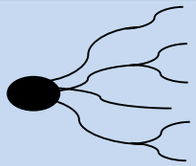
要求分析

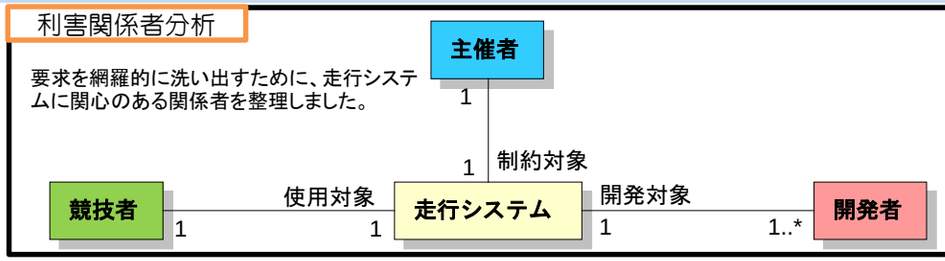


社会人に勝つために必要な要求を洗い出す

社会人に勝つために、漏れなく要求を洗い出しました。
また、品質を意識しながら要求を洗い出しました。

利害関係者分析

要求を網羅的に洗い出すために、走行システムに関心のある関係者を整理しました。



3者の関心事

主催者	ルールを決める
競技者	競技部門優勝する
開発者	楽に開発する

※ で囲んだ要求は競技において特に重要と考え、要素技術として詳しく取り上げました。
詳しくはp.4を参照してください。

※灰色文字の要求について
文字が灰色になっている要求は、既に登場した要求のため、続きの要求は省略

難所の要求は、検討の経緯は省略し、結果のみ記述しました。

走行システムの品質

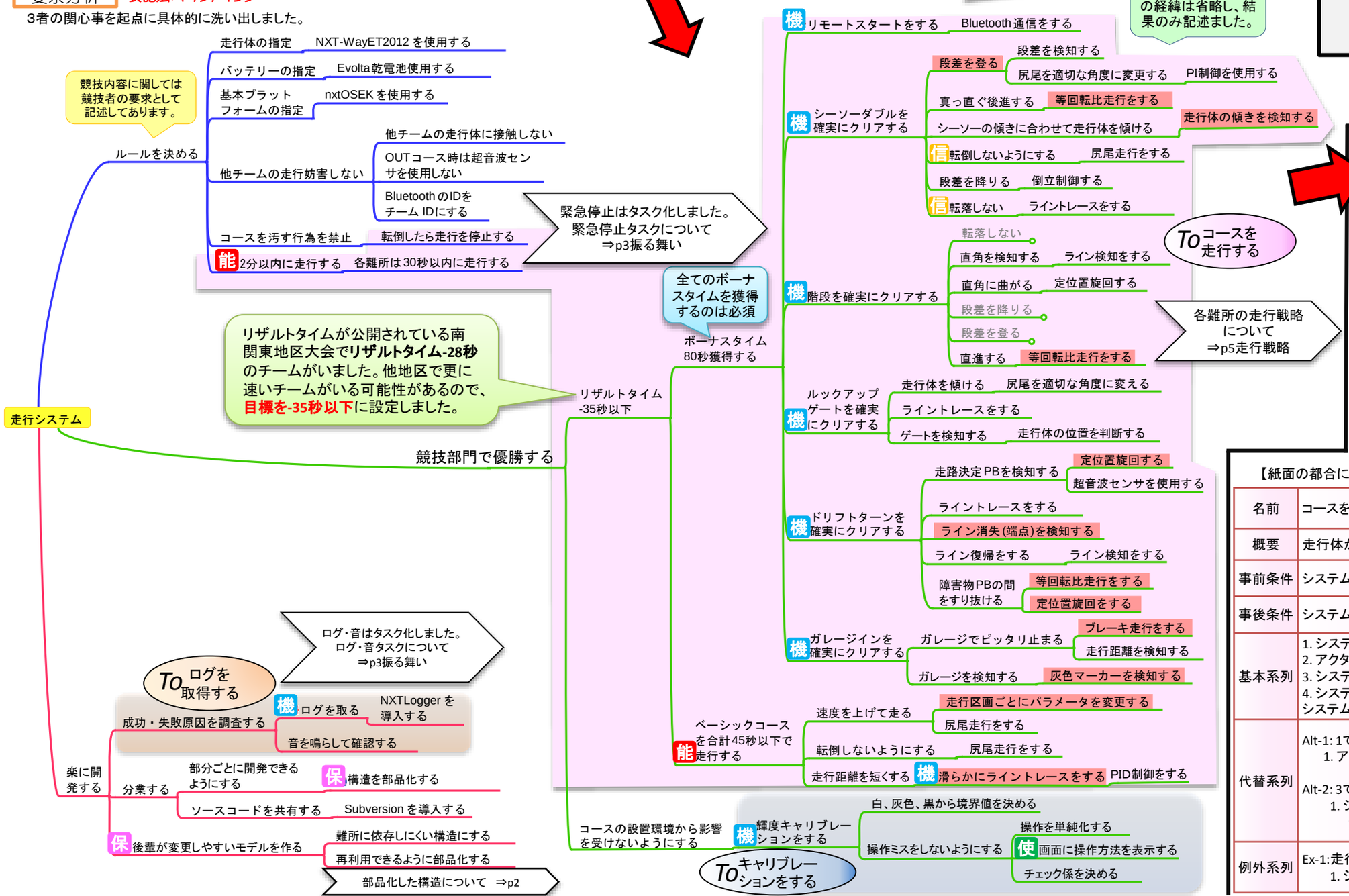
品質に関する要求を洗い出すために、FURPSを用いて走行システムに求められる品質を整理しました。

- 機能性 (Functionality) 機
- 使いやすさ (Usability) 使
- 信頼性 (Reliability) 信
- 性能 (Performance) 能
- 保守性 (Supportability) 保

