開発環境・要素技術セミナー

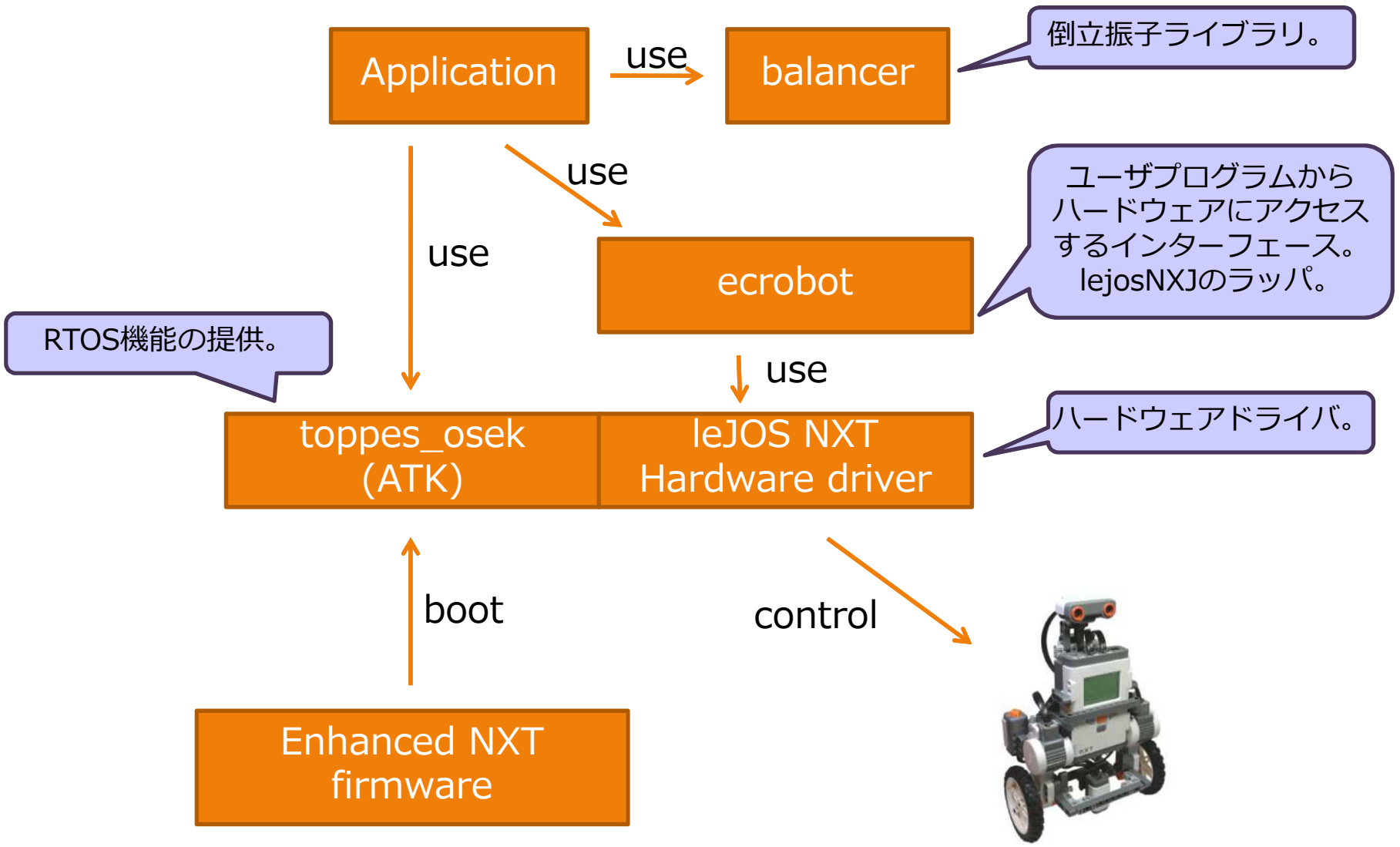
実行環境を理解する

目次

- ソフトウェア
  - 制御API:ecrobot
  - 倒立振子ライブラリ:balancer
  - OS(RTOS):OSEK
- ハードウェアの特性
  - ジャイロセンサ、光センサ、超音波センサ
  - モータ
- よくあるテクニック紹介
- チャンピオンシップ大会要素技術ハイライト

ソフトウェア

nxtOSEKのコンポーネントと役割



nxtOSEK注意点

- makeのバージョン
  - nxtOSEKサイトにあるCygwinのインストール手順では、makeのバージョンを3.81-1で選択するように記述されていますが、最新のCygwinではこのバージョンを選択できないようです。3.82-90でも動作の確認が取れていますので、こちらを選択してください。
- 拡張ファームウェア
  - nxtOSEKサイトの図では lms_arm_nbcnxc_106.rfw になっていますが、現在このバージョンは公開されていません。現在公開されている、lms_arm_nbcnxc_107.rfw、 lms_arm_nbcnxc_128.rfwどちらのバージョンを使用しても問題ないことは確認が取れています。

どちらの内容も、nxtOSEKサイトで補足説明が追記されています。

nxtOSEK注意点

- Makefile
  - NXTOSEK_ROOTマクロに絶対パスを書くとビルドできません。

    NG： NXTOSEK_ROOT = /home/etrobo/nxtOSEK

    OK： NXTOSEK_ROOT = ../../../nxtOSEK

  - Windows 7(64ビット) Home Premium SP1の環境でmakeを実行すると、以下のようなエラーが出るとの報告があります。

```
Generating OSEK kernel config files from main.oil
sg: error: cannot open file
`/usr/local/nxtOSEK/ecrobot/../toppers_osek/sg/lego_nxt.sgt'
/usr/local/nxtOSEK/ecrobot/ecrobot.mak:317: recipe for target `kernel_cfg.c'
failed
make: *** [kernel_cfg.c] Error 1
```

    報告によると、~nxtOSEK/ecrobot/ecrobot.mak の 319 行目あたりを以下のように修正するとmakeが成功するとのことです。

```
-os=ECC2 -I`cygpath -w $(TOPPERS_OSEK_ROOT_SG)/sg/impl_oil` ¥
-template=`cygpath -w $(TOPPERS_OSEK_ROOT_SG)/sg/lego_nxt.sgt`
```

nxtOSEK注意点

- nxtOSEKにsg.exeは同梱されないようになっています。

$(NXTOSEK_ROOT)/sample_c/helloworldをコンパイルしたときに、

```
$ make all
...(省略
Generating OSEK kernel config files from ./helloworld.oil
/bin/sh: ../../ecrobot/../toppers_osek/sg/sg: No such file or directory
make: *** [kernel_cfg.c] エラー 127
```

のようなエラーになるときは、http://lejos-osek.sourceforge.net/download.htm
を参照してsg.exeをインストールしてください。

お知らせ：nxtOSEKからのsg.exeの削除について(2010/03/01)

Sourceforge.netからの指摘を受け、nxtOSEKに従来含まれていたsg.exeバイナリファイル(OSEK OILパーサー/コード生成ツール)を削除することにしました。ただし、nxtOSEKに含まれていたsg.exeは、TOPPERSプロジェクトで公開されているTOPPERS/OSEK 1.1のものを使用しており、次のリンクからダウンロードすることができます：osek_os-1.1.lzh

TOPPERS/OSEK 1.1のsg.exeをnxtOSEKで使用する手順は次の通りです：
- ダウンロードしたosek-os-1.1.lzhを解凍する
- 解凍した toppers_osek/sg/sg.exe を nxtOSEK/toppers_osek/sg ディレクトリにコピーする

nxtOSEK注意点

- フック関数
  - ecrobot_device_initialize()は、起動後RUN(右ボタン押下)されるまで、何度も呼ばれます。
  - ecrobot_device_terminate()は一度だけ呼ばれます。

- ecrobot APIの中で、スリープしている箇所があります。

```
$ grep systick_wait_ms $(NXTOSEK_ROOT)/ecrobot/c/* | wc –l 26
```

制御API：ecrobot

ecrobot API

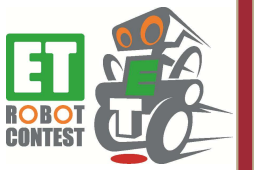
- モータ
- タッチセンサ
- 超音波センサ
- 光センサ
- LCD
- ジャイロセンサ
- Bluetooth

詳しくは [ecrobot公式APIリファレンス]

http://lejos-osek.sourceforge.net/jp/api.htm

を参照してください。

ecrobot API(Bluetooth接続)



- 接続形態によって２種類のAPIのうち１つを使うことで接続できます。
- 繋ってしまえばマスタでもスレーブでも同じです。

NXTをマスタとして初期化する場合

```
ecrobot_init_bt_master(const U8 *bd_addr, const CHAR *pin)
```

NXTをスレーブとして初期化する場合

```
ecrobot_init_bt_slave(const CHAR *pin)
```

ecrobot API(Bluetooth接続)

- APIリファレンスを読むと、
  - このAPIはループ処理(例、ecrobot_device_initialize またはバックグラウンドタスク)の中で実行してください。
- 確実な接続を担保するには、
  - ecrobot_get_bt_status()の返り値がBT_STREAMになるまで、繰り返し呼び出さないといけません。

ecrobot API(Bluetooth送信)

- データ送信用のAPIを使います。

```
ecrobot_send_bt(const void* buf, U32 off, U32 len)
```

NXTからデータを送信する例

```
char buf[] = "Hello, World!!";
ecrobot_send_bt(buf, 0, strlen(buf) + 1);
```

```
struct Log {
    int time;
    int light;
} log;

log.time = ecrobot_get_systick_ms();
log.light = ecrobot_get_light_sensor(NXT_PORT_S1);
ecrobot_send_bt(&log, 0, sizeof(struct Log));
```

ecrobot API(Bluetooth受信)

