1. 全体アーキテクチャ

DisplayG

【選択課題】:ベーシックステージ　【目標】:　最速で確実にベーシックステージを走る

Ⅰ. 開発方針

要件分析で抽出した要件に対して、あらかじめ対策方針を立てて、アーキテクチャ設計・機能仕様で対策を講じる

■「最速」で走りきるために

① 滑らかにスタート・走行し最速ゴール
② 走行経路を短縮し最速ゴール

■「確実」に走りきるために

① 会場や個体差に適合しどんな環境でも走破
② 通信途絶や外乱発生、コースアウトしても走破
③ 操作ミスの可能性を軽減

■ 目標達成し易くする為のアーキテクチャ要件

① 変更し易い
② 状況に応じて走行戦略を切り替えし易い

優勝に向けて最も最重な「最速」の要件を詳細化

反映 / 対策 → 要素技術で対応

反映 / 対策 → アーキテクチャで対応

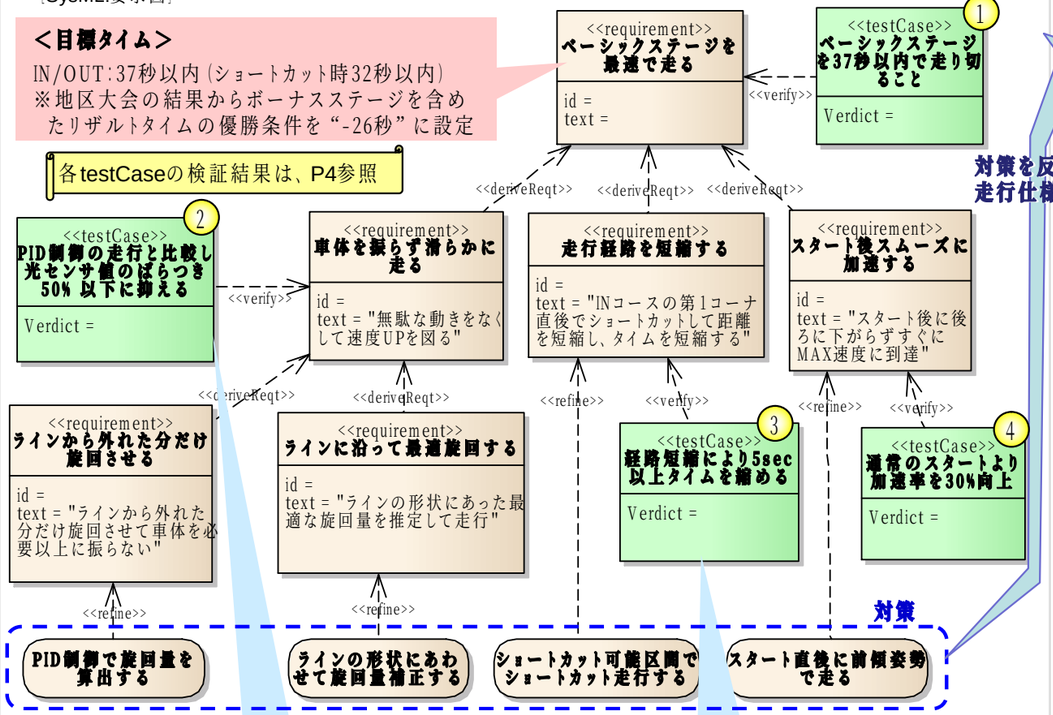
Ⅱ. 要件分析

「最速」を目標タイムで定義して、目標達成のための要件を抽出して対応方針を検討する。 また、各要件の評価基準を明確にして確実な達成を目指す。

［SysML:要求図］

＜目標タイム＞
IN/OUT:37秒以内（ショートカット時32秒以内）
※地区大会の結果からボーナスステージを含めたリザルトタイムの優勝条件を“-26秒”に設定

各testCaseの検証結果は、P4参照



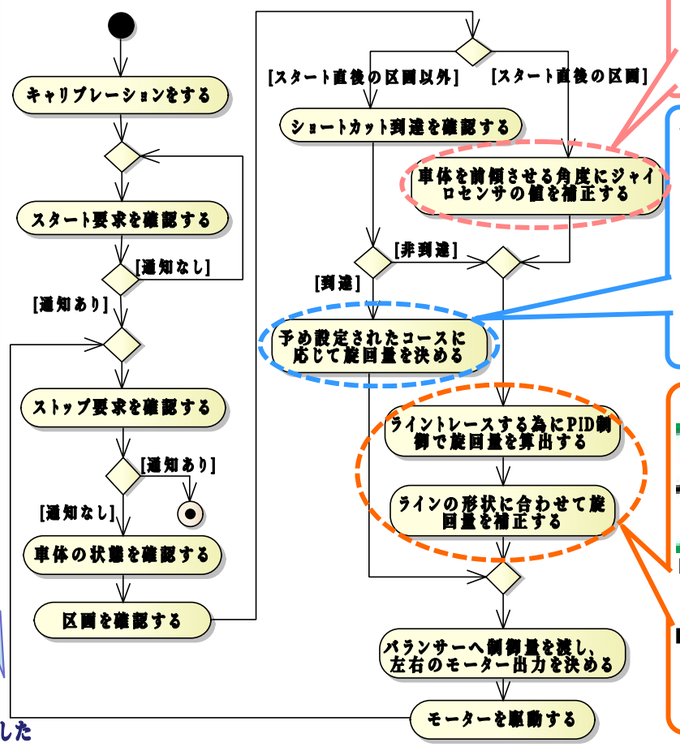
実走行時の光センサ値と目標閾値との差を比較することで、いかに目標ラインに対して無駄なく走行しているかを検証する

ショートカットすると第2ゲートを通過しないのでゲート通過のボーナスタイムがもらえない。その分タイムを短縮する必要がある

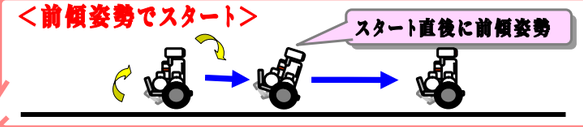
対策を反映した走行仕様

Ⅲ. 走行仕様

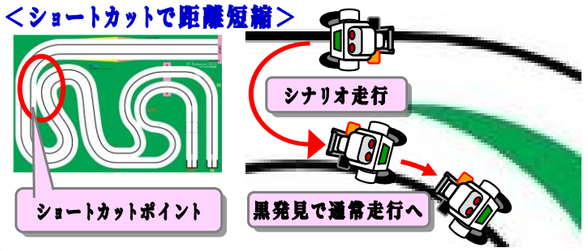
＜全体の処理概要＞ 高速化の対策を織込んだ全体の走行仕様（概要レベルの処理フロー）



