平成２４年度
ハードウェア・ソフトウェア
説明Green ET Challenge2012

平成24年11月17日

主催：福岡市

実施：NPO法人 九州組込みソフトウェアコンソーシアム（QUEST）

福岡市主催 （NPO法人 QUEST 運営）

参加費無料

Green ET Challenge 2012

スケジュール

1. 説明会 ：10月27日(土) 13:00-16:00
2. ハード､ソフト教育 ：11月17日(土) 13:00-16:00
3. 試走会－１ ：12月15日(土) 13:00-16:00
4. 試走会－２ ： 1月12日(土) 13:00-16:00
5. 大会 ： 1月19日(土) 10:00-16:00

場所：福岡県Ruby・コンテンツ産業振興センター（上記いずれも）

様々な領域で活用される組込みソフトウェアによる制御を、Project Based Leaningで学びます。習得技術をロボットの走行制御に応用し、ECO走行の実装成果を大会で競い合う、とてもエキサイティングで実践的な組込みソフトウェア開発講座です。

Green ET Challenge 2012

以下

GETC

と呼びます

目次

- 消費電力とは
- 消費電力を抑制する設計
- C言語による設計
- 実行手順
- 計測システム
- mruby-NXT
- パワーメータ（ハードウェア説明）

消費電力とは

電力

- 電気でいうところの電力は
  - P[W]　＝　I[A]　×　E[V]　で表される。
    Pは電力、Iは電流、Eは電圧である。

- 一般に家庭で利用される家電の消費電力を表す場合　kWh　（キロワットアワー）で表示される。これは、1時間あたりどのくらいの電力を使ったか。という意味であり、2.5kWの家電を5時間利用した場合、
  2.5[kW]　×　5[h]　＝　12.5[kWh]と表現できる。

消費電力

- 先ほどの12.5kwhは消費電力とイコールである
- つまり、消費電力とはある時間における瞬間電力をどのくらいの時間利用したかということになる
- GETCで利用する消費電力の定義は、1秒間における電力を競技時間利用した場合の総合電力値とする。
  総合電力値[Ws] = Σp_i（i=0～競技時間,P=1秒あたりの電力）

低消費電力

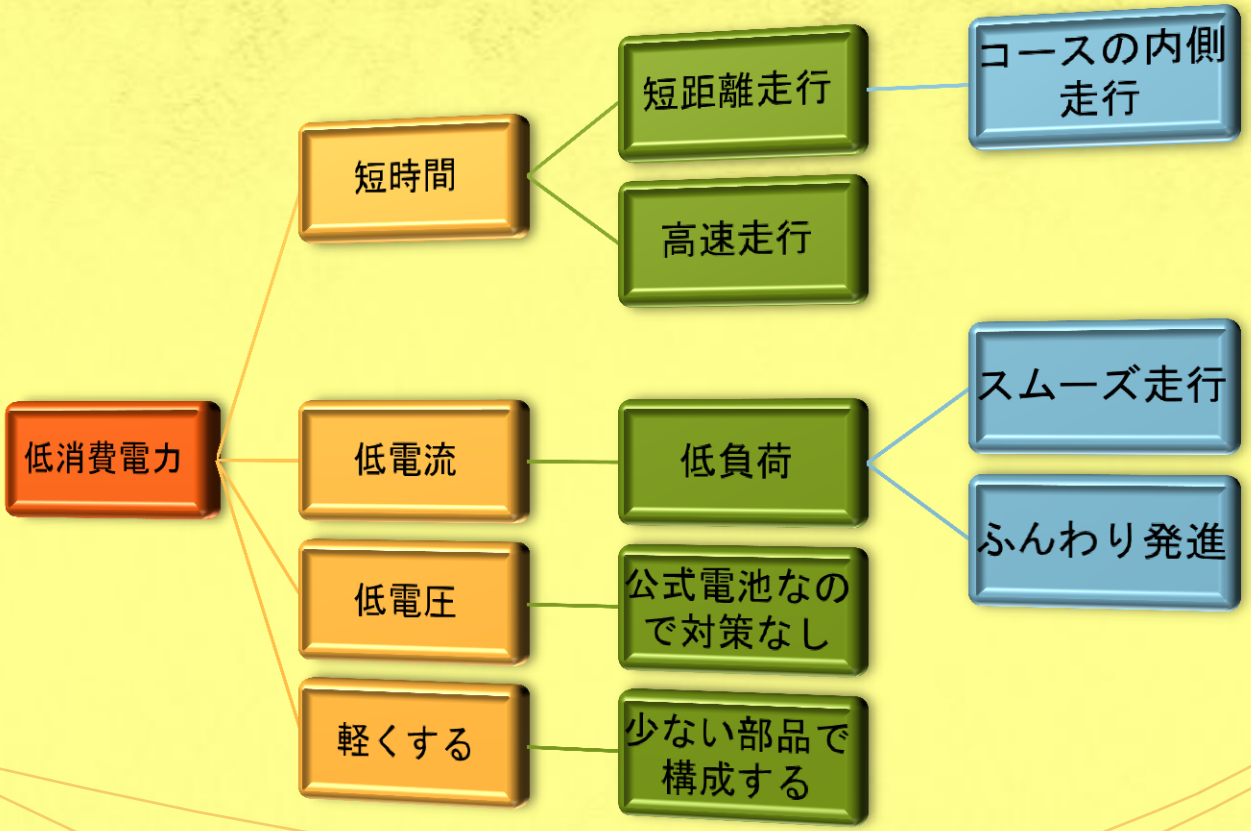
- 家電の例でも分かるように、利用時間が少ない方が、消費電力が少ない。
- GETCでも低電力で、走行時間が少ない方が低消費電力となり、有利！
- 質量が大きいほど移動するエネルギーも大きくなり、消費電力も大きくなる

どのようにすればいいのか

- 今までのことから消費電力は、電力の総和であるので、
  - 電力を低く抑える
  - 走行時間を短くする
  - 軽くする

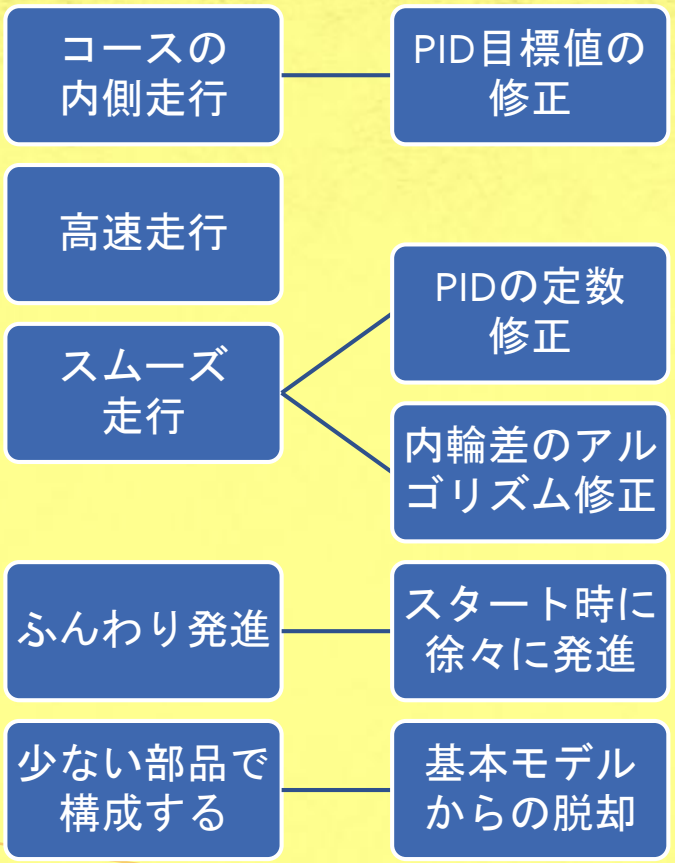
低消費電力の分析

- これらを分析してみると



具体的な戦術

- 具体的な戦術（戦略は低消費電力走行）



PID目標値の修正

- ロボットは右回り（時計回り）に動く
- 壁の内側を回るためには右側のセンサーと左側のセンサーを足して２で割る値を目標値としておく。
- もしそれよりも内側を回るのであれば、やや右寄りに目標値を補正する必要がある。

PID定数の修正

- 基本モデルでは、筐体に合わせたPIDの定数で調整しているが、タイヤの直径や位置の変更などによりPIDの定数は大きく変わる。また、現在のPID定数が最適とは限らない

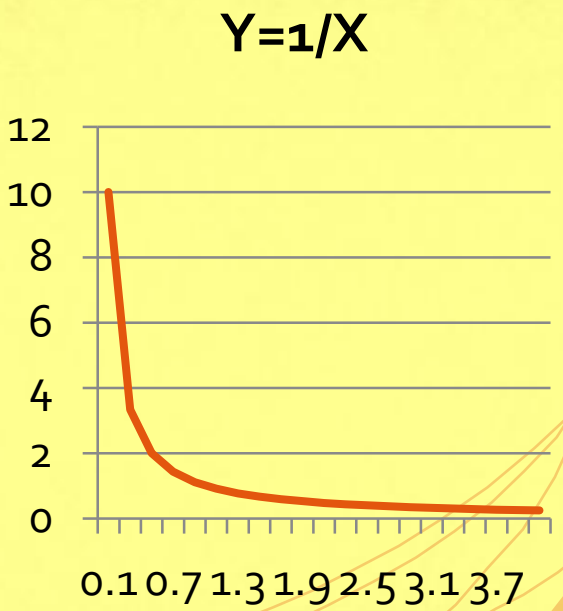
- PIDのPは、現時点での制御を行い、Iは過去のデータにより追従制御を行い、Dは未来を予測して制御を行う。

限界感度法

- PIDのうちIとDの定数をゼロにする
- Pの定数を変化させて、発振、収束、発散の限界点を見つける
- Pの値は発振する値より小さい値とする
- 次にPを一定にして、Iの値を変化させる。
- Iはカーブ等で追従できないような値を補正する
- PとIが決まれば、Dを設定する。

内輪差のアルゴリズム修正

- デモ用プログラムでは、y=1/xの関数によって構成されてる。
- Xが右タイヤの制御量
- Yが左タイヤの制御量
- 右タイヤの動きが大きいほど、左タイヤの動きは小さい
- このアルゴリズムをさらに改造すれば滑らか走行となるかもしれない。



徐々に発進

- スタート時に徐々に発進
  - モータはコイルの集合体といっても良い
  - コイルは、最初電流が流れにくく、一度流れだすと止めてもしばらく流れ続けるという特性がある
  - 電流が多く流れるということは、消費電力も多く流れるということである
- ふんわり発進、ふんわり停車
  - 急発進、急停車を避ける

基本モデルからの脱却（軽量化や構成変更）

- モータが２個から１個になれば優勝の可能性は極めて高い
  - 駆動＋ステアリングを１モータで行うことは困難
- 基本モデルはがっちりと作られている
  - 軽量化の可能性あり。壊さない程度に。
- 基本モデルの後輪は点検を怠ると発進時に横向きになる場合がある
  - 改良の余地あり
- 電池を減らす改良はNG
- ギア比を変更することは極めて有効
  - 少ない電力で回転数を上げる。自転車のトップギアを連想

