

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 the stepwise controller development.

Part II is a user guide intended for users of the laboratory. 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

I	Introduction	1
1	Marine cybernetics laboratory	2
1.1	Fixed Equipment	4
1.1.1	Qualisys motion capture system	4
1.1.2	Towing carriage	4
1.1.3	Wave generator	4
1.2	Vessels	5
1.2.1	CS Inocean Cat I Arctic Drillship	6
1.2.2	CS Saucer	7
1.2.3	ROV Neptunus	8
1.2.4	CS Enterprise I	9
2	Control system development philosophy	13
2.1	Development Steps	14
2.1.1	Model-in-the-Loop	14
2.1.2	Software-in-the-Loop	14
2.1.3	Processor-in-the-Loop	15
2.1.4	Hardware-in-the-Loop	15
2.1.5	Scale test	15
2.1.6	Full scale	15
2.2	Control Modes	16
2.2.1	Individual actuator control	16
2.2.2	Generalized force control	17
2.2.3	Regulation	18
2.2.4	Marine Operations Control	18
II	Laboratory user guide	20
3	Qualisys motion capture system	21
4	Towing carriage	22
4.1	Movement of towing wagon	22
4.2	Typical settings	24
4.3	Note	24
5	Wave generator	25

6 CS Inocean Cat I Arctic Drillship	26
6.1 Mathematical model	26
7 CS Enterprise I	27
7.1 Mathematical model	27
7.2 Model-in-the-loop simulation	30
7.3 Processor-in-the-Loop simulation	31
7.4 Basin testing	32
7.4.1 Safety - hazards and measures	32
7.4.2 Student controller implementation	33
7.4.3 Ship launching procedure - before sailing	35
7.4.4 Deploy control system	36
7.4.5 Ship docking procedure - after sailing	37
8 CS Saucer	38
9 ROV Neptunus	39
10 Simulation and control with cRIO	40
10.1 Simulink model adaptation and compilation	40
10.1.1 Modeling	40
10.1.2 Model configuration	42
10.1.3 Build	43
10.2 Simulation configuration	45
10.2.1 Project creation	45
10.2.2 System setup	46
10.2.3 Create computer interface	47
10.3 Deployment and simulation	50
10.3.1 Run	50
10.3.2 User interface side data logging	50
10.3.3 Stop	51
10.3.4 FTP data retrieval	51
III Laboratory Staff Guide	53
11 HIL lab setup	54
11.1 cRIO	54
11.1.1 Ethernet ports	54
11.1.2 Update cRIO software	55
11.1.3 Create FPGA target and XML	60
11.1.4 Installing custom device driver	73
11.2 Raspberry Pi	76
11.2.1 Raspbian installation and setup	76
11.2.2 Sixaxis installation and configuration	80
11.3 Laptop	83
12 Cybership Enterprise 1	84
12.1 Actuators	84
12.1.1 Motor control signals	85
12.1.2 Servo control signals	85

12.1.3	Measurements	85
12.2	Control software	86
12.2.1	sixaxis (currently named WL_joystick) custom device	86
12.2.2	QTM (currently named Oqus) custom device	86
12.2.3	QTM2SI	86
12.2.4	ctrl_sixaxis2thruster control module	87
12.2.5	ctrl_sixaxis2force control module	87
12.2.6	ctrl_DP_basic control module	88
12.2.7	ctrl_student control module	88
12.2.8	switch module	89
12.2.9	uao2pwm module	89
12.2.10	FPGA interface	90
12.3	Qualisys body	91
13	Qualisys motion capture system	92
13.1	Start server	92
13.2	Aquire body	92
IV	Miscellaneous	94
A	HIL lab and MC lab device network addresses	95
B	Checklist	96
C	ROV control modes	97
C.1	ROV dynamics	97
C.1.1	6 DOF model	97
C.1.2	Thruster configuration	98
C.2	ROV control modes	99
C.2.1	Direct Thruster Control	100
C.2.2	Direct Body Motion Control	100
C.2.3	Auto Control	101
C.2.4	Indirect Motion Control	101
C.2.5	Marine Operations Control	102
D	Personel and literature	103
D.1	Points of contact	103
D.2	Publications	103
D.3	Specialization projects and master theses	104
D.4	Other	104
E	Maintenance	105
F	Suppliers	106
G	To do list	107

Nomenclature

BT	bow thruster
cRIO	compact reconfigurable input/output real-time embedded industrial controller by National Instruments
CS	cyber ship
CSE1	CS Enterprise I
CSS	CS Saucer
DP	dynamic positioning
ESC	electronic speed controller
FPGA	field-programmable gate array
HIL	hardware-in-the-loop
IO	input/output
MC Lab	marine cybernetics laboratory
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).

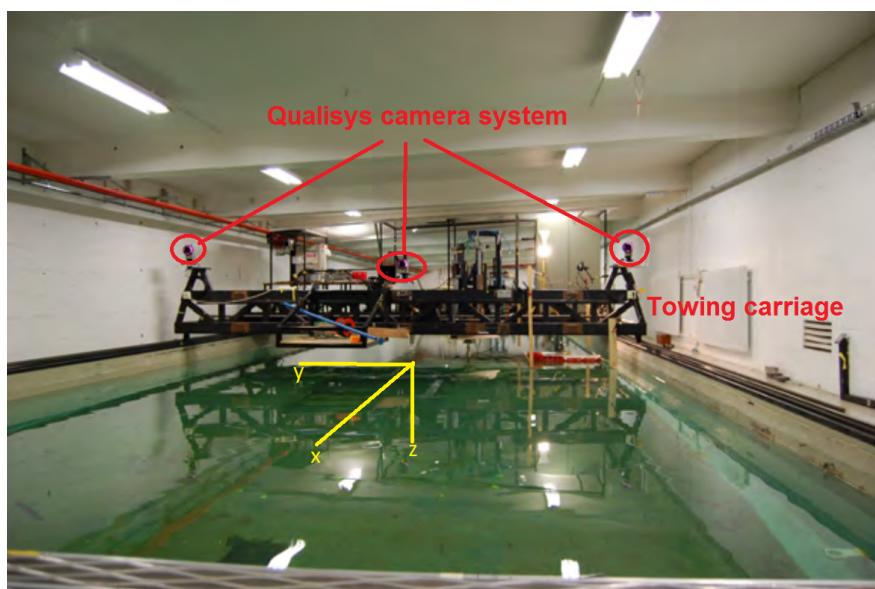


Figure 1.1: Marine cybernetics laboratory basin

	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 surface and underwater vessels.

The cyber ship class, with ship prefix CS, consists of 6 vessels:

- CS Inocean Cat I Arctic Drillship,
- CS Saucer,
- CS Enterprise I,
- Cybership III,
- Cybership II, and
- Cybership I.

Also, there lab holds an underwater craft,

- ROV Neptunus,

and a semi submersible drilling rig

- CyberRig.

1.2.1 CS Inocean Cat I Arctic Drillship

Jon

1.2.2 CS Saucer

Klipp og lim Tor

1.2.3 ROV Neptunus

Klipp og lim Jostein
et. al



Figure 1.2: CS Enterprise I

1.2.4 CS Enterprise I

The CSE1, depicted in Figure 1.2,

hva bruker vi den til

1.2.4.1 Hull

tug boat model.

1.2.4.2 Actuators

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

1.2.4.3 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 computer,
- three electronic speed controllers (ESC), and
- four servos.

The operator interfaces the system by

- laptop, and
- a Sony Sixaxis wireless gamepad for PlayStation 3.

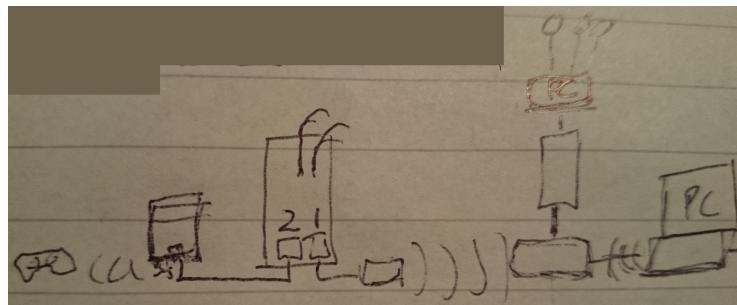


Figure 1.3: CSE1 communication diagram

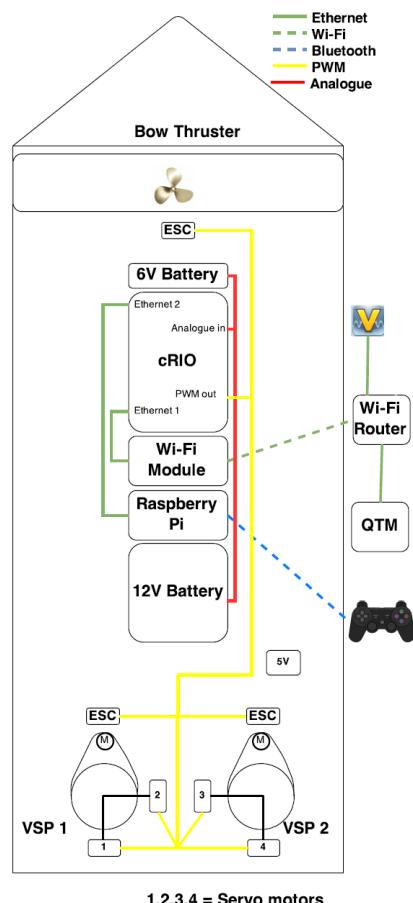


Figure 1.4: CSE1 signal paths

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 paired¹.

RPi receives Sixaxis data through the USB dongle and forwards it through its TCP/IP² 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.

1.2.4.4 Power

See Figure 1.5.

¹One-time pairing procedure described in Appendix 11.2.2.2.

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

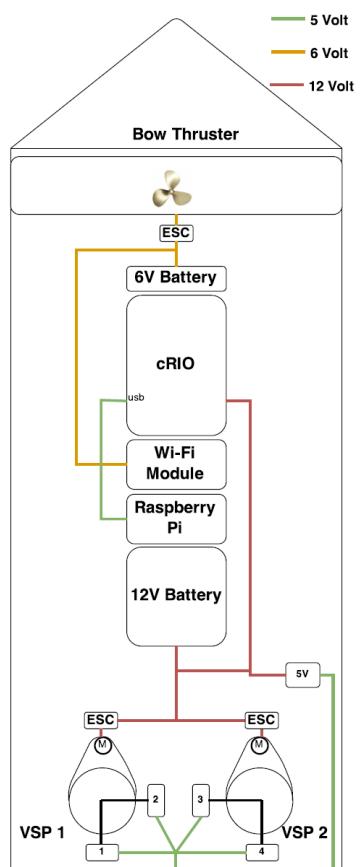


Figure 1.5: CSE1 power system

Chapter 2

Control system development philosophy

As the complexity of marine vessels and operations grows, the need for thorough testing and verification of the vessel real-time control and monitoring systems increases. More advanced integrated functionality relies on many separately designed control and monitoring systems to cooperate on performing common tasks. Regular software simulations cannot cover all aspects of this complexity.

Through steps-wise verification and validation at different levels of fidelity, errors can be discovered at earlier stages thus lowering the total development cost.

In the case of the MC lab, users qualify their experimental setups before the assigned laboratory time. This reduces debugging time, improves tuning of parameters and test scenarios, thereby increasing efficiency and maximizing the outcome of the experimental work.

2.1 Development Steps

Marine cybernetics deals with control engineering for the vessel mechatronic systems which again interact with the environment. In this text, “the controller” refers to the designed control software and “the plant” to the combination of the mechatronic system and the environment.

2.1.1 Model-in-the-Loop

2.1.1.1 Principle

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

2.1.1.2 Aim

Develop control strategies . Test principles.

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

The controller is coded in the final language, such as C or C++, and connected to the 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

The controller is 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, such as insufficient computing resources 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 subsystems.

Ensure a high level of robustness and quality.

2.1.5 Scale test

2.1.6 Full scale

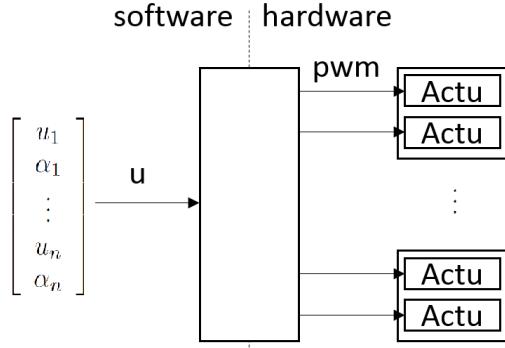


Figure 2.1: Individual actuator control

2.2 Control Modes

It is favorable to allow for five main control modes:

- Stop all actuators
- 1. Individual actuator control
- 2. Generalized force control
- 3. Regulation
- 4. Automatic/operations

In addition, sub-modes allow more functionality.

2.2.1 Individual actuator control

The most basic mode allows controlling each thruster separately. Inputs are typically normalized force $u = [-1, 1]$, angle $\alpha = [-\pi, \pi]$, and sometimes normalized rotational speed $\omega = [-1, 1]$. The software computes the corresponding physical signal, for instance a pulse width manipulated (PWM) signal as illustrated in Figure 2.1.

The user interface may be through gamepad, computer, tablet, etc.

Implementation details are discussed in Appendix C.2.1.

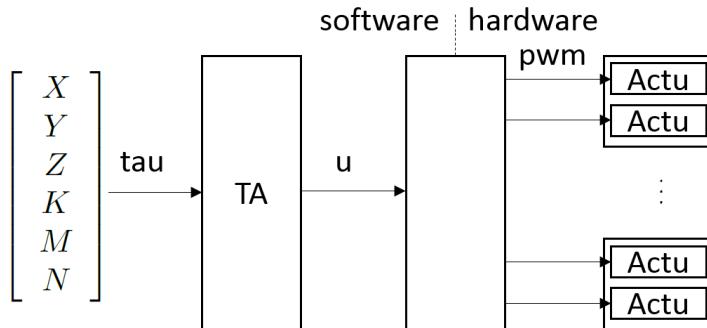


Figure 2.2: Generalized force control

2.2.2 Generalized force control

Thrust allocation allows input of the desired generalized force, as seen in Figure 2.2. For six degrees of freedom (6 DOF) control the input is

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

For surface craft, 3 DOF are typically considered:

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

The user interface may be through gamepad, computer, tablet, etc. The appropriate reference frame depends on the application.

2.2.2.1 Body frame

Most commonly, the desired thrust is given in the vessel-fixed body frame. This is the intuitive setup for an on-board operator.

Implementation details are discussed in Appendix C.2.2.1.

2.2.2.2 Inertial frame

For remote operation, it may be suitable to input the force with regard to the inertial frame, rather than the vessel orientation.

Implementation details are discussed in Appendix C.2.2.2.

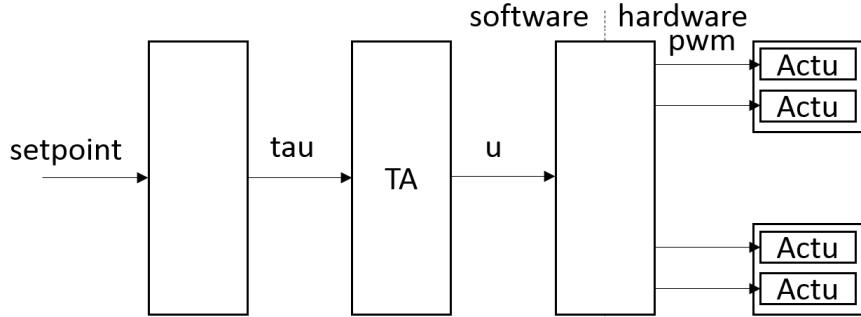


Figure 2.3: Regulation

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

Table 2.1: A selection of regulation modes

2.2.2.3 User frame

When the operator has eye contact with the vessel, it may be suitable to specify the force with respect to the line of sight between the operator and craft.

Implementation details are discussed in Appendix C.2.2.3.

2.2.3 Regulation

Maintaining a given value in one or several DOFs under the influence of disturbances is the basic automatic control mode. The given value is called setpoint, as illustrated in Figure 2.3. Typical sub-modes are listed in Table 2.1. Reference filters for changing setpoints may or may not be included.

The user interface typically allows inputting the setpoint value directly, for instance on a computer or tablet. Alternatively, the setpoint may be translated through gamepad.

Further details are discussed in Appendix C.2.3.

2.2.4 Marine Operations Control

The more complex control modes, typically combined from several of the different submodes, give automatic functions that are important for different marine operations. Input varies depending on the operation. It may be maps or waypoints, as in Figure 2.4.

Further details are discussed in Appendix C.2.5.

