Marine cybernetics laboratory handbook

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 cybernetics laboratory. The laboratory is 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 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 III.

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

Introduction

Chapter 1

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 1.1, has dimensions 40m x 6.45m x 1.5m (LxBxD).

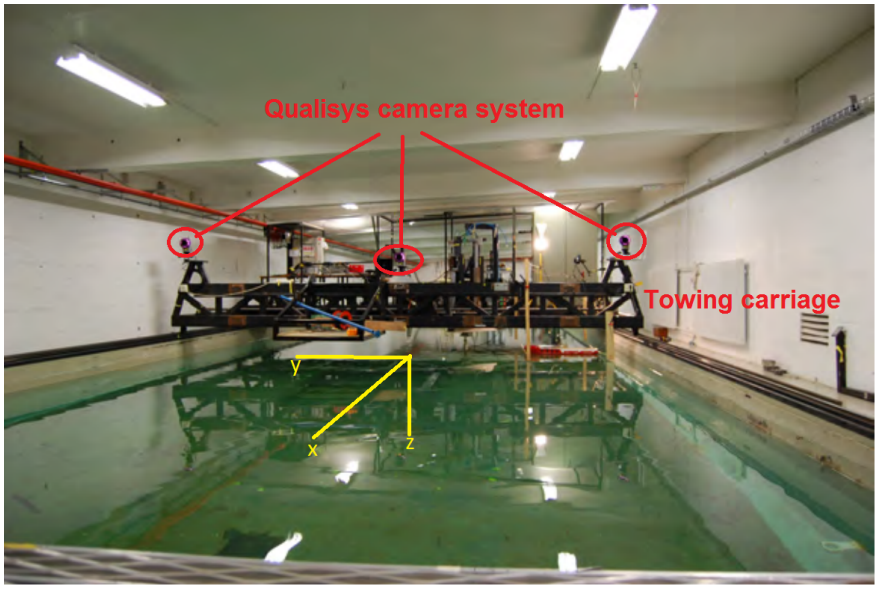


Figure 1.1: Marine cybernetics laboratory basin

	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 1.1: Wave generator capacity

1.1 Fixed Equipment

1.1.1 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.

1.1.2 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.

1.1.3 Wave generator

The single paddle wave generator is controlled by a dedicated computer. Available spectrum are first order Stoke, JONSWAP, Pierson-Moskowitz, Bretschneider, ISSC and ITTC. Table 1.1 summarizes the generation capacity.

1.2 Vessels

Both suface and underwater vessels.

The Cyber Ship class, with ship prefix CS consists of ....

1.2.1 CS Inocean Cat I Arctic Drillship

Figure 1.2: CS Enterprise I

1.2.2 CS Enterprise I

The CSE1, depicted in Figure 1.2, hva bruker vi den til?

1.2.2.1 Hull

tug boat model.

1.2.2.2 Actuators

The ship is fitted with two Voith Schneider propellers (VSP) astern and a bow thruster (BT).

1.2.2.3 Control system

The on-board control system consists of

- a National Instruments compact reconfigurable input/output (cRIO) embedded controller,
- a Raspberry Pi (RPi) single-board computerand,
- three electronic speed controllers (ESC), and
- four servos.

The operator interface is by laptop and through a Sony Sixaxis wireless gamepad for PlayStation 3.

High-level communication Following Figure 1.3 from left to right:

Sixaxis transmits its information to the RPi Bluetooth USB dongle with which it is previously paired[1].

[1] One-time pairing procedure described in Appendix 12.2.2.2.

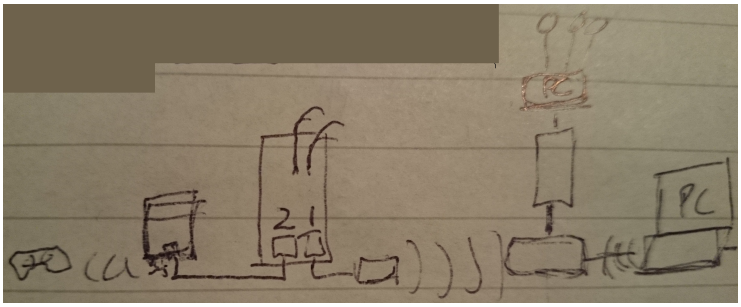


Figure 1.3: d

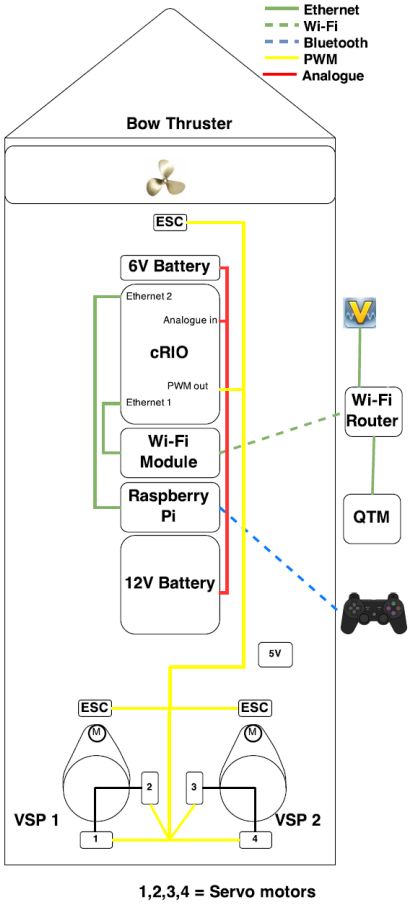


Figure 1.4: d

RPi receives Sixaxis data through the USB dongle and forwards it through its TCP/IP[2] server over Ethernet to the cRIO.

cRIO reads QTM broadcast positioning data through the Wi-Fi bridge on Ethernet port 1, Sixaxis data on Ethernet port 2. Online 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.

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.

1.2.2.4 Power

hg

[2] All IP addresses are as given in Table A.1.

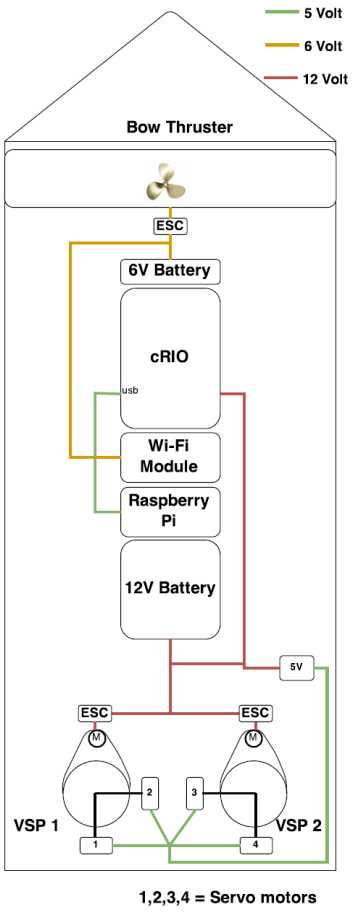

5 Volt
6 Volt
12 Volt
Bow Thruster
ESC
6V Battery
cRIO
usb
Wi-Fi Module
Raspberry Pi
12V Battery
5V
ESC
ESC
M
M
2
3
VSP 1
VSP 2
1
4
1,2,3,4 = Servo motors

Figure 1.5: d

1.2.3 CS Saucer

.

fdfd

fdfdfd

1.2.4 ROV Neptunus

.fdfd

fdfdfdfdfd

fdfdfd

Chapter 2

Control system development philosophy

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.

- Increased complexity marine vessels increases the need for testing and verification.

Verification and validation techniques applied throughout the development process enable you to find errors before they can derail your project. Most system design errors are introduced in the original specification, but aren't found until the test phase. When engineering teams use models to perform virtual testing early in the project, they eliminate problems and reduce development time by as much as 50%.

DOF	Controller	Plant	Iteration time	Flexibility
MIL	✓	Extremely short		
SIL	✓			
PIL			✓	
HIL				✓
Scale				✓
Full scale	✓	✓		Low

Table 2.1: Development steps overview

2.1 Development Steps

Most of these terms are used in the context of control systems development which means that typically you are building control software to interact with a mechatronic system. Typically, these are referred to as the "controller" and the "plant".

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.

2.1.1 Model-in-the-Loop

2.1.1.1 Principle

Model of the controller interconnected with a physical model of the plant, interconnected in a control development environment, such as MATLAB Simulink.

2.1.1.2 Aim

Develop control strategies

2.1.1.3 Iteration time

Extremely short, small changes are immediately implemented and tested.

2.1.1.4 Cost

Low

2.1.2 Software-in-the-Loop

Controller coded in final langugae, such as C or C++, and connected to plant model in a control development environment.

2.1.2.1 Aim

Test of coding system. Reveal coding failures.

2.1.2.2 Iteration time

Slightly longer than MIL.

2.1.3 Processor-in-the-Loop

2.1.3.1 Principle

Controller deployed to a representative microprocessor, connected to the plant simulation via high speed bus, such as JTAG. The plant must be synchronized with the controller.

2.1.3.2 Aim

expose problems with execution in the embedded environment. For instance, does your control loop fit within the execution time available on the embedded processor.

2.1.3.3 Iteration time

higher, regenerate and deploy code for each run

2.1.4 Hardware-in-the-Loop

2.1.4.1 Principle

Controller fully installed into the intended final hardware, connected through the plant only through the proper IO. The plant simulator must run on a real-time computer emulating the IO of a real process.

2.1.4.2 Aim

Perform regulation, security and failure tests without risk

Investigate the interaction between several ECUs and transmission control units (TCUs)

Ensure a high level of robustness and quality.

2.1.5 Scale test

2.1.6 Full scale

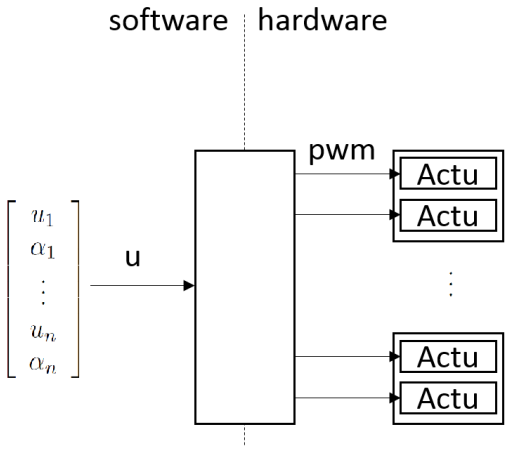


Figure 2.1: Individual actuator control

2.2 Control modes

Five control modes:

- Stop all actuators

1. Individual actuator control
2. Generalized force control
   (a) Body frame
   (b) Inertial frame
   (c) User frame
3. Regulation
   (a) Pose (3 DOF)
   (b) Heading
   (c) Depth
   (d) Roll/pitch
4. Automatic/operations

   Path following, guidance, etc

2.2.1 Individual actuator control

Inputs are

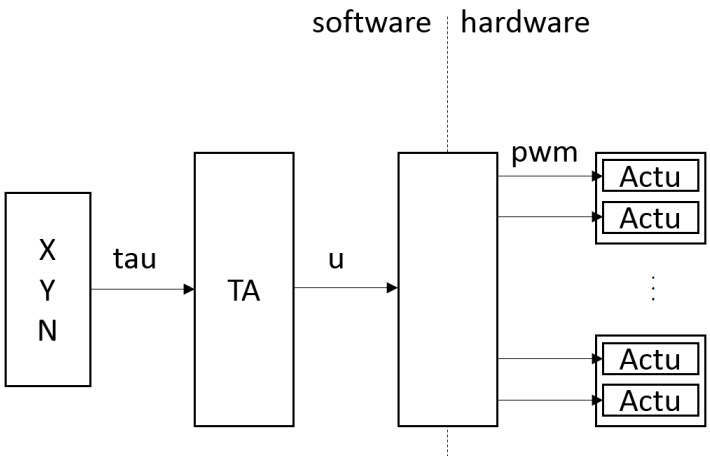


Figure 2.2: Generalized force control

2.2.2 Generalized force control

$$\tau = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix}$$

For the surface craft typically 3 DOF

$$\tau = \begin{bmatrix} X \\ Y \\ N \end{bmatrix}$$

TA

2.2.2.1 Body frame

2.2.2.2 Inertial frame

2.2.2.3 User frame

2.2.3 Regulation

DOF	Stationkeeping	Heading	Depth	Roll/Pitch
x	✓			
y	✓			
z			✓	
ϕ				✓
θ				✓
ψ	✓	✓		

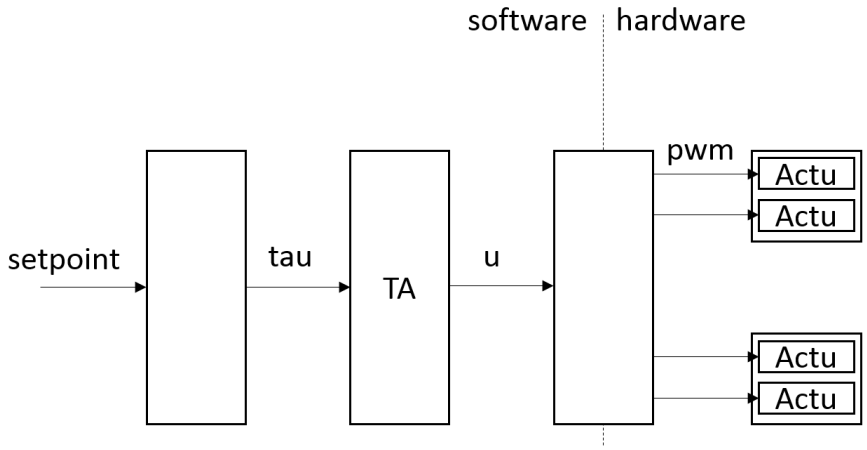


Figure 2.3: Generalized force control

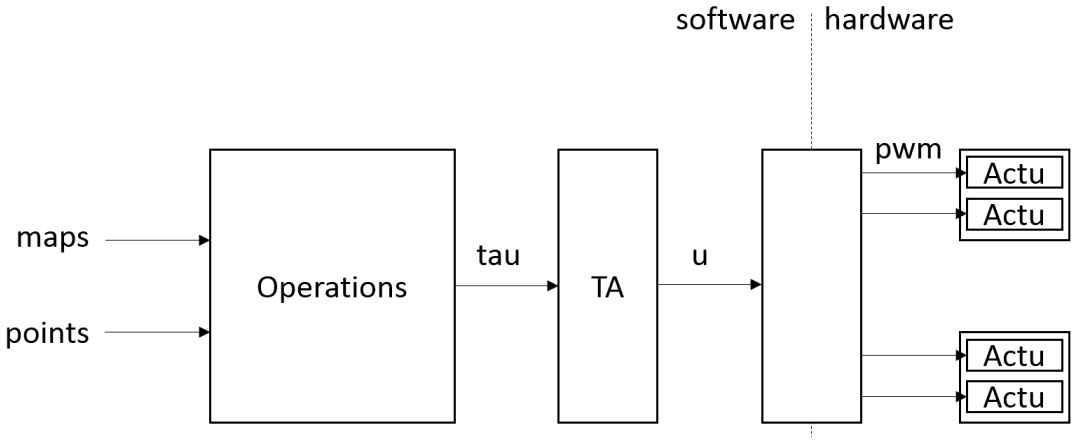


Figure 2.4: Generalized force control

2.2.4 Automatic/operations

Chapter 3

Real-time computing

3.0.1 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 user guide

Chapter 4

Qualisys motion capture system

Chapter 5

Towing carriage

In figure 5.1, a picture of the towing wagon is presented.

