure A.21: Create Labview FPGA target and XML - 13

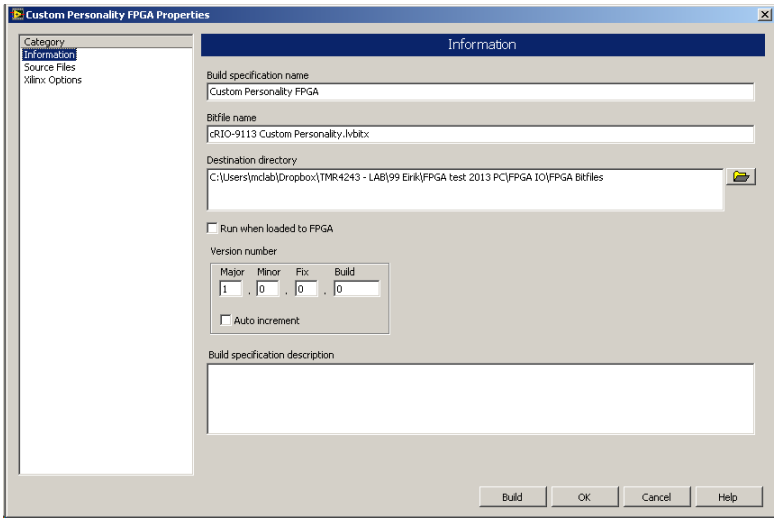


Figure A.22: Create Labview FPGA target and XML - 14

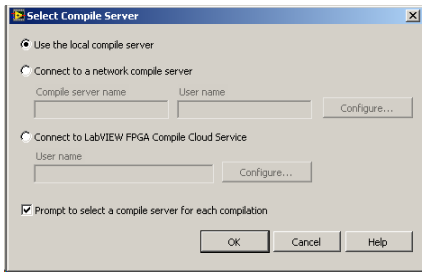


Figure A.23: Create Labview FPGA target and XML - 15

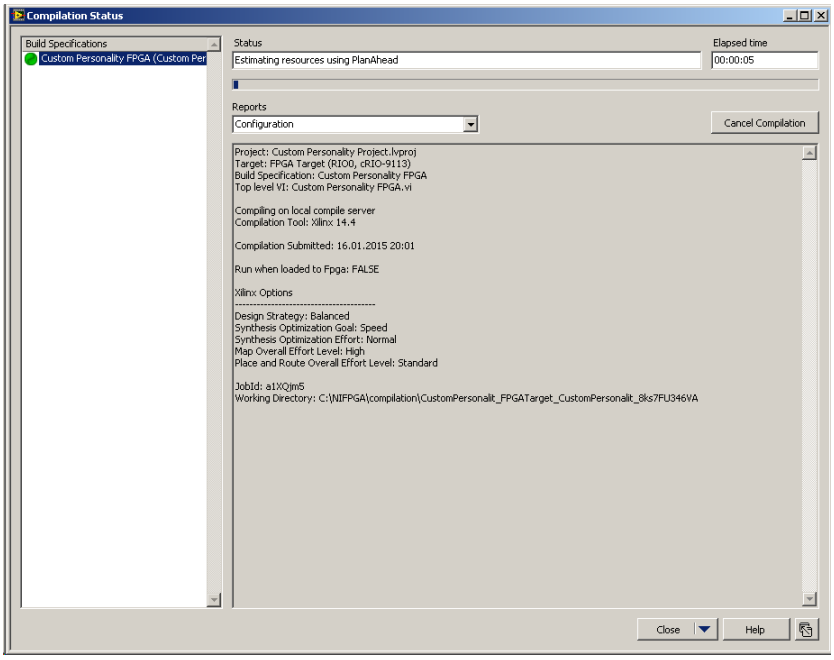


Figure A.24: Create Labview FPGA target and XML - 16

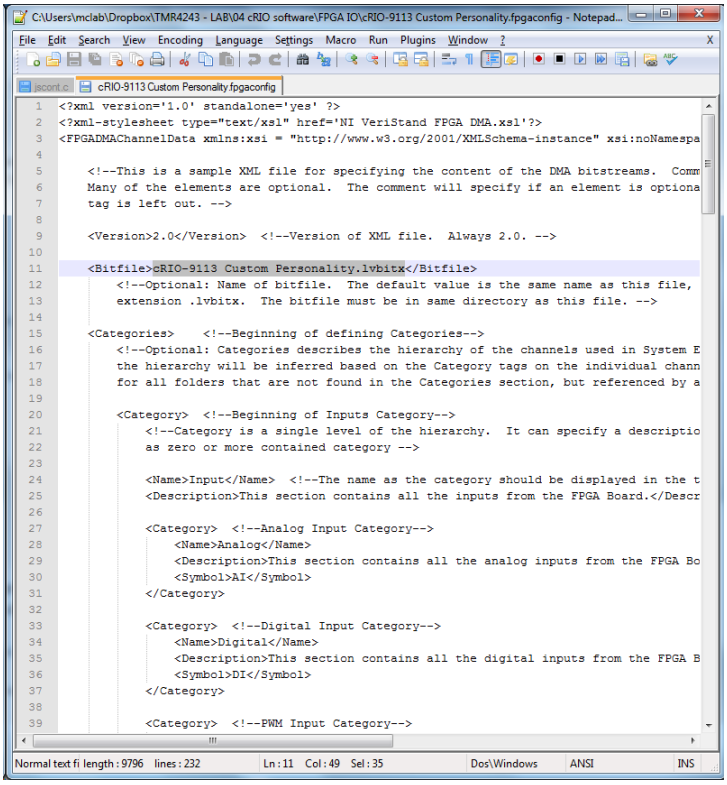


Figure A.25: Create Labview FPGA target and XML - 17

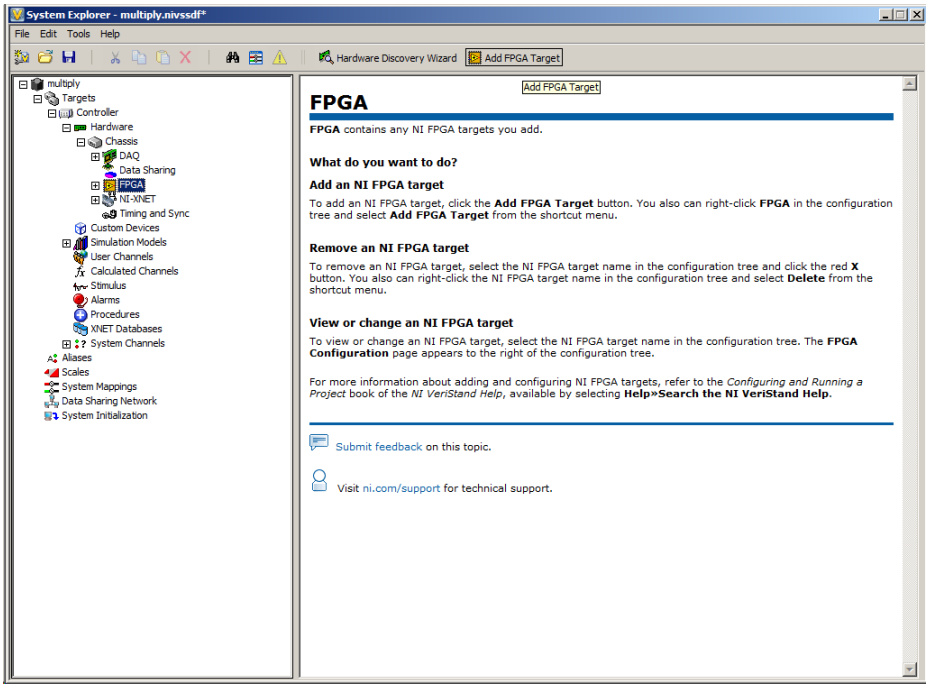


Figure A.26: FPGA1

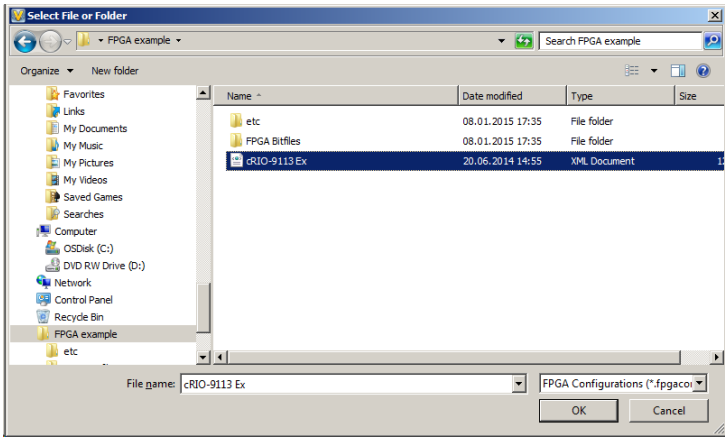


Figure A.27: FPGA2

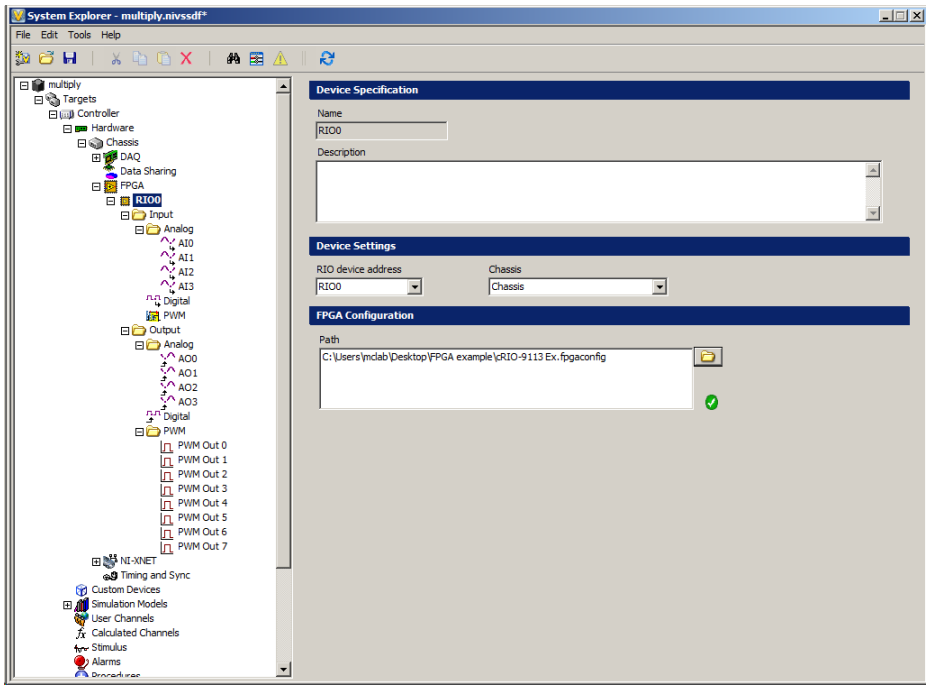


Figure A.28: FPGA3

A.1.4 Installing custom device driver

Create Torgeir Wahl has built the custom device driver for taking Oqus and PS3 controller input to Veristand