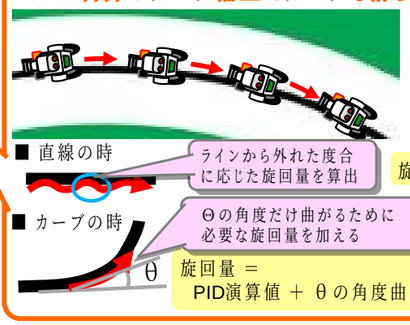
＜前傾姿勢でスタート＞



＜ショートカットで距離短縮＞



＜PID制御＋カーブ補正でカーブも滑らかに走行＞



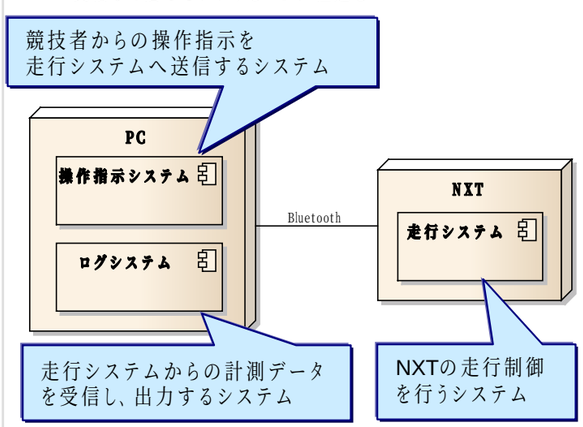
PIDで求めた旋回量を、走行体の速度、コースの曲率半径に応じて補正する

Ⅳ. システム構成

PCとNXTのシステムの分担

＜考慮した要件＞

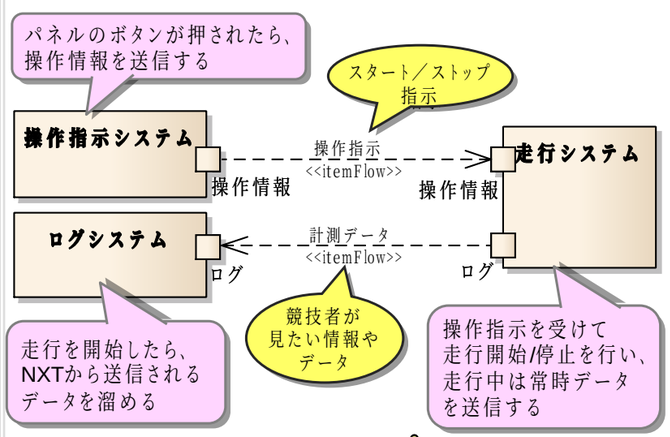
・リモートスタートのボーナス獲得の為、PCにUIアプリを組込み、PCからNXTに指示を送信する
・常時ログが 取れるようNXTが送信したデータを受信して溜めるアプリをPC に組込む



Ⅴ. システム間のやり取り

［SysML:内部ブロック図］

PCとNXT間の関係を示す



『通信途絶』に関する仕様はP5参照

DENSO

2. 詳細構造

＜考慮する要件＞

・変更し易くする ①
　⇒ **対応方針：関心事を分離して変更の影響を抑える**
・状況に応じて走行戦略を切り替えし易い ②
　⇒ **走行バリエーションを部品化**
・走行仕様、要素技術の実現 ①②①②③
　⇒ **実現する箇所（部品）の設計**

＜パッケージ分割方針＞

1）変更頻度を考えて、
以下の〈くりで分離する
・一番変更が多い走行戦略に関するもの
・繋ぐ機器で変更がある通信に関するもの
・デバイスとの入出力に関するもの
・車体やラインなどの状態推定や基本的な車体の操作などの要素技術
2）下位のパッケージを安定させて、変更頻度の高い走行戦略のパッケージだけを変更する

