

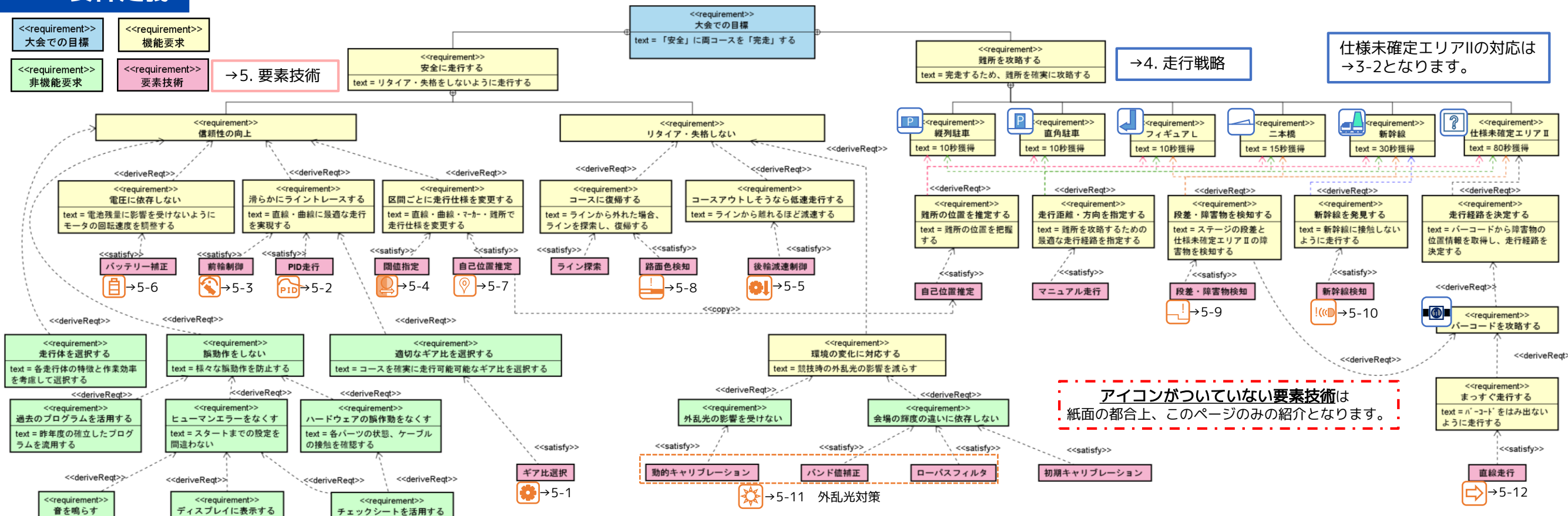
# 1. 要件分析



## ◆ 目標：『安全』に両コースを『完走』する！

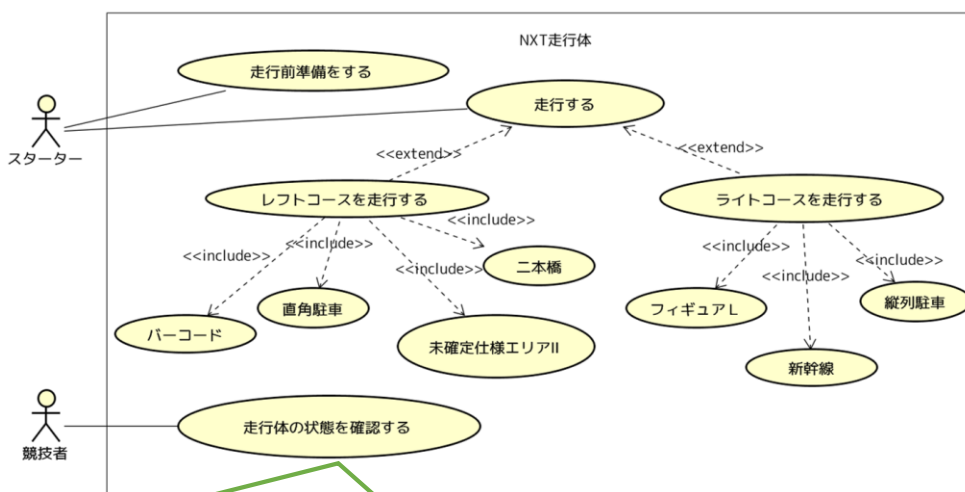
### 1-1. 要件定義

SysML要求図を用いて、目標を分析し必要な機能要件・非機能要件・要素技術の洗い出しを行う。この図では線の重なりを判別しやすくするため、一部色分けしている。



### 1-2. ユースケース分析

システムに必要なユースケースを抽出し、その全体像を表現した。紙面の都合上、ユースケース記述は省略する。



走行体の状態を確認する内容と方法は、1-3.非機能要件を参照

### 1-3. 非機能要件

非機能要件を整理し、実現方法を検討する。

- 《ヒューマンエラーをなくす》：人為的ミスをなくす。
  - ・ **チェックシート**を用いて、スタート前の準備を確認する。
  - ・ スターターはキャリブレーション時には**NXTの画面に表示**される重要なパラメータを確認する。もし、白と黒の基準値が逆の場合は**警告音**を鳴らして、再設定を行う。
  - ・ メンバー全員で事前に**指さし確認**をする。
- 《ハードウェアの誤作動をなくす》：各パーツの不具合による誤作動をなくす。
  - ・ **チェックシート**を用いて、各パーツに不備はないか確認する。
  - ・ メンバー全員でケーブルの接触状態や前輪軸・後輪の歯車などを**指さし確認**する。
- 《過去のプログラムを流用する》：確立された過去のプログラムを再利用する。
  - ・ 昨年度のプログラムを流用し、走行体の動作や要素技術の**信頼性を保証**する。
- 《走行体を選択する》：両走行体を分析して制御しやすい走行体を選択する。
  - ・ EV3TrikeVは**ステアリングの形状**により、NXTrikeより滑らかにライントレースする事が難しい。(→1-4. 前輪の形状比較)
  - ・ 段差昇降の際、**EV3TrikeVの車体下フレームが段差に引っ掛かる**ので、車輪を段差にかけることができない。
  - ・ 以上により、今大会は**NXTrike**で走行する。
- 《適切なギア比を選択する》：安全に完走するため、最適に走行可能なギア比を決定する。 →5-1に記述する。
- 《会場の輝度の違いに依存しない》：練習場の輝度に依存しない。 →5-11に記述する。
- 《外乱光の影響を受けない》：外乱光によるライントレース失敗を防ぐ。 →5-11に記述する。

