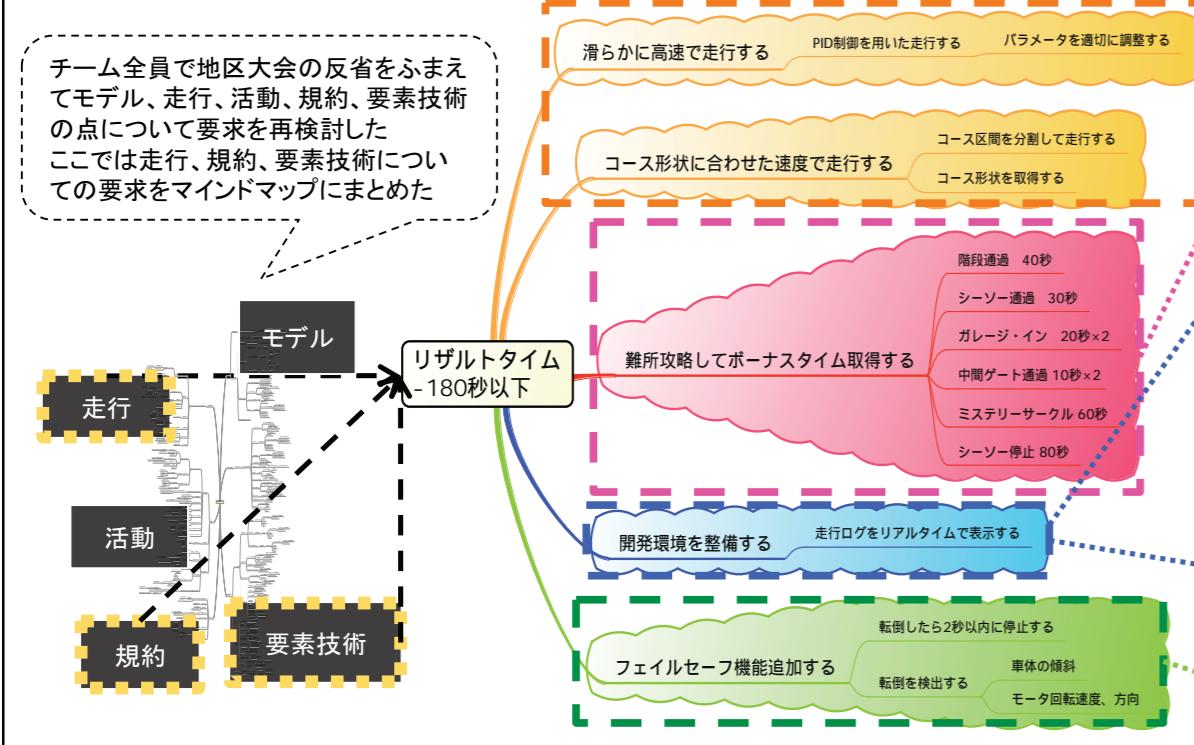


1.大会目標

走行:全ての難所を攻略してリザルトタイム-180秒
モデル:モデル評価A-以上の評価で上位入賞

全国制覇

2.マインドマップ



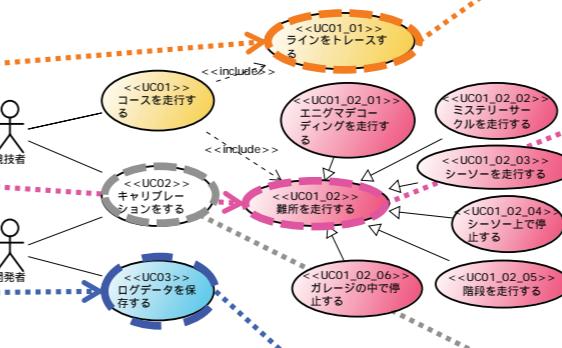
3.課題

マインドマップより課題を抽出した。地区大会で実現できなかった課題を深く検討することにした

- ①滑らかにライントレースする
- ②コースを区間に分割して適切な速度で走行する
- ③全ての難所を走行する
- ④ログデータをリアルタイムで表示する開発環境を整備する

4.ユースケース図

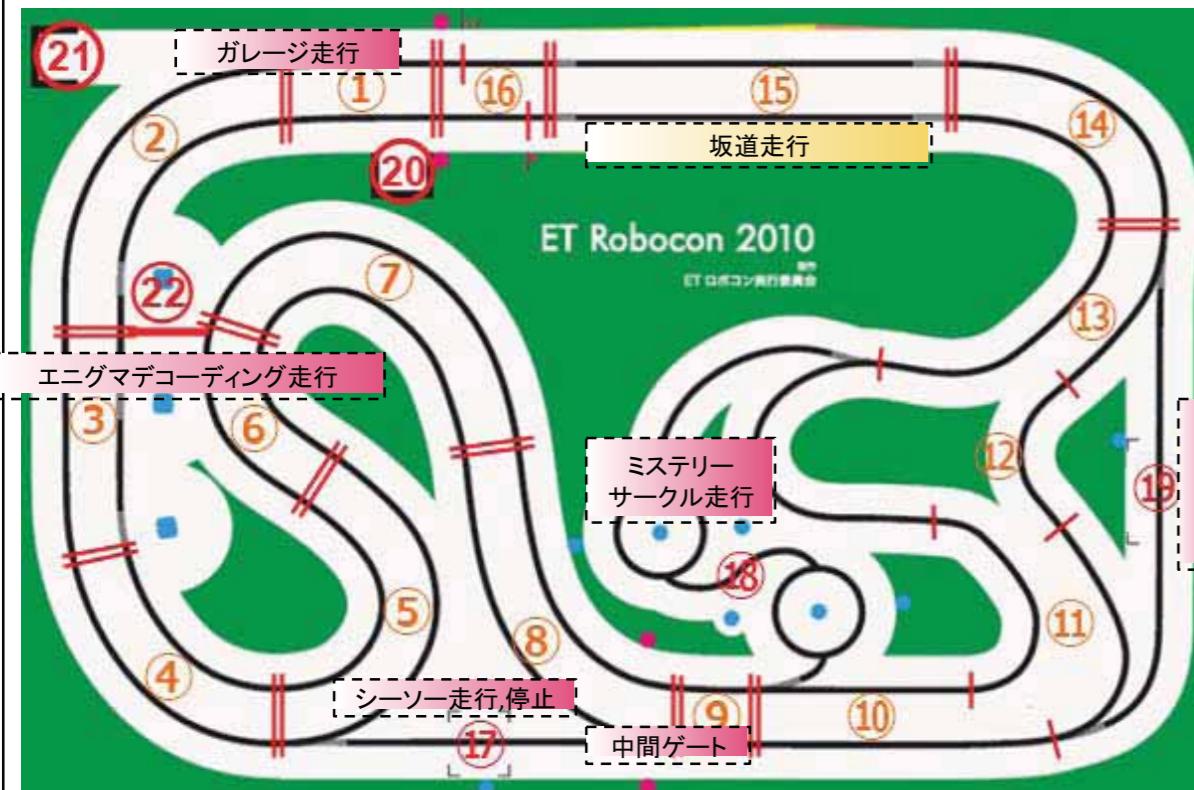
課題より機能を抽出した
ユースケース記述は「ラインをトレースする」、「難所を走行する」、「キャリブレーションをする」「ログデータを保存する」を例に示す