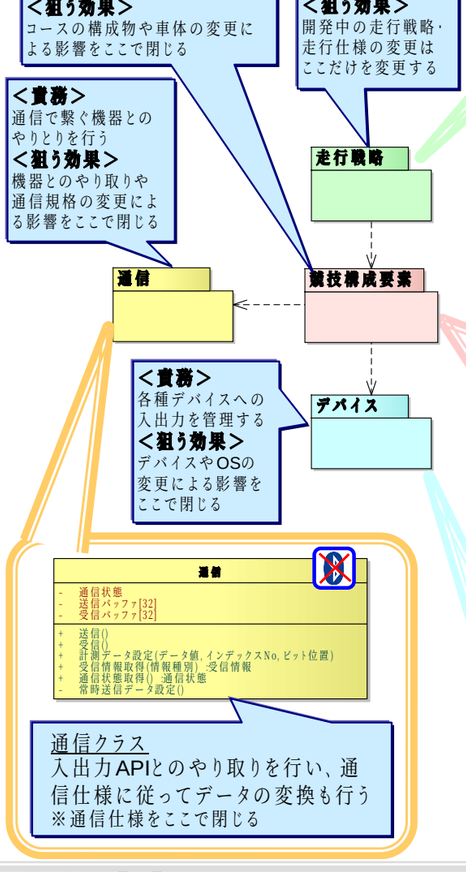
＜パッケージ内の設計方針＞

・**競技構成要素パッケージとデバイスパッケージ**
- 構成する物ごとにクラスを分割
　※構成する物は、認識するもの全般で立体でないラインも含む

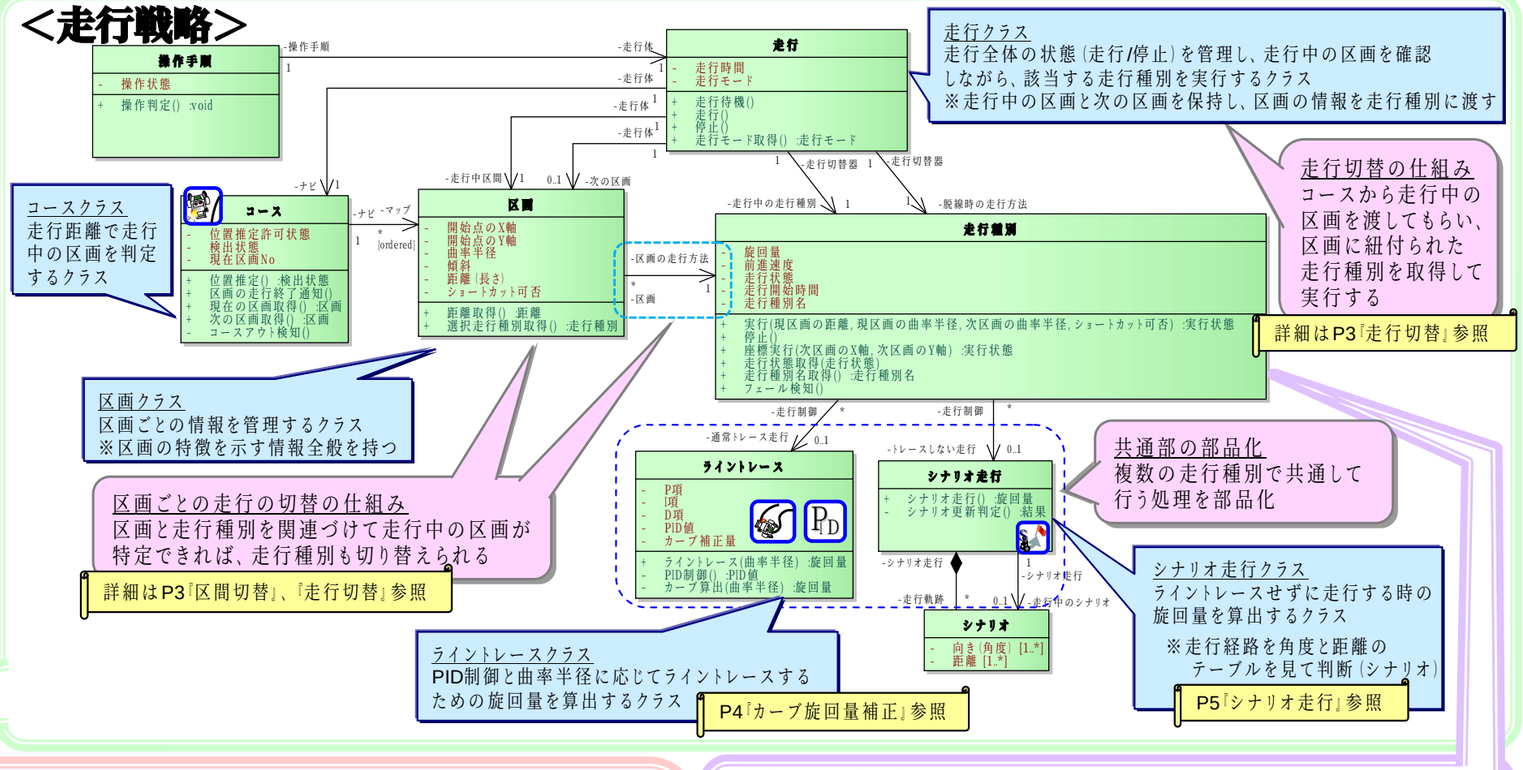
・**走行戦略パッケージ**
- 基本的に走行仕様でやろうとしている機能毎にクラスを分割
- 走行の切り替えの仕組みと適合への対応を検討する

