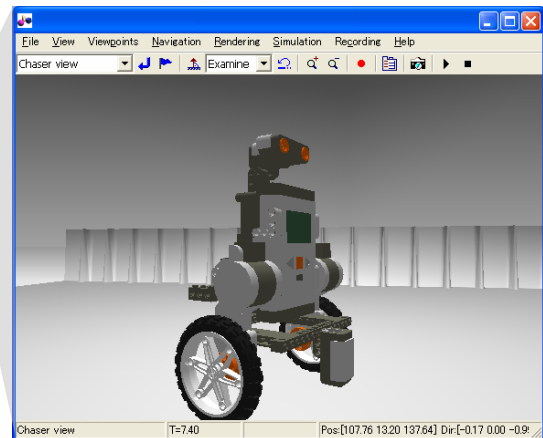
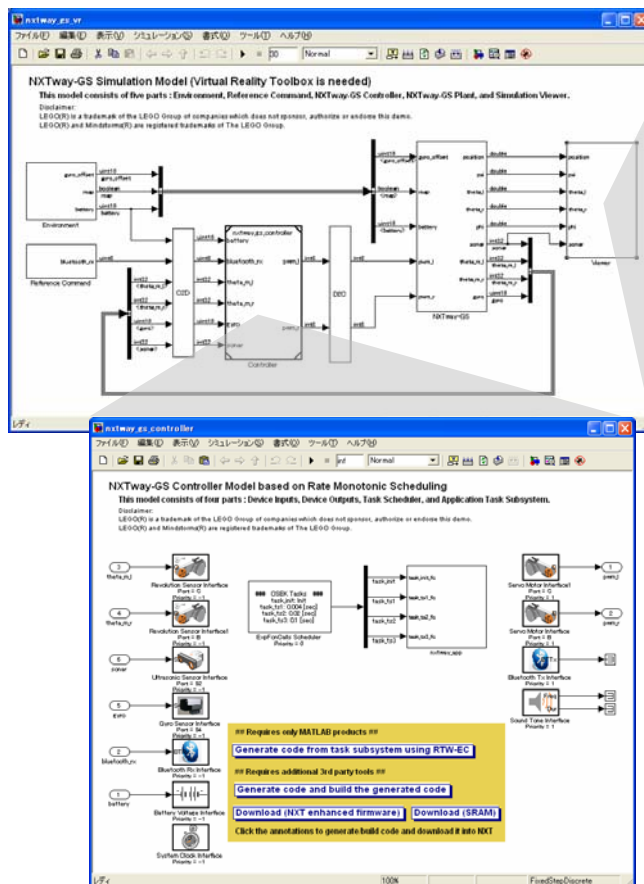


NXTway-GS のモデルベース開発 ～LEGO Mindstorms NXT を用いた 二輪型倒立振り子ロボットの制御～



つくる情熱を、支える情熱。

CYBERNET

■ 著者（初版）

サイバネットシステム株式会社

応用システム第1事業部 技術部 アドバンストサポート第1グループ

山本 順久 E-mail : y_yama@cybernet.co.jp

■ 改訂履歴

| バージョン | 年月 | 改訂内容 | 著者・編者 |
|-------|---------|---|--|
| 1.0 | 2008/02 | 初版 | 山本順久 y_yama@cybernet.co.jp |
| 1.1 | 2008/03 | 固定小数点演算モデルを追加 使用ソフトウェアのバージョンアップ | 山本順久 y_yama@cybernet.co.jp |
| 1.2 | 2008/11 | 運動方程式を一部修正 制御器モデル内アノテーション修正 使用ソフトウェアのバージョンアップ | 山本順久 y_yama@cybernet.co.jp |
| 1.3 | 2008/11 | 一般化力を修正 運動方程式・状態方程式を修正 シミュレーション動画を追加 | 山本順久 y_yama@cybernet.co.jp |
| 1.4 | 2009/05 | 図を一部修正 | 山本順久 y_yama@cybernet.co.jp |

本資料の内容・記載 URL は予告無く変更される場合があります。

はじめに

NXTway-GS は LEGO Mindstorms NXT を用いて作成した二輪型倒立振子ロボットです。本資料は MATLAB / Simulink を用いた NXTway-GS の倒立・走行制御プログラムのモデルベース開発について説明しています。主な内容は次の通りです。

- NXTway-GS の数式モデル導出
- NXTway-GS の制御器設計
- NXTway-GS モデルの解説
- シミュレーション結果および実験結果

前準備

NXTway-GSの組み立て方法についてはNXTway-GS 組み立て手順書をご覧ください。また、本資料ではモデルベース開発環境としてEmbedded Coder Robot NXTを使用しています。予めEmbedded Coder Robot NXT を下記URLからダウンロードし、Embedded Coder Robot NXT 設定手順書（Embedded Coder Robot NXT Instruction Jp.pdf）をご覧ください。必要な設定と動作確認を行ってください。

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

本資料では下記バージョンのソフトウェアを使用しています。

| ソフトウェア | バージョン番号 |
|--------------------------|---------|
| Embedded Coder Robot NXT | 3.14 |
| nxtOSEK（旧名：LEJOS OSEK） | 2.03 |
| Cygwin | 1.5.24 |
| GNU ARM | 4.0.2 |

使用製品

| 製品名 | バージョン番号 | リリース番号 |
|------------------------------------|---------|--------|
| 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 |

(N1)、(N2)の製品が無くても NXTway-GS 制御器モデルのシミュレーション・コード生成は可能です。

(N1)： 固定小数点演算モデル (nxtway_gs_controller_fixpt.mdl) を使用する際に必要です。

(N2)： 3D 表示有りモデル (nxtway_gs_vr.mdl) を使用する際に必要です。

ファイルリスト

| ファイル名 | 内容 |
|--------------------------------|--|
| iswall.m | マップ壁検出用 M-関数 |
| mywritevrtrack.m | マップ用 VRML ファイル (track.wrl) 作成用 M-関数 |
| nxtway_gs.mdl | NXTway-GS モデル (Virtual Reality Toolbox による 3D 表示無) |
| nxtway_gs_controller.mdl | NXTway-GS コントローラモデル (単精度浮動小数点演算) |
| nxtway_gs_controller_fixpt.mdl | NXTway-GS コントローラモデル (固定小数点演算) |
| nxtway_gs_plant.mdl | NXTway-GS プラントモデル |
| nxtway_gs_vr.mdl | NXTway-GS モデル (Virtual Reality Toolbox による 3D 表示有) |
| param_controller.m | 制御器 (コントローラ) パラメータ定義用 M-スクリプト |
| param_controller_fixpt.m | 固定小数点タイプ (Simulink.NumericType) 定義用 M-スクリプト |
| param_nxtway_gs.m | NXTway-GS パラメータ定義ファイル (param_***.m を実行) |
| param_plant.m | 制御対象 (プラント) パラメータ定義用 M-スクリプト |
| param_sim.m | シミュレーションパラメータ定義用 M-スクリプト |
| track.bmp | マップ用画像ファイル |
| track.wrl | マップ用 VRML ファイル |
| vrnxtwaytrack.wrl | マップ参照&NXTway-GS 用 VRML ファイル |

目次

| | |
|-------------------------------|----|
| はじめに | i |
| 前準備 | i |
| 使用製品 | ii |
| ファイルリスト | ii |
| 1 制御系モデルベース開発 | 1 |
| 1.1 モデルベース開発とは | 1 |
| 1.2 モデルベース開発プロセス | 2 |
| 1.3 モデルベース開発のメリット | 3 |
| 2 NXTway-GSの機構 | 4 |
| 2.1 構造 | 4 |
| 2.2 センサ・アクチュエータ | 4 |
| 3 NXTway-GSのモデリング | 6 |
| 3.1 二輪型倒立振子 | 6 |
| 3.2 二輪型倒立振子の運動方程式 | 7 |
| 3.3 二輪型倒立振子の状態方程式 | 10 |
| 4 NXTway-GSの制御器設計 | 12 |
| 4.1 制御系の特徴 | 12 |
| 4.2 制御器設計 | 13 |
| 5 NXTway-GSモデル | 16 |
| 5.1 モデル概要 | 16 |
| 5.2 パラメータ定義 | 20 |
| 6 プラントモデル | 21 |
| 6.1 モデル概要 | 21 |
| 6.2 アクチュエータ | 22 |
| 6.3 プラント | 23 |
| 6.4 センサ | 24 |
| 7 コントローラモデル（単精度浮動小数点演算） | 25 |
| 7.1 制御プログラム概要 | 25 |
| 7.2 モデル概要 | 27 |
| 7.3 初期化タスク : task_init | 30 |
| 7.4 4msタスク : task_ts1 | 30 |
| 7.5 20msタスク : task_ts2 | 35 |
| 7.6 100msタスク : task_ts3 | 36 |
| 7.7 チューニングパラメータ | 37 |
| 8 シミュレーション | 38 |
| 8.1 シミュレーション方法 | 38 |
| 8.2 シミュレーション結果 | 39 |

| | | |
|------------------|--------------------------|----|
| 8.3 | 3D表示 | 41 |
| 9 | コード生成と実装 | 42 |
| 9.1 | 実装環境 | 42 |
| 9.2 | コード生成・実装手順 | 43 |
| 9.3 | 実験結果 | 44 |
| 10 | コントローラモデル（固定小数点演算） | 46 |
| 10.1 | 固定小数点数とは | 46 |
| 10.2 | コントローラモデルの固定小数点化 | 47 |
| 10.3 | シミュレーション結果 | 50 |
| 10.4 | コード生成&実験結果 | 51 |
| 11 | 読者への課題 | 52 |
| 付録A | 現代制御理論 | 53 |
| A.1 | 安定性 | 53 |
| A.2 | 状態フィードバック制御 | 53 |
| A.3 | サーボ制御 | 55 |
| 付録B | 仮想空間 | 57 |
| B.1 | 座標系 | 57 |
| B.2 | マップファイルの作成 | 59 |
| B.3 | 距離算出・衝突検知 | 60 |
| 付録C | モデル生成コード | 62 |
| 参考文献 | | 66 |
| MATLABヘルプ・情報リソース | | 67 |

1 制御系モデルベース開発

制御系モデルベース開発全般について概説します。

1.1 モデルベース開発とは

モデルベース開発（Model Based Design / Development、MBD と略されます）とは、シミュレーション可能なモデルを用いるソフトウェア開発手法です。制御系 MBD では、制御器および制御対象、またはその一部をモデルで表現し、机上シミュレーション／リアルタイムシミュレーションにより制御アルゴリズムの開発・検証を行います。リアルタイムシミュレーションとは、制御系の一部を実機、その他をリアルタイムシミュレータ上で動作するモデル生成コードとし、実時間での動作検証を行うシミュレーション技術のことです。

さらに、RTW-EC 等の C コード生成ツールを用いて、制御器モデルから実際の制御器（マイコン等）に組み込む制御用 C プログラムを作成することができます。図 1-1 は MATLAB 製品を用いた制御系 MBD の概念図です。

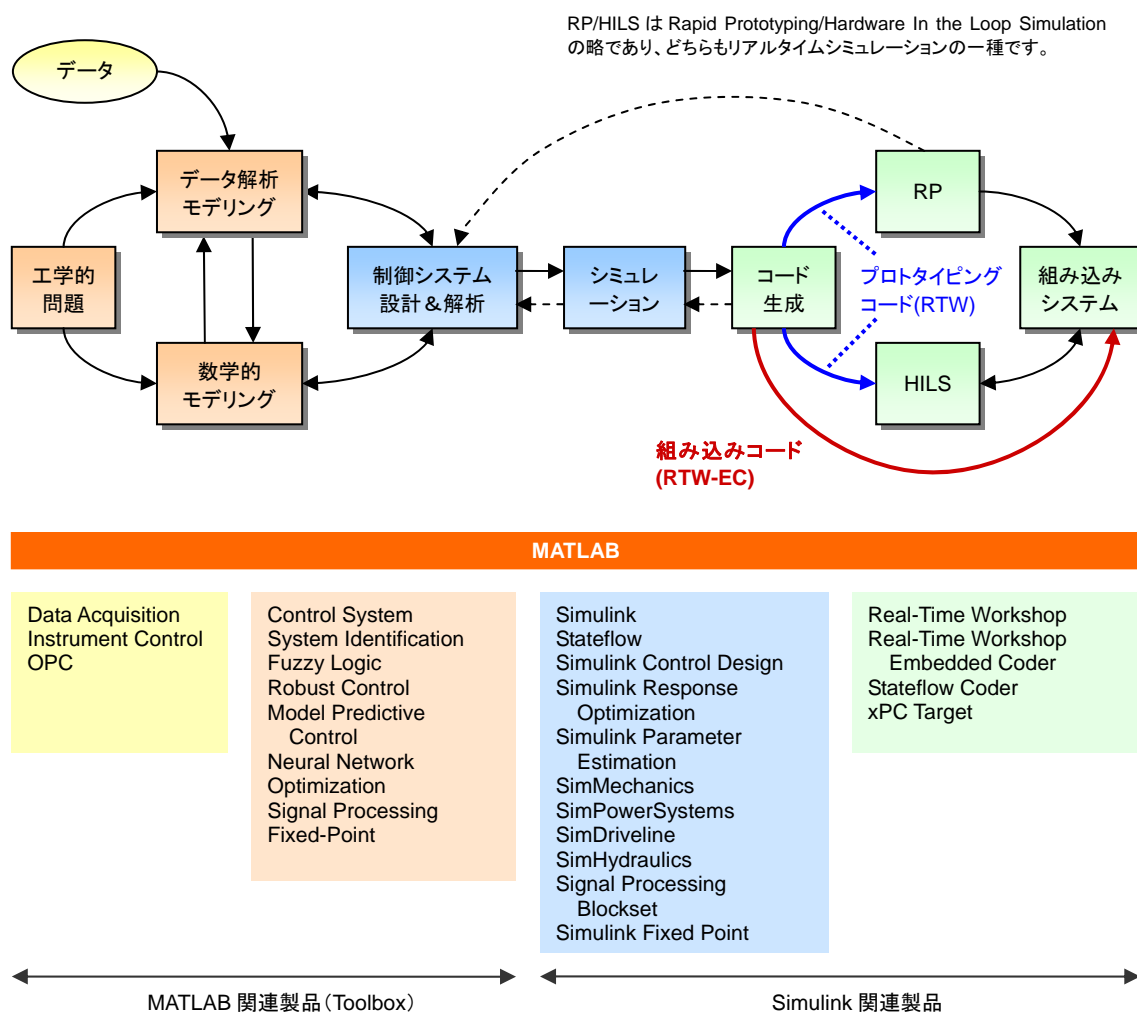


図 1-1 MATLAB 製品を用いた制御系 MBD

1.2 モデルベース開発プロセス

MBD による制御ソフトウェアの開発プロセスは、図 1-2 に示す V プロセスを用いて説明されます。V プロセスはソフトウェア開発を要求分析、各種設計、コーディングの工程に分け、各工程に検査（テスト）を対応させた V 字型の開発プロセスです。MBD では、V プロセスの左側の工程でモデリングを行い、制御仕様完成度の早期向上を図ります。また、作成したモデルを検証工程で利用することにより、コード品質・検証効率向上を図ります。

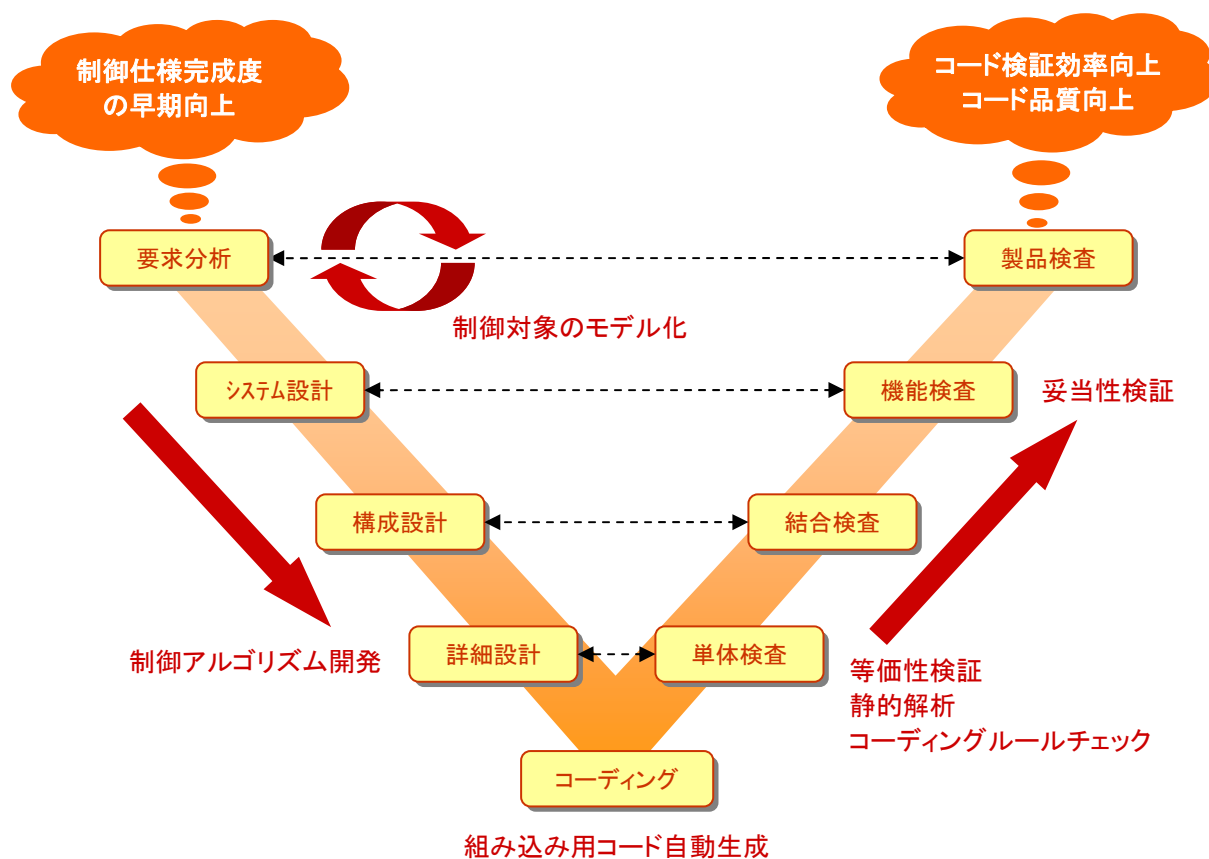


図 1-2 制御系 MBD の V プロセス

1.3 モデルベース開発のメリット

MBD には、以下のようなメリットがあります。

- 机上シミュレーションによる仕様ミスの早期検証
- リアルタイムシミュレーションによる試作工数削減・フェイルセーフ検証
- モデル検証機能によるテストの効率化
- モデル仕様の共通認識によるコミュニケーション改善
- 自動コード生成による人的コーディング工数・エラー削減

2 NXTway-GS の機構

NXTway-GS の構造およびセンサ・アクチュエータの特徴について説明します。

2.1 構造

NXTway-GS の構造を図 2-1 に示します。車体の傾斜角度を求めるために HiTechnic 社製ジャイロセンサを搭載しているのが特徴です。

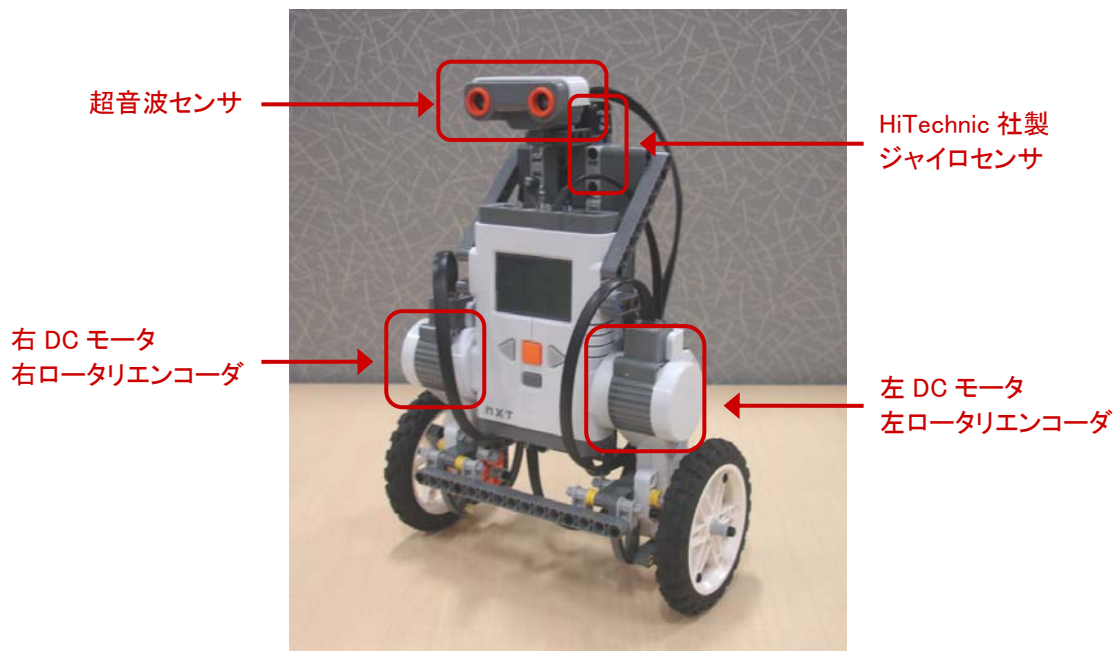


図 2-1 NXTway-GS

2.2 センサ・アクチュエータ

NXTway-GS で使用されているセンサ・アクチュエータの特性を表 2-1 および表 2-2 に示します。

表 2-1 センサの特性

| センサ | 出力値 | 単位 | データタイプ | 最大サンプル数 [1/sec] |
|-----------|-------|---------|--------|-----------------|
| ロータリエンコーダ | 回転角度 | deg | int32 | 1000 |
| 超音波センサ | 距離 | cm | int32 | 50 (注) |
| ジャイロセンサ | 傾斜角速度 | deg/sec | uint16 | 300 |

表 2-2 アクチュエータの特性

| アクチュエータ | 入力値 | 単位 | データタイプ | 最大サンプル数 [1/sec] |
|---------|-----|----|--------|-----------------|
| DC モータ | PWM | % | int8 | 500 |

(注) 超音波センサで距離を正確に測定できるサンプル数の経験的な上限値

参考文献[1]に DC モータの各種特性が紹介されています。一般に、センサ・アクチュエータには個体差があります。特に、ジャイロセンサのオフセット（ジャイロ静止時の出力値）およびドリフト（オフセットの時間変化）特性は倒立制御に与える影響が大きいのので注意が必要です。

