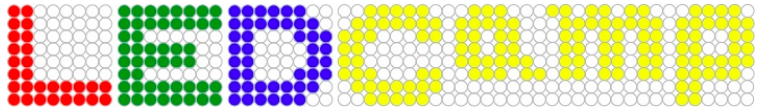
組込みシステム開発の勘所と実践

高瀬　英希



組込みシステムとは？

- 乱暴な定義をすれば，
  「汎用システム以外のコンピュータシステム」
- 各種の機器に組み込まれて，特定の機能を実現するためにその制御を行うコンピュータシステム
- アプリケーション特有の要求に対して特化して設計されるため，必然的に多様な専用システムになる
- 利用可能な資源が限られる
  - プロセッサ周波数，メモリ量，バス幅，．．．
  - 回路面積，製品サイズ，．．．
  - 信頼性，連続駆動時間，消費電力，．．．

組込みシステムの適用分野

組込みシステムの特性

- 機器の高機能化，複雑化，デジタル化
- 環境とのインタラクション
- 高い信頼性
- 特定用途向け（+「汎用機能」）
- 厳しいリソース制約
  - CPU能力，メモリ容量，製品サイズ，低消費電力
- ネットワーク接続性とセキュリティ強化
- リアルタイム性

組込みシステム開発の勘所

リアルタイム性を確保する

既存資産の仕様を理解する

– 実習教材システムの紹介

理想と現実のぶれを考慮する

– 外部環境とのシステム同定

クロス開発環境を整備する

– リモートデバッグ環境

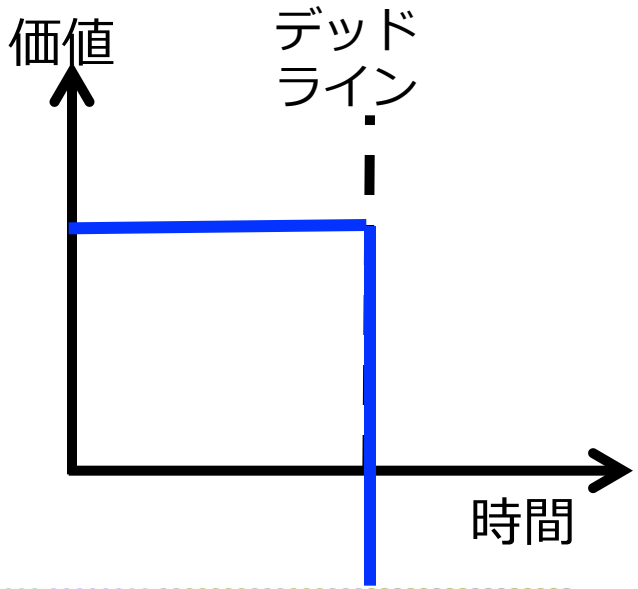
リアルタイム性

組込みシステムにはリアルタイム性が必要

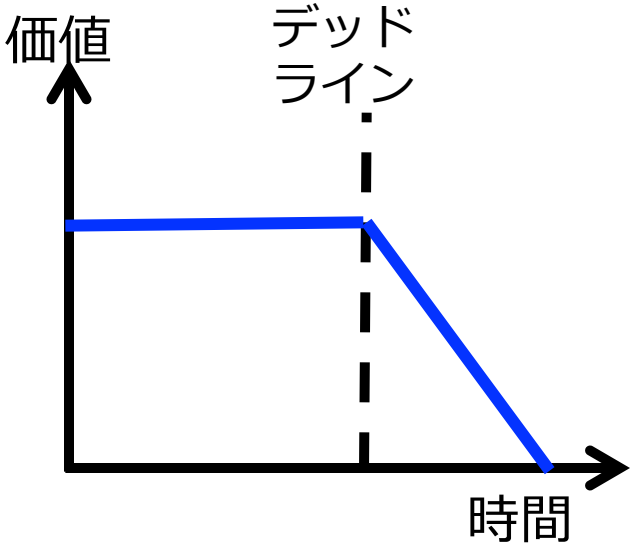
- タスクが定められた時刻（デッドライン）までに処理を終えるための性能
  - 処理結果の正確さに加えて，処理時間の正確さが求められる
- 入力から応答までの処理時間が一定に見積もれることが重要
  - 単に高速であればいいわけではない！
    - 例１：ブレーキシステム
      ○：ペダルを踏んだら常に3ms以内に制動が働く
      ×：最速1msで制動が働く，たまに10msくらい掛かることがある
    - 例２：動画再生プレーヤ
      ○：常に30fpsの処理性能を保証する
      ×：最高50fps，頻繁に10fpsまで性能が落ちる

リアルタイム性の分類

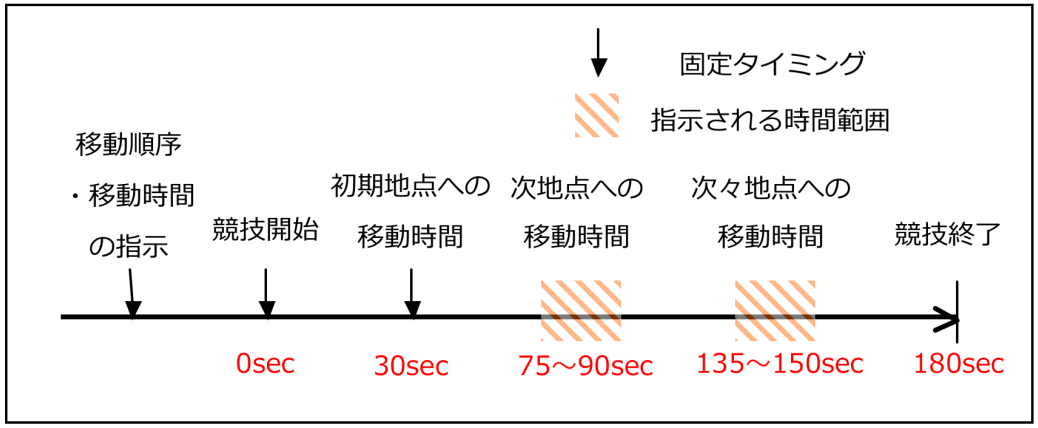
- ハードリアルタイム
  - デッドラインを過ぎると処理の価値がただちに0（以下）になる
  - 例：ブレーキ制御