Install In order to use a RPi to send joystick commands to the cRIO it is necessary to build a custom device driver. In our case Torgeir Wahl has built a driver, and this guide will show how to install the driver.

The first step is to copy the whole directory (folder named WL_Joystick) of the custom device driver into the correct directory on your computer:

C:\Users\Public\Documents\National Instruments\NI VeriStand 2014\Custom Devices

The directory should now contain something like Figure A.29.

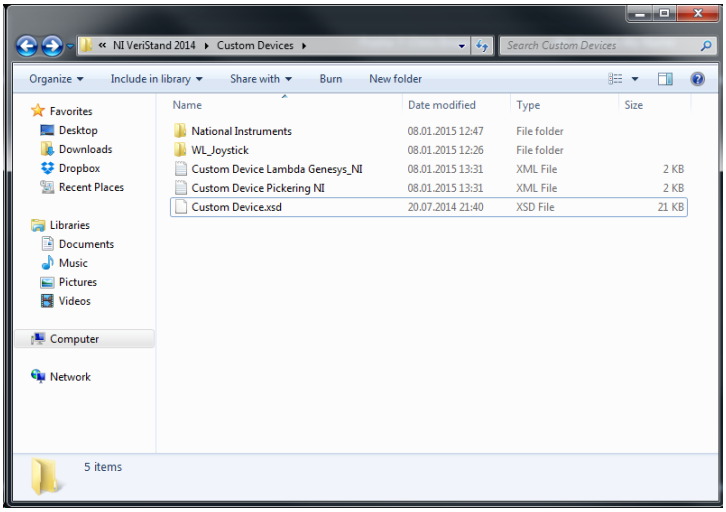


Figure A.29: Custom device folder

The next step is to add custom device to your project. This is done in the system explorer, which is found as seen in Figure A.30.