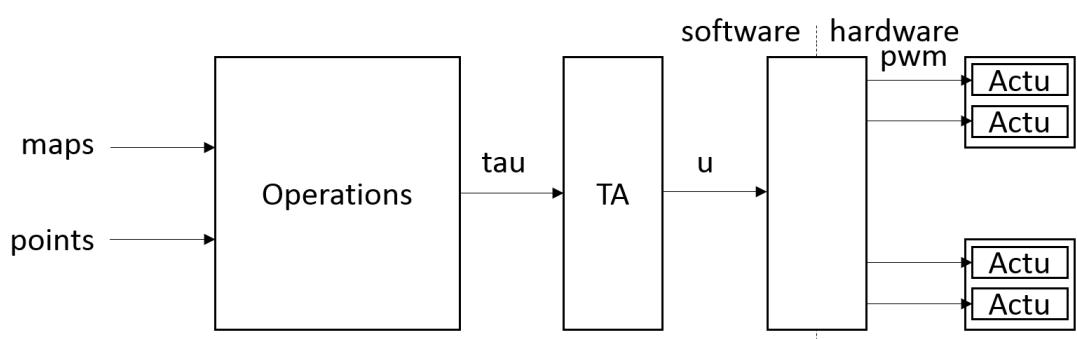


Figure 2.4: Marine Operations Control

Part II

Laboratory user guide

Chapter 3

Qualisys motion capture system

Chapter 4

Towing carriage

In figure 4.1, a picture of the towing wagon is presented.

Control of the wagon is done on board; see the red circle in figure 4.1. For our purposes, to perform towing tests of CS Enterprise 1, only a certain computer commands are used. Further, in figure 4.2, a close up of the computer is shown. An explanation of the different commands, used for our purposes, is presented below.

4.1 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 4.1: Towing wagon in the MC-lab. Control of the wagon is done using the computer marked in red.

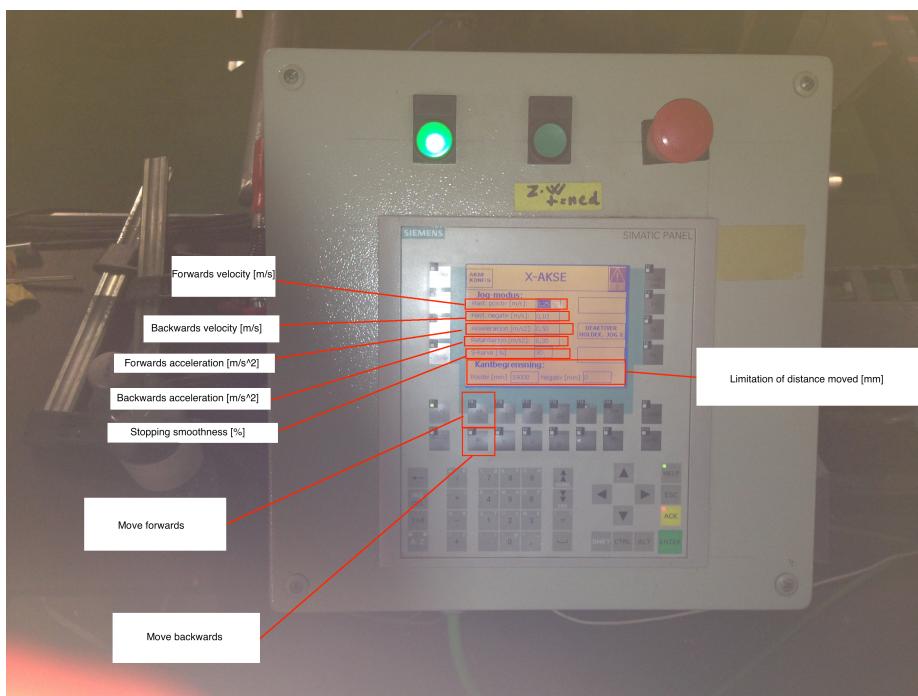


Figure 4.2: Computer to operate the towing wagon

4.2 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.

4.3 Note

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

Chapter 5

Wave generator

Kanskje kan Astrid skrive om grunnleggende bruk, og om hvordan hun hacker bølgeserier

Chapter 6

CS Inocean Cat I Arctic Drillship

6.1 Mathematical model

Chapter 7

CS Enterprise I

7.1 Mathematical model

The proposed control design model is

$$\dot{\eta} = R(\psi) \nu \quad (7.1)$$

$$\dot{\nu} = M^{-1}(-C(\nu)\nu - D(\nu)\nu + \tau) \quad (7.2)$$

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 7.1: 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 7.2: CSE1 damping parameters

- the coriolis and centripetal matrix is

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

- 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

$$d_{11}(u) = -X_u - X_{|u|u}|u| - X_{uuu}u^2 \quad (7.3)$$

$$d_{22}(v, r) = -Y_v - Y_{|v|v}|v| - Y_{vvv}v^2 - Y_{|r|v}|r| \quad (7.4)$$

$$d_{23}(v, r) = -Y_r - Y_{|v|r}|v| - Y_{|r|r}|r| - Y_{rrr}r^2 \quad (7.5)$$

$$d_{32}(v, r) = -N_v - N_{|v|v}|v| - N_{vvv}v^2 - N_{|r|v}|r| \quad (7.6)$$

$$d_{33}(v, r) = -N_r - N_{|v|r}|v| - N_{|r|r}|r| - N_{rrr}r^2 \quad (7.7)$$

The rigid body inertia and hydrodynamic added mass parameters are given in Table 7.1.

The hydrodynamic damping parameters are given in Tables 7.2 and 7.3.

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 7.3: CSE1 damping parameters

7.2 Model-in-the-loop simulation

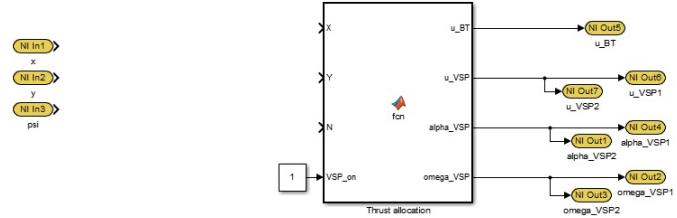


Figure 7.1: CSE1 `ctrl_student.slx` including thrust allocation

7.3 Processor-in-the-Loop simulation

Implementation of the student controller is done in the `ctrl_student.slx` Simulink template, depicted in Figure 7.1. Detailed implementation steps are given in Section 7.4.2.

The generic control system consists of several Simulink modules, the details of which are given in Appendix 12.2, a FPGA driver, described in Appendix 11.1.3, and two custom device drivers.

7.4 Basin testing

7.4.1 Safety - hazards and measures

7.4.1.1 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.

7.4.1.2 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.

7.4.2 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 10.1.1.
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 10.1.2.
The remaining configuration, such as target selection is preselected in the file.
 - (c) Compile the model as described in Section 10.1.3. 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 7.2, 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 10.2.3, for your controller in control screen 4: `ctrl_student`. Figure 7.3 shows control screen 4 before customization.

¹Other paths are possible but require changing all VeriStand project paths.

²Not `CSE1.nivssdf`, since not only the system definition should be altered.

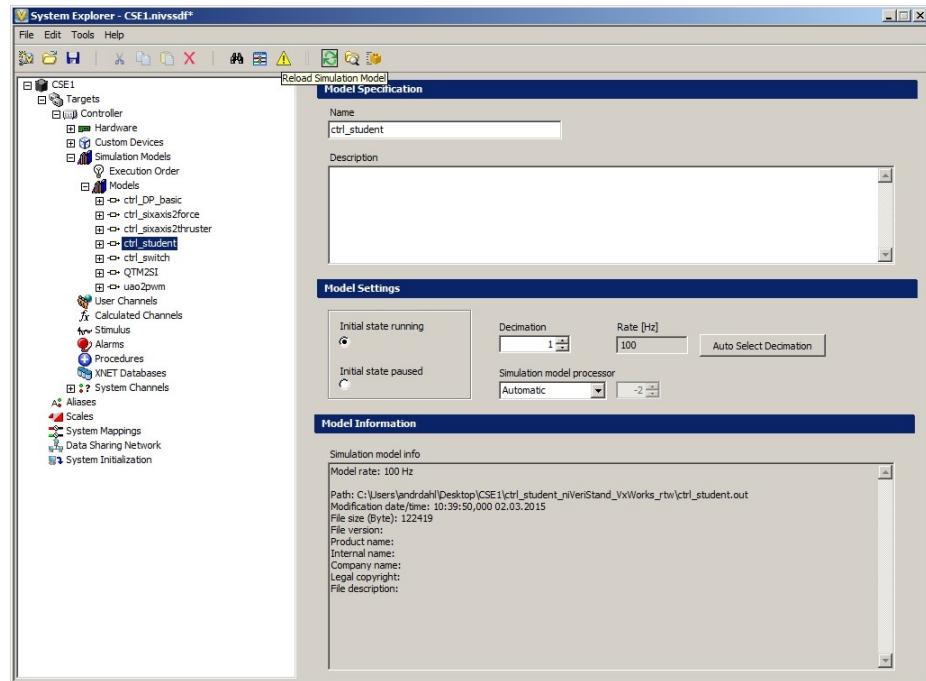


Figure 7.2: CSE1 Veristand Project simulation models

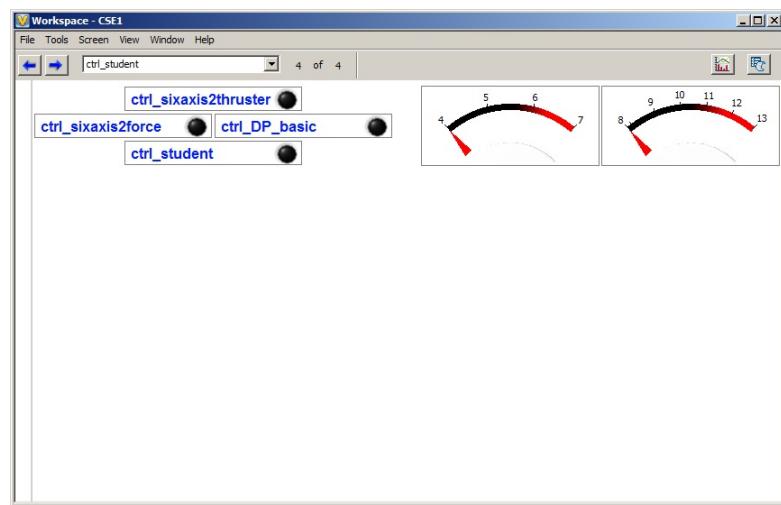
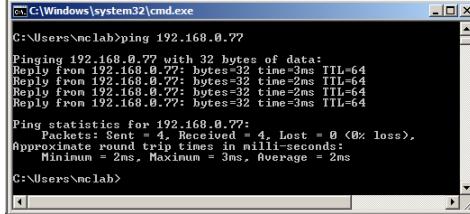


Figure 7.3: CSE1 Veristand Project ctrl_student workspace

7.4.3 Ship launching procedure - before sailing

7.4.3.1 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 pole³.
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 successfully 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 secondary 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.



```
C:\Windows\system32\cmd.exe
C:\Users\mclab>ping 192.168.0.77

Pinging 192.168.0.77 with 32 bytes of data:
Reply from 192.168.0.77: bytes=32 time=3ms TTL=64
Reply from 192.168.0.77: bytes=32 time=2ms TTL=64
Reply from 192.168.0.77: bytes=32 time=2ms TTL=64
Reply from 192.168.0.77: bytes=32 time=3ms TTL=64

Ping statistics for 192.168.0.77:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 3ms, Average = 2ms

C:\Users\mclab>
```

Figure 7.4: Ping, successful access to CSE1

7. Verify laptop access: ping the CSE1 IP in the command prompt, as in Figure 7.4. 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.

7.4.3.2 Positioning system

³The connection order of the wires should not matter. However, experiences favor 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 7.4: Generic control modes

7.4.4 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 7.4.

7.4.5 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 8

CS Saucer

k

Chapter 9

ROV Neptunus

k

Chapter 10

Simulation and control with cRIO

10.1 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. Mostly¹ 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 11.3.

10.1.1 Modeling

10.1.1.1 Input and output

In order for the model to interact with VeriStand, special input and output blocks must be added to the block diagram². These are found in the Simulink Library Browser under NI VeriStand Blocks.

10.1.1.2 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

¹It 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.

²Ordinary 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.

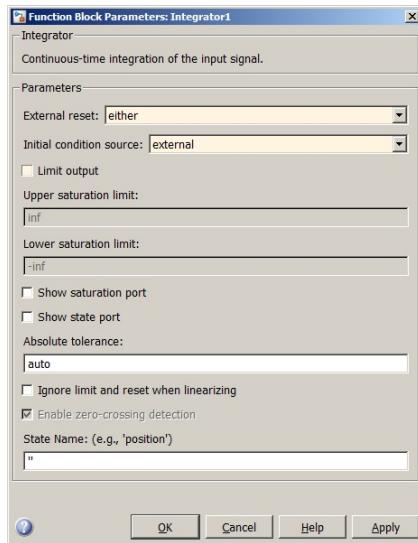


Figure 10.1: Integrator function block parameters

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 10.1.

10.1.1.3 Real-time data logging

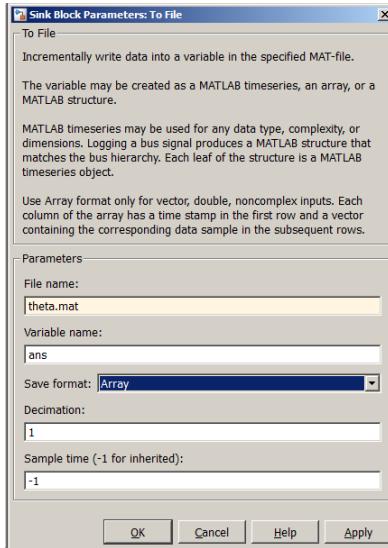


Figure 10.2: To File block parameters

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

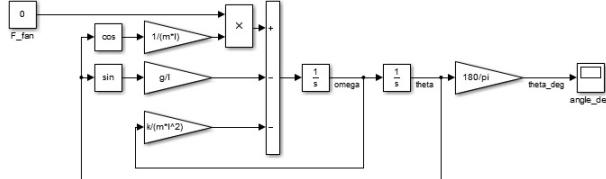


Figure 10.3: Simulink model for offline simulation

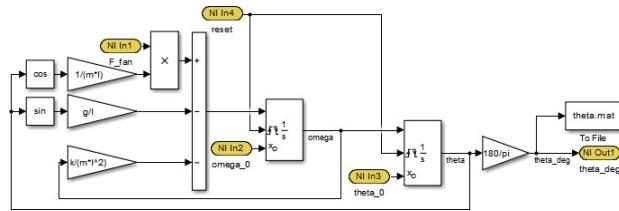


Figure 10.4: Simulink model for adjusted for compilation

Example: For a simple pendulum, $\dot{\omega} = -\frac{g}{l} \sin(\theta) - \frac{k}{ml^2} \omega + \frac{F_{fan}}{ml} \cos(\theta)$, the offline simulation block diagram could look as Figure 10.3. Figure 10.4 shows the same system adapted for VeriStand input, including reset and initial conditions, and output. The VeriStand blocks are yellow. ω_0 and θ_0 are ports corresponding to the initial conditions $(\omega(0), \theta(0))$. The integrators take these values whenever reset is rising or falling.

10.1.2 Model configuration

The code generation toolbox compiles the Simulink diagram to an output shared library in *.out format³. 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 10.5. 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. second⁴.

³The *.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.

⁴This can also be achieved by use of decimation, as described in Section 10.2.2.

