ヒカリバクシンオー

「確実に走破する」ために必要な要求を分析！

P1-1：活動方針の検討

■ＣＳ大会で全難所クリアして完走！

歴代メンバーが達成したことのない目標にチャレンジするという意気を込めてチーム目標を決定。目標到達のために施策を検討し、昨年までの傾向や現メンバーの希望も考慮し以下の活動方針を立てました。

1. Rコース/Lコースを確実に走破する
- 制限時間内にゴールできるスピードを確保する
- コースアウト/リタイアの要因分析に注力

2. バーコードと仕様未確定エリアⅡを確実に走破する
- 仕様未確定エリアⅡはスタート直前に障害物の配置が決まる
- 全256通りのルートを事前に用意しておくのは現実的ではない

⇒ 動的にルートを作る必要がある
⇒ どのルートでも走破できるようにする

3. 使い慣れたNXTで出場

NXTかEV3どちらか選択できる今大会、諸先輩方含め長年使い慣れているNXTで大会に出場します

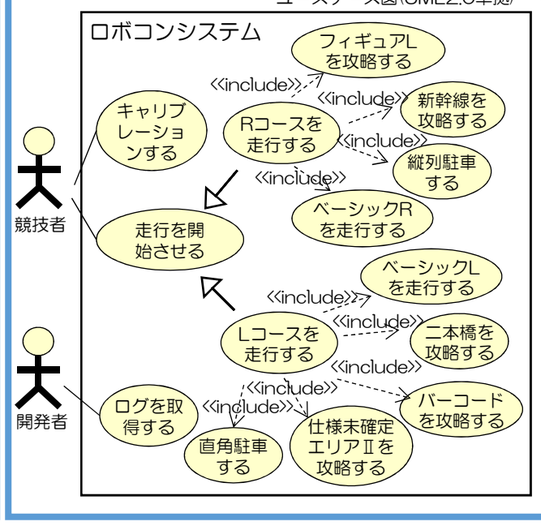
4. モデルを評価されたい！

やっぱりモデルを褒められたい！(by メンバー一同)

P1-2：ユースケースの抽出

活動方針「Rコース/Lコースを確実に走破する」において、本システムに必要なユースケースを抽出した

ユースケース図(UML2.0準拠)



※用語に対して以下のように定義する

：難所
：ベーシックR
：ベーシックL

ベーシックR+ベーシックLを「ベーシックコース」とする。

※ここではユースケース記述の一例として「走行を開始させる」をあげる

UC名	走行を開始させる
概要	走行を開始させる
アクター	競技者
事前条件	キャリブレーションが終了していること
事後条件	走行が開始していること
基本フロー	1. 競技者は走行開始を指示する 2. システムはRまたはLコースの走行を開始する 3. システムはRまたはLコースの走行が開始したか判定する 4. ユースケースを終了する

P1-3：機能要件

ユースケースを満たすために必要な機能要件をユースケース記述から抽出した
(用語についての説明はP1-5で説明する)

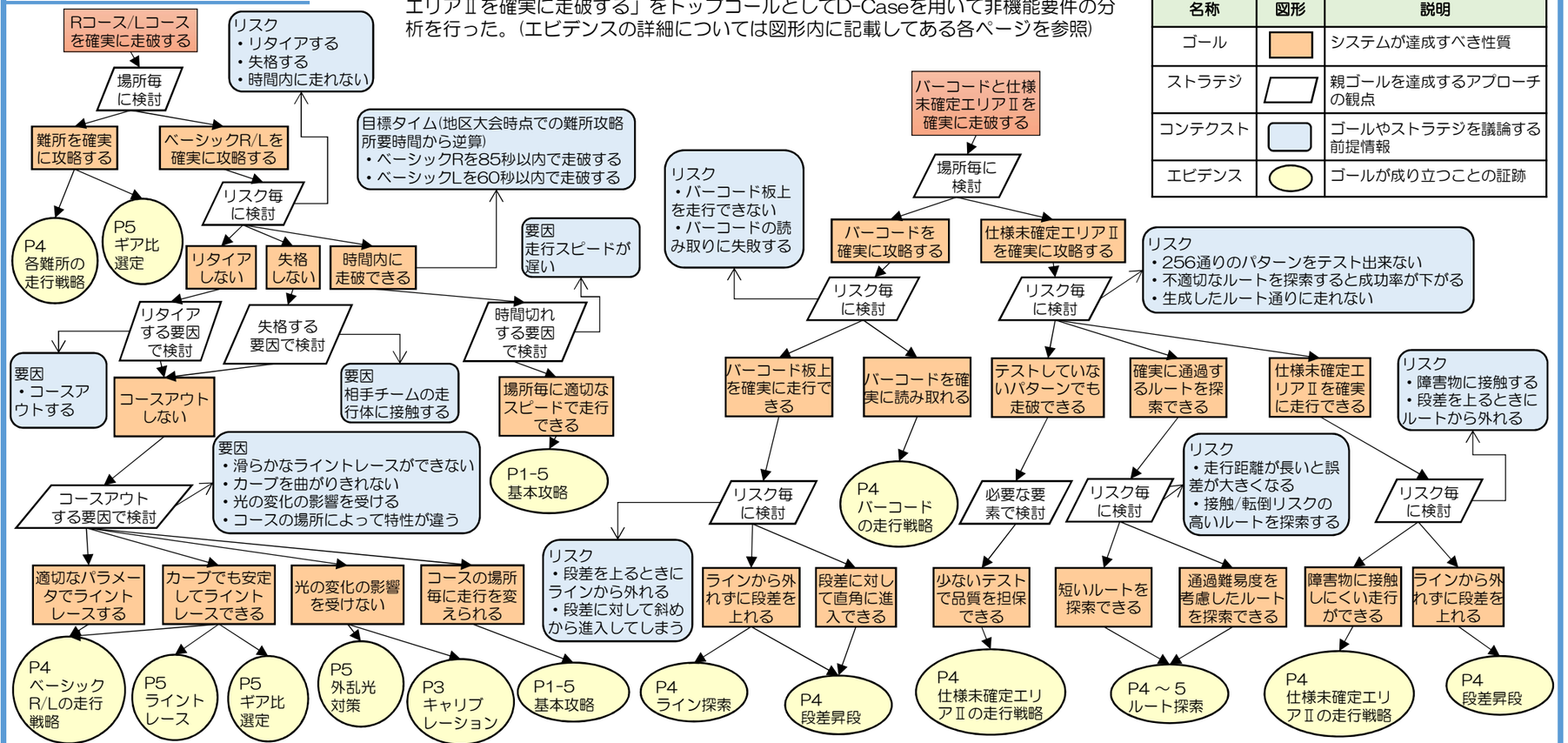
ユースケース ＼ 機能要件	キャリブレーション	スタート	ログ	走行：ラインレース	走行：停止	走行：回転走行	走行：直進走行	走行：PWM値指定走行	検知：走行モード移行検知	検知：距離検知	検知：回転角検知	検知：時間検知	検知：白/黒検知	検知：衝撃検知	検知：障害物検知	検知：前輪回転数検知	検知：後輪停止検知	追加アクション：バーコード読み取り	追加アクション：ルート探索	追加アクション：区間生成
キャリブレーション	○																			
走行開始		○																		
ベーシックR/L走行				○						○				○						
ログ取得			○																	
フィギュアL攻略				○	○	○	○	○		○	○	○	○	○		○	○			
新幹線攻略				○	○		○			○		○		○	○					
縦列/直角駐車				○	○	○	○	○	○	○	○	○								
二本橋攻略				○	○	○	○	○		○	○	○	○	○		○	○			
バーコード攻略				○	○	○	○	○		○	○	○	○	○		○	○	○		
仕様未確定エリアⅡ攻略				○	○	○	○	○	○	○	○		○	○		○	○		○	○

