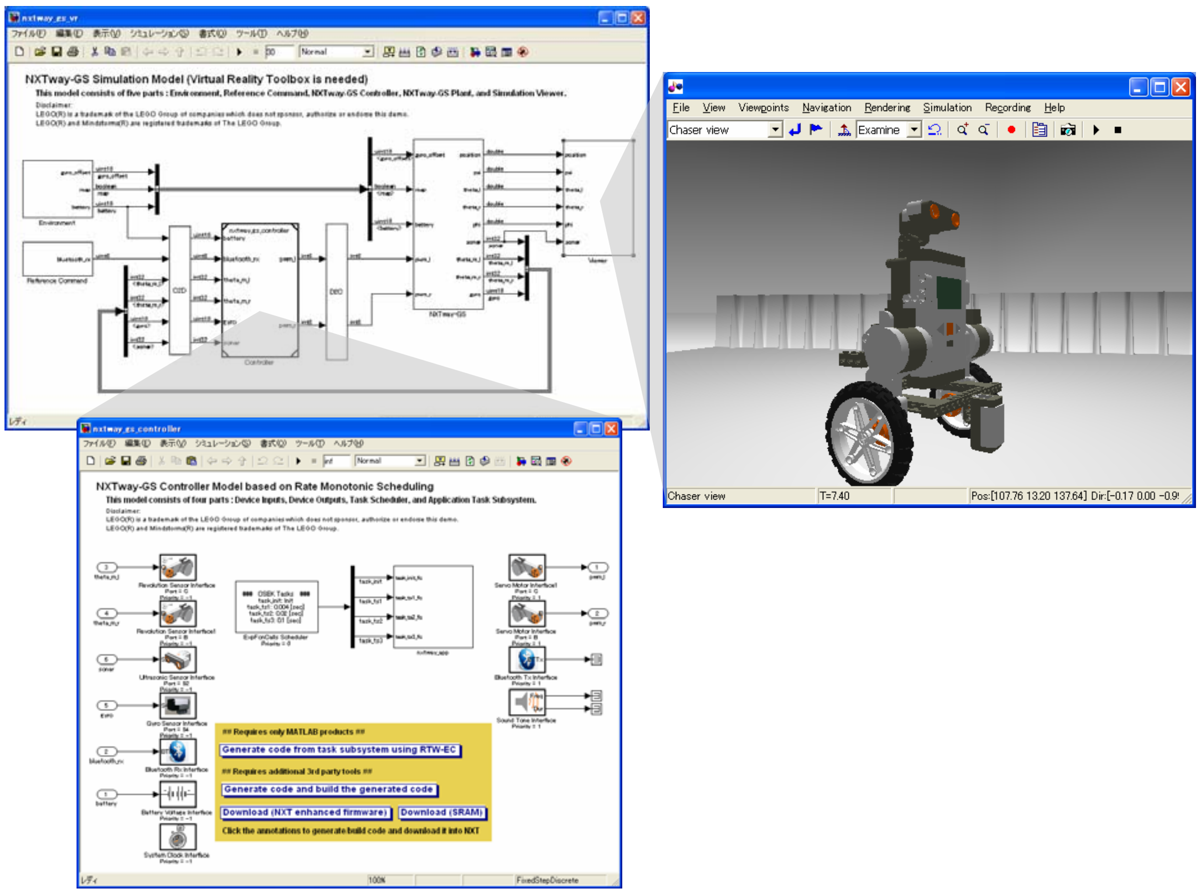
NXTway-GS Model-Based Design

- Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT -


nxtway_gs_vr
NXTway-GS Simulation Model (Virtual Reality Toolbox is needed)
This model consists of five parts : Environment, Reference Command, NXTway-GS Controller, NXTway-GS Plant, and Simulation Viewer.
Environment
Reference Command
Controller
NXTway-GS
Viewer
nxtway_gs_controller
NXTway-GS Controller Model based on Rate Monotonic Scheduling
This model consists of four parts : Device Inputs, Device Outputs, Task Scheduler, and Application Task Subsystem.
## Requires only MATLAB products ##
Generate code from task subsystem using RTW-EC
## Requires additional 3rd party tools ##
Generate code and build the generated code
Download (NXT enhanced firmware)
Download (SRAM)
Click the annotations to generate build code and download it into NXT
File View Viewpoints Navigation Rendering Simulation Recording Help
Chaser view
Examine
T=7.40

■ Author (First Edition)

Yorihisa Yamamoto

y_yama@cybernet.co.jp

Application Engineer

Advanced Support Group 1 Engineering Department

Applied Systems First Division

CYBERNET SYSTEMS CO., LTD.

■ Revision History

Revision	Date	Description	Author / Editor
1.0	29 Feb 2008	First edition	Yorihisa Yamamoto y_yama@cybernet.co.jp
1.1	3 Mar 2008	Added fixed-point controller model Updated software	Yorihisa Yamamoto y_yama@cybernet.co.jp
1.2	7 Nov 2008	Modified motion equations Modified annotations in controller models Updated software	Yorihisa Yamamoto y_yama@cybernet.co.jp
1.3	28 Nov 2008	Modified generalized forces Modified motion equations and state equations Added simulation movie	Yorihisa Yamamoto y_yama@cybernet.co.jp
1.4	1 May 2009	Modified text	Yorihisa Yamamoto y_yama@cybernet.co.jp

Introduction

NXTway-GS is a self-balancing two-wheeled robot built with LEGO Mindstorms NXT. This document presents Model-Based Design about balance and drive control of NXTway-GS by using MATALB / Simulink. The main contents are the following.

- Mathematical Model
- Controller Design
- Model Illustration
- Simulation & Experimental Results

Preparation

To build NXTway-GS, read NXTway-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
Cygwin	1.5.24
GNU ARM	4.0.2

Required Products

Product	Version	Release
MATLAB®	7.5.0	R2007b
Control System Toolbox	8.0.1	R2007b
Simulink®	7.0	R2007b
Real-Time Workshop®	7.0	R2007b
Real-Time Workshop® Embedded Coder	5.0	R2007b
Fixed-Point Toolbox (N1)	2.1	R2007b
Simulink® Fixed Point (N1)	5.5	R2007b
Virtual Reality Toolbox (N2)	4.6	R2007b

You can simulate NXTway-GS models and generate codes from it without the products (N1) and (N2).

(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).

File List

File	Description
iswall.m	M-function for detecting wall in map
mywritevrtrack.m	M-function for generating map file (track.wrl)
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

Table of Contents

1 Model-Based Design

This chapter outlines Model-Based Design briefly.

1.1 What is Model-Based Design?

Model-Based Design is a software development technique that uses simulation models. Generally, it is abbreviated as MBD. For control systems, a designer models a plant and a controller or a part of them, and tests the controller algorithm based on a PC simulation or real-time simulation. The real-time simulation enables us to verify and validate the algorithm in real-time, by using code generated from the model. It is Rapid Prototyping (RP) that a controller is replaced by a real-time simulator, and Hardware In the Loop Simulation (HILS) is a plant version of Rapid Prototyping.

Furthermore, auto code generation products like RTW-EC enables us to generate C/C++ code for embedded controllers (microprocessor, DSP, etc.) from the controller model. Figure 1-1 shows the MBD concept for control systems based on MATLAB product family.

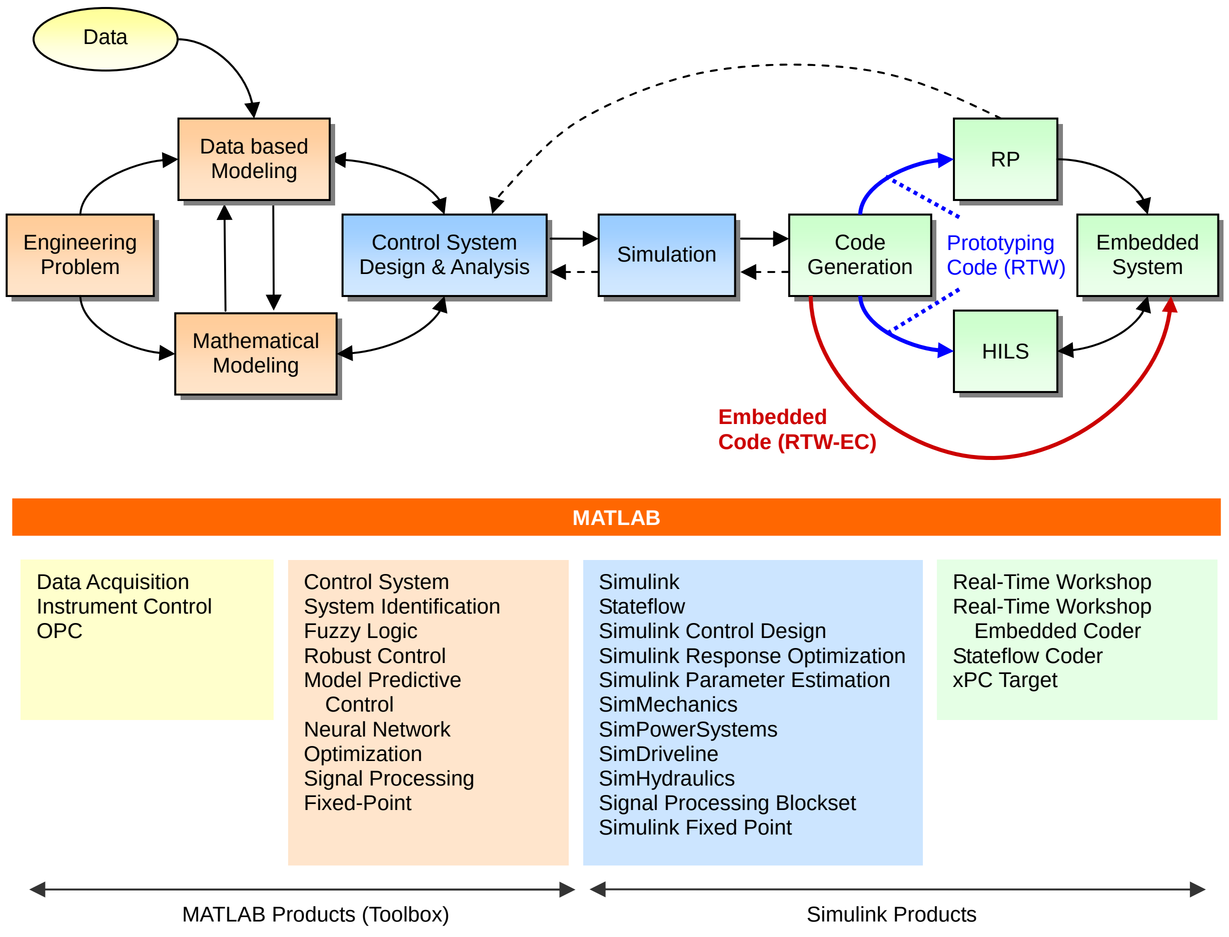


Figure 1-1 MBD for control systems based on MATLAB product family

1.2 V-Process

The V-process shown in Figure 1-2 describes the MBD development process for control systems. The V-process consists of the Design, Coding, and Test stage. Each test stages correspond to the appropriate Design stages. A developer makes plant / controller models in the left side of the V-process for early improvement of controller algorithm, and reuses them in the right side for improvement of code verification and validation.

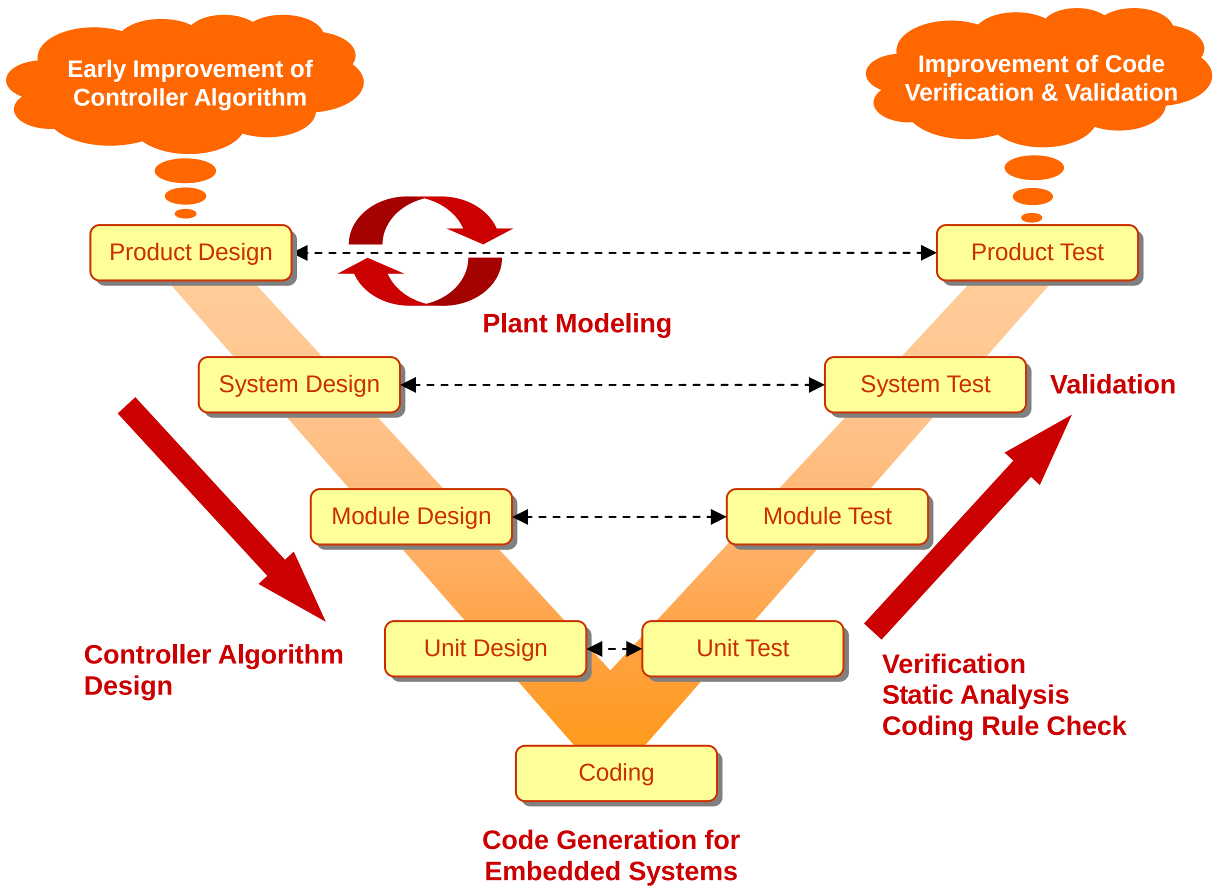


Figure 1-2 V-process for control systems

1.3 Merits of MBD

MBD has the following merits.

- Detection about specification errors in early stages of development
- Hardware prototype reduction and fail-safe verification with real-time simulation
- Efficient test by model verification
- Effective communication by model usage
- Coding time and error reduction by auto code generation

2 NXTway-GS System

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

2.1 Structure

Figure 2-1 shows the structure of NXTway-GS. A Hitechnic gyro sensor is used to calculate body pitch angle.

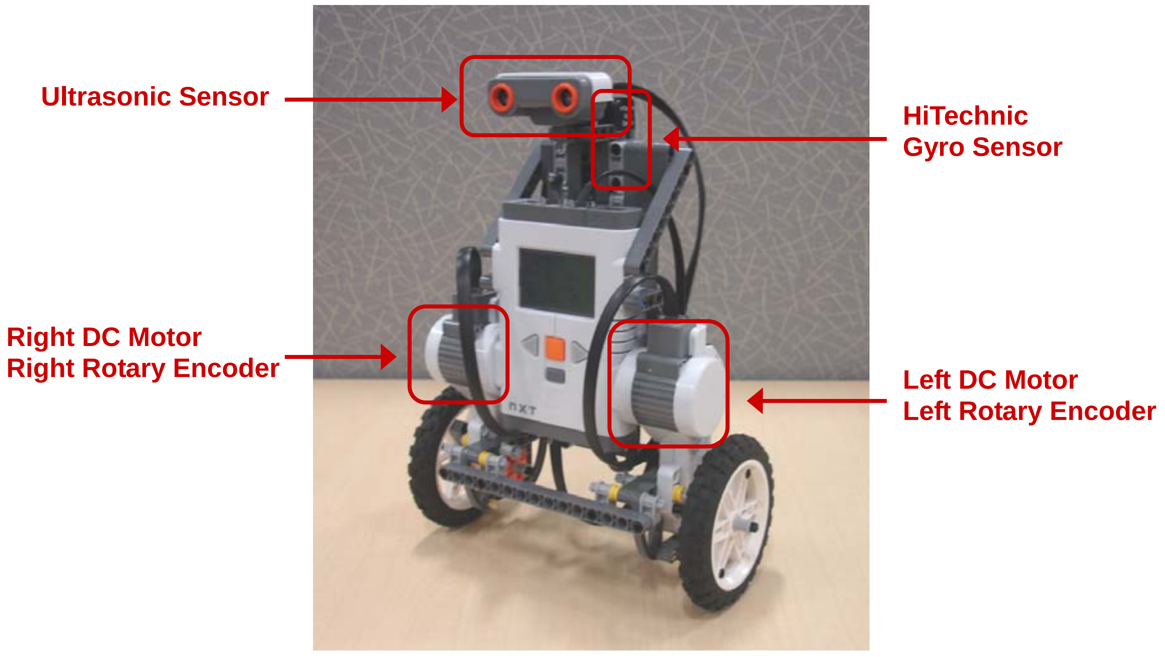

Ultrasonic Sensor
HiTechnic Gyro Sensor
Right DC Motor Right Rotary Encoder
Left DC Motor Left Rotary Encoder

Figure 2-1 NXTway-GS

2.2 Sensors and Actuators

Table 2-1 and Table 2-2 show sensor and actuator properties.

Table 2-1 Sensor Properties

Sensor	Output	Unit	Data Type	Maximum Sample [1/sec]
Rotary Encoder	angle	deg	int32	1000
Ultrasonic Sensor	distance	cm	int32	50 (N1)
Gyro Sensor	angular velocity	deg/sec	uint16	300

Table 2-2 Actuator Properties

Actuator	Input	Unit	Data Type	Maximum Sample [1/sec]
DC Motor	PWM	%	int8	500

(N1) : The heuristic maximum sample rate for measuring accurate distance.

The reference [1] illustrates many properties about DC motor. Generally speaking, sensors and actuators are different individually. Especially, you should note that gyro offset and gyro drift have big impact on balance control. Gyro offset is an output when a gyro sensor does not rotate, and gyro drift is time variation of gyro offset.

3 NXTway-GS Modeling

This chapter describes mathematical model and motion equations of NXTway-GS.

3.1 Two-Wheeled Inverted Pendulum Model

NXTway-GS can be considered as a two wheeled inverted pendulum model shown in Figure 3-1.

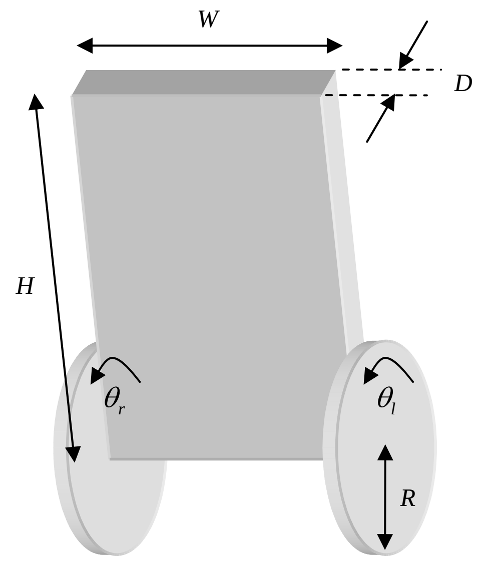
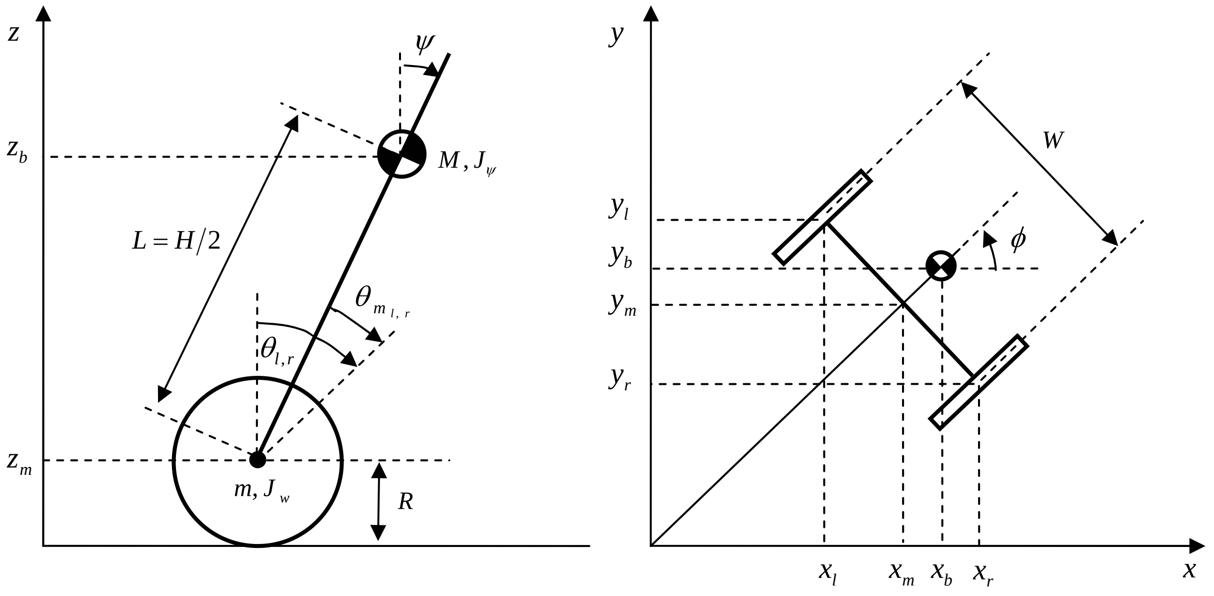


Figure 3-1 Two-wheeled inverted pendulum

Figure 3-2 shows side view and plane view of the two wheeled inverted pendulum. The coordinate system used in 3.2 Motion Equations of Two-Wheeled Inverted Pendulum is described in Figure 3-2.



ψ : body pitch angle $\theta_{l,r}$: wheel angle (l, r indicates left and right) $\theta_{m_{l,r}}$: DC motor angle

Figure 3-2 Side view and plane view of two-wheeled inverted pendulum

Physical parameters of NXTway-GS are the following.

$g = 9.81$	$[m/\sec^2]$	:	Gravity acceleration
$m = 0.03$	$[kg]$	:	Wheel weight
$R = 0.04$	$[m]$	:	Wheel radius
$J_W = mR^2/2$	$[kgm^2]$	:	Wheel inertia moment
$M = 0.6$	$[kg]$	:	Body weight
$W = 0.14$	$[m]$	:	Body width
$D = 0.04$	$[m]$	:	Body depth
$H = 0.144$	$[m]$	:	Body height
$L = H/2$	$[m]$	:	Distance of the center of mass from the wheel axle
$J_\psi = ML^2/3$	$[kgm^2]$	:	Body pitch inertia moment
$J_\phi = M(W^2 + D^2)/12$	$[kgm^2]$	:	Body yaw inertia moment
$J_m = 1 \times 10^{-5}$	$[kgm^2]$	:	DC motor inertia moment
$R_m = 6.69$	$[\Omega]$	:	DC motor resistance
$K_b = 0.468$	$[V \sec/rad]$	:	DC motor back EMF constant
$K_t = 0.317$	$[Nm/A]$	:	DC motor torque constant
$n = 1$		:	Gear ratio
$f_m = 0.0022$		:	Friction coefficient between body and DC motor
$f_W = 0$		:	Friction coefficient between wheel and floor.

- We use the values described in reference [2] for R_m, K_b, K_t.
- We use the values that seems to be appropriate for J_m, n, f_m, f_W, because it is difficult to measure.

3.2 Motion Equations of Two-Wheeled Inverted Pendulum

We can derive motion equations of two-wheeled inverted pendulum by the Lagrangian method based on the coordinate system in Figure 3-2. If the direction of two-wheeled inverted pendulum is x-axis positive direction at $t = 0$, each coordinates are given as the following.

$$(\theta, \quad \phi) = \left(\frac{1}{2}(\theta_l + \theta_r) \quad \frac{R}{W}(\theta_r - \theta_l) \right) \tag{3.1}$$

$$(x_m, \quad y_m, \quad z_m) = \left(\int \dot{x}_m dt, \quad \int \dot{y}_m dt, \quad R \right), \quad (\dot{x}_m, \quad \dot{y}_m) = \left(R\dot{\theta}\cos\phi \quad R\dot{\theta}\sin\phi \right) \tag{3.2}$$

$$(x_l, \quad y_l, \quad z_l) = \left(x_m - \frac{W}{2}\sin\phi, \quad y_m + \frac{W}{2}\cos\phi, \quad z_m \right) \tag{3.3}$$

$$(x_r, \quad y_r, \quad z_r) = \left(x_m + \frac{W}{2}\sin\phi, \quad y_m - \frac{W}{2}\cos\phi, \quad z_m \right) \tag{3.4}$$

$$(x_b, \quad y_b, \quad z_b) = \left(x_m + L\sin\psi\cos\phi, \quad y_m + L\sin\psi\sin\phi, \quad z_m + L\cos\psi \right) \tag{3.5}$$

The translational kinetic energy T_1, the rotational kinetic energy T_2, the potential energy U are

$$T_1 = \frac{1}{2}m\left(\dot{x}_l^{\,2} + \dot{y}_l^{\,2} + \dot{z}_l^{\,2}\right) + \frac{1}{2}m\left(\dot{x}_r^{\,2} + \dot{y}_r^{\,2} + \dot{z}_r^{\,2}\right) + \frac{1}{2}M\left(\dot{x}_b^{\,2} + \dot{y}_b^{\,2} + \dot{z}_b^{\,2}\right) \tag{3.6}$$

$$T_2 = \frac{1}{2}J_w\dot{\theta}_l^{\,2} + \frac{1}{2}J_w\dot{\theta}_r^{\,2} + \frac{1}{2}J_\psi\dot{\psi}^2 + \frac{1}{2}J_\phi\dot{\phi}^2 + \frac{1}{2}n^2J_m(\dot{\theta}_l - \dot{\psi})^2 + \frac{1}{2}n^2J_m(\dot{\theta}_r - \dot{\psi})^2 \tag{3.7}$$

$$U = mgz_l + mgz_r + Mgz_b \tag{3.8}$$

The fifth and sixth term in T_2 are rotation kinetic energy of an armature in left and right DC motor. The Lagrangian L has the following expression.

$$L = T_1 + T_2 - U \tag{3.9}$$

We use the following variables as the generalized coordinates.

