

NXTbike-GS



Joep Mutsaerts
Mechatronic System Design II
WB 1185756

May, 18 2010

Disclaimer

LEGO® is a trademark of the LEGO Group of companies which do not sponsor, authorize or endorse this project. LEGO® and Mindstorms® are registered trademarks of The LEGO Group.

According to LEGO Mindstorms NXT Hardware Developer Kit,

“Important note: When the NXT is disassembled or when third party firmware is used with the NXT, all warranties are rendered invalid”.

Therefore, make sure that the author of this document does not take any responsibility for any loss or damage of any kind incurred as a result of the use or the download of this document, nxtOSEK, Embedded Coder Robot NXT and other related third party software/hardware.

This document extends on software and control as used in lejos-osek NXTway-GS set-up. View this here: http://lejos-osek.sourceforge.net/nxtway_gs.htm

This document is not intended for commercial purposes.

Author (1st edition NXTbike-GS)

J.T.M. Mutsaerts

J.T.M.Mutsaerts@student.tudelft.nl

Delft University of Technology

In cooperation with Bicycle Dynamics Lab, <http://www.bicycle.tudelft.nl>.

In assignment of Jo W. Spronck, Associate professor.

Author of NXTway-GS

Yorihisa Yamamoto

Revision History

Revision	Date	Description	Author
1.0	May 21, 2010	NXTbike-GS first report	J.T.M. Mutsaerts

1 Introduction

In the beginning of February 2010, Ir. J.W. Spronck suggested to build a bike out of Lego NXT parts. The idea that this would work, came from the already working Lego NXTway-GS that can be found on various video websites. The start of the project was building a Lego segway (two wheels inverted pendulum) and later convert this into a bike.

The already existing Lego segway uses Matlab and Simulink development environments to compile a simulink control scheme into a C programmed code. The ARM processor in the NXT Brick can use this code to control the system. After showing the successful function of the segway set-up, a Lego Bike has been made. This report shows the characteristics of the NXTbike-GS, as well as the control scheme, JBIke6 parameters and future developments.

2 Preparation

To build the NXTbike-GS, read the NXTbike-GS Building Instructions.

You need to download Embedded Coder Robot NXT from the following URL because it is used as Model-Based Design Environment in this document.

<http://www.mathworks.com/matlabcentral/fileexchange/13399>

Read Embedded Coder Robot NXT Instruction Manual (Embedded Coder Robot NXT Instruction En.pdf) and test sample models / programs preliminarily. The software versions used in this document are as follows.

Software	Version
Embedded Coder Robot NXT	3.14
nxtOSEK (previous name is LEJOS OSEK)	2.03
GNU ARM	4.0.2
Cygwin	1.5.24

Commercial software

- for the NXTbike-GS, release R2008b is used
- **Real Time Workshop Embedded Coder is not included in TU Delft package. Get it yourself!**

Product	Version
MATLAB®	7.5.0
Control System Toolbox	8.0.1
Simulink®	7.0
Real-Time Workshop®	7.0
Real-Time Workshop® Embedded Coder	5.0
Fixed-Point Toolbox (N1)	2.1
Simulink® Fixed Point (N1)	5.5
Virtual Reality Toolbox (N2)	4.6

You can simulate original NXTway-GS models and generate codes from it without the products (N1) and (N2). The meaning of these notes:

(N1) : It is required to run fixed-point arithmetic controller model (nxtway_gs_controller_fixpt.mdl).

(N2) : It is required to run 3D visualization (nxtway_gs_vr.mdl).