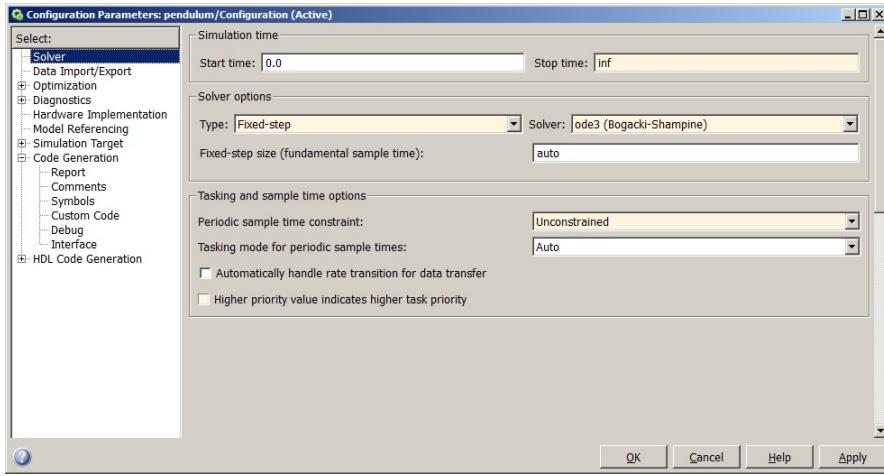


Figure 10.5: Simulink configuration parameters - solver

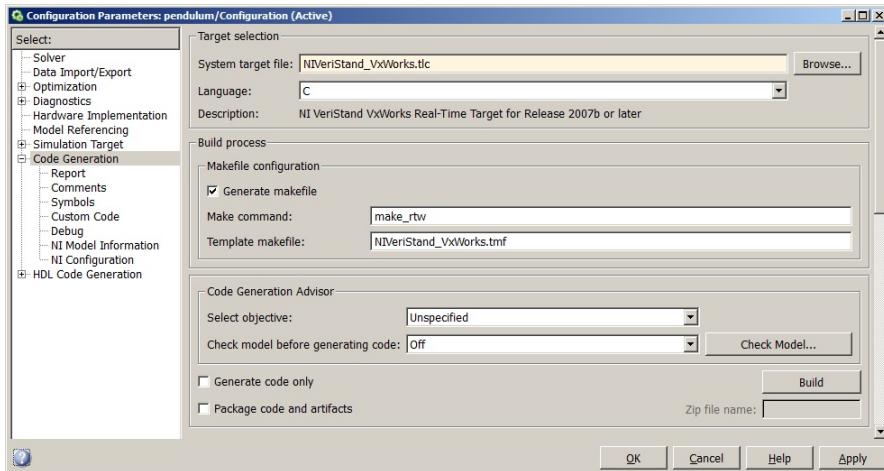


Figure 10.6: Simulink configuration parameters - target selection

The correct target file should be selected depending on the target device. Select `NIVeriStand_VxWorks.tlc` for VxWorks targets⁵, such as cRIO-9024, as in Figure 10.6.

The WindRiver GNU Toolchain must be present in the folder specified under NI Configuration, as in Figure 10.7.

10.1.3 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

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

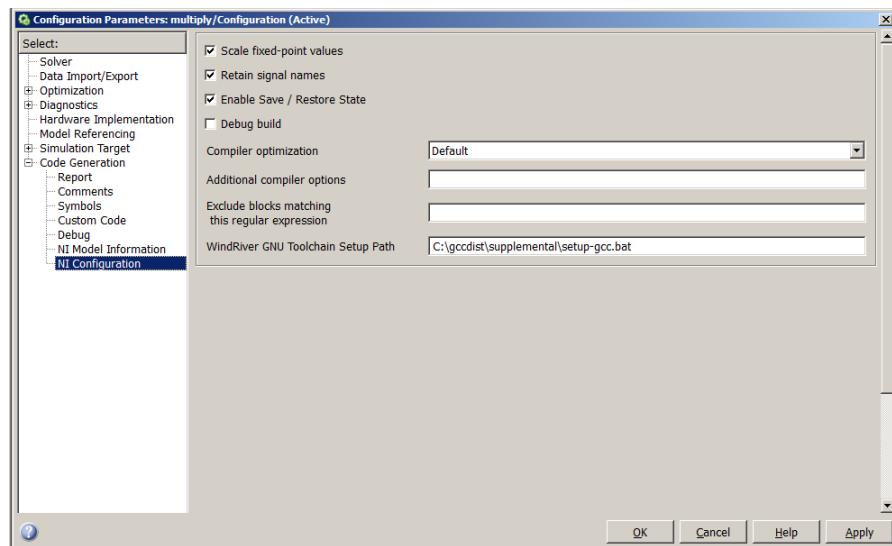


Figure 10.7: Simulink model configuration - NI configuration

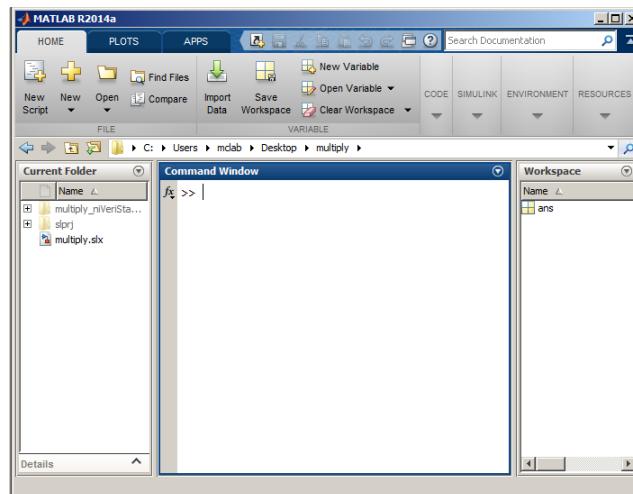


Figure 10.8: MATLAB console

Figure 10.8, before compiling. The build subfolder name is [simulink model name]_niVeriStand_VxWorks_rtw.

The build is done 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.

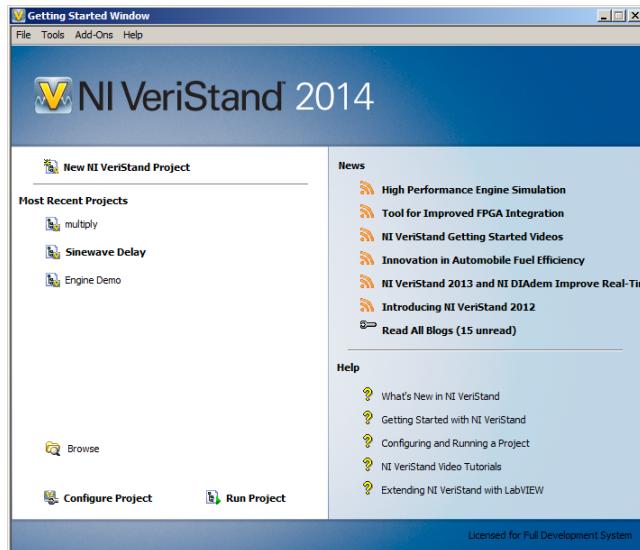


Figure 10.9: VeriStand start screen

10.2 Simulation configuration

Simulations are set up, deployed and interfaced through VeriStand. Figure 10.9 shows the start screen. Already configured projects can be run directly from here, or reconfigured.

10.2.1 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

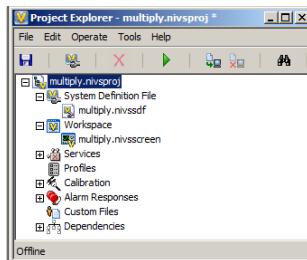


Figure 10.10: VeriStand Project Explorer

in given location and opens the Project Explorer, as in Figure 10.10. In this section, the example project name is multiply.

10.2.2 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.

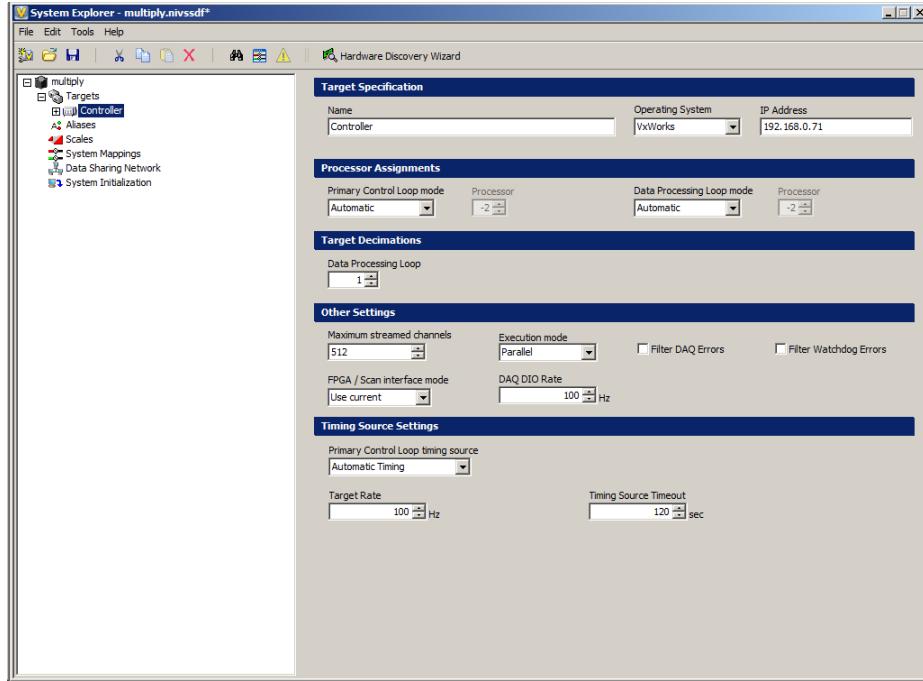


Figure 10.11: VeriStand - System Explorer - Controller

1. Set the correct controller operating system and IP address, as in Figure 10.11. 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 10.12. Browse to the output of the Simulink compilation, as seen in Figure 10.13. 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 device⁶. Figure 10.14 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 10.15.
5. Save and close to return to the Project Explorer.

⁶If the required device is not present, refer to the device driver installation instructions in Section 11.1.4.

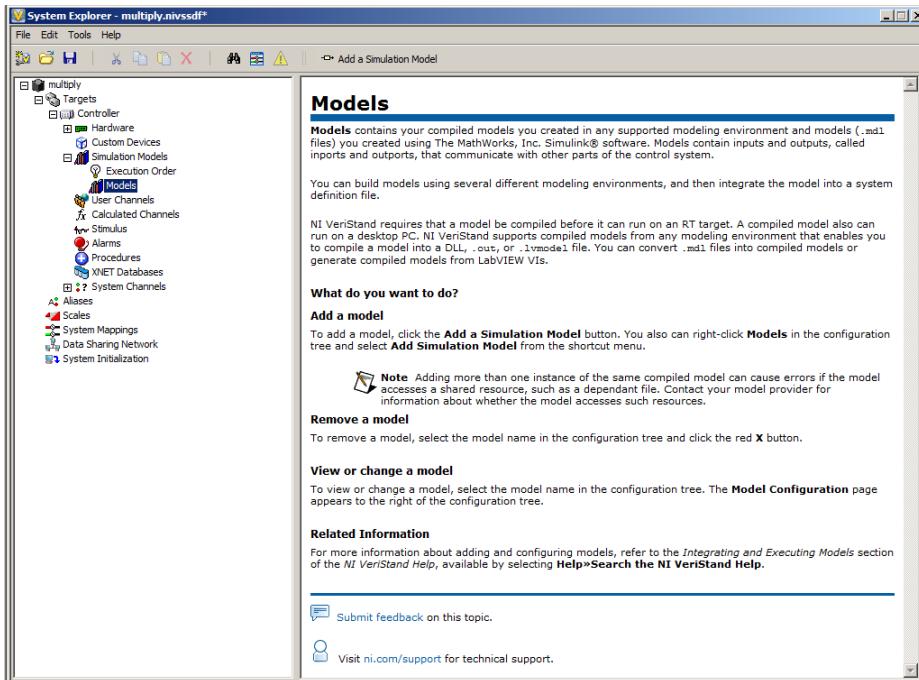


Figure 10.12: VeriStand - System Explorer - Models

10.2.3 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.

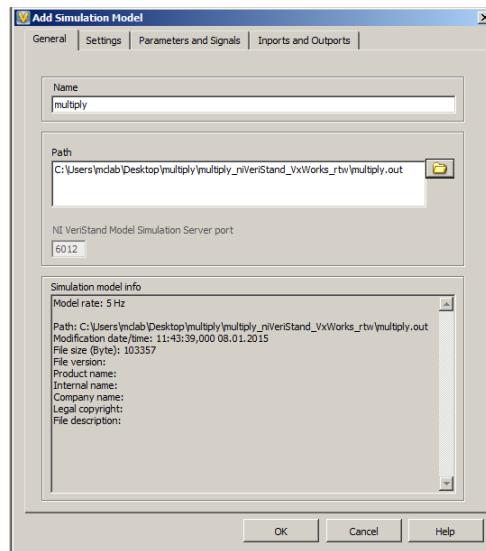


Figure 10.13: VeriStand - System Explorer Model

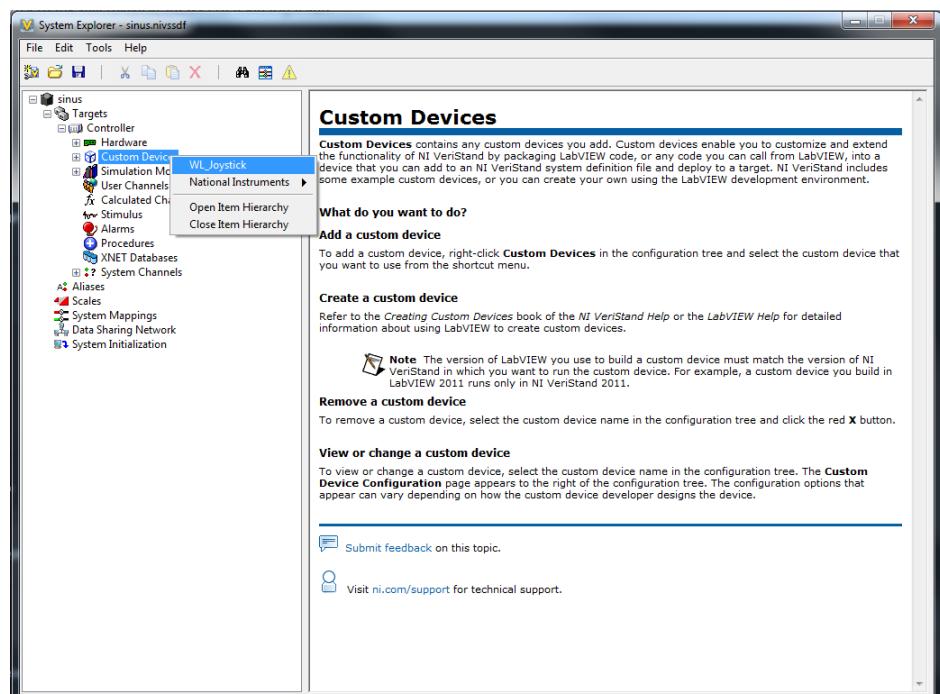


Figure 10.14: Custom device selection

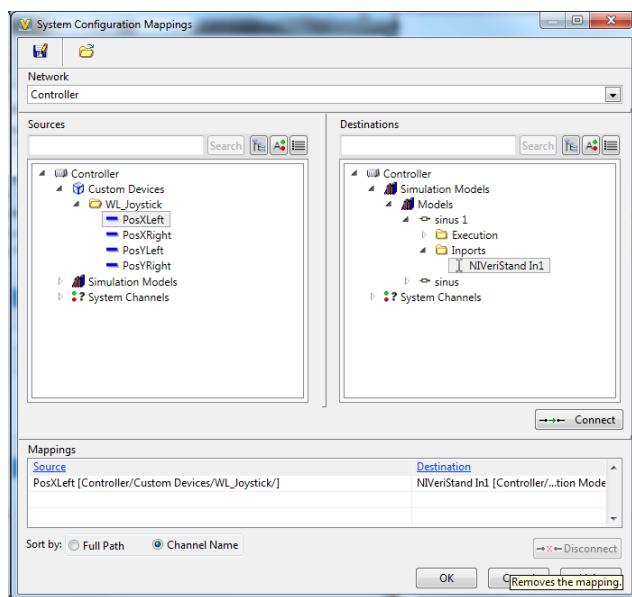


Figure 10.15: VeriStand System Configuration Mappings

10.3 Deployment and simulation

10.3.1 Run

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

10.3.2 User interface side data logging

For reliability, it is recommended to log data directly on the cRIO during simulation, as described in Section 10.1.1.3. It is also possible to log via the laptop user interface.

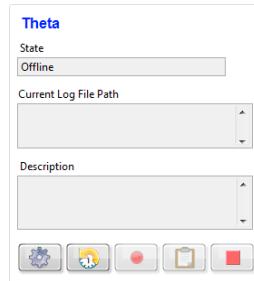


Figure 10.16: Logging Control

A Logging Control, as seen in Figure 10.16, must be added to the workspace to export data from the simulation. The control is added as described in Section 10.2.3.