- データ受信用のAPIを使います。

```
ecrobot_read_bt(void* buf, U32 off, U32 len)
```

NXTでデータを受信する例

```
struct Command {
    int command_type;
    int value;
};

Command com;
U32 len = 0;
len = ecrobot_read_bt(&com, 0, sizeof(struct Command));
if (len == 0) {
    // 受信データなし
}
```

ecrobot API(Bluetooth通信)

- 注意
  - Windows側のBluetoothスタックによっては、動かない、データが化けるといった症状が出ることがあります。

倒立振子ライブラリ:balancer

レゴ倒立振子ロボットの歴史

LegWay
Steve Hassenplug

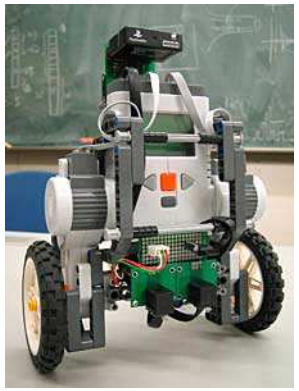
NXTway-DT
Takashi Chikamasa

NXTway-GS
Yorihisa Yamamoto

- 当初は光センサによる倒立制御
- ジャイロ販売に伴いジャイロが主流に
- NXTway-GSをベースにNXT走行体を作成

2006 → 2007 → 2008 → 2009 → 2010 → 2011

NXTway
by Philo



NXTway-G
by Ryo Watanabe

NXTway-ETの
倒立振子制御アルゴリズムは
NXTway-GSと同じです

倒立振子制御

- 倒立振子はもちろん不安定。どうやって安定化（倒立）させる？

一輪車

頭
目・体から得た傾斜角度を元に足に指令を出す

コンピュータに相当

目・体
傾斜角度を検出する

センサに相当

足
倒れる方向に車輪をこぐ

アクチュエータに相当

人間は目・体からの情報を元に傾斜角度を推定、体の挙動を予測しながら最適な指令値を足に送って車輪をこぐことでバランスをとっている

倒立振子制御

- 倒立振子はもちろん不安定。どうやって安定化（倒立）させる？

NXT Brick（コンピュータ）
ジャイロセンサから得た傾斜角速度
等を元にモータに指令を出す

ジャイロセンサ（センサ）
傾斜角速度を検出する

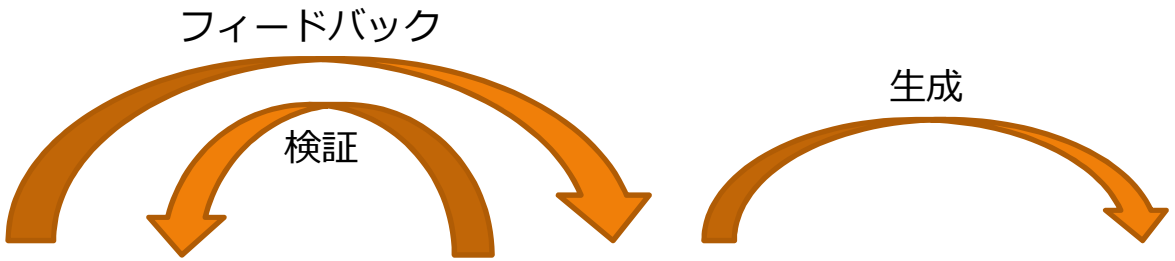
モータ（アクチュエータ）
モータ回転角度を検出しながら、
倒れる方向に車輪を回転させる

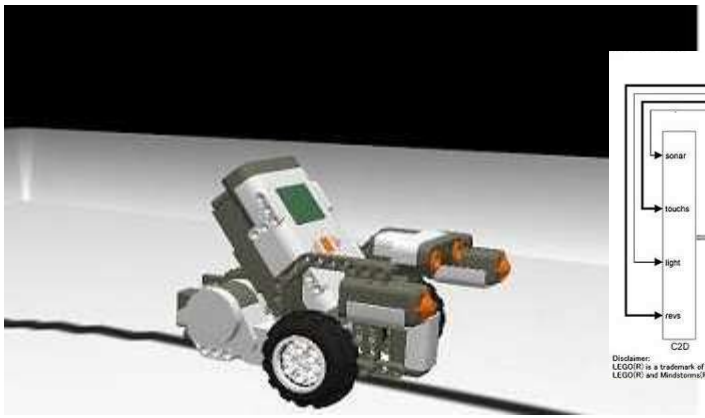
ジャイロセンサから傾斜角度を計算し、最適な指令値をモータに送って車輪を回転させることでバランスをとればよい

どうやって最適なモータ回転量を計算する？

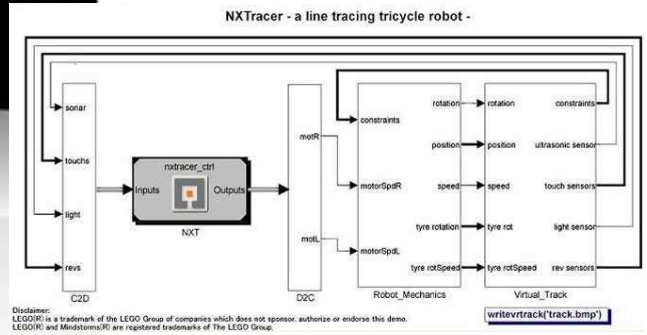
倒立振子ライブラリ

- 制御モデルから自動生成されたライブラリ(関数)





シミュレーション



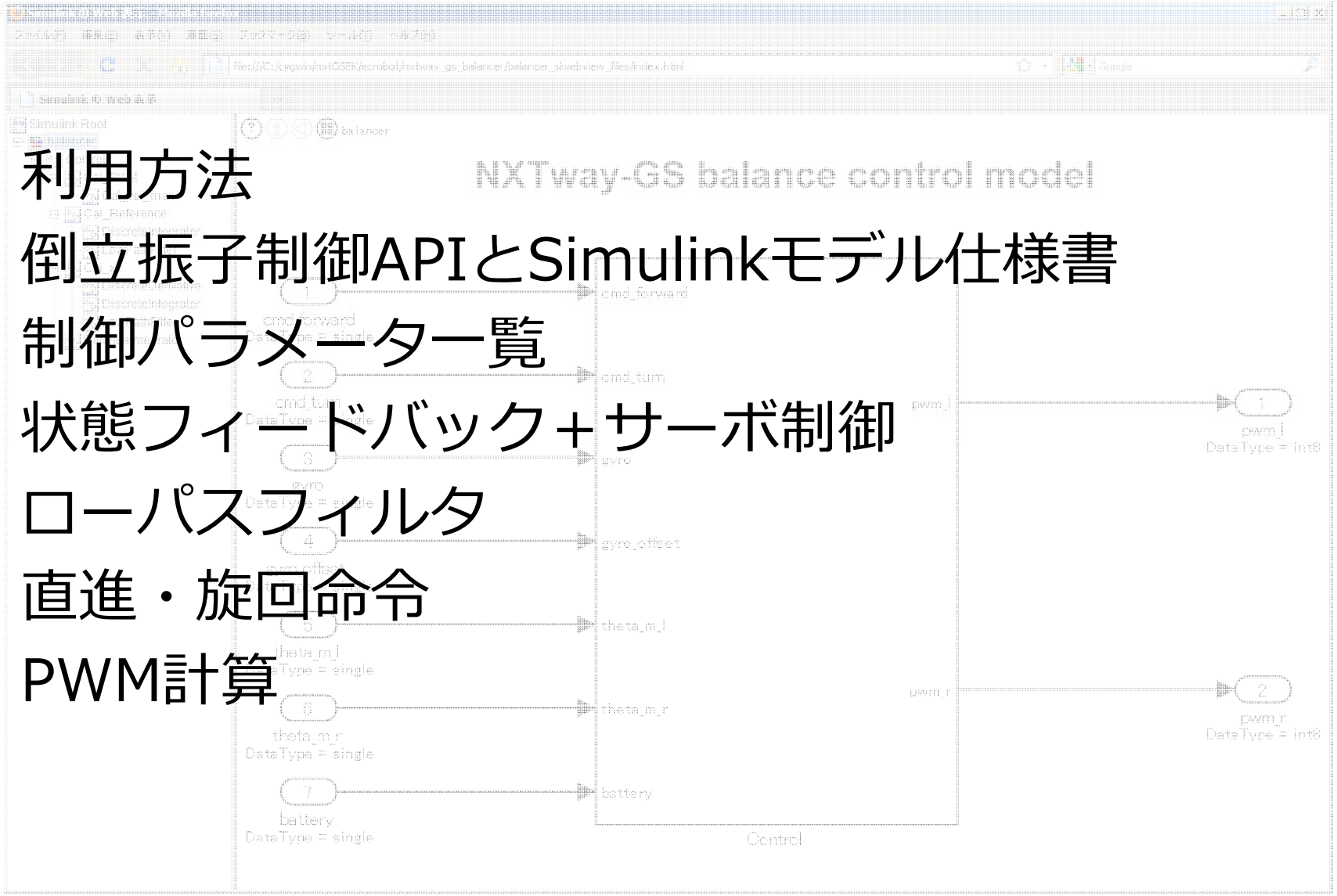
制御モデル

```
void
balance_control(
    F32 args_cmd_forward,
    F32 args_cmd_turn,
    F32 args_gyro,
    F32 args_gyro_offset,
    F32 args_theta_m_l,
    F32 args_theta_m_r,
    F32 args_battery,
    S8 *ret_pwm_l,
    S8 *ret_pwm_r)
```