C言語による設計

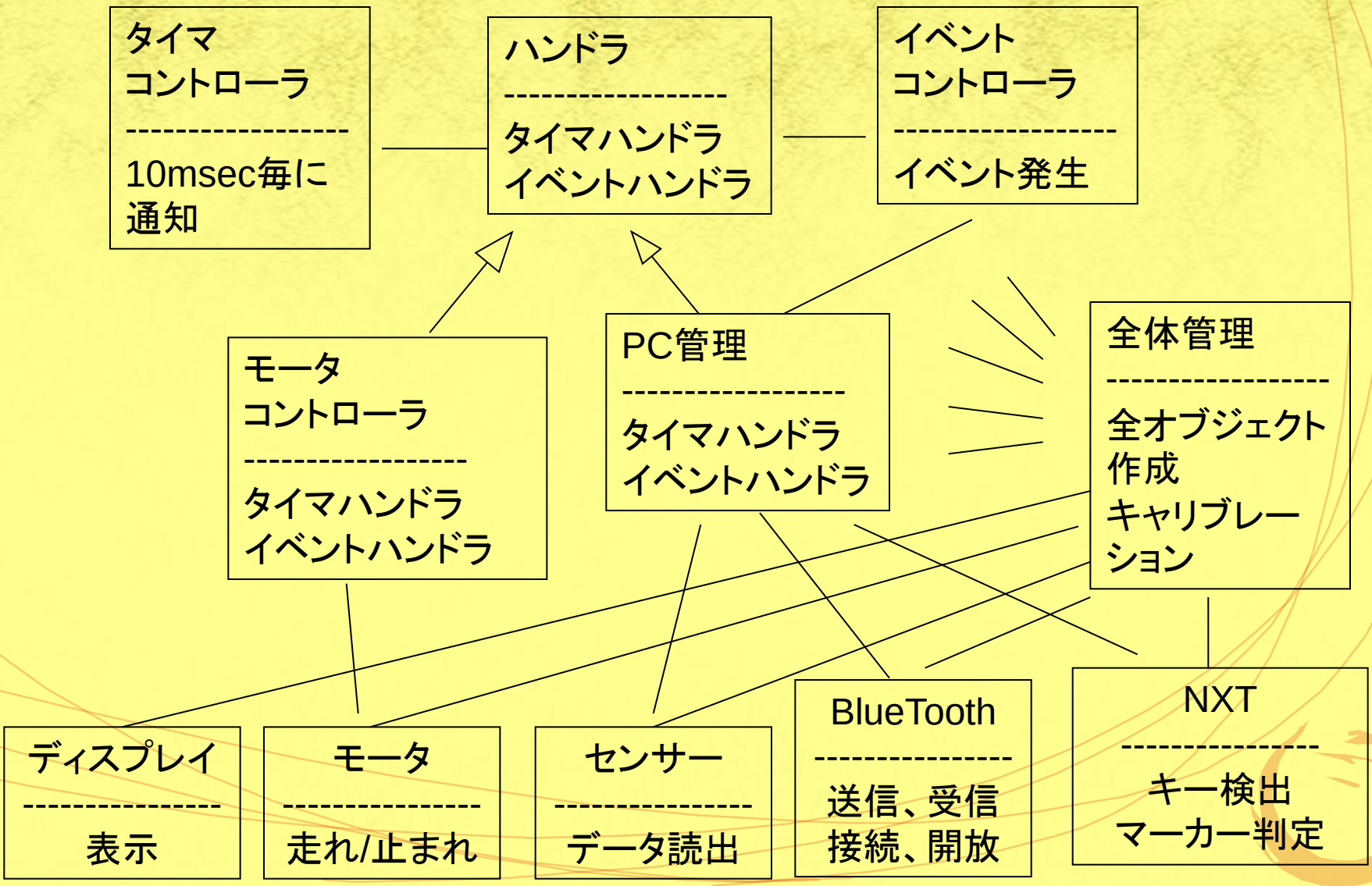
GETC2011との違い

- C++からC言語へ修正
- 去年は秘密だったy=1/x関数を使ったturn制御を公開
- 自作パワーメータを利用した電力測定

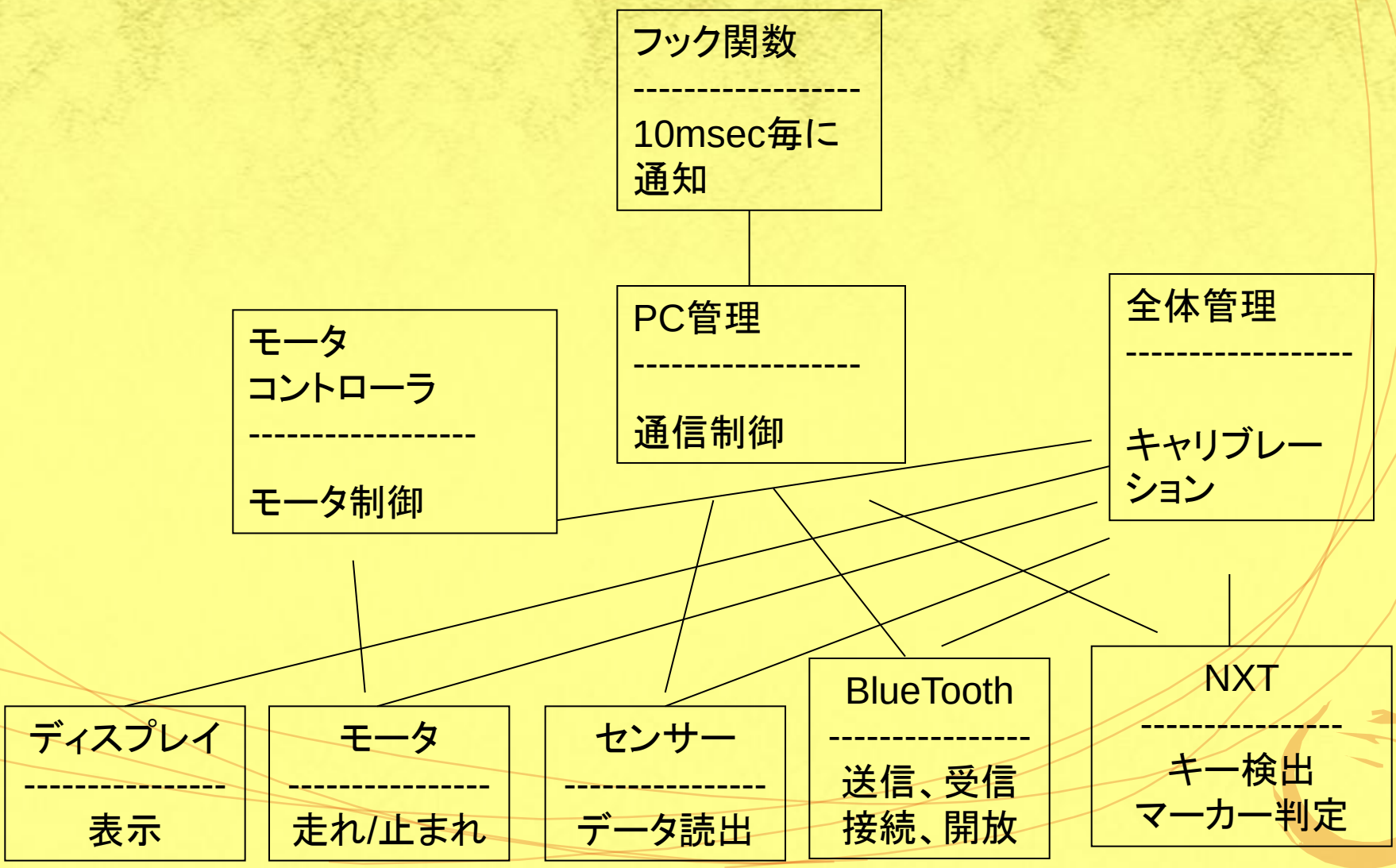
C++からC言語へ修正

- 昨年はオブジェクト指向プログラミングを学習するために、C++を利用した
- 今年はプログラムそのものを容易に理解するためにC言語によって基本モデルを公開する
- 去年利用したポリモフィズムは難解だったので今年は利用しない
- 各関数は、クラス名を頭につけてメソッド名とする

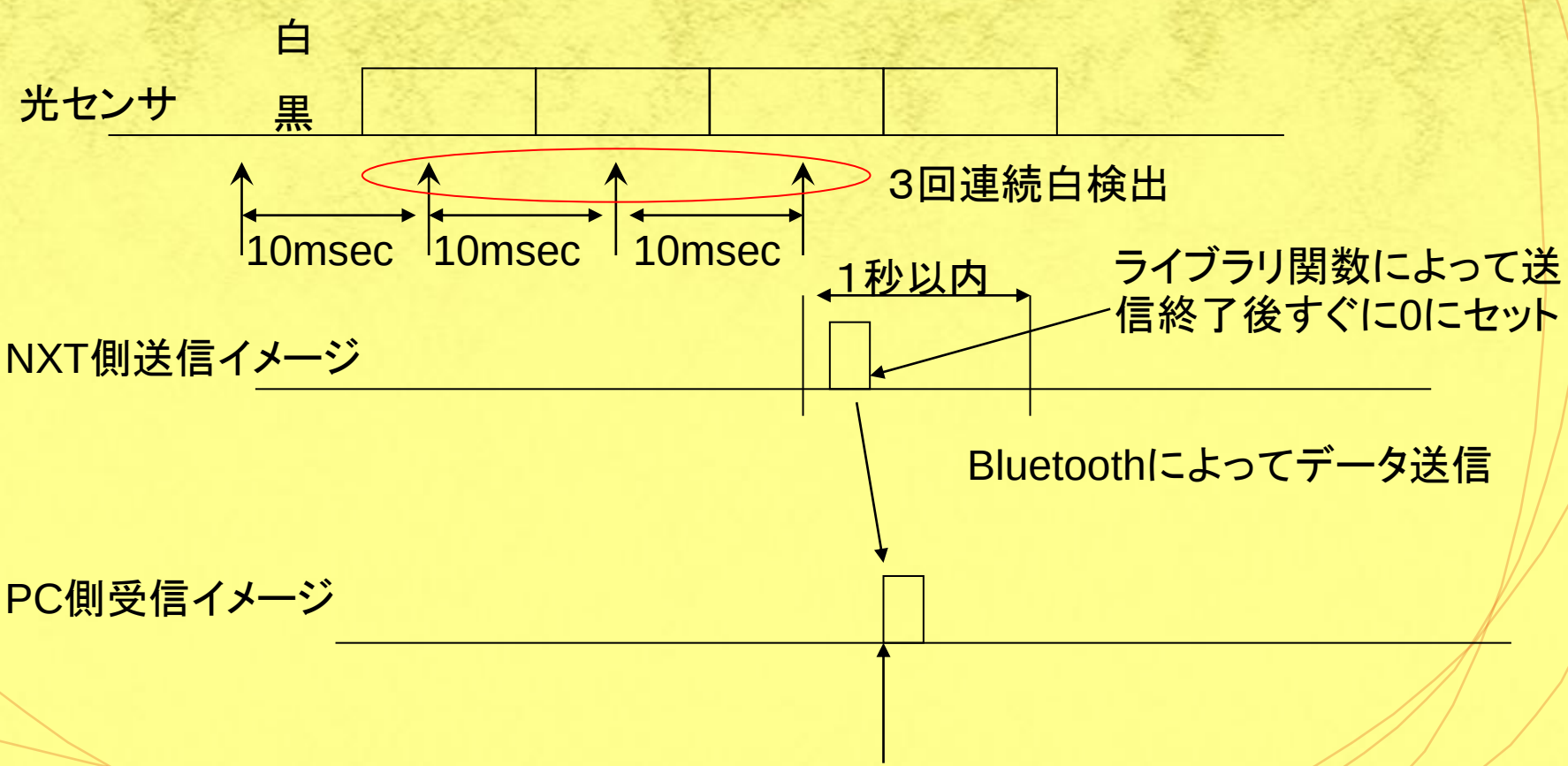
去年の設計


タイマ
コントローラ
10msec毎に
通知
ハンドラ
タイマハンドラ
イベントハンドラ
イベント
コントローラ
イベント発生
モータ
コントローラ
タイマハンドラ
イベントハンドラ
PC管理
タイマハンドラ
イベントハンドラ
全体管理
全オブジェクト
作成
キャリブレー
ション
ディスプレイ
表示
モータ
走れ/止まれ
センサー
データ読出
BlueTooth
送信、受信
接続、開放
NXT
キー検出
マーカー判定

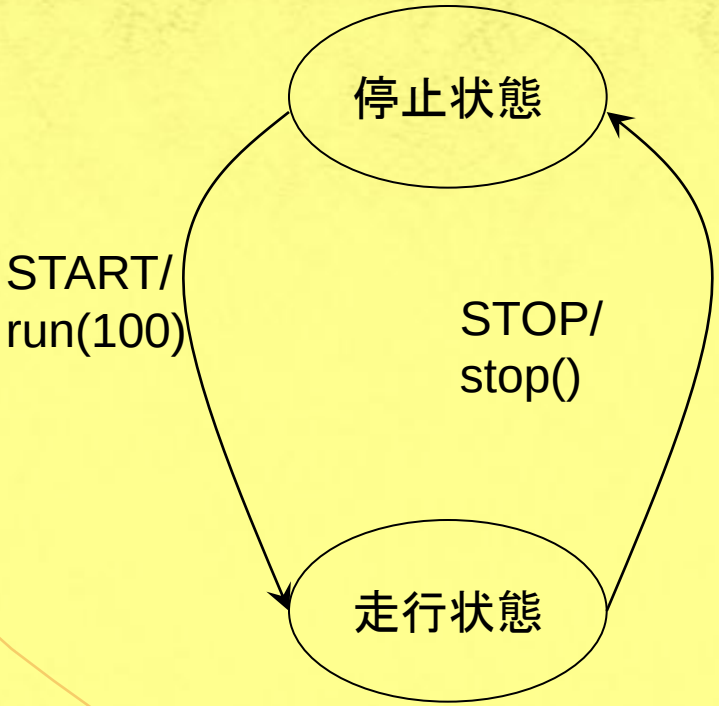
今年の設計


フック関数
10msec毎に
通知
PC管理
通信制御
モータ
コントローラ
モータ制御
全体管理
キャリブレー
ション
ディスプレイ
表示
モータ
走れ/止まれ
センサー
データ読出
BlueTooth
送信、受信
接続、開放
NXT
キー検出
マーカー判定

