ETロボコン2012

地区：東北　　地域：宮城県仙台市

チームＩＤ： 10　チーム名：良いこんぶ　　所属：　仙台高等専門学校　広瀬キャンパス

## ☆チーム紹介

高専の３年生から７年生（専攻科２年生）７名で構成幅広い年代のチームです.所属学科もバラバラで,異なるバックグランドを持ったメンバーがお互いに補い合いながら取り組んできました.昨年度に続き今年２度目のチャレンジです.昨年度の経験を活かし、モデル,走行共にパワーアップしました. チーム名の由来説明？こんなチームで大会に望みます。

## ☆組込み，そしてモデリングの未来へ一言

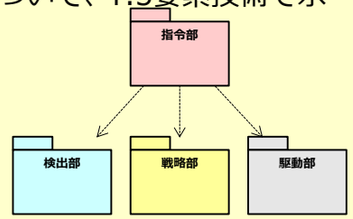
モデリングの根底に流れる重要な考え方のひとつは「抽象化思考」です。これは新しい技術がどんどん登場しても廃れることなく常に通用する技術です。組込みシステムが肥大化する昨今、この技術を手に入れることは、当然の流れと言えます。若手社会人や学生が参加するこのコンテストを通してこの武器が広く日本に普及すれば、組み込み業界だけでなく、すべてのエンジニアがハッピーになれる未来が待っているはずです。学生らしく・・・

## ☆コンテストにかける意気込み，アピール

昨年果たせなかった悲願の全国大会出場を果たします！！
高専で15歳から受けた教育をベースにした高専生の実力をお見せします！

## ☆モデルの概要

UMLとSysMLを用いてモデルを構成しました.大会における目標に対して要求図を用いて要素を抽出しました。要素の一つである区間について詳細な分析をP.2で行い,下図のパッケージ構成が導き出されました。パッケージ分けの詳細はP2.構造を参照.要求とパッケージ構成に基づいて構造を分析しました.区間の切り替えと駆動の振る舞いについて並行性設計を踏まえながら分析を行うことで実現可能性を検証しました。詳細はP.３振る舞い参照。難所をどのように攻略すべきかを4P.に示し、そこで使われている主な要素技術について、P.5要素技術で示しました.




## ☆設計思想

パッケージ分けを開発の初期に行い、責務が分散しないように意識することにより、モデルに一貫性を持たせました．双方向の関連を禁止しました.区間の切り替え通知はデザインパターンであるObserverパターンを拡張した構成を用いることで双方向の関連を避けました。区間ごとにチームで分担して開発することにより開発スピードを上げ、結合は区間をつなげることのみで行えるので容易になりました。

## ☆モデルのここに注目！

ETロボコンはコースを分割した区間の連続．その区間に応じたパラメータと区間切替条件を設計すれば完走することができる．このコンセプトが読み取れる構造,振る舞いになっています.そして、責務が明確に別れた単方向・疎結合な構造にご注目ください。

## ☆追加課題への取り組み

並行性設計・要求モデルについて取り組みました。

・並行性設計について

設計指針
走行体のバランス動作などのモータの駆動が一番優先するべきことです.それに対して,区間の切替ははるかに遅い周期でも十分に性能は得られると考えました.その根拠の詳細はP3.振る舞い参照.よって駆動関連を一番高い優先度のタスクとし,それ以外は駆動よりも優先度が低いタスクとすることで,駆動が求められる周期で確実に行われるように設計した。
なお,タスクの構造を示すために２つのステレオタイプを用いた.採用するRTOSの提供する機能をnxtOSEK,ひとつひとつのタスクをTASKとしました.

・要求モデルについて

大会における目標についてSysMLの要求図を使って分析しました。そこから機能要件、非機能要件を洗い出して、構造、振る舞い、走行戦略で使われる技術要素を導きだしました。

