ETロボコン2018

デベロッパー部門 アドバンストクラス

地区：東海　　地域：愛知県刈谷市

チームNo. 113　　チーム名：ダントツ戦隊デンソルジャー　　所属：株式会社デンソー

チーム紹介、目標、意気込み

前回は惜しくも地区予選で敗退してしまったが、正義はくじけない！
昨年メンバー５人に加え、新たに４人のソルジャーが仲間になり、パワーアップして帰ってきた！！
ＣＳ大会優勝を目指し、モデリングの力で組込みソフト開発の闇を暴く！
そう我らこそ、**ダントツ戦隊デンソルジャー！！**

モデルの概要

“確実性”を重視した戦略

ポイント１．信頼性の向上

- 通信途絶等を考慮し、ブロックの初期情報をカメラシステムだけでなく、手入力、走行体からも取得する機構を作成し冗長的な設計とした。
- 走行体を安定して移動させるため、ブロック並べでは、「フリー走行」を用いる場合を最小限に抑え、極力「ライントレース走行」を選択するようにした。AIアンサーでは、「フリー走行」のバラツキが許容範囲か判断する仕組みを導入した。

ポイント２．確実性を支えるソフトウェア構造

- 制御の妥当性を検証するため、機能毎に検証をしやすいソフトウェア構造にした。
- 当日の環境に適応しやすいように、パラメータ調整が容易なソフトウェア構造にした。

モデルの構成

1. 要求モデル

「確実な範囲で最小のリザルトタイムを獲得する」という目標のもと、D-caseを用いて方針を決め、主に以下の要件を抽出した。

①. 確実に目標タイムを獲得するために、センサやアクチュエータのバラツキ、環境など外的要因への対策を立てる。

②. 確実な範囲で、目標タイムを最小化する戦略を立てる。

③. 上記を満たすために開発しやすい仕組みを導入する。

2. 分析モデル

①を満たすため、カメラシステムとの通信が途絶しても、ブロックの位置と色を取得できるロジックを用いた。

①・②を満たすため、ライントレース優先・残り時間を考慮した経路を決めるアルゴリズムを用いた。

3. 設計モデル

③を満たすため、攻略順序・外部入力・動作指示・走行体制御・走行体情報に機能を分割し、実装・検証の分担がしやすい仕組みにした。

③を満たすため、センサ、アクチュエータとのやり取りを標準のフォーマットで指示できるようなソフトウェア構造にした。

4. 制御モデル

（AIアンサー） ①・③を満たすため、対角線を走行するという共通の動作で左右の数字を判定する。数字を判定できるか、実際に走行させたデータを用いて検証した。

（ブロック並べ） ①を満たすため、必ず「ライントレース走行」でブロック置き場にブロックを置く。

（ブロック並べ） ①・②を満たすため、以降の処理に影響を加味したタイムアウト時間で、通信途絶を判定する。

1.要求モデル

1-1.目標

確実な範囲で最小のリザルトタイムを獲得する

目標タイム

実走結果を踏まえて目標走行タイムと目標ボーナスタイムを定め、目標リザルトタイムを-3秒とした。確実性との両立が困難な場合は、**確実性を優先**させる方針とする。

- スピード競技　走行タイム　：25秒
- ブロック並べ　ボーナスタイム　：23秒
- 直角駐車　ボーナスタイム　：5秒

25秒 － (23秒 ＋ 5秒) ＝ **－3秒**
[走行タイム]　[ボーナスタイム]　[目標タイム]

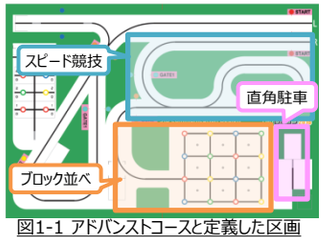


図1-1 アドバンストコースと定義した区画

目標タイムの根拠

【スピード競技】2つの中間ゲートを通るという制約と場所を見失うリスクを回避して、黒線をトレースする戦略とし、実走タイムと走行安定性を考慮して、目標走行タイムを25秒とする。

【ブロック並べ】
パワーブロックの位置ごとにボーナスポイントを比較し、制限時間の制約とパワーブロックの移動に失敗するリスクを考慮してブロック運搬に注力するため、表1の方針①を選択した。

表1-1 パワーブロックの移動有無による比較

番号	パワーブロック	パワーポット重複	ボーナスタイム
方針①	初期位置のまま	1箇所	23秒
方針②	移動させる	2箇所	25秒

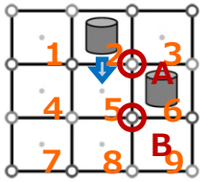


図1-2 パワーブロックの初期位置と移動例

確実性

開発目標として、本プロジェクトでの「確実性」を次のように定めた。

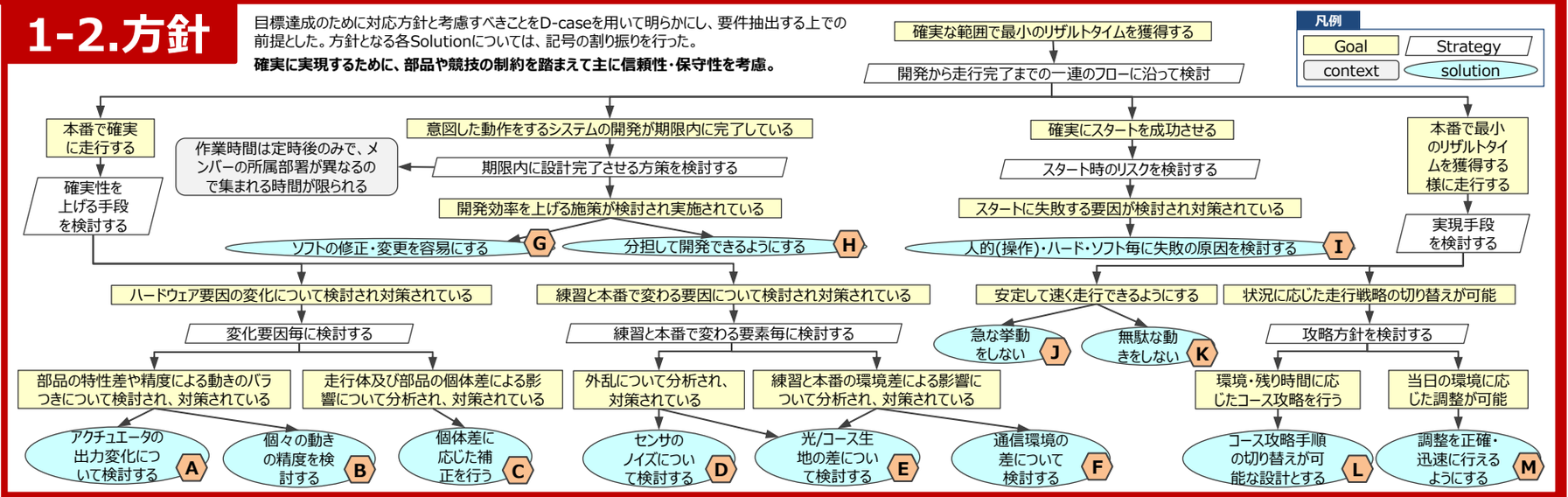
本番当日のフロー（準備～Rコース競技終了）において、完走し意図通りの目標タイムを獲得できる確率　：**95%**

・練習環境と異なる環境で走行
・再走を考慮する
・2回走行して成功率9割以上が、チームメンバーの納得する基準
⇒以上より目標値を決定。

1-2.方針

目標達成のために対応方針と考慮すべきことをD-caseを用いて明らかにし、要件抽出する上での前提とした。方針となる各Solutionについては、記号の割り振りを行った。

確実に実現するために、部品や競技の制約を踏まえて主に信頼性・保守性を考慮。



1-3.要件

システムに必要な機能を、ユースケース図（及びユースケース記述、本モデルでは省略）を用いて導出した。また、導出された各機能毎に、1-2.方針で導出した方針を満たすように、詳細な要件を抽出した。

