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3年から7年（専攻科2年）まで7名で構成される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，大会に向けて取り組んできました．昨年度の経験を活かし，モデル・走行共にパワーアップした良いこんぶです！

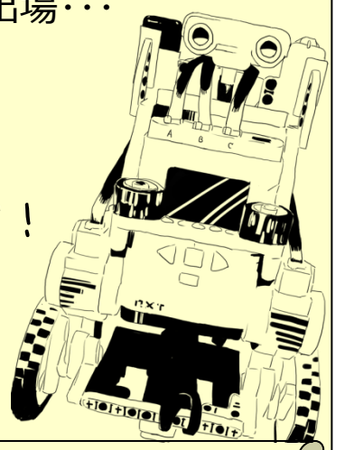
☆組込み，そしてモデリングの未来へ一言

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，新技術がどんどん登場しようと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！！

☆コンテストにかける意気込み，アピール

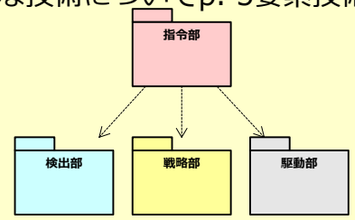
昨年果たせなかった，悲願の全国大会出場･･･
高専生の実力お見せします！

こんぶは頭の栄養！いいこんぶ！



☆モデルの概要

UMLとSysMLを用いてモデルを構成しました．要求図を用いて，大会における目標を実現するための要求を定義しました．（p. 1要求分析を参照）区間を中心に据えた動作要件を満たす為，下図のパッケージ構成を導きだし，構造を分析しました．（→区間を含む構造分析の詳細はp. 2構造を参照）p. 3では区間切替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．各難所での戦略をp. 4に示し，走行を通し使用される主要な技術についてp. 5要素技術で示しています．



☆設計思想

パッケージの分割を開発の初期に行い，パッケージ毎の責務が分散しないよう意識することで，モデルに一貫性を持たせました．双方向の関連を禁止した上で，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の使用を避けました．

☆モデルのここに注目！

コースを「区間」の集合と定義し，それぞれの区間に応じた走行パラメータと区間切替条件を設計すれば完走できることをコンセプトにモデルを構成しました．要素の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．
また，各区間をチームで分担し，並行して開発することで開発スピードの向上が狙えます．それぞれ開発した区間をつなげるだけで容易に結合が行えます．

☆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

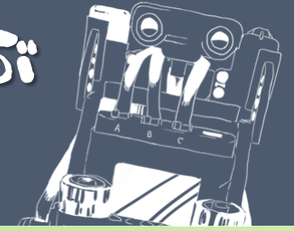
並行性設計の必要性
最優先すべき処理は走行体のバランス制御などにかかせないモータ駆動です．一方で，区間切替はより長い周期でも十分に動作要件を満たすと考えました．（詳細はp. 3振る舞い参照）駆動関連のタスクを最優先とし，それ以外のタスクの優先度を駆動よりも低く設定することで，駆動が求められる周期で確実に行われるように設計しました．
タスクの構造を示すために２つのステレオタイプを用いました．採用するRTOSが提供する機能をnxtOSEK，タスクをTASKとします．

・要求モデルについて

前述の通り，大会における目標についてSysMLの要求図を用いて分析しました．そこから機能要件，非機能要件を洗い出し，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

