AI/IoTセンサのしくみを知ろう（応用編）

徳島大学技術支援部常三島技術部門
技術専門職員　辻　明典　博士（工学）
E-mail: a-tsuji@is.tokushima-u.ac.jp

2019年11月30日（土）

Copyright 2018-2019, Akinori Tsuji, All rights reserved.

講座内容

- 講師：辻　　明典（徳島大学技術支援部）
  桑折　範彦（徳島大学名誉教授）
  川上　博　　（徳島大学名誉教授）
- 土曜日：10:00～11:30
- 日程：

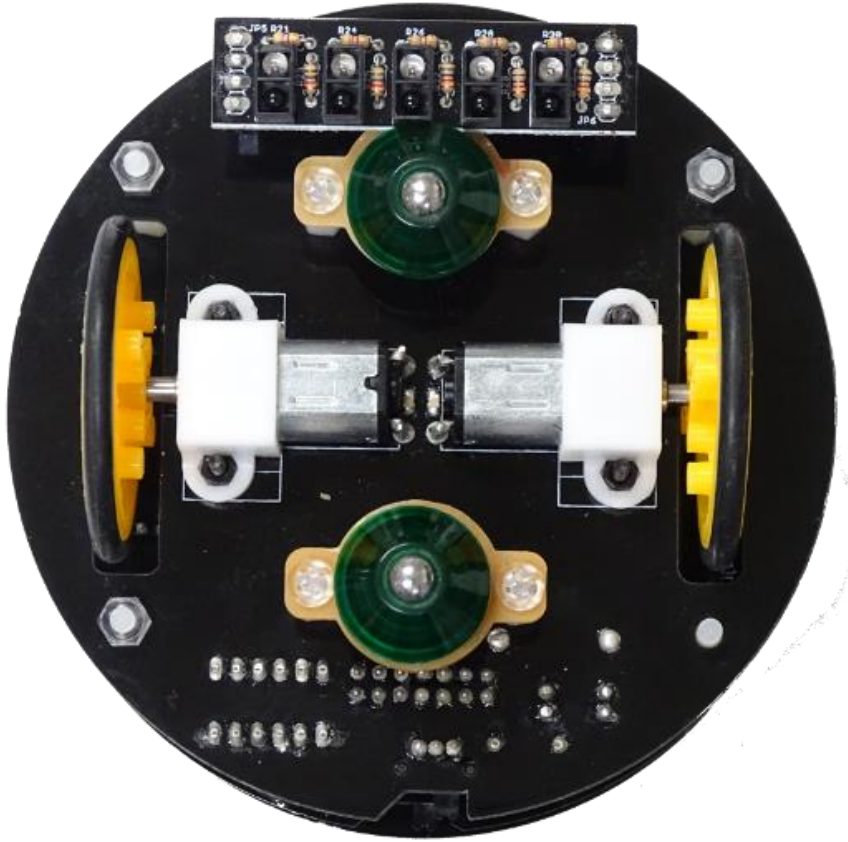
① 10 / 5　概要，環境設定，配布部品の確認
② 10 / 19　復習
③ 10 / 26　ロボットのモーター1（基本動作）
④ 11 / 9　ロボットのモーター2（応用動作）
⑤ 11 / 16　ロボットのセンサ1
（距離センサとモーター）
⑥ 11 / 30　ロボットのセンサ2
（フォトリフレクタ）
⑦ 12 / 7　ロボットの制御1
（ライントレース1）
⑧ 12 / 14　ロボットの制御2
（ライントレース2）
⑨ 12 / 21　ロボットの制御3
（迷路探索）

Copyright 2018-2019, Akinori Tsuji, All rights reserved.

フォトリフレクタの取り付け

- ロボットにフォトリフレクタを取り付け
  ※ 取り付け向き注意

フォトリフレクタ

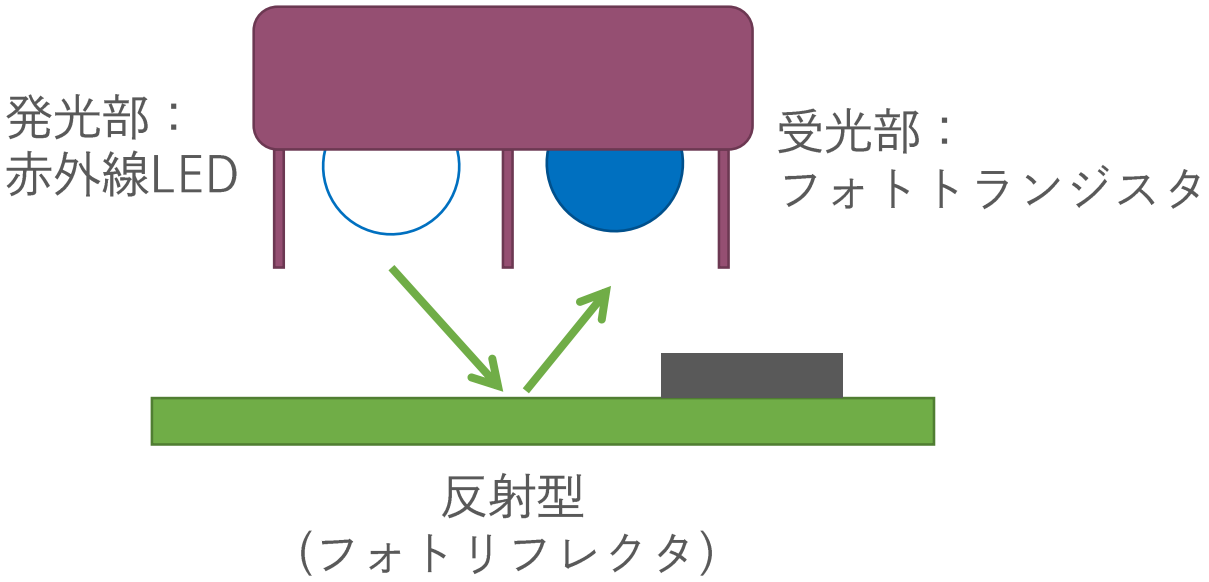


取り付け向き（ロボット裏）

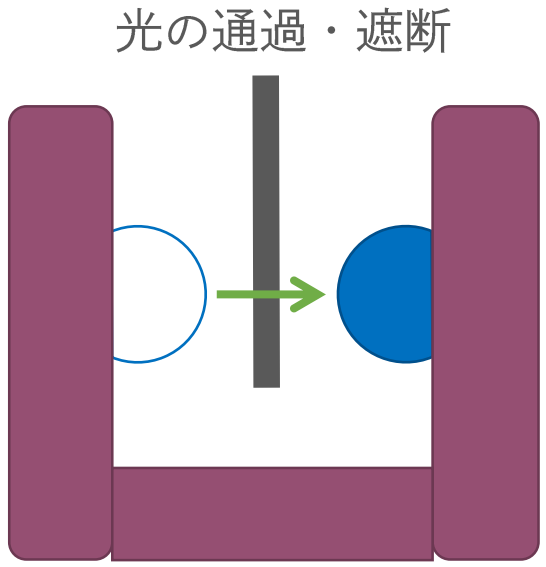
Copyright 2018-2019, Akinori Tsuji, All rights reserved.