5.ユースケース記述

ユースケース	<<UC01_01>>ラインをトレースする
概要	ラインをエッジ走行する
事前条件	キャリブレーション完了する
事後条件	なし
基本フロー	①コース形状(直線、カーブなど)にあわせて加速、減速し、ラインエッジをトレースする
ユースケース	<<UC01_02>>難所を走行する
概要	難所を通過しボーナスタイムを獲得する
事前条件	難所区間入口まで走行している
事後条件	難所区間出口まで走行する
基本フロー	①難所区間中のサブ区間を走行する ②サブ区間を走行終了したか判定する ③難所区間が終了するまで①と②を繰り返す
ユースケース	<<UC02>>キャリブレーションをする
概要	滑らかに安定して走行するために走行環境を整える
アクター	競技者、開発者
事前条件	ロボットを測定場所に固定する
事後条件	測定情報が設定される
基本フロー	①ボタンを押してキャリブレーション開始する ②1秒間センサ値を計測し続ける ③平均値を用いて、しきい値またはオフセット値を設定する
ユースケース	<<UC03>>ログデータを保存する
概要	走行体から得られるデータを数値化して保存する
アクター	開発者
事前条件	大会本番ではないこと、PC側プログラムが起動していること
事後条件	データがPCに送られる、データを保存する
基本フロー	①ログデータを保存するPC側でログ保存プログラムを起動する ②走行体側でBluetoothの設定を行ってから、走行開始する

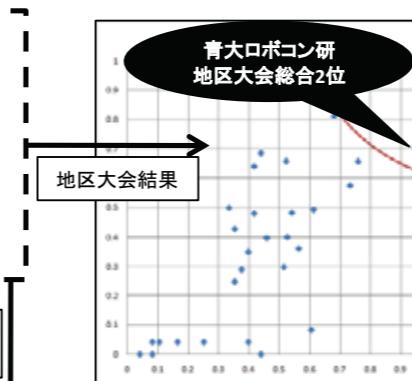
7.コース戦略、区間について



地区大会では安全策をとり、リスクの高い難所は走行せず、難所はガレージだけ挑戦した。チャンピオンシップでは守りに入らず全難所に強気で挑戦する!!

挑戦する難所&予想リザルトタイム算出

トラック	コース	ボーナス	走行タイム	リザルト
インコース	中間	10秒	45秒	-45秒
	ミステリー	60秒		
	ガレージ	20秒		
	シーソー	30秒	45秒	-135秒
アウトコース	中間	10秒	45秒	-135秒
	階段	40秒		
	ガレージ	20秒		
	シーソー停止	80秒		
	エニグマデコーディング			リザルトタイム -180秒以下実現



コースを、ライン形状、難所への分岐・合流個所から22区間に分割し、区間形状に合わせた適切な方法・速度で走行する。走行距離が区間距離に達したら、走行区間変更と走行方法切替を行う。走行方法切替については、P2のシーケンス図にて!!
難所区間番号の()内はサブ区間を示す

詳しくはこれら

区間形状	区間番号	走行
直線	1,3,9,10,16	ライントレース
左カーブ	2,4,5,8,11,13,14	
右カーブ	6,7,12	各難所走行
坂道	15	
シーソー	17(A,B,C,D,E)	各難所走行
ミステリーサークル	18(A,B,C,D)	
階段	19(A,B,C,D)	各難所走行
ガレージ・イン(イン)	20(A,B,C)	
ガレージ・イン(アウト)	21(A,B,C)	各難所走行
エニグマデコーディング	22(A,B,C)	

戦略を実現するための構造、走行の流れ、キャリブレーションの流れについては
P2へ

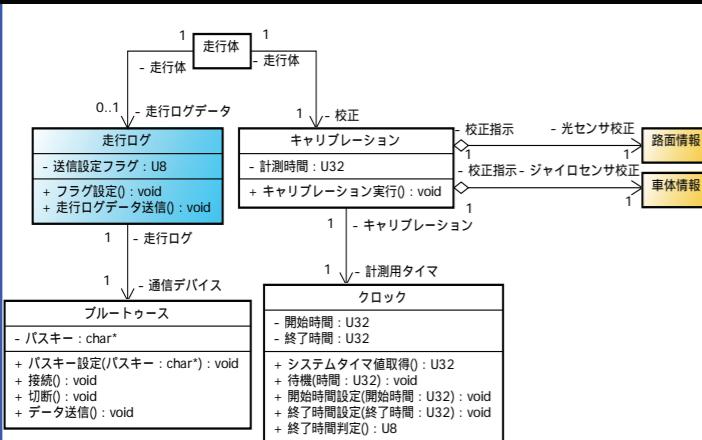
走行タイムイン、アウト合計で90秒。ボーナスタイム合計270秒。リザルトタイム-180秒実現。詳しい難所の走行については