# 要求分析

## チーム目標

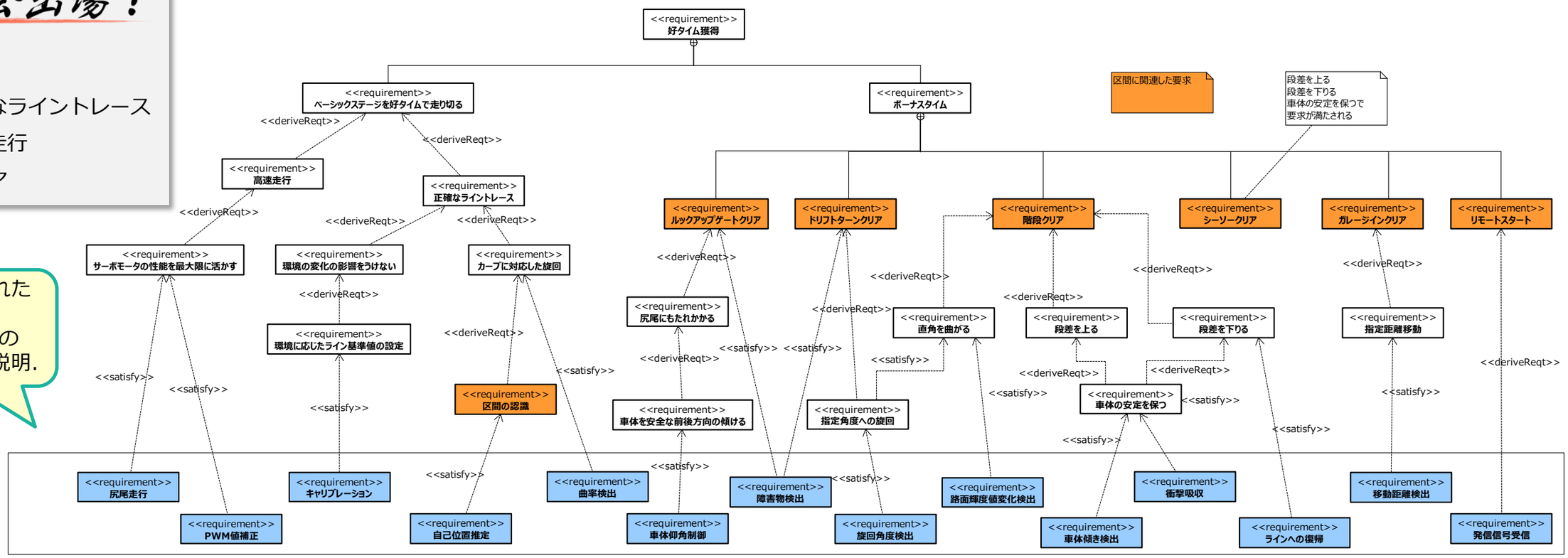
全国大会出場！

そのために･･･

- 高速かつ正確なライントレース
- 区間に応じた走行
- 全難所のクリア

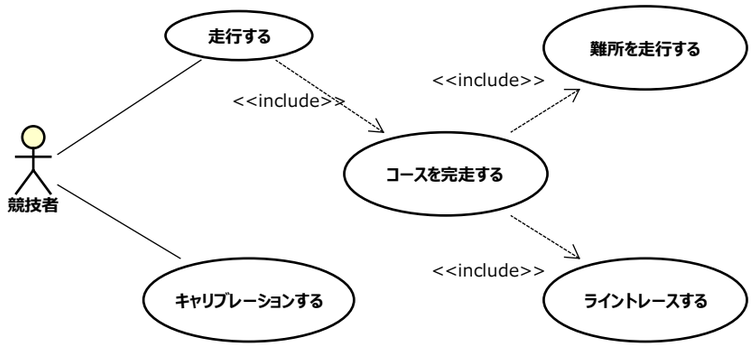
## 要求図(SysML)

目標を実現するためにシステムに何が要求されるのか，要求図を用いて分析．




## 機能要件

要求の中からシステムの機能とされるものを抽出した




**ユースケース記述**

| | |
|---|---|
| ユースケース名 | コースを完走する |
| 事前条件 | キャリブレーションが終わっている |
| 事後条件 | ガレージイン区間で完全停止状態になっている |
| 基本フロー | 1. 競技者は走行体をスタート位置に設置する.<br>2. 競技者は走行体に走行スタートを指示する.<br>3.走行体がコースを走行する.<br>4.走行体がガレージで停止する. |

## 非機能要件

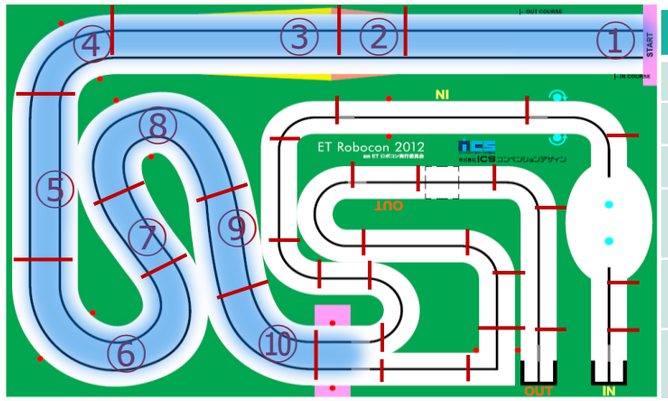
要求図から非機能要件として安全性や，性能面で重要と考えられることを抽出した．それが満たされない時の問題点と対処を分析．

| 非機能要件 | 問題点 | 対処 | 対応する要素技術 |
|---|---|---|---|
| 高速走行 | カーブを曲がり切れない | モータ性能を引き出す | PWM値補正 |
| 急カーブを曲がりきる | 曲がりきれずコースアウト | 区間を判別できる | 自己位置推定 |
| 車体を安定して前後方向に傾ける | しっぽの制御が車体の安定に悪影響を与える | 適切なしっぽ角度制御 | 車体仰角制御安定化 |

