Handbook of
Marine HIL simulation laboratory
and
Marine cybernetics laboratory

Introduction

Hva skal denne labben handle om?

Structure

Part I explains the concepts and motivations for real-time and hardware-in-the-loop (HIL) testing.

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

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

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

Contents

Nomenclature

cRIO	National Instruments compact reconfigurable input/output real-time embedded industrial controller
CSE1	Cybership Enterprise 1
HIL	Hardware-in-the-loop
MC	Marine cybernetics
RPi	Raspberry Pi single-board computer
VI	virtual instrument, a LabVIEW program

Part I

Theory

1 Realtime

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

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

2 HIL

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

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

Smogeli, Fremtidens verifikasjon av kontrollsystemer for Skip og offshorefartøy

DNV, Hardware in the Loop Testing (HIL)

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

Smogeli, Introduction to third- party HIL testing

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

Pivano, Experiences from seven years of DP software testing

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

Ambrosovskaya, Approach for Advanced Testing of DP Control System

Selvam, System Verification Helps Validate Complex Integrated Systems

A. Veksler prøveforelesning:

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

Asgeir

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

2.1 Real-time computing

2.2 SIL

2.3 Model-in-the-loop

2.4 Hybrid simulation

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

Part II

Laboratory descriptions

3 Marine HIL simulation laboratory

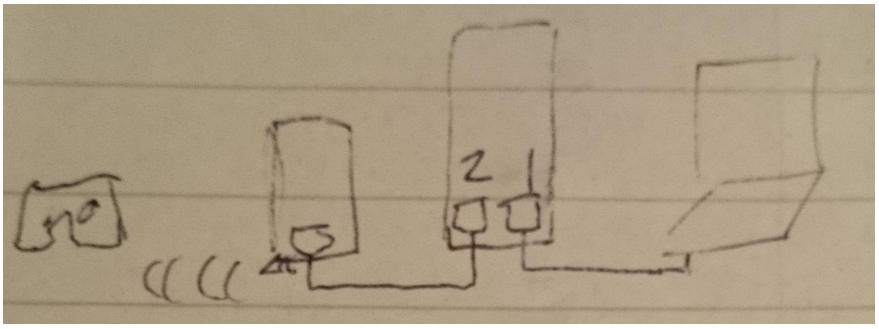


Figure 1: HIL setup

The lab consists of three equivalent portable setups, as illustrated in Figure 1, each including

- Sony Sixaxis wireless gamepad for PlayStation 3 (PS3)
- Raspberry Pi with Bluetooth dongle
- National Instruments (NI) cRIO-9024 compact reconfigurable input/output embedded real-time control and acquisition device
- Dell Latitude E6440 laptop

The setup is suitable for implementation qualification and comprehensive testing

- marine control algorithms
- communication interfaces,
- human-machine interfaces,
- experimental test scenarios, and
- experimental setups before proceeding to other laboratories (such as CSE1)

3.1 Hardware

3.1.1 cRIO-9024

The CompactRIO system's rugged hardware architecture includes I/O modules, a reconfigurable FPGA chassis, and an embedded controller. Additionally, CompactRIO is programmed with NI LabVIEW graphical programming tools and can be used in a variety of embedded control and monitoring applications. For more info vistit the producer website

Sakset fra ni.com/compactrio

3.2 Software

Only PC software, RPI and cRIO software in appendix.

MATLAB	Mathworks	Modelling
LabVIEW	National Instruments	
VeriStand	National Instruments	Interfacing, Real-time testing and simulation
NI MAX	National Instruments	Measurement & Automation Explorer: configurering av cRIO
Qualisys?		

Table 1: Software

Hva brukes det enkelte programmet til?

3.2.1 MATLAB

3.2.2 LabVIEW

LabVIEW - Real time module
National Instruments real-time technology offers reliable, deterministic performance for your time-critical applications. Use the LabVIEW Real-Time Module to develop and deploy complex real-time systems quickly and efficiently to the CompactRIO microprocessor.

3.2.3 VeriStand

3.2.4 NI MAX

3.2.5 Qualisys positioning system

The positioning system works by tracking reflectors placed on the ship with the use of high speed cameras.
Qualisys consists of three systems

- Qualisys Oqus: The cameras used to register/see the IR markers
- Qualisys Motion Capture Systems: is the system that process the data from Oqus
- Qualisys Track Manager: The userinterface to interact with Motion Capture System

3.3 Communication

.VxWorks RT targets - .out

7

4 Marine cybernetics laboratory

http://www.ntnu.no/imt/lab/cybernetics

The Marine Cybernetics Laboratory is the newest test basin at the Marine Technology Centre. It is located in what was originally a storage tank for ship models made of paraffin wax.

As the name indicates, the facility is especially suited for tests of marine control systems, due to the relatively small size and advanced instrumentation package. It is also suitable for more specialised hydrodynamic tests, mainly due to the advanced towing carriage , which has capability for precise movement of models in 6 degrees of freedom.

The MCLab is operated by the Department of Marine Technology, and has been a Marie Curie EU Training Site (2002-2008). It is mainly used by Master and PhD-students, but it is also available for MARINTEK and external users.

The software in use was developed using rapid prototyping techniques and automatic code generation under Matlab/Simulink and Opal. The target PC onboard the vessel runs the QNX real-time operating system while experimental results are presented in real-time on a host PC using Labview.

Sakset rett fra nettsiden http://www.ntnu.edu/imt cybernetics-lab

4.1 CSE1 - CS Enterprise I

Figure 2: C/S Enterprise I

Control system

PWM

Digital output

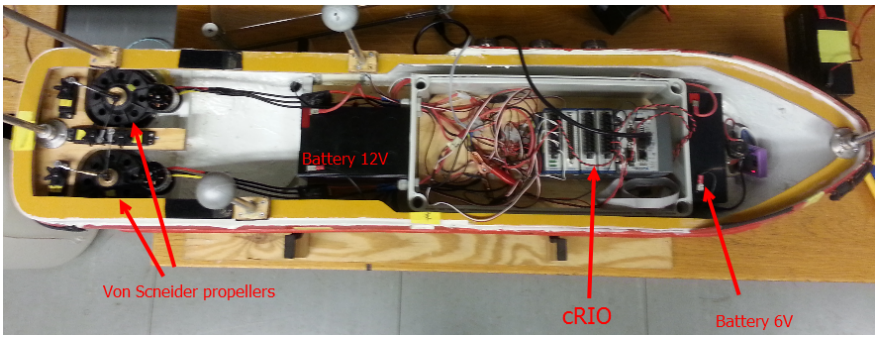


Figure 3: CSE1 - Hardware

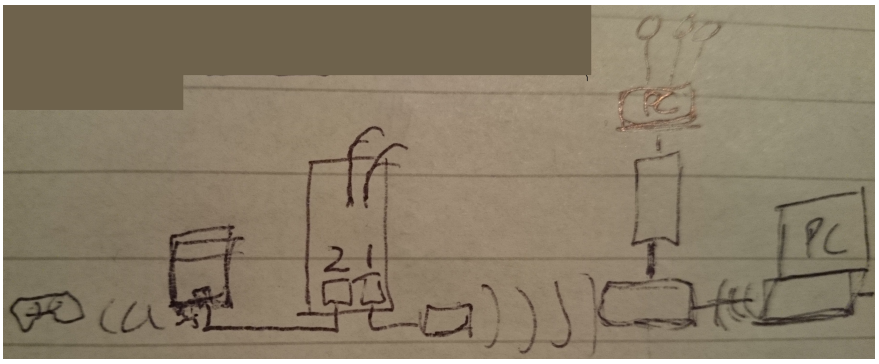


Figure 4: CSE1 setup

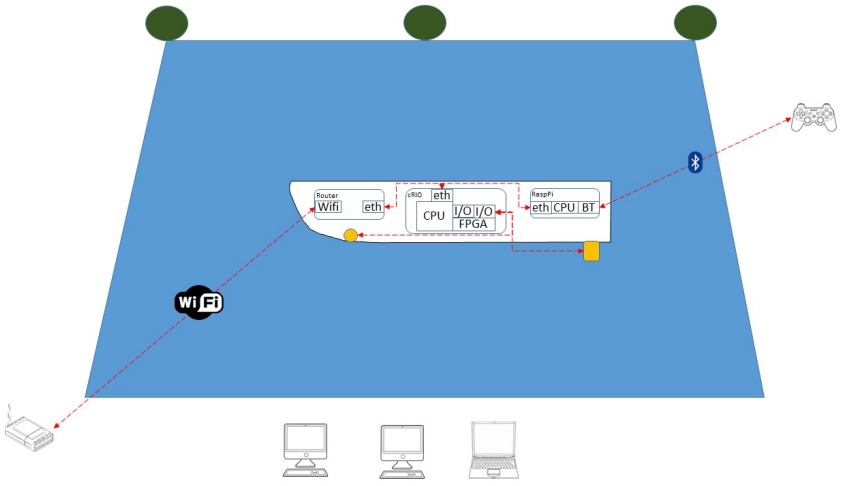


Figure 5: Towing carriage

4.2 Qualisys postitioning system

4.3 Communication

For communication with the ship, the wireless network HILlab is used

4.4 Towing carriage

Carriage : towing speed 2 m/s, 5 (6) DOFs forced motions Current generation: 0-0.15m/s

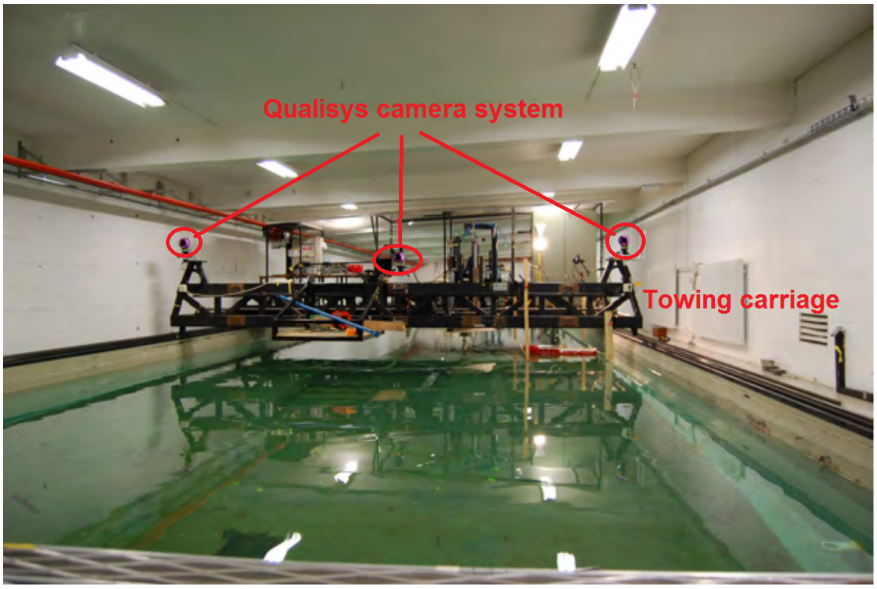


Figure 6: Towing carriage

4.5 Wave generator

The wave genertor is located at the end of the tank and is operated from its own copmuter. It has the capability to create first order Stoke waves or irregular recreate different wave spectras such as JONSWAP or PM spectras.
Significant wave height $H_s = 0.3$ [m] with period T between 0.6 [s] and 1.5 [s]

4.5.1 First order Stoke waves

First order stoke waves are regular linear waves. Very nice to do calculations with, but not so representative for real life conditions. Described by potential theory

4.5.2 Irregular waves

Part III

Laboratory user guide

5 HIL simulation and testing

5.1 Simulink model adaptation and compilation

Complete the following steps to convert your model you created in The MathWorks, Inc. Simulink® software into a compiled model that runs on RT targets.

5.1.1 Modeling

In order for the model to interact with VeriStand, special input and output blocks must be added to the block diagram[1]. These are found in the Simulink Library Browser under NI VeriStand Blocks.

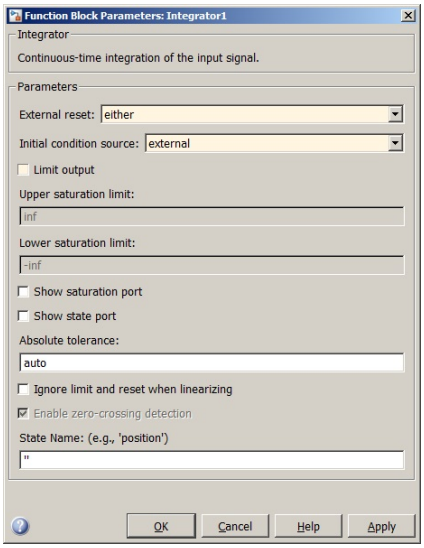


Figure 7: Integrator function block parameters

If the simulation is to be run with different initial conditions, one possible method is to allow external reset of the integrators. This is done right-click the integrator and selecting Block Parameters (Integrator) in the drop-down menu. Here, the reset condition is set. The initial condition source should be external, as in Figure 7.