P1-4：非機能要件の分析

活動方針の「Rコース/Lコースを確実に走破する」および「バーコードと仕様未確定エリアⅡを確実に走破する」をトップゴールとしてD-Caseを用いて非機能要件の分析を行った。(エビデンスの詳細については図形内に記載してある各ページを参照)

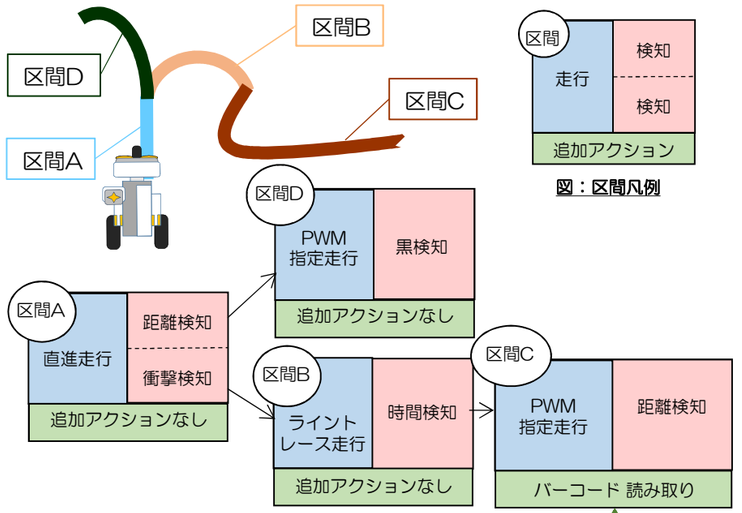
表：D-Case凡例

名称	図形	説明
ゴール		システムが達成すべき性質
ストラテジ		親ゴールを達成するアプローチの観点
コンテクスト		ゴールやストラテジを議論する前提情報
エビデンス		ゴールが成り立つことの証跡



P1-5：基本攻略の検討

これまでに抽出した機能要件および非機能要件の分析結果から、下記の基本攻略を検討した。
- コースをその場所の特性に応じて『区間』という単位に分割する
- それぞれの区間では1つの「走行」と複数の「検知」を保持している
- 「検知」の結果によって次の区間が決定される
- 区間によっては「走行」「検知」の他に、「追加アクション」を実行することでバーコード読み取りやルート探索、区間生成を実行する



FUJI Xerox

ヒカリバクシンオー

変更に強い構造を実現！

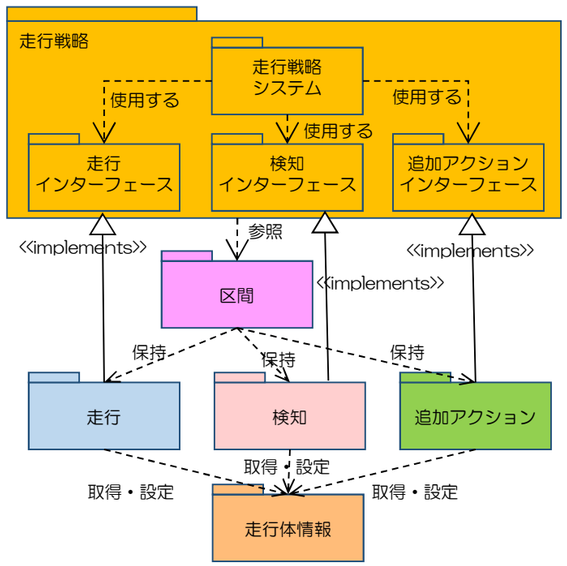
基本攻略の走行方法と要件を満たすために必要な構造の検討を行った。走行戦略が使用するインターフェースと各パッケージの実装を切り離すことにより、走行、検知、追加アクションに変更や追加があっても容易に対応が可能な構造とした。仕様変更などでバーコードと仕様未確定エリア以外の場所で何かの動作が必要になった場合にも任意のアクションが追加可能となっている。また、仕様未確定エリアでは動的に走行ルートを決定する必要があるため、走行体が扱う単位を区間として組み換えがしやすい構造とした。

P2-1：パッケージ構造

基本攻略の走り方(走行・検知・追加アクション)や要件を分析し、
各パッケージに分割した。

【特徴】
走行、検知、追加アクションのインターフェースを走行戦略パッケージで定義し、具体的なロジックやパラメータなどは走行パッケージなどで管理する構造とした。

走行戦略パッケージでインターフェースを定義することで、走行などのロジックの追加時にも走行戦略の構造や振る舞いに影響しない構造とした。使用するインターフェースも「初期化()」や「走行する()」、などのシンプルな設計(P2-2)とすることで、走行戦略パッケージが走行ロジックの実装に影響を受けなくなり、よりシンプルなステートマシンで実現できた。(P3-2.ステートマシン設計参照)