# 構造

## ドメイン分析(区間について)

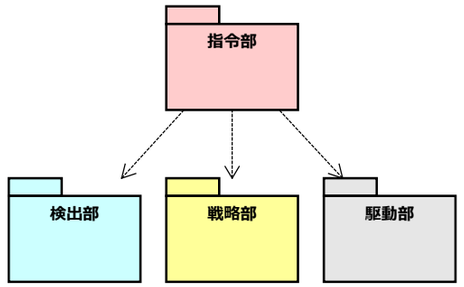
コースは，細かく分割された区間の連続によって構成されるものと分析した．
区間ごとに最適な前進量などの目標駆動パラメータと区間の切替条件がある．走行体は区間が切り替わるまで同一のパラメータを用いて走行する．区間切替に用いる情報をトリガーと称する．各区間クラスはトリガーの集合を区間切替条件として持つ．難所エリアには図示されているより詳細な区間が存在する．
→ボーナス・ステージ各難所での動作はP4.走行戦略を参照．




| トリガー | 詳細 |
|---|---|
| 発進信号受信 | Bluetooth信号受信機能から、発進信号の受信を検出 |
| 移動距離 | 自己位置推定機能から、指定した距離の移動を検出 |
| 衝撃検出 | ジャイロセンサから、段差にぶつかったことを検出 |
| 障害物検出 | 超音波センサから、指定した距離内に障害物が存在することを検出 |
| 路面輝度値変化 | 光センサから、指定した輝度値 |
| 旋回角度 | 自己位置推定機能から、指定した角度だけの旋回を検出 |

## パッケージ図(SysML)

区間に応じた走行を実現するため，下図のパッケージ構成を考案した．
指令部を除く各パッケージは互いを知らず，与えられた責務を実行し続ける．
この構成により，開発者は区間の情報を設計することに専念でき，かつ，それらは他の要素に影響を及ぼさないのでチームでの開発が容易になった．
来年度以降についても，戦略部の区間情報のみを再構成する事で新規約に容易に対応することが可能である．




| | |
|---|---|
| 指令部 | 戦略部，検出部の情報をもとに区間切替を行う． |
| 検出部 | センサ類の入力から区間の切替を検出し，指令部に通知する． |
| 戦略部 | 区間走行に必要なパラメータや，次区間への切替条件などの区間情報を保持する． |
| 駆動部 | モータ制御に必要な演算処理を行い，車輪としっぽを駆動する．必要なパラメータは指令部より受け取る． |

## クラス図(走行関連のみ)

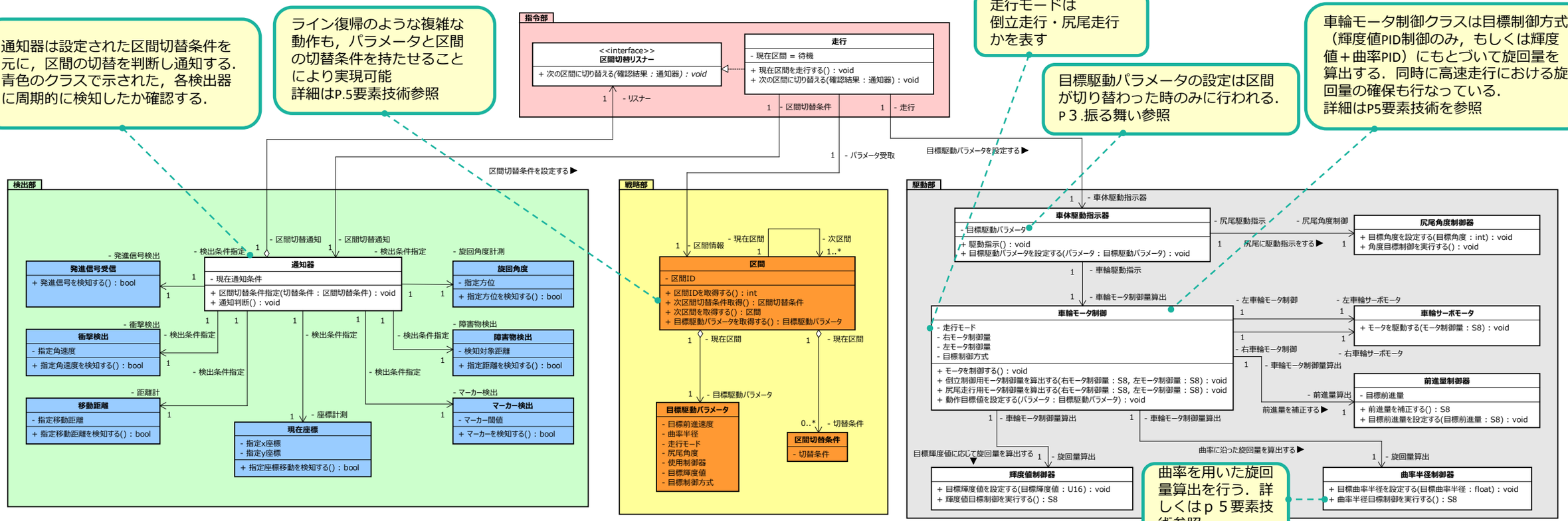
通知器は設定された区間切替条件を元に，区間の切替を判断し通知する．青色のクラスで示された，各検出器に周期的に検知したか確認する．

ライン復帰のような複雑な動作も，パラメータと区間の切替条件を持たせることにより実現可能
詳細はP.5要素技術参照

走行モードは
倒立走行・尻尾走行
かを表す

目標駆動パラメータの設定は区間が切り替わった時のみに行われる．
P３.振る舞い参照

車輪モータ制御クラスは目標制御方式（輝度値PID制御のみ，もしくは輝度値＋曲率PID）にもとづいて旋回量を算出する．同時に高速走行における旋回量の確保も行なっている．
詳細はP5要素技術を参照