要求分析

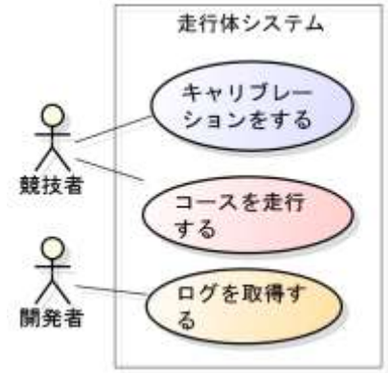
表記法:マインドマップ

3者の関心事を起点に具体的に洗い出しました。



ユースケース分析

アクターにとって価値のある機能をユースケースとして整理しました。



【紙面の都合により他のユースケース記述は省略しています】

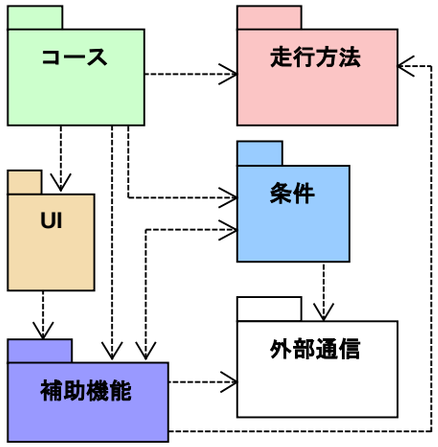
名前	コースを走行する
概要	走行体が走行ルートを走行する
事前条件	システムがキャリブレーションを完了している
事後条件	システムはガレージインで停止している
基本系列	1. システムは走行を開始する 2. アクターはBluetooth通信で走行開始の指示を出す 3. システムは走路を指定された走行方法で走行する 4. システムは条件が満たされたら走行方法を切り替える システムはガレージインをクリアするまで3, 4を繰り返す
代替系列	Alt-1: 1でBluetoothで開始命令を出せなかった場合 1. アクターがタッチセンサを用いて開始命令を出す Alt-2: 3で走路が分岐している場合 1. システムはペットボトルが右側にあった場合は右の走路へ行き、左側にあった場合は左の走路へ行く
例外系列	Ex-1:走行体が転倒した場合 1. システムは走行動作を停止する

後輩が変更しやすいモデルを目指す

多種多様な難所があるため、難所に依存しにくい構造を設計しました。（コースパッケージ）
また、再利用できるよう部品化に努めました。（走行方法パッケージ、条件パッケージ）

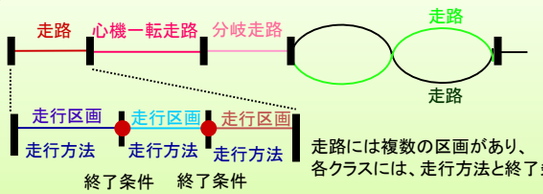
パッケージ構成

後輩が変更しやすいモデルにするため、
パッケージ間の関係をシンプルにし、
再利用できる構造にしました。



「条件」と「補助機能」間の関係は、クラス同士の相互依存ではないため許容する。

パッケージ名	説明
コース	コース、走路、走行区画、などをまとめたもの
走行方法	各区画で使う走行方法をまとめたもの
条件	各区画を終了する条件
UI	使用者に関わる機能
補助機能	転倒時停止など、走行を補助する機能
外部通信	PCと通信する機能