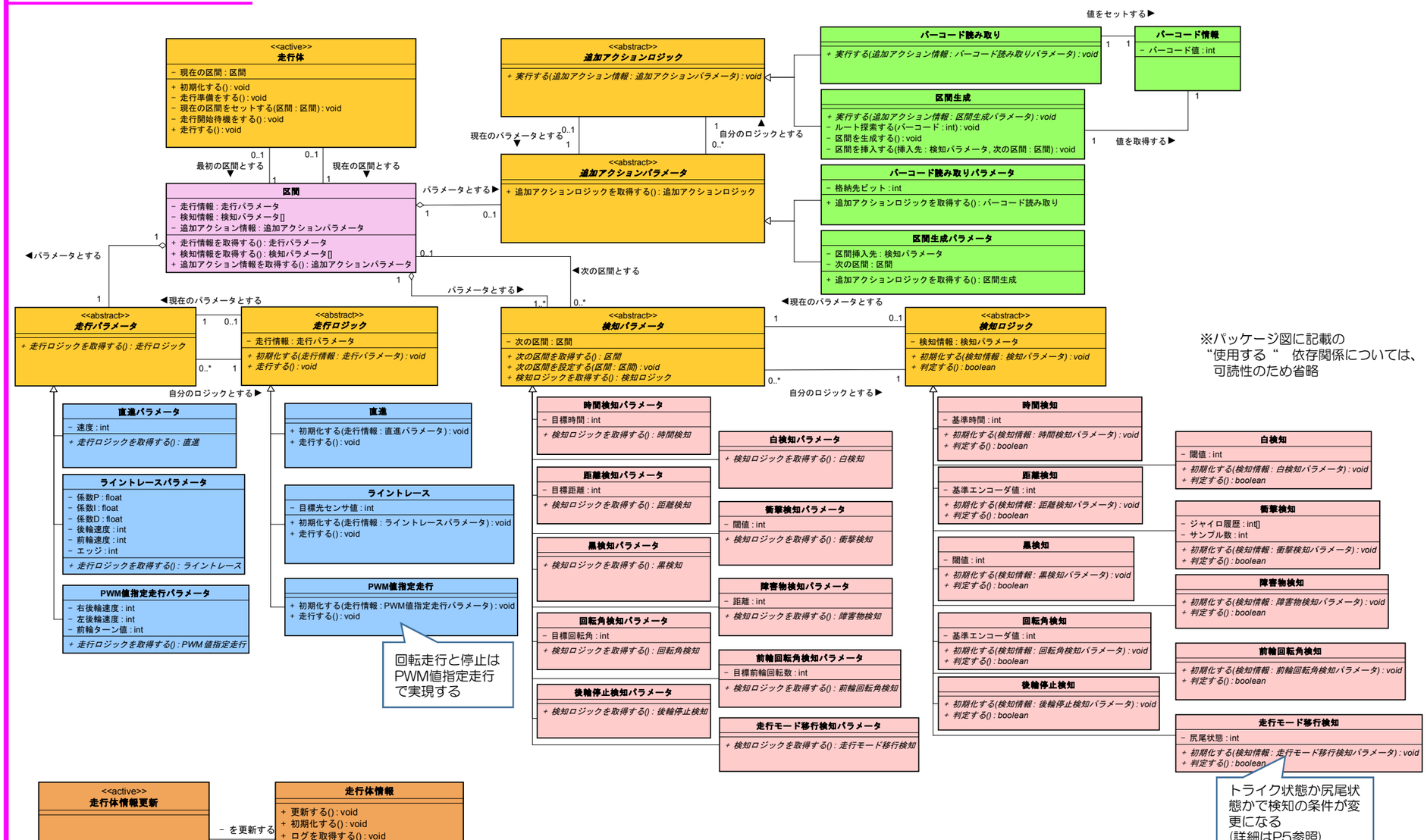
各パッケージと責務は以下のとおり。

表：各パッケージの責務一覧

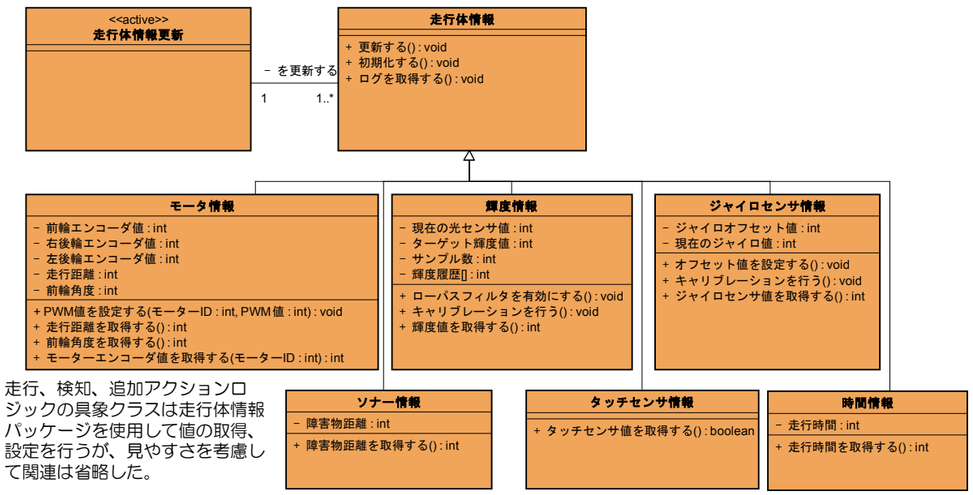
パッケージ		責務
走行戦略	走行戦略システム	走行・検知・追加アクションの実行区間の切り替えなど
	走行インターフェース	走行パッケージのインターフェース
	検知インターフェース	検知パッケージのインターフェース
	追加アクションインターフェース	追加アクションのインターフェース
区間		走行・検知・追加アクションを保持
走行		走行ロジックとパラメータを提供
検知		検知ロジックとパラメータを提供
追加アクション		追加アクションのロジックとパラメータを提供
走行体情報		デバイスに依存する情報を保持 情報を設定・取得する機能を提供 デバイスのキャリブレーションなど

P2-2：クラス図

パッケージ構造に基づいて、詳細な構造を分析し、クラス図を作成した。走行、検知、追加アクションの組み合わせを各パラメータのインスタンスと関連で表現した。



※パッケージ図に記載の
“使用する“ 依存関係については、
可読性のため省略



走行、検知、追加アクションロジックの具象クラスは走行体情報パッケージを使用して値の取得、設定を行うが、見やすさを考慮して関連は省略した。

ステレオタイプの説明：
<<active>>: P3-1でタスクに割りつけられるクラス
<<abstract>>: 走行戦略パッケージに定義される抽象クラス

クラス図(UML2.0準拠)

FUJI Xerox

ヒカリバクシンオー

シンプルな振る舞いで走行戦略を実行！

シンプルでわかりやすくするためにタスクの数や状態の数ができる限り少なくなるように検討した。シーケンスは基本的なシーケンスのみで区間に設定されている情報に応じた走行戦略をとることができるため、ベーシックコース、各難所、仕様未確定エリアを攻略できるようになっている。