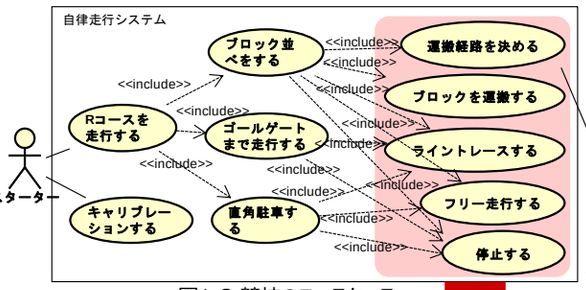
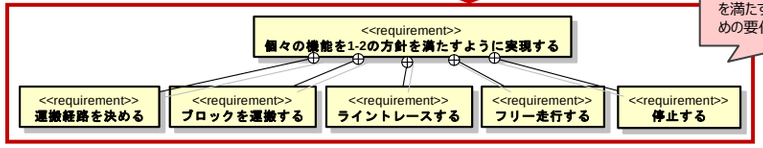
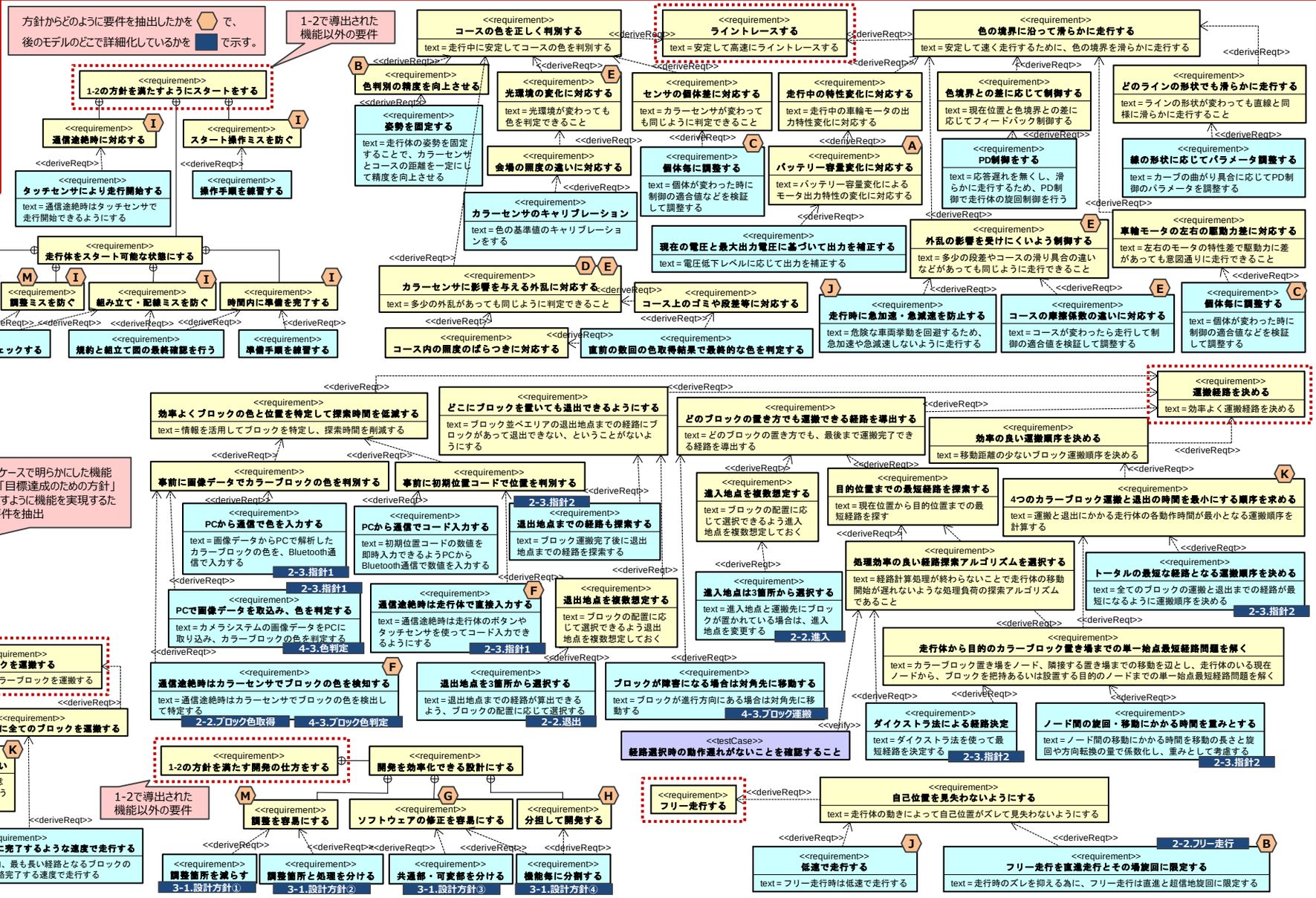


図1-3 競技のユースケース

ユースケースで明らかにした機能毎に、「目標達成のための方針」を満たすように機能を実現するための要件を抽出





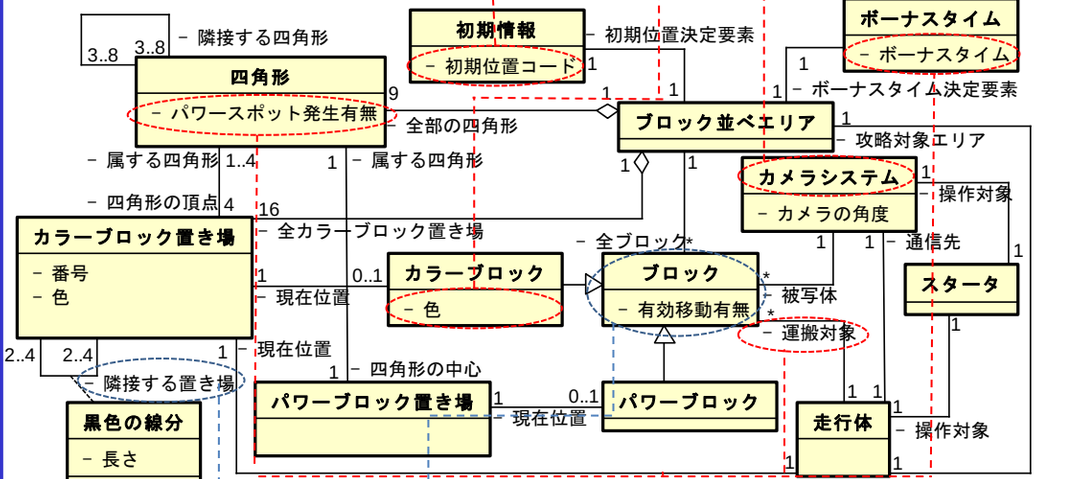
2.分析モデル

2-1.ゲームの構成要素

ブロック並べゲームを構成する要素を図2-１のクラス図に示す。
右記＜凡例＞のように、「走行体の動作定義」につながる箇所や、要求モデルの要件と関わる箇所を抽出し、2-2.「走行体の動作定義」と2-3.「指針」で分析した。

＜凡例＞
- 2-2.「走行体の動作定義」と関係する部分
- 2-3.「指針」と関係する部分

要件①　初期位置コードからブロック初期位置、画像データから色情報を事前に特定して探索時間を低減する。



2-2.走行体の動作定義
＜ライントレース＞
＜カラーブロック置き場の通過＞
＜ブロック把持＞を参照

2-2.走行体の動作定義
＜ブロック運搬中の旋回＞
＜ブロックのリリース＞を参照

要件②　運搬前に以下を満たす運搬手順を決定する。
・ボーナスタイム23秒（「1-1.目標」参照）
・４つのカラーブロック運搬と退出の時間が最小な経路。

図2-1 ブロック並べを構成する要素を示したクラス図

2-2.走行体の動作定義

走行体がカラーブロックを運搬する一例を図2-2に示す。
一連の流れの中で、「安定」に関する要求に基づく各動作を定義した。

＜ライントレース＞
カラーブロック置き場間を確実に移動するため、カラーブロック置き場間の接続を利用。

＜カラーブロック置き場の通過＞
要求「自己位置を見失わないようにする」を満たすため、一定量の旋回または直進。

＜ブロック把持＞
要求「安定してブロックを把持する」を満たすため、ライントレースでカラーブロックへ確実に向かう。

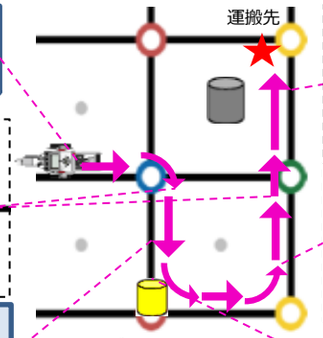
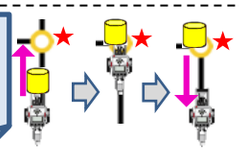
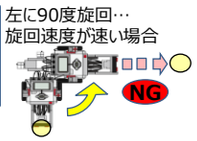