P3へ

要素技術についての検証結果については
P4へ

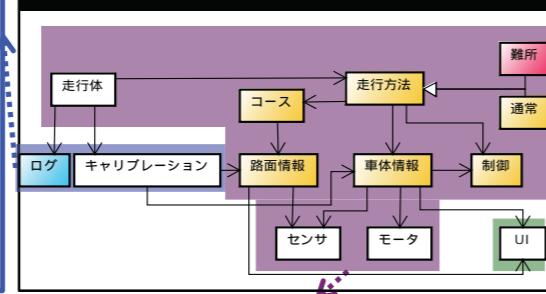
難所を確実に走れるという証拠をリアルタイムログで紹介。リアルタイムの結果をもとに思考、改善をした。検証結果については
P5へ

ログ、キャリブレーションクラス

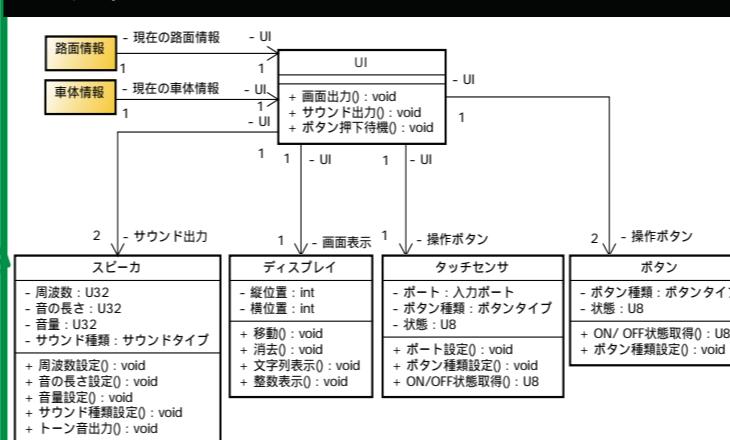


要求、機能を整理し、それらを実現するクラスを抽出し、関連をつけた

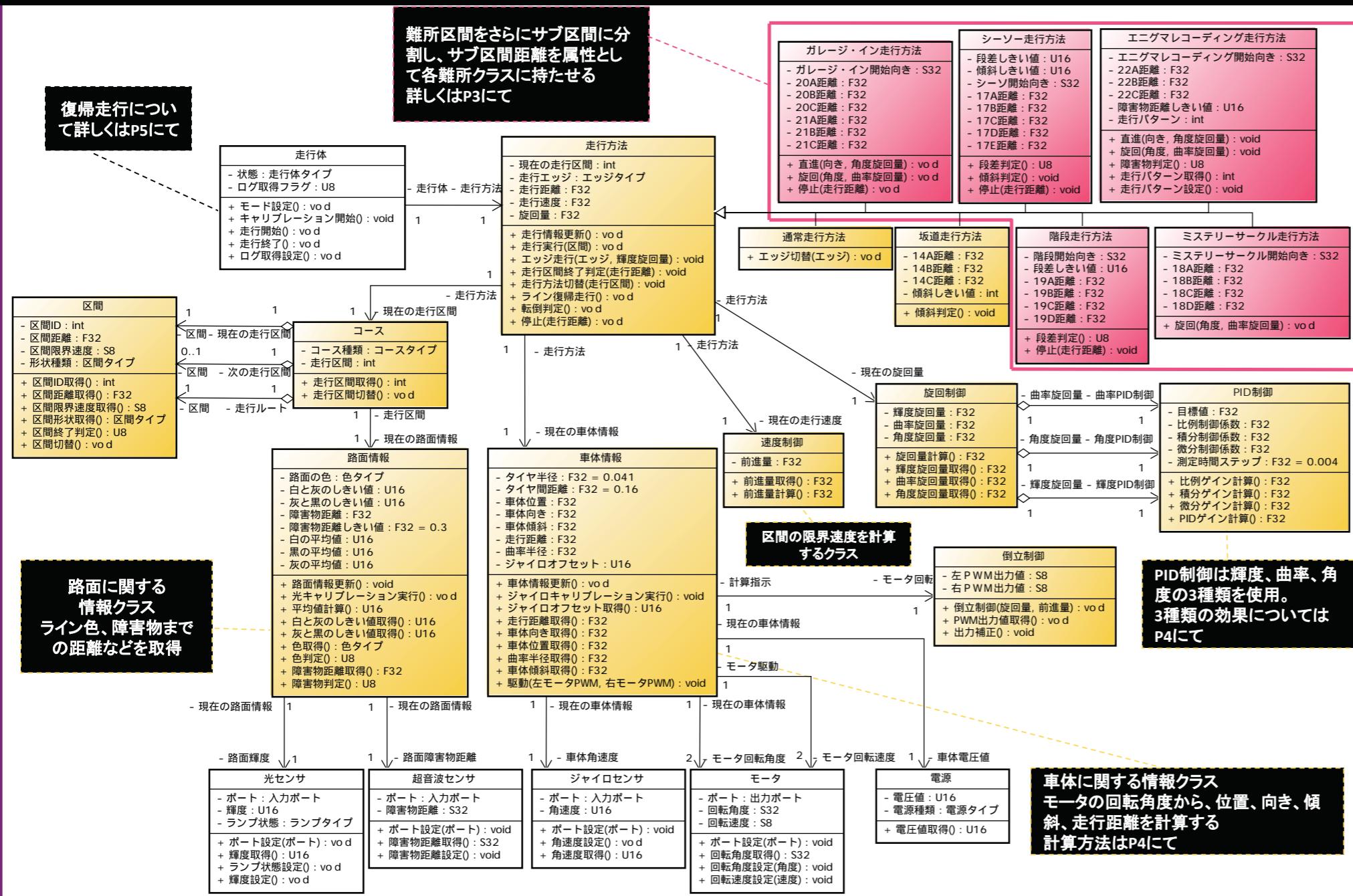
基本構造



UIクラス



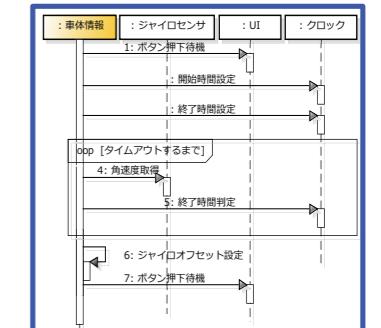
走行関連クラス



キャリブレーションの流れ

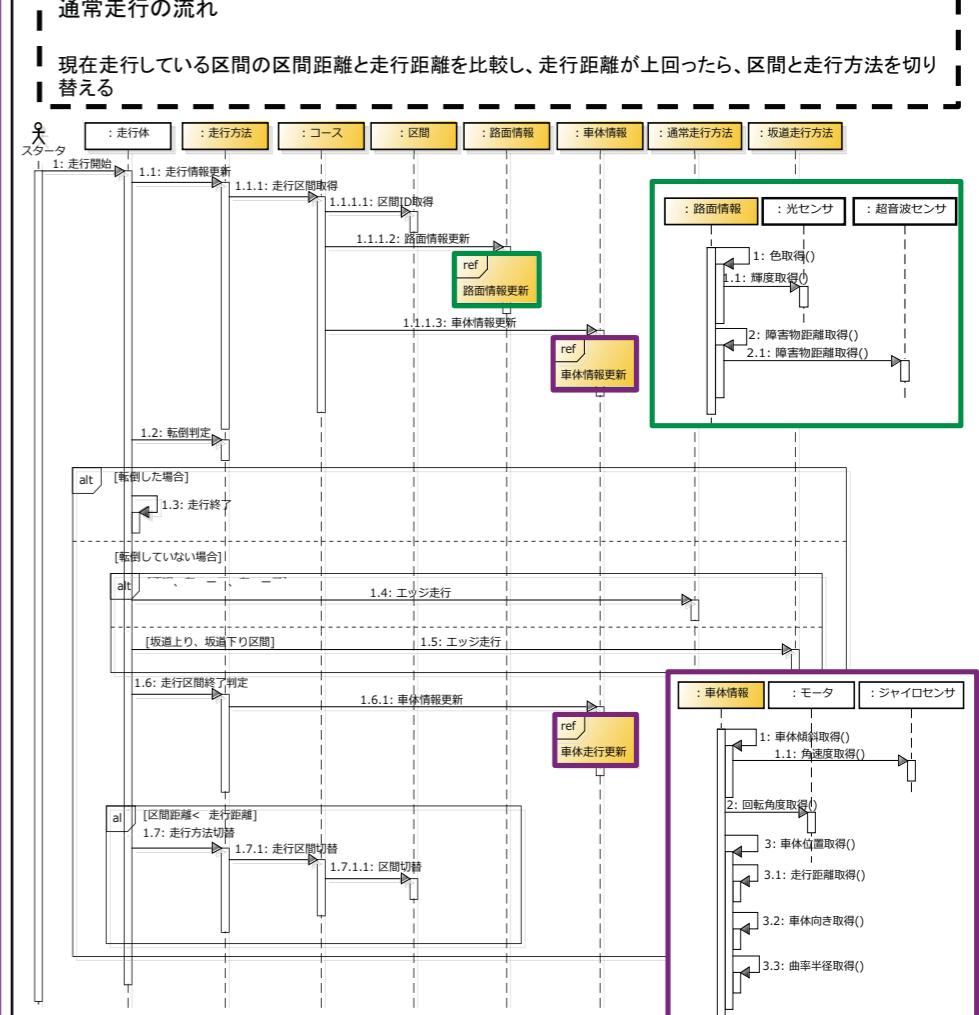
キャリブレーション