- ソフトリアルタイム
  - デッドラインを過ぎると処理の価値が緩やかに減少していく
  - 例：動画再生プレーヤ

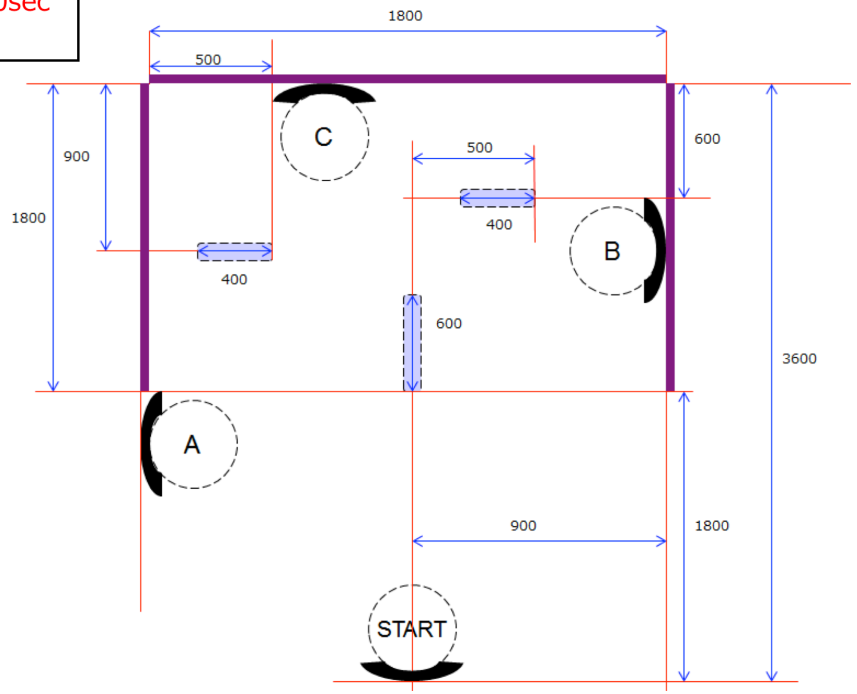


LED-Camp2競技会のミッション



地点間を指定された
順序かつ時間で移動する

処理時間を考慮して
ソフトウェアを設計する



リアルタイム性向上のために

- 時間制約を洗い出す
  - 入力から処理までの時間
  - 処理そのものに掛ける時間 ← 本実習の着眼点
  - 処理終了から出力までの時間
- 処理を適切にタスク分割する
  - システムの機能と重要度に応じて分割する
    - 重要度＝優先度
  - モジュール化のためにも重要（生産性・保守性UP）