図2-2 ブロック運搬の一例

＜ブロックのリリース＞
要求「輪の中に安定してブロックを設置する」を満たすため、ラインに沿って後退しリリース。

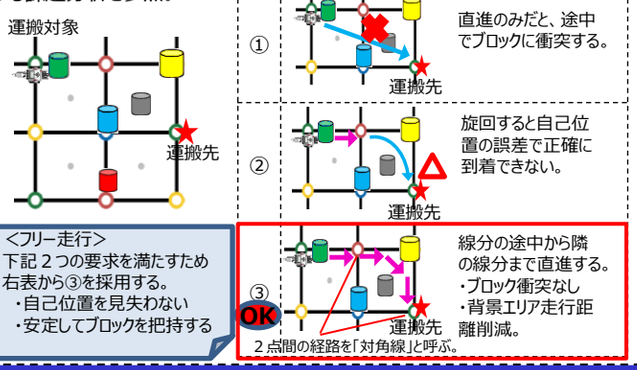


＜ブロック運搬中の旋回＞
要求「意図せずブロックを離さない」を満たすため、一定の角速度以下で旋回。



＜カラーセンサーによるブロック色取得＞
要求「通信途絶に対応する」を満たすため、アームを持ち上げてカラーセンサーをブロックに向ける。

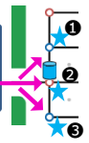
下図に示すように、ライントレースのみで運搬先に到達不可能な場合などは、背景エリア上をフリー走行する。フリー走行する理由は2-3「指針」の要件②に対する課題分析を参照。



＜フリー走行＞
下記２つの要求を満たすため右表から③を採用する。
・自己位置を見失わない
・安定してブロックを把持する

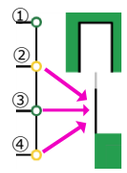
ブロック並べエリアへの進入と退出の動作を定義。
進入地点と退出地点が複数ある理由は、2-3「指針」の要件②に対する課題分析を参照。

＜進入＞
・右図の❷にはライントレースで進入。
・❶、❸にはフリー走行で進入。



＜退出＞
要求「どこにブロックを置いても退出できるようにする」を満たすため、右図②～④※からフリー走行でブロック並べエリアを退出する。

※①は直角駐車場の柵に走行体が接触する可能性がある。



2-3.指針

2-1.「ゲームの構成要素」クラス図中の２つの要件を満たすための各手順の特長、想定される課題とその対策を分析し、指針を導出した。

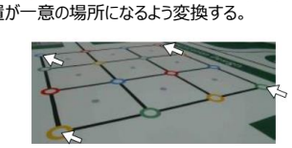
課題分析１【要件①】
初期位置コードからブロック初期位置、画像データから色情報を事前に特定して探索時間を低減する。

＜アプローチ＞カメラシステムの活用
――特長――
・ブロック並べスタート前に全カラーブロックの色を取得することで事前に運搬経路を生成可能
・走行体によるカラーブロックの色識別動作が不要

初期位置コードで色検出範囲を限定

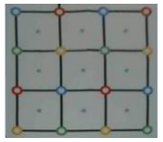
カメラシステムでカラーブロックの色を取得する様子

＜課題①＞カラーブロック置き場のピクセル位置が一意になるようカメラ角度等を設置することは難しい
＜対策①＞カメラの位置・角度が異なっても同じカメラ座標系へ変換することでカラーブロックのピクセル位置が一意の場所になるよう変換する。



変換前の元画像で四隅を指定

変換



毎回同一のピクセル位置に各ブロック置き場が位置するように変換

＜課題②＞カメラシステムと走行体の通信途絶時、走行体はブロック初期位置と色情報を取得不能。
＜対策②-1＞スタータが初期位置コードを走行体に直接入力。
＜対策②-2＞走行体が初期位置まで行き、その都度カラーセンサでブロックの色を特定。

【指針１　-ブロック初期位置と色特定-】
カメラシステムを活用した方法をメインとし、Bluetooth通信途絶時には初期位置コードを走行体に直接入力、色特定をカラーセンサで行う。

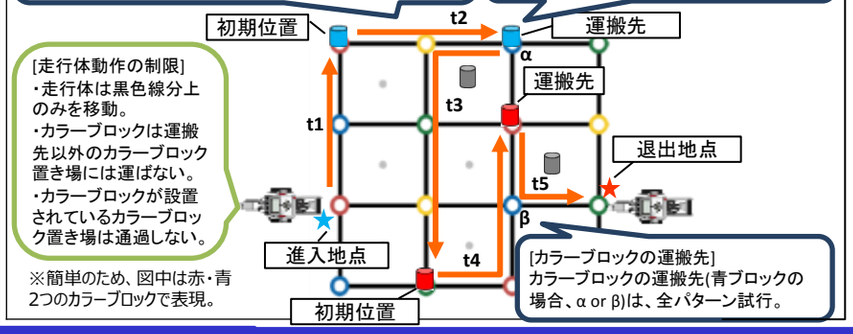
課題分析2【要件②】運搬前に以下を満たす運搬手順を決定する。
・ボーナスタイム23秒（「1-1.目標」参照）
・４つのカラーブロック運搬と退出の時間が最小な経路。

＜アプローチ＞ブロック並べの制限時間を80秒と定義し、80秒以内に運搬可能な運搬手順を探索する。
⇒[120秒(競技時間)−25秒(スピード競技)−5 秒(スピード競技-ブロック並べ間移動)−10秒(直角駐車)]

[最短運搬手順の探索方法]
運搬手順は下図"進入地点"から"退出地点"までの移動順序を指す。運搬手順の時間"$\sum_{i=1}^{5} t_i$"が最小となる、運搬手順を探索する。
運搬時間に影響を与える、"経路"(オレンジ線)、"カラーブロックの運搬順序"、"カラーブロックの運搬先"の求め方および"走行体動作の制限"は下図の通り。

[経路]
各カラーブロック置き場間の経路(各オレンジ線)はダイクストラ法で最短経路を求める。走行体の各動作に要する時間(測定値)を重みとして、経路の時間を算出する。

[カラーブロックの運搬順序]
カラーブロックの運搬順序(青→赤or赤→青)は全パターン試行。



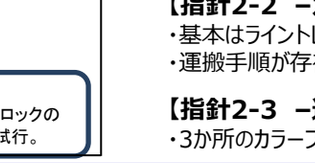
[走行体動作の制限]
・走行体は黒色線分上のみを移動。
・カラーブロックは運搬先以外のカラーブロック置き場には運ばない。
・カラーブロックが設置されているカラーブロック置き場は通過しない。

※簡単のため、図中は赤・青2つのカラーブロックで表現。

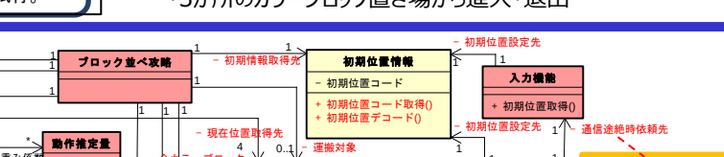
[カラーブロックの運搬先]
カラーブロックの運搬先(青ブロックの場合、α or β)は、全パターン試行。

本アプローチで要件を満たせているか分析した。
[ボーナスタイム23秒を獲得できるか分析]
＜課題＞
一部のブロック初期位置パターンに対して、全カラーブロックでパワースポット設置を成立させる運搬手順が存在しない場合がある。(全初期位置パターン115200中、4354パターン)

＜要因①＞進入地点ブロックを運搬できず　＜対策①＞進入地点を3か所に（2-2.「進入」を参照）



＜要因②＞退出地点までの経路無し　＜対策②＞フリー走行を導入（2-2.「フリー走行」を参照）

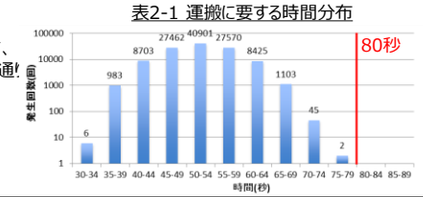


＜要因③＞退出地点にカラーブロックが存在　＜対策③＞退出地点を3か所に（2-2.「退出」を参照）

対策①～③により、全初期位置パターン115200全てに対して、運搬手順を探索可能となる。

[運搬時間の分析]
全ブロック初期位置パターンに対して、運搬に要する時間の分布は右表の通り。
"80秒未満"を全パターンに於いて満たすことができている。

表2-1 運搬に要する時間分布



【指針2-1　–運搬手順の探索方法-】
・各カラーブロック置き場間の経路はダイクストラ法で算出
・カラーブロックの運搬先は全パターン試行
・カラーブロックの運搬順序は全パターン試行