Control of the wagon is done on board; see the red circle in figure 5.1. For our purposes, to perform towing tests of CS Enterprise 1, only a certain computer commands are used. Further, in figure 5.2, a close up of the computer is shown. An explanation of the different commands, used for our purposes, is presented below.

5.1 Movement of towing wagon

The towing wagon can be moved approximately 20 meters in total. The end is located closest to the MC-laboratory.

- Move forwards: Press the button marked as move forwards (see figure 2).
- Move backwards: Press the button marked move backwards.
- Stop: If forwards, press backwards button.

If backwards, press forwards button. In both forwards and backwards direction, the velocity can be set between 0 m/s and 2 m/s. In addition, the acceleration can be changed between 0 and 0.5 m/s2. How aggressive the wagon is stopping can be regulated to a value between 0 % and 100 %, where 100 % is the most aggressive.

Moreover, it is possible to limit the distance covered by the wagon. Consequently, it can be stopped before it reaches the end of the basin. This is necessary for example when the wagon has a high velocity. The limit can be set between 0 mm and 20 000 mm.

Figure 5.1: Towing wagon in the MC-lab. Control of the wagon is done using the computer marked in red.

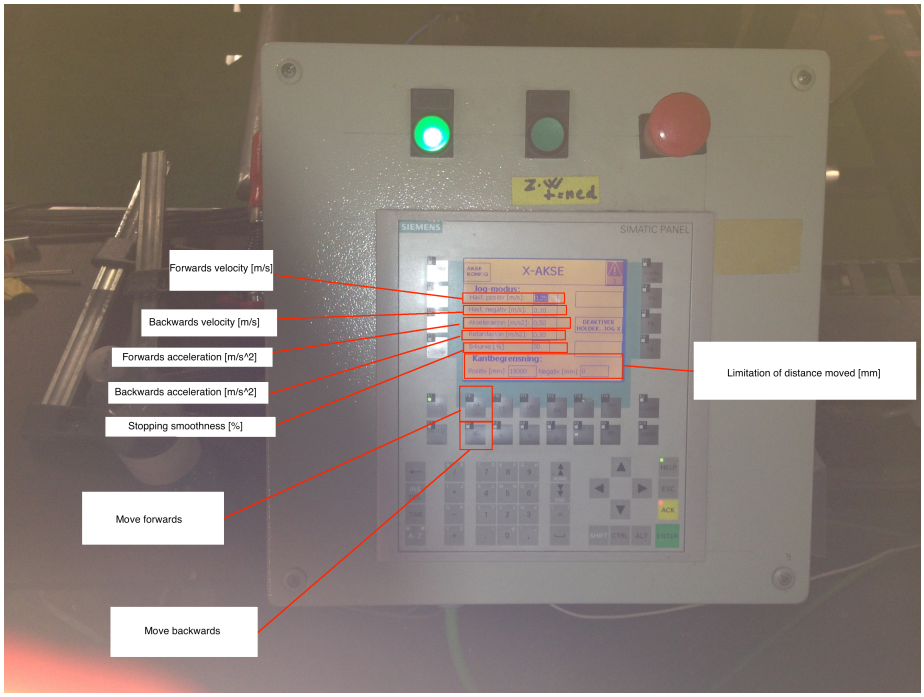


Figure 5.2: Computer to operate the towing wagon

5.2 Typical settings

For a typical towing run, the wagon has a velocity in forwards and backwards between 0 m/s and 1 m/s, depending on the towed object. The acceleration is set to 0.3 m/s2. The stopping smoothness is set to 30 %. Finally, the distance for the wagon to cover is between 0 mm and 19 000 mm.

5.3 Note

When the wagon is moving, no one is allowed to move on the sides of the basin.

Chapter 6

Wave generator

Chapter 7

CS Inocean Cat I Arctic Drillship

7.1 Mathematical model

Chapter 8

CS Enterprise I

8.1 Mathematical model

The proposed control design model is

$$\dot{\eta} = R(\psi)\,\nu \tag{8.1}$$
$$\dot{\nu} = M^{-1}\left(-C(\nu)\,\nu - D(\nu)\,\nu + \tau\right) \tag{8.2}$$

where

- the pose and velocity vectors are

$$\eta = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \in \mathbb{R}^3, \text{ and } \nu = \begin{bmatrix} u \\ v \\ r \end{bmatrix} \in \mathbb{R}^3,$$

  respectively. (x, y) is the position and ψ the yaw angle or heading in the marine cybernetics (MC) basin frame. (u, v) are the surge and sway velocities in the CSE1 vessel frame, and r is the yaw rate.

- the thrust force and moment vector is

$$\tau = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} \in \mathbb{R}^3,$$

  where (X, Y) is the surge and sway force vector, and N is the yaw moment.

- the three degrees of freedom (3 DOF) rotation matrix is

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

- the vessel inertia matrix is

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix} = M^\top > 0.$$

Rigid body		Added mass	
Parameter	Value	Parameter	Value
m	14.79	$X_{\dot{u}}$	-2
I_z	1.76	$Y_{\dot{v}}$	-10
x_g	0.0375	$Y_{\dot{r}}$	-0
y_g	0.0	$N_{\dot{r}}$	-1

Table 8.1: CSE1 rigid body and added mass parameters

Hydro surge		Hydro sway		Hydro yaw	
Parameter	Value	Parameter	Value	Parameter	Value
X_u	-0.4543	Y_v	-3.083	N_v	$4.1117 \cdot 10^{-4}$
		Y_r	-7.25	N_r	-1.9

Table 8.2: CSE1 damping parameters

- the coriolis and centripetal matrix is

$$C(\nu) = \begin{bmatrix} 0 & -mr & Y_{\dot{v}} v + (Y_{\dot{r}} - m x_g) r \\ mr & 0 & -X_{\dot{u}} u \\ -Y_{\dot{v}} v - (Y_{\dot{r}} - m x_g) r & X_{\dot{u}} u & 0 \end{bmatrix} = -C^{\top}(\nu).$$

- the damping matrix is

$$D(\nu) = \begin{bmatrix} d_{11}(u) & 0 & 0 \\ 0 & d_{22}(v,r) & d_{23}(v,r) \\ 0 & d_{32}(v,r) & d_{33}(v,r) \end{bmatrix},$$

where the damping components are

$$\begin{aligned} d_{11}(u) &= -X_u - X_{|u|u}|u| - X_{uuu}u^2 && (8.3) \\ d_{22}(v,r) &= -Y_v - Y_{|v|v}|v| - Y_{vvv}v^2 - Y_{|r|v}|r| && (8.4) \\ d_{23}(v,r) &= -Y_r - Y_{|v|r}|v| - Y_{|r|r}|r| - Y_{rrr}r^2 && (8.5) \\ d_{32}(v,r) &= -N_v - N_{|v|v}|v| - N_{vvv}v^2 - N_{|r|v}|r| && (8.6) \\ d_{33}(v,r) &= -N_r - N_{|v|r}|v| - N_{|r|r}|r| - N_{rrr}r^2 && (8.7) \end{aligned}$$

The rigid body inertia and hydrodynamic added mass parameters are given in Table 8.1.

The hydrodynamic damping parameters are given in Tables 8.2 and 8.3.

The model is valid for low-speed.

Hydro surge		Hydro sway		Hydro yaw	
Parameter	Value	Parameter	Value	Parameter	Value
X_u	-0.6555	Y_v	-1.33	N_v	0.0
X_{uu}	0.3545	Y_{vv}	-2.776	N_{vv}	-0.2088
X_{uuu}	-3.787	Y_{vvv}	-64.91	N_{vvv}	0.0
X_v	0.0	Y_r	-7.25	N_r	-1.9
X_{vv}	-2.443	Y_{rr}	-3.45	N_{rr}	-0.75
X_{vvv}	0.0	Y_{rrr}	0.0	N_{rrr}	0.0
.	.	Y_{rv}	-0.805	N_{rv}	0.130
.	.	Y_{vr}	-0.845	N_{vr}	0.080

Table 8.3: CSE1 damping parameters

8.2 Model-in-the-loop simulation

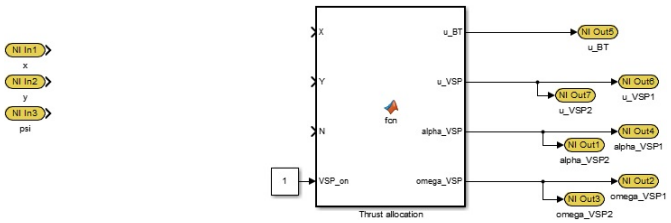


Figure 8.1: CSE1 `ctrl_student.slx` including thrust allocation

8.3 Processor-in-the-Loop simulation

Implementation of the student controller is done in the `ctrl_student.slx` Simulink template, depicted in Figure 8.1. Detailed implementation steps are given in Section 8.4.2.

The generic control system consists of several Simulink modules, the details of which are given in Appendix 13.2, a FPGA driver, described in Appendix 12.1.3, and two custom device drivers.

8.4 Basin testing

8.4.1 Safety - hazards and measures

8.4.1.1 Personnel injury

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

Electric shock The towing catenary should not be approached or touched.

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

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.

8.4.1.2 Material damage

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.

Towing carriage Stop before automatic stop at high speeds.

8.4.2 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 11.1.1.
   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 11.1.2.
   The remaining configuration, such as target selection is preselected in the file.
   (c) Compile the model as described in Section 11.1.3. 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 8.2, 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 11.2.3, for your controller in control screen 4: ctrl_student. Figure 8.3 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.

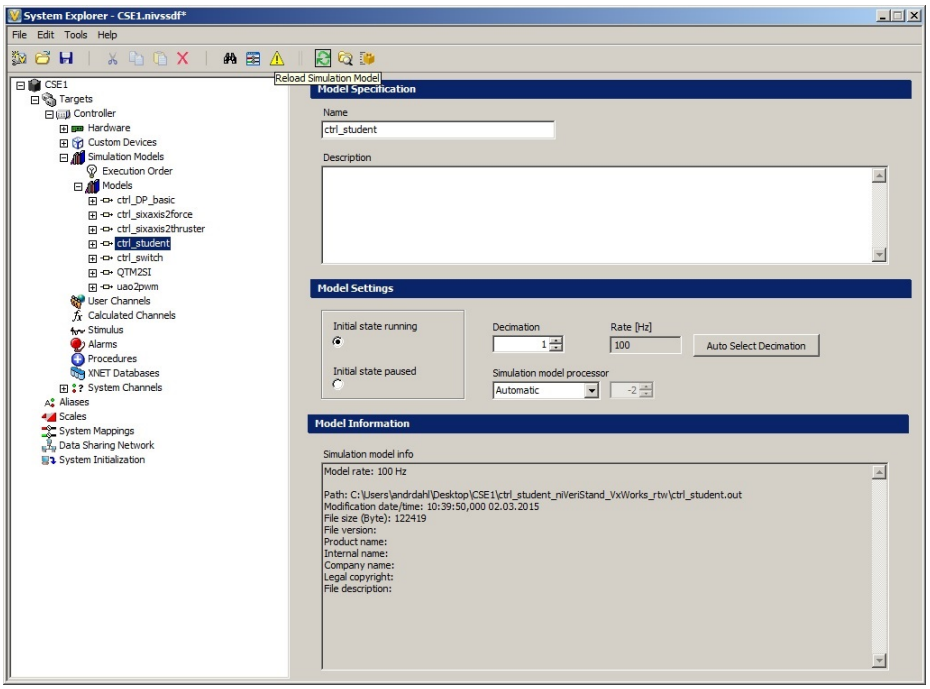


Figure 8.2: CSE1 Veristand Project simulation models

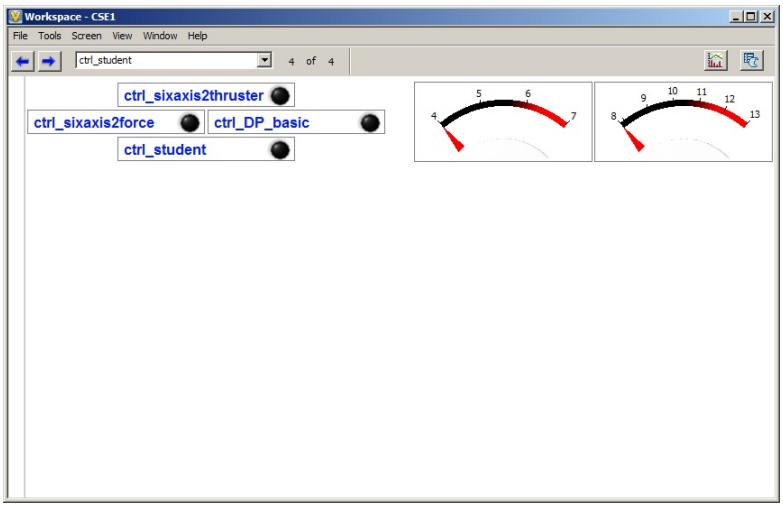


Figure 8.3: CSE1 Veristand Project ctrl_student workspace

8.4.3 Ship launching procedure - before sailing

8.4.3.1 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.

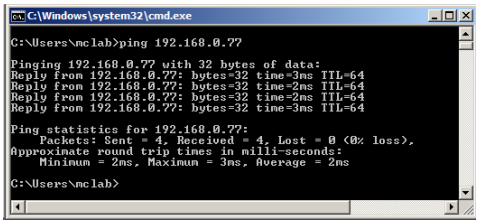


Figure 8.4: Ping, successful access to CSE1

7. Verify laptop access: ping the CSE1 IP in the command promt, as in Figure 8.4. 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.