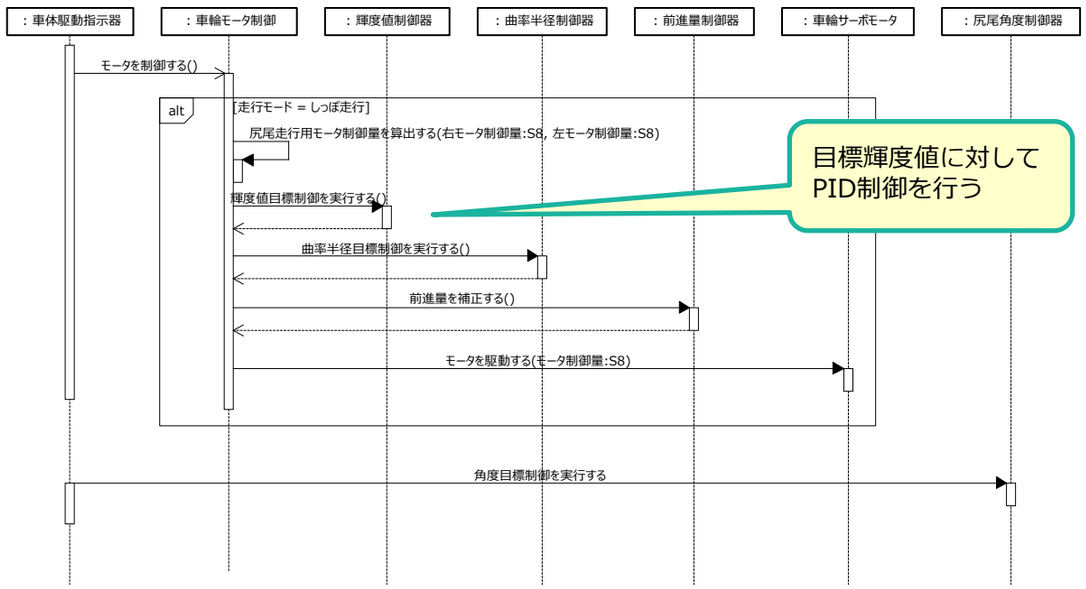



曲率を用いた旋回量算出を行う．詳しくはｐ５要素技術参照

# 振る舞い

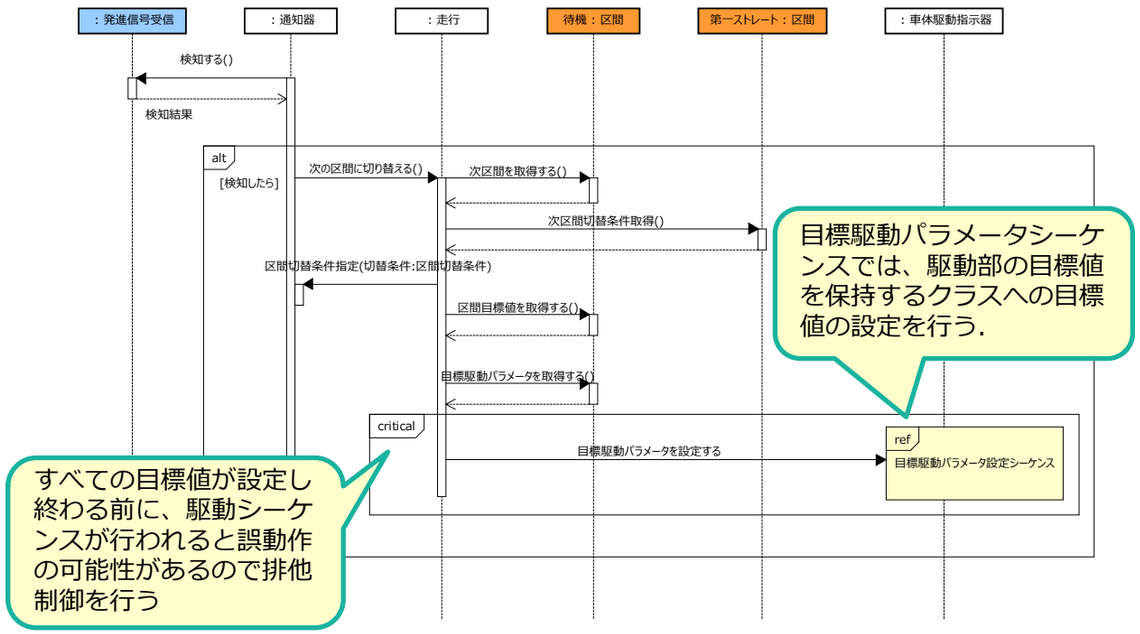
## 駆動シーケンス

走行中の振る舞い．
区間切替時に設定されるパラメータを用いて旋回量を算出し，モータを駆動する．
どの区間でも同様に振る舞い，走行することが可能．




## 区間切替シーケンス

走行区間を切り替え，駆動部に目標駆動パラメータを設定する振る舞い．




## 並行性設計

### 制約

APIの仕様上，倒立制御は4ms周期で実行する必要がある．

### 設計方針

①RTOSオーバヘッドを考慮し，タスクの数は最小限に抑える．
②最優先にすべきモータ駆動処理の動作周期が保障される．
③区間切替の検知に必要十分な周期を割り当てる．