・**デバイスパッケージ、通信パッケージ**
- 基本的に種類ごとにクラスを分割（デバイスの種類、通信の種類）

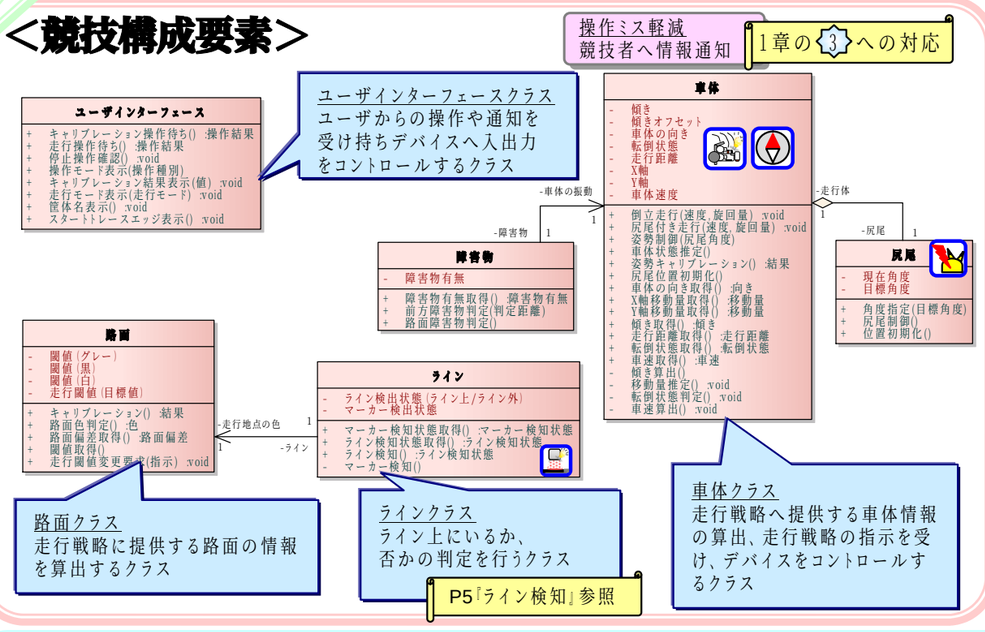
＜パッケージ構造＞



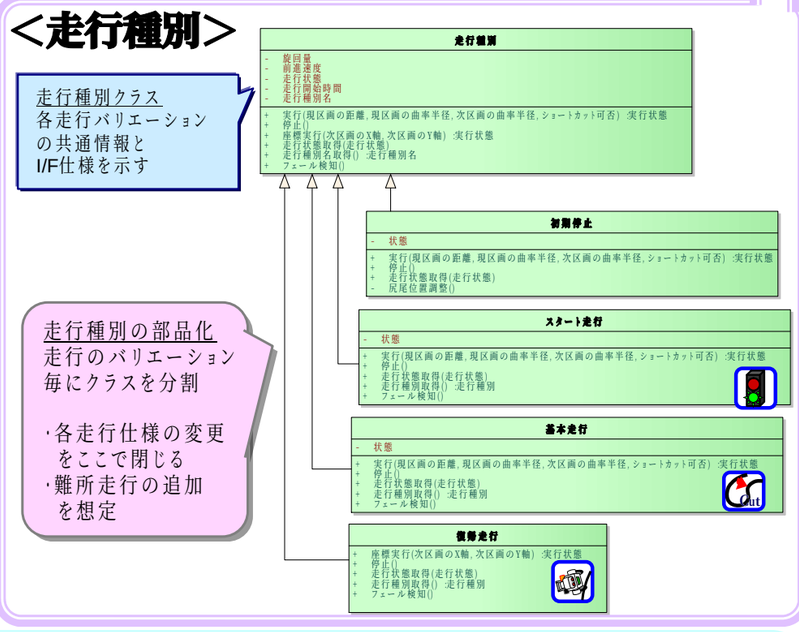
＜走行戦略＞

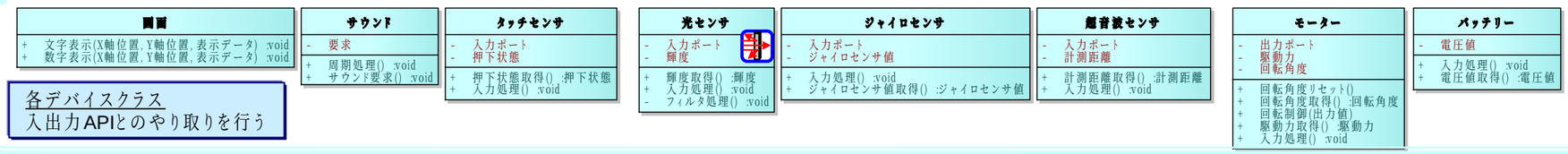


＜競技構成要素＞



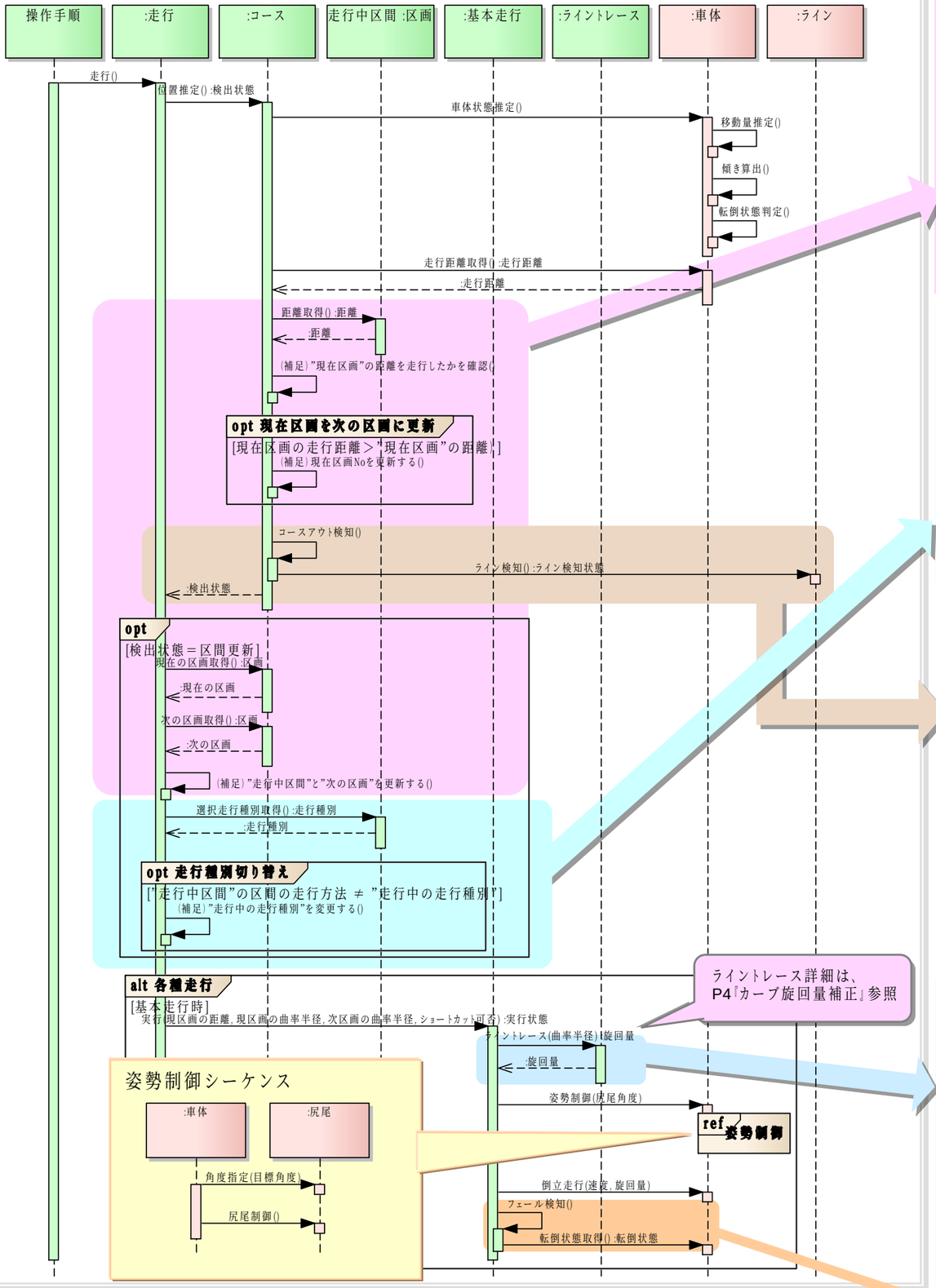
＜走行種別＞





3. 振る舞い

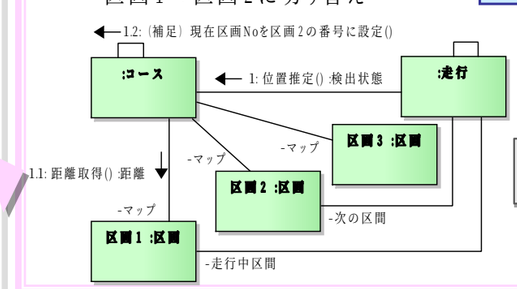
基本走行

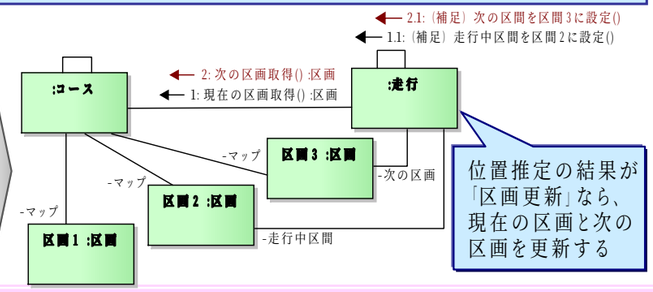


ラントレース詳細は、
P4『カーブ旋回量補正』参照

区画切替

例：走行中の区画を
区画1→区画2に切り替え



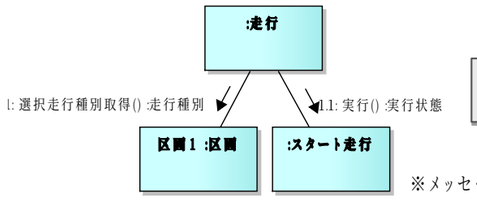


位置推定の結果が「区画更新」なら、現在の区画と次の区画を更新する

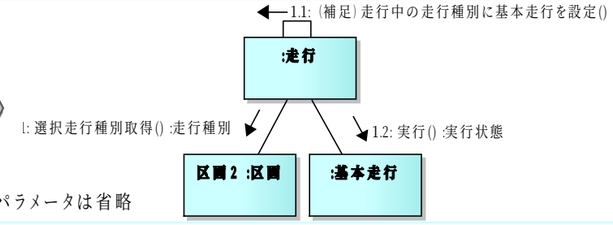
＜区画切替判定の仕様＞
コースは走行距離と区画の距離を比較してどの区画にいるかを判断
・「現在区画内の走行距離＞区画の距離」ならば、「現在区画（区画No）を次区画に更新し、区画更新を走行クラスに返す

走行切替

例：スタート走行→基本走行の切替



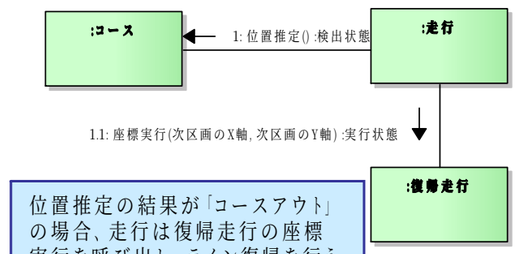
※メッセージのパラメータは省略



＜走行切替時の振る舞い＞
位置推定の結果が「区画更新」なら、走行は現在の区画から該当する走行種別を受け取り走行を実行する

フェール動作

＜コースアウト時の振る舞い＞

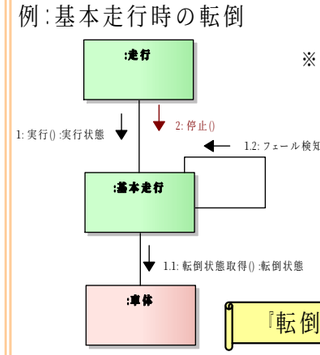


位置推定の結果が「コースアウト」の場合、走行は復帰走行の座標実行を呼び出し、ライン復帰を行う

『復帰走行』詳細はP5参照

＜転倒時の振る舞い＞

例：基本走行時の転倒



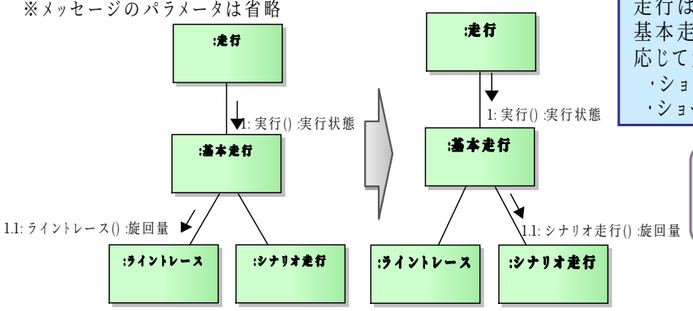
