



チーム紹介、目標、意気込み

ETロボコン3回目の出場となる「下町ロボット」です！普段はバラバラの現場で働くメンバーですが、業務時間外や休日の限られた時間で地道に活動を続け、昨年はプライマリークラスでCS大会の舞台に立つことができました。

チームの熱い思いをチーム名の元になったドラマ「下町ロケット」の主人公・佃航平の名台詞を借りて…
「毎日壁にぶつかってばかりだ、だからこそ必死に腕を磨いて徹夜で開発に没頭して、次こそはって信じてる！」

アドバンストクラスへの参戦は初ですが、持ち前の雑草魂で今年こそCS大会優勝を目指します！

モデルの概要

・全般的な記述内容

すべての難所を確実の攻略するため、各難所を分析し、リスクを考慮して必要な機能を実現できるよう多くの機能盛り込みました

・モデルの構成

時間的制約からブロック並べの攻略にカメラシステムは重要と判断し、走行システムの他にカメラシステムも記載省略せず記載しました。

・特徴

時間がなくても難所はすべて開発する…という理念があり今後の活動を踏まえ、AIやカメラシステムの積極的に使用しています。L/Rの難所攻略のため、制御モデルにはAIアンサーの読み取り機能を記述しました。

モデルの構成

要求分析

- 具体的な開発目標にしたがって機能や非機能要求を「要求図」を使用して抽出しました

分析モデル

- ゲームの説明の後、モデルの要素定義をクラス図を用いて行い、その図から課題や特徴を抽出し、図の周りに抽出内容を記載しています。
- 分析クラスからゲームを解くために必要な動作を定義し、その内容から開発目標を達成するための指針を検討しその内容と結果をまとめました。
- 2-4では、2-1のクラス定義に追加する形で、要素を定義しました。追加したクラスは既存で使っていない色を使ってました。

設計モデル（構造）

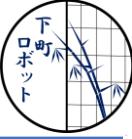
- 「走行システム」、「ターミナルシステム」、「カメラシステム」の3システムを作成した為、それぞれの設計を記載しています※ターミナルシステムの構造は割愛し「振る舞い」にスタートマシン図を記載した

設計モデル（振る舞い）

- 走行システム：全体の振る舞いと重要な「シミュレーション処理」と「運搬走行部分」の振る舞いを記載しています
- カメラシステム：スターターが操作する部分の制御と、ブロック色判定する部分の振る舞いを記載しています

制御モデル

- カメラシステム：ブロックの色をスタート前に認識する為に、自動的にカメラに映ったブロックの色をターミナルシステムを介して走行体に連携するシステムを実現しています
- AIアンサー：フォントの違う数字を確実に読み取るため、代替機能（AIを使わない判定）を踏まえ、直進制御やカラーセンサー読み取り領域縮小化の要素技術を使っています



1. 要求モデル

1-1. 開発目標 Rコースの目標リザルトタイムを達成する

開発目標は目標リザルトタイムの達成をもって競技を完了することとした。

$$(目標走行タイム) - (目標ボーナスタイム) = (目標リザルトタイム)$$

$$22.0\text{秒} - 28.0\text{秒} = -6.0\text{秒}$$

【全体基本戦略】

ブロック並べの完全攻略には時間が必要であり、攻略にかかる時間と獲得できるボーナスタイムの値を検討した結果、「初期位置のパワースポットのいずれの周囲にカーブロックを設置することを最優先とする」とを全体基本戦略としてシステムを構築することとした。検討したゲームにおける目標ボーナスタイムの内訳を表1.【目標ボーナスタイムの内訳】に表す。また、ブロック並べに関する獲得ボーナスタイム分析結果は分析モデル2-3.指針に記す。

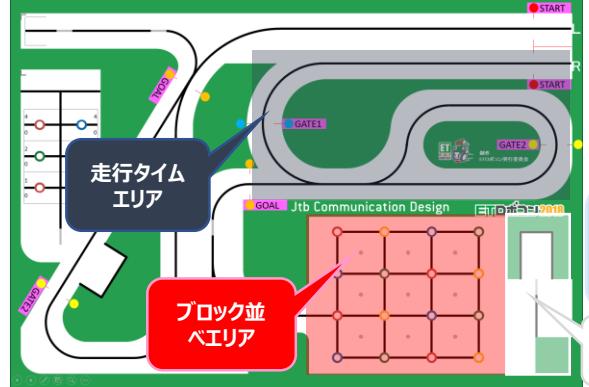


図1-1.【R-course内のエリア定義】

ブロック有効移動	4個	13.0秒
パワースポット設置	5個	10.0秒
直角駐車場停止		5.0秒
合計		28.0秒

表1.【目標ボーナスタイムの内訳】

【コース定義】
Rコースの競技特徴からコースを分割してそれぞれを「エリア」として定義し本モデル内で呼称する。エリアを分割し開発、評価することで効率化を図る。なお、エリアの範囲と呼称を図1-1.【Rコースのエリア定義】に示す。