マーカ検出の方法


白
光センサ
黒
3回連続白検出
10msec
10msec
10msec
1秒以内
ライブラリ関数によって送信終了後すぐに0にセット
NXT側送信イメージ
Bluetoothによってデータ送信
PC側受信イメージ
0から1へ変化したエッジを見つけて、
マーカーカウントアップ

MotorController内の状態遷移図



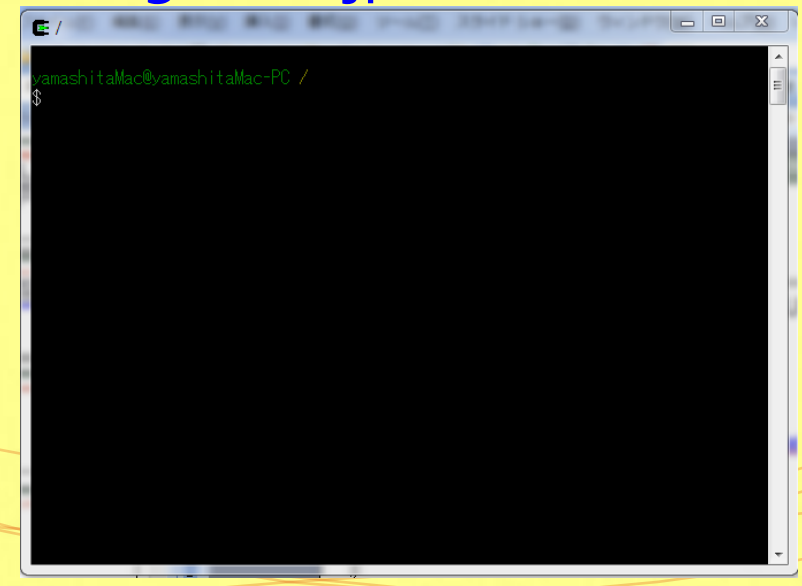
- このような簡単なシステムでも状態遷移を用いる理由として
  - 同じイベントが２度続いてくる場合の対処
    - STARTが２回きてもOK
  - 状態や、イベントが仕様変更により増減しても対処が比較的楽にできる。
    - 状態遷移で設計した場合はコードへの変換がルーチンワーク的にできる

開発環境（WindowsPC）

- cygwin
- nxtOSEK
- ディレクトリ構造
- Makefileの使い方
- ライブラリファイルの作り方
- プログラムアップロード
- 実行

Cygwin

- Microsoft Windows OS上で動作するUNIXライクな環境であり、フリーソフト
- nxtOSEKを使うために必要
- インストール方法
  - http://lejos-osek.sourceforge.net/jp/installation_enf.htm を参照



nxtOSEK

- オープンソースのLEGO MINDSTOMES NXT用の開発/実行環境
- leJOS NXTという開発環境に含まているI/Oドライバ及びTOPPERSプロジェクトの成果物の一つである、TOPPERS/ATK、TOPPERS/JSPをNXTのハードへ移植したリアルタイムOSで構成されている
  - gccツールチェインを使用して、ANSI　C/C++言語が利用可能
  - NXTモータ、センサーのデバイスへの制御APIを用意
  - 自動車電子制御用のOSEKに対応したマルチタスクスケジューリング
  - UITRON4.0に対応したマルチタスクスケジューリング

ディレクトリ構成

- cygwin
  - nexttool
  - nxtOSEK
  - ユーザディレクトリ / (例： GETC_1)
    - ユーザ用プロジェクトディレクトリ / (例：getc2012_sample_c)
      - ユーザ用ソースコード　　xxx.c xxx.h

Makeファイルの書き方

```
# nxtOSEKルートディレクトリ
NXTOSEK_ROOT = ../../nxtOSEK

# ターゲット実行形式ファイル名
TARGET = getc2012sample

# インクルードパス
#USER_INC_PATH=
$(NXTOSEK_ROOT)/ecrobot/nxtway_gs_balancer
USER_INC_PATH= ../

# ライブラリ
#USER_LIB = nxtway_gs_balancer
USER_LIB = keisoku
```

```
# Cソースファイル
TARGET_SOURCES = ¥
        pc_manager.c ¥
        display.c ¥
        bluetooth.c ¥
        motor_controller.c ¥
        nxt.c ¥
        getc_sample.c ¥
        manager.c
# CPP(.cpp)ソースファイル
TARGET_CPP_SOURCES = ¥

# TOPPERS/ATK(OSEK)設定ファイル
TOPPERS_OSEK_OIL_SOURCE = getc_sample.oil

# 下記のマクロは変更しないでください
O_PATH ?= build
include $(NXTOSEK_ROOT)/ecrobot/ecrobot.mak
```

make clean

- make clean
- 上記コマンドを実行すると、makeによって生成されたオブジェクトや必要のないファイル、実行ファイルがすべて削除される。
- make allを行なって、どうしてもコンパイルが通らない場合は、一度make cleanを行なって、再度make allを行うと通る場合がある。
  - これは、あるファイルを修正したら、影響のあるファイルをコンパイルしなければならないのにもかかわらず（依存関係）コンパイルしてなかったという現象がたまに起こるからである。
  - この現象はUNIXではMakefileの修正を行うことで対策できるのだが、今回紹介しているMakefileではその修正が困難

ライブラリファイル

ディレクトリ構造

- cygwin
  - nexttool
  - nxtOSEK
  - ユーザディレクトリ/(例： GETC_1)
    - ライブラリ用ディレクトリ/(例：lib)
      - アーカイブ名ディレクトリ/(例：keisoku)
        - ライブラリ関数 xxx.c xxx.h
    - ユーザ用プロジェクトディレクトリ/(例：getc2012_sample_c)
      - ユーザ用ソースコード　　xxx.c xxx.h

ライブラリのMakefile

```
# Makefile for NXTway-GS balancer library

# modified to support new directory structure by takshic
ROOT := $(dir $(lastword $(MAKEFILE_LIST)))../../../nxtOSEK

ECROBOT_ROOT = $(ROOT)/ecrobot
# added to support new directory structure by takshic
ECROBOT_C_ROOT = $(ECROBOT_ROOT)/c

LEJOSNXJSRC_ROOT = $(ROOT)/lejos_nxj/src/

LEJOS_PLATFORM_SOURCES_PATH = $(LEJOSNXJSRC_ROOT)/nxtvm/platform/nxt
LEJOS_VM_SOURCES_PATH = $(LEJOSNXJSRC_ROOT)/nxtvm/javavm

C_LIB_SOURCES = ¥
        send_bt.c

C_OPTIMISATION_FLAGS = -Os
include $(ECROBOT_ROOT)/tool_gcc.mak

INC_PATH := ¥
        $(LEJOS_PLATFORM_SOURCES_PATH) ¥
        $(LEJOS_VM_SOURCES_PATH) ¥
        $(ECROBOT_ROOT) ¥
        $(ECROBOT_C_ROOT)

O_FILES = $(C_LIB_SOURCES:c=o)

TARGET = ../../libkeisoku.a

.PHONY: all
all: $(TARGET)

$(TARGET): $(O_FILES)
        @echo "Creating $@"
        $(AR) rv $(TARGET) $(O_FILES)

%.o: %.c
    @echo "Compiling $< to $@"
    $(CC) $(CFLAGS) -o $@ $<

%.oram: %.c
    @echo "Compiling $< to $@"
    $(CC) $(CFLAGS) -o $@ $<

%.o: %.s
    @echo "Assembling $< to $@"
    $(AS) $(ASFLAGS) -o $@ $<

%.oram: %.s
    @echo "Assembling $< to $@"
    $(AS) $(ASFLAGS) -o $@ $<

%.obmp : %.bmp
    @echo "Converting $< to $@"
    $(OBJCOPY) -I binary -O elf32-littlearm -B arm ¥
    $< $@

.PHONY: release
release:
    rm $(O_FILES)

.PHONY: clean
clean:
    rm $(TARGET)
    rm $(O_FILES)
```

プログラムのアップロード

- USBケーブルを使って、開発用PCとNXT本体を接続する
- 初めて接続する場合は、USBケーブルが接続されたというメッセージがでるので、それまで待つ（OSによって表示が変わる）
- NXTの電源をONにする
- Makeしたディレクトリに移動する
- 次のコマンドを打つ
  - 拡張NXTファームウェアの場合
    - $sh rxeflash.sh ($はcygwinのプロンプトなので、打たない。）
  - NXT BIOSの場合
    - $sh appflash.sh

アップロード

- 次のメッセージがでてきたらOK
  - Executing NeXTTool to upload getc2012sample.rxe...
  - getc2012sample.rxe=xxxxxx
  - NeXTTool is terminated.
    - ファイル名や、サイズはプログラムによって変化します
- この時に、 getc2011sample.rxe=xxxxxが表示されない場合はアップロードに失敗したと考えたほうが良い
  - 失敗する主な原因は次の通り
    - NXTの電源が入っていない場合
    - PCへUSBケーブルが接続されていない場合
    - NXTでプログラムが実行されている場合（これが一番多いので注意）

ライブラリについて（静的ライブラリ）

- ライブラリとは
  - libxx.aとあるのは、一般的に関数コンパイルしたものをアーカイブしたものである。
  - 単体では動作できないので実行ファイルではない。
  - arコマンドでアーカイブされるので、拡張子がaになっている
  - Make時にリンクされるので、静的ライブラリとも呼ばれる。
  - 組み込みシステムでは、動的ライブラリの置く場所がもったいないので、静的ライブラリの活用がほとんど

- libkeisoku.aに含まれる、void send_bt()関数の説明

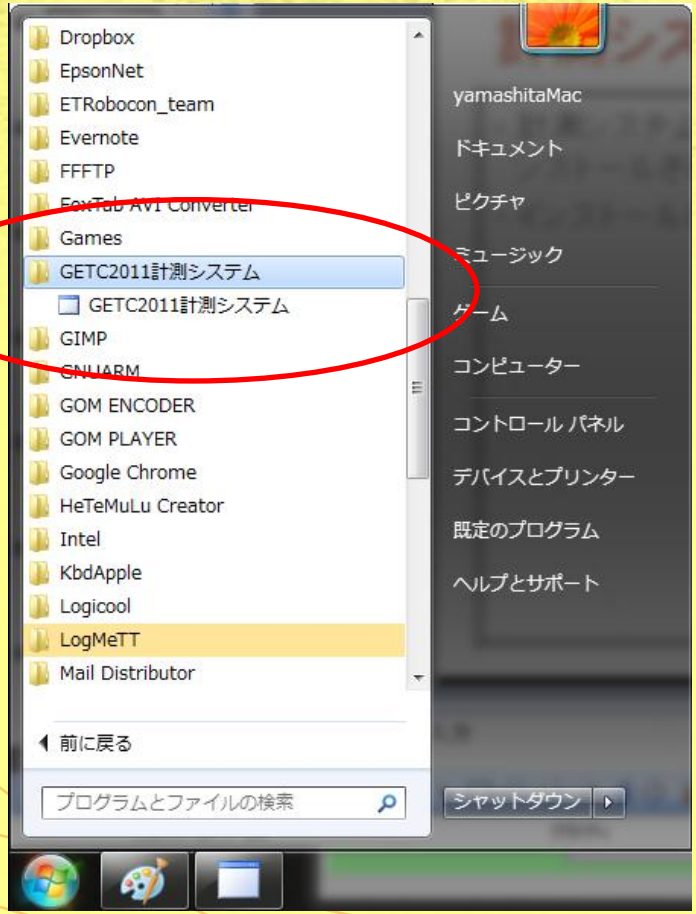
```
//説明：interval_time/reduce_time毎に電力の平均をとり，light_dataの値とともに
//　計測器へ送信する．送信が終わったら，light_dataはoにセットされる
//引数説明 int interval_time Bluetooth送信間隔 msec　値変更不可
//引数説明 int reduce_time　PcManagerへの通知間隔 msec 値変更不可
//引数説明 int *light_data マーカ検出値 1:検出 0:未検出。ライブラリの中でoに設定される。
//引数説明 U8 data1,data2 ユーザデータ
//引数説明 S32 data21,data22,data23,data24 ユーザーデータ
//引数説明 S16 data31,data32,data33　　ユーザーデータ
 void send_bt(int interval_time , int reduce_time , int *light_data ,
                        U8 data11 , U8 data12,
                        S32 data21, S32 data22,S32 data23, S32 data24,
                        S16 data31 , S16 data32 ,S16 data33 );
```