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

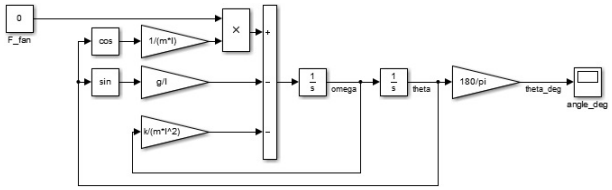


Figure 8: Simulink model for offline simulation

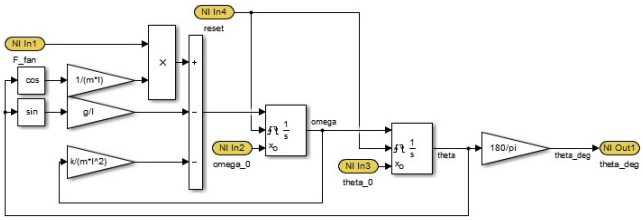


Figure 9: Simulink model for adjusted for compilation

Example: For a simple pendulum, $\dot{\omega} = -\frac{g}{l}\sin(\theta) - \frac{k}{ml^2}\omega + \frac{F_{\text{fan}}}{ml}\cos(\theta)$, the offline simulation block diagram could look as Figure 8. Figure 9 shows the same system adapted for VeriStand input, including reset and initial conditions, and output. The VeriStand blocks are yellow. omega_0 and theta_0 are ports corresponding to the initial conditions $(\omega(0), \theta(0))$. The integrators take these values whenever reset is rising og falling.

5.1.2 Model configuration

The code generation toolbox compiles the Simulink diagram to an output shared library in *.out format[2]. 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.

The correct target file should be selected depending on the target device. Select `NIVeriStand_VxWorks.tlc` for VxWorks targets[3], such as cRIO-9024, as in Figure 11.

The WindRiver GNU Toolchain must be present in the folder specified under NI Configuration, as in Figure 12.

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

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

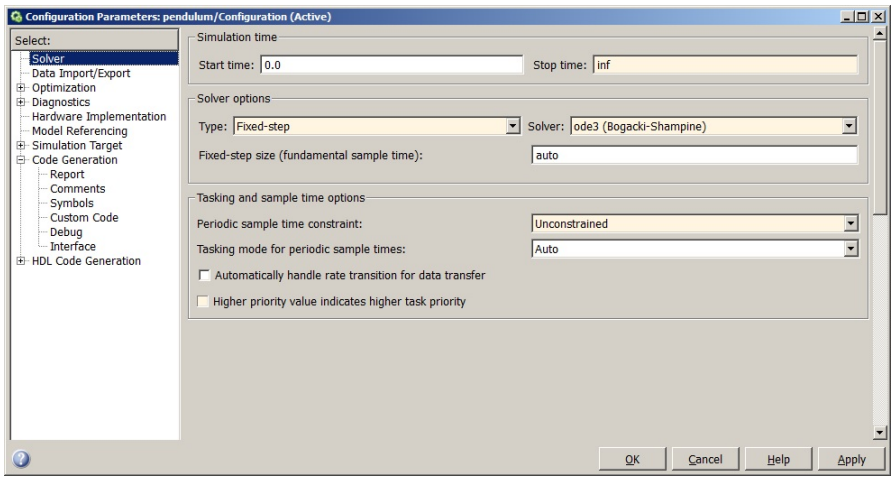


Figure 10: Simulink configuration parameters - solver

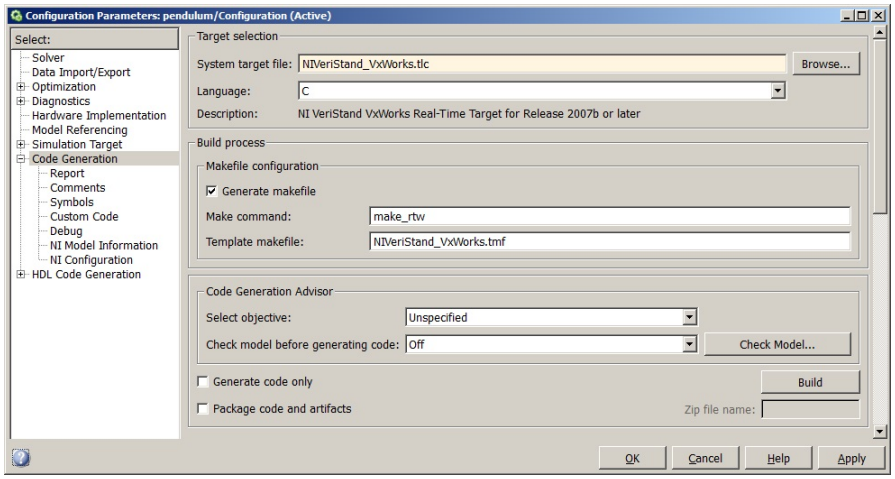


Figure 11: Simulink configuration parameters - target selection

5.1.3 Build

The build output is placed in a subfolder in the MATLAB Current Folder. The desired folder must therefore be active in the MATLAB main window, as in Figure 13, before compiling. The build subfolder name is [simulink model name]_niVeriStand_VxWorks_rtw.

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

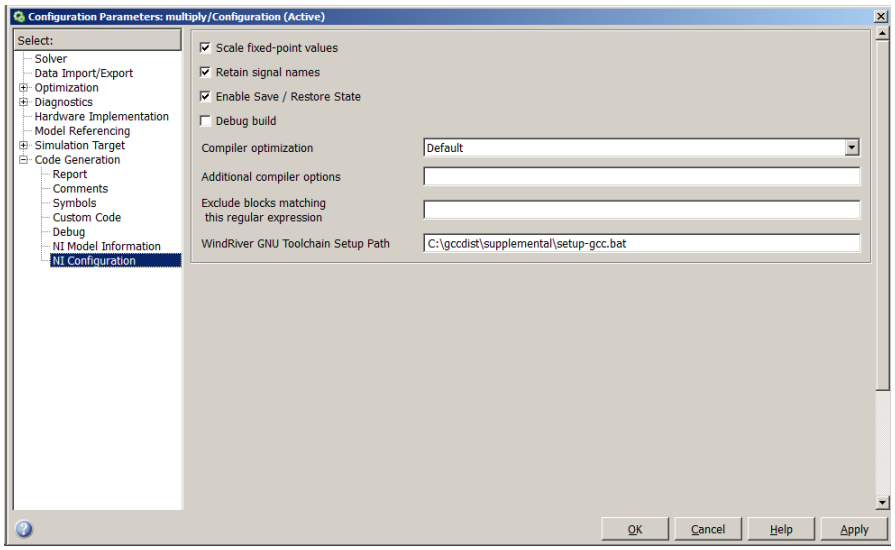


Figure 12: Simulink model configuration - NI configuration

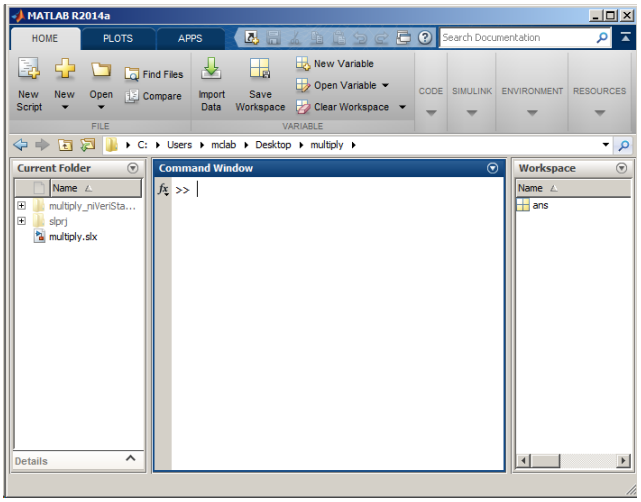


Figure 13: Matlab console

5.2 Model deployment

Simulations are set up, deployed and interfaced through VeriStand. Figure 14 shows the start screen. Already configured projects can be run directly from here, or reconfigured.

5.2.1 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

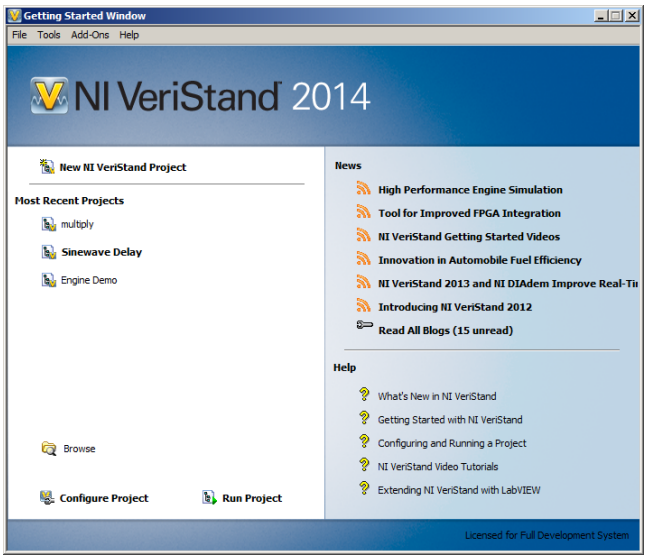


Figure 14: VeriStand

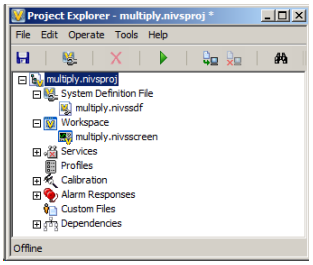


Figure 15: VeriStand Project Explorer

files in given location and opens the Project Explorer, as in Figure 15. In this section, the example project name is multiply.

5.2.2 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.

1. Set the correct controller operating system and IP address, as in Figure 16. All HIL and MC lab IP addresses are given in Table 7. Also, note the target rate.

2. Click Add a Simulation Model, as seen at the top of Figure 17. Browse to the output of the Simulink compilation, as seen in Figure 18. 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

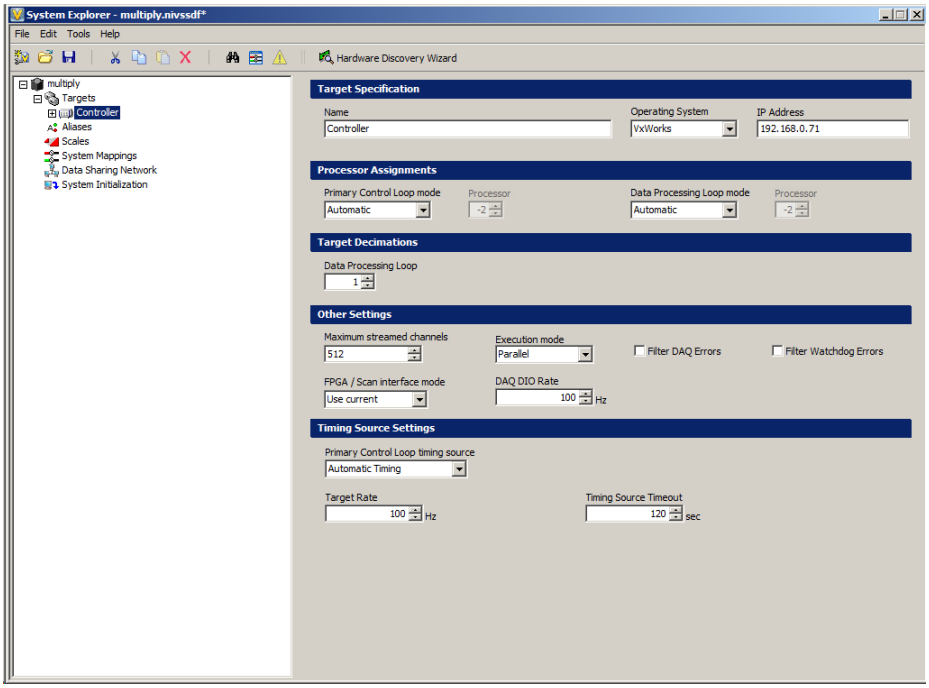


Figure 16: VeriStand - System Explorer - Controller

device pane and choosing the required device[4]. Figure 19 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 20.

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

5.2.3 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 popup dialog.

4. Close the Workspace editor.

[4]If the required device is not present, refer to the device driver installation instructions in Section A.3.