フォトリフレクタの動作原理

- 赤外線の光を使い、物体の光の反射を用いて計測
  - 反射型と透過型の2種類
  - 検出距離：1mm～10mm程度
  - プリンタ，モーター，スマートフォンなど



反射型
（フォトリフレクタ）

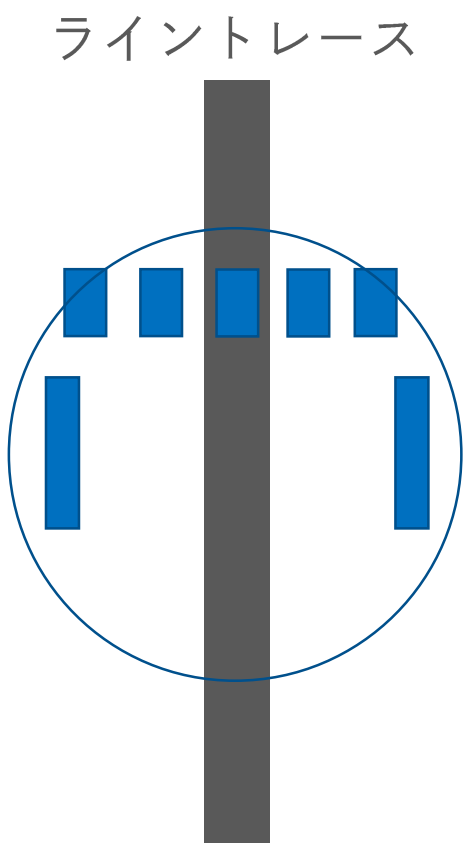


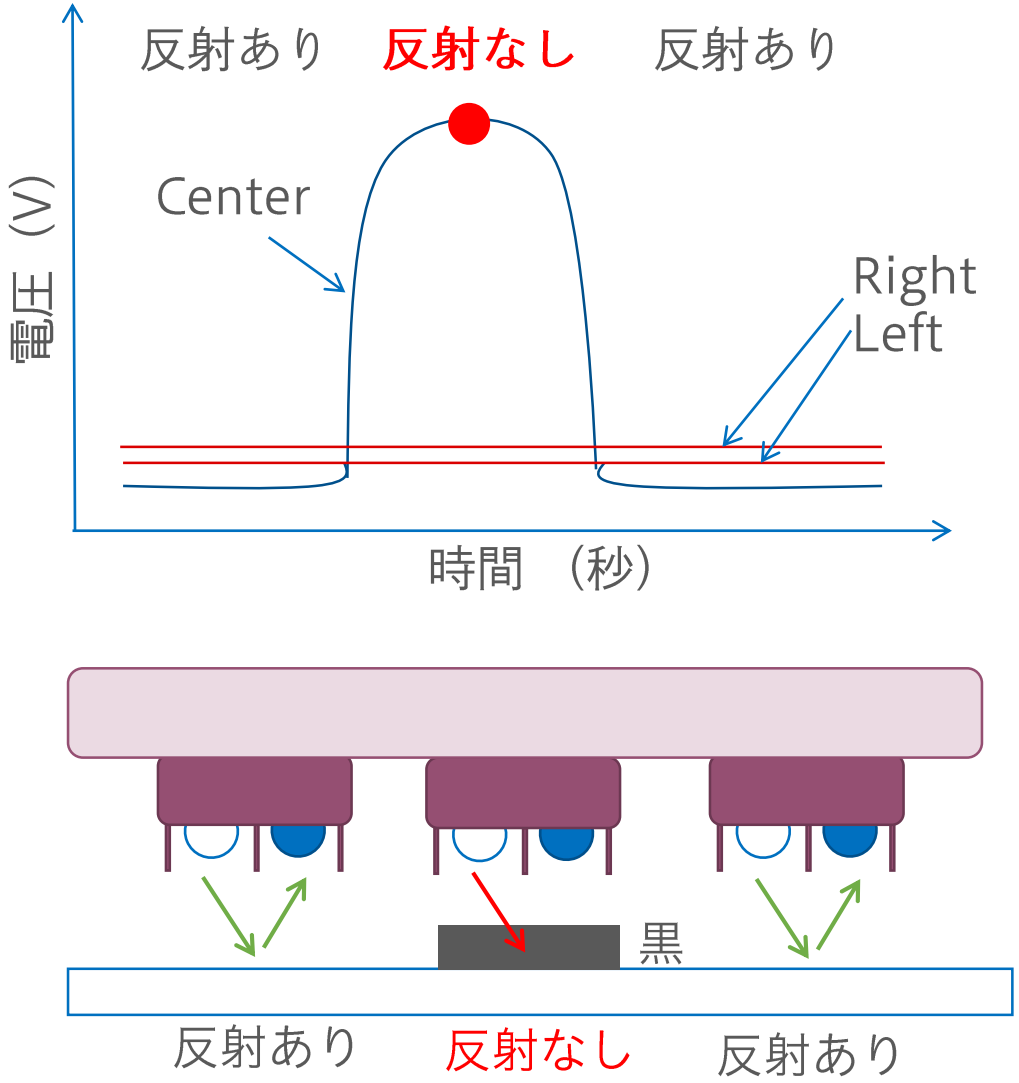
透過型
（フォトインタラプタ）

Copyright 2018-2019, Akinori Tsuji, All rights reserved.