- 今回の教材では実践が難しい，，，

組込みシステム開発の勘所

リアルタイム性を確保する

既存資産の仕様を理解する

– 実習教材システムの紹介

理想と現実のぶれを考慮する

– 外部環境とのシステム同定

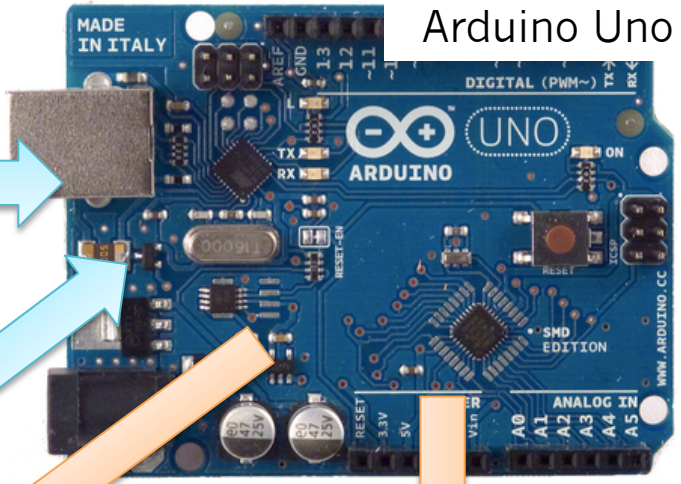
クロス開発環境を整備する

– リモートデバッグ環境

実習教材システムの構成

超音波距離センサ

物体との距離



Arduino Uno R3

検知系の状態

制御コマンド

デバッグデータ
書き込み

ROS

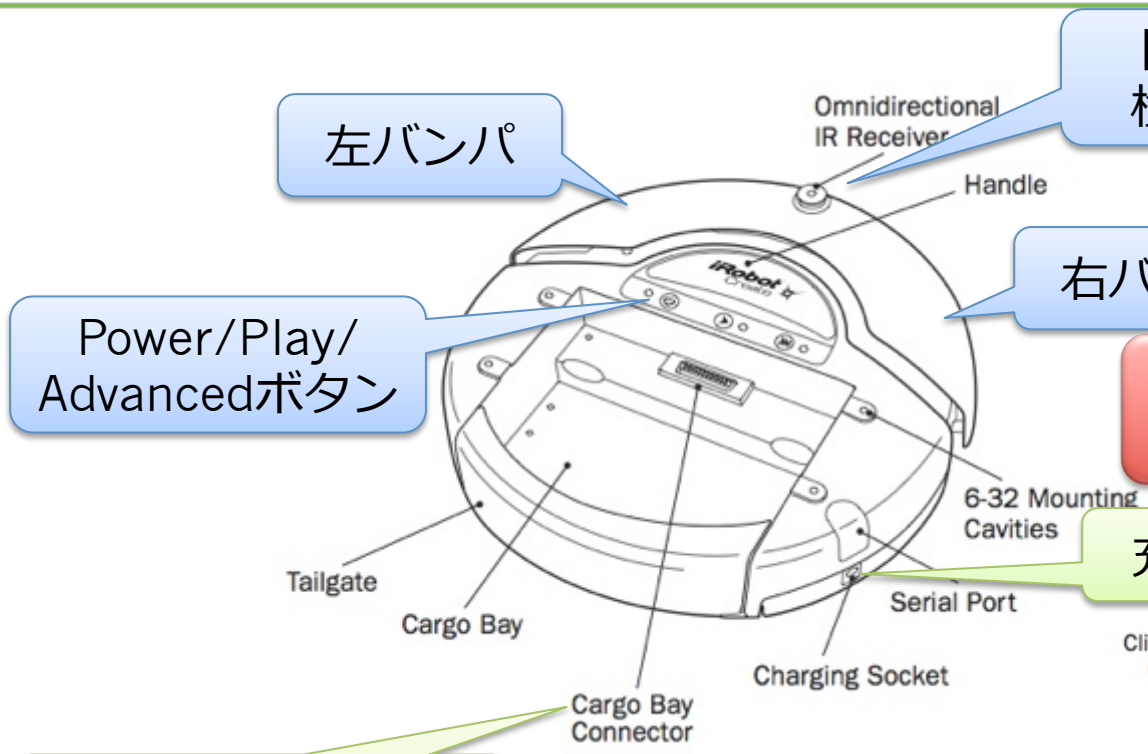
FlashAir

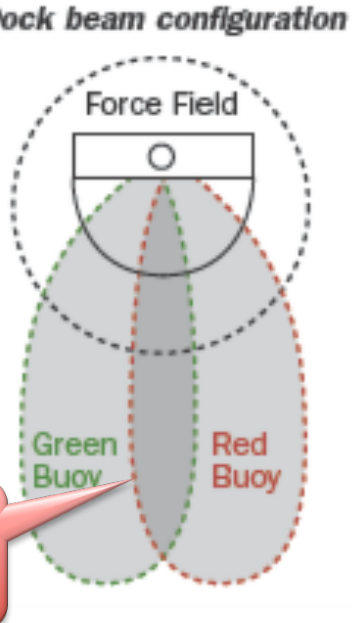
iRobot Create

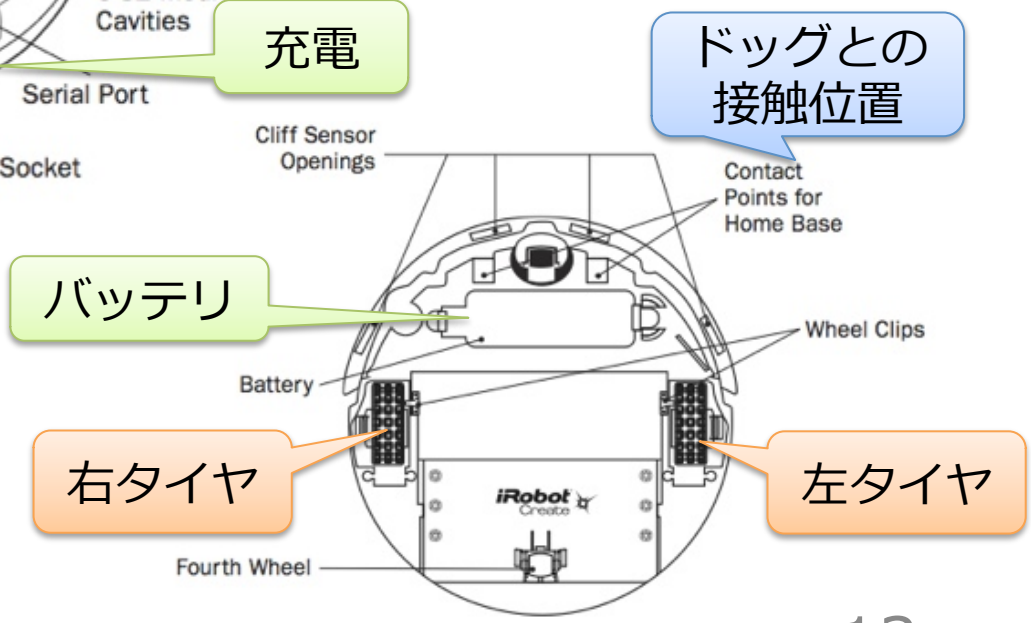
iRobot Create

- 掃除機型ロボット
  - お掃除機能の無いRoomba
- 主要な検知系
  - 左右バンパ
  - IRセンサ（ドッグ位置検知）
  - クリフセンサ
  - ボタン×3
- 主要な制御系
  - 左右タイヤ
  - LED×3
- コマンドモジュール
  - ホストからは8ビットのシリアル通信でやり取り
  - ROS (Robot Operating Systems) による隠蔽化
    - CPU : Atmel AVR ATMega168 8-bit RISK 18.432MHz
    - Flash memory 14336bytes (14kbyte) EEPROM 512bytes

iRobot Create







Createの動作モード

- Off Mode
- Passive Mode
  - Play/Advancedの操作で10種類のデモが動作できる
  - センサ値の取得はできる
- Safe Mode
  - 制御コマンドのやり取りによって，センサ値の取得および動作の制御ができる
  - 落下検知，脱輪または充電時には緊急停止する
- Full Mode
  - 制御コマンドのやり取りによって，センサ値の取得および動作の制御ができる

Create API

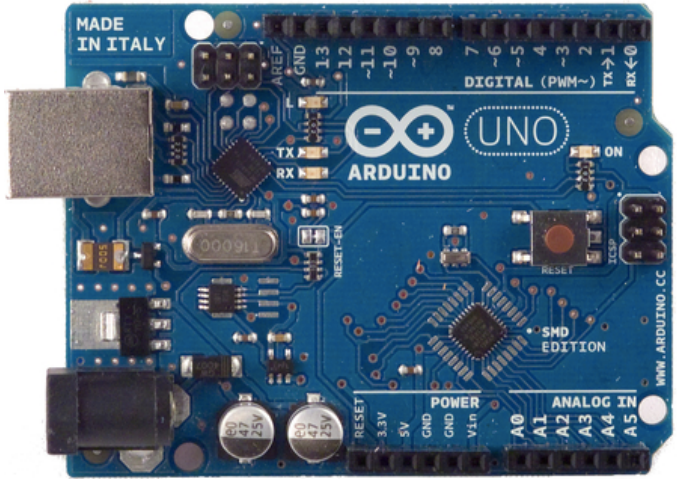
- 詳細は資料集末尾のAPIリファレンスにて
- センサ系
  - 各種検知系の状態や値を取得する
  - APIによって振る舞いや戻り値の意味が異なることに注意すること
- スタート制御系
  - 動作モードを指定しつつスタートする
  - 動作モードを変更する
- 動作制御系
  - Createの動作（主にタイヤ）を制御する

Maxbotix LV-EZ1

- 超音波の送受信で物体までの距離を測るセンサ
- 主要なスペック
  - 測定範囲：0インチ～254インチ（約6.45mまで）
  - 分解能：1インチ（約2.54cm）
  - 超音波周波数：42kHz
- 出力形式
  - シリアル
  - アナログ電圧（PWMをCRで平滑）
  - パルス幅出力
- API (getRange) の戻り値の単位は「mm」

Arduino

- オープンソース・ハードウェア
  - ハードウェア設計が公表されている
  - CC BY-SA（表示・継承）で再利用が可能
- Arduino IDE：統合開発環境
  - 詳細は後述
- Arduino Uno R3
  - マイコン：ATmega328P (AVR 8-bit MCU / 16MHz)
  - 動作電圧：5V　入力電圧（推奨）：7～12V
  - デジタルI/Oピン：14本（うち6本はPWM出力可）
  - アナログ入力ピン：6本（デジタルI/Oとしても使用可）
  - Flash：32KB （うち0.5KBはブートローダ）
  - SRAM:2KB　EEPROM:1KB