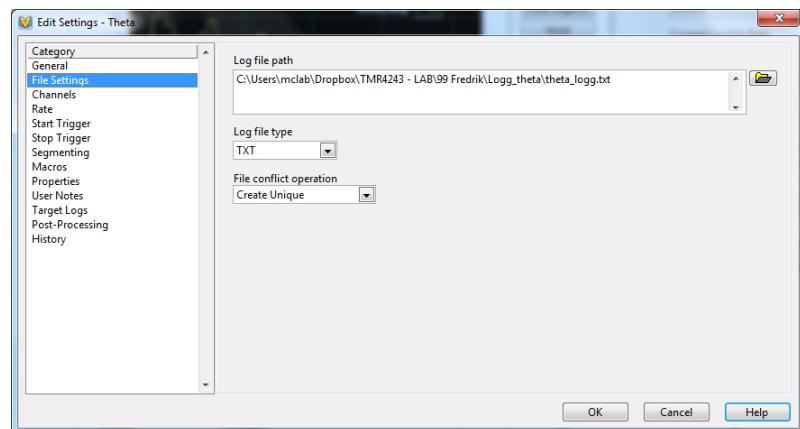


Figure 10.17: Logging Control file settings

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 10.17. Under Channels, the desired channels can be selected and added, as in Figure 10.18.

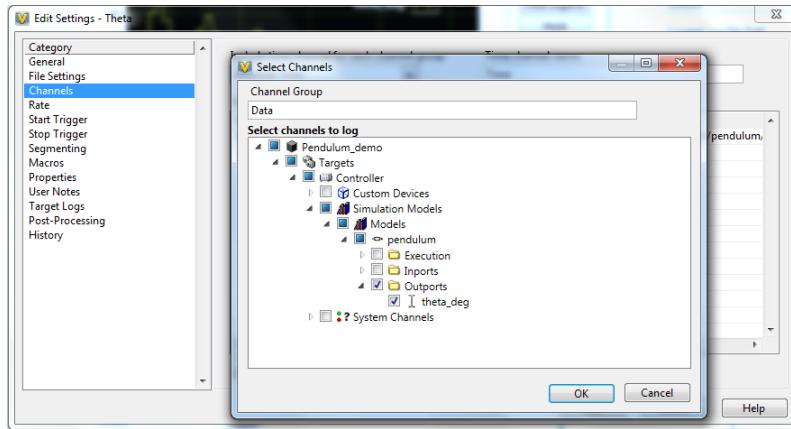


Figure 10.18: Logging Control add channel

10.3.3 Stop

button

10.3.4 FTP data retrieval

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

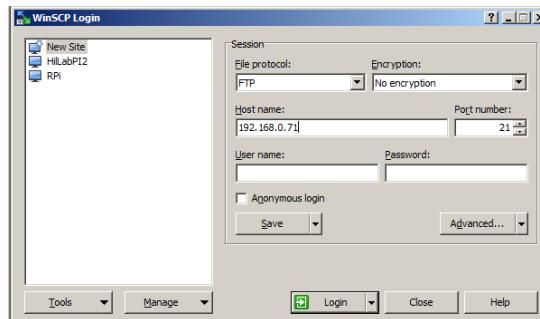


Figure 10.19: WinSCP login

To connect to the cRIO, the correct IP must be specified, as in Figure 10.19. 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 10.20. Data is transferred to the laptop by drag and drop to the desired location in the left pane.

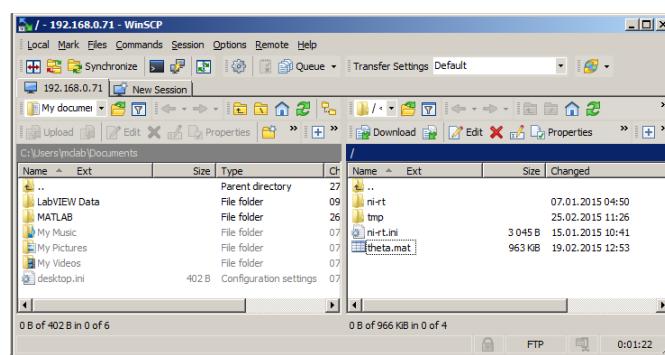


Figure 10.20: WinSCP

Part III

Laboratory Staff Guide

Chapter 11

HIL lab setup

11.1 cRIO

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

11.1.1 Ethernet ports

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

11.1.1.1 Primary

Set fixed IP, set fixed IP on HIL-computers

11.1.1.2 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*

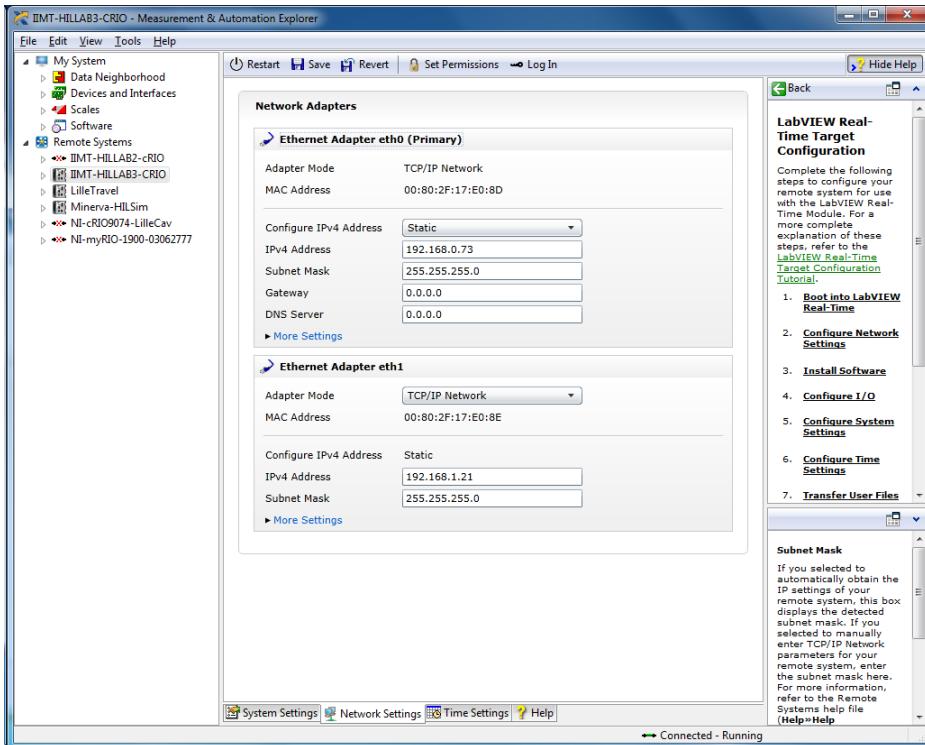


Figure 11.1: NI MAX - Network Settings

11.1.2 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.

11.1.2.1 Update

1. Open NI Max
2. Find your cRIO on the left hand side and click it
3. Click Software, and then Add/Remove Software located on the top pane, see Figure 11.2
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 11.3
5. Click next without making any changes
6. Click next without making any changes
7. Wait for the installation to finish and the cRIO to reboot

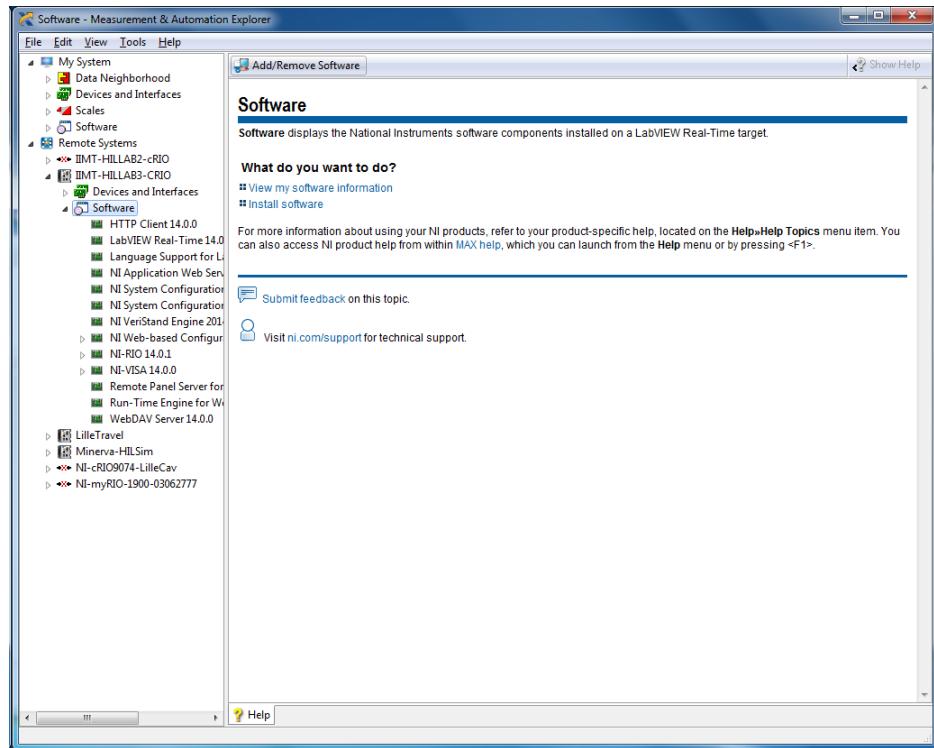


Figure 11.2: NI MAX - Software Update 1

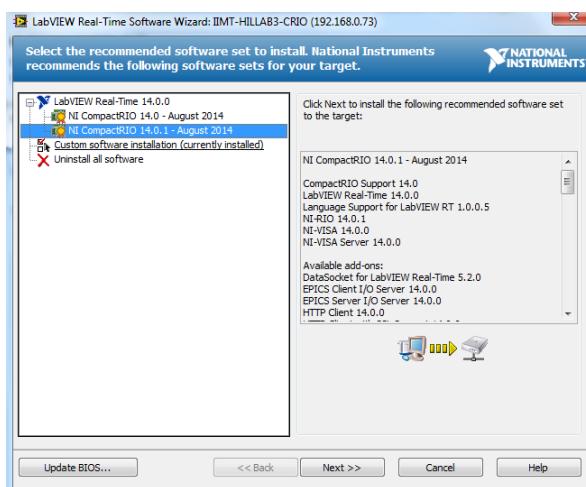


Figure 11.3: NI MAX - Software Update 1

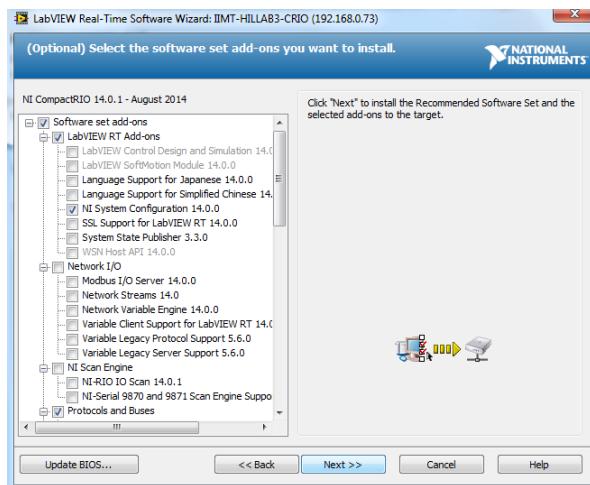


Figure 11.4: NI MAX - Software Update 3

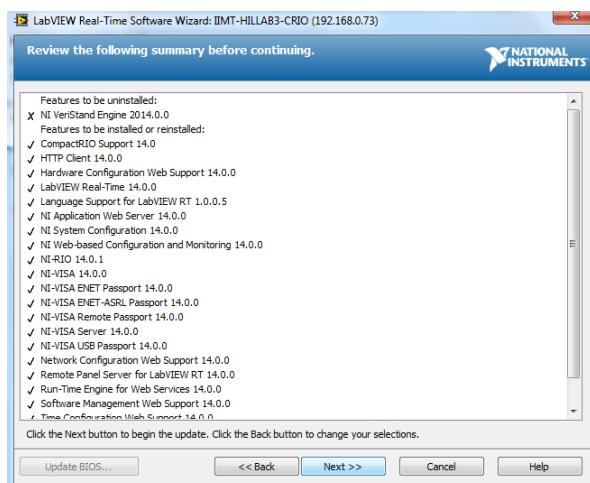


Figure 11.5: NI MAX - Software Update 4

11.1.2.2 NI Veristand Engine

1. Repeat step 1-3 from the previous guide
2. Now you choose Custom Software installation in the menu, see Figure 11.6
3. Ignore the warning, See Figure 11.7
4. Locate NI Veristand Engine 2014 0.0 and click install feature. See Figure 11.8
5. Click your way through the rest of the installation and let the cRIO reboot.

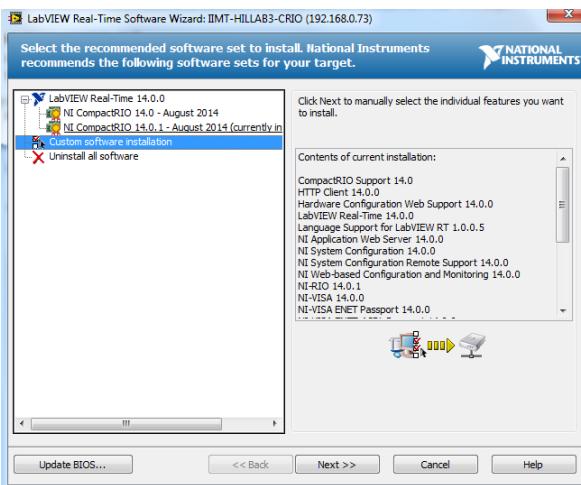


Figure 11.6: NI MAX - NI Veristand Engine installation 1

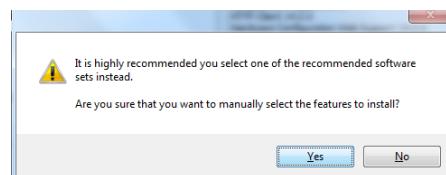


Figure 11.7: NI MAX - NI Veristand Engine installation 1

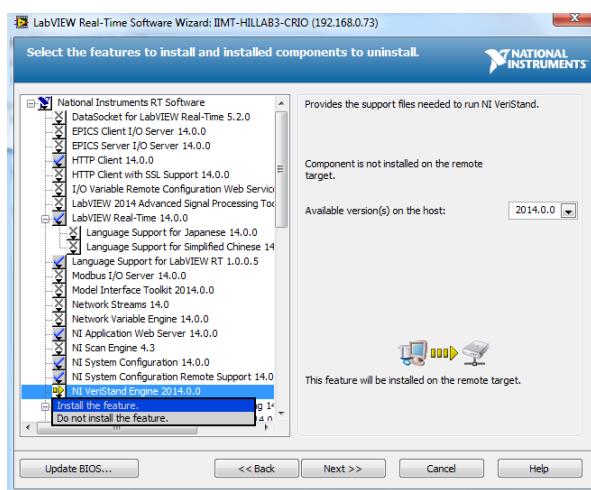


Figure 11.8: NI MAX - NI Veristand Engine installation 1

11.1.3 Create FPGA target and XML

If you do not have 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 installation 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 Reconfigurable 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 recommended. 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 11.18. Select the LabVIEW VI as demonstrated ours 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 11.19. You will now have to redesign this to look like Figure 11.20. 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 your 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 11.25, 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 and the XML file is in the same folder.