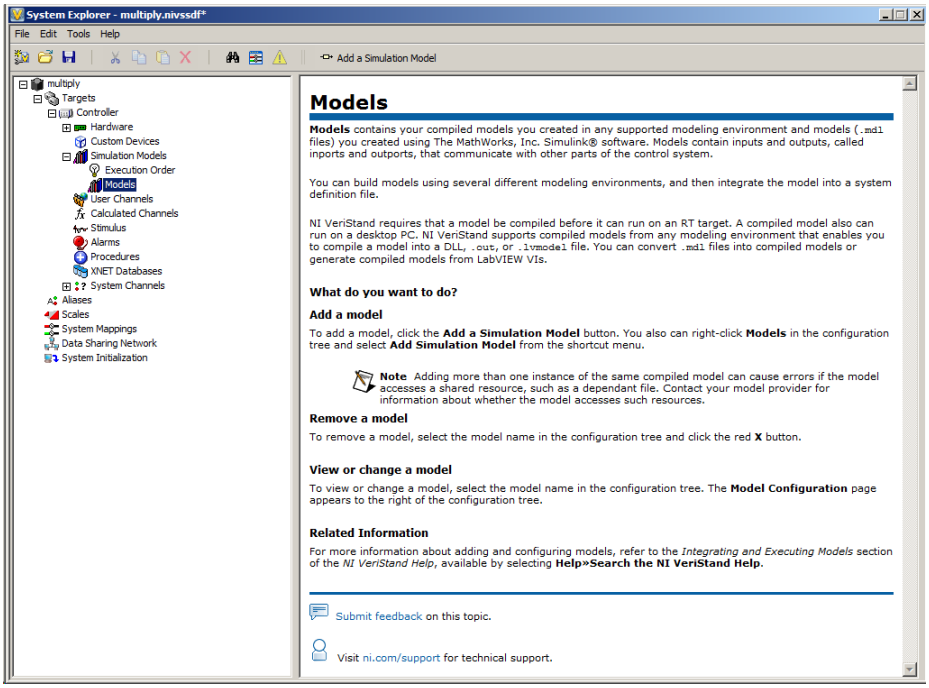


Figure 17: VeriStand - System Explorer - Models

5.2.4 Run

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

5.2.5 Stop

button

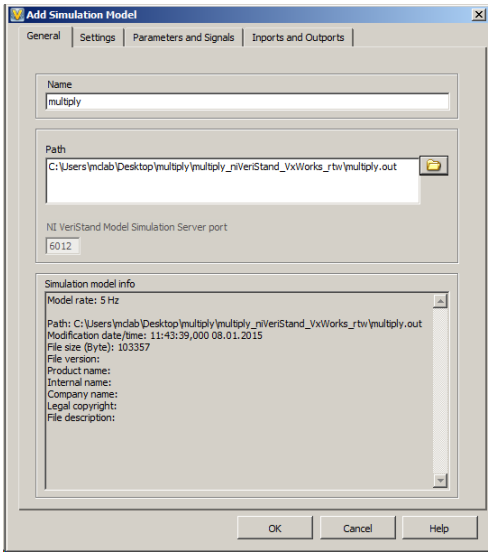


Figure 18: VeriStand - System Explorer Model

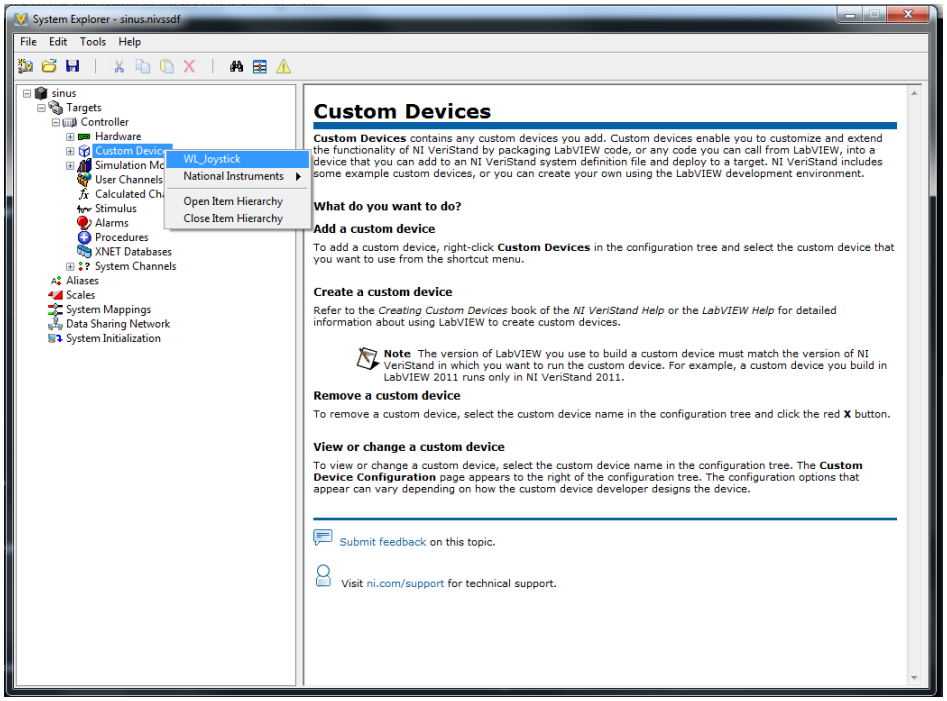


Figure 19: Custom device selection

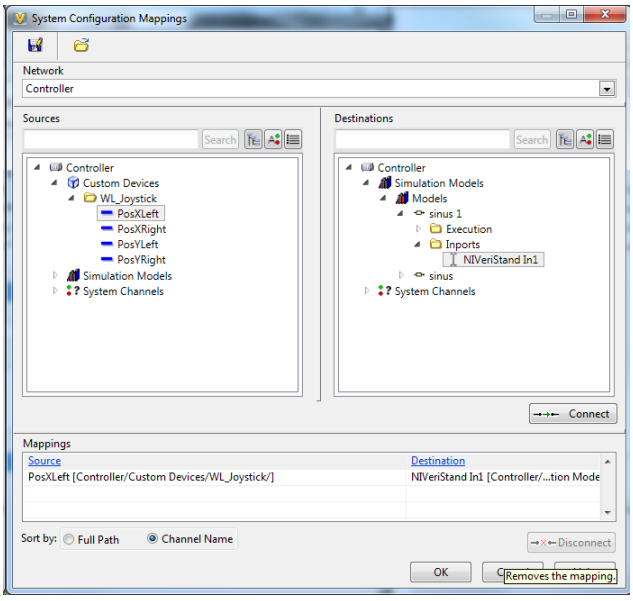


Figure 20: VeriStand System Configuration Mappings

5.3 Data logging

Figure 21: Logging Control

A Logging Control, as seen in Figure 21, must be added to the workspace to export data from the simulation. The control is added as described in Section 5.2.3.

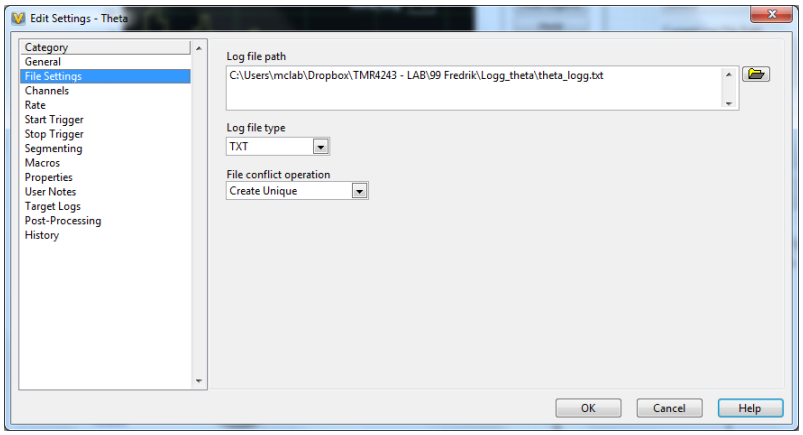


Figure 22: Logging Control file settings

Once the control is added, a popup window allows to edit the settings. The log file path is specified under File Settings, see Figure 22. Under Channels, the desired channels can be selected and added, as in Figure 23.

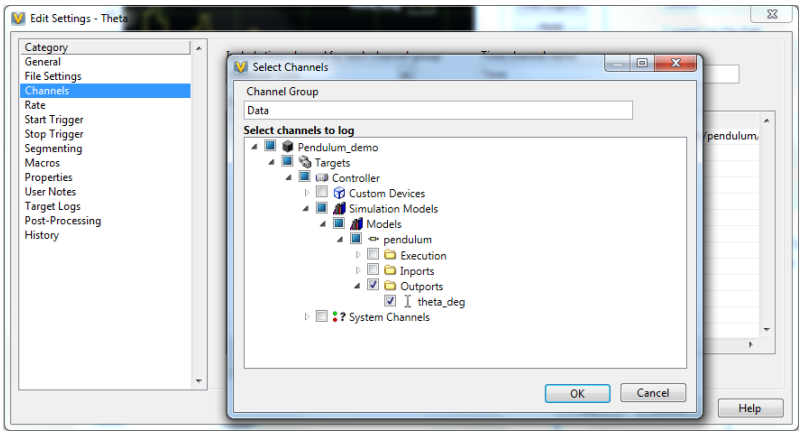


Figure 23: Logging Control add channel

6 CSE1 model scale testing

6.1 Ship launching procedure - before sailing

Batteries should be placed as in Figure 3.

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

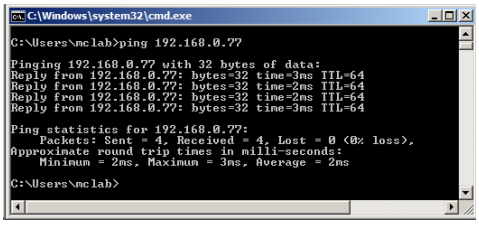


Figure 24: Ping, successful access to CSE1

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

6. Verify laptop access: ping the CSE1 IP in the command promt, as in Figure 24. While the round trip times may vary, it is essential to have 0% loss.

6.2 Deploy control system

Veristand osv

6.3 Ship docking procedure - after sailing

Undeploy the running project to disable all actuators.

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

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

Connect the Sixaxis gamepad to the laptop for charging.

Part IV

TMR4243 exercises and expected results

7 Pendulum lab

Adjust Simulink model for VeriStand.

Deploy

Create suitable workspace

Actuate fan manually.

Simulate to get same results as Simulink.

Export data

Try with handed out model (surprises).

8 Internal dynamics lab

9 Estimation lab

10 The maneuvering lab

Part V

Equipment setup and configuration

A cRIO

A.1 Ethernet ports

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

A.1.1 Primary

Set fixed IP, set fixed IP on HIL-computers

A.1.2 Secondary ethernet port

Enabling the 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*

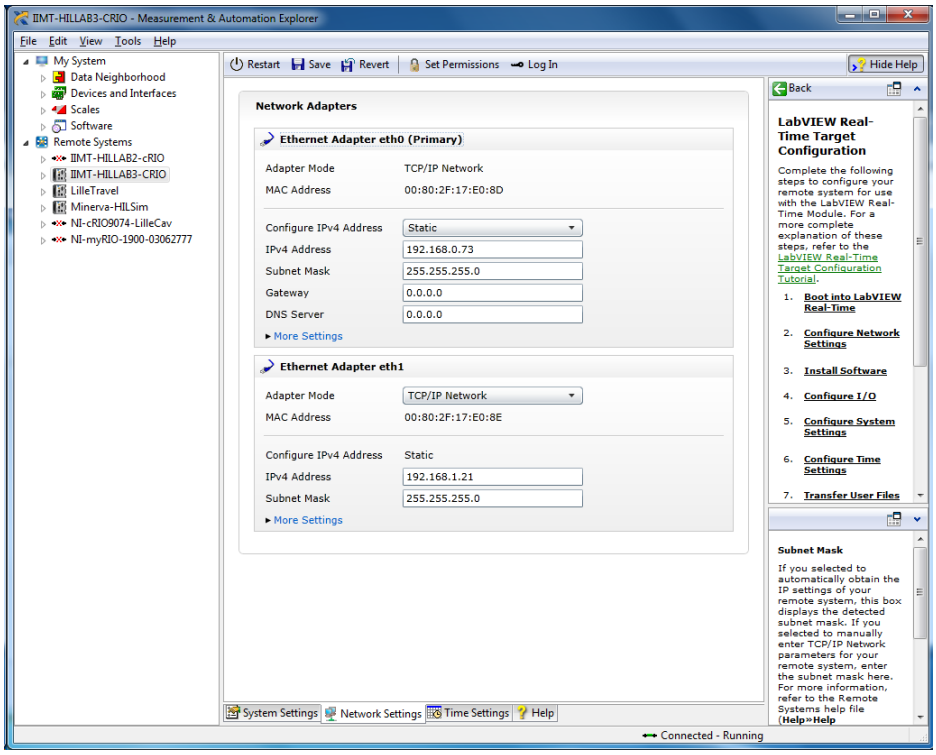


Figure 25: NI MAX - Network Settings

A.2 Update cRIO software

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

A.2.1 Update

1. Open NI Max
2. Find your cRIO on the left hand side and click it
3. Click Software, and then Add/Rem ove Software located on the top pane, see Figure 26
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 27
5. Click next without making any changes28
6. Click next without making any changes29
7. Wait for the installtion to finish and the cRIO to reboot

