1. 要求分析

追跡線隊レッド

目標 **リザルトタイム -30秒を達成する**

★ 要求から機能と要件、制御戦略、要素技術、ソフトウェア構造につなげるトレース情報を **“要件定義+戦略トレーサビリティマトリクス”** です。

要求分析のポイント

① 目標から3つの要求に詳細化　② 3つの要求から機能としてユースケースを定義　③ ユースケースを実現する要件（要求仕様）を定義　④ 要求仕様を実現する制御戦略、要素技術、ソフトウェア構造を検討
［要件定義の表記には、清水吉男氏が考案したUSDM（Universal Specification Describing Manner）を適用］

ソフトウェア構造におけるレイヤ、パッケージ、クラスを列挙し、各クラスと要求仕様との対応を示す

要件定義			機能	制御戦略						要素技術									基盤技術			
		要求／要求仕様	ユースケース	基本走行	駐車場	二本橋	フィギュアL	新幹線	仕様未確定エリアⅡ攻略	定常円旋回	Advanced PID	バーコード読み取り	光センサ値の補正	自己位置推定	ライン復帰	後輪停止検知	段差昇段	ステアリング角度最適化	光センサ値取得	ソナーセンサ値取得	バッテリ電圧取得	時間計測
<ボーナスタイム獲得要求>																						
要求	REQ01	全6箇所の難所を攻略したい																				
	理由	6箇所の難所に設定されたボーナスタイム合計155秒を獲得するため																				
	説明	難所ごとに考案した制御戦略を実現させ、各難所を攻略する																				
		<二本橋を攻略する>	●																			
□□□	REQ01-01	段差接触での後輪停止から二本橋到達を判断して、前輪を浮かせ橋上に昇る				●											●	●					
□□□	REQ01-02	橋に昇った後、橋上を一定距離ライントレースする				●					●			●								
□□□	REQ01-03	降段後、バック走行して二本橋の縁に後輪を合わせる				●											●						
□□□	REQ01-04	バック走行後、ラインを見失っていた場合、ラインを探索して、ライン上に復帰する				●										●							
		<仕様未確定エリアⅡを攻略する>	●																			
□□□	REQ01-05	バーコードを読み取って、障害物の配置を把握する							●			●										
□□□	REQ01-06	把握した障害物配置に応じて走行経路を決定する							●			●										
□□□	REQ01-07	直進、旋回を組み合わせて、決定した経路を走行する							●	●				●								
□□□	REQ01-08	障害物エリア攻略後、ラインを探索をしてライン上に復帰する							●						●							
		<フィギュアLを攻略する>	●																			
□□□	REQ01-09	前輪を段差に接触させフィギュアＬ到達を判断し、前輪を浮かせライン右側に昇段する					●									●	●					
□□□	REQ01-10	昇段後、ライントレース走行を行い、T字路を検知して停車する					●				●								●			
□□□	REQ01-11	T字路検知後、一定距離バック走行して、板の端からの脱輪を防ぐ					●						●								
□□□	REQ01-12	バック走行後、右に55deg旋回する					●						●								
□□□	REQ01-13	右旋回後、ライントレース走行を行い、段差を降りる					●				●			●								
		<新幹線を攻略する>	●																			
□□□	REQ01-14	フィギュアL出口に後輪を合わせて、初期位置を決定する						●								●						
□□□	REQ01-15	新幹線前の区間をライントレース走行し、新幹線検知位置まで移動して停止する						●			●			●								
□□□	REQ01-16	新幹線検知位置で停止後、ソナーセンサで新幹線を検知する						●											●		
□□□	REQ01-17	検知後、新幹線が通り過ぎるまで停止する						●													●
□□□	REQ01-18	新幹線が通過した後、新幹線と接触せずに線路を通過する						●						●								
□□□	REQ01-19	初めの線路を通過後、段上をライントレース走行し、新幹線検知位置で停止する						●			●			●								
		<駐車場を攻略する>	●																			
□□□	REQ01-20	走行位置を把握して、駐車動作の開始位置を決定する			●									●								
□□□	REQ01-21	駐車開始位置に到達後、目的の位置に駐車する			●									●								
<走行タイム確保要求>																						
要求	REQ02	Rコースを45秒、Lコースを80秒で走行させたい																				
	理由	難所6箇所のボーナスタイム合計155秒と合わせてリザルトタイムを-30秒を達成するため																				
	説明	難所の攻略や折り返しにかかる時間を考慮し、区間ごとの目標タイムを設定する																				
		<Rコースを45秒で走行する>	●																			
□□□	REQ02-01	ライントレース走行を行い、第1ゴールゲートRをスタートから20秒以内に通過する		●							●								●				
□□□	REQ02-02	第1ゴールゲートRを通過後、折り返す		●										●	●							
□□□	REQ02-03	ライントレース走行を行い、折り返し後、20秒以内にゴールゲートを通過する		●							●								●				
□□□	REQ02-04	ゴールゲート通過後、フィギュアLまで30秒以内に到達する		●							●								●				
□□□	REQ02-05	Rコースを区間に分割して管理し、走行区間に応じて走行戦略を切り替える		●										●								
		<Lコースを80秒で走行する>	●																			
□□□	REQ02-06	ライントレース走行を行い、スタートから15秒以内に二本橋まで到達する		●							●								●				
□□□	REQ02-07	Lコースの二本橋から仕様未確定エリアⅡを50秒以内に攻略する		●							●								●				
□□□	REQ02-08	仕様未確定エリアⅡを攻略後、15秒以内にゴールゲートを通過する		●							●								●				
□□□	REQ02-09	Lコースを区間に分割して管理し、走行区間に応じて走行戦略を切り替える		●										●								
<走行安定性要求>																						
要求	REQ03	スタートからゴールまで走行の安定性を確保したい																				
	理由	不安定な走行でコースアウトしてしまうと、リザルトタイム-30秒の達成が不可能になるため																				
	説明	環境やバッテリ残量に依存せず、安定した駆動でライントレースを行い、高い完走率を実現する																				
		<キャリブレーションをする>	●																			
□□□	REQ03-01	ライン上の黒い部分の光センサ値を取得する																	●			
□□□	REQ03-02	コース上の白い部分の光センサ値を取得する																	●			
		<環境に依存しない走行をする>	●																			
□□□	REQ03-03	取得した白と黒の光センサ値の偏差を正規化する											●						●			
□□□	REQ03-04	正規化した光センサ値から閾値を決定する											●									
		<バッテリに依存しない走行をする>	●																			
□□□	REQ03-05	バッテリ残量に依存しないモータ出力を決定する																			●	
		<安定したライントレースする>	●																			
□□□	REQ03-06	光センサの閾値と取得値の偏差から後輪のモータ出力をPID制御する									●											
□□□	REQ03-07	後輪のモータ出力比から前輪のステアリング角度を決定する															●					