8.4.3.2 Positioning system

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

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 8.4: Generic control modes

8.4.4 Deploy control system

Veristand osv

A diagram representation of CSE1's control system is given in Figure **??**. The first three controllers are predefined, while users may implement their own controller in the fourth.

The vessel can switch among four control modes, summarized in Table 8.4.

8.4.5 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.

Chapter 9

CS Saucer

k

Chapter 10

ROV Neptunus

k

Chapter 11

Simulation and control with cRIO

11.1 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 12.3.

11.1.1 Modeling

11.1.1.1 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.

11.1.1.2 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

[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.

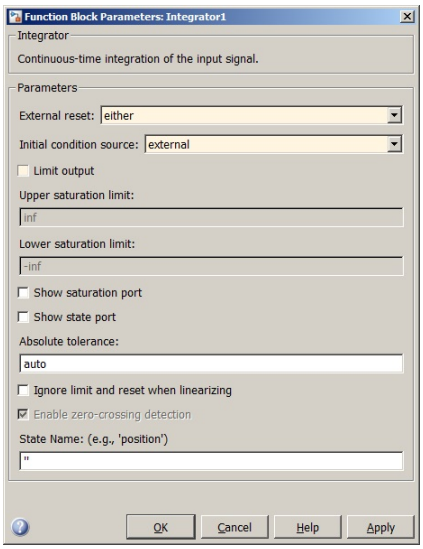


Figure 11.1: Integrator function block parameters

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 11.1.

11.1.1.3 Real-time data logging

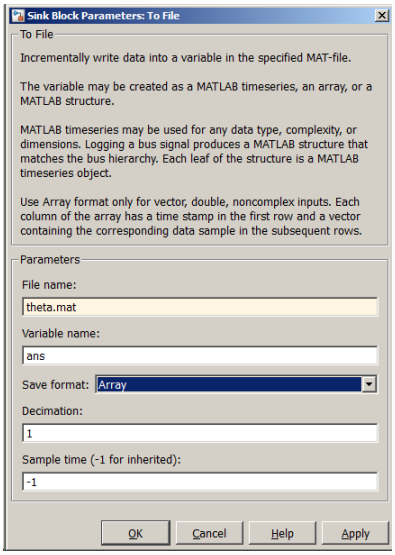


Figure 11.2: To File block parameters

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 11.2. The format should be set to Array, since the cRIO does not support the Timeseries format.

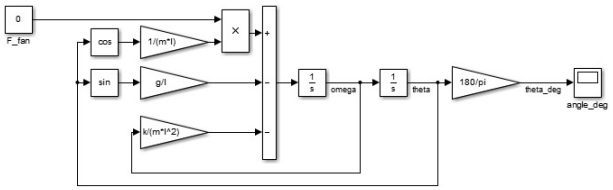


Figure 11.3: Simulink model for offline simulation

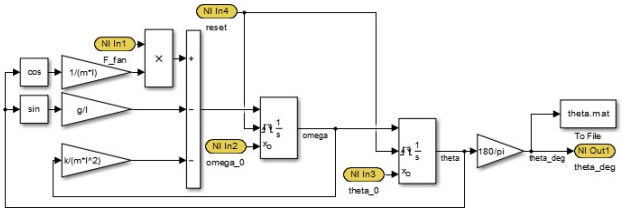


Figure 11.4: Simulink model for adjusted for compilation

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 11.3. Figure 11.4 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.

11.1.2 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 11.5. 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].

[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 11.2.2.

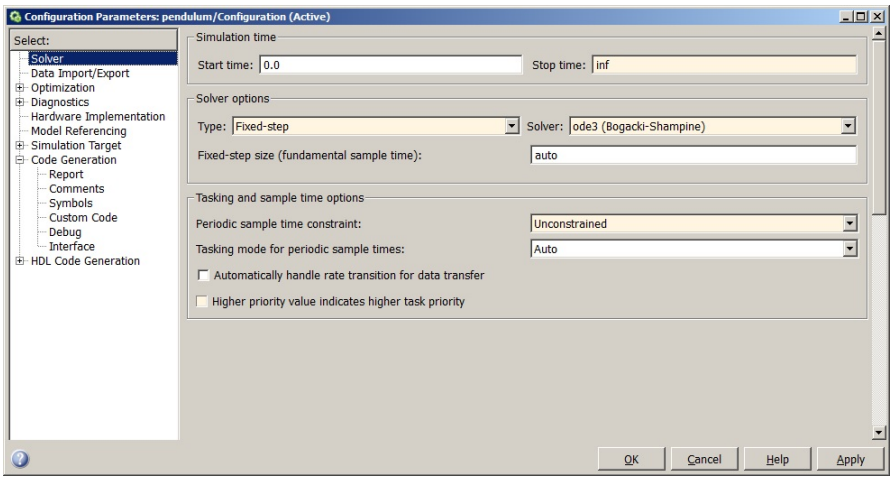


Figure 11.5: Simulink configuration parameters - solver

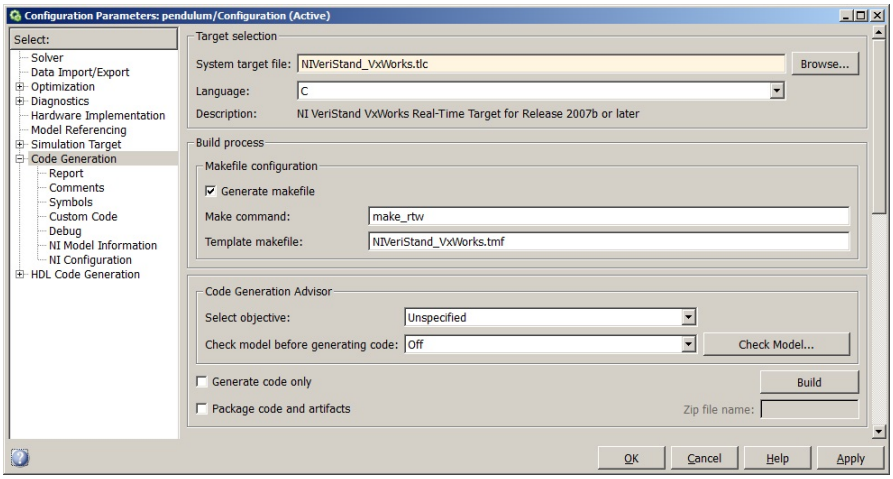


Figure 11.6: Simulink configuration parameters - target selection

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 11.6.

The WindRiver GNU Toolchain must be present in the folder specified under NI Configuration, as in Figure 11.7.

11.1.3 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

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

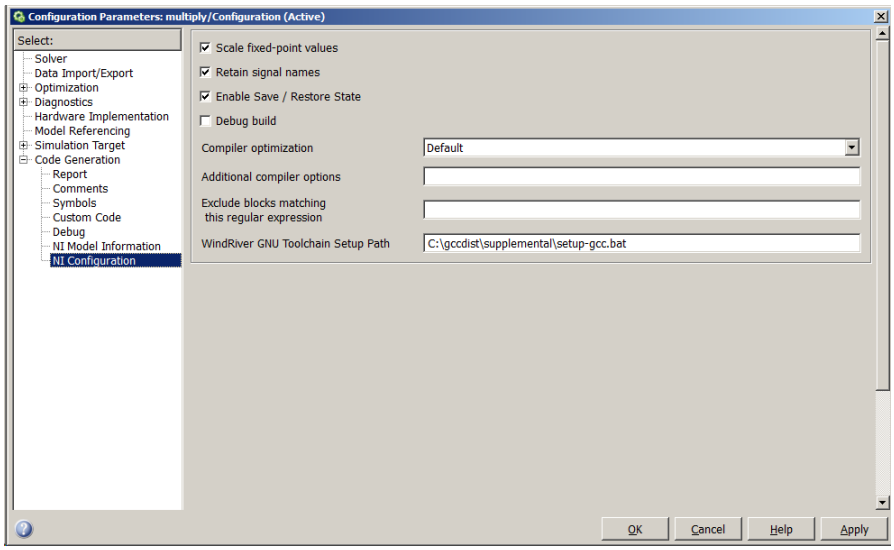


Figure 11.7: Simulink model configuration - NI configuration

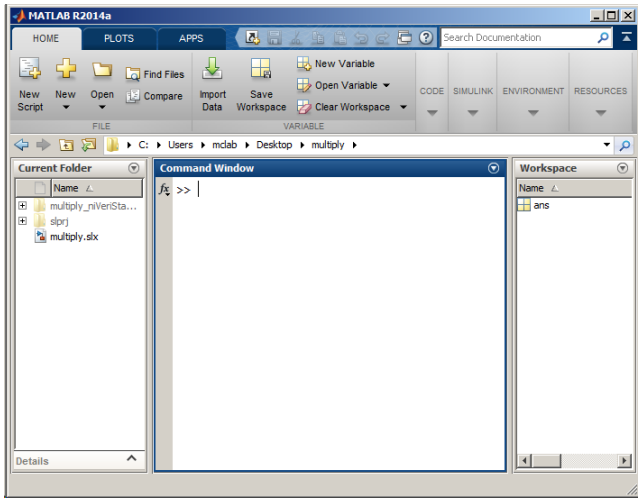


Figure 11.8: MATLAB console

Figure 11.8, 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 button.

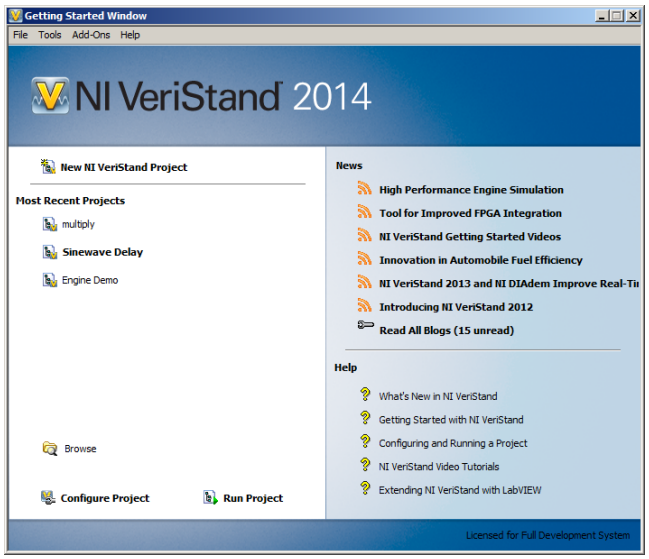


Figure 11.9: VeriStand start screen

11.2 Simulation configuration

Simulations are set up, deployed and interfaced through VeriStand. Figure 11.9 shows the start screen. Already configured projects can be run directly from here, or reconfigured.

11.2.1 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

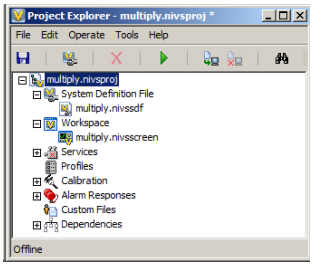


Figure 11.10: VeriStand Project Explorer

in given location and opens the Project Explorer, as in Figure 11.10. In this section, the example project name is multiply.

11.2.2 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.

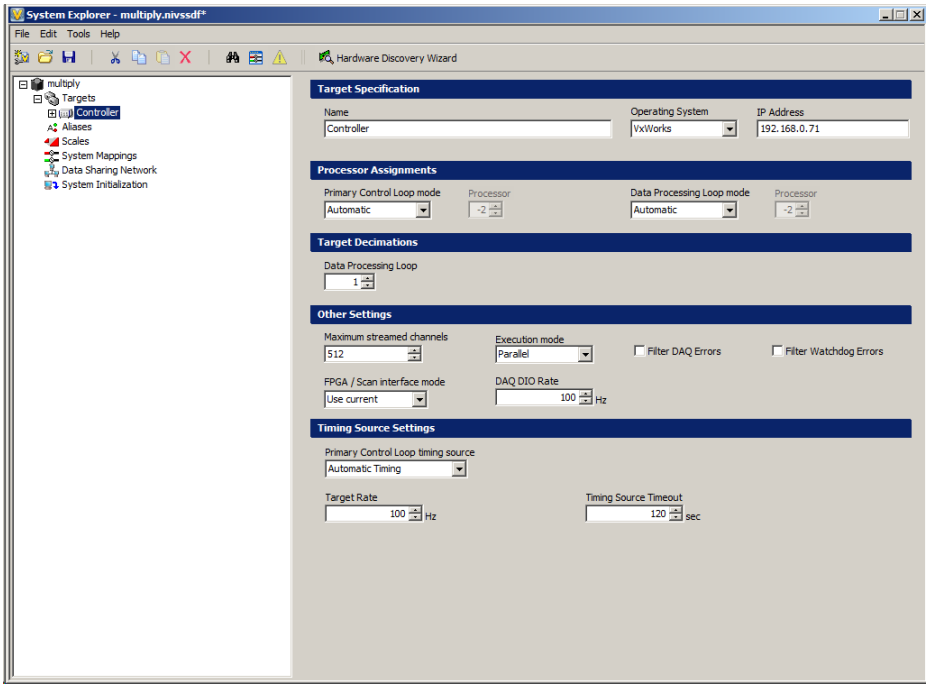


Figure 11.11: VeriStand - System Explorer - Controller

1. Set the correct controller operating system and IP address, as in Figure 11.11. All HIL and MC lab IP addresses are given in Table A.1. Also, note the target rate.

2. Click Add a Simulation Model, as seen at the top of Figure 11.12. Browse to the output of the Simulink compilation, as seen in Figure 11.13. 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 11.14 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 11.15.

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 12.1.4.