【指針2-2　–カラーブロック置き場間の移動-】
・基本はライントレースで移動
・運搬手順が存在しない場合はフリー走行で移動

【指針2-3　–進入/退出地点までの経路探索-】
・3か所のカラーブロック置き場から進入・退出

2-4.解法

2-2.「指針」を反映した解法を示すため、2-1.「課題」のクラス図を更新した。なお、解法に直接関係しない要素や、単純なset/getの操作は紙面の都合上省略した。

＜凡例＞
- 解法での新規追加クラス
- 動作実行クラス
- 赤字　新規追加の関連・属性・操作(新規クラス同士含まず)

補足：
目標位置は下記４種類ある。
運搬経路が右図の経路Ⅰと経路Ⅱがそれぞれ最短となるように決定される。

①運搬前のブロックが置いてある場所
②運搬先
③進入地点（2-2.＜進入＞参照）
④退出地点（2-2.＜退出＞参照）

補足：
通信途絶時、入力機能が初期位置・パワースポット発生有無をカラーブロック置き場に設定する。

補足：
通信途絶時、カラーブロックの色は「未定」となり、それを受けて走行体動作管理が色取得する動作を設定する。

2.分析モデル

2-4.解法（続き）

前ページに記載したクラス図中の関連を検証するため、一連の流れをシーケンス図で記述した。

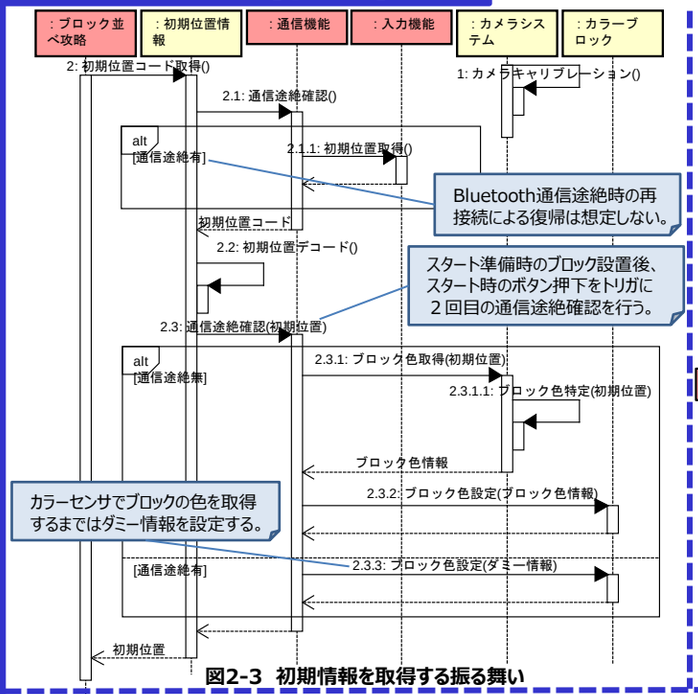


図2-3 初期情報を取得する振る舞い

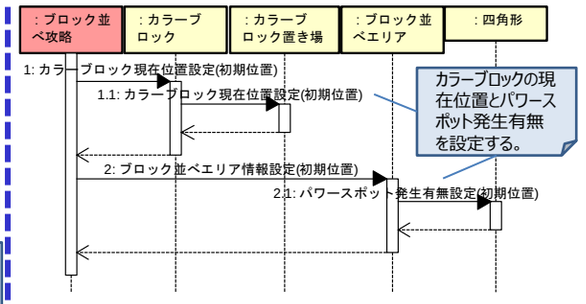


図2-4 初期情報を設定する振る舞い

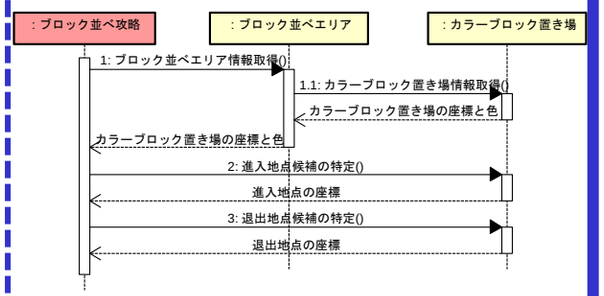


図2-5 経路算出用の現在位置/目標位置の取得

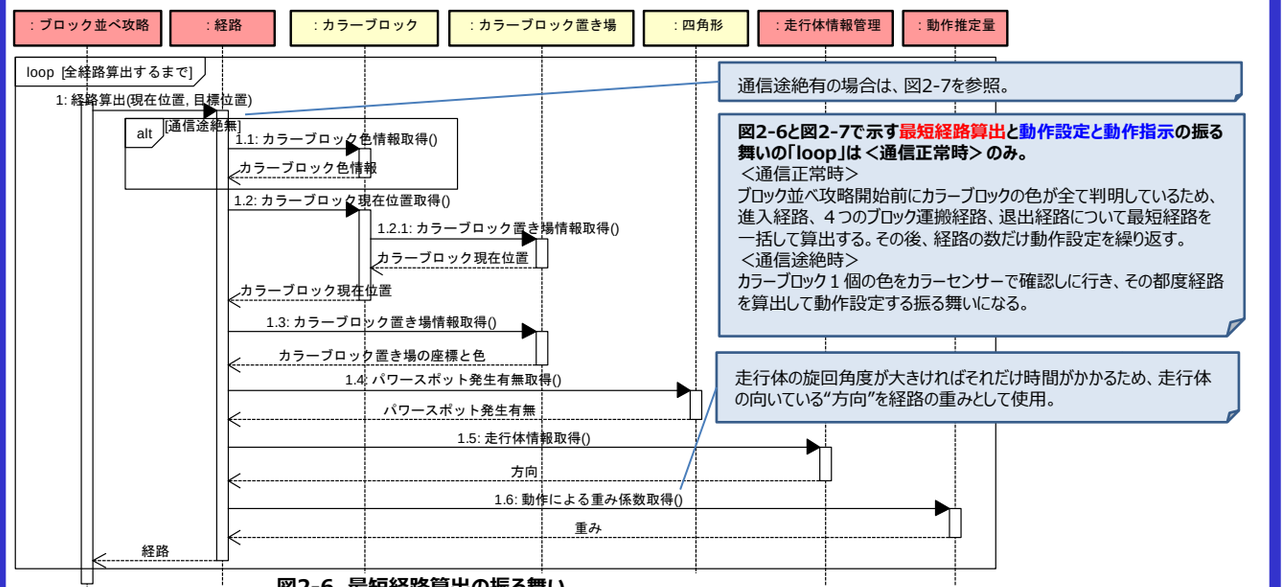


図2-6 最短経路算出の振る舞い

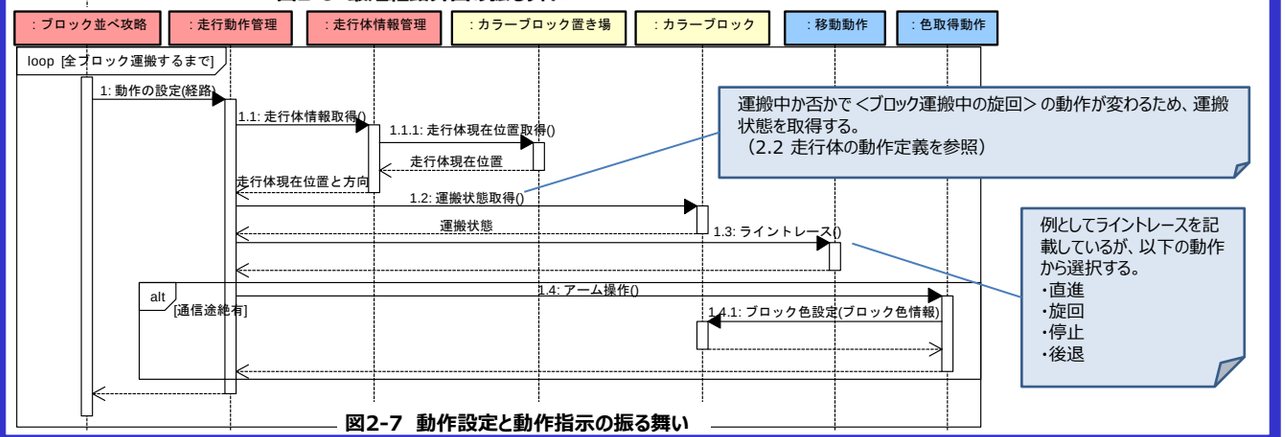


図2-7 動作設定と動作指示の振る舞い

3.設計モデル

3-1.設計方針・設計意図

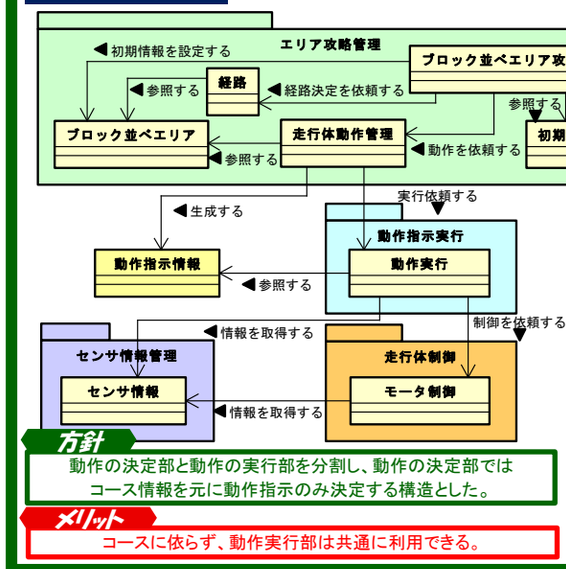
ソフトウェア設計方針の検討 要求分析にて抽出したソフトウェア設計に関する要件から設計の方針を決定した。

要件	意図	設計方針	
調整が容易にできる	・パラメータの調整にかかる時間を減らしたい ・実装や制御に関する知識がないメンバーもパラメータの調整ができるようしたい ・パラメータ調整時に発生するリスク(人為的な変更ミス)を減らしたい	①	・調整パラメータを抽象度の高いものにする(人間も調整しやすいパラメータにする) ・具体的な値(PID値やPWM値)は調整パラメータから自動算出するようにする
		②	・調整箇所(走行体動作の定義部)と実行処理(走行体動作の実行部)を分ける
機能の追加・変更に対応できる	・後から新しい戦略や要素技術を追加したくなった場合でもソフトウェアの変更箇所を少なくしたい	③	・共通部と可変部を分けた構造とし、可変部へのインターフェースを共通化する