直角駐車場
エリア

1-2. 機能検討 要求図を用いて最重要1-スケスとした「Rコースの目標リザルトタイムを達成する」の達成を上位目標としてレーザーを用いた。必要な機能要求、非機能要求、要素技術の抽出を行った。



1-3. 要件定義

開発目標を達成するために開発するシステムを整理する。
1-スケス図を用いて、システムが提供する機能を明示する。

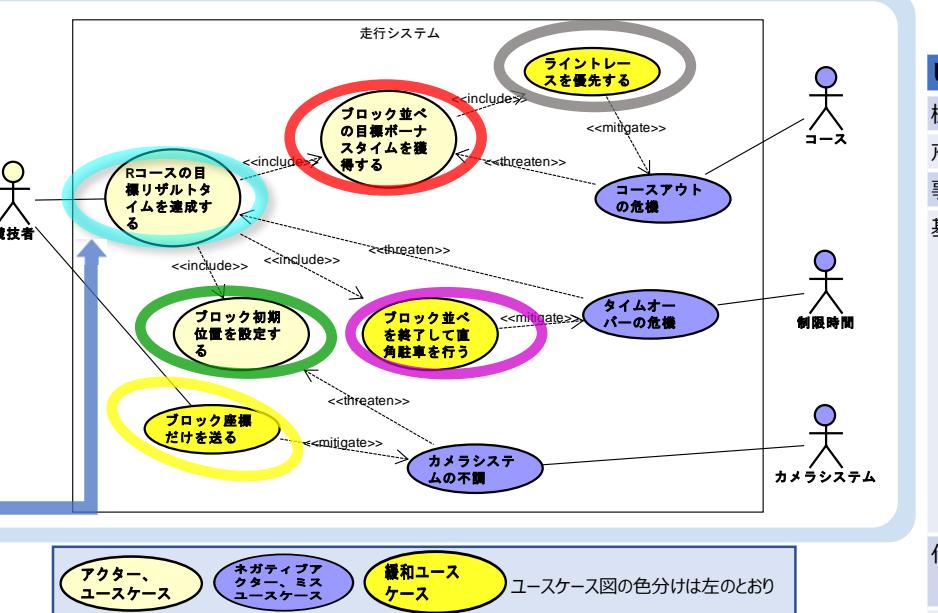


図1-2.【ユースケース図】

開発目標を達成するための走行システムを開発する。
本走行システムは走行システム(走行体)、ターミナルシステム、カメラシステムより構成され、競技者によって操作される。システム構成は設計モデル3-1.コンポーネント図に表す。
本走行システムの最重要ユースケースを開発目標でもある「Rコースの目標リザルトタイムを達成する」とし必要な要件をユースケース図を用いて分析した。作成されたユースケース図を図1-2に示す。

【R-courseの目標リザルトタイムを達成する1-スケス記述】

UC名 R-courseの目標リザルトタイムを達成する

概要 R-courseの競技で目標リザルトタイムを達成する

アクター 競技者

事前条件 R-courseのキャリブレーションが開始されていないこと

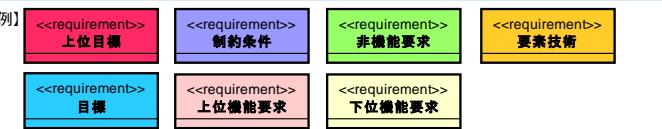
- 基本プロセス
- 競技者は走行システムを起動する
 - 競技者は初期位置コードを入力する
 - 走行システムは初期位置コードをデコードする
 - 走行システムはデータコードした結果を保持する
 - 走行システムはカメラシステムよりブロック色情報を取得する
 - 走行システムはR-courseの走行タイムエリアをクリアする
 - 走行システムは直角駐車エリアをクリアする
 - 走行システムは終了する