フォトリフレクタの応用

- ロボットのライントレース
  - 線(ライン)の有無の検出

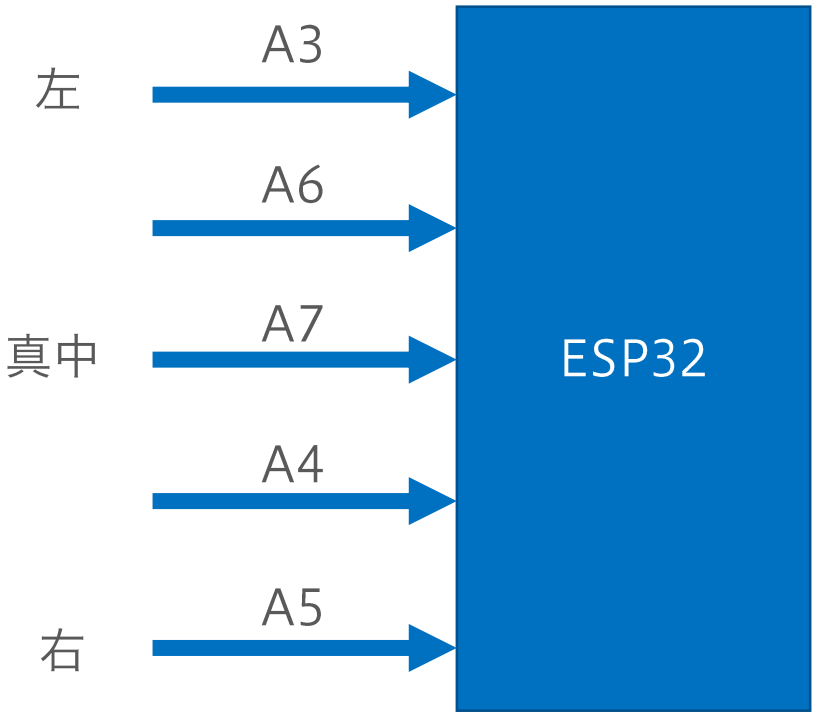


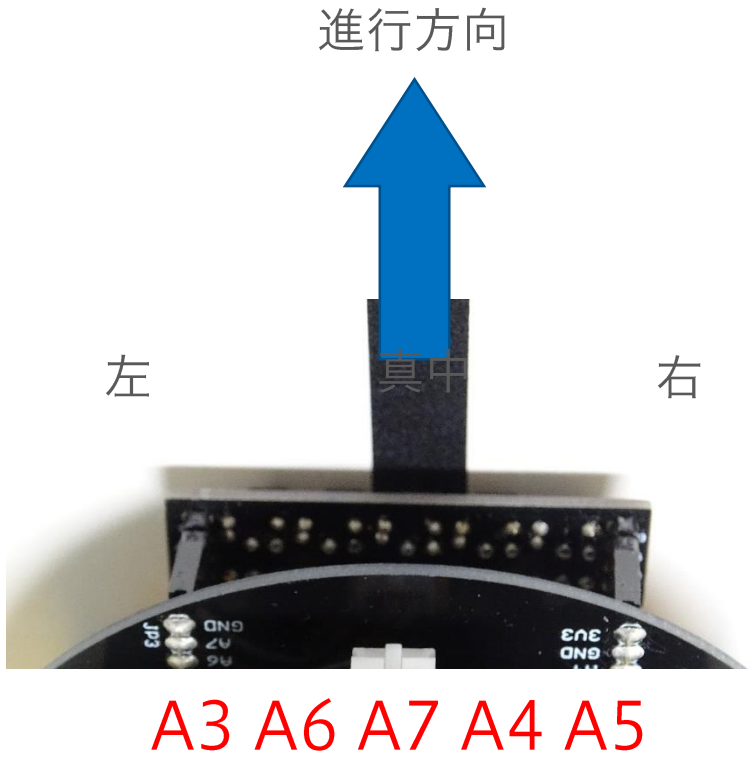


Copyright 2018-2019, Akinori Tsuji, All rights reserved.

フォトリフレクタの配線

- フォトリフレクタ5個
- アナログ端子に接続



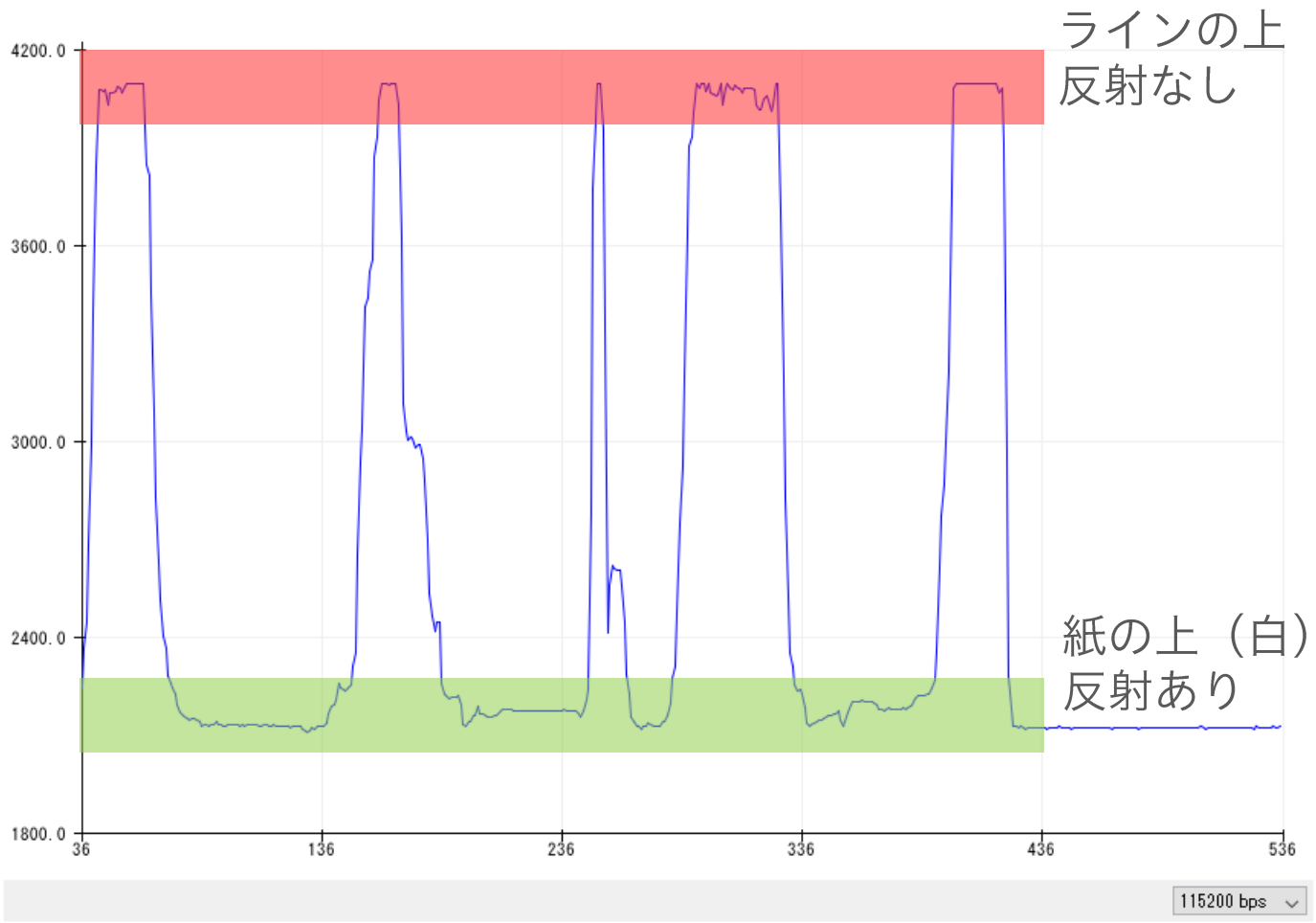


Copyright 2018-2019, Akinori Tsuji, All rights reserved.