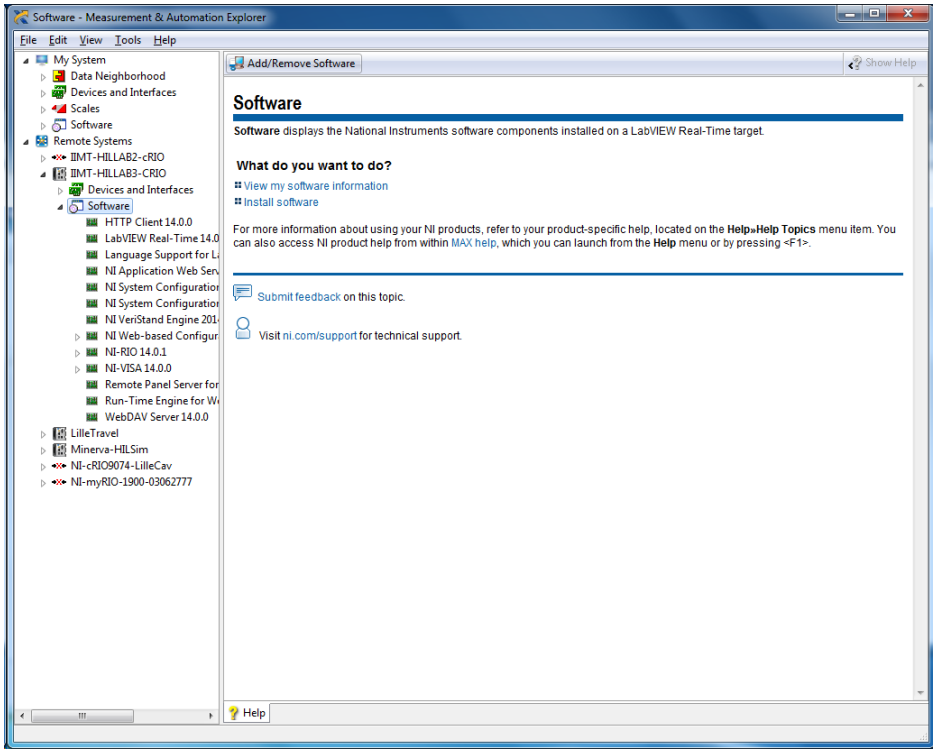


Figure 26: NI MAX - Software Update 1

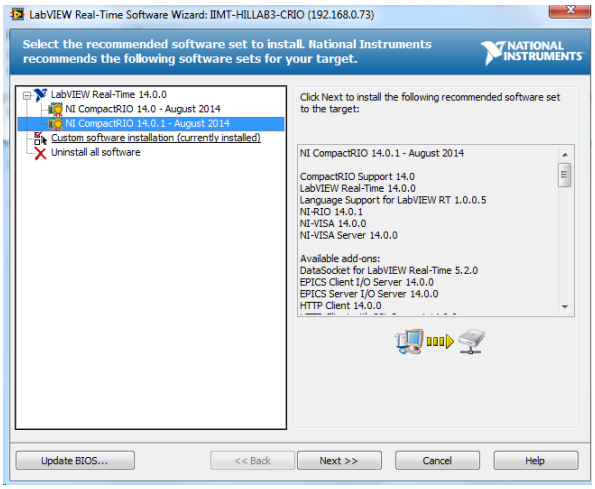


Figure 27: NI MAX - Software Update 1

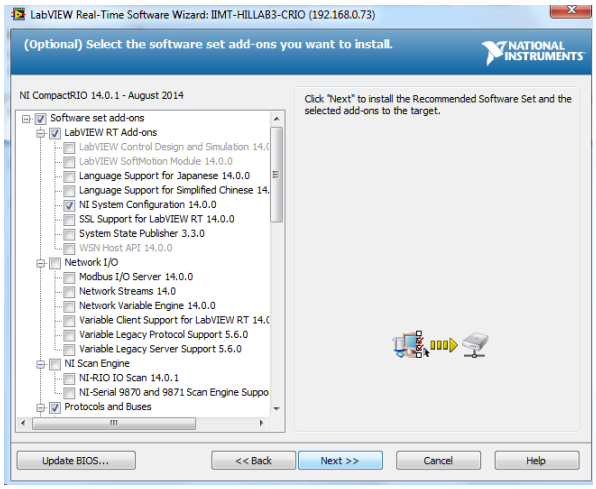


Figure 28: NI MAX - Software Update 3

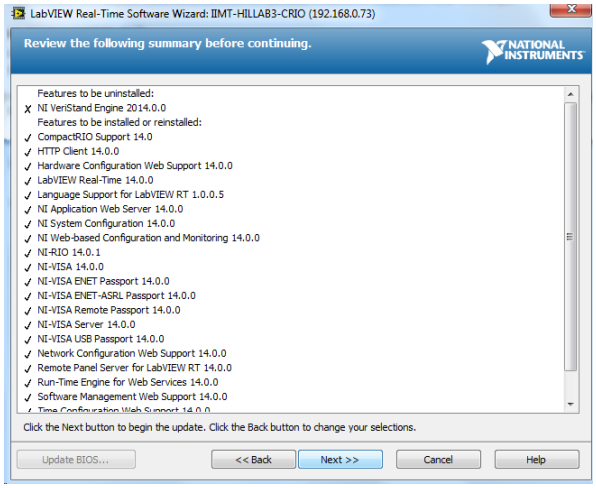


Figure 29: NI MAX - Software Update 4

A.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 30
3. Ignore the warning, See Figure 31
4. Locate NI Veristand Engine 2014 0.0 and click install feature. See Figure 32
5. Click your way through the rest of the installation and let the cRIO reboot.

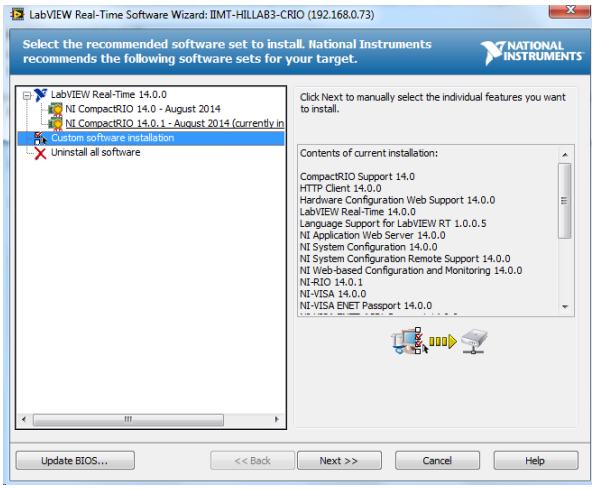


Figure 30: NI MAX - NI Veristand Engine installation 1

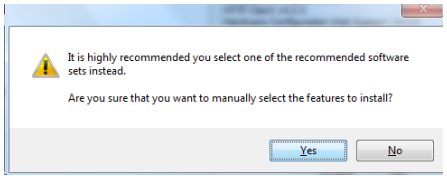


Figure 31: NI MAX - NI Veristand Engine installation 1

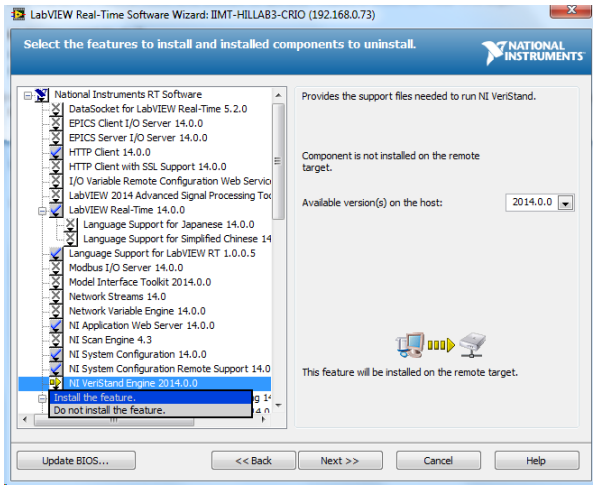


Figure 32: NI MAX - NI Veristand Engine installation 1

A.3 Installing custom device driver

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 33.

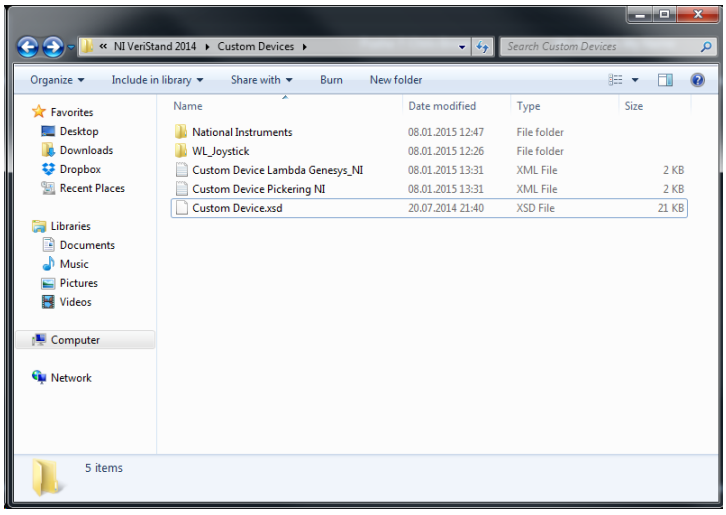


Figure 33: Custom device folder

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 34.

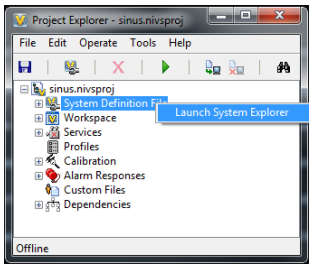


Figure 34: VeriStand launch system explorer

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 35. 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.

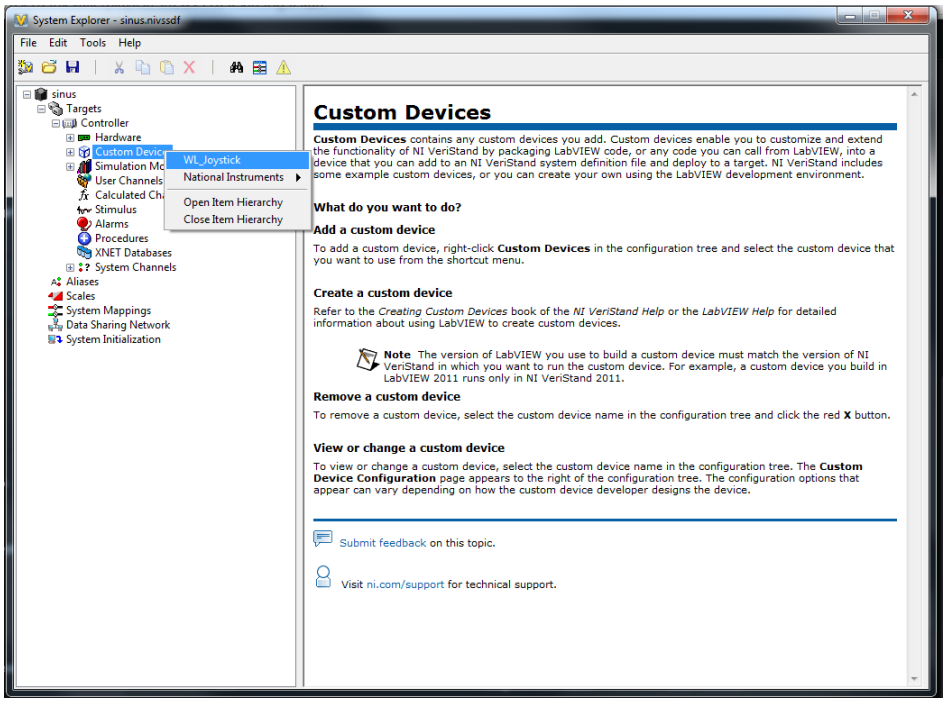


Figure 35: Custom device selection

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 36. 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 36).