機能毎に実装・テストができる	・複数人で分担して実装・テストできるようにしたい ・不具合発生時に原因箇所の特定を容易にしたい	④	・機能ごとに独立して動作する構造にする

3-2.構造設計(全体)

パッケージ設計 ブロック並べゲームの分析結果および、ソフトウェアの設計方針を元にソフトウェア全体の構造設計を行った。

ポイント
カメラシステムとの通信部やユーザからの入力受付部を独立させ、分担して実装・テストできるようにした。



方針
コース攻略に必要な動作を全て「動作指示」として定義し、「動作指示」を元に走行体の制御を行う構造とした。

データ名	役割
動作指示情報	１つの動作を実行するのに必要な情報を保持する。

メリット
走行体の動作に必要な情報を「動作指示」に集約することで、パラメータの調整や修正が容易かつ安全になる。

パッケージ名	役割
エリア攻略管理	コース情報(コース形状やブロック等コース上のオブジェクト位置など)を参照して、攻略に必要な動作を決定する。→ 分析モデルの「解法」部分に相当する
動作指示実行	攻略順序管理パッケージより動作指示を受け取り、内容を解釈してセンサやアクチュエータとやり取りする
センサ情報管理	走行に必要な情報をセンサから取得し、保持する。 また、センサの校正やデータの加工を行う。
走行体制御	動作指示管理パッケージの依頼に基づいて、モータの制御量を決定する。 また、走行体情報を参照して、最終的な制御量に補正を加える。

方針
動作の決定部と動作の実行部を分割し、動作の決定部ではコース情報を元に動作指示のみ決定する構造とした。

メリット
コースに依らず、動作実行部は共通に利用できる。

3-3.振舞設計(全体)

タスク設計 ブロック並べゲームの分析結果および、ソフトウェアの設計方針を元にソフトウェア全体の構造設計を行った。

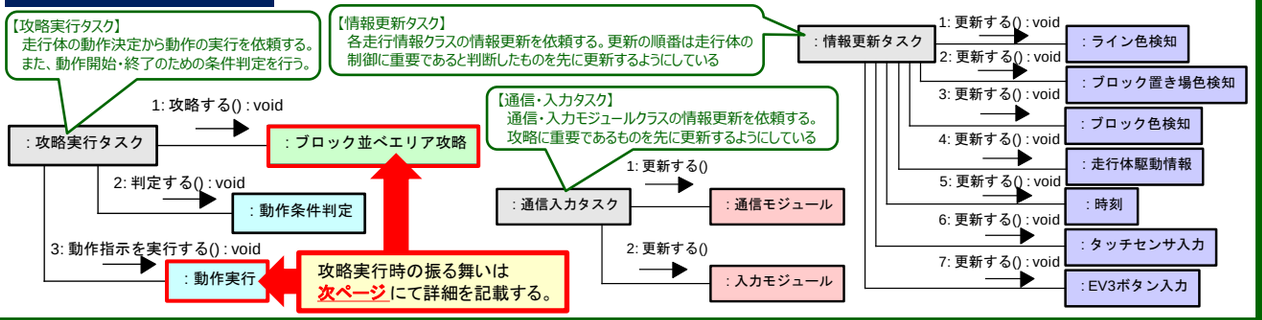
タスク	周期	優先度	役割と設計意図	実行パッケージ
初期化タスク	—	—	プログラム起動時に1度だけ実行されるタスク。他の周期タスク起動前に実施しておきたい処理(変数や各クラスの初期化)を実施する。 各種初期化処理を実行後、下記周期タスクを起動し、終了する。	・なし
攻略実行タスク	4ms	高	動作の決定から走行体の制御までを実行するタスク。 走行体が停止するとコースの攻略が不可能になるため。最も優先度を高く設定した。	・エリア攻略管理 ・動作指示実行 ・走行体制御
情報更新タスク	4ms	中	各センサ情報の更新を実行するタスク。 攻略実行の方が優先されるため、攻略実行タスクに対して優先度を下げた。	・センサ情報管理
通信・入力タスク	20ms	低	カメラシステムとの通信やユーザからの入力を受け付ける処理を実行するタスク。 走行体の制御やセンサ情報の更新に比べ、優先度は低く、更新頻度も多くないため、左記の設定とした。	・外部入力管理

ポイント
・各クラス(特にセンサやアクチュエータ)の初期化が完了してから、各処理の実行を開始することで安全に動作を開始できるようにした。
・走行体制御ほど実行頻度が要らない機能を適切な周期タスクに分けて実行することで、処理が干渉し、制御が妨げられるリスクを下げた。