Ex0601：フォトリフレクタの読み込み(真中)

- analogRead関数
  - アナログ値：0～4095
- シリアルプロッタで確認

```
int pr = 0;
void setup() {
  Serial.begin(115200);
  delay(100);
}
void loop() {
  get_pr();
  Serial.println(pr);
  delay(100);
}
void get_pr() {
  pr = analogRead(A7);
}
```



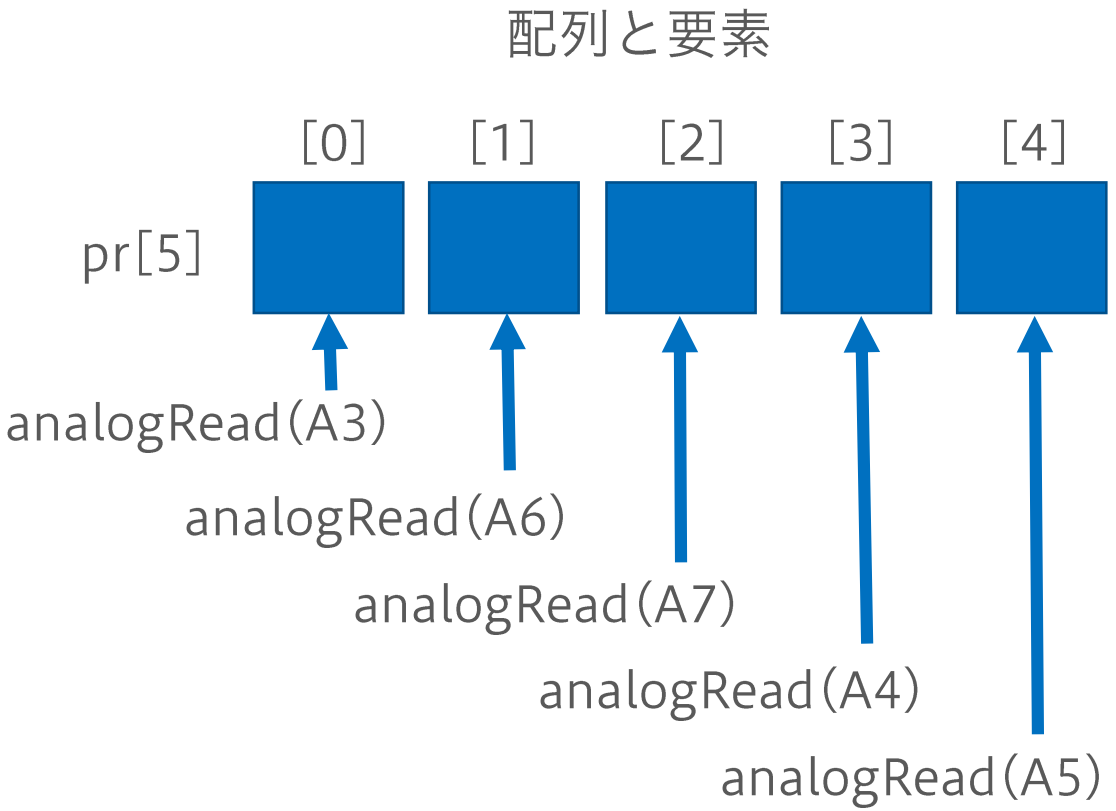
Copyright 2018-2019, Akinori Tsuji, All rights reserved.

Ex0602：フォトリフレクタの読み込み(5個)

- フォトリフレクタ5個をまとめて読み込み
  - 配列を使用

```
int pr[5] = {0}; // フォトリフレクタ5個分の配列

void get_pr() {
  pr[0] = analogRead(A3); // left
  pr[1] = analogRead(A6);
  pr[2] = analogRead(A7);
  pr[3] = analogRead(A4);
  pr[4] = analogRead(A5); // right
}
```



Copyright 2018-2019, Akinori Tsuji, All rights reserved.

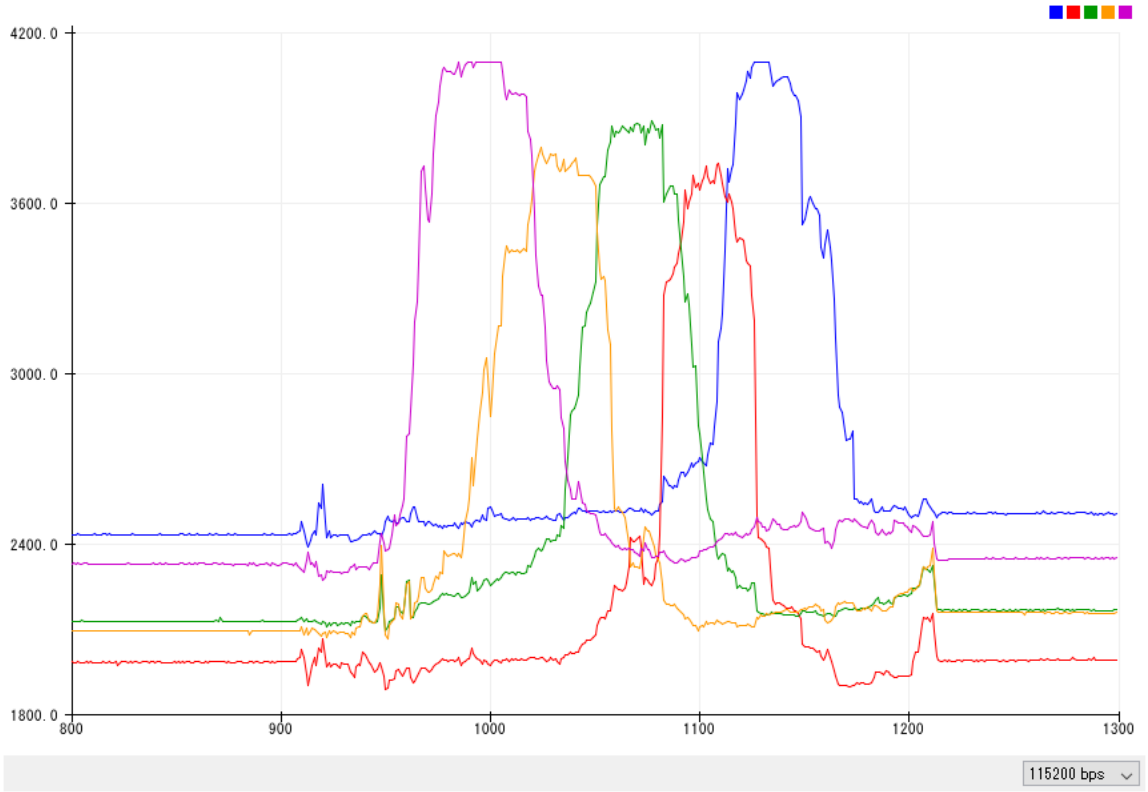
Ex0603：フォトリフレクタの読み込み(Ex0602改良)