3 NXTway-GS のモデリング

NXTway-GS を二輪型倒立振子としてモデリングし、運動方程式を導出します。

3.1 二輪型倒立振子

NXTway-GS を図 3-1 に示す二輪型倒立振子としてモデリングします。

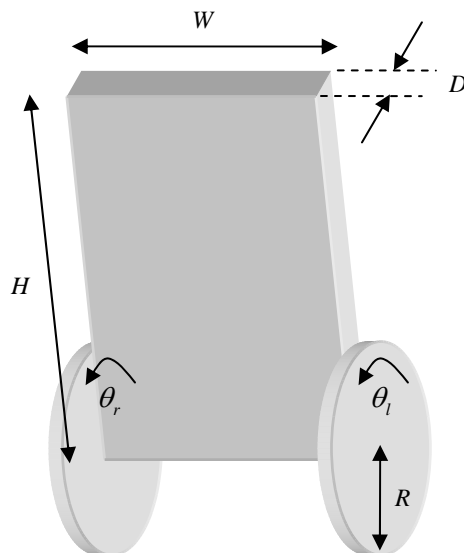
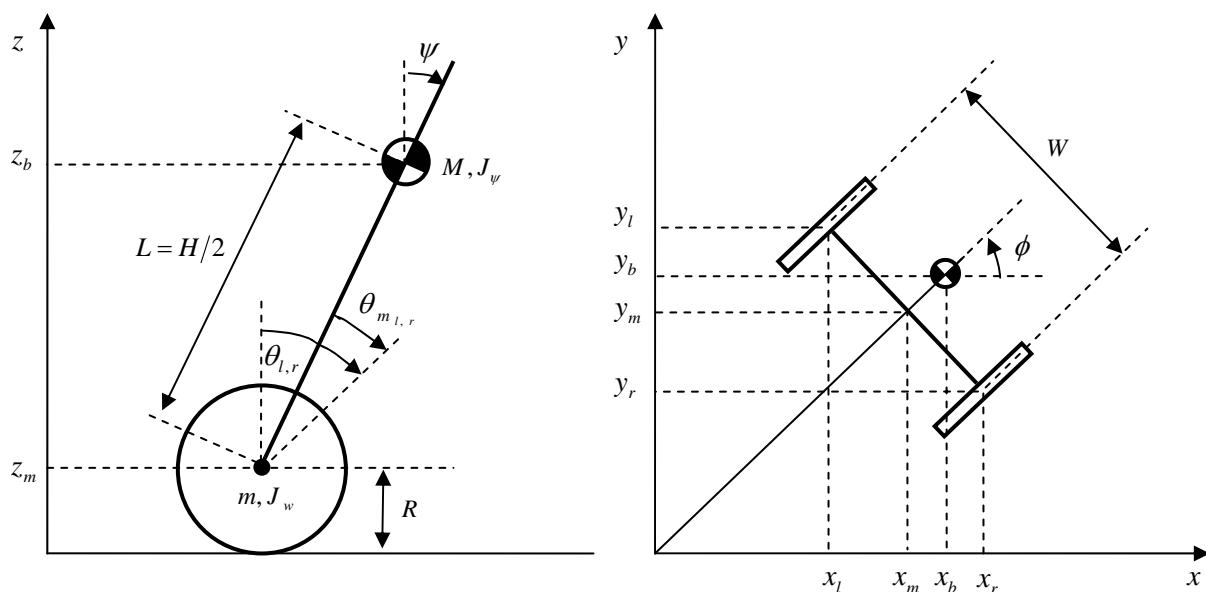


図 3-1 二輪型倒立振子

図 3-2 は二輪型倒立振子の側面図と平面図です。図中には3.2 二輪型倒立振子の運動方程式で用いる座標系を記述しています。



ψ : 本体の傾斜角度 $\theta_{l,r}$: 車輪の回転角度 (l, r は左右を表す) $\theta_{m_{l,r}}$: DC モータの回転角度

図 3-2 二輪型倒立振子の側面図と平面図

NXTway-GS の物理パラメータは次の通りです。

| | | | |
|----------------------------|------------------------------|---|------------------|
| $g = 9.81$ | $[m / \text{sec}^2]$ | : | 重力加速度 |
| $m = 0.03$ | $[kg]$ | : | 車輪質量 |
| $R = 0.04$ | $[m]$ | : | 車輪半径 |
| $J_w = mR^2/2$ | $[kgm^2]$ | : | 車輪慣性モーメント |
| $M = 0.6$ | $[kg]$ | : | 車体質量 |
| $W = 0.14$ | $[m]$ | : | 車体幅 |
| $D = 0.04$ | $[m]$ | : | 車体奥行き |
| $H = 0.144$ | $[m]$ | : | 車体高さ |
| $L = H/2$ | $[m]$ | : | 車輪中心から車体重心までの距離 |
| $J_\psi = ML^2/3$ | $[kgm^2]$ | : | 車体慣性モーメント (ピッチ) |
| $J_\phi = M(W^2 + D^2)/12$ | $[kgm^2]$ | : | 車体慣性モーメント (ヨー) |
| $J_m = 1 \times 10^{-5}$ | $[kgm^2]$ | : | DC モータ慣性モーメント |
| $R_m = 6.69$ | $[\Omega]$ | : | DC モータ抵抗 |
| $K_b = 0.468$ | $[V \text{ sec}/\text{rad}]$ | : | DC モータ逆起電力定数 |
| $K_t = 0.317$ | $[Nm/A]$ | : | DC モータトルク定数 |
| $n = 1$ | | : | ギヤレシオ |
| $f_m = 0.0022$ | | : | 車体と DC モータ間の摩擦係数 |
| $f_w = 0$ | | : | 車輪と路面間の摩擦係数 |

- R_m, K_b, K_t の値は参考文献[2]の値を使用しました。
- J_m, n, f_m, f_w の値は測定が困難なため、適当と思われる値を設定しました。

3.2 二輪型倒立振子の運動方程式

図 3-2 の座標系を用いて二輪型倒立振子の運動方程式を導出します。運動方程式の導出にはラグランジュ方程式を用います。時刻 $t=0$ で二輪型倒立振子の向きが x 軸正方向であるとする、各座標は次式で表されます。

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

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

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

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

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

並進方向の運動エネルギー T_1 、回転方向の運動エネルギー T_2 、位置エネルギー U はそれぞれ

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

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

$$U = mgz_l + mgz_r + Mgz_b \quad (3.8)$$

となります。 T_2 の第5・6項はモータ電気子の回転運動エネルギーです。ラグランジアン L は以下のように表されます。

$$L = T_1 + T_2 - U \quad (3.9)$$

以下、一般化座標として次の3変数を用いることにします。

- θ : 左右車輪の平均回転角度
- ψ : 車体の傾斜角度（ピッチ角度）
- ϕ : 車体の平面回転角度（ヨー角度）

ラグランジュ方程式は次式で与えられます。

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

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

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

(3.10)～(3.12)式の左辺を計算すると、次式となります。

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

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

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

一方、右辺の一般化力は、DC モータの駆動トルクおよび粘性摩擦を考慮すると次式となります。

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

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

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

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

ここで、 $i_{l,r}$ は DC モータに流れる電流です。

DC モータのアクチュエータは PWM（電圧）制御のため、電流制御を行うことはできません。そこで、DC モータの方程式から電流 $i_{l,r}$ と電圧 $v_{l,r}$ の関係を求めることにします。DC モータの方程式は、一般に次式で与えられます。

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

ここで、DC モータのインダクタンスが十分小さいとしてその効果を見捨てると、(3.20)式から電流は

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

となります。(3.21)式を(3.16)～(3.19)式に代入すると、一般化力を電圧を用いて表すことができます。

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

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

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

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

3.3 二輪型倒立振子の状態方程式

直立姿勢近傍で運動方程式の線形化を行い、現代制御理論における状態方程式を求めます。すなわち、車体の傾斜角度が十分小さいとして、 $\psi \rightarrow 0$ の極限をとります（ $\sin \psi \rightarrow \psi, \cos \psi \rightarrow 1, \dot{\psi}^2$ 等 2 次の項を無視します）。すると、運動方程式(3.13)～(3.15)は次式となります。

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

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

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

(3.26)式および(3.27)式は θ と ψ の連立方程式、(3.28)式は ϕ のみの式となっています。(3.26)～(3.27)式をまとめて記述すると、次式のようになります。

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

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

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

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

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

また、(3.28)式は

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

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

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

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

となります。

ここで、状態量 $\mathbf{x}_1, \mathbf{x}_2$ および入力 \mathbf{u} として、以下の変数を採用します (x^T は x の転置)。

$$\mathbf{x}_1 = [\theta, \psi, \dot{\theta}, \dot{\psi}]^T, \mathbf{x}_2 = [\phi, \dot{\phi}]^T, \mathbf{u} = [v_l, v_r]^T \quad (3.31)$$

(3.29)式および(3.30)式から、二輪型倒立振子の状態方程式を求めることができます。

$$\dot{\mathbf{x}}_1 = A_1 \mathbf{x}_1 + B_1 \mathbf{u} \quad (3.32)$$

$$\dot{\mathbf{x}}_2 = A_2 \mathbf{x}_2 + B_2 \mathbf{u} \quad (3.33)$$

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

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

$$A_1(3,2) = -gMLE(1,2)/\det(E)$$

$$A_1(4,2) = gMLE(1,1)/\det(E)$$

$$A_1(3,3) = -2[(\beta + f_w)E(2,2) + \beta E(1,2)]/\det(E)$$

$$A_1(4,3) = 2[(\beta + f_w)E(1,2) + \beta E(1,1)]/\det(E)$$

$$A_1(3,4) = 2\beta[E(2,2) + E(1,2)]/\det(E)$$

$$A_1(4,4) = -2\beta[E(1,1) + E(1,2)]/\det(E)$$

$$B_1(3) = \alpha[E(2,2) + E(1,2)]/\det(E)$$

$$B_1(4) = -\alpha[E(1,1) + E(1,2)]/\det(E)$$

$$\det(E) = E(1,1)E(2,2) - E(1,2)^2$$

4 NXTway-GS の制御器設計

現代制御理論を用いて、NXTway-GS（二輪型倒立振り子モデル）の制御器設計を行います。

4.1 制御系の特徴

NXTway-GS の制御系としての特徴は次の通りです。

入出力

(3.31)式の入力 \mathbf{u} は左右 DC モータの電圧ですが、PWM 制御を行いますので実際には PWM デューティ値が NXTway-GS の入力となります。センサからの出力値は左右車輪の回転角度 $\theta_{l,r}$ および車体の傾斜角速度 $\dot{\psi}$ です。

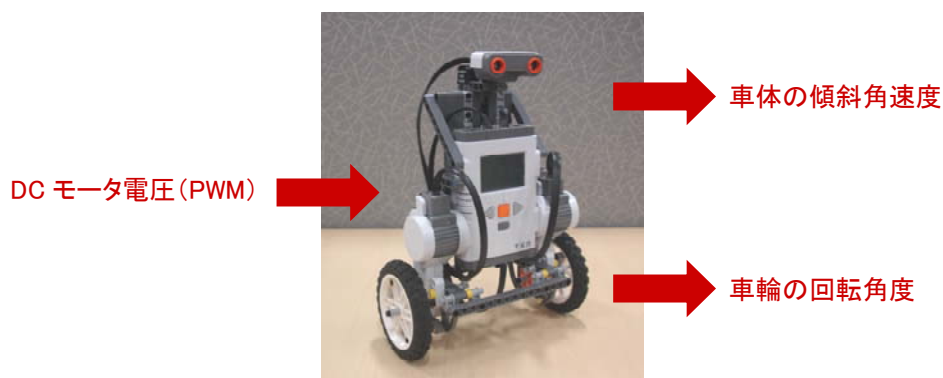


図 4-1 NXTway-GS の入出力

(3.1)式を用いて $\theta_{l,r}$ から θ, ϕ を簡単に求めることができます。 $\dot{\psi}$ から ψ を求めるには、以下の2通りの方法があります。

1. 角速度を数値積分して角度を求める。
2. 現代制御理論に基づく状態オブザーバを構成して角度を推定する。

以下の制御器設計では、方法 1. を使用します。

安定性

容易に分かるように、NXTway-GSの倒立状態は不安定です。倒立状態を安定に保つためには、車体の傾斜角度を検出してNXTway-GSが倒れないように制御する必要があります。具体的には、車体が倒れようとしたらそれと同じ方向にNXTway-GSを移動させて倒立状態を保持するように制御します。不安定な系を安定化する制御には様々な手法が提案されています（付録A参照）。

4.2 制御器設計

(3.29)式はバネ・マス・ダンパ系の一般式と同じ式構造を持っています。図 4-2 は二輪型倒立振子をバネ・マス・ダンパ系として解釈した図です。

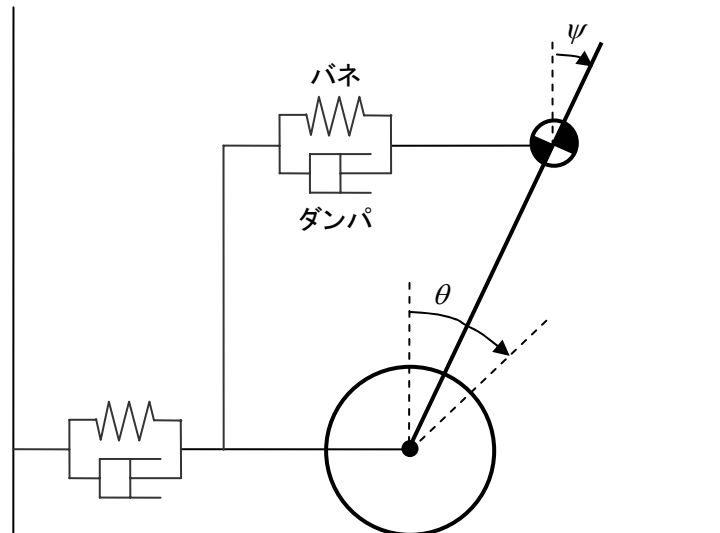
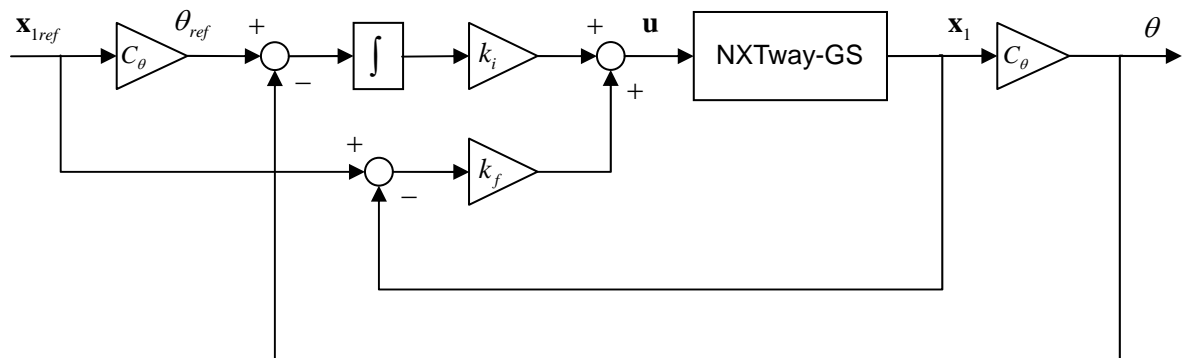


図 4-2 二輪型倒立振子の等価系

図 4-2 から、二輪型倒立振子を安定化させるためには、倒立状態を保持するようにバネのバネ定数およびダンパの減衰係数を適当に調節すればよいということが分かります。付録Aで紹介されている制御手法は、このバネ定数および減衰係数の理論的な計算方法を提供していると解釈することができます。

本資料では、NXTway-GSの倒立制御手法として付録A.3で説明されているサーボ制御を適用します。サーボ制御の目標値には左右車輪の平均回転角度 θ を選択します。 θ 以外の状態をサーボ制御の目標値にすることはできません（系が可制御でなくなるため）。図 4-3 はNXTway-GS用サーボ制御器のブロック線図です。



C_θ は \mathbf{x}_1 から θ を抽出する出力行列

図 4-3 NXTway-GS 用サーボ制御器のブロック線図

サーボ制御のゲイン計算には最適レギュレータ法を用います。実験結果から、最適レギュレータの重み行列 Q と R に以下の値を使用することにしました。

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 6 \times 10^5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 \times 10^2 \end{bmatrix}, R = \begin{bmatrix} 1 \times 10^3 & 0 \\ 0 & 1 \times 10^3 \end{bmatrix}$$

$Q(2,2)$ は車体の傾斜角度に対する重み、 $Q(5,5)$ は車輪の平均角度とその目標値の偏差の時間積分に対する重みになっています。

最適レギュレータのゲイン計算および各種パラメータ設定は `param_controller.m` で行われています。以下に `param_controller.m` のゲイン計算部分を抜粋します。

```
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;
```

ゲインの計算結果は次の通りです。

$$k_f = [-0.8351, -34.1896, -1.0995, -2.8141]$$

$$k_i = -0.4472$$

速度ゲイン $k_f(3)$ が大きいと倒立状態を維持するための揺動が大きくなるため、`param_controller.m` では速度ゲインの計算値を適当に調節しています。

NXTway-GS に実装する制御器モデルでは、サーボ制御器に加えて以下の走行制御を行っています。

- 左右 DC モータに与える PWM 値に差を設けて NXTway-GS を回転させる。
- 前進時に左右車輪の回転角度を同期させるように P 制御を行う（左右 DC モータの特性が異なる場合、同じ PWM 値を与えても回転角度が等しくならないため）。

図 4-4 は上記処理を追加した NXTway-GS 制御器のブロック線図です。ゲイン $k_{\dot{\theta}}$ 、 $k_{\dot{\phi}}$ 、 k_{sync} はチューニングパラメータです。

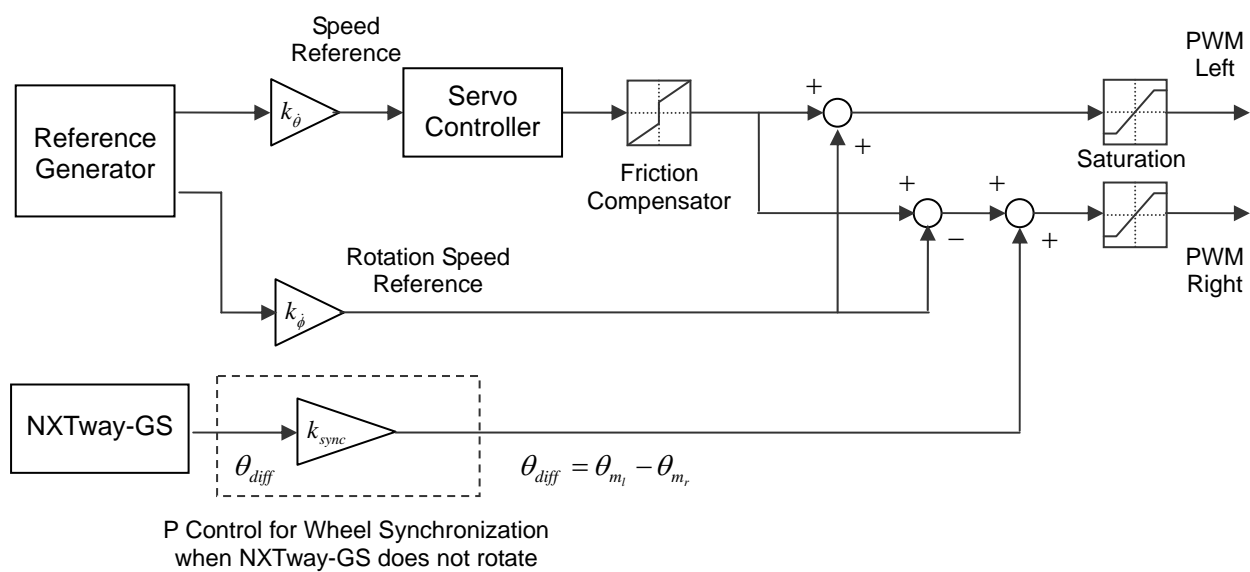


図 4-4 NXTway-GS 制御器のブロック線図

5 NXTway-GS モデル

NXTway-GS モデルの概要およびパラメータ定義ファイルについて説明します。

5.1 モデル概要

nxtway_gs.mdl および nxtway_gs_vr.mdl は NXTway-GS の制御系全体を表したモデルです。両モデルは基本的に同じであり、Virtual Reality Toolbox による 3D 表示を含むかどうかのみが異なります。

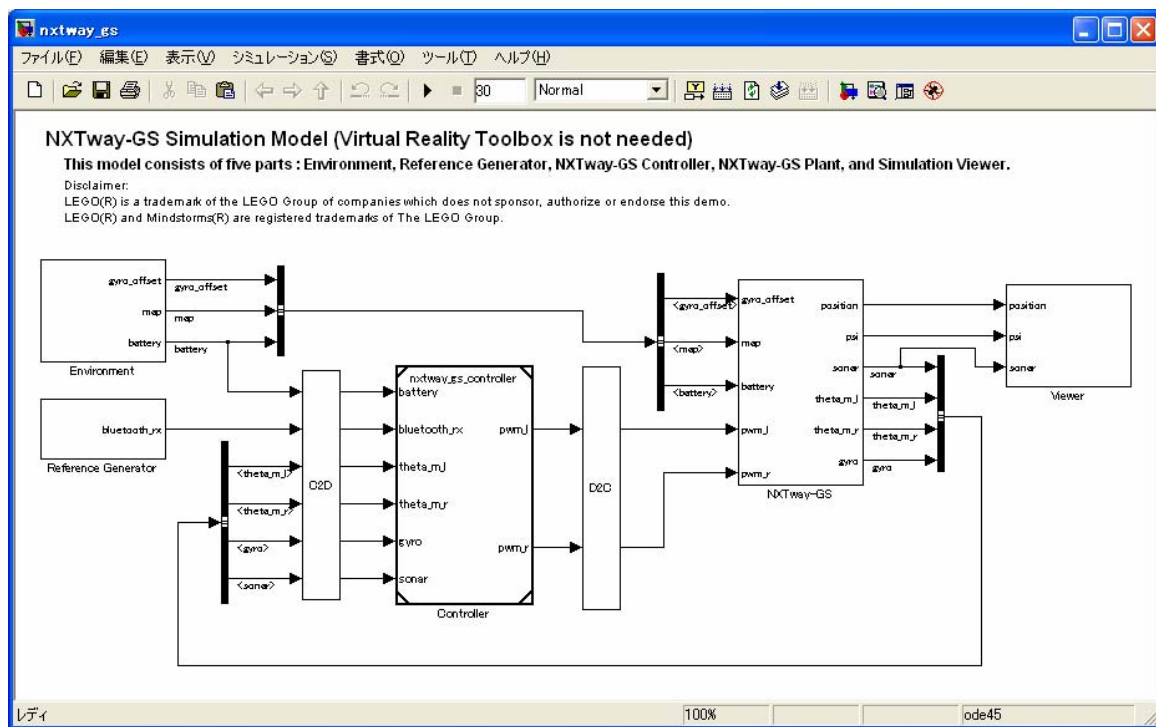


図 5-1 nxtway_gs.mdl

nxtway_gs.mdl および nxtway_gs_vr.mdl の主な構成要素は次の通りです。

Environment

マップデータやジャイロのオフセット値等、外部・内部環境の設定を行っています。

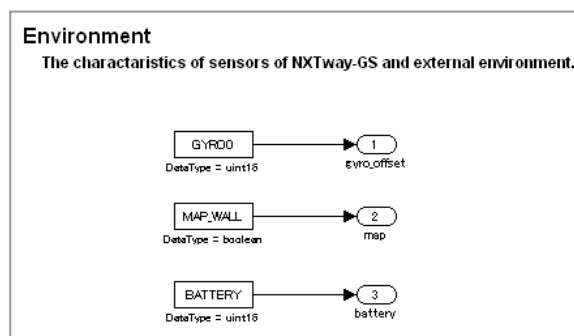


図 5-2 Environment サブシステム

Reference Generator

NXTway-GS に対する目標信号発生器です。Signal Builder ブロックを使用して速度・回転速度の目標値を設定できるようになっています。出力信号は NXT GamePad の仕様に合うようにダミーデータを付け加えた 32 バイトのデータとなります。

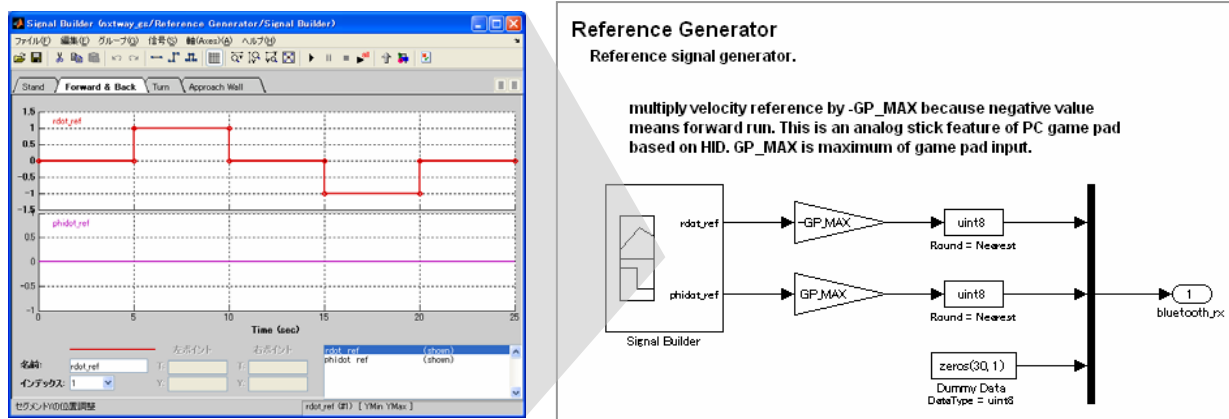
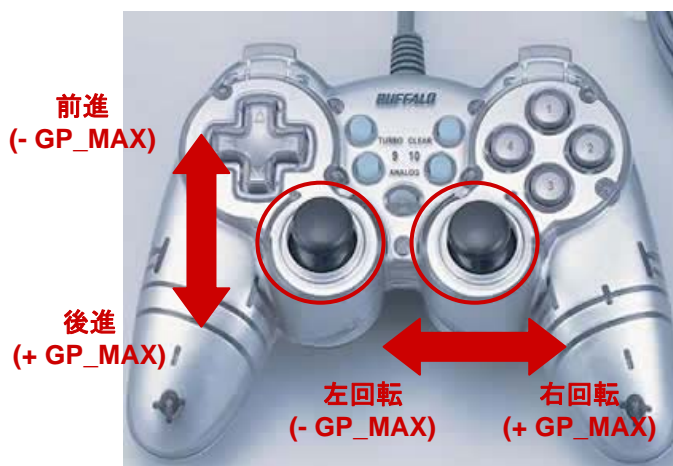