※メッセージのパラメータは省略

各走行種別で実行の最後にフェール検知を行い、転倒を検知したら、走行停止する

『転倒検知』詳細はP5参照

ショートカット切替

※メッセージのパラメータは省略



＜ショートカットの振る舞い＞
走行は走行中区画のショートカット可否の情報を基本走行に渡し、基本走行でショートカット可否に応じて走行を判定する
・ショートカット「否」なら、ラントレース走行を実行
・ショートカット「可」なら、シナリオ走行を実行

区画のショートカット可否の設定を変更することで、当日の状況に応じてショートカットする／しないの戦略変更可能

『ショートカット走行』詳細はP4参照

4. 走行仕様

確実に走るための方針： 環境差・個体差の影響とミスケース（通信途絶・外乱など）を課題として抽出し、対策を考える

カーブ旋回量補正

■目標：滑らかなコーナリング

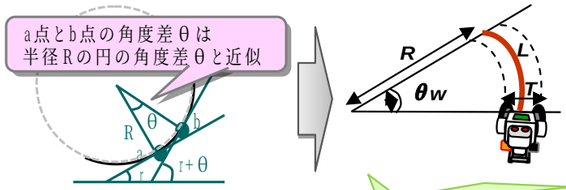
→P5の PD を使用

【要件】 カーブに応じて最適な旋回量に補正する

【課題】
（1）開発中は床に敷いて弛んだコースで走行し、本番コースとは張り具合が異なるので距離に誤差がでる可能性がある

【解決策】
（課題1）走行ロジックに影響しないように区画を適合できるようにする

＜補正旋回量の算出仕様＞ 角度θに近似した半径Rを旋回するのにかかる遠心力から旋回量（turn値）を算出する



＜演算式＞

$\theta w = Rw(\theta r - \theta l)/T$
$L = Rw(\theta r + \theta l)/2$
$V = L/t$
$Fr = (m*V^2)/R$
$Req\_turn = \alpha \times Fr$

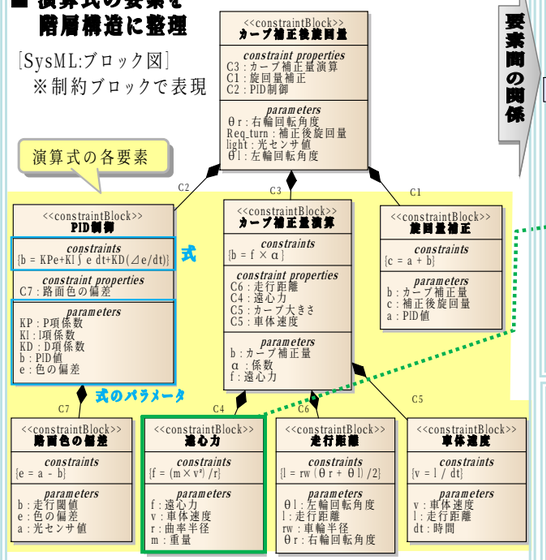
遠心力から旋回量への変換係数αは、合わせ込み

θr, θl ： 車輪回転角度
Rw ： 車輪半径
T ： 車輪間距離
m ： 重量
θw ： 車体旋回角度
L ： 車体移動距離
t ： 時間
V ： 速度
R ： 曲率半径
Fr ： 遠心力
Req_turn ： 旋回量（要求値）