### 1-4. 前輪の形状比較

NXTrike



EV3TrikeV



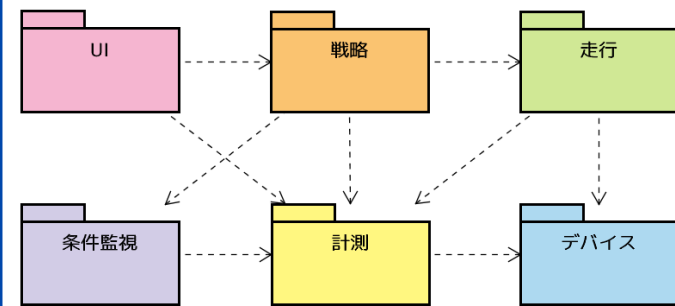
NXTrikeの前輪はEV3TrikeVの前輪と異なり、丸みを帯びている。直線を走行するのはEV3TrikeVの方が安定するが、曲線をライントレースする際は、前輪の角で旋回するので、NXTrikeと比較して走行が不安定になる。

# 2. 構造設計



## 2-1-1. パッケージ図

走行システムの構成をパッケージ図を用いて表現する。  
今回は機能別に6種類に分割した。



パッケージ名	内容
UI	ユーザによる操作を受け付ける
戦略	区間ごとに各パラメータを設定したシナリオ情報を管理する
走行	戦略で管理されているシナリオを実行する
条件監視	シナリオの終了条件を設定・判定する
計測	デバイスより値を取得し、走行体とバーコードの状態を計測・保持する
デバイス	各デバイスの値を取得・出力を行う

## 2-1-2. シナリオ定義

戦略パッケージ内のシナリオ情報を定義する。  
また、どのような条件でシナリオ(区間)を切り替えるのか記述する。

シナリオは直線・曲線・各難所の攻略情報の集合体である。  
戦略パッケージ内には両コースのシナリオが存在する。  
シナリオで設定できるのは、

1. 走行方法 (PIDトレース走行 or 直進走行)
  2. 走行エッジ (ライン右側 or ライン左側)
  3. 走行速度 (前輪旋回速度・後輪速度)
  4. 前輪角度の操作
  5. 走行体の向き操作 である。
- 上記を組み合わせることで区間を走行・攻略する。

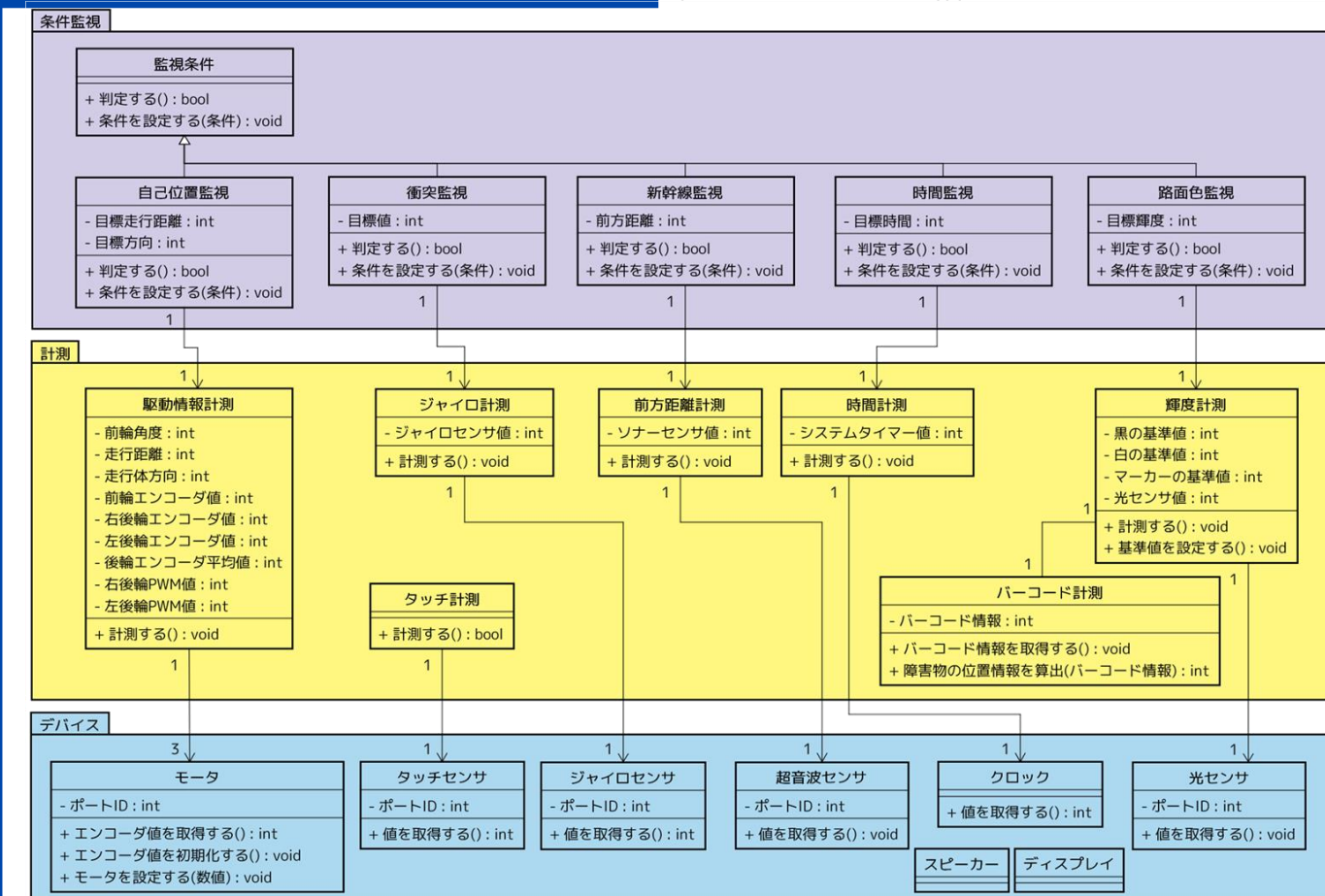
シナリオを切り替える主な条件として、

1. 走行距離 ... 目標距離を走行したか?
  2. 衝突 ... 段差・障害物を検知したか?
  3. 路面色 ... 目標の輝度を取得したか?
  4. 時間 ... 指定の時間に到達したか?
- を条件としている。