3-5.振舞設計(詳細)-①

※紙面の都合上、「3-4.構造設計(詳細)」は次のページに記載

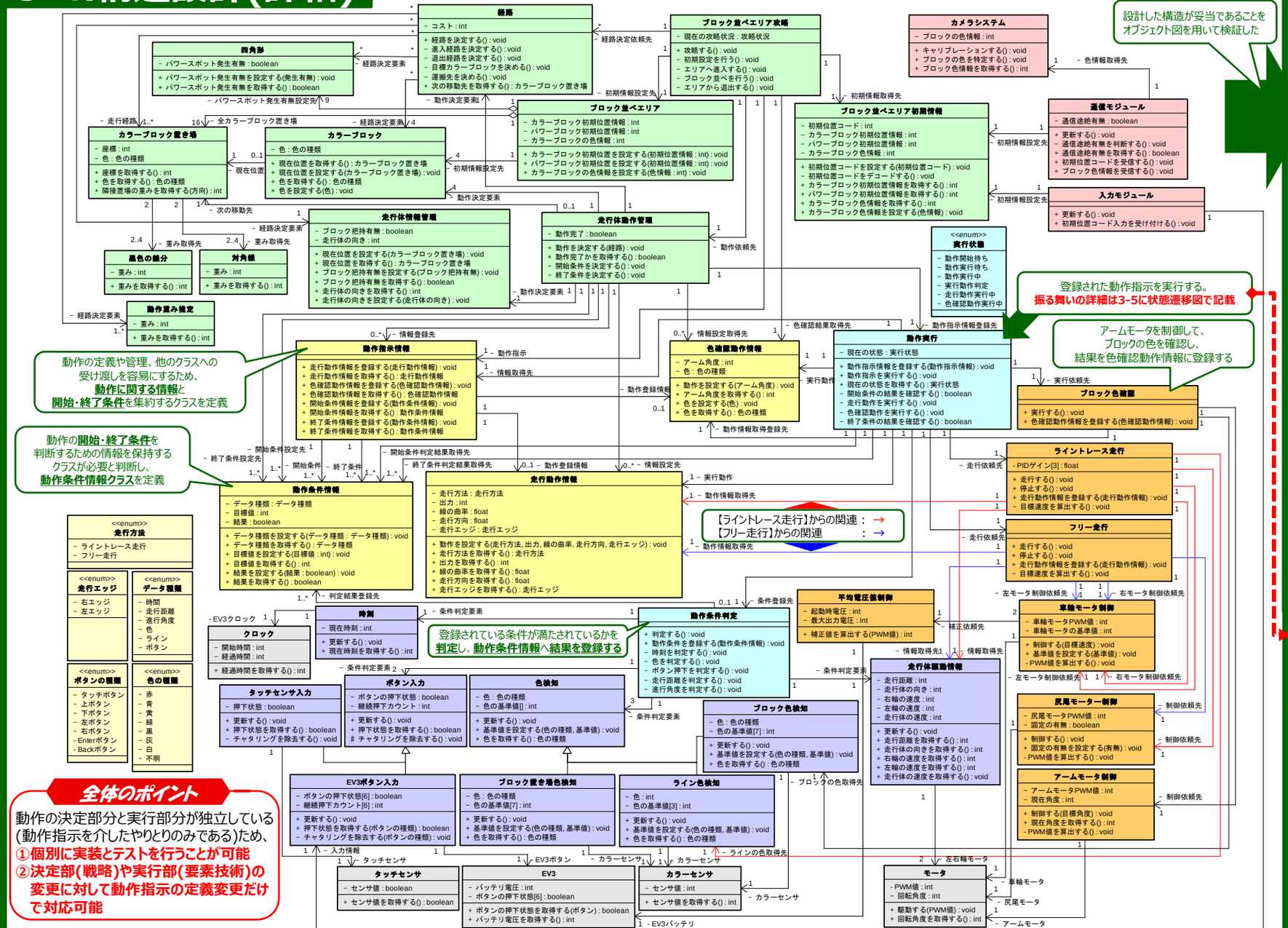
各タスクの振る舞い タスク設計(3-3.)及び、構造設計(3-1.と3-4.)の結果を元に、各タスクの振る舞いを設計した。



3 .設計モデル

3-4.構造設計(詳細)

構造の詳細 ゲームの分析結果および、【3-1】～【3-3】の結果より、構造の詳細を決定した。



設計した構造が妥当であることを
オブジェクト図を用いて検証した

登録された動作指示を実行する。
振る舞いの詳細は3-5に状態遷移図で記載

アームモータを制御して、
ブロックの色を確認し、
結果を色確認動作情報に登録する

動作の定義や管理、他のクラスへの
受け渡しを容易にするため、
動作に関する情報と
開始・終了条件を集約するクラスを定義

動作の**開始・終了条件**を
判断するための情報を保持する
クラスが必要と判断し、
動作条件情報クラスを定義

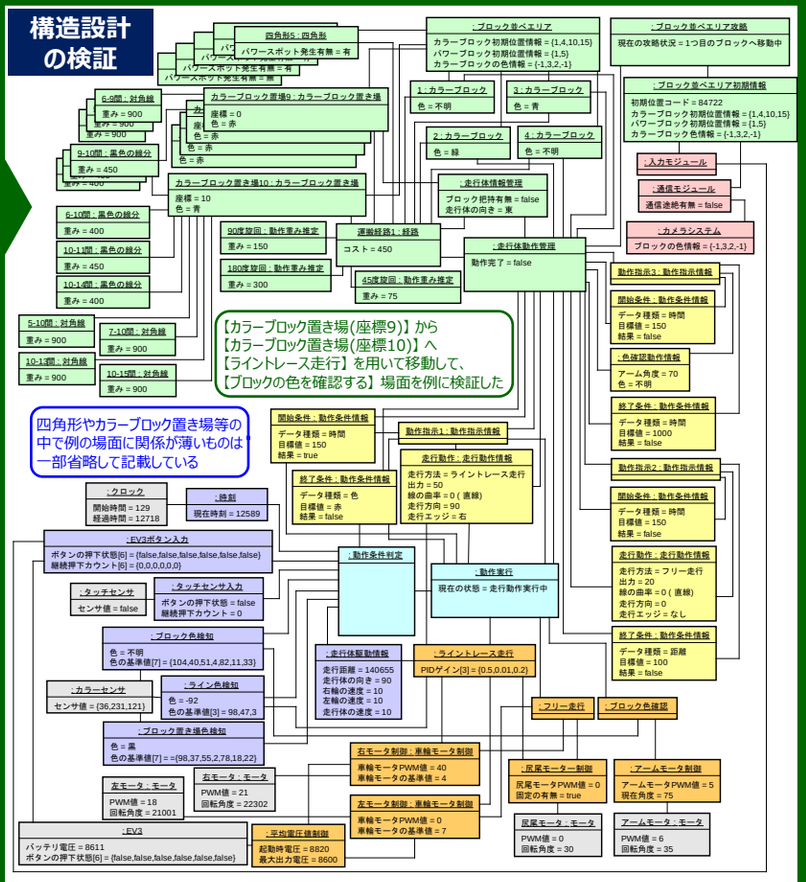
【ライントレース走行】からの関連 : →
【フリー走行】からの関連 : →

登録されている条件が満たされているかを
判定し、**動作条件情報へ結果を登録する**

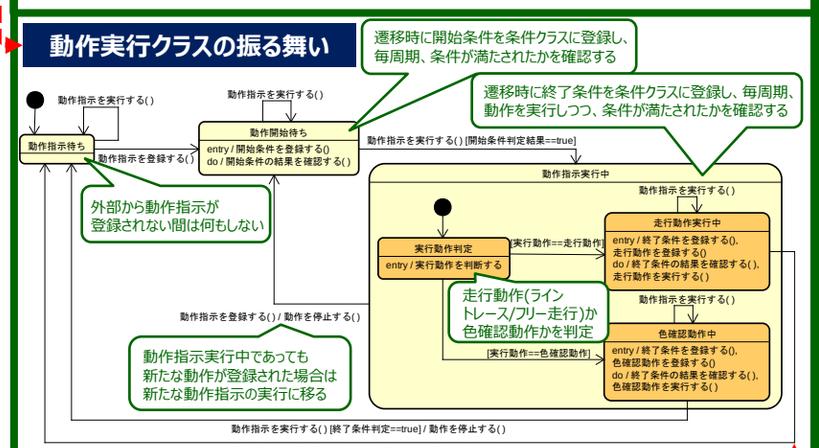
全体のポイント

動作の決定部分と実行部分が独立している
(動作指示を介したやりとりのみである)ため、
①個別に実装とテストを行うことが可能
②決定部(戦略)や実行部(要素技術)の
変更に対して動作指示の定義変更だけ
で対応可能