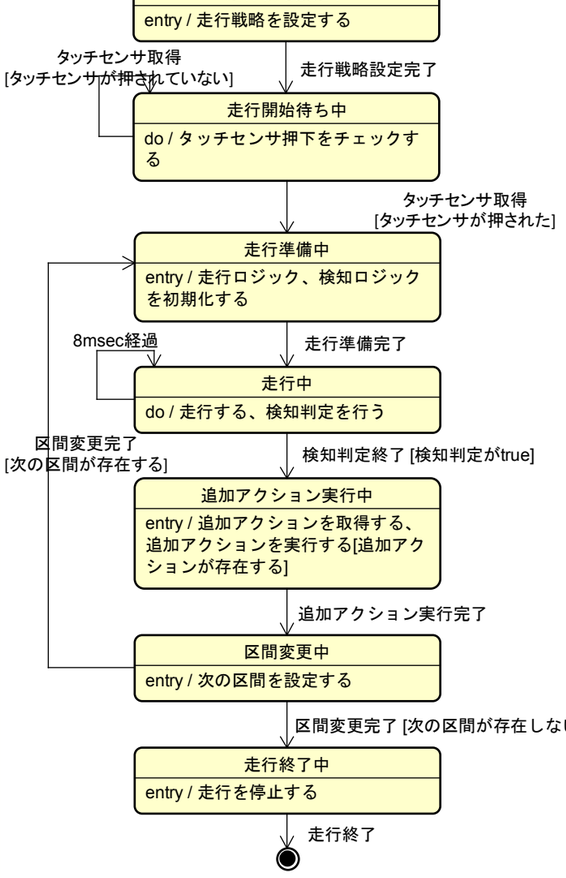
P3-1：タスク設計

走行の周期と走行体情報を更新する周期が異なるため、各種センサからの情報を取得する走行体情報更新のタスクと、シンプルに走行を行うための走行タスクの２つのタスクに分割した。
NXTWay-ETのように２輪倒立振子制御を行う必要がないため、走行ロジックで参照するパラメータも含め、すべてのセンシングパラメータは走行体情報更新タスクで更新を行うこととした。
各パッケージは最新の情報を使用することが望ましいため優先度を高とした。
走行タスクの周期はライントレースの安定度を保てる周期とした。

タスク名	割りつけるクラス	処理内容	優先度	周期
走行体情報更新タスク	走行体情報更新	走行体の各種センシングパラメータを更新する	高	4[ms]
走行タスク	走行体	走行および区間の切り替え	中	8[ms]

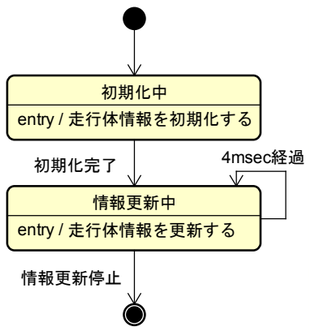
P3-2：ステートマシン設計

ステレオタイプ<<active>>のクラスについて、ステートマシン分析を行った。
単純な状態と遷移のみで区間の切り替えと戦略の実行を実現した。



ステートマシン図(UML2.0準拠)