- フォトリフレクタ5個をまとめて読み込み
  - 配列を使用
  - マクロ定義(#define)
  - forループを使用

```
#define NUM_PR  5 // センサの数
const int pr_pins[NUM_PR] = {A3, A6, A7, A4, A5};
int pr[NUM_PR] = {0};

void get_pr() {
  for (int i = 0; i < NUM_PR; i++) {
    pr[i] = analogRead(pr_pins[i]);
  }
}
```

Ex0602, Ex0603の結果

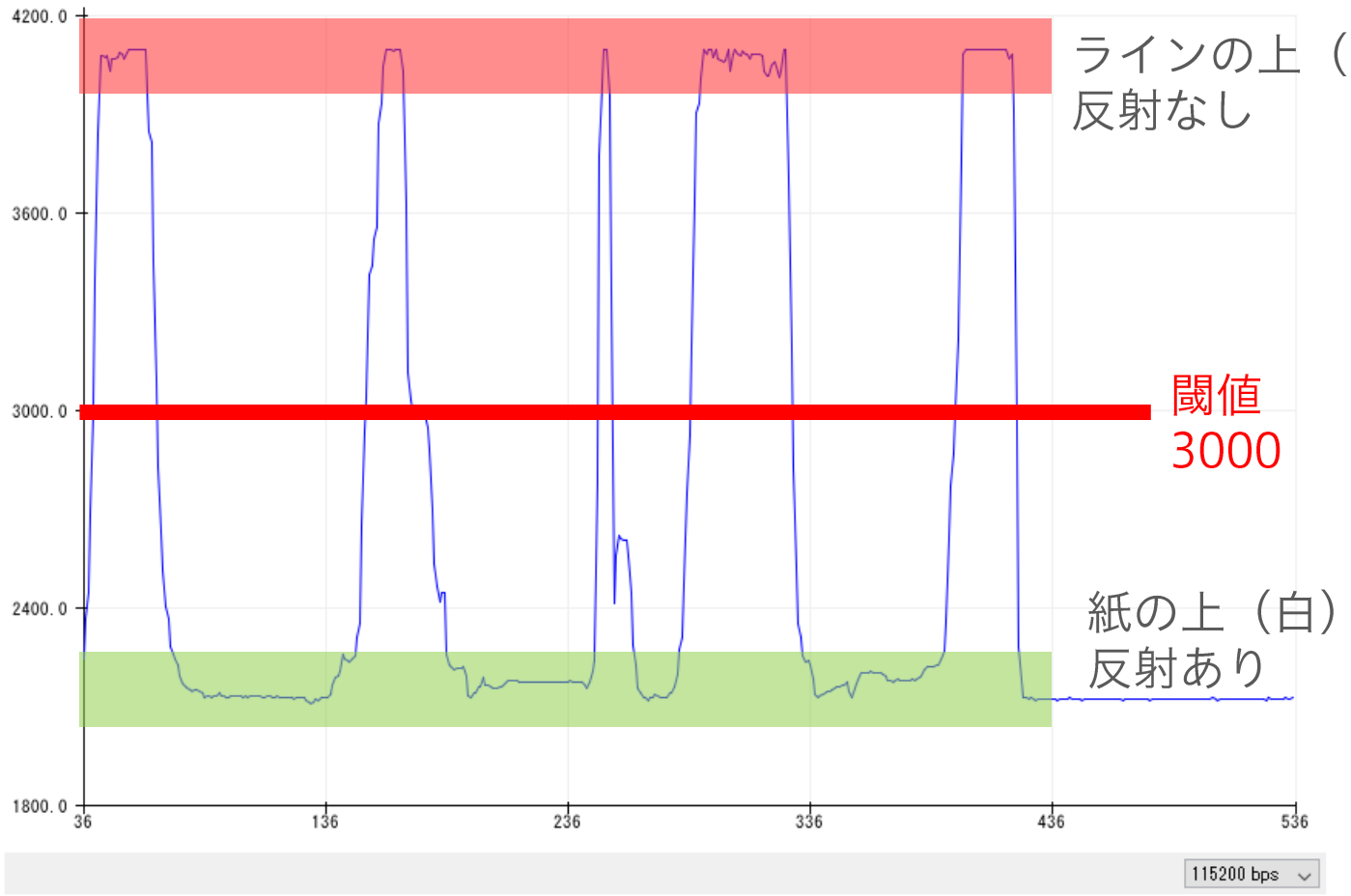


Copyright 2018-2019, Akinori Tsuji, All rights reserved.

ラインの検出

- センサがライン上にあるか？
  - ある値(閾値)より大きい → 1
  - それ以下 → 0
- 0か1の状態にする(2値化)

```
if (pr > 3000) {
    pr_bin = 1;
} else {
    pr_bin = 0;
}
```