| タスク名 | 優先度※ | 周期 [ms] | 理由 |
|---|---|---|---|
| 駆動タスク | 1 | 4 | 制約よりバランサーは4ms周期で実行される必要がある．加えて設計方針①と②より，それに関連するモータ駆動処理は同じタスクで処理すべきであると考えた． |
| 区間切替監視タスク | 2 | 10 | 区間切替は1cm以内の精度で行えば十分であると考えた． |

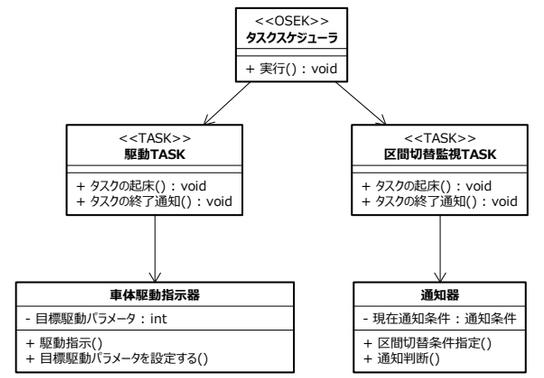
※値が若いほど優先度は高い

最高速度(60cm/s)で走行中に走行距離をトリガーとして区間を切り替える場合，最大で0.6cm移動する間での区間切替が可能なので，10msの周期は妥当であると判断した．
また，他のセンサをトリガーに区間切替を行う場合も十分な応答が得られた．

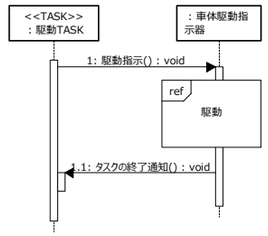
### タスクの構造と振る舞い

設計方針に基づいてタスクを設計した．

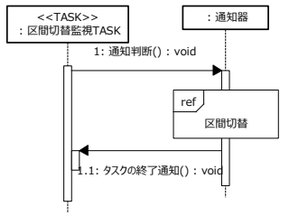
#### 構造




■駆動TASK

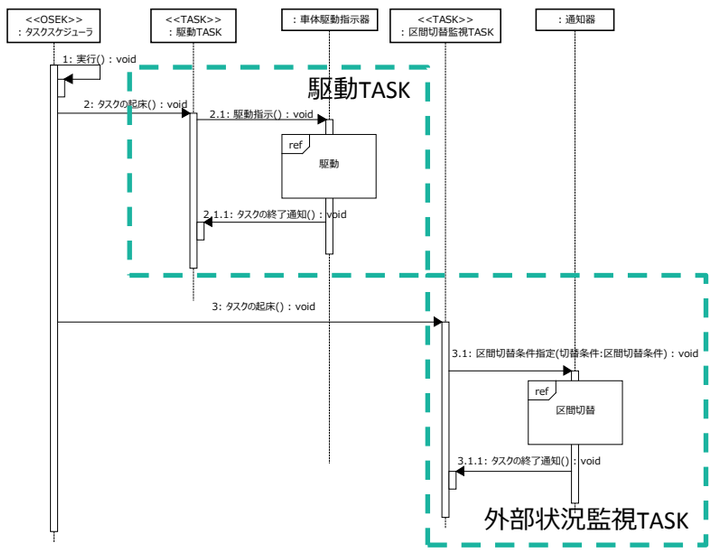



■区間切替監視TASK




#### 振る舞い

別タスクで起動する２つの振る舞いから
走行システムが構成される．



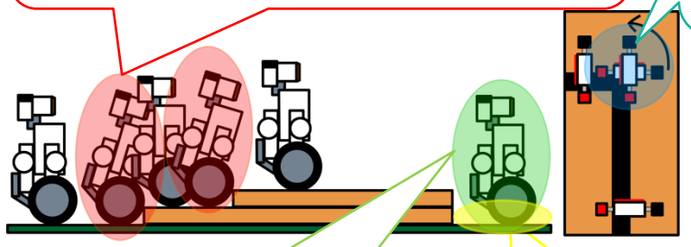
# 難所走行戦略

## 階段

階段突破のためには厚さ１ｃｍの段差を乗り越え,限られたスペースで直角に引かれたラインをトレースしなければならない.そこに潜む危険とその解決策を考え,それらを踏まえてステートチャート図を作成した.（他の難所についても同様の手順でステートマシン図を作成）

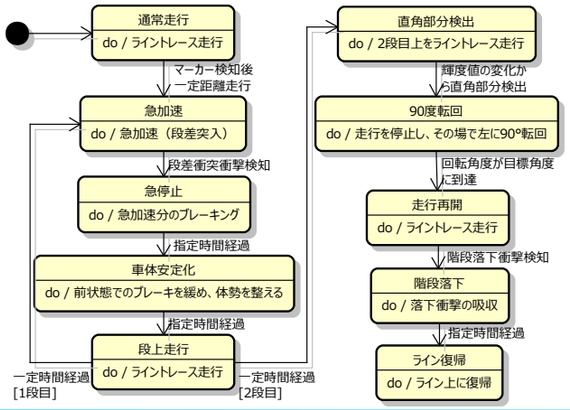
・段差進入時の速度不足
段差を上るためには加速により勢いをつける必要がある.そこで倒立制御APIで用いるジャイロセンサのオフセット値を調節し,走行体を強制的に前傾させることで短距離での急加速を実現.