上記の条件が達成したなら、今のシナリオを終了し、  
次に設定しているシナリオを実行する。

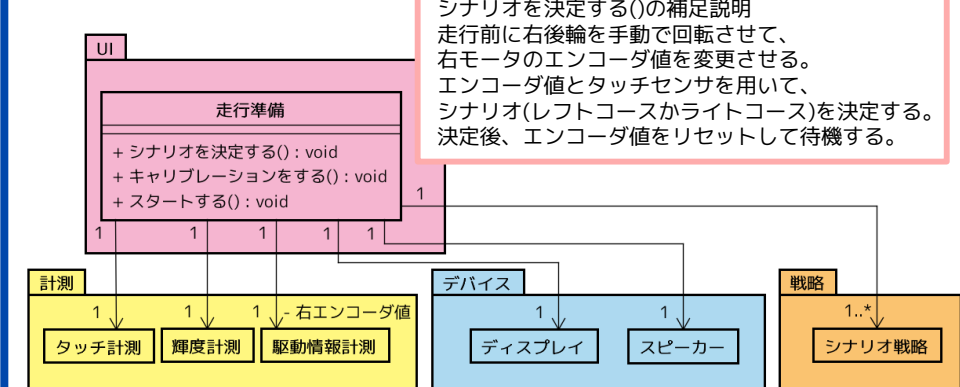
## 2-3. 条件監視・計測・デバイス

条件監視・計測・デバイスのパッケージ図の詳細構造を示す。  
紙面の都合上、一部の操作区画は省略する。



## 2-2. UI

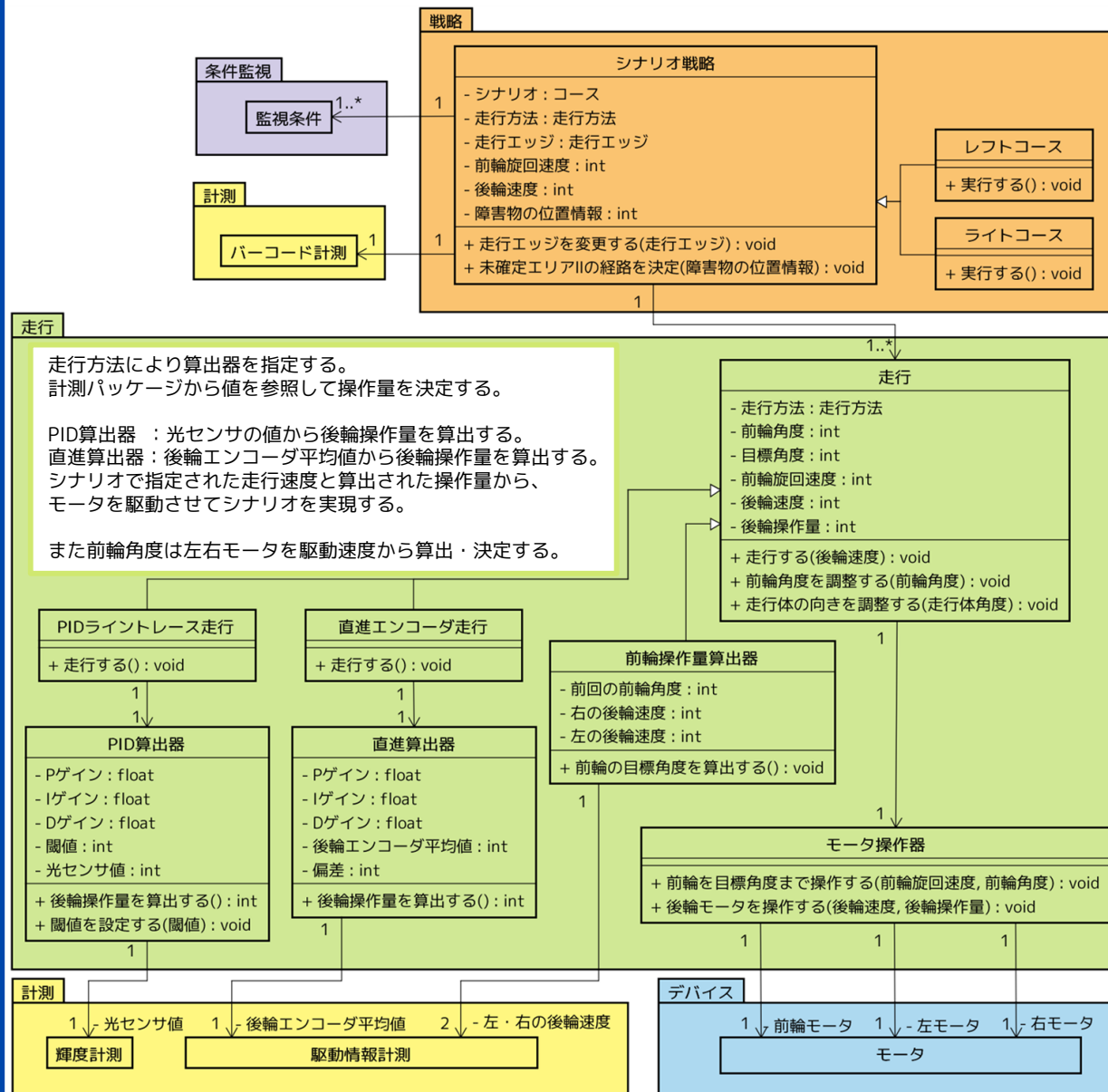
UIのパッケージ図の詳細構造を示す。



シナリオを決定する()の補足説明  
走行前に右後輪を手動で回転させて、  
右モータのエンコーダ値を変更させる。  
エンコーダ値とタッチセンサを用いて、  
シナリオ(レフトコースかライトコース)を決定する。  
決定後、エンコーダ値をリセットして待機する。