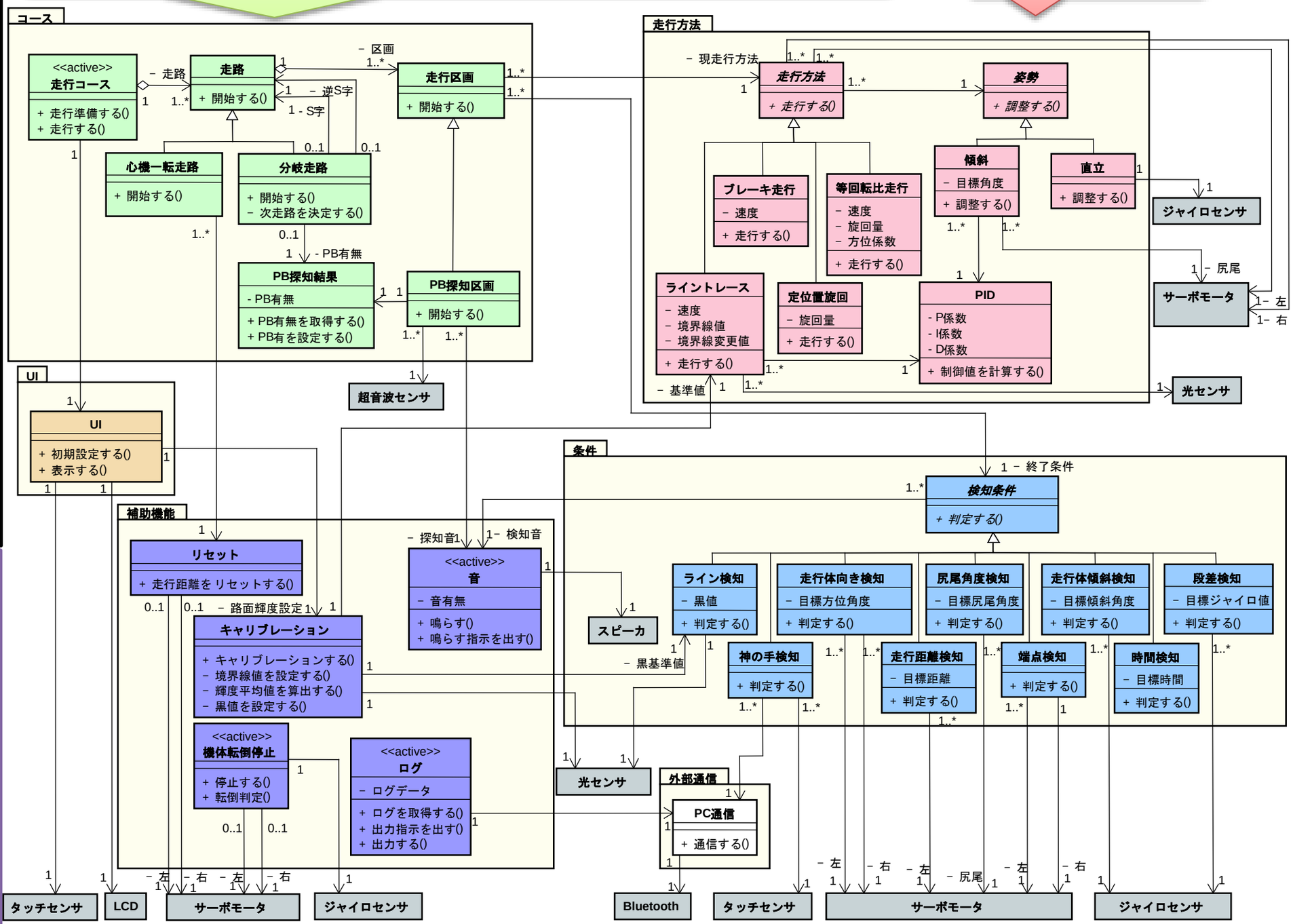


走路には複数の区画があり、
各クラスには、走行方法と終了条件がある。

心機一転走路･･･走路の開始地点を基準に走行距離を測定するためのクラス。
具体的には走行距離や向きを0にリセットする。

分岐走路･･･走路の開始地点で2つに走路が分岐していることが表すためのクラス。
どの走路を選択するかはPB探知結果の情報をもとに本クラスが決定する。

姿勢･･･走行体の傾き具合のこと
傾斜とは尻尾走行のことで、直立とは倒立走行のことである。尻尾角度やバランスを制御する。



クラスの補足説明

＜全体＞

※ ディバイスは、見やすさの都合上、複数記載してあります。

※1 アクティブクラスの表記について･･･クラス図をastahで作成したが、表記が分かりにくいためステレオタイプで<<active>>と記入しました。

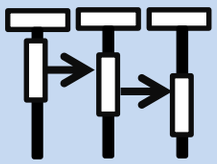
＜条件パッケージ＞

※2 「神の手検知」とは･･･Bluetooth通信やタッチセンサを通じたユーザーからの介入を検知する。リモートスタートで利用。

＜補助機能パッケージ＞

※3 ログへの関連は多いため省略した。

※4 「リセット」とは･･･サーボモータの回転角度を0（ゼロ）にセットする。走行距離や向きがリセットされるので、その地点を基準にサーボモータの回転角度を累積していくことになる。

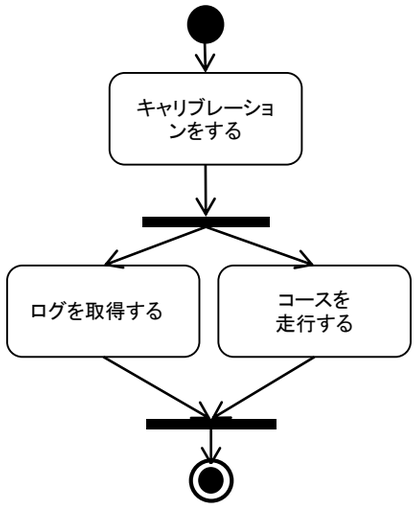


構築した構造で機能の実現可能性を検証する

「コースを走行する」ユースケースの実現可能性を検証しました。
難所に依存しにくい構造であることをドリフトターンを例に検証しました。

全体の振る舞い

ユースケース同士の関係性を整理しました。



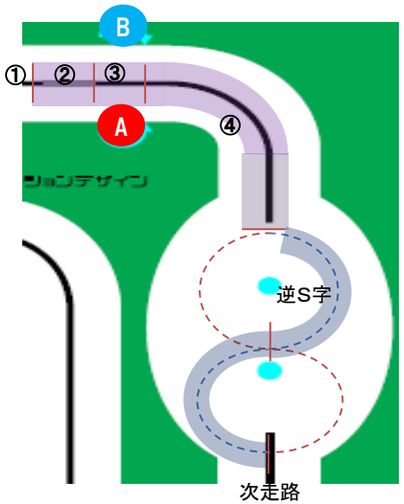
<<「コースを走行する」のユースケース記述>>

1.システムは走行を開始する
2.アクターはBluetooth通信で走行開始の指示を出す
3. システムは走路を指定された走行方法で走行する
4. システムは条件が満たされたら走行方法を切り替える
システムはガレージインで停止するまで3, 4を繰り返す

ドリフトターンの振る舞い

ドリフトターンではS字走行をする場合と逆S字走行をする場合とで走路を分岐する必要がある。そのために分岐走路クラスがあり、スーパークラスの走路クラスの振る舞いとは一部異なる動きをする。その振る舞いを右図に示す。合わせて基本的な走行の振る舞いの枠組みの中で実現できるかどうかについても検証した。

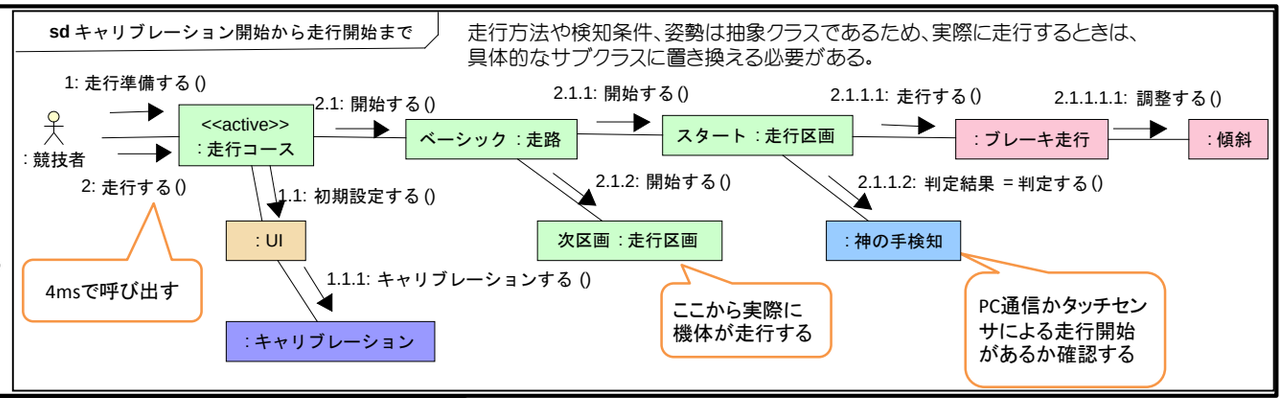
※今回はペットボトル(PB)が左(B側)に
あり図中の青いルートを通る場合を想定した。
※図中のLTとはライントレース走行を表す。