ジャイロセンサのオフセット値、白/灰/黒の色を判定するためのしきい値を求める
どちらも、1秒間測定した平均値をもとにして計算する



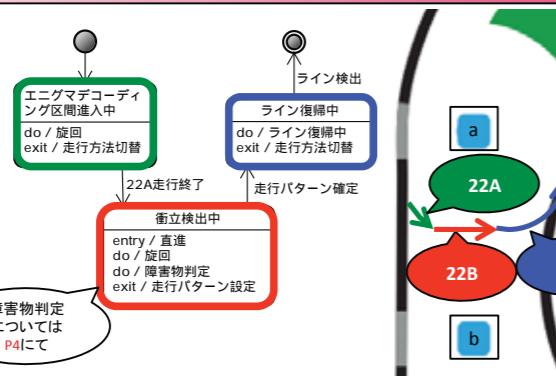
走行の流れ(通常走行)

通常走行の流れ



エニグマデコーディング走行(区間22)

- ①区間2の走行を終了したらエニグマデコーディング区間へ進入する
- ②衝立a,bの中間まで移動して、旋回しながら衝立の有無を判定する
- ③ミステリーサークルの通過ルートを決定し、ライン復帰状態になる
- ④区間7へ進入する

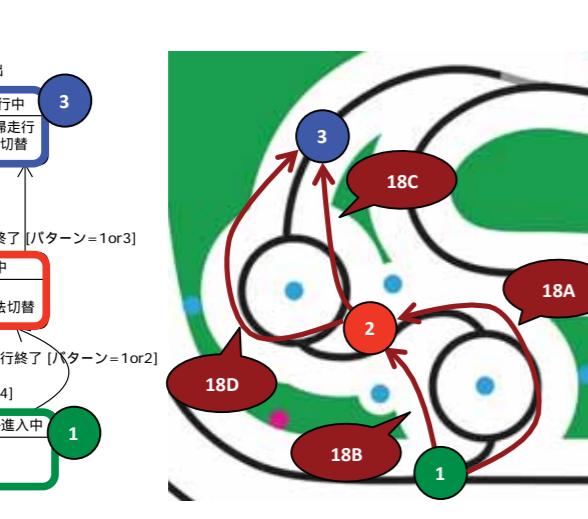


難所の振る舞いについて

難所区間をさらにサブ区間に分割した(区間22は、22A、22B、22Cの3つのサブ区間に分割)
サブ区間の切替は走行距離や段差、傾斜判定で行う
難所の判定の説明についてはP4にて
難所の検証結果についてはP5にて