・直角部分を曲がりきることが出来ない
直角部分では輝度値の変化を検知すると停止して転回する.この時,検知が遅れるとラインから外れた位置で転回するためライントレースを続行出来ない.そこで、輝度値変化が発生した瞬間の傾向を調べ,その傾向が見られた瞬間に停止することでラインを見失わない転回を実現.



・落下時に走行体が不安定になる
走行体が落下した際に走行体が後傾姿勢になる傾向がある。そこで、落下時の衝撃を検知した際に,ジャイロオフセットの値を調節し補正を行うことで倒立制御の安定化を実現.

・落下時に走行体がラインから外れている
ライン復帰動作（p.5 要素技術参照）によってラインへの復帰を実現.



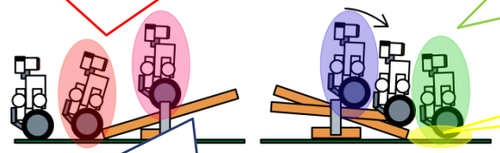

## シーソー（シングル）

シーソー突破のためには段差を乗り越え,傾斜を上り,シーソーの動きに対して

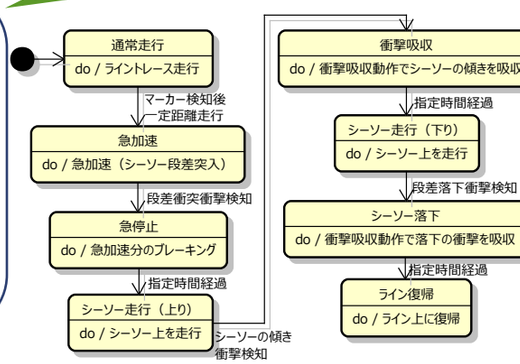
・段差進入時の速度不足
階段での動作と同様にして実現.

・落下時に走行体が不安定になる
階段での動作と同様にして実現.

・落下時に走行体がラインから外れている
階段での動作と同様にして実現.

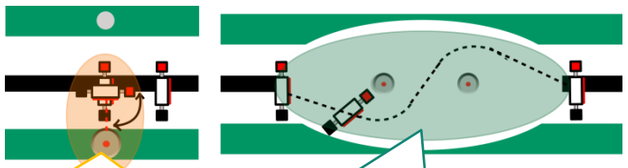


・シーソーの動きと角度によって走行が不安定になる
シーソー上走行中,シーソーの角度によって走行体が姿勢を維持することが難しくなる。また、シーソーが降下した際に



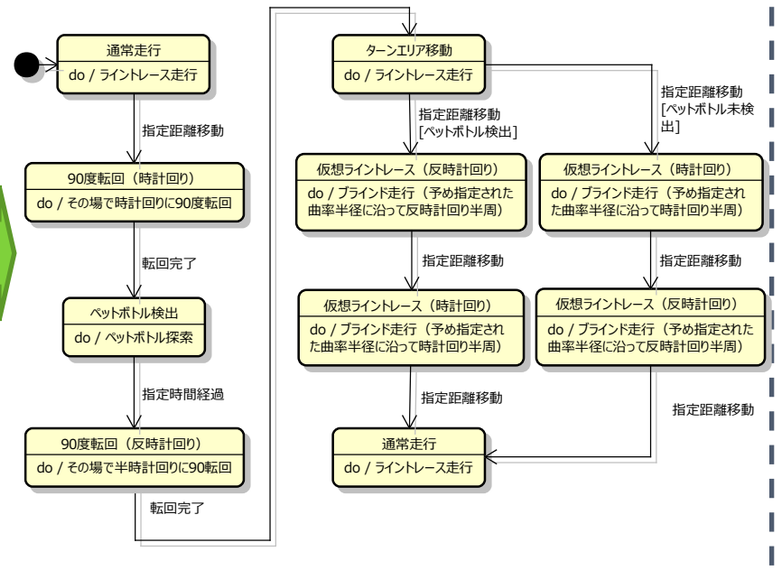

## ドリフトターン

ドリフトターン突破のためにはペットボトルの誤検知を防ぎ,ラインの無い区間を走行しなければならない.



・ペットボトルの誤検知
大会のコース上にはいくつかのオブジェが置いてあり,それらを誤って検知する可能性がある.そこでライン上かつペットボトルに最も近い位置で停止,転回し検知を行うことで誤検知防止を実現.