FlashAir

- 一般的には「無線LAN搭載SDHCメモリカード」
- 実際は，
  フラッシュメモリ
  ＋無線LAN
  ＋Webサーバ

- LED-Camp2では，無線でのリアルタイムデバッグに利用する
  - 詳細は後述

組込みシステム開発の勘所

リアルタイム性を確保する

既存資産の仕様を理解する

– 実習教材システムの紹介

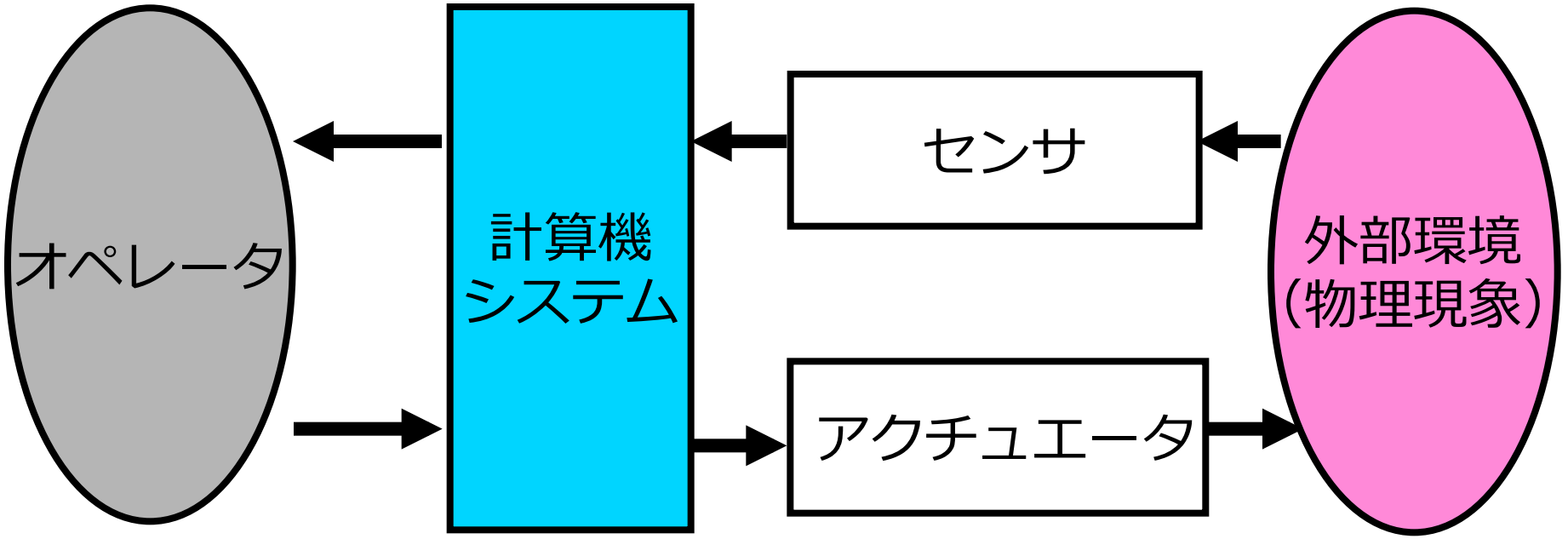
理想と現実のぶれを考慮する

– 外部環境とのシステム同定

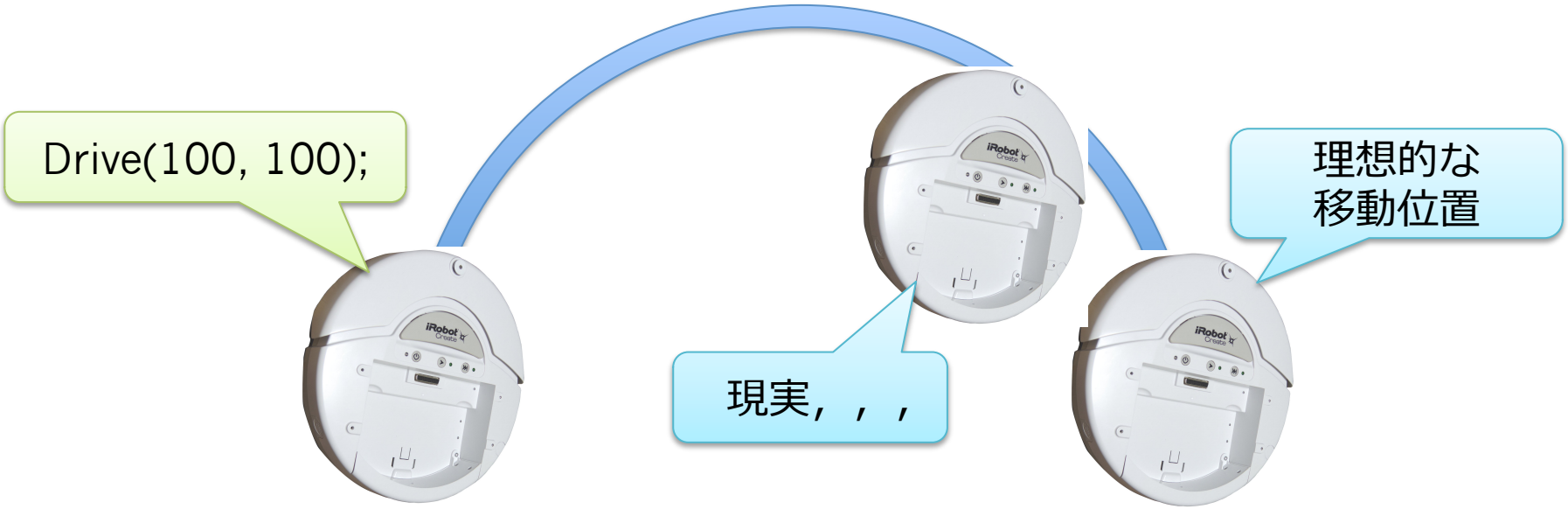
