



### チーム紹介、目標、意気込み

#### 【チームの活動目標】

- ロボコン活動を通じて、SysMLを使ったMBD設計技術の習得を目指す。
- 初心者から経験者までを教育できる仕組みを作る。

#### 【設計目標】

- 難所をクリアして「完走」（目標リザルトタイム：16秒）
- 「意のままの走り」を実現する（理想のG波形で走る）

#### 【チームメンバと意気込み】

- 太田：ロボコンを社内の人材育成に活用できる仕組みを作る。
- 池上：SW実装設計の技術を身につける。
- 坂下：MBDとはなんぞやを知りたい初心者。

### モデルの概要

- SysMLをつかってモデル記述を行う。
- 走行体システムは、①認知（状況を認知する）、②判断（走行軌道を判断する）、③操作（走行する）の3つのサブシステムから構成する。
- 基本的にライントレースによりコース走行を行うが、ライン喪失した場合でも安全にライン復活しゴールまで走行するために、走行目的地を算出して走行するマップトレース機能を導入する。この両トレース機能により、「コース完走」と「意のままの走り」を両立させている。

### モデルの構成

#### 1. 機能モデル

- 「コースを完走する」ために必要な機能を、以下の3つの大機能で分類する。
  - 認知：各種センサから走行体の座標や速度を計算する。
  - 判断：認知結果から走行モードを決め、「ライントレース」と「マップトレース」とを切り替えて、目標車速やヨーレートを決める。
  - 操作：目標車速からモーター出力を決定する。

#### 2. 構造モデル

- 認知／判断／操作の各機能ごとにサブシステムを構成する。
- 操作システム内にモーターおよびジャイロの倒立APIを有し、走行体に制御信号を送信する。また滑らかなG波形を生成するため、ヨーレート制御モデルの組込む。
- 判断システム内には航法設定と目標運動量演算のブロックを持ち、コース上を安全に走行できる経路を算出する。

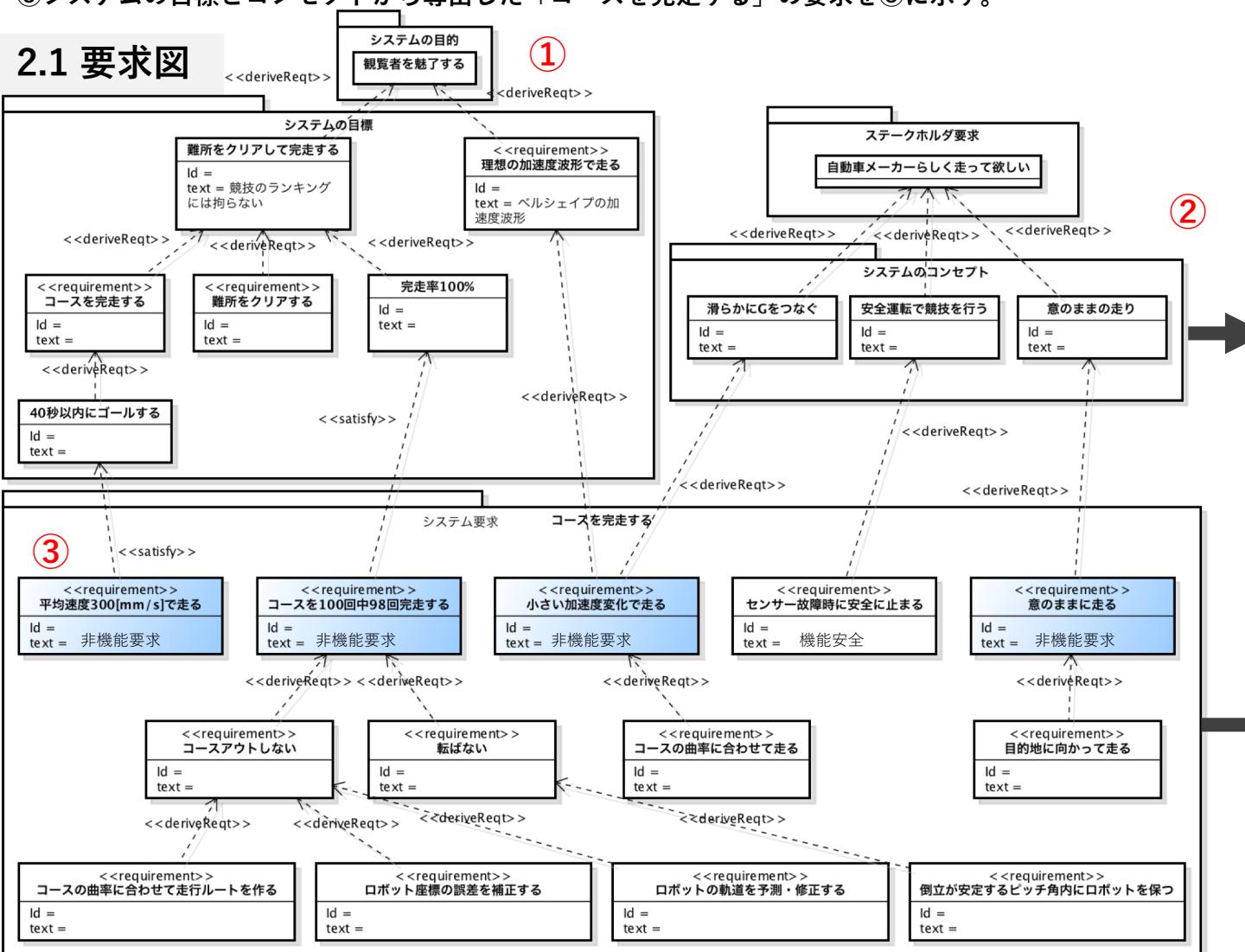
#### 3. 振舞いモデル

- 走行中は、認知／判断／操作の各モードを行き来し、操作モード時にゴール通過を判断したら、走行体を倒立モードへ移行し、停止する。
- コースを走行するシーケンスの周期は倒立APIの周期に合わせて4msecとする。
- マップトレース機能は、走行体の速度ベクトルと目標ベクトルから、目標ヨーレートを決める機能で、滑らかなG波形生成が可能となる。

## 2 機能モデル 要求分析

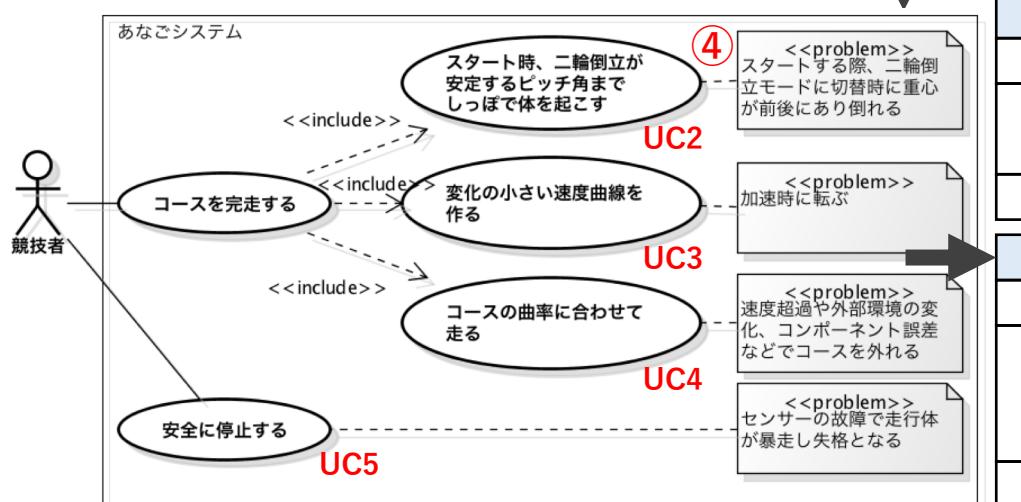
- ①競技結果だけではなく、見ている人を楽しませるためにシステムの目的を「観覧者を魅了する」とした。  
 ②ステークホルダ要求からコンセプトを立案した。  
 ③システムの目標とコンセプトから導出した「コースを完走する」の要求を③に示す。