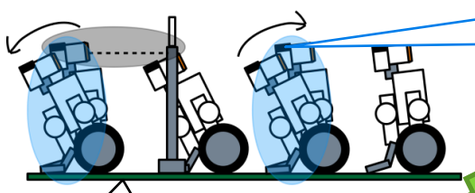
・ターンエリアでの走行失敗
ターンエリア上にはラインが無いため,仮想ラインを定義し,輝度値によるライントレースを行わず曲率半径PID制御（p.5 要素技術参照）のみを利用して走行することで,ラインの無い区間での走行を実現.



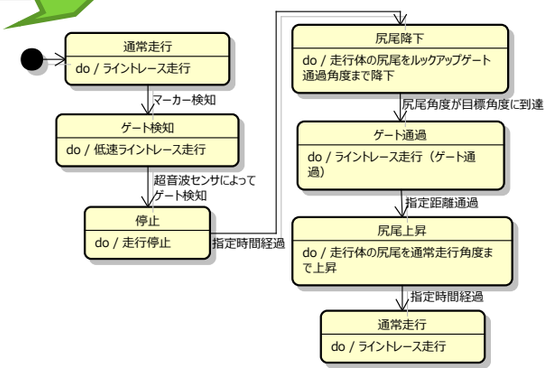

## ルックアップゲート

ルックアップゲート突破のためにはゲートを検知し,その下を通過出来る角度まで走行体を傾け,通過後に元の角度に戻らなければならない.

・目標尻尾角度への制御失敗
走行体仰角制御（p.5 要素技術参照）によって,安定した尻尾角度制御を実現.

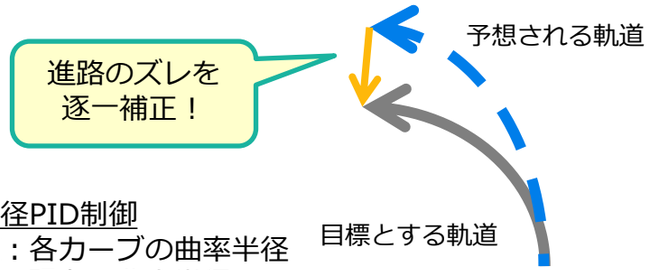


・ゲート検知の失敗
ゲートの検知には超音波センサを用いるが,値の取得は超音波センサAPIの仕様上50ms周期で行わなければならない.よってゲート検知をする際の速度が速すぎると検知をする前にゲートを衝突してしまうため,ゲート前の区間で速度を下げることで正確な検知を実現.



# 要素技術

## 曲率半径PID制御




曲率半径PID制御
目標値：各カーブの曲率半径
計測値：現在の曲率半径

各カーブ毎の曲率半径を目標値，自己位置推定機能によって算出される現在の曲率半径を計測値として，PID制御方式による目標制御を行い操作量を算出する．

→輝度値PID制御と組み合わせたハイブリッドPID制御を実装．

より柔軟なライントレースの実現．

## 自己位置推定

車輪回転角度から走行体の移動距離・移動方向・座標を算出する．推定より得られる情報から走行区間の推定を行うことで，各区間に応じた走行方法の変更を実現する．

移動距離

$$l_n = \frac{(l_L + l_r)}{2} \quad [cm]$$

移動方向

$$\theta_n = \frac{R_w}{L_w} \times (\Phi_R - \Phi_L) \quad [度]$$

曲率半径

$$R = \frac{180}{\pi} \times \frac{l_n - l_{n-1}}{\theta_n - \theta_{n-1}}$$

座標

$$\begin{cases} x_n = x_{n-1} + (l_n - l_{n-1}) \times \cos\theta_n \\ y_n = y_{n-1} + (l_n - l_{n-1}) \times \sin\theta_n \end{cases}$$

その他必要な定義等

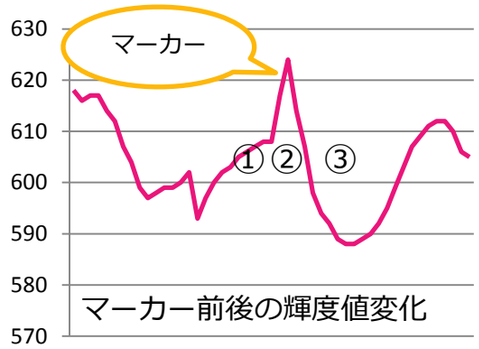
$$\begin{cases} l_L = \frac{\pi}{180}(\Phi_L \times R_w) \quad [cm] \\ l_R = \frac{\pi}{180}(\Phi_R \times R_w) \quad [cm] \end{cases}$$

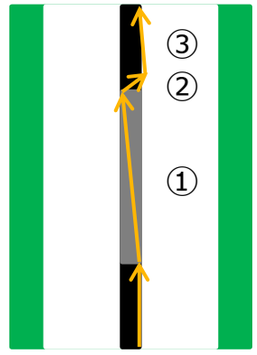
$$\begin{cases} \Phi_L = \text{左車輪モータ回転角度} \quad [度] \\ \Phi_R = \text{右車輪モータ回転角度} \quad [度] \end{cases}$$