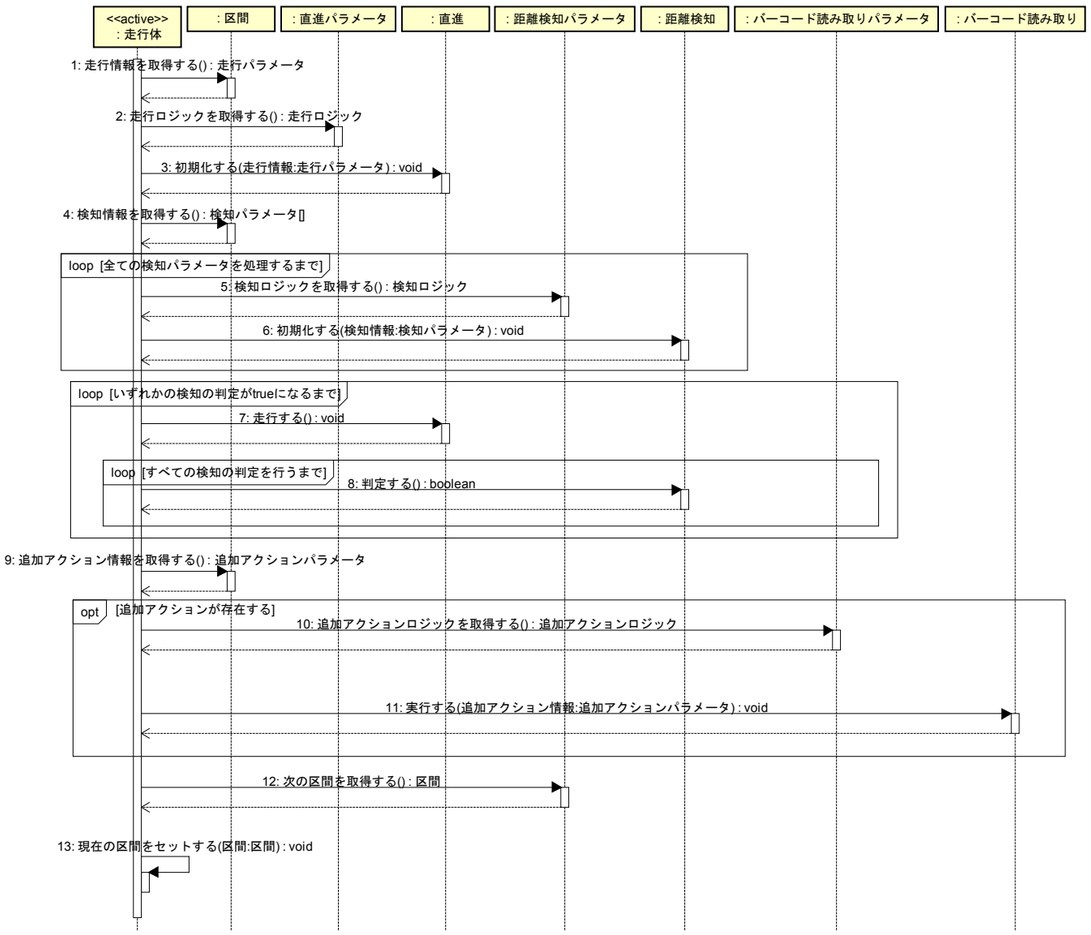
走行体情報更新クラス



P3-3：シーケンス設計

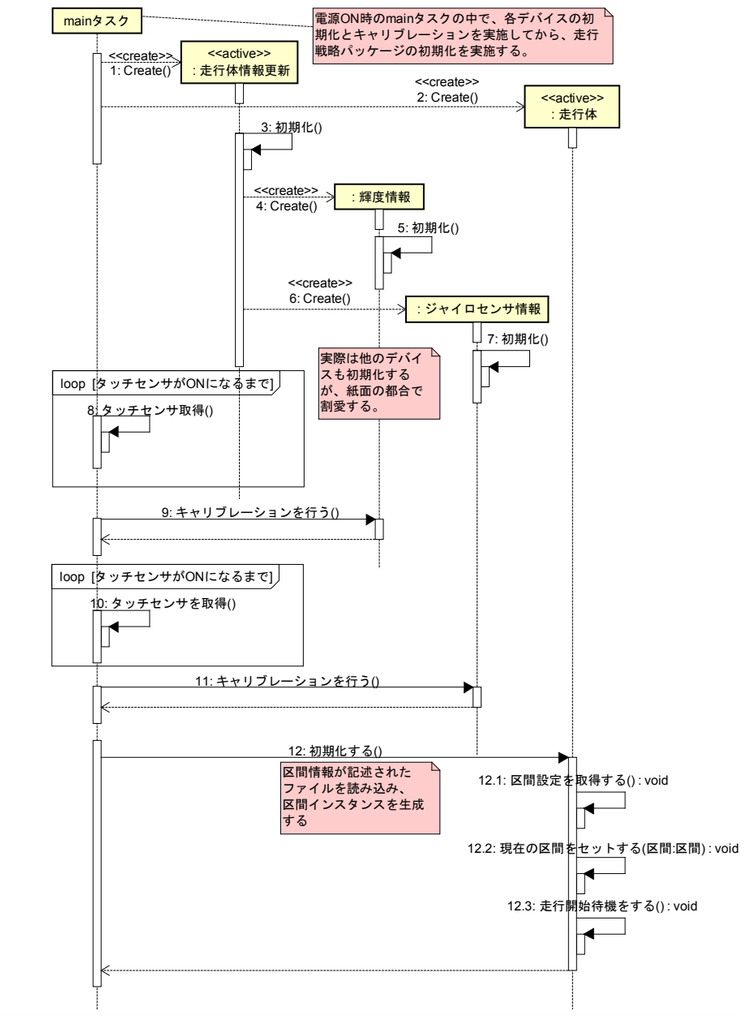
基本走行シーケンス

バーコードで読み取る区間を例として基本的な走行シーケンスを示す。各区間では、「走行」「検知」「追加アクション」を実行することでそれぞれの区間に対応した走行戦略で走行することができる。
また、仕様未確定エリアⅡについても同様の走行シーケンスで攻略することができる。



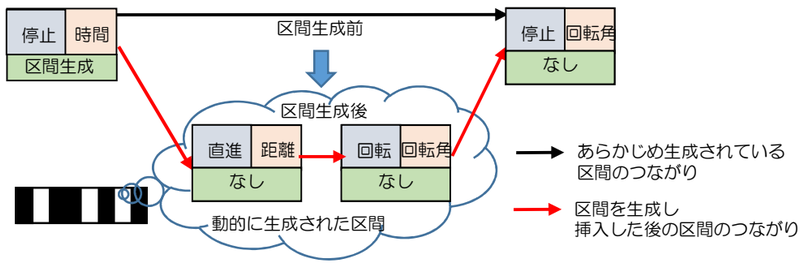
起動シーケンス

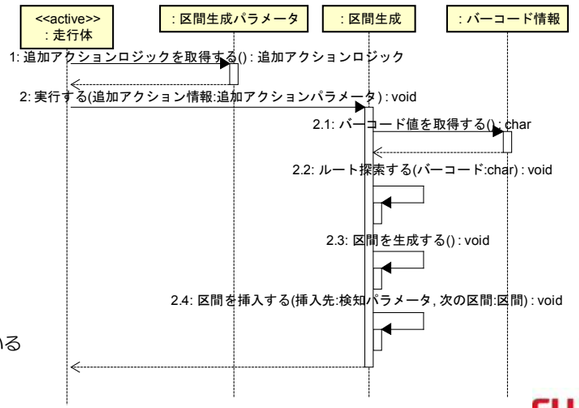
システム起動時のシーケンスを示す。各インスタンスの生成、初期化、キャリブレーションを行う。



区間生成シーケンス

区間生成時のシーケンスと振る舞いのイメージを示す。
区間生成では以下の4つを行う
- バーコード値取得：バーコード情報からバーコード値を取得する
- ルート探索：バーコード値を入力として障害物の位置を考慮した走行ルートを計算する
- 区間の生成：計算した走行ルートを区間に分割し、区間のインスタンスを生成する
- 区間の挿入：引数で渡された検知パラメータと次の区間との間に生成した区間を挿入する





シーケンス図(UML2.0準拠)

FUJI Xerox

ヒカリバクシンオー

区間を組み合わせることで確実にコースを攻略！

区間の走行、検知、追加アクションを組み合わせて各エリアを確実に走破する。
仕様未確定エリアⅡでは、確実に走破する走行と開発効率を両立した独自の攻略法を実現！

P4-1 読み方

各区間の走行、検知、追加アクションを下記の表記の通り記載。
パラメータは特徴的な値のみ記載。追加アクションを生成しない区間は表記を省略。区間の表記はP.1-5を参照。

走行	
表示	意味
PWM	PWM値指定走行(右,左,前)
LT	ライントレース
直進	直進走行
停止	PWM値指定走行(0,0,0)
回転(右)	PWM値指定走行(-60,60,0)
回転(左)	PWM値指定走行(60,-60,0)
共通走行	
表示	意味
段差昇段	段差を昇段
ライン探索	ラインを探索しエッジを検出
尻尾移行	尻尾走行へ移行
トライク移行	トライク走行へ移行
追加アクション	
表示	意味
読取	バーコードの読み取り
区間生成	仕様未確定エリアの区間生成

検知	
表示	意味
距離	距離検知
時間	時間検知
衝撃	衝撃検知
後輪	後輪停止検知
障害	障害物検知
回転	回転角検知(目標角)
前輪	前輪回転角検知(目標角)
尻尾	走行モード移行検知(尻尾→トライク)
トライク	走行モード移行検知(トライク→尻尾)
黒	黒検知
白	白検知

P4-2 ベーシックコース ～難所以外の攻略法！～

・凡例
LT (①、②)
①：前輪制御の有無
②：後輪の速度

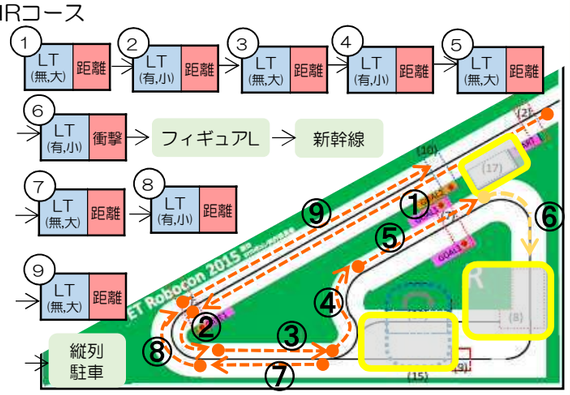
LT閾値

	黒用	グレイ用
右エッジ	→	→
左エッジ	→	→

■Lコース

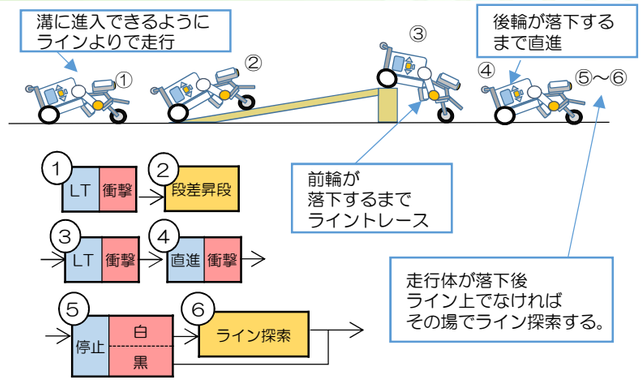
ベーシックコースは下図のように区間分けしてライントレースする。
前輪制御「有・無」、後輪速度「大・中・小」の三段階で表記している。