### 2.1 要求図



### 2.3 ユースケース分析

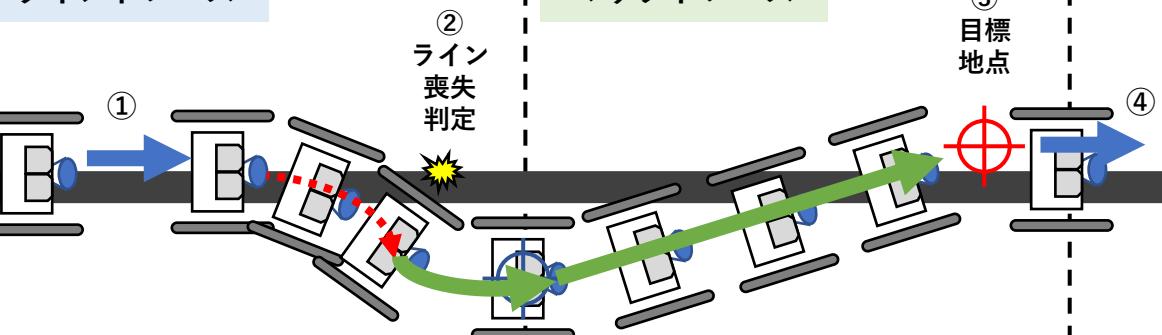
- ④astah SysMLではミスユースケース記述がないため、  
 problemでミスユースケースを表現した。(ネガティブアクターは割愛)  
 ⑤簡単なユースケース記述を右図に示す。(安全に停止するは割愛)