代替プロセス カメラシステムの不調によりブロック色情報が得られない場合
・ブロック座標情報だけを送信する

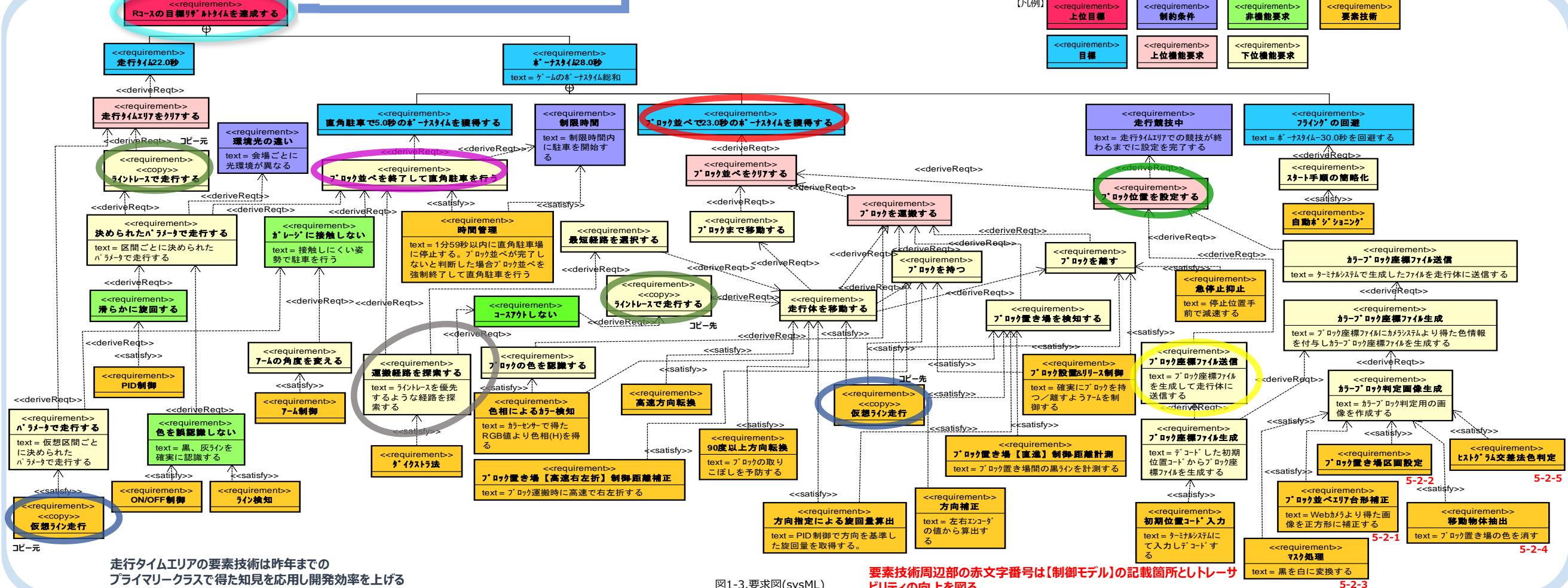
事後条件 システムは終了している

- 脅威
- コースアウトの危機
 - タイムオーバーの危機
 - カメラシステムの不調

【凡例】



1-3. 要求図(sysML) 要素技術周辺部の赤文字番号は【制御モデル】の記載箇所とトレーリティの向上を図る。



走行タイムエリアの要素技術は昨年までの
プライマリークラスで得た知見を応用し開発効率を上げる

図1-3.要求図(sysML)