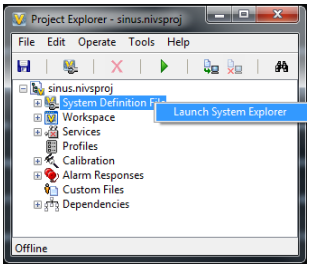


Figure A.30: VeriStand launch system explorer

When in the system explorer, adding the custom device should be as simple as right clicking the custom device pane and choosing WL_Joystick, as in Figure

A.31. If you do not find the custom device WL_Joystick, the most likely problem is that the placement of the custom device folder from step 1 is wrong.

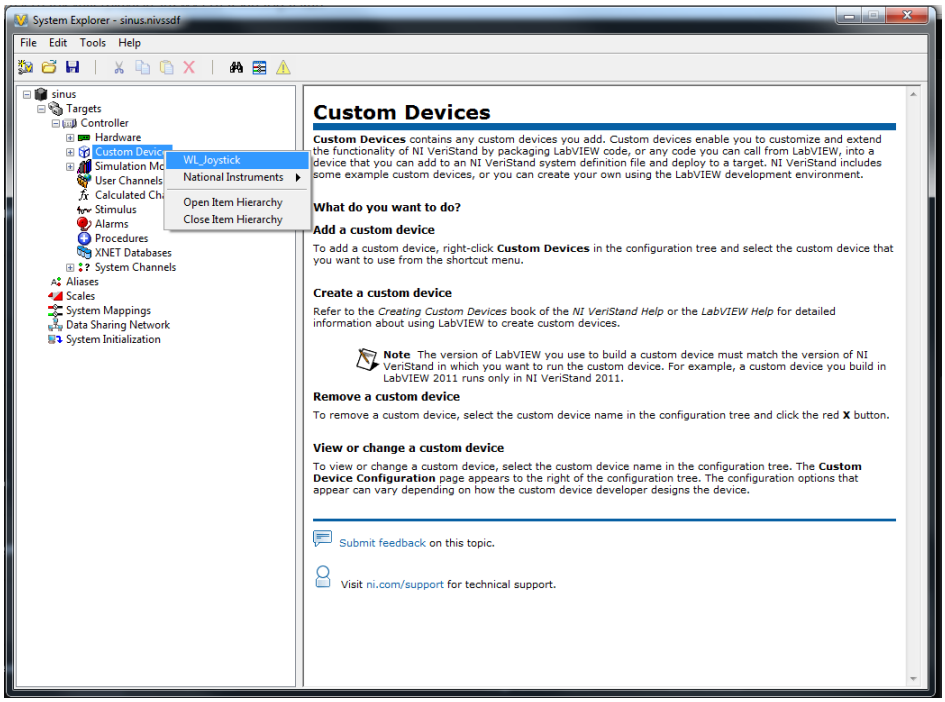


Figure A.31: Custom device selection

If the installation is successful you should be able to see WL_Joystick folder under custom devices as seen in the red box in Figure A.32. Here you will also see the different inputs from the custom device, in this case it is joystick axis.

To connect the joystick to the input ports of the Simulink model. You open the system configuration mappings (click the button marked by the arrow in Figure A.32).

You the simply find the ports you would like to connect, mark them and click the connect button. Figure A.33 a joystick output is connected to a input port on the Simulink model.

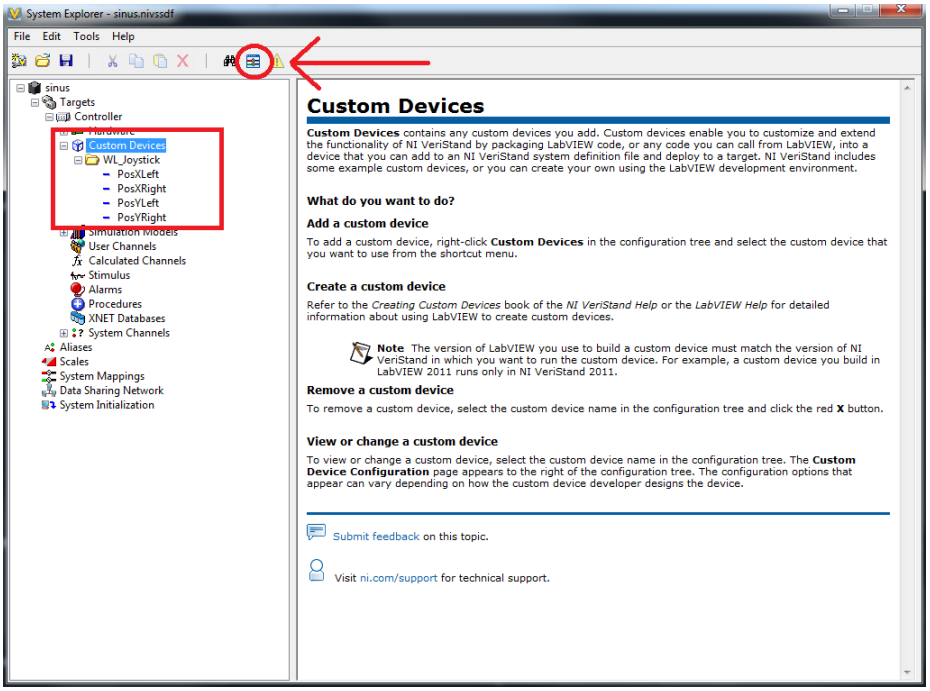


Figure A.32: VeriStand

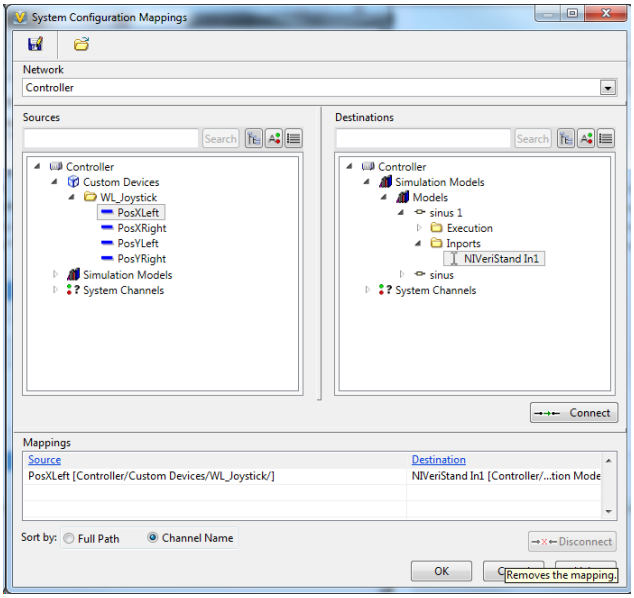


Figure A.33: VeriStand System Configuration Mappings

A.2 Raspberry Pi

the unit is configured with Raspbian Linux-kernel-based operating system

A.2.1 Raspbian installation and setup

This section describes how to install and access the Raspbian operating system on the RPi from a Windows computer. The operations are also possible from an OSX or Linux computer.