## 2-4. 戦略・走行

戦略・走行のパッケージ図の詳細構造を示す。



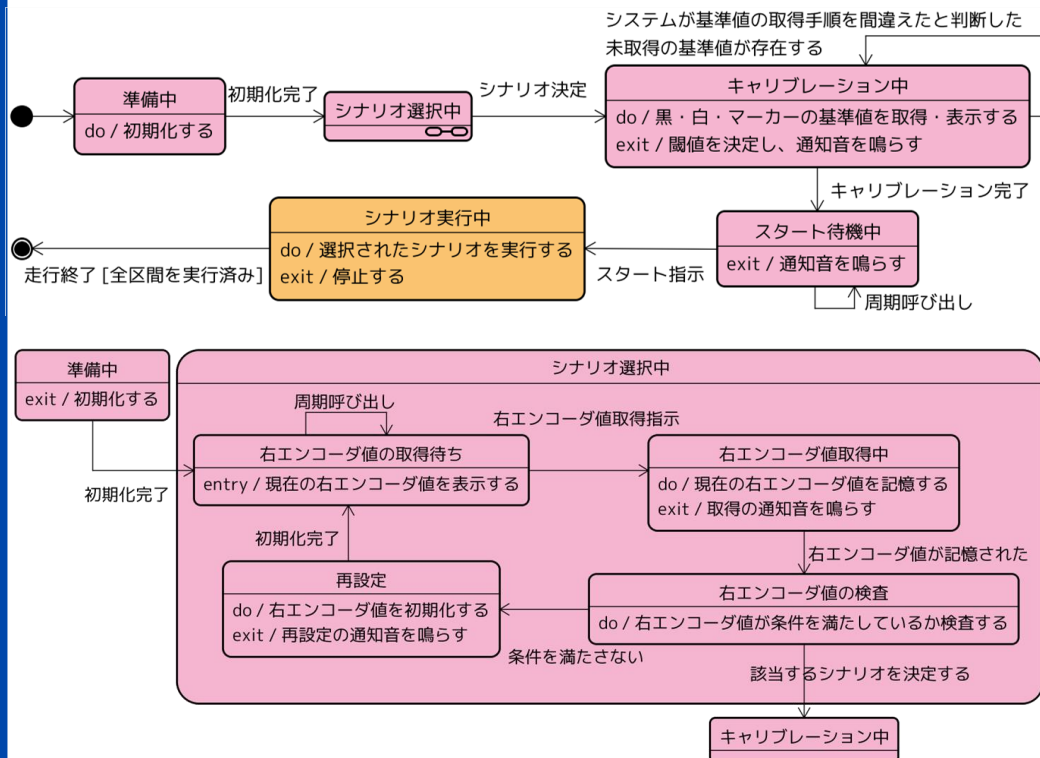




### 3-1. 状態遷移図

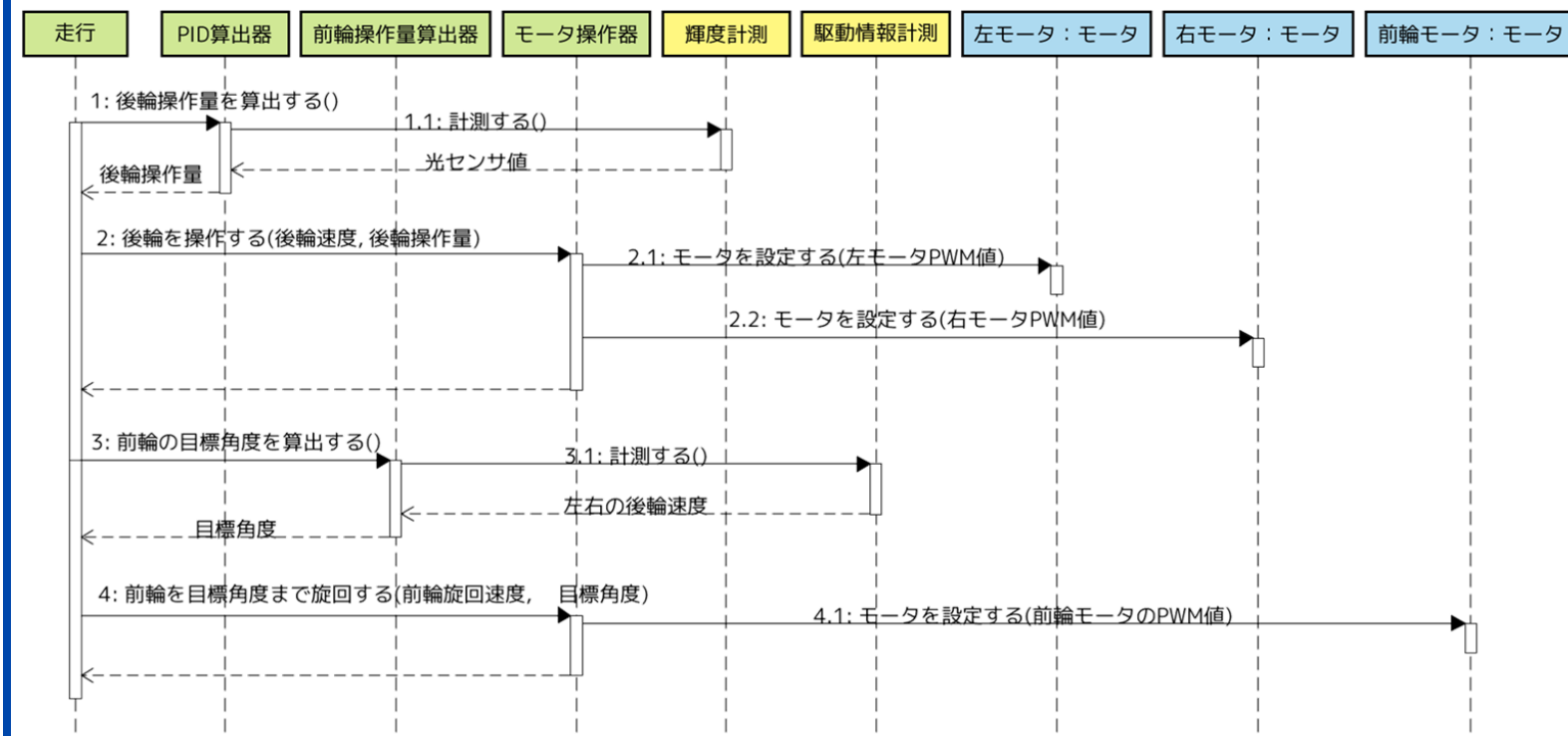
走行前にモータのエンコード値を用いてシナリオを決定する。  
その後、キャリブレーションを行い、走行前の準備を行う。