θ	:	Average angle of left and right wheel
ψ	:	Body pitch angle
ϕ	:	Body yaw angle

Lagrange equations are the following

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = F_\theta \tag{3.10}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) - \frac{\partial L}{\partial \psi} = F_\psi \tag{3.11}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) - \frac{\partial L}{\partial \phi} = F_\phi \tag{3.12}$$

We derive the following equations by evaluating Eqs. (3.10) - (3.12).

$$\left[(2m+M)R^2 + 2J_w + 2n^2J_m\right]\ddot{\theta} + \left(MLR\cos\psi - 2n^2J_m\right)\ddot{\psi} - MLR\dot{\psi}^2\sin\psi = F_\theta \tag{3.13}$$

$$\left(MLR\cos\psi - 2n^2J_m\right)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2J_m\right)\ddot{\psi} - MgL\sin\psi - ML^2\dot{\phi}^2\sin\psi\cos\psi = F_\psi \tag{3.14}$$

$$\left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2J_m\right) + ML^2\sin^2\psi\right]\ddot{\phi} + 2ML^2\dot{\psi}\dot{\phi}\sin\psi\cos\psi = F_\phi \tag{3.15}$$

In consideration of DC motor torque and viscous friction, the generalized forces are given as the following

$$\left(F_\theta, \quad F_\psi, \quad F_\phi\right) = \left(F_l + F_r, \quad F_\psi, \quad \frac{W}{2R}\left(F_r - F_l\right)\right) \tag{3.16}$$

$$F_l = nK_t i_l + f_m(\dot{\psi} - \dot{\theta}_l) - f_w\dot{\theta}_l \tag{3.17}$$

$$F_r = nK_t i_r + f_m(\dot{\psi} - \dot{\theta}_r) - f_w\dot{\theta}_r \tag{3.18}$$

$$F_\psi = -nK_t i_l - nK_t i_r - f_m(\dot{\psi} - \dot{\theta}_l) - f_m(\dot{\psi} - \dot{\theta}_r) \tag{3.19}$$

where $i_{l,r}$ is the DC motor current.

We cannot use the DC motor current directly in order to control it because it is based on PWM (voltage) control. Therefore, we evaluate the relation between current $i_{l,r}$ and voltage $v_{l,r}$ using DC motor equation. If the friction inside the motor is negligible, the DC motor equation is generally as follows

$$L_m \dot{i}_{l,r} = v_{l,r} + K_b(\dot{\psi} - \dot{\theta}_{l,r}) - R_m i_{l,r} \tag{3.20}$$

Here we consider that the motor inductance is negligible and is approximated as zero. Therefore the current is

$$i_{l,r} = \frac{v_{l,r} + K_b(\dot{\psi} - \dot{\theta}_{l,r})}{R_m} \tag{3.21}$$

From Eq.(3.21), the generalized force can be expressed using the motor voltage.

$$F_\theta = \alpha(v_l + v_r) - 2(\beta + f_w)\dot{\theta} + 2\beta\dot{\psi} \tag{3.22}$$

$$F_\psi = -\alpha(v_l + v_r) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \tag{3.23}$$

$$F_\phi = \frac{W}{2R}\alpha(v_r - v_l) - \frac{W^2}{2R^2}(\beta + f_w)\dot{\phi} \tag{3.24}$$

$$\alpha = \frac{nK_t}{R_m}, \quad \beta = \frac{nK_t K_b}{R_m} + f_m \tag{3.25}$$

3.3 State Equations of Two-Wheeled Inverted Pendulum

We can derive state equations based on modern control theory by linearizing motion equations at a balance point of NXTway-GS. It means that we consider the limit $\psi \to 0$ ($\sin\psi \to \psi$, $\cos\psi \to 1$) and neglect the second order term like $\dot{\psi}^2$. The motion equations (3.13) – (3.15) are approximated as the following

$$\left[(2m+M)R^2 + 2J_w + 2n^2 J_m\right]\ddot{\theta} + \left(MLR - 2n^2 J_m\right)\ddot{\psi} = F_\theta \tag{3.26}$$

$$\left(MLR - 2n^2 J_m\right)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2 J_m\right)\ddot{\psi} - MgL\psi = F_\psi \tag{3.27}$$

$$\left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2 J_m\right)\right]\ddot{\phi} = F_\phi \tag{3.28}$$

Eq. (3.26) and Eq. (3.27) has θ and ψ, Eq. (3.28) has ϕ only. These equations can be expressed in the form

$$E\begin{bmatrix}\ddot{\theta}\\ \ddot{\psi}\end{bmatrix} + F\begin{bmatrix}\dot{\theta}\\ \dot{\psi}\end{bmatrix} + G\begin{bmatrix}\theta\\ \psi\end{bmatrix} = H\begin{bmatrix}v_l\\ v_r\end{bmatrix} \tag{3.29}$$

$$E = \begin{bmatrix}(2m+M)R^2 + 2J_w + 2n^2 J_m & MLR - 2n^2 J_m\\ MLR - 2n^2 J_m & ML^2 + J_\psi + 2n^2 J_m\end{bmatrix}$$

$$F = 2\begin{bmatrix}\beta + f_w & -\beta\\ -\beta & \beta\end{bmatrix}$$

$$G = \begin{bmatrix}0 & 0\\ 0 & -MgL\end{bmatrix}$$

$$H = \begin{bmatrix}\alpha & \alpha\\ -\alpha & -\alpha\end{bmatrix}$$

$$I\ddot{\phi} + J\dot{\phi} = K(v_r - v_l) \tag{3.30}$$

$$I = \frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2 J_m\right)$$

$$J = \frac{W^2}{2R^2}\left(\beta + f_w\right)$$

$$K = \frac{W}{2R}\alpha$$

Here we consider the following variables $\mathbf{x_1}, \mathbf{x_2}$ as state, and $\mathbf{u}$ as input. $\mathbf{x}^T$ indicates transpose of $\mathbf{x}$.

$$\mathbf{x_1} = [\theta, \quad \psi, \quad \dot{\theta}, \quad \dot{\psi}]^T, \quad \mathbf{x_2} = [\phi, \quad \dot{\phi}]^T, \quad \mathbf{u} = [v_l, \quad v_r]^T \tag{3.31}$$

Consequently, we can derive state equations of two-wheeled inverted pendulum from Eq. (3.29) and Eq. (3.30).

$$\dot{\mathbf{x}}_\mathbf{1} = A_1 \mathbf{x_1} + B_1 \mathbf{u} \tag{3.32}$$

$$\dot{\mathbf{x}}_\mathbf{2} = A_2 \mathbf{x_2} + B_2 \mathbf{u} \tag{3.33}$$

$$A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_1(3,2) & A_1(3,3) & A_1(3,4) \\ 0 & A_1(4,2) & A_1(4,3) & A_1(4,4) \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ B_1(3) & B_1(3) \\ B_1(4) & B_1(4) \end{bmatrix} \tag{3.34}$$

$$A_2 = \begin{bmatrix} 0 & 1 \\ 0 & -J/I \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 & 0 \\ -K/I & K/I \end{bmatrix} \tag{3.35}$$

$$A_1(3,2) = -gMLE(1,2)/\det(E)$$
$$A_1(4,2) = gMLE(1,1)/\det(E)$$
$$A_1(3,3) = -2[(\beta + f_w)E(2,2) + \beta E(1,2)]/\det(E)$$
$$A_1(4,3) = 2[(\beta + f_w)E(1,2) + \beta E(1,1)]/\det(E)$$
$$A_1(3,4) = 2\beta[E(2,2) + E(1,2)]/\det(E)$$
$$A_1(4,4) = -2\beta[E(1,1) + E(1,2)]/\det(E)$$
$$B_1(3) = \alpha[E(2,2) + E(1,2)]/\det(E)$$
$$B_1(4) = -\alpha[E(1,1) + E(1,2)]/\det(E)$$
$$\det(E) = E(1,1)E(2,2) - E(1,2)^2$$

4 NXTway-GS Controller Design

This chapter describes the controller design of NXTway-GS (two-wheeled inverted pendulum) based on modern control theory.

4.1 Control System

The characteristics as control system are as follows.

Inputs and Outputs

The input to an actuator is PWM duty of the left and right DC motor even though input $\mathbf{u}$ in Eq. (3.31) is voltage. The outputs from sensors are the DC motor angle $\theta_{m_{l,r}}$ and the body pitch angular velocity $\dot{\psi}$.

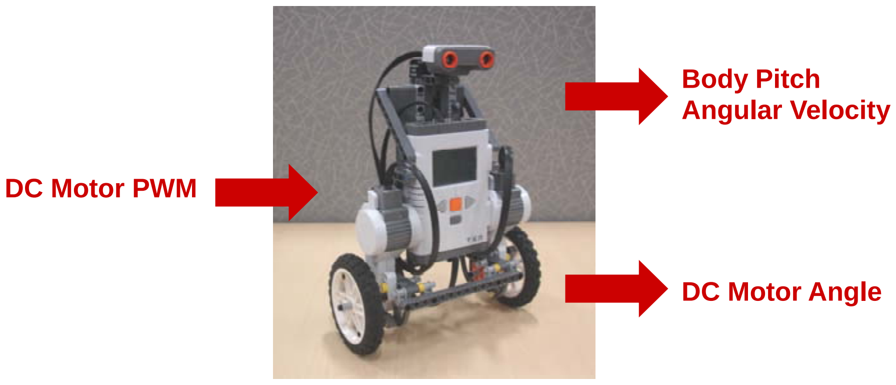


Figure 4-1 Inputs and Outputs

It is easy to evaluate θ, ϕ by using $\theta_{m_{l,r}}$. There are two methods to evaluate ψ by using $\dot{\psi}$.

1. Derive ψ by integrating the angular speed numerically.
2. Estimate ψ by using an observer based on modern control theory.

We use method 1 in the following controller design.

Stability

It is easy to understand NXTway-GS balancing position is not stable. We have to move NXTway-GS in the same direction of body pitch angle to keep balancing. Modern control theory gives many techniques to stabilize an unstable system. (See Appendix A)

4.2 Controller Design

Eq. (3.29) is a similar equation as mass-spring-damper system. Figure 4-2 shows an equivalent system of two-wheeled inverted pendulum interpreted as mass-spring-damper system.

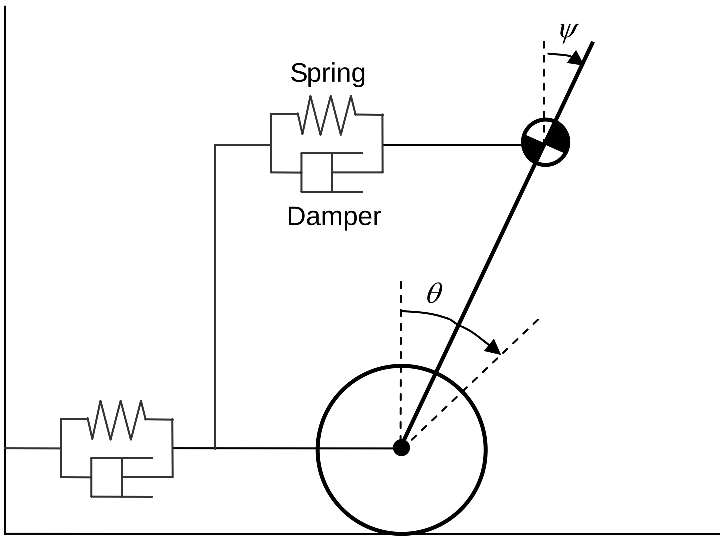
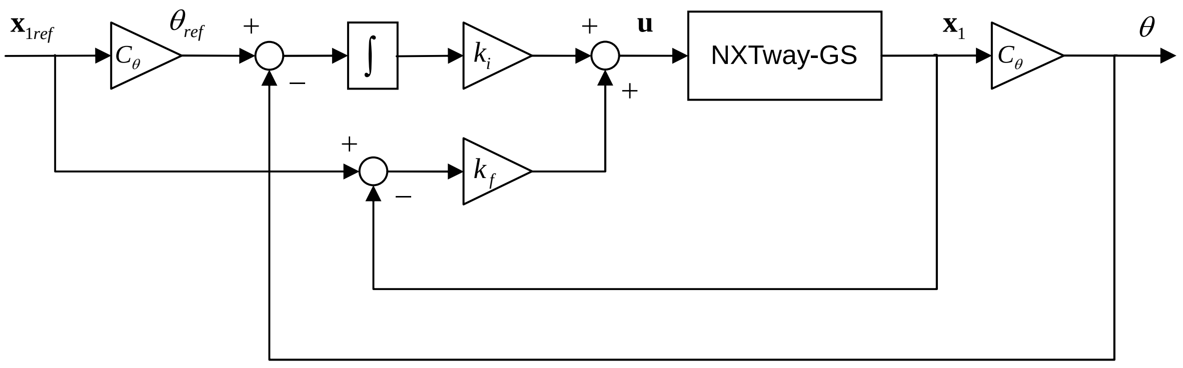


Figure 4-2 Equivalent mass-spring-damper system

We find out that we can make the two-wheeled inverted pendulum stable by adjusting spring constants and damper friction constants in Figure 4-2. It is possible to think that the control techniques described in Appendix A provides theoretical method to calculate those constants.

We use a servo controller described in Appendix A.3 as NXTway-GS controller and select θ as a reference of servo control. It is important that we cannot use variables other than θ as the reference because the system goes to be uncontrollable. Figure 4-3 shows the block diagram of NXTway-GS servo controller.



C_θ *is an output matrix to derive* θ *from* $\mathbf{x}_1$

Figure 4-3 NXTway-GS servo controller block diagram

We calculate feedback gain and integral gain by linear quadratic regulator method. We choose the following weight matrix Q and R by experimental trial and error.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 6\times10^5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4\times10^2 \end{bmatrix}, R = \begin{bmatrix} 1\times10^3 & 0 \\ 0 & 1\times10^3 \end{bmatrix}$$

$Q(2,2)$ is a weight for body pitch angle, and $Q(5,5)$ is a weight for time integration of difference between measured average angle and referenced one.

param_controller.m calculates linear quadratic regulator and defines other parameters. The gain calculation part of it is extracted as the following.

param_controller.m

```
% Controller Parameters

% Servo Gain Calculation using Optimal Regulator
A_BAR = [A1, zeros(4, 1); C1(1, :), 0];
B_BAR = [B1; 0, 0];
QQ = [
      1, 0,   0, 0, 0
      0, 6e5, 0, 0, 0
      0, 0,   1, 0, 0
      0, 0,   0, 1, 0
      0, 0,   0, 0, 4e2
      ];
RR = 1e3 * eye(3);
KK = lqr(A_BAR, B_BAR, QQ, RR);
k_f = KK(1, 1:4);          % feedback gain
k_i = KK(1, 5);            % integral gain
% suppress velocity gain because it fluctuates NXTway-GS
k_f(3) = k_f(3) * 0.85;
```

The result is

$$k_f = [-0.8351,\ -34.1896,\ -1.0995,\ -2.8141]$$
$$k_i = -0.4472$$

We adjust the value of speed gain $k_f(3)$ after linear quadratic regulator calculation because it fluctuates NXTway-GS excessively.

Furthermore, we add the following control

- Rotate NXTway-GS by giving different value to the left and right motor.
- Use P control for wheel synchronization when NXTway-GS runs straightforward because the rotation angle of DC motors is not same even though same PWM is applied.

Consequently, we derive NXTway-GS controller shown in Figure 4-4. Here $k_{\dot{\theta}}$, $k_{\dot{\phi}}$, k_{sync} are tuning parameters.

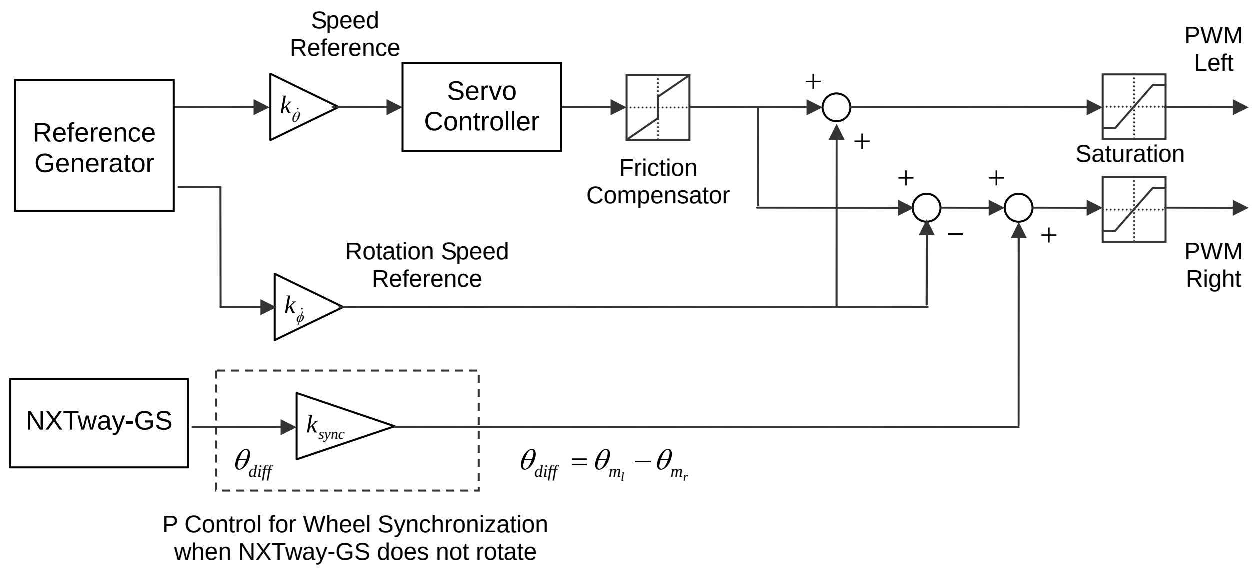


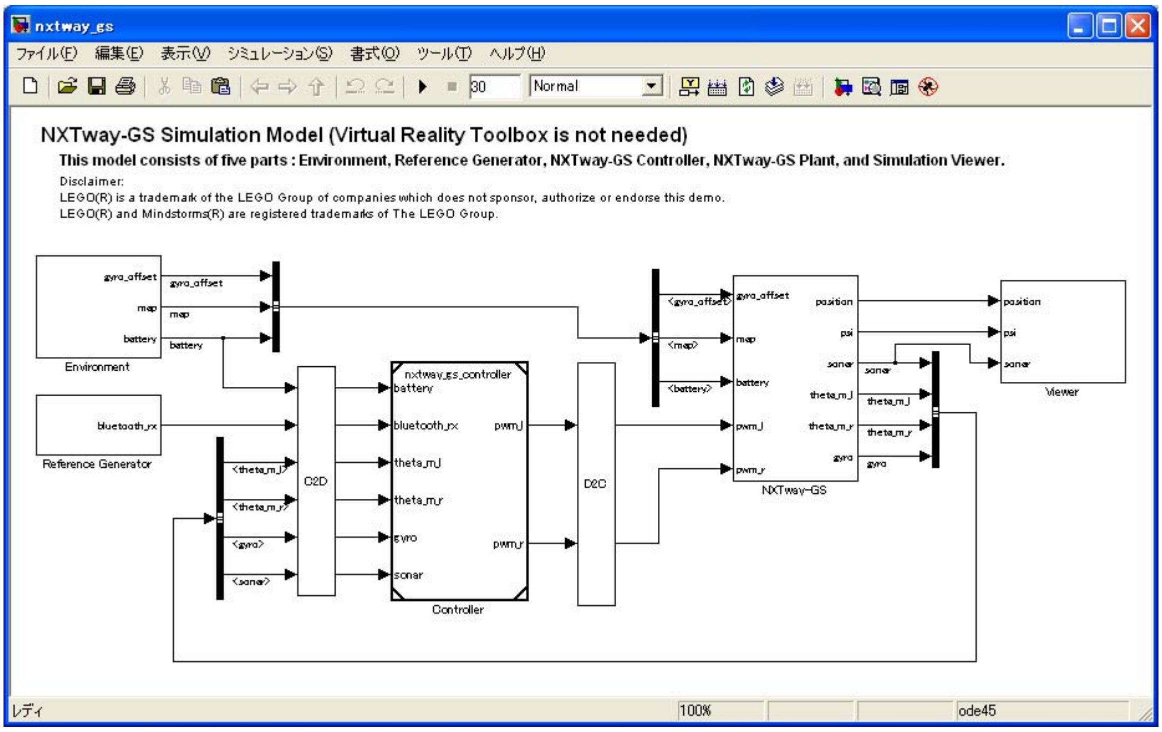
Figure 4-4 NXTway-GS controller block diagram

5 NXTway-GS Model

This chapter describes summary of NXTway-GS model and parameter files.

5.1 Model Summary

The nxtway_gs.mdl and nxtway_gs_vr.mdl are models of NXTway-GS control system. Both models are identical but different in point of including 3D viewer provided by Virtual Reality Toolbox.



NXTway-GS Simulation Model (Virtual Reality Toolbox is not needed)

This model consists of five parts : Environment, Reference Generator, NXTway-GS Controller, NXTway-GS Plant, and Simulation Viewer.

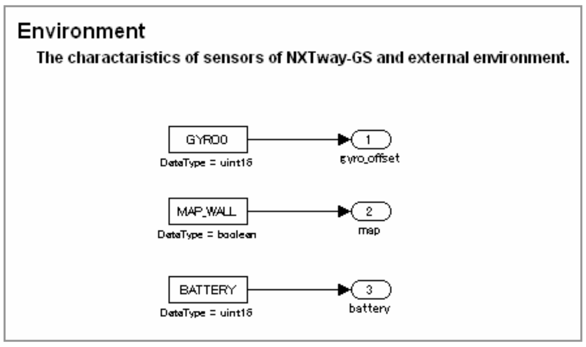
Disclaimer:
LEGO(R) is a trademark of the LEGO Group of companies which does not sponsor, authorize or endorse this demo.
LEGO(R) and Mindstorms(R) are registered trademarks of The LEGO Group.

Figure 5-1 nxtway_gs.mdl

Main parts of nxtway_gs.mdl and nxtway_gs_vr.mdl are as follows.

Environment

This subsystem defines environmental parameters. For example, map data, gyro offset value, and so on.



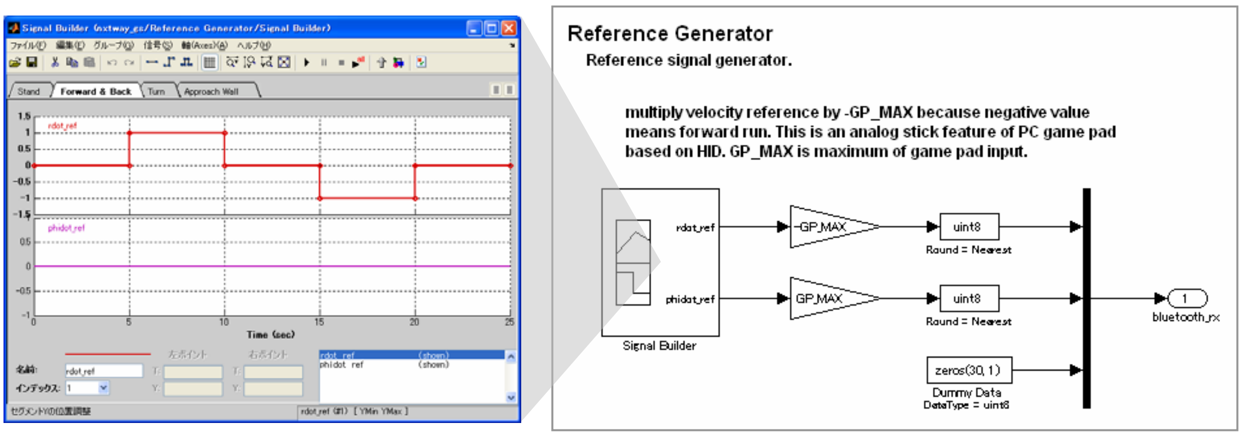
Environment

The charactaristics of sensors of NXTway-GS and external environment.

Figure 5-2 Environment subsystem

Reference Generator

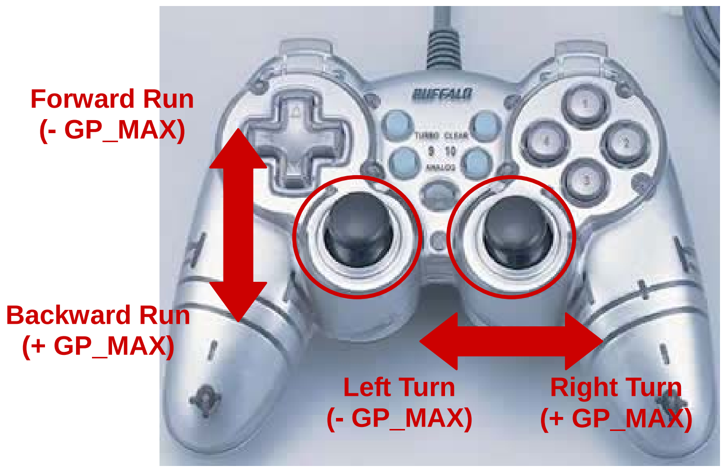
This subsystem is a reference signal generator for NXTway-GS. We can change the speed reference and the rotation speed reference by using Signal Builder block. The output signal is a 32 byte data which has dummy data to satisfy the NXT GamePad utility specification.



multiply velocity reference by -GP_MAX because negative value means forward run. This is an analog stick feature of PC game pad based on HID. GP_MAX is maximum of game pad input.

Figure 5-3 Reference Generator subsystem

Figure 5-4 shows a relation between speed and rotation speed reference value and PC game pad analog sticks.

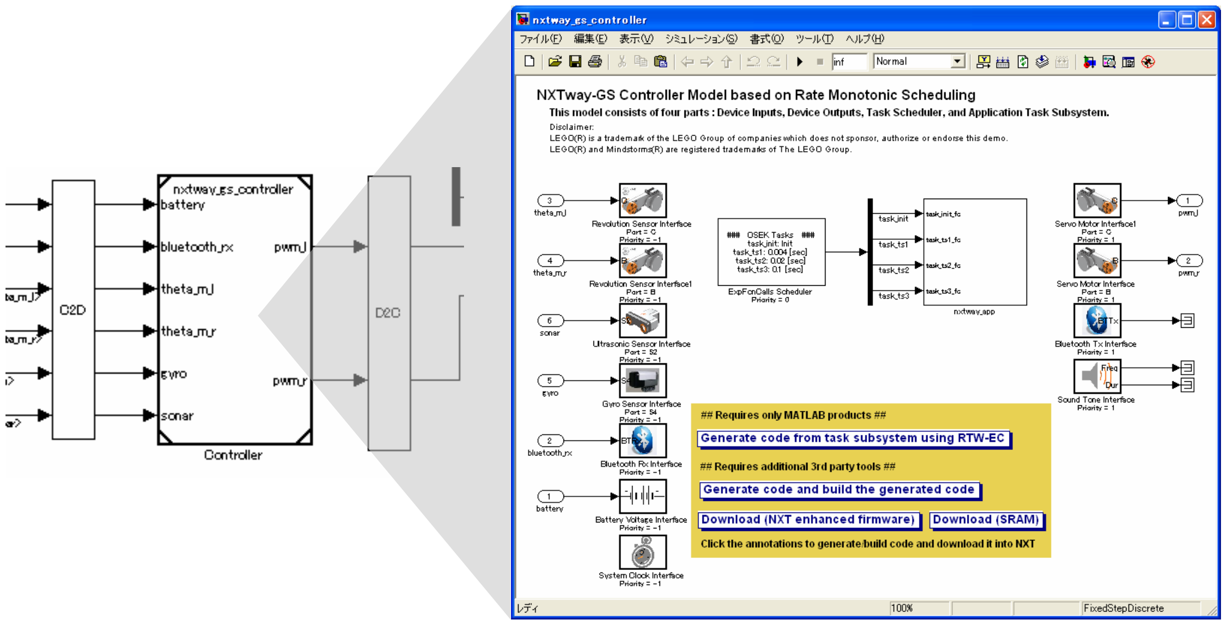


GP_MAX is a maximum input from PC game pad.
(We set GP_MAX = 100)

Figure 5-4 Inputs from PC game pad analog sticks

Controller

This block is NXTway-GS digital controller and references nxtway_gs_controller.mdl with Model block. Refer 7 Controller Model (Single Precision Floating-Point Arithmetic) for more details.



NXTway-GS Controller Model based on Rate Monotonic Scheduling

This model consists of four parts : Device Inputs, Device Outputs, Task Scheduler, and Application Task Subsystem.

Disclaimer:
LEGO(R) is a trademark of the LEGO Group of companies which does not sponsor, authorize or endorse this demo.
LEGO(R) and Mindstorms(R) are registered trademarks of The LEGO Group.