良いこんぶ



チーム目標

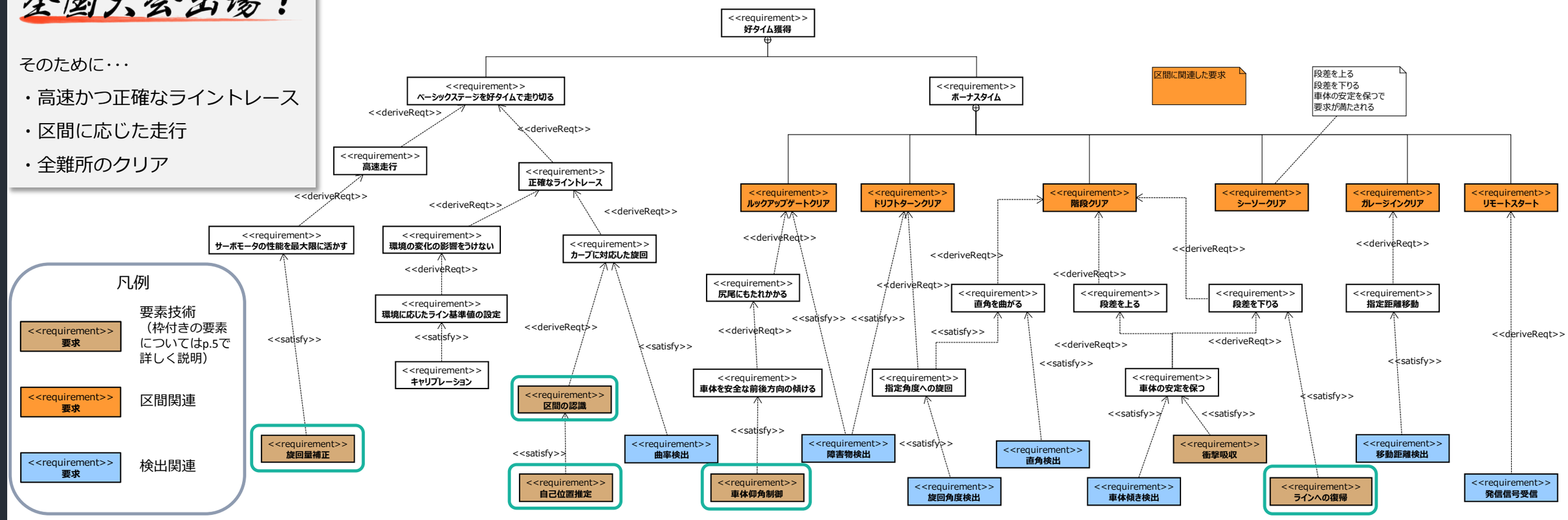
全国大会出場！

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

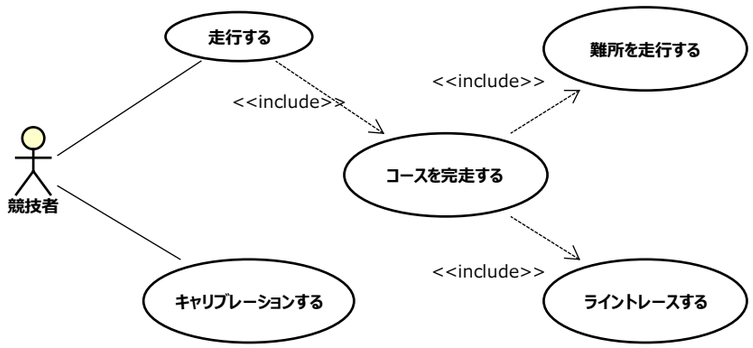
要求図(SysML)

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. 2. 競技者は走行体に走行スタートを指示する. 3.走行体がコースを走行する. 4.走行体がガレージで停止する.

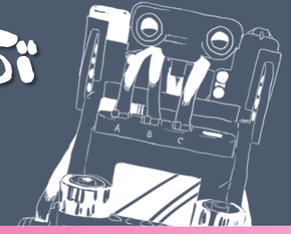
非機能要件

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
走行体を安定して前後方向に傾ける	しっぽの制御が走行体の安定に悪影響を与える	適切なしっぽ角度制御	走行体仰角制御 安定化

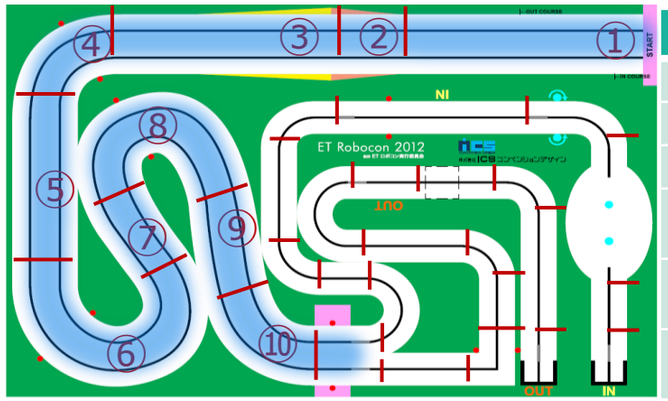
構造

良いこんぶ



ドメイン分析(区間について)

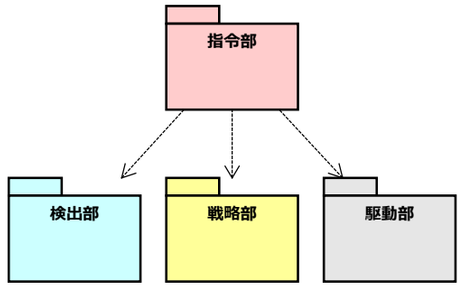
コースは，細かく分割された区間の連続によって構成されるものと分析した．
各区間には，最適な前進量などの目標駆動パラメータと区間の切替条件がある．走行体は区間が切り替わるまで同一のパラメータを用いて走行する．区間切替に用いる情報をトリガーと称する．各区間クラスはトリガーの集合を区間切替条件として持つ．難所エリアには図示されているより詳細な区間が存在する．
→ボーナス・ステージ各難所での動作はp. 4走行戦略参照．



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.
マーカー検出	マーカー特有の輝度値の変化を検出．詳細はp.5要素技術参照.
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.