ここではシナリオ選択・キャリブレーション終了時にパラメータ値を検査して、競技者が間違えて設定外の値を入力した場合、システムは最初からやり直すように通知する。



### 3-2. PID走行

基本走行となるPIDライントレースの走行シーケンスを例示する。



### 3-3. 仕様未確定エリアIIの対応



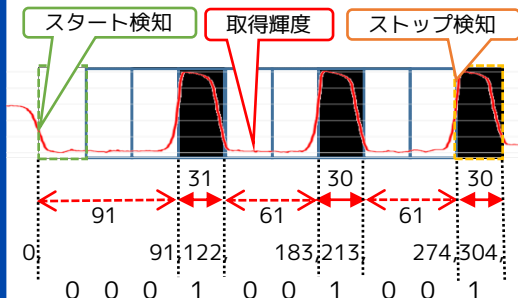
：使用要素技術

**【設計課題】** 仕様未確定エリアIIを攻略するために、①. バーコードからの位置情報の読み取り方法、②. 最適な走行経路の算出方法、③. 算出した走行経路の走行方法を記述する。

**【設計①】** バーコードからの位置情報の読み取り方法 (バーコードの走行戦略→4-5)

スタートビットの白、ストップビットの黒を読み込む。  
 ストップの判定条件：1~9ビットを走行後(約270mm以上)に判定する。  
 (…、9,10bitが連続して黒だった場合、直前のビットを10bitだと誤検知するのを防ぐため。)  
 スタートとストップ基準に走行距離を用いて全体の長さを調整する。

1. バーコード上の路面色を判定する。閾値以上なら黒、閾値未満は白と判定する。
2. 路面色を判定した後、**同一色が何mm継続して計測されたか**走行距離を記憶する。
3. その走行距離より、白または黒のビット情報を対応するビットに記憶する。
4. バーコード上の総走行距離が終了目標距離になるまで1~3を繰り返す。




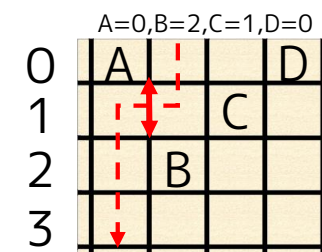
ここがポイント！！

斜めに走行したり取得開始位置が前後した際、走行距離に誤差が生じる事を考慮してNビット分に相当する基準距離を  $N \times 30 + \alpha [\text{mm}]$  以内とする。  
 $\alpha = 5 \text{ mm}$  として、  
 1bit分の距離 … 0 ~ 35 mm  
 2bit分の距離 … 36 ~ 65 mm …  
 10bit分の距離 … 276 ~ 305 mm  
 を基準距離とする。

誤差に強いビット取得が可能になる!

斜め走行 : 1bit分の距離 : 約32[mm]  
理想走行 : 1bit分の距離 : 30[mm]

- 差分が正なら左に方向転換可能、
- 
1. [0 0] [1 0] [0 1] [0 0]  
[0] [2] [1] [0]  
A=0, B=2, C=1, D=0
2. A-B=-2 B-C=1 C-D=1



図：障害物の位置と走行可能な空間

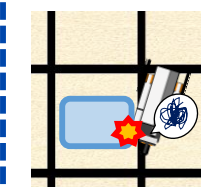
3. この場合、パターン[A-B]の差分が-2なので、  
 入口：「bc間」 出口：「ab間」 方向転換：「右方向」とする

### 【設計③】 算出した走行経路の走行方法

1. 未確定エリア侵入後、ペットボトルに衝突するまで前進。
2. 衝突したら少し後退し、斜め右または斜め左に方向転換する。
3. 段差（出口）を検知するまで**進入した列に赤るように斜めに走行する。**
4. 直進して脱出する。

ここがポイント！！

各列に障害物は一個なので、障害物を回避した後の進入した列には障害物は存在しない。

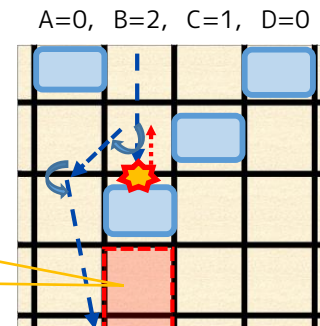


図③-2：衝突の例

前進または方向転換の際、走行距離が変化していない場合、つかえていると判断し、復帰動作を行う。

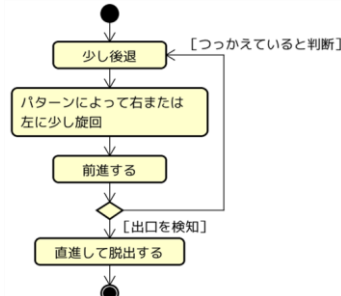
(枠の都合により、アクティビティ図は3.の復帰動作のみ記載)

□ : 障害物の位置



図③-1：走行経路

【課題】 走行体が障害物につっかえる





# 4. 走行戦略



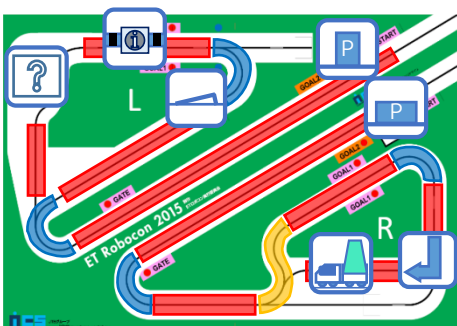
## 4-1. 区間攻略 & 特殊走行

直線・カーブ・S字カーブごとにコースを区間に分ける。  
各区間のライトレースに適した前輪・後輪PWM値を設定する。(→4-1-1を参照)  
難所の位置はアイコンで示す。(各難所の戦略→4-2~4-6)  
そして、各難所の攻略に用いる特殊走行を紹介する。(→4-1-2, 4-1-3を参照)