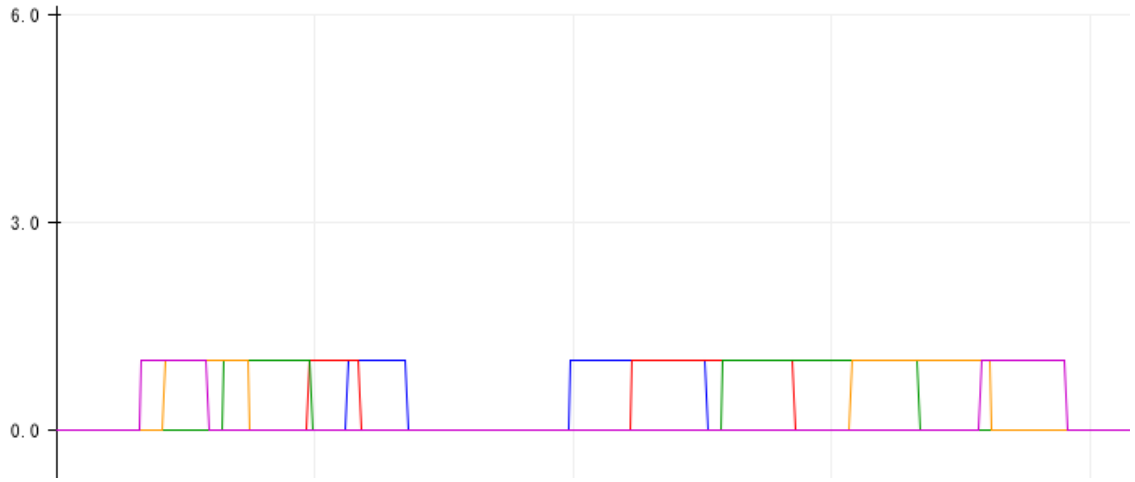
Copyright 2018-2019, Akinori Tsuji, All rights reserved.

Ex0604：ラインの検出

- Ex0603を実行して，閾値(PR_TH)を確認
  - ラインの有無：pr_bin[]

```
#define NUM_PR  5 // センサの数
const int pr_pins[NUM_PR] = {A3, A6, A7, A4, A5};
int pr[NUM_PR] = {0};
int pr_bin[NUM_PR] = {0}; // ラインの有無
const int PR_TH = 3000; // 閾値

void get_pr() {
  for (int i = 0; i < NUM_PR; i++) {
    pr[i] = analogRead(pr_pins[i]);
  }
  for (int i = 0; i < NUM_PR; i++) {
    if (pr[i] > PR_TH) pr_bin[i] = 1;
    else pr_bin[i] = 0;
  }
}
```

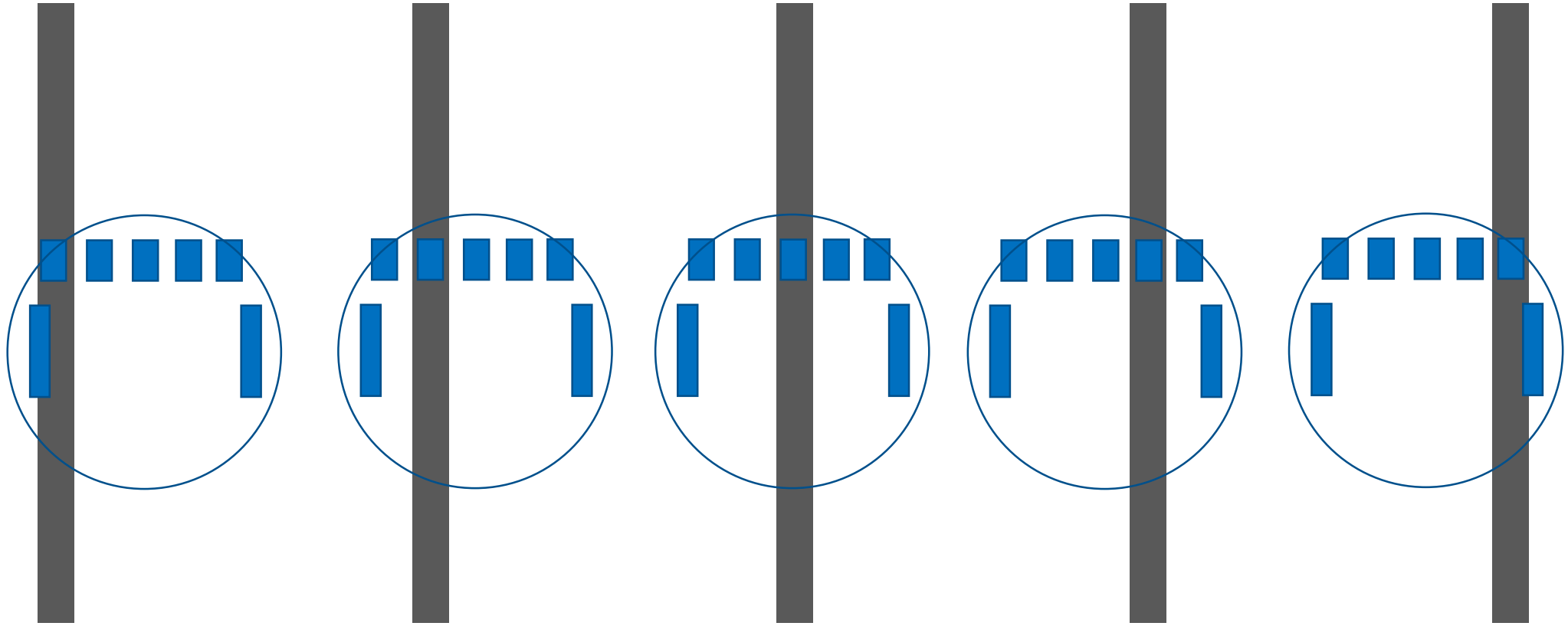


実行結果：ラインを検出すると1
ライン外のとき0

Copyright 2018-2019, Akinori Tsuji, All rights reserved.

ロボットのラインに対する位置

- ロボットがライン(直線)上の何処にいるか？



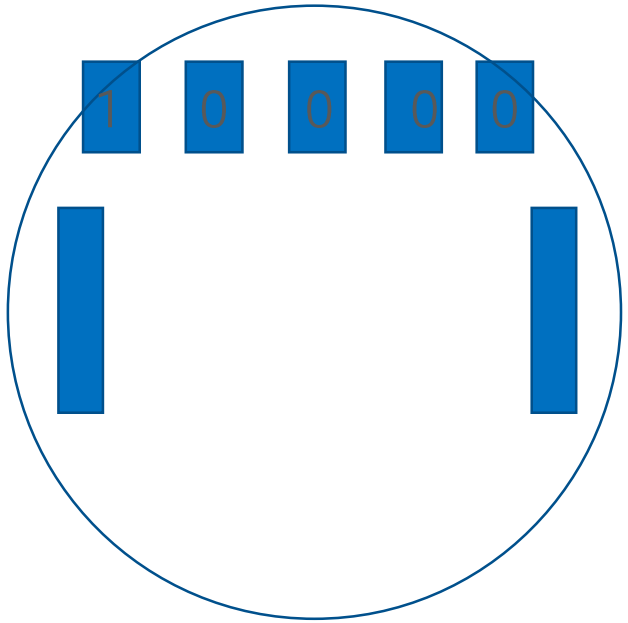
pr_bin[5]　1 0 0 0 0　0 1 0 0 0　0 0 1 0 0　0 0 0 1 0　0 0 0 0 1

Copyright 2018-2019, Akinori Tsuji, All rights reserved.