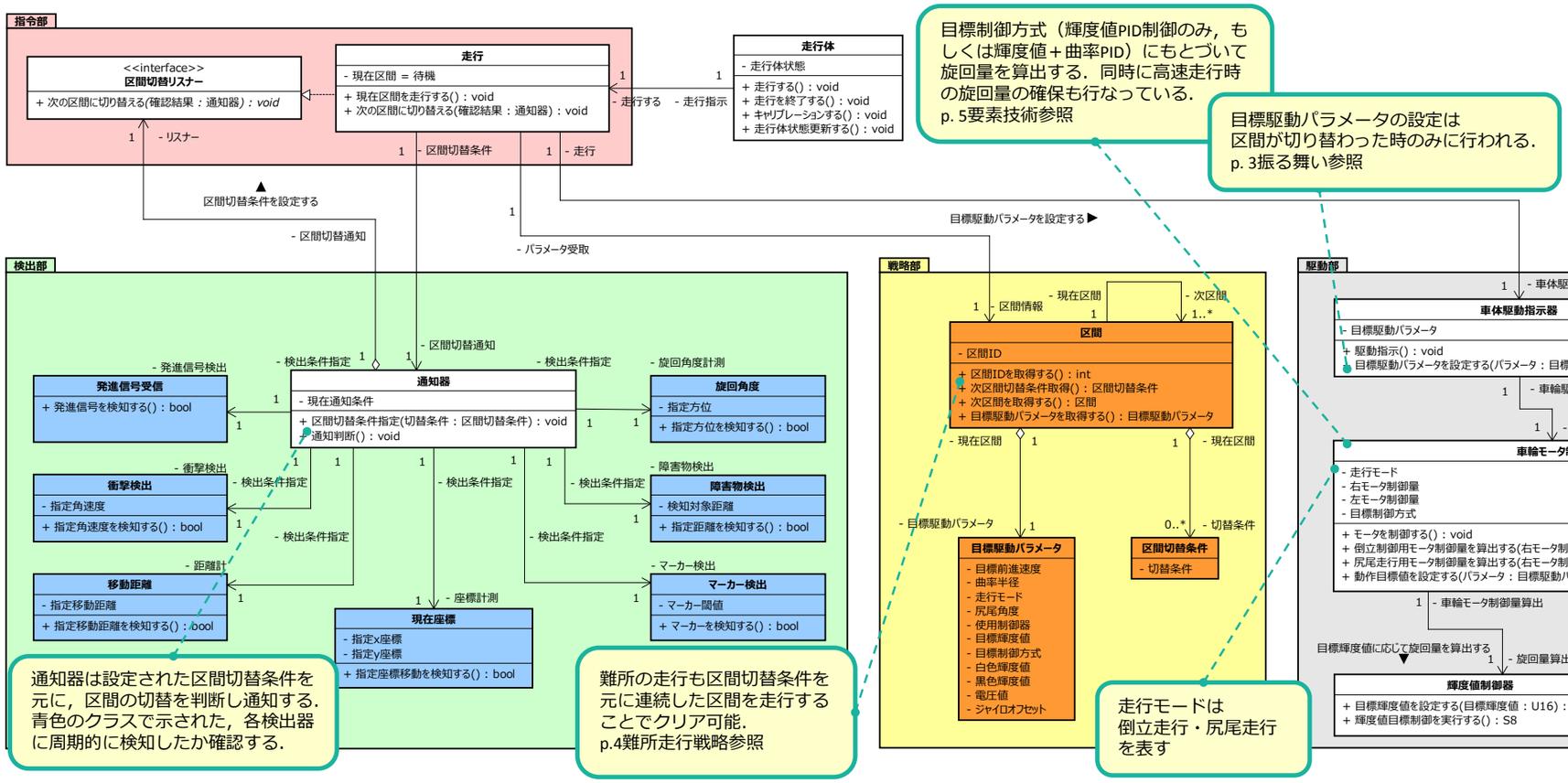
パッケージ図

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．

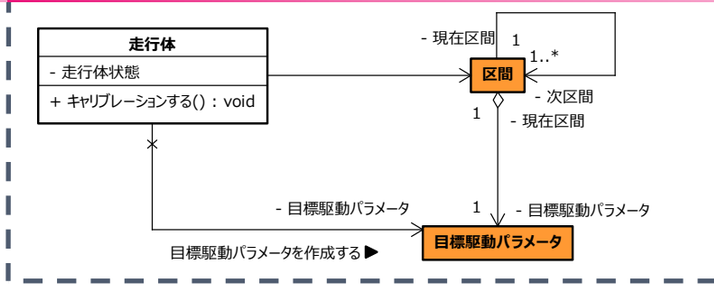


指令部	戦略部，検出部の情報をもとに区間切替を行う.
検出部	センサ類の入力から区間の切替を検出し，指令部に通知する.
戦略部	区間走行に必要なパラメータや，次区間への切替条件などの区間情報を保持する.
駆動部	モータ制御に必要な演算処理を行い，車輪としっぽを駆動する．必要なパラメータは指令部より受け取る.