実行手順

実行手順

- 計測システムのインストール方法
- Bluetoothについて
- NXTのBluetoothの設定
- PCとNXTのBluetoothのコネクション
- COMポート番号の確認
- プログラム実行からコース設置までの手順
- 計測システムの使用方法
- 既知のバグ

計測システムのインストール



- 計測システムは、setup.exeファイルを実行すると自動的にインストールされる
- インストールされた計測システムは、スタートメニューに登録される
- アンインストールは、コントロールパネルから通常のアプリケーションと同様に行う。
- 新しいリリースをインストールする場合は、古いバージョンは必ずアンインストールすること

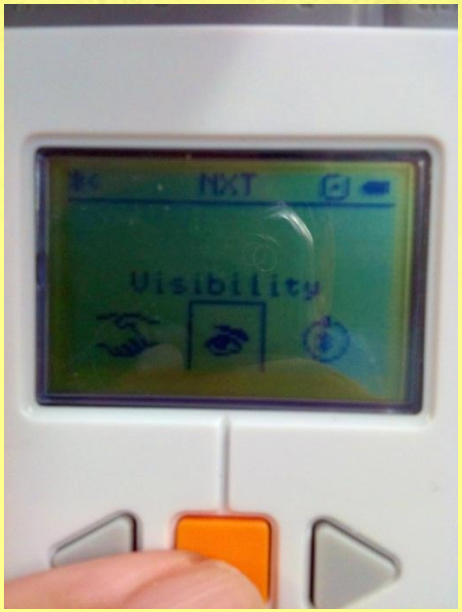
Bluetoothについて

- Bluetoothは、PCに内蔵されているものや、市販のものが利用できる
- よく知られている問題として、Bluetoothに付属しているTOSHIBAのドライバをインストールすると、NXTとの通信ができないことがあげられる。
  - ドングル等に付属しているドライバーはあえてインストールしなくても、Windowsに付属しているマイクロソフトのドライバーが自動的にインストールされる
  - 試していないがLEGO社が販売している公式のBluetoothドングルも利用できると考えられる。（ETロボコンでは推奨ドングル）
  - 下記の事例で動作確認しています
    - Mac内蔵のBluetooth＋bootcamp＋Windows7 Home Premium
    - Corega製マイクロサイズブルートゥースUSBアダプタ＋Panasonic CW-7 Windows Vista

NXTのBluetoothの設定

- 電源が入ったあとのTOPメニューから左右どちらかの矢印ボタンを３回押して、Bluetoothの設定メニューを出し、ENTERキーを押下する
- BluetoothをONするために、そのままENTERキーを押下する

NXTのBluetoothの設定



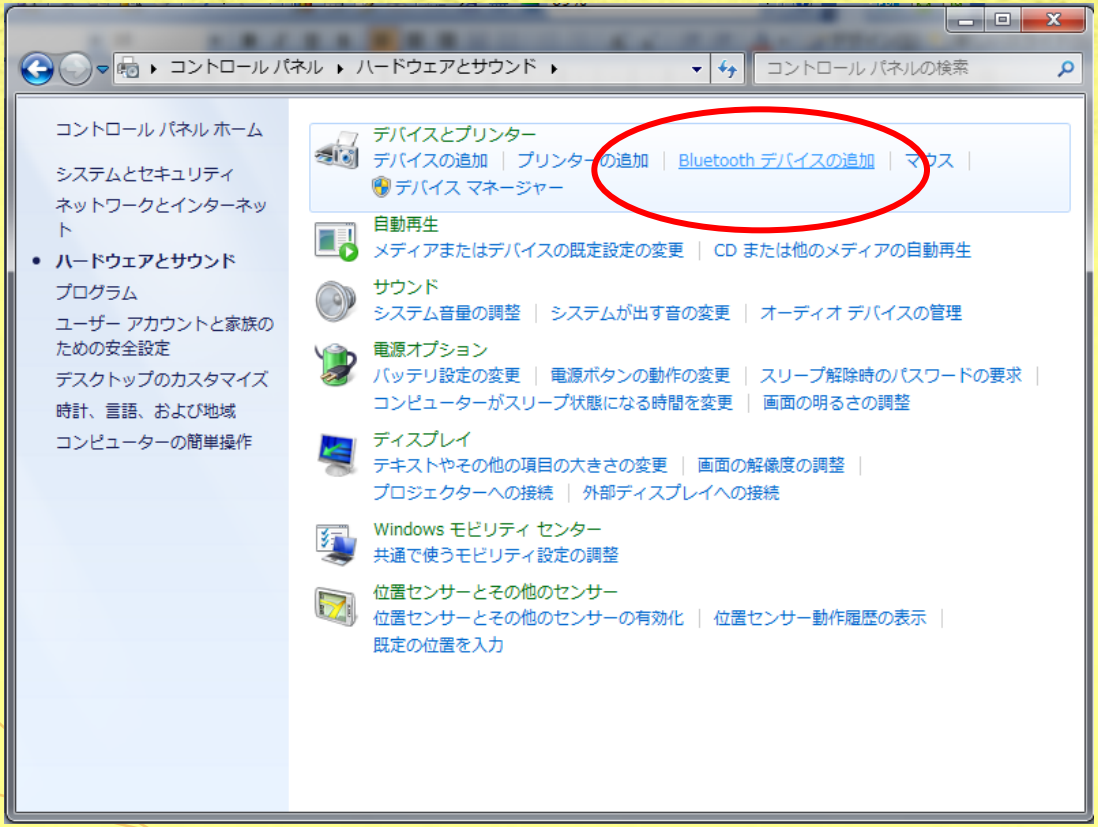
- ONすると、左上にBluetoothのマークが点灯する
- 左右ボタンを複数回押下してVisibilityを表示させる
- ENTERキーを２回押下し、左上にブルートゥースマークの横に、“<”が表示されれば、OK
- この設定は１回行うだけでよい。
- このマークが消えた場合は再度行う必要がある

B.Tの追加

Windows7の場合

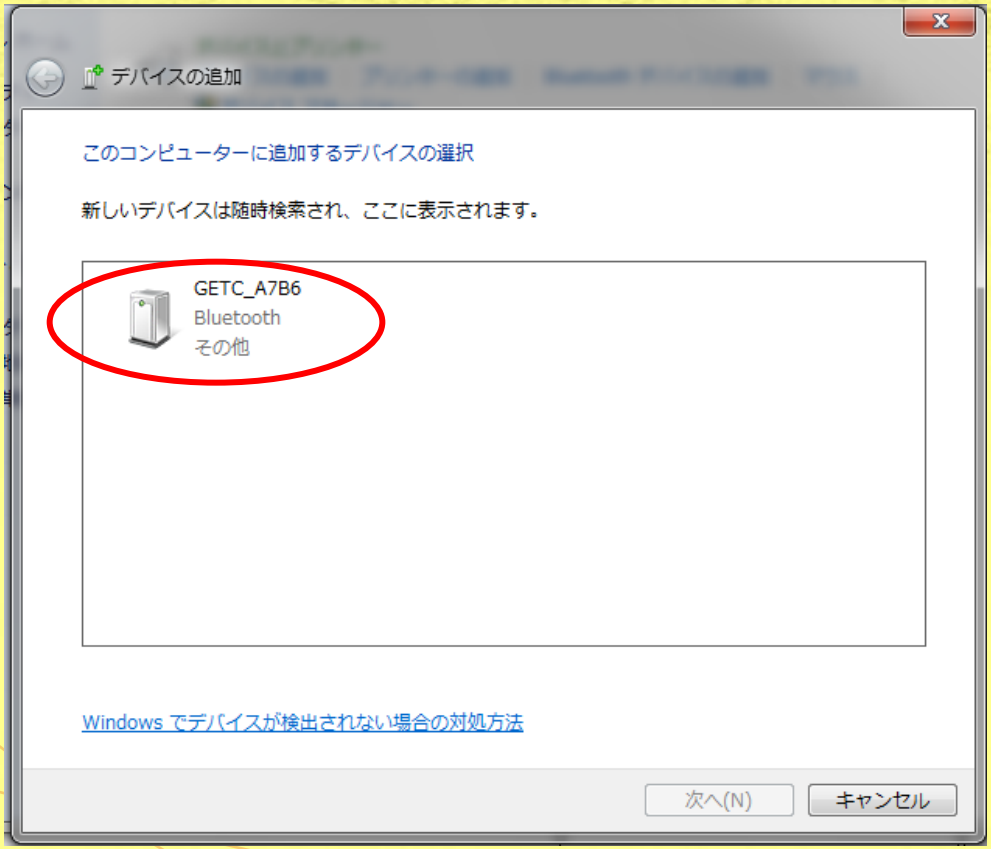
- スタートーコントロールパネルを開く
- ハードウェアとサウンドの中にある「デバイスの追加」をクリック

B.Tの追加



- Windows7の場合
  - 「Bluetoothデバイスの追加」をクリック
  - NXTの電源を入れる

B.Tの追加

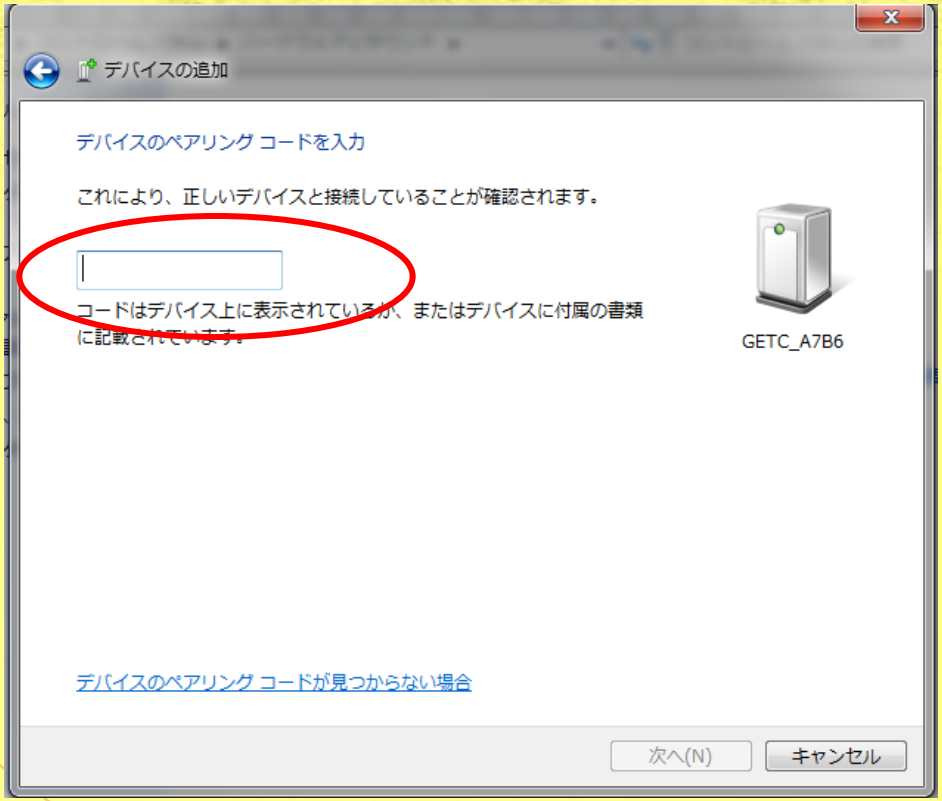


- Windows7の場合
  - デバイスの追加画面がでる
  - しばらくすると、NXTが検出される
  - 検出されたNXTをダブルクリックする

パスキー

- Windows7の場合
  - NXT側の画面が切り替わりパスキーの入力を促される
  - ENTERキーの下の灰色キーでクリア
  - 矢印キーでパスキーの選択
  - ENTERキーで決定
  - ✓を選択すると入力終了となる
  - １２３４というパスキーはデフォルトになっているので、適当な４桁の数字を入れる
  - **大会時のパスキーは1234固定とする**