図 5-3 Reference Generator サブシステム

図 5-4 は速度・回転速度の目標値とアナログスティックからの入力値の関係を示した図です。



GP_MAX はゲームパッド入力の最大値
(本モデルでは GP_MAX = 100 を使用)

図 5-4 アナログスティック入力値

Controller

NXTway-GS の制御器（コントローラ）です。モデルリファレンス機能を使用して `nxtway_gs_controller.mdl` を参照しています。同モデルの詳細については7 コントローラモデルを参照してください。

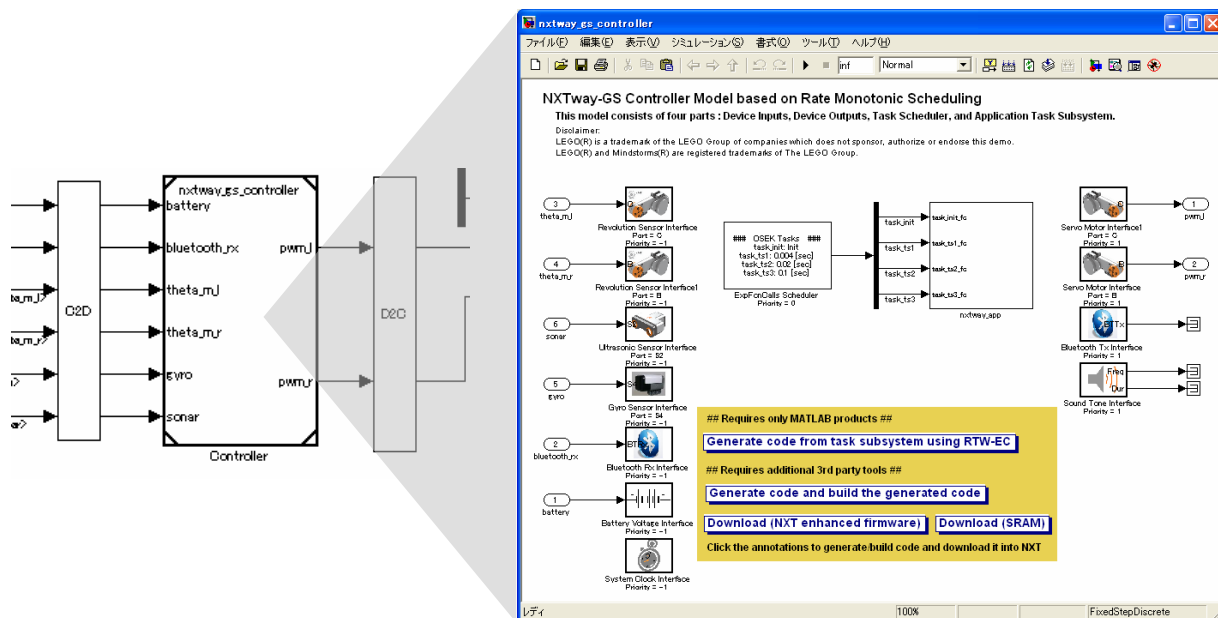


図 5-5 Controller ブロック (`nxtway_gs_controller.mdl`)

Controller ブロックは離散時間（ベースサンプル時間： $TS = 1$ [ms]）、NXTway-GS サブシステムは連続時間（サンプル時間：0 [s]）で動作します。連続系と離散系が混在しているため、両者の間に Rate Transition ブロックを挿入して連続・離散変換を行っています。

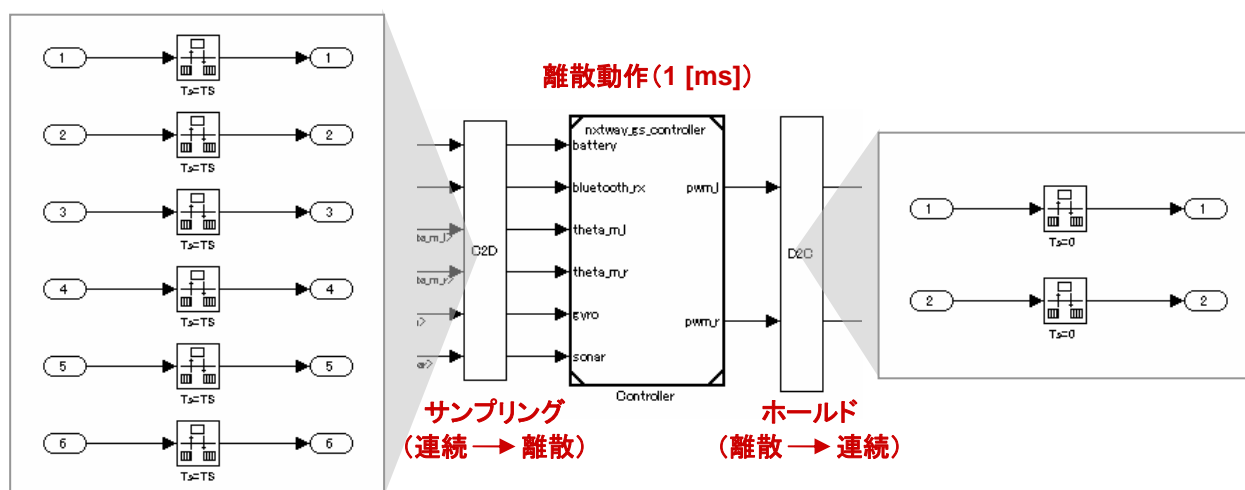


図 5-6 入出力信号の連続・離散変換

NXTway-GS

NXTway-GSの数式モデルです。センサ・アクチュエータ・線形プラントモデルから構成されています。線形プラントモデルはモデルリファレンス機能を使用してnxtway_gs_plant.mdlを参照しています。詳細については6 プラントモデルを参照してください。

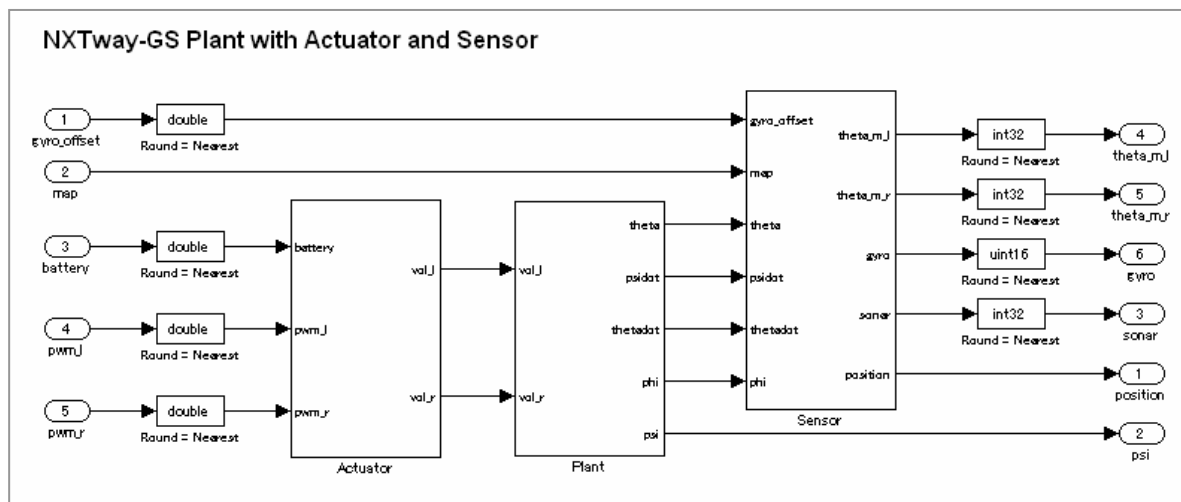


図 5-7 NXTway-GS サブシステム

Viewer

シミュレーション結果表示部です。nxtway_gs.mdl では XY Graph ブロックによる位置表示、nxtway_gs_vr.mdl では Virtual Reality Toolbox による 3D 表示を行うことができます。

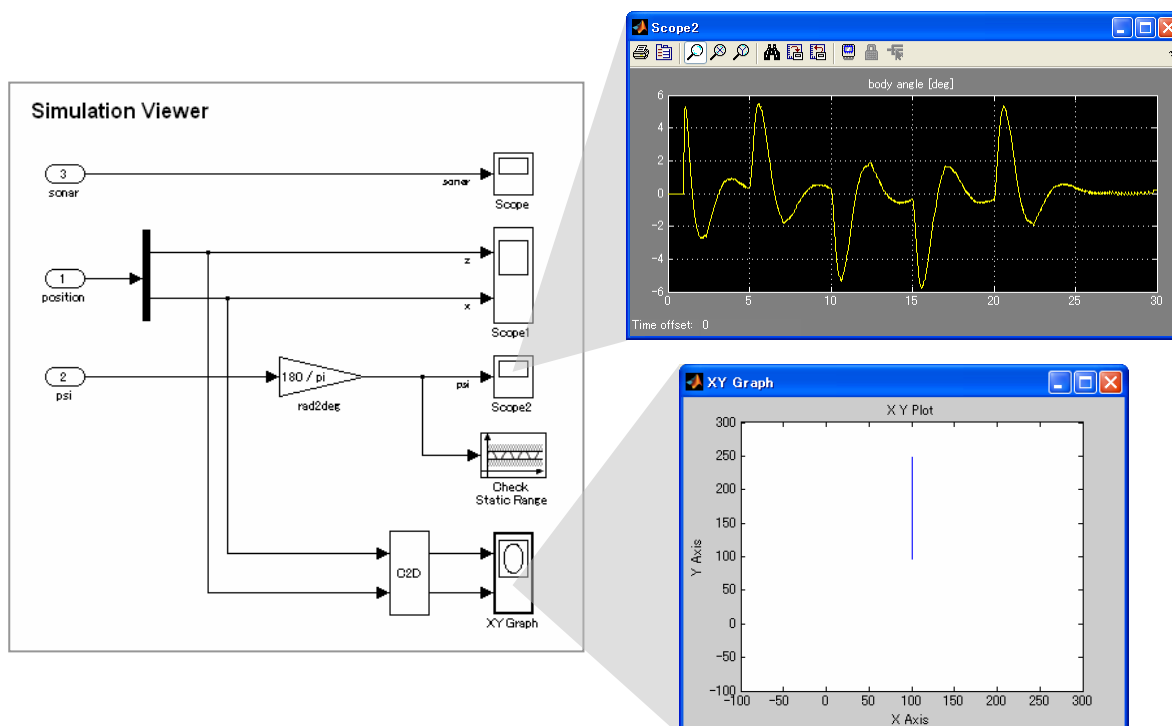


図 5-8 Viewer サブシステム (nxtway_gs.mdl)

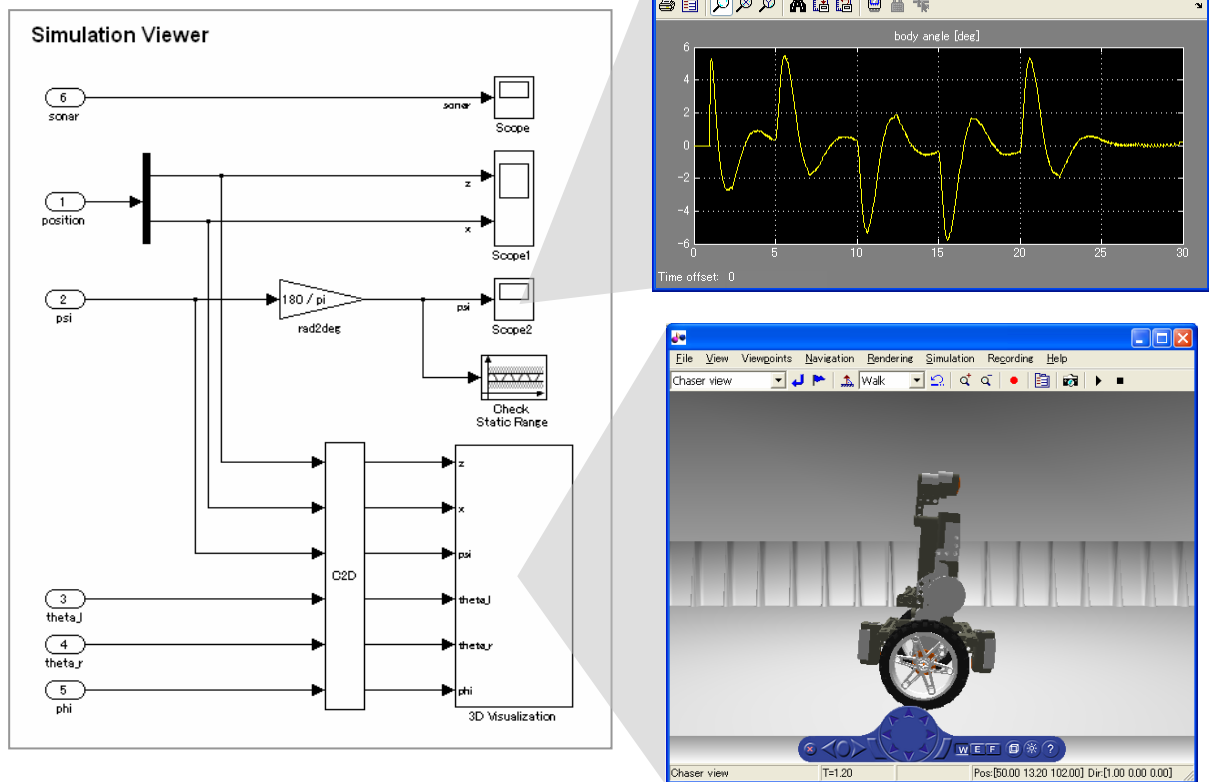


図 5-9 Viewer サブシステム (nxtway_gs_vr.mdl)

5.2 パラメータ定義

表 5-1 にシミュレーション・コード生成に必要なパラメータを定義する M-ファイルを示します。

表 5-1 パラメータ定義ファイル

| ファイル名 | 内容 |
|--------------------------|---|
| param_controller.m | 制御器（コントローラ）パラメータ定義用 M-スクリプト |
| param_controller_fixpt.m | 固定小数点タイプ（Simulink.NumericType）定義用 M-スクリプト |
| param_nxtway_gs.m | NXTway-GS パラメータ定義ファイル（param_***.m を実行） |
| param_plant.m | 制御対象（プラント）パラメータ定義用 M-スクリプト |
| param_sim.m | シミュレーションパラメータ定義用 M-スクリプト |

param_nxtway_gs.m を実行すると、param_***.m（***は controller, plant, sim）が呼び出され、各パラメータがワークスペース上に定義されます。デモモデルでは、モデル起動時に param_nxtway_gs.m を自動実行するようにモデルコールバック関数を設定しています。

モデルコールバック関数設定を確認するには、モデルウィンドウ内で右クリックして表示されるコンテキストメニューから [モデルプロパティ] を選択後、[コールバック] を選択してください。

6 プラントモデル

制御対象（プラント）モデルである NXTWay-GS サブシステムについて説明します。

6.1 モデル概要

NXTWay-GS サブシステムはセンサ・アクチュエータ・線形プラントモデルから構成されています。制御器（コントローラ）からの入力信号のデータタイプを `double` に変換してから倍精度浮動小数点演算を行い、その結果を適当に量子化してからコントローラへ出力しています。

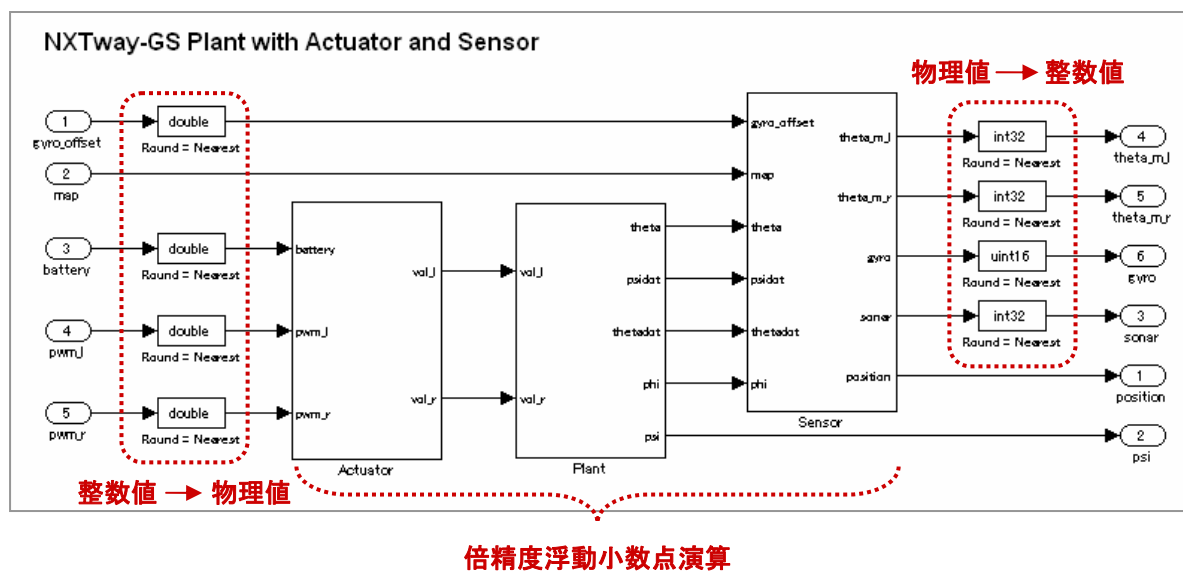


図 6-1 NXTWay-GS サブシステム

6.2 アクチュエータ

コントローラで計算された PWM デューティ値から DC モータの印加電圧を求めます。Actuator サブシステムでは、電圧を計算する前に駆動系のクーロン摩擦と粘性摩擦を考慮した不感帯を設けています。

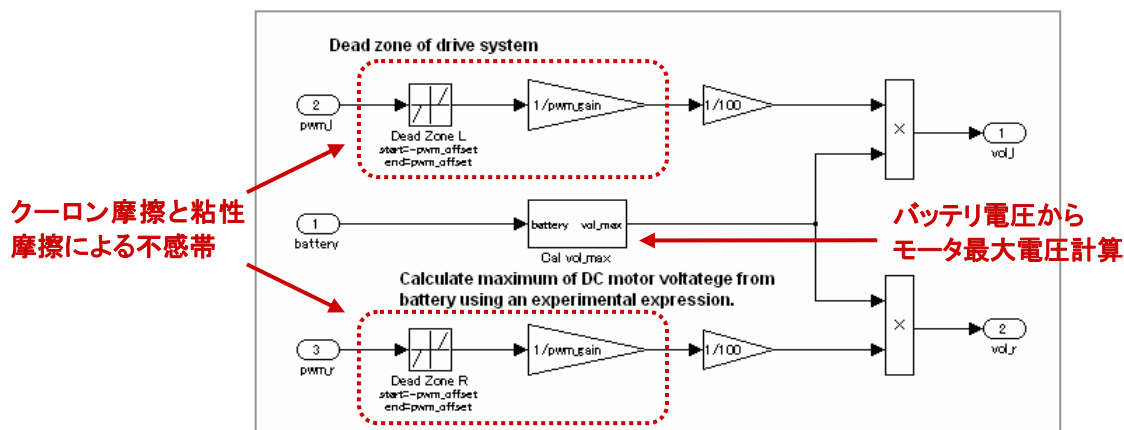


図 6-2 Actuator サブシステム

PWM デューティ値から DC モータの印加電圧を求めるには、モータの最大電圧が必要になります。Cal vol_max サブシステムでは、実験的に得られた次式を用いてバッテリー電圧 V_B からモータ最大電圧 V_{\max} を求めています。

$$V_{\max} = 0.001089 \times V_B - 0.625 \quad (6.1)$$

(6.1)式の導出方法は次の通りです。DC モータの印加電圧および回転速度はバッテリー電圧に比例します。図 6-3 は無負荷状態で PWM デューティ値を 100%にした時のバッテリー電圧とモータ回転速度の実測値です。

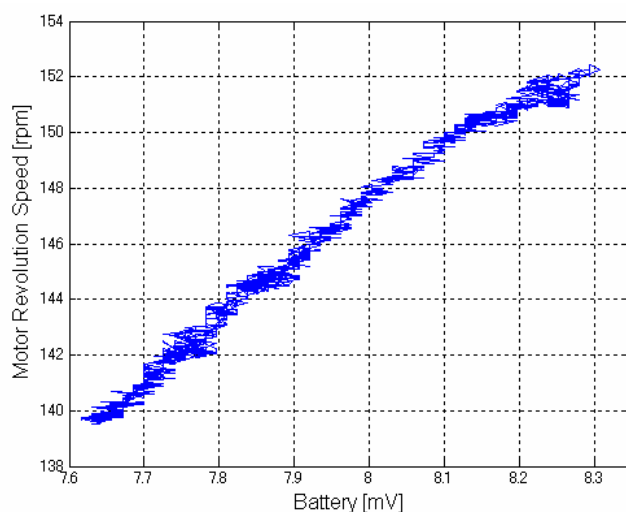
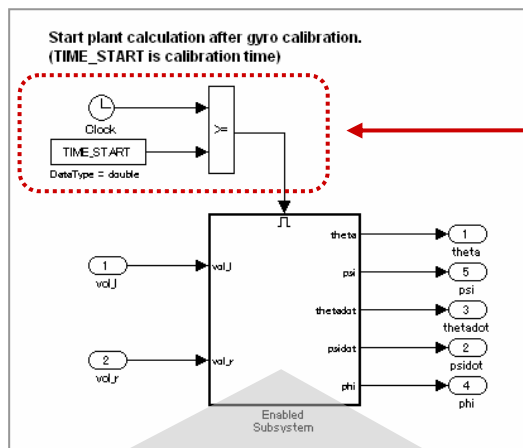


図 6-3 バッテリー電圧とモータ回転速度の実測値 (PWM=100%)

図 6-3 と参考文献[1]のデータを組み合わせて (6.1)式を得ました。

6.3 プラント

nxtway_gs_plant.mdlは3.3 二輪型倒立振子の状態方程式で求めた状態方程式をモデリングしたモデルです。モデルリファレンス機能を用いてPlantサブシステムから同モデルを参照しています。なお、制御器のジャイロオフセット校正処理を考慮して、ジャイロオフセット校正時間が経過してからnxtway_gs_plant.mdlが動作するようにモデリングしています。ジャイロオフセットの校正については7.1 制御プログラム概要を参照してください。