### 4-1-1. 区間攻略



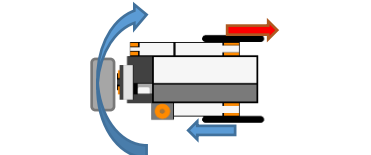
：主な使用要素技術



区間	直線	曲線	S字
前輪回転速度	小	中~大	大
後輪速度	大	中~小	小

### 4-1-2. その場回転

走行体をその場で回転させる。  
前輪角度を車軸に対して90度にし、  
左右後輪のPWM値を符号逆転させる。



### 4-1-3. 段差昇降

前輪を持ち上げ、難所に昇段する。  
後退後、急発進することで前輪が浮く。  
両後輪をステージ入口に引っ掛けると、  
走行体をまっすぐにできる。



## 4-2. 新幹線



：使用要素技術

[課題1] 移動する新幹線を検知しないと、**走行体**が**新幹線と衝突**する。

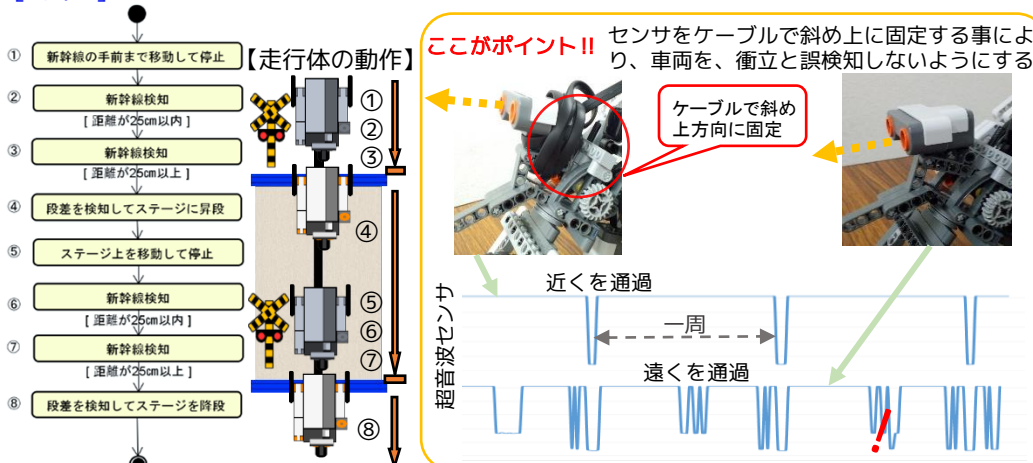
[対策1] **新幹線を検知するまで待機**し、新幹線が通過後に走行を再開する。

[課題2] 新幹線の車両の数が増えると、衝突と衝突の間で「**新幹線が全て通過した**」と誤検知し、  
通過中の車両と衝突する。

[対策2] アクティビティ図の②、③を**複数回実行**し、確実に新幹線が通過した後に走行を再開する。

[課題3] フィギュアLの降段時、**走行体の後輪が滑り走行距離に誤差**が生じて超音波で**新幹線を検知する**  
**ことが可能な地点**まで達しない。

[対策3] 一定時間、超音波センサが新幹線を検知しない場合、5cm**前進して超音波による検知を再開**する。



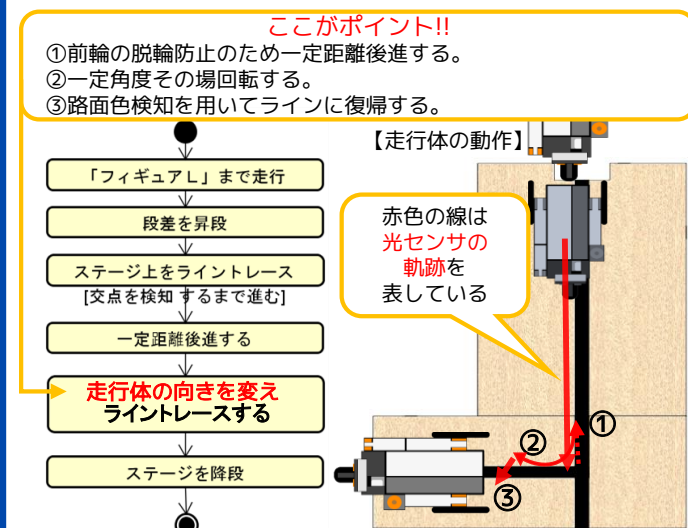
## 4-3. フィギュアL



：使用要素技術

[課題] 走行体が通常のライトレースで直角に曲がる場合、  
**前輪または後輪がステージから脱輪**する可能性がある。

[対策] **路面色検知を多用**し走行体がラインから離れないように、  
旋回・走行することで、前後輪の脱輪を防ぐ。



## 4-4. 二本橋



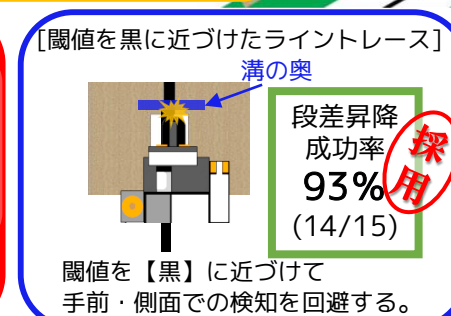
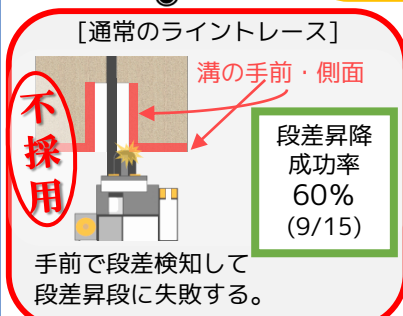
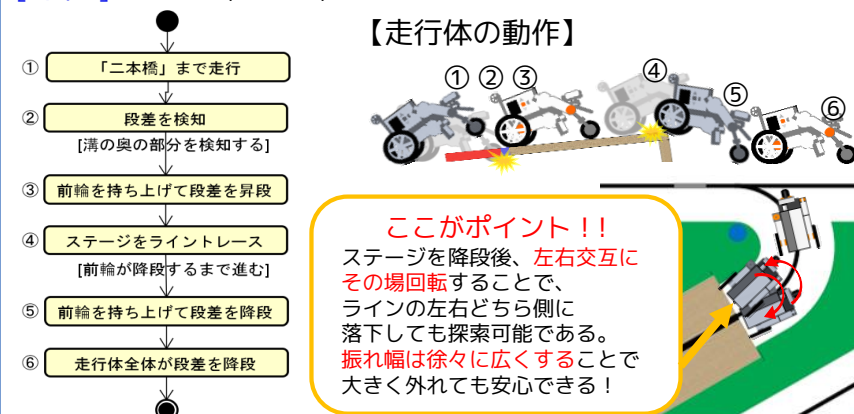
：使用要素技術