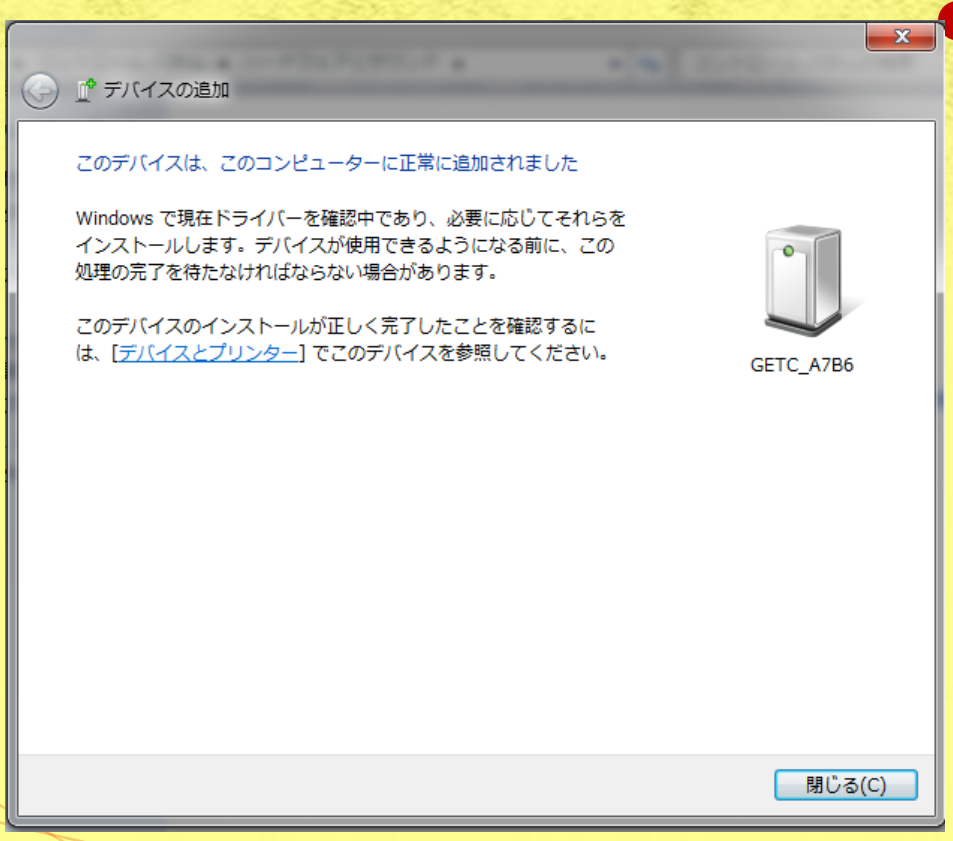
パスキー



Windows7の場合

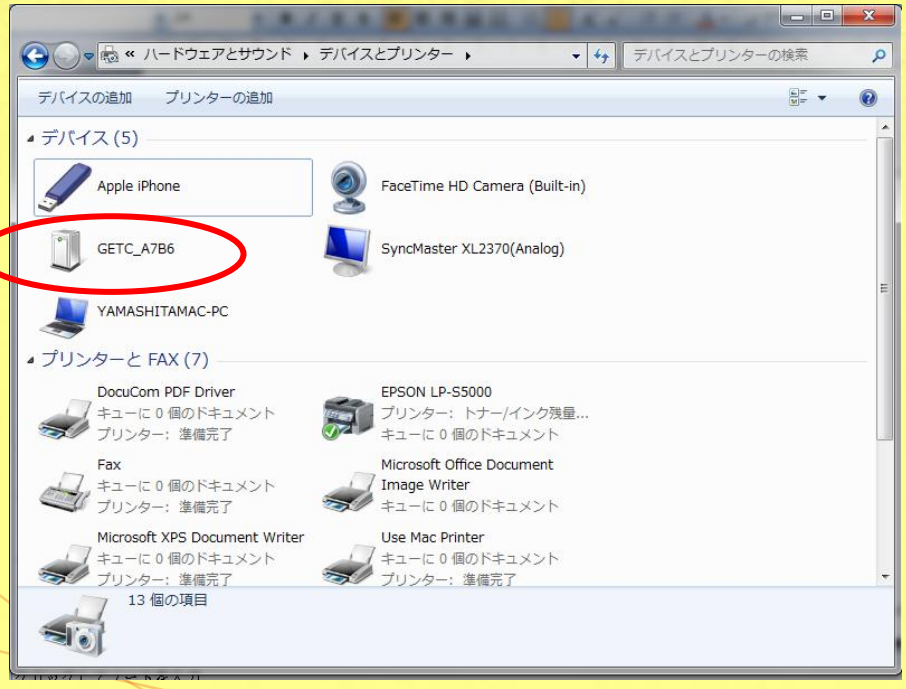
- NXTでパスキーを入れるとWindowsに左図のような画面が出る。
- ここで、先程NXTへ入力した４桁の数字１２３４を入れる

パスキー



- Windows7の場合
  - 左図の画面がでれば登録OK「閉じる」を押下

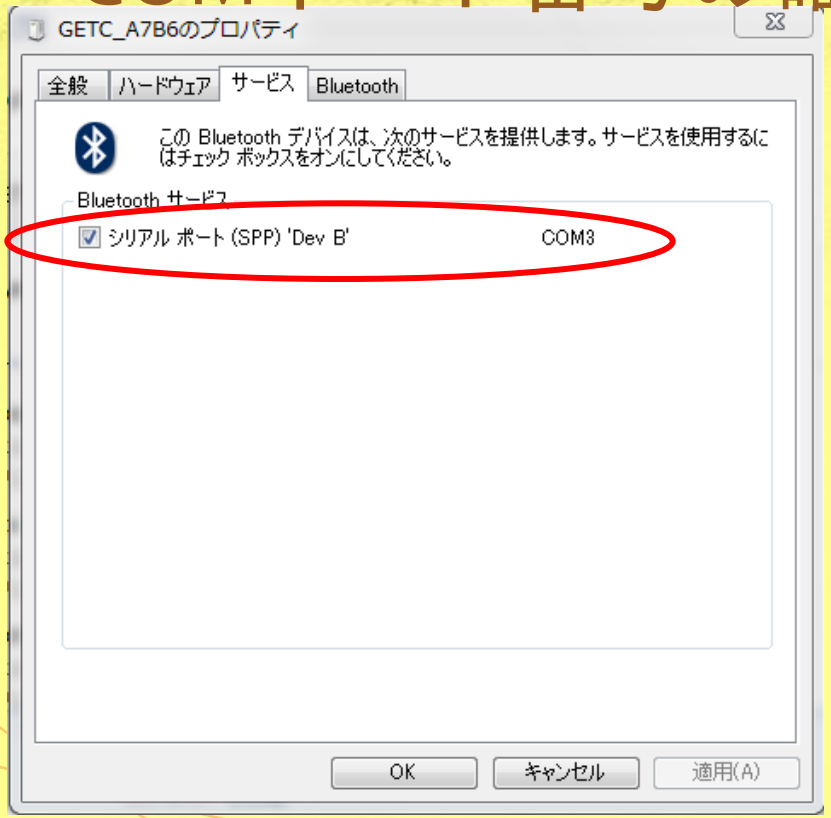
COMポート番号の確認



- Windows7の場合
  - スタートーコントロールパネルーハードウェアとサウンドーデバイスとプリンターをクリック
  - 画面に追加したNXTがあるので、ダブルクリック

COMポート番号の確認

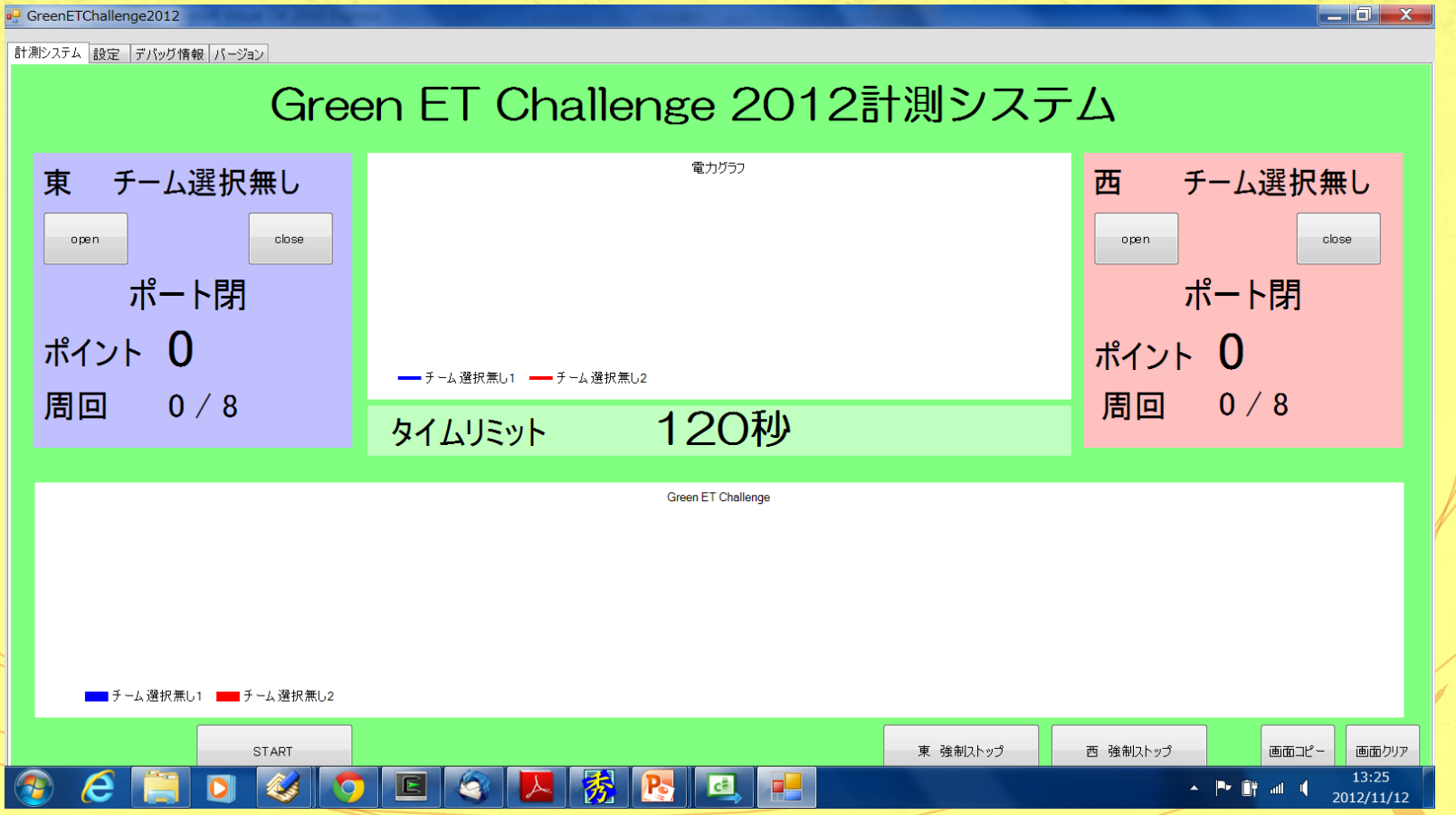
- COMポート番号の確認



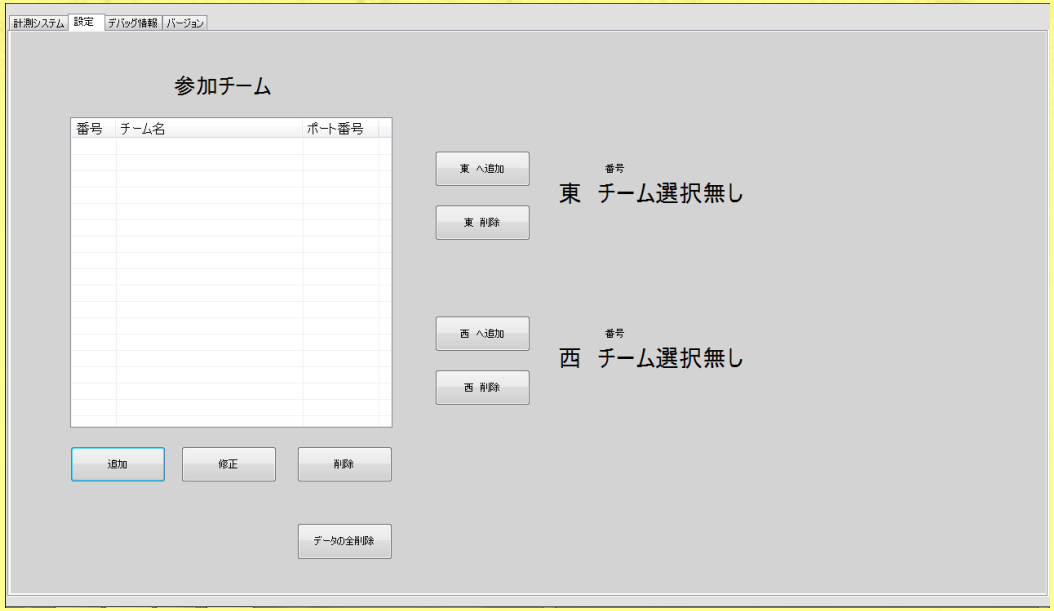
- Windows7の場合
  - プロパティ画面がでるので、「サービス」タブをクリックすると、COMポートの番号が確認できる。
  - 注意）Windows7はNXTを削除、追加を繰り返してもこの番号は変わらない。
  - Windows Vistaは、削除、追加を繰り返すとポート番号が増える一方になり、その番号を戻すには手こずるので注意