ジャイロオフセット校正時間が経過してから動作するようにモデリング

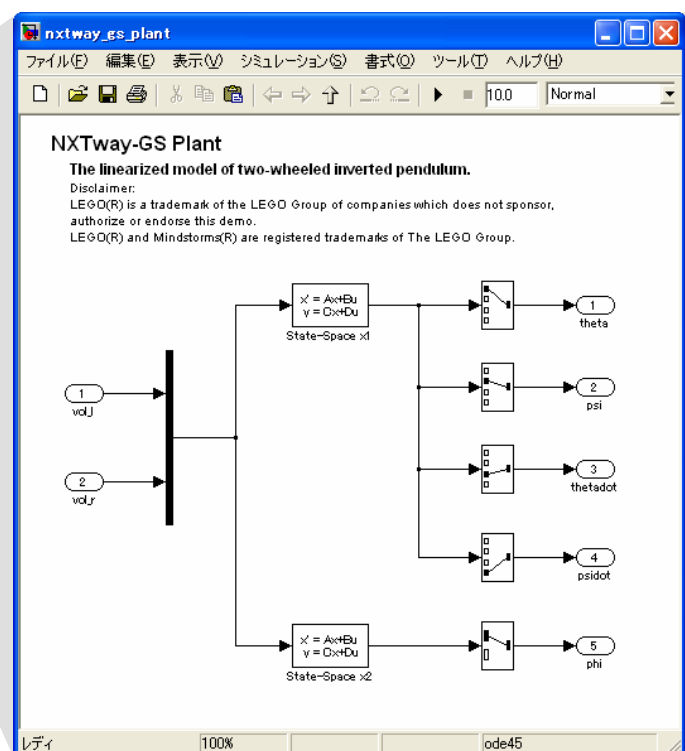
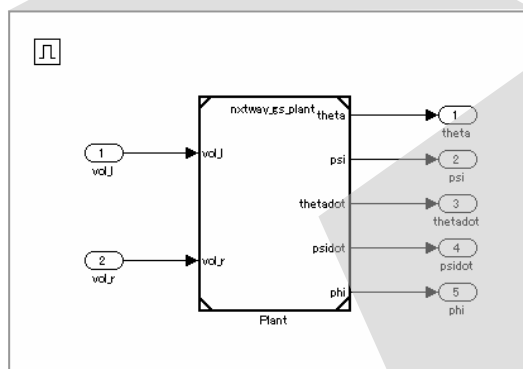


図 6-4 Plant サブシステム

6.4 センサ

プラントで計算された状態値をセンサ出力値に変換します。超音波センサによる壁との距離算出および壁との衝突検知は計算負荷が高いため、**Rate Transition**ブロックを挿入して計算ステップを間引いています。壁との距離算出・衝突検知については付録B.3を参照してください。

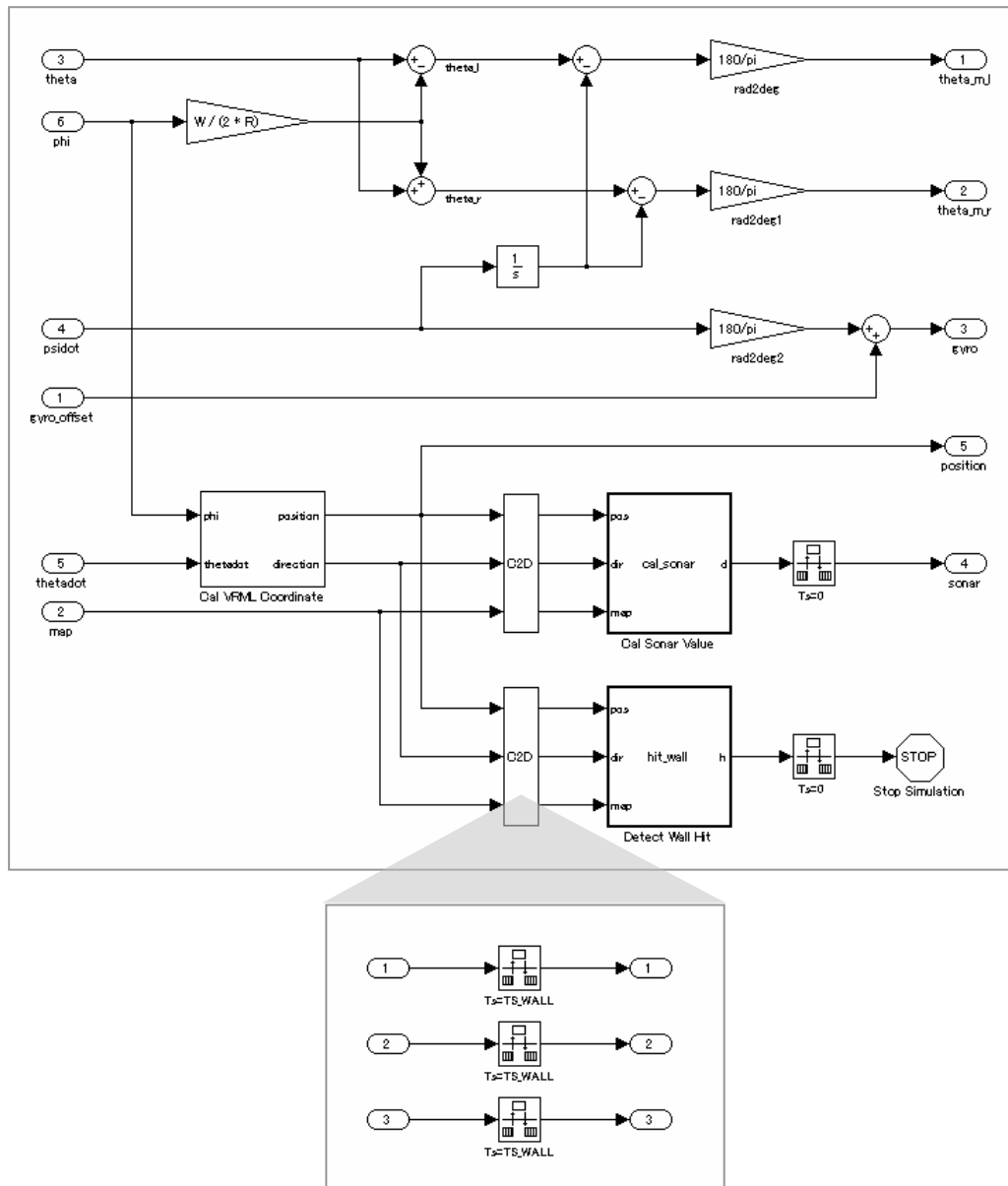


図 6-5 Sensor サブシステム

7 コントローラモデル（単精度浮動小数点演算）

制御器（コントローラ）モデルである `nxtway_gs_controller.mdl` の制御プログラム・タスク構成・モデル内容について説明します。

7.1 制御プログラム概要

ステートマシン図

図 7-1 は制御プログラムのステートマシン図です。`nxtway_gs_controller.mdl` に実装されている制御プログラムには自律走行モードと PC ゲームパッドによるリモコン操縦モードの 2 つの動作モードが用意されています。

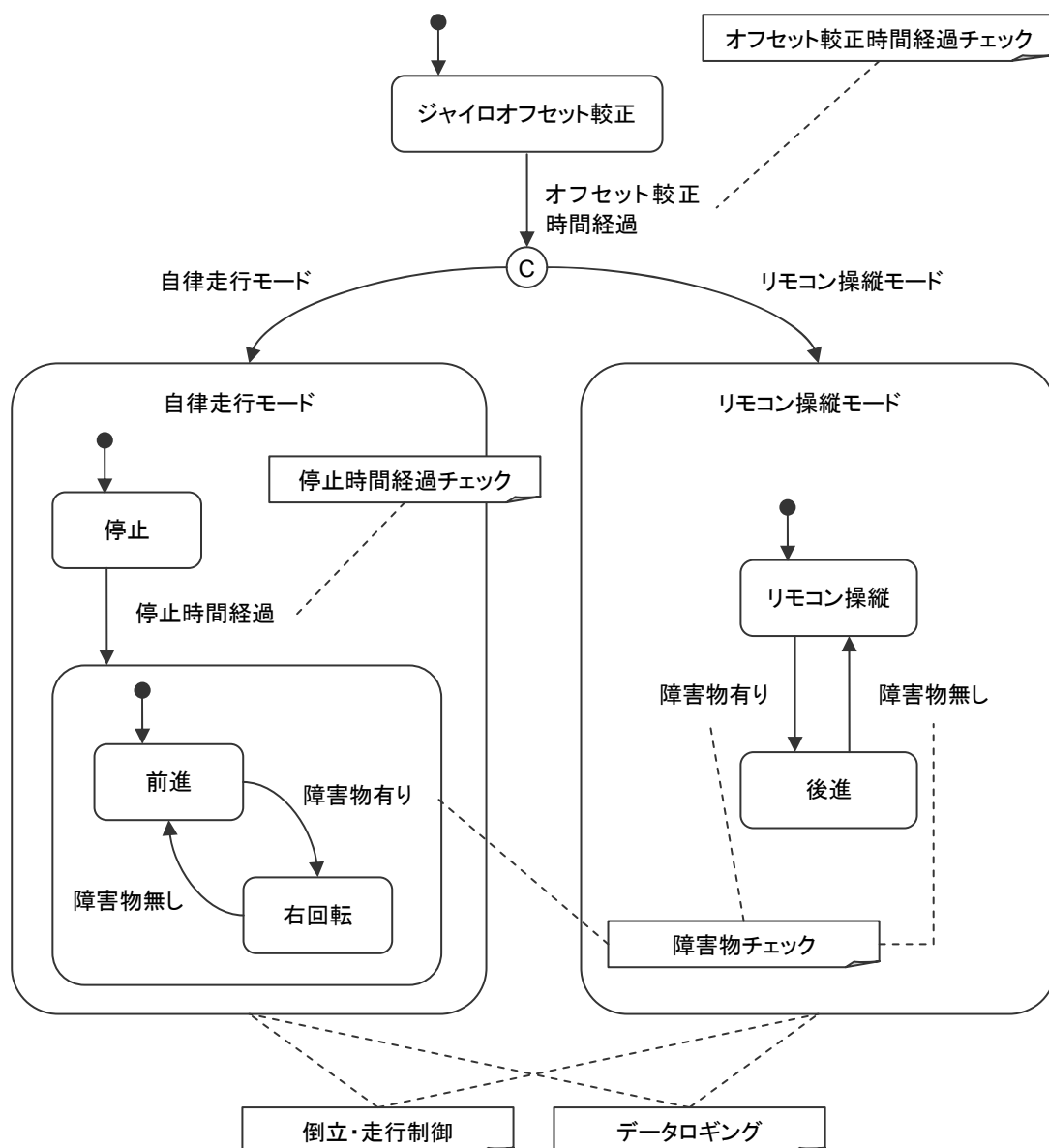


図 7-1 制御プログラムのステートマシン図

タスク構成

制御プログラムを表 7-1 の 4 つのタスクで構成します。

表 7-1 制御プログラムのタスク構成

| タスク名 | 駆動タイミング | 主な処理 |
|-----------|-------------|-----------------------------------|
| task_init | 起動時のみ | 初期値設定 |
| task_ts1 | 4 [ms] 周期 | 倒立・走行制御 データロギング ジャイロオフセット較正 |
| task_ts2 | 20 [ms] 周期 | 障害物チェック |
| task_ts3 | 100 [ms] 周期 | 経過時間チェック バッテリー電圧平均値計算 |

ジャイロセンサの 1 秒当たり最大サンプル数を考慮して、倒立・走行制御のタスク周期を 4 [ms] に設定しています。同様の理由で、超音波センサを用いた障害物チェック処理のタスク周期を 20 [ms] に設定しています。倒立・走行制御には**4.2 制御器設計**で説明した制御器を使用します。

データタイプ

演算誤差を抑える目的で、倒立・走行制御演算を単精度浮動小数点データで行うことにします。LEGO Mindstorms NXT に搭載されている ARM 7 プロセッサには FPU（浮動小数点演算装置）が搭載されていませんが、GCC の浮動小数点演算ライブラリを用いてソフトウェア的に単精度浮動小数点演算を行うことができます。

7.2 モデル概要

nxtway_gs_controller.mdl は Embedded Coder Robot NXT フレームワークに基づいてモデリングされています。

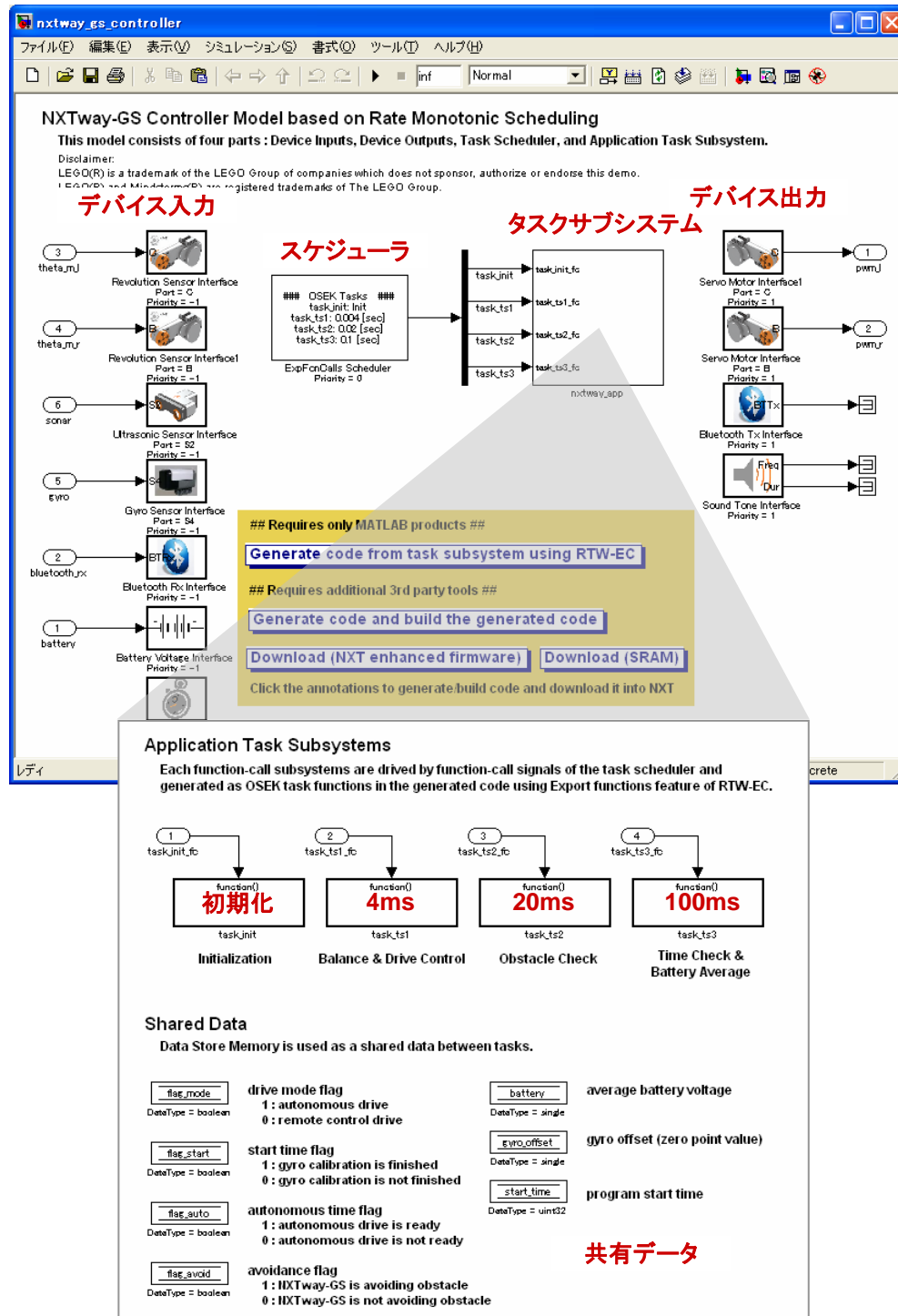


図 7-2 nxtway_gs_controller.mdl

デバイスインタフェース

Embedded Coder Robot NXT ライブラリに用意されている各種センサ・アクチュエータ用ブロックを用いてデバイス入出力インタフェースを作成しています。

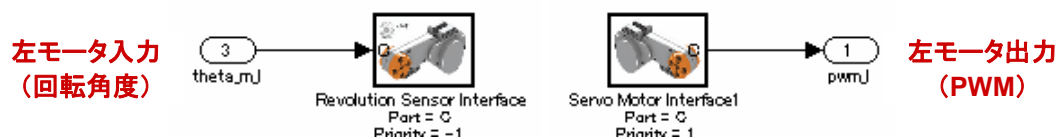


図 7-3 DC モータの入出力インタフェース

スケジューラ&タスク

ExpFcnCalls Scheduler ブロックを用いてタスク設定（タスク名、サンプル時間、プラットフォーム、スタックサイズ）を行い、同ブロックから出力される Function-Call 信号を Function-Call Subsystem に接続してタスク用サブシステムを作成しています。

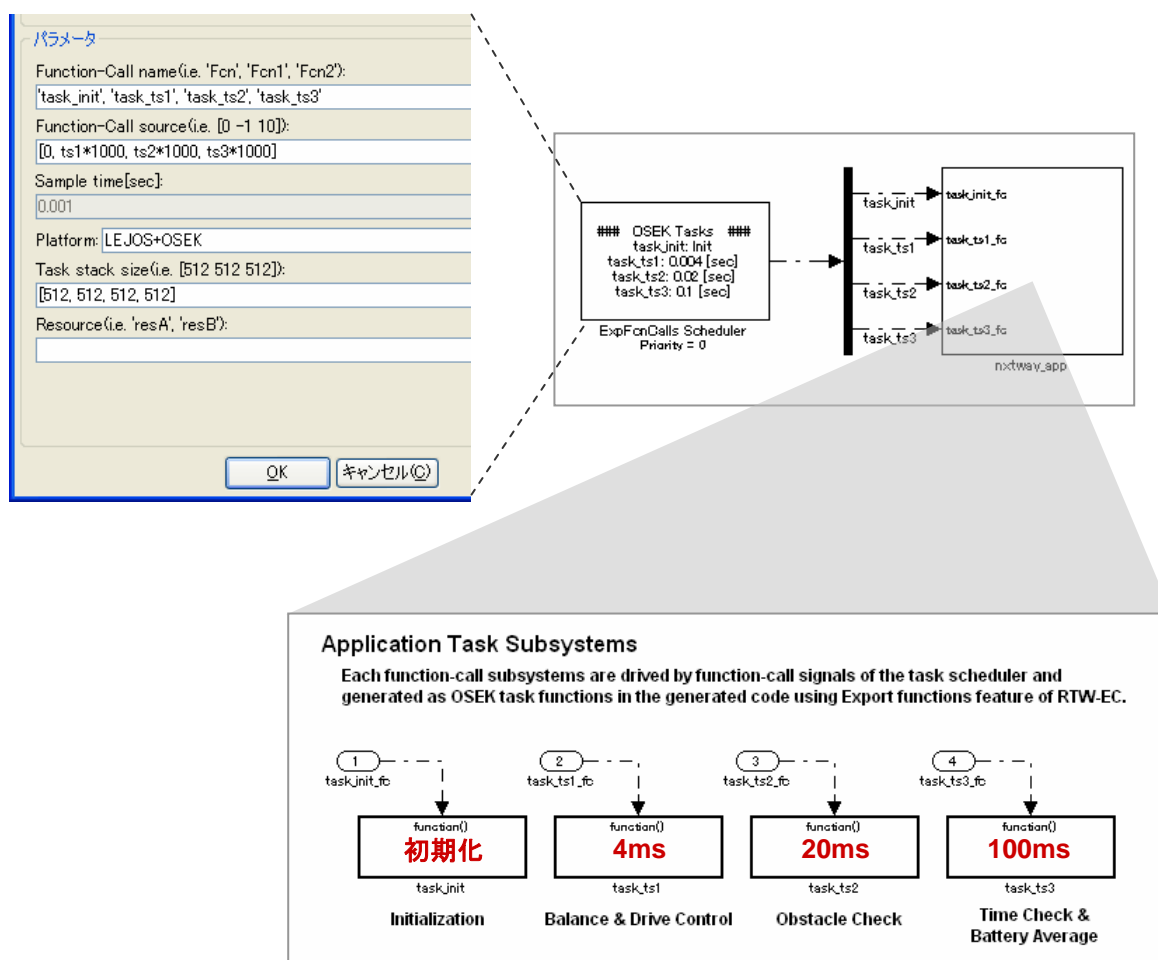


図 7-4 スケジューラ&タスク

優先度

デバイス入力→タスク→デバイス出力の順で処理が行われるように、各ブロックに優先度を設定しています。数値が小さいと優先度が高いことを示します。負の値も設定可能です。

優先度を設定するには、ブロックを右クリックして表示されるコンテキストメニューから [ブロックプロパティ] を選択してください。

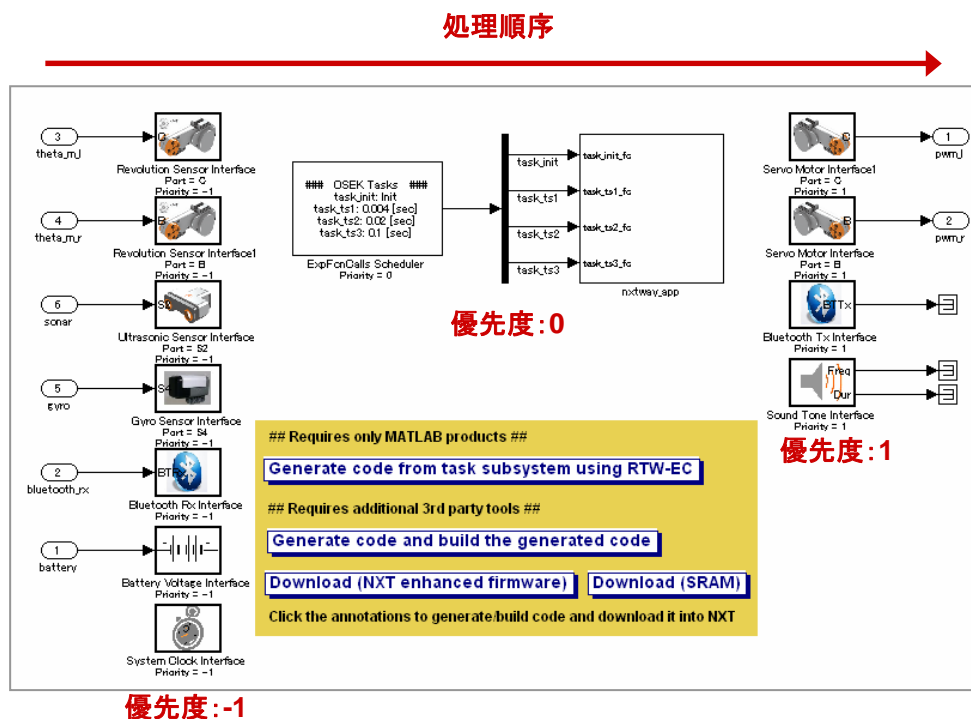


図 7-5 優先度設定による処理順序

共有データ

タスク間で共有するデータとして Data Store Memory を使用しています。

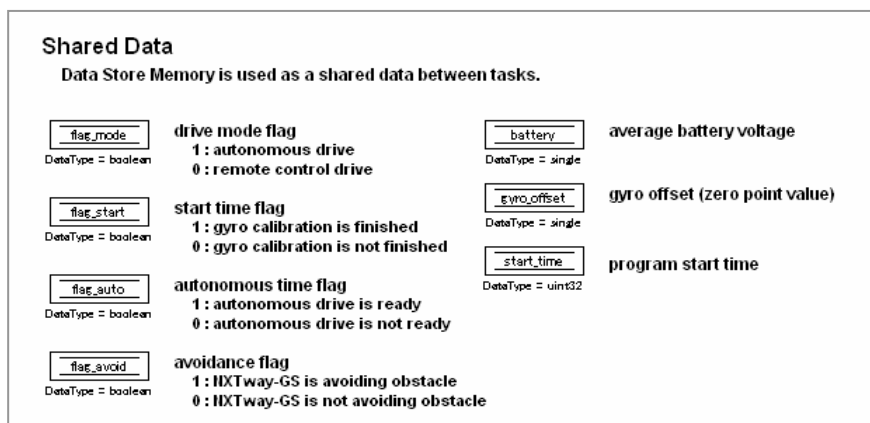


図 7-6 共有データ

7.3 初期化タスク : task_init

初期値設定を行うタスクです。同タスク内でモード切り替え（自律走行 or リモコン操縦）を切り替えることができます。

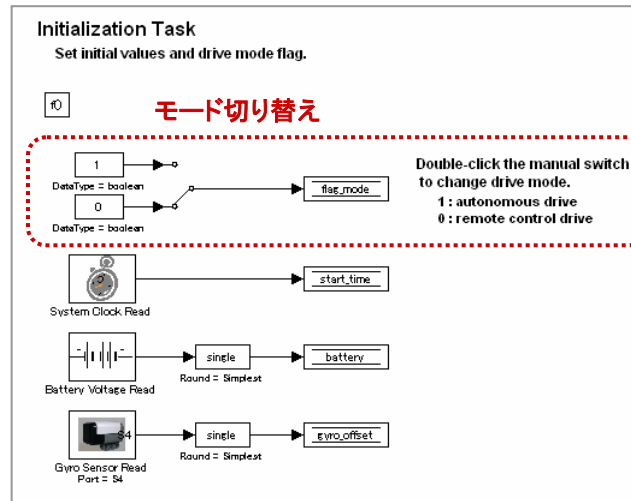


図 7-7 task_init

7.4 4ms タスク : task_ts1

ジャイロオフセット校正・倒立・走行制御およびデータロギングを行うタスクです。ジャイロオフセット校正後、倒立・走行制御を行います。ジャイロオフセットの校正時間は param_controller.m 内で time_start として定義されています。