[課題1] 前輪が**溝の手前・側面**に触らないようにして、『**溝の奥**』で段差検知する。

[対策1] ライン寄りに走行すると、**溝の奥**を検知して安定した段差昇降が可能になる。

[課題2] 降段後、その先のラインが曲線になっているため、通常のトレースを行うと  
**ラインを見失う可能性**がある。

[対策2] その場回転(→4-1-2)を用いて、**広範囲のライン探索**を行う。



## 4-5. バーコード



：使用要素技術

[課題1] バーコード上では**ライトレースができない**ため、コースアウトしてしまう。

[対策1] 走行体の方向をラインと並行になるように合わせて、**直進走行**でバーコード上を走行する。

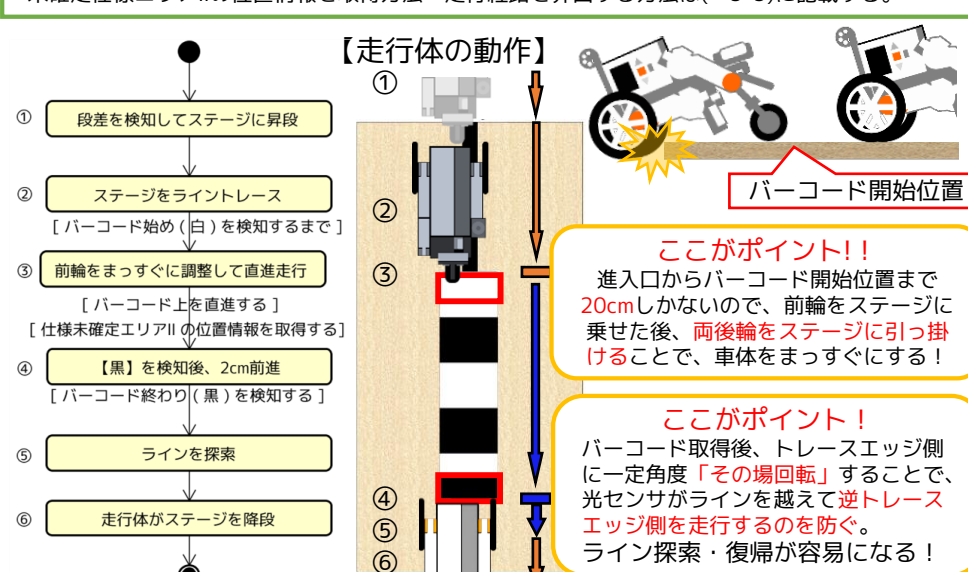
[課題2] バーコードを読み取った後、**ラインを見失う可能性**がある。

[対策2] その場回転を用いて、ステージ横から降りない程度の**狭い範囲でライン探索**を行う。

[課題3] バーコードの横へ外れる等の原因で、バーコード情報を**正しく読み取らない可能性**がある。

[対策3] ストップビット(10ビット目)が白を検知したとき、または算出した障害物の位置情報が、  
「走行が保証されない」と考えられるパターン(障害物が横一列に並ぶパターン)の場合は、  
**スタートビット(1ビット目)まで走行体を後退して再読み込み**を行う。

未確定仕様エリアIIの位置情報を取得方法・走行経路を算出する方法は(→3-3)に記載する。



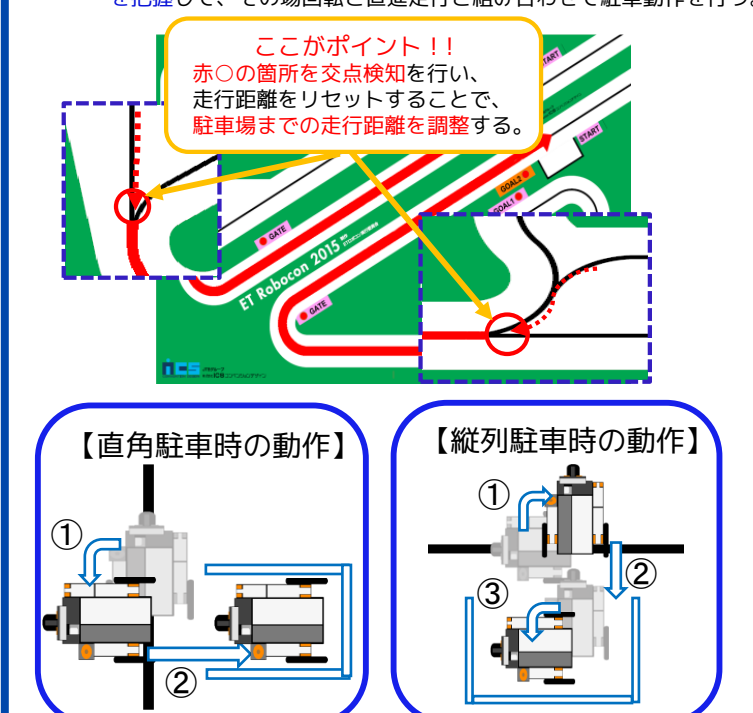
## 4-6. 縦列・直角駐車



：使用要素技術

[課題] 駐車上の**位置を把握することが困難**である。昨年度のコースとは  
異なり、駐車場付近には**目印となるマーカーが存在しない**。

[対策] 路面色検知による**交点検知**により**現在の位置から駐車場まで距離**  
**を把握**して、その場回転と直進走行と組み合わせて駐車動作を行う。



# 5. 要素技術

## 5-1. ギア比選択



難所、二本橋の段差昇段時に着目。走行体が下の図のような状態で段差を昇段できるかを調べた。



図：走行体の状態

ギア比 **1:1** の走行体で出場する！

ギア比	段差昇段成功率
1 : 1	◎
1 : 3	△
1 : 5	×

## 5-2. PID走行



【目的】滑らかなライトレースを実現する。

【方法】閾値と光センサの値から偏差を求め、PID制御による後輪モータの旋回量を算出する。

偏差  $X = \text{閾値} - \text{輝度}$

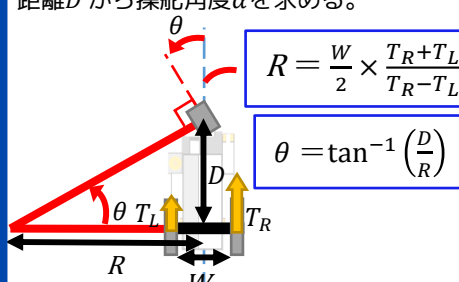
$\Delta X = \text{今回の輝度} - \text{前回の輝度}$

旋回量  $= X \times K_p + \int X \times K_i + \Delta X \times K_d$

## 5-3. 前輪制御



【目的】両後輪モータの出力  $T_L, T_R$  から幾何学的に曲率半径  $R$  を求め、曲率半径  $R$  と後輪の中心部～前輪までの距離  $D$  から操舵角度  $\alpha$  を求める。



$$R = \frac{W}{2} \times \frac{T_R + T_L}{T_R - T_L}$$

$$\theta = \tan^{-1} \left( \frac{D}{R} \right)$$

## 5-5. 後輪減速制御



【目的】ライトレース時に急なカーブ等では走行体の後輪速度によって、前輪操作の制御が追い付かずに曲がり切れない場合がある。

【方法】以下の式で求めた減速量を後輪出力に乗算することで、閾値から離れるほど減速するようにした。

減速量  $= 1 - |(X + \Delta X) \times 0.01|$   
 $X, \Delta X$  の定義は → 5-2

後輪PWM値70以上でRコースのS字カーブ(→4-1)を走行したときの成功率：		
制御なし	制御あり	
53% (8/15)	53%	86% (13/15)