計測システム

- 計測システムの使用方法

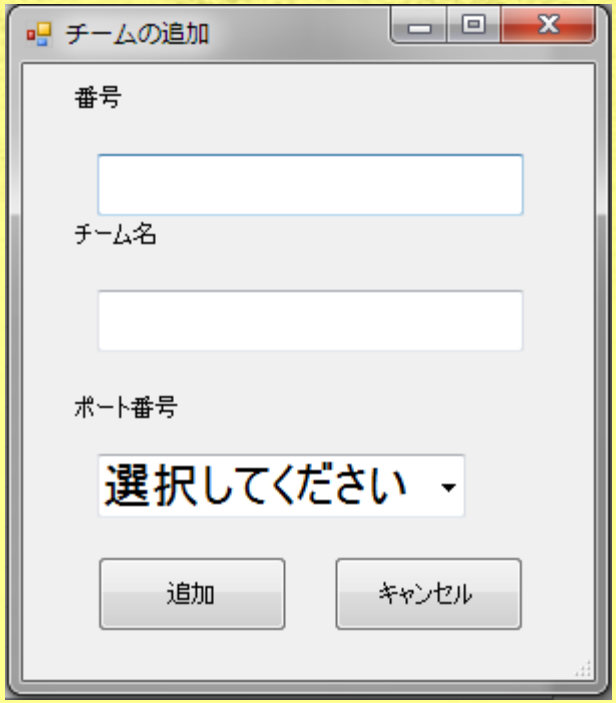


計測システム：チームの登録



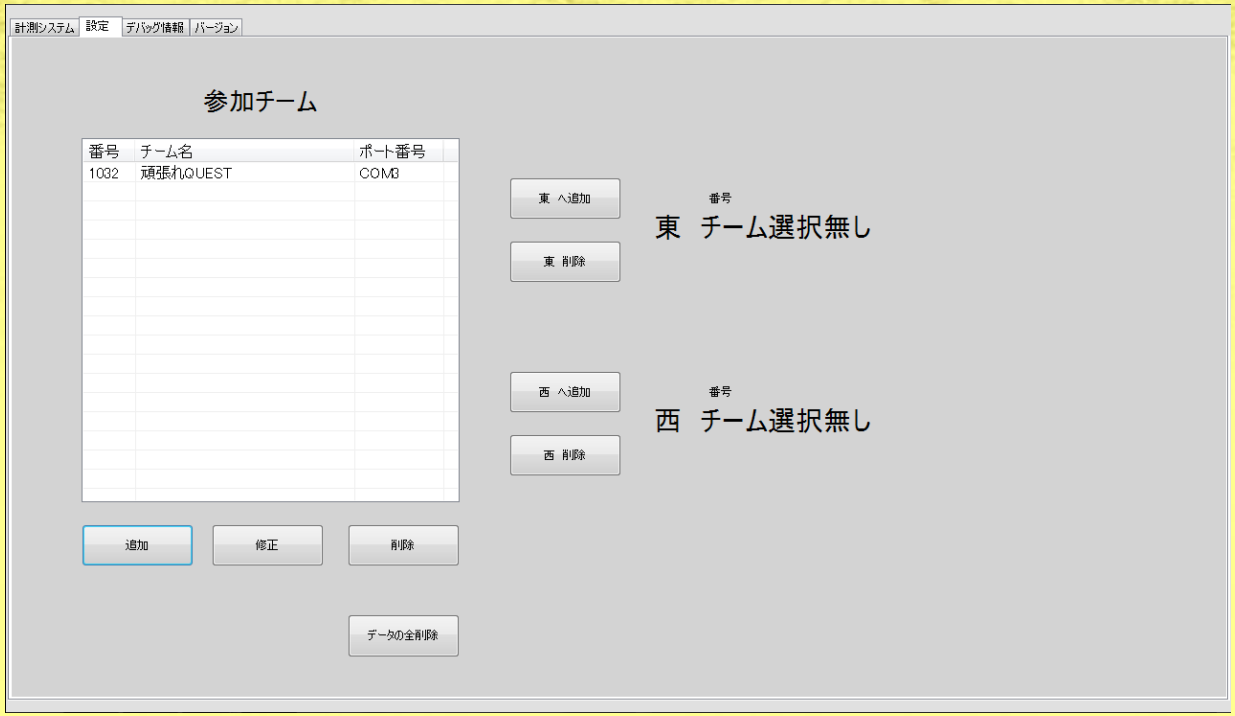
- 設定タブをクリックすると設定画面が表示される
- 追加ボタンを押下すると、チームが登録できる

計測システム：チームの登録



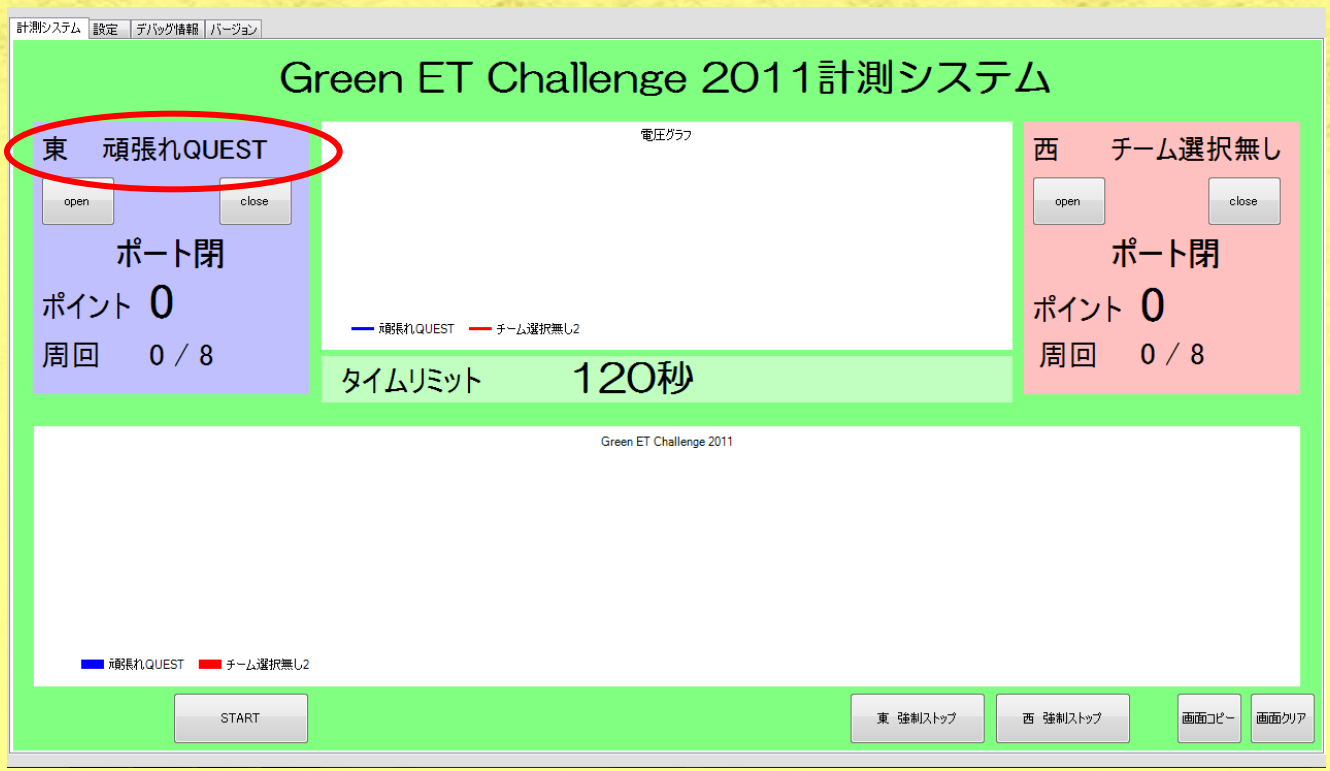
- 追加画面で、
  - 番号：チームID
  - チーム名：チーム名
  - ポート番号：上記で調べたCOMポート番号
- を入力して追加ボタンを押下する。

計測システム：チームの登録



- チームを東へ登録する場合は、リスト上のチームを選択して、「東へ追加」を押下する

計測システム：チームの登録



- 左上の計測システムのタグをクリックすると
- 東へチーム名が登録される。

プログラム実行からコース設置までの手順

- 電源を入れる
- 電源投入は、橙色のENTERボタンで行う
- 中央のにMyFilesがあればENTERキーを押下
  - なければ、灰色の矢印キーで中央にもってくる

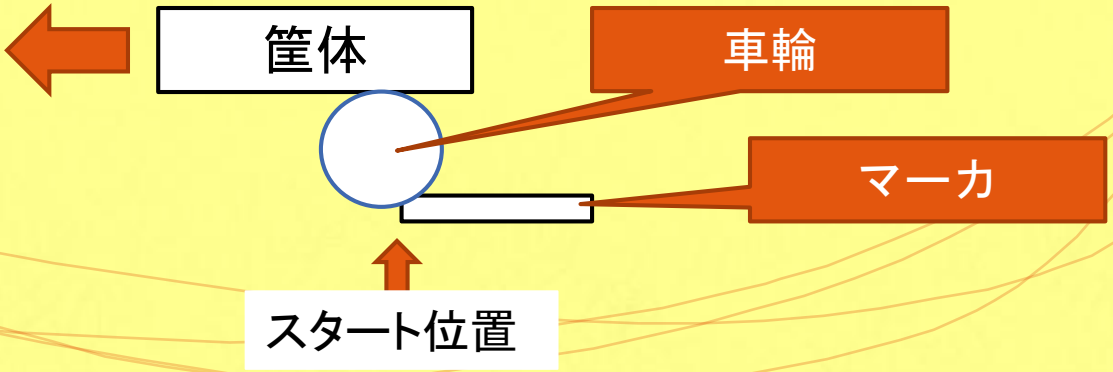
プログラム実行からコース設置までの手順

- 中央にSoftware filesがあれば、ENTERキーを押下
- 自分で作成したプログラム名が中央にあれば、ENTERキーを押下
- Runが中央にあれば、ENTERキーを押下

プログラム実行からコース設置までの手順

- プログラムが実行して一時停止する。
- 今年は、ここでBluetooth待ちとなるので、計測システムで「open」を押下
- その後、RUNをするために右矢印キーを押下
- ここから、実際のプログラムが走るので、自分が作成した通りに操作する。
  Demoプログラムの動きを下記に示す
  - 「黒の上に置け」とメッセージが出るので、マーカー以外のコース場所において、ENTERキーを押下
  - 「白の上に置け」とメッセージが出るので、マーカーの上において、ENTERキーを押下
  - この状態でコースに設置する