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

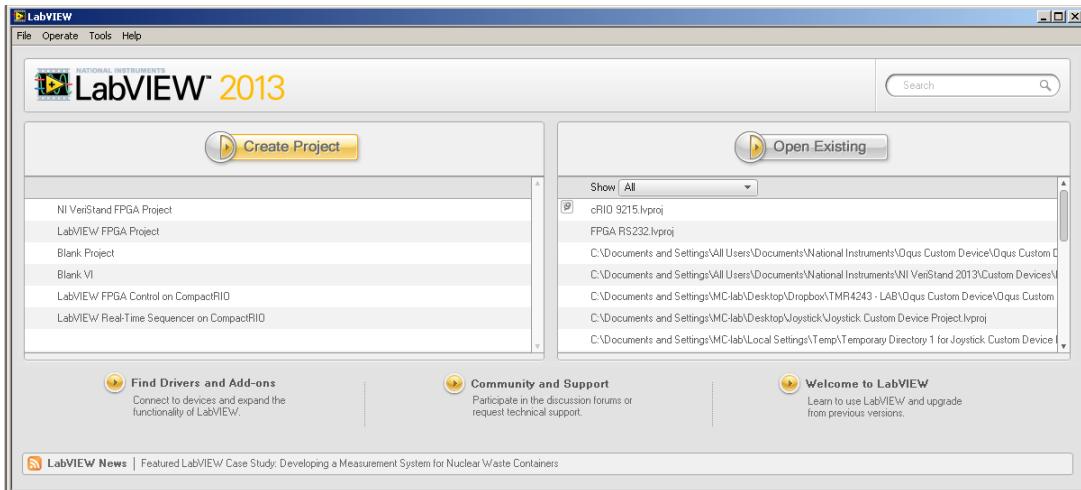


Figure 11.9: Create Labview FPGA target and XML - 1

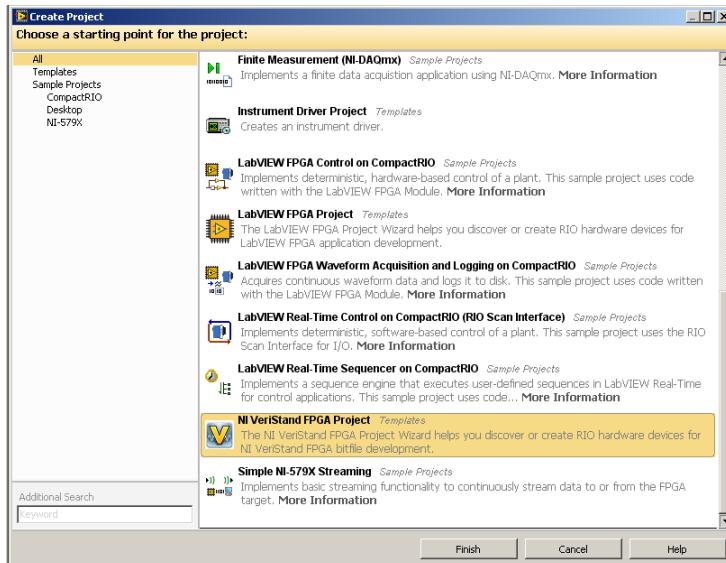


Figure 11.10: Create Labview FPGA target and XML - 2

Veristand FPGA programming

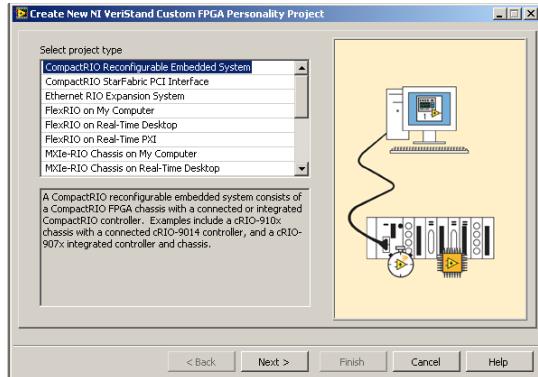


Figure 11.11: Create Labview FPGA target and XML - 3

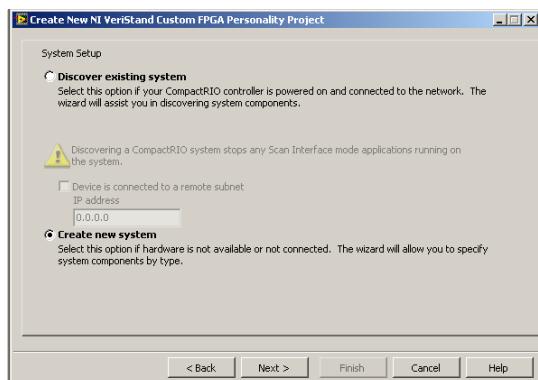


Figure 11.12: Create Labview FPGA target and XML - 4

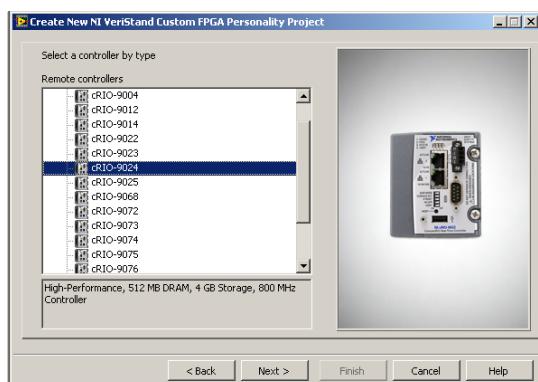


Figure 11.13: Create Labview FPGA target and XML - 5

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.

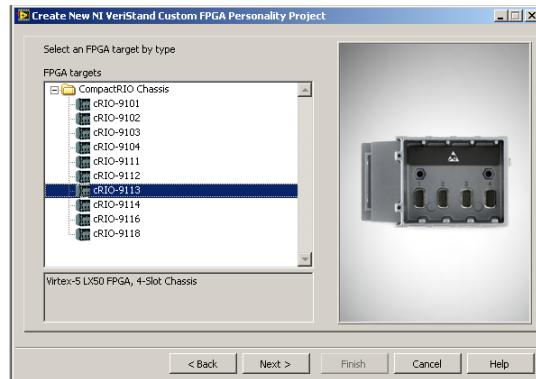


Figure 11.14: Create Labview FPGA target and XML - 6

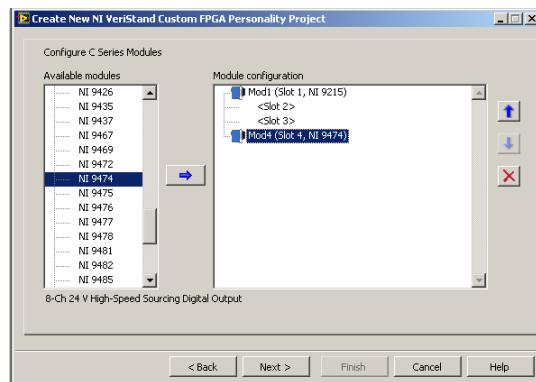


Figure 11.15: Create Labview FPGA target and XML - 7

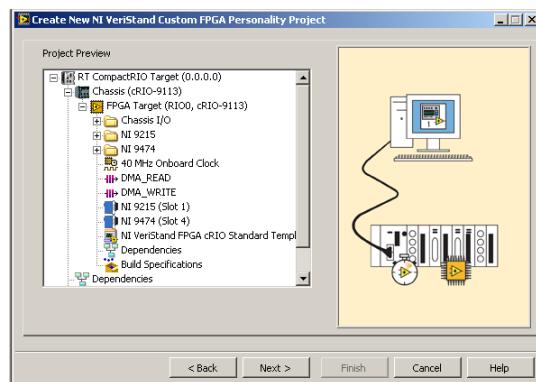


Figure 11.16: Create Labview FPGA target and XML - 8

11.1.3.1 Install in veristand

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

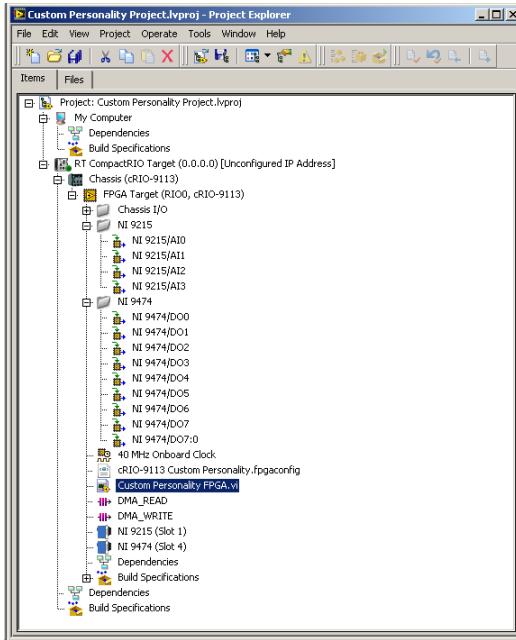


Figure 11.17: Create Labview FPGA target and XML - 9

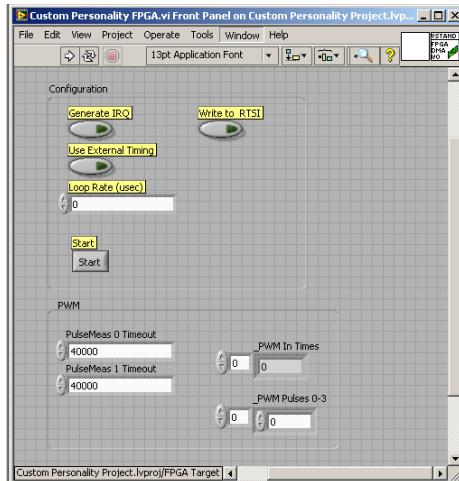


Figure 11.18: Create Labview FPGA target and XML - 10

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 11.26.

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 11.27, 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 Figure11.33.

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

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

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 overfølgende 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