＜演算仕様の整理と対応箇所＞

■ 演算式の要素を階層構造に整理

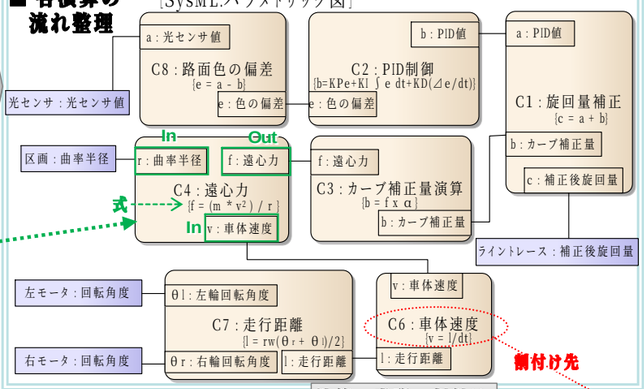
［SysML:ブロック図］
※制約ブロックで表現



要素間の関係

■ 各演算の流れ整理

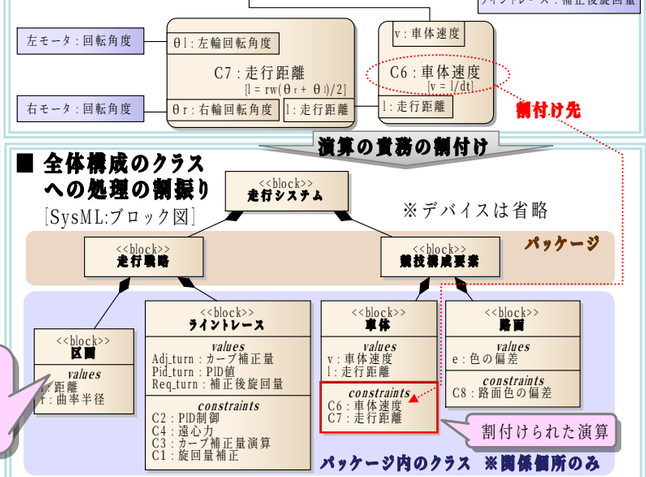
［SysML:パラメトリック図］



演算の責務の割付け

■ 全体構成のクラスへの処理の割振り

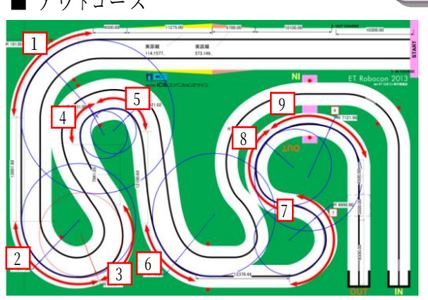
［SysML:ブロック図］



＜コースの区画分割＞
カーブの曲率半径を計測し、
直線と曲率半径が異なる曲線部を分割

区画の適合のみで当日のコース環境に対応可能

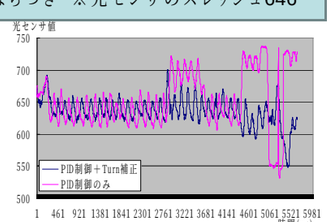
■ アウトコース



＜結果＞

TestCase ① ②	Result
ベーシックステージ走行 37秒以内	IN：36.2秒 OUT：36.5秒
光センサのばらつき	52%低減

参考：旋回量補正有り・無し時の光センサのばらつき ※光センサのスレッシュ646



スタート高速化

■目標：スタートから100msで速度MAX

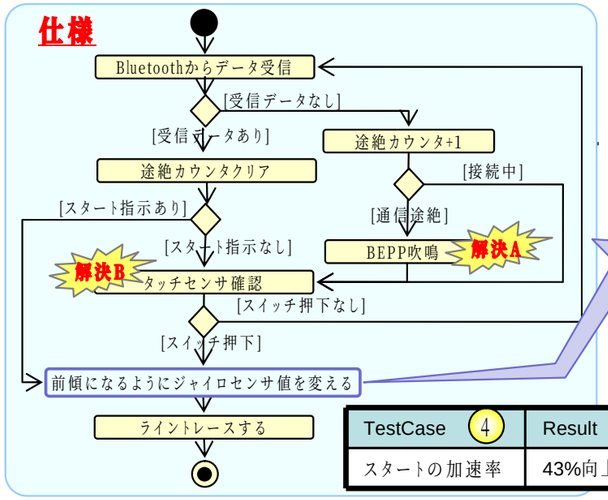
→P5の を使用

【要件】 スタート直後に前傾させて、後退しないでスムーズに加速させる

【課題】
（1）Bluetoothの通信が途絶するとスタート指示ができない

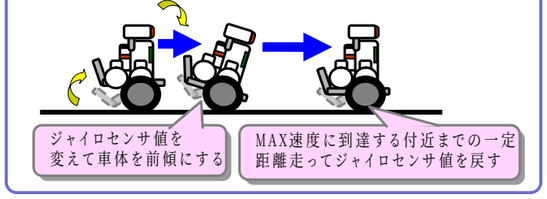
【解決策】
（課題1）常時通信を監視し、通信途絶を検知したらすぐにBEEP音で通知する ⇒解決A
再接続には手間がかかるので、タッチセンサでもスタートできるようにする ⇒解決B

仕様

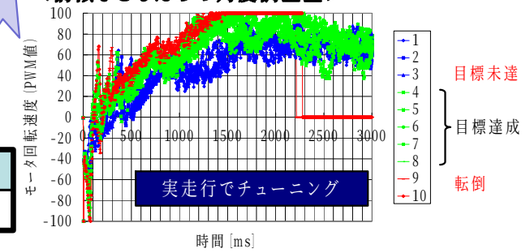


TestCase ④	Result
スタートの加速率	43%向上

＜仕様イメージ＞ 前傾させてスタートする動きのイメージ



＜前傾させるための角度調整量＞



ショートカット走行

■目標：最短距離の走行で時間短縮

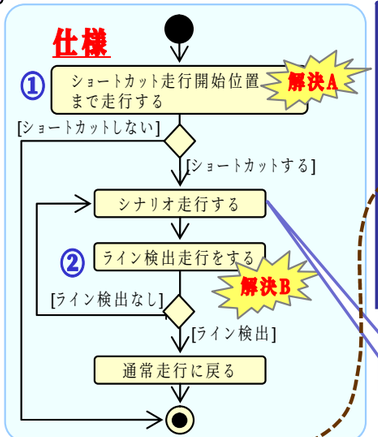
→P5の を使用

【要件】 インコース第1コーナー後にショートカットして走行距離を短縮する

【課題】
（1）スタート時の走行体の設置場所や走行中のタイヤ空転により距離がずれ、ショートカット走行を開始する位置が一定にならない
（2）シナリオ走行のターンでは電圧などの違いでずれが生じるため、同じ目標地点に到着せず、ラインを見失う可能性がある