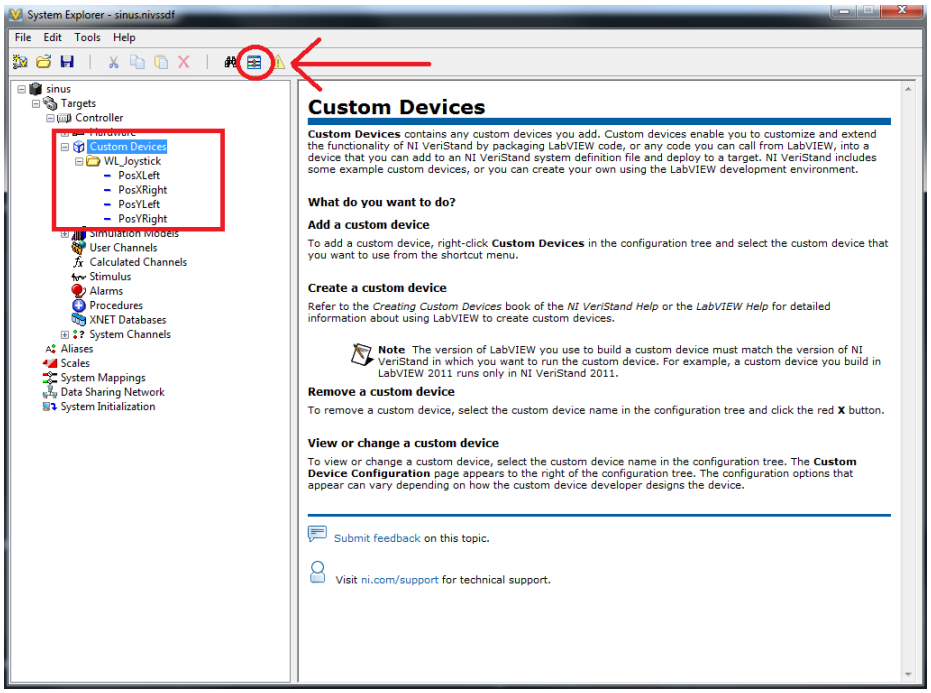


Figure 36: VeriStand

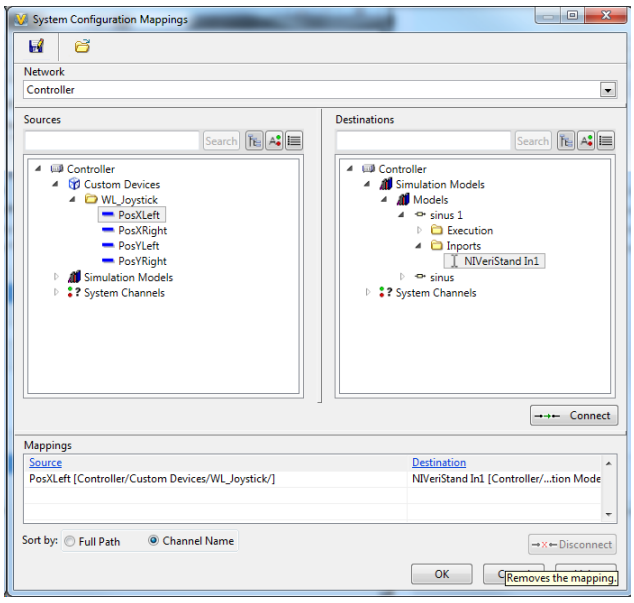


Figure 37: VeriStand System Configuration Mappings

You the simply find the ports you would like to connect, mark them and click the connect button. Figure 37 a joystick output is connected to a input port on the Simulink model.

A.3.1 Creating custom device driver

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

A.3.2 PWM output

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

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

A.3.3 Analog input

Eirik: FPGA-greier osv

A.4 Veristand FPGA programmering

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.

A.4.1 Create Labview FPGA target and XML

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

1. Open LabVIEW and create new project. We have done this in LabVIEW 2013 because of some instalation issues with LabVIEW 2014, but the procedure should be the same for both editions.
2. Choose NI Veristand FPGA Project in project templates and proceed.
3. Choose CompactRIO Reconfiguarble Embedded System and click next.
4. You will now get the choice between letting LabVIEW detect your cRIO system or configure it yourself. If you are connected to the cRIO and it has all of the I/O ports connected, the option “Discover existing system” is simpler and therefore recommneded. If you do not have your cRIO connected choose “Create new system”, this is the version that will be worked through here.
5. Select your controller, in our case cRIO-9024.

6. Select your FPGA target, in our case cRIO-9113.
7. Then you select your I/O modules to the correct slots. In our case NI 9215 in slot 1 and NI 9474 in slot 4.
8. You are now finished with configuring your project. Press next.
9. The project menu will now appear and should look something like Figure 47. Select the LabVIEW VI as demonstrated our is called Custom Personality FPGA.vi
10. The UI window will now present itself, select window and show block diagram.
11. You should now see a block diagram similar to Figure 48. You will now have to redesign this to look like Figure 49. This will be valid for our system, if you have different I/O modules the block diagram need to reflect this.
12. Now, return to the Project explorer and select Build Specifications and Custom Personality FPGA
13. A new window will open. Check that the name and project path is correct and press build.
14. Select your preferred compile server. The compilation process will take quite some time (approx 15-30 min).
15. When the compilation process is finished, the last step is to edit the automatically generated XML file. You will now have to find you project directory in Windows. Here there will be a folder called bitfiles which contains the files you compiled in the last step, there will also be a .XML file. The point of editing this file is to match the actually compiled VI, meaning the packets must match the connected I/O. The recommended way to edit the file is to copy our XML file from: Dropbox\TMR4243 - LAB\04 cRIO software\FPGA IO. You will have to make sure that the name of your bitfile matches the name in the XML file as seen in Figure 54, also make sure the I/O modules matches your setup.
16. Copy the bitfiles from the bitfile folder to the level above so that the bitfile aand the XML file is in the same folder.