## 機能「コースを完走する」をSysMLで記述する

### 2.2 走行コンセプト

#### ライントレース

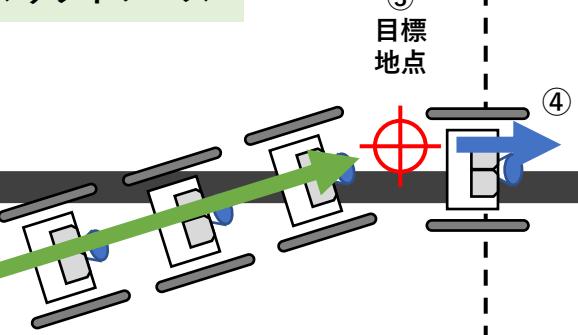


- ①通常はコースの曲率に合わせて  
 ラインをトレースし走る  
 ②ラインを見失った場合、地図座標を  
 もとに走るマップトレースに切替える

「意のままに走る」

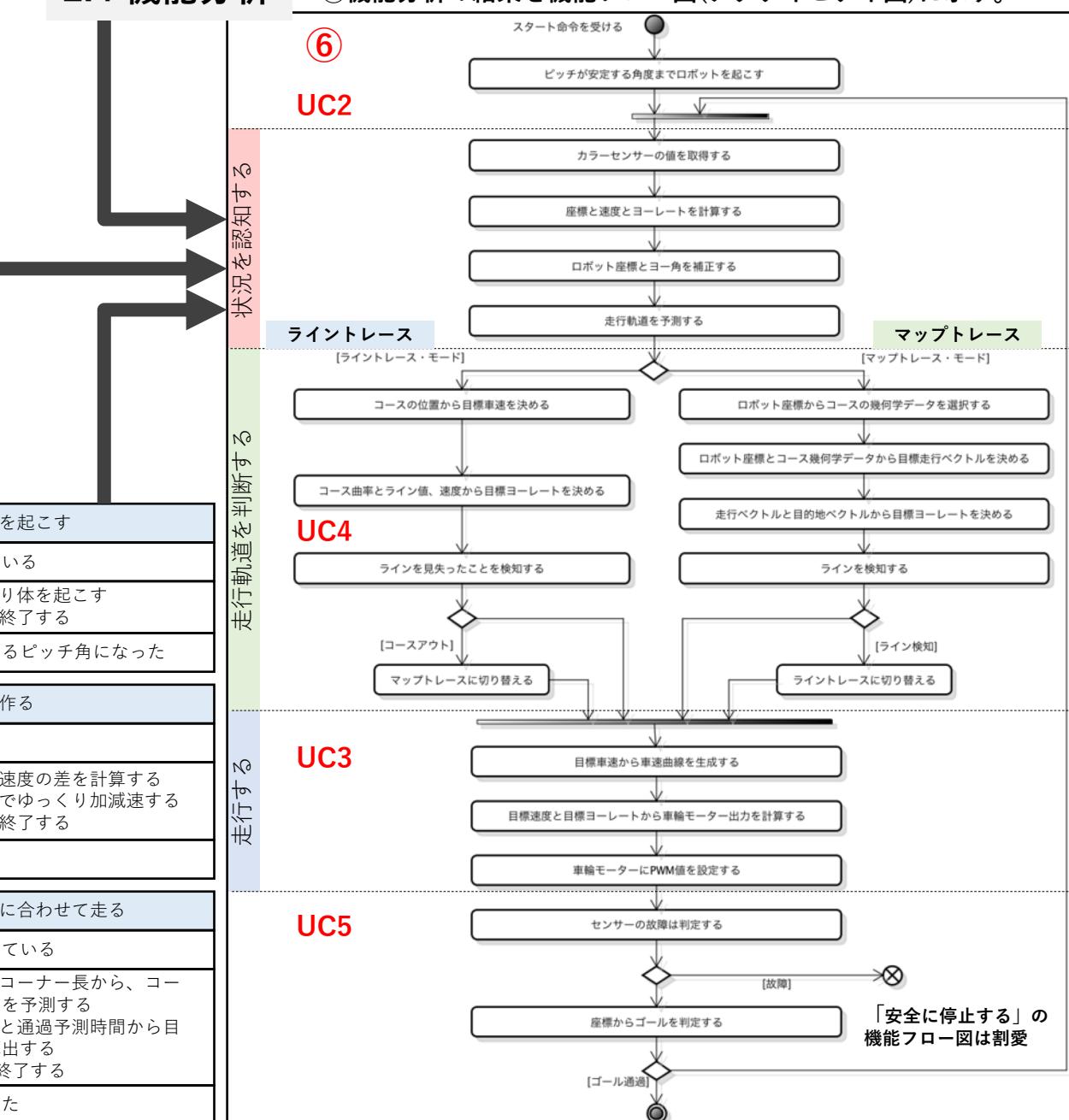
詳細は6頁の工夫点を参照

#### マップトレース



### 2.4 機能分析

- ⑥機能分析の結果を機能フロー図(アクティビティ図)に示す。



### 3.構造モデル

#### 3.1 機能展開図

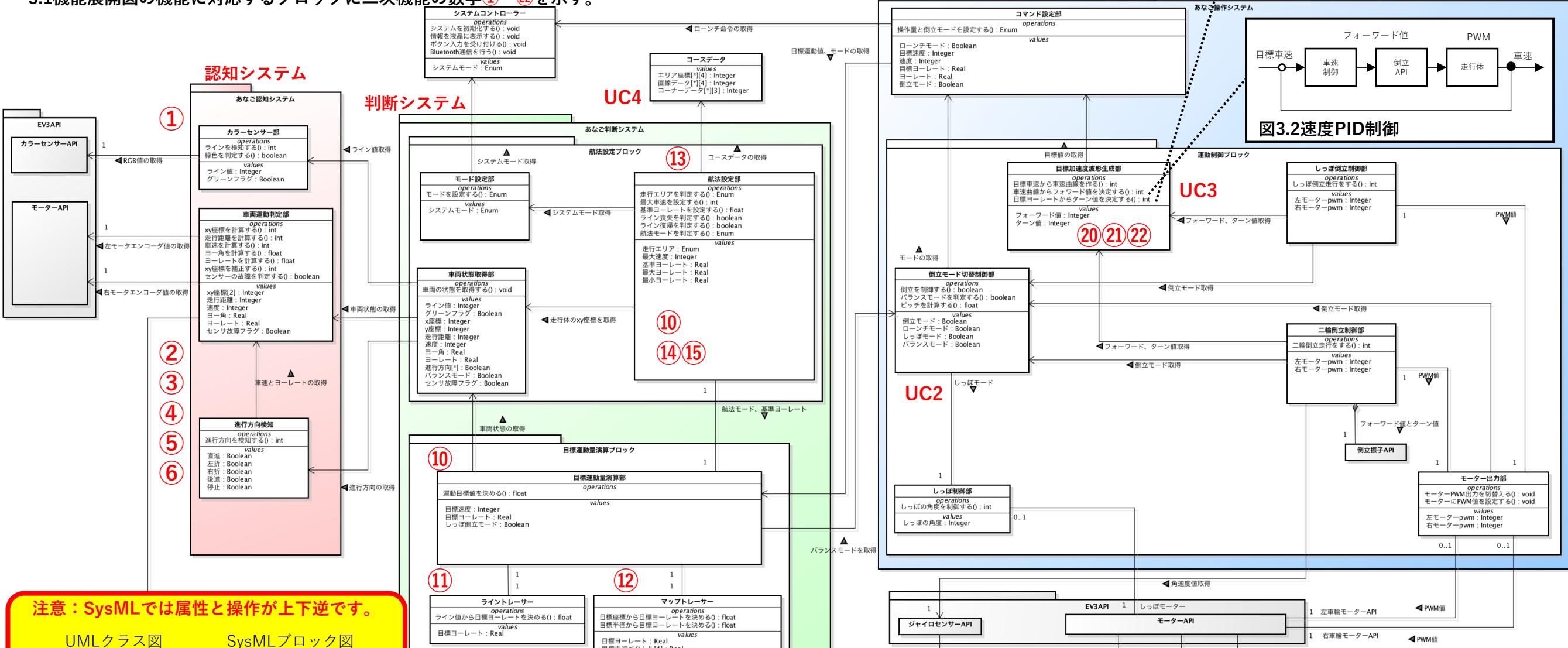
2.4機能分析で定義した機能「状況を認知する」「走行軌道を判断する」「走行する」を二次機能まで展開したものを下記に示す。

| 上位機能      | 一次機能        | 二次機能                            |   |
|-----------|-------------|---------------------------------|---|
| 状況を認知する   | ラインを検知する    | RGB値から輝度を計算し、輝度をライン値0-100%に変換する | ① |
|           | 幾何学的状況を認知する | 走行体の座標を計算する                     | ② |
|           |             | 走行体の進行角度を計算する                   | ③ |
|           | 運動状態を認知する   | 車速を計算する                         | ④ |
|           |             | ヨーレートを計算する                      | ⑤ |
|           | センサ故障を認知する  | センサ出力パターンを監視する                  | ⑥ |
| 走行軌道を判断する |             | 目標車速を決める                        | ⑩ |
|           |             | 目標ヨーレートを決める                     | ⑪ |
|           |             | 走行軌道曲率を計算する                     | ⑫ |
|           |             | 航法を選択する                         | ⑬ |
|           |             | 走行体の座標から走行アリアを判定する              | ⑭ |
|           |             | ライントレースをする                      | ⑮ |
| 走行する      |             | マップトレースをする(実現方法は6.工夫点)          | ⑯ |
|           |             | 車速とコースの曲率から基準ヨーレートを計算する         | ⑰ |
|           |             | ライン喪失を検出する                      | ⑱ |
|           |             | ライン復帰を検出する                      | ⑲ |
|           |             | 前後加速度を作る                        | ⑳ |
|           |             | 横加速度を作る                         | ㉑ |

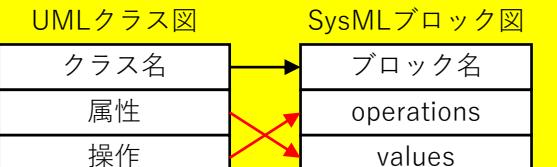
#### 3.2 構造モデル

機能「コースを完走する」の構造モデルをSysMLのブロック図で記述する。

- UMLクラス図の属性はSysMLブロック図のvaluesで表現する(下図)。合成関係はパッケージで表現する。
- システムは「認知システム」「判断システム」「操作システム」の3つのサブシステムで構成する。
- 3.1機能展開図の機能に対応するブロックに二次機能の数字①～㉑を示す。