Required files:

File	Description
Nxtbike_controller.m*	NXTBike-GS model
iswall.m	M-function for detecting wall in map
mywritevrtrack.m	M-function for generating map file (track.wrl)
Simulink®	7.0
nxtway_gs.mdl	NXTway-GS model (It does not require Virtual Reality Toolbox)
nxtway_gs_controller.mdl	NXTway-GS controller model (single precision floating-point)
nxtway_gs_controller_fixpt.mdl	NXTway-GS controller model (fixed-point)
nxtway_gs_plant.mdl	NXTway-GS plant model
nxtway_gs_vr.mdl	NXTway-GS model (It requires Virtual Reality Toolbox)
param_controller.m	M-script for controller parameters
param_controller_fixpt.m	M-script for fixed-point settings (Simulink.NumericType)
param_nxtway_gs.m	M-script for NXTway-GS parameters (It calls param_***.m)
param_plant.m	M-script for plant parameters
param_sim.m	M-script for simulation parameters
track.bmp	map image file
track.wrl	map VRML file
vrnxtwaytrack.wrl	map & NXTway-GS VRML file

- * **The only new file added to the NXTway-GS package. This file contains the whole bicycle controller.**

3 NXTbike-GS system

This chapter describes the structure and sensors/actuators of the NXTbike-GS.

3.1 Structure



The figure above shows the active parts of the NXTbike-GS. A Hitechnics Gyro Sensor measures the angular roll velocity. The Steer DC Motor is able to actively steer the NXTbike-GS. Because of friction in the motor, the steer is not freely rotating and therefore the natural stability of a bicycle in a certain velocity region will not occur.

3.2 Sensors and actuators

The table underneath shows the used sensors in the NXTbike-GS set-up.

Sensor	Output	Unit	Data type	Max Sample / sec
Rotary Encoder	angle	deg	Int32	1000
Gyro Sensor	Angular velocity	Deg/sec	Uint16	300

The next table shows the used actuators.

Actuator	Input	Unit	Data type	Max Sample / sec
DC Motor	PWM	%	Int8	500

Please take into account that sensors are different individually and the gyro sensor has gyro offset (the value when the system does not rotate) and gyro drift (the time variation of the gyro offset)

4 NXTbike-GS modeling

For the modeling of a normal bicycle, the JBIke6 model is often used. For this model, the parameters are given in the next table.

Physical parameter	Unit	Description
$g=9.81$	[m/sec ²]	Gravity acceleration
$m=0.03$	[kg]	Mass of wheel
$R=0.04$	[m]	Radius of wheel
$M=0.678$	[kg]	Total weight of NXTbike-GS
$M_{\text{front}}=0.2$	[kg]	Weight on front wheel
$M_{\text{rear}}=0.478$	[kg]	Weight on rear wheel
$L_{\text{base}}=0.225$	[m]	Wheel base
$L_{\text{trail}}=0.011$	[m]	Trail
$\text{Head_angle}=67$	[deg]	Head angle
$V=0.6$	[m/sec]	Velocity of NXTbike-GS

5 Control scheme

This chapter shows the control scheme behind the NXTbike-GS.

Used variables

Parameter	Unit	Description
theta	[deg]	Rear wheel position
thetadot	[deg/sec]	Rear wheel speed
phi	[deg]	Body roll angle
phidot	[deg/sec]	Body roll angular velocity
phidotdot	[deg/sec ²]	Body roll angular acceleration
delta	[deg]	Steer angle
X1	[array]	[phi phidot phidotdot] state variables

Inputs and Outputs

Input	Sensor	Description
Theta	DC Motor	Rear wheel position
Delta	DC Motor	Steer position
Phidot	Gyro Sensor	Angular roll velocity