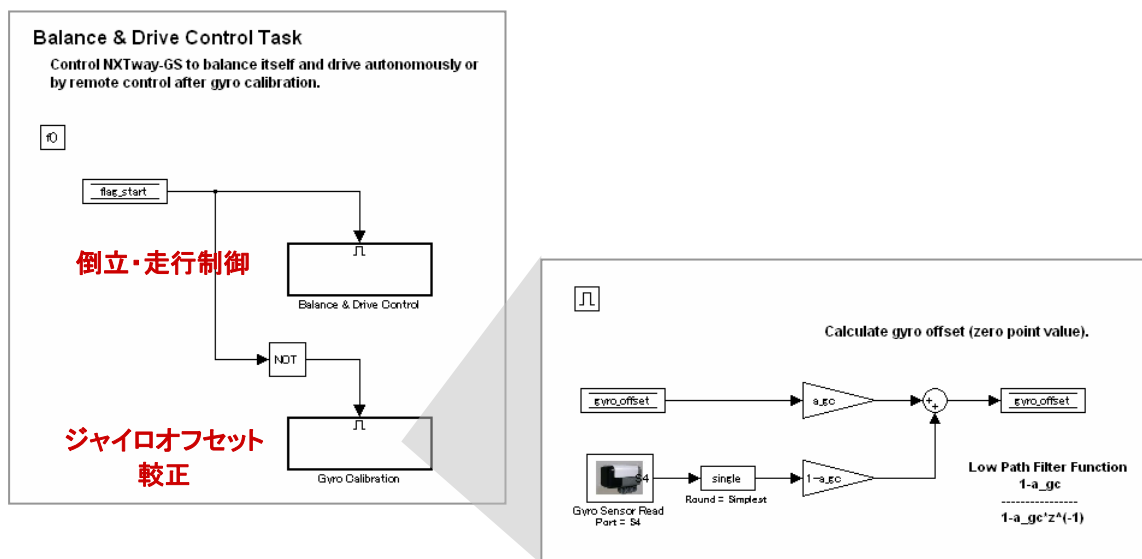


図 7-8 task_ts1

Balance & Drive Controlサブシステム内には4.2 制御器設計で説明した制御器がモデリングされています。

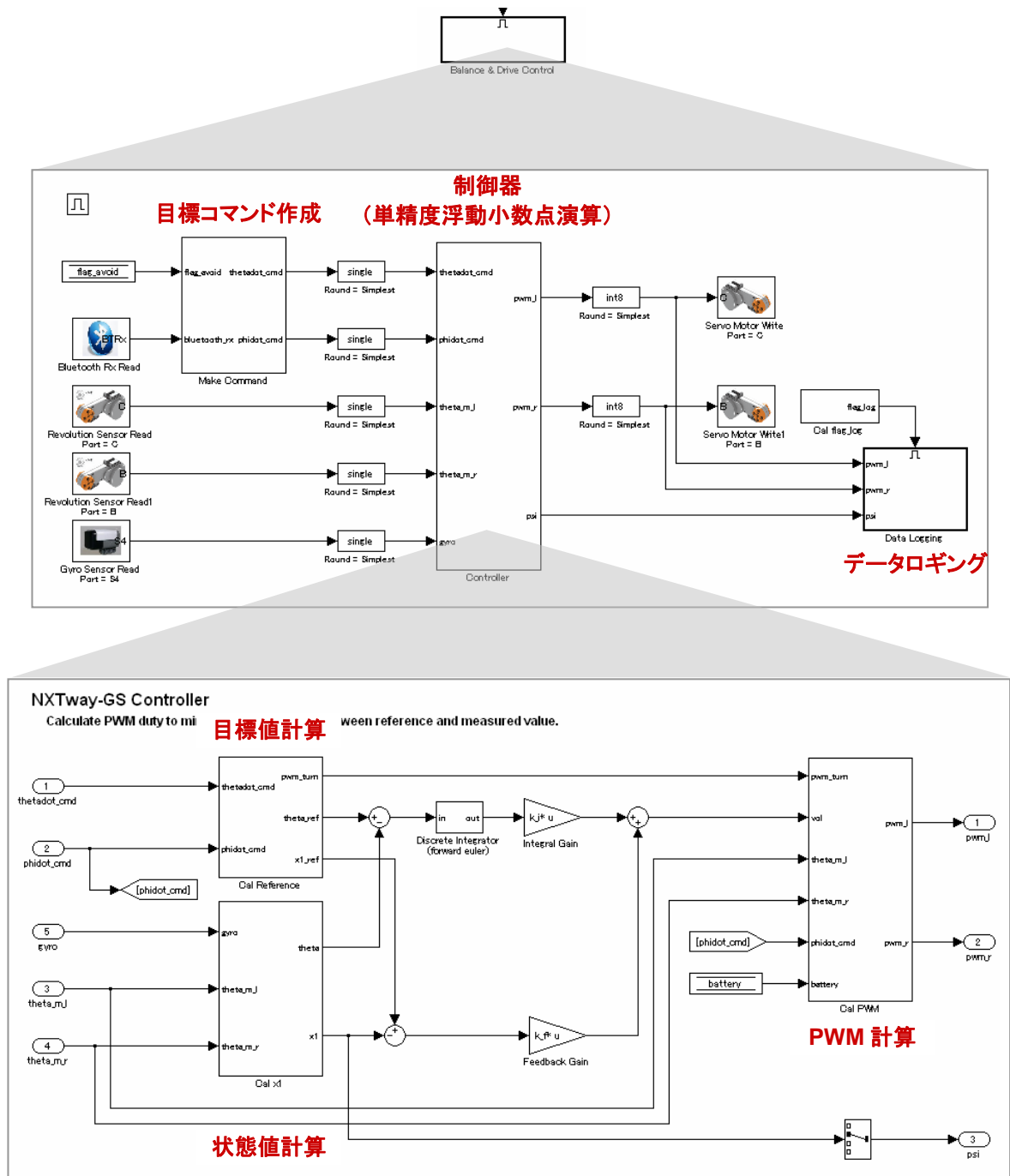


図 7-9 倒立・走行制御器モデル

数値微分・積分ブロック

微分計算には後退差分法、積分計算には前進 Euler 法を使用しています。

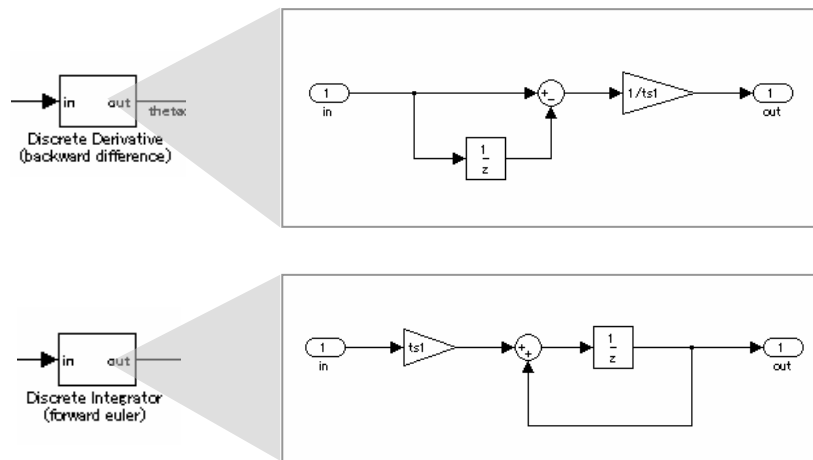


図 7-10 数値微分・積分ブロック

目標値計算 (Cal Reference)

目標値の急激な変化によるオーバーシュートを抑えるために、目標値にローパスフィルタをかけています。

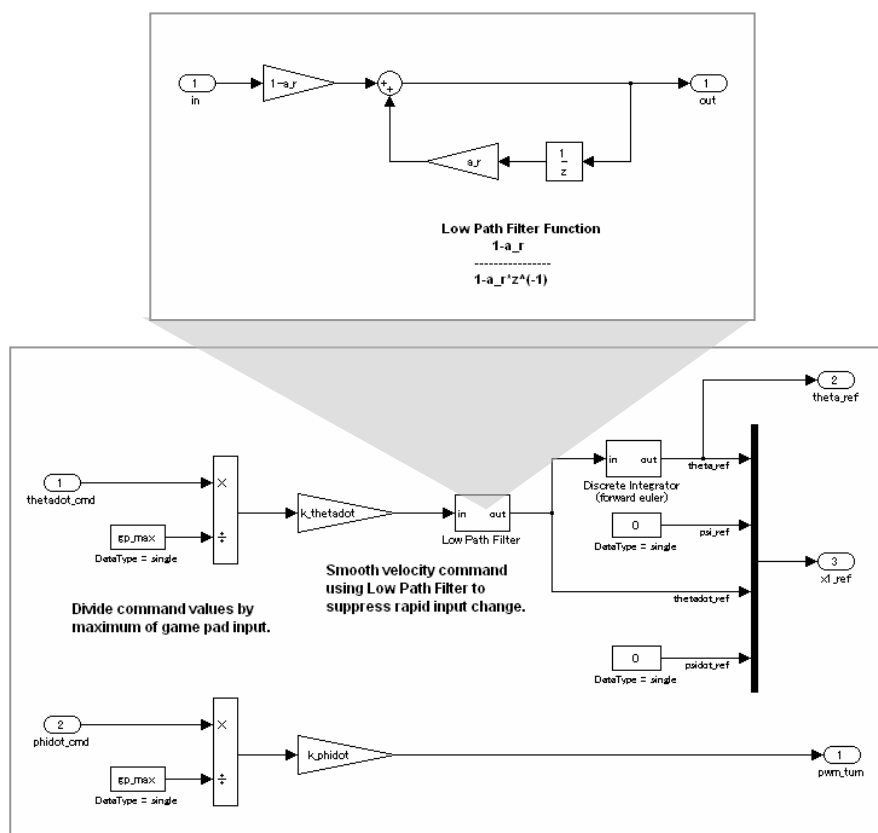


図 7-11 目標値計算

状態値計算 (Cal x1)

ジャイロのドリフト対策として、ジャイロの長時間平均値をジャイロオフセットとして用いています。また、速度信号のノイズを除去するためにローパスフィルタを用いています。

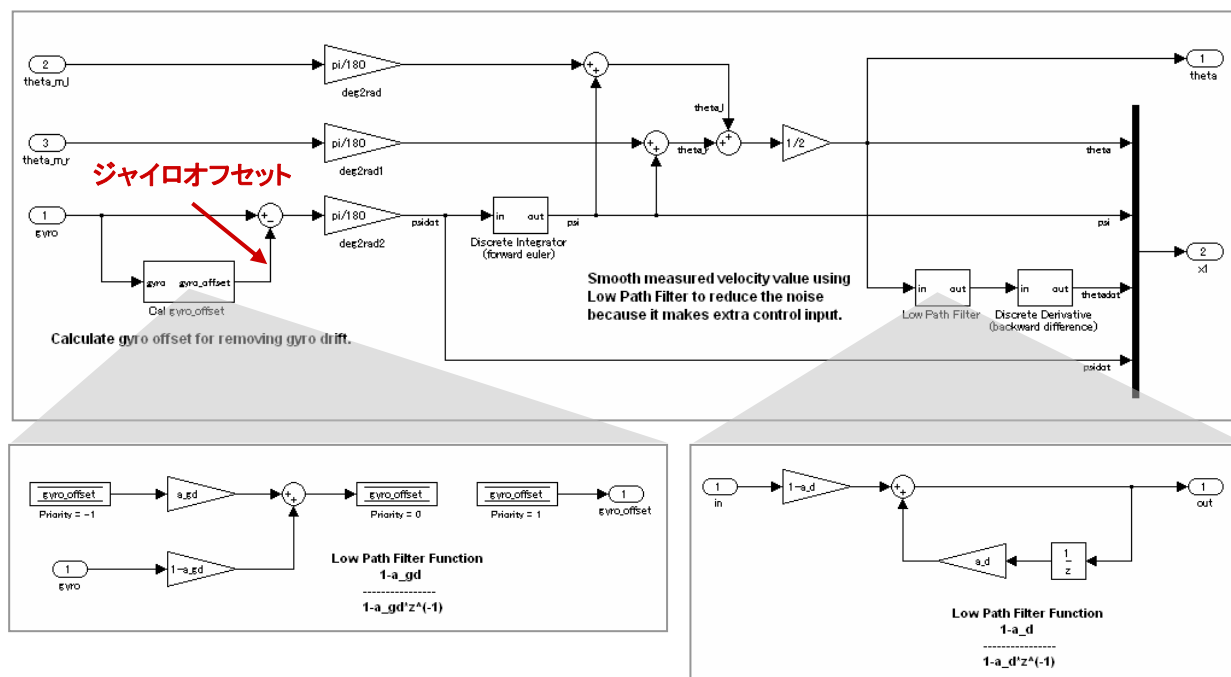


図 7-12 状態値計算

PWM計算 (Cal PWM)

図 4-4 のブロック線図を基に PWM デューティ値を計算しています。電圧→PWM 計算には(6.1)式を用いています。また、PWM 計算値に対して駆動系の摩擦補償を行っています。

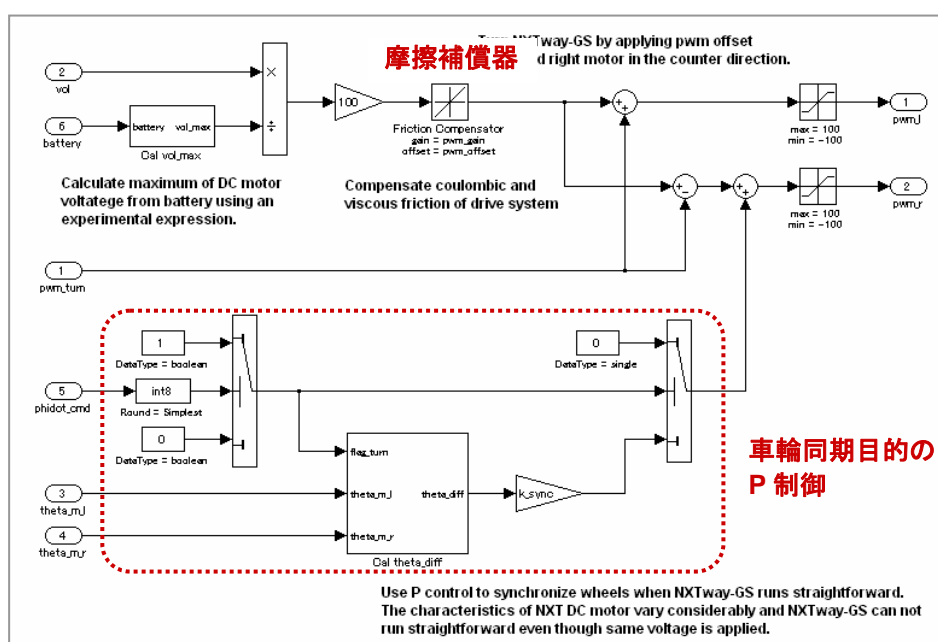


図 7-13 PWM 計算

データロギング

センサ値および計算結果をロギングするために NXT GamePad ADC Data Logger を配置しています。ロギングタイミングを変更するには param_controller.m 内で定義されている log_count の値を変更します。

NXT GamePad ADC Data Logger でロギングできるデータタイプは 8 ビットおよび 16 ビットの符号有り整数データとなっています。浮動小数点データをそのままロギングすることはできないため、浮動小数点データを適当な値 (= 整数データ 1 ビット当りの値。この値をスロープまたは **LSB** と呼びます) で割って整数データにキャストしてからロギングする必要があります。ロギング結果にスロープ (**LSB**) をかければ元の値を得ることができます。

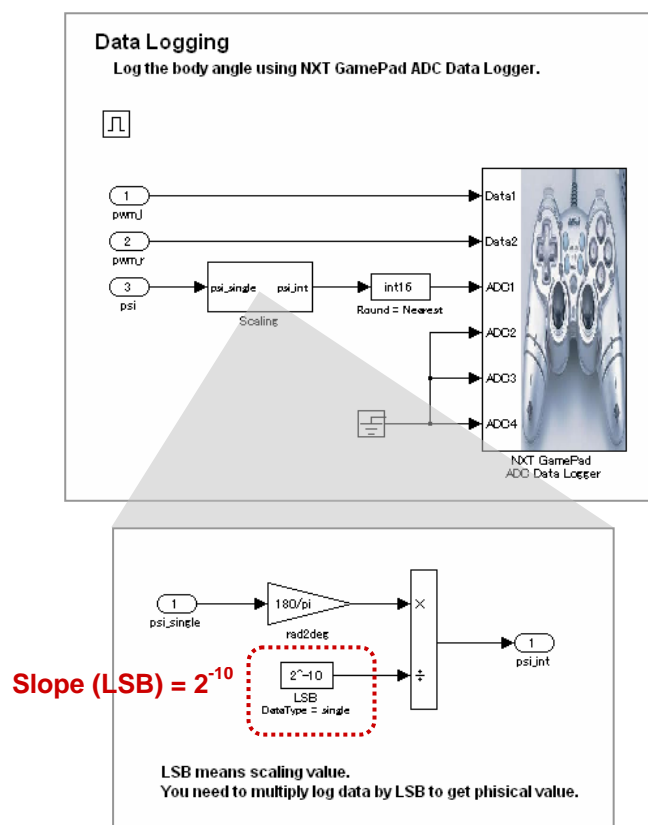


図 7-14 データロギング

7.5 20ms タスク : task_ts2

超音波センサを用いて障害物のチェックを行うタスクです。障害物を検知すると、自律走行モードでは右回転、リモコン操縦モードでは後進します。

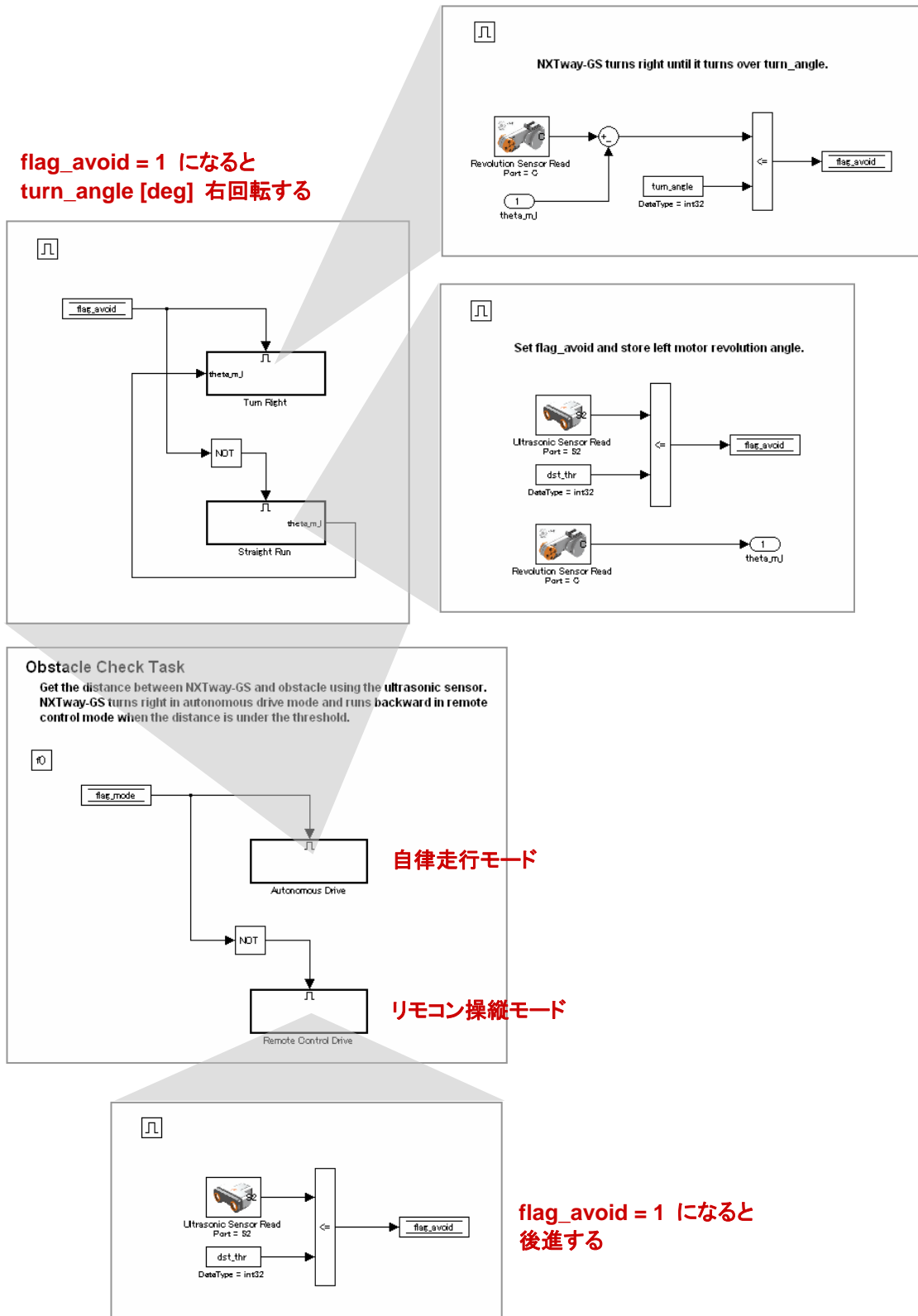


図 7-15 task_ts2

7.6 100ms タスク : task_ts3

制御プログラム経過時間のチェックおよびバッテリー電圧の平均値を求めるタスクです。ジャイロオフセット較正が終了すると、sound_dur [ms] サウンドが出力されるようになっています。