クロス開発環境を整備する

– リモートデバッグ環境

外部環境とのインタラクション



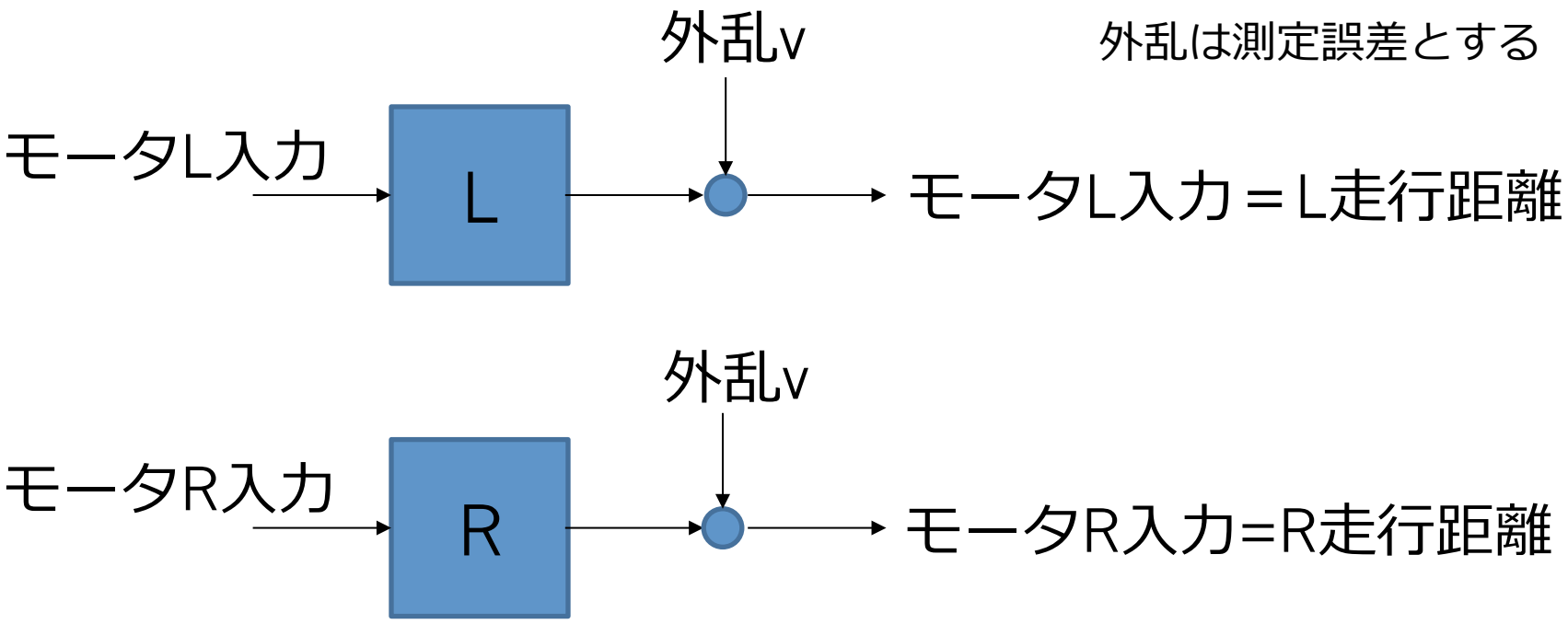
理想とのぶれ



ずれの要因：
- 個体差
- センサ特性
- 床面の材質
- タイヤの摩耗度
- etc, etc, …

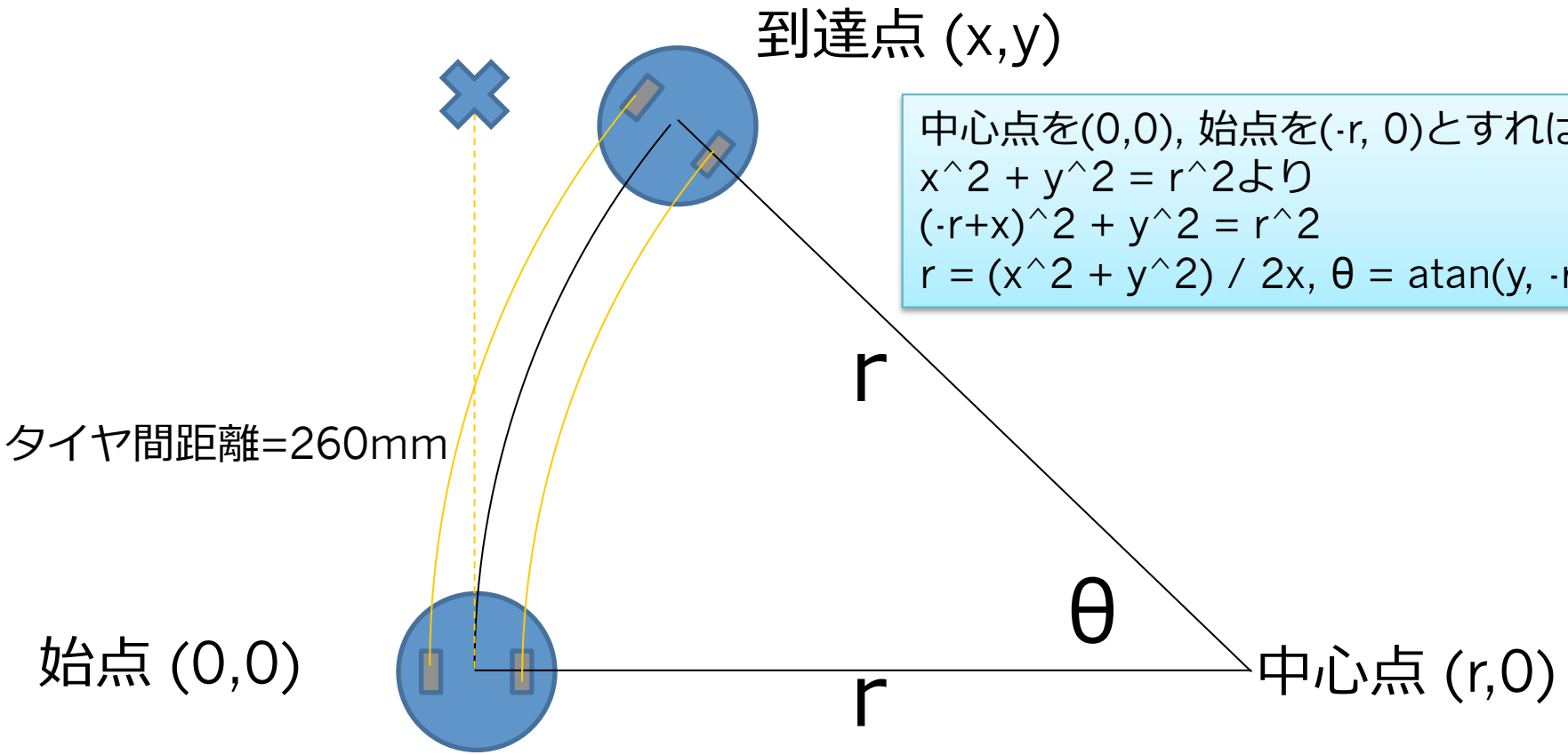
→ 最小二乗法を用いて
パラメータを調整する

左右モータの出力



- L，Rそれぞれのずれの要因になる関数を知りたい
- 逆関数を求められれば，ちゃんと指定した通りの距離を走ってくれるようになるはず