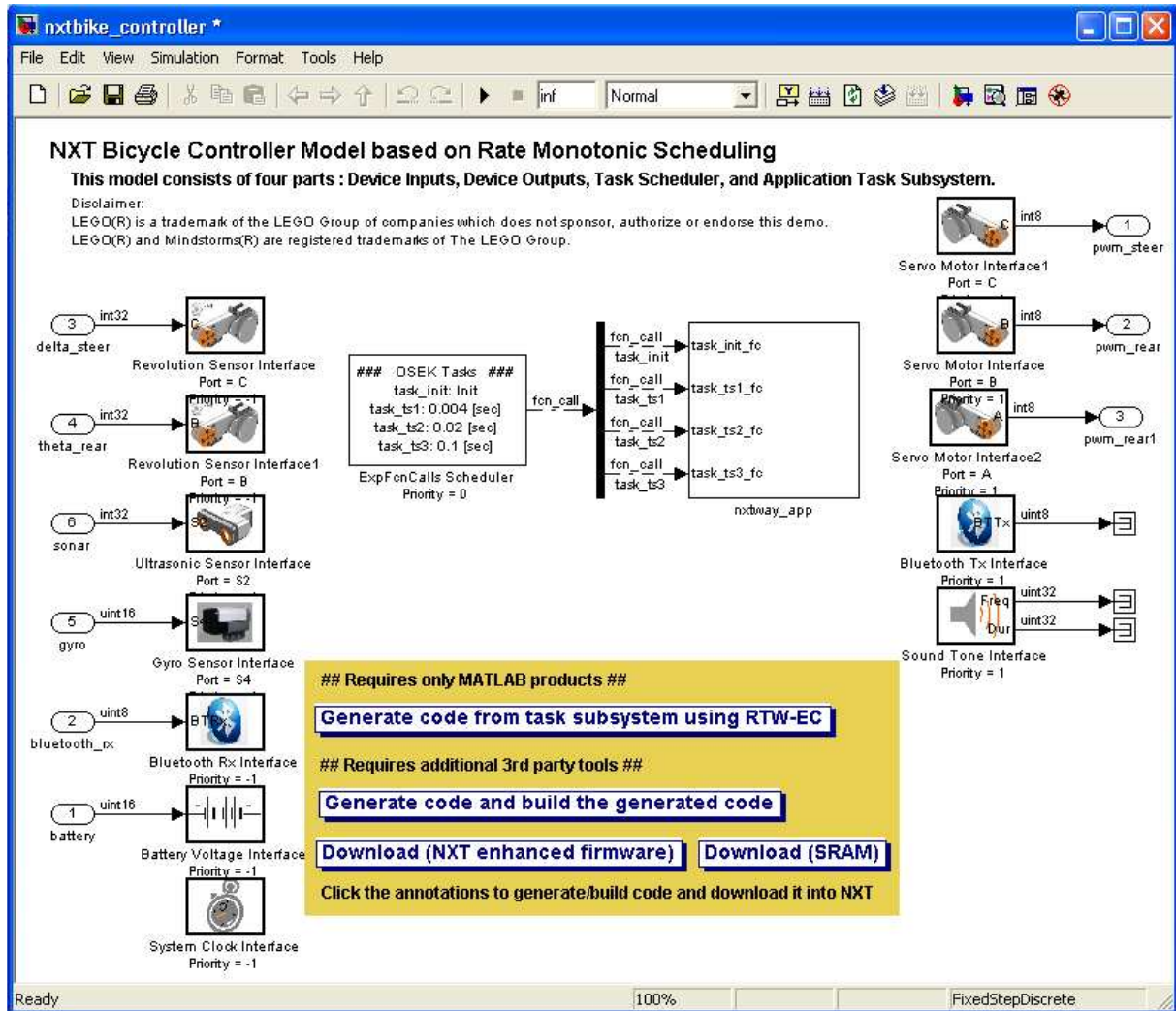
Output	Actuator	Description
PWM_theta	DC Motor	Rear wheel voltage
PWM_delta	DC Motor	Steer voltage