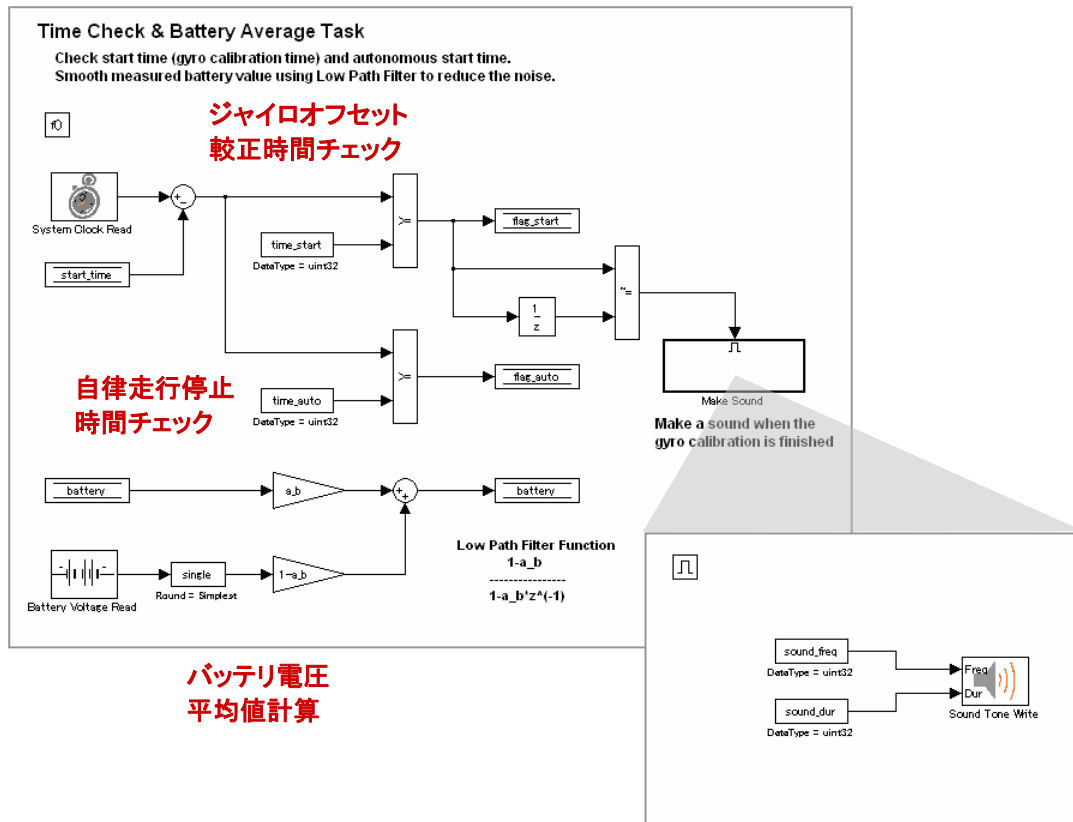


図 7-16 task_ts3

7.7 チューニングパラメータ

nxtway_gs_controller.mdl で使用するパラメータは全て param_controller.m 内で定義されています。倒立・走行制御性能を決めるチューニングパラメータは表 7-2 の通りです。LEGO Mindstorms NXT のパーツ・ブロックやセンサ・アクチュエータの個体差の関係で、チューニングパラメータを再調節する必要がある可能性がありますのでご注意ください。

表 7-2 チューニングパラメータ

| パラメータ | 機能 |
|------------|---------------------------|
| k_f | サーボ制御用フィードバックゲイン |
| k_i | サーボ制御用積分ゲイン |
| k_thetadot | 速度目標用ゲイン |
| k_phidot | 回転速度目標用ゲイン |
| k_sync | 車輪同期 P 制御用ゲイン |
| a_b | ローパスフィルタ係数（バッテリー電圧平均値計算用） |
| a_d | ローパスフィルタ係数（速度ノイズ抑制用） |
| a_r | ローパスフィルタ係数（オーバーシュート抑制用） |
| a_gc | ローパスフィルタ係数（ジャイロオフセット較正用） |
| a_gd | ローパスフィルタ係数（ジャイロドリフト対策用） |
| pwm_gain | 摩擦補償器ゲイン |
| pwm_offset | 摩擦補償器オフセット |
| dst_thr | 障害物回避距離 |
| turn_angle | 自律走行時右回転角度 |

8 シミュレーション

NXTway-GS モデルのシミュレーション方法およびシミュレーション結果について説明します。また、nxtway_gs_vr.mdl の 3D 表示について説明します。

8.1 シミュレーション方法

シミュレーション方法は通常の Simulink モデルと同じです。Reference Generator 内 Signal Builder ブロックを編集することにより、様々な目標信号を生成することができます。

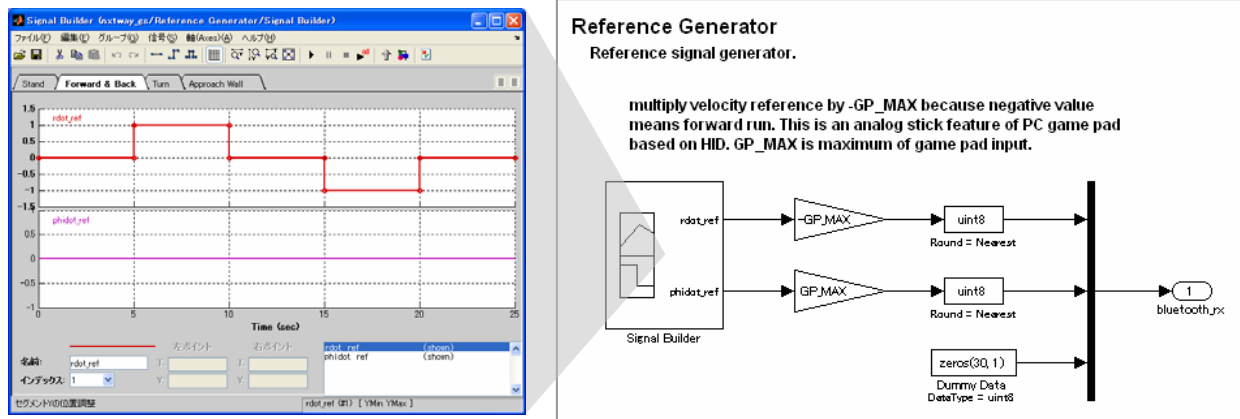


図 8-1 Reference Generator サブシステム

8.2 シミュレーション結果

倒立静止

図 8-2 は車体傾斜角度の初期値を 5 [deg] に設定して倒立させた際のシミュレーション結果です。車体傾斜角度が速やかに 0 [deg] に収束していることが分かります。なお、最初の 1 [s] はジャイロオフセット較正の時間であり、NXTway-GS は動作していません。

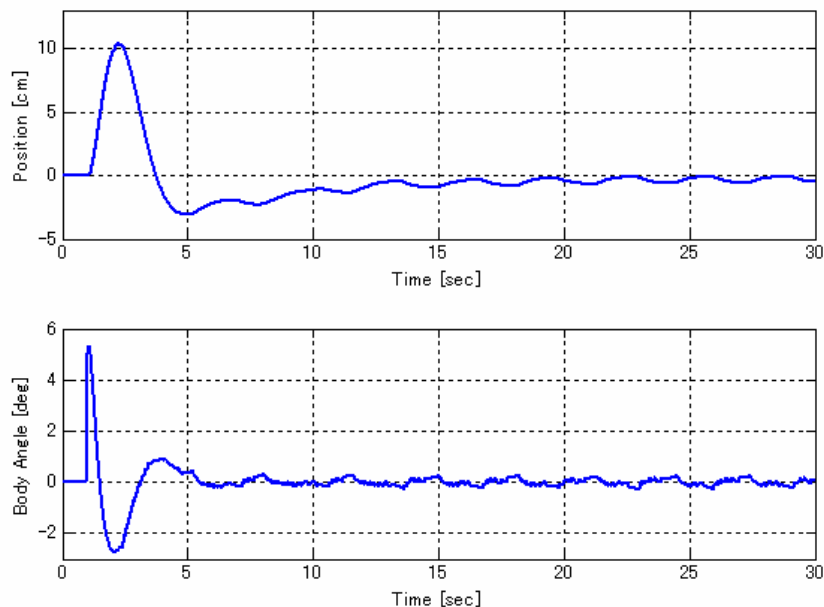


図 8-2 倒立静止時の位置・車体傾斜角度

前進&後進

図 8-3 は 20 秒静止→5 秒前進→20 秒静止→5 秒後進→20 秒静止した際のシミュレーション結果です。前進・後進時に大きな揺れが発生していますが、次第に収束していることが分かります。

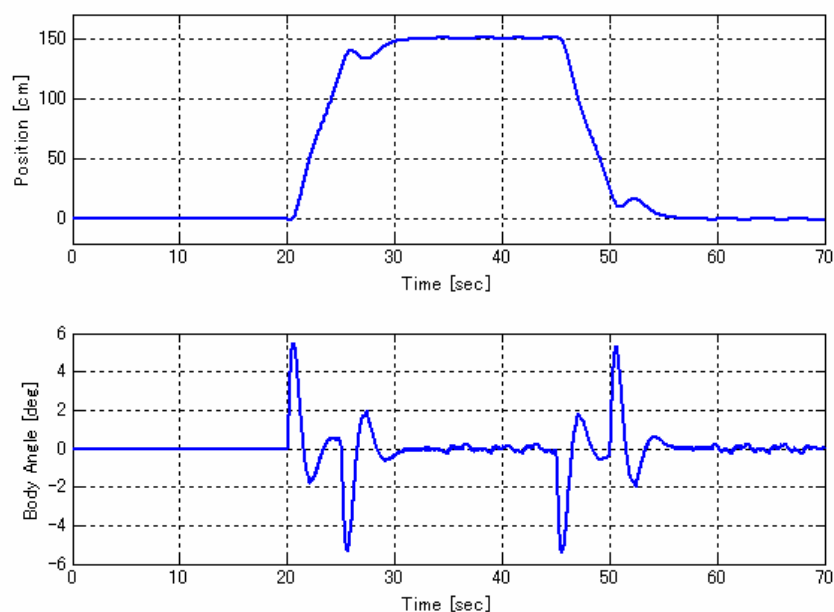


図 8-3 前進&後進時の位置・車体傾斜角度

下記 URL にシミュレーション動画が公開されています。前進&後進および旋回動作のシミュレーションの様子をご覧になれます。

<http://www.youtube.com/watch?v=EHPIGTLQHRc>

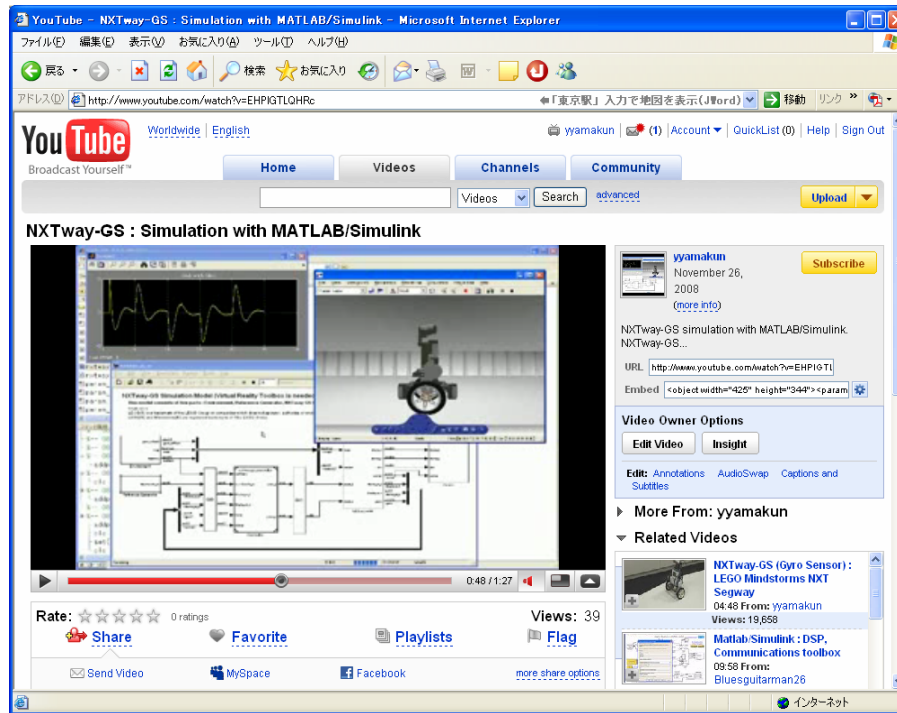
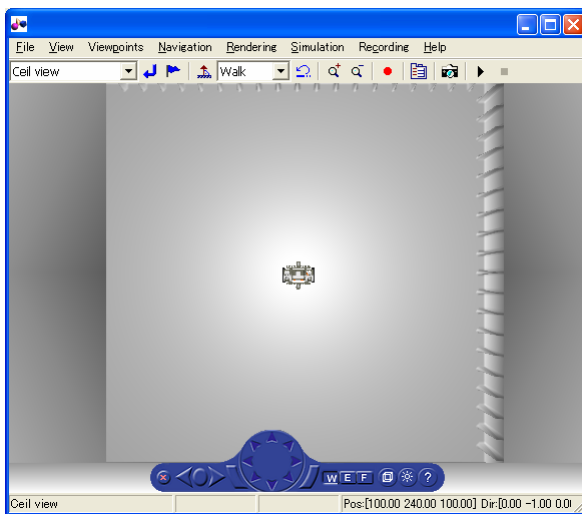


図 8-4 シミュレーション動画

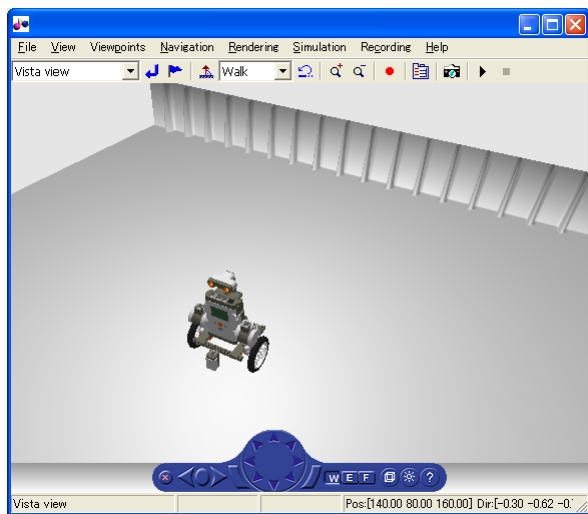
8.3 3D 表示

nxtway_gs_vr.mdl では、Virtual Reality Toolbox を用いた 3D 表示を行うことができます。Virtual Reality ウィンドウのビューモードを変更することにより、カメラ位置を変更することができます。nxtway_gs_vr.mdl では次の 4 つのビューモードが用意されています。

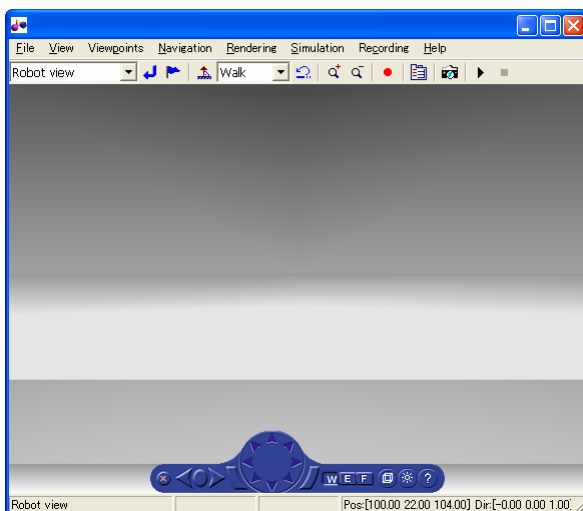
- Ceil View : 俯瞰視点
- Vista View : 固定カメラ視点
- Robot View : NXTWay-GS 超音波センサ視点
- Chaser View : NXTWay-GS 追跡カメラ視点



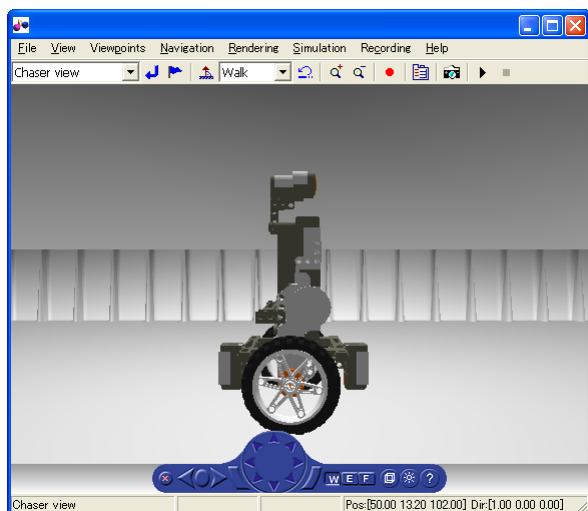
Ceil View



Vista View



Robot View



Chaser View

図 8-5 ビューモード

9 コード生成と実装

nxtway_gs_controller.mdl からのコード生成および生成コードの実装手順について説明します。また、実機を用いた実験結果を紹介します。

9.1 実装環境

表 9-1 は LEGO Mindstorms NXT の実装環境としての特徴および Embedded Coder Robot NXT のソフトウェア仕様をまとめた表です。

表 9-1 LEGO Mindstorms NXT & Embedded Coder Robot NXT 仕様

| | | |
|---------|----------|---|
| ハードウェア | マイコン | ATMEL 32-bit ARM 7 (AT91SAM7S256) 48MHz |
| | フラッシュメモリ | 256 K バイト（書き込みは 10000 回まで保証） |
| | RAM | 64 K バイト |
| インタフェース | アクチュエータ | DC モータ×3 |
| | センサ | 超音波センサ、接触センサ、光センサ、サウンドセンサ |
| | 液晶表示 | 100×64 ピクセル |
| | 通信 | Bluetooth |
| ソフトウェア | RTOS | LEJOS C / LEJOS OSEK |
| | コンパイラ | GCC |
| | ライブラリ | GCC ライブラリ（C 標準・浮動小数点演算ライブラリ） |

制御プログラムを NXT にダウンロードする際、以下の制約条件が課されます。

- プログラムをフラッシュメモリにダウンロードする場合
全てのプログラム領域をフラッシュサイズ（256K バイト）に収める必要がある。
全ての静的領域、スタック領域を RAM サイズ（64K バイト）に収める必要がある。
- プログラムを RAM にダウンロードする場合
全てのプログラム領域、静的領域、スタック領域を RAM サイズ（64K バイト）に収める必要がある。

9.2 コード生成・実装手順

図 9-1 に示す通り、nxtway_gs_controller.mdl 内のアノテーションをクリックすることにより、コード生成・ビルド・プログラムダウンロードを行うことができます。その手順は次の通りです。

1. [**Generate code and build the generated code**] をクリックしてモデルからコードを生成し、生成コードをビルドします。
2. NXT と PC を USB ケーブルで接続します。NXT のブートモード (NXT 拡張ファームウェア / SRAM ブート) に合わせて、[**Download (NXT enhanced firmware)**] または [**Download (SRAM)**] をクリックして NXT にプログラムをダウンロードします。

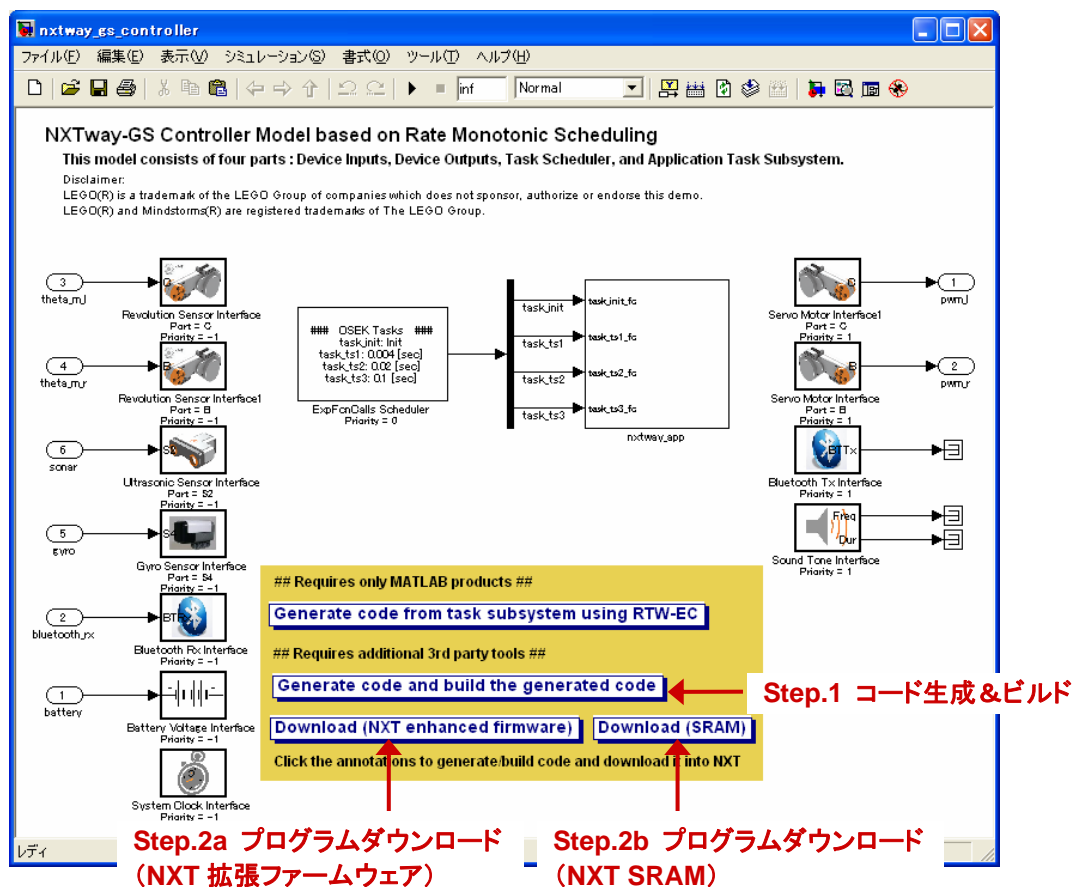


図 9-1 コード生成&ビルド / プログラムダウンロード用アノテーション

付録Cにコード生成結果の一部が掲載されています。

9.3 実験結果

NXT 実機を用いた実験結果は次の通りです。シミュレーションとほぼ類似の結果が得られました。

倒立静止

図 9-2 は倒立静止状態の実験結果です。位置および車体傾斜角度が 0 近傍で安定していることが分かります。

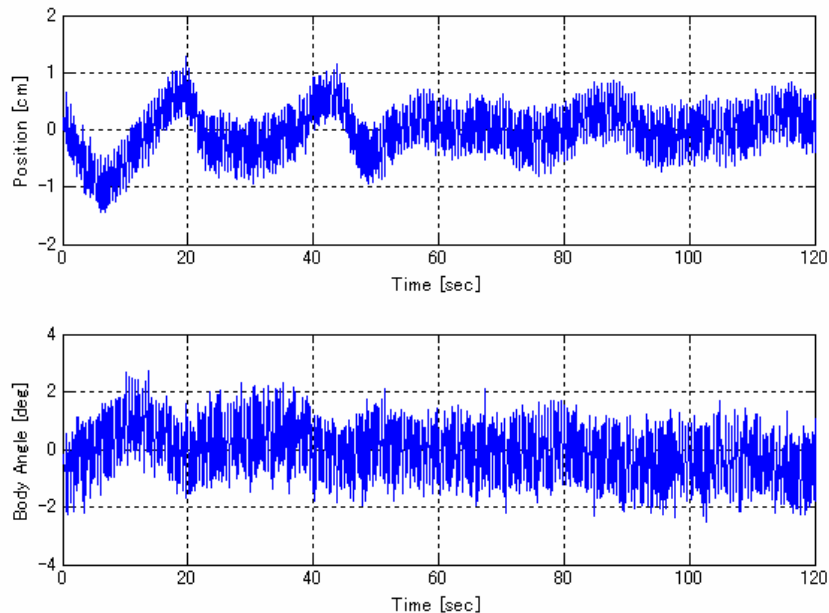


図 9-2 倒立静止時の位置・車体傾斜角度

前進&後進

図 9-3 は 20 秒静止→5 秒前進→20 秒静止→5 秒後進→20 秒静止した際の実験結果です。図 8-3 とほぼ同じ結果となっています。

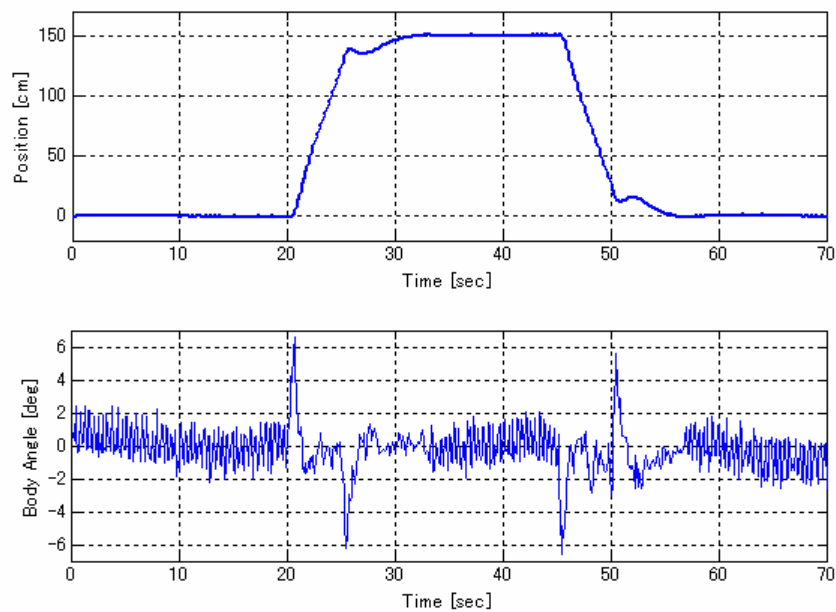


図 9-3 前進・後進時の位置・車体傾斜角度

下記 URL に NXTway-GS の制御実験動画が公開されています。外乱応答や PC ゲームパッドによるリモコン操作の様子をご覧になれます。

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



図 9-4 NXTway-GS 制御実験動画

10 コントローラモデル（固定小数点演算）

制御器モデル内で使用されるデータを単精度浮動小数点数から固定小数点数に変換し、精度およびオーバーフローが制御性能に与える影響について考察します。

10.1 固定小数点数とは

固定小数点数とは、コンピュータにおける実数の近似値の表現形式の一種です。浮動小数点数とは異なり、固定小数点数では整数型データを用いて実数を表現します。固定小数点数は一般に次のスケール式で表すことができます。