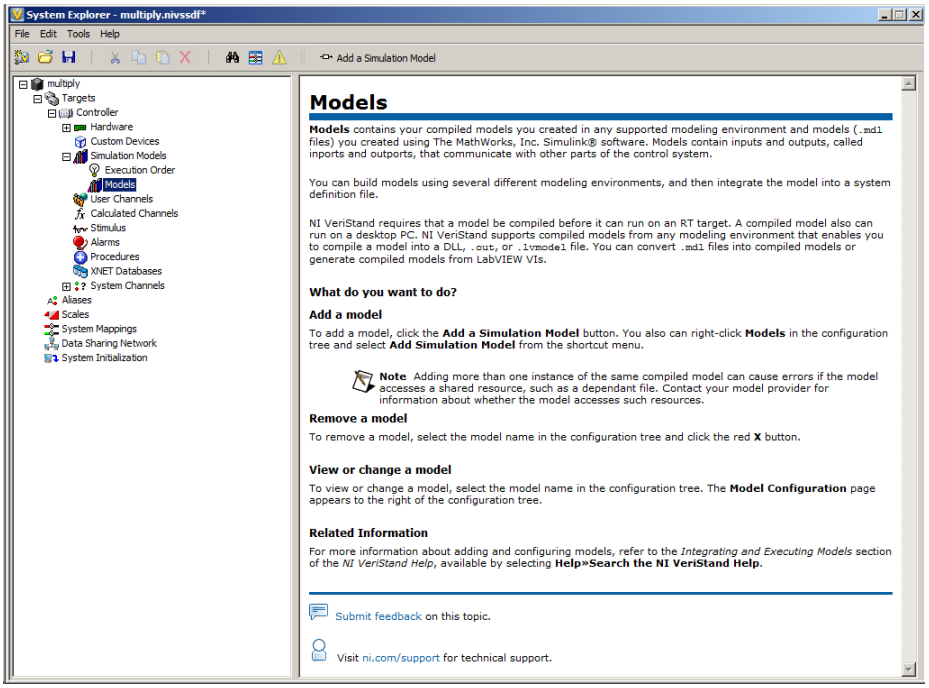


Figure 11.12: VeriStand - System Explorer - Models

11.2.3 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.

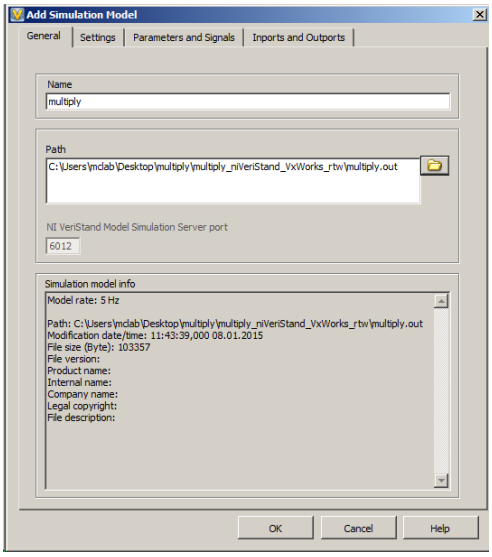


Figure 11.13: VeriStand - System Explorer Model

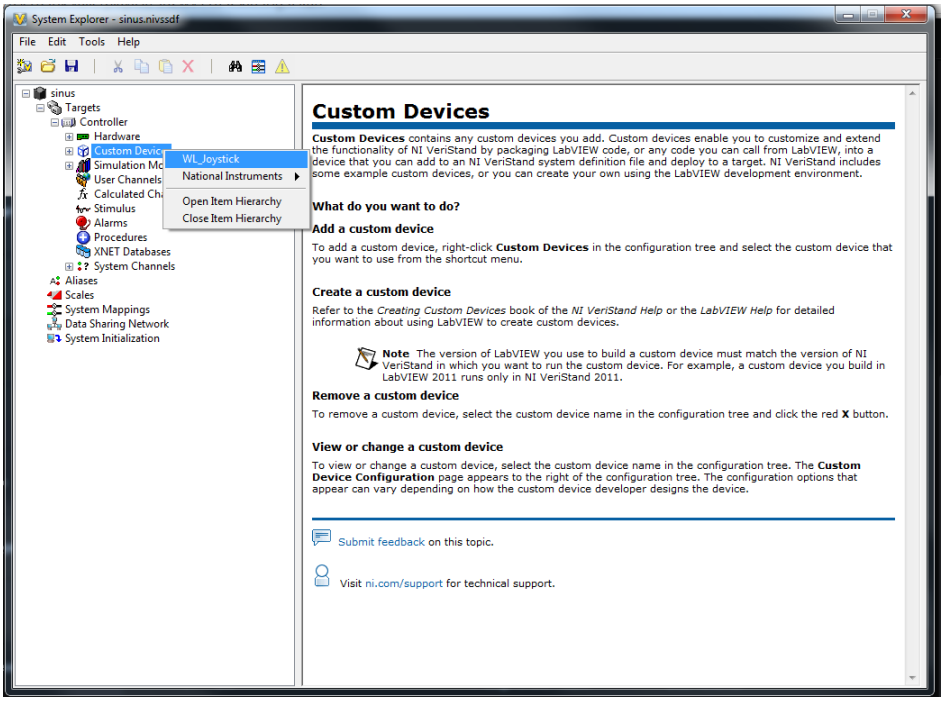


Figure 11.14: Custom device selection

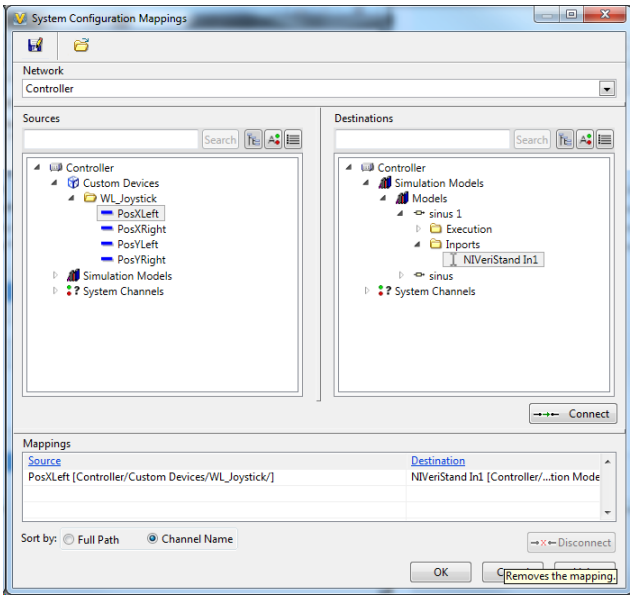


Figure 11.15: VeriStand System Configuration Mappings

11.3 Deployment and simulation

11.3.1 Run

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

11.3.2 User interface side data logging

For reliability, it is recommended to log data directly on the cRIO during simulation, as described in Section 11.1.1.3. It is also possible to log via the laptop user interface.

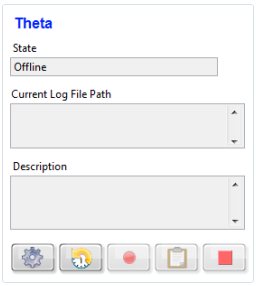


Figure 11.16: Logging Control

A Logging Control, as seen in Figure 11.16, must be added to the workspace to export data from the simulation. The control is added as described in Section 11.2.3.

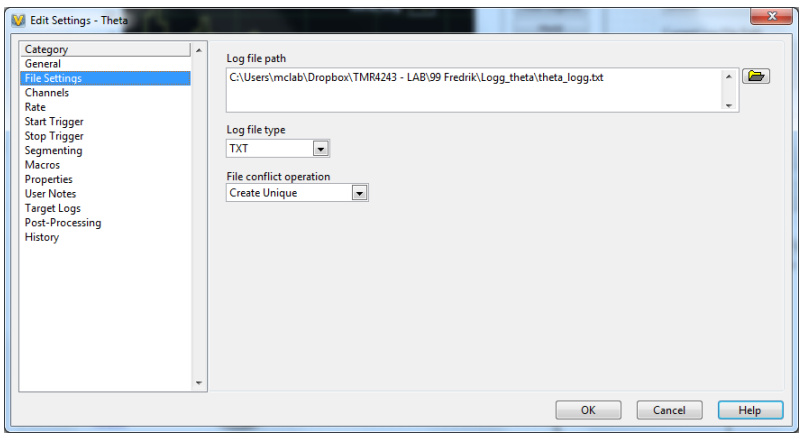


Figure 11.17: Logging Control file settings

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 11.17. Under Channels, the desired channels can be selected and added, as in Figure 11.18.

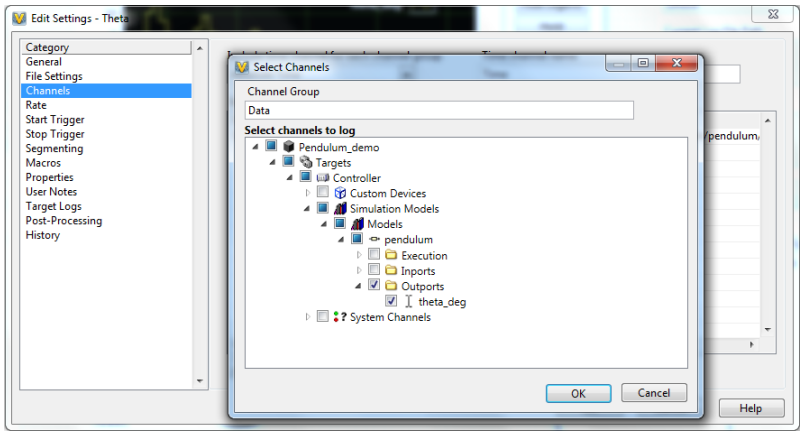


Figure 11.18: Logging Control add channel

11.3.3 Stop

button

11.3.4 FTP data retrieval

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

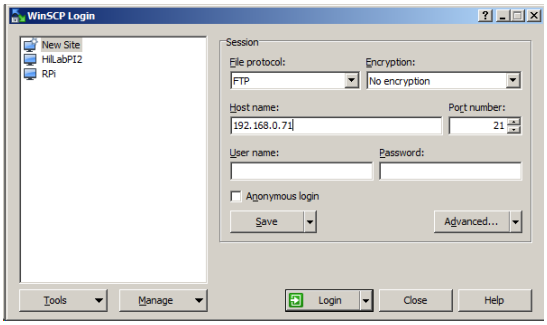


Figure 11.19: WinSCP login

To connect to the cRIO, the correct IP must be specified, as in Figure 11.19. 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 11.20. Data is transferred to the laptop by drag and drop to the desired location in the left pane.

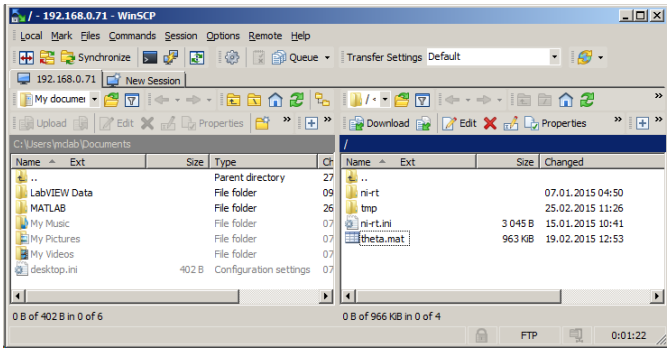


Figure 11.20: WinSCP

Part III

Laboratory Staff Guide

Chapter 12

HIL lab setup

12.1 cRIO

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

12.1.1 Ethernet ports

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

12.1.1.1 Primary

Set fixed IP, set fixed IP on HIL-computers

12.1.1.2 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*

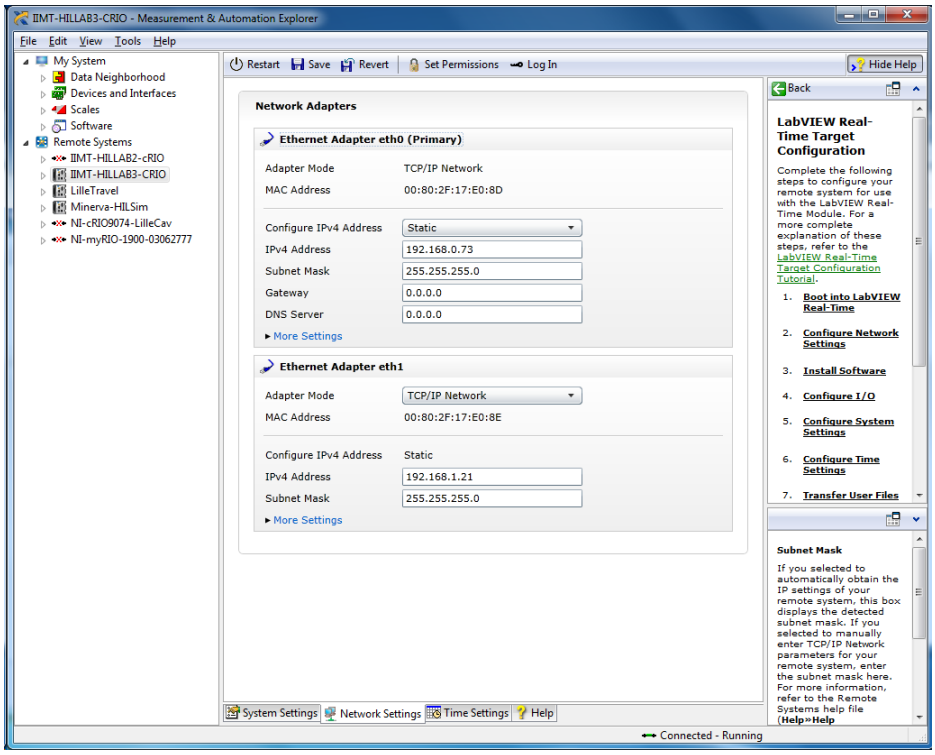


Figure 12.1: NI MAX - Network Settings

12.1.2 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.