■Rコース

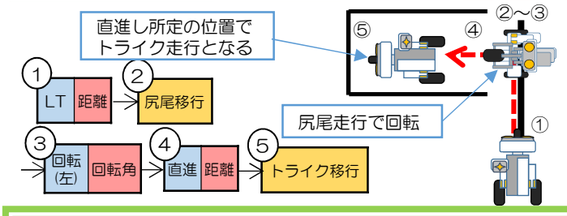


P4-3 難所 ～各難所ごとの攻略法！～

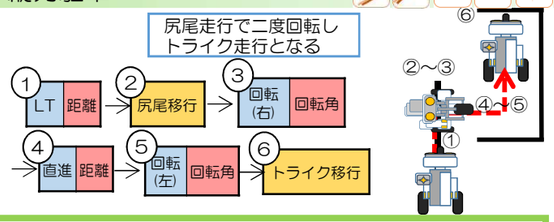
二本橋



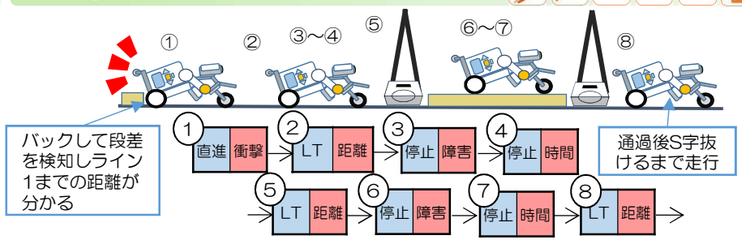
直列駐車



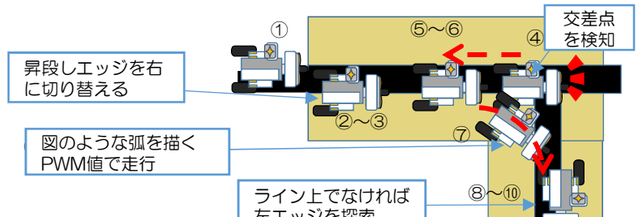
縦列駐車



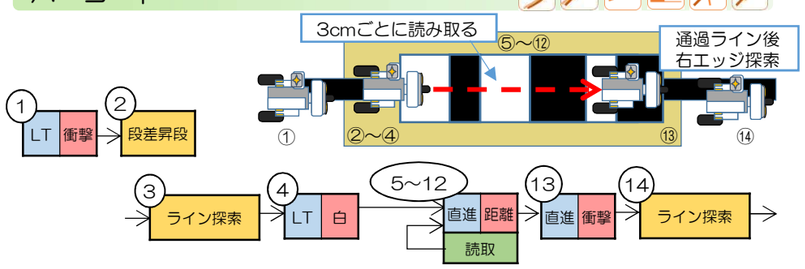
新幹線



フィギュアL

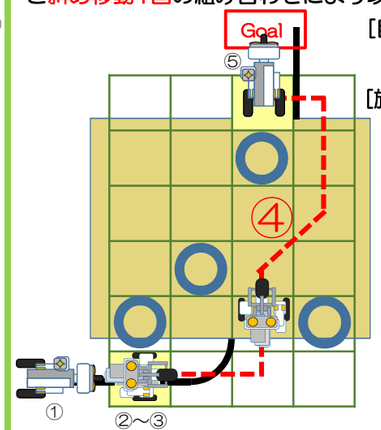


バーコード



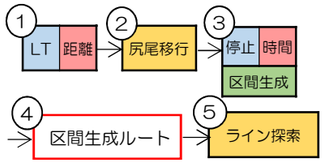
仕様未確定エリアⅡ

障害物の位置情報をもとにルート探索を行い、盤上は全パターン直進2回と斜め移動1回の組み合わせにより攻略可能！！



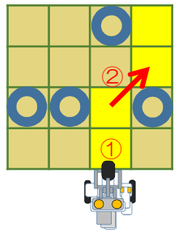
[目的]
どの位置に障害物が配置されても、仕様未確定エリアⅡを確実に攻略できる。

[施策]
1. エリア(盤上)への昇段および降段時は、転倒を防ぐため斜め移動を禁止
2. 障害物の接触を少なくするため尻尾走行で移動
3. エリア内ではP5に記載のルート探索を用いて、直進2回、斜め移動1回の走行パターンでクリア
4. ライン復帰を行うため、走行ラインまで走行



●なぜ直進2回、斜め1回でクリアできる？
①P5に記載のルート探索により、昇段に最適な列が選択→P5を参照

②格子の各列に対して必ず１つしか障害物が配置されない。よって、自身の位置に対して真横に障害物が配置されているときに斜め方向に進むことで残りは直進で攻略できる。



●効率的なテスト！
盤上のテスト
・板上は直進2回と斜め1回のみ
→ 左右の斜めがあるため、盤上は3x2通りのテストで網羅可能！
盤上以外のテスト
・start～段差昇段までの各列(4パターン)
・段差降段～ライン探索までの各列(4パターン)

盤上のテストおよび盤上以外のテストで仕様未確定エリアⅡの品質を担保！
切り離してのテストが可能なのでテストも簡単！

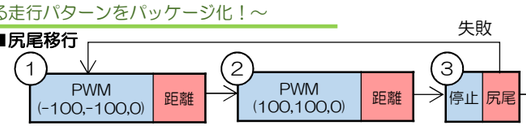
P4-4 共通走行 ～各難所で使用する走行パターンをパッケージ化！～

走行モード移行

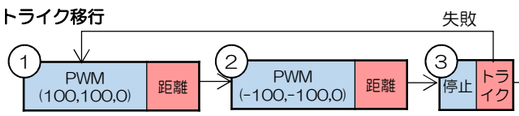
[目的]
トライク走行から尻尾走行、尻尾走行からトライク走行にモードを移行する

[施策]
尻尾移行は、最大速度でバックしてから(①)急激に前進へ切り替えることで(②)、慣性の力を利用して尻尾走行に移行する。
トライク移行は、逆に前進してから(①)バックすることで(②)トライク走行に移行する。