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

V_R : 実数 V_0 : 固定小数点数 Q : 整数 S : スロープ B : バイアス

スロープは1ビット当たりの実数値、バイアスは整数値が0の時の実数値です。スロープを分解能や（慣習的に）LSB、バイアスをオフセットと呼ぶこともあります。

固定小数点数は有限の分解能を用いて実数を表すため、数値によっては丸め誤差が発生します。また、整数型データを用いるため表現できる値に制限があります。この値の表現範囲を決めるデータサイズを語長と呼びます。同一語長のもとでは、精度と表現範囲は一般にトレードオフの関係にあります。つまり、精度を上げる（スロープを小さくする）と、表現範囲が小さくなります。

例題

$V_r=2$ を8ビット符号有り整数データ、 $S=0.3$ 、 $B=1$ で固定小数点数化した場合

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



固定小数点数を用いると、浮動小数点演算器（FPU）が無いマイコン・DSP 上での実数表現・計算が可能になります（固定小数点数は整数型データを用いるため）。一般に、固定小数点演算は浮動小数点演算よりも高速であるという利点があります。一方、固定小数点数は浮動小数点数に比べて表現できる値の範囲が狭く、オーバーフローが起きやすいという欠点があります。表 10-1 は固定小数点数と浮動小数点数の比較表です。

表 10-1 固定小数点数・浮動小数点数の比較表

| 比較項目 | 固定小数点数 | 浮動小数点数 |
|--------------|--------|--------|
| 実行速度 | 速い | 遅い |
| 消費電力 | 小さい | 大きい |
| RAM / ROM 消費 | 少ない | 多い |
| 語長&分解能 | 変更可能 | 変更不可 |
| 表現範囲 | 狭い | 広い |
| 量子化誤差 | 大きい | 小さい |
| 開発時間 | 長い | 短い |

10.2 コントローラモデルの固定小数点化

MATLAB / Simulink では、固定小数点数を用いたプログラミング・シミュレーション・コード生成を行うための製品として、Fixed-Point Toolbox / Simulink Fixed Point が提供されています。両製品の使用方法については、参考文献 [4]、[5]、[6] を参照してください。

nxtway_gs_controller_fixpt.mdl は、nxtway_gs_controller.mdl 内の車輪角度・車体傾斜角度・バッテリーの各信号・パラメータのデータタイプを単精度浮動小数点数から固定小数点数に置き換えたモデルです。以下、固定小数点化に伴う主な変更点について説明します。

固定小数点データタイプ

param_controller_fixpt.m に nxtway_gs_controller_fixpt.mdl で使用する Simulink.NumericType オブジェクトが定義されています。Simulink.NumericType オブジェクトは各種データタイプを表すためのオブジェクトです。

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);
```

一般に、制御器の性能は使用するデータタイプに依存します。**10.1 固定小数点数とは**で説明しましたように、同一語長のもとでは精度と表現範囲はトレードオフの関係にあります。制御設計者は、このトレードオフを考慮して最適な固定小数点設定を行う必要があります。

NXTway-GS の場合、特に車輪角度・車体傾斜角度の固定小数点設定が重要です。何故かという、両変数の精度が不足している、またはオーバーフローが発生すると、NXTway-GS が倒立できなくなるためです。NXTway-GS の最大移動距離は次式を用いて計算することができます。

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

ここで、R は車輪半径、S は dt_theta のスロープです。最大移動距離は表 10-2 の通りです。

表 10-2 NXTway-GS の最大移動距離

| dt_theta スロープ | 2^{-10} | 2^{-14} | 2^{-18} | 2^{-22} |
|---------------|-----------|-----------|-----------|-----------|
| 最大移動距離 [m] | 1464.1 | 91.5 | 5.7 | 0.36 |

固定小数点モデル (nxtway_gs_controller_fixpt.mdl)

制御器モデルの主な変更点は次の通りです。

- 単精度浮動小数点データを固定小数点データに置き換えている。
- Mux ブロックでベクトル化する信号を同じ固定小数点データで統一している（異なるデータタイプの信号をベクトル化できないため）。
- PWM 計算部分でスロープ合わせの設定を行っている。

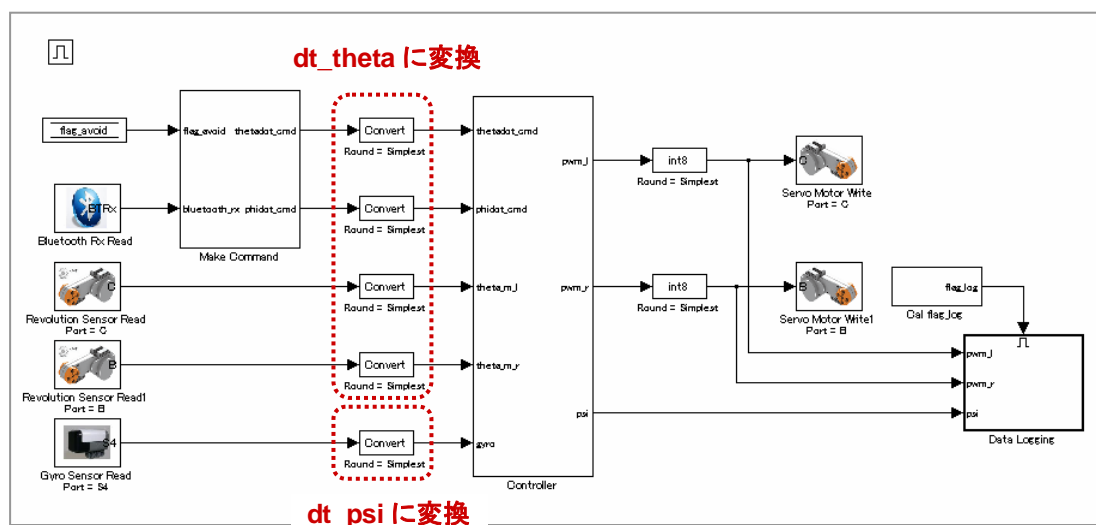


図 10-1 NXTway-GS 制御器の固定小数点設定

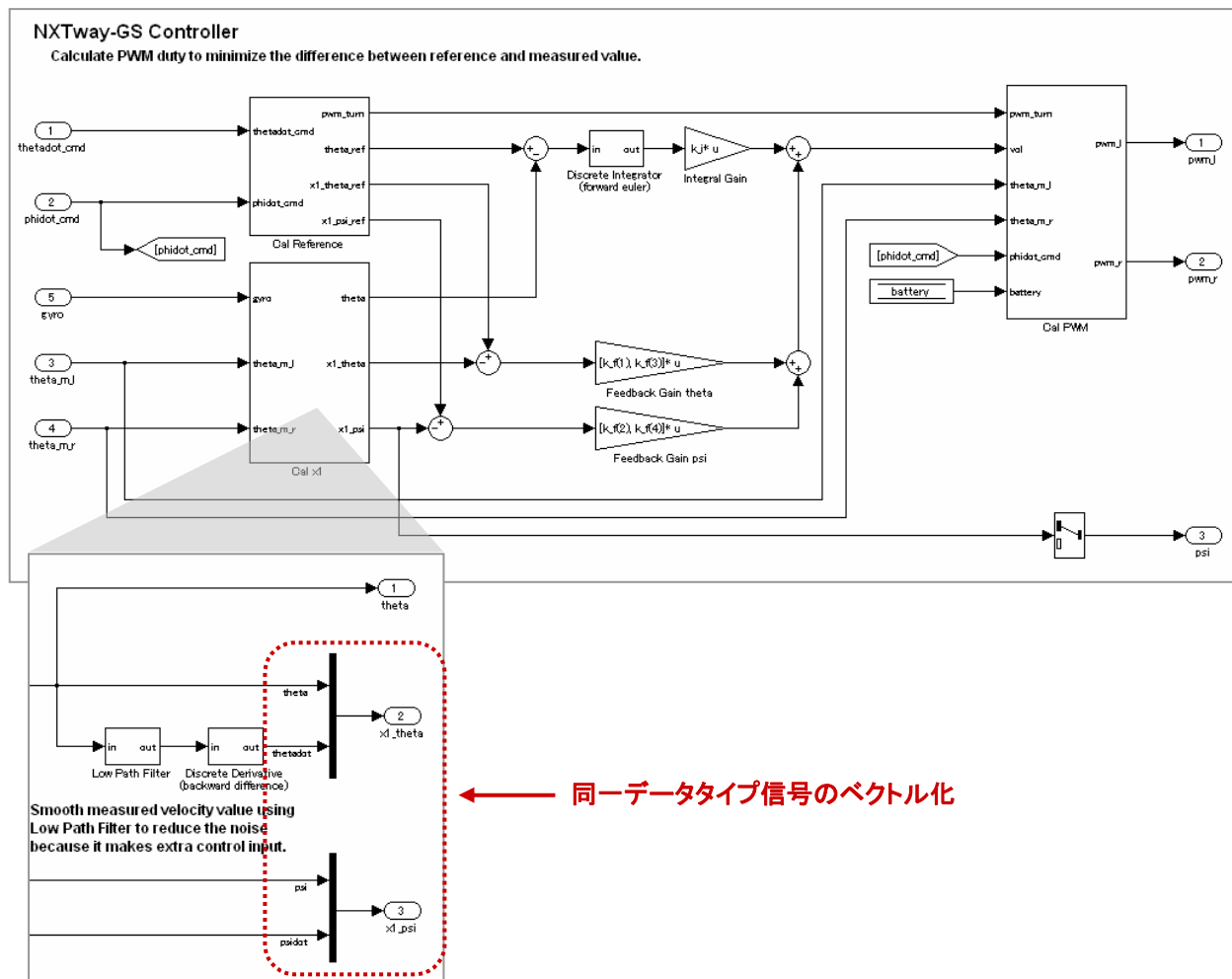


図 10-2 同一データタイプ信号のベクトル化

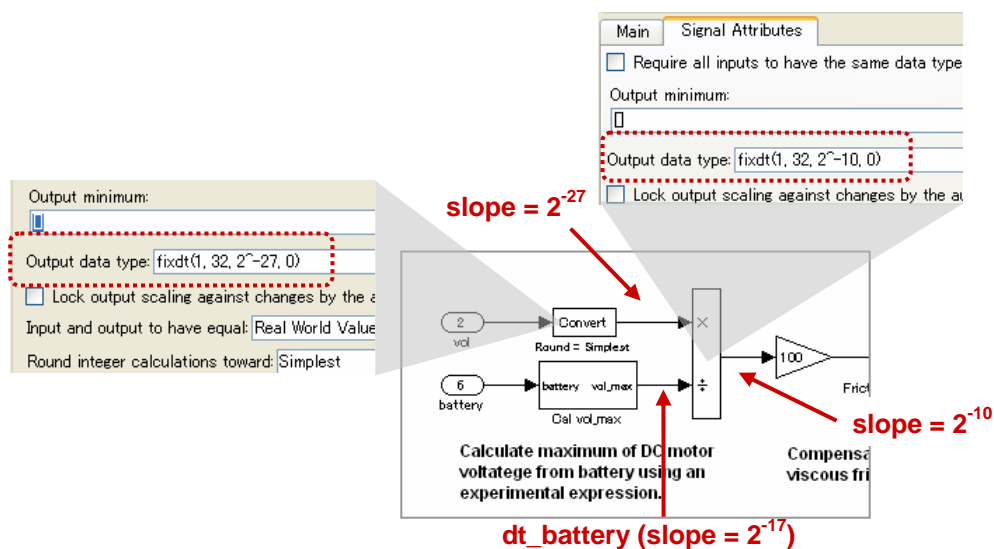


図 10-3 PWM 計算のスロープ合わせ

10.3 シミュレーション結果

nxtway_gs_controller_fixpt.mdl のシミュレーションを行うには、nxtway_gs.mdl / nxtway_gs_vr.mdl 内 Controller ブロックが参照しているモデルを nxtway_gs_controller.mdl から同モデルに変更してください。

Model ブロックの参照モデルを変更するには、同ブロックを右クリックして表示されるコンテキストメニューから **[Model Reference パラメータ]** を選択して **[モデル名]** を編集してください。

車輪角度の固定小数点データタイプ (dt_theta) のスロープを変えてシミュレーションを行うと、固定小数点精度（スロープ）が制御性能に与える影響を確認することができます。図 10-4 は、dt_theta のスロープを変更して倒立・静止状態のシミュレーションを行った結果です（車体傾斜角度の初期値は 5 [deg]）。スロープが大きいと静止状態への収束時間および静止状態での揺動が悪化していることが分かります。なお、スロープを 2^5 以上に大きくすると、精度不足により NXTway-GS は倒立することができません。

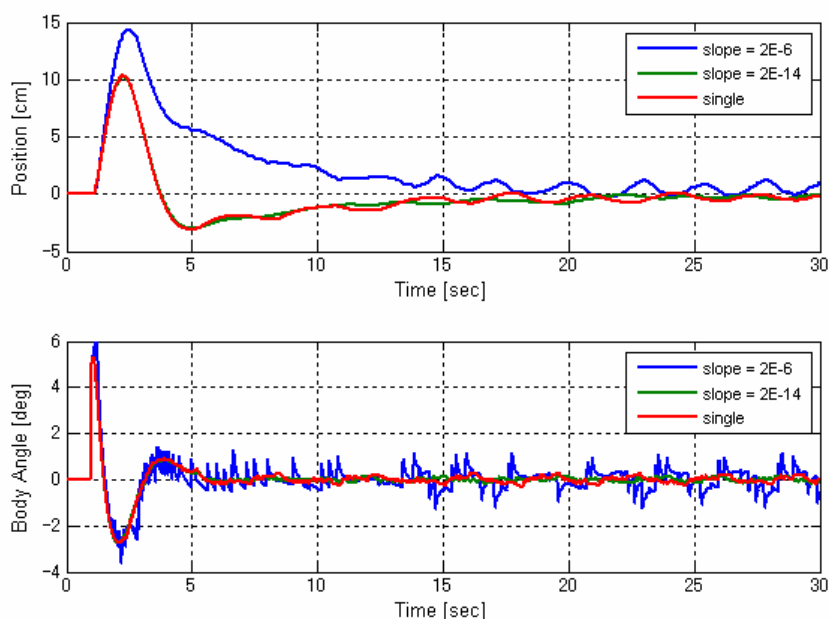


図 10-4 スロープの影響 ($2E-6 = 2^{-6}$)

一方、スロープを小さくするとNXTway-GSの最大移動距離が短くなります。例えば、スロープを 2^{22} に設定すると、表 10-2 より 0.36 [m] 直進後にオーバーフローが発生してNXTway-GSが転倒することが分かります。図 10-5 はオーバーフローによりNXTway-GSが転倒した様子をキャプチャした画像です。

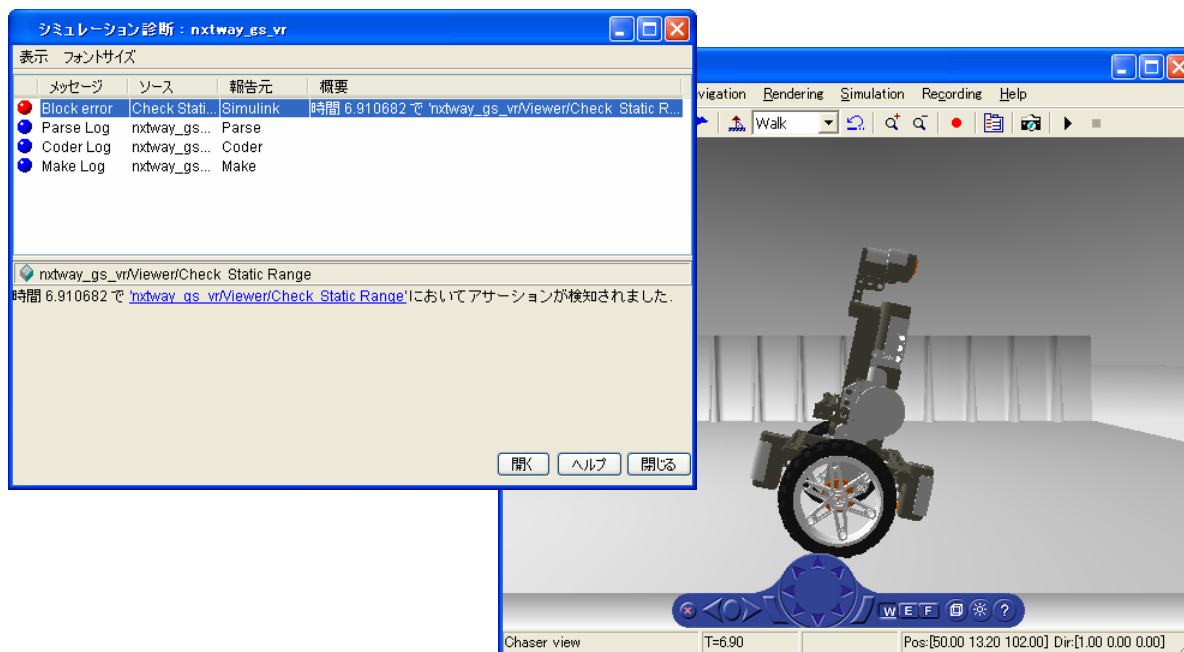


図 10-5 オーバーフローによる転倒

10.4 コード生成&実験結果

nxtway_gs_controller_fixpt.mdl のコード生成および実装手順は nxtway_gs_controller.mdl と同じです。生成コードは整数データのみを用いた固定小数点演算コードとなります。NXT 実機に固定小数点演算コードを実装すると、シミュレーション結果と同じ動作（精度不足による制御性能の悪化、オーバーフローによる転倒）を確認することができます。

11 読者への課題

以下の問題を読者への課題とします。興味のある方はトライして下さい。

- 制御器の改善（ゲインチューニング、ロバスト制御の適用等）
- NXT 標準小径車輪の適用
- 光センサによるライントレーシング
- 障害物距離に応じたサウンド出力

付録 A 現代制御理論

本付録では、NXTway-GS の制御器設計で用いる現代制御理論について簡単に説明します。詳細については参考文献[3]を参照してください。

A.1 安定性

一般に、入力 \mathbf{u} が 0 のとき任意の初期値に対して

$$\lim_{t \rightarrow \infty} \mathbf{x}(t) = 0 \quad (\text{A.1})$$

となる系を漸近安定であるといいます。系の状態方程式を

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

とすると、系が漸近安定であるための必要十分条件は、システム行列 \mathbf{A} の全ての固有値の実部が負であることです。実部が正である固有値を含む場合、系は不安定となります。

A.2 状態フィードバック制御

状態 \mathbf{x} を検出し、ゲイン \mathbf{K} を乗じてフィードバックする制御手法を状態フィードバック制御と呼びます。状態フィードバック制御は、古典制御の PD 制御に相当します。図 A-1 は状態フィードバック制御のブロック線図です。 \mathbf{x}_{ref} は \mathbf{x} に対する目標値です。

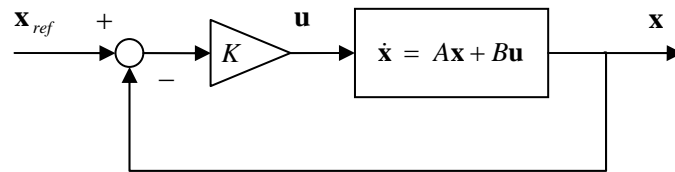


図 A-1 状態フィードバック制御のブロック線図

入力および状態方程式は次式の通りです。

$$\mathbf{u}(t) = -\mathbf{K}(\mathbf{x}(t) - \mathbf{x}_{ref}) \quad (\text{A.3})$$

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{BK})\mathbf{x}(t) + \mathbf{BK}\mathbf{x}_{ref} \quad (\text{A.4})$$

このフィードバック系のシステム行列は $\mathbf{A} - \mathbf{BK}$ ですので、ゲイン \mathbf{K} を適当に調節することにより系を安定にすることができます。

状態フィードバック制御で状態を任意に制御するためには、系が可制御である必要があります。可制御であるための必要十分条件は、可制御行列

$$M_c = [B, AB, \dots, A^{n-1}B] \quad (\text{A.5})$$

がフルランク ($\text{rank}(M_c) = n$) となることです (n は \mathbf{x} の次数)。Control System Toolbox では可制御行列を求める関数として `ctrb` 関数が提供されています。

例題

$A = [0, 1; -2, -3]$ 、 $B = [0; 1]$ の系は可制御か? → 可制御

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

状態フィードバックゲイン K を求める代表的な方法として、次の 2 通りの方法があります。

1. 極配置法

フィードバック系のシステム行列 $A - BK$ が指定された極 (固有値) を持つようにゲイン K を求める方法です。チューニングパラメータは極であり、試行錯誤で適切なゲインを求める必要があります。Control System Toolbox では極配置用の関数として `place` 関数が提供されています。

例題

$A = [0, 1; -2, -3]$ 、 $B = [0; 1]$ の系に対して、極を $[-5, -6]$ に配置するフィードバックゲインを求めよ。

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

2. 最適レギュレータ

次式で与えられる評価関数 J を最小化するゲイン K を求める方法です。

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

チューニングパラメータは状態に対する重み行列 Q および入力に対する重み行列 R であり、試行錯誤で適切なゲインを求める必要があります。Control System Toolbox では最適レギュレータ用の関数として `lqr` 関数が提供されています。

例題

$A = [0, 1; -2, -3]$ 、 $B = [0; 1]$ の系に対して、 $Q = [100, 0; 0, 1]$ 、 $R = 1$ で最適レギュレータのゲインを求めよ。

```
>> 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 サーボ制御

出力を目標値に追従させる制御を一般にサーボ制御と呼びます。古典制御の PID 制御や I-PD 制御もサーボ制御の一種です。出力をステップ状の目標値に追従させる場合、制御ループの中に積分器を一つ含めます。図 A-2 は（PID 型）サーボ制御のブロック線図です。

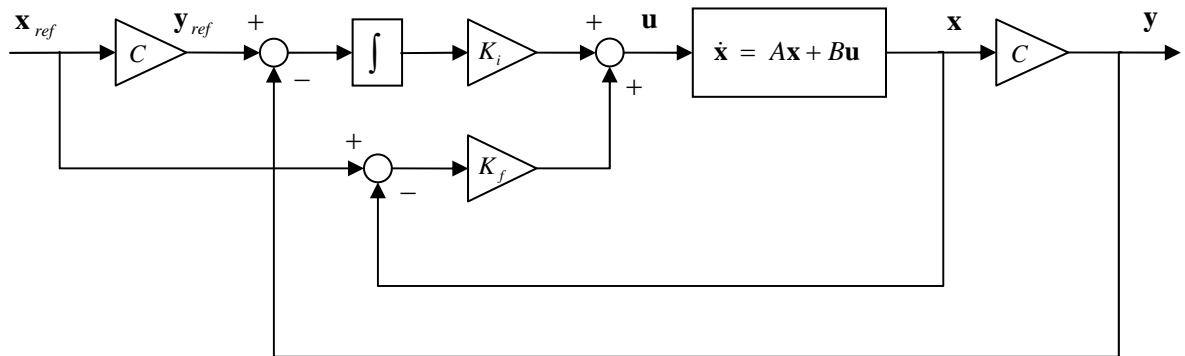


図 A-2 （PID 型）サーボ制御のブロック線図

観測出力と目標値の偏差を状態として加えた拡大系を考えることにより、状態フィードバック制御と同じ方法でサーボ制御のゲインを求めることができます。以下、その方法について説明します。