A.2.1.1 Download operating system and utilities

Download and extract the newest Raspbian[1] operating system (OS) image.

Necessary utilities for the setup are

- Win32 Disk Imager[2] to write the OS image to the RPi SD card
- Advanced IP scanner[3] to find the RPi address on the network
- Putty terminal emulator[4] for SSH connection
- WinSCP[5] for file transfer

Windows	Linux, OSX
Win32 Disk Imager	dd
Advanced IP scanner	nmap
Putty	ssh
WinSCP	sftp

Table A.1: RPi installation and setup utilities

See Table **??** for a list of the equivalent software for OSX and Linux.

A.2.1.2 Write image to SD card

Since the .iso file is raw, it needs to be written to the SD card in way that makes it bootable. Win32 Disk Imager does this.

Run the program as administrator. Select the correct image file and device, as in Figure A.34. Make sure that you have slected the correct drive before you push Write.

Once the write is complete, insert the SD card in the RPi and boot.

[1] raspberrypi.org/downloads
[2] sourceforge.net/projects/win32diskimager
[3] by Famatech, advanced-ip-scanner.com
[4] www.chiark.greenend.org.uk/~sgtatham/putty/download.html
[5] by Martin Prikryl, winscp.net/eng/download.php

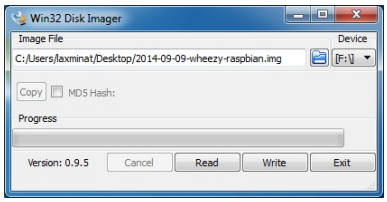


Figure A.34: Disk Imager

A.2.1.3 Terminal access

RPi can be accessed through the network, i.e. without having to directly connect a monitor and keyboard.

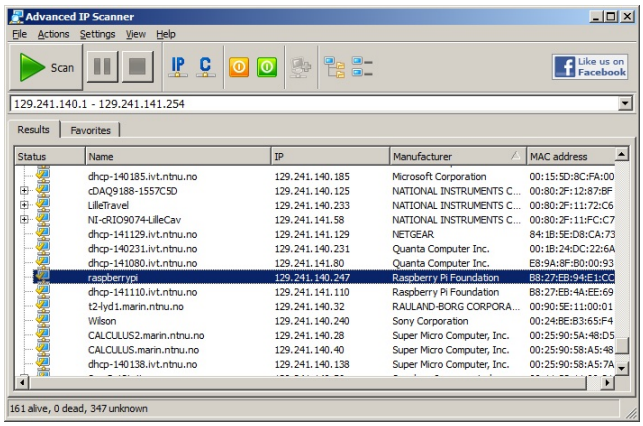


Figure A.35: Advanced IP Scanner

At first boot, the RPi by default waits to be assigned an IP address by DHCP. If this address is not known, scan the network with Advanced IP Scanner. It is advicible to sort the results by manufacturer since it is fixed (*Raspberry Pi Foundation*). The name is typically *raspberrypi*. See Figure A.35.

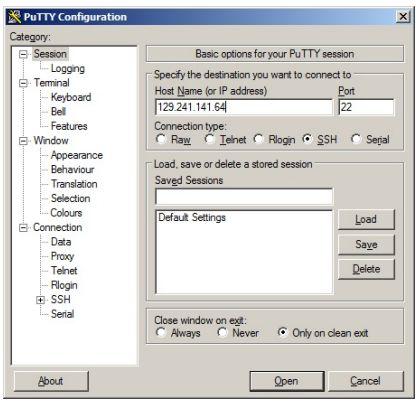


Figure A.36: Putty settings

Once the IP is known, it is specified in the Putty settings, as in Figure A.36, and a connection can be opened.

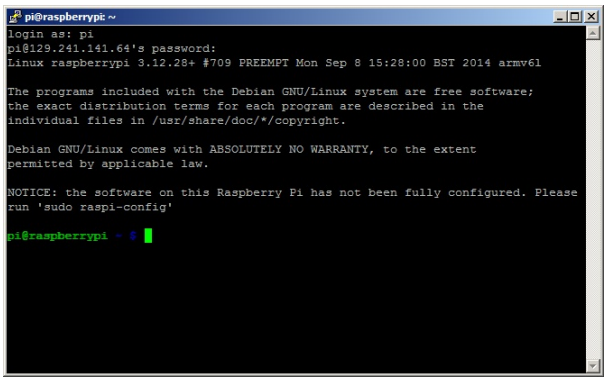


Figure A.37: SSH connection

The default login is `pi`, and the default password `raspberry`. Figure A.37 shows the terminal output on first login.

A.2.1.4 Finalize configuration

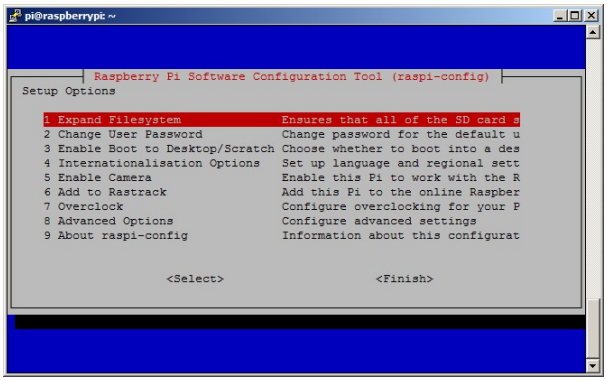


Figure A.38: RPi configuration tool

Enter the

```
sudo raspi-config
```

command to start the RPi Software Configuration Tool, as in Figure A.38. Use the menu to apply the following

1. Update configuration tool: 8 Advanced Options >A9 Update
2. Change password: 2 Change User Password
3. Expand filesystem: 1 Expand Filesystem >Finish

Exit the configuration tool and select Yes for reboot. Reconnect through Putty.

Finally, update the repository package lists and upgrade all packages currently installed on the RPi:

```
sudo apt-get update
sudo apt-get upgrade -y
```

This process took approximately 10 minutes on a 90 Mbps internet connection.

A.2.1.5 Transfer files to RPi from computer

WinSCP can be used to transfer files to the RPi. This is useful for instance when transferring code, or when the RPi is not directly connected to the internet.