■尻尾移行



■トライク移行

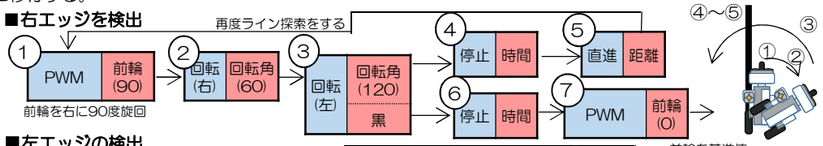


ライン探索

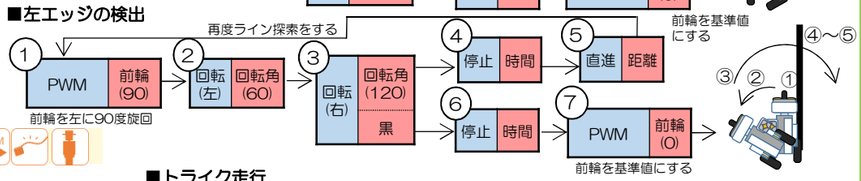
[目的]
ライン(エッジ)を探索する

[施策]
トライク走行で左右60度回転しラインを探索する。ライン(黒)を検知するとその場で停止する。捕えたいエッジにより動作が異なる。

■右エッジを検出



■左エッジの検出

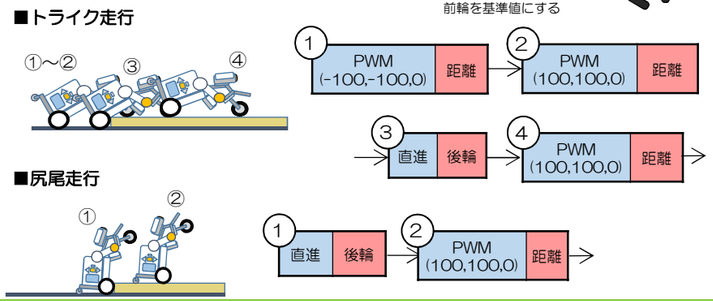


段差昇段

[目的]
安定した段差の昇段を実現する

[施策]
・トライク走行
前輪の遊びにより昇段時にラインから外れることがあるため、前輪を一瞬浮かせて段差にのせた後、後輪を段差に密着させ段差を昇段する。なお、尻尾走行にならない程度バックし(①)前進する(②)ことで前輪を浮かせることができる。
・尻尾走行
トライク走行時同様後輪を段差に密着させ段差を昇段する。

■トライク走行



■尻尾走行

ヒカリバクシンオー

「確実に走破する」ための各要件に対応した要素技術！

ギア比選定

[説明]
制限時間内にコースを確実に走破する最適なギア比を選定する

表1：ラインとレースと段差昇段走行検証結果

	ライントレース箇所		段差昇段走行時の左右モータのPWM指定値		
ギア比	直線	カーブ	40	50	100
1:1	○	○	×	○	○
1:3	○	×	×	×	×
1:5	×	×	×	×	×

[施策]
1: ギア比毎のライントレースの安定性とトライク状態での段差昇段走行を検証(表１)
2: コースの難所クリアに必要な平均所要時間とベーシックコースに割ける時間を算出(表2)

施策１の検証結果より、確実にコースを走破するためにはギア比１:1が最も最適であることを確認した。

表２ではギア比1:1の時の各難所の攻略所要時間を示す。

表2：難所攻略所要時間（ギア比1:1の時）

コース	難所	平均所要時間[s]
R	Figure L	12.8
	新幹線エリア	11.2
	縦列駐車	8
	ベーシックR	79
L	二本橋	8
	仕様未確定エリア	22
	バーコード	14
	直列駐車	10
	ベーシックL	56

RコースのベーシックRに割ける時間は、制限時間120[s]から各難所の平均所要時間を引いた88[s]。
ベーシックRの検証結果では走行時間は79[s]、最遅でも85[s]以内であればRコースをクリアできることを確認出来た。同様にしてLコースのベーシックLに割ける時間は66[s]、最遅でも60[s]以内であればLコースをクリアできることを確認出来た。
以上より、安定的にかつ制限時間内に走破可能なギア比1:1を選定。

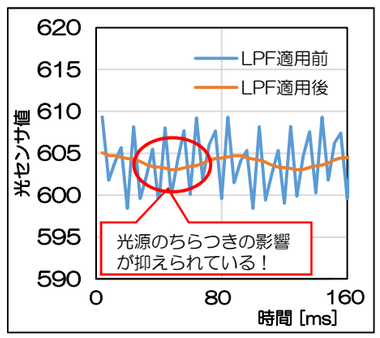
外乱光対策

[説明]
蛍光灯や水銀灯等の光源のノイズによる影響を低減させ、安定したライントレースを可能にする

[施策]
ローパスフィルタを導入して光の急激な変化を抑止する。
以下の式より輝度値を算出する。

輝度値 = (x[s]+x[s−1]+x[s−2]+x[s−3]+…+x[s−(n−1)])/n

x[s]: 光センサの値
s: 現在のサンプリング
n: 平均を取る回数



図：ローパスフィルタ適用結果(n=10)

右図にローパスフィルタ適用結果を示す。
ローパスフィルタの平均を取る回数(n)は、外乱光のちらつきを抑え、ライントレースのライン追従性に影響が出ないn=10に決定した。