スタートまでの振る舞い

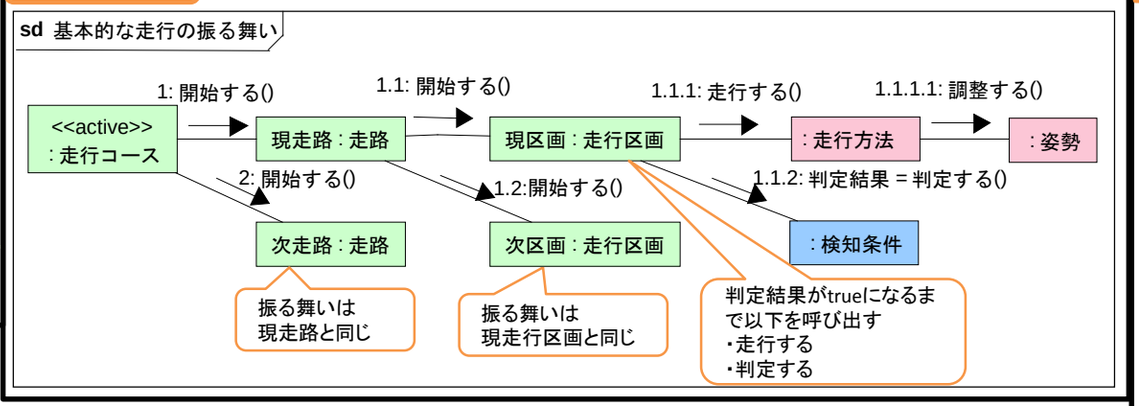
<<スタートまでの振る舞いを検証しました>>

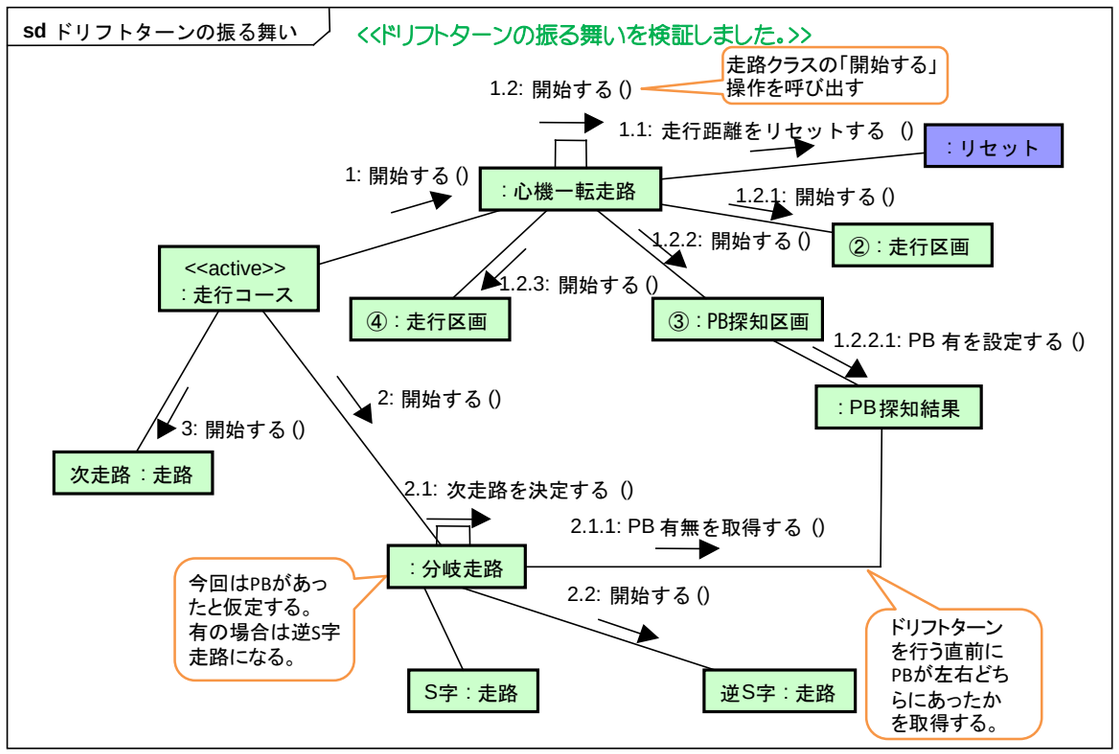
スタート時はまずUIを通じてキャリブレーションをした後にスタート用の区画を用意した。
スタート命令がでてからユースケースの「コースを走行する」が開始されるのではなく、「コースを走行する」の中で、スタートを待つように設計した。
走行開始命令を待ってる間はブレーキ走行を行っていると考え、スタート区画の終了条件は神の手検知とし、PCかタッチセンサからの走行開始命令を待つ。
命令が出れば次区画に移行し、実際に機体が走行を開始する
これが基本的な走行の振る舞いと同一な振る舞いで実現できるかどうかについて検証した。



コースを走行する

<<「コースを走行する」の基本的な振る舞いが実現できるか検証しました。>>





並行性設計

<<設計指針>>

次の理由でタスク分割を行いました。
- 倒立制御やサウンド等の異なる周期で実行しなければならない処理がある。
- 倒立制御などの優先的に実行しなければならない処理がある。