Stability

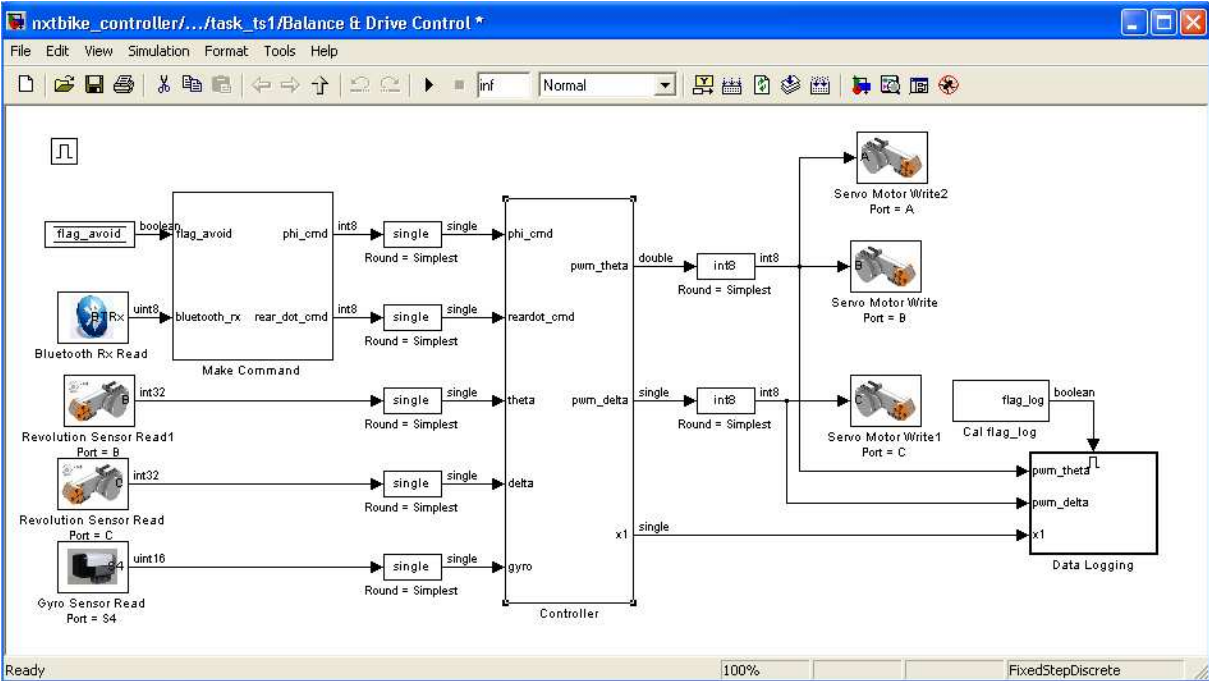
Of course the NXTbike-GS is not stable without any control. The NXTbike-GS has to steer into the same direction as in which it is falling. This is done with proportional gain on the roll angular velocity into the steer DC motor.

5.1 Main control scheme

The Simulink environment below is exactly the same as the one of the NXTway-GS. Off course, one DC motor is added for the rear wheel. On