注意: SysMLでは属性と操作が上下逆です。

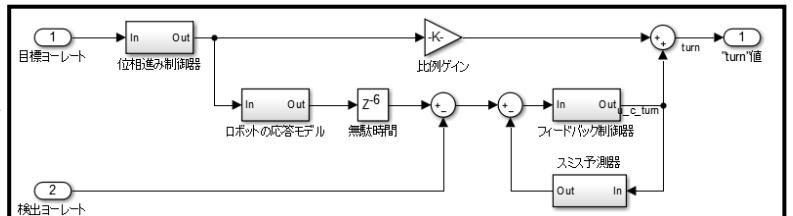


属性の型floatはRealで表現しています。

詳細は6頁の工夫点を参照



図3.1ヨーレート制御モデル



操作システム

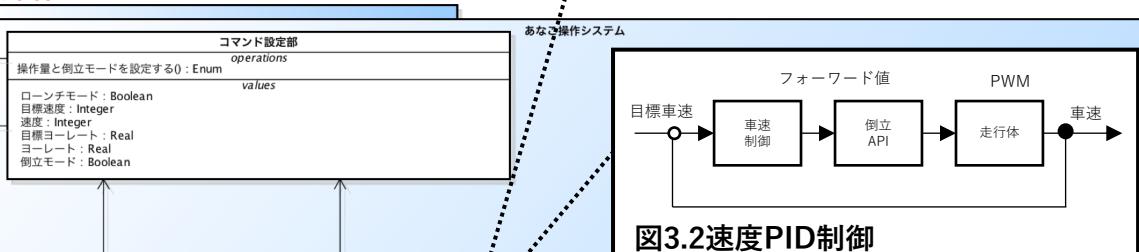
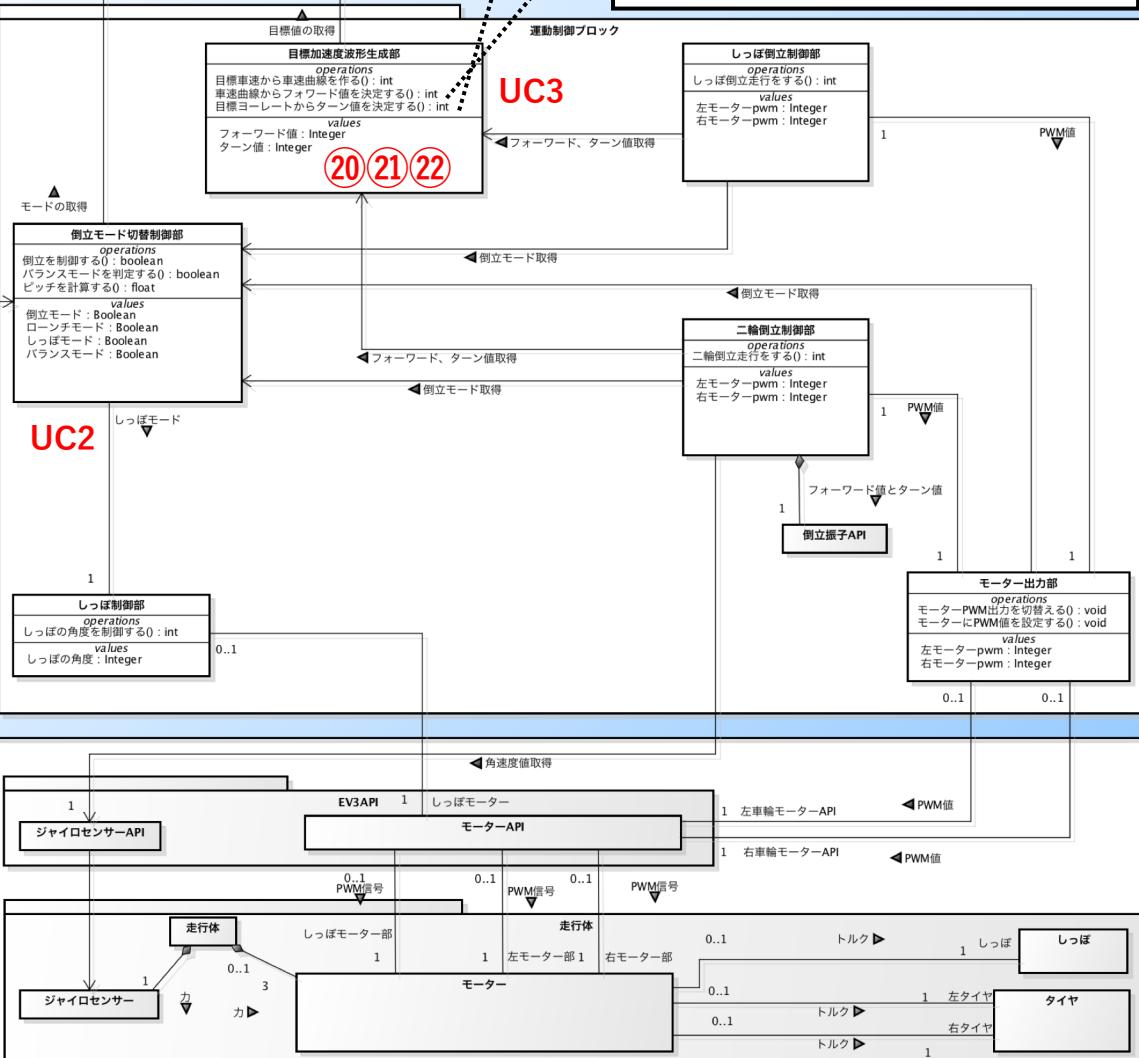