ストラテジ・エグゼキューション

要求仕様	車体制御	車体状態制御	コース管理	キャリブレータ	Lコース走行区間管理	Rコース走行区間管理	走行戦略	二本橋攻略	フィギュアL攻略	縦列駐車攻略	直角駐車攻略	新幹線攻略	折り返し攻略	バーコード攻略	仕様未確定エリアⅡ攻略
REQ01-01								●							
REQ01-02								●							
REQ01-03								●							
REQ01-04								●							
REQ01-05														●	
REQ01-06															●
REQ01-07															●
REQ01-08															●
REQ01-09									●						
REQ01-10									●						
REQ01-11									●						
REQ01-12									●						
REQ01-13									●						
REQ01-14												●			
REQ01-15												●			
REQ01-16												●			
REQ01-17												●			
REQ01-18												●			
REQ01-19												●			
REQ01-20										●	●				
REQ01-21										●	●				
REQ02-01															
REQ02-02													●		
REQ02-03															
REQ02-04															
REQ02-05		●				●									
REQ02-06															
REQ02-07															
REQ02-08															
REQ02-09		●			●										
REQ03-01				●											
REQ03-02				●											
REQ03-03															
REQ03-04															
REQ03-05															
REQ03-06															
REQ03-07															

メカニズム・コントロール

要求仕様	走行メカニズム	ライントレース走行	シナリオ走行	ライン復帰	格子走行	ウィリー走行	曲率指定走行	走行	検知・検出メカニズム	後輪停止検知	ライン検知	時間検知	前方障害物検知	自己位置検知	タッチ検知	光検出	制御メカニズム	ルート計算	バッテリ補正	バーコードリーダ	走行ルート決定
REQ01-01						●		●		●											
REQ01-02		●						●								●					
REQ01-03			●				●	●													
REQ01-04				●			●	●			●										
REQ01-05			●					●						●		●		●		●	
REQ01-06																					●
REQ01-07			●		●		●	●			●			●							
REQ01-08			●	●			●	●			●										
REQ01-09		●				●	●	●					●			●					
REQ01-10		●						●			●					●					
REQ01-11			●				●	●						●							
REQ01-12			●				●	●						●							
REQ01-13		●						●								●					
REQ01-14			●				●	●		●											
REQ01-15		●						●						●		●					
REQ01-16													●								
REQ01-17												●									
REQ01-18			●				●	●						●							
REQ01-19		●						●						●		●					
REQ01-20		●						●						●		●					
REQ01-21			●				●	●						●							
REQ02-01		●						●								●					
REQ02-02			●				●	●			●			●							
REQ02-03		●						●								●					
REQ02-04		●						●								●					
REQ02-05																					
REQ02-06		●						●								●					
REQ02-07		●						●								●					
REQ02-08		●						●								●					
REQ02-09																					
REQ03-01															●	●					
REQ03-02															●	●					
REQ03-03																●					
REQ03-04																●					
REQ03-05																			●		
REQ03-06		●														●					
REQ03-07		●																			

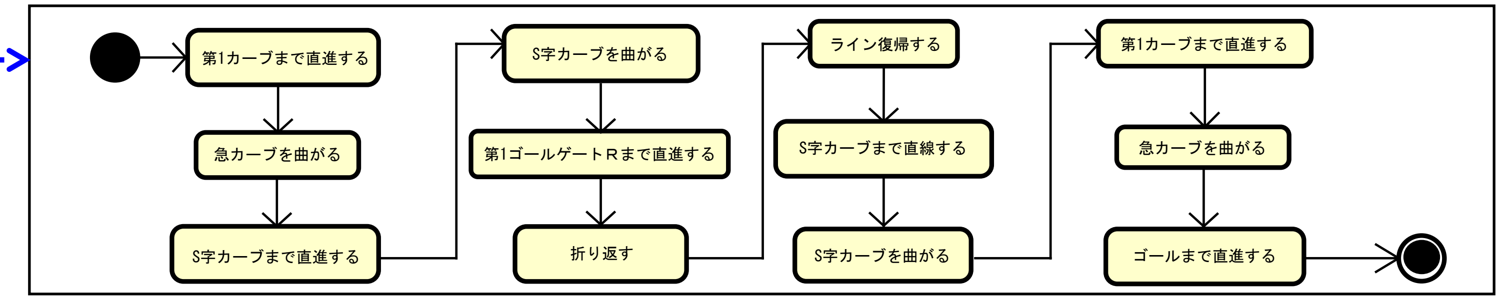
デバイス・アクセス

要求仕様	デバイス制御	デバイスマネージャ	バッテリ	ソナーセンサ	タッチセンサ	ディスプレイ	スピーカ	光センサ	サーボモータ	タイマ
REQ01-01		●					●		●	
REQ01-02		●					●	●	●	
REQ01-03		●					●		●	
REQ01-04		●					●	●	●	
REQ01-05		●					●	●	●	
REQ01-06		●					●			
REQ01-07		●					●	●	●	
REQ01-08		●					●	●	●	
REQ01-09		●					●	●	●	
REQ01-10		●					●	●	●	
REQ01-11		●					●	●	●	
REQ01-12		●					●		●	
REQ01-13		●					●	●	●	
REQ01-14		●					●	●	●	
REQ01-15		●					●	●	●	
REQ01-16		●		●			●			
REQ01-17		●					●			●
REQ01-18		●					●		●	
REQ01-19		●					●	●	●	
REQ01-20		●					●		●	
REQ01-21		●					●		●	
REQ02-01		●						●	●	
REQ02-02		●							●	
REQ02-03		●						●	●	
REQ02-04		●						●	●	
REQ02-05		●				●	●			
REQ02-06		●						●	●	
REQ02-07		●						●	●	
REQ02-08		●						●	●	
REQ02-09		●				●	●			
REQ03-01		●			●			●		
REQ03-02		●			●			●		
REQ03-03		●						●		
REQ03-04										
REQ03-05		●	●							
REQ03-06		●						●		
REQ03-07										

上記の要件定義の中で定義したユースケースのうち、代表として <Rコースを45秒以内で走行する> のユースケース記述を以下に示す。また、このユースケース実現の上で考えられるリスクを検討し、考察した対策方法を表1.1に示す。

<Rコースを45秒以内で走行する> のユースケース記述

ユースケース名	<Rコースを45秒以内で走行する>
アクタ	競技者
事前条件	Rコースの走行準備が終了し、スタートしている
事後条件	ゴールゲートを通過し折り返し動作を行う
イベントフロー(メイン)	右図参照
例外ケース	ライン形状によるライントレースの不安定化