クラス図(走行関連のみ)

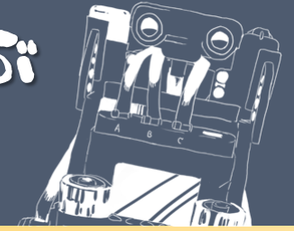


クラス図(キャリブレーション関連)



■ 振る舞い

良いこんぶ



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

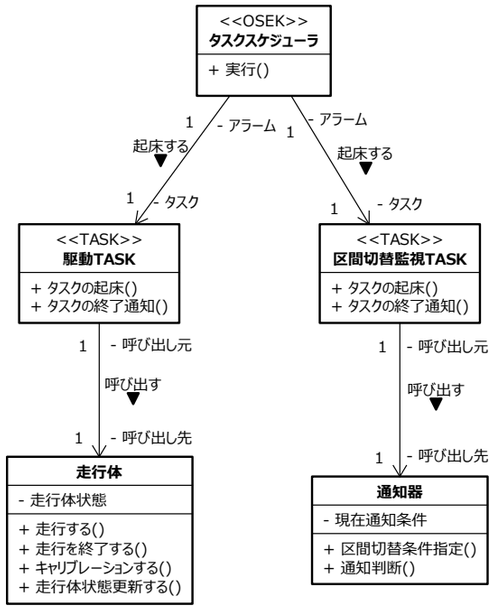
最高速度(60cm/s)で走行中に走行距離をトリガーとして区間を切り替える場合，最大で0.6cm移動する間での区間切替が可能なので，10msの周期は妥当であると判断した．
また，他のセンサをトリガーに区間切替を行う場合も十分な応答が得られた．