図3.2速度PID制御

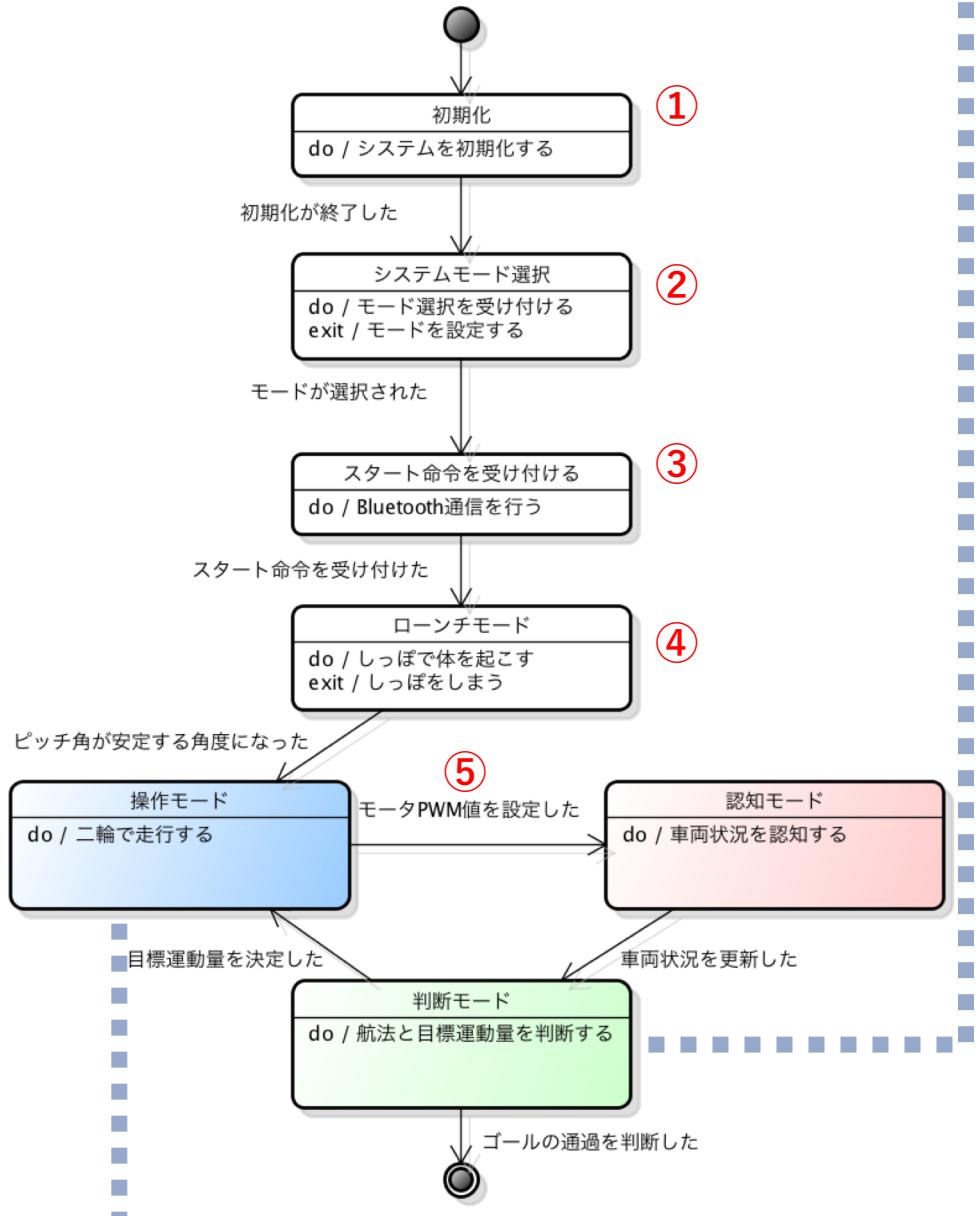


## 4.振る舞いモデル:状態遷移図

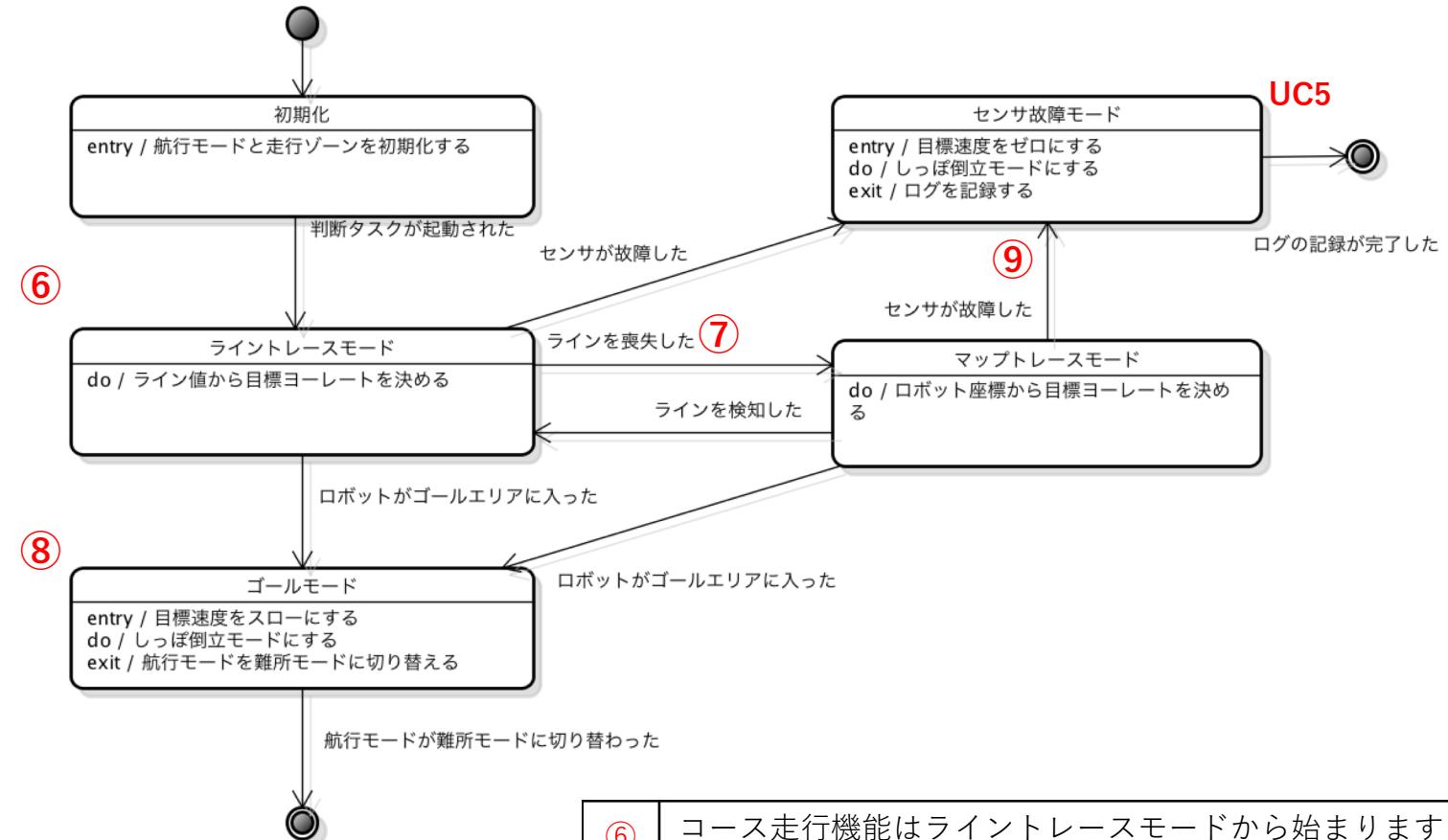
機能「コースを完走する」の状態遷移図をSysMLで記述する。

システム全体の状態遷移図を左図に示す。  
判断と操作システムの状態遷移図を右図に示す。  
認知システムの状態遷移図は割愛する。

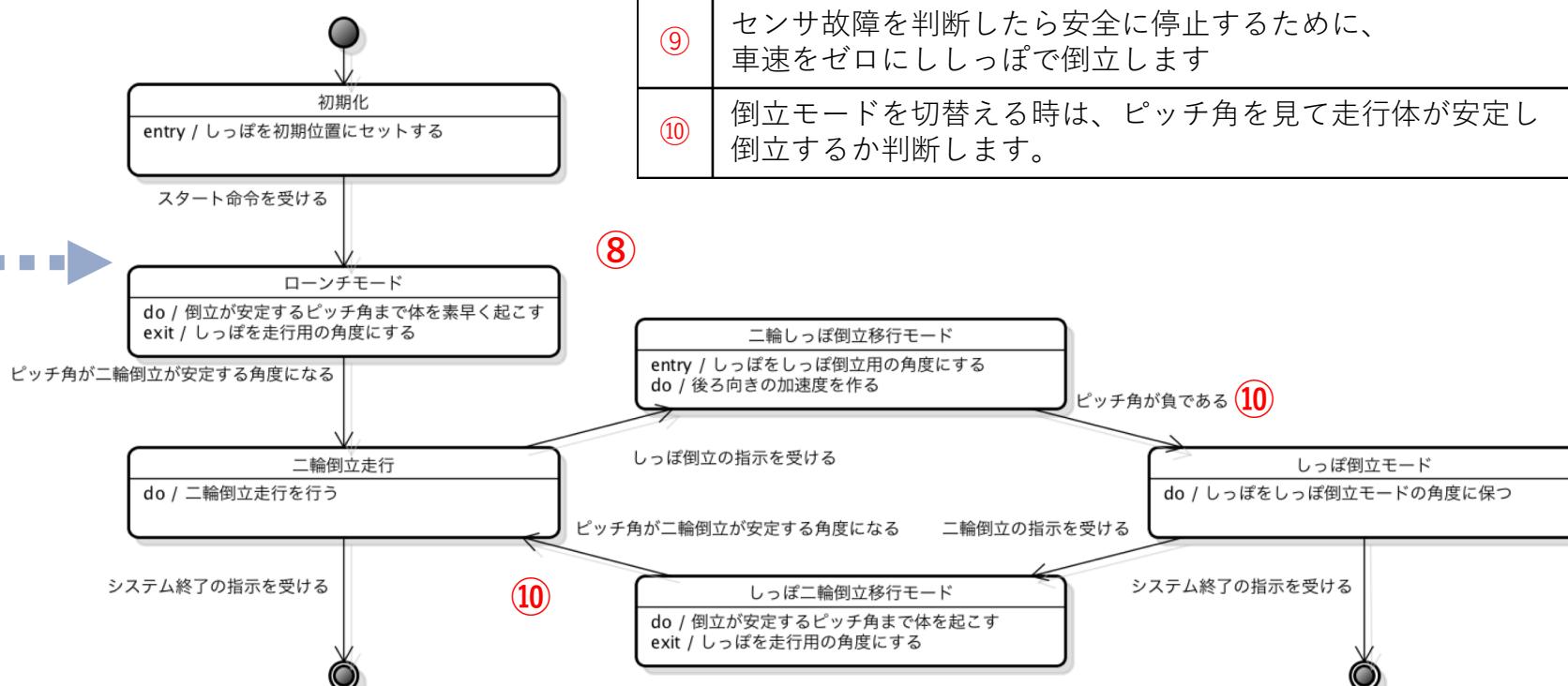
### 4.1 システム全体の状態遷移図



### 4.2 判断システムの状態遷移図



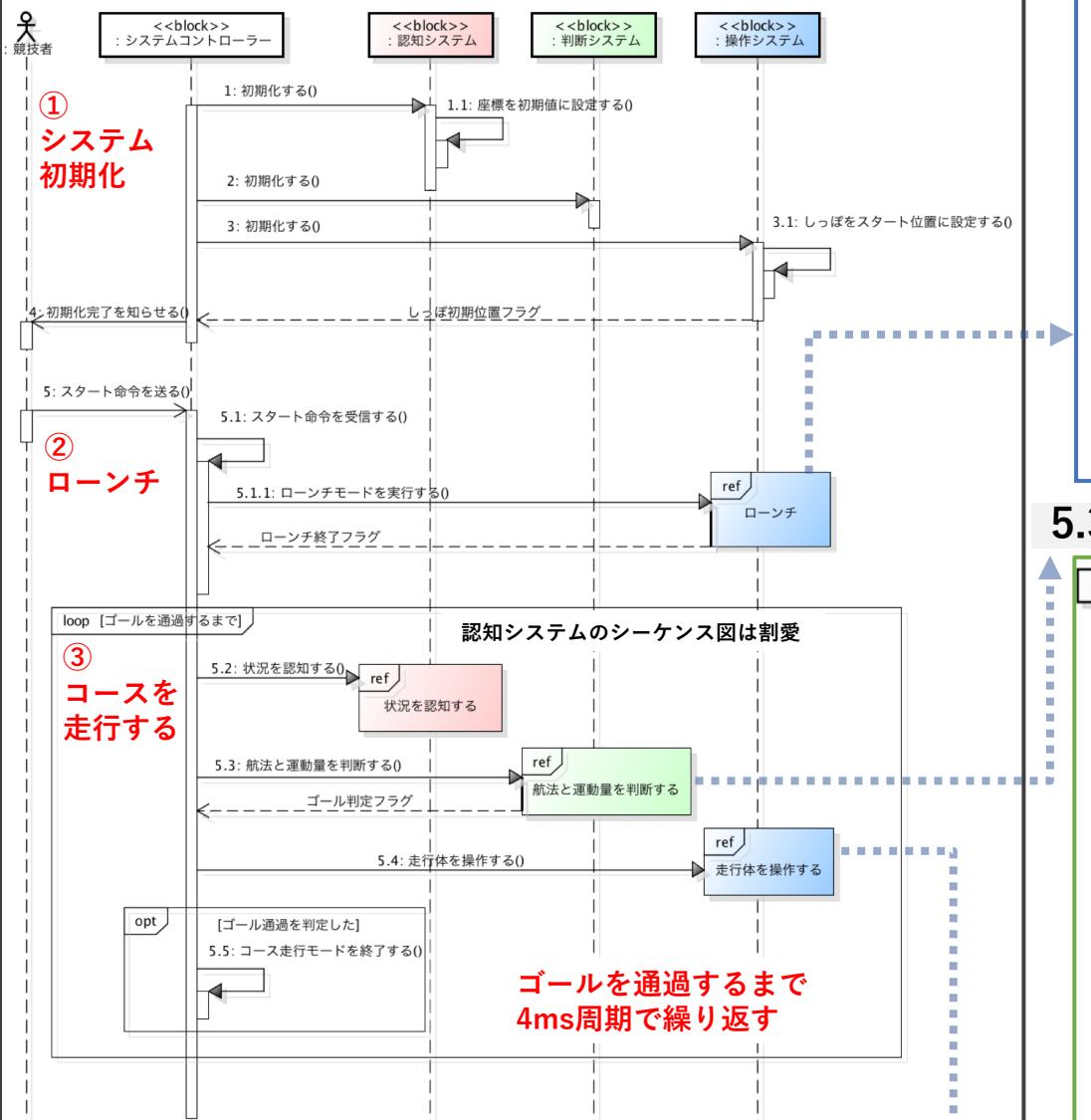