<Rコースを45秒以内で走行する>のイベントフロー(メイン)をアクティビティ図で記述する。

<Rコースを45秒以内で走行する>実現の上で考えられるリスクを検討した。考えられるリスクは、直線からカーブへの切り替え時に直線のPIDパラメータでカーブを行うことである。そこで要素技術のAdvanced PIDを用いることでそのリスクの対策をとった。

表1.1 <Rコースを45秒以内で走行する>のリスク対策表

リスク	リスク発生の結果	対策
直線→カーブでライントレースに失敗する	失格、リタイヤ	Advanced PID
カーブ→直線でライントレースに失敗する	失格、リタイヤ	

本モデルで使用するUMLはすべてUML2.0に準拠している

2. 構造設計

ソフトウェア構成

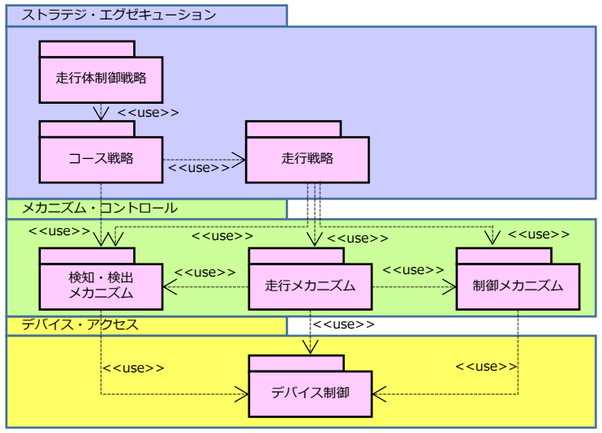
ソフトウェア構成の概要を、パッケージ図を用いて示す。

ソフトウェア構成の複雑さを軽減するため、以下の設計を行った。

【3層構造】
最上位のレイヤには、走行戦略を実行するための「ストラテジ・エグゼキューション」レイヤ、最下位のレイヤには、NXTのハードウェア制御情報にアクセスする「デバイス･アクセス」レイヤ、中間のレイヤには、各種の要素技術を実現させる「メカニズム・コントロール」レイヤを配備した。

【インタフェース構造】
走行戦略は、必要な検知・検出メカニズム、走行メカニズム、制御メカニズムを組み替えるだけでコースを走行することができる。

【汎化関係・コンポジット構造】
同型のクラスを抽象クラスでまとめ、全体と部分のクラスをコンポジット構造でまとめた。



パッケージ構成概要

3層（レイヤ）構造の各レイヤに配備したパッケージの責務について概要を示す。

3層（レイヤ）の責務を以下に示す。

1.ストラテジ・エグゼキューション

走行体制御やリザルトタイム-30秒を獲得するために必要な戦略を実行する。

2.メカニズム･コントロール

走行戦略を実現するためのメカニズムを制御する。

3.デバイス･アクセス

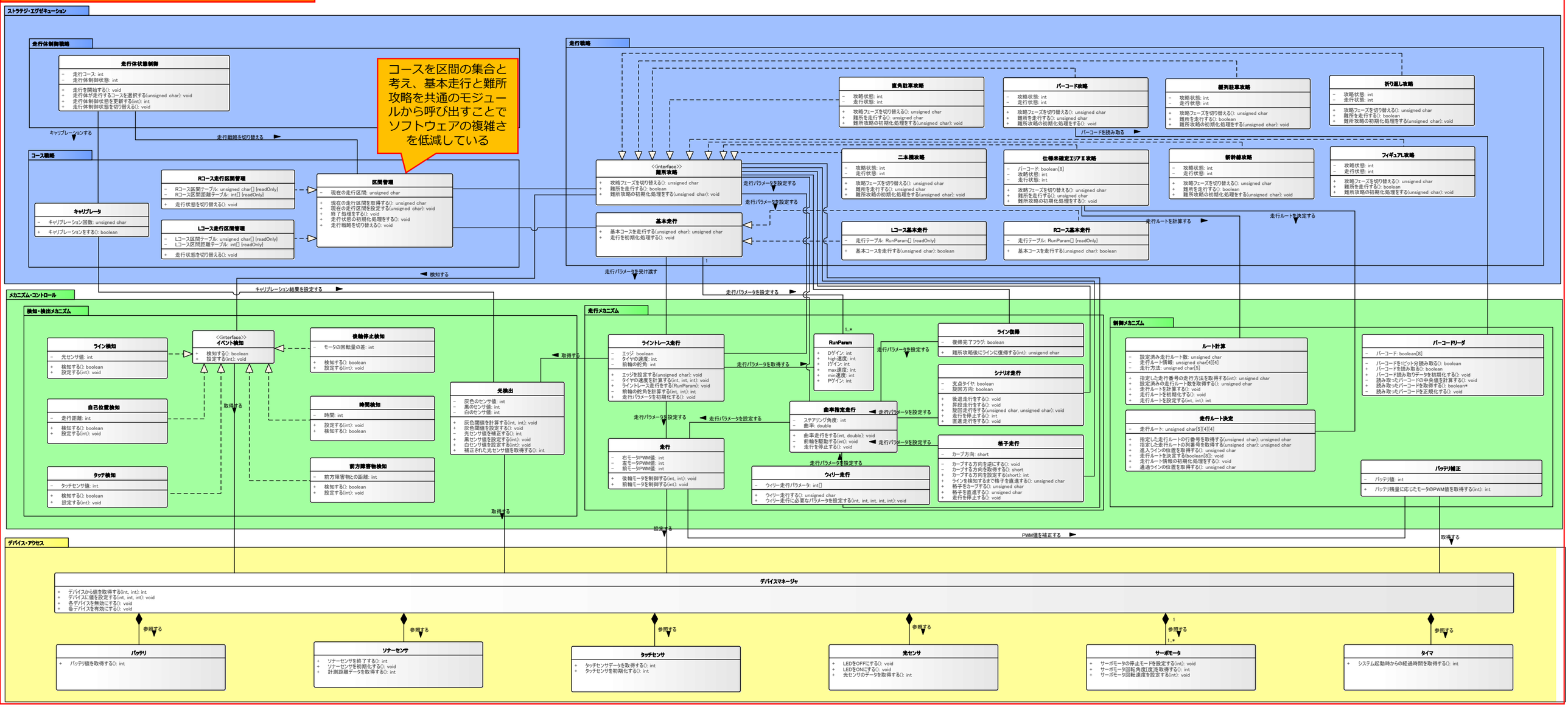
デバイスと間接的にやり取りをするインタフェースを持つ。

レイヤ構造におけるパッケージの責務

レイヤ	パッケージ名	責　務
ストラテジ・エグゼキューション	走行体制御戦略	走行体の走行状態を制御する
	コース戦略	走行戦略を切り替えてコースを攻略する
	走行戦略	基本走行と難所攻略の制御戦略を実行する
メカニズム･コントロール	検知・検出メカニズム	走行戦略に必要な情報を検知・検出する
	走行メカニズム	走行戦略に従った走行メカニズムを制御する
	制御メカニズム	走行戦略に従った制御メカニズムを実行する
デバイス･アクセス	デバイス制御	デバイス制御のためのデバイス情報を取得する