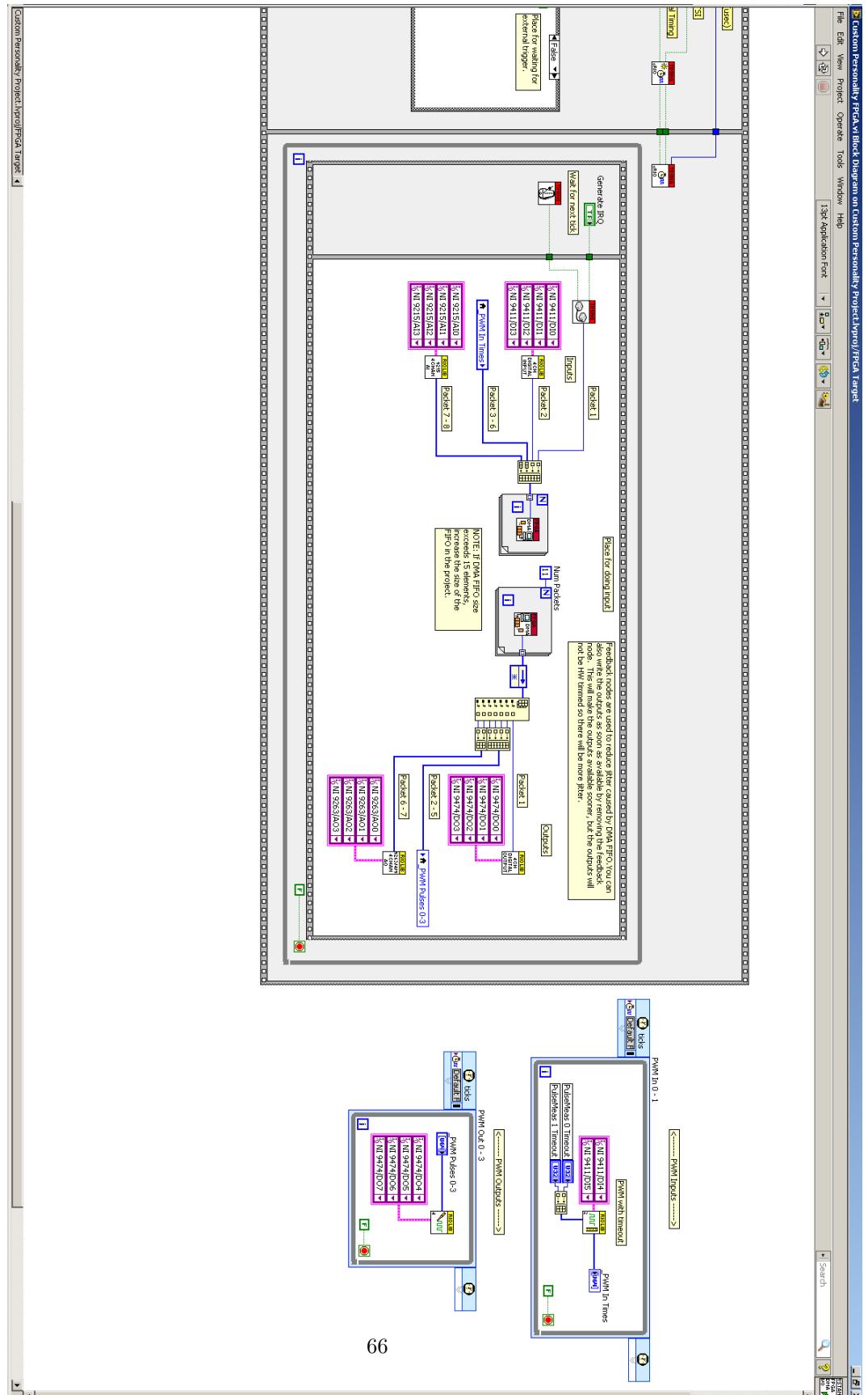


Figure 11.19: Create Labview FPGA target and XML - 11

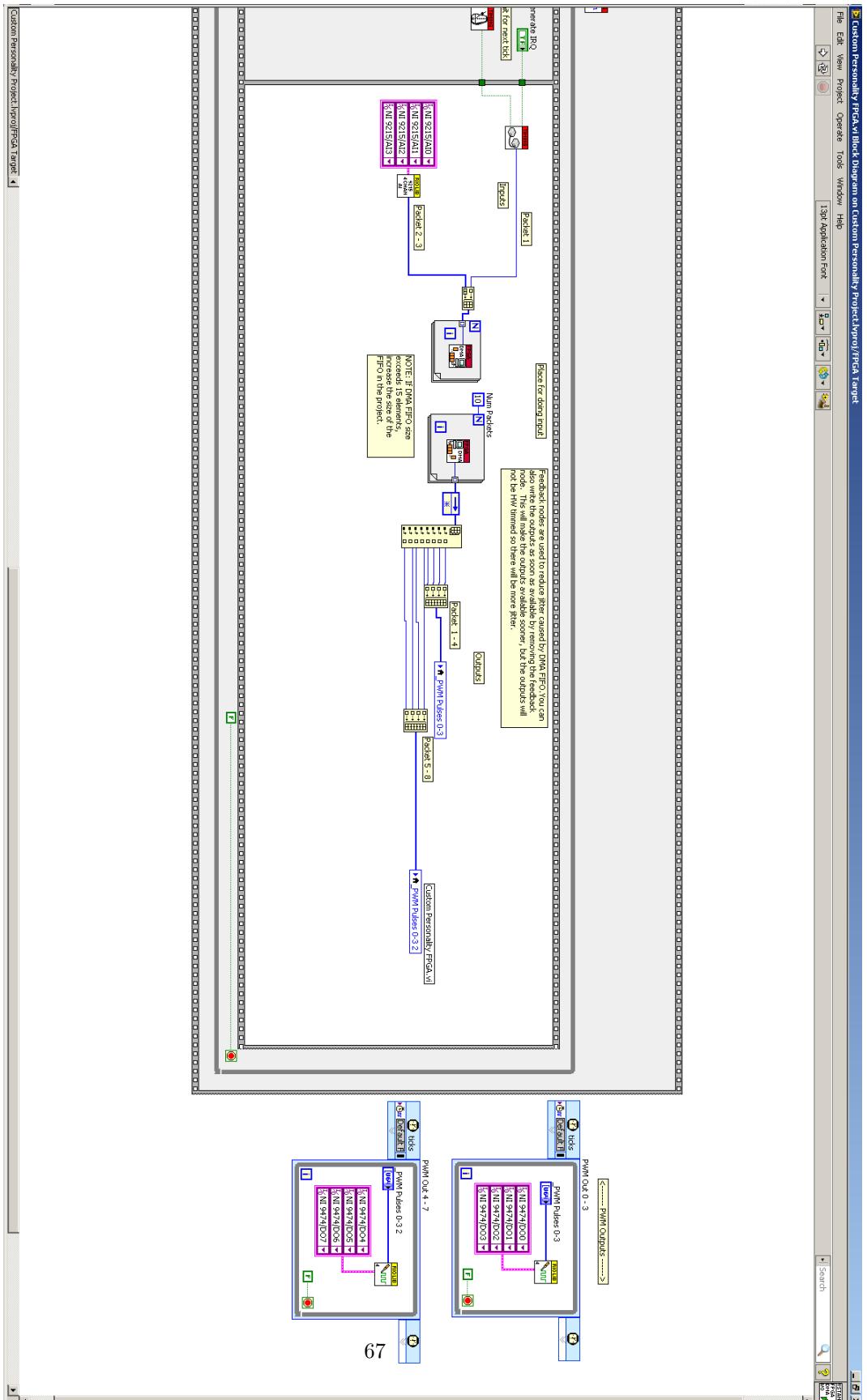


Figure 11.20: Create Labview FPGA target and XML - 12

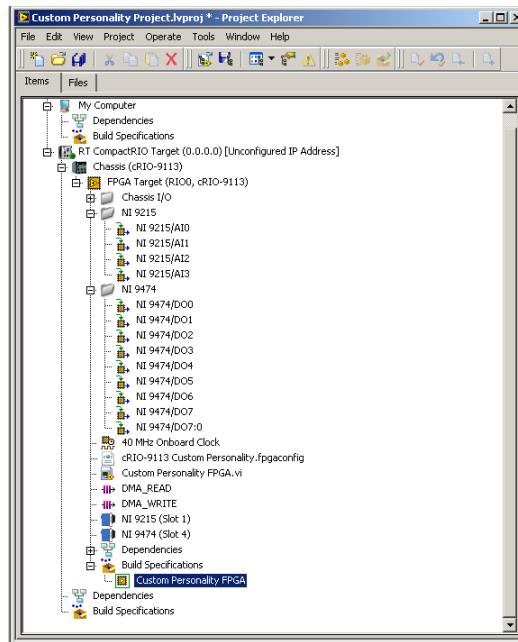


Figure 11.21: Create Labview FPGA target and XML - 13

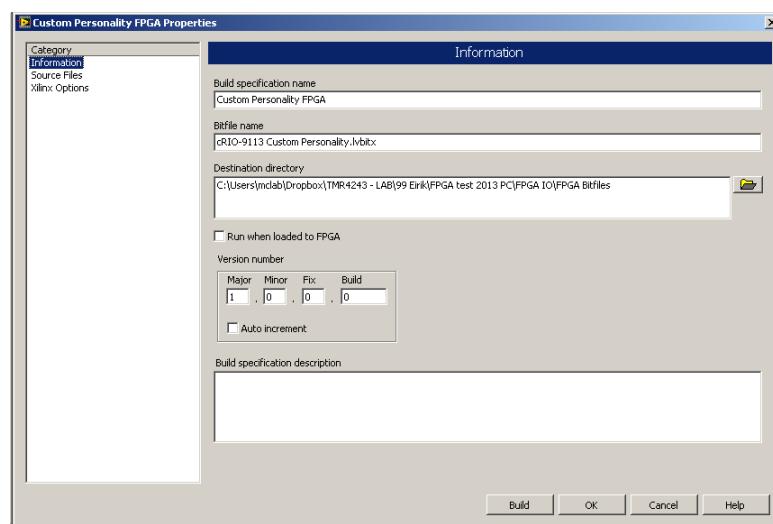


Figure 11.22: Create Labview FPGA target and XML - 14

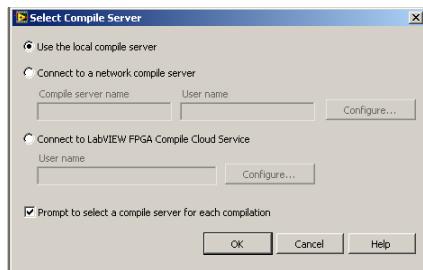


Figure 11.23: Create Labview FPGA target and XML - 15

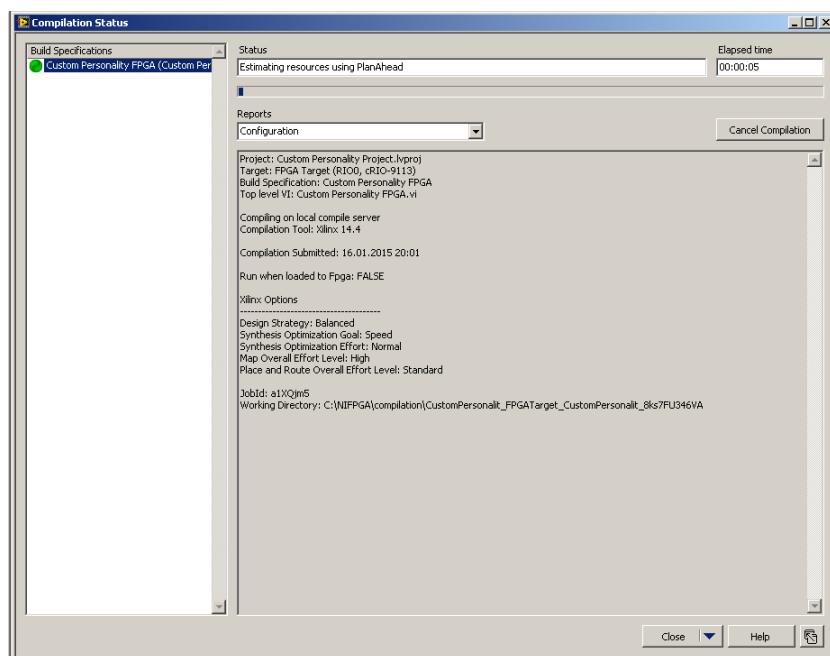
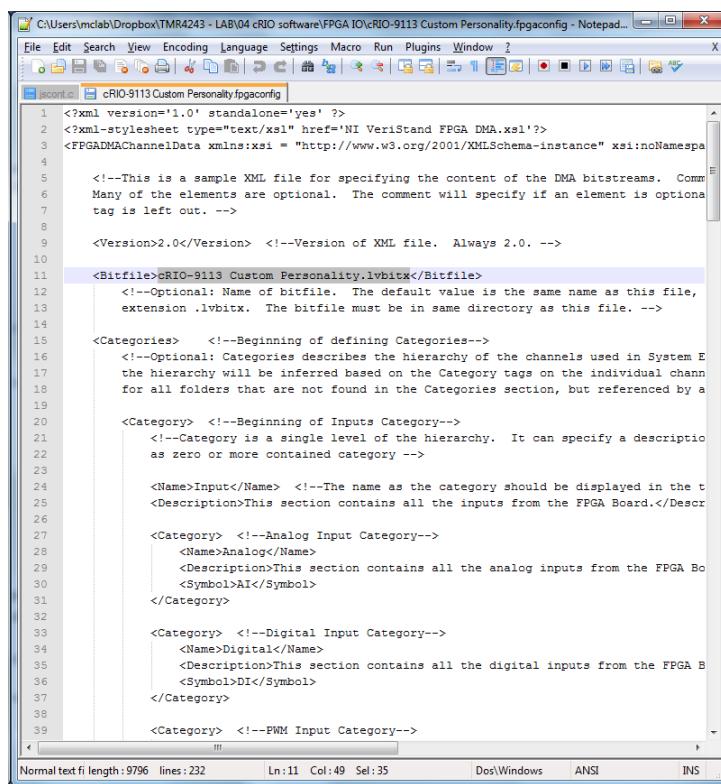


Figure 11.24: Create Labview FPGA target and XML - 16



```

1 <?xml version='1.0' standalone='yes' ?>
2 <?xmlstylesheet type="text/xsl" href="NI VeriStand FFGA DMA.xsl'?>
3 <FFGADMAChannel1Data xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="NI VeriStand FFGA DMA.xsd">
4
5   <!--This is a sample XML file for specifying the content of the DMA bitstreams. Comment Many of the elements are optional. The comment will specify if an element is optional or not. -->
6
7   <Version>2.0</Version> <!--Version of XML file. Always 2.0. -->
8
9   <Bitfile>cRIO-9113 Custom Personality.lvbitx</Bitfile>
10  <!--Optional: Name of bitfile. The default value is the same name as this file, extension .lvbitx. The bitfile must be in same directory as this file. -->
11
12  <Categories> <!--Beginning of defining Categories-->
13    <!--Optional: Categories describes the hierarchy of the channels used in System E. The hierarchy will be inferred based on the Category tags on the individual channels for all folders that are not found in the Categories section, but referenced by a
14      <Category> <!--Beginning of Inputs Category-->
15        <!--Category is a single level of the hierarchy. It can specify a description as zero or more contained category -->
16
17        <Name>Input</Name> <!--The name as the category should be displayed in the target -->
18        <Description>This section contains all the inputs from the FPGA Board.</Description>
19
20        <Category> <!--Analog Input Category-->
21          <Name>Analog</Name>
22          <Description>This section contains all the analog inputs from the FPGA Board.</Description>
23          <Symbol>AI</Symbol>
24        </Category>
25
26        <Category> <!--Digital Input Category-->
27          <Name>Digital</Name>
28          <Description>This section contains all the digital inputs from the FPGA Board.</Description>
29          <Symbol>DI</Symbol>
30        </Category>
31
32        <Category> <!--PWM Input Category-->
33          <Name>PWM</Name>
34          <Description>This section contains all the PWM inputs from the FPGA Board.</Description>
35          <Symbol>PWMS</Symbol>
36        </Category>
37
38      </Category>
39    </Categories>

```

Figure 11.25: Create Labview FPGA target and XML - 17

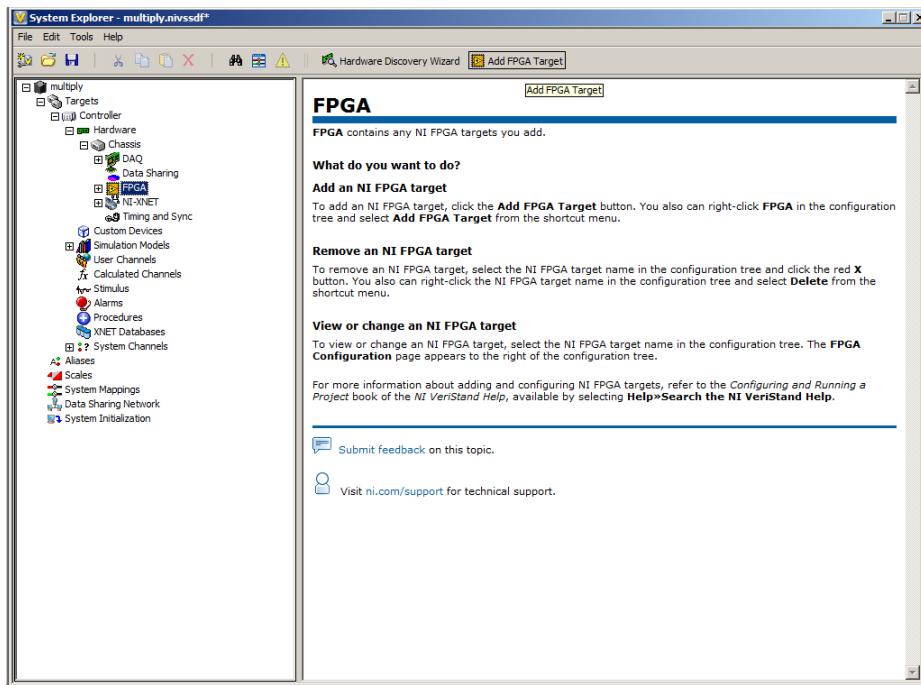


Figure 11.26: FPGA1

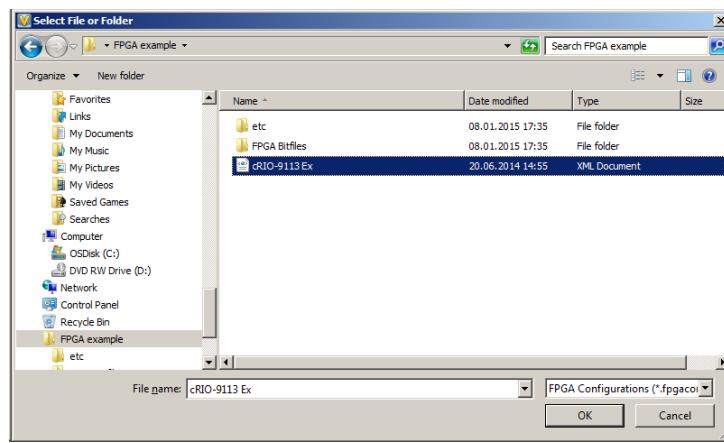


Figure 11.27: FPGA2

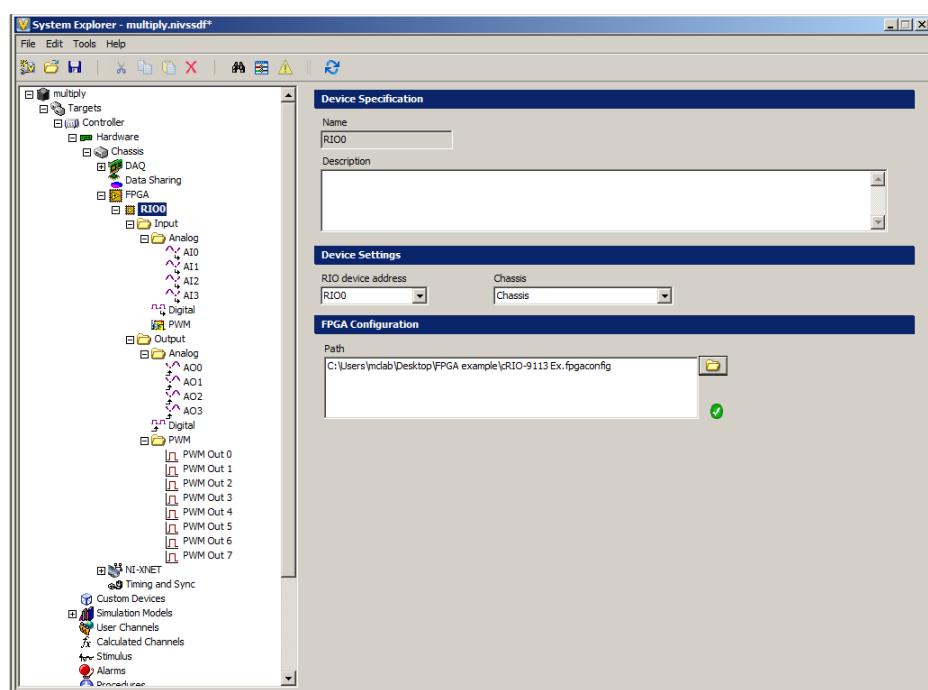


Figure 11.28: FPGA3

11.1.4 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 11.29.

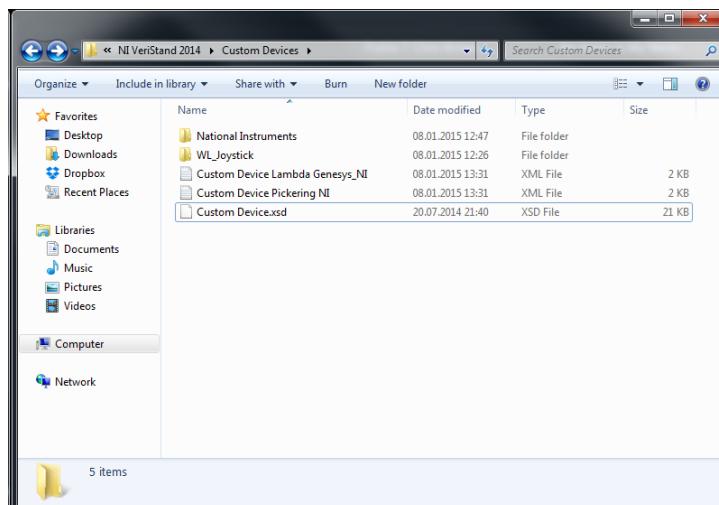


Figure 11.29: Custom device folder

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 11.30.

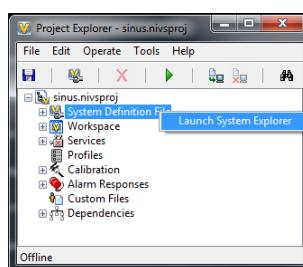


Figure 11.30: VeriStand launch system explorer

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

11.31. 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.

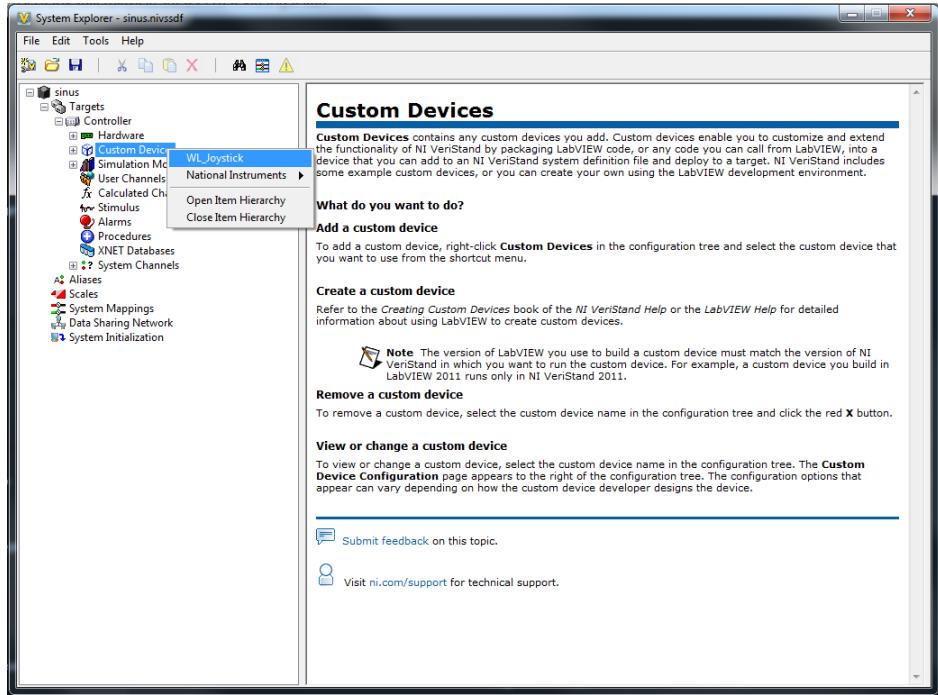


Figure 11.31: Custom device selection

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 11.32. 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 11.32).