自動生成

※nxtOSEKには、Webブラウザで確認できる倒立振子ライブラリのモデルも同梱されています。

倒立振子制御API 解説



- 利用方法
- 倒立振子制御APIとSimulinkモデル仕様書
- 制御パラメーター覧
- 状態フィードバック＋サーボ制御
- ローパスフィルタ
- 直進・旋回命令
- PWM計算

倒立振子制御 C API利用方法

- balance_init()で初期化後、balance_control()を呼び出し、前進量・旋回量から左右モータの制御量を算出させます。
- balance_control()は、4msec周期で呼び出す必要があることに注意。
- →倒立振子制御 C API解説

制御量算出

```
void
balance_control(
    F32 args_cmd_forward,
    F32 args_cmd_turn,
    F32 args_gyro,
    F32 args_gyro_offset,
    F32 args_theta_m_l,
    F32 args_theta_m_r,
    F32 args_battery,
    S8 *ret_pwm_l,
    S8 *ret_pwm_r)
```

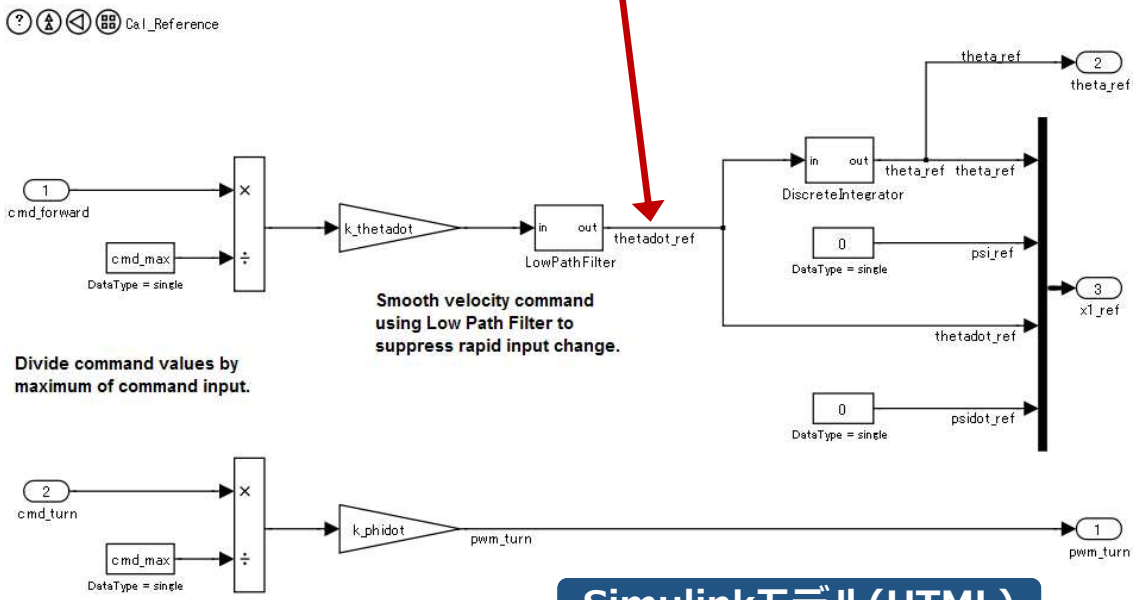
初期化

```
void
balance_init(void)
```

倒立振子制御APIとSimulinkモデル仕様書

倒立振子制御APIとSimulinkモデル仕様書を
比較することによりアルゴリズムを把握します

```
197      *  Product: '<S8>/Product'
198      *  Sum: '<S8>/Sum1'
199      *  UnitDelay: '<S8>/Unit Delay'
200      */
201     tmp_thetadot_cmd_lpf = (((args_cmd_forward / CMD_MAX) * K_THETADOT) * (1.0F
202       - A_R)) + (A_R * ud_thetadot_cmd_lpf);
203
204     /* Gain: '<S4>/Gain' incorporates:
205      *  Gain: '<S4>/deg2rad'
206      *  Gain: '<S4>/deg2rad1'
```



Simulinkモデル(HTML)

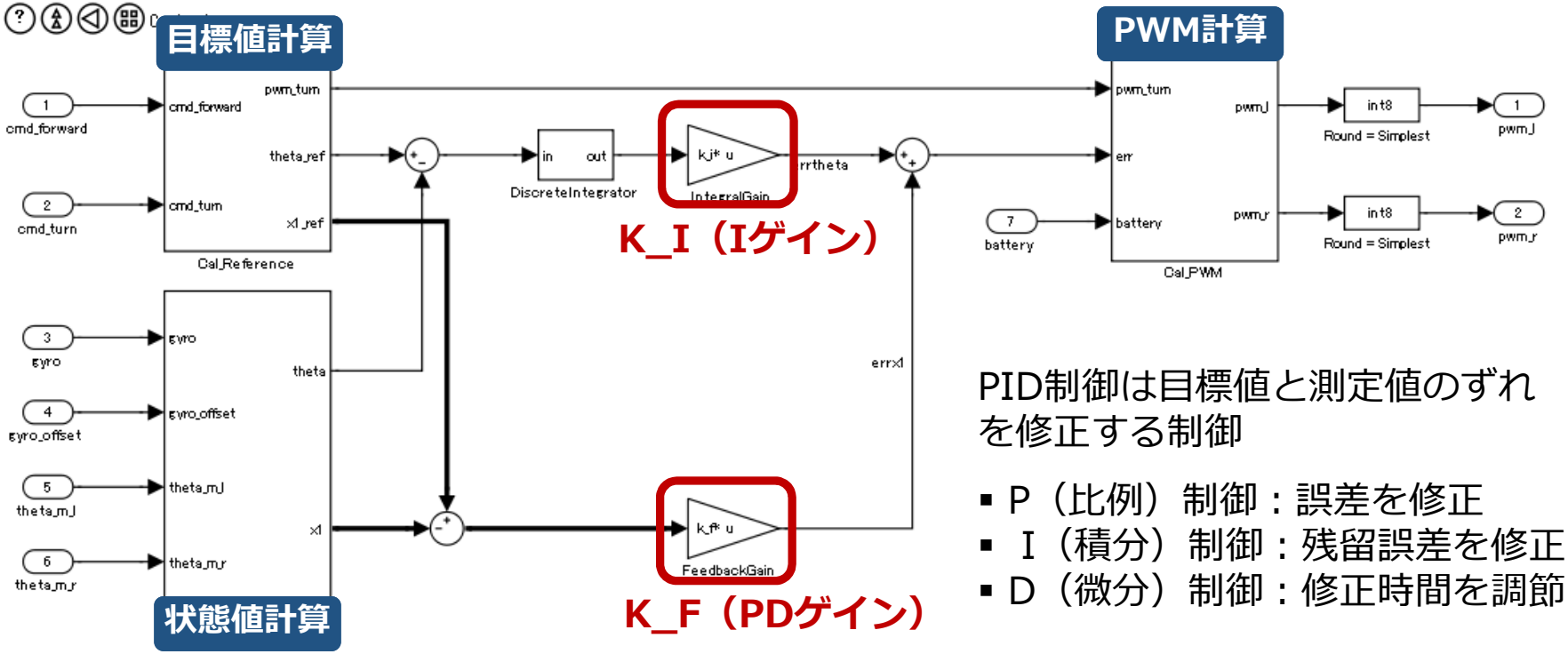
※nxtOSEK¥ecrobot¥nxtway_gs_balancer¥balancer_slwebview.html

制御パラメーター一覧@balancer_param.c

制御パラメータ	説明
A_D	左右車輪の平均回転角速度用ローパスフィルタ係数
A_R	左右車輪の目標平均回転角度用ローパスフィルタ係数
K_F[4]	状態フィードバック係数（PID制御のPD係数に相当） K_F[0] ： 左右車輪の平均回転角度係数 K_F[1] ： 車体傾斜角度係数 K_F[2] ： 左右車輪の平均回転角速度係数 K_F[3] ： 車体傾斜角速度係数
K_I	サーボ制御係数（PID制御のI係数に相当）
K_PHIDOT	目標旋回角速度係数（旋回方向の命令係数）
K_THETADOT	目標平均角速度係数（直進方向の命令係数）
BATTERY_GAIN	PWM計算用電圧係数
BATTERY_OFFSET	PWM計算用電圧オフセット

状態フィードバック＋サーボ制御

- K_F, K_IはPID制御のPIDゲインに相当する
- 各ゲインを適当に調節することにより安定性・追従性が増す
- 車輪平均角度と傾斜角度の制御特性はトレードオフの関係にある
- 角速度に対するゲインをあまり大きくしないほうがいい（速度ノイズに過敏になる）