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

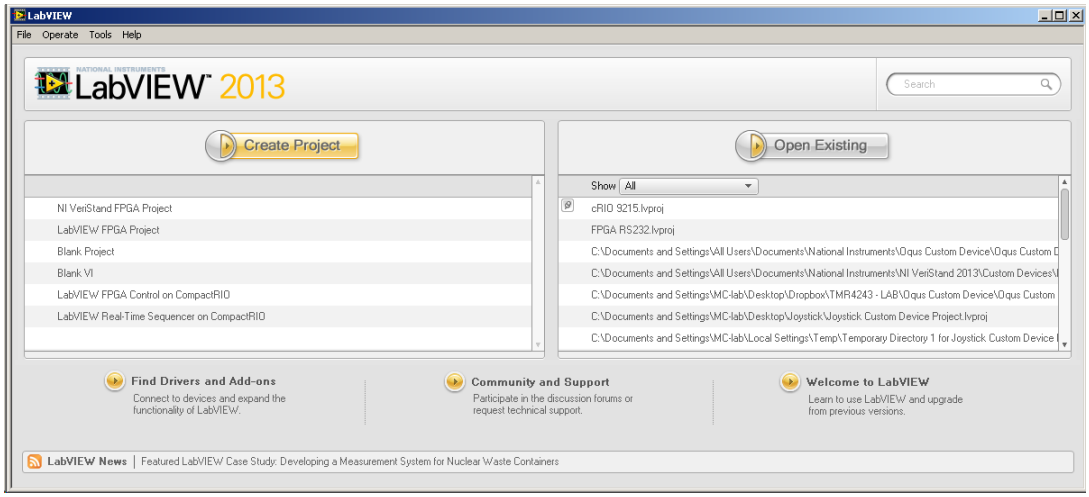


Figure 38: Create Labview FPGA target and XML - 1

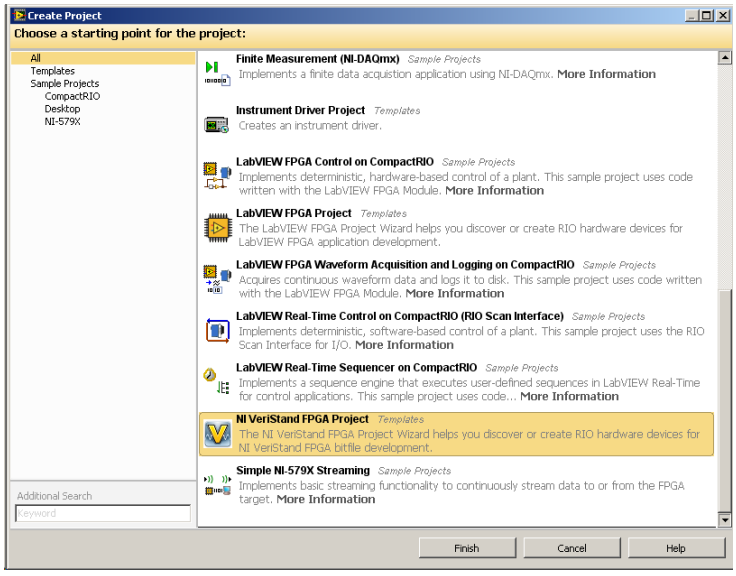


Figure 39: Create Labview FPGA target and XML - 2

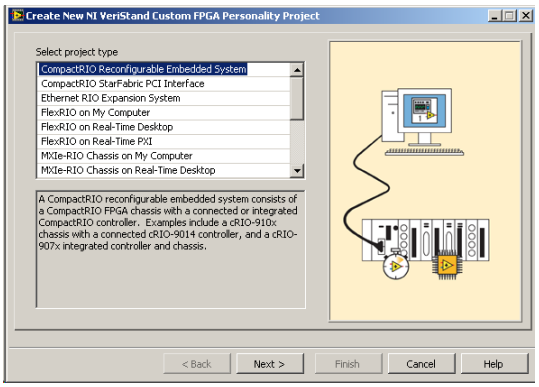


Figure 40: Create Labview FPGA target and XML - 3

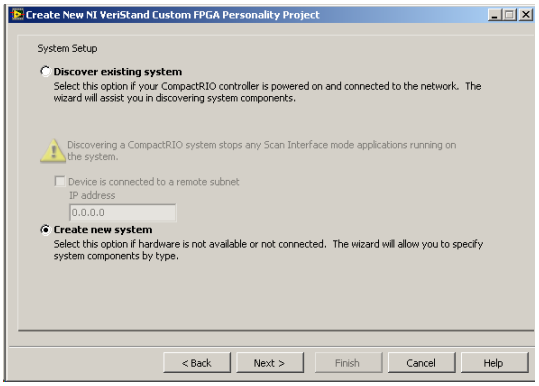


Figure 41: Create Labview FPGA target and XML -4

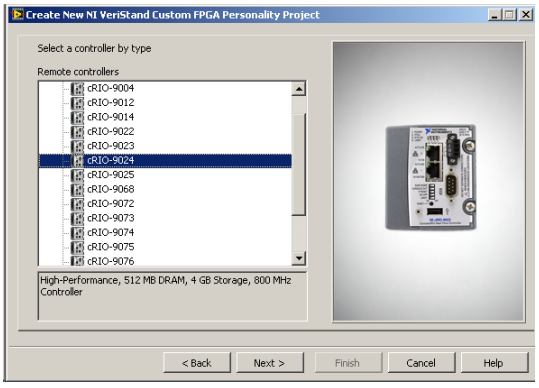


Figure 42: Create Labview FPGA target and XML - 5

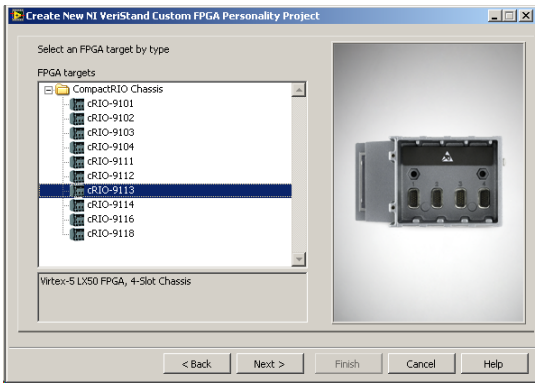


Figure 43: Create Labview FPGA target and XML - 6

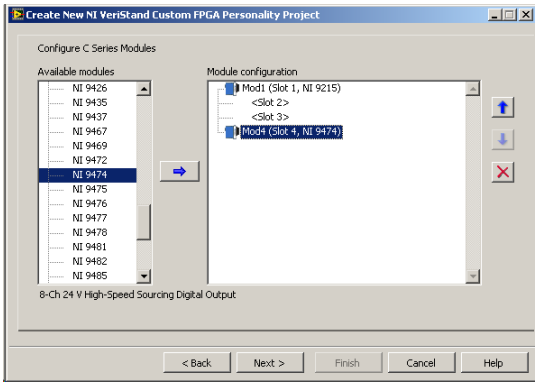


Figure 44: Create Labview FPGA target and XML - 7

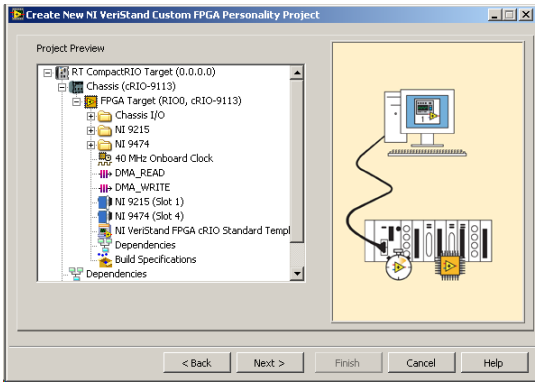


Figure 45: Create Labview FPGA target and XML - 8

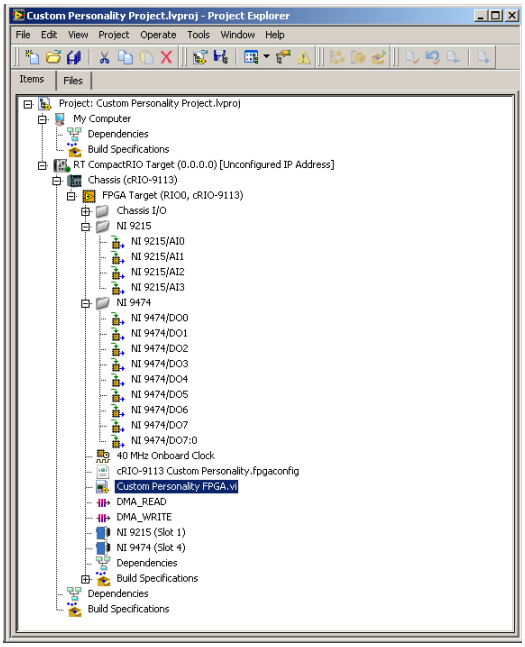


Figure 46: Create Labview FPGA target and XML -9

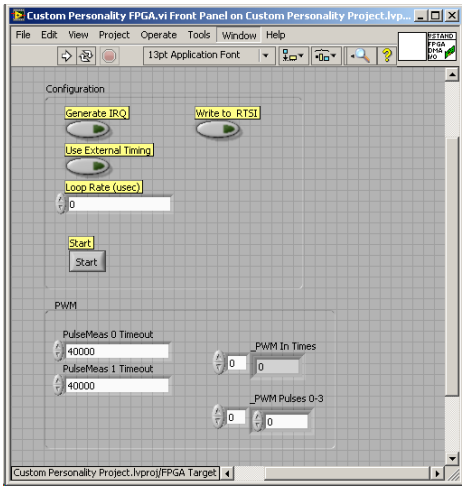


Figure 47: Create Labview FPGA target and XML - 10

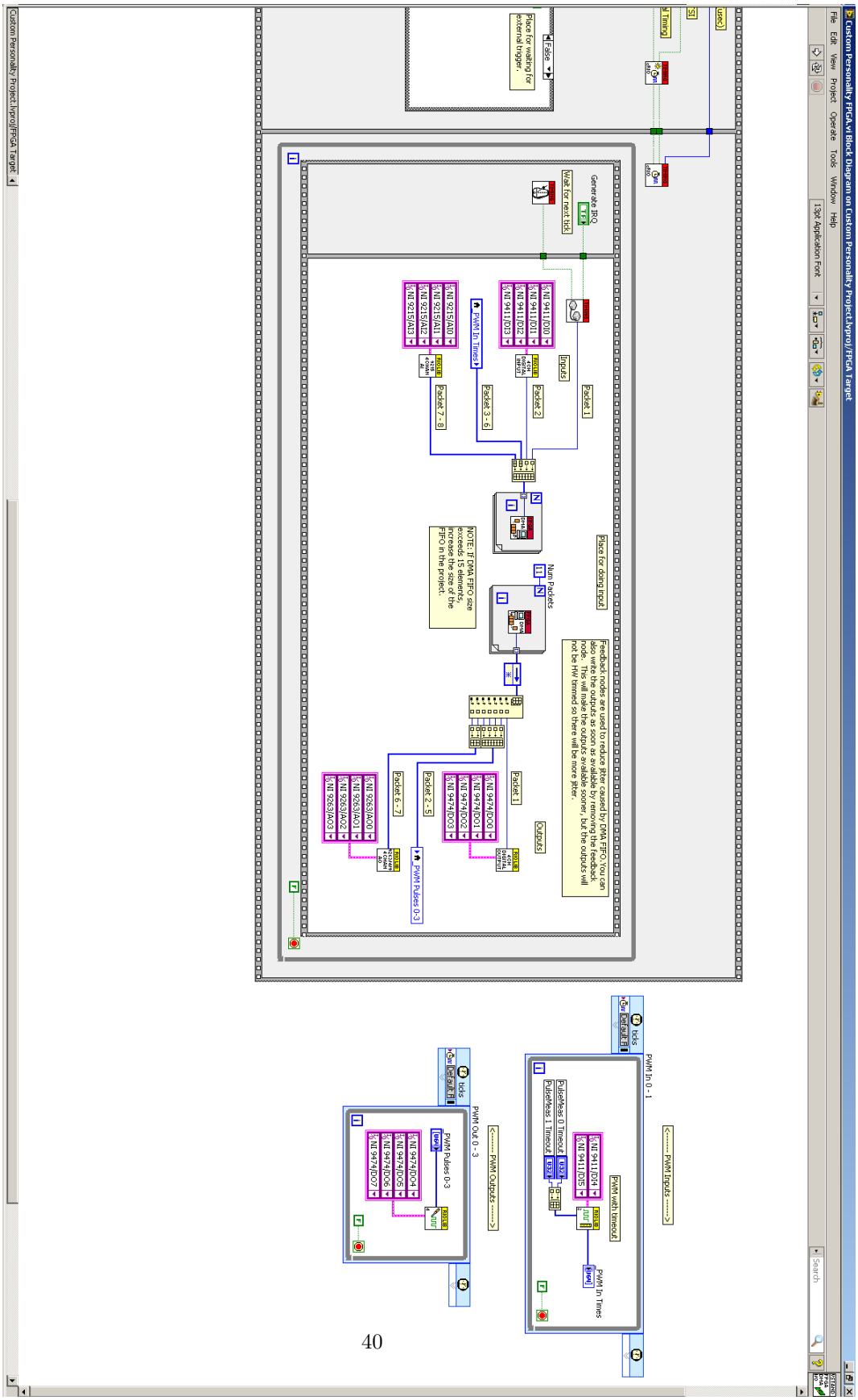


Figure 48: Create Labview FPGA target and XML - 11

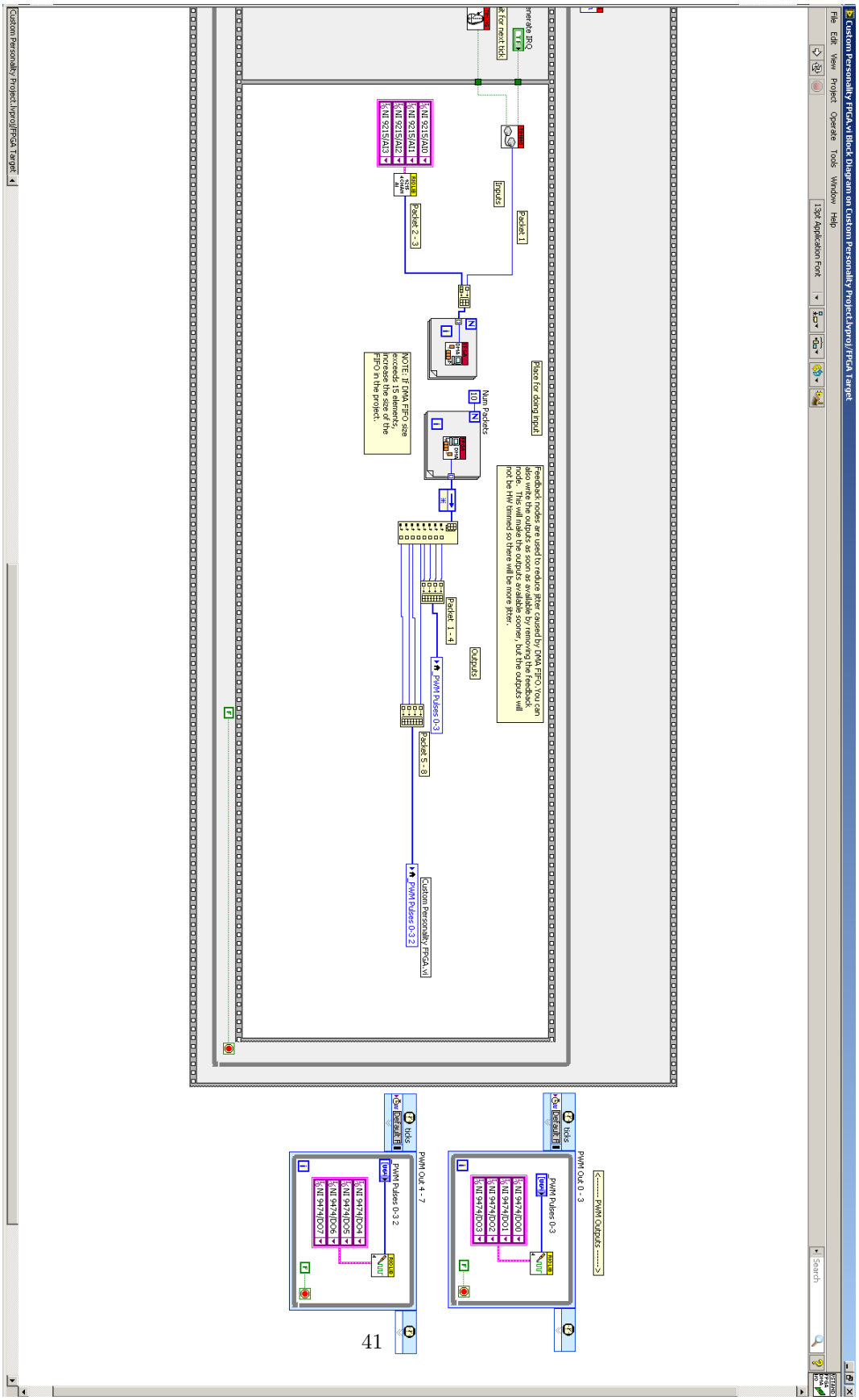


Figure 49: Create Labview FPGA target and XML - 12

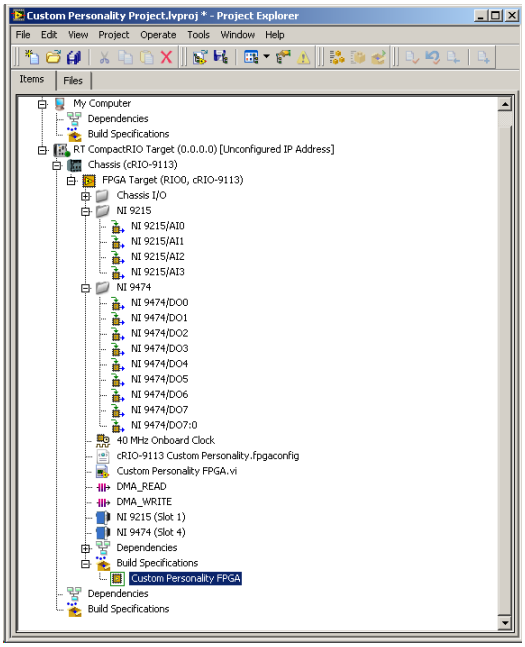


Figure 50: Create Labview FPGA target and XML - 13

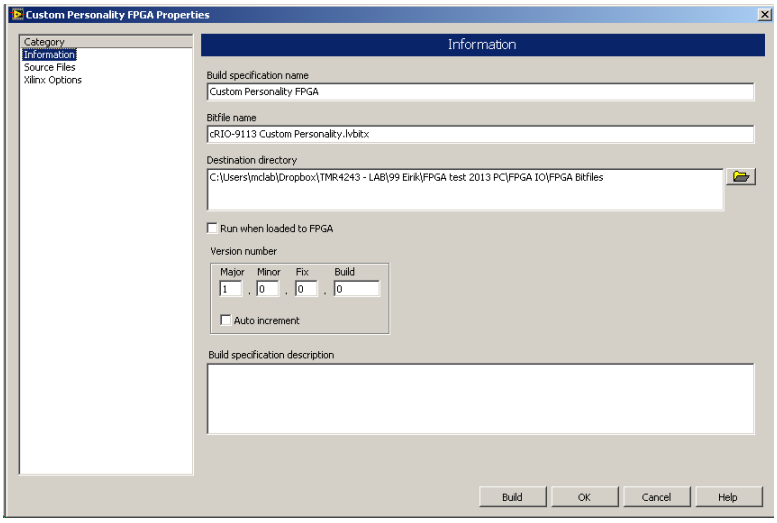


Figure 51: Create Labview FPGA target and XML - 14

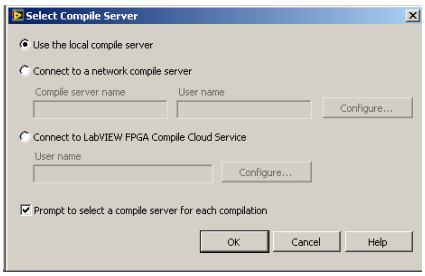


Figure 52: Create Labview FPGA target and XML - 15

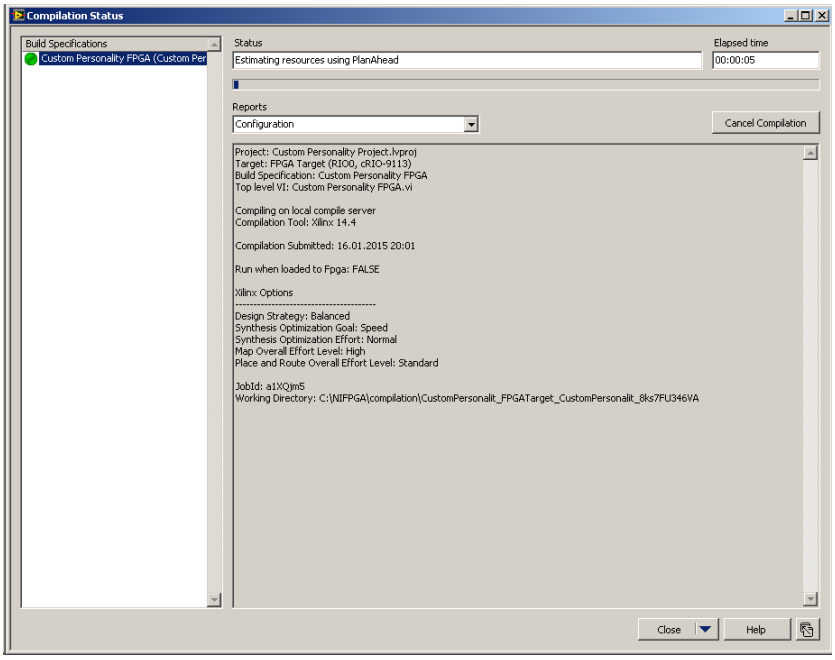


Figure 53: Create Labview FPGA target and XML - 16

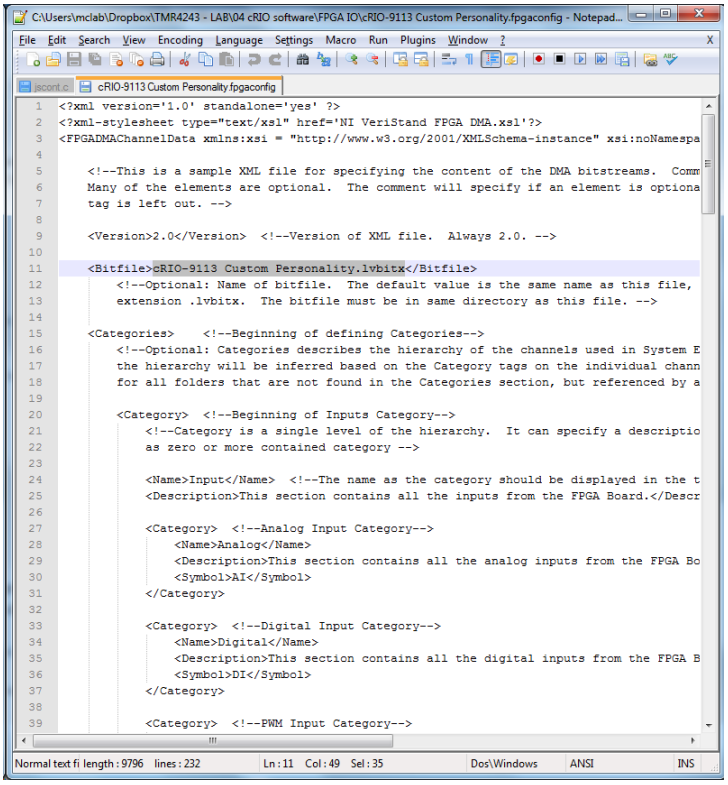


Figure 54: Create Labview FPGA target and XML - 17

A.4.2 Install in veristand

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

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 55.

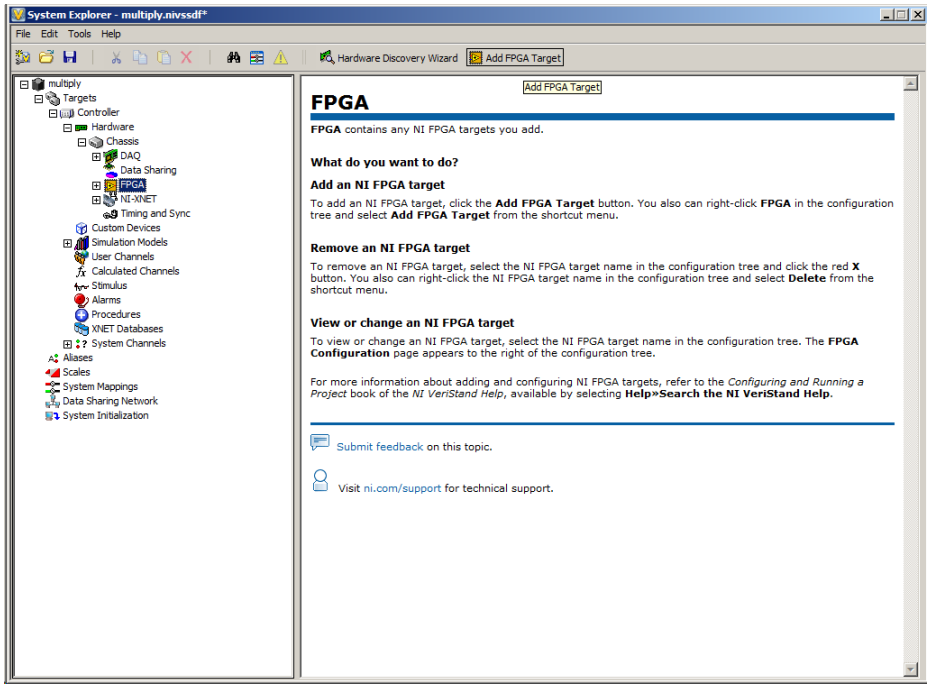


Figure 55: FPGA1

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 56, 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 Figure37.

PWM

A.4.3 Ticks og sånt

tick = FPGA clock pulse

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

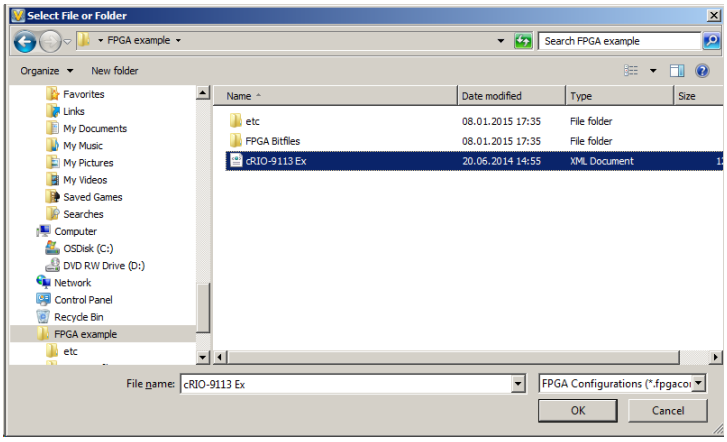


Figure 56: FPGA2