ミステリーサークル走行(区間18)

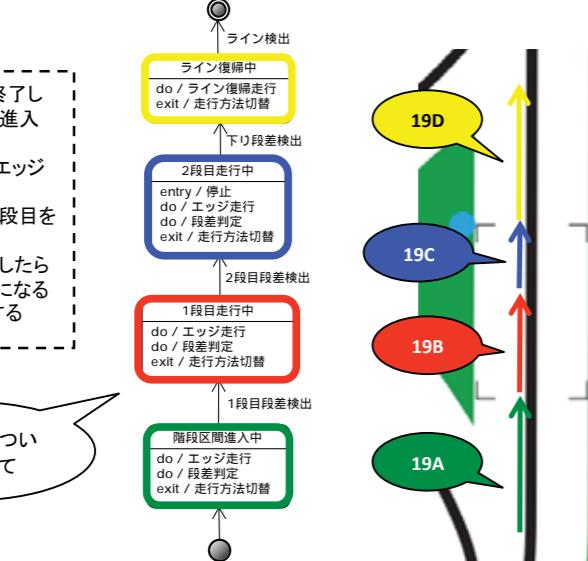
- ①区間9の走行終了したらミステリーサークル区間へ进入する
- ②基本的にラインを無視して旋回と直進で走行する
- ③地点1でミステリーサークルに入る向きを決め地点2へ向かう
- ④地点2では後半のコースの向きを決め、地点3でライン復帰状態になる
- ⑤区間12へ进入する



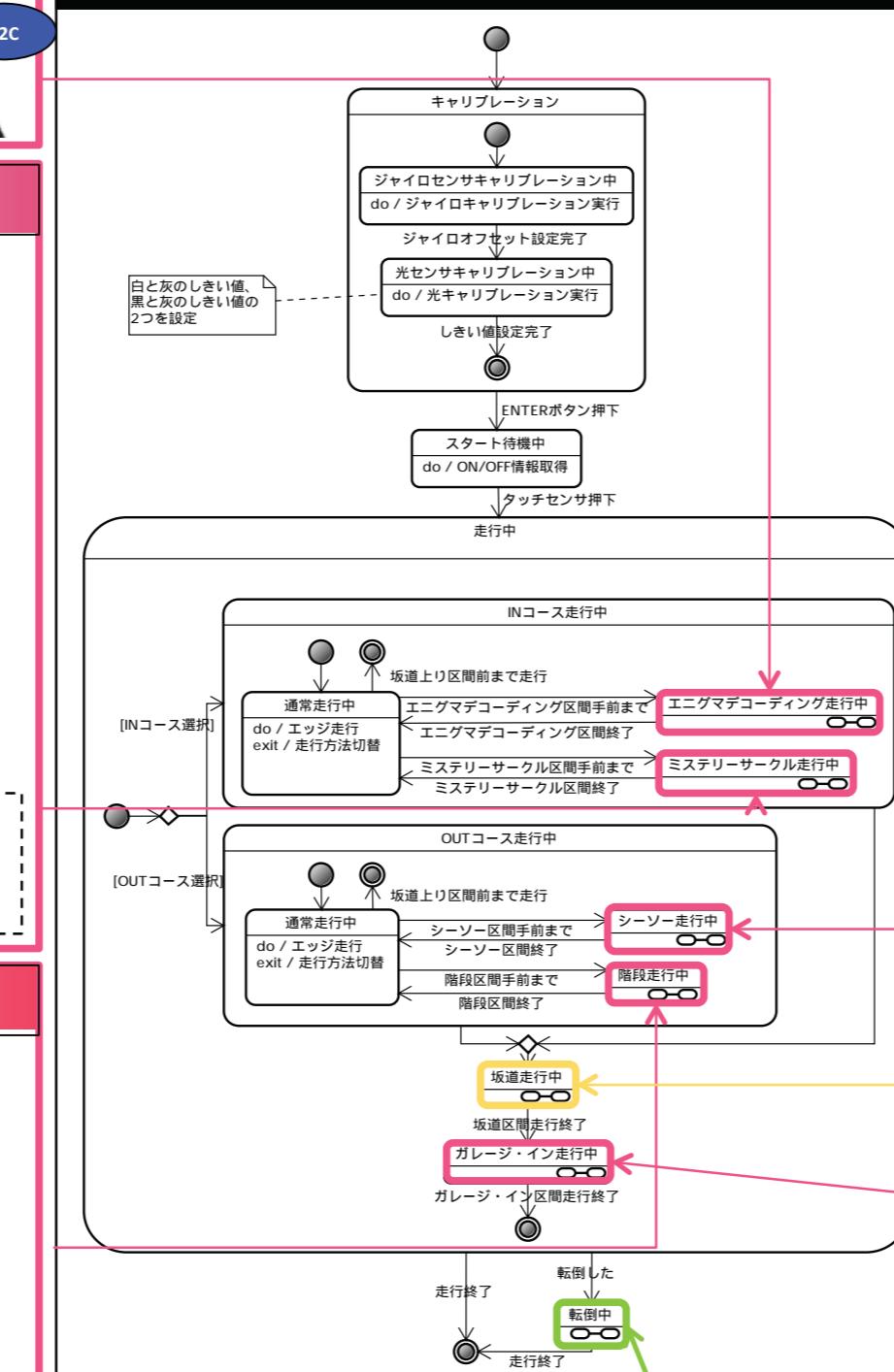
階段走行(区間19)

- ①区間10の走行終了したら階段区間へ进入する
- ②階段のラインをエッジ走行する
- ③段差判定で1、2段目を検出終了する
- ④下り段差を検出したらライン復帰状態になる
- ⑤区間14に进入する