## 5-6. バッテリー補正



【課題】回転速度がバッテリー残量に依存しているため、安定した走行や段差昇降ができない。

【対策】『バッテリーの基準値』を設定し、現在のバッテリー残量との差分を偏差としてP制御して『補正值』を求める。『補正值』をモータのPWM値に加算することで、『バッテリーの基準値』と同じモータの回転速度を再現できる。

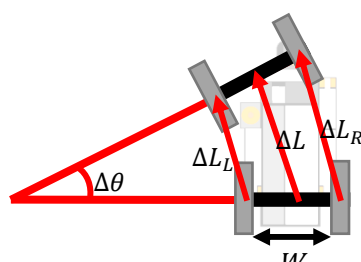
二本橋の段差昇降が成功した回数：(成功回数/試行回数)		
電圧[V]	補正なし	補正あり
9.0	10回/10回	10回/10回
8.5	9回/10回	10回/10回
8.0	8回/10回	10回/10回
7.5	6回/10回	9回/10回

電圧が低下しても、安定して段差昇降ができた。

## 5-7. 自己位置推定



【目的】走行体の方向と移動距離を求めて幅広い戦略を実現する。  
【方法】以下の計算式より走行体の方向  $\theta$ 、移動距離  $L$  を算出する。



$$\Delta \theta = (\Delta L_R - \Delta L_L) / W$$
$$\theta_{i+1} = \theta_i + \Delta \theta$$

$$\Delta L = \pi \times \text{後輪の直径} \times (\text{後輪エンコーダの平均値} / 360)$$

## 5-8. 路面色検知



【目的】路面色(黒・白・灰色マーカー)を検知する。

【方法】輝度補正值が指定した色の基準帯内の値を取得したとき、指定した色を検知したと判断する。(輝度補正值の算出方法 → 5-11【対策③】)

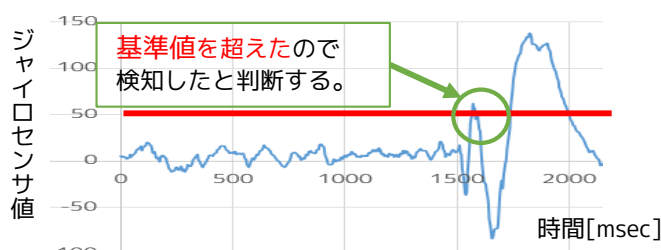


## 5-9. 段差・障害物検知



【目的】段差・障害物を検知する。

【方法】ジャイロセンサの値が基準値を超えたとき、前輪が段差に衝突したと判断することで段差を検知することができる。

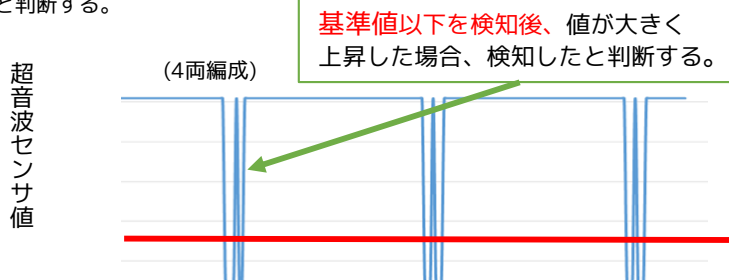


## 5-10. 新幹線検知



【目的】障害物(新幹線)を検知する。

【方法】超音波センサの値が基準値以下を一定時間取得したとき、障害物を検知したと判断する。



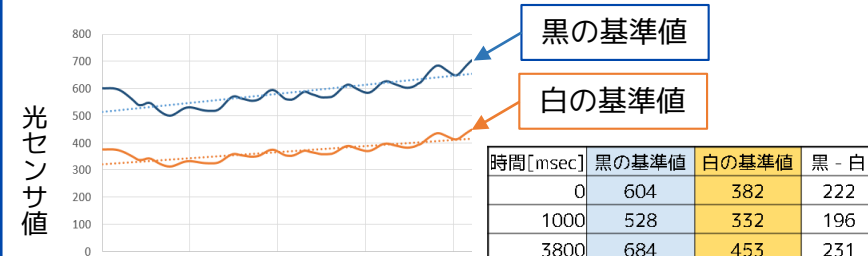
## 5-11. 外乱光対策



【課題】環境光などの原因で外乱が発生し、ライトレースの精度が低下する。練習場と本番会場では輝度が異なるため、PIDゲインも異なる。

【対策①】動的キャリブレーション：

一定周期ごとに輝度の平均値を算出し、その変化量から新たな黒と白の基準値を設定する。



図：黒と白の基準値 時間[msec]

【対策②】ローパスフィルタ(LPF)：

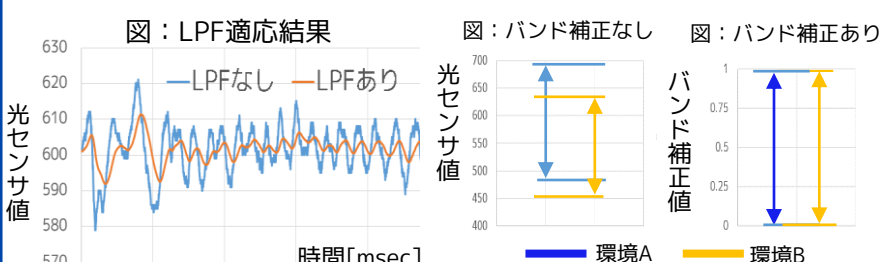
蛍光灯や走行体が走行中に光センサが上下に振動して生じるノイズを除去し、輝度パラメータを安定させる。

$$\text{輝度(LPF)} = \text{今回の輝度} \times (1 - \alpha) + \text{前回の輝度} \times \alpha$$

【対策③】バンド幅補正：

キャリブレーションをするとき、練習場と本番会場ではバンド幅(黒ライン上と白部分の明暗差)が異なる場合がある。この幅を同一にするため補正する。

$$\text{補正值} = (\text{光センサ値} - \text{白の基準値}) / (\text{黒の基準値} - \text{白の基準値})$$



ノイズの影響が取り除かれている。

バンド幅のズレが補正できている。

## 5-12. 直線走行



【課題】後輪モータの個体差と走行体の重心の偏りが原因で、同一のPWM値を入力しても直進走行できない。

【対策】両後輪モータのエンコーダ値の平均値を目標値として、左右モータのエンコーダとの偏差を求めて、PID制御することで左右モータの回転速度が同一になる。