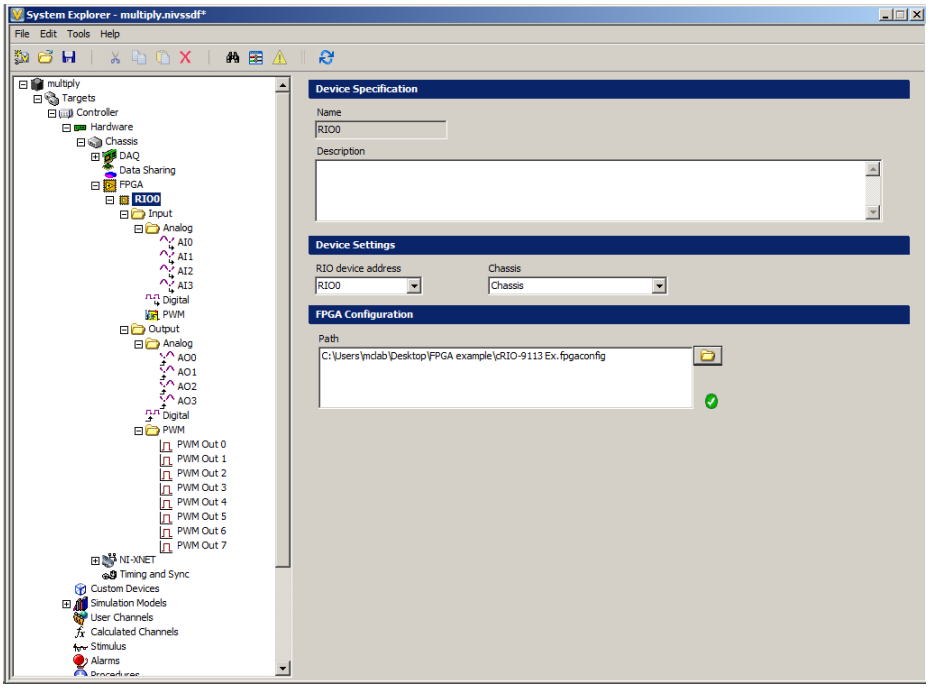


Figure 57: FPGA3

output at 50 Hz demands output every

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

B Raspberry Pi

B.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.

B.1.1 Download operating system and utilities

Download and extract the newest Raspbian[6] operating system (OS) image.

Necessary utilities for the setup are

- Win32 Disk Imager[7] to write the OS image to the RPi SD card
- Advanced IP scanner[8] to find the RPi address on the network
- Putty terminal emulator[9] for SSH connection
- WinSCP[10] for file transfer

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

Table 2: RPi installation and setup utilities

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

B.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.

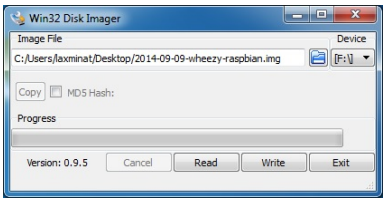


Figure 58: Disk Imager

[6] raspberrypi.org/downloads
[7] sourceforge.net/projects/win32diskimager
[8] by Famatech, advanced-ip-scanner.com
[9] www.chiark.greenend.org.uk/~sgtatham/putty/download.html
[10] by Martin Prikryl, winscp.net/eng/download.php

Run the program as administrator. Select the correct image file and device, as in Figure 58. Make sure that you have slected the correct drive before you push WRITE.

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

B.1.3 Terminal access

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

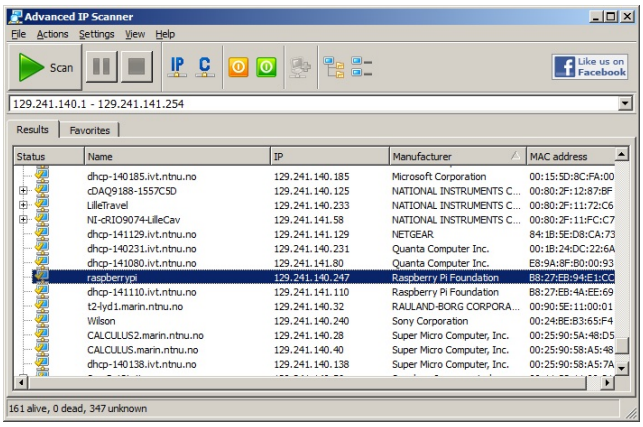


Figure 59: Advanced IP Scanner

At first boot, the RPi by default waits to be assigned an IP address by DHCP. If this address is not known, scan the network with Advanced IP Scanner. It is advicible to sort the results by manufacturer since it is fixed (*Raspberry Pi Foundation*). The name is typically *raspberrypi*. See Figure 59.

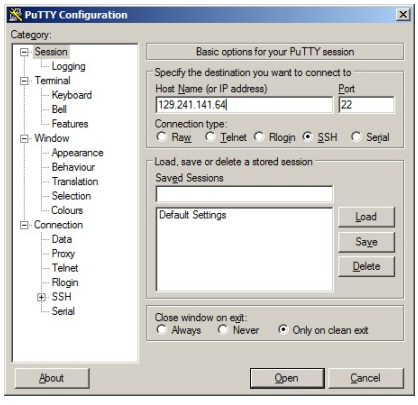


Figure 60: Putty settings

Once the IP is known, it is specified in the Putty settings, as in Figure 60, and a connection can be opened.

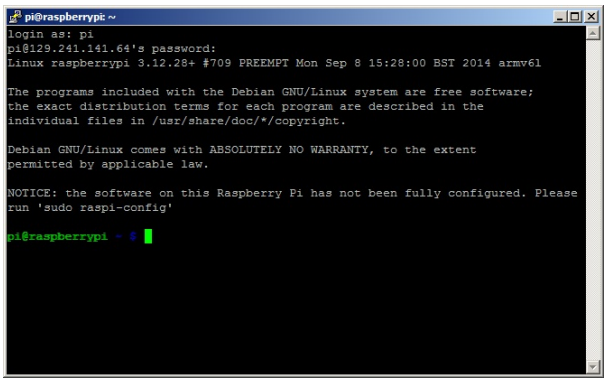


Figure 61: SSH connection

The default login is `pi`, and the default password `raspberry`. Figure 61 shows the terminal output on first login.