12.1.2.1 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 12.2
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 12.3
5. Click next without making any changes12.4
6. Click next without making any changes12.5
7. Wait for the installtion to finish and the cRIO to reboot

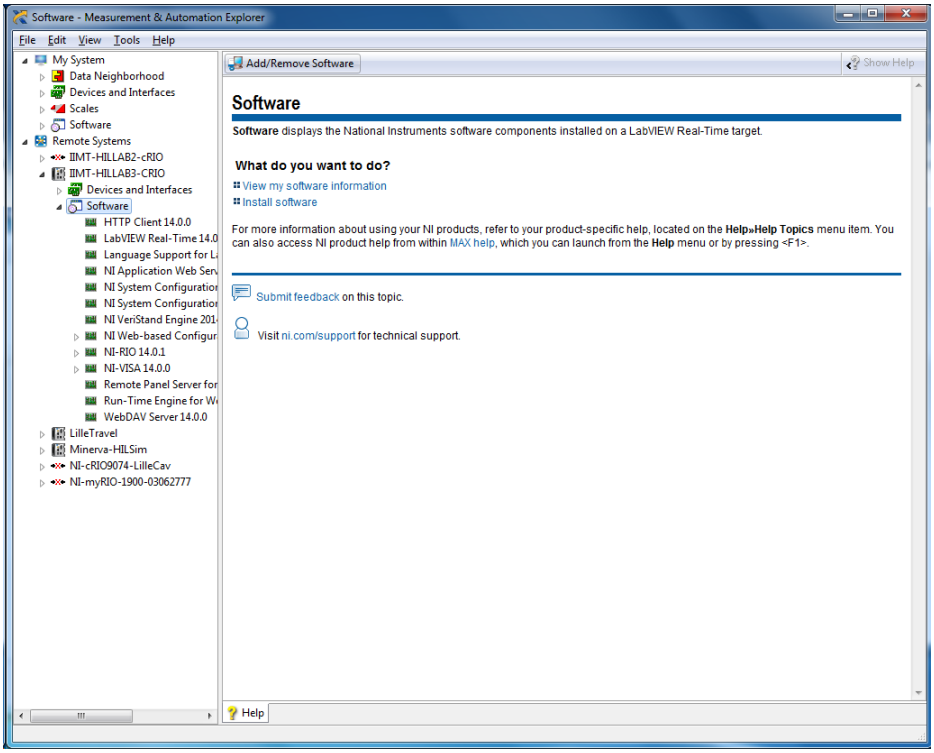


Figure 12.2: NI MAX - Software Update 1

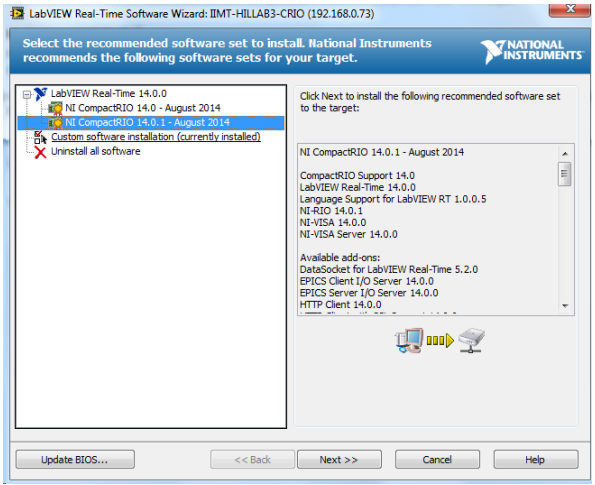


Figure 12.3: NI MAX - Software Update 1

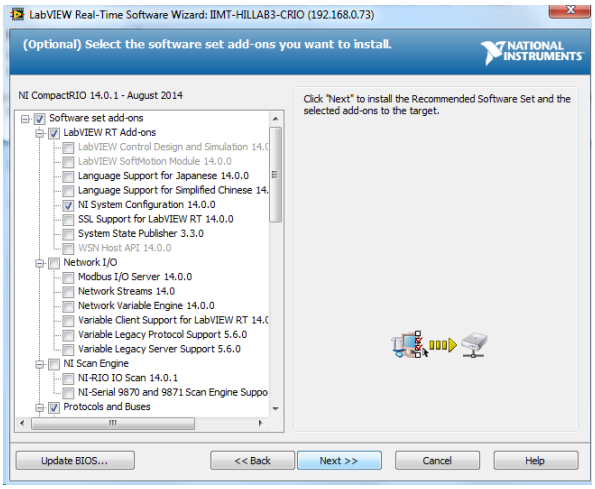


Figure 12.4: NI MAX - Software Update 3

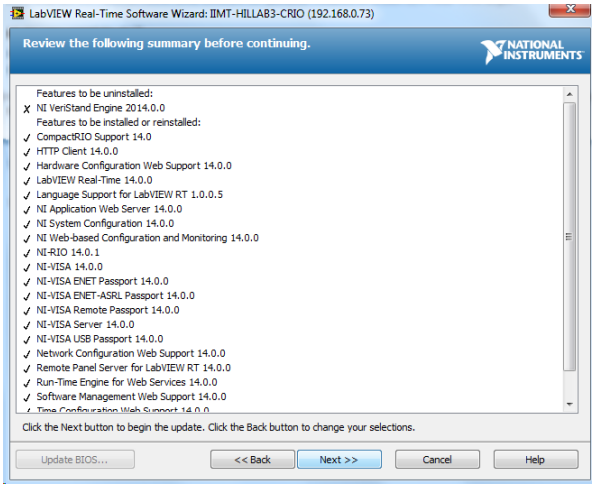


Figure 12.5: NI MAX - Software Update 4

12.1.2.2 NI Veristand Engine

1. Repeat step 1-3 from the previous guide
2. Now you choose Custom Software installation in the menu, see Figure 12.6
3. Ignore the warning, See Figure 12.7
4. Locate NI Veristand Engine 2014 0.0 and click install feature. See Figure 12.8
5. Click your way through the rest of the installation and let the cRIO reboot.

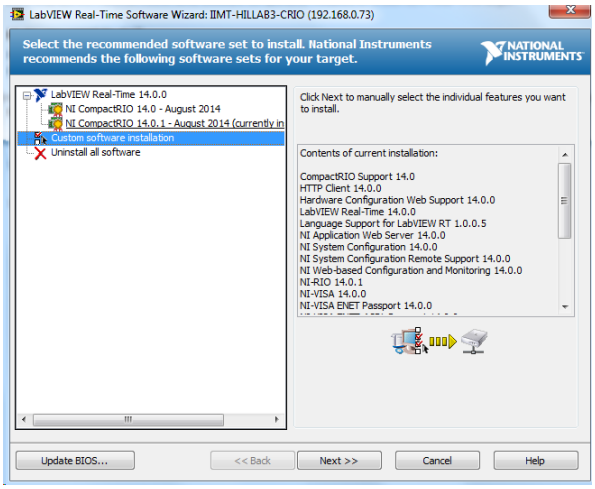


Figure 12.6: NI MAX - NI Veristand Engine installation 1

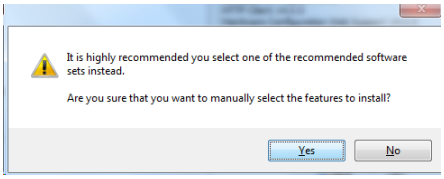


Figure 12.7: NI MAX - NI Veristand Engine installation 1

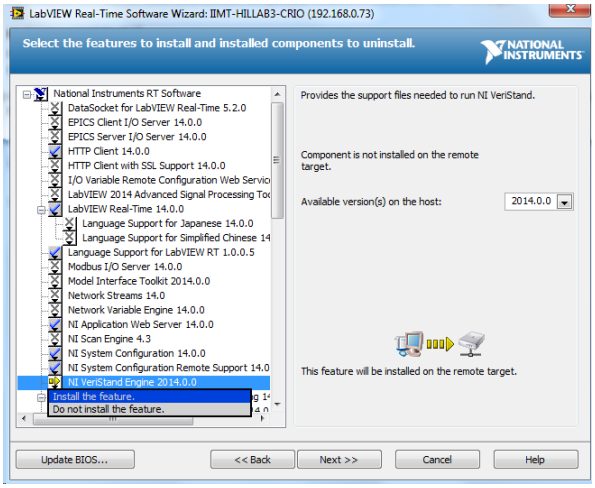


Figure 12.8: NI MAX - NI Veristand Engine installation 1

12.1.3 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 12.18. 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 12.19. You will now have to redesign this to look like Figure 12.20. 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 12.25, 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`

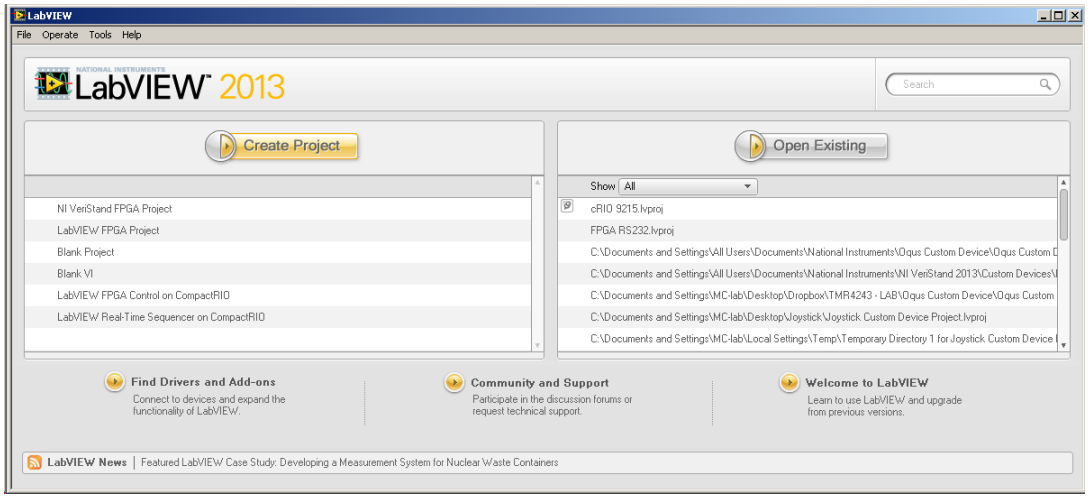


Figure 12.9: Create Labview FPGA target and XML - 1

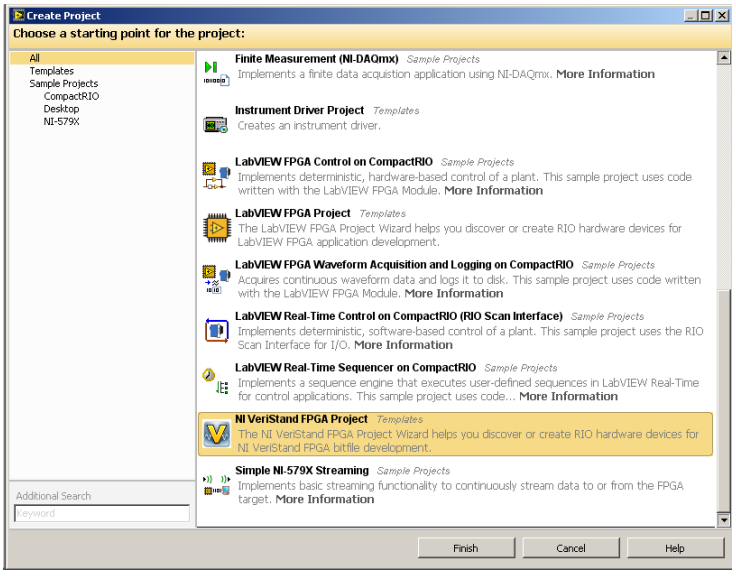


Figure 12.10: Create Labview FPGA target and XML - 2

Veristand FPGA programmering

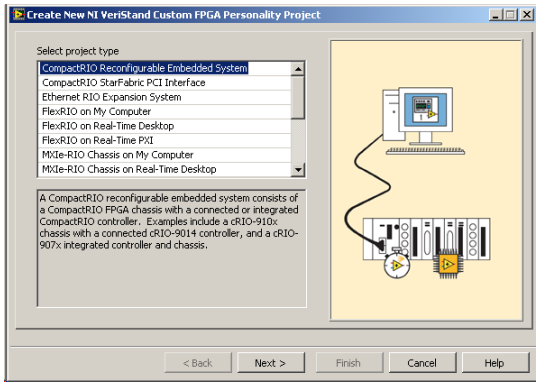


Figure 12.11: Create Labview FPGA target and XML - 3

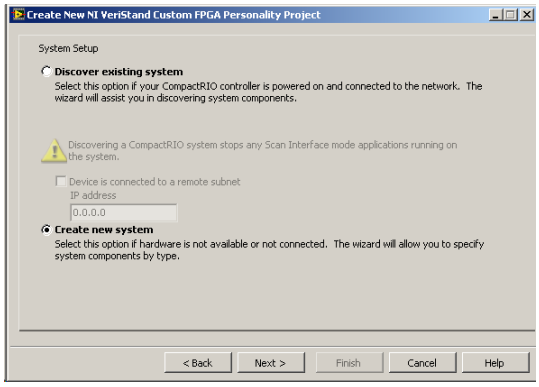


Figure 12.12: Create Labview FPGA target and XML -4

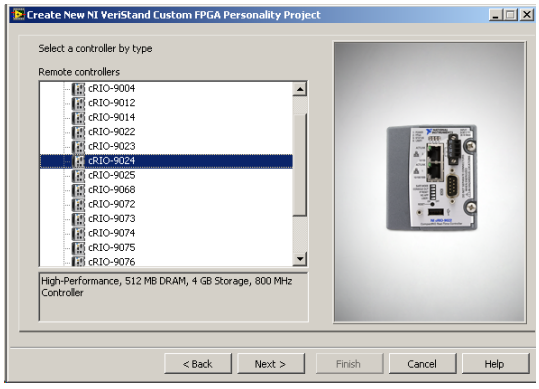


Figure 12.13: Create Labview FPGA target and XML - 5

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.

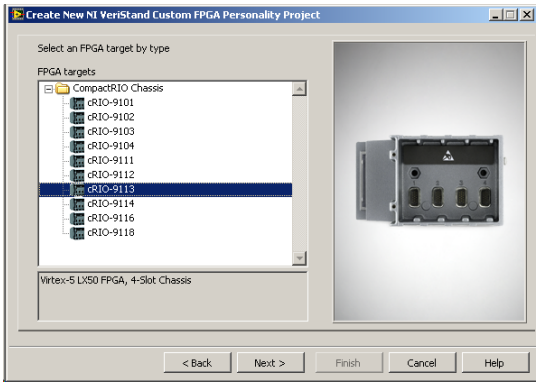


Figure 12.14: Create Labview FPGA target and XML - 6

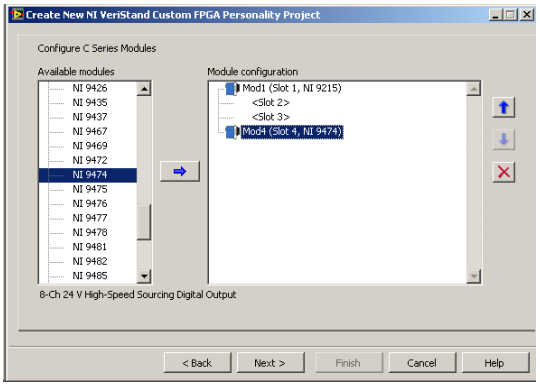


Figure 12.15: Create Labview FPGA target and XML - 7

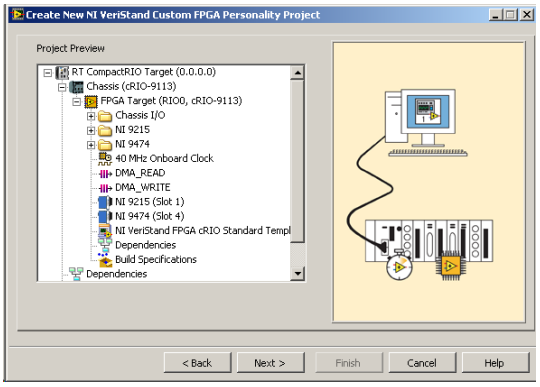


Figure 12.16: Create Labview FPGA target and XML - 8

12.1.3.1 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.