Requires only MATLAB products

Generate code from task subsystem using RTW-EC

Requires additional 3rd party tools

Generate code and build the generated code

Download (NXT enhanced firmware) | Download (SRAM)

Click the annotations to generate/build code and download it into NXT

nxtway_gs_controller
ファイル(F) 編集(E) 表示(V) シミュレーション(S) 書式(O) ツール(T) ヘルプ(H)
inf
Normal
C2D
battery
bluetooth_rx
theta_m_l
theta_m_r
gyro
sonar
pwm_l
pwm_r
Controller
D2C
Revolution Sensor Interface Port = C Priority = -1
Revolution Sensor Interface1 Port = B Priority = -1
Ultrasonic Sensor Interface Port = S2 Priority = -1
Gyro Sensor Interface Port = S4 Priority = -1
Bluetooth Rx Interface Priority = -1
Battery Voltage Interface Priority = -1
System Clock Interface Priority = -1
### OSEK Tasks ### task_init: Init task_ts1: 0.004 [sec] task_ts2: 0.02 [sec] task_ts3: 0.1 [sec]
ExpFcnCalls Scheduler Priority = 0
task_init
task_ts1
task_ts2
task_ts3
task_init_fc
task_ts1_fc
task_ts2_fc
task_ts3_fc
nxtway_app
Servo Motor Interface1 Port = C Priority = 1
Servo Motor Interface Port = B Priority = 1
Bluetooth Tx Interface Priority = 1
Sound Tone Interface Priority = 1
Freq
Dur
レディ
100%
FixedStepDiscrete

Figure 5-5 Controller block (nxtway_gs_controller.mdl)

The Controller block runs in discrete-time (base sample time = 1 [ms]) and the plant (NXTway-GS subsystem) runs in continuous-time (sample time = 0 [s]). Therefore it is necessary to convert continuous-time to discrete-time and vice versa by inserting Rate Transition blocks between them properly.

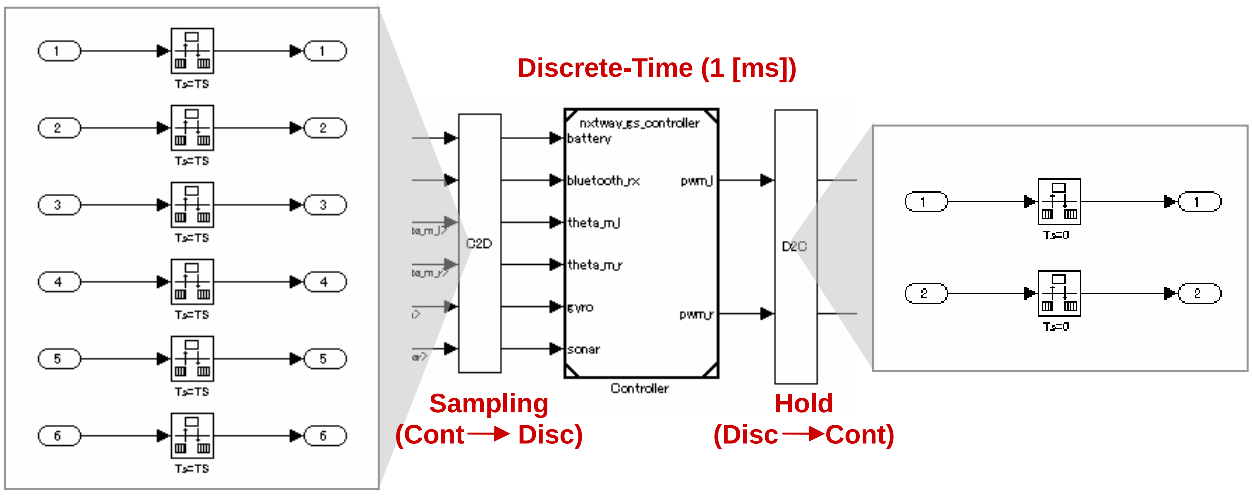


Figure 5-6 Rate conversions between the controller and the plant

NXTway-GS

This subsystem is mathematical model of NXTway-GS. It consists of sensor, actuator, and linear plant model. The plant references nxtway_gs_plant.mdl with Model block. Refer 6 Plant Model for more details.

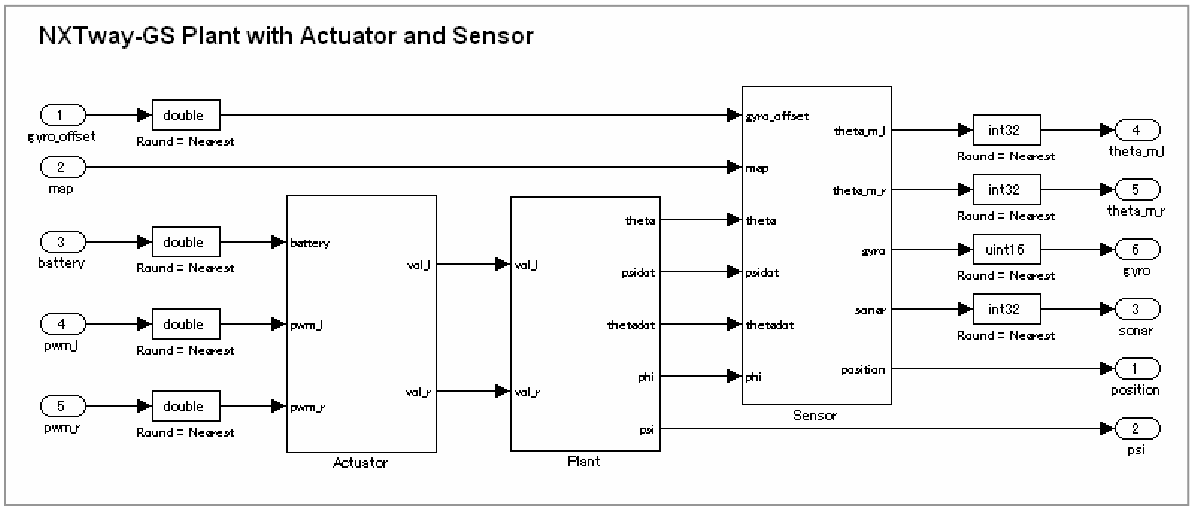


Figure 5-7 NXTway-GS subsystem

Viewer

This subsystem includes simulation viewers. nxtway_gs.mdl includes a position viewer with XY Graph block, and nxtway_gs_vr.mdl does 3D viewer provided by Virtual Reality Toolbox.

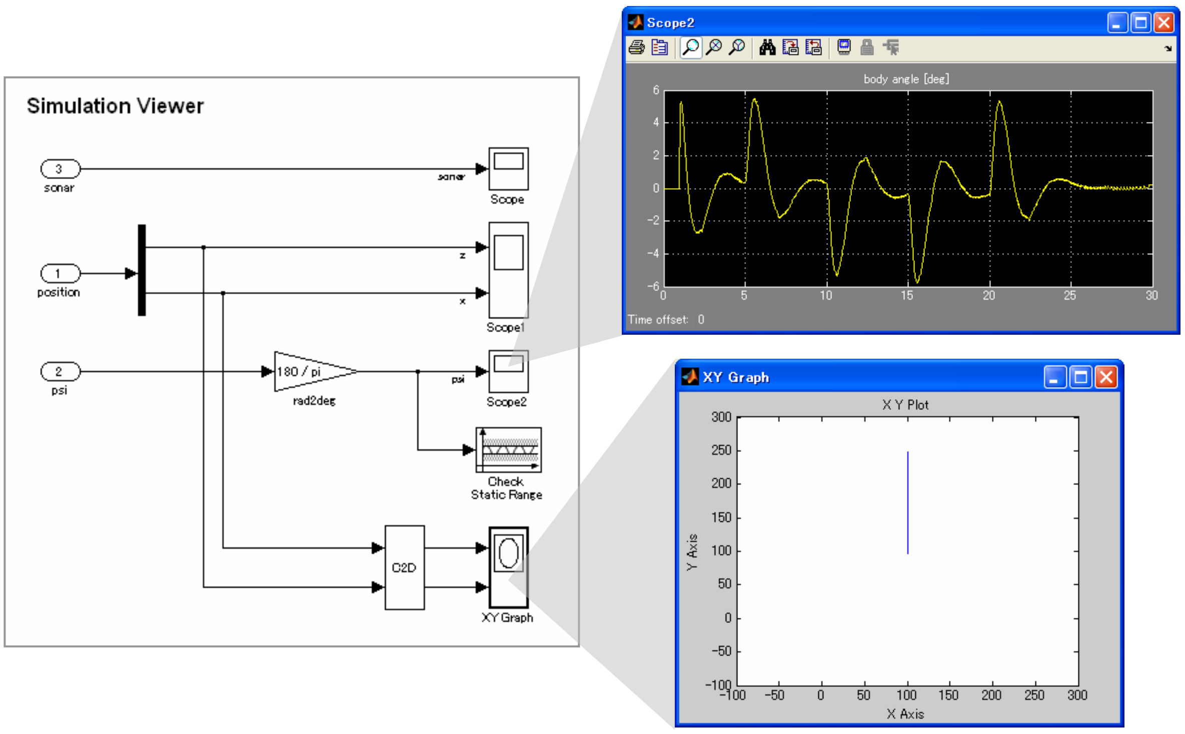


Figure 5-8 Viewer subsystem (nxtway_gs.mdl)

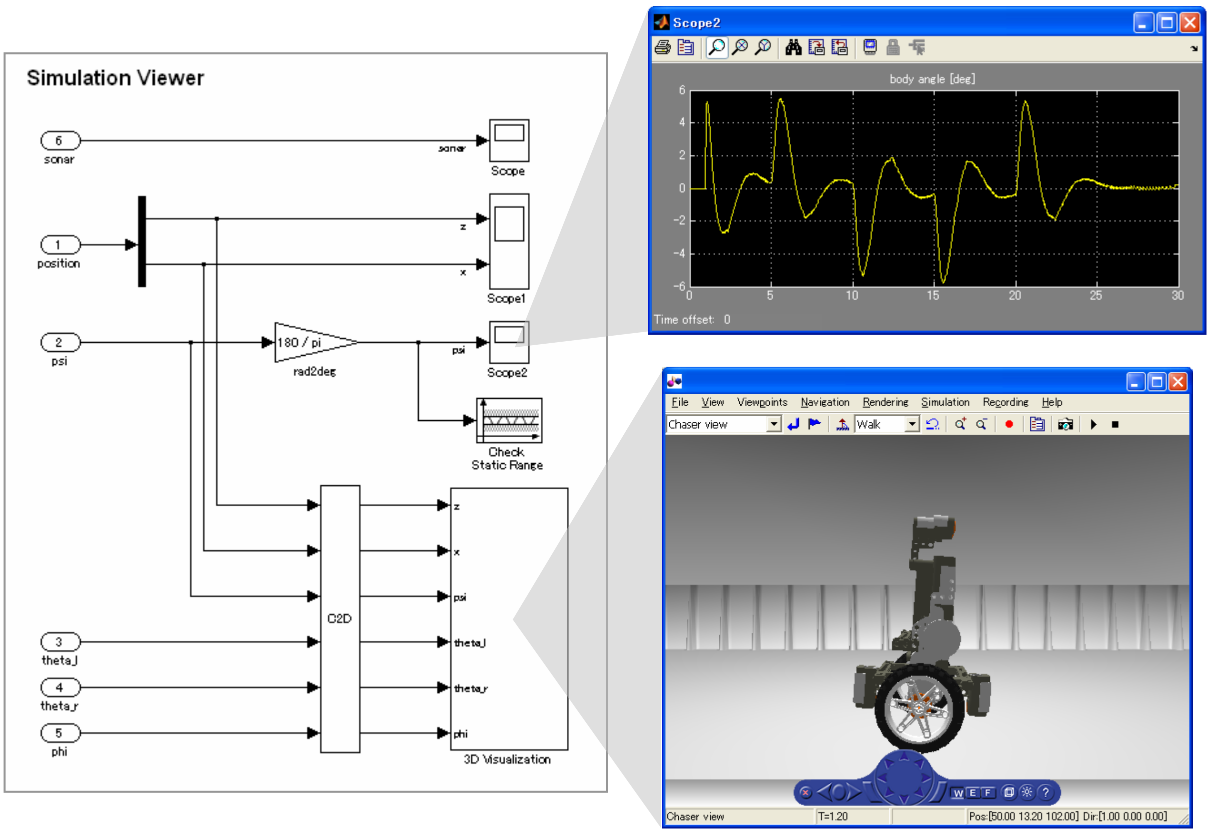


Figure 5-9 Viewer subsystem (nxtway_gs_vr.mdl)

5.2 Parameter Files

Table 5-1 shows parameter files for simulation and code generation.

Table 5-1 Parameter files

File Name	Description
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

param_nxtway_gs.m calls param_***.m (*** indicates controller, plant, sim) and creates all parameters in base workspace. Model callback function is used to run param_nxtway_gs.m automatically when the model is loaded.

To display model callback function, choose [**Model Properties**] from the Simulink [**File**] menu.

6 Plant Model

This chapter describes NXTway-GS subsystem in nxtway_gs.mdl / nxtway_gs_vr.mdl.

6.1 Model Summary

The NXTway-GS subsystem consists of sensors, actuators, and linear plant model. It converts the data type of input signals to double, calculates plant dynamics using double precision floating-point arithmetic, and outputs the results after quantization.

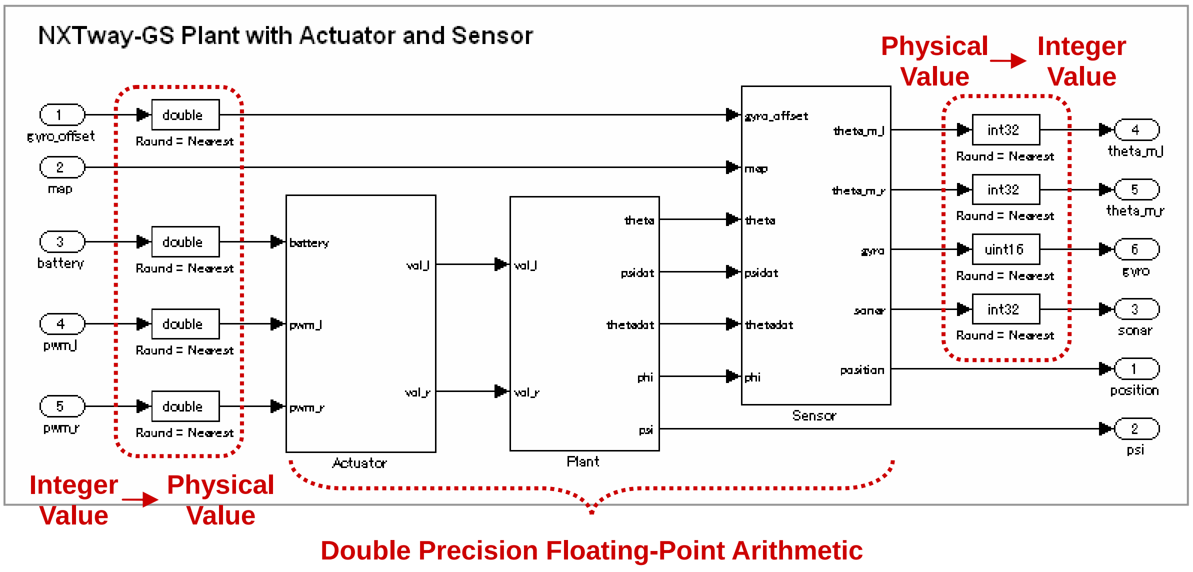

NXTway-GS Plant with Actuator and Sensor
Physical Value → Integer Value
Integer Value → Physical Value
Double Precision Floating-Point Arithmetic
1 gyro_offset
2 map
3 battery
4 pwm_l
5 pwm_r
double
Round = Nearest
Actuator
Plant
Sensor
int32
uint16
4 theta_m_l
5 theta_m_r
6 gyro
3 sonar
1 position
2 psi

Figure 6-1 NXTway-GS subsystem

6.2 Actuator

Actuator subsystem calculates the DC motor voltage by using PWM duty derived from the controller. Considering the coulomb and viscous friction in the driveline, we model it as a dead zone before calculating the voltage.

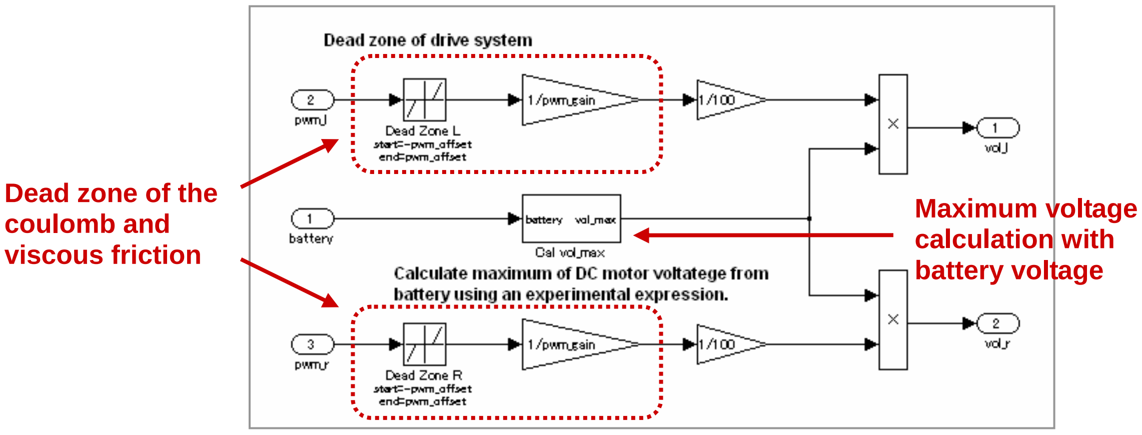

Dead zone of drive system
2 pwm_l
Dead Zone L start=-pwm_offset end=pwm_offset
1/pwm_gain
1/100
1 vol_l
Dead zone of the coulomb and viscous friction
1 battery
battery vol_max
Cal vol_max
Maximum voltage calculation with battery voltage
Calculate maximum of DC motor voltatege from battery using an experimental expression.
3 pwm_r
Dead Zone R start=-pwm_offset end=pwm_offset
2 vol_r

Figure 6-2 Actuator subsystem

DC motor maximum voltage is necessary to calculate PWM duty. In the Cal vol_max subsystem, we use the following experimental equation which is a conversion rule between the battery voltage V_B and the maximum DC motor voltage $V_{\max}$.

$$V_{\max} = 0.001089 \times V_B - 0.625 \tag{6.1}$$

We have derived Eq. (6.1) as the following. Generally speaking, DC motor voltage and rotation speed are proportional to battery voltage. Figure 6-3 shows an experimental result of battery voltage and motor rotation speed at PWM = 100% with no load.

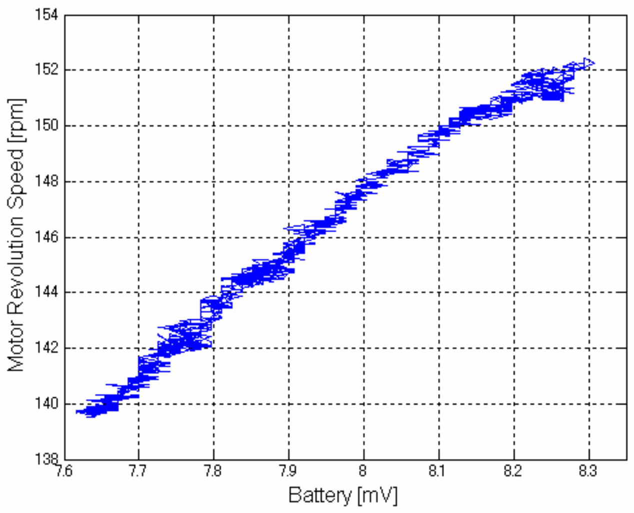
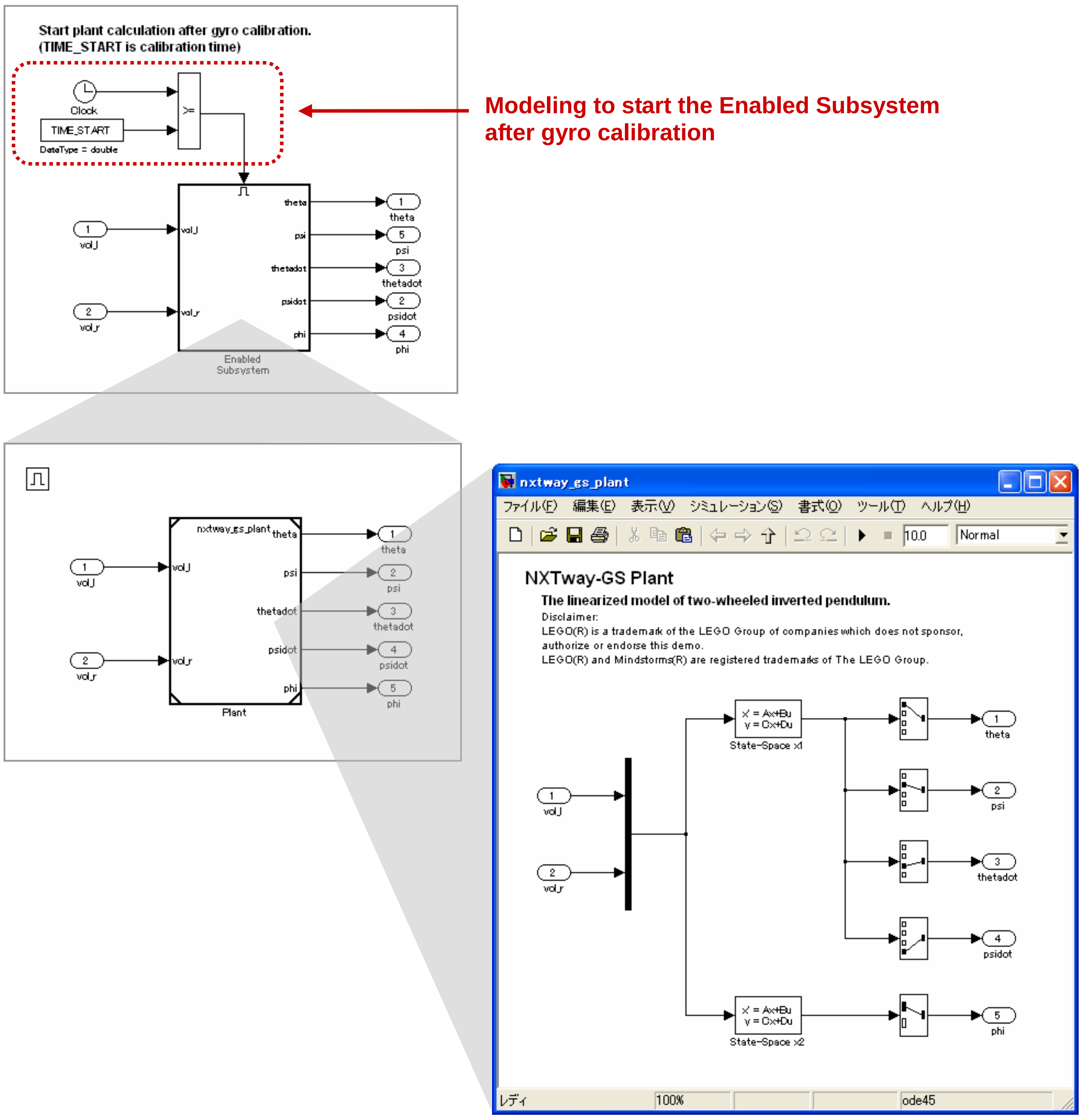


Figure 6-3 Experimental result of battery voltage and motor rotation speed at PWM = 100%

Eq. (6.1) is derived by relating Figure 6-3 and the data described in the reference [1].

6.3 Plant

nxtway_gs_plant.mdl is a model of the state equation derived in 3.3 State Equations of Two-Wheeled Inverted Pendulum. It is referenced in Enabled Subsystem with Model block. Considering the calibration of gyro offset, the plant is modeled so as to start calculation after gyro calibration. Refer 7.1 Control Program Summary for more details about the gyro calibration.



Start plant calculation after gyro calibration.
(TIME_START is calibration time)
Clock
TIME_START
DataType = double
>=
Modeling to start the Enabled Subsystem after gyro calibration
vol_l
vol_r
theta
psi
thetadot
psidot
phi
Enabled Subsystem
nxtway_gs_plant
Plant
nxtway_gs_plant
ファイル(F) 編集(E) 表示(V) シミュレーション(S) 書式(O) ツール(T) ヘルプ(H)
10.0
Normal
x' = Ax+Bu
y = Cx+Du
State-Space x1
State-Space x2
レディ
100%
ode45

NXTway-GS Plant

The linearized model of two-wheeled inverted pendulum.

Disclaimer:
LEGO(R) is a trademark of the LEGO Group of companies which does not sponsor, authorize or endorse this demo.
LEGO(R) and Mindstorms(R) are registered trademarks of The LEGO Group.

Figure 6-4 Plant subsystem

6.4 Sensor

Sensor subsystem converts the values calculated in plant subsystem to several sensor outputs. Because the computation cost of distance (an output of ultrasonic sensor) calculation and wall hit detection are heavy, we cut the computation steps by inserting Rate Transition blocks. Refer Appendix B.3 for more details about the distance calculation and wall hit detection.

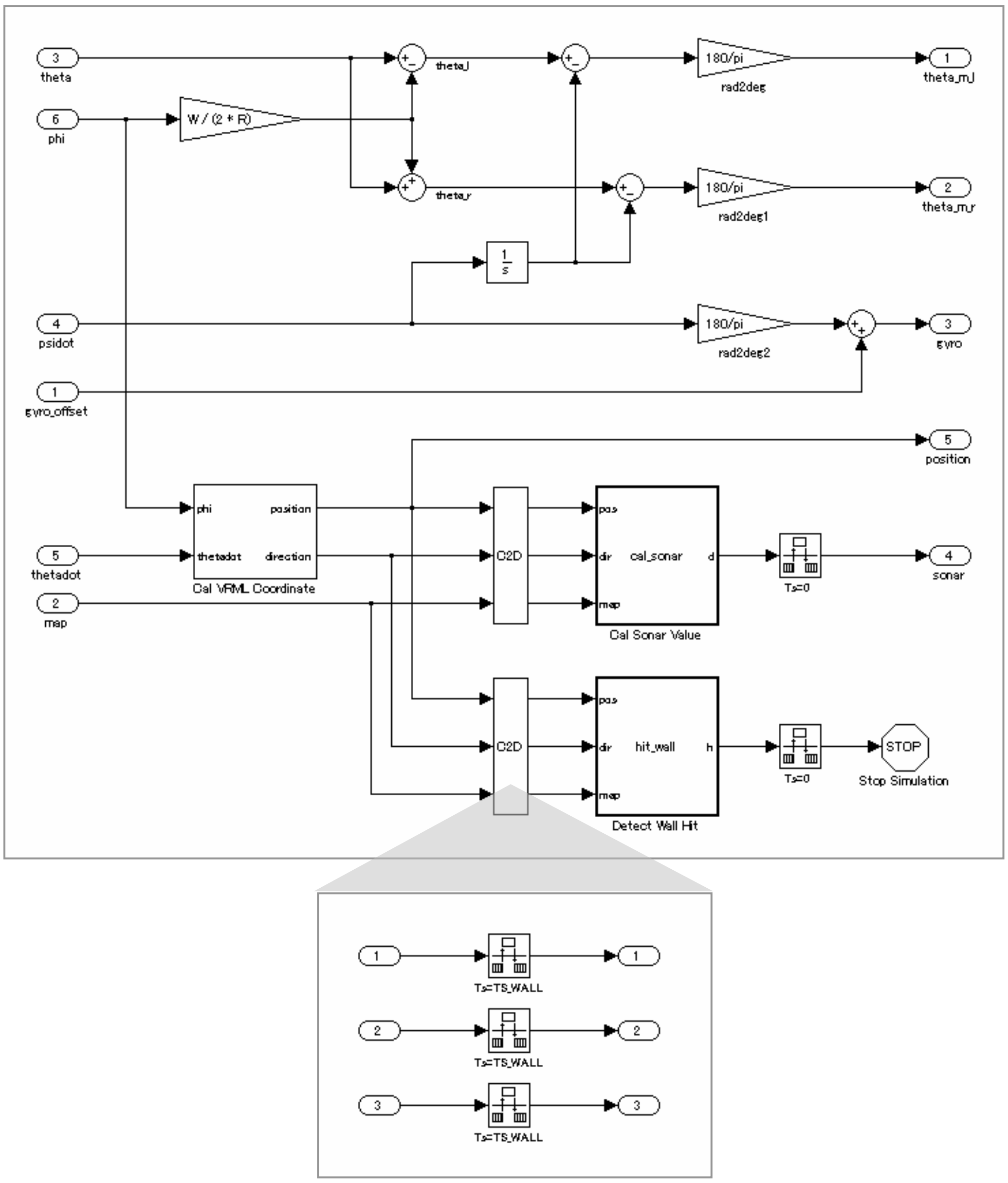


Figure 6-5 Sensor subsystem

7 Controller Model (Single Precision Floating-Point Arithmetic)

This chapter describes control program, task configuration, and model contents of nxtway_gs_controller.mdl.