状態方程式および出力方程式が次式で与えられているとします。

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

観測出力と目標値の偏差を $\mathbf{e}(t) = \mathbf{C}(\mathbf{x}(t) - \mathbf{x}_{ref})$ 、その積分値を $\mathbf{z}(t)$ 、拡大系の状態を $\bar{\mathbf{x}}(t) = [\mathbf{x}(t), \mathbf{z}(t)]^T$ 、 \mathbf{x} と \mathbf{u} の次数をそれぞれ n 、 m とすると、拡大系の状態方程式は

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

となります。拡大系が安定な場合、十分な時間が経つと(A.7)式は以下の式に収束します。

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

$\mathbf{x}_e = \mathbf{x}(t) - \mathbf{x}(\infty)$ 、 $\mathbf{z}_e = \mathbf{z}(t) - \mathbf{z}(\infty)$ 、 $\mathbf{u}_e = \mathbf{u}(t) - \mathbf{u}(\infty)$ とすると、(A.7)式と(A.8)式の差が次の状態方程式になっていることが分かります。

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

この拡大系を安定化させるため、状態フィードバック制御を行います。すなわち、以下の入力を与えます。

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

$\mathbf{x}(\infty) \rightarrow \mathbf{x}_{ref}$ 、 $\mathbf{z}(\infty) \rightarrow 0$ 、 $\mathbf{u}(\infty) \rightarrow 0$ とすると、入力 $\mathbf{u}(t)$ は

$$\mathbf{u}(t) = -K_f (\mathbf{x}(t) - \mathbf{x}_{ref}) - K_i C \int (\mathbf{x}(t) - \mathbf{x}_{ref}) dt \quad (\text{A.11})$$

となります。

通常、制御理論のテキストでは現代制御理論に基づくサーボ制御として I-PD 型 ((A.11)式の第 1 項の \mathbf{x}_{ref} を 0 とした式) が紹介されています。ここでは、追従性改善のために PID 型 ((A.11)式の第 1 項の \mathbf{x}_{ref} を 0 にしない式) を採用しています。

付録 B 仮想空間

本付録では、NXTway-GS の仮想空間の座標系およびマップファイルについて説明します。また、壁との距離算出・衝突検知について説明します。

B.1 座標系

nxtway_gs_vr.mdl では NXTway-GS の 3D 表示に Virtual Reality Toolbox を使用しています。Virtual Reality Toolbox は 3D 表示に VRML 言語を使用しています。VRML 言語の座標系は図 B-1 のように定義されています。

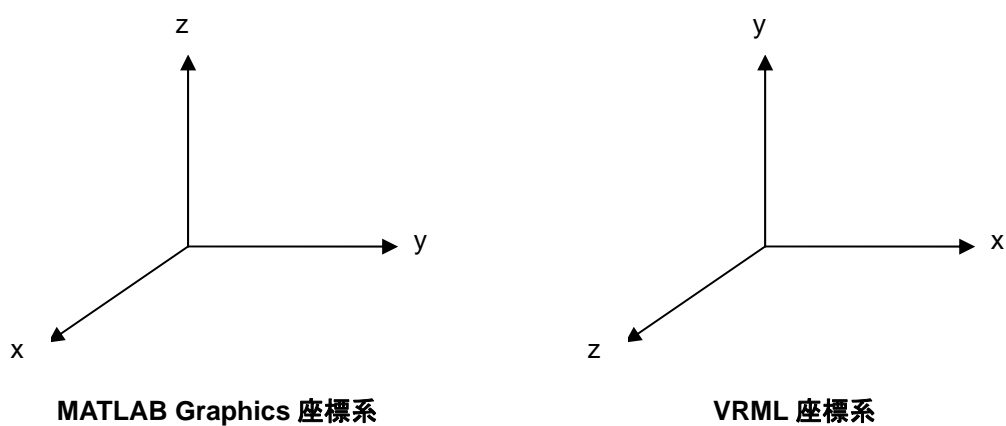


図 B-1 VRML 座標系

マップ用 VRML ファイル track.wrl の座標系を図 B-2 に示します。

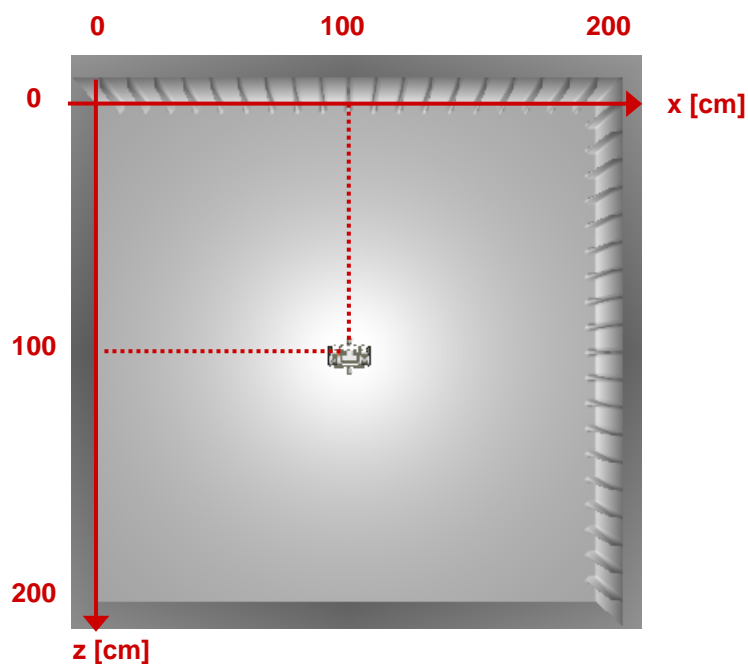


図 B-2 track.wrl の座標系

nxtway_gs.mdl / nxtway_gs_vr.mdl では、NXTway-GS / Sensor / Cal VRML Coordinate サブシステムで NXTway-GS の VRML 座標を計算しています。

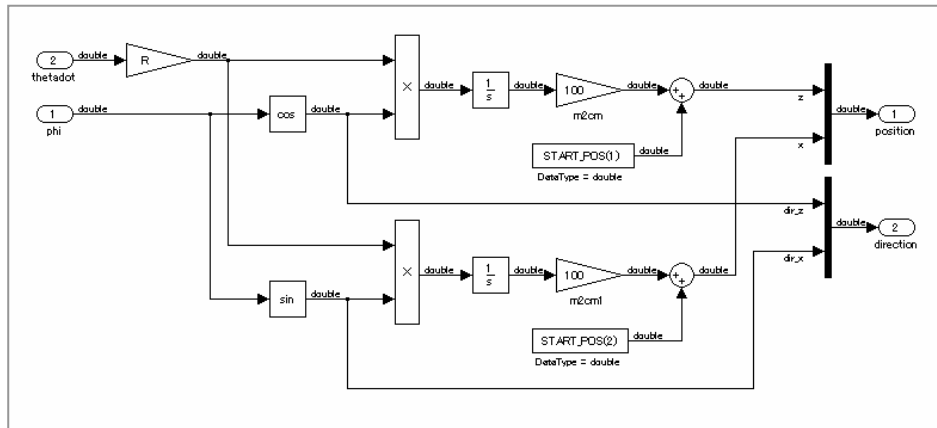


図 B-3 Cal VRML Coordinate サブシステム

B.2 マップファイルの作成

mywritevrtrack.m を用いると、画像ファイル track.bmp からマップ用 VRML ファイル track.wrl を作成することができます。track.bmp から track.wrl を作成するには、次のコマンドを入力します。

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

mywritevrtrack.m の画像→マップ変換規則は次の通りです。

- 1 pixel → 1×1 [cm²]
- 白色 pixel (RGB = [255, 255, 255]) → 床
- 灰色 pixel (RGB = [128, 128, 128]) → 壁 (デフォルトでは高さは 20 [cm])
- 黒色 pixel (RGB = [0, 0, 0]) → 黒線 (ライントレース用)

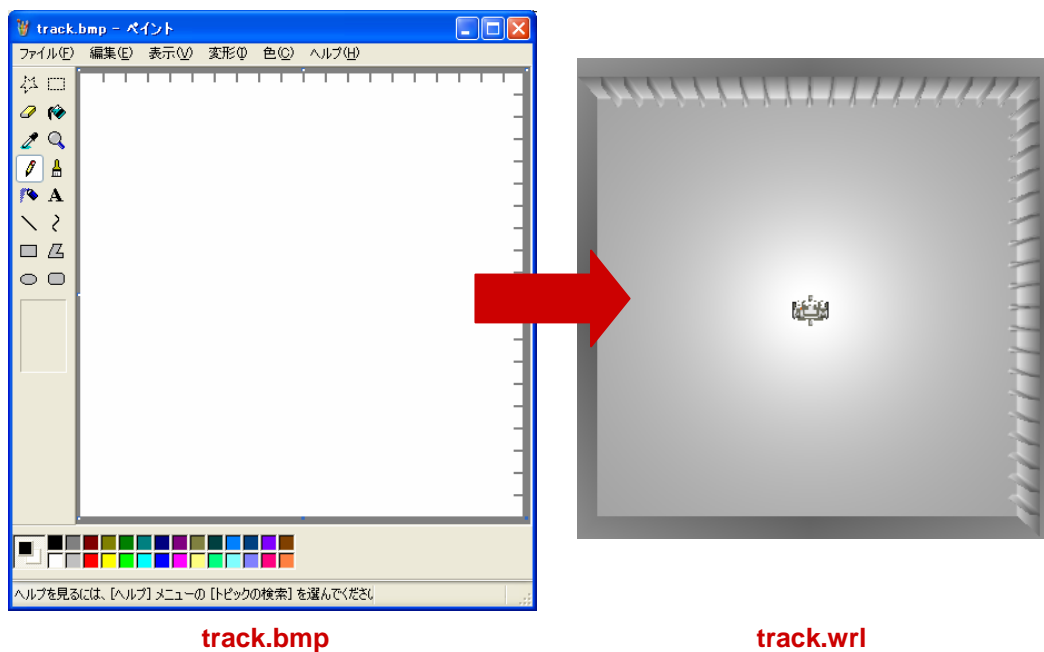


図 B-4 マップ用 VRML ファイルの作成

B.3 距離算出・衝突検知

NXTway-GS と壁との距離算出および衝突検知の計算は、Sensor サブシステム内の Embedded MATLAB Function ブロックで行われています。

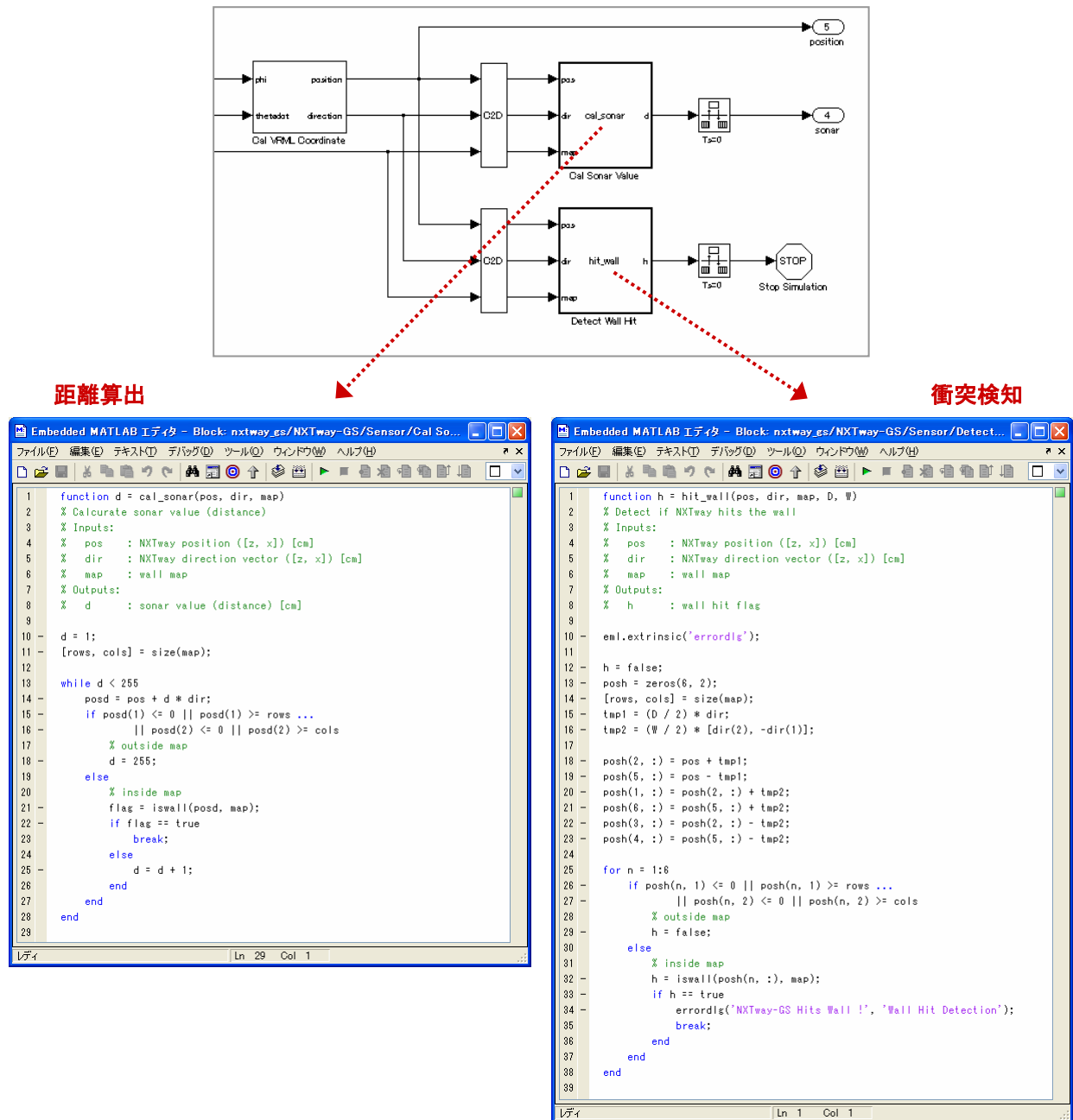


図 B-5 距離算出・衝突検知用 Embedded MATLAB Function ブロック

NXTway-GS と壁が衝突すると、下記エラーダイアログが表示されます。

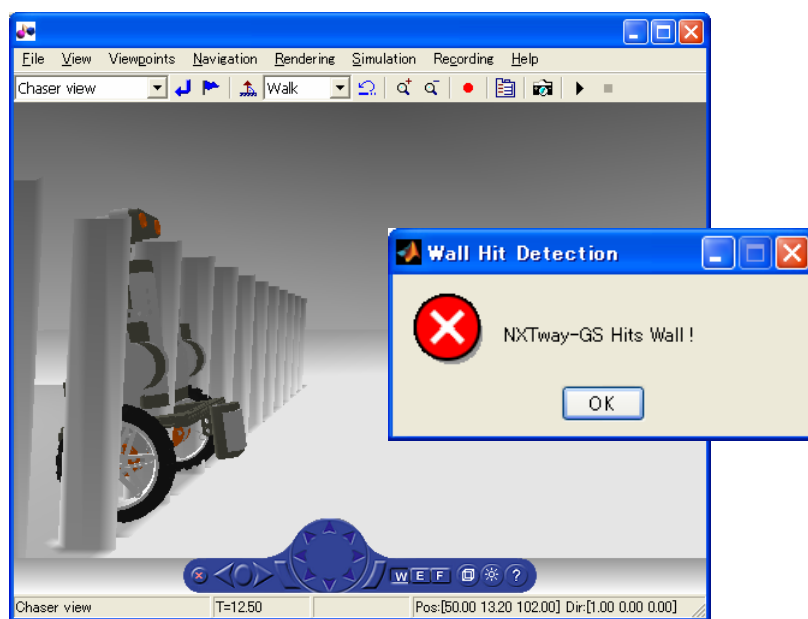


図 B-6 壁との衝突

付録 C モデル生成コード

nxtway_gs_controller.mdl から生成される C コードのうち、タスクを実装しているコードを紹介しま
す。RTW-EC では、Simulink データオブジェクトを用いてユーザ変数情報（変数名、記憶クラス、修
飾子等）を設定することができますが、下記コードではそのような設定を行っていません（つまり、
RTW-EC のデフォルト生成コード）。スペース節約のためコメントは省略しています。

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_tsl(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 +
9.9599999919E-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 + 8.0000000119E-001F *
    rtDWork.UnitDelay_DSTATE_hp;
rtb_Sum_b = (real32_T)ecrobot_get_gyro_sensor(NXT_PORT_S4);
rtDWork.gyro_offset = 9.9900000129E-001F * rtDWork.gyro_offset +
    1.0000000047E-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_g1 = (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_g1, 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)
{
}

```


参考文献

- [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] 島田明、大石潔、柴田昌明、市川修 EE テキスト モーションコントロール オーム社 2004
- [4] Fixed-Point Toolbox チュートリアル
<http://www.cybernet.co.jp/matlab/library/library/detail.php?id=TT043>
- [5] Simulink Fixed Point／RTW-ECを用いた固定小数点コード生成 チュートリアル
<http://www.cybernet.co.jp/matlab/library/library/detail.php?id=TT038>
- [6] Tips for Fixed-Point Modeling and Code Generation for Simulink 6
<http://www.mathworks.com/matlabcentral/fileexchange/7197>

MATLAB ヘルプ・情報リソース


■ 関数・コマンドの使用方法を調べる

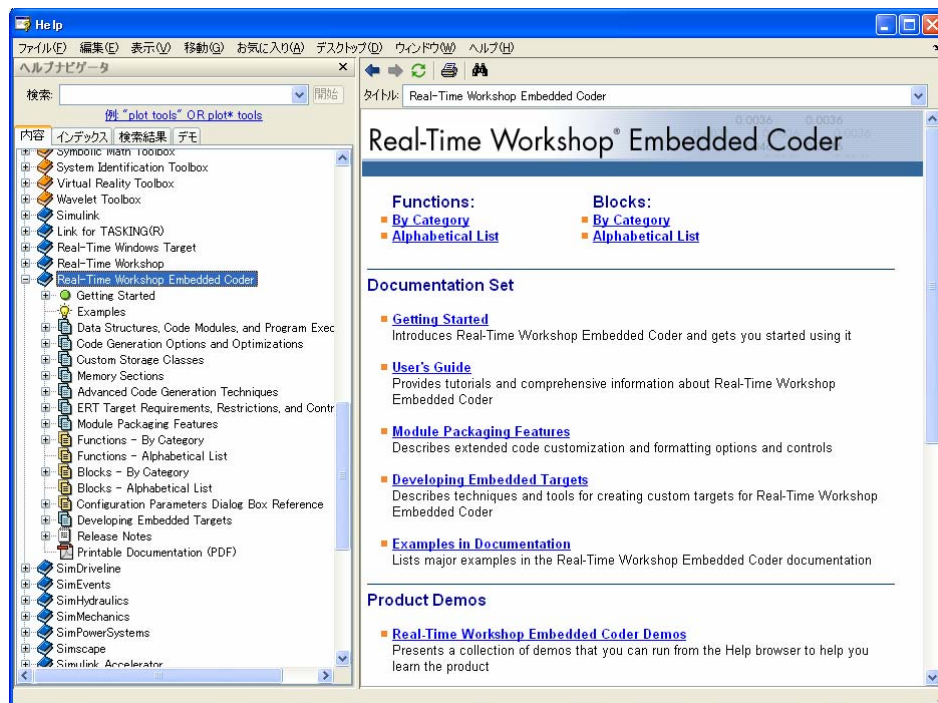
```
>> help 関数・コマンド名  
  
or  
  
>> doc 関数・コマンド名
```

■ 関数・コマンドの一覧リストを表示する

```
>> helpwin
```

■ MATLABヘルプブラウザ

MATLABの [ヘルプ] メニュー→ [MATLABヘルプ] を選択するか、 アイコンをクリックすることによりヘルプブラウザを開くことができます。ヘルプブラウザからはMATLABマニュアルやデモを参照することができます。



■ 弊社MATLABホームページ

<http://www.cybernet.co.jp/matlab/>

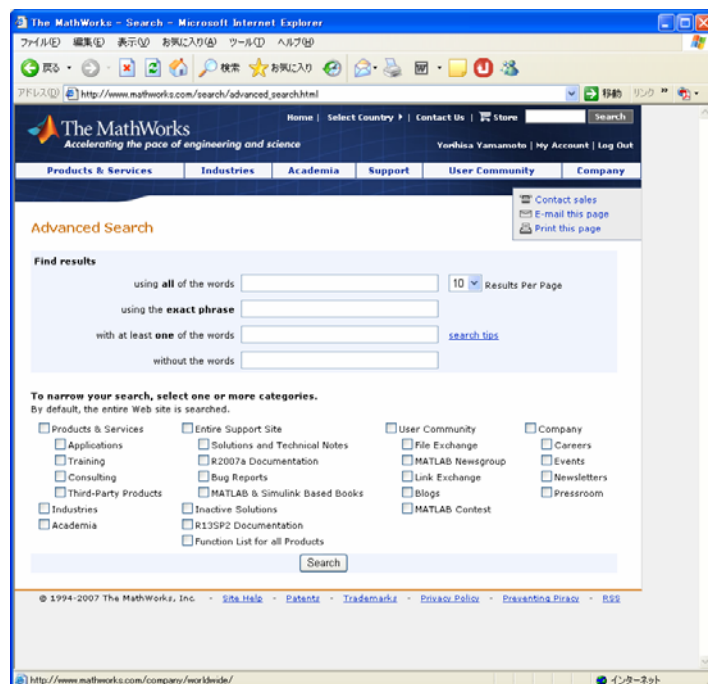
製品情報・日本語ドキュメント・FAQ・技術資料等を掲載しています。



■ 開発元ソリューション検索ページ

http://www.mathworks.com/search/advanced_search.html

MATLABに関する最新の情報を検索することができます。



■ 著作権

本資料の一部あるいは全部を無断で転載、複製、複写されると著作権等の権利侵害となる場合がありますのでご注意ください。

■ 商標の帰属

MATLAB®は米国The MathWorks,Inc.の登録商標です。また、LEGO®はレゴ社の登録商標です。その他、本資料に記載されている製品名とブランド名は、それぞれ該当する各社の商標または登録商標です。

■ 免責事項

本資料に掲載されている操作の影響につきましては、弊社では責任を負いかねますので予めご了承ください。

■ 問合せ先

本資料の内容についてお気づきの点がありましたら、弊社までご連絡頂ければ幸いです。

NXTway-GS のモデルベース開発

2008 年 3 月作成

～LEGO Mindstorms NXT を用いた二輪型倒立振子ロボットの制御～

CYBERNET
サイバネットシステム株式会社

応用システム第 1 事業部 <http://www.cybernet.co.jp/MATLAB>
応用システム第 1 事業部 営業部 E-mail: infomatlab@cybernet.co.jp
応用システム第 1 事業部 技術部 E-mail: techmatlab@cybernet.co.jp

| | | | | | | |
|---------|-----------|------------------|-------------|-----------------------|------------------|-------------------|
| 本 社 | 〒101-0022 | 東京都千代田区神田練堀町 3 | 富士ソフトビル14 階 | Tel: 03-5297-3565(営業) | 03-5297-3546(技術) | Fax: 03-5297-3648 |
| 中 部 支 社 | 〒460-0003 | 愛知県古屋市中区錦 1-6-26 | 富士ソフトビル 3 階 | Tel: 052-219-5197(営業) | 052-219-5198(技術) | Fax: 052-219-5970 |
| 西日本支社 | 〒542-0028 | 大阪市中央区常盤町 1-3-8 | 中央大通 FN ビル | Tel: 06-6940-3613(営業) | 06-6940-3611(技術) | Fax: 06-6940-3602 |