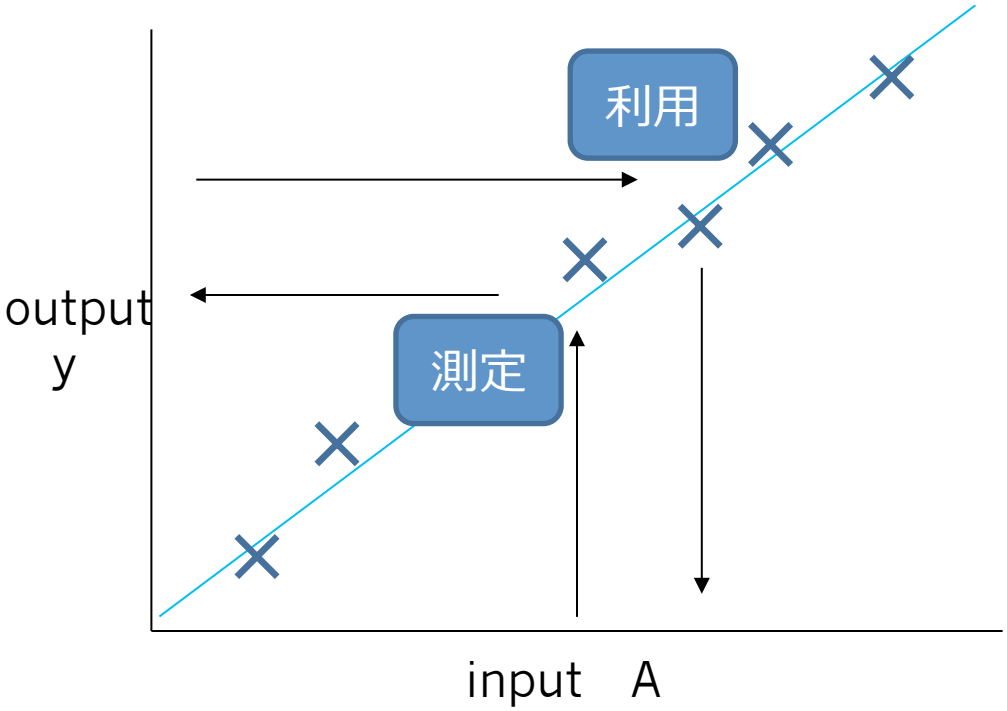
計測



左モータ走行距離 = 2 π * (r+L) / θ
右モータ走行距離 = 2 π * (r-L) / θ

最小二乗法による逆関数の導出

$$y = A\,x_1 + x_2$$



1入力1出力　1次関数

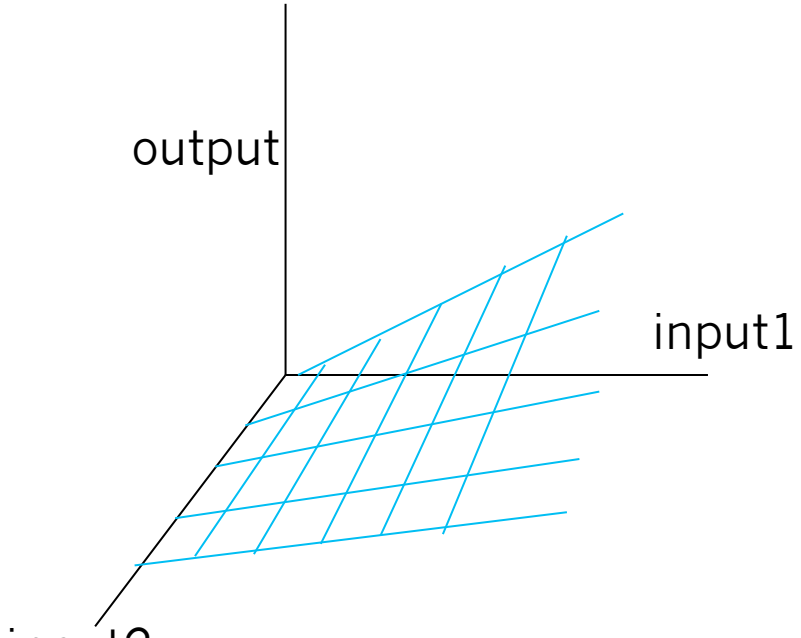
$$x_1 = \frac{N\sum AiYi - \sum Ai\sum Yi}{N\sum Ai^2 - (\sum Ai)^2}$$