### 4.3 操作システムの状態遷移図



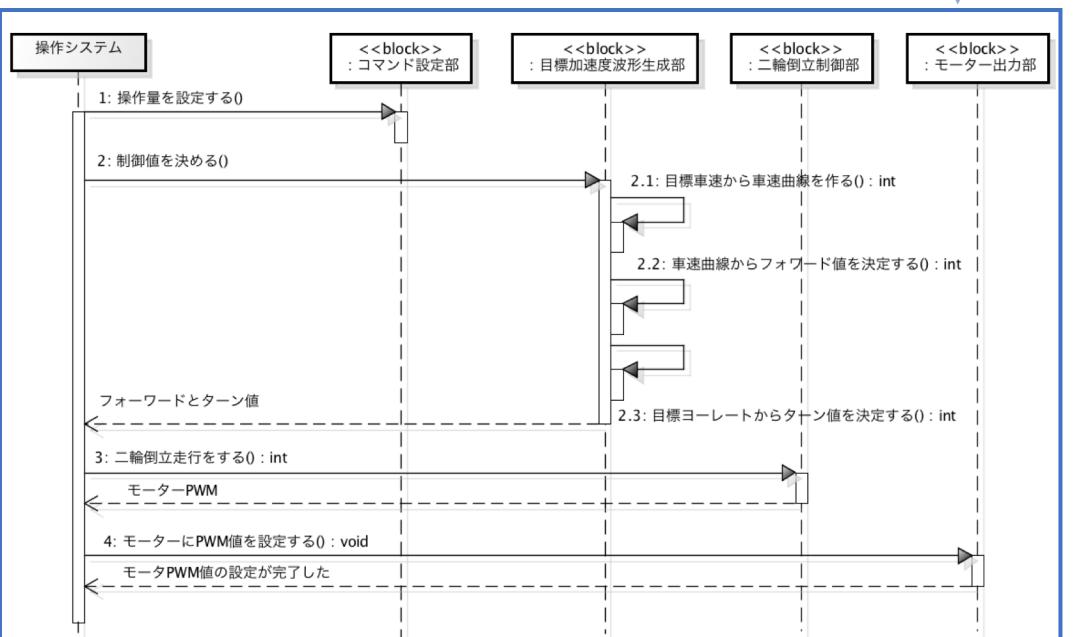
|   |  |
|---|--|
| ① | ジャイロのオフセット、各クラスの変数を初期化する                     |
| ② | 左コース、右コースを選択する                               |
| ③ | Bluetooth通信を確立し、スタート命令を受ける                   |
| ④ | 倒立モードに移行する時に、走行体が後ろに倒れないように、しつぽで前方方向に走行体を傾ける |
| ⑤ | コースを走行中は3つのモードを順々に実行していく                     |