7.1 Control Program Summary

State Machine Diagram

Figure 7-1 is a state machine diagram of NXTway-GS controller. There are two drive modes. One is autonomous drive mode and the other is remote control drive mode by analog sticks of PC game pad with NXT GamePad utility.

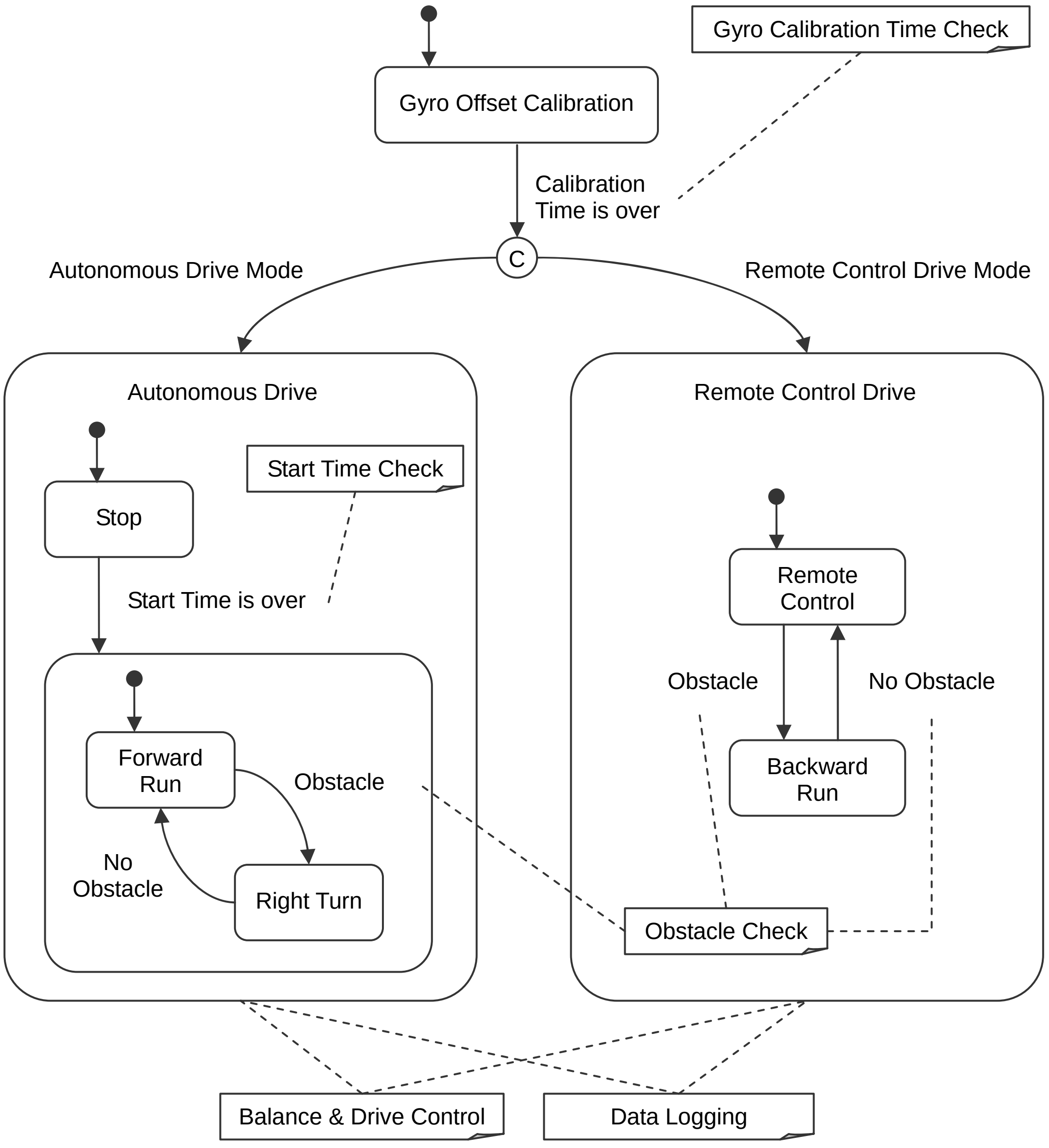


Figure 7-1 State machine diagram of NXTway-GS controller

Task Configuration

NXTway-GS controller has four tasks described in Table 7-1.

Table 7-1 NXTway-GS controller tasks

Task	Period	Works
task_init	Initialization only	Initial value setting
task_ts1	4 [ms]	Balance & drive control Data logging Gyro offset calibration
task_ts2	20 [ms]	Obstacle check
task_ts3	100 [ms]	Time check Battery voltage averaging

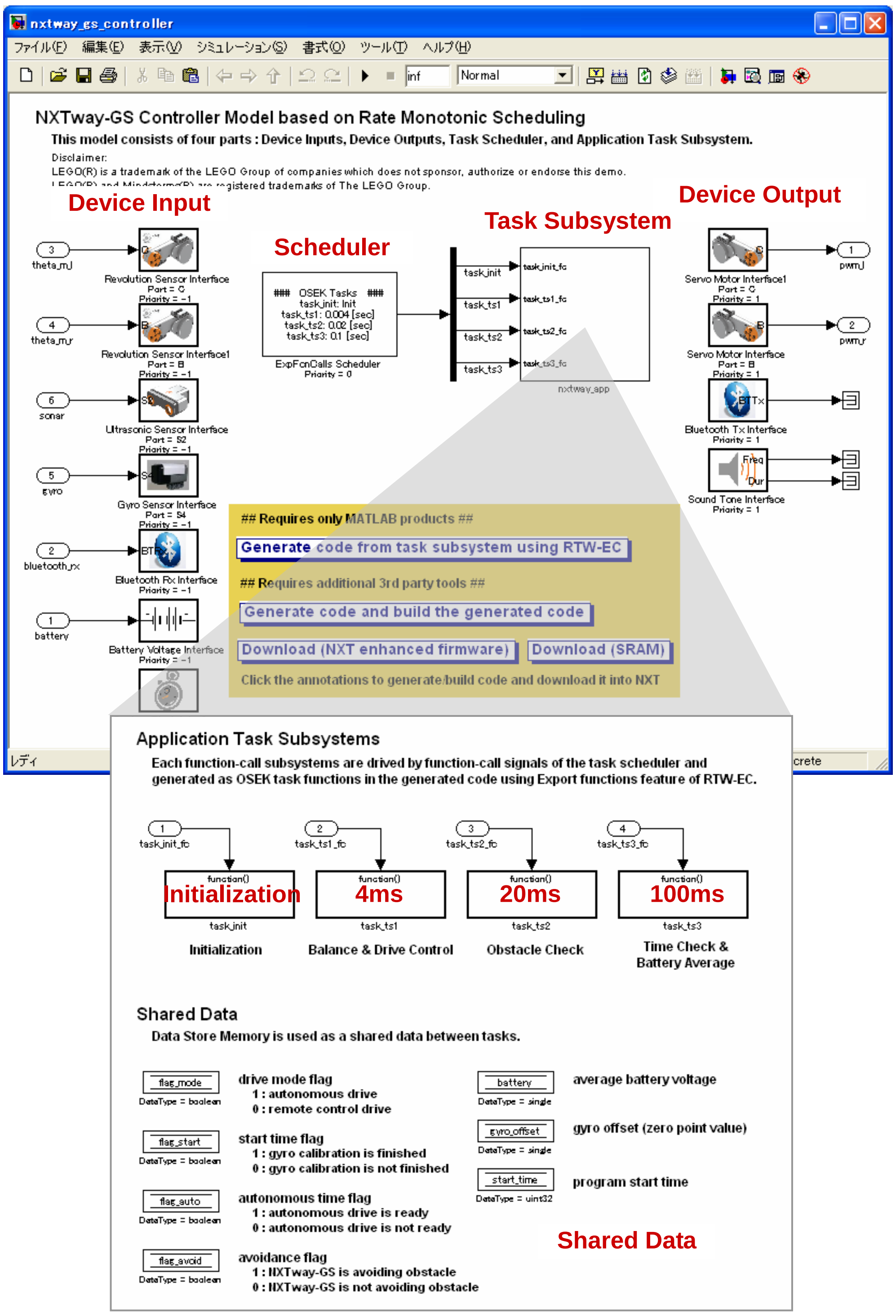
We use 4 [ms] period for the balance & drive control task considering max sample numbers of gyro sensor a second. As same reason, we use 20 [ms] period for the obstacle check task by using ultrasonic sensor. The controller described in 4.2 Controller Design used for the balance & drive control.

Data Type

We use single precision floating-point data for the balance & drive control to reduce the computation error. There are no FPU (Floating Point number processing Unit) in ARM7 processor included in NXT Intelligent Brick, but we can perform software single precision floating-point arithmetic by using floating-point arithmetic library provided by GCC.

7.2 Model Summary

nxtway_gs_controller.mdl is based on Embedded Coder Robot NXT framework.


nxtway_gs_controller
ファイル(F) 編集(E) 表示(V) シミュレーション(S) 書式(O) ツール(T) ヘルプ(H)
inf
Normal
Device Input
Scheduler
Task Subsystem
Device Output
3 theta_m_l
Revolution Sensor Interface Port = C Priority = -1
4 theta_m_r
Revolution Sensor Interface1 Port = B Priority = -1
6 sonar
Ultrasonic Sensor Interface Port = S2 Priority = -1
5 gyro
Gyro Sensor Interface Port = S4 Priority = -1
2 bluetooth_rx
Bluetooth Rx Interface Priority = -1
1 battery
Battery Voltage Interface Priority = -1
ExpFcnCalls Scheduler Priority = 0
task_init
task_ts1
task_ts2
task_ts3
task_init_fc
task_ts1_fc
task_ts2_fc
task_ts3_fc
nxtway_app
Servo Motor Interface1 Port = C Priority = 1
1 pwm_l
Servo Motor Interface Port = B Priority = 1
2 pwm_r
Bluetooth Tx Interface Priority = 1
Sound Tone Interface Priority = 1
Freq
Dur
Generate code from task subsystem using RTW-EC
Generate code and build the generated code
Download (NXT enhanced firmware)
Download (SRAM)
レディ
crete
1 task_init_fc
2 task_ts1_fc
3 task_ts2_fc
4 task_ts3_fc
function()
Initialization
task_init
function()
4ms
task_ts1
function()
20ms
task_ts2
function()
100ms
task_ts3
Shared Data

NXTway-GS Controller Model based on Rate Monotonic Scheduling

This model consists of four parts : Device Inputs, Device Outputs, Task Scheduler, and Application Task Subsystem.

Disclaimer:
LEGO(R) is a trademark of the LEGO Group of companies which does not sponsor, authorize or endorse this demo.
LEGO(R) and Mindstorms(R) are registered trademarks of The LEGO Group.

```
###   OSEK Tasks   ###
task_init: Init
task_ts1: 0.004 [sec]
task_ts2: 0.02 [sec]
task_ts3: 0.1 [sec]
```

Requires only MATLAB products

Requires additional 3rd party tools

Click the annotations to generate/build code and download it into NXT

Application Task Subsystems

Each function-call subsystems are drived by function-call signals of the task scheduler and generated as OSEK task functions in the generated code using Export functions feature of RTW-EC.

task_init	task_ts1	task_ts2	task_ts3
Initialization	Balance & Drive Control	Obstacle Check	Time Check & Battery Average

Shared Data

Data Store Memory is used as a shared data between tasks.

Data Store	DataType	Description
flag_mode	boolean	drive mode flag 1 : autonomous drive 0 : remote control drive
flag_start	boolean	start time flag 1 : gyro calibration is finished 0 : gyro calibration is not finished
flag_auto	boolean	autonomous time flag 1 : autonomous drive is ready 0 : autonomous drive is not ready
flag_avoid	boolean	avoidance flag 1 : NXTway-GS is avoiding obstacle 0 : NXTway-GS is not avoiding obstacle
battery	single	average battery voltage
gyro_offset	single	gyro offset (zero point value)
start_time	uint32	program start time

Figure 7-2 nxtway_gs_controller.mdl

Device Interface

We can make device interfaces by using the sensor and actuator blocks provided by Embedded Coder Robot NXT library.

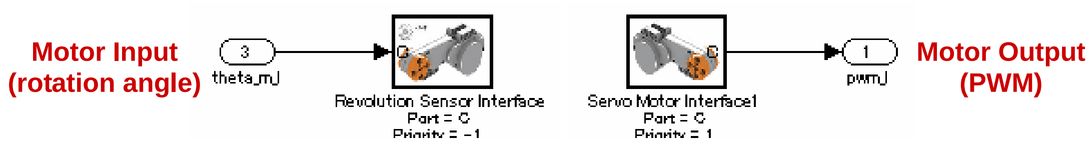
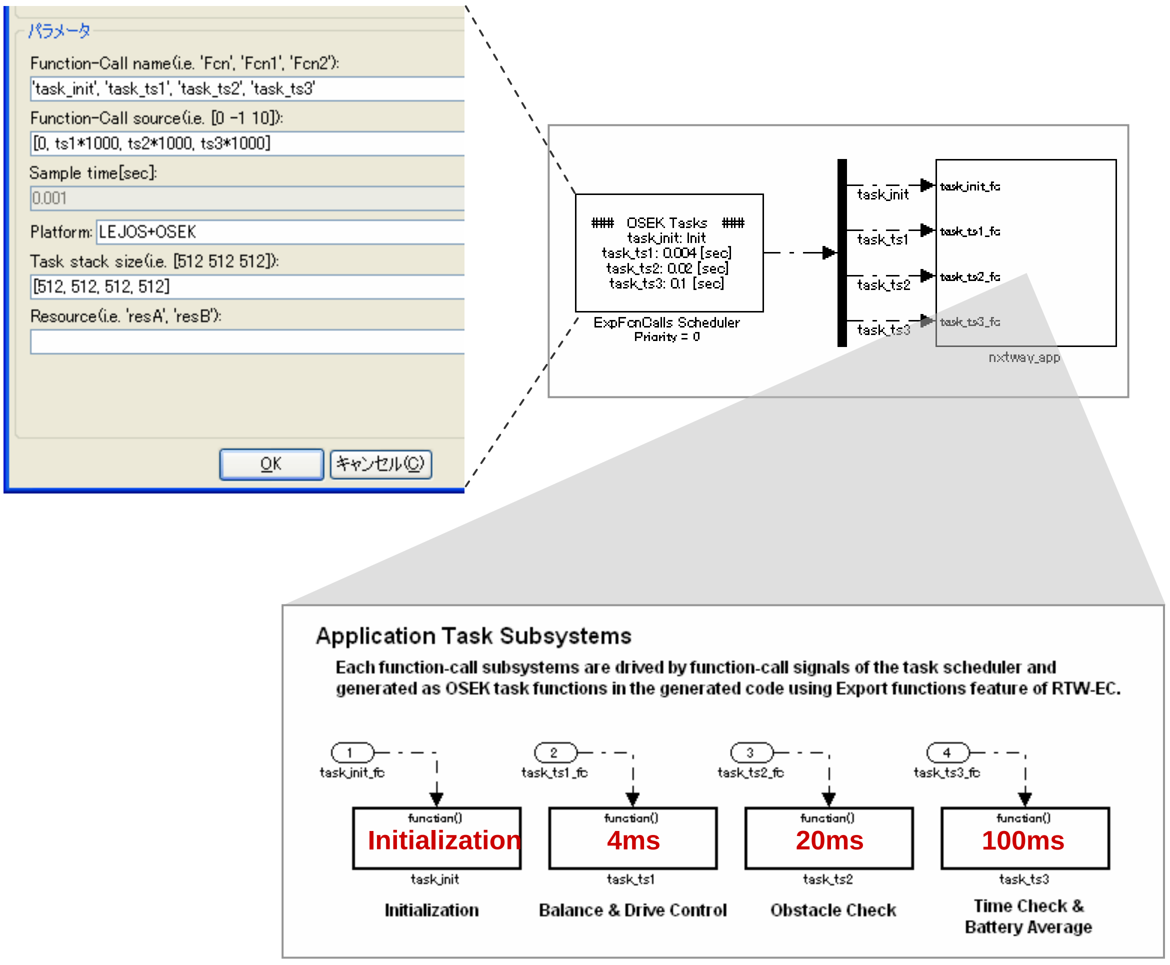


Figure 7-3 DC motor interface

Scheduler & Tasks

The ExpFcnCalls Scheduler block has task configuration such as task name, task period, platform, and stack size. We can make task subsystems by connecting function-call signals from the scheduler to function-call subsystems.



パラメータ

Function-Call name(i.e. 'Fcn', 'Fcn1', 'Fcn2'):
'task_init', 'task_ts1', 'task_ts2', 'task_ts3'

Function-Call source(i.e. [0 -1 10]):
[0, ts1*1000, ts2*1000, ts3*1000]

Sample time[sec]:
0.001

Platform: LEJOS+OSEK

Task stack size(i.e. [512 512 512]):
[512, 512, 512, 512]

Resource(i.e. 'resA', 'resB'):

OSEK Tasks
task_init: Init
task_ts1: 0.004 [sec]
task_ts2: 0.02 [sec]
task_ts3: 0.1 [sec]

Application Task Subsystems

Each function-call subsystems are drived by function-call signals of the task scheduler and generated as OSEK task functions in the generated code using Export functions feature of RTW-EC.

Figure 7-4 Scheduler and tasks

Priority

We have to set priority of the device blocks and ExpFcnCalls Scheduler block at root level to sort them in the order 1. device inputs $\rightarrow$ 2. tasks $\rightarrow$ 3. device outputs. The low number indicates high priority and negative numbers are allowed.

To display priority, right click the block and choose [**Block Properties**].

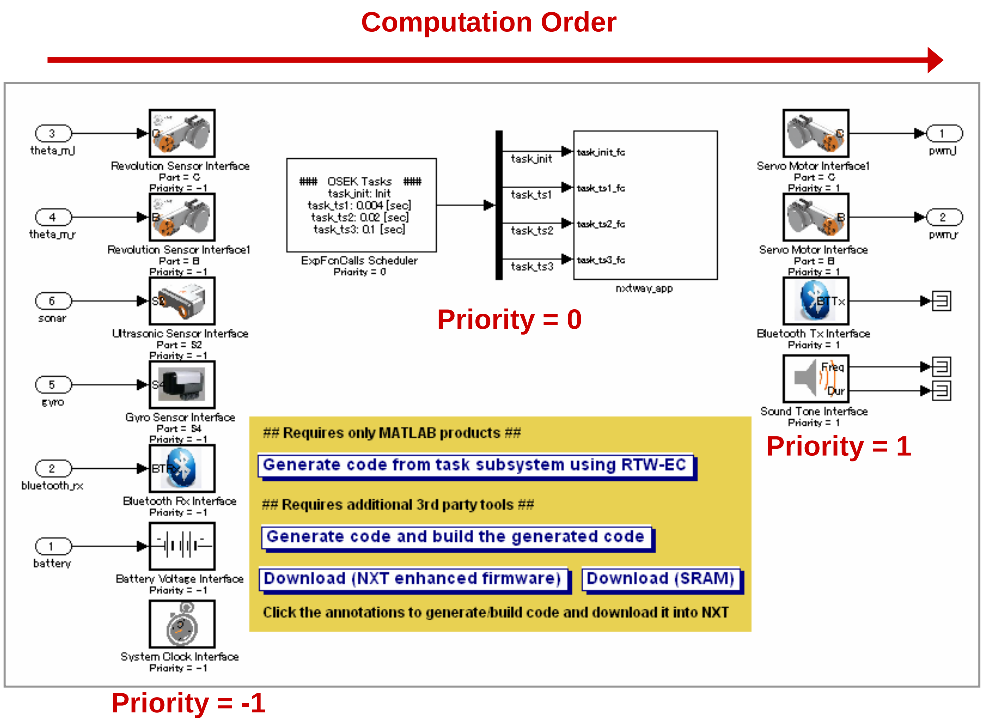
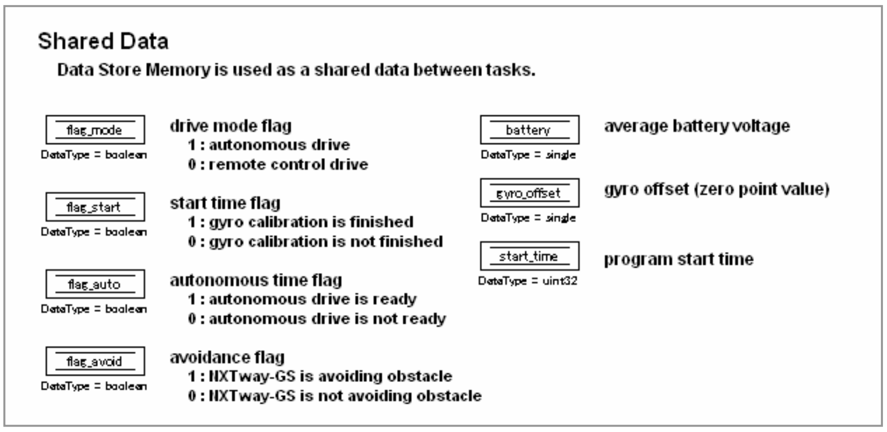


Figure 7-5 Priority setting

Shared Data

We use Data Store Memory blocks as shared data between tasks.



Shared Data

Data Store Memory is used as a shared data between tasks.

flag_mode DataType = boolean	drive mode flag 1 : autonomous drive 0 : remote control drive
flag_start DataType = boolean	start time flag 1 : gyro calibration is finished 0 : gyro calibration is not finished
flag_auto DataType = boolean	autonomous time flag 1 : autonomous drive is ready 0 : autonomous drive is not ready
flag_avoid DataType = boolean	avoidance flag 1 : NXTway-GS is avoiding obstacle 0 : NXTway-GS is not avoiding obstacle
battery DataType = single	average battery voltage
gyro_offset DataType = single	gyro offset (zero point value)
start_time DataType = uint32	program start time

Figure 7-6 Shared data

7.3 Initialization Task : task_init

This task sets the initial values. We can switch the drive mode. (autonomous drive mode or remote control drive mode)

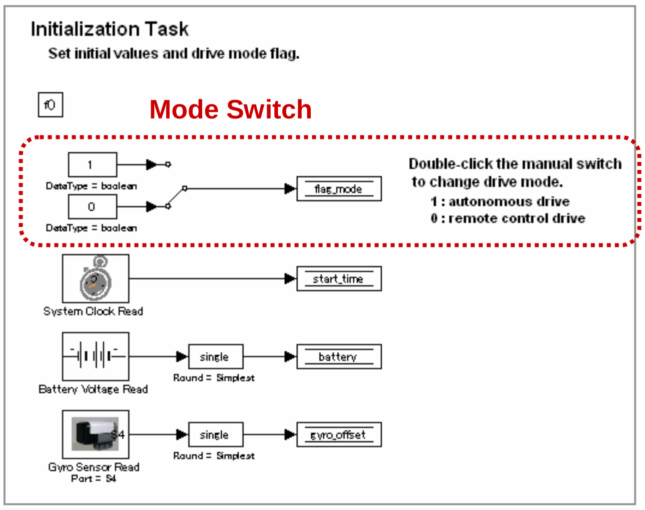


Figure 7-7 task_init subsystem

7.4 4ms Task : task_ts1

This is a task for gyro offset calibration, balance & drive control, and data logging. The balance & drive control runs after the gyro calibration. The gyro offset calibration time is defined as `time_start` in param_controller.m.

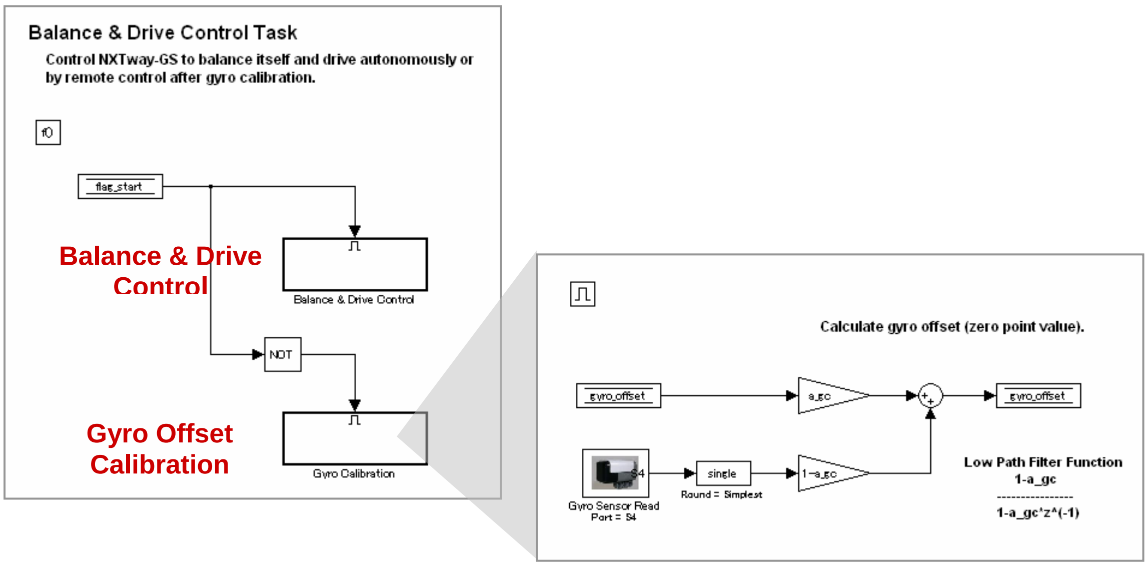


Figure 7-8 task_ts1 subsystem

Balance & Drive Control subsystem is a controller described in 4.2 Controller Design.