コースへの設置



計測システム

- 「停止中」と状態がなっているのを確認する
- 「停止中」であれば、左下の「START」ボタンで走行体が走りだす
- 状態は「走行中」に変化する
- 止める場合は、東あるいは西の強制ストップボタンを押下する
- 画面コピーボタンを押下すると画面のハードコピーを得ることができる

既知のバグ

- 計測システムにおいて、「画面クリア」を行なってもクリアしない場合がある。
  - 対策：アプリの再起動
- ２０１１年度のバグ
  - 計測システムでOPENエラーした時の対処が悪く、二度とOPEN処理することができなかった。
  - ２０１２年度ではNXTのプログラムがスタートする前にOPEN処理を行うことができるように変更した。
  - BluetoothのOPENエラーは色々ある
    - NXTがOPEN状態で接続できない
    - 計測システムがOPEN待ちでタイムアウトするまで接続できない等

既知のバグ

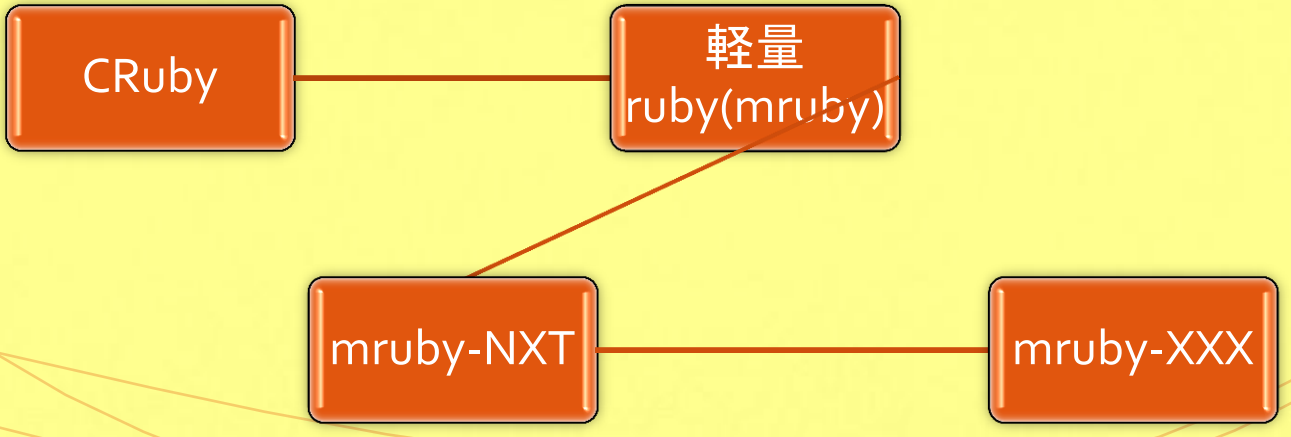
- 計測システムとNXTが接続されていない場合
  - NXTの画面で、[BT]と表示されている時はなんらかの原因で、ポートが掴まれている状態なので、NXT側の電源を落として再起動する

mruby-NXT

mruby-NXT

mruby-NXT

- 軽量rubyとは
  - 組込み開発向けに軽量化された新しいRuby
  - VM上でバイトコードを実行



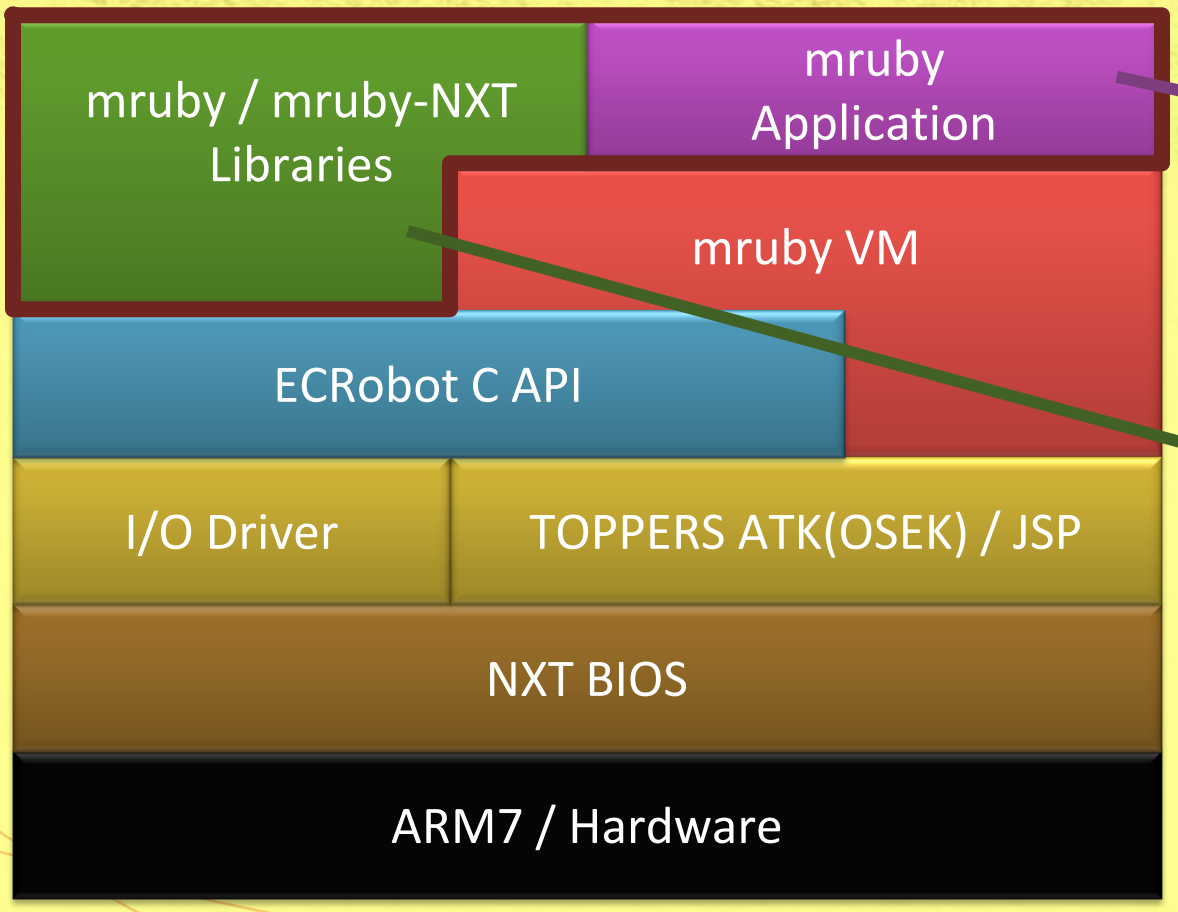
mruby-NXT

ETロボコン用にカスタマイズされたmruby

- 組込み向けRubyである **mruby** を広く知ってもらい、使ってもらうために、ETロボコンの開発環境として利用できるようにしたい・・・というところがスタート
- 素のmrubyではメモリサイズ的にNXTに搭載できないため、派生ソフトとして開発
- **mruby-NXT** は開発コードネーム
- 約190KB（ROM:157KB, RAM:34KB）まで軽量化

注）mruby-NXT 平成24年11月14日 NPO法人 QUEST, 福岡CSK　より抜粋

mruby-NXT



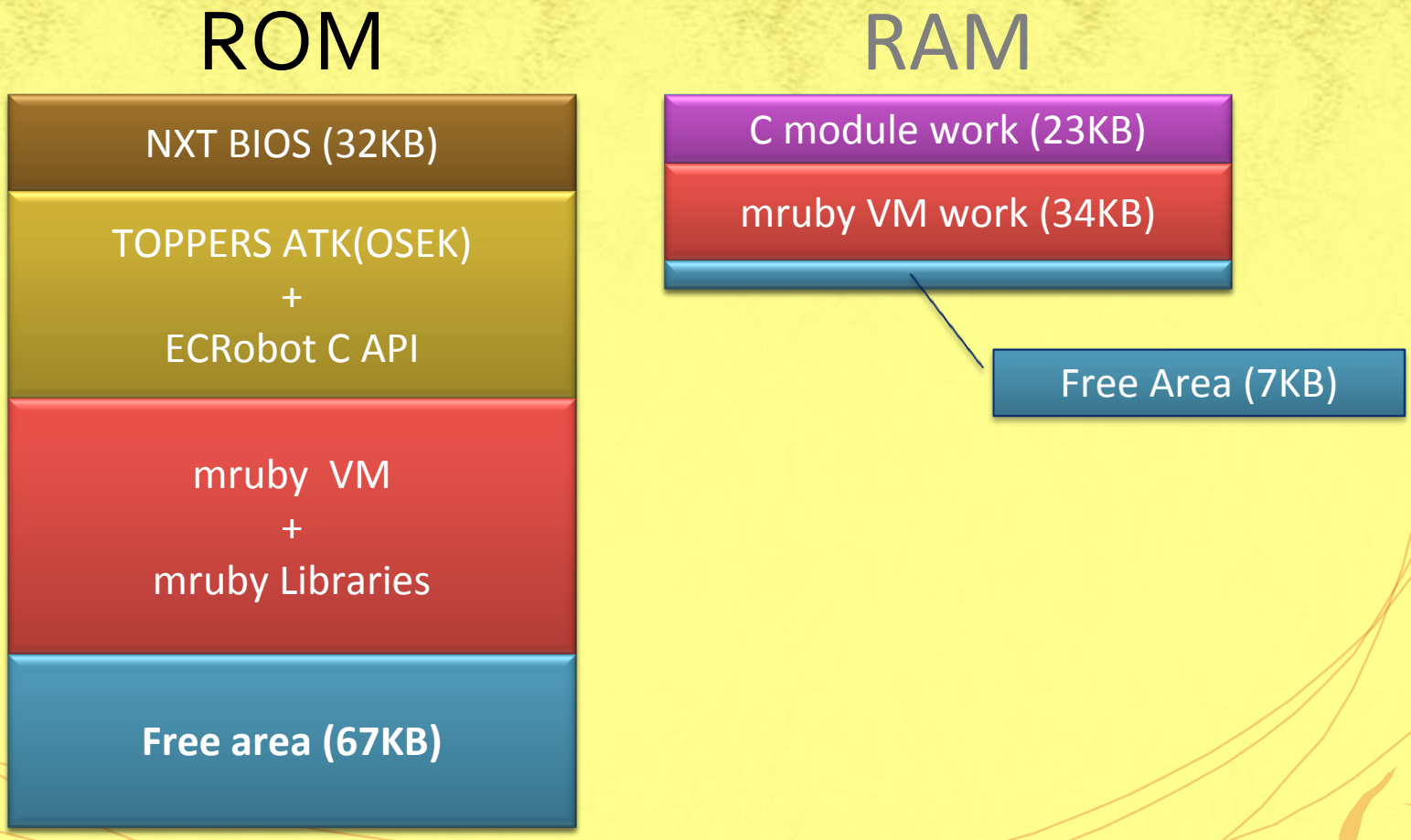
mrubyでアプリ開発を容易に!!