<<設計内容>>

※優先度は数字が大きい方が高い。

タスク名	実行周期	優先度	処理内容
走行タスク	4ms	3	コースを走行する
ログ・音タスク	10ms	2	サウンドを鳴らす。データをPCに送信する
緊急停止タスク	100ms	1	転倒時に走行を停止させる

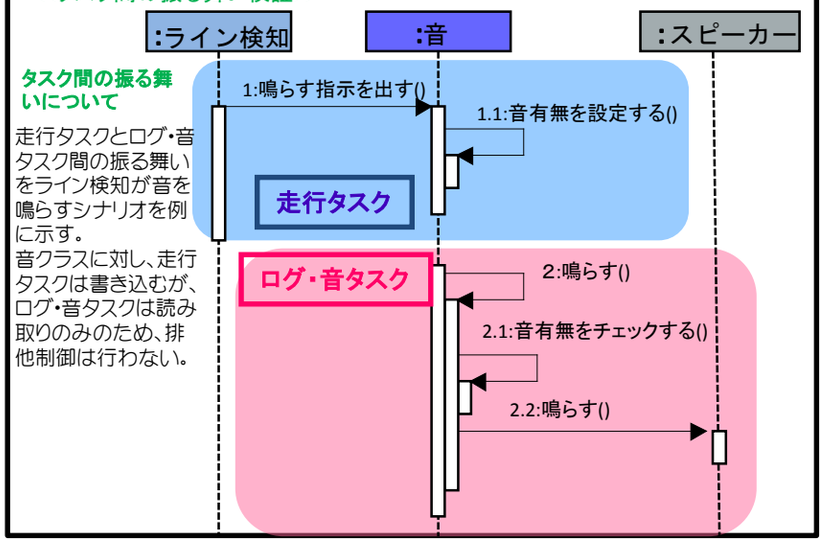
走行タスクは倒立制御を4ms周期で確実に実行する必要があり、優先度高にする。
緊急停止タスクは処理に負担をかけさせないため100ms周期で実行し優先度低にする
ログ・音タスクは最短10msで音を鳴らす必要がある。また、ログは通信時の時間的ロスが走行に影響しないように音と同じタスクにした。相対的に優先度中にする。

超音波探査の処理は走行タスクで10回に1回動作させることで実現した。

<<緊急停止について>>

転倒時に「緊急停止タスク」がモーターに停止命令を出しても、「走行タスク」から駆動命令が来る可能性があるため「緊急停止タスク」はモーター制御に排他をかける。

<<タスク間の振る舞い検証>>



要素技術

社会人に勝つために確実に難所をクリアする技術を開発する

尻尾の角度について

階段 / ドリフトターン / 坂(ベーシック) / シーソーダブル / ガレージイン

走行体傾斜検知

注目 傾斜が認識可能

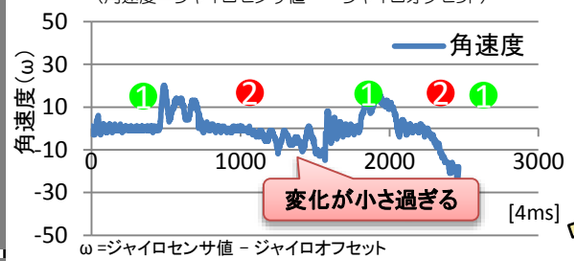
課題

・尻尾走行時において、シーソーの傾斜は影響が大きく、走行体が転倒してしまう可能性がある。そのため、傾きをジャイロセンサ値で検知したいが、ジャイロセンサ値をそのまま使用しても変化が小さいため、検知できない。