B.1.4 Finalize configuration

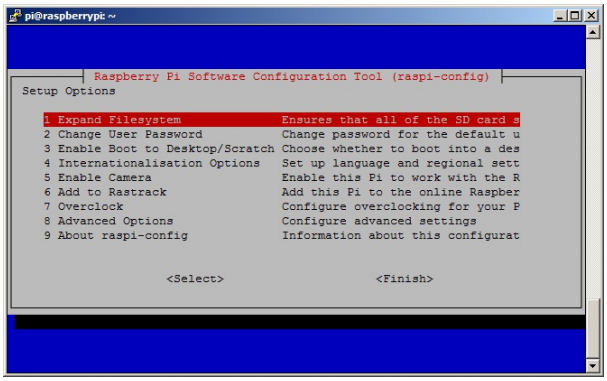


Figure 62: RPi configuration tool

Enter the

```
sudo raspi-config
```

command to start the RPi Software Configuration Tool, as in Figure 62. 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.

B.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.

B.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.

B.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.

B.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.

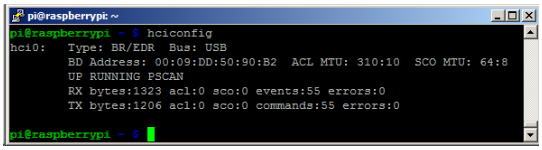


Figure 63: Bluetooth configuration tool

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

Most experienced errors were due to typos.

B.2.2 Bluetooth pairing

Sixaxis does not support the standard Bluetooth paring prcedure, instead, pairing is done over USB. The `sixpair` command-line utility[11] searches USB buses for Sixaxis devices and tells them to connect to a new Bluetooth master.

Download and compile the program by the following commands:

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

Connect the Sixaxis by USB before running the paring utility

```
sudo ./sixpair
```

The output should be similar to

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

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

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

The Sixaxis USB cable may now be disconnected.

B.2.3 Joystick manager system service

`QtSixA`[12] 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.

B.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 B.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 64. To exit the server use the key combination Ctrl+C.

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

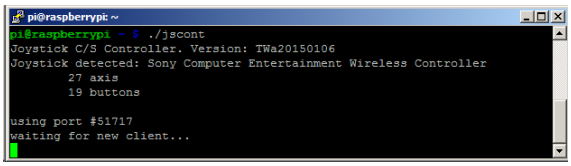


Figure 64: Joystick signal server test

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

1. Open the file for editing

   `sudo nano /etc/inittab`

2. Change the line that reads

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

   by adding `--autologin pi` to get

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

   Warning: Typos here may result consequences hard to correct.

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

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

1. Open the file for editing

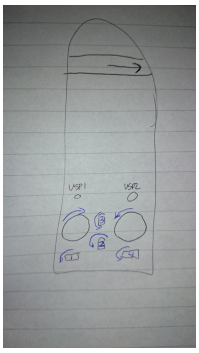
   `sudo nano /home/pi/.bashrc`

2. At the very end of the file, add

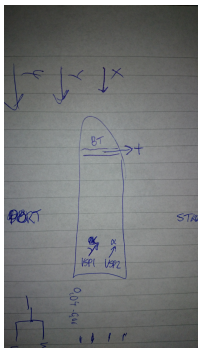
   `sudo ./jscont`

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

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



(a) Measurments



(b) Second extrapolation

Figure 65: Servo, rod position tuning

C CSE1

Increased servo percentage results in clockwise? motion.

Hystersis on motors

C.1 Actuators

Antall, posisjon, aktivert med pwm.

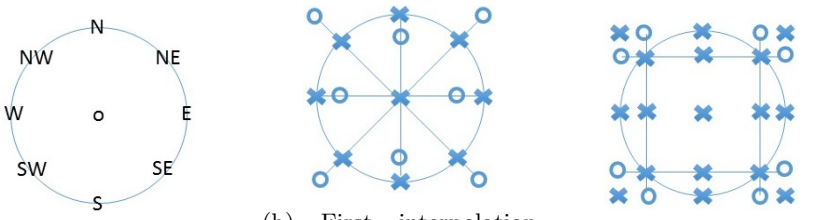
Port	Component
pwm0	Bow thruster motor
pwm1	VSP1 motor
pwm2	VSP2 motor
pwm3	not in use
pwm4	servo1
pwm5	servo2
pwm6	servo3
pwm7	servo4

Table 3: 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.



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

Figure 66: Servo, rod position tuning

C.1.1 Motor control signals

C.1.2 Servo control signals

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 4: Servo pwm ranges

Ikke lineært, ikke rett frem.

Foreslått metode

C.2 Measurements

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

Table 5: CSE1 cRIO analog input

C.3 Control software

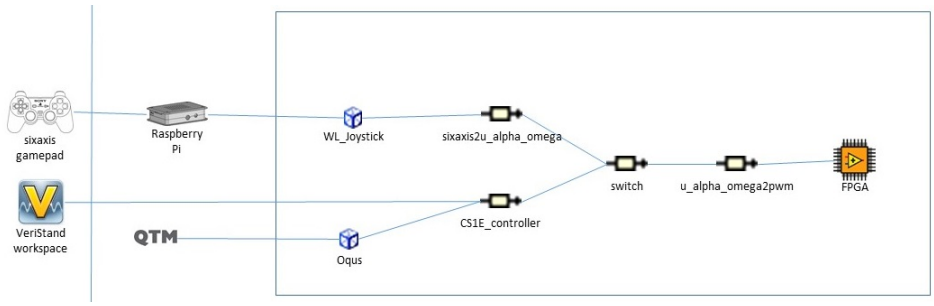


Figure 67: Software module communication

C.3.1 sixaxis2uao

maps

1. joysticks to individual VSP angle and arm
2. L/R3 to tunnel thruster

Discrete block, no states

C.3.2 sixaxis2tau

maps Sixaxis to resultant thrust forces

1. left joystick to tau_x
2. right joystick to tau_y
3. L/R3 to tau_psi

C.3.3 CS1Ectrl

contains motion control such as dynamic positioning and thruster allocation

C.3.4 switch

switches between

1. Sixaxis thruster control
2. Sixaxis force control
3. Fallback standard DP mode
4. CS1Ectrl

C.3.5 uao2pwm

transforms angles and actuation amount to pwm signals

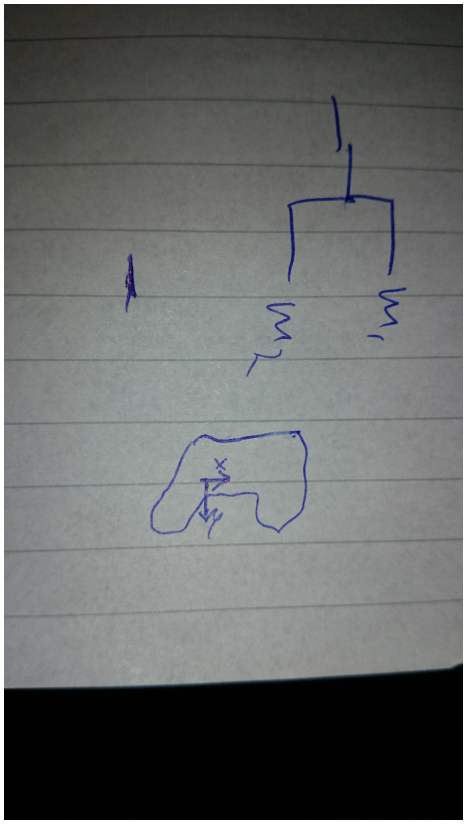


Figure 68: Sixaxis coordinate system

From	**Sixaxis gamepad**	**sixaxis2ualphaomega**	ualphaomega2pwm	**FPGA**	CSE
To	**sixaxis2ualphaomega**	**ualphaomega2pwm**	**FPGA**	**Digital out**	
	$L2_{\mathbf{cont}}$, $R2_{\mathbf{cont}}$	$u_{\mathbf{BT}}$	$\mathbf{pwm_{BT}}$	**pwm0**	Bow
	arrow_up arrow_down	$\omega_{\mathbf{VSP1}}$	$\mathbf{pwm_{VSP1}}$	**pwm1**	VSP
		$\omega_{\mathbf{VSP2}}$	$\mathbf{pwm_{VSP2}}$	**pwm2**	VSP
	PosXLeft, PosYLeft	$u_{\mathbf{VSP1}}\alpha_{\mathbf{VSP1}}$	$\mathbf{pwm_{servo1}}$	**pwm4**	servo
			$\mathbf{pwm_{servo2}}$	**pwm5**	servo
	PosXRight, PosYRight	$u_{\mathbf{VSP2}}\alpha_{\mathbf{VSP2}}$	$\mathbf{pwm_{servo3}}$	**pwm6**	servo
			$\mathbf{pwm_{servo4}}$	**pwm7**	servo

Table 6: Connections

D Qualisys

calibration

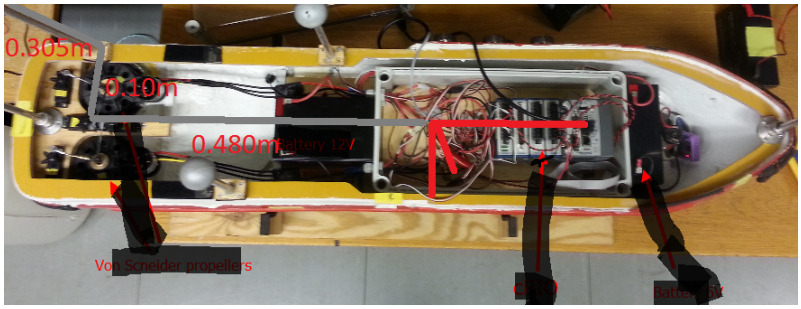


Figure 69: Matlab console

leverer med 50Hz på nettet

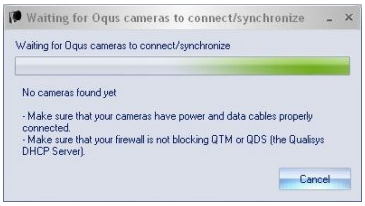


Figure 70: Matlab console

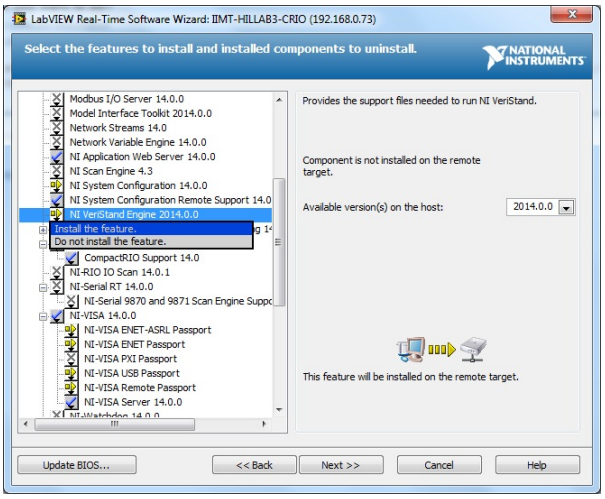


Figure 71: FPGA2

Part VI

Miscellaneous

E HIL lab and MC lab device network addresses

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

Table 7: IP addresses

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

Note: to connect the RPi directly to the computer, both need to be on the same

domain and the computer IP thus needs to change to 192.168.**1**.xx.

F Checklist

Device driver in place

Custom indicators in place

PC and cRIO IP correct

Sixaxis charged

G Personel and literature

G.1 Points of contact

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

Øivind Kåre Kjerstad Build CSE1

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

Dinh Nam Tran oppryddingsarbeid

Andreas Orsten brukt mye, skrevet artikel om sleping av isberg

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

Eirik Valle Teaching assistant TMR4243, Sixaxis for RPi setup

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

Jostein Follestad Teaching assistant TMR4243, CS1E HIL model

Fredrik Sandved Teaching assistant TMR4243, Custom displays

G.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).

G.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.

G.4 Other

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

H Maintenance

Oil VSP

I Suppliers

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

Table 8: Suppliers

J To do list

- Etablere fargekoder for simulinkblokker (spesielt "ikke røre"-farge)
- Konsekvent notasjon: C/S Enterprise 1 eventuelt CSE1
- Forklaring av hva realtime betyr i HW og SW
- Troubleshooting-prosedyrer for de vanligste feilene
- Implementere "fail to zero" for når kommunikasjonen avbrytes.
- legge til IMU/gyro på båten

K Software

Compatability between software is very important, See NI VeriStand Version Compatibility KnowlegeBase[13].

K.1 Order of installation

1. Microsoft .NET
2. Microsoft SDK
3. Matlab
4. Labview
5. Veristand
   (a) Including NI VeriStand Model Framework!
6. Additional for model compilation
   (a) VxWorks:
      i. WindRiver GNU Toolchain[14]
      ii. Real-Time Workshop software
   (b) PharLap:
      i. Microsoft Visual C++
      ii. The MathWorks, Inc. Real-Time Workshop® software

K.2 needed

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

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