ソフトウェア構成の詳細

ソフトウェア構成の詳細をクラス図を用いて示す。　多重度1:1は省略する。



本モデルで使用するUMLはすべてUML2.0に準拠している

3. 振る舞い設計

ユースケース＜Rコースを45秒以内で走行する＞の振る舞い

P1.で定義したユースケース実現の振る舞いをシーケンス図で示す。

■ ユースケース実現のため、第1ゴールゲートR通過後に折り返しを行う。

■ 折り返し後、走行タイムを確保して難所攻略に挑戦する。

■ 本走行を実現するため、「直線」、「カーブ」、「S字カーブ」の単位でコースを区間に分割している。

★ 区間を分割したことで、各区間の最適なライントレースのためのPID値を計算し、走行タイムの短縮につなげている。

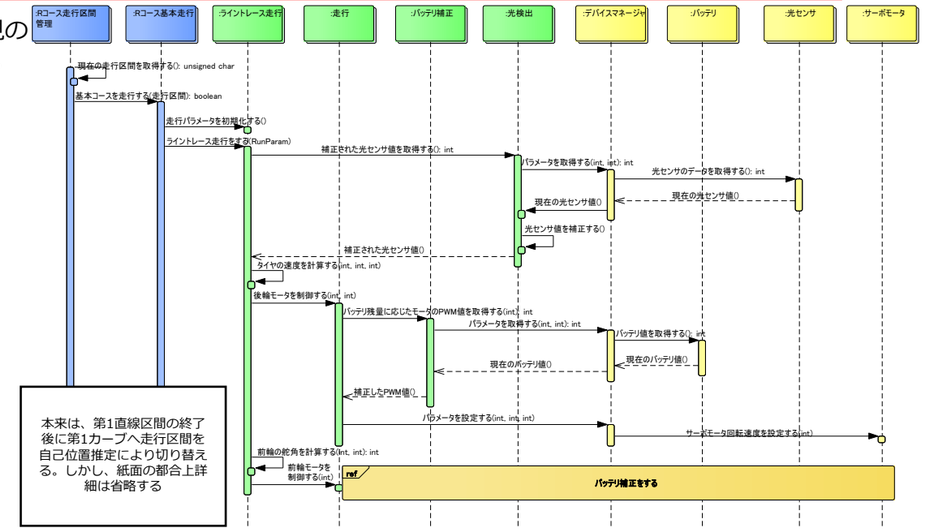


図3.1 ＜Rコースを45秒以内で走行する＞のシーケンス図

仕様未確定エリアⅡ攻略の振る舞い

仕様未確定エリアⅡにて読み取ったバーコードデータから走行ルートを算出し走行する。
ここで、算出したルートが走行不可能とならないよう、走行ルート決定の補償 [下記(1)を参照] を行う。

■ シーケンス図(右図)を用いて障害物エリアの走行方法を設計した。

■ 本項下段にて、走行ルート決定から障害物エリアを走行するまでの戦略を説明する。

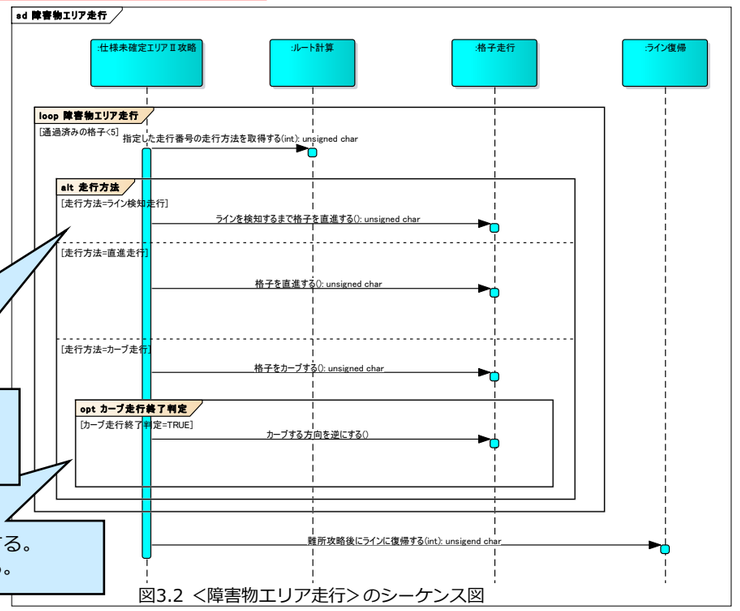


図3.2 ＜障害物エリア走行＞のシーケンス図

障害物配置にかかわらず、障害物エリアは５マス分の格子を通過するルートで攻略できる。
[下記(2.1)を参照]

直進走行、カーブ走行を格子ごとに切り替えて走行する。
走行方法は格子ごとにルート計算クラスから取得する。

仕様未確定エリアⅡ攻略の制御戦略

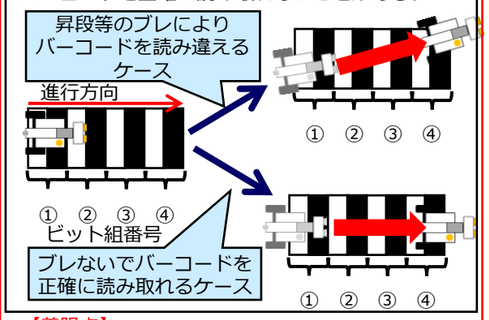
使用要素技術: 自己位置推定、段差昇段、後輪停止検知、バーコード読み取り、定常円旋回