要素技術周辺部の赤文字番号は【制御モデル】の記載箇所とトレーリティの向上を図る。

2. 分析モデル

2-1. ゲームの要素定義

ゲームの構成要素の特徴や関係をモデルを使って定義する

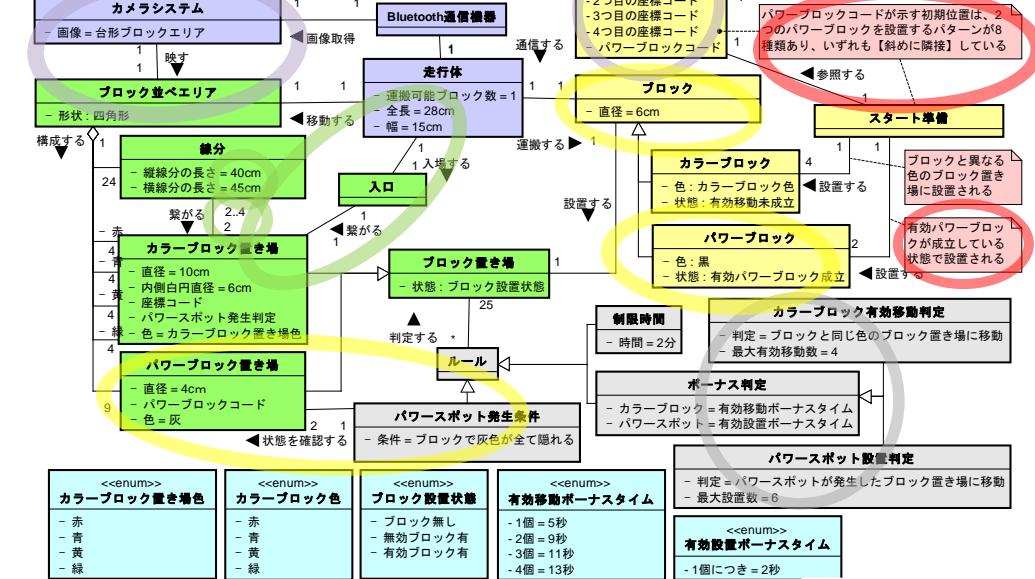
<ゲーム説明>

制限時間内に、ブロック並ベエリア上のブロック置き場にあるブロックを移動し、ブロックの移動先の結果によりボーナスタイムが獲得するゲーム

- ①ブロックの位置はランダムに決定され事前に初期位置コードとして公開される
※カラーブロックはブロックと同じ色のブロック置き場には設置されない
- ②カラーブロックと同じ色のブロック置き場に移動させるとボーナスを獲得できる
- ③②の移動先にパワースポットが発生した場合、追加ボーナスを獲得できる

<構成要素の特徴や関係の定義>

関係や特徴を整理し、ゲームのポイント・課題を導出する



2-1-7. パワーブロック移動難度

- ①パワーブロック置き場に繋がる線分がなく、ライントレースで移動できない。
- ②有効判定を受けるには「直径6cmのブロックで、4cmのパワーブロック置き場を完全に隠す」必要がある



2-1-6. 完全攻略の条件

- ①パワーブロック周辺のカラーブロック置き場に移動する。
- ②有効判定を受けるには「直径6cmのブロックで、4cmのパワーブロック置き場を完全に隠す」必要がある