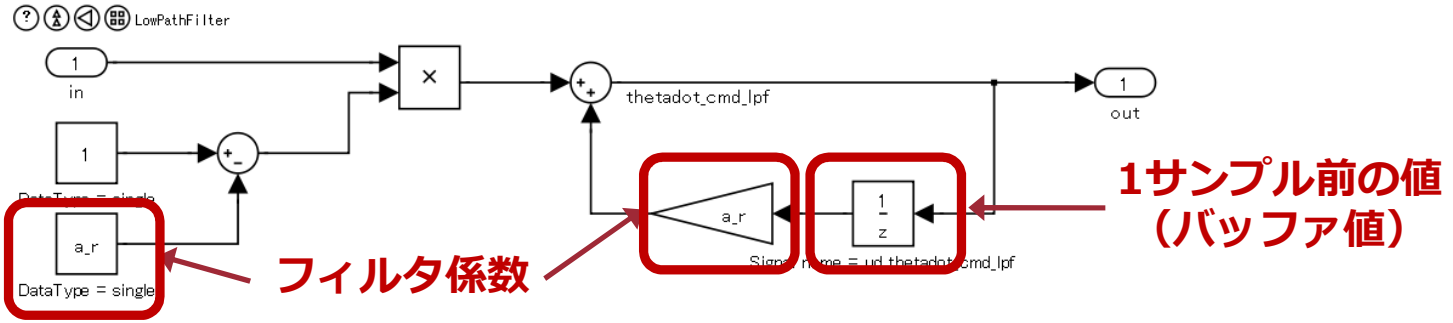


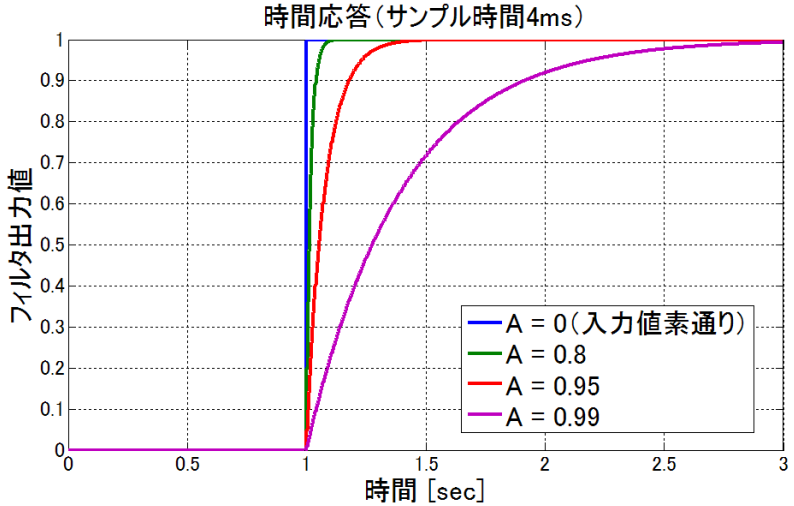
PID制御は目標値と測定値のずれを修正する制御

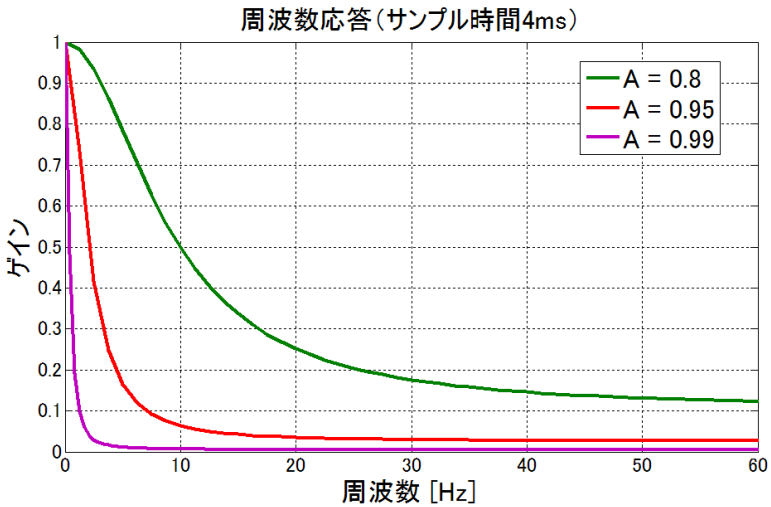
- P（比例）制御：誤差を修正
- I（積分）制御：残留誤差を修正
- D（微分）制御：修正時間を調節

ローパスフィルタ

- A_Dは数値微分により発生した角速度ノイズを除去するためのローパスフィルタ係数
- A_Rは目標角度指令値の急激な変化を滑らかにするためのローパスフィルタ係数
- 係数を大きくする程カットオフ周波数は小さくなるが、速応性が悪くなる
- 同じフィルタ係数でもサンプル時間（タスク周期）が異なるとフィルタ性能は異なる