$$x_2 = \frac{-\sum Ai\sum AiYi + \sum Ai^2\sum Yi}{N\sum Ai^2 - (\sum Ai)^2}$$

・逆関数　$A = \dfrac{y - x_2}{x_1}$

ここまでの例は1入力1出力

- 実際には考慮すべき入力値はまだある
  - 速度：複数の設定速度それぞれで求める
  - バッテリー：本システムではほぼ影響しない



2入力1出力
平面関数

組込みシステム開発の勘所

リアルタイム性を確保する

既存資産の仕様を理解する

– 実習教材システムの紹介

理想と現実のぶれを考慮する

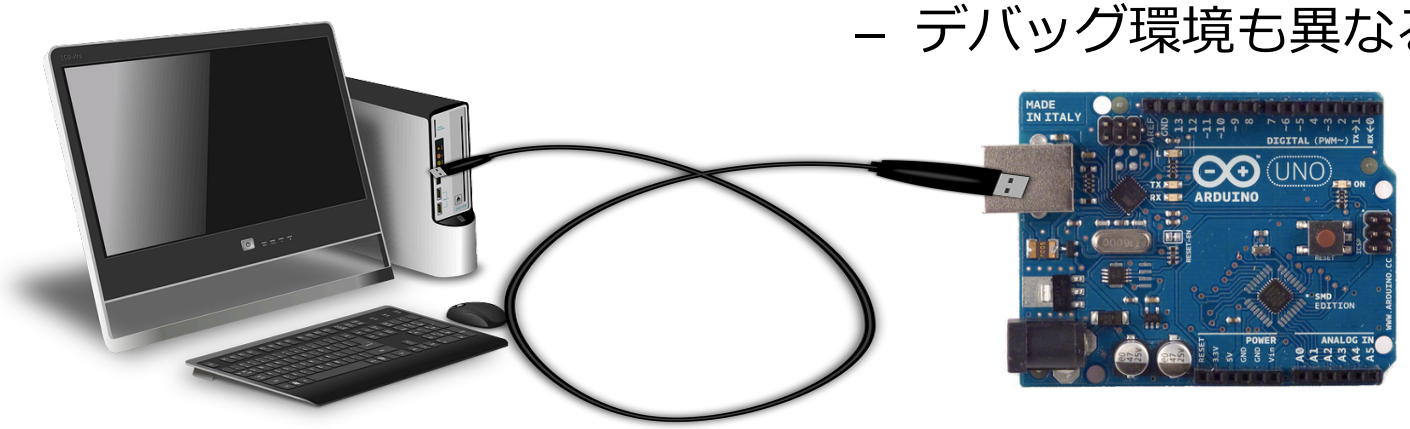
– 外部環境とのシステム同定

クロス開発環境を整備する

– リモートデバッグ環境

クロス開発環境

- セルフ開発
  - プログラムの開発環境と実行環境が同じ
- クロス開発
  - プログラムの開発環境と実行環境が異なる
  - デバッグ環境も異なる

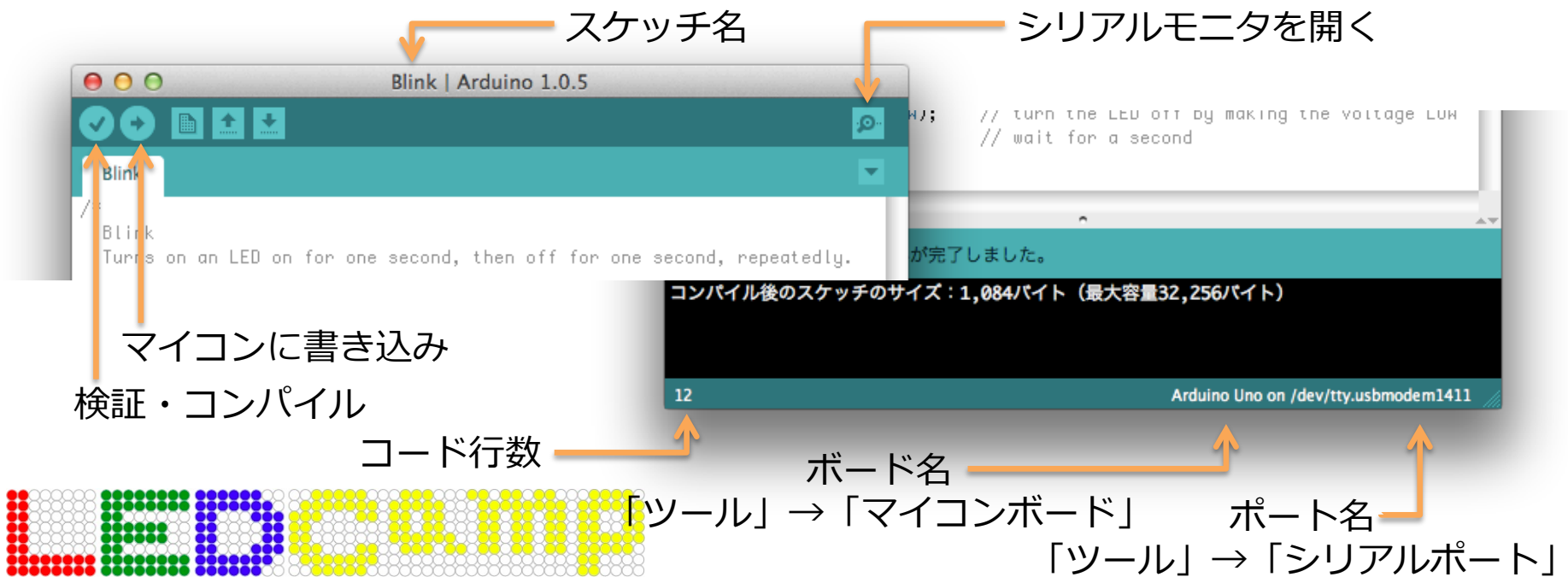


組込みではクロス開発が当たり前

開発環境・デバッグ環境を
整備することが重要！！

Arduino IDE

- エディタ･コンパイラ･ダウンロード･デバッグの各機能が統合された開発環境
  - 内部で gcc や avrdude が動作
  - 注意点：Arduinoのプログラム書き換えの際は Createの電源を切っておくこと！