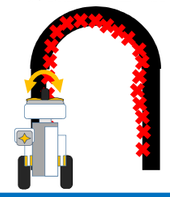
ライントレース

[説明]
どんなカーブでも滑らかなライントレースを行なう

[施策]
後輪のPID制御および前輪のOFF/ON制御を組み合わせて実現することで、滑らかな走行を維持しつつ極端なカーブも走行可能となる

前輪のOFF/ON制御
理由：前輪はOFF/ON制御だけでも、精度の良い走行が可能。PID制御を使用するよりもテスト工数が削減できるため、開発効率の向上が図れる

右エッジでのOFF/ON制御場合
輝度値＞閾値(ラインの内側)：前輪を右に旋回
輝度値＜閾値(ラインの外側)：前輪を左に旋回
輝度値＝閾値：前輪を旋回させない



輝度値より前輪をOFF/ONに切り替えることでラインに追従した走行が可能

✖光センサの軌跡

後輪のPID制御
理由：後輪ではラインに追従した滑らかな走行を必要とするため、PID制御で操作量の極端な変化を抑える

PID制御値 $= K_P e + K_I \int_0^t e\,dt + K_D \frac{d}{dt} e$

K_P：P制御ゲイン
K_I：I制御ゲイン
K_D：D制御ゲイン
e：制御偏差

本ライントレースではPID制御値はPWM値
偏差eは、目標輝度と取得輝度値の差であるまた、各ゲイン値は実験的に最も追従性の高い値を選択している

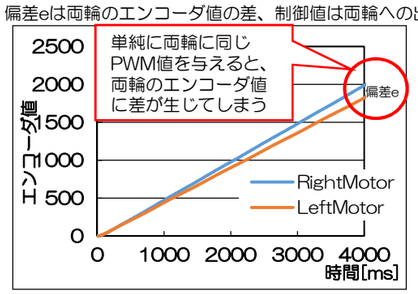
使用するパラメータは、
K_P=0.3333　K_I=0　K_D=0.01

直進走行

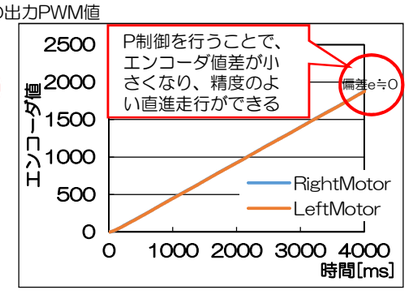
[説明]
走行体を滑らかに直進させる

[施策]
左右のモータのエンコーダ値の差(偏差e)が小さくなる（ゼロに近づける）ようP制御を行なう

偏差eは両輪のエンコーダ値の差、制御値は両輪への出力PWM値



図：P制御なし



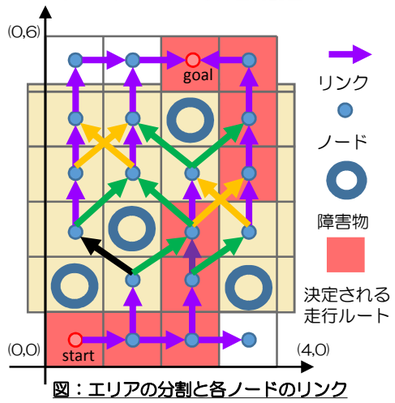
図：P制御あり

ルート探索

[説明]
仕様未確定エリアⅡの障害物の情報から走行ルートを決定する

[施策]
※ルート探索アルゴリズムA*を用いて転倒リスクが少なく最短でエリアを攻略できるルートを決定する

仕様未確定エリアⅡへの適用方法
仕様未確定エリアⅡを拡張して縦横6×4マスに分割し、障害物が無いマスにノードを割り当てる。
スタート地点とゴール地点を座標(1,1)、(3,6)にそれぞれ割り当てる。
P4記載の走行パターンに従ってノード間にリンクを設定する。
各リンクの移動コストを移動パターンの特徴に合わせて4種類設定する。
直進2回、斜め移動1回の組み合わせで盤上を攻略でき、障害物衝突による転倒のリスクが最も低くなるように下表のとおり移動コストを設定した。
これによって、どのような障害物配置のパターンに対しても安全にエリアを攻略できるようになった。



図：エリアの分割と各ノードのリンク

表：移動パターンと移動コスト

移動パターン	矢印の色	移動コスト
①直進	→	低
②斜め移動1(障害物なし)	→	中
③斜め移動2(片方障害物あり)	→	高
④斜め移動3(両方障害物あり)	→	最高

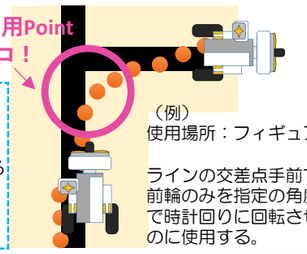
※A*はある地点から別の地点に移動する際の移動コストをあらかじめ設定しておくことで、合計の移動コストが最も低くなるような経路を選択するアルゴリズム。

PWM値指定走行

[説明]
走行体の前輪、左右の後輪に特定の動きをさせる

[施策]
前輪、左右の後輪のPWM値を直接指定する。

前輪　：車輪の向きが変わる
指定値：＋　時計回り
　　－　反時計回り
後輪　：車輪の回転向きが変わる(走行体の向きが変わる)
指定値：＋　時計回り
　　－　反時計回り



（例）
使用場所：フィギュアL

ラインの交差点手前で前輪のみを指定の角度まで時計回りに回転させるのに使用する。

距離検知

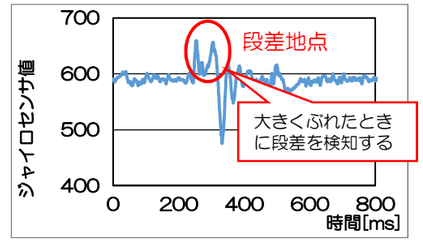
[説明]
走行体の走行距離を検知する

[施策]
後輪モータの回転角度から以下の式より走行距離を算出する。
走行距離 =左右後輪のエンコーダ平均値×π×タイヤの直径/360

衝撃検知

[説明]
走行体が段差にぶつかった、あるいは段差から下りた際に起こる衝撃を検知する

[施策]
段差にぶつかる、あるいは段差から下りた際はジャイロセンサ値が大きく変動する。
ジャイロセンサの値は、過去10回分の移動平均を保持しており、スピードに応じた閾値を超えたことで段差にぶつかる、あるいは段差から下りたことを検知することができる。

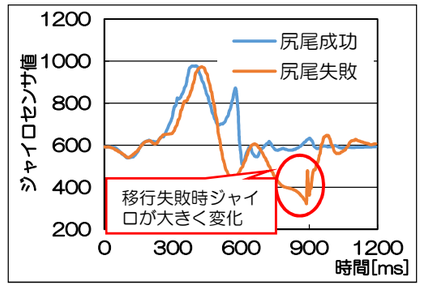


図：段差地点のジャイロセンサ値

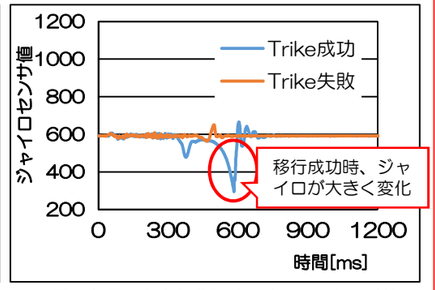
走行モード移行検知

[説明]
走行体がトライク走行から尻尾走行に移行したこと、または尻尾走行からトライク走行に移行したことを検知する

[施策]
トライク走行から尻尾走行に移行失敗時、また、尻尾走行からトライク走行に移行成功時、ジャイロセンサの値が正常動作時の約600から大きく下に振れる。ジャイロセンサ値が一定期間（最大900ｍｓ）の間にこの閾値を超えたか否かで成功の可否を判断することができる。



図：尻尾走行移行時のジャイロセンサ値



図：トライク走行移行時のジャイロセンサ値

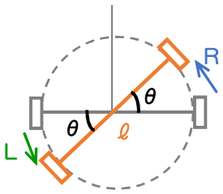
回転角検知

[説明]
その場での任意の回転角を検知する

[施策]
検知開始前の左右の後輪のエンコーダ値を基準とし、左右の後輪の回転角を検知する。
右の計算式より、左右後輪のエンコーダ値の差から回転角を算出する

$\theta = (|R-L|)/2\pi\ell \times 360$

θ：回転角[°]
ℓ：左右後輪の車軸の長さ[cm]
R：後輪右側のエンコーダ値から算出した距離
L：後輪左側のエンコーダ値から算出した距離



障害物検知

[説明]
障害物を検知する

[施策]
走行体の正面に障害物があると、超音波センサの値が変動する。
超音波センサ値が閾値を超えた場合に障害物があると判断することで障害物検知ができる。

黒検知／白検知

[説明]
ラインの黒色、または白色を検知する

[施策]
白 → 黒ラインに移動すると光センサの値が変動する。
光センサの値が閾値を超えた場合に黒ラインにいると判断することでラインを検知できる。
閾値を超えない場合は白と判断する。

後輪停止検知

[説明]
後輪が停止していることを検知する

[施策]
左右の後輪のエンコーダ値が一定時間の間にどれだけ変化しているかを確認する。
エンコーダ値が一定時間変化しなかった場合、停止していると判断する。

前輪回転角検知

[説明]
前輪の回転角を検知する

[施策]
走行開始前の前輪のエンコーダ値を基に、前輪の回転角を検知する。

FUJI Xerox