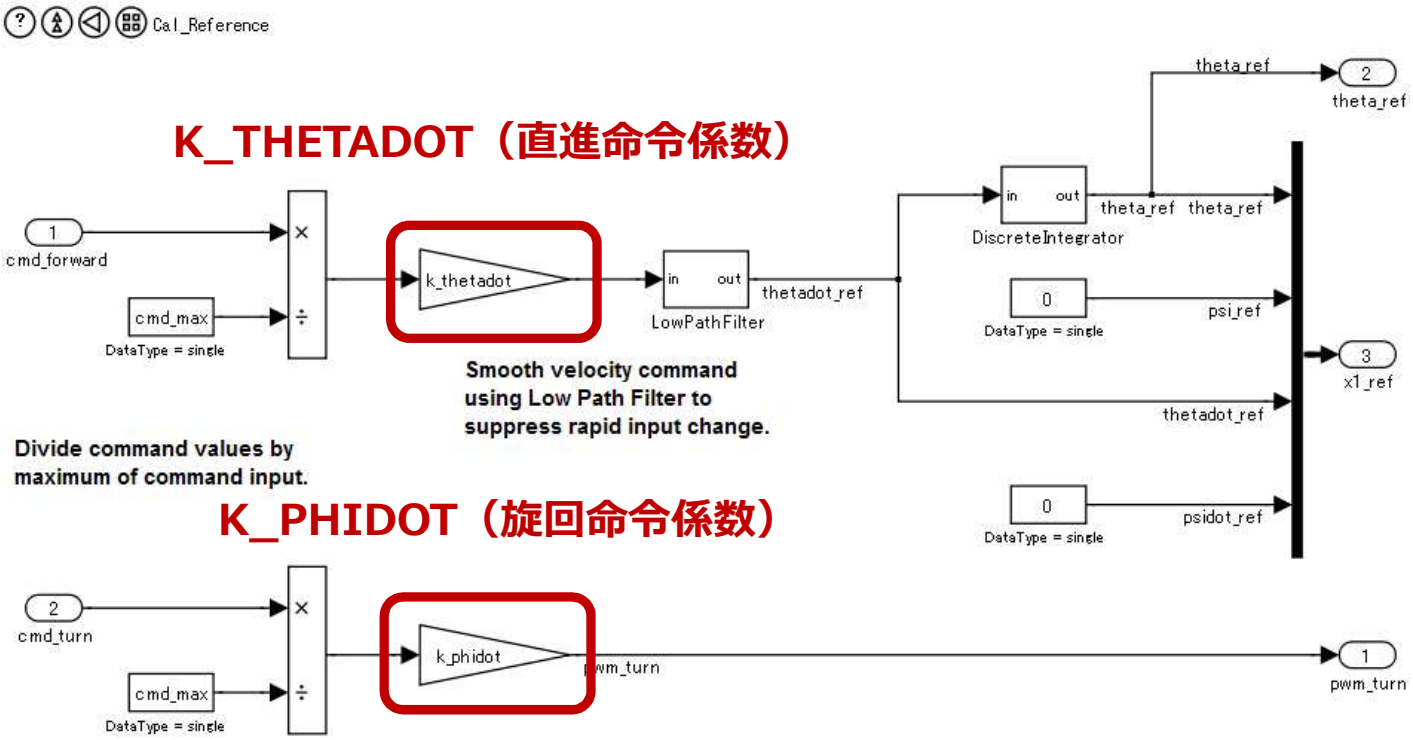






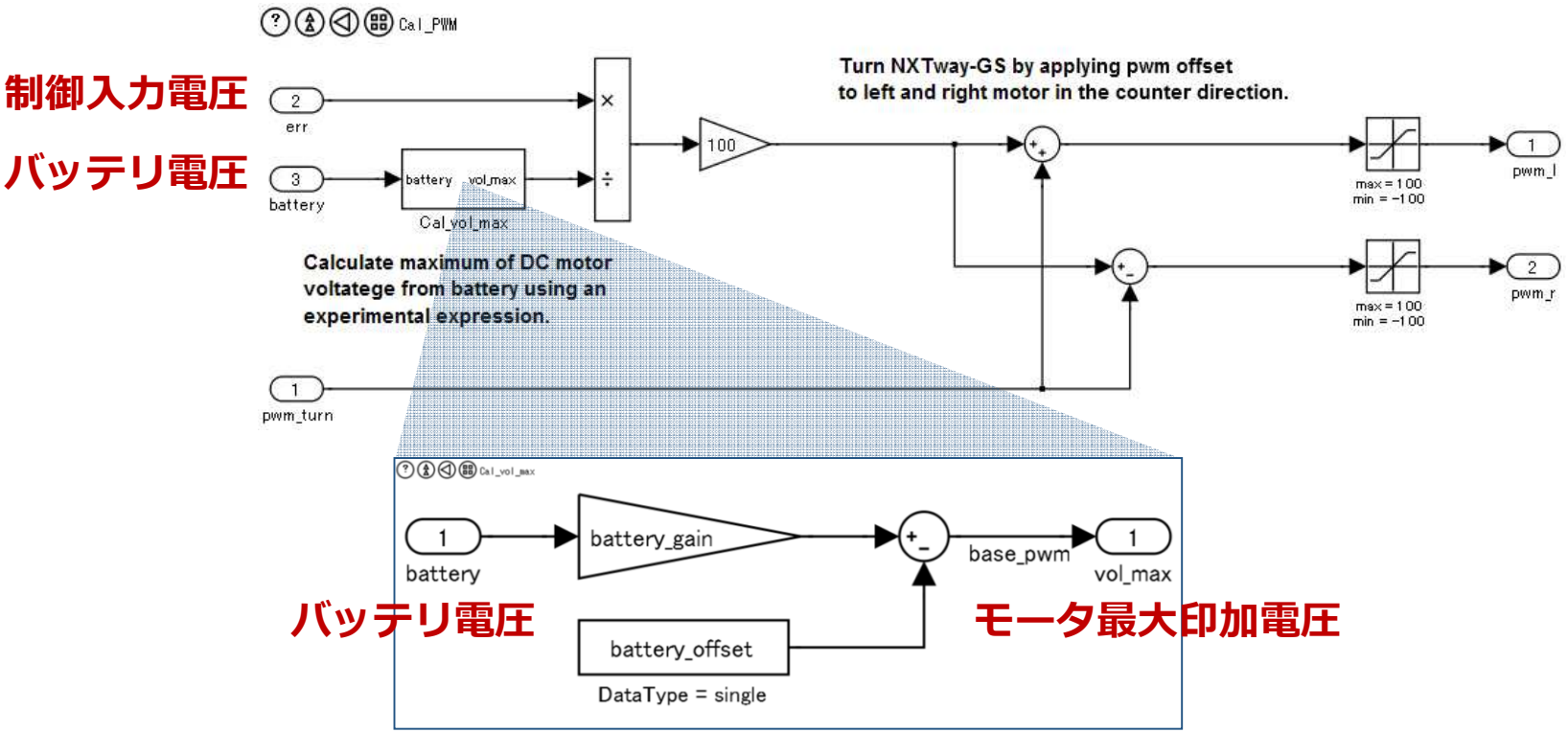
直進・旋回命令

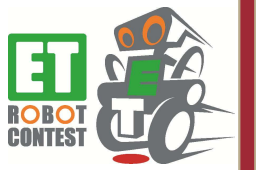
- K_THETADOTは直進速度に対する命令係数
- K_PHIDOTは旋回速度に対する命令係数
- 命令係数を上げると速度が向上するが、上げすぎると不安定になる
- PWMを100%以上にすることはできないので、自ずと係数の上限は決まる



PWM計算

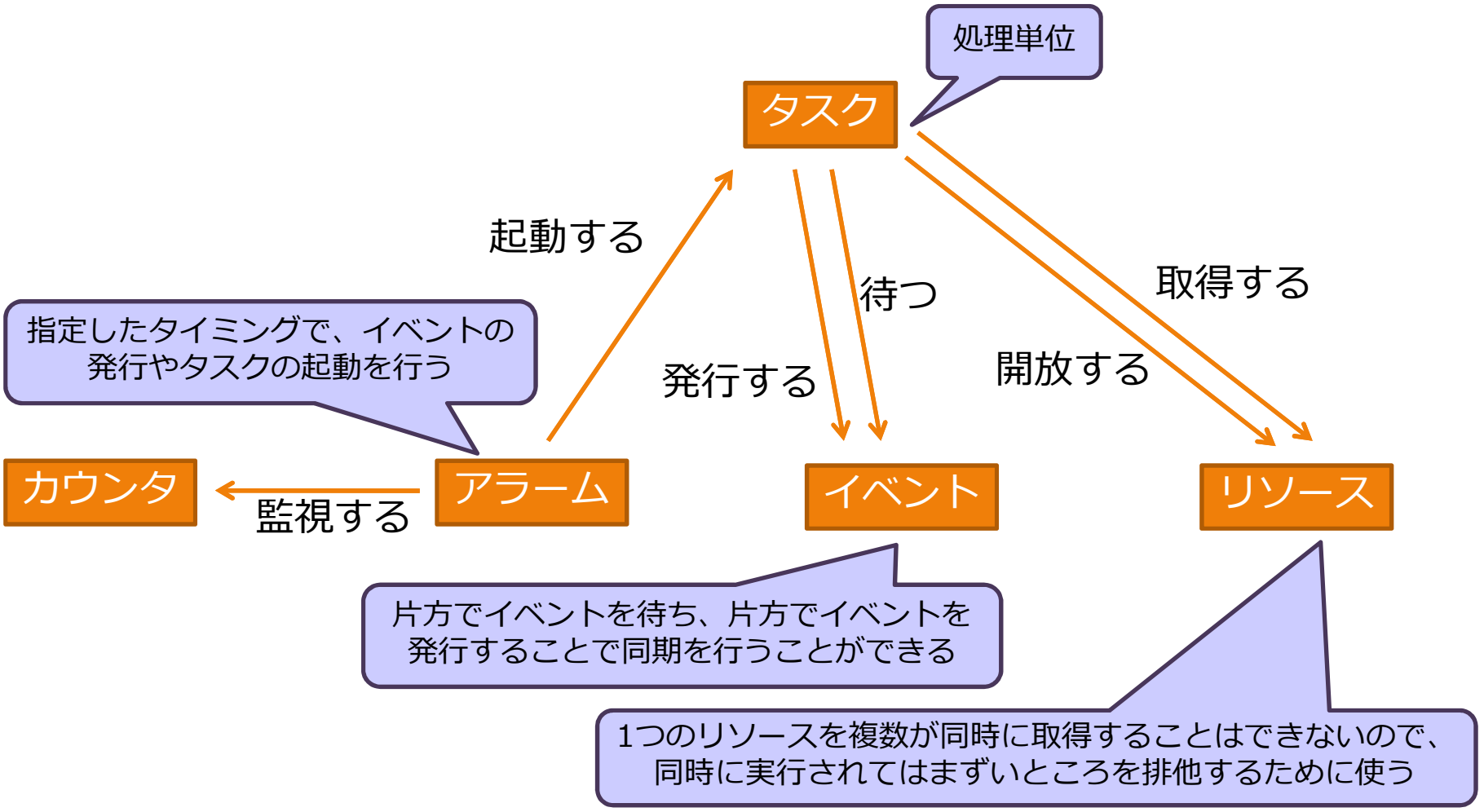
- 制御入力電圧をモータ最大印加電圧で割ってPWM値を計算
- モータ最大印加電圧をバッテリ電圧から計算
- BATTERY_GAIN、BATTERY_OFFSETのデフォルト値は実験データから同定
- デフォルト値を変更する必要は特に無し
- 旋回用PWM値を偶力として与えた場合は、前進/バランス制御から独立して扱える。ただしモータトルク (= 直進トルク + 旋回トルク + バランストルク)は有限であることを考慮する必要がある





OS(RTOS):OSEK

OSEK概要



※おおよその関連です。これ以外の関連もあります。

代表的なシステムコール

- タスク関係
  - ActivateTask(), TerminateTask(), ChainTask()
- イベント関係
  - SetEvent(), WaitEvent()
- リソース関係
  - GetResource(), ReleaseResource()
- アラーム関係
  - SetRelAlarm(), SetAbsAlarm(), CancelAlarm()

OIL

- システムの構成を静的に定義するためのファイル。
  - システムの構成：タスク、カウンタ、アラーム…
  - 静的：ビルド時に構成を決定します。
- 例えば、どんなことを書くか。
  - どんなイベント/リソース/カウンタを使うか
  - タスク：優先度はどのくらいか、どんなリソースを使うか、どんなイベントを受信できるか。
  - アラームが満了したときに何をするか。

詳しくは

[nxtOSEKのサンプル]
http://lejos-osek.sourceforge.net/jp/rms.htm
http://lejos-osek.sourceforge.net/jp/eds.htm
http://lejos-osek.sourceforge.net/jp/resource.htm