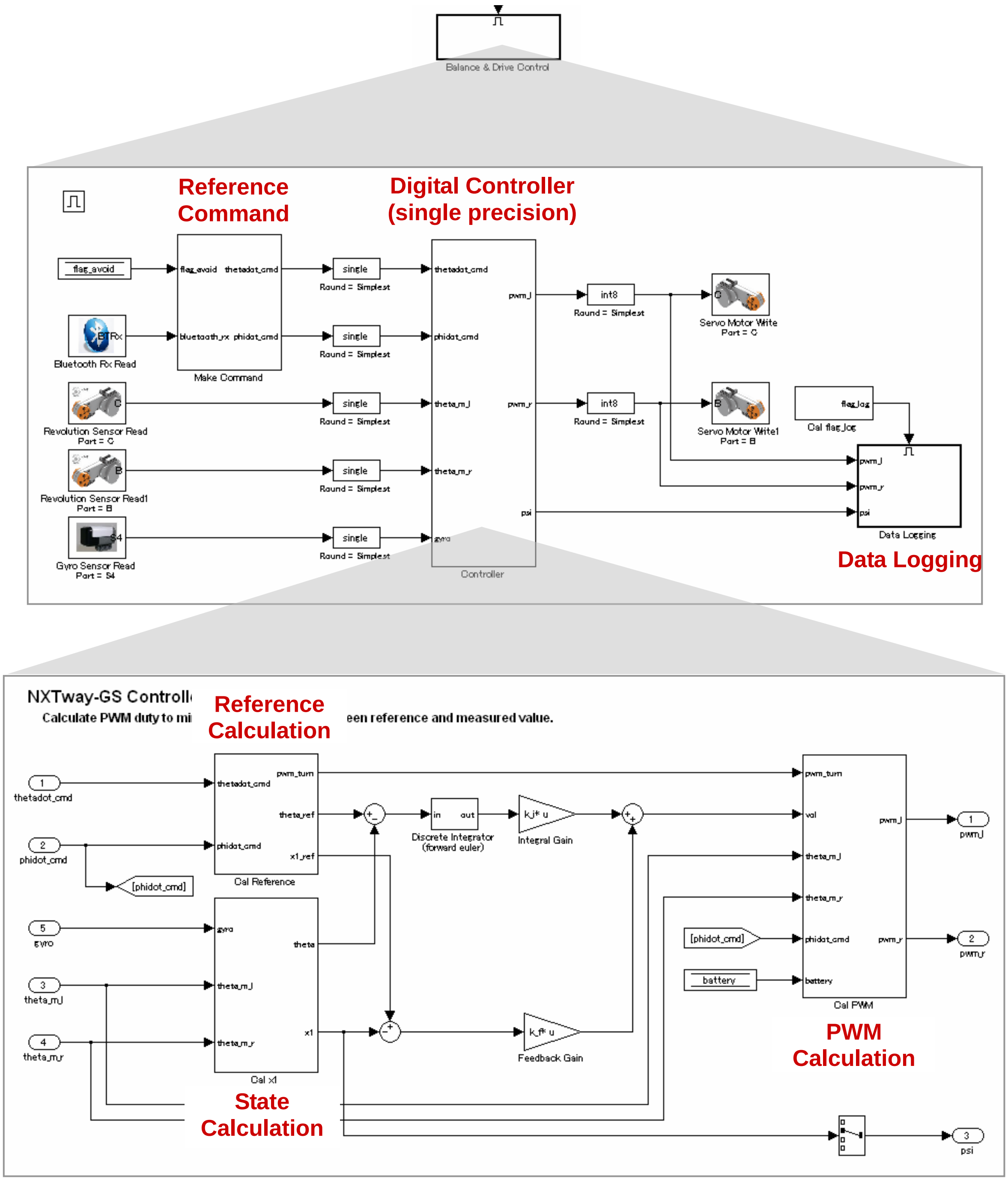


Figure 7-9 Controller model for balance & drive control

Discrete Derivative and Discrete Integrator block

Discrete Derivative block computes a time derivative by backward difference method, and Discrete Integrator block does a time integral by forward euler method.

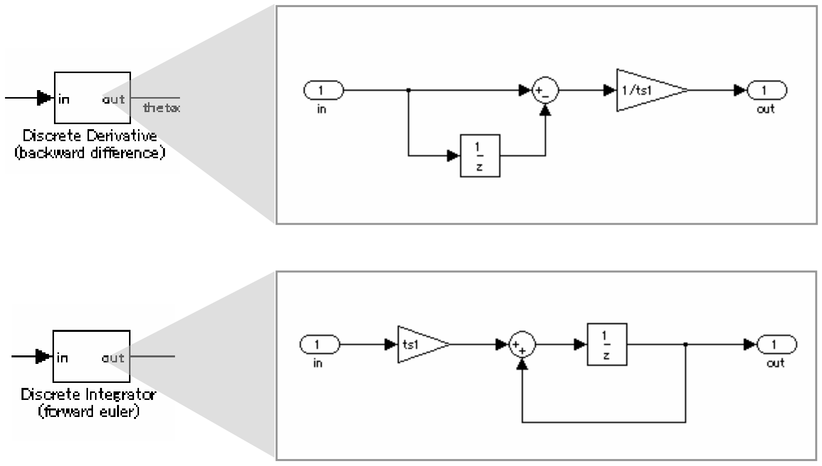


Figure 7-10　Discrete Derivative and Discrete Integrator block

Reference Calculation (Cal Reference subsystem)

This subsystem calculates the reference using a low path filter to reduce an overshoot induced by a rapid change of speed reference signal.

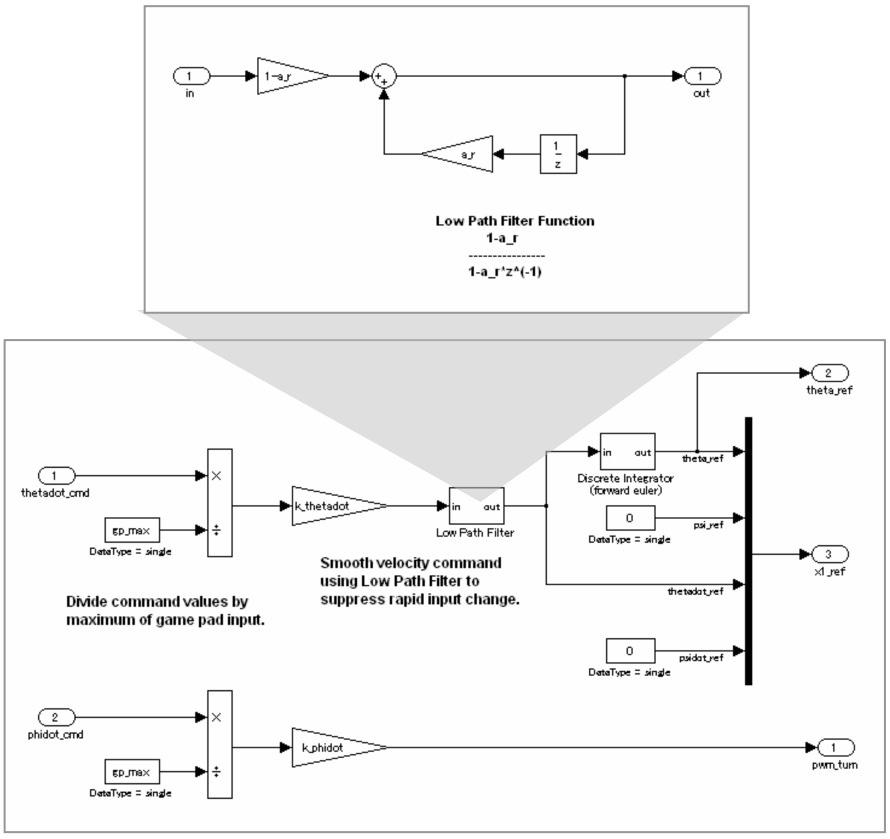


Figure 7-11　Reference calculation

State Calculation (Cal x1)

This subsystem calculates the state using outputs from the sensors. We use a long-time average of gyro output as gyro offset to remove gyro drift effect, and a low path filter to remove noise in the velocity signal.

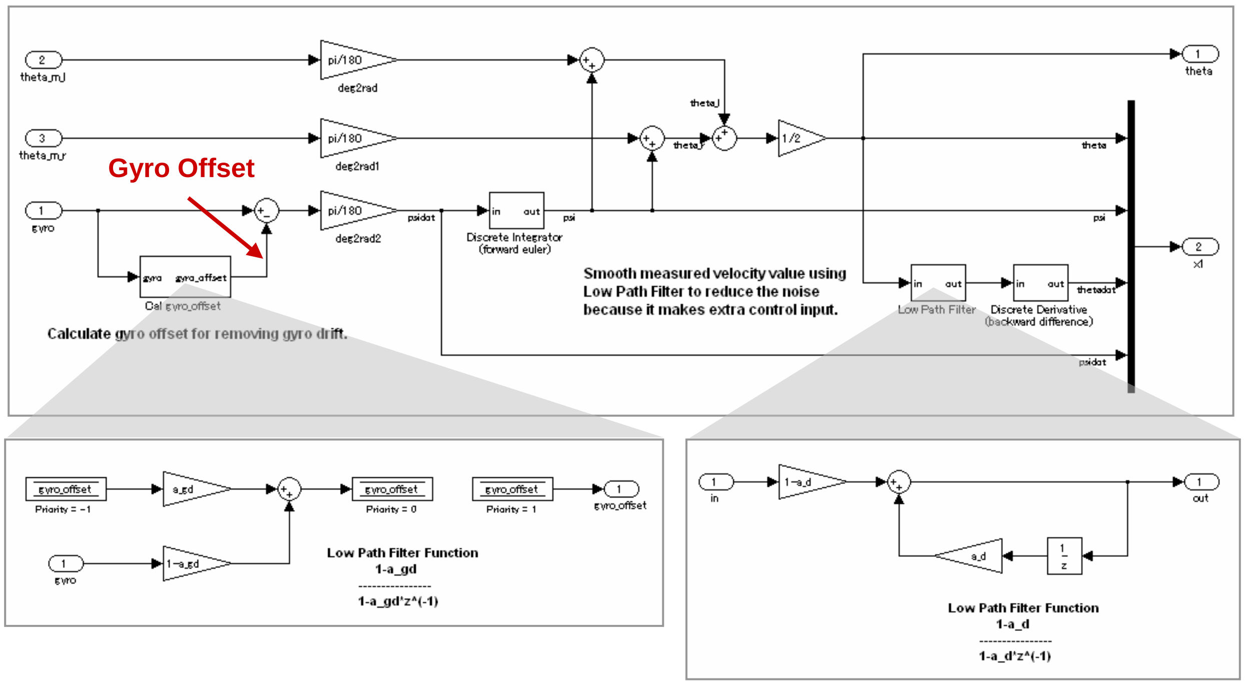
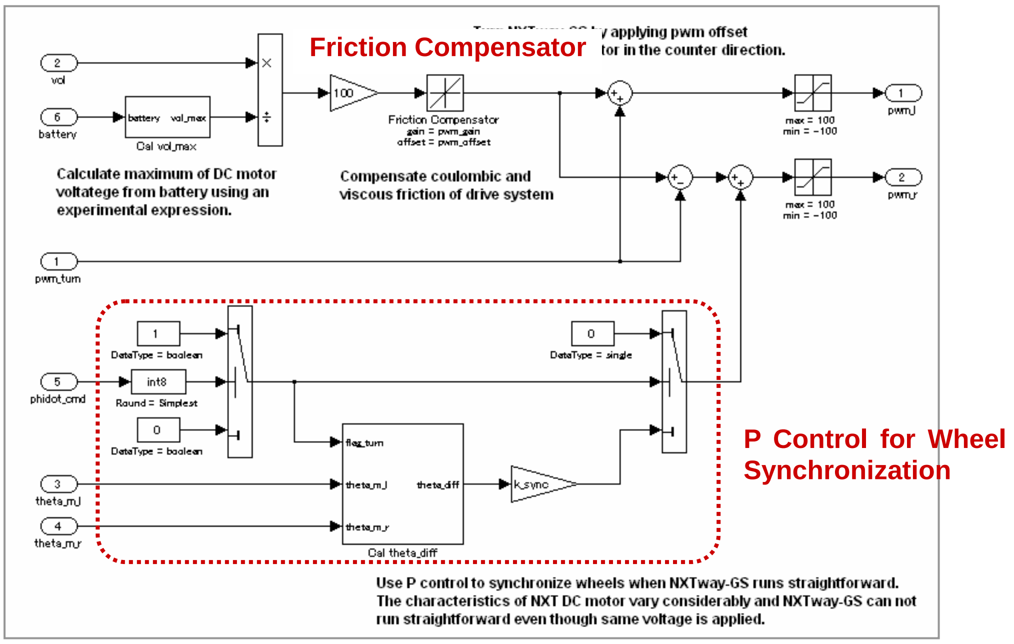


Figure 7-12 State calculation

PWM Calculation (Cal PWM)

This subsystem calculates the PWM duty based on the block diagram shown in Figure 4-4 and maximum motor voltage by using Eq. (6.1).



Friction Compensator
applying pwm offset
tor in the counter direction.
vol
battery
Cal vol_max
100
Friction Compensator
gain = pwm_gain
offset = pwm_offset
max = 100
min = -100
pwm_l
pwm_r
Calculate maximum of DC motor voltage from battery using an experimental expression.
Compensate coulombic and viscous friction of drive system
pwm_turn
phidot_cmd
DataType = boolean
int8
Round = Simplest
DataType = single
flag_turn
theta_m_l
theta_m_r
theta_diff
k_sync
Cal theta_diff
P Control for Wheel Synchronization
Use P control to synchronize wheels when NXTway-GS runs straightforward. The characteristics of NXT DC motor vary considerably and NXTway-GS can not run straightforward even though same voltage is applied.

Figure 7-13 PWM calculation

Data Logging

The NXT GamePad ADC Data Logger block is placed in this subsystem for logging the sensor output and calculation result. To change the logging period, change `log_count` defined in param_controller.m.

The NXT GamePad ADC Data Logger block can log sensor outputs and 8-bit, 16-bit signed integer data. It is necessary for logging floating-point data to quantize it by dividing appropriate value, because it does not support floating-point data directly. The denominator of the dividing operation is called as Slope or LSB, it corresponds a value per 1-bit. We can derive an original value by multiplying the logged data and Slope (LSB).

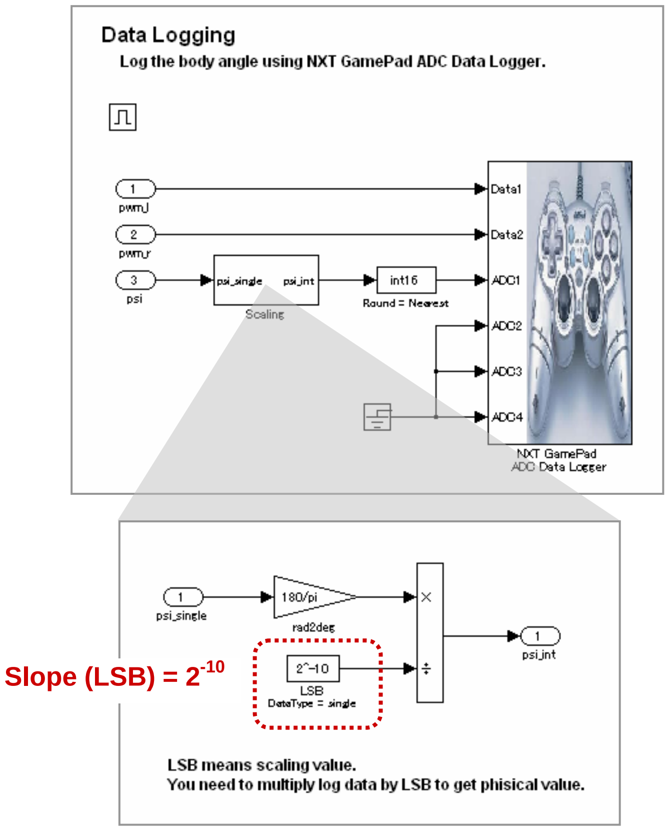
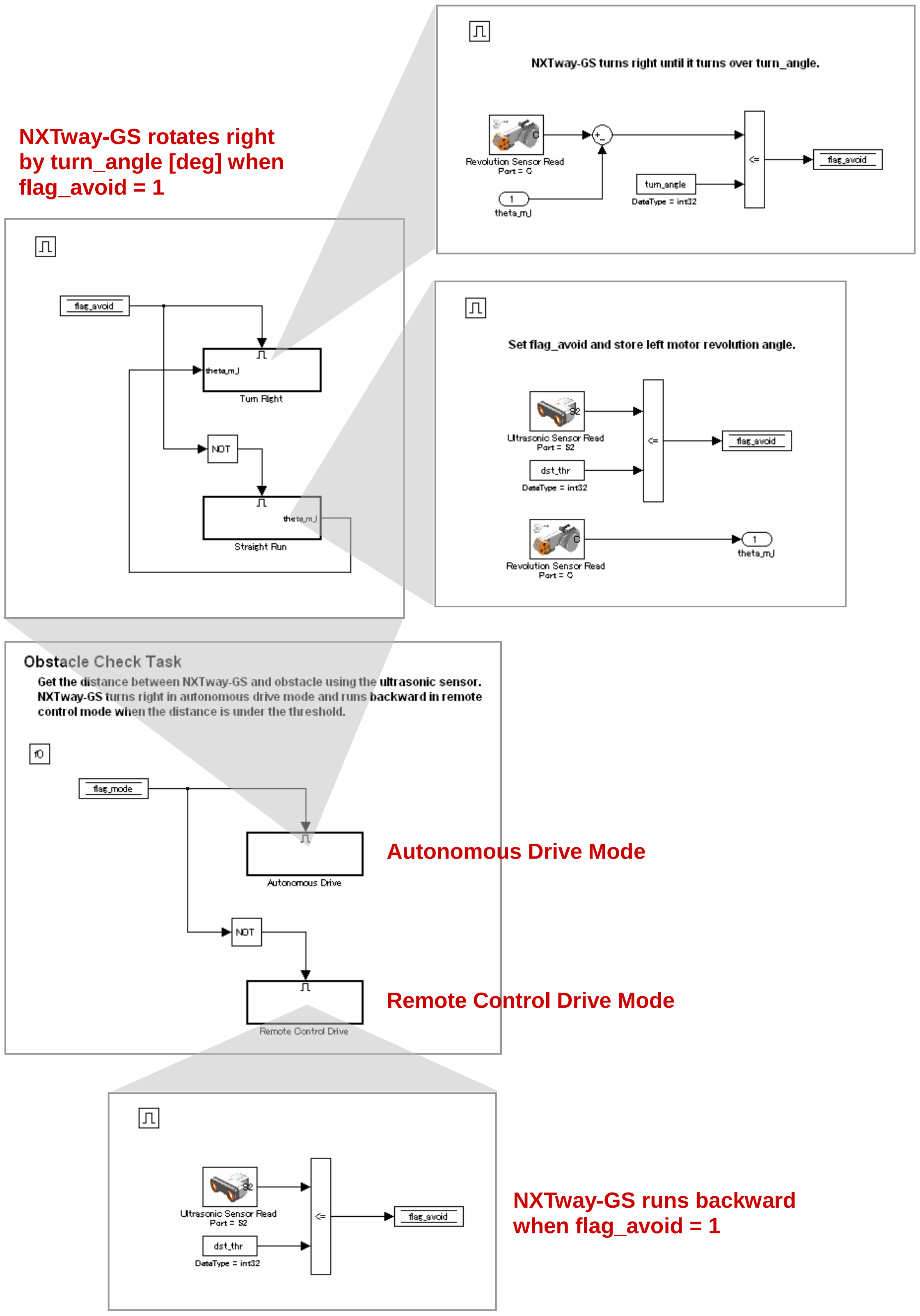


Figure 7-14　Data Logging

7.5 20ms Task : task_ts2

This is a task to check an obstacle with ultrasonic sensor. NXTway-GS rotates right in autonomous drive mode and runs backward in remote control drive mode when finding an obstacle.


NXTway-GS rotates right by turn_angle [deg] when flag_avoid = 1
NXTway-GS turns right until it turns over turn_angle.
Revolution Sensor Read Port = C
theta_m_l
turn_angle DataType = int32
<=
flag_avoid
Turn Right
NOT
Straight Run
Set flag_avoid and store left motor revolution angle.
Ultrasonic Sensor Read Port = S2
dst_thr DataType = int32
Obstacle Check Task
f()
flag_mode
Autonomous Drive Mode
Autonomous Drive
Remote Control Drive Mode
Remote Control Drive
NXTway-GS runs backward when flag_avoid = 1

Get the distance between NXTway-GS and obstacle using the ultrasonic sensor. NXTway-GS turns right in autonomous drive mode and runs backward in remote control mode when the distance is under the threshold.

Figure 7-15 task_ts2 subsystem

7.6 100ms Task : task_ts3

This is a task to check time and average battery voltage. NXTway-GS sounds during `sound_dur` [ms] when the gyro calibration is finished.

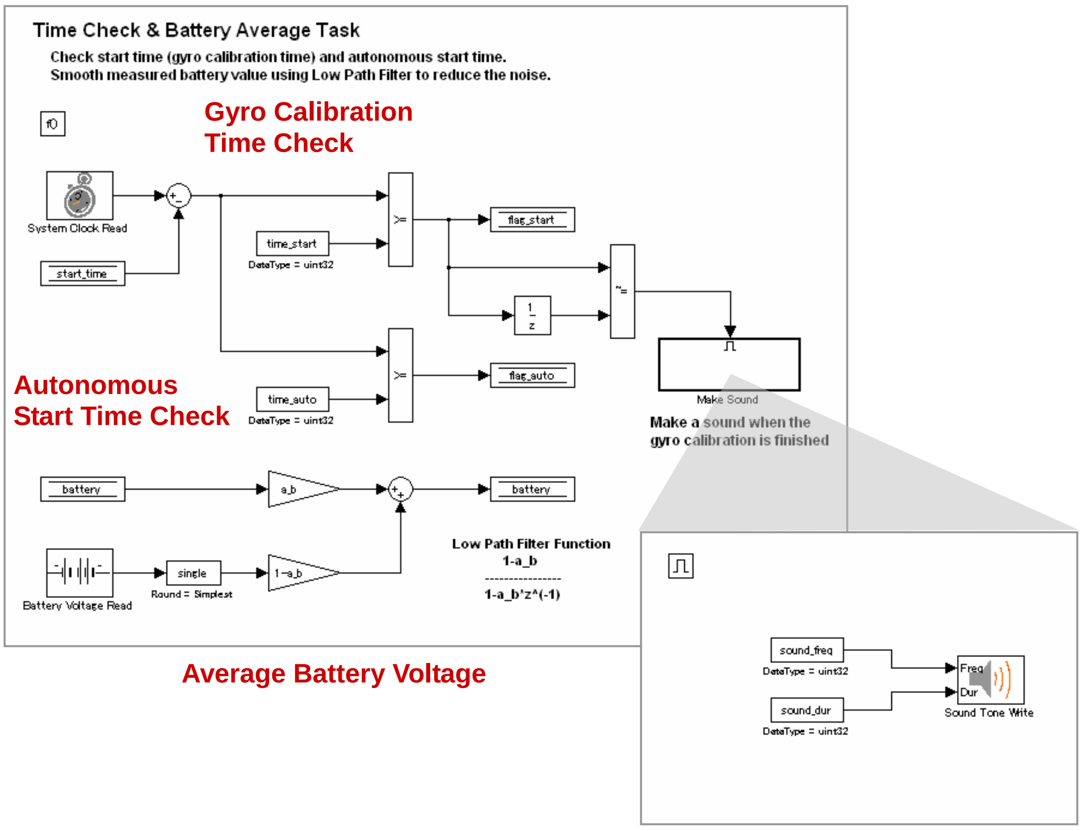


Figure 7-16 task_ts3 subsystem

7.7 Tuning Parameters

All parameters used in nxtway_gs_controller.mdl are defined in param_controller.m. Table 7-2 shows the tuning parameters for balance & drive control. You might have to tune these parameters because the parts, blocks, sensors, and actuators are different individually.

Table 7-2 Tuning parameters

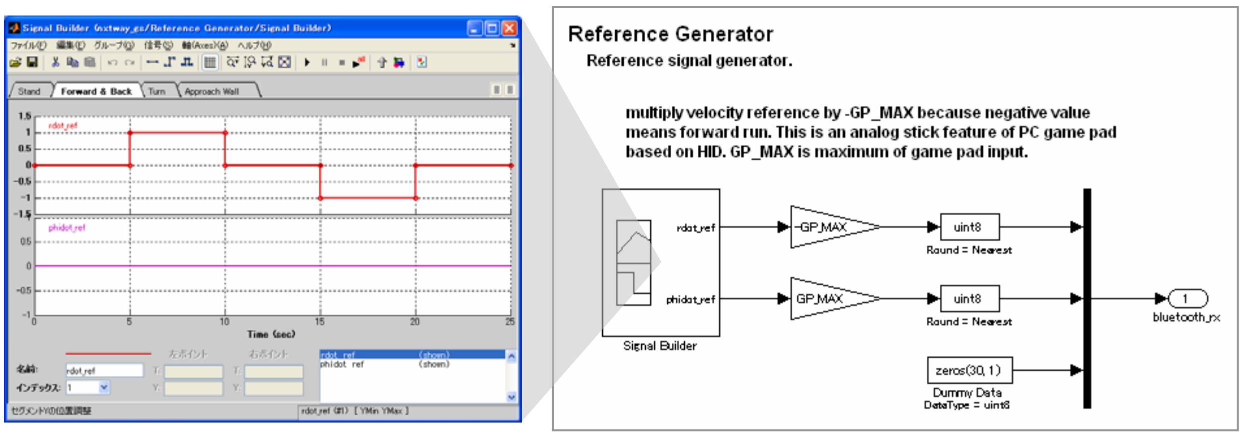
Parameter	Description
`k_f`	Servo control feedback gain
`k_i`	Servo control integral gain
`k_thetadot`	Speed reference gain
`k_phidot`	Rotation speed reference gain
`k_sync`	P control gain for wheel synchronization
`a_b`	Low path filter coefficient for averaging battery voltage
`a_d`	Low path filter coefficient for removing velocity noise
`a_r`	Low path filter coefficient for removing overshoot
`a_gc`	Low path filter coefficient for gyro calibration
`a_gd`	Low path filter coefficient for removing gyro drift effect
`pwm_gain`	Friction compensator gain
`pwm_offset`	Friction compensator offset
`dst_thr`	Obstacle avoidance threshold (distance)
`turn_angle`	Rotation angle in autonomous drive mode

8 Simulation

This chapter describes simulation of NXTway-GS model, its result, and 3D viewer in nxtway_gs_vr.mdl.

8.1 How to Run Simulation

It is same way as usual models to run simulation. We can change the speed reference and the rotation speed reference by using Signal Builder block in Reference Generator subsystem.



multiply velocity reference by -GP_MAX because negative value means forward run. This is an analog stick feature of PC game pad based on HID. GP_MAX is maximum of game pad input.

Figure 8-1 Reference Generator subsystem

8.2 Simulation Results

Stationary Balancing

Figure 8-2 shows a simulation result of stationary balancing at initial value of body pitch angle equals to 5 [deg]. The body pitch angle goes to be 0 [deg] quickly. NXTway-GS does not run until 1 [s] passes because of gyro calibration.

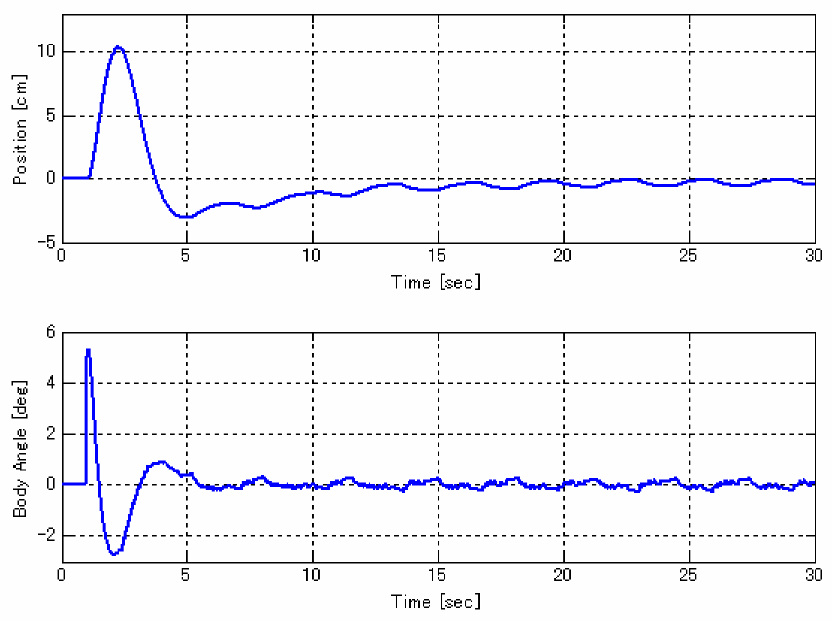


Figure 8-2 Position and body pitch angle of stationary balancing

Forward and Backward Run

Figure 8-3 shows a simulation result of forward and backward run. NXTway-GS fluctuates at the time it runs or stops, but it goes to be stable gradually.