解決策

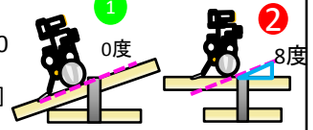
角速度を足した値で傾きを検知する。

（角速度＝ジャイロセンサ値 － ジャイロオフセット）



ω =ジャイロセンサ値 － ジャイロオフセット

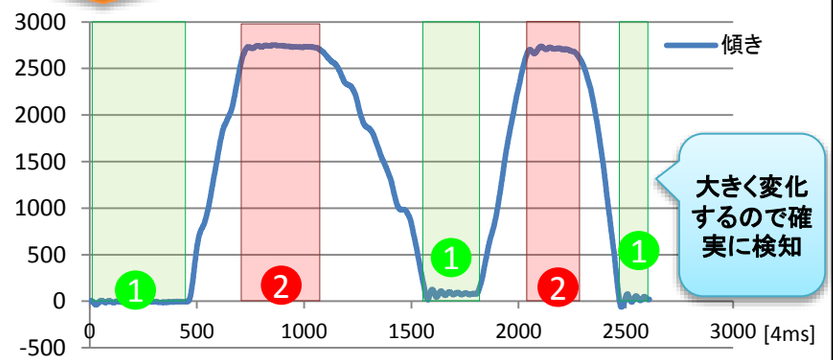
左のグラフは、斜度を8度変える操作（①、②）を2回行ったときのデータ



そこで、右記の式を使用して、角速度の偏差を足していくと

$$\theta = \sum_i \omega(t_i)\Delta t_i$$

t ： かかった時間
$\omega(t_i)$ ： t_iの時の角速度
θ ： 傾斜角度



積分したことで変化が大きくなり、データのノイズを無視できました。
これで確実に走行体の傾斜を検知することが可能になりました。

《検証》 シーソーが傾きはじめたときの検知確率
導入前 0%（0回/30回） 導入後 100%（30回/30回）

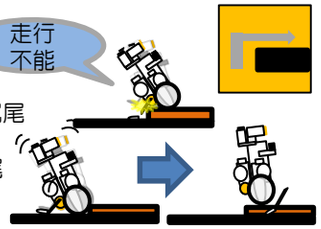
※加速・減速時は、その影響を大きく受けるので、等速度時でのみ使用できます。

使う場面

段差登り

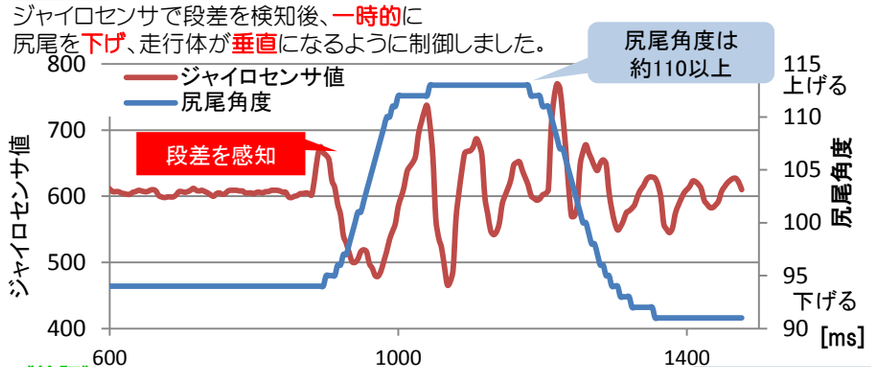
課題

段差は走行体が転倒する確率が高いため、タイヤと尻尾の3点で登り、転倒を回避することにしました。
しかし、尻尾走行の尻尾角度のまま段差を登ると、尻尾が**段差に引っ掛かってしまい**登ることができません



解決策

ジャイロセンサで段差を検知後、**一時的**に尻尾を**下げ**、走行体が**垂直**になるように制御しました。



《検証》
段差2段分の成功率
導入前（倒立） 12回/30回
導入後 30回/30回

この段差を登る一連の動作を右のアクションで表記します。（次ページで使用します）

段差登り
［登った］

使う場面

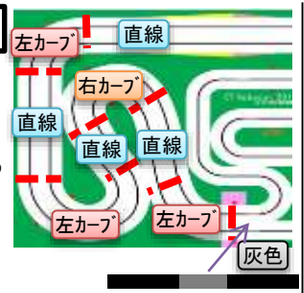
ライントレース走行

課題

・ON/OFF制御のライントレースはタイムが遅くなってしまう。
・全速力で走行するとカーブや灰色マーカーで脱線してしまう

解決策

・輝度PD制御を用いてなめらかにライントレースをする。
・**境界線値**を区画に合わせて変更する。



境界線値の変更方法
直線 ：基準値
右カーブ ：黒値寄り
左カーブ ：白値寄り
灰色 ：白値寄り

<<検証>> 脱線した回数
導入前 10回/10回
導入後 0回/10回

定位置旋回

課題

旋回時に旋回前と旋回後では走行隊の位置が変わってしまう。

解決策

右モーターに指定した回転速度に -1 を掛けた値を左モーターの回転速度に指定する。

<<検証>> -30 ⇩ ⇧ +30

誤差 1回転につき5mm

使う場面

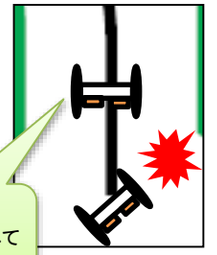
端点検知

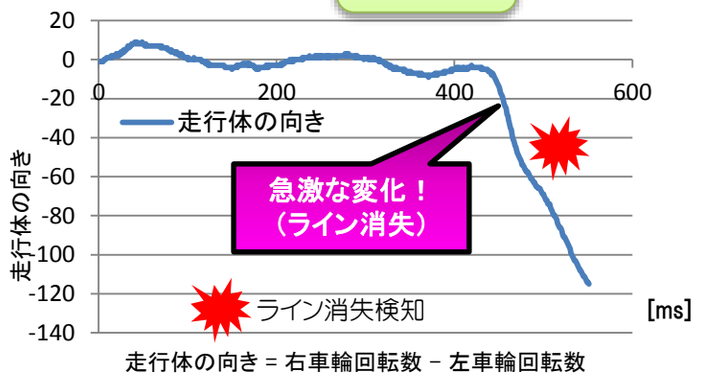
課題

ライン消失の瞬間を起点として次の動作を始めるため、ライン消失が確実に検知できないと次の動作に影響が出てしまう。

解決策

この走行体の急激な向きの変化でラインの消失を検知します。





走行体の向き = 右車輪回転数 － 左車輪回転数

※同様に**灰色マーカー**に進入した時にも走行体の向きが急激に変化するので、灰色マーカー検知も可能です。

《検証》
灰色マーカー検出回数
導入後 30回/30回

使う場面

PB検知

課題

PBと走行体の距離が遠すぎたり、近すぎたりすると検知ができない。

解決策

最適な距離で検知する。

<<検証>> 検知した回数
導入後 9回/10回

	1回	2回	3回
20cm	○	○	○
25cm	○	○	○
30cm	×	×	○

以上の票より、PBと走行体の距離は**20～28cm**が最適とした。

使う場面

等回転比走行

注目 特定の角度での旋回や、直進走行をするための技術

課題

ドリフトターンではラインが無いので、常に一定の旋回が要求されるが、単に旋回しようとするとモーターの個体差や電圧の影響を受け毎回旋回後の位置が異なってしまう。

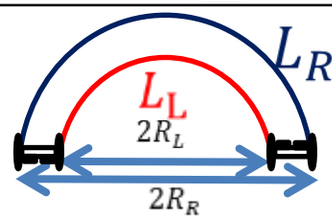
解決策

左右車輪の回転数の比率を一定になるように調節しながら旋回する。

係数は右記のように算出します。
旋回したい軌道の半径が求まればその値から算出できます。

$L_L : L_R$ = 1:k（一定） kは係数
となるように旋回すれば良い

※係数を 1 にすると、左右の回転数が同じになるので**直進走行**をします。



L_R:右車輪が描く弧の長さ R_L:左車輪が描く弧の半径
L_L:左車輪が描く弧の長さ R_R:右車輪が描く弧の半径

係数kは、軌道の半径から求める

$$L_L = 2R_L \times \pi \times \frac{\theta}{2\pi}$$
$$L_R = 2R_R \times \pi \times \frac{\theta}{2\pi}$$