構造設計の検証



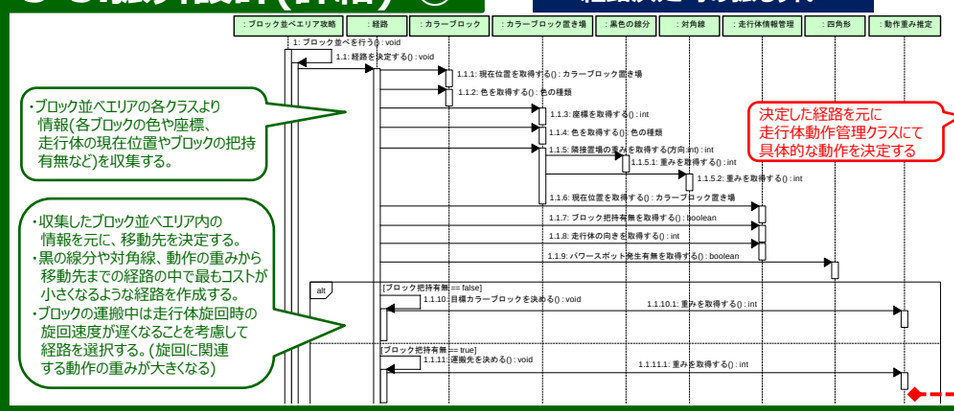
動作実行クラスの振る舞い



動作指示実行中であっても
新たな動作が登録された場合は
新たな動作指示の実行に移る

3-5.振舞設計(詳細)-②

経路決定時の振る舞い



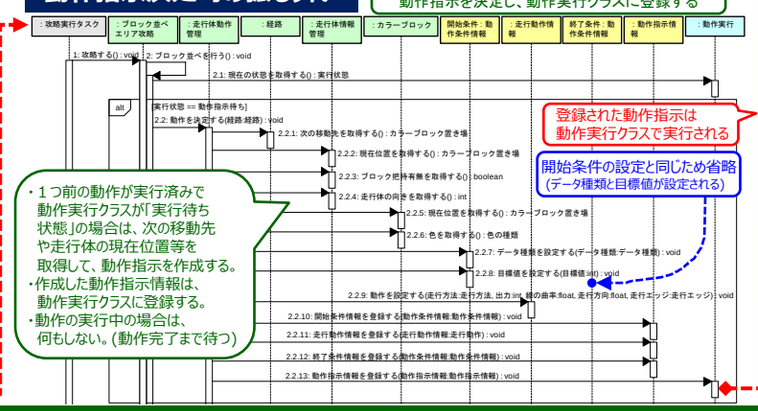
・ブロック並べエリアの各クラスより
情報(各ブロックの色や座標、
走行体の現在位置やブロックの把持
有無など)を収集する。

・収集したブロック並べエリア内の
情報を元に、移動先を決定する。
・黒の線分や対角線、動作の重みから
移動先までの経路の中で最もコストが
小さくなるような経路を作成する。
・ブロックの運搬中は走行体旋回時の
旋回速度が遅くなることを考慮して
経路を選択する。(旋回に関連
する動作の重みが大きくなる)

決定した経路を元に
走行体動作管理クラスにて
具体的な動作を決定する

動作指示決定時の振る舞い

【表現しているシーン】
動作指示を決定し、動作実行クラスに登録する



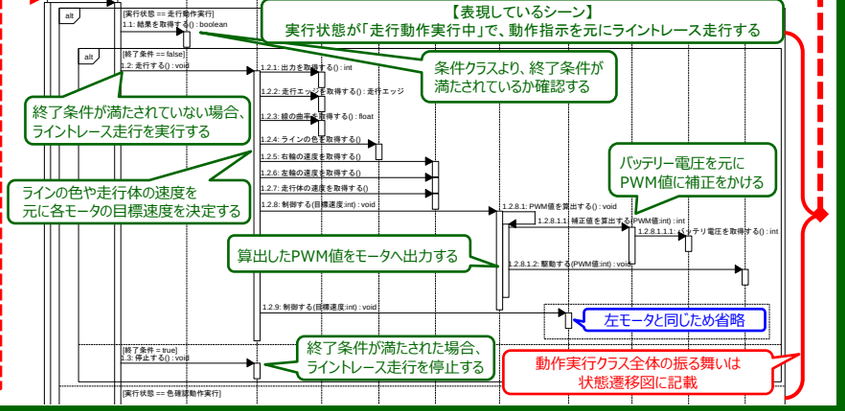
・１つ前の動作が実行済みで
動作実行クラスが「実行待ち
状態」の場合は、次の移動先
や走行体の現在位置等を
取得して、動作指示を作成する。
・作成した動作指示情報は、
動作実行クラスに登録する。
・動作の実行中の場合は、
何もしない。(動作完了まで待つ)

登録された動作指示は
動作実行クラスで実行される

開始条件の設定と同じため省略
(データ種類と目標値が設定される)

動作実行時の振る舞い

【表現しているシーン】
実行状態が「走行動作実行中」で、動作指示を元にライントレース走行する



条件クラスより、終了条件が
満たされているか確認する

終了条件が満たされていない場合、
ライントレース走行を実行する

ラインの色や走行体の速度を
元に各モータの目標速度を決定する

バッテリー電圧を元に
PWM値に補正をかける

算出したPWM値をモータへ出力する

左モータと同じため省略

終了条件が満たされた場合、
ライントレース走行を停止する

動作実行クラス全体の振る舞いは
状態遷移図に記載

4.制御モデル

4-1.AIアンサー制御戦略の検討

数字を判別するための走行経路を検討する。

<走行方法の決定>
２種類の走行方法について検討した。
①直進しながらライン情報を取得する。
②エリアを分割しラインの有無を確認する。

(左)①の例
⇒外枠からラインまでの距離を取得する。
(右)②の例
⇒経路上のラインの有無を取得する。

⇒①の方が単純な経路であり、外的要因の影響を受けにくい。
よって、①を選択する。

<走行経路の決定>
スタート地点の違いから二つの候補を挙げた。
・辺の中心を通る経路
・対角線を通る経路

(左)中心の経路
(右)対角線の経路

⇒辺の中心を通る場合、取得距離のバラツキを抑えるためには正確な位置からのスタートが必要。角の方が正確な位置を捉えやすいので、対角線走行を選択する。

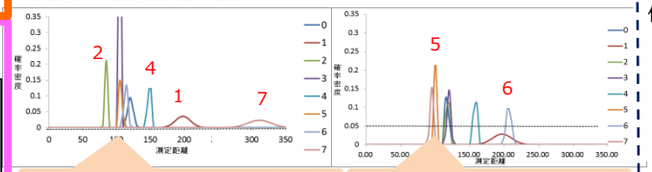
<数字判別が可能な走路の決定>
左出題数字
机上の測定から左下から右上の経路が有効と判断したが、誤差を考慮して、実際に走行調査を行った。
右出題数字
机上の測定から左下から右上と左上から右下の経路の二つの組み合わせが有効だが、実際に調査を行った。
詳細は4-2

4-2.AIアンサーの要素技術と制御戦略

実際に走行調査を行ったことで確実に数字を判別するための要素技術を確立した。また、要素技術をもとに、制御戦略を記載する。

要素技術Ⅰ 距離による数字判定

<目的・方法>
対角線走行で、ラインに到達するまでの距離を実際に測定し、数字を判別できることを確認する。
<結果と課題>　※左出題数字について記載
計測結果から最尤推定で確率密度関数を求めた(図4-1)。左下角からの距離で1,2,4,7を判別でき、右上角からの距離で5,6を判別できた(図4-1参照)。しかし、 0と3は確実に判別できるとは言えなかった。
<対策・・・要素技術Ⅱを参考>



左下の距離で1,2,4,7を判別可能　右上の距離で5,6を判別可能
図4-1 (左)左下からの距離の分布、(右)右上からの距離の分布

要素技術Ⅱ 時系列による数字判定

<目的・方法>
左下から右上までの対角線走行中に、白から黒に切り替わる回数が異なる(図4-2参照)ことを利用して0と3を判別できることを確かめる。そこで、毎ステップの色情報を黒1、白0として走行時の時系列データを保存する。
<結果>
測定した結果、0は2回、3は3回の切り替え回数となり、0と3を判別できた。
<工夫点>
図4-3に示すように、ノイズが検出されたため、直近5回のセンサー値の平均値を使用したことで一回の色の切り替えを正確に検出した。

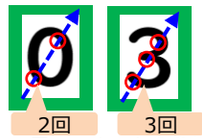


図4-2 0と3の対角線走行における切り替え回数の違い

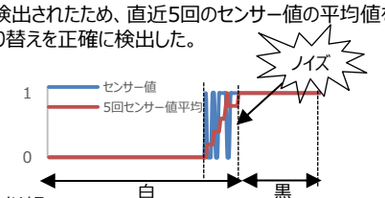


図4-3 観測時系列に移動平均を適用した例

左出題数字制御戦略

<走行経路>
対角線走行（図4-4に示す①～③の順。）
②→③までの距離が異常な場合、③→②に向けて再度走行する。

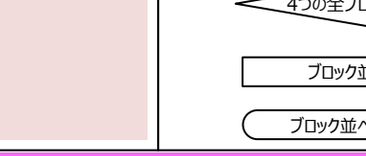


図4-4 左出題数字の走行経路

<データ収集方法>
Ⅰ 図4-5に示すd_1、d_2の距離を走行しながら測定する。
d_1：スタート地点から最初に黒に到達するまでの距離
d_2：最後の黒からゴール地点までの距離
Ⅱ 毎ステップの床の色を黒1、白0として時系列yを保存する。
<数字判定方法>
数字判定ロジックを図4-6に示す。
要素技術Ⅰで1,2,4,7,5,6を判別
要素技術Ⅱで0,3を判別