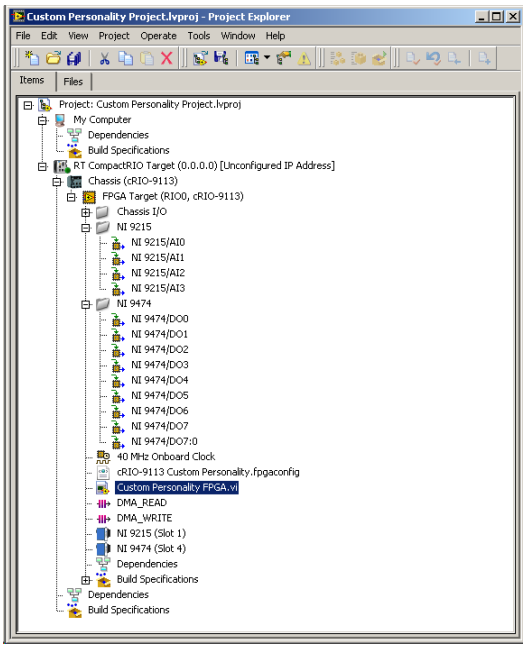


Figure 12.17: Create Labview FPGA target and XML -9

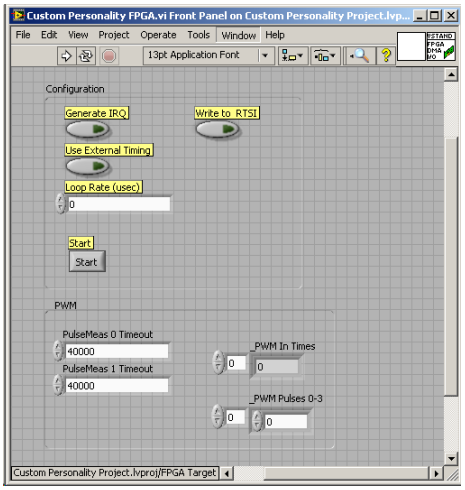


Figure 12.18: Create Labview FPGA target and XML - 10

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 12.26.

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 12.27, 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 Figure12.33.