【解決策】
（課題1）第1コーナーを曲がったことを判定して走行距離を補正する ⇒解決A
（課題2）走行体が走行ラインと平行に近くなるまでシナリオ走行し、必ず鋭角でラインを検知できるようにする ⇒解決B

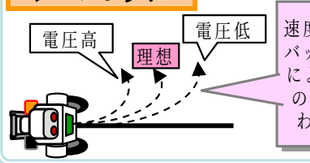
仕様



＜シナリオ走行とは＞
予め決めた経路を「向き」と「距離」の指示で走行させること

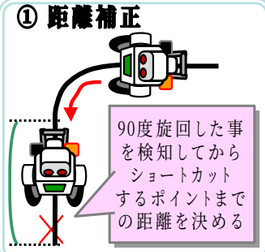
TestCase ③	Result
ショートカットの短縮時間	-6.5秒の短縮

ターンのずれ



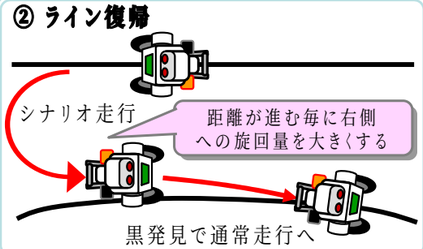
速度一定でも、バッテリ電圧等によってターンの大きさが変わってしまう

① 距離補正



90度旋回した事を検知してからショートカットするポイントまでの距離を決める

② ライン復帰



DENSO

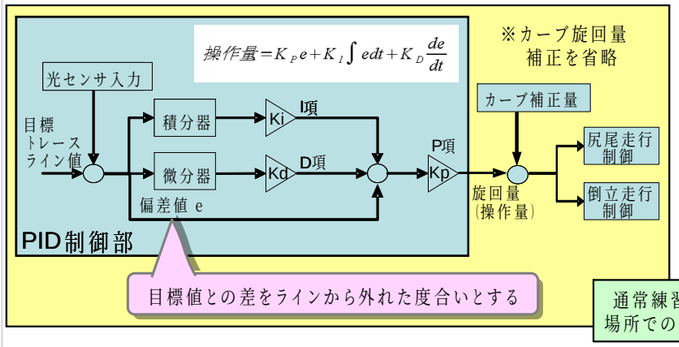
5. 要素技術

確実に走るための方針： 環境差・個体差の影響とミスケース（外乱・誤検知など）を課題として抽出し、対策を考える

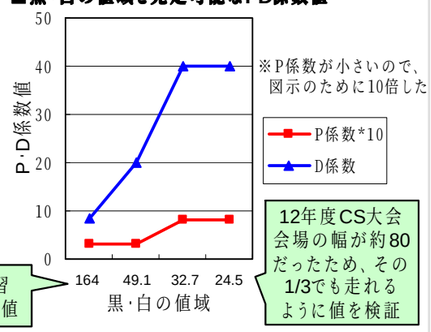
PID制御

1章の①への対応

【要件】ラインと走行体の距離（ラインから離れている距離）に応じて旋回させる
【課題】会場の照明によっては光センサで取れる黒・白の値域が小さくなり、PID制御の旋回量が足らずコースアウトする
【解決策】光センサのキャリブレーション時に、その時の黒・白値域にあわせて完走できるPID係数を設定する



■黒・白の値域と完走可能なPD係数値



走行体位置推定

【要件】走行データとして、走行距離、方位、座標を算出する
【課題】走行体の個体差により測定値に誤差が発生する
【解決策】走行体をテストコースを測定させ、コースの実測値に対する平均値のずれ（個体差）を補正する

＜演算仕様＞

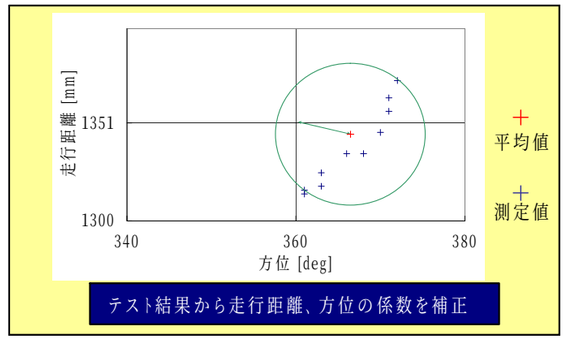
$$\begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{T} & -\frac{R}{T} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix}$$

直進速度：v
方位角速度：ω
右車輪角速度：ω_r
左車輪角速度：ω_l
車輪直径：R=40mm
車輪間距離：T=160mm

＜走行データ＞

$$L = \int \upsilon c dt \qquad \theta = \int \omega dt$$
$$x = \int \upsilon \cos\theta dt \qquad y = \int \upsilon \sin\theta dt$$

走行距離：L、方位：θ、位置：x, y

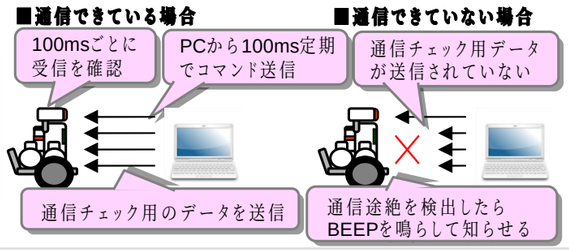


通信途絶

1章の②への対応

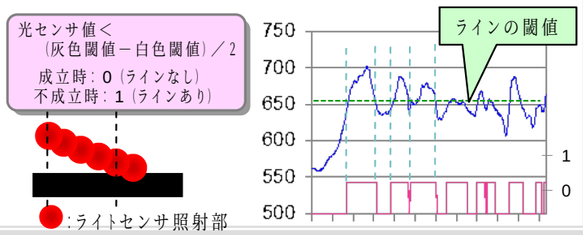
【要件】Blutooth通信が途絶していることを検出する
【課題】早く検出して通知しないと、スタートの合図に間に合わない
【解決策】常時、誤検出しない間隔で監視して速やかに通知する