ハードウェアの特性

走行体に繋っているもの

超音波センサ

タッチセンサ

※本資料では扱いません

ジャイロセンサ

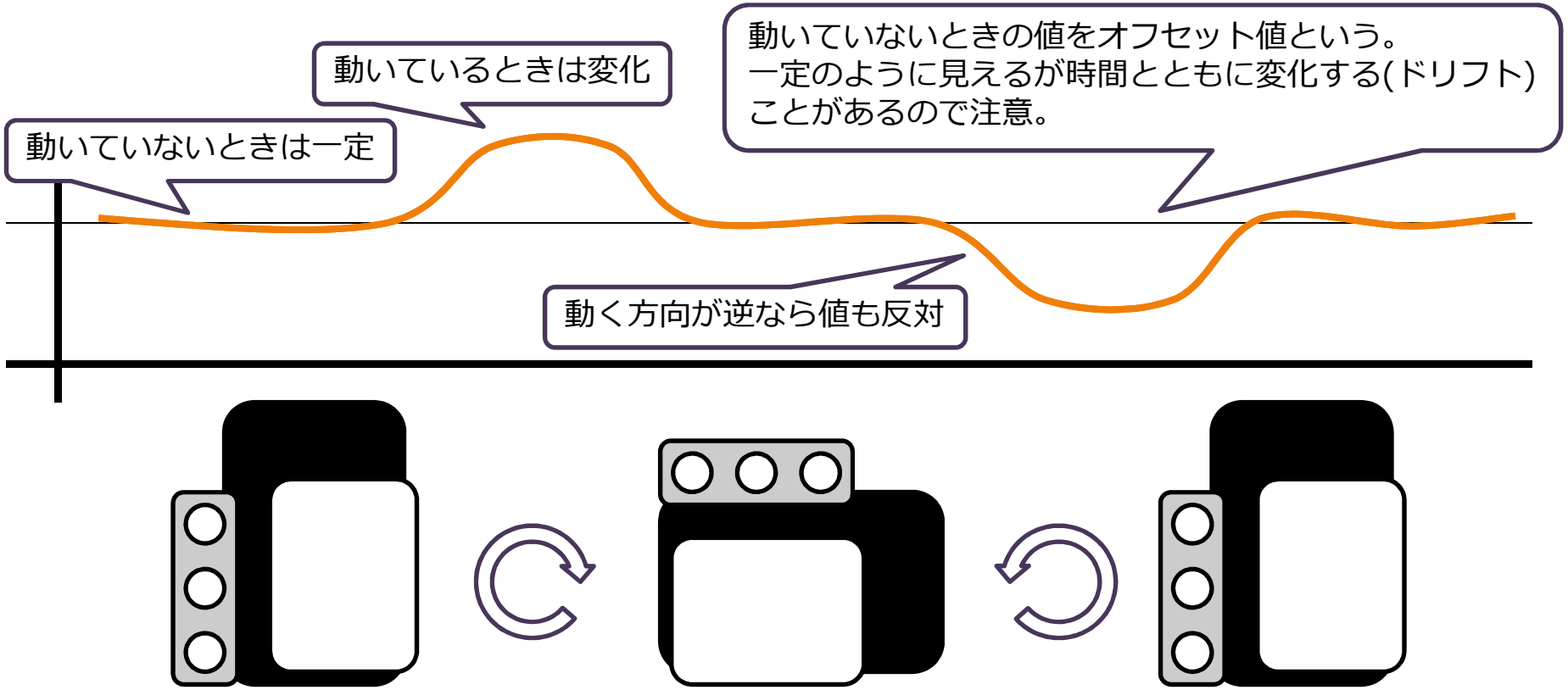
モータ

※エンコーダ付き

光センサ

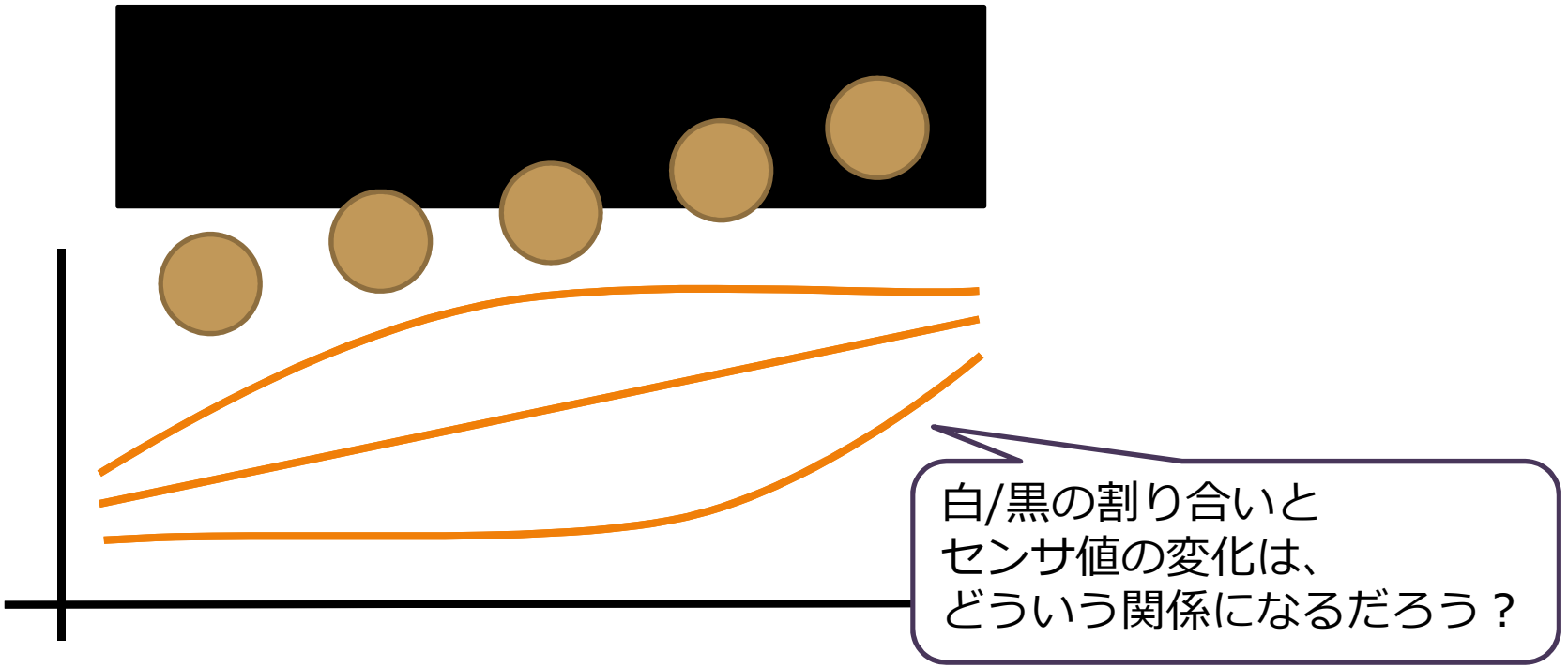
ジャイロセンサ

- センサ自身の**角速度**を計測するセンサ
  - センサの置かれている角度を計るものではありません。
  - ある時から、ある時までの動きを検出します。



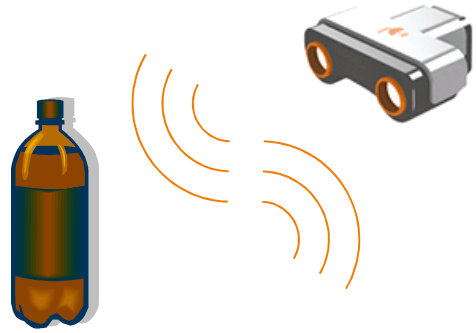
光センサ

- 周囲の明るさを検出する。
- 明るさとセンサ値の関係
  - 明るさとセンサ値の関係はリニア(単純な比例関係)ではない。



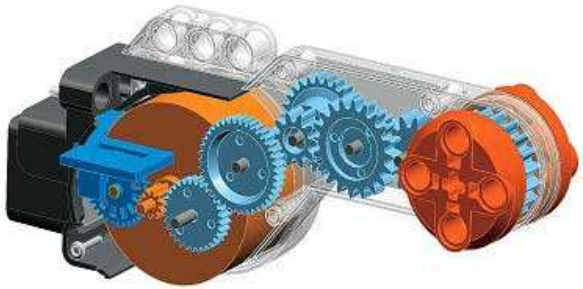
超音波センサ

- 超音波センサは、片方から超音波を発射、もう片方で超音波を検知する。
- 発射から検知までの時間と、音速を使うことで対象物までの距離を測定することができる。
- 減衰までの時間
  - 超音波は反響するため、測距の間隔が短いと誤検知することがある。
- 指向性
  - 検知するものの形状によっては、超音波が反射しなかったり、別の方向へ反射してしまい、測距できないことがある。



モータ

- トルク
  - 速く回すと高トルク、遅く回すと低トルク
  - 坂道を上りきるには？
- エンコーダ
  - 分解能1度のエンコーダが内蔵
  - ギヤがあるため精度はよくない。つまり、確実に１度の角度を検知することができることはない。



よくあるテクニック紹介

定番化している戦略

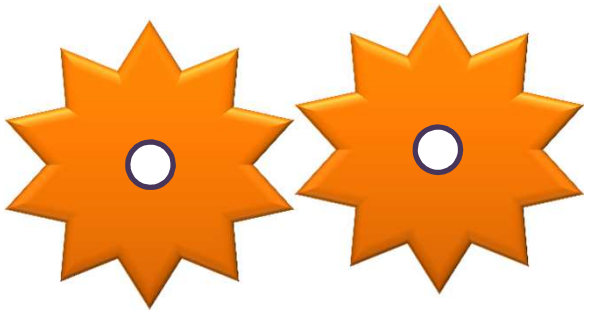
- PID
- 自己位置推定
- まいまい式

ＰＩＤ制御

- ＰＩＤ制御を入れると何故速くなるのか。
  - 「右へ行く」「左へ行く」のみだと、いつまでたっても目標(白と黒の境界)へ近付きません。
- そこで、
  - 目標と離れていたら大きく右/左へ行く、近いときは、小さく右/左へ行く。(P:比例制御)
  - 目標との差分を溜め込み、目標に近付ける。(I:積分制御)
  - 目標との差が急に大きくなったときに、制御量を増やす。(D：微分制御)

自己位置推定

- 走行体が、コース上のどこに居るかを把握することができれば、戦略を増やすことができます。
- モータのエンコーダを使い移動距離を計算することで、推定可能？
  - エンコーダの精度は厳密では無いので、走れば走るほど誤差が蓄積されます。
  - 尻尾を降ろしたまま走れば、タイヤがスリップするので誤差が発生する可能性も。（尻尾を降ろさないといけない難所もあったような．．．）



まいまい式

- 光センサのLEDをON-OFFすることで、外乱光に対応します。
  - LEDがOFFのときは、外乱光の影響を検知する。
  - LEDがONのときは、LEDの光がコースに反射した成分と、外乱の影響を検知する。
  - (ONのとき) – (OFFのとき)とすることで外乱光の影響を除去できる。
- ただし、定周波の外乱に対しては、工夫しないと除去できません。
  - 定周波の外乱＝たとえば、蛍光灯。人の目ではほとんど分かりませんが、50Hz/60Hzで点滅している。