$R_w$：車輪半径$[cm]$

$L_w$：二輪間の距離$[cm]$

## マーカー検知

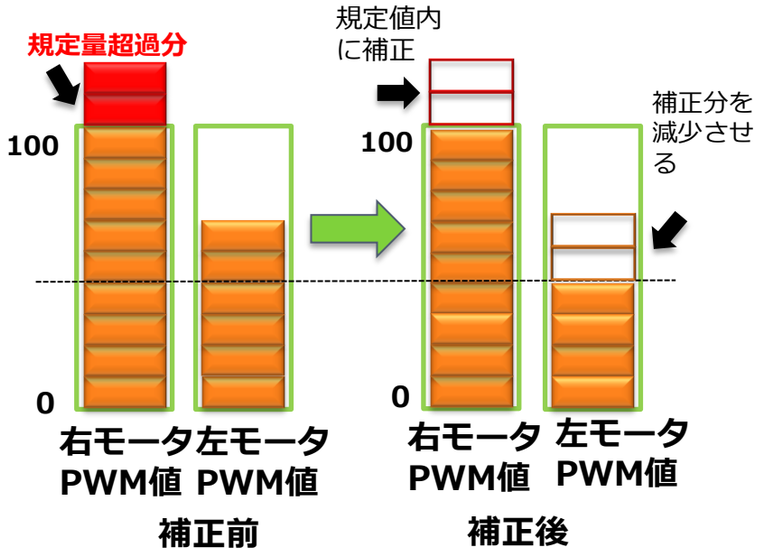



マーカーの灰色領域の路面輝度値は黒色領域より明るく，灰色領域走行中の走行体はラインの内側へと食い込んで走行する．ラインが灰色→黒色へ変化する際の急峻な輝度値の変化をトリガーとして利用し，マーカーを検知する．

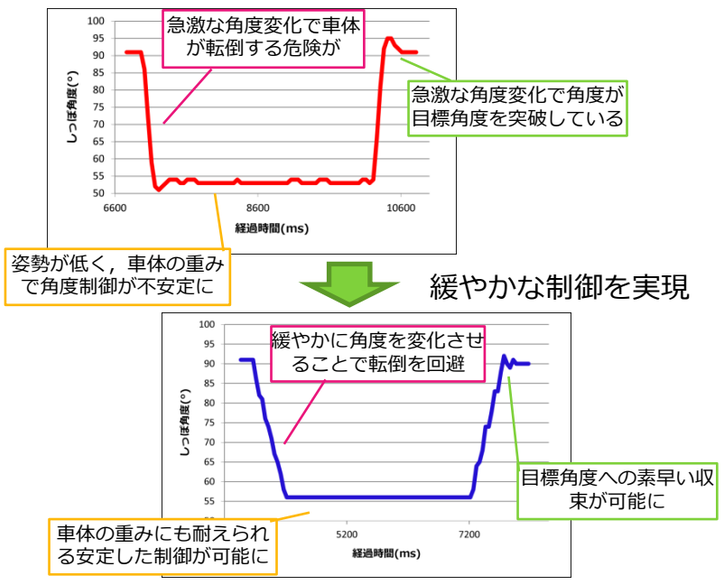
## 旋回量補正

高速走行中のカーブでは算出された左右のモータのPWM値がAPIの入力範囲を超えることがある．その結果を単純に入力範囲に収まるよう値を調整すると，旋回量が不足し曲がり切れない．そこで，モータのPWM値が規定量を超えたら，それを反対側のモータの制御量に反映させることで高速走行における旋回制御を実現している．




## 車体仰角制御

ルックアップゲートを通過するためにはしっぽの角度を変化させ車体を傾け，通過後に元の角度に戻す必要がある．しかし，しっぽの角度の急激な変化によって車体が倒れてしまうなどの問題があった．そこで，しっぽ制御の目標角度自体を最終的な目標角度に達するまで，1°ずつ変化させることで，急激な角度の変化を抑える．




## ライン復帰

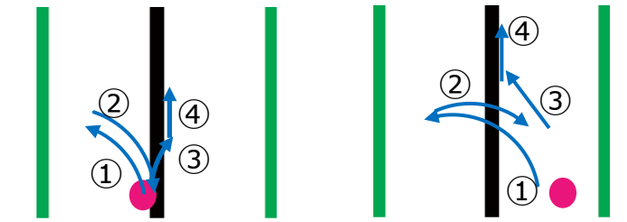
階段、シーソクリア後はラインを見失うことがある.そこで，ラインを探し出しライントレースを再開する必要がある．しかし，難所クリア後の走行ログから自己位置推定するには誤差が多く意図しない動作をする可能性がある．自己位置推定に頼らずラインの左右どちらに外れてしまっても復帰できるようにする必要がある．ラインの存在はラインエッジの検出回数により判断する.なお,必ず右エッジに復帰するように設計した．

**ライン左側に落ちた場合**

①左へ旋回
②エッジ検出回数が2回未満なのでラインは左側にはないと判断
③右へ旋回
④ラインへ復帰

**ライン右側に落ちた場合**

①左へ旋回しエッジ検出を2回するためラインが左側にあると判断
②右へ旋回
③ラインに向かって直進
④ラインへ復帰




**※黒ライン上や左エッジ上に落下した場合でも、エッジ検出回数は2回未満になるので、右エッジに復帰することが可能である**