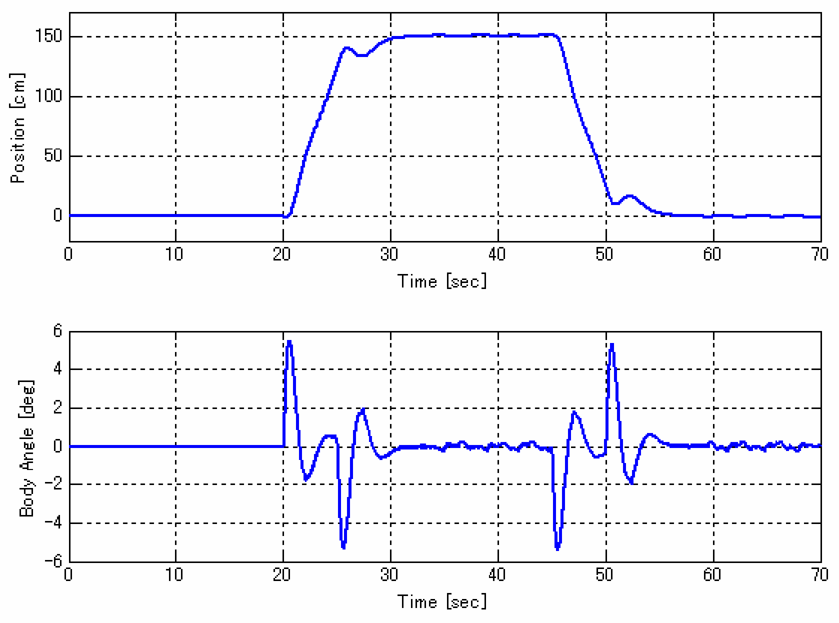


Figure 8-3 Position and body pitch angle of forward and backward run

You can watch the movie of NXTway-GS simulation at the following URL.

http://www.youtube.com/watch?v=EHPlGTLQHRc

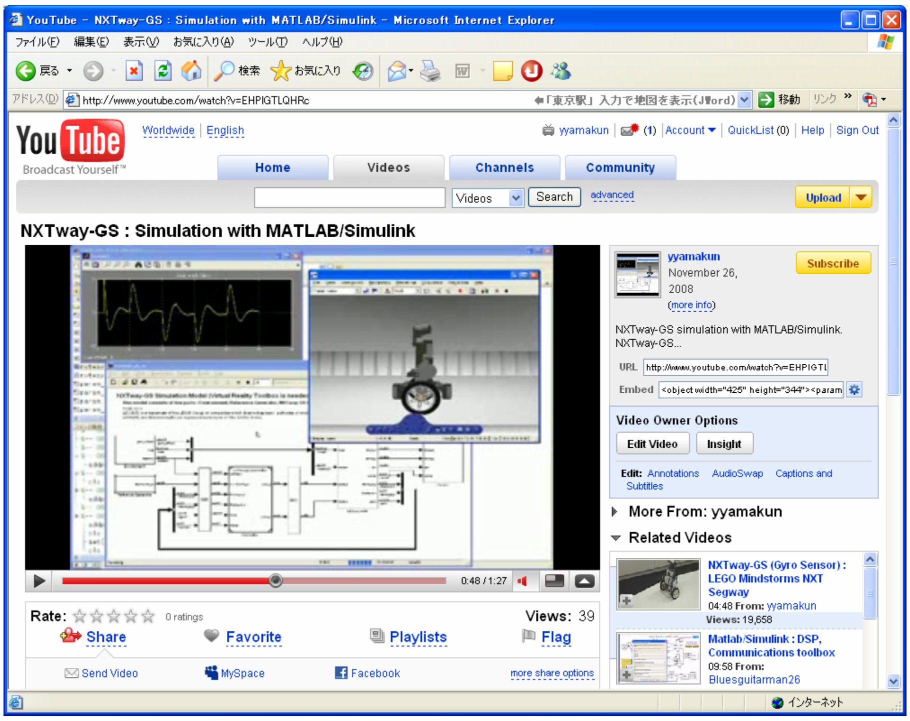
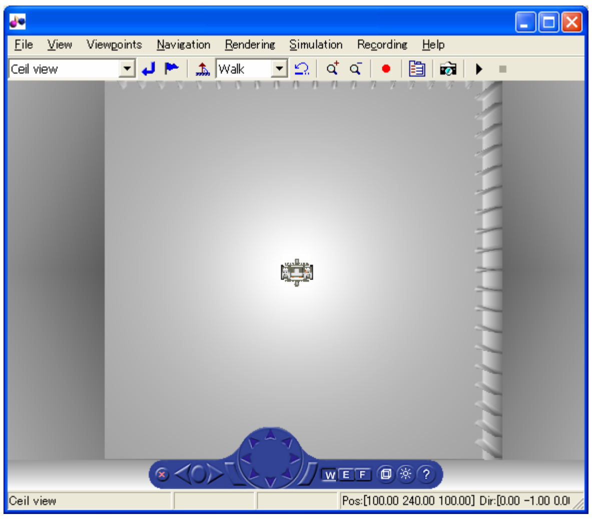

YouTube - NXTway-GS : Simulation with MATLAB/Simulink - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り
http://www.youtube.com/watch?v=EHPlGTLQHRc
移動 リンク
Worldwide | English
yyamakun (1) Account QuickList (0) Help Sign Out
Home Videos Channels Community
Videos Search advanced Upload
NXTway-GS : Simulation with MATLAB/Simulink
yyamakun November 26, 2008 (more info) Subscribe
NXTway-GS simulation with MATLAB/Simulink. NXTway-GS...
URL http://www.youtube.com/watch?v=EHPlGTL
Embed <object width="425" height="344"><param
Video Owner Options
Edit Video Insight
Edit: Annotations AudioSwap Captions and Subtitles
More From: yyamakun
Related Videos
NXTway-GS (Gyro Sensor) : LEGO Mindstorms NXT Segway
04:48 From: yyamakun
Views: 19,658
Matlab/Simulink : DSP, Communications toolbox
09:58 From: Bluesguitarman26
0:48 / 1:27
Rate: 0 ratings
Views: 39
Share Favorite Playlists Flag
Send Video MySpace Facebook more share options
インターネット

Figure 8-4　Movie of NXTway-GS simulation

8.3 3D Viewer

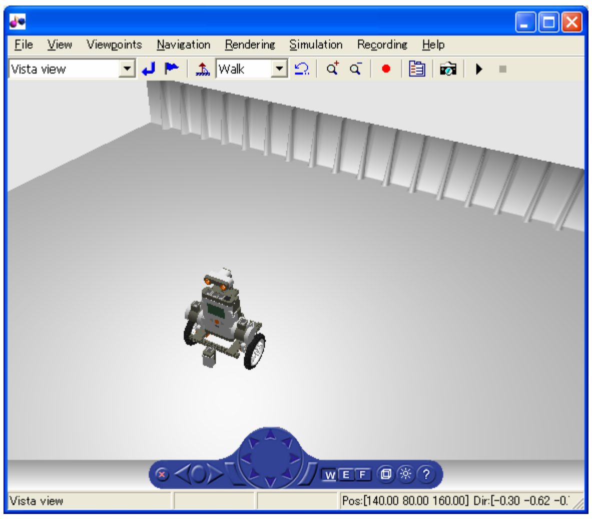
In nxtway_gs_vr.mdl, we can watch 3D simulation with the viewer provided by Virtual Reality Toolbox. We can change the viewpoint by switching the view mode in Virtual Reality window. nxtway_gs_vr.mdl has four view modes.

● Ceil View	:	Overhead viewpoint
● Vista View	:	Fixed viewpoint
● Robot View	:	Viewpoint on the ultrasonic sensor
● Chaser View	:	Tracking viewpoint


File View Viewpoints Navigation Rendering Simulation Recording Help
Ceil view
Walk
W E F

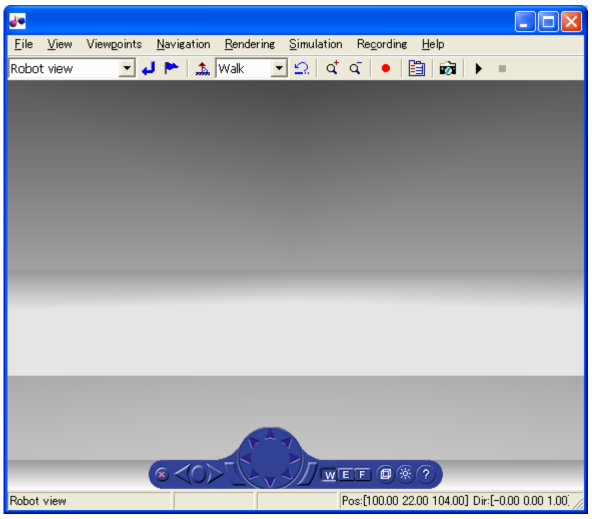
```
Ceil view    Pos:[100.00 240.00 100.00] Dir:[0.00 -1.00 0.0
```

Ceil View


File View Viewpoints Navigation Rendering Simulation Recording Help
Vista view
Walk
W E F

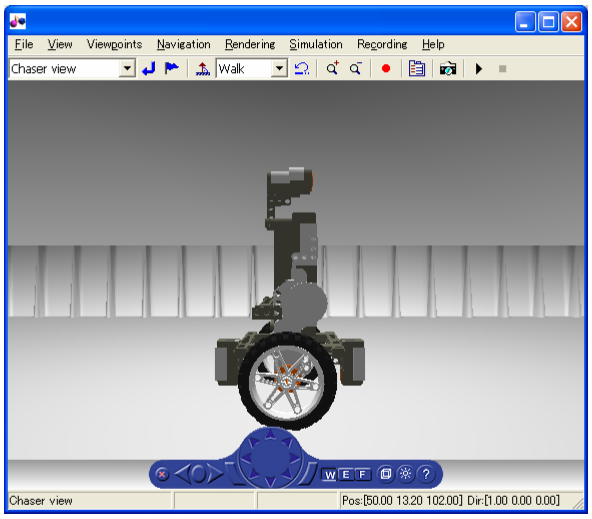
```
Vista view    Pos:[140.00 80.00 160.00] Dir:[-0.30 -0.62 -0.
```

Vista View


File View Viewpoints Navigation Rendering Simulation Recording Help
Robot view
Walk
W E F

```
Robot view    Pos:[100.00 22.00 104.00] Dir:[-0.00 0.00 1.00
```

Robot View


File View Viewpoints Navigation Rendering Simulation Recording Help
Chaser view
Walk
W E F

```
Chaser view    Pos:[50.00 13.20 102.00] Dir:[1.00 0.00 0.00]
```

Chaser View

Figure 8-5 View modes

9 Code Generation and Implementation

This chapter describes how to generate code from nxtway_gs_controller.mdl and download it into NXT Intelligent Brick, and the experimental results are shown.

9.1 Target Hardware & Software

Table 9-1 shows the target hardware specification of LEGO Mindstorms NXT and the software used in Embedded Coder Robot NXT.

Table 9-1 LEGO Mindstorms NXT & Embedded Coder Robot NXT specification

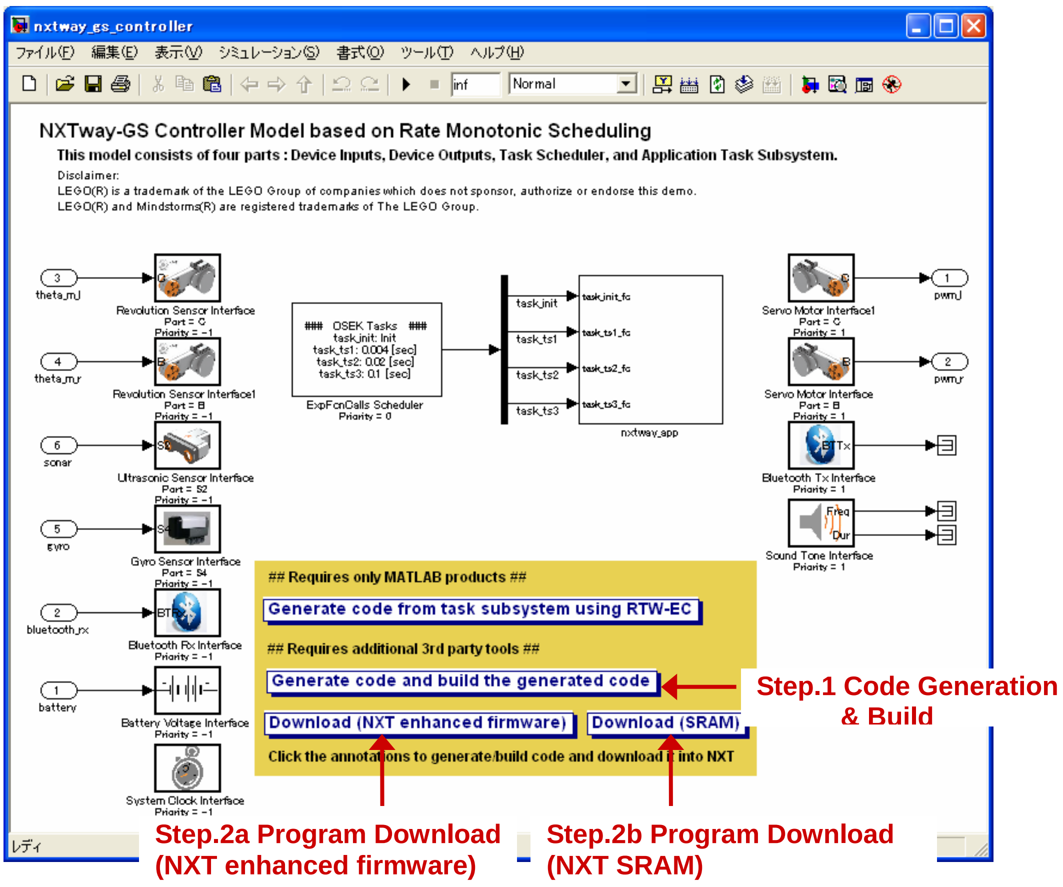
Hardware	Processor	ATMEL 32-bit ARM 7 (AT91SAM7S256) 48MHz
	Flash Memory	256 Kbytes (10000 time writing guarantee)
	RAM	64 Kbytes
Interface	Actuator	DC motor
	Sensor	Ultrasonic, Touch, Light, Sound
	Display	100 * 64 pixel LCD
	Communication	Bluetooth
Software	RTOS	LEJOS C / LEJOS OSEK
	Compiler	GCC
	Library	GCC library

You can not download a program when the program size is greater than SRAM or flash memory size (you will encounter a download error).

9.2 How to Generate Code and Download

You can generate code from the model, build it, and download the program into NXT by clicking the annotations in nxtway_gs_controller.mdl shown in Figure 9-1. The procedure is the following.

1. Generate code and build the generated code by clicking [**Generate code and build the generated code**].
2. Connect NXT and PC via USB. Download the program into NXT by clicking [**Download (NXT enhanced firmware)**] or [**Download (SRAM)**] in accordance with the boot mode of NXT (enhanced firmware or SRAM boot).



nxtway_gs_controller
ファイル(F) 編集(E) 表示(V) シミュレーション(S) 書式(O) ツール(T) ヘルプ(H)
inf
Normal
NXTway-GS Controller Model based on Rate Monotonic Scheduling
This model consists of four parts : Device Inputs, Device Outputs, Task Scheduler, and Application Task Subsystem.
Disclaimer:
LEGO(R) is a trademark of the LEGO Group of companies which does not sponsor, authorize or endorse this demo.
LEGO(R) and Mindstorms(R) are registered trademarks of The LEGO Group.
theta_m_l
Revolution Sensor Interface
theta_m_r
Revolution Sensor Interface1
sonar
Ultrasonic Sensor Interface
gyro
Gyro Sensor Interface
bluetooth_rx
Bluetooth Rx Interface
battery
Battery Voltage Interface
System Clock Interface
### OSEK Tasks ###
task_init: Init
task_ts1: 0.004 [sec]
task_ts2: 0.02 [sec]
task_ts3: 0.1 [sec]
ExpFcnCalls Scheduler
nxtway_app
Servo Motor Interface1
pwm_l
Servo Motor Interface
pwm_r
Bluetooth Tx Interface
Sound Tone Interface
## Requires only MATLAB products ##
Generate code from task subsystem using RTW-EC
## Requires additional 3rd party tools ##
Generate code and build the generated code
Download (NXT enhanced firmware)
Download (SRAM)
Click the annotations to generate/build code and download it into NXT
Step.1 Code Generation & Build
Step.2a Program Download (NXT enhanced firmware)
Step.2b Program Download (NXT SRAM)
レディ

Figure 9-1 Annotations for code generation / build / download

A part of the generated code is described in Appendix C.

9.3 Experimental Results

The experimental results are shown as follows. We have derived similar results as the simulation results.

Stationary Balancing

Figure 9-2 shows an experimental result of stationary balancing. NXTway-GS is stable very well.

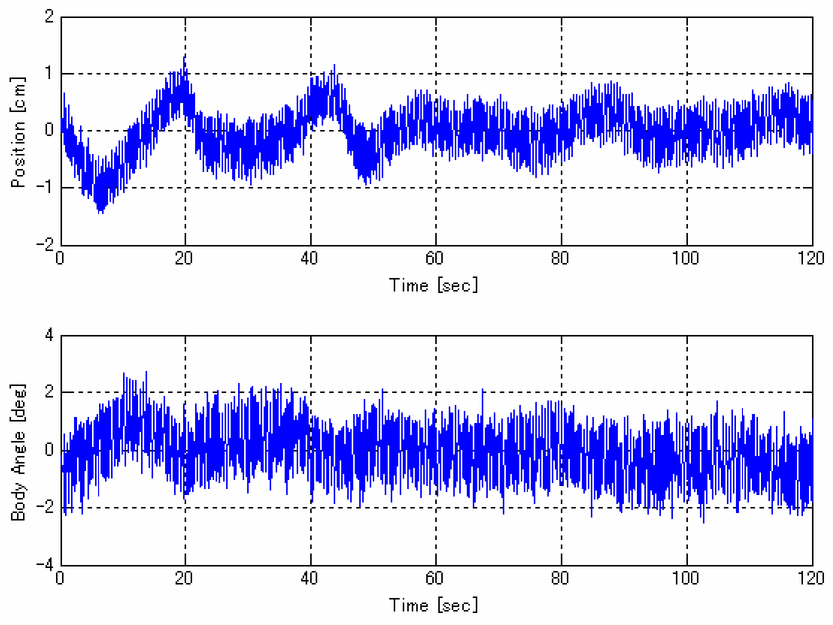


Figure 9-2 Position and body pitch angle of stationary balancing

Forward and Backward Run

Figure 9-3 shows an experimental result of forward and backward run. It is almost the same figure as Figure 8-3.

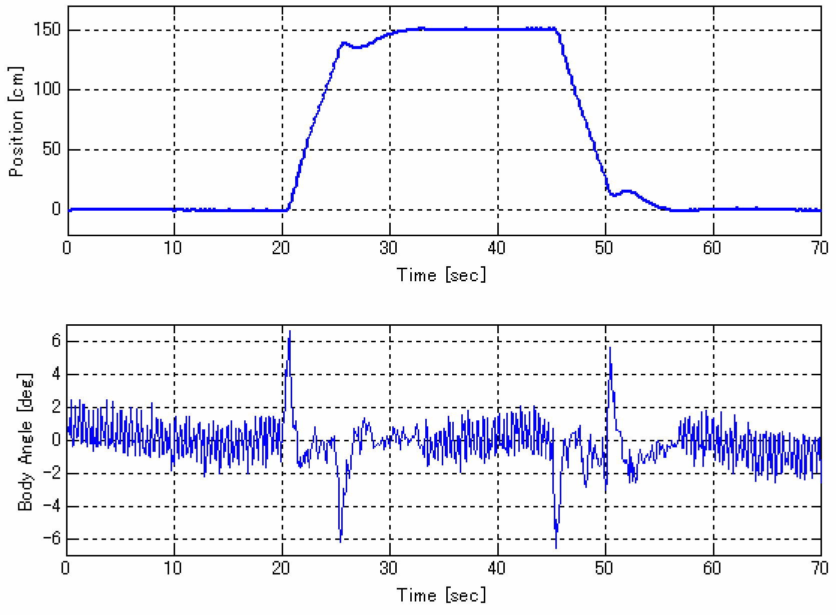
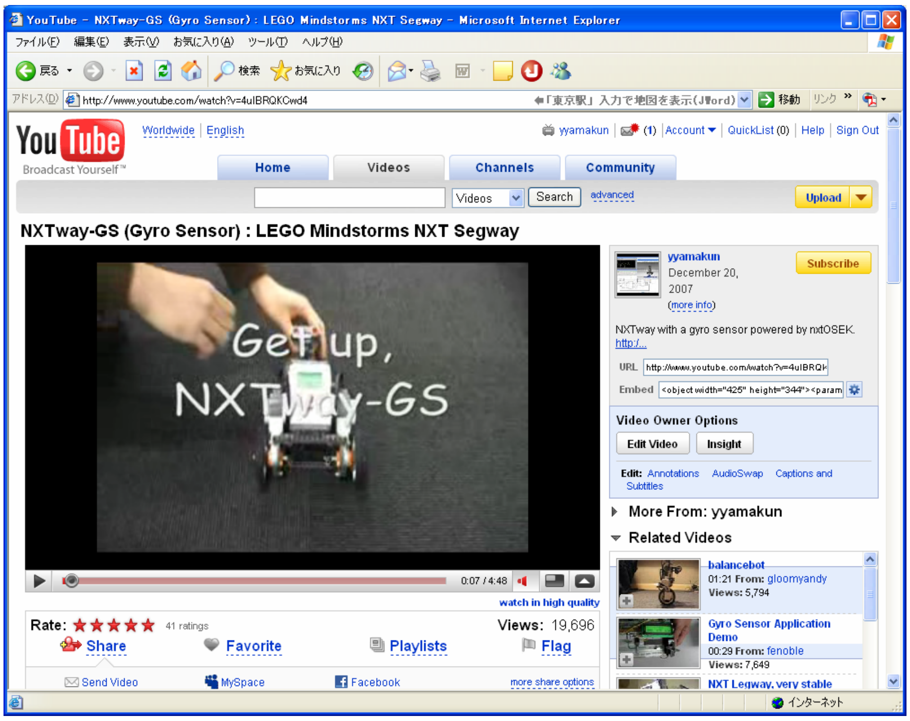


Figure 9-3 Position and body pitch angle of forward and backward run

You can watch the movie of NXTway-GS control experiment at the following URL.

http://www.youtube.com/watch?v=4ulBRQKCwd4


YouTube - NXTway-GS (Gyro Sensor) : LEGO Mindstorms NXT Segway - Microsoft Internet Explorer

http://www.youtube.com/watch?v=4ulBRQKCwd4

NXTway-GS (Gyro Sensor) : LEGO Mindstorms NXT Segway

yyamakun
December 20, 2007
(more info)

NXTway with a gyro sensor powered by nxtOSEK.
http://...

URL http://www.youtube.com/watch?v=4ulBRQK

Embed <object width="425" height="344"><param

Rate: ★★★★★ 41 ratings
Views: 19,696

balancebot
01:21 From: gloomyandy
Views: 5,794

Gyro Sensor Application Demo
00:29 From: fenoble
Views: 7,649

NXT Legway, very stable

Figure 9-4 Movie of NXTway-GS control experiment

10 Controller Model (Fixed-Point Arithmetic)

This chapter describes the fixed-point version of NXTway-GS controller model and considers a precision loss and overflow impact on the controller performance.

10.1 What is Fixed-Point Number?

Fixed-point number is a representation for approximating a real number in computing. It uses an integer data type to represent a real number, and does not use a floating-point data type. We can evaluate a fixed-point number by the following scaling equation.

$$V_R \cong V_Q = SQ + B$$

V_R : Real number V_Q : Fixed-point number Q : Integer S : Slope B : Bias

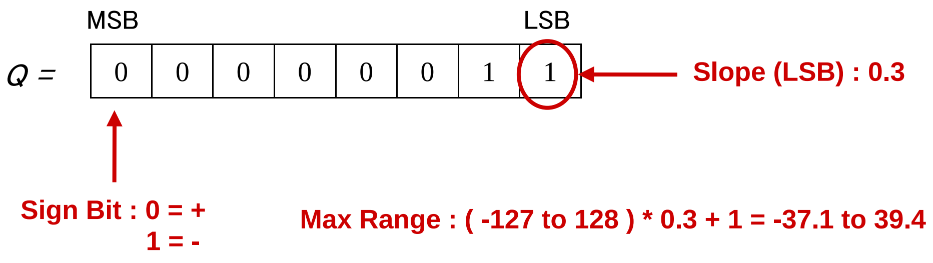
where slope represents real value per 1-bit and bias is real value when the integer value equals to zero. Slope is called as LSB or resolution and bias is called as offset conventionally.

Fixed-point number might have a rounding error because it represents a real number with finite resolution. Also max range represented by fixed-point number is restricted depending on its word length. The word length is total bit number of integer data type. Generally, fixed-point number has precision vs. max range trade-off under the condition the word length is fixed.

Example

Calculate fixed-point number corresponding to $V_R = 2$ under the condition 8-bit signed integer, $S = 0.3$, $B = 1$.

$$Q = (2-1)/0.3 \approx 3 \quad \rightarrow \quad V_Q = 0.3 \times Q + 1 = 1.9$$



Fixed-point number enables us to calculate real number on the micro processor or DSP with no FPU. Generally speaking, fixed-point arithmetic has a merit that it runs faster rather than floating-point arithmetic. On the other hand, it has a demerit that it has restricted range and overflow may occur. Table 10-1 shows the comparison with fixed-point and floating-point.

Table 10-1 Fixed-Point vs. Floating-Point

Consideration	Fixed-Point	Floating-Point
Execution Speed	Fast	Slow
Hardware Power Consumption	Low	High
RAM / ROM Consumption	Small	Large
Word Size and Scaling	Flexible	Inflexible
Max Range	Narrow	Wide
Error Prone	High	Low
Development Time	Long	Short

10.2 Floating-Point to Fixed-Point Conversion

Fixed-Point Toolbox / Simulink Fixed Point enable the intrinsic fixed-point capabilities of the MATLAB / Simulink product family. Refer the reference [3] to learn fixed-point modeling.

nxtway_gs_controller_fixpt.mdl is a fixed-point version of nxtway_gs_controller.mdl. It uses fixed-point data types for wheel angle, body pitch angle, and battery variable instead of single precision data types. The key points of fixed-point conversion are the following.