(1) 走行ルート決定の補償

頑強な走行ルート決定のために走行ルート決定の補償を取り入れる

【問題点】

昇段時の走行体のブレや重心の偏りによりバーコードを正確に読み取れないことがある。

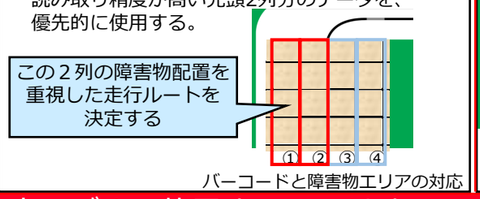


【着眼点】

・バーコードを読み違えても、前半部分 (下図①、②) では読み取り精度が高い。

【解決策】

読み取り精度が高い先頭2列分のデータを、優先的に使用する。



(2) 走行ルート決定方法

バーコード読み取り結果から障害物エリアを攻略するために障害物エリアの走行ルートを決定する

(2.1) 走行列の決定

バーコード読み取り結果から走行可能な走行列を決定

走行列決定手順

【攻略策】
障害物エリアの進入ラインから通過ラインまでの最小移動距離を下記に示す。
・左方向に格子1つ分
・下方向に格子4つ分

【着眼点】
・2列分の情報があれば走行ルートを決定できる。

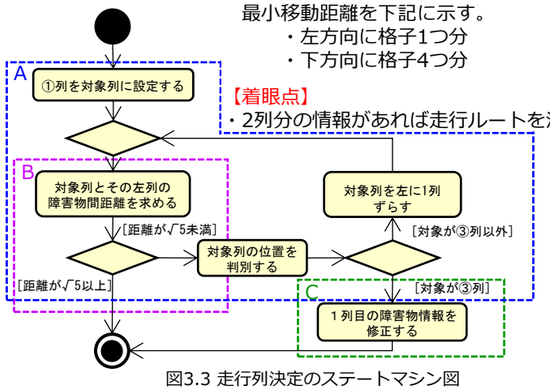
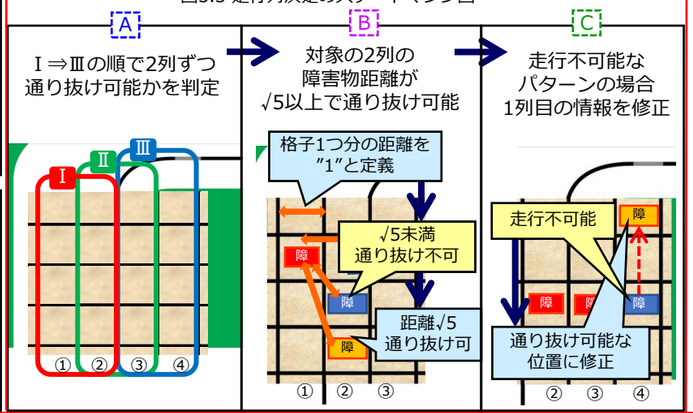


図3.3 走行列決定のステートマシン図



(2.2) 走行する格子の決定

障害物を避けて走るために走行する障害物エリアの格子を決定

走行する格子の決定手順

【攻略策】
(2.1)で決めた走行列を元に左図と下図のような手順で走行する格子を決定する。

【着眼点】
・進入列は、通過ライン側の障害物位置で決定できる。
・進入列側の障害物手前で横移動すれば障害物を避けて突破できる。

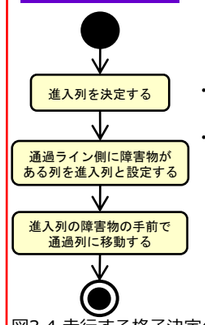
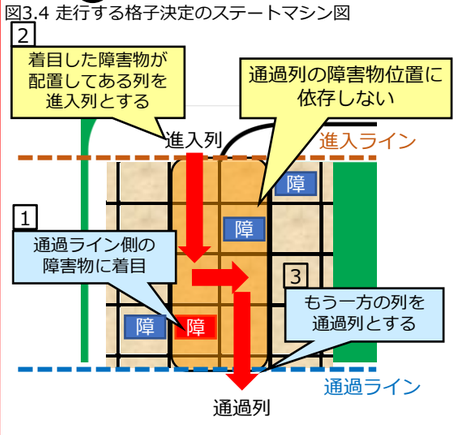


図3.4 走行する格子決定のステートマシン図

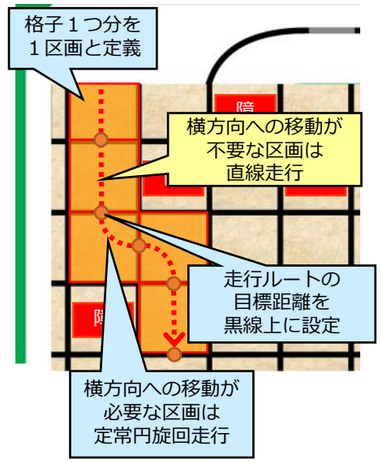


(2.3) 格子走行方法の決定

障害物エリアの格子の走行方法を決定

【攻略策】
(2.2)で走行する格子を決定した後、格子内の走行方法を決める。
直線に走行する格子内では直進走行、障害物を避けなければならない部分は定常円旋回走行を行う。

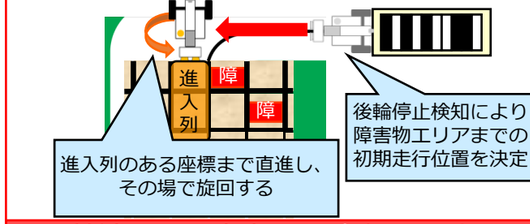
【着眼点】
・格子1つ分を区間とし、その1区間毎に走行方法を切り替える。
・直進と定常円旋回の走行方法を組み合わせれば突破が可能である。



(3) スタート位置までの移動

バーコードから障害物エリアまで走行する

【攻略策】
バーコード攻略後から自己位置推定を用いて、障害物エリアのスタート地点まで走行する。



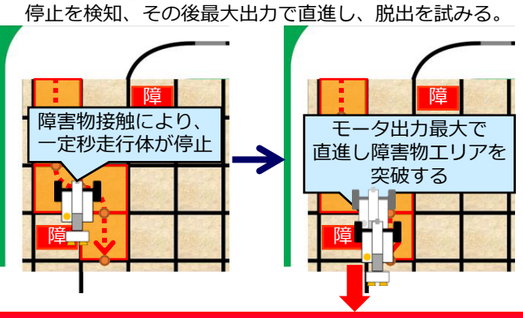
(4) 前述の2で決定した走行ルートを実際に走行する

ex.障害物接触時の緊急脱出

障害物によって停止してもボーナスタイムを狙う

【問題点】
決定した走行ルートを走行する際、下左図のように障害物間を曲がりきれず、接触し停止することがあった。

【解決策】
障害物に接触した場合、「後輪停止検知」で走行体の停止を検知、その後最大出力で直進し、脱出を試みる。



4. 制御戦略

基本走行と各難所を攻略するため、事前に「リスク抽出」を行い、その対策として各区間における制御戦略を立案した。

基本走行

使用要素技術: Advanced PID、自己位置推定、ステアリング角度最適化

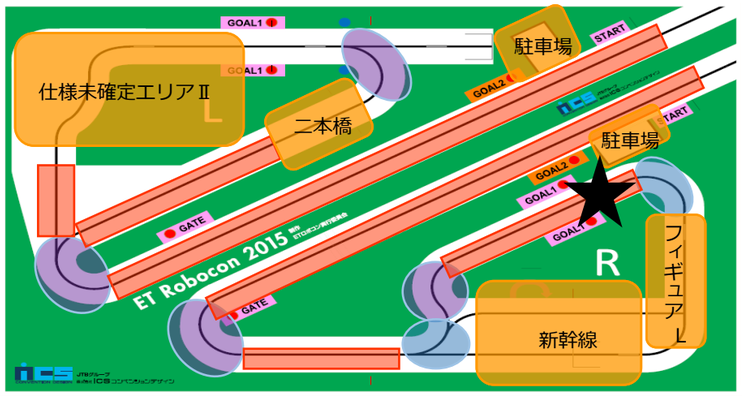
基本走行において、ラインの曲率などからコースを直進区間、カーブ区間、急カーブ区間の3領域に分類した。