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

Eirik: FPGA-greier osv

Analog input

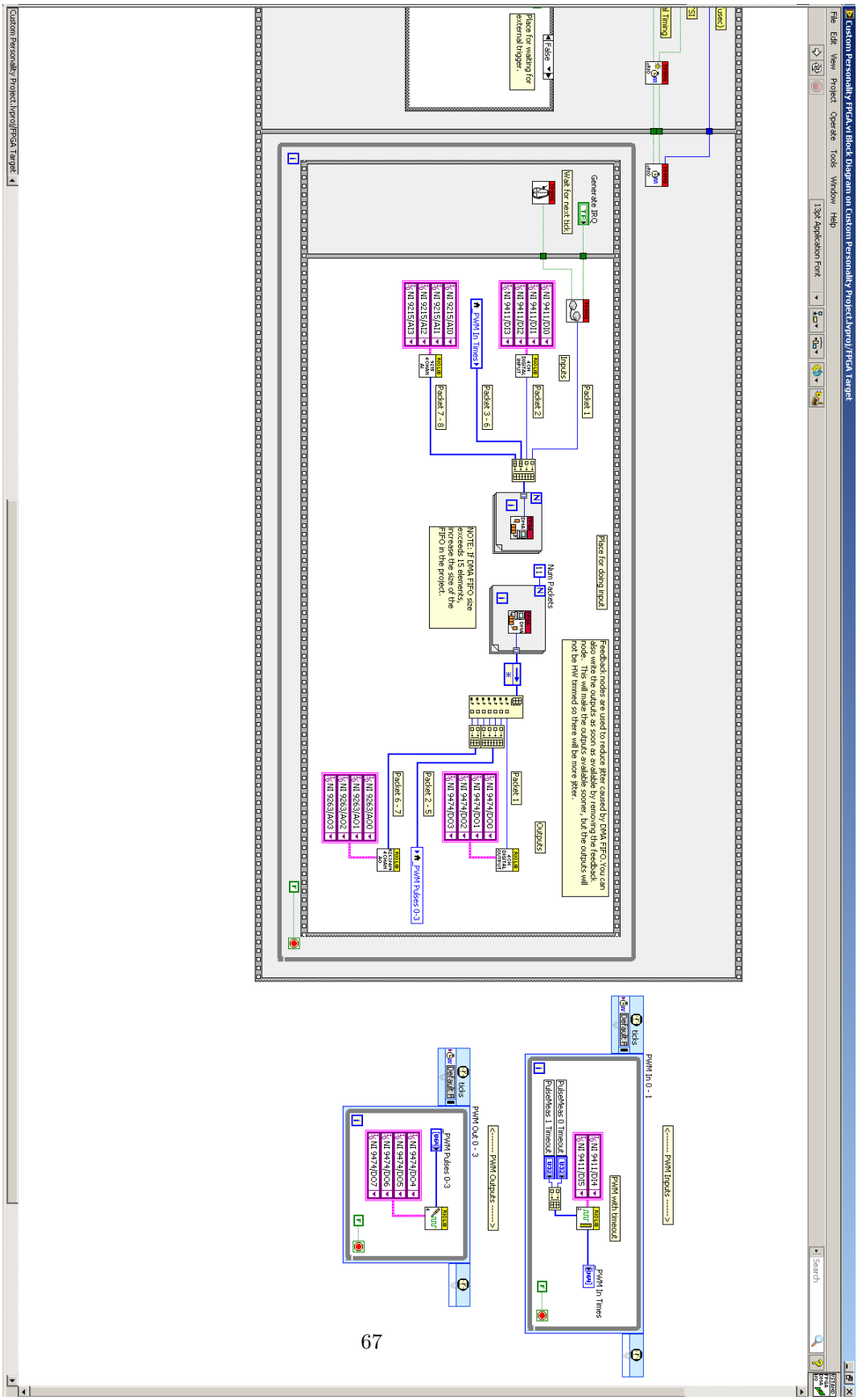


Figure 12.19: Create Labview FPGA target and XML - 11

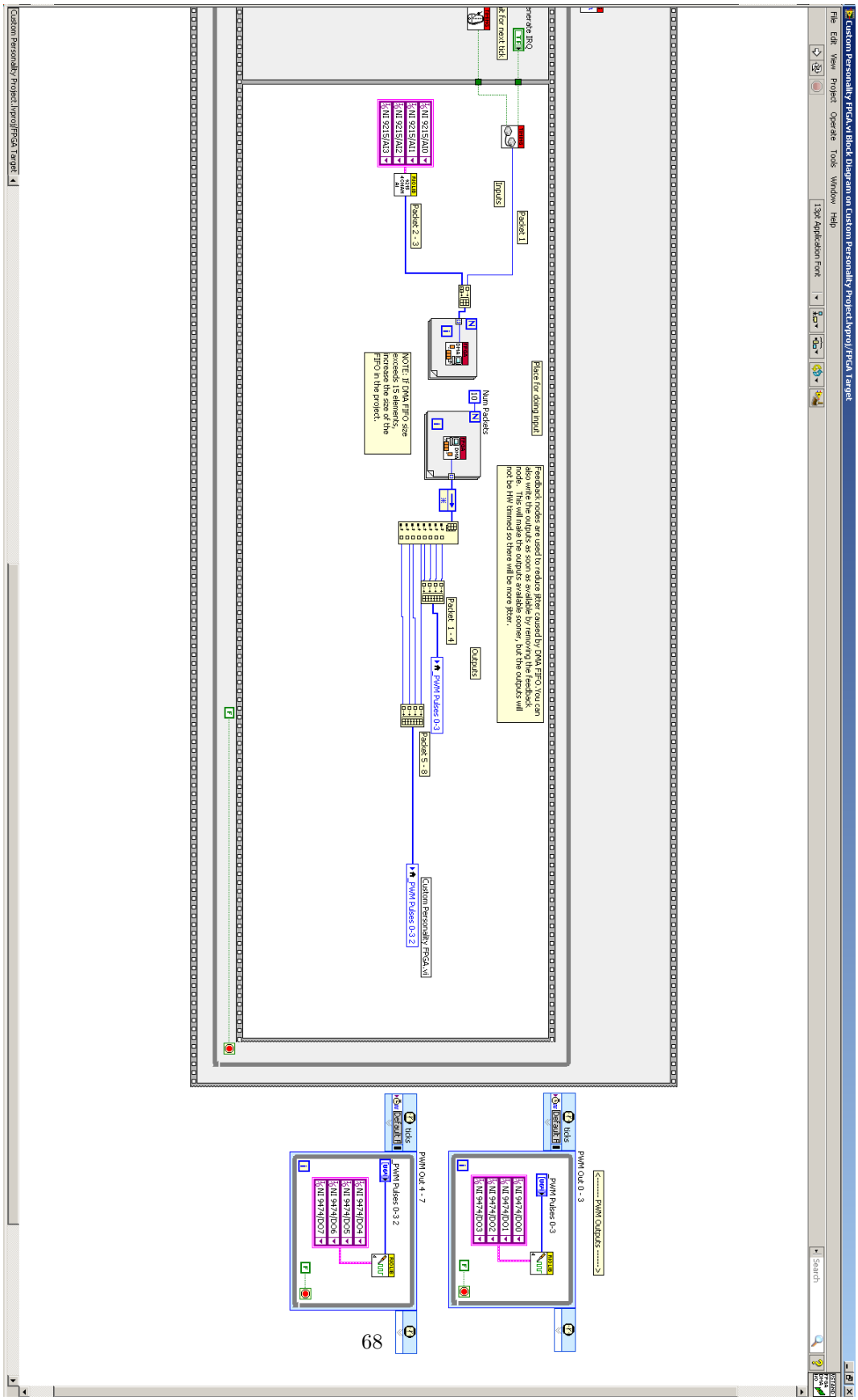


Figure 12.20: Create Labview FPGA target and XML - 12

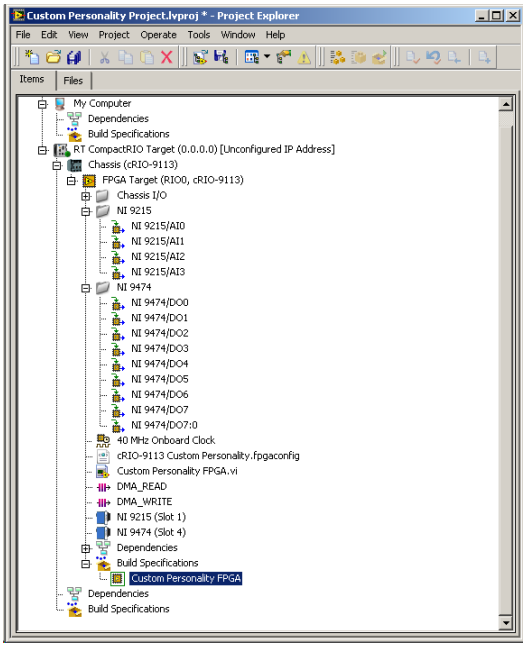


Figure 12.21: Create Labview FPGA target and XML - 13

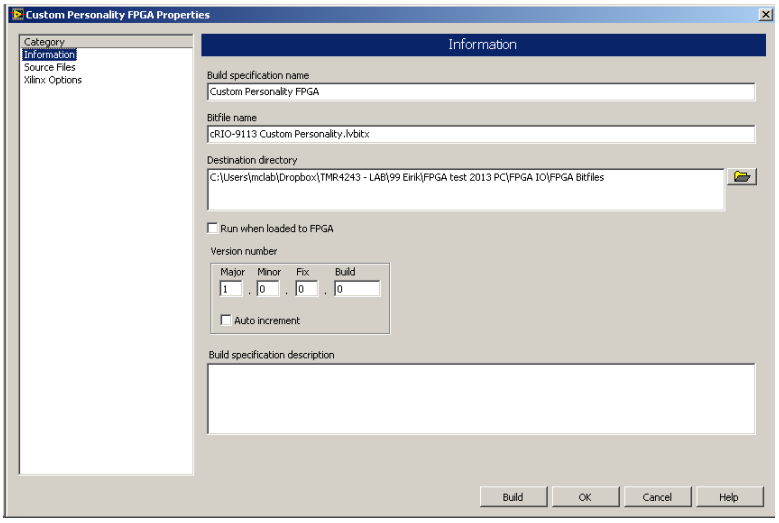


Figure 12.22: Create Labview FPGA target and XML - 14

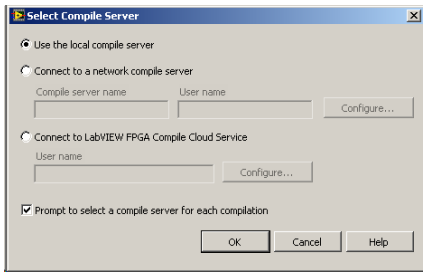


Figure 12.23: Create Labview FPGA target and XML - 15

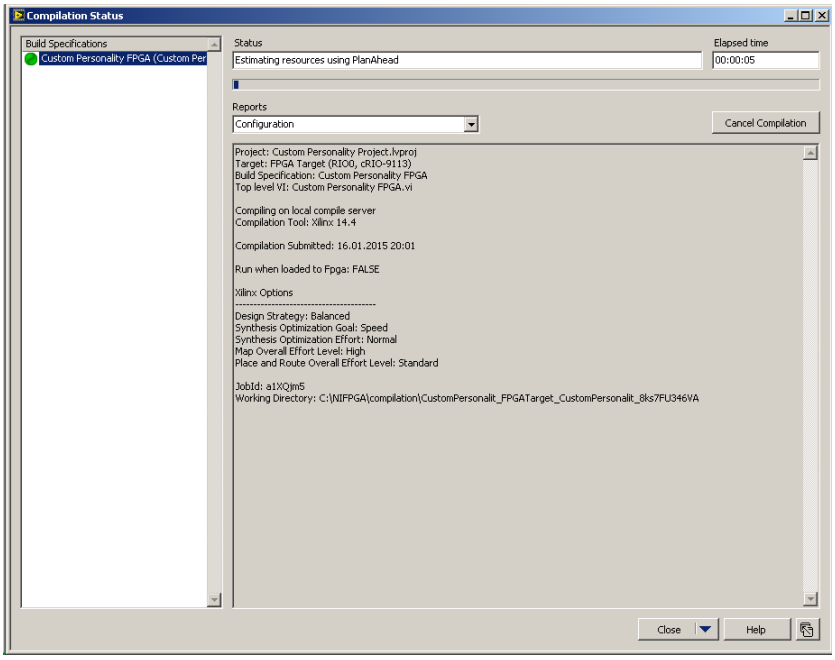


Figure 12.24: Create Labview FPGA target and XML - 16

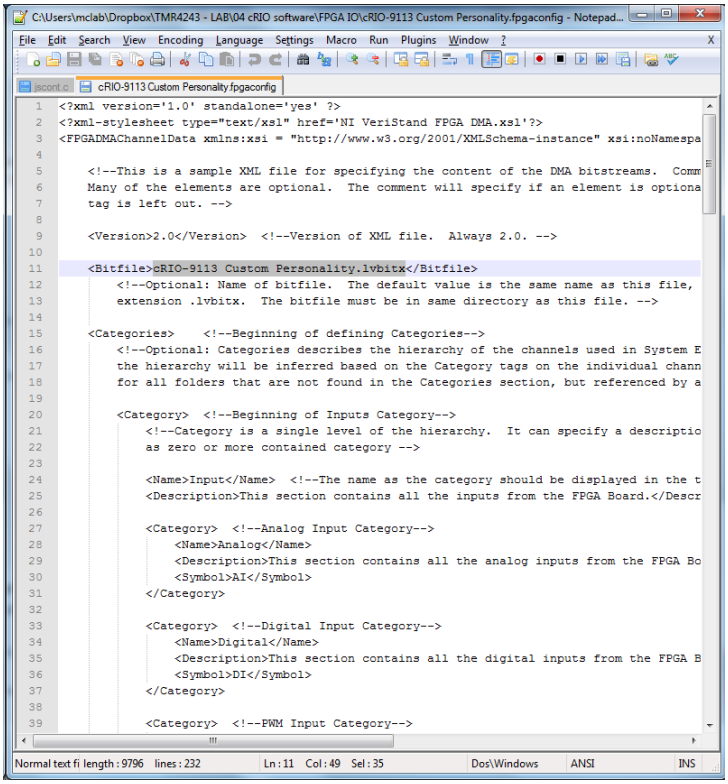


Figure 12.25: Create Labview FPGA target and XML - 17

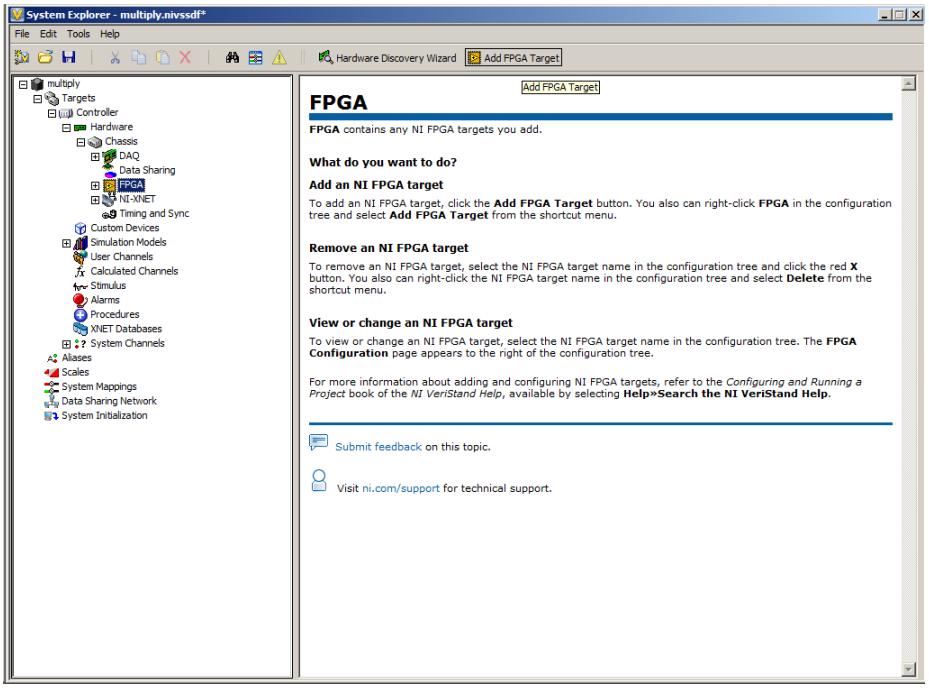


Figure 12.26: FPGA1

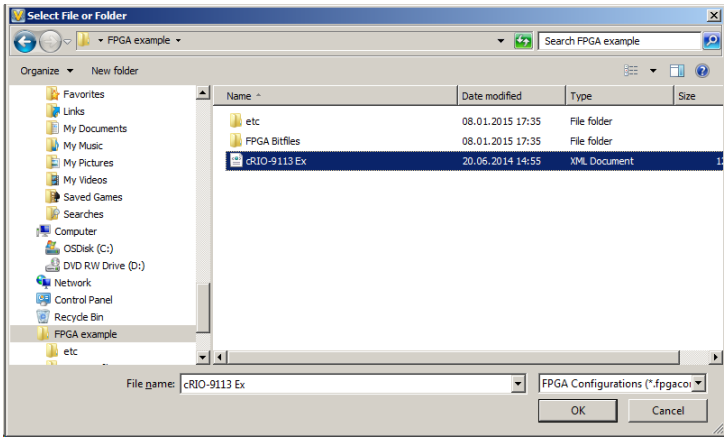


Figure 12.27: FPGA2

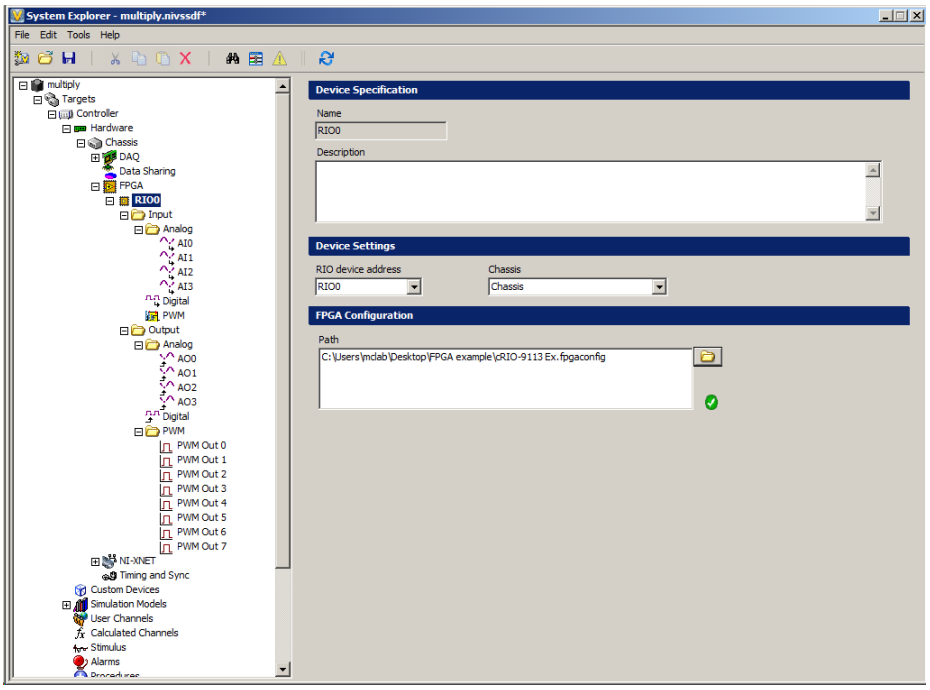


Figure 12.28: FPGA3

12.1.4 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 12.29.

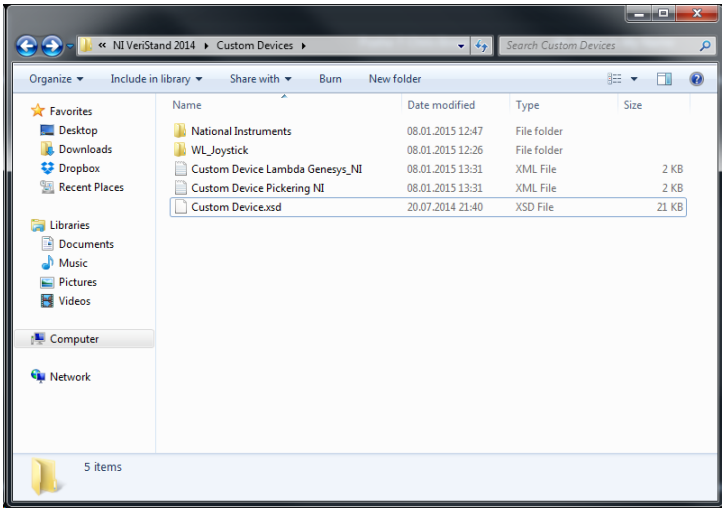


Figure 12.29: Custom device folder

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 12.30.

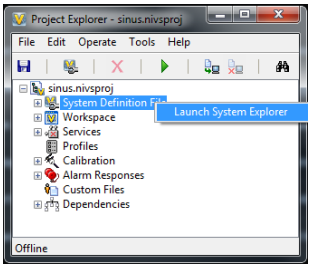


Figure 12.30: VeriStand launch system explorer

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

12.31. 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.

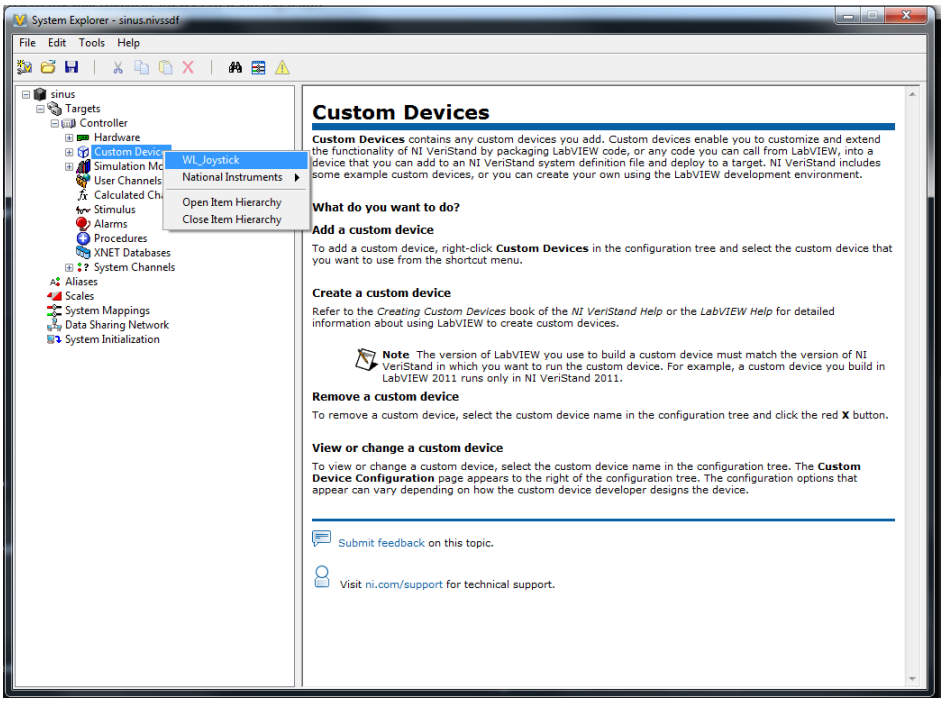


Figure 12.31: Custom device selection

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 12.32. 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 12.32).

You the simply find the ports you would like to connect, mark them and click the connect button. Figure 12.33 a joystick output is connected to a input port on the Simulink model.

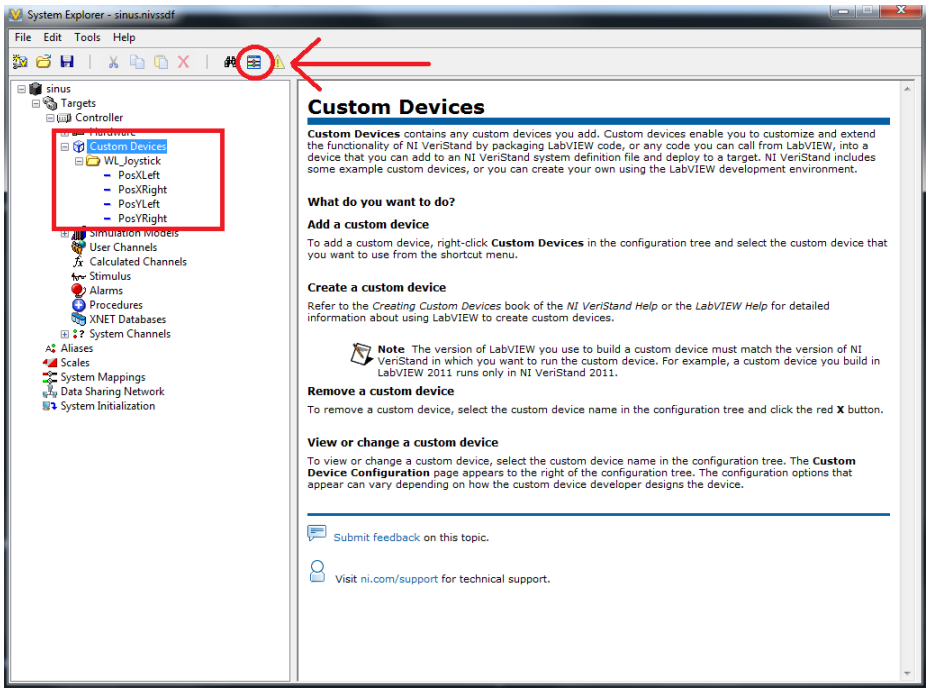


Figure 12.32: VeriStand

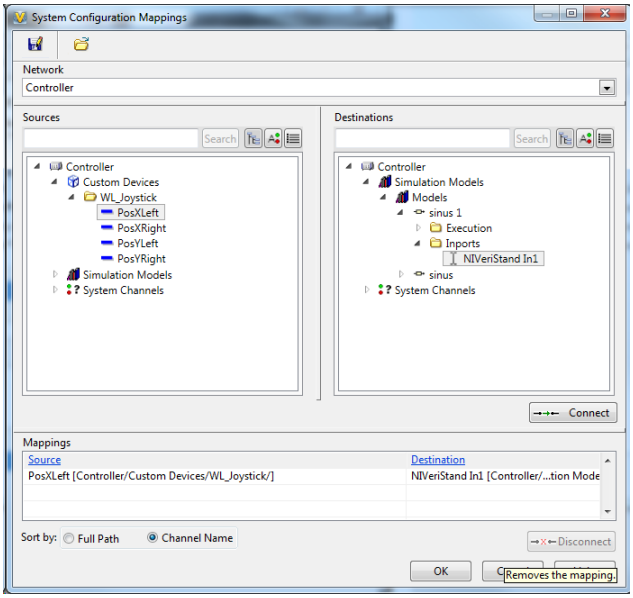


Figure 12.33: VeriStand System Configuration Mappings

12.2 Raspberry Pi

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

12.2.1 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.