Fixed-Point Data Type

param_controller_fixpt.m defines Simulink.NumericType objects used in nxtway_gs_controller_fixpt.mdl. Simulink.NumericType specifies any data types, for example, integer, floating-point, and fixed-point.

param_controller_fixpt.m

```
% Fixed-Point Parameters
% We can calculate how far NXTway-GS can move by using the following equation
% >> double(intmax('int32')) * pi / 180 * R * S
% where R is the wheel radius and S is the slope of dt_theta (S = 2^-14 etc.)
%
%     S     :   Range [m]
%   2^-10   :   1464.1
%   2^-14   :     91.5
%   2^-18   :      5.7
%   2^-22   :      0.36

% Simulink.NumericType for wheel angle
% signed 32-bit integer, slope = 2^-14, bias = 0
dt_theta = fixdt(true, 32, 2^-14, 0);

% Simulink.NumericType for body pitch angle
% signed 32-bit integer, slope = 2^-20, bias = 0
dt_psi = fixdt(true, 32, 2^-20, 0);

% Simulink.NumericType for battery
% signed 32-bit integer, slope = 2^-17, bias = 0
dt_battery = fixdt(true, 32, 2^-17, 0);
```

Generally speaking, controller performance depends on data types used in it. There is trade-off between precision and range in using same word length illustrated in 10.1 What is Fixed-Point Number?. It is necessary for a controller designer to consider this trade-off in order to find best fixed-point settings.

In the case of NXTway-GS, the fixed-point settings for wheel angle and body pitch angle are important because NXTway-GS cannot balance if a precision loss or overflow occurs. We can estimate how far NXTway-GS can move by the following equation.

```
>> double(intmax('int32')) * pi / 180 * R * S
```

where R is the wheel radius and S is the slope of `dt_theta`. Table10-2 shows max movement distance of NXTway-GS using several slopes.

Table10-2 Max movement distance of NXTway-GS

`dt_theta` slope	2^{-10}	2^{-14}	2^{-18}	2^{-22}
Max distance [m]	1464.1	91.5	5.7	0.36

Fixed-Point Model (nxtway_gs_controller_fixpt.mdl)

nxtway_gs_controller_fixpt.mdl is a modified version of nxtway_gs_controller.mdl as follows.

- Convert single precision data type to fixed-point data types.
- Vectorize signals of same data type with a Mux block because it is impossible to vectorize signals of different data types.
- Adjust slope in PWM calculation with Data Type Conversion block.

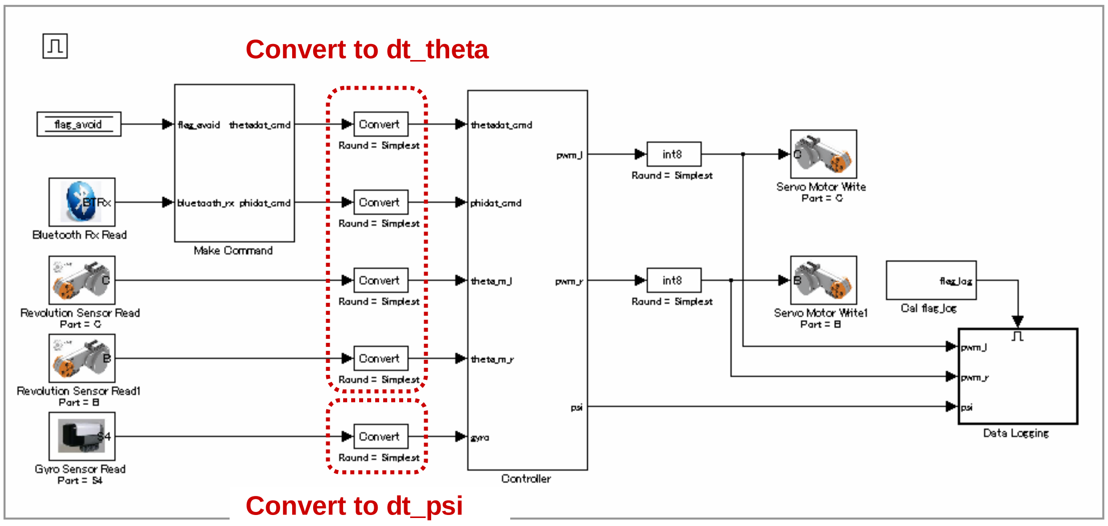


Figure 10-1 Fixed-point settings of NXTway-GS controller

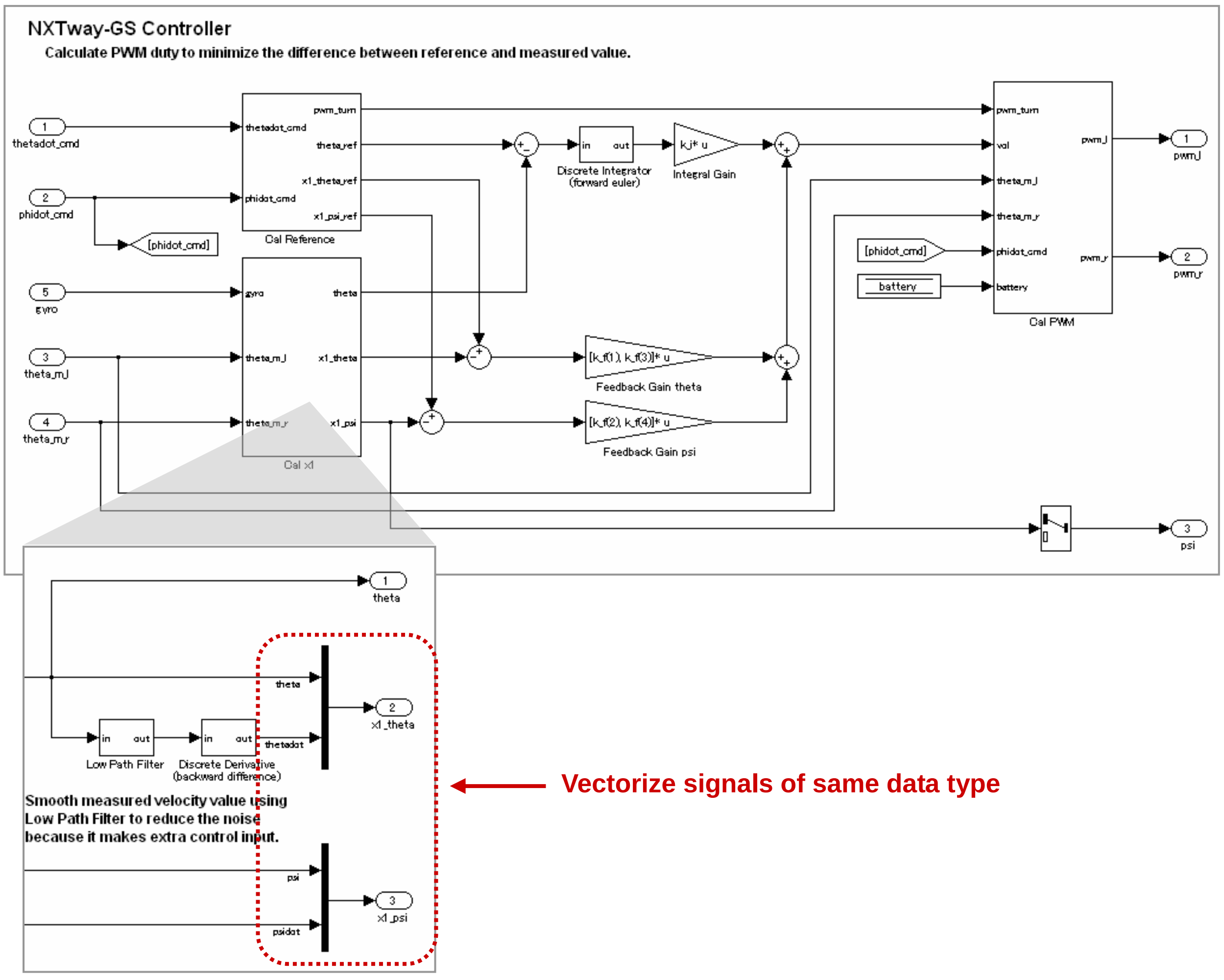

NXTway-GS Controller
Calculate PWM duty to minimize the difference between reference and measured value.
thetadot_cmd
phidot_cmd
[phidot_cmd]
Cal Reference
gyro
theta_m_l
theta_m_r
Cal x1
Discrete Integrator (forward euler)
Integral Gain
Feedback Gain theta
Feedback Gain psi
Cal PWM
battery
pwm_l
pwm_r
psi
Low Path Filter
Discrete Derivative (backward difference)
Smooth measured velocity value using Low Path Filter to reduce the noise because it makes extra control input.
x1_theta
x1_psi
Vectorize signals of same data type

Figure 10-2 Signal vectorization

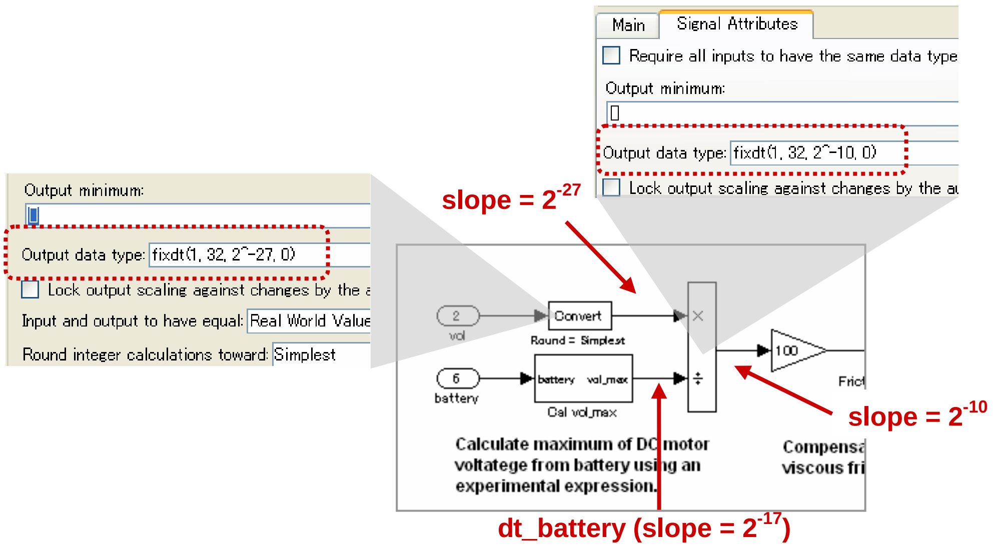

Main
Signal Attributes
Require all inputs to have the same data type
Output minimum:
Output data type: fixdt(1, 32, 2^-10, 0)
Lock output scaling against changes by the a
Output minimum:
Output data type: fixdt(1, 32, 2^-27, 0)
Lock output scaling against changes by the a
Input and output to have equal: Real World Value
Round integer calculations toward: Simplest
slope = 2^-27
slope = 2^-10
dt_battery (slope = 2^-17)
vol
battery
Convert
Round = Simplest
Cal vol_max
Calculate maximum of DC motor voltatege from battery using an experimental expression.
Compensa viscous fri

Figure 10-3 Slope adjustment in PWM calculation

10.3 Simulation Results

To simulate nxtway_gs_controller_fixpt.mdl, change referenced model of the controller in nxtway_gs.mdl / nxtway_gs_vr.mdl to this model.

To change referenced model, right click the block and chose [**Model Reference Parameter**] and edit [**Model name**].

We can see the controller performance is varied depending on a precision of fixed-point data type (slope) by using several slopes of wheel angle (`dt_theta`). Figure10-4 is a stationary balancing simulation result with different slopes of `dt_theta` (the initial value of body pitch angle is 5 [deg]). A convergence time and fluctuation are proportional to the slope. NXTway-GS cannot balance if the slope is bigger than 2^{-5} because of a precision loss.

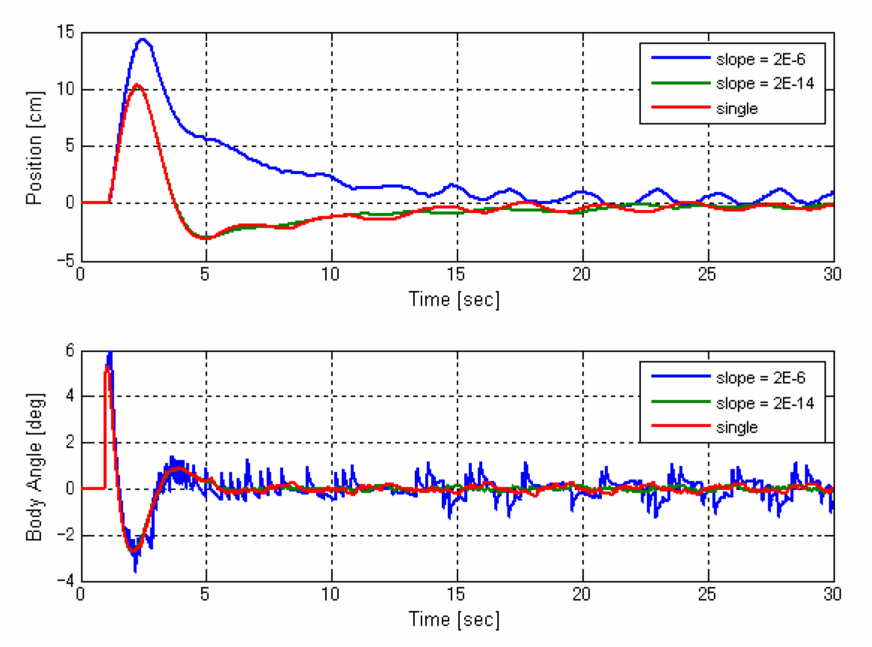
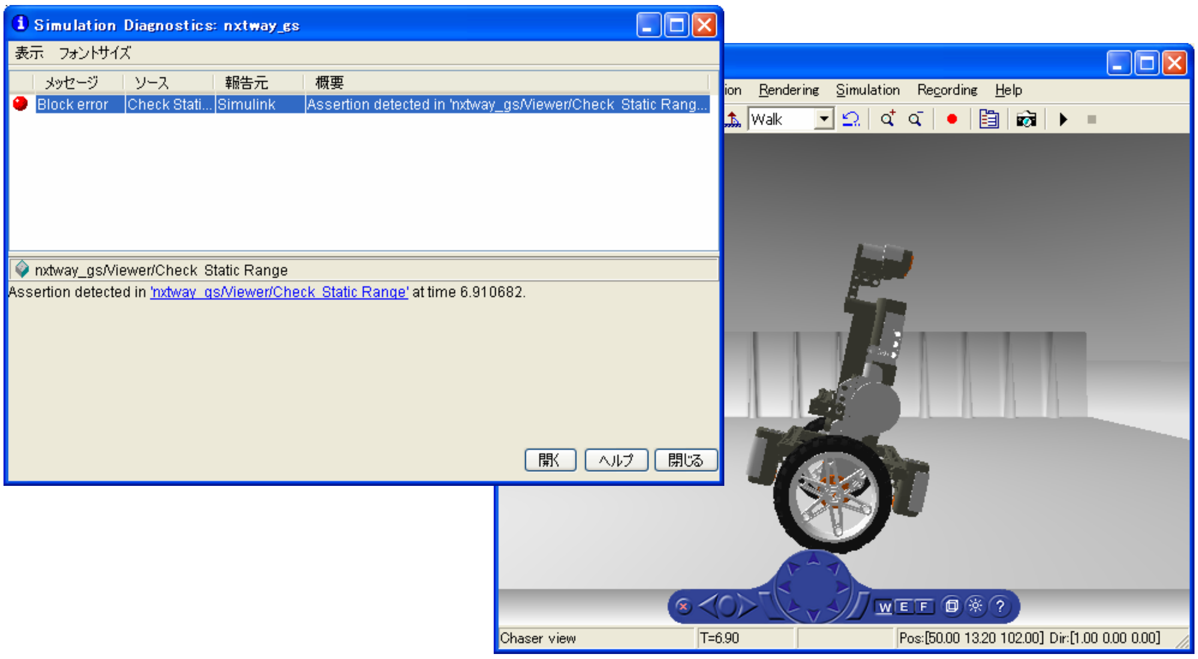


Figure10-4 Simulation result with different slopes (2E-6 = 2^{-6})

On the other hand, smaller slope means shorter max movement distance of NXTway-GS. For example, when the slope is set to 2^{-22}, NXTway-GS will fall after it runs 0.36 [m] because of overflow (refer Table10-2). Figure10-5 is a captured image when NXTway-GS falls by the overflow.


Simulation Diagnostics: nxtway_gs
表示 フォントサイズ
開く
ヘルプ
閉じる
ion Rendering Simulation Recording Help
Walk

メッセージ	ソース	報告元	概要
Block error	Check Stati...	Simulink	Assertion detected in 'nxtway_gs/Viewer/Check Static Rang...

nxtway_gs/Viewer/Check Static Range

Assertion detected in 'nxtway_gs/Viewer/Check Static Range' at time 6.910682.

Chaser view T=6.90 Pos:[50.00 13.20 102.00] Dir:[1.00 0.00 0.00]

Figure10-5 NXTway-GS falls by overflow

10.4 Code Generation & Experimental Results

The procedure of code generation and download the program from nxtway_gs_controller_fixpt.mdl is the same as nxtway_gs_controller.mdl. The generated code uses integer data types only. We can see same results as simulation (controller performance and overflow) when we download the fixed-point codes into NXT Intelligent Brick.

11 Challenges for Readers

We provide the following problems as challenges for readers. Try them if you are interested in.

- Controller performance improvement (gain tuning, robust control technique, etc.)
- Use small wheels included in LEGO Mindstorms Education NXT Base Set.
- Line tracing with a light sensor
- Proportional sound to the distance between NXTway-GS and obstacle.

Appendix A　Modern Control Theory

This appendix describes modern control theory used in controller design of NXTway-GS briefly. Refer textbooks about control theory for more details.

A.1　Stability

We define a system is asymptotically stable when the state goes to zero independent of its initial value if the input $\mathbf{u}$ equals to zero.

$$\lim_{t\to\infty} \mathbf{x}(t) = 0 \tag{A.1}$$

A state equation is given as the following

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \tag{A.2}$$

It is necessary and sufficient condition for asymptotically stable that the real part of all eigenvalues of system matrix A is negative. The system is unstable when it has some eigenvalues with positive real part.

A.2　State Feedback Control

State feedback control is a control technique that multiplies feedback gain K and deviation between reference state value $\mathbf{x}_{ref}$ and measured value $\mathbf{x}$ and feedbacks it to the system. It is similar to PD control in classical control theory. Figure A-1 shows block diagram of state feedback control.

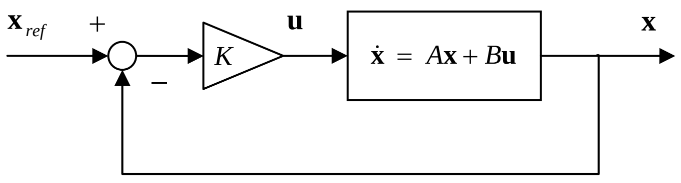


Figure A-1　Block diagram of state feedback control

The input and state equation of the system shown in Figure A-1 are given as the following

$$\mathbf{u}(t) = -K\left(\mathbf{x}(t) - \mathbf{x}_{ref}\right) \tag{A.3}$$

$$\dot{\mathbf{x}}(t) = \left(A - BK\right)\mathbf{x}(t) + BK\mathbf{x}_{ref} \tag{A.4}$$

We can make the system stable by calibrating feed back gain K because we can change the eigenvalues of the system matrix $A - BK$.

It is necessary to use state feedback control that the system is controllability. It is necessary and sufficient condition that a controllability matrix M_c is full rank. ($rank(M_c) = n$, n is order of $\mathbf{x}$)

$$M_c = \left[B, AB, \cdots A^{n-1}B\right] \tag{A.5}$$

Control System Toolbox provides `ctrb` function for evaluating a controllability matrix.

Example

Is the following system controllable? A = [0, 1; -2, -3], B = [0; 1] --> controllable

```
>> A = [0, 1; -2, -3]; B = [0; 1];
>> Mc = ctrb(A, B);
>> rank(Mc)
ans =
    2
```

There are two methods to evaluate feedback gain K.

1. Direct Pole Placement

   This method calculates feedback gain K so as to assign the poles (eigenvalues) of system matrix $A - BK$ at the desired positions. The tuning parameters are the poles and you need to decide appropriate gain value by trial and error. Control System Toolbox provides `place` function for direct pole placement.

Example

Calculate feedback gain for the system A = [0, 1; -2, -3], B = [0; 1] so as to assign the poles in [-5, -6].

```
>> A = [0, 1; -2, -3]; B = [0; 1];
>> poles = [-5, -6];
>> K = place(A, B, poles)
K =
   28.0000   8.0000
```

2. Linear Quadratic Regulator

   This method calculates feedback gain K so as to minimize the cost function J given as the following

$$J = \int_0^\infty \left(\mathbf{x}(t)^T Q\mathbf{x}(t) + \mathbf{u}(t)^T R\mathbf{u}(t)\right)dt$$

   The tuning parameters are weight matrix for state Q and for input R. You need to decide appropriate gain value by trial and error. Control System Toolbox provides `lqr` function for linear quadratic regulator.

Example

Calculate feedback gain for the system A = [0, 1; -2, -3], B = [0; 1] by using Q = [100, 0; 0, 1], R = 1.

```
>> A = [0, 1; -2, -3]; B = [0; 1];
>> Q = [100, 0; 0, 1]; R = 1;
>> K = lqr(A, B, Q, R)
K =
    8.1980    2.1377
```

A.3 Servo Control

Servo control is a control technique so that an output of a system tracks an expected behavior. PID control and I-PD control in classical control theory are a kind of servo control. It is necessary to add an integrator in the closed loop if you want the output tracks step shaped reference. Figure A-2 shows block diagram of servo control (PID type).

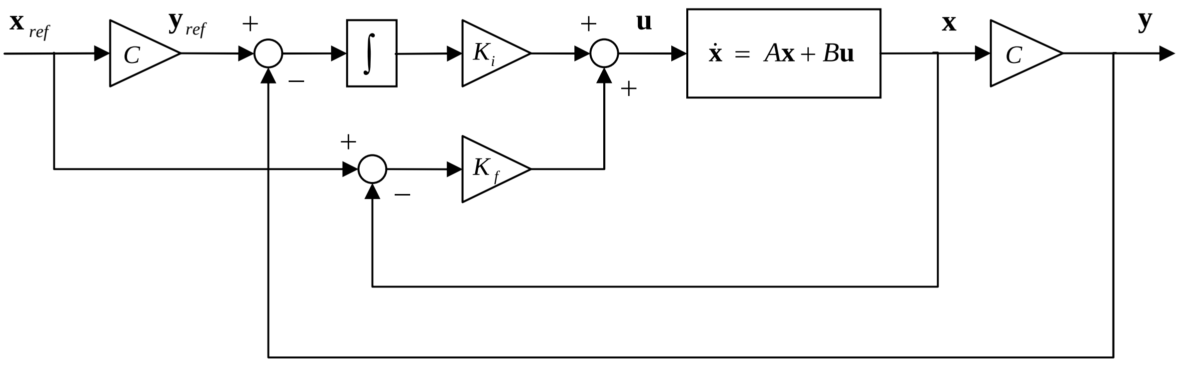


Figure A-2 Block diagram of servo control (PID type)

We can calculate servo control gains in the same way as feedback control by considering an expansion system. It has new state that is the difference between the output and referenced value. The servo control gains are derived as the following.

The state equation and output function are given as the following.

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) &= C\mathbf{x}(t) \end{aligned} \tag{A.6}$$

We use a difference $\mathbf{e}(t) = C(\mathbf{x}(t) - \mathbf{x}_{ref})$, an integral of the difference $\mathbf{z}(t)$, states of the expansion system $\bar{\mathbf{x}}(t) = [\mathbf{x}(t), \mathbf{z}(t)]^T$, and the orders of $\mathbf{x}$ and $\mathbf{u}$ are n, m. The state equation of the expansion system is the following

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{z}}(t) \end{bmatrix} = \begin{bmatrix} A & 0_{n\times m} \\ C & 0_{m\times m} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{z}(t) \end{bmatrix} + \begin{bmatrix} B \\ 0_{m\times m} \end{bmatrix} \mathbf{u}(t) - \begin{bmatrix} 0_{n\times m} \\ I_{m\times m} \end{bmatrix} C\mathbf{x}_{ref} \tag{A.7}$$

Eq. (A.7) converges to Eq. (A.8) if the expansion system is assumed to be stable.

$$\begin{bmatrix} \dot{\mathbf{x}}(\infty) \\ \dot{\mathbf{z}}(\infty) \end{bmatrix} = \begin{bmatrix} A & 0_{n\times m} \\ C & 0_{m\times m} \end{bmatrix} \begin{bmatrix} \mathbf{x}(\infty) \\ \mathbf{z}(\infty) \end{bmatrix} + \begin{bmatrix} B \\ 0_{m\times m} \end{bmatrix} \mathbf{u}(\infty) - \begin{bmatrix} 0_{n\times m} \\ I_{m\times m} \end{bmatrix} C\mathbf{x}_{ref} \tag{A.8}$$

The state equation (A.9) is derived by considering the subtraction between Eq. (A.7) and Eq. (A.8).

$$\begin{bmatrix} \dot{\mathbf{x}}_e(t) \\ \dot{\mathbf{z}}_e(t) \end{bmatrix} = \begin{bmatrix} A & 0_{n\times m} \\ C & 0_{m\times m} \end{bmatrix} \begin{bmatrix} \mathbf{x}_e(t) \\ \mathbf{z}_e(t) \end{bmatrix} + \begin{bmatrix} B \\ 0_{m\times m} \end{bmatrix} \mathbf{u}_e(t) \rightarrow \frac{d}{dt}\bar{\mathbf{x}}_e(t) = \bar{A}\bar{\mathbf{x}}_e(t) + \bar{B}\mathbf{u}_e(t) \tag{A.9}$$

where $\mathbf{x}_e = \mathbf{x}(t) - \mathbf{x}(\infty)$, $\mathbf{z}_e = \mathbf{z}(t) - \mathbf{z}(\infty)$, $\mathbf{u}_e = \mathbf{u}(t) - \mathbf{u}(\infty)$. We can use feedback control to make the expansion system stable. The input is

$$\mathbf{u}_e(t) = -K\bar{\mathbf{x}}_e(t) = -K_f\mathbf{x}_e(t) - K_i\mathbf{z}_e(t) \tag{A.10}$$

If it is possible that $\mathbf{x}(\infty) \rightarrow \mathbf{x}_{ref}$, $\mathbf{z}(\infty) \rightarrow 0$, $\mathbf{u}(\infty) \rightarrow 0$ are assumed, we derive the following input $\mathbf{u}(t)$.

$$\mathbf{u}(t) = -K_f(\mathbf{x}(t) - \mathbf{x}_{ref}) - K_iC\int(\mathbf{x}(t) - \mathbf{x}_{ref})dt \tag{A.11}$$

Modern control theory textbooks usually describe I-PD type expression ($\mathbf{x}_{ref}$ in the first term of Eq. (A.11) equals to zero) as servo control input. However we use PID type expression Eq. (A.11) because of improving the tracking performance.

Appendix B　Virtual Reality Space

This appendix describes the virtual reality space used in NXTway-GS model. For example, coordinate system, map file, distance calculation, and wall hit detection.

B.1　Coordinate System

We use Virtual Reality Toolbox for 3D visualization of NXTway-GS. Virtual Reality Toolbox visualizes an object based on VRML language. The VRML coordinate system is defined as shown in Figure B-1.