A.2.1.6 Set fixed IP address

When the RPi is connected directly to the cRIO or computer, a fixed IP is necessary since there is no DHCP server in that network. During most of this setup, however, it is preferable to keep the default DHCP assigned IP setting.

To set a fixed IP

1. Open the network interface configuration information file for editing

   ```
   sudo nano /etc/network/interfaces
   ```

2. Alter the eth0 settings from `dhcp` to `static` and add address and netmask as

   ```
   auto eth0
   iface eth0 inet static
      address 192.168.1.22
      netmask 255.255.255.0
   ```

3. Save the changes by the key combination CTRL+X.

The new IP is applied on the next reboot.

A.2.2 Sixaxis installation and configuration

This section describes how to install and configure the Sixaxis gamepad for Bluetooth connection to the RPi, and how to add a server for sending joystick signals to the cRIO.

A.2.2.1 Download and install bluetooth support

BlueZ is the official Linux Bluetooth stack. It provides support for core Bluetooth layers and protocols.

To download and install, type

```
sudo apt-get install bluez-utils bluez-compat bluez-hcidump
   libusb-dev libbluetooth-dev joystick checkinstall -y
```

The process takes a few minutes.

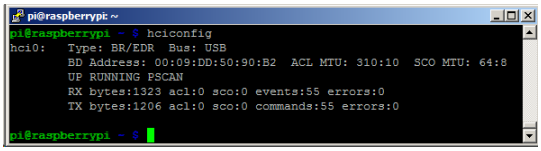


Figure A.39: Bluetooth configuration tool

To confirm the installation, use the `hciconfig` command to print name and basic information about Bluetooth devices installed in the system. The output should include `UP RUNNING PSCAN`, as in Figure A.39. If instead it says `DOWN`, some error har occured.

Most experienced errors were due to typos.

A.2.2.2 Bluetooth pairing

Sixaxis does not support the standard Bluetooth paring prcedure, instead, pairing is done over USB. The `sixpair` command-line utility[6] searches USB buses for Sixaxis devices and tells them to connect to a new Bluetooth master.

Download and compile the program by the following commands:

```
wget http://www.pabr.org/sixlinux/sixpair.c
gcc -o sixpair sixpair.c -lusb
```

Connect the Sixaxis by USB before running the paring utility

```
sudo ./sixpair
```

The output should be similar to

```
Current Bluetooth master: 00:02:72:BF:BC:8F
Setting master bd_addr to: 00:02:72:BF:BC:8F
```

[6] by Pabr Technologies, www.pabr.org

The addresses at the end of each line will only be the same if you have already paired the Sixaxis with the Bluetooth dongle. First time they will be different.

The Sixaxis USB cable may now be disconnected.

A.2.2.3 Joystick manager system service

`QtSixA`[7] reads the Sixaxis signals and makes them available to other programs. This program needs to run automatically whenever the RPi is booted.

To download the program, type

```
wget http://sourceforge.net/projects/qtsixa/files/QtSixA%201.5.1/QtSixA-1.5.1-src.tar.gz
```

To install, type

```
tar xfvz QtSixA-1.5.1-src.tar.gz
cd QtSixA-1.5.1/sixad
make
sudo mkdir -p /var/lib/sixad/profiles
sudo checkinstall -y
```

Update the system service list with sixad driver and reboot

```
sudo update-rc.d sixad defaults
sudo reboot
```

To test the program, turn on the Sixaxis (round PS button in the middle) and start the test program

```
sudo jstest /dev/input/js0
```

The terminal should now fill up with numbers that change as you move the analogue sticks and press the buttons on the Sixaxis. Exit the program by the key combination Ctrl+C.

A.2.2.4 Joystick signal server

A server must run to make joystick signals available over the RPi ethernet port. This should also start whenever the RPi is booted.

Transfer the source file `jscont.c` to the RPi (see Section A.2.1.5), then compile:

```
g++ -o jscont jscont.c
```

To verify that the program runs correctly, turn off (hold PS3 button for about 10 seconds) the previously paired Sixaxis and start the program

```
./jscont
```

The program should then wait until you turn on the Sixaxis before giving output simular to Figure A.40. To exit the server use the key combination Ctrl+C.

[7] the Sixaxis Joystick Manager by falkTX, qtsixa.sourceforge.net

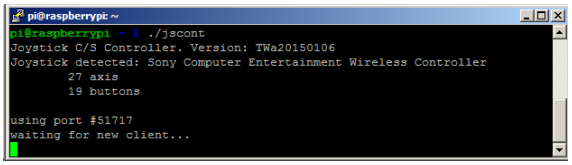


Figure A.40: Joystick signal server test

Next, disable login at start-up in the bootup service description `inittab`:

1. Open the file for editing

   `sudo nano /etc/inittab`

2. Change the line that reads

   `1:2345:respawn:/sbin/getty --noclear 38400 tty1`

   by adding `--autologin pi` to get

   `1:2345:respawn:/sbin/getty --autologin pi --noclear 38400 tty1`

   Warning: Typos here may result consequences hard to correct.

3. Save and exit the changes by the key combination Ctrl+X.

Finally, add `jscont` to the login execution file:

1. Open the file for editing

   `sudo nano /home/pi/.bashrc`

2. At the very end of the file, add

   `sudo ./jscont`

3. Save the changes by the key combination Ctrl+X.

RPi should now be sending joystick signals at start-up.

A.3 Laptop

Compatability between software is very important, See NI VeriStand Version Compatibility KnowlegeBase[8].

A.3.0.1 Order of installation

1. Microsoft .NET
2. Microsoft SDK
3. Matlab
4. Labview
5. Veristand
   (a) Including NI VeriStand Model Framework!
6. Additional for model compilation
   (a) VxWorks:
      i. WindRiver GNU Toolchain[9]
      ii. Real-Time Workshop software
   (b) PharLap:
      i. Microsoft Visual C++
      ii. The MathWorks, Inc. Real-Time Workshop® software

A.3.0.2 needed

- Matlab
- Labview
- LabVIEW development system
- LabVIEW Real-Time Module
- LabVIEW FPGA Module (recommended)
- NI-RIO driver
- VeriStand

[8] `http://digital.ni.com/public.nsf/allkb/2AE33E926BF2CDF2862579880079D751`
[9] `ftp://ftp.ni.com/pub/devzone/epd/gccdist_vxworks6.3_gcc3.4.4.zip`

Appendix B

Cybership Enterprise 1

Increased servo percentage results in clockwise? motion.