ロボットのラインに対する位置

- ロボットがライン(直線)上の何処にいるかを数値で知る
  - センサの反応した位置毎に重み付け

pr_bin [0] [1] [2] [3] [4]
重み： x100 x200 x300 x400 x500



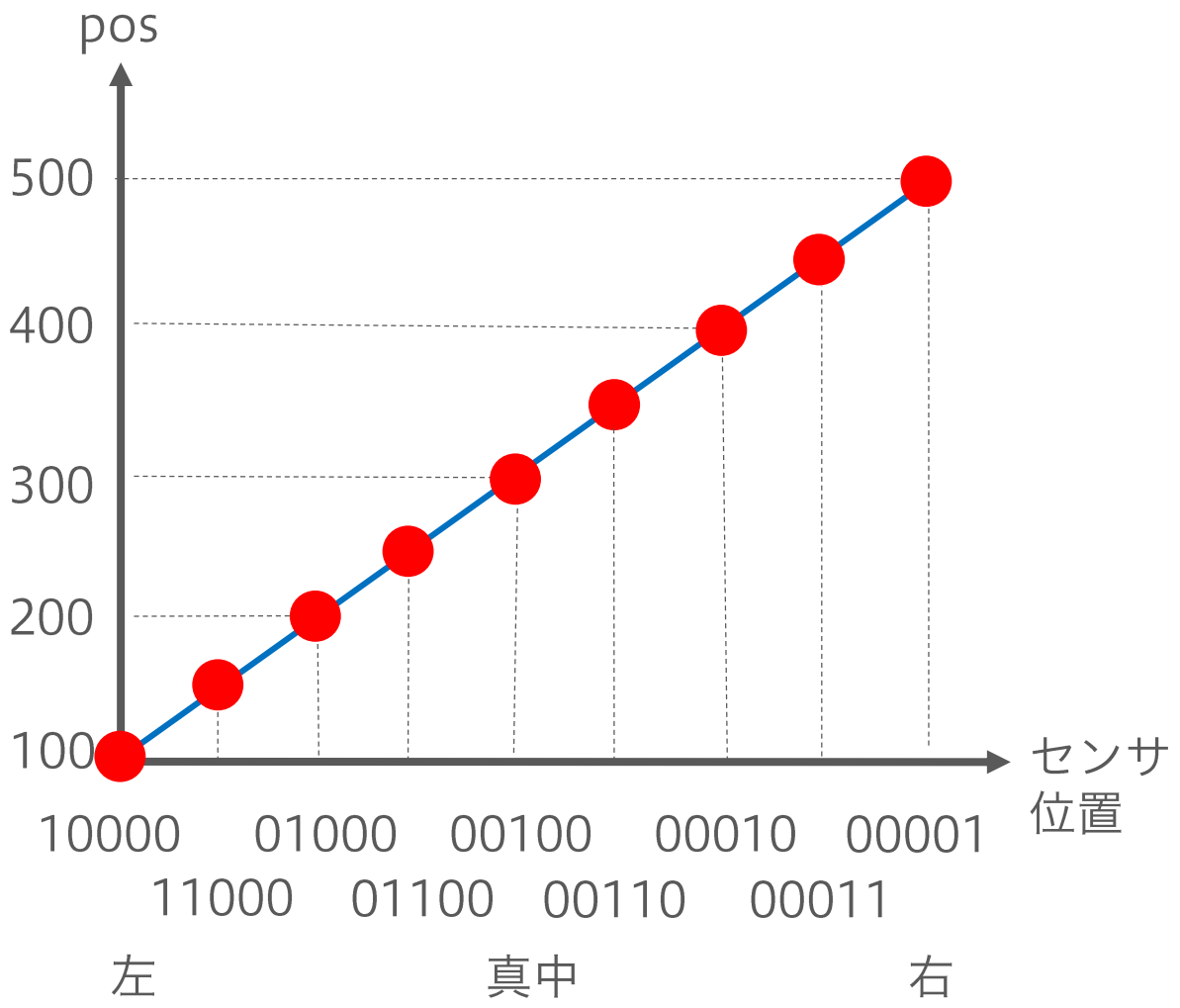
	ラインの有無(0,1)					反応数	合計	ラインの位置
pr_bin	[0]	[1]	[2]	[3]	[4]	n	sum	pos (sum/n)
	0	0	0	0	0	0	0	0 ------
	1	0	0	0	0	1	100	100 (100/1)
	1	1	0	0	0	2	100+200	150 (300/2)
	0	1	0	0	0	1	200	200 (200/1)
	0	1	1	0	0	2	200+300	250 (500/2)
	0	0	1	0	0	1	300	300 (300/1)
	0	0	1	1	0	2	300+400	350 (700/2)
	0	0	0	1	0	1	400	400 (400/1)
	0	0	0	1	1	2	400+500	450 (900/2)
	0	0	0	0	1	1	500	500 (500/1)

Copyright 2018-2019, Akinori Tsuji, All rights reserved.

Ex0605：ラインの位置

- ラインの位置に応じた値を出力
  - ラインあり：100～500
  - ラインなし：0

```
int line_pos() {
  int pos;
  int sum = 0, n = 0;
  for (int i = 0; i < 5; i++) {
    sum += pr_bin[i] * ((i+1) * 100);
    if (pr_bin[i] == 1) n++; // ラインを検出した数
  }
  if (n > 0) {
    pos = sum / n;  // ライン位置
  } else {
    pos = 0; // ライン未検出
  }
  return pos;
}
```



Copyright 2018-2019, Akinori Tsuji, All rights reserved.