the left and right side, one can clearly distinguish the in/outputs. In this window, code generation, building and downloading through USB interface is easily done.



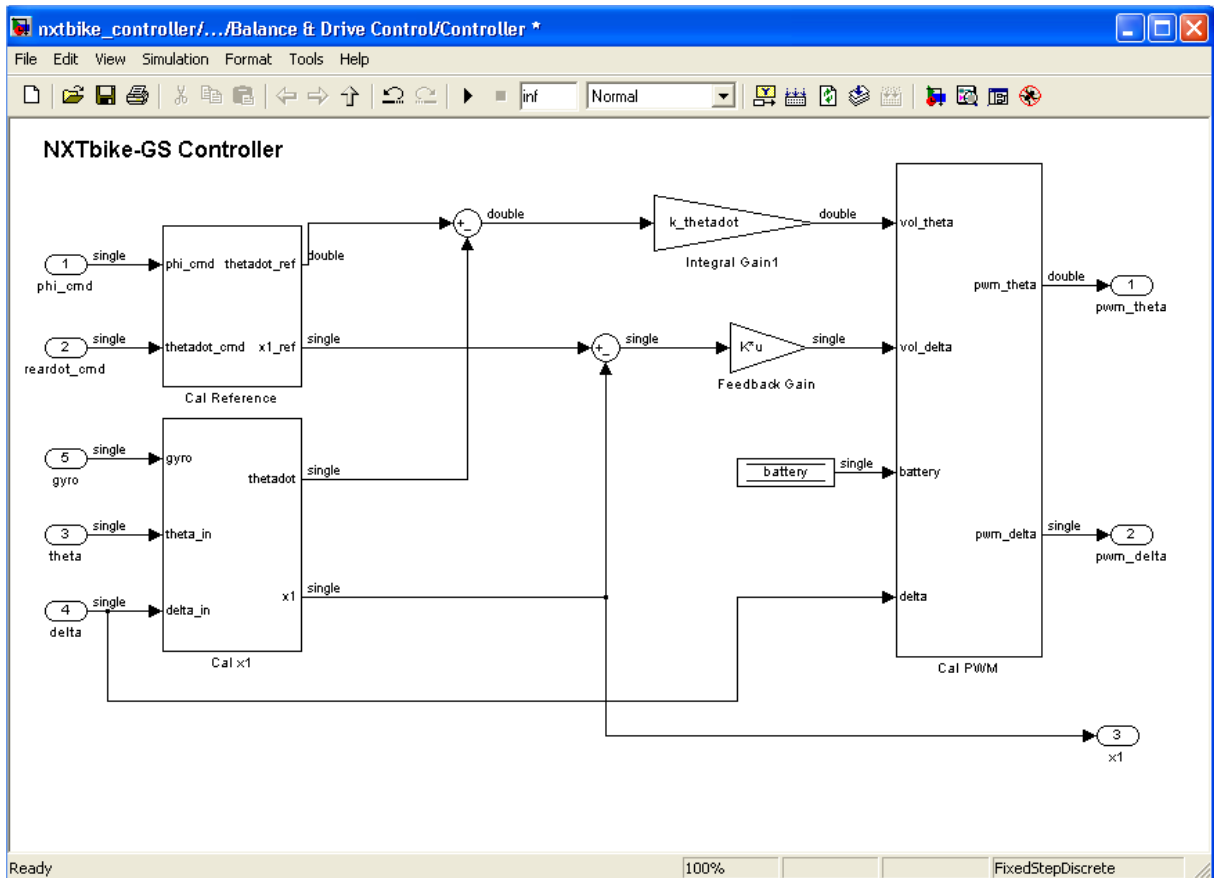
Now the next scheme shows the global controller layout, with the inputs and outputs. Also notice signals leading to data logging, this is a datalog connection over Bluetooth. The DC Motor A and B both receive the same PWM control.