Rコース45秒以内、**Lコース80秒以内という走行タイム**を獲得するため、それぞれの区間に最適な制御量を与え、コース形状に合った走行を行う。

Rコースでは、難所を経由してからゴールをめざすと、走行タイム45秒以内という要求を満たせなくなるため、**第１ゴールゲートR通過後折り返し**、走行タイムを獲得する戦略をとった。

	P	I	D	max	high	low
直進区間	小	小	0	100	100	84
カーブ区間	大	中	中	80	80	32
急カーブ区間	大	大	大	65	65	26

max：最高出力。光センサが理想の位置（閾値）上にある時の速度
high-low：左右輪の最高出力差。光センサが白 or 黒上にある時の速度差



駐車場

使用要素技術: 自己位置推定

旋回パラメータ

ステアリング角度　：67deg
左モータエンコーダ値：455
右モータエンコーダ値：100

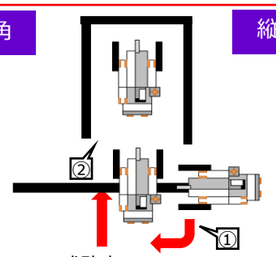
戦略

①旋回パラメータを用いて旋回する

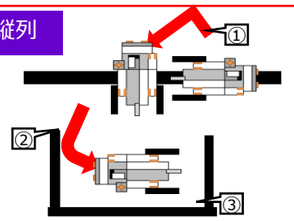
②49mmバック走行する（直角駐車は300mm）

③①と同様の旋回を行う

直角



縦列



成功率

27/30回成功したので、この戦略を採用した。

リスク	対策
目標の走行タイムを獲得するために速度を上げると、カーブに対応できずに脱線してしまう	コースを区間に分割して管理し、それぞれの区間に最適な制御量を与えることで、コースの形状に合った走行を可能にした
布を巻き込むことにより、走行体の回転が不足して壁にぶつかってしまう	その場で90deg回転して方向を変えるのではなく、大きく旋回することで、布を巻き込むリスクを削減した
板に溝があるため段差が大きく昇段ができない	前輪持ち上げを通常の昇段より大きくすることで前輪が溝にはまるリスクを削減した
橋から降りた後の位置が安定せず、ライン復帰が困難になってしまう	着地後バック走行し、ライン復帰開始位置を定位置にすることで復帰できないリスクを削減した
T字路検知の後、 走行体が板を曲がりきれずに転落してしまう	T字路検知後一定距離バック走行することで、前方の距離に余裕を作り、落ちるリスクを削減した
走行体が新幹線を検知後、直ぐに走行を開始すると接触してしまう	新幹線を検知後1秒間停車した後、走行を開始することで、接触のリスクを削減した

二本橋

使用要素技術: Advanced PID、自己位置推定、ライン復帰、後輪停止検知、 段差昇段

走行制御の状態遷移

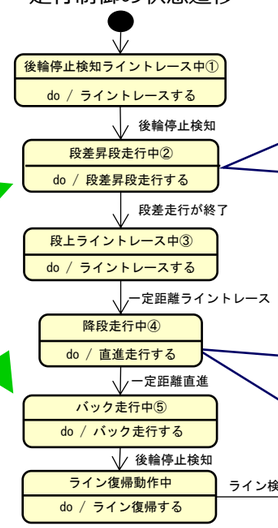
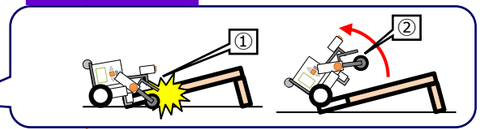
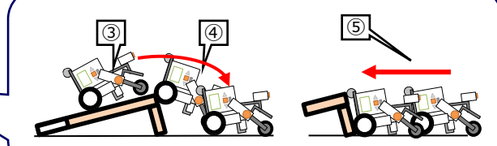


図4.1 走行制御のステートマシン図

昇段走行



バック走行で位置を決定



成功率

27/30回成功したので、この戦略を採用した。

フュギュアL

使用要素技術: Advanced PID、自己位置推定、後輪停止検知、 段差昇段

※T字路検知で旋回動作開始位置を固定し、T字路検知後の走行も工夫した

走行制御の状態遷移

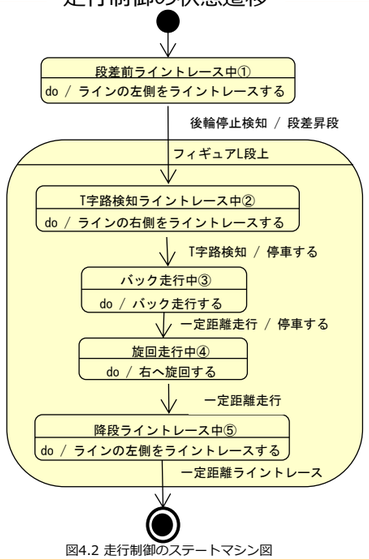


図4.2 走行制御のステートマシン図

T字路検知後の走行

状態	遷移条件	前輪角度	右後輪出力	左後輪出力
③バック走行	15mm走行	0deg	-30	-30
④旋回走行	50mm走行	55deg	0	80

※前輪角度：右方向＝プラス、左方向＝マイナス　後輪出力：前進＝プラス、バック＝マイナス

バック走行距離	30回テスト中の成功回数
0mm	24回
15mm	30回
30mm	24回

T字路検知後15mmバック走行すると、旋回時に前輪が板から落ちなくなった。

また、上表に示した旋回パラメータの時、バック走行距離が0～30mmの範囲外だと、板から落ちてしまった。

そこで、バック走行距離0～30mmの間で、試行を重ねたところ、特に走行距離が15mmだと他の走行距離に比べ成功率が高くなった。
よって、15mmを採用した。
これらの戦略を図にまとめると下図の動作を行う。