Hystersis on motors

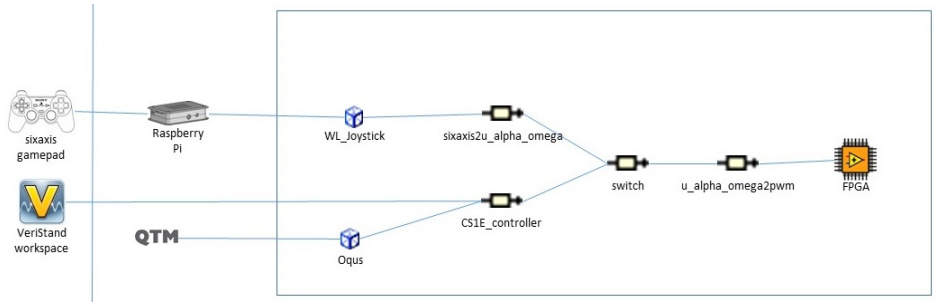


Figure B.1: CSE1 component communication

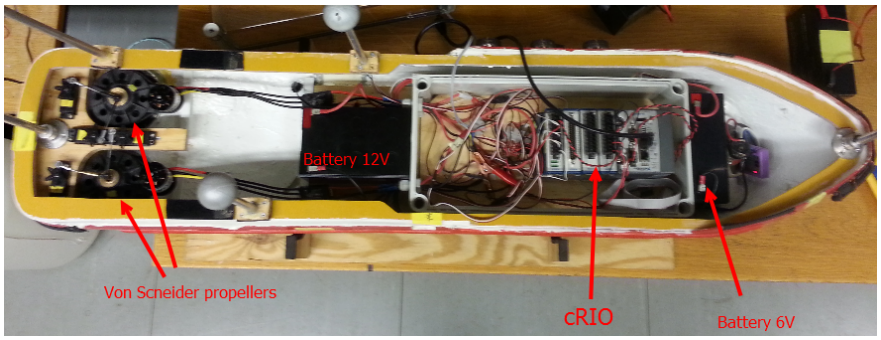
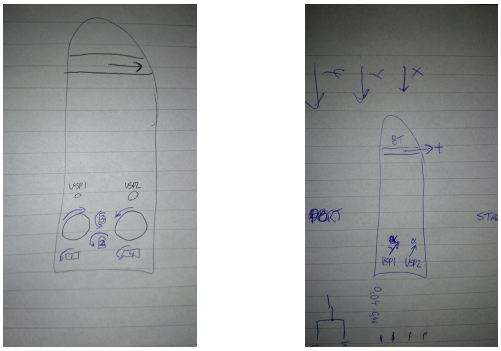


Figure B.2: CSE1 - Hardware

B.1 Actuators

Antall, posisjon, aktivert med pwm.

Figure B.3: CSE1 diagram



(a) Hardware input (b) Control input

Figure B.4: Thruster configuration

Port	Component
pwm0	Bow thruster motor
pwm1	VSP1 motor
pwm2	VSP2 motor
pwm3	not in use
pwm4	servo1
pwm5	servo2
pwm6	servo3
pwm7	servo4

Table B.1: CSE1 cRIO digital output

50Hz. Table **??**

Motors motor control, servos directly.

PWM signals are found experimentally. Remeasure to account for wear and tear and flexibility.

B.1.1 Motor control signals

B.1.2 Servo control signals

B.1.3 Measurements

Port	Component
AI0	6V Battery
AI1	Unknown
AI2	Unknown
AI3	12V Battery

Table B.2: CSE1 cRIO analog input

B.2 Control software

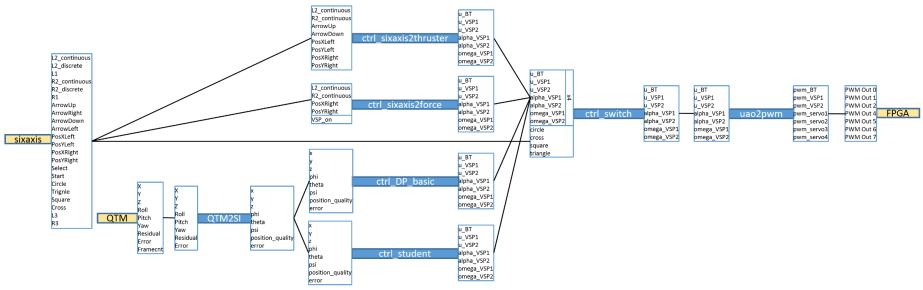


Figure B.5: CSE1 control software modules

B.2.1 sixaxis (currently named WL_joystick) custom device

Reads sixaxis gamepad input from the RPi server.

B.2.2 QTM (currently named Oqus) custom device

Reads pose (position and orientation) information broadcasted by Qualisys Track Manager. The data is in milimeters and degrees.

Additional outputs are

- a residual which is 0 for perfect measurement, increases with reduced quality of measurement and -1 for no measurement.
- an error code
- framecounter

B.2.3 QTM2SI

Converts QTM data to standard international (SI) units: position in meters and orientation in radians. The yaw angle is mapped to the interval $\psi \in [-\pi, \pi]$.

B.2.4 ctrl_sixaxis2thruster control module

Provides manual thruster control.

B.2.4.1 Voith Schneider Propellers

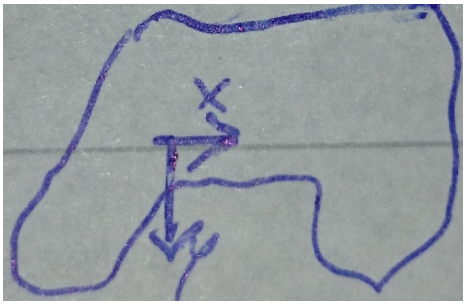


Figure B.6: Sixaxis coordinate system

The left and right joysticks, repectively, give the VSP deflections, u_{VSP1} and u_{VSP2}, and angles, α_{VSP1} and α_{VSP2}, depicted in Figure B.4b. The joystick coordinates PosX and PosY axes point right and down, as seen in Figure B.6. The deflection is

$$u_{\mathrm{VSP}i} = \min\left(\sqrt{(\mathrm{PosX})^2 + (\mathrm{PosY})^2}, 1\right).$$

The $\min(\cdot)$ ensures contraining $u_{\mathrm{VSP}i} \in [0, 1]$. The angle is

$$\alpha_{\mathrm{VSP}i} = \arctan 2\left(\mathrm{PosX}, -\mathrm{PosY}\right).$$

The VSP rotational speeds, ω_{VSP1} and ω_{VSP2}, are set in ± 0.1 increments by use of the directional pad up and down buttons.