In the next window, the control scheme is shown. Go here through :
 nxtway_app / Balance & Drive control / Balance & Drive control / Controller

The upper Gain: k_{thetadot} is used to P control the speed of the rear wheel to the setpoint of thetadot_ref . The second Gain is multiplied with the error signal ($x1_{\text{ref}} - x1$). Since $x1 = [\text{phi phidot phidotdot}]$, this is actually a PID control on the body roll angular velocity.

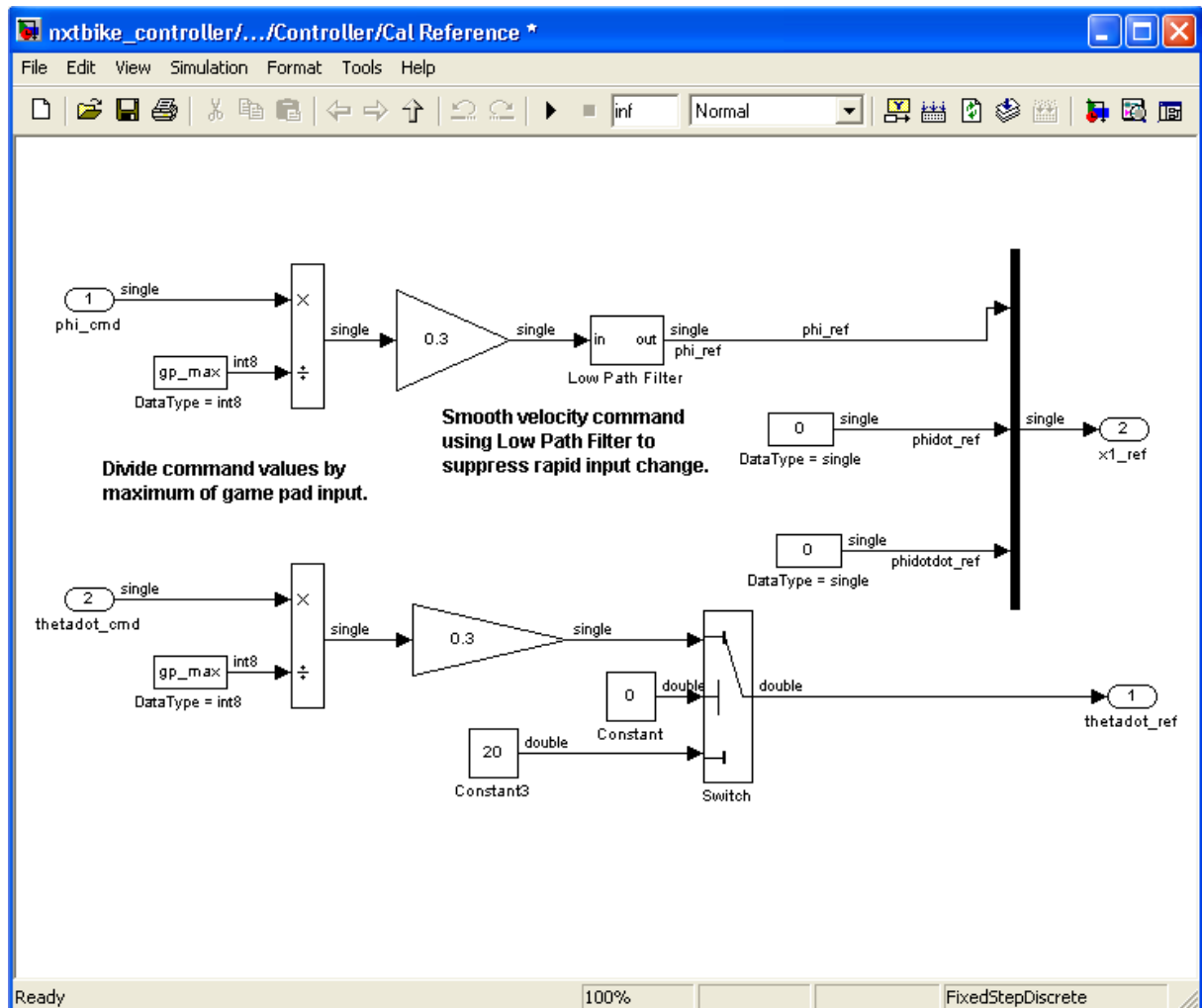
Now I have used $[-0.01 -8 -0.01]$ as second Gain value. This means there is no I and no D action. However, the bicycle has shown to be pretty stable, although there is a mechanical play in the steering wheel.



How to determine the reference

Here phi_cmd, the commanded roll angle is 0. However, when this angle would be changed, for example with the gamepad input, we could be able to steer.

Also thetadot_cmd is determined here. With a switch it is now set to the constant value. Also this speed could be adjusted by the gamepad, via Bluetooth.