2-1-5. 線分の長さ

- ①パワーブロック周辺のカラーブロック置き場に移動する。
- ②パワーブロックを隣接する上下左右のいづれかに移動させる



2-1-4. パワーブロックの初期設定状態について

- ①初期位置は斜めに隣接する
- ②パワーブロックの初期位置は有効パワーブロックの状態で設置される。



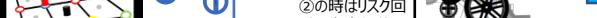
2-1-3. 線分による移動不可

- ①パワーブロック置き場は線分が2つ以上ある
- ②移動不可



2-1-2. ブロック並ベエリア入場不可

- ①パワーブロック置き場は線分が2つ以上ある
- ②移動不可



2-1-1. スタート前: ブロックの色認識

- ①初期位置コードから色までは判らない。
- ②ブロックの色はスタート直前にブロックが設置される事で確定するが、その時点色を認識出来るのは、カメラシステムのみ



2-2. 走行体の動作定義

ゲームを解く上で前提となる走行体の動作を定義する

コース内の形状や距離、ルール上の制約、実際の走行結果より導出された動作

2-2-1. ブロック位置 + 情報の取得

- ①初期位置コード入力 → ②位置情報をカメラシステムへ送信 → ③位置 + 情報を受信 → ④走行体へ送信



2-2-4. カラーブロック取得 & 方向転換

ブロックをアームから外さないように、ゆっくり回転し90°回転する毎に持ち直し（少し前進）の動作を入れる。主にブロックを保持して180°ターンする時に実施する動作



2-2-2. ライントレース～ブロック置き場直進

- ①黒線は距離を計測しながらライントレース走行する
- ②ブロック置き場は色を検知 → 一定距離を直進する



2-2-5. 仮想ライン走行

- ①車輪を前後に回転させる事でその場で方向転換する
- ②縦横の線分の距離が異なる為、仮想ライン走行（方向指定走行）する時の角度はそれぞれ異なる



2-2-3. ブロック置き場右左折

- ①ブロック置き場の色を検知
- ②片輪のみ前進させ90°方向転換する → 走行再開



2-2-6. ブロック設置 & リリース

- ①車輪を前後に回転させる事でその場で方向転換する
- ②前後のブロック置き場にブロックがある場合を想定して接触しないように中間地点まで後進した後、ターンする

2-2-7. ブロック取扱い

- ①ブロックをアームから外さないように、ゆっくり回転し90°回転する毎に持ち直し（少し前進）の動作を入れる。主にブロックを保持して180°ターンする時に実施する動作

2-3. 指針

定義したゲームの要素と走行体の動作を前提にゲームを解く有効な指針を記述する

検討方針：ゲーム中に起こりうる否定的な事象の影響を最小限にし、攻略の機会を最大限に利用する

2-3-1. ボーナスタイム獲得戦略～完全攻略を目指すか？【2-1. ゲームの構成定義】から導出した戦略を比較検討する～

【比較結果】案①は運搬先が1つに限定されるため有効移動には距離が必要になる。またパワーブロックの移動に失敗するリスクある。

案②を採用する上記を踏まえ、開発目標達成のために案②を採用する

初期位置のパワースポット(発生予定期地)にカラーブロックを有効移動させる

【戦略一指針】

初期位置のパワースポット(発生予定期地)にカラーブロックを有効移動させる

【新】課題

仮想ラインを追加したことで新しく「コースアウトのリスク」が発生した

走行距離が長くなる→制限時間をオーバーする可能性がある

【残】課題

通常はライントレースで走行し、移動経路が無い時のみ仮想ラインを使用する

方針 ライントレース優先走行

【デメリット】コースアウトリスク高・移動不可状態:無・走行距離:長い

【新】課題

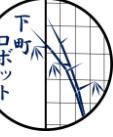
全カラーブロック(4つ)の運搬順序を全てシミュレーション(4x6=24)し、コストが少ない順序を採用する

【戦略一指針】ブロック並ベエリアまでに解析が終わらない場合算出した中で一番コストが少ない順序を採用する

【新】課題

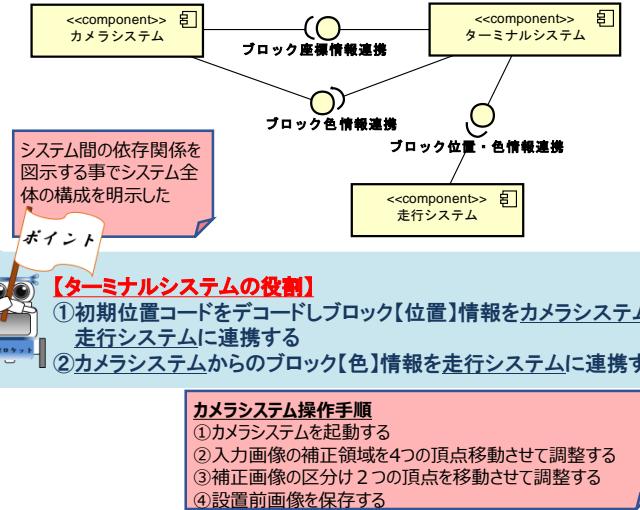
全カラーブロック(4つ)の運搬順序を全てシミュレーション(4x6=24)し、コストが少ない順序を採用する