段差判定についてはP4にて



全体の流れ



フェイルセーフ

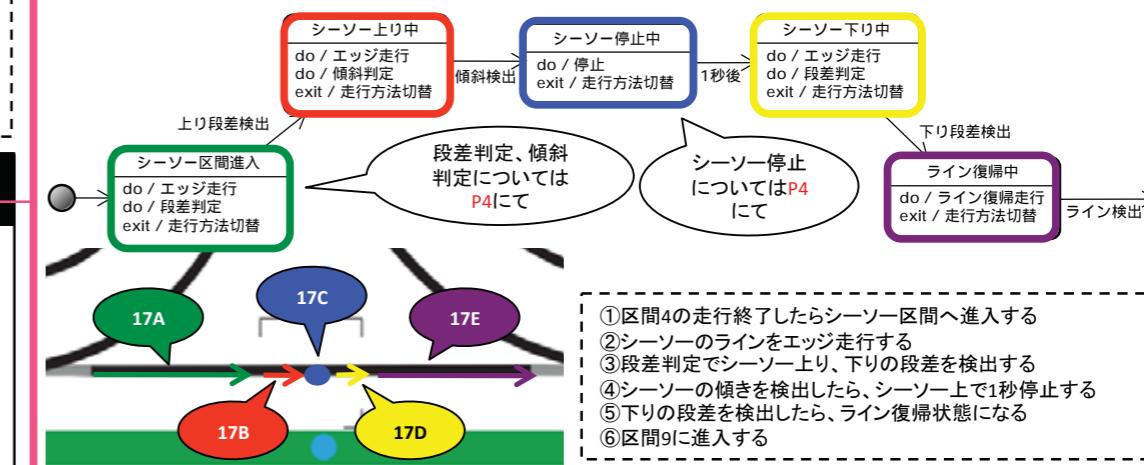
走行中に転倒と判定したらバランスをとって停止する。停止後、左右のモータ回転角度が上がり又は下降し続いている場合、実際に転倒していると見なし走行終了する。

走行終了 [回転角度上昇中又は下降中]

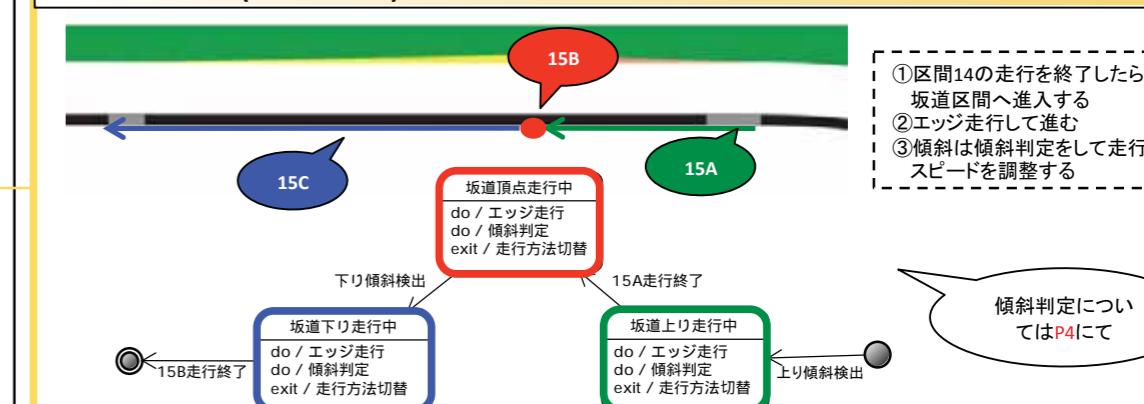
転倒確認中
do / 停止

1秒停止

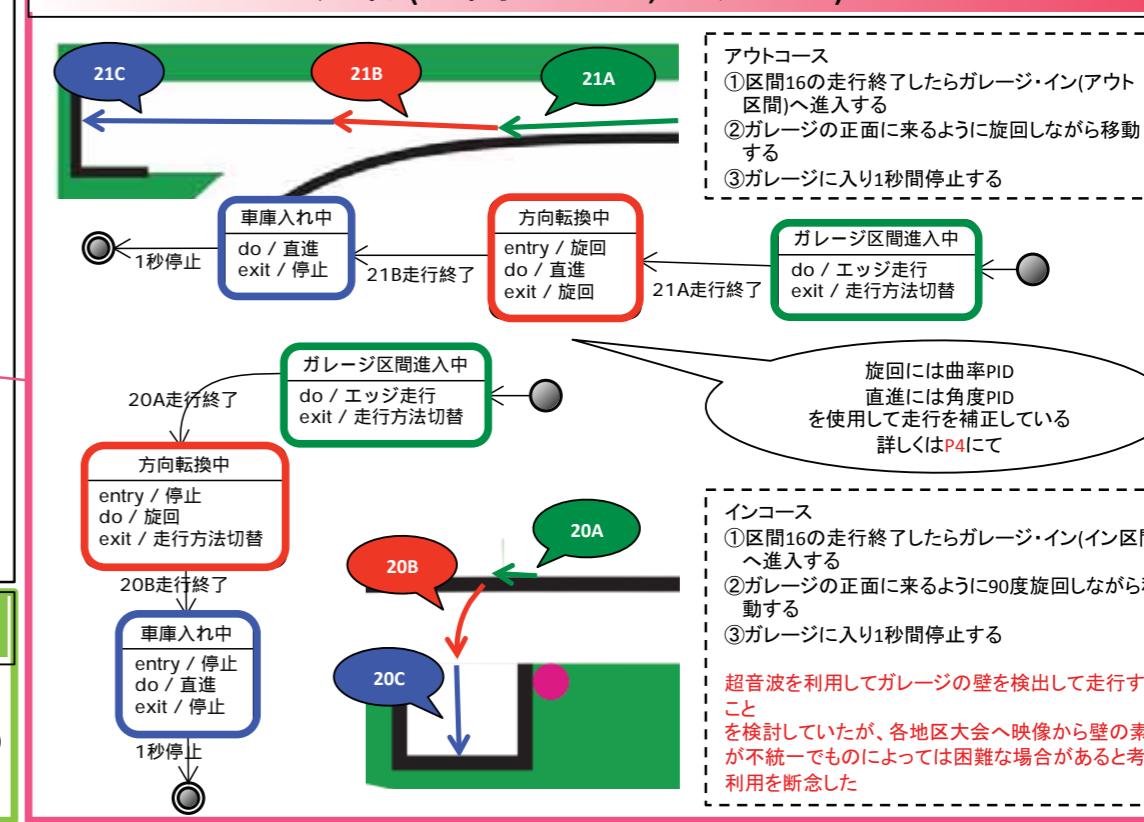
シーソー走行(区間17)