You then simply find the ports you would like to connect, mark them and click the connect button. Figure 11.33 shows a joystick output connected to an input port on the Simulink model.

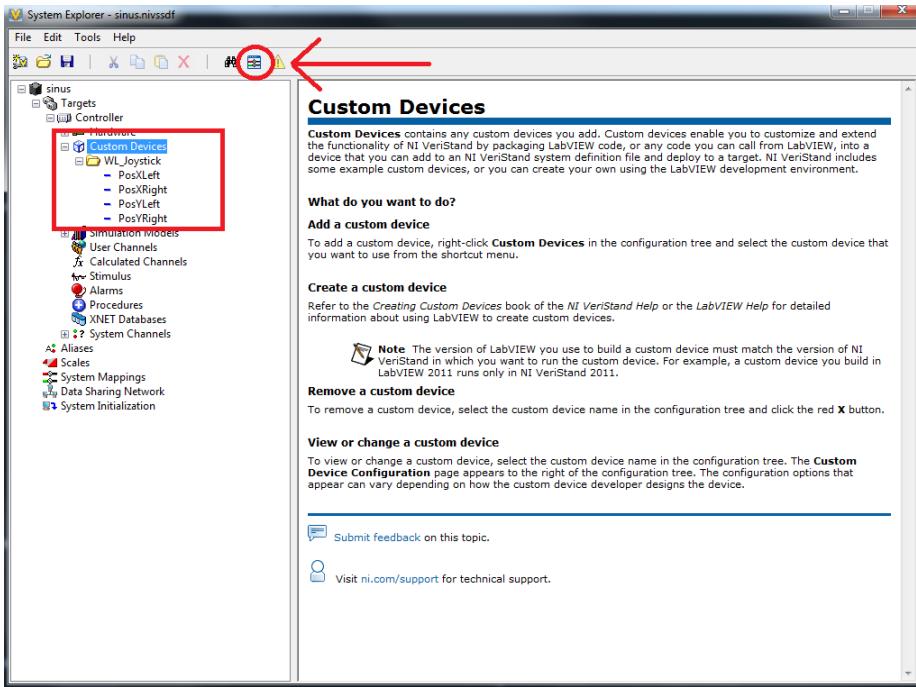


Figure 11.32: VeriStand

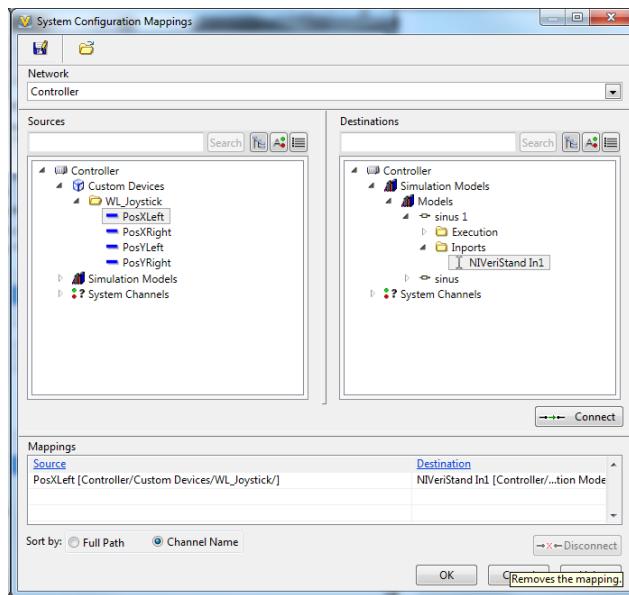


Figure 11.33: VeriStand System Configuration Mappings

11.2 Raspberry Pi

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

11.2.1 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.

11.2.1.1 Download operating system and utilities

Download and extract the newest Raspbian¹ operating system (OS) image.

Necessary utilities for the setup are

- Win32 Disk Imager² to write the OS image to the RPi SD card
- Advanced IP scanner³ to find the RPi address on the network
- Putty terminal emulator⁴ for SSH connection
- WinSCP⁵ for file transfer

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

Table 11.1: RPi installation and setup utilities

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

11.2.1.2 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 11.34. Make sure that you have selected the correct drive before you push WRITE.

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

¹raspberrypi.org/downloads

²sourceforge.net/projects/win32diskimager

³by Famatech, advanced-ip-scanner.com

⁴www.chiark.greenend.org.uk/~sgtatham/putty/download.html

⁵by Martin Prikryl, winscp.net/eng/download.php

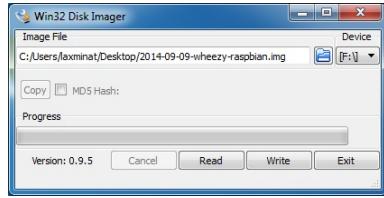


Figure 11.34: Disk Imager

11.2.1.3 Terminal access

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

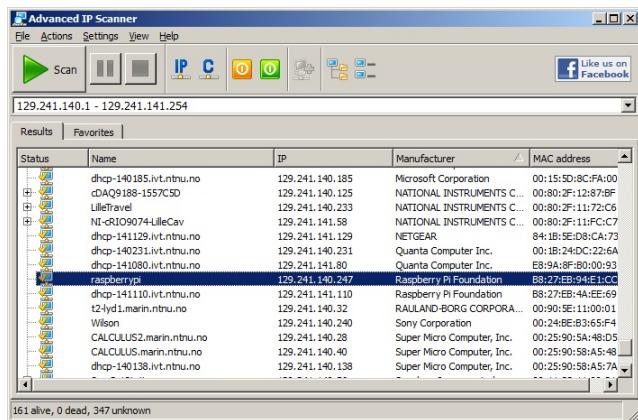


Figure 11.35: Advanced IP Scanner

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 advisable to sort the results by manufacturer since it is fixed (*Raspberry Pi Foundation*). The name is typically *raspberrypi*. See Figure 11.35.

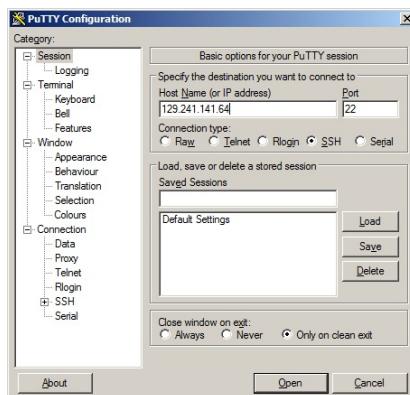


Figure 11.36: Putty settings

Once the IP is known, it is specified in the Putty settings, as in Figure 11.36, and a connection can be opened.

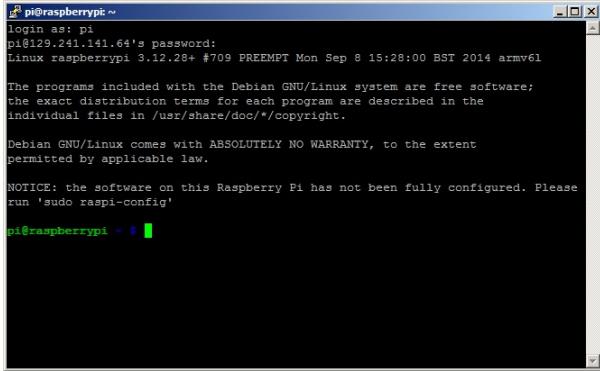


Figure 11.37: SSH connection

The default login is **pi**, and the default password **raspberry**. Figure 11.37 shows the terminal output on first login.

11.2.1.4 Finalize configuration

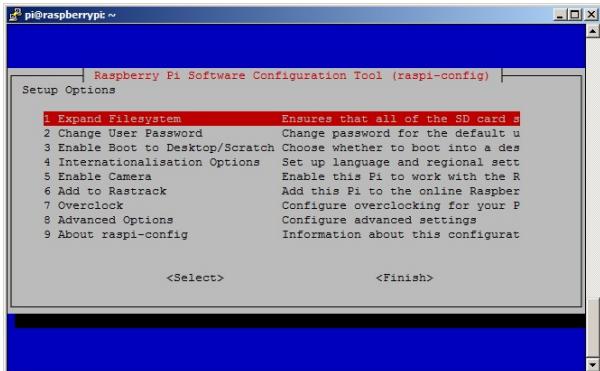


Figure 11.38: RPi configuration tool

Enter the

```
sudo raspi-config
```

command to start the RPi Software Configuration Tool, as in Figure 11.38. 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.

11.2.1.5 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.

11.2.1.6 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.

11.2.2 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.

11.2.2.1 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.

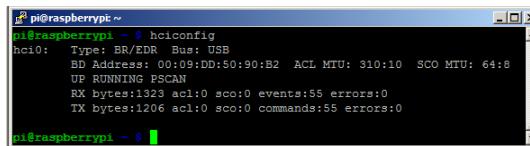


Figure 11.39: Bluetooth configuration tool

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 11.39. If instead it says DOWN, some error has occurred.

Most experienced errors were due to typos.

11.2.2.2 Bluetooth pairing

Sixaxis does not support the standard Bluetooth pairing procedure, instead, pairing is done over USB. The `sixpair` command-line utility⁶ 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 pairing 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
```

⁶by 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.

11.2.2.3 Joystick manager system service

QtSixA⁷ 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**.

11.2.2.4 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 11.2.1.5), 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 11.40. To exit the server use the key combination **CTRL+C**.

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