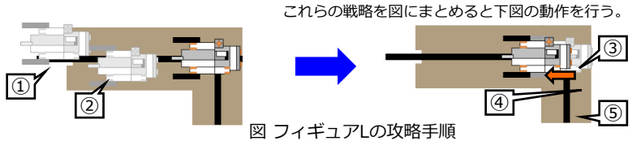


図 フィギュアLの攻略手順

①:ラインの左側をライントレースしながらフュギュアLに進入する。
②:T字路を検知するため、ラインの右側を昇段し、ライントレースする。
③:T字路検知後、前輪の脱輪を防ぐために、15mmバック走行する。
④:ラインの左側に光センサが来るように旋回する。
⑤:ラインの左側をライントレースしながら、段上から降りる。

新幹線

使用要素技術: Advanced PID、自己位置推定、後輪停止検知、ライン復帰

※新幹線検知後1秒停止し、接触のリスクを抑えた

走行制御の状態遷移

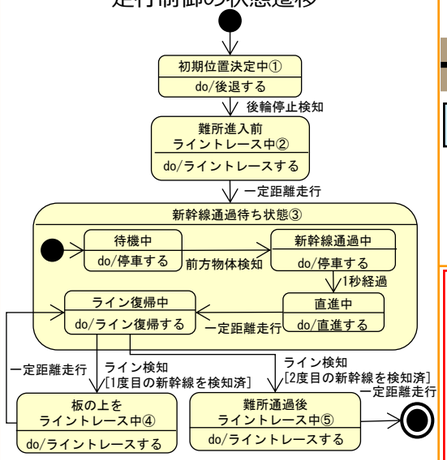
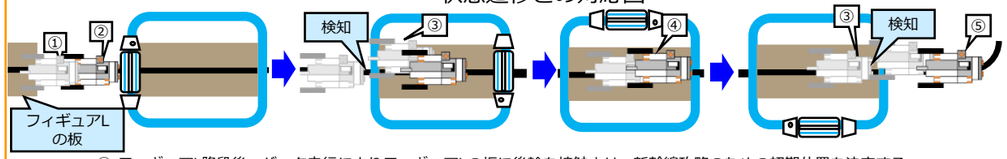


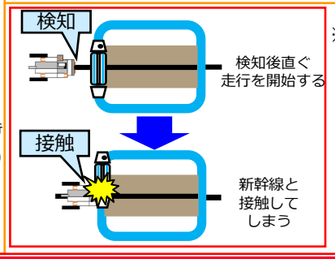
図4.3 走行制御のステートマシン図

状態遷移との対応図



①:フィギュアL降段後、バック走行によりフィギュアLの板に後輪を接触させ、新幹線攻略のための初期位置を決定する。
②:初期位置から新幹線が検知する位置までライントレース走行を行い、停車する。
③:新幹線検知後、1秒間停車し、新幹線の通過を待つ。そして直進走行し環状線を通過後、ライン復帰によってラインに復帰する。
④:ラインに復帰した後、再び新幹線を検知する位置までライントレース走行を行い停車する。
⑤:③と同じ処理を行った後、ライントレース走行を一定距離行い、次の区間に遷移する。

新幹線通過待ち秒数の根拠



※1秒と設定した根拠に関して
・新幹線の全長は約40cm
・走行体が新幹線を検知するのは、新幹線全長のおよそ半分(20cm)の地点
・また、 新幹線の速度は電池残量によって
→最高速度：約40[cm/s](新品の電池使用時)
最低速度：約25[cm/s](電池消耗時)
∴1秒間停車すると新幹線は、60cm～37.5cm進むため、1秒停車すれば、新幹線に当たる可能性が低い為、1秒に決定した。
上記の条件により、30/30回成功した。

本モデルで使用するUMLはすべてUML2.0に準拠している

5. 要素技術　追跡線隊レッド

バーコード読み取り

サンプルの中央値を使った白黒判定によって、
正確にバーコードを読み取る

【概要】
10個ある一定幅(30mm)のビットのうち、読み取り開始位置となるビット(スタートビット)と最後の1ビット(ストップビット)は読み飛ばし、間の8個の白(0)と黒(1)を読み取り、障害物エリアの走行ルートを決定する。

【実現方式】
後輪がバーコードに昇段した後、スタートビットまで約5cmある。そのスペースを低速ライントレースすることで、走行体の向きを整える。その後、30mmずつ直進しながら、光センサの値を取得することでバーコードを読み取る。

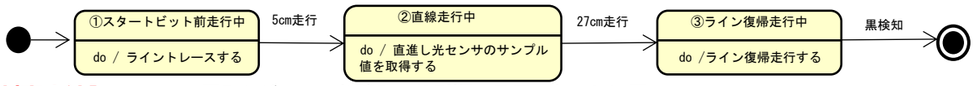
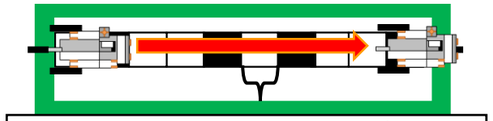
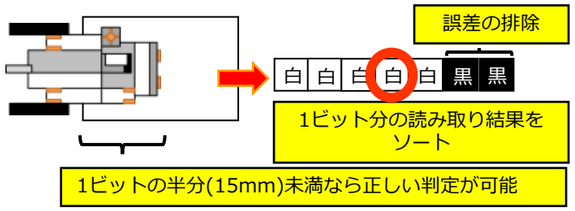


図5.1 バーコード読み取り手順のステートマシン図

【0と1の判定方法】
ビットごとに取得した光センサ値 43個の サンプルから中央値を求め、白黒判定を行う。判定位置のズレが1ビットの半分(15mm)未満なら正しい判定ができる。


1[bit] = 30 [mm] ≒ 42.9[encoder] (ギア比1:1)
約43サンプル(1ビット分)の光センサ値を取得可能

【読み取り精度向上策】
低速走行し後輪エンコーダ値が1変化するたびに光センサの値を取得する。できる限り多くの光センサのサンプル数を確保することで精度の高い白黒判定を行うことができる。



自己位置推定

現在までの走行距離を後輪エンコーダ値から求め、走行位置を把握して走行区間の切り替えに用いる。

[車輪走行距離]
= [後輪直径] × π × [後輪エンコーダ値] / 360

[車体走行距離]
= ([左後輪走行距離] + [右後輪走行距離]) / 2

左右の後輪の走行距離を平均して
車体中央部分の走行距離を算出する

難所検知のタイミングで補正を行い、自己位置推定ズレの蓄積を防ぐ。

ライン復帰

難所からコースに復帰するため、ライン探索を行う。

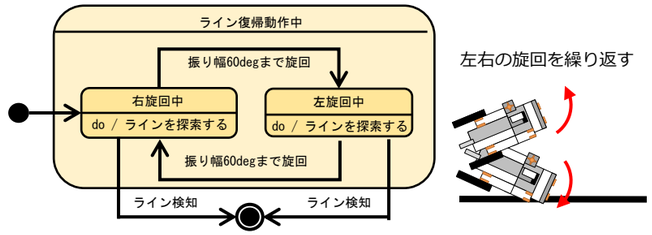


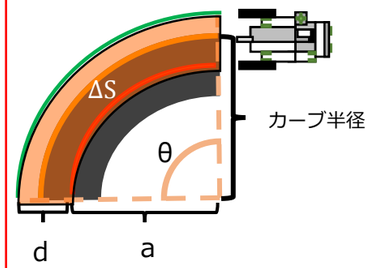
図5.2 ライン復帰動作のステートマシン図

左右に旋回を繰り返してラインを探索する

振り幅60degで、1往復以内にラインを30/30回発見できた。

定常円旋回

カーブ形状に沿った最適な後輪の出力を求め、モータを制御する。



- 外輪走行距離：$\Delta l'_L$
- 内輪走行距離：$\Delta l'_R$
- カーブ外周
- 後輪走行軌跡を弧に持つ扇形： ΔS
- カーブ角度：θ
- 車体の左右輪間の距離：d
- カーブ中心から内輪までの距離：a