## 5.振る舞いモデル：シーケンス図

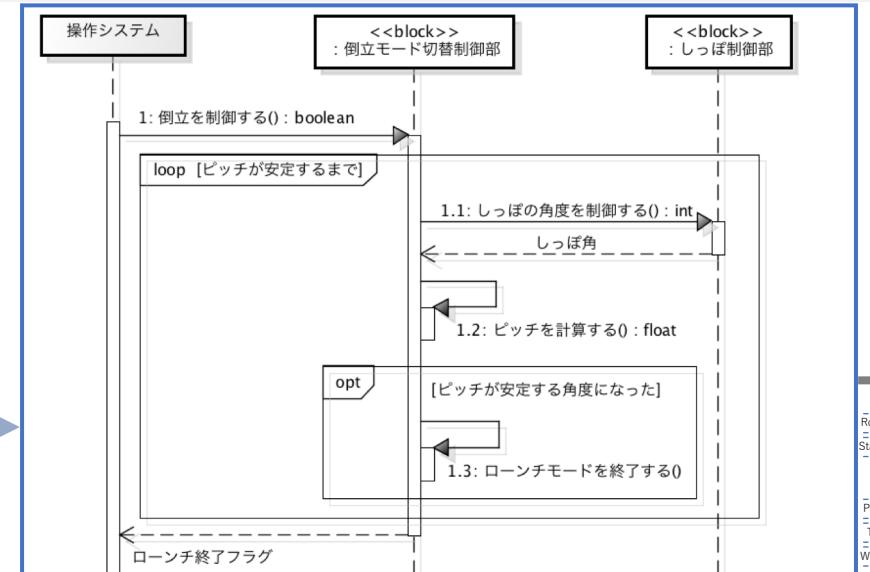
### 5.1 システム全体のシーケンス図



### 5.4 操作システムの操作・シーケンス図

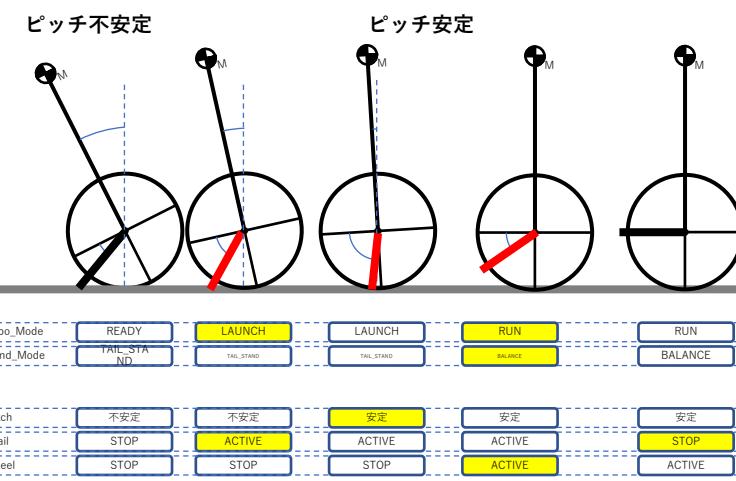


## 5.2 操作システムのローンチ・シーケンス図

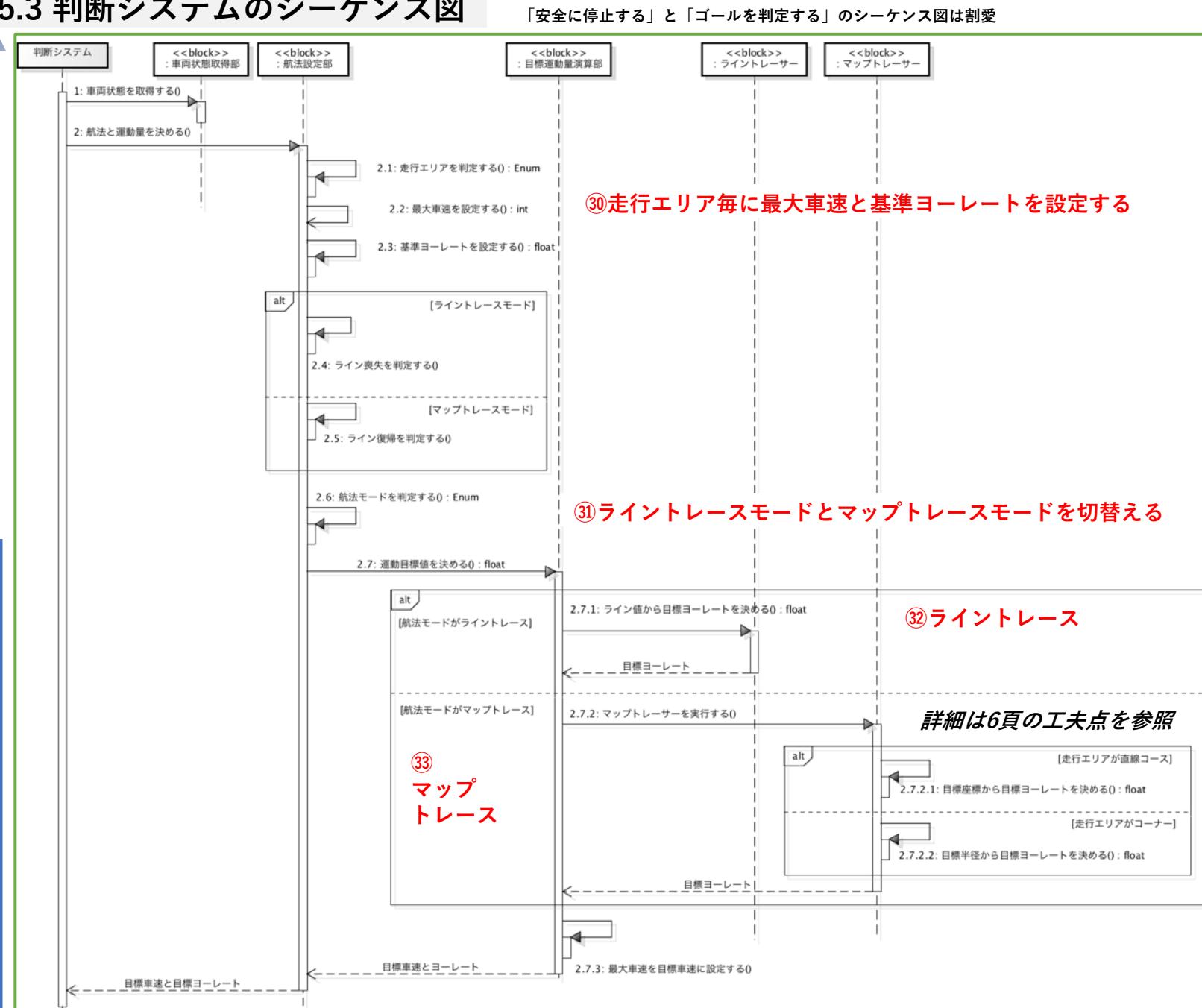


機能「コースを完走する」のシーケンス図を SysML で記述する。

### ローンチ・シーケンス



## 5.3 判断システムのシーケンス図



## 6.工夫点

### 6.1 マップトレース機能

**6.1.1 目標** 要求分析時にシステムの目標「**完走率100%**」を実現するために「コースを100回中98回完走する」という要求を導出し、「意のままに走る」「滑らかにGをつなぐ」というコンセプトを定義した。

**6.1.2 問題** 従来のライントレースでは、光センサー値50を基準とし、50より明るければ右旋回、暗ければ左旋回する。ライン追従性を上げるには旋回ヨーレートのゲインを上げる必要があるが、ヨーレートのゲインを上げると横Gの変化が大きく「**滑らかにGをつなぐ**」事が出来ない。ヨーレートのゲインを下げるするとライン追従性が下がり、ラインの喪失が発生し「**完走率100%**」を実現しない。またカラーセンサーの故障、ケーブルの断線、コネクタの不具合、環境光の変化などのリスクがあり回避策が必要である。

**6.1.3 課題** 上記の問題を解決するための課題は「**ライン喪失を検出した場合、安全にラインに復活する**」と「**カラーセンサーの機能の故障を検出し、安全にゴールまで走行する**」である。

**6.1.4 対策** 「**安全にラインに復活する**」と「**安全にゴールまで走行する**」を実現するために地図データをもとに走行する**マップトレース機能**を導入する。マップトレース機能とは走行体の速度ベクトルとコース上の走行曲線と走行体からの接線から求めた目標ベクトルから目標ヨーレートを決める機能である(右図)。実装を容易にするために走行曲線(コースデータ)は直線( $x_0, y_0, x_1, y_1$ )と円( $x_0, y_0, r$ )で表現した(右図)。直線時には接点を直線コースの終端( $x_1, y_1$ )にし目標ベクトルを計算し、コーナー時は、走行体と円の中心点( $x_0, y_0$ )との距離と円の半径 $r$ の差分から簡易的に目標ヨーレートを決定することで計算コストを削減した(下図)。ライン喪失やセンサ故障時はマップトレースに切り替えゴールを目指す(2.2走行コンセプト参照)。

**6.1.5 効果** ライントレースのみの場合の成功率 15/20回に対し、18/20回に成功率が向上した。ささらにヨーレートゲインを下げる事が可能となり滑らかなG曲線を実現できた(右下図)。ただし障害物がない条件で行った検証結果である。今後は障害物へ衝突するリスクも考慮した対策が必要である。