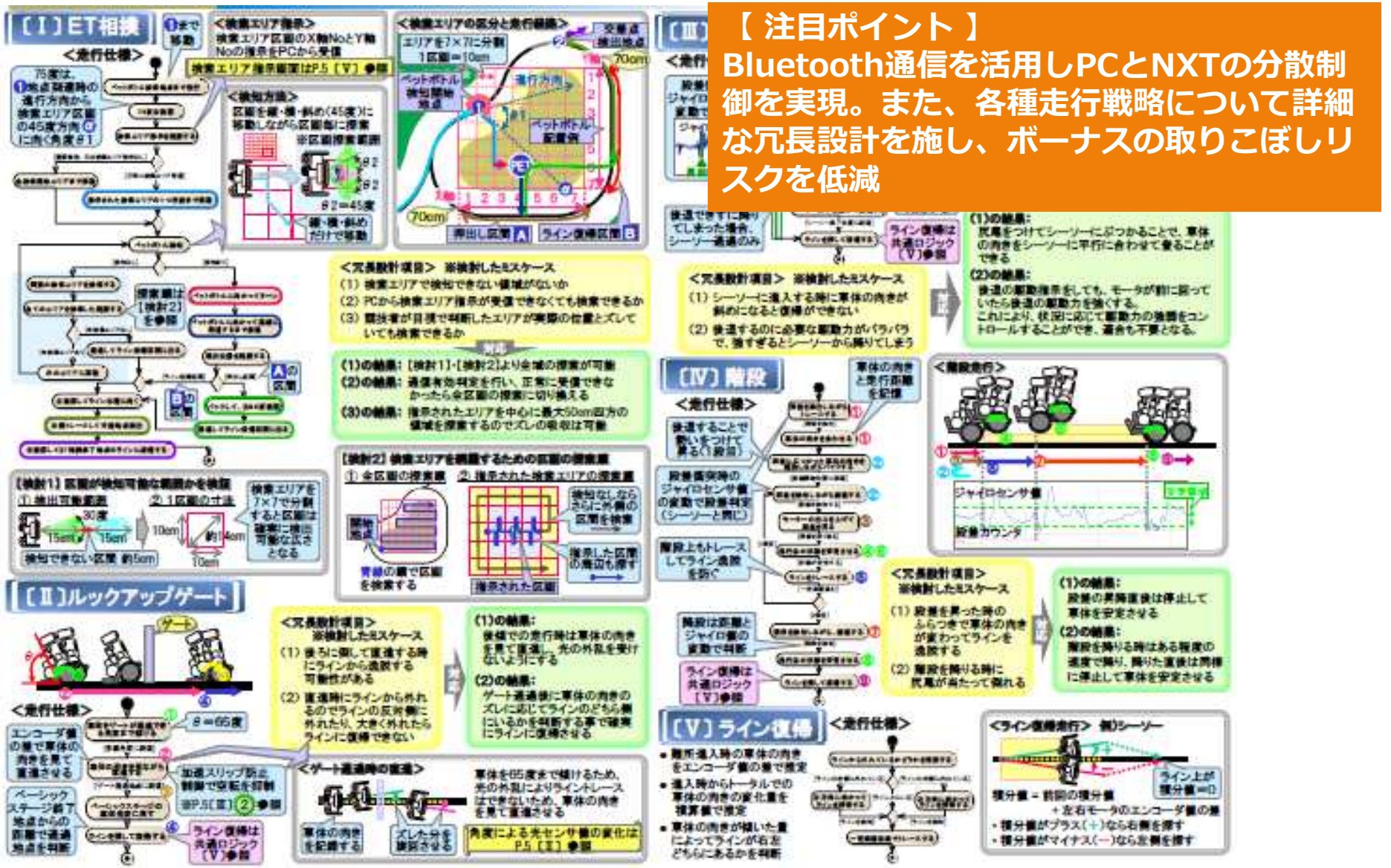
チャンピオンシップ大会
要素技術ハイライト

2011年 競技部門1位

Champagne Fight (北海道地区)

【 注目ポイント 】
ベーシックステージの路面傾斜によって尻尾角度を可変させ高速走行を実現。また、ボーナスステージの全難所を制覇する戦略と要素技術を満遍なく記載

2011年 競技部門2位
ＨＥＬＩＯＳ (東海地区)

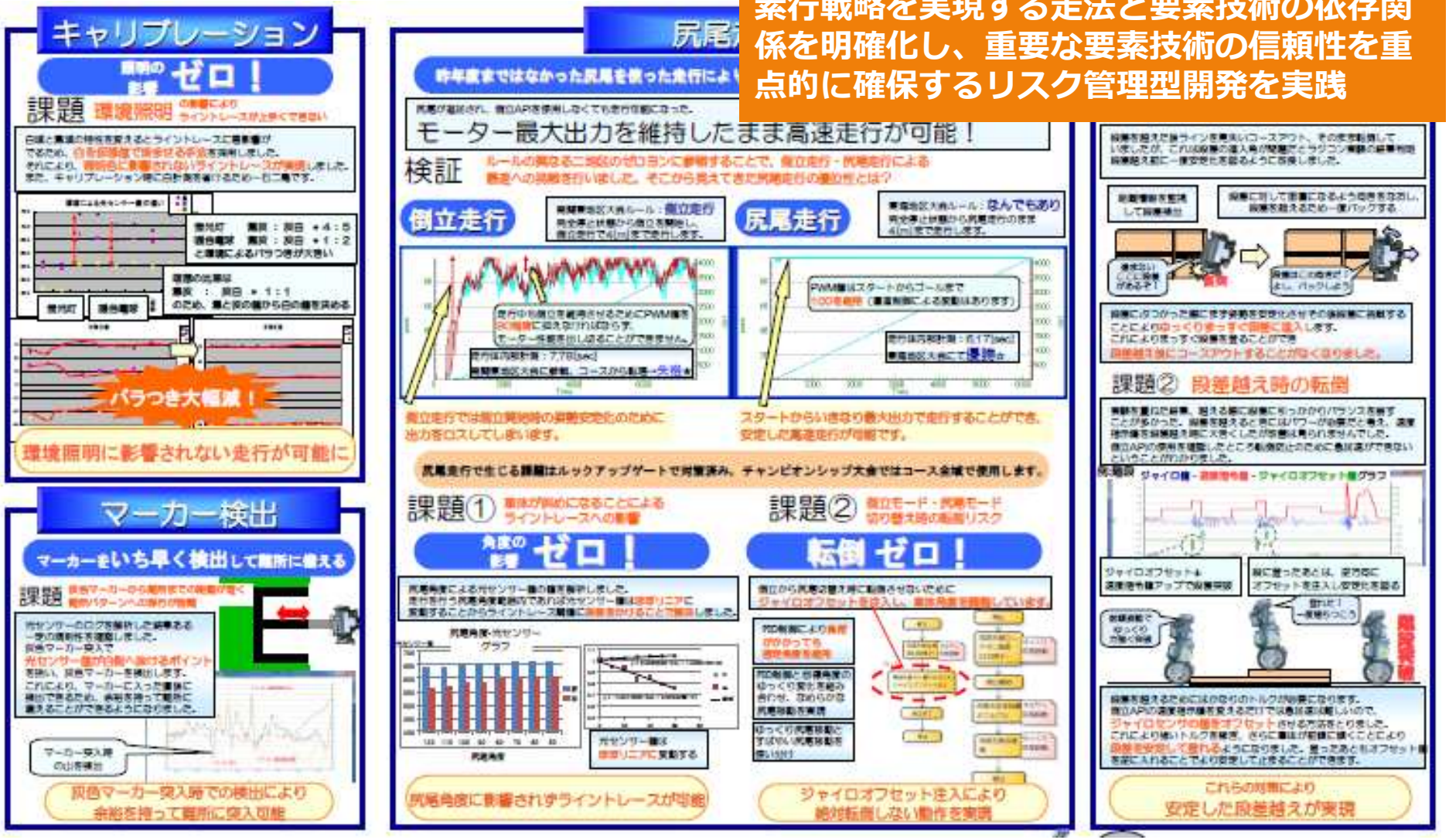


【 注目ポイント 】
Bluetooth通信を活用しPCとNXTの分散制御を実現。また、各種走行戦略について詳細な冗長設計を施し、ボーナスの取りこぼしリスクを低減

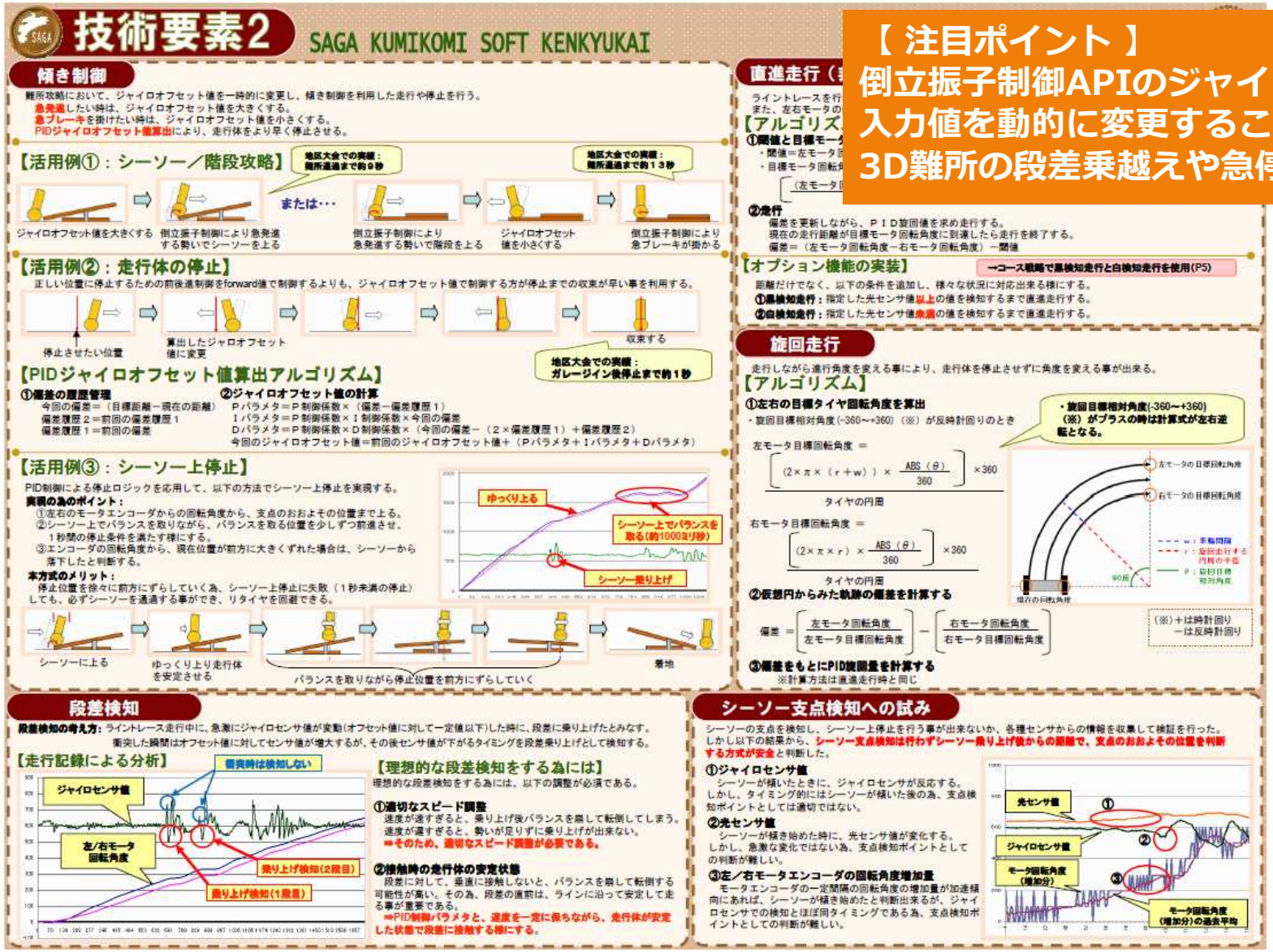
2011年 競技部門3位

Ｉｍｐｒｅｓｓｉｏｎｓ (東海地区)