【戦略一指針】ブロック並ベエリアまでに解析が終わらない場合算出した中で一番コスト

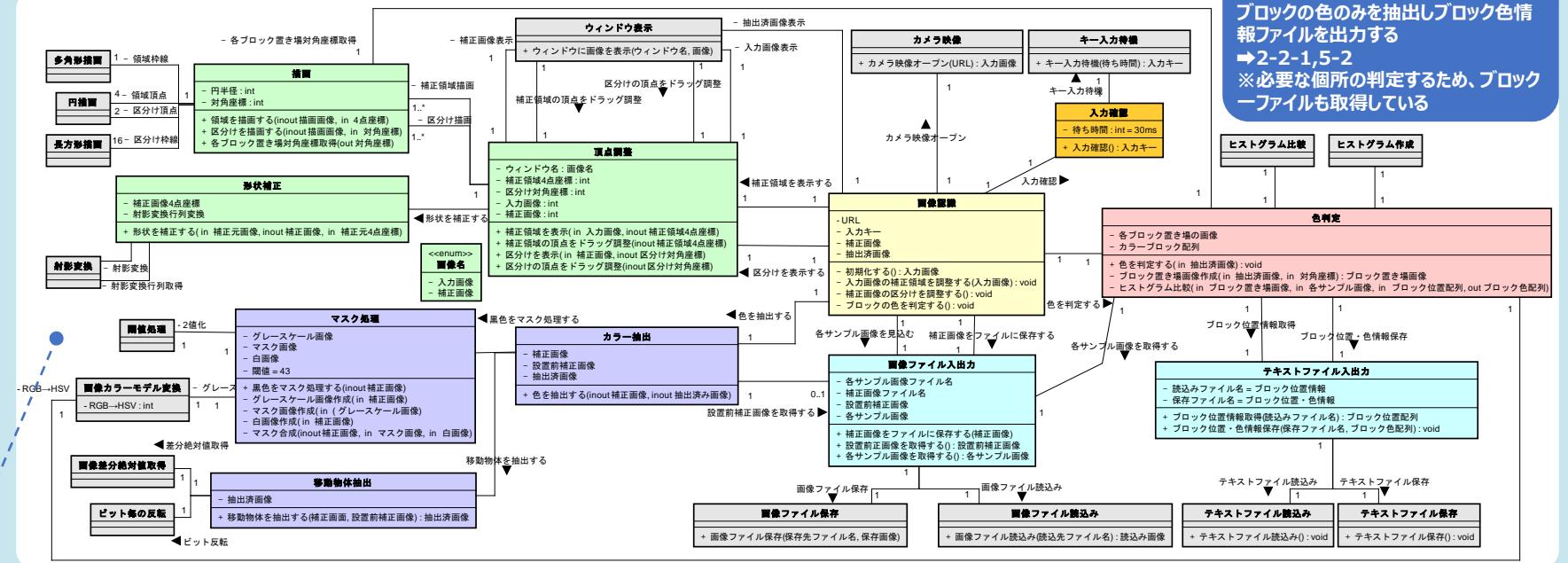


3. 設計モデル（構造）

3-1. ターミナルシステム：コンポーネント図

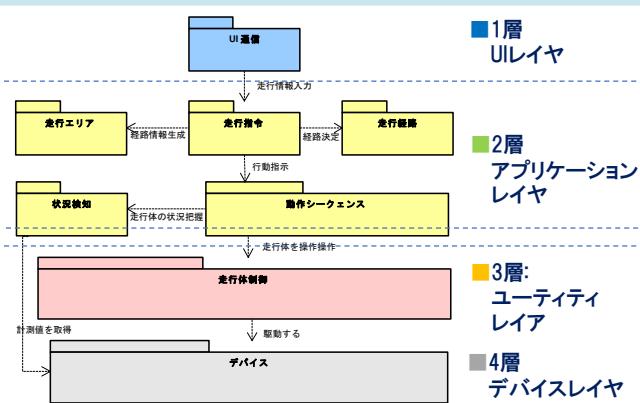


3-2. カメラシステム：クラス図



3-3. 走行システム：構成概要図

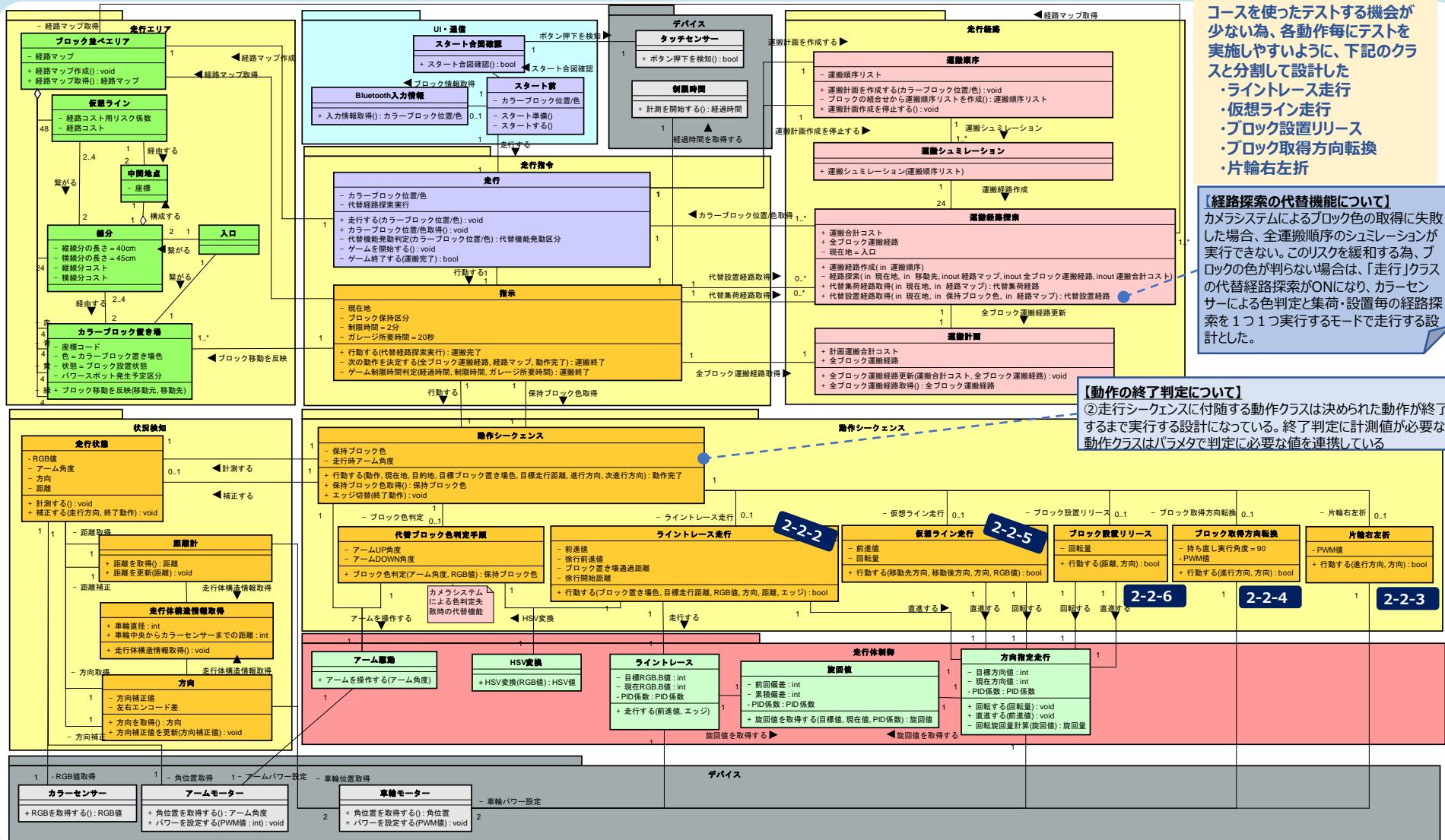
システム構成の意図
品質を維持し、平行開発を可能とする為、下記のレイア構成で設計しました。3層のユーティリティは各担当が使いまわせる用独立性を高めています。



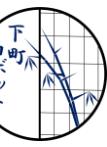
パッケージ	役割
UI・通信	Bluetooth通信でタッチセンサー等の入力機器からの情報を処理し、後続機能へ連携する
走行指令	走行開始後、全ての動作の指示を出す
走行経路	効率よくブロックを集荷・運搬する為、計画を立てる
走行エリア	ブロック並べエリアの構造から走行経路情報を生成・保持する
状況検知	各デバイスの情報を取得し、動作に必要な形で連携する
動作シーケンス	ゲームを解くために必要な複数の動作を順番に実行する
走行体制御	デバイス情報を加工して駆動デバイスを制御する
デバイス	各種センサー値を提供、各駆動デバイスを制御可能とする

上から、1層目がUIレイヤ、2層目がアプリケーションレイヤ、3層目がユーティリティレイヤ、4層目がデバイスレイヤとしています。アプリケーションレイヤは、今回のコースの分析に基づく機能を提供します。ユーティリティレイヤは上位アpriが変わっても影響がない汎用的な機能を提供します。