よって
$$L_L : L_R = R_L : R_R = 1 : \frac{R_R}{R_L}$$

《検証》
半径21.75cmの弧を180度走行したときの位置のずれ
導入前 ±5cm 導入後 ±1.5cm （平均値）

使う場面

ブレーキ走行

課題

急に速度を0にして走行を停止させると転倒してしまう。

解決策

急に速度を0にするのではなく、実行周期(4ms)ごとに減速させ、速度を0にする。

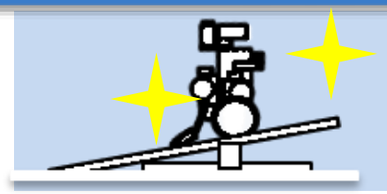
使う場面

<<検証>> 転倒する確率
導入前 8回/10回
導入後 0回/10回

5 走行戦略

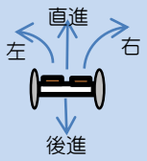
科学の妖精 聖望学園中学校高等学校

要求分析 | 構造 | 振る舞い | 要素技術 | 走行戦略



要素技術の組み合わせで難所をクリアできることを示す
構造との対応関係を示す

等回転比走行について
走行方法に記載された「等回転比走行(進行方向)」進行方向は右図参照



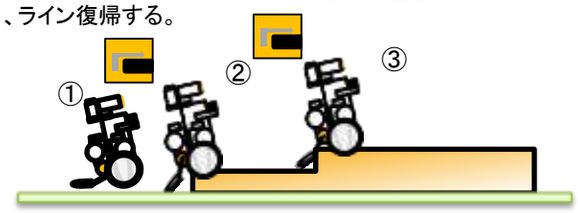
水色のアクションは直立(倒立中)を示します　直立

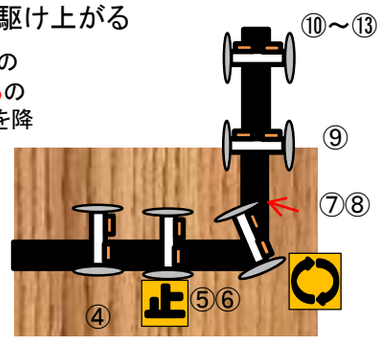
このアクションは傾斜(尻尾走行中)を表します　傾斜

階段

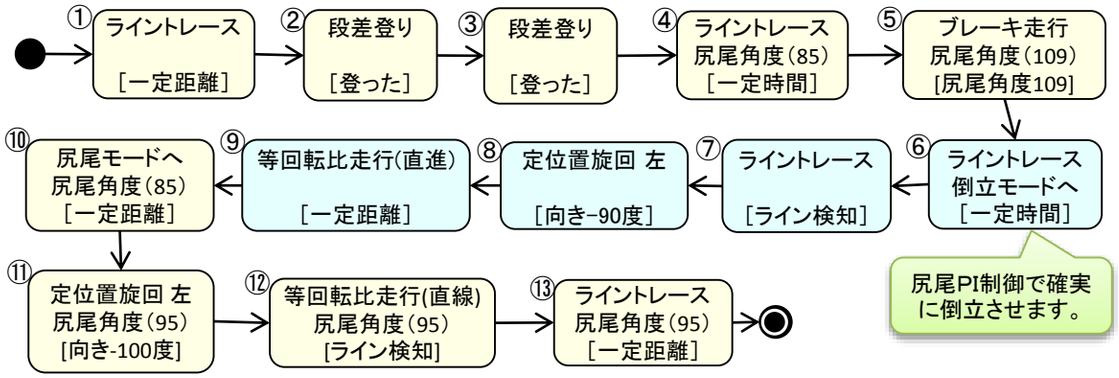
尻尾走行で一気に階段を駆け上がる

<<概要>>要素技術の段差登りで階段を登る。登ったら倒立し、ラインの左エッジをライントレースする。すると、直角部でライン上を通り過ぎるので、ライン検知で直角部を認識。定位置旋回し、まっすぐ進んで段差を降り、ライン復帰する。





向きは①を0度とし、右が＋、左がーになります。



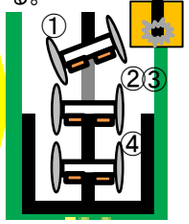
「段差登り」は要素技術p.4を参照してください。

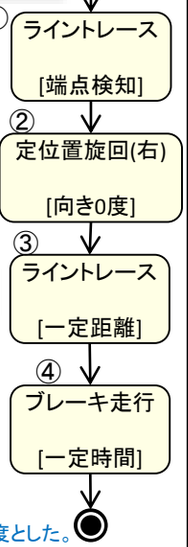
ポイント

段差を登った後、尻尾走行から倒立走行に切り替えます。この時、P制御だけだと、目標値とのズレが出てしまうので、尻尾の角度をPI制御で制御しました。これで、倒立直後も走行体のぶれがほとんどありません。

ガレージイン

<<概要>>
灰色検知をし、ラインと平行になるように向きを補正、ガレージの壁に当たらないように低速で滑らかなライントレース走行をし、停止する。

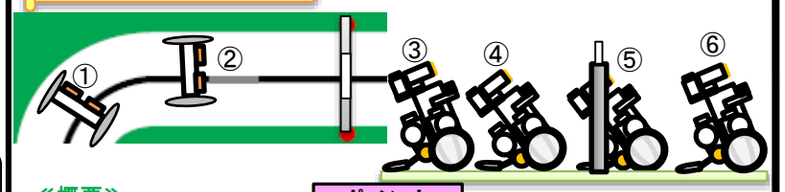




向きは①を0度とした。

ルックアップゲート

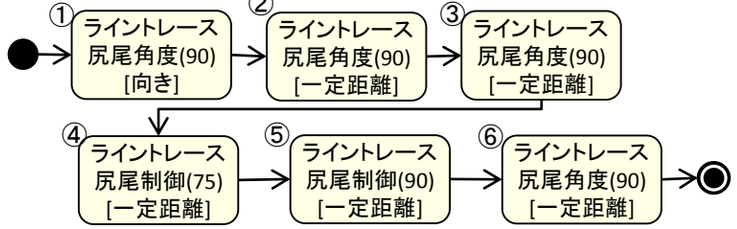
速度を下げずに通り抜ける



<<概要>>
ゲートの前に来たら走行体を傾けてゲートをくぐる。

ポイント

ゲート位置は超音波センサを使わず、手前のカーブを利用して、走行体の向きで検知します。
走行体を傾けた状態から走り出すと尻尾に重心がいき、タイヤが空転してしまう。そのため、走行速度を下げずにゲートを通過します。