Arduinoのプログラミングモデル

- Arduino言語
  - C言語のすべての構造
  - 一部のC++の機能
    - クラスなど

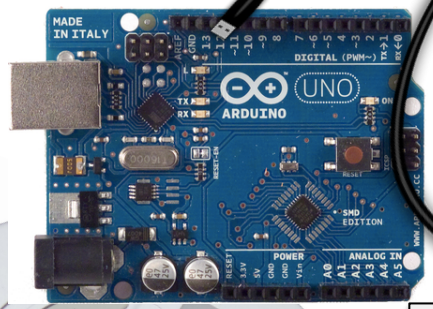
setup()
最初に一回だけ実行される

loop()
繰り返し実行される

delay()
m秒単位で遅延させる

シリアルデバッグ

Arduinoと
ノートPCを
ケーブル接続

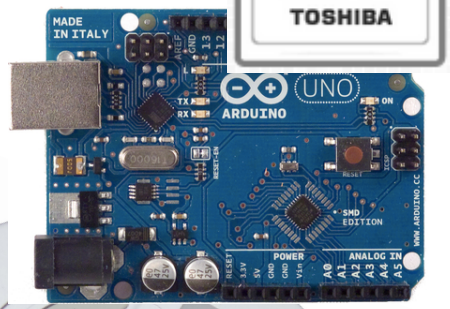


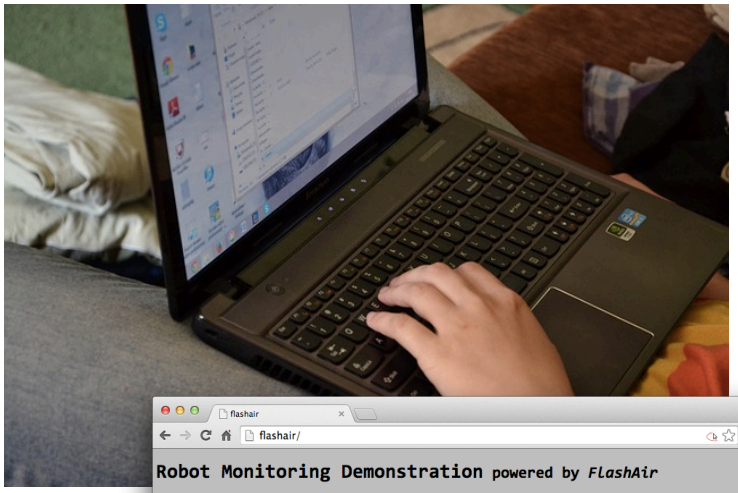

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("Hello!!");
  Serial.println(num);
  num++;
  delay(1000);
}
```

シリアルモニタで
出力を確認

FlashAirを使った無線デバッグ

無線で
データ通信





flashair/

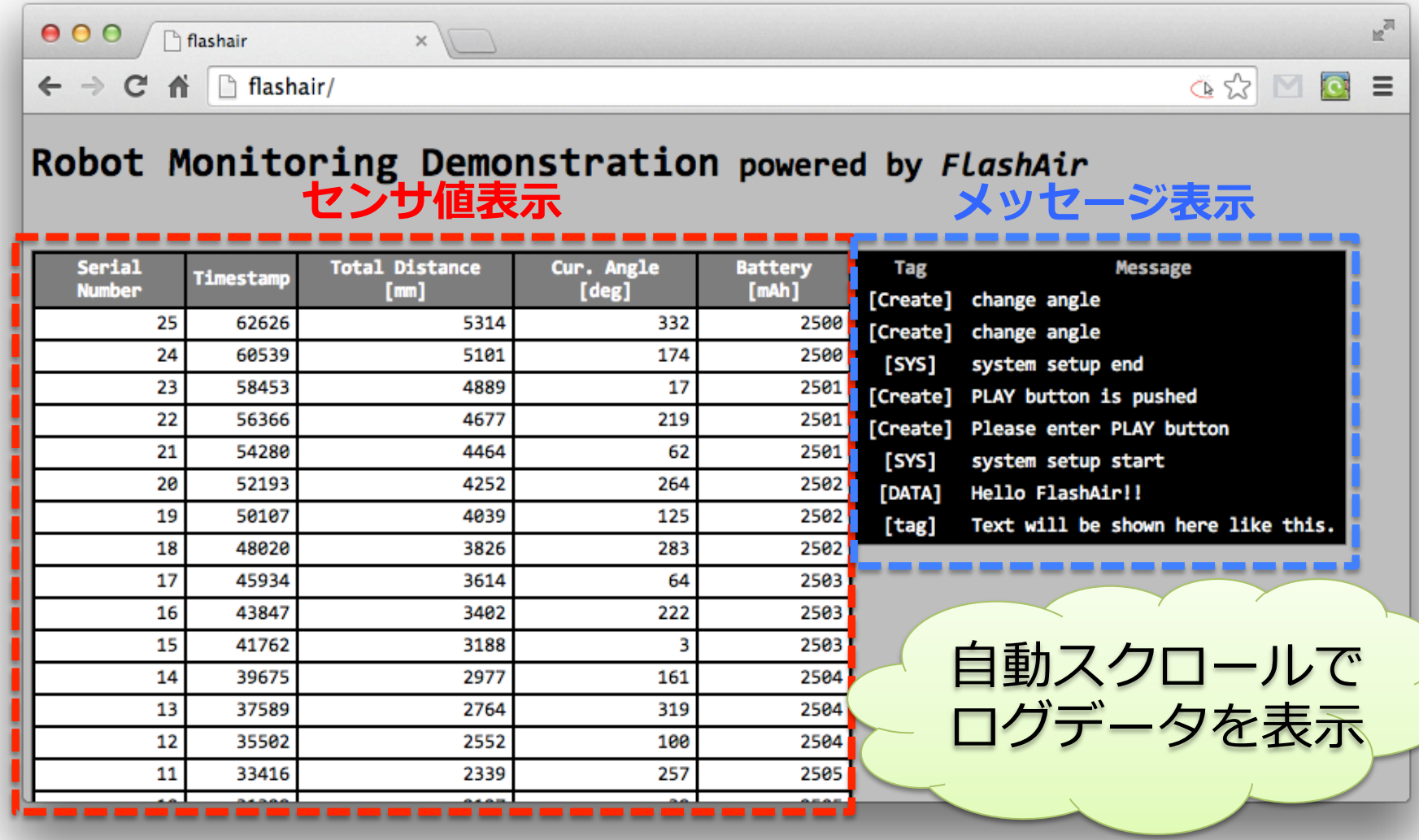
Robot Monitoring Demonstration powered by *FlashAir*

Serial Number	Timestamp	Total Distance [mm]	Cur. Angle [deg]	Battery [%]
6173	158	100	300	94
6075	139	100	300	94
5977	120	100	300	94
5880	102	100	300	94
5782	83	100	300	94
5683	64	100	300	94
5585	45	100	300	94
5487	26	100	300	94
5389	7	100	300	94
5292	348	100	300	94
5194	329	100	300	94
5096	311	100	300	94
4998	292	100	300	94
4900	273	100	300	94
4802	254	100	300	94
4704	236	255	300	94
4606	217	100	300	94
4509	198	100	300	94
4411	179	100	300	94

Tag	Message
[INFO]	System shut down.
[ERROR]	Gave up
[WARN]	Trying to recovery...
[WARN]	Trying to recovery...
[WARN]	Trying to recovery...
[INFO]	Tire trouble detected.
[INFO]	System started.
[INFO]	Initializing system...
[tag]	Text will be shown here like this.

Webブラウザ上で
リアルタイム表示

デバッグ画面のブラウザ表示



Serial Number	Timestamp	Total Distance [mm]	Cur. Angle [deg]	Battery [mAh]
25	62626	5314	332	2500
24	60539	5101	174	2500
23	58453	4889	17	2501
22	56366	4677	219	2501
21	54280	4464	62	2501
20	52193	4252	264	2502
19	50107	4039	125	2502
18	48020	3826	283	2502
17	45934	3614	64	2503
16	43847	3402	222	2503
15	41762	3188	3	2503
14	39675	2977	161	2504
13	37589	2764	319	2504
12	35502	2552	100	2504
11	33416	2339	257	2505

Tag	Message
[Create]	change angle
[Create]	change angle
[SYS]	system setup end
[Create]	PLAY button is pushed
[Create]	Please enter PLAY button
[SYS]	system setup start
[DATA]	Hello FlashAir!!
[tag]	Text will be shown here like this.

デバッグデータの出力方法

資料集から修正

- 各種センサ値の出力：DATA.CSV
  - 形式：[serialNumber], [elapsedTime], [totalDistance], [currentAngle], [battery]
  - API例：CreateDataSDWriter.printCreateData(create);
- printf デバッグ：PRINT.CSV
  - 形式：[Num],[Tag],[Message]
    - [Num] は1から開始してインクリメントすること
  - API例：SDWriter.printf(“print.csv”, "%d,DATA,Hello FlashAir!!\n", + +print_counter);
- FlashAirへの接続方法は配布資料を参照

FlashAirデバッグの注意点①

資料集から修正

- printCreateData APIの内部で getDistance(), getAngle() を使用している
  - getDistance(), getAngle() は前回実行時からの差分を返しており，printCreateData APIは内部で値を累積している
- ログの取得／表示にオーバヘッドが掛かる
  - ログ取得：SDカードへの書き込み
  - ログ表示：ネットワーク接続・スクリプト実行

→ デバッグ機能の有無で振る舞いは変わる！！

FlashAirデバッグの注意点②

資料集から修正

- FlashAirへの接続が安定しない場合にはモバイルバッテリでArduinoに給電する
  - FlashAirの電力供給が安定しないため
  - CreateとArduinoの電源ライン（赤線）は抜いておく
- 未使用時はFlashAir接続を遮断する
  - トラフィックの負荷軽減のため
    - ルータの同時接続数は50
  - FlashAirのWebサーバは1リクエストずつ逐次処理する

組込みシステム開発の勘所

リアルタイム性を確保する

既存資産の仕様を理解する

理想と現実のぶれを考慮する

クロス開発環境を整備する

本講義の内容を
“まずは”チーム開発実習に
活かして取り組みましょう！