■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．

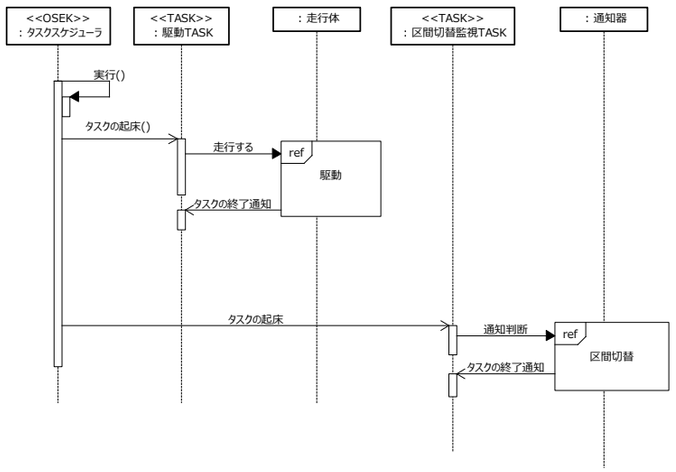
構造

導出した２つのタスクの構造と
呼び出し関係を示した．



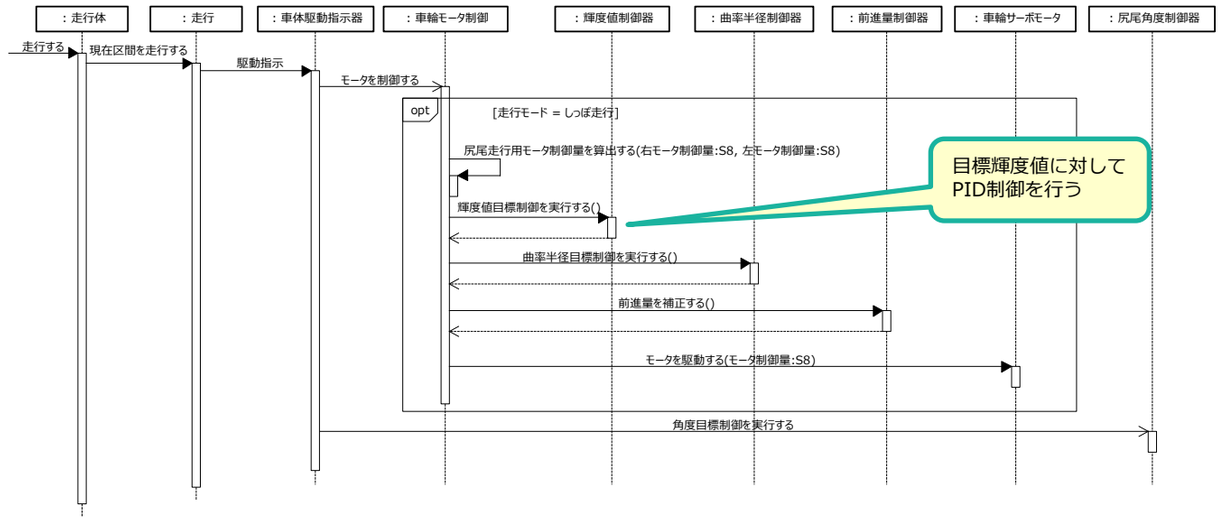
振る舞い

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．

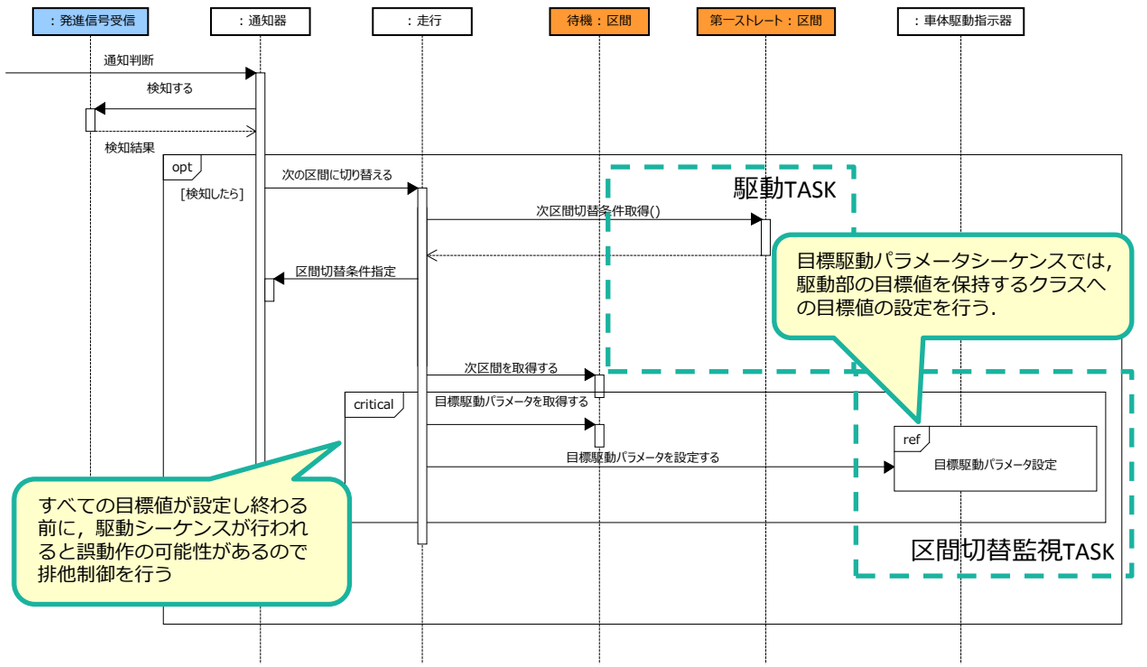


駆動シーケンス

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．

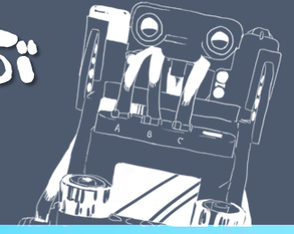


区間切替シーケンス



難所走行戦略

良いこんぶ

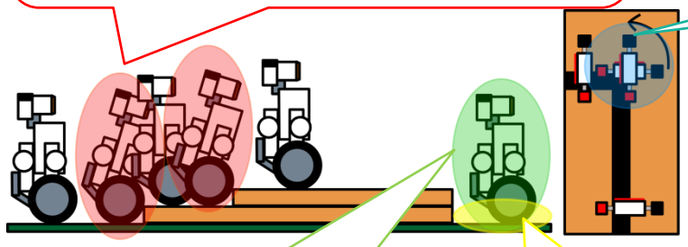


階段

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マシン図を作成した．各状態は各区間に対応している．（他の難所についても同様にステートマシン図を作成）

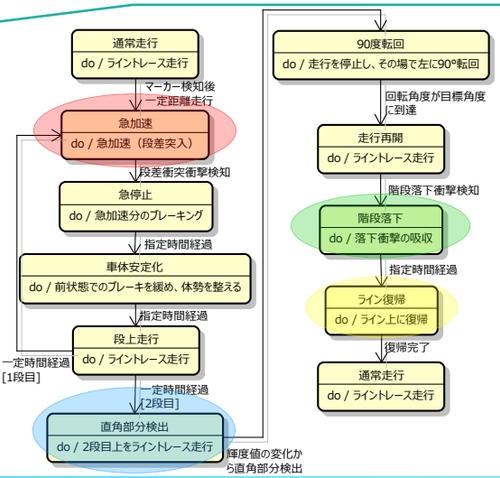
・段差進入時の速度不足