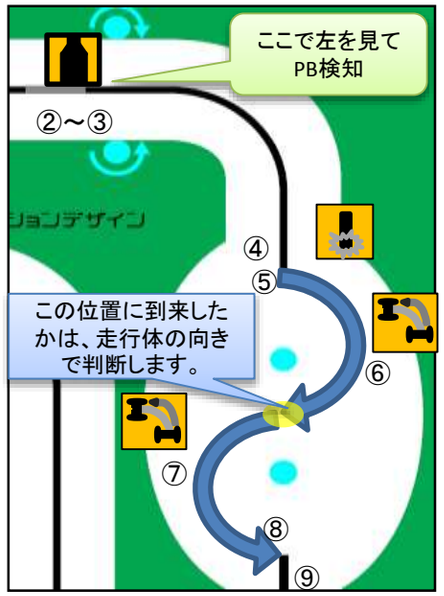
ドリフトターン

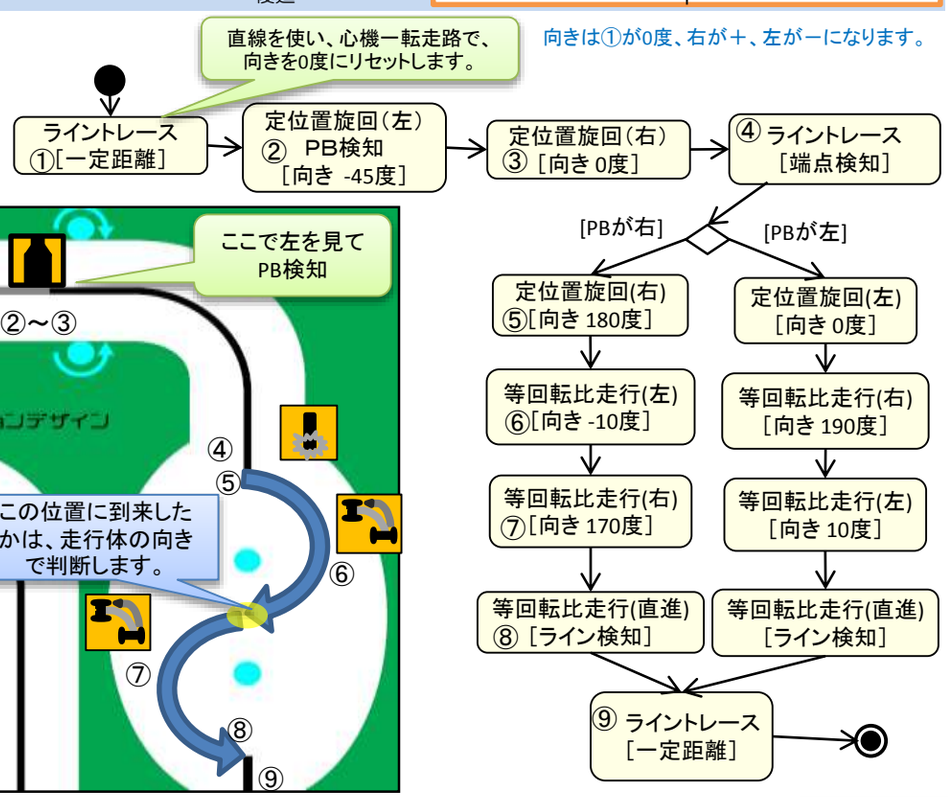
尻尾走行でPB間をすり抜ける

≪概要≫
ペットボトルの配置を検知したのち、ラインが消失するまでライントレース、そこから等回転比走行でペットボトル間をすりぬけ、ライン復帰する。

ポイント

ドリフトターンはラインがないため、車輪の回転数のみで現在位置を判断していきます。そのため、等回転比走行と端点検知が有効になります。

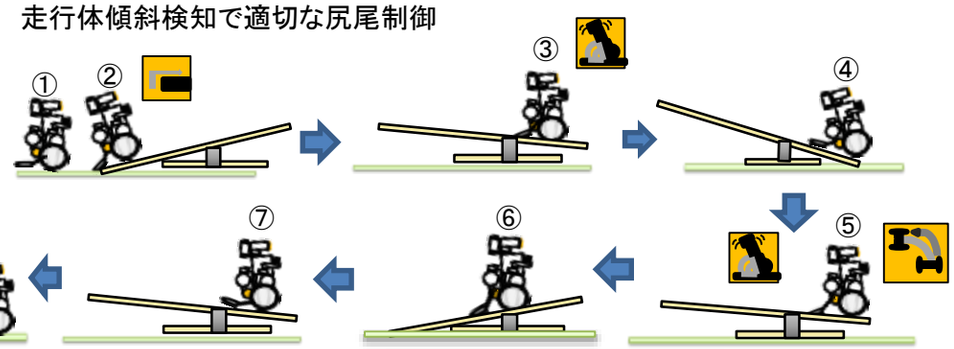




シーソーダブル

走行体傾斜検知で適切な尻尾制御

≪概要≫転落しないように、ライントレースでシーソーを登り、シーソーが傾きだしたのを傾斜検知で認識し、尻尾を制御する。降りるときは倒立走行で一気に駆け下りる。



「段差登り」は要素技術p.4を参照してください。