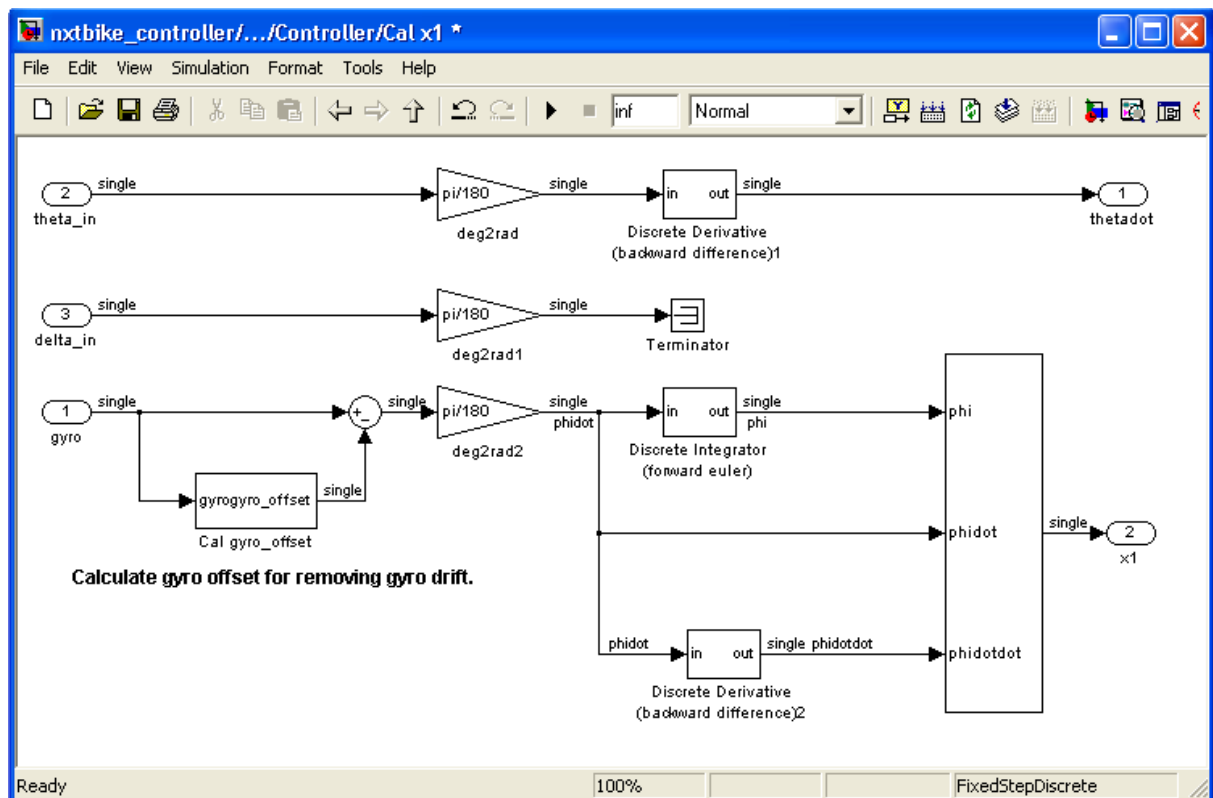


How to evaluate the input signals

As presented in the next scheme, the theta_in (rear wheel position) is converted into rad and differentiated, to obtain the speed thetadot (rad/sec). The derivative is done with respect to the sample time (0.004 ms), to obtain the speed in (rad/sec).

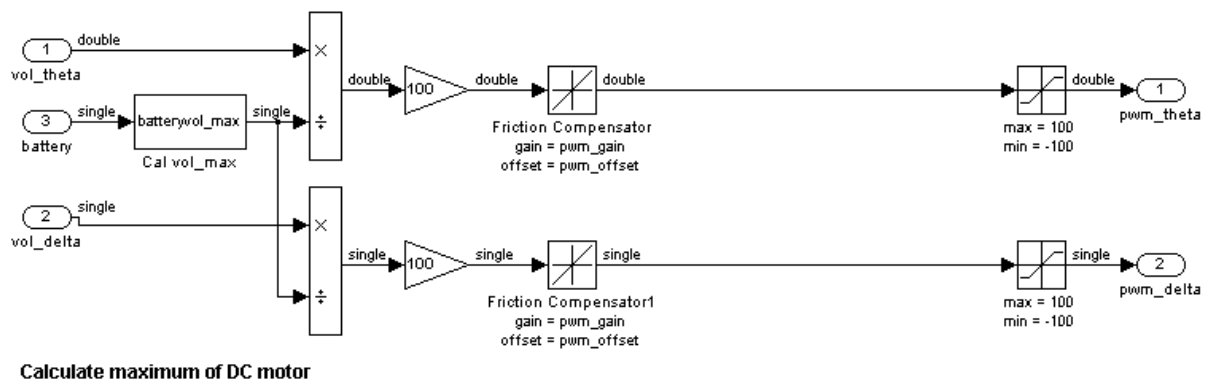
Delta_in, the feedback of the steer position, is not used. Therefore, when the bicycle fell down, it might keep rotating the steering wheel.

Gyro signal is corrected by its measured offset and then transformed into the x1 array. This is done by differentiating to phi_dotdot and integrating to phi. Both with respect to the sample time.



How the PWM signals are made

Now it is visible how the PWM volumes are corrected with the actual battery voltages, using an experimental formula. Also there is a friction compensation and signal limitation.



6 Challenges for further research

- Backlash in steering servo causes disturbances
- Path following by roll angle offset (I control)
- Dynamic gain, related to NXTbike-GS velocity
- Simulation in Matlab / Simulink with JBike6
- Performing a 'wheelie'