必要に応じてライブラリを追加可能!!
(C / Ruby)

注）mruby-NXT 平成24年11月14日 NPO法人 QUEST, 福岡CSK　より抜粋

mruby-NXT

mruby-NXTメモリ構成



注）mruby-NXT 平成24年11月14日 NPO法人 QUEST, 福岡CSK　より抜粋

mruby-NXT

ETロボコンへのチャレンジを容易に

- アプリケーション開発はmruby（Ruby言語）で記述
- モデルとコードの一貫性保持が容易
- ETロボコン用（NXT用）クラスライブラリ
  モータ、各種センサ、二輪倒立振子制御、Bluetooth …
- C言語による拡張も可能

注）mruby-NXT 平成24年11月14日 NPO法人 QUEST, 福岡CSK　より抜粋

mruby-NXT

- 狭いROM/RAM空間なのに、何がいいのか
  - 狭いROM/RAM空間だからこそ必要な機能がすでに組み込まれている
  - ユーザはアプリの動作に集中できる
  - 例えばモータ２個を動作させるためには
    ml = Motor.new Nxt::Constants::NXT_PORT_C
    mr = Motor.new Nxt::Constants::NXT_PORT_B
    mr.set_speed 100
    ml.set_speed 100
    これだけで、スピード100で前進します。
    mainもなければ、#includeもいりません

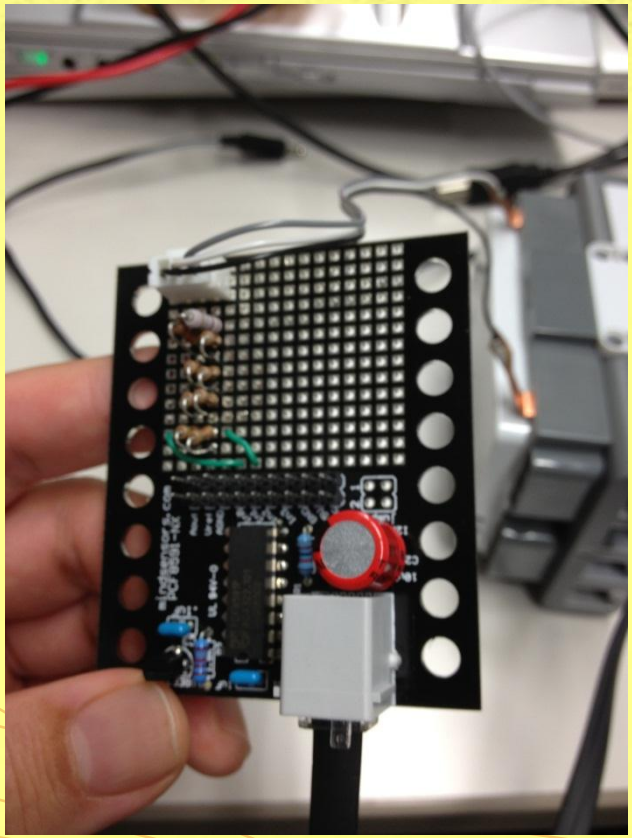
mruby-NXTの意味

- -NXTとついているが、NXT独自の拡張ライブラリを削除し、他の組込みシステムへ移植することが可能となる
- これは、ROM224k、RAM64k程度のマイコンでrubyで書かれたコードが実装できることを示している。
- 試作はruby、実装はC言語のシステムであれば、rubyコードをC言語に移植することなしに、システムへ導入することが可能

パワーメータ

パワーメータ

- 全体図



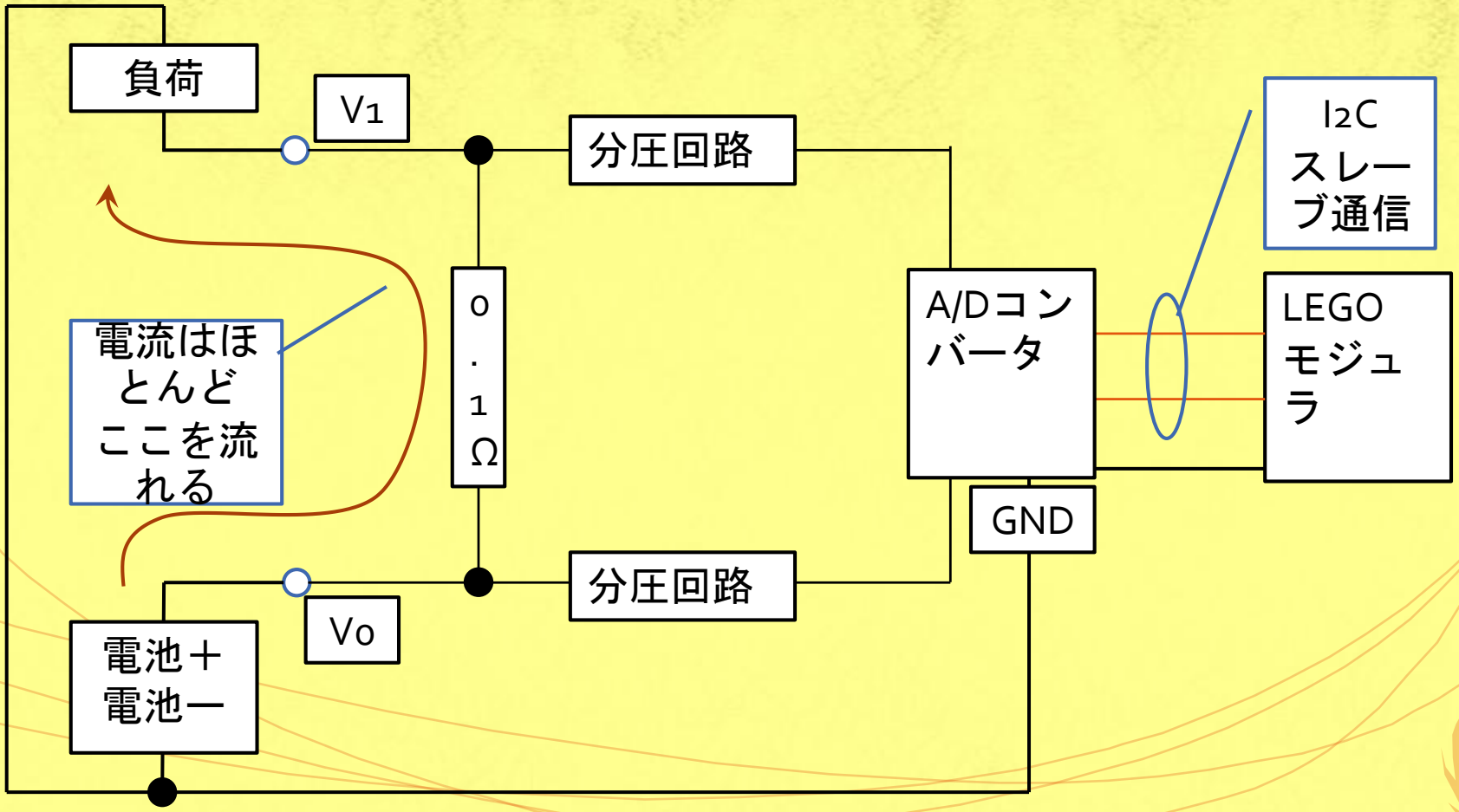
測定点

パワーメータ

- NXTの消費電力を単体で測定することは不可能
- ２０１１年度は電圧の変化の標準偏差を計算して測定結果としていた
- ２０１２年度は実際に電力を測定しその結果を計測システムへBluetoothへ飛ばすようにしている

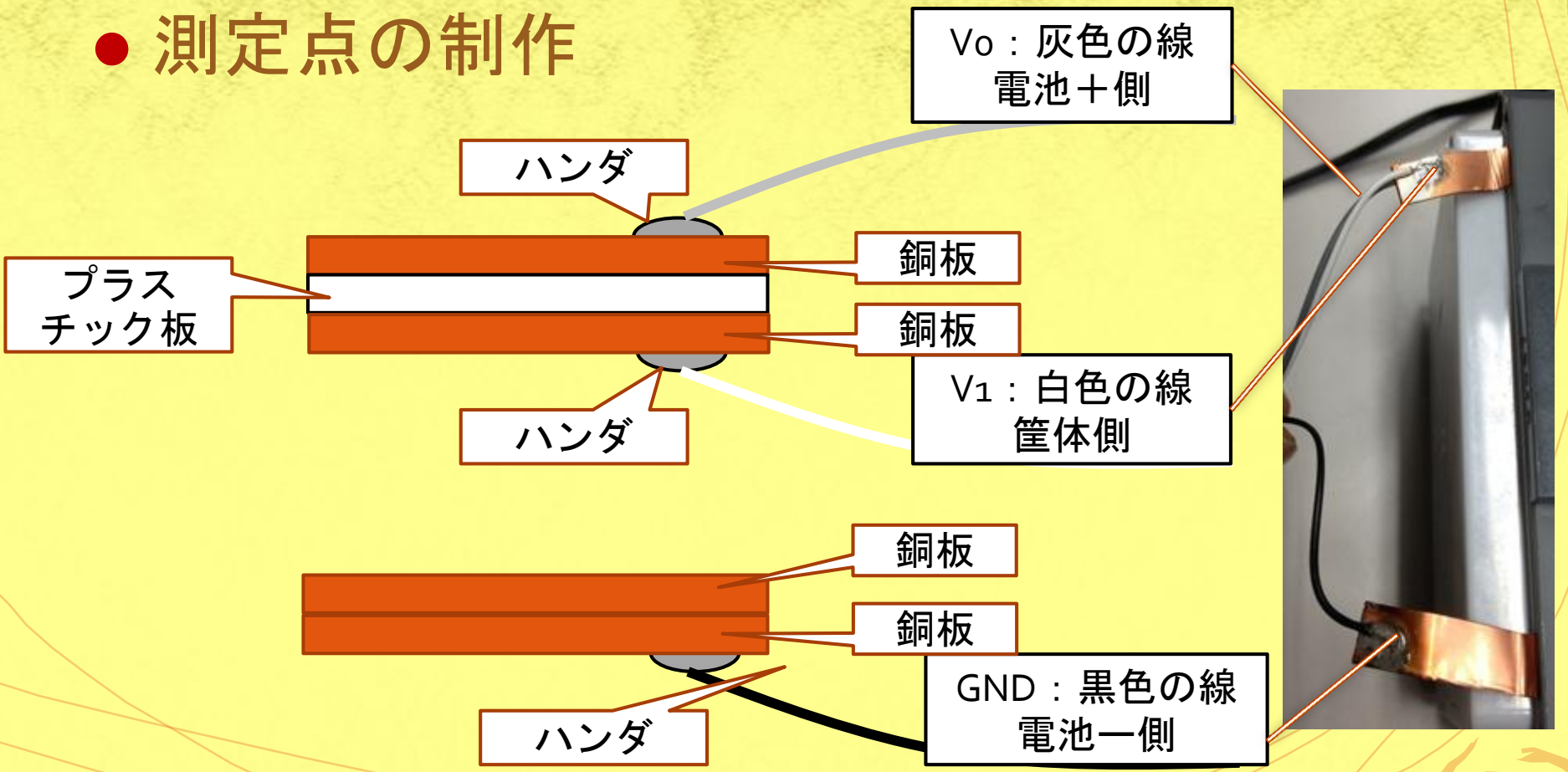
パワーメータ

- ブロック図



パワーメータ

- 測定点の制作



パワーメータ

- 計算式
  - 電力＝電圧 × 電流

P[W] = E[V] * I[A]
P[W] = Vo * (Vo – V1)/0.1
= Vo *(vo – V1) * 10

パワーメータ（I2C部プログラム）

```
#include <string.h>
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"
#include "power_meter.h"

static U8 v0 = 0;
static U8 v1 = 0;

void Pmr_init(int port){
        ecrobot_init_i2c(port , LOWSPEED);
}
```

```
void Pmr_adc(int port)
{
        static U8 data[2] = {0};
        //      ecrobot_read_i2c(port , 0x90>>1 , 0x04, data , 1);
        ecrobot_read_i2c(port , 0x90>>1 , 0x00, data , 1);

        v0 = data[0];
        ecrobot_read_i2c(port , 0x90>>1 , 0x01, data , 1);

        v1 = data[0];

}
U8 Pmr_get_v0()
{
        return v0;
}
U8 Pmr_get_v1()
{
        return v1;
}
```

アドレスは0x90
これを1ビット
シフト

コントロール
コードは0x00で
ch0、0x01でch1
指定

大会まで時間が足りません

- 大丈夫です
  - まずは、基本プログラムで動作確認してください
  - 基本プログラムでも十分低消費電力走行をします
  - PID制御で利用している定数kp,ki,kdを変更してみてください
    - PID制御用のこれらの定数は大変デリケートです。
    - この定数の変化に伴い、補正値が大きく変わってきます。
    - この補正値をグラフ化することにより、PIDの見える化ができますので、デバッグ機能を使ってチャレンジすると新しい発見があるかもしれません。

余裕でクリアしました

- 時間が余ったら
  - 基本プログラムを設計変更してみてください。
    - 筐体を独自に開発
      - 軽量化する（バラバラにならない程度に）
      - モータの数を減らす（無理かも）
      - 駆動モータ１、ステアリングモータ１にしてデフギアを作れば、消費電力はかなり減ります。
      - 製品として売っているデフギアは使用不可です（ETロボコンキットに入っていないため）
    - 悪いところを改良
    - 気になったところを改良
    - 性能的なところを改良
    - 状態遷移を改良（ふんわり発進等）

おわりに

ご清聴ありがとうございました。

修正履歴

日付	修正者	内容	備考
2012/11/12	山下	初版	1.0.0