B.2.4.2 Bow thruster

BT is controlled by L2 and R2. Both buttons output -1 when released and increasing to 1 when fully pushed. The thruster input

$$u_{\mathrm{BT}} = -\frac{L2 - R1}{2}$$

maps to the interval $u_{\mathrm{BT}} \in [-1, 1]$ with positive direction according to Figure B.4b.

B.2.5 ctrl_sixaxis2force control module

Provides manual forces and moment control. Surge and sway forces, X and Y, are given by the left joystick. Yaw moment N is given by the L2 and R2 buttons.

Thrust allocation is based on the configation shown in Figure B.4b. The thrust is thus

$$\tau = T(\alpha) K u$$

where

$$\tau = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} \text{ are the forces and moment,}$$

$$T(\alpha) = \begin{bmatrix} \cos(\alpha_{\mathrm{VSP1}}) & \cos(\alpha_{\mathrm{VSP2}}) & 0 \\ \sin(\alpha_{\mathrm{VSP1}}) & \sin(\alpha_{\mathrm{VSP2}}) & 1 \\ l_{x,\mathrm{VSP1}}\cos(\alpha_{\mathrm{VSP1}}) - l_{y,\mathrm{VSP1}}\sin(\alpha_{\mathrm{VSP1}}) & l_{x,\mathrm{VSP2}}\cos(\alpha_{\mathrm{VSP2}}) - l_{y,\mathrm{VSP2}}\sin(\alpha_{\mathrm{VSP2}}) & l_{\mathrm{BT}} \end{bmatrix}$$

is the configuration matrix,

$$\alpha = \begin{bmatrix} \alpha_{\mathrm{VSP1}} \\ \alpha_{\mathrm{VSP2}} \end{bmatrix} \text{ are the thruster angles,}$$

$$K = \begin{bmatrix} K_{\mathrm{VSP1}} & 0 & 0 \\ 0 & K_{\mathrm{VSP2}} & 0 \\ 0 & 0 & K_{\mathrm{BT}} \end{bmatrix} \text{ is the force coefficient matrix, and}$$

$$u = \begin{bmatrix} u_{\mathrm{VSP1}} \\ u_{\mathrm{VSP2}} \\ u_{\mathrm{BT}} \end{bmatrix} \text{ are the control forces.}$$

Since solving the thrust equation for u and α is complicated, a virtual is azimuthing thruster VSP representing the joint forces from VSP1 and VSP2 is considered instead. It is further assumed that $\alpha_{\mathrm{VSP1}} = \alpha_{\mathrm{VSP2}}$, $K_{\mathrm{VSP1}} = K_{\mathrm{VSP2}}$ and $u_{\mathrm{VSP1}} = u_{\mathrm{VSP2}}$. Considering an extended control force vector

$$u_e = \begin{bmatrix} u_{\mathrm{VSP},x} \\ u_{\mathrm{VSP},y} \\ u_{\mathrm{BT}} \end{bmatrix},$$

where the VSP control forces are decomposed, yields

$$\underbrace{\begin{bmatrix} X \\ Y \\ N \end{bmatrix}}_{\tau_e} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & l_{x,\mathrm{VSP}} & l_{\mathrm{BT}} \end{bmatrix}}_{T_e} \underbrace{\begin{bmatrix} K_{\mathrm{max,VSP}} & 0 & 0 \\ 0 & K_{\mathrm{max,VSP}} & 0 \\ 0 & 0 & K_{\mathrm{max,BT}} \end{bmatrix}}_{K_e} u_e.$$

This is solved for u_e by simple inversion. Finally, the actual control forces are

$$\begin{aligned} u_{\mathrm{VSP1}} = u_{\mathrm{VSP2}} &= \sqrt{(u_{\mathrm{VSP},x})^2 + (u_{\mathrm{VSP},y})^2}, \text{ and} \\ \alpha_{\mathrm{VSP1}} = \alpha_{\mathrm{VSP2}} &= \arctan2(u_{\mathrm{VSP},y}, u_{\mathrm{VSP},x}). \end{aligned}$$

B.2.6 ctrl_DP_basic control module

Provides basic dynamic positioning capability.

B.2.7 ctrl_student control module

-

min	control input	max
$-1 \leq$	u_BT	≤ 1
$0 \leq$	u_VSP1	≤ 1
$0 \leq$	u_VSP2	≤ 1
$-\pi \leq$	alpha_VSP1	$\leq \pi$
$-\pi \leq$	alpha_VSP2	$\leq \pi$
$-1 \leq$	omega_VSP1	≤ 1
$-1 \leq$	omega_VSP2	≤ 1

Table B.3: Control input ranges

B.2.8 switch module

Selects one out of four control modules

- ctrl_sixaxis2thruster when △ is pushed
- ctrl_sixaxis2force when □ is pushed
- ctrl_DP_basic when ○ is pushed
- ctrl_student when × is pushed

The module also saturates the control signals according to Table B.3 and issues a warning signal if the current controller exceedes the bounds.

B.2.9 uao2pwm module

Converts the unitized controller inputs to signals suitable for pwm output to the ESC.

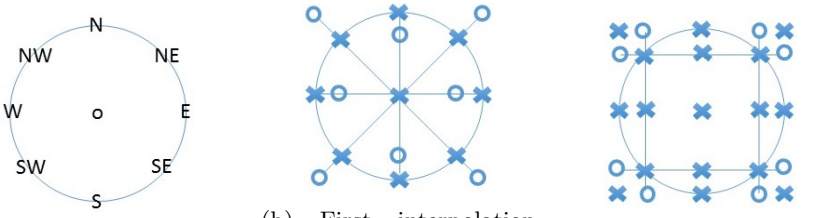
The position of the VSP steering rods are controlled by a pair of servos for each.

Position	VSP1		VSP2	
	servo1 [%]	servo2 [%]	servo3 [%]	servo4 [%]
N	4.25	5.20	4.95	3.85
NE	4.30	4.50	5.60	3.90
E	4.90	4.05	5.89	4.38
SE	5.40	4.10	5.60	5.00
S	5.99	4.70	4.95	5.50
SW	5.75	5.50	4.35	5.40
W	5.25	5.75	4.15	4.85
NW	4.60	5.65	4.20	4.30
Origo	4.90	4.82	4.83	4.52

Table B.4: Servo pwm ranges

Ikke lineært, ikke rett frem.