【 注目ポイント 】
素行戦略を実現する走法と要素技術の依存関係を明確化し、重要な要素技術の信頼性を重点的に確保するリスク管理型開発を実践

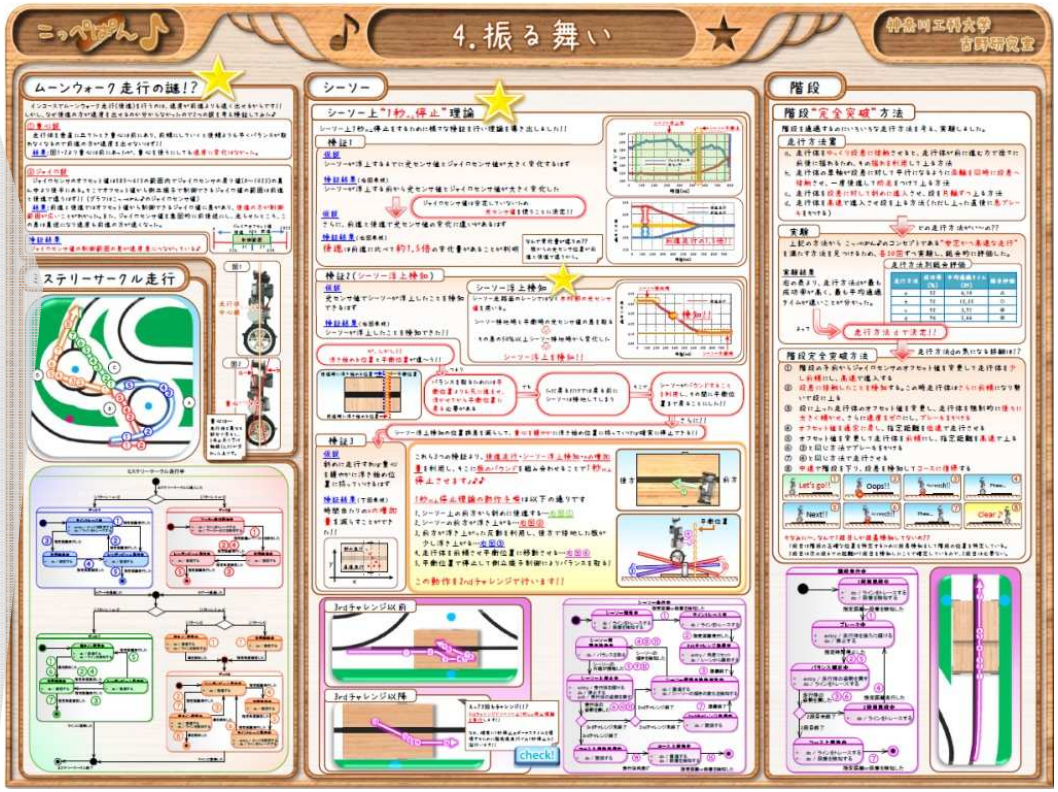


2010年 競技部門1位
SAGA組込ソフト研究会 (九州地区)



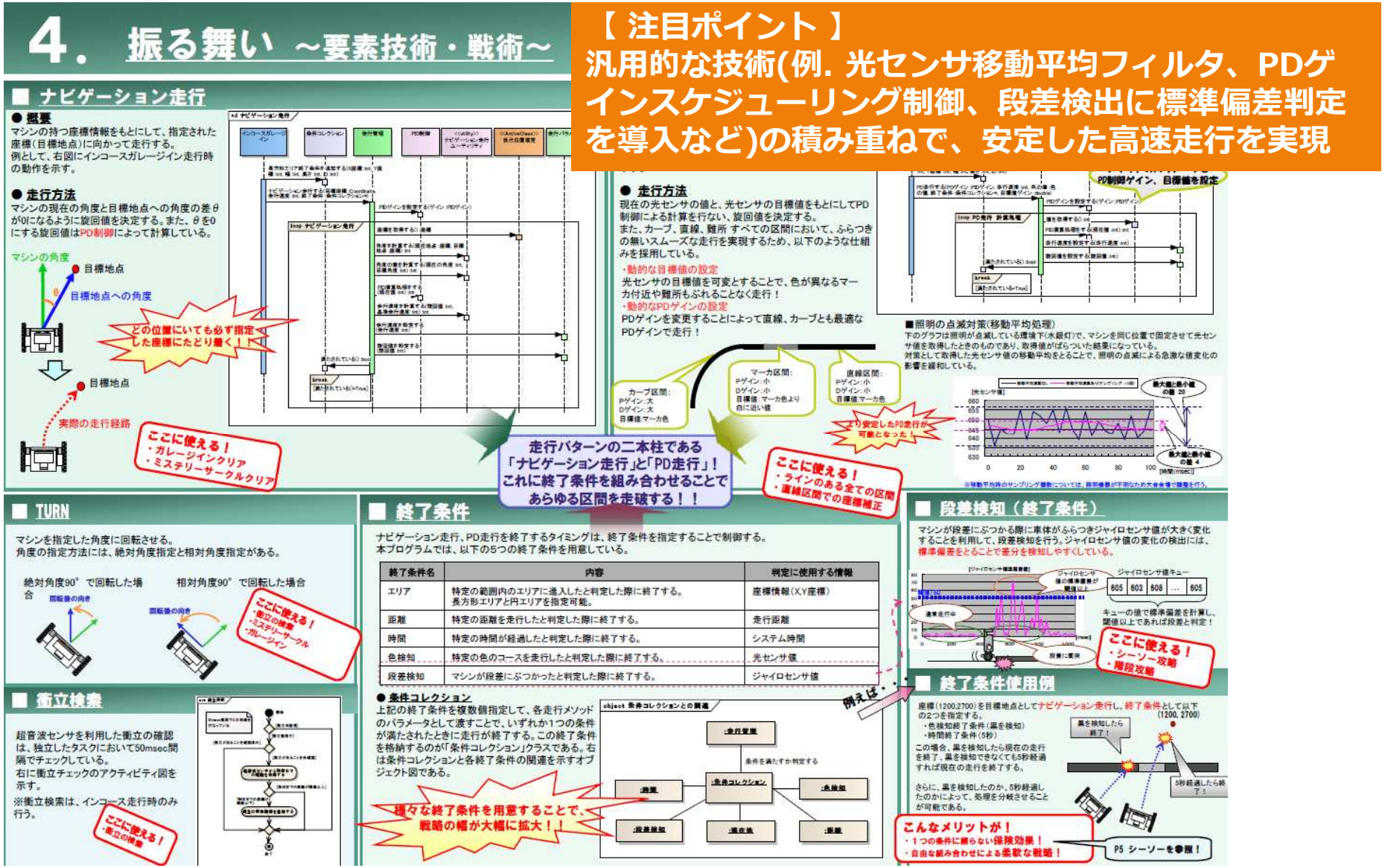
【 注目ポイント 】
倒立振子制御APIのジャイロオフセット入力値を動的に変更することにより、3D難所の段差乗越えや急停止を実現

2010年 競技部門2位
こっぺぱん♪ (南関東地区)



【 注目ポイント 】
走行体の重心位置のずれに着目したユニークなムーンウォーク(高速バック)走行の実現

2010年 競技部門3位
AEK RUNNER 10 (南関東地区)



最後にお願い

- 2010年
  - 3D難所では走行体転倒の可能性があります。競技を円滑に進めるためにも、**「走行体が転倒などのバランス制御不能状態になった場合は、自動的に両輪モータを停止するフェイルセーフ機能」**の搭載を検討してください。走行体の転倒状態を検出する(できる)ことは、実は3D難所攻略に必要な要素技術の一つでもあります。
- 2011年
  - 競技中のBluetooth通信が解禁されましたが、**「競技中のBluetooth通信の不調を想定し、Bluetooth通信無しでも走行(完走)できるようなシステムの冗長化対策」**の搭載を検討してください。せっかく頑張って開発してきたのに、通信不調でスタートできずorz では悲し過ぎるので…
- 2012年
  - 組込みシステムも機器単体だけで動作するのはなく、外部のシステムと連携して動作するものも増えてきています。前年のETロボコン競技では、まだ外部システムと連携して動作させているチームは少なかったように思いますので、**もっとBluetooth通信を活用**してみてはいかがでしょうか。今年は**動きさえすれば、ボーナスタイムをGet**できるところもありますし(^^)


ET
ROBOT
CONTEST
E
T