段差を上るためには加速し勢いをつける必要がある.そこで倒立制御APIで用いるジャイロセンサのオフセット値を調節し，走行体を強制的に前傾させることで短距離での急加速を実現.

・直角部分を曲がりきることが出来ない
直角部分の検知が遅れると走行体がラインを見失い，転回後にラントレースを継続出来なくなる．そこで，光センサの値の目標値を走行体が完全にラインを見失う前の値に設定することで，転回後のラントレース継続を実現.



・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.



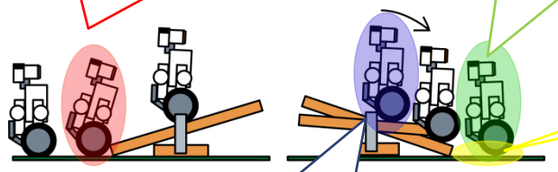
シーソー（シングル）

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応出来なければならない。しかし、階段での戦略を使いまわすことで対応が可能である.

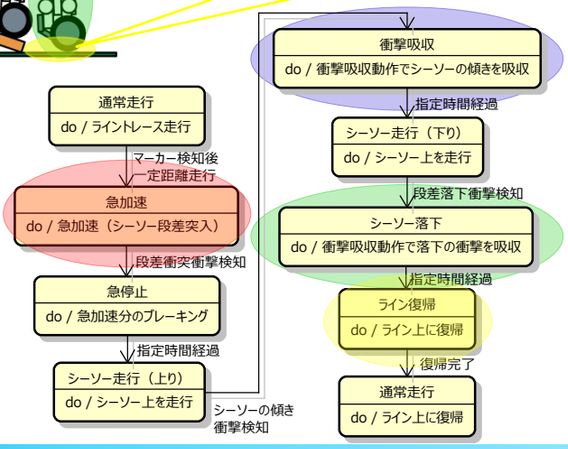
・段差進入時の速度不足
階段での動作と同様にして実現.

・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.

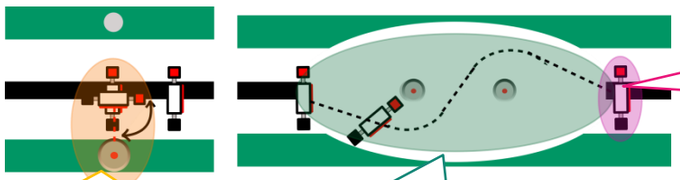


・シーソーの動きによって走行が不安定になる
シーソーが降下した際に走行体が大きく前傾してしまう。そこで，階段落下時と同様に，シーソーの降下に合わせて走行体を後傾させることによって，シーソー上での倒立制御の安定化を実現.