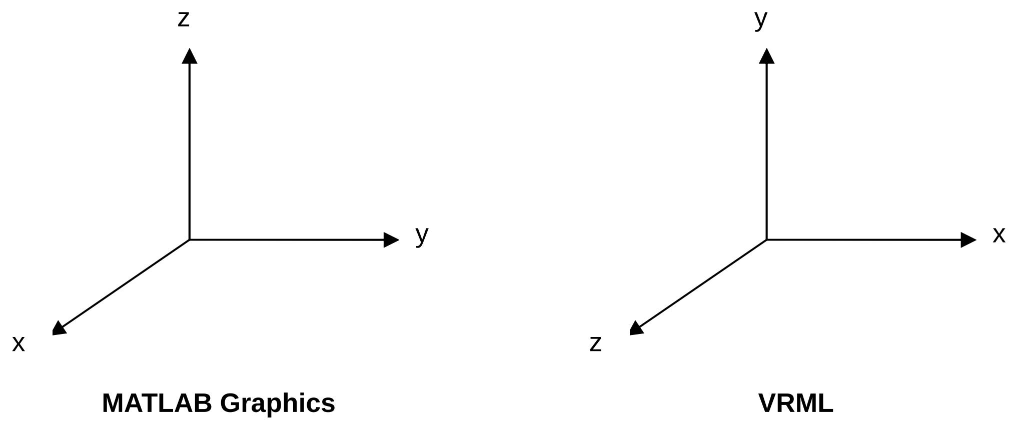


Figure B-1　MATLAB and VRML coordinate system

Figure B-2 shows the coordinate system defined by track.wrl, a map file written in VRML.

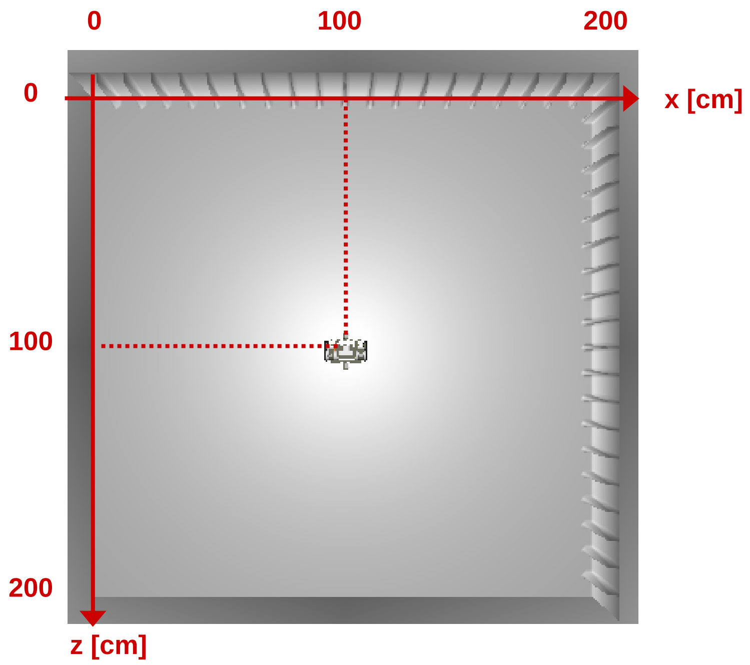


Figure B-2　The coordinate system defined by track.wrl

The VRML position of NXTway-GS is calculated in NXTway-GS / Sensor / Cal VRML Coordinate subsystem.

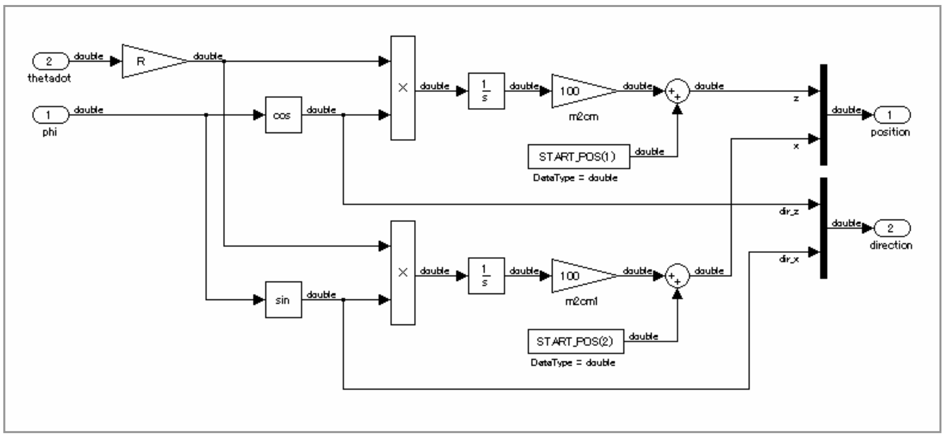


Figure B-3　Cal VRML Coordinate subsystem

B.2 Making Map File

You can make track.wrl from track.bmp by using mywritevrtrack.m. Enter the following command to make track.wrl.

```
>> mywritevrtrack('track.bmp')
```

mywritevrtrack.m makes track.wrl using the following conversion rules.

● 1 pixel	→	1 * 1 [cm^2]
● white pixel (RGB = [255, 255, 255])	→	floor
● gray pixel (RGB = [128, 128, 128])	→	wall (default height is 20 [cm])
● black pixel (RGB = [0, 0, 0])	→	black line

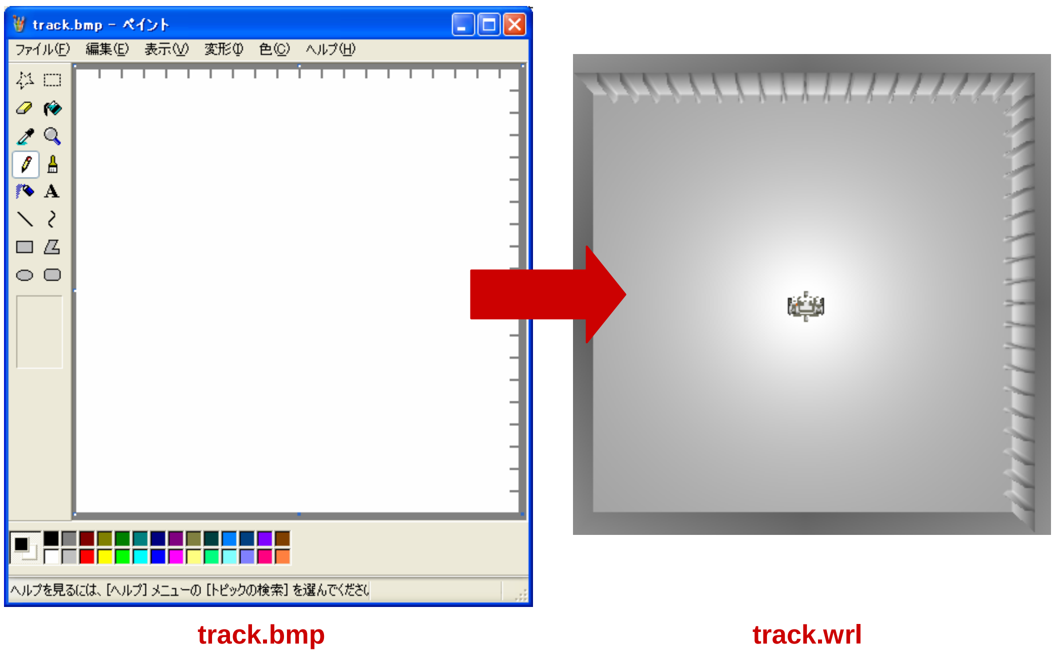
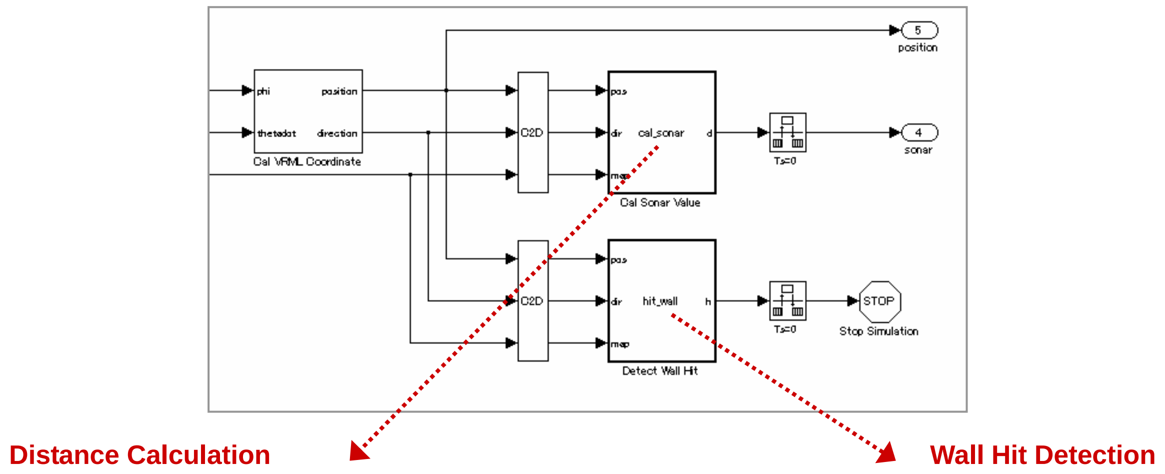
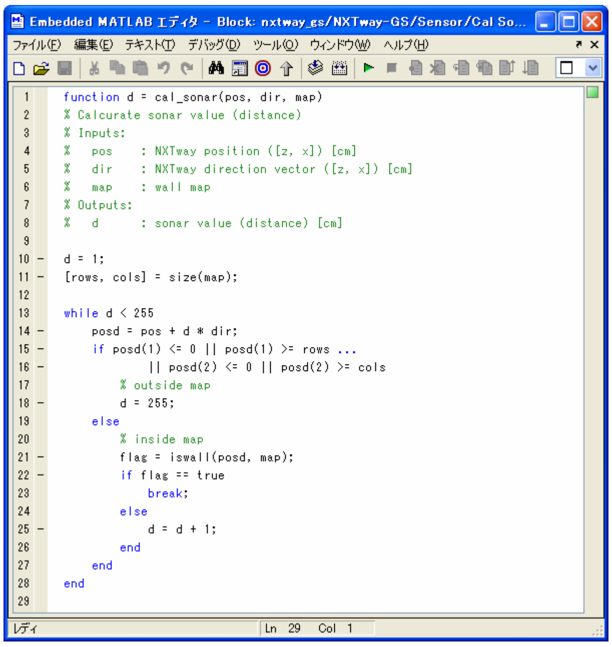

track.bmp - ペイント
ファイル(F) 編集(E) 表示(V) 変形(I) 色(C) ヘルプ(H)
ヘルプを見るには、[ヘルプ] メニューの [トピックの検索] を選んでくださ
track.bmp
track.wrl

Figure B-4 Making a map file

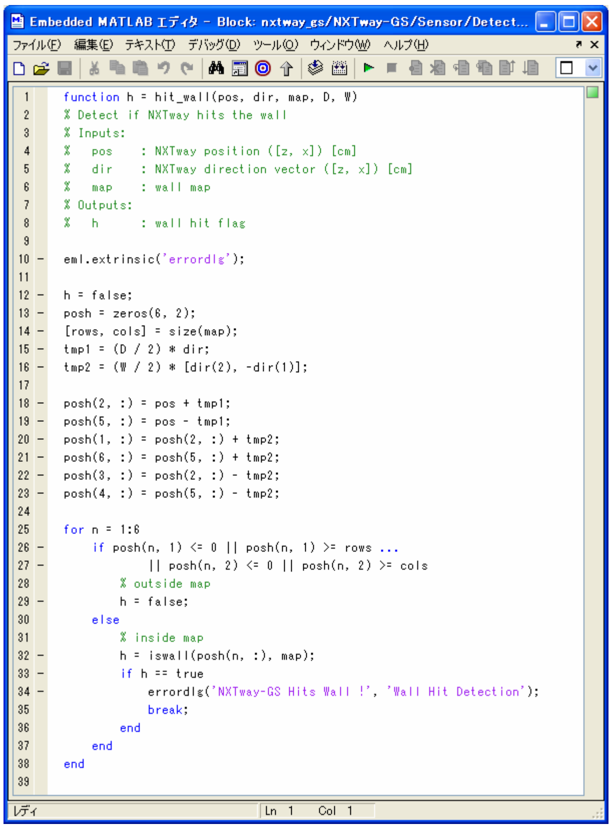
B.3 Distance Calculation and Wall Hit Detection

The distance between NXTway-GS and wall and wall hit detection are calculated with Embedded MATLAB Function blocks in Sensor subsystem.




Embedded MATLAB エディタ - Block: nxtway_gs/NXTway-GS/Sensor/Cal So...
ファイル(F) 編集(E) テキスト(T) デバッグ(D) ツール(O) ウィンドウ(W) ヘルプ(H)
レディ
Ln 29 Col 1

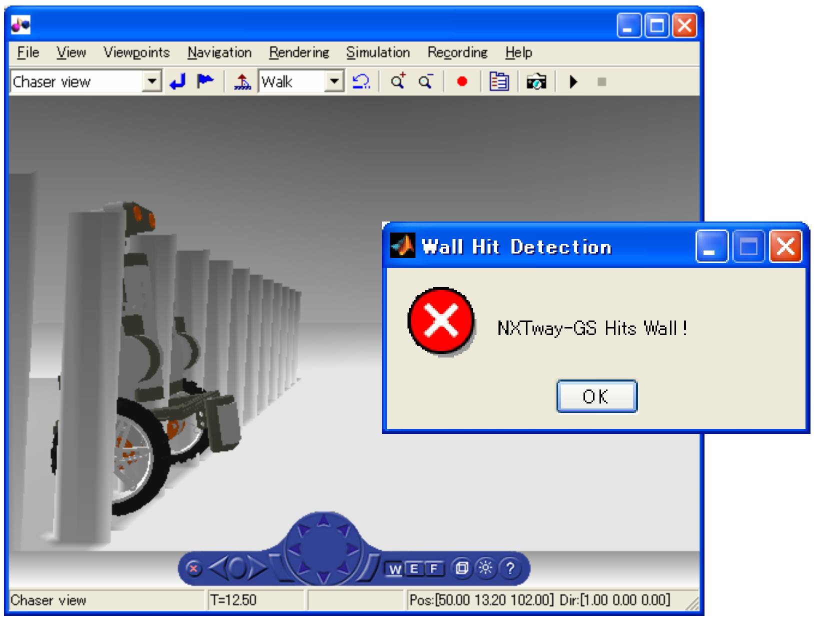
```
function d = cal_sonar(pos, dir, map)
% Calcurate sonar value (distance)
% Inputs:
%   pos    : NXTway position ([z, x]) [cm]
%   dir    : NXTway direction vector ([z, x]) [cm]
%   map    : wall map
% Outputs:
%   d      : sonar value (distance) [cm]

d = 1;
[rows, cols] = size(map);

while d < 255
    posd = pos + d * dir;
    if posd(1) <= 0 || posd(1) >= rows ...
            || posd(2) <= 0 || posd(2) >= cols
        % outside map
        d = 255;
    else
        % inside map
        flag = iswall(posd, map);
        if flag == true
            break;
        else
            d = d + 1;
        end
    end
end

```


Embedded MATLAB エディタ - Block: nxtway_gs/NXTway-GS/Sensor/Detect...
ファイル(F) 編集(E) テキスト(T) デバッグ(D) ツール(O) ウィンドウ(W) ヘルプ(H)
レディ
Ln 1 Col 1

```
function h = hit_wall(pos, dir, map, D, W)
% Detect if NXTway hits the wall
% Inputs:
%   pos    : NXTway position ([z, x]) [cm]
%   dir    : NXTway direction vector ([z, x]) [cm]
%   map    : wall map
% Outputs:
%   h      : wall hit flag

eml.extrinsic('errordlg');

h = false;
posh = zeros(6, 2);
[rows, cols] = size(map);
tmp1 = (D / 2) * dir;
tmp2 = (W / 2) * [dir(2), -dir(1)];

posh(2, :) = pos + tmp1;
posh(5, :) = pos - tmp1;
posh(1, :) = posh(2, :) + tmp2;
posh(6, :) = posh(5, :) + tmp2;
posh(3, :) = posh(2, :) - tmp2;
posh(4, :) = posh(5, :) - tmp2;

for n = 1:6
    if posh(n, 1) <= 0 || posh(n, 1) >= rows ...
            || posh(n, 2) <= 0 || posh(n, 2) >= cols
        % outside map
        h = false;
    else
        % inside map
        h = iswall(posh(n, :), map);
        if h == true
            errordlg('NXTway-GS Hits Wall !', 'Wall Hit Detection');
            break;
        end
    end
end

```

Figure B-5 Embedded MATLAB Function blocks for the distance calculation and wall hit detection

An error dialog displays when NXTway-GS hits the wall.



NXTway-GS Hits Wall !

Figure B-6　Wall hit

Appendix C Generated Code

This appendix describes a main code generated from nxtway_gs_controller.mdl. It is a default code generated by RTW-EC. RTW-EC allows us to assign user variable attributes such as variable name, storage class, modifiers etc. by using Simulink Data Object, but we does not use it here. The comments are omitted for compactness.

nxtway_app.c

```
#include "nxtway_app.h"
#include "nxtway_app_private.h"

BlockIO rtB;
D_Work rtDWork;
void task_init(void)
{
  rtDWork.battery = (real32_T)ecrobot_get_battery_voltage();
  rtDWork.flag_mode = 0U;
  rtDWork.gyro_offset = (real32_T)ecrobot_get_gyro_sensor(NXT_PORT_S4);
  rtDWork.start_time = ecrobot_get_systick_ms();
}

void task_ts1(void)
{
  real32_T rtb_UnitDelay;
  real32_T rtb_IntegralGain;
  real32_T rtb_Sum_g;
  real32_T rtb_UnitDelay_j;
  real32_T rtb_Sum_a;
  real32_T rtb_UnitDelay_l;
  real32_T rtb_DataTypeConversion4;
  real32_T rtb_theta;
  real32_T rtb_Sum2_b[4];
  real32_T rtb_Gain2_d;
  real32_T rtb_DataTypeConversion3;
  real32_T rtb_psidot;
  real32_T rtb_Sum_b;
  real32_T rtb_Saturation1;
  int16_T rtb_DataTypeConversion2_gl;
  int8_T rtb_DataTypeConversion;
  int8_T rtb_DataTypeConversion6;
  int8_T rtb_Switch;
  int8_T rtb_Switch2_e;
  int8_T rtb_DataTypeConversion_m[2];
  uint8_T rtb_UnitDelay_lv;
  boolean_T rtb_DataStoreRead;
  boolean_T rtb_DataStoreRead1;
  boolean_T rtb_Switch3_m;

  {
    int32_T i;
    rtb_DataStoreRead = rtDWork.flag_start;
    if (rtDWork.flag_start) {
      rtb_UnitDelay = rtDWork.UnitDelay_DSTATE;
      rtb_IntegralGain = -4.472135901E-001F * rtDWork.UnitDelay_DSTATE;
      rtb_UnitDelay_l = rtDWork.UnitDelay_DSTATE_j;
      rtb_DataStoreRead1 = rtDWork.flag_avoid;
      if (rtDWork.flag_mode) {
        if (rtDWork.flag_auto) {
          if (rtDWork.flag_avoid) {
            rtb_Switch = 0;
          } else {
            rtb_Switch = 100;
          }

          if (rtDWork.flag_avoid) {
            rtb_Switch2_e = 100;
          } else {
            rtb_Switch2_e = 0;
          }
        } else {
          rtb_Switch = 0;
          rtb_Switch2_e = 0;
        }
      } else {
        ecrobot_read_bt_packet(rtB.BluetoothRxRead, 32);
        for (i = 0; i < 2; i++) {
          rtb_DataTypeConversion_m[i] = (int8_T)rtB.BluetoothRxRead[i];
        }

        if (rtb_DataStoreRead1) {
          rtb_Switch = 100;
        } else {
          rtb_Switch = rtb_DataTypeConversion_m[0];
        }

        rtb_Switch = (int8_T)(-rtb_Switch);
        rtb_Switch2_e = rtb_DataTypeConversion_m[1];
      }

      rtb_Sum_g = (real32_T)rtb_Switch / 100.0F * 7.5F * 4.000000190E-003F +
        9.959999919E-001F * rtDWork.UnitDelay_DSTATE_e;
      rtb_DataTypeConversion3 = (real32_T)ecrobot_get_motor_rev(NXT_PORT_C);
      rtb_UnitDelay_j = rtDWork.UnitDelay_DSTATE_h;
      rtb_DataTypeConversion4 = (real32_T)ecrobot_get_motor_rev(NXT_PORT_B);
      rtb_theta = ((1.745329238E-002F * rtb_DataTypeConversion3 +
                  rtDWork.UnitDelay_DSTATE_h) + (1.745329238E-002F *
        rtb_DataTypeConversion4 + rtDWork.UnitDelay_DSTATE_h)) / 2.0F;
      rtb_Sum_a = 2.000000030E-001F * rtb_theta + 8.000000119E-001F *
        rtDWork.UnitDelay_DSTATE_hp;
      rtb_Sum_b = (real32_T)ecrobot_get_gyro_sensor(NXT_PORT_S4);
      rtDWork.gyro_offset = 9.990000129E-001F * rtDWork.gyro_offset +
        1.000000047E-003F * rtb_Sum_b;
      rtb_psidot = (rtb_Sum_b - rtDWork.gyro_offset) * 1.745329238E-002F;
      rtb_Sum2_b[0] = rtb_UnitDelay_l - rtb_theta;
      rtb_Sum2_b[1] = 0.0F - rtDWork.UnitDelay_DSTATE_h;
      rtb_Sum2_b[2] = rtb_Sum_g - (rtb_Sum_a - rtDWork.UnitDelay_DSTATE_m) /
        4.000000190E-003F;
      rtb_Sum2_b[3] = 0.0F - rtb_psidot;

      {
        static const int_T dims[3] = { 1, 4, 1 };

        rt_MatMultRR_Sgl((real32_T *)&rtb_Sum_b, (real32_T *)
                         &rtConstP.FeedbackGain_Gain[0],
                         (real32_T *)rtb_Sum2_b, &dims[0]);
      }

      rtb_IntegralGain += rtb_Sum_b;
      rtb_Sum_b = rtDWork.battery;
      rtb_Sum_b = rtb_IntegralGain / (1.089000027E-003F * rtb_Sum_b - 0.625F) *
        100.0F;
      rtb_Sum_b += 0.0F * rt_FSGN(rtb_Sum_b);
      rtb_IntegralGain = (real32_T)rtb_Switch2_e;
      rtb_Gain2_d = rtb_IntegralGain / 100.0F * 25.0F;
      rtb_Saturation1 = rtb_Sum_b + rtb_Gain2_d;
      rtb_Saturation1 = rt_SATURATE(rtb_Saturation1, -100.0F, 100.0F);
      rtb_DataTypeConversion = (int8_T)rtb_Saturation1;
      ecrobot_set_motor_speed(NXT_PORT_C, rtb_DataTypeConversion);
      if ((int8_T)rtb_IntegralGain != 0) {
        rtb_DataStoreRead1 = 1U;
      } else {
        rtb_DataStoreRead1 = 0U;
```

```
      }

      rtb_IntegralGain = rtb_DataTypeConversion3 - rtb_DataTypeConversion4;
      if ((!rtb_DataStoreRead1) && rtDWork.UnitDelay_DSTATE_c) {
        rtB.theta_diff = rtb_IntegralGain;
      }

      if (rtb_DataStoreRead1) {
        rtb_Saturation1 = 0.0F;
      } else {
        rtb_Saturation1 = (rtb_IntegralGain - rtB.theta_diff) *
          3.499999940E-001F;
      }

      rtb_Saturation1 += rtb_Sum_b - rtb_Gain2_d;
      rtb_Saturation1 = rt_SATURATE(rtb_Saturation1, -100.0F, 100.0F);
      rtb_DataTypeConversion6 = (int8_T)rtb_Saturation1;
      ecrobot_set_motor_speed(NXT_PORT_B, rtb_DataTypeConversion6);
      rtb_UnitDelay_lv = rtDWork.UnitDelay_DSTATE_k;
      if (rtDWork.UnitDelay_DSTATE_k != 0U) {
        rtb_Switch3_m = 0U;
      } else {
        rtb_Switch3_m = 1U;
      }

      if (rtb_Switch3_m) {
        rtb_DataTypeConversion2_gl = (int16_T)floor((real_T)(5.729578018E+001F *
          rtb_UnitDelay_j / 0.0009765625F) + 0.5);
        ecrobot_bt_adc_data_logger(rtb_DataTypeConversion,
          rtb_DataTypeConversion6, rtb_DataTypeConversion2_gl, 0, 0, 0);
      }

      rtb_UnitDelay_lv = (uint8_T)(1U + (uint32_T)rtb_UnitDelay_lv);
      if (25U == rtb_UnitDelay_lv) {
        rtDWork.UnitDelay_DSTATE_k = 0U;
      } else {
        rtDWork.UnitDelay_DSTATE_k = rtb_UnitDelay_lv;
      }

      rtDWork.UnitDelay_DSTATE = (rtb_UnitDelay_l - rtb_theta) *
        4.000000190E-003F + rtb_UnitDelay;
      rtDWork.UnitDelay_DSTATE_j = 4.000000190E-003F * rtb_Sum_g +
        rtb_UnitDelay_l;
      rtDWork.UnitDelay_DSTATE_e = rtb_Sum_g;
      rtDWork.UnitDelay_DSTATE_h = 4.000000190E-003F * rtb_psidot +
        rtb_UnitDelay_j;
      rtDWork.UnitDelay_DSTATE_hp = rtb_Sum_a;
      rtDWork.UnitDelay_DSTATE_m = rtb_Sum_a;
      rtDWork.UnitDelay_DSTATE_c = rtb_DataStoreRead1;
    }

    if (!rtb_DataStoreRead) {
      rtDWork.gyro_offset = 8.000000119E-001F * rtDWork.gyro_offset +
        2.000000030E-001F * (real32_T)ecrobot_get_gyro_sensor(NXT_PORT_S4);
    }
  }
}

void task_ts2(void)
{
  boolean_T rtb_DataStoreRead2_j;
  if (rtDWork.flag_mode) {
    rtb_DataStoreRead2_j = rtDWork.flag_avoid;
    if (!rtDWork.flag_avoid) {
      rtDWork.flag_avoid = (ecrobot_get_sonar_sensor(NXT_PORT_S2) <= 20);
      rtB.RevolutionSensor_C = ecrobot_get_motor_rev(NXT_PORT_C);
    }

    if (rtb_DataStoreRead2_j) {
      rtDWork.flag_avoid = (ecrobot_get_motor_rev(NXT_PORT_C) -
                        rtB.RevolutionSensor_C <= 210);
    }
  }

  if (!rtDWork.flag_mode) {
    rtDWork.flag_avoid = (ecrobot_get_sonar_sensor(NXT_PORT_S2) <= 20);
  }
}

void task_ts3(void)
{
  uint32_T rtb_Sum1_i;
  boolean_T rtb_RelationalOperator_e0;
  rtDWork.battery = 8.000000119E-001F * rtDWork.battery + 2.000000030E-001F *
    (real32_T)ecrobot_get_battery_voltage();
  rtb_Sum1_i = (uint32_T)((int32_T)ecrobot_get_systick_ms() - (int32_T)
    rtDWork.start_time);
  rtb_RelationalOperator_e0 = (rtb_Sum1_i >= 1000U);
  rtDWork.flag_start = rtb_RelationalOperator_e0;
  rtDWork.flag_auto = (rtb_Sum1_i >= 5000U);
  if (rtb_RelationalOperator_e0 != rtDWork.UnitDelay_DSTATE_a) {
    sound_freq(440U, 500U);
  }

  rtDWork.UnitDelay_DSTATE_a = rtb_RelationalOperator_e0;
}

void nxtway_app_initialize(void)
{
}
```

References

[1] Philo’s Home Page　LEGO Mindstorms NXT
http://www.philohome.com/

[2] Ryo’s Holiday　LEGO Mindstorms NXT
http://web.mac.com/ryo_watanabe/iWeb/Ryo%27s%20Holiday/LEGO%20Mindstorms%20NXT.html

[3] Tips for Fixed-Point Modeling and Code Generation for Simulink 6
http://www.mathworks.com/matlabcentral/fileexchange/7197