$$\Delta S = \frac{1}{2}(a+d)\Delta l'_L - \frac{1}{2}a\Delta l'_R \quad \cdots ①$$

$$\Delta S = \frac{\theta}{2\pi}\{(a+d)^2\pi - a^2\pi\} \quad \cdots ②$$

式①、②より

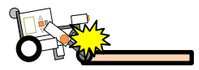
$$\Delta l'_R = \frac{a+d}{a}\Delta l'_L - \left(\frac{d^2}{a} + 2d\right)\theta$$

カーブ角度θとカーブ半径$(a+\frac{d}{2})$がわかれば、外輪出力に対しての最適内輪出力を算出できる。

後輪停止検知

後輪の停止を検知することで
段差などの障害物への接触を検知する。

障害物に引っ掛かった際の走行体停止を後輪エンコーダ値の変化量から観測する。

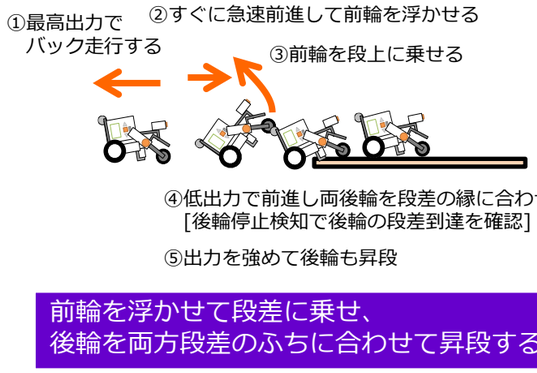


一定時間後輪エンコーダ値の
変化がない場合、
障害物接触と判定する

ジャイロセンサによる段差検知と比較して、走行振動による誤検知や検知漏れが減少する。

段差昇段

前輪持ち上げによる段差昇段で安定して難所に進入する。

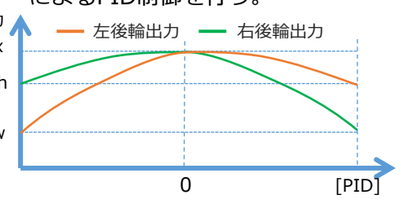


前輪を浮かせて段差に乗せ、
後輪を両方段差のふちに合わせて昇段する

Advanced PID

ライントレースの安定性を高めるため収束が早く、安定性の高い、二次関数によるPID制御を行う。

PIDゲイン値の
二次関数で
後輪出力を決定する

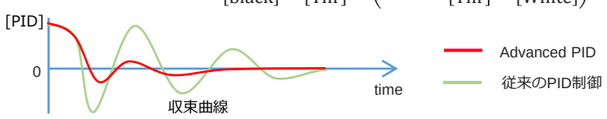


【モータ出力の公式】
(右後輪について掲載,
左後輪は対称形)

[Max] ：最高出力　[PID] : PID制御値 $(k_pP + K_iI + K_dD)$
[High] − [Low] : 左右後輪の最大出力差
[Black] ：黒基準値　[White] ：白基準値　[Thr] ：閾値

$$\text{if } [\text{PID}] > 0; \text{then: } v_{\text{Right}} = -\frac{[\text{Max}]-[\text{High}]}{[\text{black}]-[\text{Thr}]} \times [\text{PID}]^2 + [\text{Max}]$$

$$\text{if } [\text{PID}] < 0; \text{then: } v_{\text{Right}} = \frac{[\text{Max}]-[\text{Low}]}{[\text{black}]-[\text{Thr}]} \times \left([\text{PID}] \times \frac{[\text{Black}]-[\text{Thr}]}{[\text{Thr}]-[\text{White}]}\right)^2 + [\text{Max}]$$



従来のPID制御よりも素早く目標値に収束する

光センサ値の補正

環境光の変化の影響を抑え、ライントレースの安定性を向上させる。

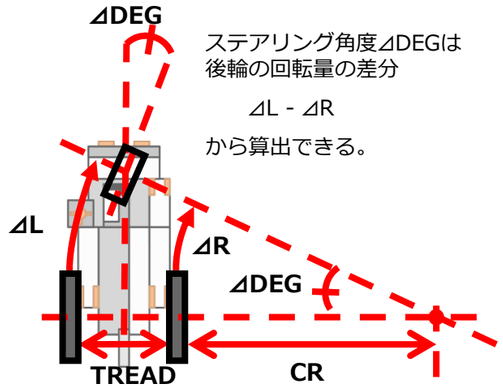
【問題点】環境により、黒と白の光センサ基準値の幅が異なるため、PID制御の制御量が変化することから走行の安定性に影響が出る。

【解決策】上記の影響を抑えるため、光センサ値を正規化して制御量の計算に使用する。

[正規化後のセンサ値]
= ([センサ値] − [黒基準値]) / ([白基準値] − [黒基準値])

ステアリング角度最適化

左右後輪の回転量差から最適な前輪ステアリング角度を算出し、安定した旋回を実現する。



ステアリング角度⊿DEGは
後輪の回転量の差分
⊿L - ⊿R
から算出できる。

【問題点】前輪と後輪の制御を個別に行うと前輪のステアリング角度と左右後輪の回転量が噛み合わず、走行が不安定になる

【解決策】後輪の回転量差からステアリング角度を算出する

⊿L、⊿R	：左右後輪の回転量
TREAD	：走行体の左右輪間の距離
CR	：回転半径
⊿DEG	：前輪のステアリング角度

⊿L = 2π(CR+TREAD) * (⊿DEG / 360)
⊿R = 2π * CR * (⊿DEG / 360)
⊿L - ⊿R = TREAD * ⊿DEG * (π / 180)
⊿DEG = (⊿L - ⊿R) * (π / 180) / THREAD

本モデルで使用するUMLはすべてUML2.0に準拠している