ドリフトターン

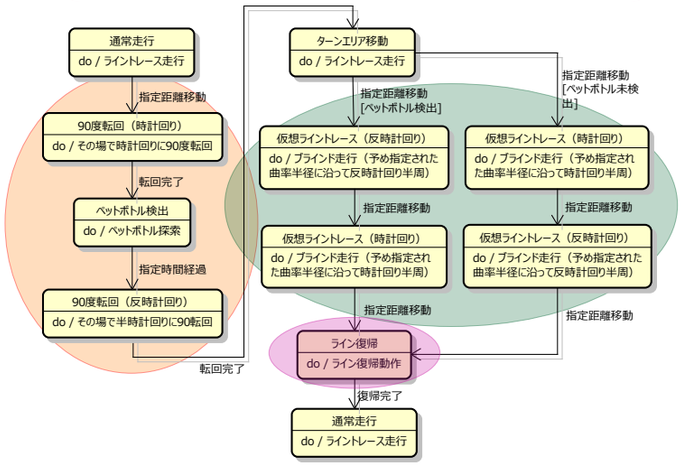
ドリフトターン突破のためには経路選択用ペットボトルの誤検知を防ぎ，ラインの無いエリアを走行する必要がある.



・ペットボトルの誤検知
大会のコース上ではペットボトル付近にもオブジェが置いてありそれを誤検知する可能性がある.そこで検知位置をライン上でペットボトルに最も近い位置にすることで精度の高い検知を実現.

・ラインの無いエリアでの走行
曲率半径PID制御（p. 5 要素技術参照）を利用し，定義した仮想ラインを2つ用意することにより規定された走行を実現.

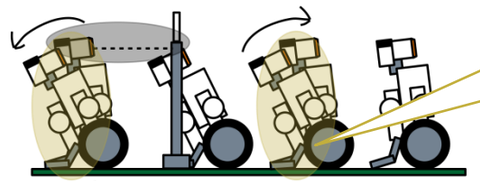
・ターンエリア終了後に走行体がラインに復帰する必要がある
ラインへ斜めに進入しラインエッジを検出後，輝度値PID制御に切り替えることでラインへ復帰.



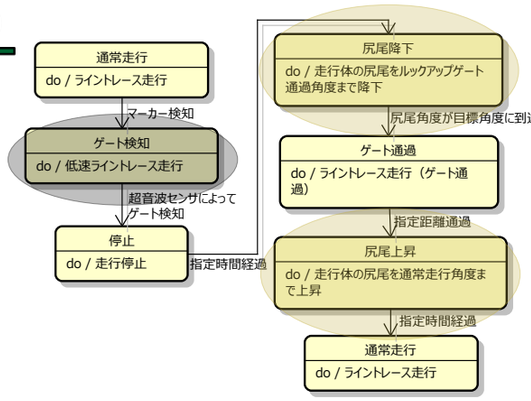
ルックアップゲート

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.

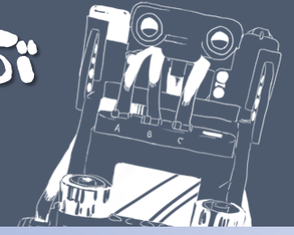


・ゲート検知の失敗
ゲート検知に用いる超音波センサのAPIは仕様上50ms周期で使用しなければならない． よってゲートに接近する際の速度が速すぎると検知が出来ないためゲート接近前に減速することで精度の高い検知を実現.

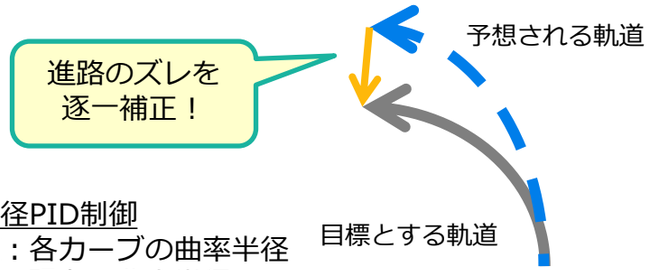


要素技術

良いこんぶ



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．
曲率半径と移動距離算出を組み合わせることにより、仮想のラインをトレースすることも可能である．

→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

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

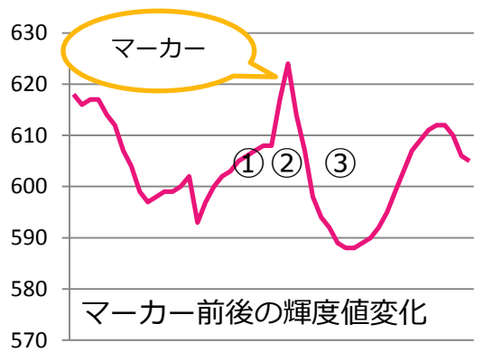
その他必要な定義等

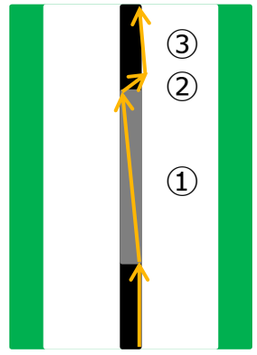
$$\begin{cases} l_L = \frac{\pi}{180}(\Phi_L \times R_w) \quad [cm] \\ l_R = \frac{\pi}{180}(\Phi_R \times R_w) \quad [cm] \end{cases}$$