坂道走行(区間15)



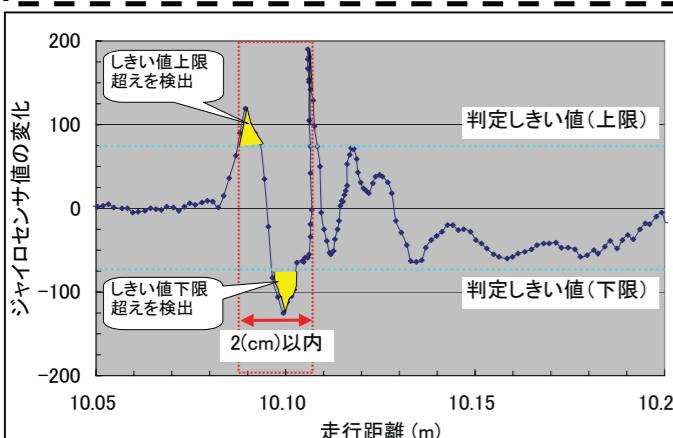
ガレージ・イン走行(区間イン:20,アウト:21)



段差判定

階段区間およびシーソー区間では、段差判定を以下の様に行い、満たす場合に段差を検出したと見なす

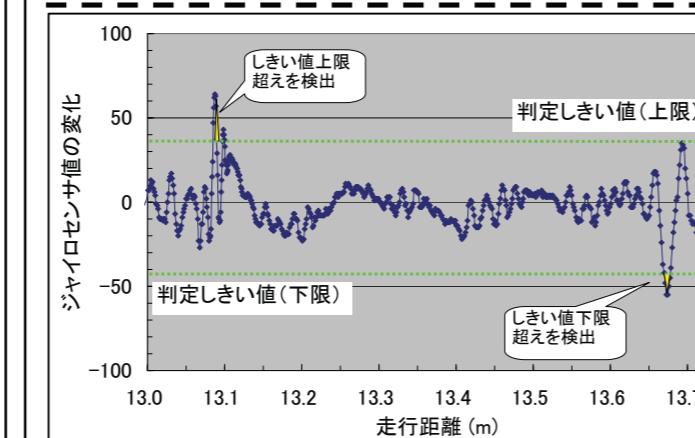
1. ジャイロセンサの角速度を取得する
(角速度からオフセットを差し引いた値)
2. 角速度が判定しきい値(上限)を超える
3. 角速度が判定しきい値(下限)を超える
4. 上記2,3を走行距離2(cm)以内に同時に満たす



傾斜判定

坂道区間およびシーソー区間(走行体がシーソー上にいる区間)では、傾斜判定を以下の様に行い、満たす場合に傾斜を検出したと見なす

1. ジャイロセンサの角速度を取得する
(角速度からオフセットを差し引いた値)
2. 角速度が判定しきい値(上限)を超える
3. 角速度が判定しきい値(下限)を超える
4. 上記2,3のどちらか一方を満たす



PID制御

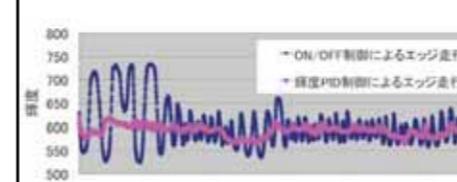
走行体の旋回量(turn)をPID制御で以下のように求める。
 e_n : 偏差=(測定値-目標値)
 K_p, K_i, K_d : 比例、積分、微分パラメータ
 Δt : 時間ステップ(4[ms])
 $\text{旋回量} = K_p e_n + K_i \Delta t \sum_n e_n + K_d \frac{(e_n - e_{n-1})}{\Delta t}$
エッジ走行・直進・旋回の各走行の旋回量の目標値と測定値から計算する

輝度PID制御(エッジ走行)

輝度を測定し、ライントレースする
•目標値:エッジ走行の輝度しきい値
•測定値:現在の輝度

テストコース走行結果

ON/OFF制御と輝度PID制御を比較し、輝度PID制御によるエッジ走行で、ラインからのぶれが少ないことを確認した。

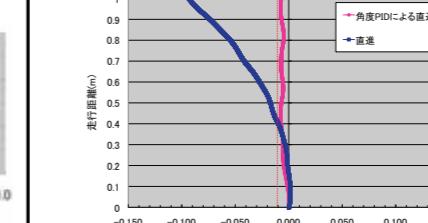


角度PID制御(直進)

走行体を指定方向(0~360°)へ直進させる
•目標値:指定した走行体の進行方向
•測定値:現在の走行体の方向

1m直進走行結果

走行体を0°方向に1m直進する動作を、角度PID制御を使用／不使用で比較した。その結果、角度PID制御による直進で左右の誤差を1(cm)以下に出来ることを確認した。

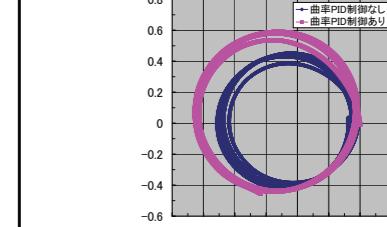


曲率PID制御(旋回)