Foreslått metode



(a) Measurments (b) First interpolation and extrapolation (c) Second extrapolation

Figure B.7: Servo, rod position tuning

B.2.10 FPGA interface

Outputs pwm signals through the digital output modules.

uao2pwm	**FPGA**
$\mathbf{pwm_{BT}}$	**pwm0**
$\mathbf{pwm_{VSP1}}$	**pwm1**
$\mathbf{pwm_{VSP2}}$	**pwm2**
$\mathbf{pwm_{servo1}}$	**pwm4**
$\mathbf{pwm_{servo2}}$	**pwm5**
$\mathbf{pwm_{servo3}}$	**pwm6**
$\mathbf{pwm_{servo4}}$	**pwm7**

Table B.5: PWM connections

B.3 Qualisys body

calibration

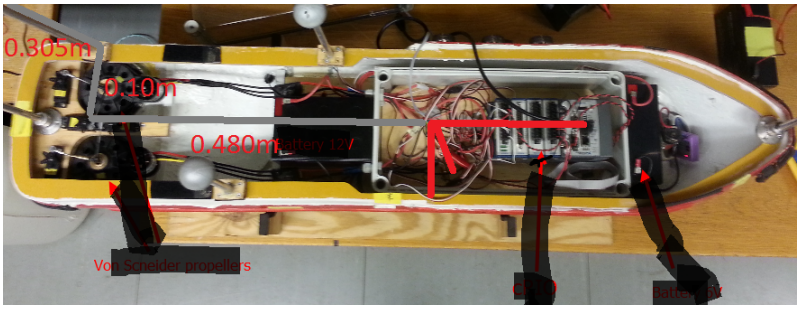


Figure B.8: Matlab console

Appendix C

Qualisys motion capture system

C.1 Start server

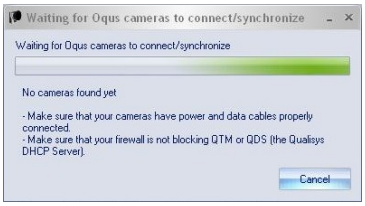


Figure C.1: Matlab console

C.2 Aquire body

leverer med 50Hz på nettet

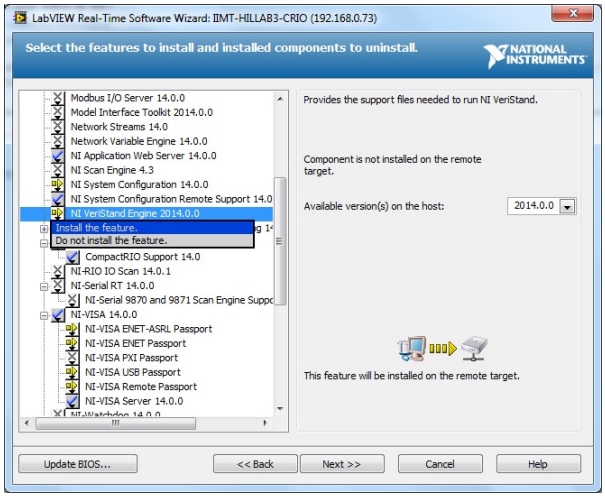


Figure C.2: FPGA2

Part VI

Miscellaneous

Appendix D

HIL lab and MC lab device network addresses

RPi	192.168.1.22	for all
cRIO secondary ethernet	192.168.1.21	for all
cRIO primary ethernet	192.168.0.71	iimt-HILlab1-cRIO
	192.168.0.72	iimt-HILlab2-cRIO
	192.168.0.73	iimt-HILlab3-cRIO
	192.168.0.76	CSE1
Computer	192.168.0.10	Qualisys PC
	192.168.0.41	iimt-HILlab1-PC
	192.168.0.42	iimt-HILlab2-PC
	192.168.0.43	iimt-HILlab3-PC
	192.168.0.47	MClab
Subnet mask	255.255.255.0	for all

Table D.1: IP addresses

All RPis and have the same IP address, but there is no IP conflict since the cRIO-RPi networks are separate and closed. The same goes for the cRIO secondary ethernet ports

Note: to connect the RPi directly to the computer, both need to be on the same domain and the computer IP thus needs to change to 192.168.**1**.xx.

Appendix E

Checklist

Device driver in place

Custom indicators in place

PC and cRIO IP correct

Sixaxis charged

Appendix F

Personel and literature

F.1 Points of contact

Håkon Nødset Skåtun Hakon.Nodset.Skatun@km.kongsberg.com, built CSE1

Øivind Kåre Kjerstad Built CSE1.

Torgeir Wahl Custom devices (Qualisys client, Sixaxis client), Sixaxis RPi server

Dinh Nam Tran oppryddingsarbeid

Andreas Orsten brukt mye, skrevet artikel om sleping av isberg

Robert Kanajus rkajanus@gmail.com brukt HIL-lab og Minerva

Eirik Valle Teaching assistant TMR4243, Sixaxis for RPi setup

Andreas Reason Dahl andreas.r.dahl@ntnu.no, Laboratory assistant TMR4243

Jostein Follestad Teaching assistant TMR4243. Comprehensive CSE1 identification, including towing. CS1E HIL model.

Fredrik Sandved Teaching assistant TMR4243, Custom displays

Tor Kvestad Idland Built CSS spring 2015.

Trond Innset Trond.Innset@marintek.sintef.no, cut out the CSS hull foam

F.2 Publications

2014

Andreas Orsten, Petter Norgren, Roger Skjetne, LOS guidance for towing an iceberg along a straight-line path. Proceedings of the 22nd IAHR International Symposium on ICE 2014 (IAHR-ICE 2014).

F.3 Specialization projects and master theses

2011

Håkon Nødset Skåtun Development of a DP system for CS Enterprise I with Voith Schneider thrusters. Master thesis.

2013

Nam Dinh Tran Development of a modularized control architecture for CS Enterprise I for path-following based on LOS and maneuvering theory. Specialization project.

2014

Andreas Orsten Automatic Reliability-based Control of Iceberg Towing in Open Waters. Master thesis and poster.

Nam Dinh Tran Line-Of-Sight-based maneuvering control design, implementation, and experimental testing for the model ship C/S Enterprise I. Master thesis.

F.4 Other

Håkon Nødset Skåtun `http://www.youtube.com/watch?v=MiESJsIZO04`

Appendix G

Maintenance

Oil VSP

Appendix H

Suppliers

Laptops	Dell
cRIO	National Instruments
VSP	Thrusters were ordere at `www.cornwallmodelboats.co.uk/acatalog/voith_schottel.html`. Per 2014, availability is variable.

Table H.1: Suppliers

Appendix I

To do list

- Etablere fargekoder for simulinkblokker (spesielt “ikke røre”-farge)
- Konsekvent notasjon: CS Enterprise 1 eventuelt CSE1
- Troubleshooting-prosedyrer for de vanligste feilene
- Implementere “fail to zero” for når kommunikasjonen avbrytes.
- legge til IMU/gyro på båten