```

pi@raspberrypi: ~ ./jscont
Joystick C/S Controller. Version: TWa20150106
Joystick detected: Sony Computer Entertainment Wireless Controller
    27 axis
    19 buttons
using port #51717
waiting for new client...

```

Figure 11.40: 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.

11.3 Laptop

Compatibility between software is very important, See NI VeriStand Version Compatibility KnowledgeBase⁸.

11.3.0.1 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 Toolchain⁹
 - ii. Real-Time Workshop software
 - (b) PharLap:
 - i. Microsoft Visual C++
 - ii. The MathWorks, Inc. Real-Time Workshop® software

11.3.0.2 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>
⁹ftp://ftp.ni.com/pub/devzone/epd/gccdist_vxworks6.3_gcc3.4.4.zip

Chapter 12

Cybership Enterprise 1

Increased servo percentage results in clockwise? motion.

Hysteresis on motors

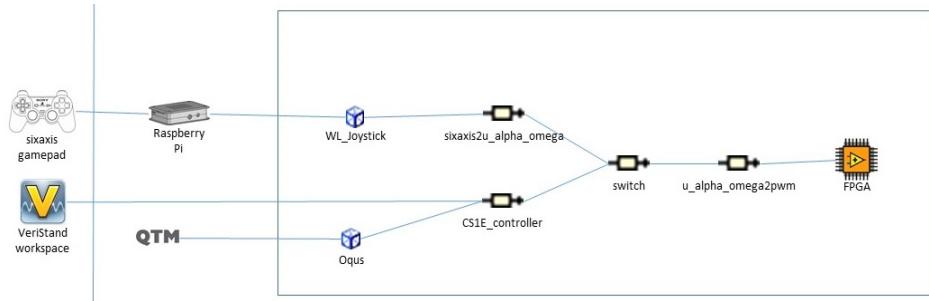


Figure 12.1: CSE1 component communication

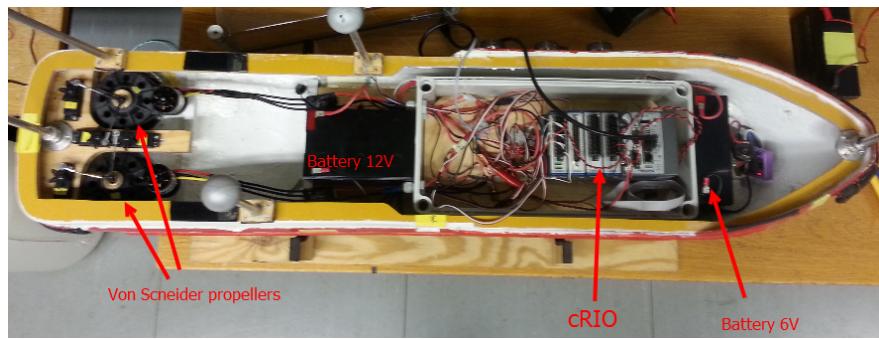


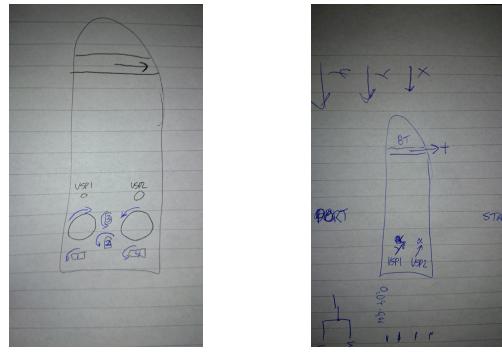
Figure 12.2: CSE1 - Hardware

12.1 Actuators

Antall, posisjon, aktivert med pwm.



Figure 12.3: CSE1 diagram



(a) Hardware input (b) Control input

Figure 12.4: 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 12.1: 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.

12.1.1 Motor control signals

12.1.2 Servo control signals

12.1.3 Measurements

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

Table 12.2: CSE1 cRIO analog input

12.2 Control software

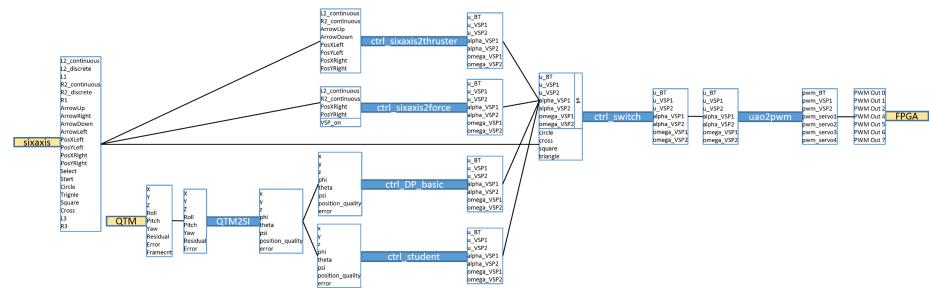


Figure 12.5: CSE1 control software modules

12.2.1 sixaxis (currently named WL_joystick) custom device

Reads sixaxis gamepad input from the RPi server.

12.2.2 QTM (currently named Oqus) custom device

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

12.2.3 QTM2SI

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

12.2.4 ctrl_sixaxis2thruster control module

Provides manual thruster control.

12.2.4.1 Voith Schneider Propellers

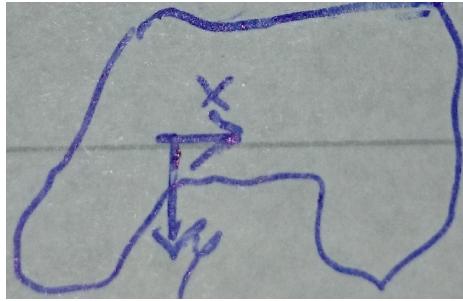


Figure 12.6: Sixaxis coordinate system

The left and right joysticks, respectively, give the VSP deflections, u_{VSP_1} and u_{VSP_2} , and angles, α_{VSP_1} and α_{VSP_2} , depicted in Figure 12.4b. The joystick coordinates PosX and PosY axes point right and down, as seen in Figure 12.6. The deflection is

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

The $\min(\cdot)$ ensures constraining $u_{VSP_i} \in [0, 1]$. The angle is

$$\alpha_{VSP_i} = \arctan 2(\text{PosX}, -\text{PosY}).$$

The VSP rotational speeds, ω_{VSP_1} and ω_{VSP_2} , are set in ± 0.1 increments by use of the directional pad up and down buttons.

12.2.4.2 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

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

maps to the interval $u_{BT} \in [-1, 1]$ with positive direction according to Figure 12.4b.

12.2.5 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 configuration shown in Figure 12.4b. The thrust is thus

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

where

$$\begin{aligned} \tau &= \begin{bmatrix} X \\ Y \\ N \end{bmatrix} \text{ are the forces and moment,} \\ T(\alpha) &= \begin{bmatrix} \cos(\alpha_{VSP1}) & \cos(\alpha_{VSP2}) & 0 \\ \sin(\alpha_{VSP1}) & \sin(\alpha_{VSP2}) & 1 \\ l_x,VSP1 \cos(\alpha_{VSP1}) - l_y,VSP1 \sin(\alpha_{VSP1}) & l_x,VSP2 \cos(\alpha_{VSP2}) - l_y,VSP2 \sin(\alpha_{VSP2}) & l_{BT} \end{bmatrix} \text{ is the configuration matrix,} \\ \alpha &= \begin{bmatrix} \alpha_{VSP1} \\ \alpha_{VSP2} \end{bmatrix} \text{ are the thruster angles,} \\ K &= \begin{bmatrix} K_{VSP1} & 0 & 0 \\ 0 & K_{VSP2} & 0 \\ 0 & 0 & K_{BT} \end{bmatrix} \text{ is the force coefficient matrix, and} \\ u &= \begin{bmatrix} u_{VSP1} \\ u_{VSP2} \\ u_{BT} \end{bmatrix} \text{ are the control forces.} \end{aligned}$$

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

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

where the VSP control forces are decomposed, yields

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

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

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

12.2.6 ctrl_DP_basic control module

Provides basic dynamic positioning capability.

12.2.7 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 12.3: Control input ranges

12.2.8 switch module

Selects one out of four control modules

- ctrl_sixaxis2thruster when Δ is pushed
- ctrl_sixaxis2force when \square is pushed
- ctrl_DP_basic when \circ is pushed
- ctrl_student when \times is pushed

The module also saturates the control signals according to Table 12.3 and issues a warning signal if the current controller exceeds the bounds.

12.2.9 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 12.4: Servo pwm ranges

Ikke lineært, ikke rett frem.

Foreslått metode

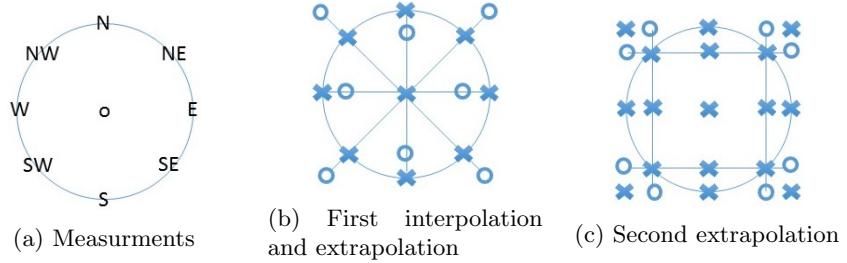


Figure 12.7: Servo, rod position tuning

12.2.10 FPGA interface

Outputs pwm signals through the digital output modules.

uao2pwm	FPGA
pwm _{BT}	pwm0
pwm _{VSP1}	pwm1
pwm _{VSP2}	pwm2
pwm _{servo1}	pwm4
pwm _{servo2}	pwm5
pwm _{servo3}	pwm6
pwm _{servo4}	pwm7

Table 12.5: PWM connections

12.3 Qualisys body

calibration

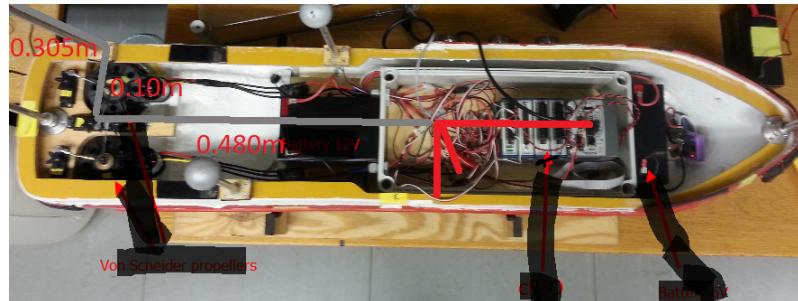


Figure 12.8: Matlab console

Chapter 13

Qualisys motion capture system

13.1 Start server

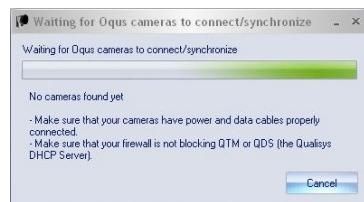


Figure 13.1: Matlab console

13.2 Aquire body

leverer med 50Hz på nettet

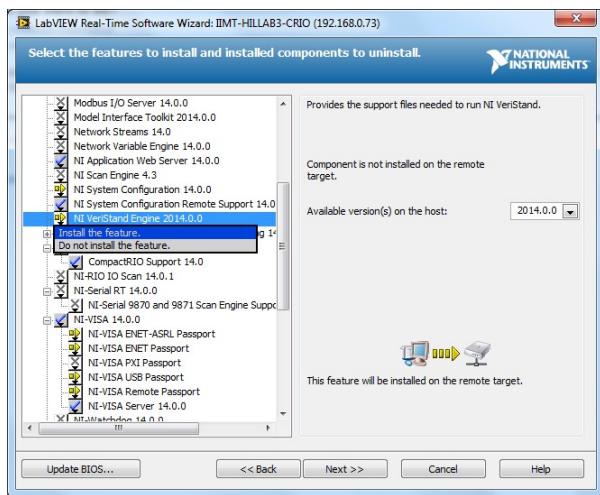


Figure 13.2: FPGA2

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	192.168.0.71 192.168.0.72 192.168.0.73 192.168.0.76	iimt-HILLab1-cRIO iimt-HILLab2-cRIO iimt-HILLab3-cRIO CSE1
Computer	192.168.0.10 192.168.0.41 192.168.0.42 192.168.0.43 192.168.0.47	Qualisys PC iimt-HILLab1-PC iimt-HILLab2-PC iimt-HILLab3-PC MClab
Subnet mask	255.255.255.0	for all

Table A.1: IP addresses

All RPis 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

ROV control modes

C.1 ROV dynamics

C.1.1 6 DOF model

We let the 6 DOF position/angle vector be $\eta = \text{col}(p, \Theta)$, where $p = \text{col}(x, y, z)$ and $\Theta = \text{col}(\phi, \theta, \psi)$, and the corresponding body-fixed velocity vector $\nu = \text{col}(u, v, r, p, q, r)$. The ROV kinematic model is then

$$\dot{\eta} = J(\Theta)\nu. \quad (\text{C.1})$$

The kinetic control model is given by

$$M\ddot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau + J(\Theta)^{-1}b(t), \quad (\text{C.2})$$

where, in particular, $M = M_{rb} + M_a$ is the system inertia matrix, $g(\eta)$ is the restoring vector, and $b(t)$ is an external slowly varying bias vector. The thrust load vector is

$$\tau = \begin{bmatrix} \mathcal{F} \\ \mathcal{M} \end{bmatrix}, \quad \mathcal{F} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} [\text{N}], \quad \mathcal{M} = \begin{bmatrix} K \\ M \\ N \end{bmatrix} [\text{Nm}] \quad (\text{C.3})$$

According to Fossen, if we assume starboard-port symmetry with $y_g = I_{xy} = I_{yz} = I_{zx} = 0$, then M is given by

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & -X_{\dot{w}} & 0 & mz_g - X_{\dot{q}} & 0 \\ 0 & m - Y_{\dot{v}} & 0 & -mz_g - Y_{\dot{p}} & 0 & mx_g - Y_{\dot{r}} \\ -X_{\dot{w}} & 0 & m - Z_{\dot{w}} & 0 & -mx_g - Z_{\dot{q}} & 0 \\ 0 & -mz_g - Y_{\dot{p}} & 0 & I_x - K_{\dot{p}} & 0 & -I_{zx} - K_{\dot{r}} \\ mz_g - X_{\dot{q}} & 0 & -mx_g - Z_{\dot{q}} & 0 & I_y - M_{\dot{q}} & 0 \\ 0 & mx_g - Y_{\dot{r}} & 0 & -I_{zx} - K_{\dot{r}} & 0 & I_z - N_{\dot{r}} \end{bmatrix}. \quad (\text{C.4})$$

Expressions for the Coriolis and centripetal matrix $C(\nu)$, and damping matrix $D(\nu)$, and restoring loads $g(\eta)$ can be found in Fossen.

C.1.2 Thruster configuration

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

$$f = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \quad l = \begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix}, \quad (\text{C.5})$$

respectively. Then the corresponding thrust loads become

$$\tau = \begin{bmatrix} f \\ l \times f \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_z \\ l_y f_z - l_z f_y \\ l_z f_x - l_x f_z \\ l_x f_y - l_y f_x \end{bmatrix}. \quad (\text{C.6})$$

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

$$\tau = \sum_{i=1}^m \tau_i = \sum_{i=1}^m \begin{bmatrix} f_i \\ l_i \times f_i \end{bmatrix} = B(\alpha)Ku \quad (\text{C.7})$$

where $B(\alpha)$ is the thruster configuration matrix with $\alpha = \text{col}(\alpha_1, \dots, \alpha_m)$, $K = \text{diag}(k_1, \dots, k_m)$ is a matrix of scaling gains, and $u = \text{col}(T_1, \dots, T_m)$ is the individual thruster forces (possibly normalized using the scaling in K).

C.1.2.1 Example: Minerva

For ROV Minerva the thruster configuration is shown in Figure C.1 (where f_p should point in the opposite direction).

Figure C.1: Thruster configuration for ROV Minerva. The thrust force f_p should point in opposite direction.

Considering only the thrusters for the 3DOF horizontal motion, the thruster configuration is:

- T1: Aft starboard thruster.
- T2: Aft port thruster.
- T3: Bow thruster, lateral thrust in positive y-direction.

Thruster	Location [m]	Azimuth []	Saturation [RPM]	Saturation [N]
Thr1	$l_1 = (-0.57, 0.24)$	$\alpha_2 = 10$	1450	$[T_{1,\min}, T_{1,\max}]$
Thr2	$l_2 = (-0.57, -0.24)$	$\alpha_3 = -10$	1450	$[T_{2,\min}, T_{2,\max}]$
Thr3	$l_3 = (0.166, 0)$	$\alpha_1 = 90$	1450	$[T_{3,\min}, T_{3,\max}]$

Table C.1: ROV Minerva thruster data

These thrusters data is given in Table C.1.

With the generalized force/moment vector $\tau := \text{col}(\tau_u, \tau_v, \tau_r)$, and $u = \text{col}(T_1, T_2, T_3)$ being the thrust force [N] from each thruster, this gives the following configuration

$$\begin{aligned} \tau &= B(\alpha)u && \text{(C.8)} \\ B(\alpha) &= \begin{bmatrix} \cos \alpha_1 & \cos \alpha_2 & \cos \alpha_3 \\ \sin \alpha_1 & \sin \alpha_2 & \sin \alpha_3 \\ l_{1x} \sin \alpha_1 - l_{1y} \cos \alpha_1 & l_{2x} \sin \alpha_2 - l_{2y} \cos \alpha_2 & l_{3x} \sin \alpha_3 - l_{3y} \cos \alpha_3 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \cos \alpha_2 & \cos \alpha_3 \\ 1 & \sin \alpha_2 & \sin \alpha_3 \\ l_{1x} & l_{2x} \sin \alpha_2 - l_{2y} \cos \alpha_2 & l_{3x} \sin \alpha_3 - l_{3y} \cos \alpha_3 \end{bmatrix}. \end{aligned}$$

The commanded thrust for (T_1, T_2, T_3) , based a commanded generalized thrust load vector τ_c , is then calculated by a **thrust allocation** algorithm.

C.2 ROV control modes

We have discussed control modes for an ROV. We typically have different input devices:

- Keyboard
- Joystick
- Touchscreen

Each of these can be set up to control the ROV in a number of control modes. The proposed modes, open for debate, is the following:

1. **Direct Thruster Control:** Direct command of individual thrusters.
2. **Direct Motion Control:** Direct command of ROV motions, distinguishing between:
 - (a) **Direct Body-relative motion:** Direct command of motion relative to Body reference frame.
 - (b) **Direct NED-relative motion:** Direct control of motion relative to NED reference frame.
 - (c) **Direct User-relative motion:** Direct control of motion relative to User reference frame.
3. **Auto Control:** Automatic control modes

- **AutoPos:** Automatic control of (x, y) .
 - **AutoDepth:** Automatic control of (z) .
 - **AutoHead:** Automatic control of (ψ) .
 - **CompRollPitch:** Automatic compensation control of (ϕ, θ) .
4. **Indirect Motion Control:** Instantaneously command a guidance filter and use feedback to automatically control the ROV correspondingly.
 5. **Marine Operations Control:** More complex modes specific for various marine operations of the ROV, such as:
 - Waypoint tracking control
 - Path-following
 - ...

Typically in a motion control system it is possible to combine several of these modes, such that one can for instance use *Direct Motion Control* in horizontal position, while using AutoHead and AutoDepth. Other modes may not be possible to combine, since they may be in conflict. For instance, one should not combine the submodes *Direct Body-relative Motion Control* with *Direct NED-relative Motion Control* since these may come in conflict. The user should therefore be given a mode *Direct Motion Control* and a switch to select Body-, NED-, or User-relative submodes.

C.2.1 Direct Thruster Control

Use the input device to directly command the individual thrust vector

$$u = \text{col}(T_1, T_2, \dots, T_m) = u_{cmd}.$$

C.2.2 Direct Body Motion Control

Direct Motion Control uses thrust allocation to enable the user to command motions of the fully actuated vehicle without need to consider individual thruster setting. Motions can be commanded relative to body-frame, NED-frame, or user-frame.

C.2.2.1 Direct Body-relative motion

Use the input device to command the generalized **body-relative** thrust vector

$$\tau = \text{col}(\mathcal{F}, \mathcal{M}) = \tau_{cmd}. \quad (\text{C.9})$$

Then use thrust allocation to calculate the corresponding individual commanded thrust vector, e.g.

$$u_{cmd} = B(\alpha)^\dagger \tau_{cmd}. \quad (\text{C.10})$$

C.2.2.2 Direct NED-relative motion

Use the input device to command the generalized **NED-relative** thrust vector τ_{cmd}^{NED} .

Assuming we have the orientation vector Θ for the vehicle available, we use the NED-Body transformation to calculate the corresponding body-fixed thrust vector, i.e.

$$\tau_{cmd}^{Body} = J(\Theta)^{-1} \tau_{cmd}^{NED}. \quad (\text{C.11})$$

Finally we allocate this thrust vector to the individual thrusters, e.g.

$$u_{cmd} = B(\alpha)^\dagger \tau_{cmd}^{Body} = B(\alpha)^\dagger J(\Theta)^{-1} \tau_{cmd}^{NED}. \quad (\text{C.12})$$

C.2.2.3 Direct User-relative motion

Use the input device to command the generalized **User-relative** thrust vector τ_{cmd}^{User} .

Assuming we have the orientation vector Θ_{User} for the user relative to the NED-frame, we then calculate

$$\tau_{cmd}^{NED} = J(\Theta_{User}) \tau_{cmd}^{User} \quad (\text{C.13})$$

Then, with the orientation vector Θ for the vehicle available, we use the NED-Body transformation to calculate the corresponding body-fixed thrust vector, i.e.

$$\tau_{cmd}^{Body} = J(\Theta)^{-1} \tau_{cmd}^{NED} = J(\Theta)^{-1} J(\Theta_{User}) \tau_{cmd}^{User}. \quad (\text{C.14})$$

Finally we allocate this thrust vector to the individual thrusters, e.g.

$$u_{cmd} = B(\alpha)^\dagger \tau_{cmd}^{Body} = B(\alpha)^\dagger J(\Theta)^{-1} J(\Theta_{User}) \tau_{cmd}^{User}. \quad (\text{C.15})$$

C.2.3 Auto Control

The automatic feedback control modes are proposed as:

1. AutoPos: Automatic control of (x, y) .
2. AutoDepth: Automatic control of (z) .
3. AutoHead: Automatic control of (ψ) .
4. CompRollPitch: Automatic control of (ϕ, θ) .

These submodes are self-explanatory. To be detailed in the student projects.

C.2.4 Indirect Motion Control

The idea is that the input devices are used to instantaneously command positions/velocities/accelerations, feed these through a reference filter or a guidance algorithm, and then use feedback to automatically control the ROV correspondingly.

Here we can be creative and try different options.

C.2.5 Marine Operations Control

These are a set of more complex control modes, typically combined from several of the different submodes, to give automatic functions that are important for different marine operations of the ROV. An example is Path Following, where a sophisticated path-generation algorithms and a guidance filter is combined with automatic control of (x, y) , (z) , and (ψ) , possibly while compensating (ϕ, θ) .

Appendix D

Personel and literature

D.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 artikkel om sleping av isberg

Robert Kanajus rkajanu@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 Kvæstad Idland Built CSS spring 2015.

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

D.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).

D.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.

D.4 Other

Håkon Nødset Skåtun <http://www.youtube.com/watch?v=MiESJsIZ004>

Appendix E

Maintenance

Oil VSP

Appendix F

Suppliers

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

Table F.1: Suppliers

Appendix G

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