走行体を指定した曲率半径で旋回させる
•目標値:指定した旋回半径
•測定値:現在の曲率半径

半径50cm走行結果

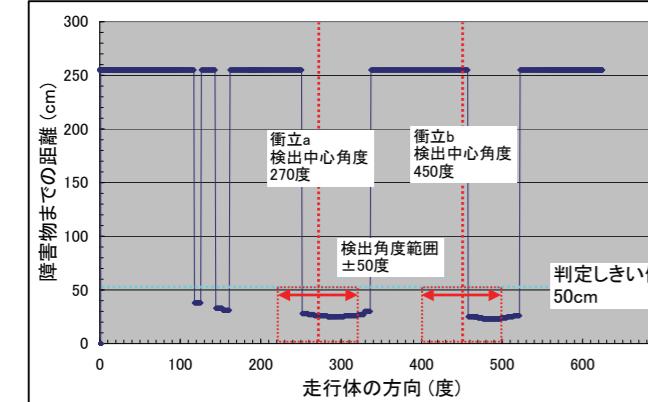
半径50cmの旋回動作を曲率PID制御で使用／不使用で比較した。その結果曲率PID制御による旋回が、より正確でぶれの少ない円軌道で旋回できることを確認した。



障害物判定

エニグマデコーディング区間では障害物判定を以下のように行い、満たす場合に衝立を検出したと見なす

1. 衝立a,bの中間で旋回しながら超音波センサを用いて障害物距離を測定する
2. 以下の条件で衝立の有無を判定する
 - a) 衝立a:方向270°±50°の範囲で50cm以内の測定値がある
 - b) 衝立b:方向450°±50°の範囲で50cm以内の測定値がある



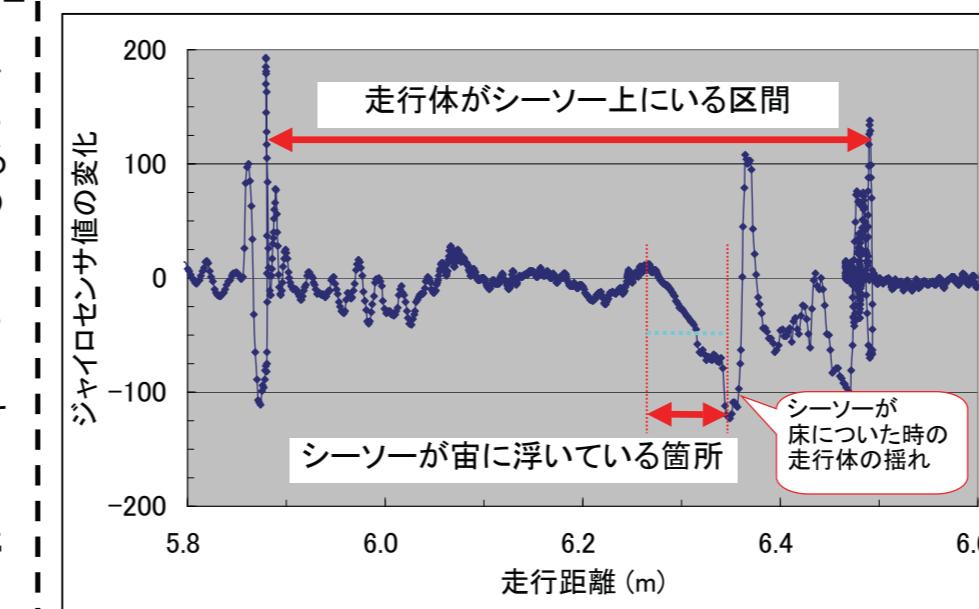
シーソー停止

走行ログを検証した結果、シーソー出入口の段差とシーソーが床についた時の揺れを検出できることがわかった。床につく直前の角速度変化に着目し、以下の条件でシーソー停止の動作を行う

条件

1. 段差判定により走行体がシーソー上にいることを満たしている
2. 走行体がシーソーの支点(段差から28cm)より先にいる
3. 角速度がしきい値下限を超える
4. 1,2,3を同時に満たす

以上の条件を満たす場合に、走行体を1秒間停止する



並行性設計

地区大会では、モータ制御、光センサ、超音波センサの3タスクを並行動作したが、以下の理由で並行性設計を行わないことにした。

1. 超音波センサの検出精度の再検証を行った結果、床面にテストコースを敷いた場合やガレージ壁の素材によって、検出可能な範囲が極端に変化することが判った。衝立検出以外では使用しないことにした。
2. まいまい式輝度測定を使用しないエッジ走行の方が走行タイムが速い結果が出たため、チャンピオンシップ大会では通常の輝度測定でエッジ走行することにした。

車体情報計算式

車体情報クラスで使用する計算式一覧

$$L_n = \frac{L_n^l + L_n^r}{2}, L_n^l = \frac{2\pi r}{360} \phi_n^l, L_n^r = \frac{2\pi r}{360} \phi_n^r$$

$$\theta_n = \frac{r}{d} (\phi_n^r - \phi_n^l)$$

$$R_n = \frac{360}{2\pi} \cdot \frac{dL}{d\theta} = \frac{360}{2\pi} \cdot \frac{L_n - L_{n-1}}{\theta_n - \theta_{n-1}}$$

$$x_n = x_{n-1} + (L_n - L_{n-1}) \sin \theta_{n-1}$$

$$y_n = y_{n-1} + (L_n - L_{n-1}) \cos \theta_{n-1}$$

走行距離[m]

左モータ走行距離[m]

右モータ走行距離[m]

左モータ回転角度[度]

右モータ回転角度[度]

タイヤ半径 (0.041[m])

タイヤ間距離(0.16[m])

走行体の向き[度]

曲率半径[m]

走行体の位置[m]

リアルタイムログ

LabVIEWを使い、ログデータを収集・表示するシステムを作成した。昨年までは、ログ収集およびグラフ化作業と、撮影したビデオとグラフを見ながら検討する作業を、20~30分程度かけて行っていたが、このシステムを使用して作業・検討時間を、5分程度まで短縮することに成功した。