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

R_w：車輪半径$[cm]$
L_w：二輪間の距離$[cm]$

マーカー検知

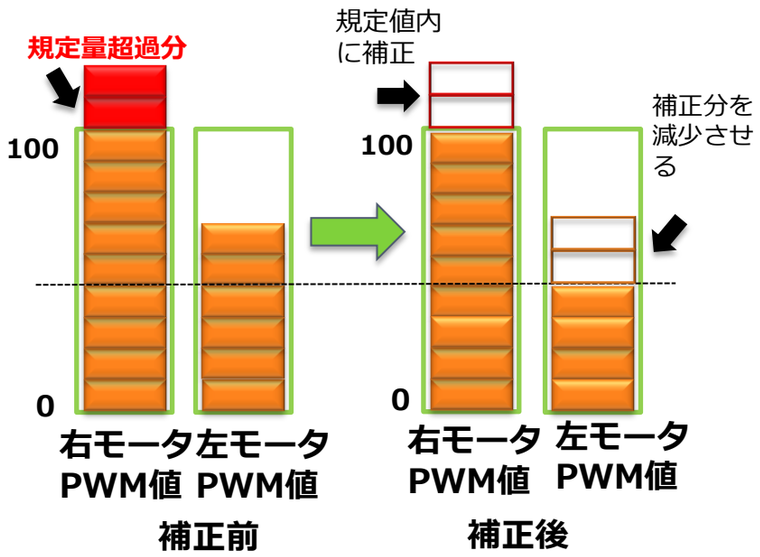




マーカーの灰色領域の路面輝度値は黒色領域より明るく，灰色領域走行中の走行体はラインの内側へと食い込んで走行する．
ラインが灰色→黒色へ変化する際の急峻な輝度値の変化をトリガーとして利用し，マーカーを検知する．

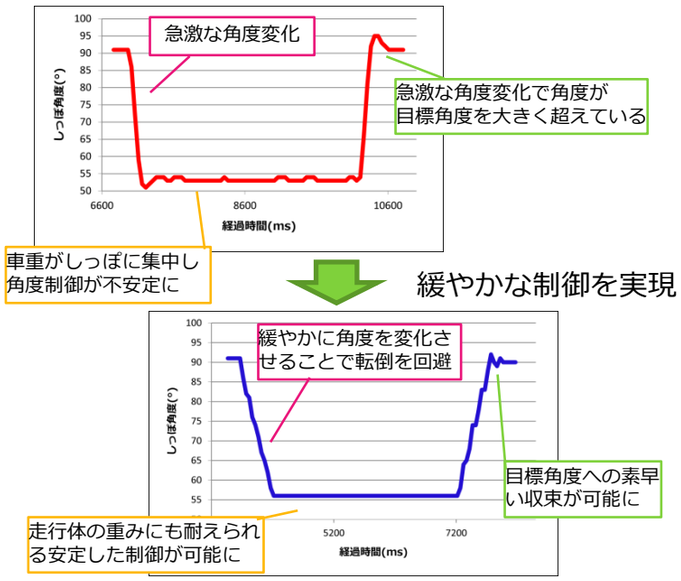
旋回量補正

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，旋回量が飽和し曲がり切れないことがある．そこで，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．



走行体仰角制御

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走行体を傾け，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走行体が倒れてしまうなどの問題があった．そこで，目標とするしっぽ角度に到達するまで、目標角度自体を1°ずつ変化させることで角度の急激な変化を抑える．



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，クリア後に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をするには誤差が多く信頼できない．よって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．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９０度旋回
②エッジ検出回数が2回未満落下地点へ戻る．
③右へ旋回，エッジ検出２回検目でライントレース再開
④ラインへ復帰

ライン右側に落ちた場合

①左へ旋回しエッジを２回検出.ラインが左側にあると判断
②右へ旋回.エッジを２回検出後，さらに旋回
③ラインへ直進，エッジ検出後ライントレース再開
④ラインへ復帰