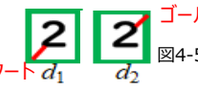


図4-5 計測対象

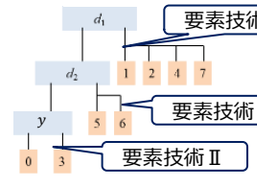


図4-6 判定ロジック

右出題数字制御戦略

右出題数字は２本の対角線走行で数字を判定する。
※1と7の判別を確実にするため、上側セグメントのみ別途検出する。

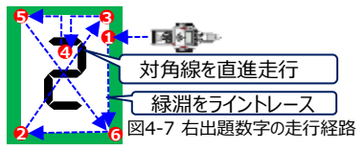


図4-7 右出題数字の走行経路

<走行経路>
対角線走行（図4-7に示す①～⑥の順。）
左出題数字と同様、異常値の場合、再走行する。
<データ収集方法>
Ⅰ 図4-5に示すd_1、d_2の距離を走行しながら測定する。
d_1：スタート地点から最初に黒に到達するまでの距離
d_2：最後の黒からゴール地点までの距離
Ⅱ 図4-8に示すd_3、d_4の距離を測定する。
<数字判定方法>
d_1～d_4の値で0,2,3,4,5,6を判別
上側セグメントで1,7を判別

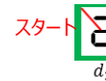


図4-8 計測対象

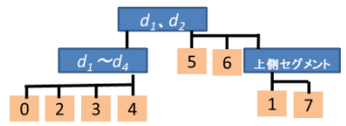


図4-9 判定ロジック

4-3.ブロック並べの要素技術

制御戦略でポイントに挙げる3つの要素技術について記載する

A.カメラシステムでの色判定

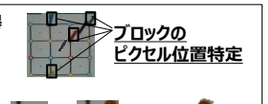
(注)分析モデル[課題分析1]記載の真上画像を使用

カラーブロック描画ピクセル(注)に対して誤判定しないピクセルを対象にHSV色空間のH成分の閾値によって色判定する。
<Point>
・カラーブロック描画領域の**重心位置のピクセル**を色判定に使用する
・重心位置近傍複数ピクセルの判別結果の**多数決**によって決定する

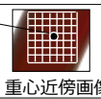
判定位置	赤(正)	黄(誤)	緑(誤)	青(誤)
①上面近傍	37	12	0	0
②重心近傍	48	1	0	0
③下面近傍	31	5	13	0

【検証】色判定に用いる対象領域
①ブロック上面近傍
②ブロック重心近傍
③ブロック下面近傍
赤ブロックの①～③に対して49ピクセル分の色判定を**検証**
正解の多い**②重心近傍ピクセル採用**

【前準備】重心位置計算のためのブロック領域取得
■4つのカラーブロック位置
初期位置コードから得られる座標コードから
カラーブロックが置かれているピクセル位置を特定
■ブロック領域の特定
キャリブレーション時にブロックの無し画像を取得し、ブロック有り画像との差分からブロック領域を特定



【判定手法】複数ピクセルの多数決による色判定
上記検証結果の黄色のような誤判定ノイズを削除するため、重心近傍49(7×7)ピクセルに対しての多数決で色判定を行う



重心近傍画像

B.ブロック運搬

黒色線分上をライントレースのみで目標へ到達不可能な場合にフリー走行する。
(分析モデル[2-2.走行体の動作定義]に記載の到達不可能条件時)
<Point>
・走行体とブロック間の**距離を確保**した経路をフリー走行(ブロックへの衝突回避)
・ブロック置き場**到達前は必ずライントレース走行**(ブロック設置・把持の安定性)

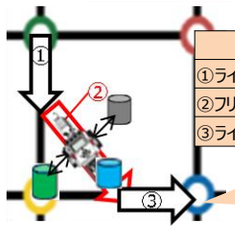


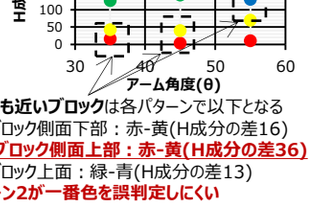
図　ブロック並べエリア

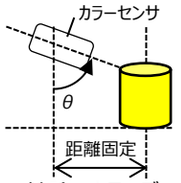
走行動作	終了条件
①ライントレース	線分中点まで移動
②フリー走行(旋回→直進)	黒ライン検出
③ライントレース	ブロック置き場色判定

ブロック置き場へは必ずライントレースで突入し、走行体が輪の中心を向くため安定◎

C.走行体カラーセンサによるカラーブロック色判定

カラーセンサによる色判定では、競技会場の明るさの影響を受けにくいHSV色空間の“H成分”により色判定を行う。本記述では、アームの角度による色判定精度の違いを検討し、判定精度の高い**アーム角度を決定**する。
<検証内容>
4色のカラーブロックに対してカラーブロックのどの位置を計測するかを以下の3パターンに分けてH成分を計測した。
・パターン1:ブロック側面下部(θ=35°)
・パターン2:ブロック側面上部(θ=45°)
・パターン3:ブロック上面(θ=55°)





<結果>
H成分が最も近いブロックは各パターンで以下となる
パターン1 ブロック側面下部：赤-黄(H成分の差16)
パターン2 ブロック側面上部：赤-黄(H成分の差36)
パターン3 ブロック上面：緑-青(H成分の差13)
→パターン2が一番色を誤判定しにくい

<結論>カラーブロック色判定時の**アーム角度を45°**に決定する

4-4.ブロック並べの制御戦略

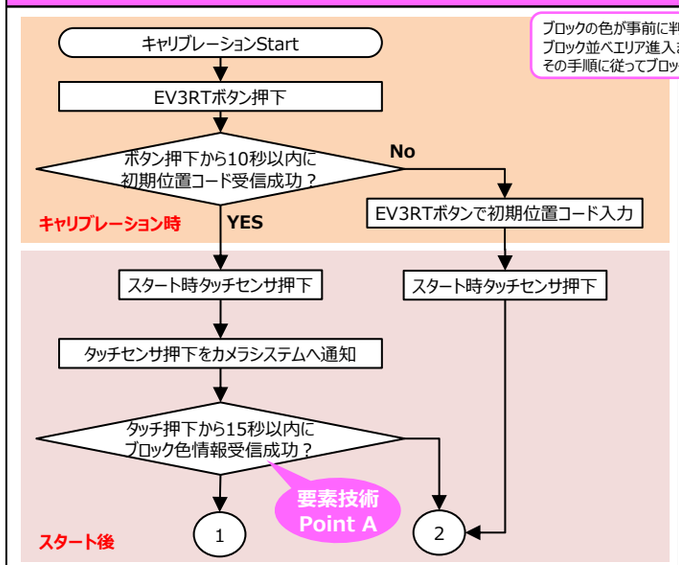
ブロック並べの走行体動作に関する記述する。
走行体は右記の通信途絶3パターンに分岐して攻略する。

	通信途絶パターン別の特徴		
	①通信途絶無し	②スタート前後に通信途絶	③完全通信不可
初期位置	通信により受信	通信により受信	走行体へ直接入力
色情報	攻略前に受信	ブロック位置到達時にカラーセンサで逐次取得	ブロック位置到達時にカラーセンサで逐次取得
運搬経路	4つのブロック運搬全体の最短経路	各ブロックごとに最短経路	各ブロックごとに最短経路

・キャリブレーション時の初期位置コード受信失敗は10秒タイムアウトで判定する
【理由】時間内にキャリブレーションを完了させるための最悪時間が10秒のため
タイムアウト(10秒)+コード入力(30秒)+センサ校正(10秒)<キャリブレーション時間(60秒)

・色情報受信失敗は15秒タイムアウトで判定する
【理由】ブロック並べ突入までに経路探索を終えるための最悪時間が15秒のため
タイムアウト(15秒)+運搬経路全探索(10秒)<ブロック並べエリア到達(30秒)

パターン分岐までの制御全体フロー



ブロックの色が事前に判明している場合の制御フロー

ブロックの色が判明していない場合の制御フロー