＜検出仕様＞
- データ更新なしが400ms間継続 成立時：通信途絶
  ※瞬断で途絶判定しないよう連続4回更新なしで途絶
- データ更新あり 成立時：通信途絶なし



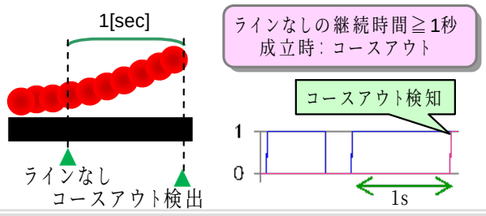
ライン検知

【要件】光センサの値からラインの有無を検知する
【課題】目標トレースライン値は灰色に近い値なので、マーカー（灰色線）が検知できない
【解決策】灰色と白色の中間の閾値でライン検出を行う



コースアウト検知

【要件】ラインの検出状態からコースアウトを検知する
【課題】カーブなど一時的にラインから離れて走行した場合にコースアウトと誤判定する
【解決策】コースアウトの判定時間をラインに復帰させた方が良いと思うくらいの長さ（1秒）にする



光センサフィルタ

1章の①②への対応

【要件】光センサの値をフィルタし瞬間的な外乱の影響を抑える
【課題】高速走行した場合に、フィルタする回数が多いと通常の光センサの変化に対して反応が遅れ、コースアウトする
【解決策】走行体の速度に対してコースアウトしないフィルタ回数を検証し、当日の外乱の状況に応じて走行を変更する

■フィルタ回数ごとの走行検証

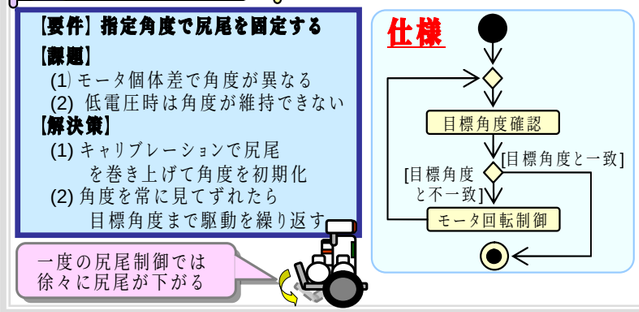
速度	フィルタ回数 ※2	ゴールタイム（INコース）
170 ※1	2回	36.5秒
150	4回	40.9秒
130	6回	45.1秒

※1 速度を170以上にすると走行が安定しないため、上限を170としている
※2 完走率9割以上のフィルタ回数

尻尾制御

1章の①への対応

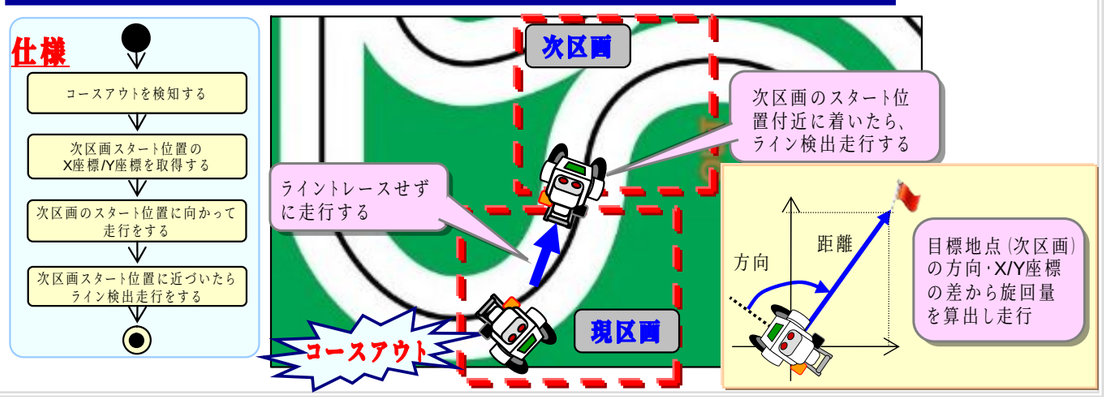
【要件】指定角度で尻尾を固定する
【課題】
(1) モータ個体差で角度が異なる
(2) 低電圧時は角度が維持できない
【解決策】
(1) キャリブレーションで尻尾を巻き上げて角度を初期化
(2) 角度を常に見てずれたら目標角度まで駆動を繰り返す



復帰走行

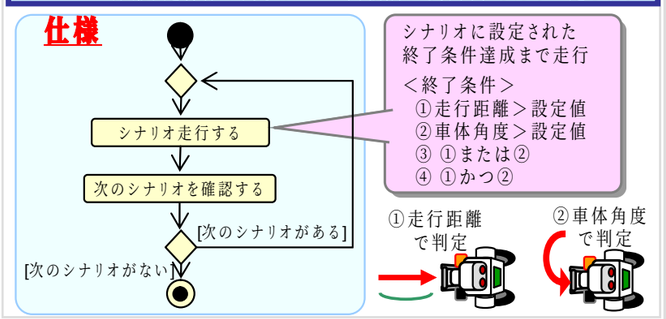
1章の②への対応

【要件】コースアウトを検知したらラインを探して復帰させる
【課題】タイヤの空転などで推定した座標には誤差が生じるので、ラインに復帰できない可能性がある
【解決策】復帰位置付近に来たらラインを探索して確実に復帰させる



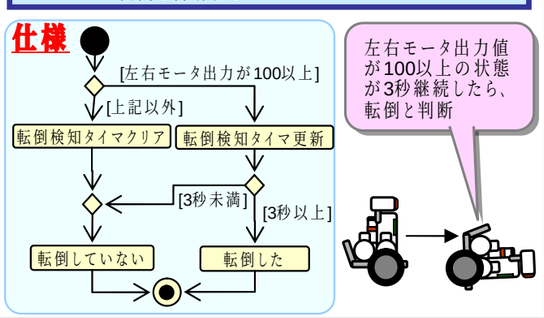
シナリオ走行

【要件】ラインのないところを自在に走行できること
【課題】電池残量の違いで、同じシナリオでも走行内容が異なる
【解決策】終了条件を距離、走行体の向きの到達で判断する
※距離、走行体の向きはモータ回転数から算出するため一定となる



転倒検知

【要件】モータの出力値から走行体の転倒を検知する
【課題】瞬間的なモータ高出力により転倒を誤検知してしまう
【解決策】左右モータの出力値が3秒以上、100を超えたら転倒と判断する



DENSO