12.2.1.1 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 12.1: RPi installation and setup utilities

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

12.2.1.2 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 12.34. 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

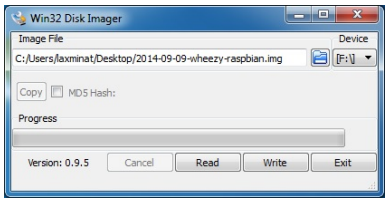


Figure 12.34: Disk Imager

12.2.1.3 Terminal access

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

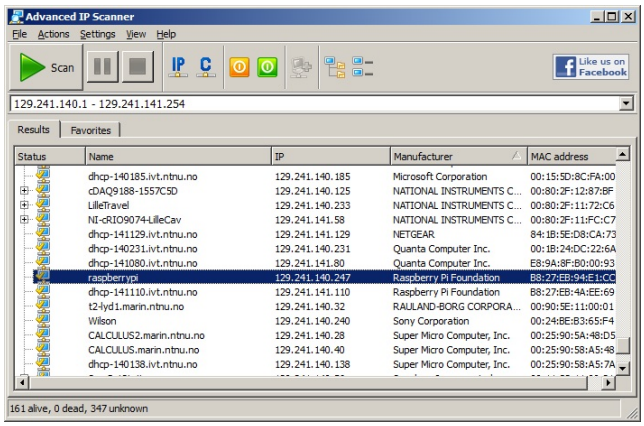


Figure 12.35: Advanced IP Scanner

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 12.35.

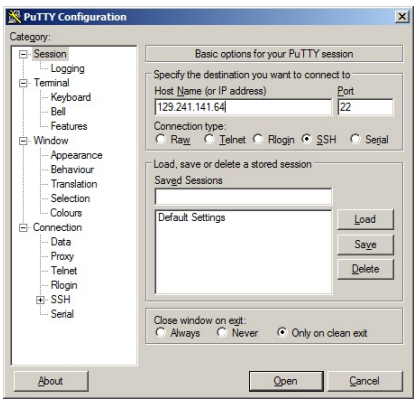


Figure 12.36: Putty settings

Once the IP is known, it is specified in the Putty settings, as in Figure 12.36, and a connection can be opened.

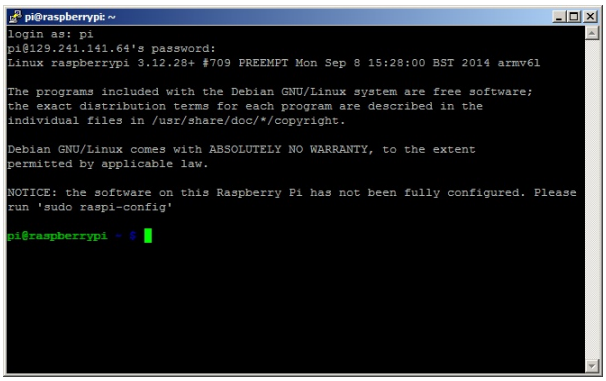


Figure 12.37: SSH connection

The default login is `pi`, and the default password `raspberry`. Figure 12.37 shows the terminal output on first login.

12.2.1.4 Finalize configuration

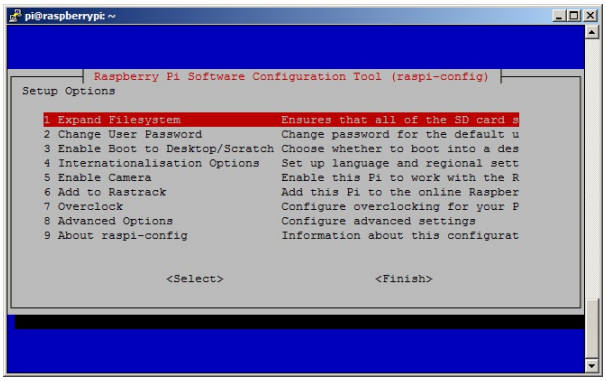


Figure 12.38: RPi configuration tool

Enter the

```
sudo raspi-config
```

command to start the RPi Software Configuration Tool, as in Figure 12.38. 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.

12.2.1.5 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.

12.2.1.6 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.

12.2.2 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.

12.2.2.1 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.

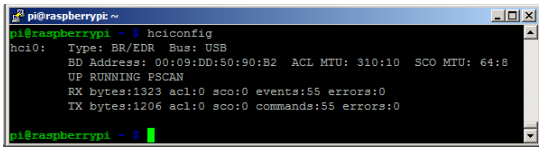


Figure 12.39: Bluetooth configuration tool

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 12.39. If instead it says `DOWN`, some error har occured.

Most experienced errors were due to typos.

12.2.2.2 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.

12.2.2.3 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
```

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.

12.2.2.4 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 12.2.1.5), 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 12.40. To exit the server use the key combination Ctrl+C.

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

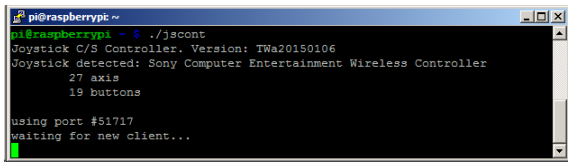


Figure 12.40: Joystick signal server test

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.

12.3 Laptop

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

12.3.0.1 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

12.3.0.2 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

Chapter 13

Cybership Enterprise 1

Increased servo percentage results in clockwise? motion.

Hystersis on motors

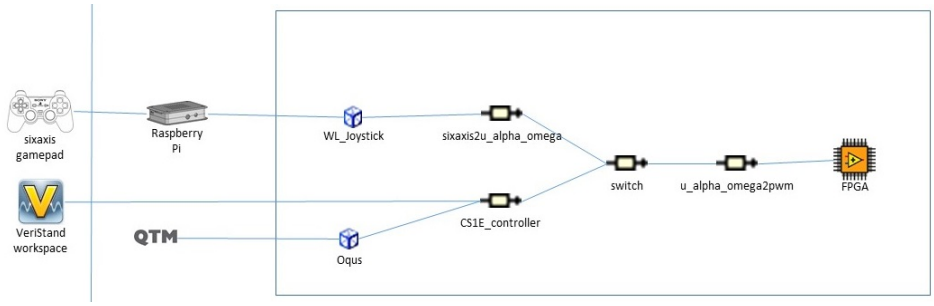


Figure 13.1: CSE1 component communication

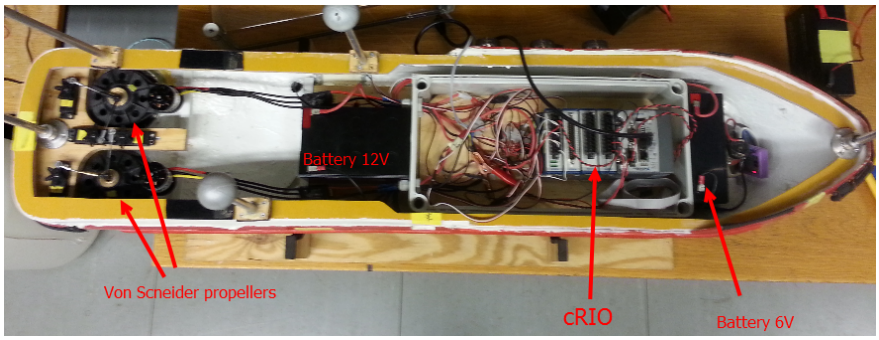
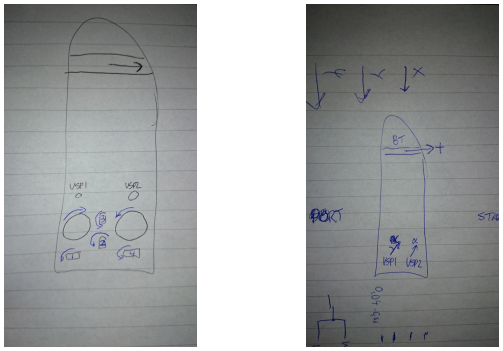


Figure 13.2: CSE1 - Hardware

13.1 Actuators

Antall, posisjon, aktivert med pwm.

Figure 13.3: CSE1 diagram



(a) Hardware input (b) Control input

Figure 13.4: 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 13.1: 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.

13.1.1 Motor control signals

13.1.2 Servo control signals

13.1.3 Measurements

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

Table 13.2: CSE1 cRIO analog input

13.2 Control software

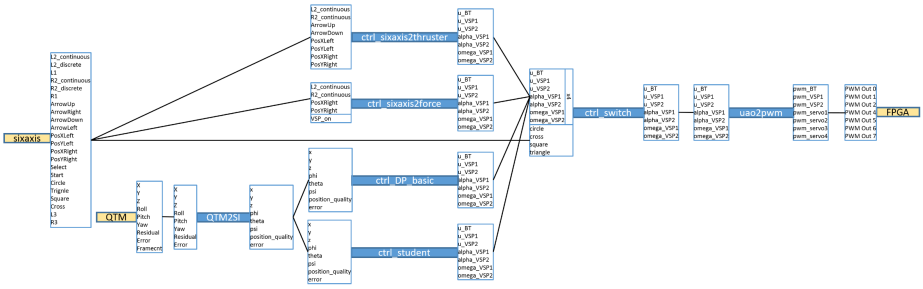


Figure 13.5: CSE1 control software modules

13.2.1 sixaxis (currently named WL_joystick) custom device

Reads sixaxis gamepad input from the RPi server.

13.2.2 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

13.2.3 QTM2SI

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

13.2.4 ctrl_sixaxis2thruster control module

Provides manual thruster control.

13.2.4.1 Voith Schneider Propellers

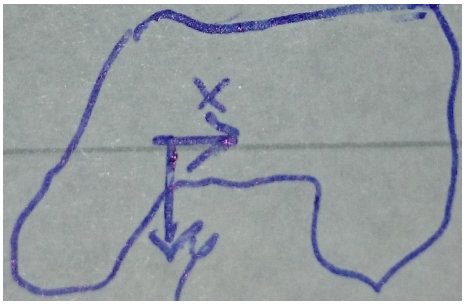


Figure 13.6: Sixaxis coordinate system

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 13.4b. The joystick coordinates PosX and PosY axes point right and down, as seen in Figure 13.6. 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.

13.2.4.2 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 13.4b.

13.2.5 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 13.4b. 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

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

13.2.6 ctrl_DP_basic control module

Provides basic dynamic positioning capability.

13.2.7 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 13.3: Control input ranges

13.2.8 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 13.3 and issues a warning signal if the current controller exceedes the bounds.

13.2.9 uao2pwm module

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

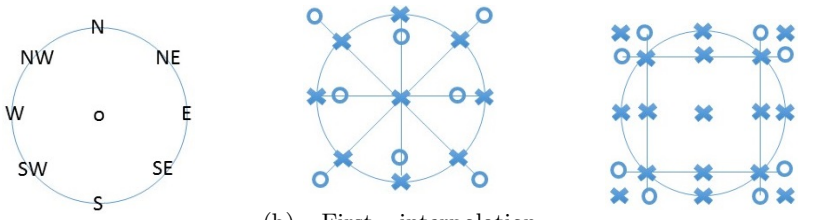
The position of the VSP steering rods are controlled by a pair of servos for each.

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 13.4: Servo pwm ranges

Ikke lineært, ikke rett frem.

Foreslått metode



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

Figure 13.7: Servo, rod position tuning

13.2.10 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 13.5: PWM connections

13.3 Qualisys body

calibration

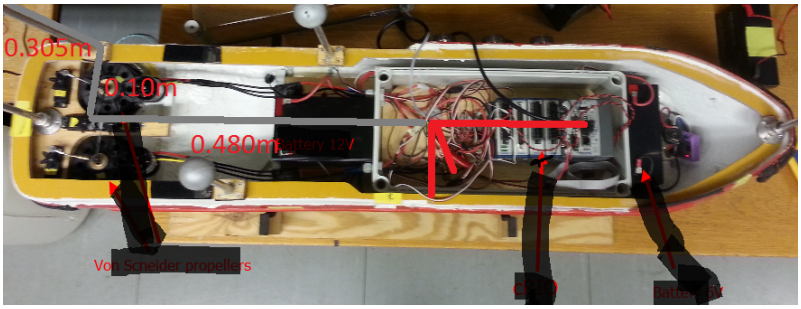


Figure 13.8: Matlab console

Chapter 14

Qualisys motion capture system

14.1 Start server

Figure 14.1: Matlab console

14.2 Aquire body

leverer med 50Hz på nettet

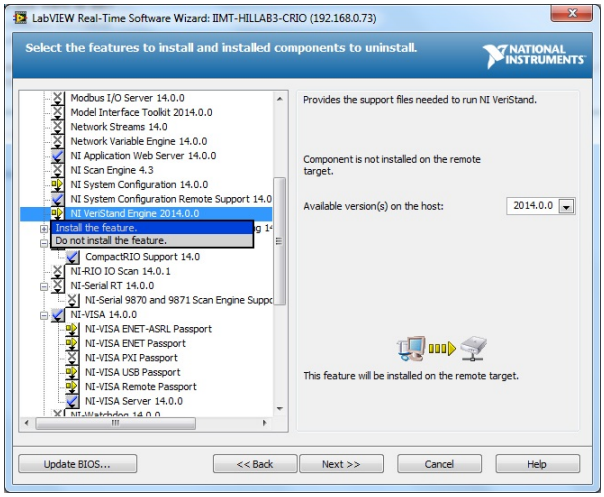


Figure 14.2: FPGA2

Part IV

Miscellaneous

Appendix A

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 A.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 B

Checklist

Device driver in place

Custom indicators in place

PC and cRIO IP correct

Sixaxis charged

Appendix C

Personel and literature

C.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

C.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).

C.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.

C.4 Other

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

Appendix D

Maintenance

Oil VSP

Appendix E

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 E.1: Suppliers

Appendix F

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