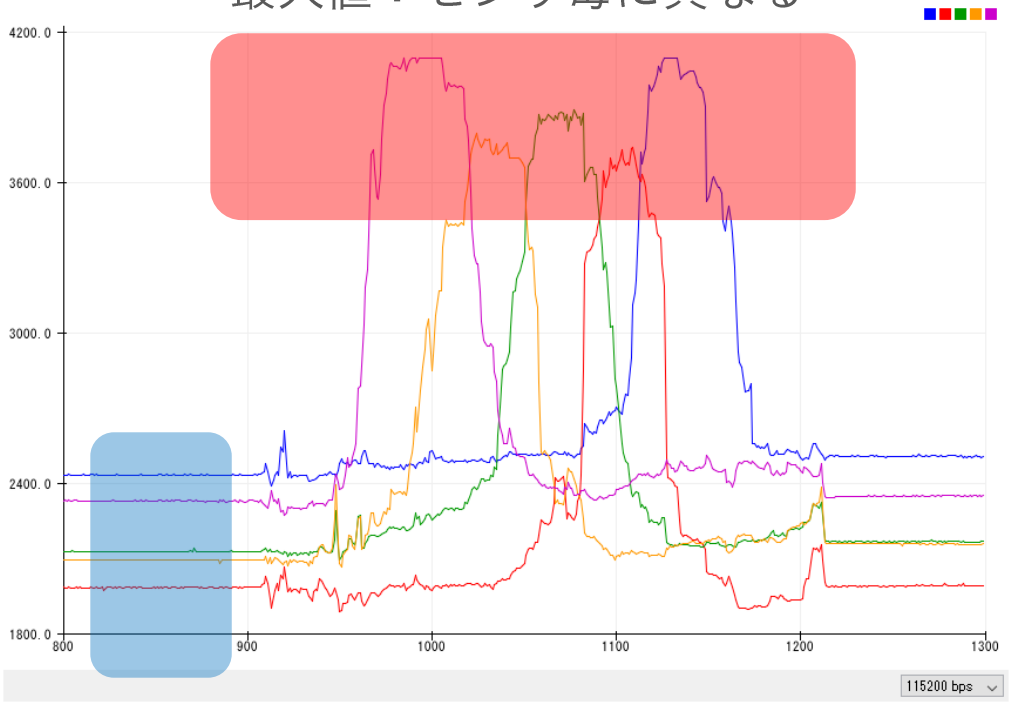
Ex0606： ラインの検出(閾値の自動計算)

- ラインの濃さや周囲の環境によって環境が異なる。
- ライン検出時に，最大値と最小値を探す
- (現在の値 – 最小値) / (最大値 – 最小値)
- 正規化：0～1.0
- 閾値：0.6

```
int pr_min[NUM_PR] = {4095, 4095, 4095,
                      4095, 4095}; // 最小値
int pr_max[NUM_PR] = {0}; // 最大値

for (int i = 0; i < NUM_PR; i++) {
   if (pr[i] < pr_min[i]) pr_min[i] = pr[i];
   if (pr[i] > pr_max[i]) pr_max[i] = pr[i];
 }
```

最大値：センサ毎に異なる



最小値：センサ毎に異なる

Copyright 2018-2019, Akinori Tsuji, All rights reserved.

ロボットをスムーズに動かすために

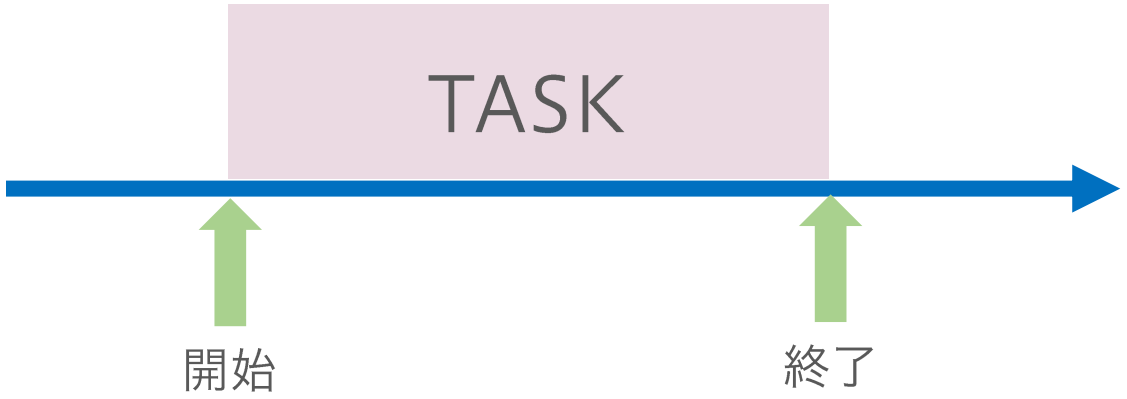
- センサ読み込み，データ処理，モーター駆動などには時間がかかる。
- それぞれ，どのぐらい時間を要するかmillis, micros関数を使って計測してみよう。

開始 → `unsigned long st = millis();`

```
・・・
Do something task
・・・
```

終了 → `unsigned long en = millis();`

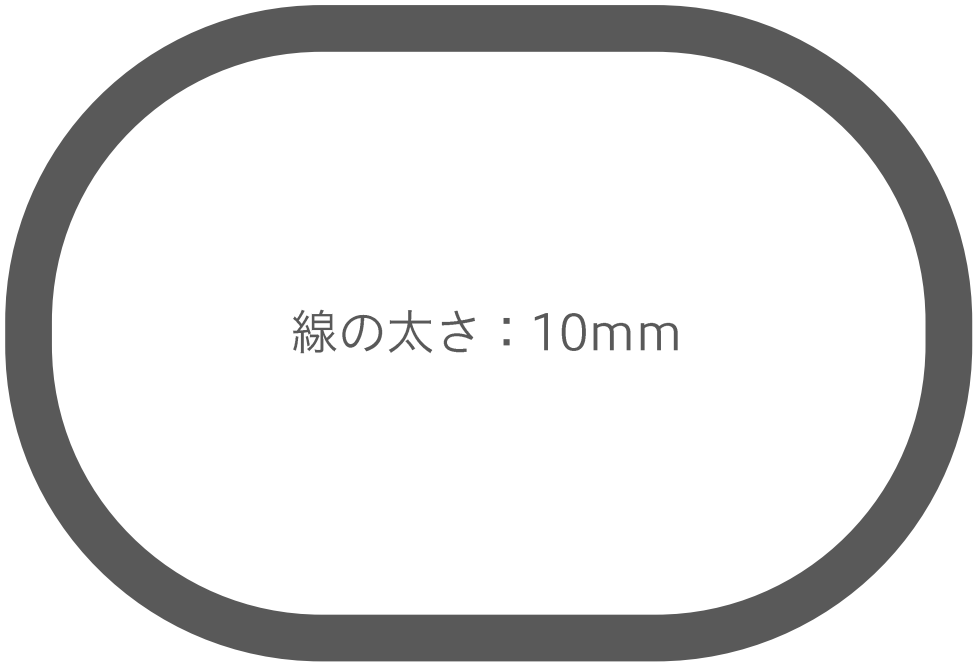
`Serial.println(en – st);`



Copyright 2018-2019, Akinori Tsuji, All rights reserved.

ロボットのライントレース

- ロボットをライン(黒)に沿って走らせる
  - センサでラインの位置を確認して
  - 左右のモーターの回転速度を調整



Copyright 2018-2019, Akinori Tsuji, All rights reserved.