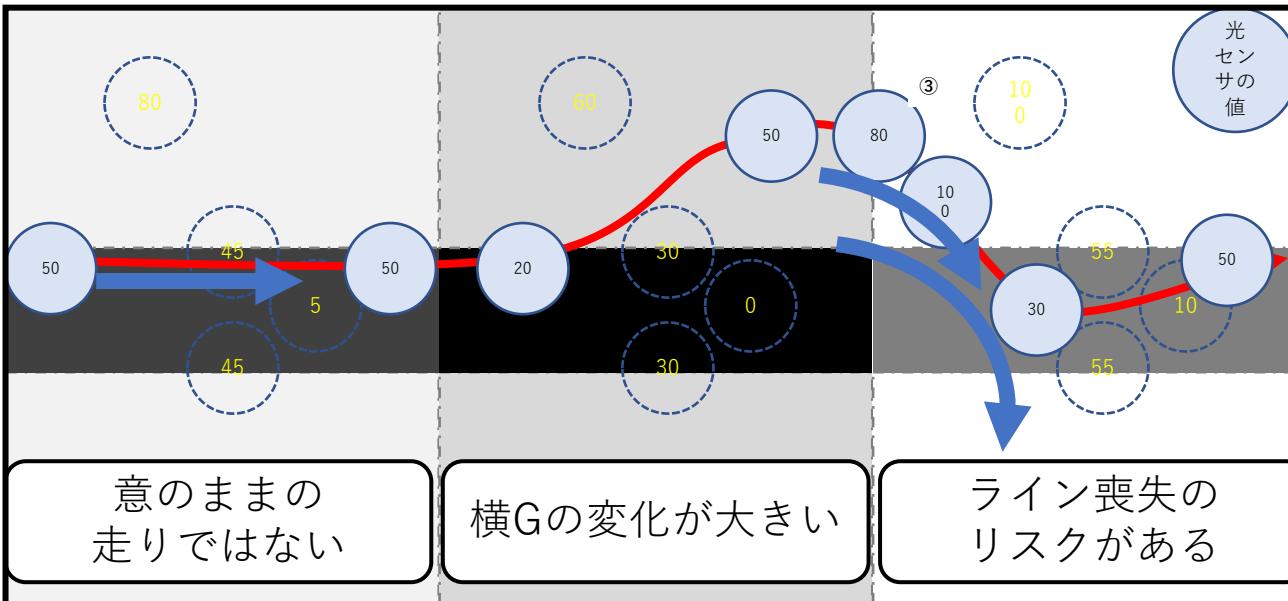
図：コースデータ

```
int STRAIGHT_01[4] = { 0, 160, 2770, 160};
int CIRCLE_01[3] = {2770, 1160, 1000};
int STRAIGHT_02[4] = {3770, 1160, 3770, 2820};
int CIRCLE_02[3] = {4220, 2820, 450};
int STRAIGHT_03[4] = {4220, 3270, 4660, 3270};
int CIRCLE_03[3] = {4660, 2910, 360};
int STRAIGHT_04[4] = {5000, 2790, 3630, 610};
int CIRCLE_04[3] = {3870, 450, 290};
int STRAIGHT_05[4] = {3870, 160, 5200, 160};
int CIRCLE_05[3] = { 0, 0, 0};
```

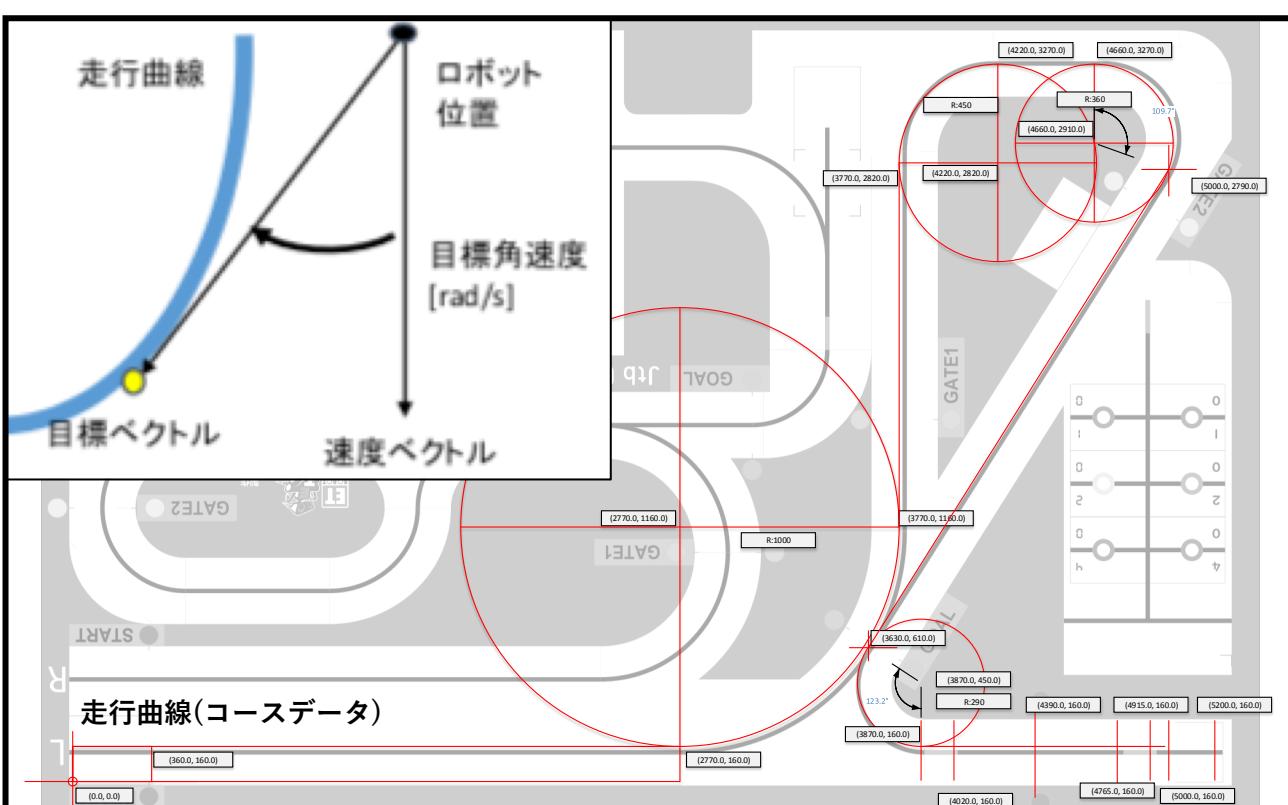
図：簡易マップトレースの実装(コーナー時)

```
case FIRST_CORNER_ZONE:
    x0 = mXvalue+Virtual_point_dist*cos(mYawangle);
    y0 = mYvalue+Virtual_point_dist*sin(mYawangle);
    x1 = CIRCLE_01[0];
    y1 = CIRCLE_01[1];
    a = x1 - x0;
    b = y1 - y0;
    a2 = a * a;
    b2 = b * b;
    r2 = a2 + b2;
    y_t = sqrt(r2) - CIRCLE_01[2];
    if(y_t > 20.0) y_t = 20.0;
    if(y_t < -20.0) y_t = -20.0;
    yawratecmd = (y_t/4.0)*(pg + df*(y_t-y_t_prev));
    y_t_prev = y_t;
    break;
```

図：従来のライントレース



図：マップトレース機能



図：ヨーレート比較（青線：本システム 赤線：従来のライントレース）