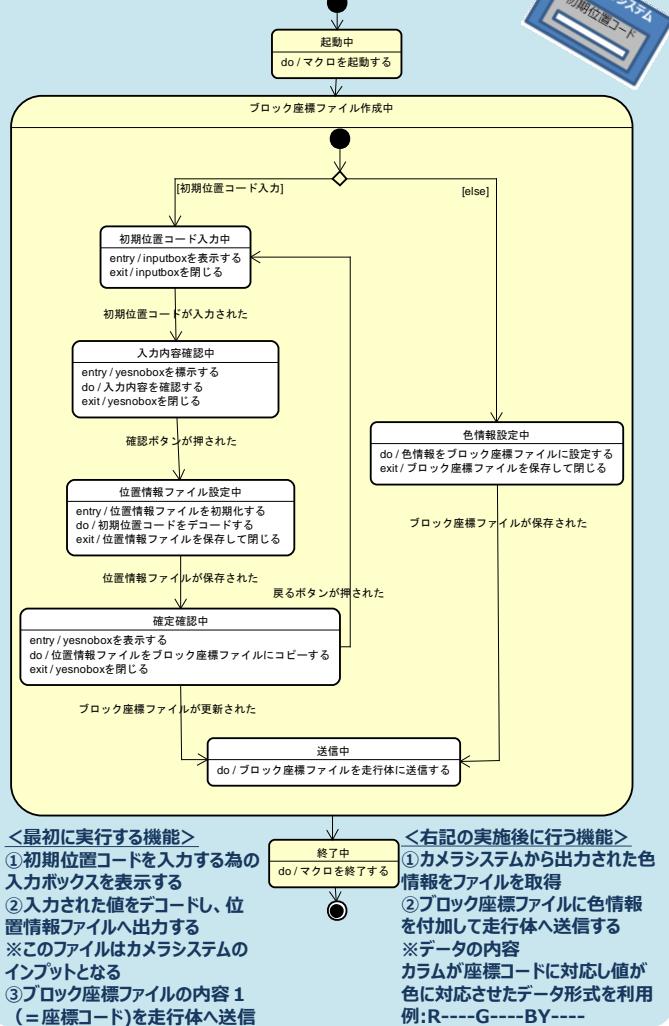
3-4. 走行システム：クラス図



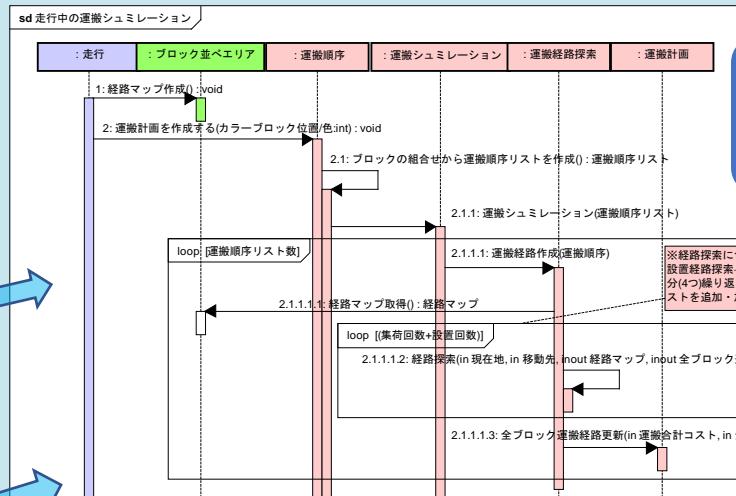
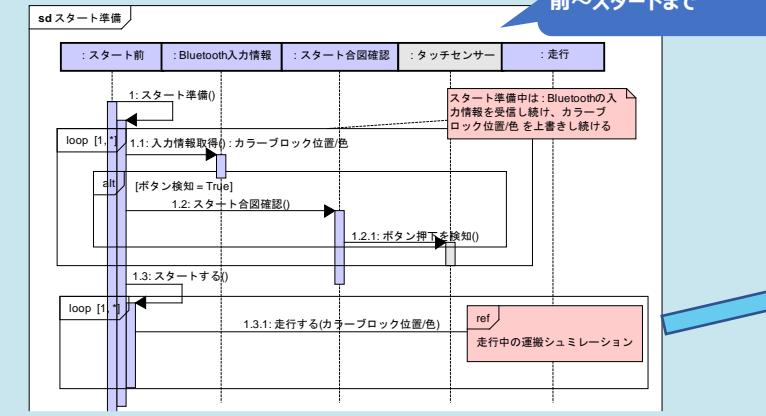
4. 設計モデル(振る舞い)



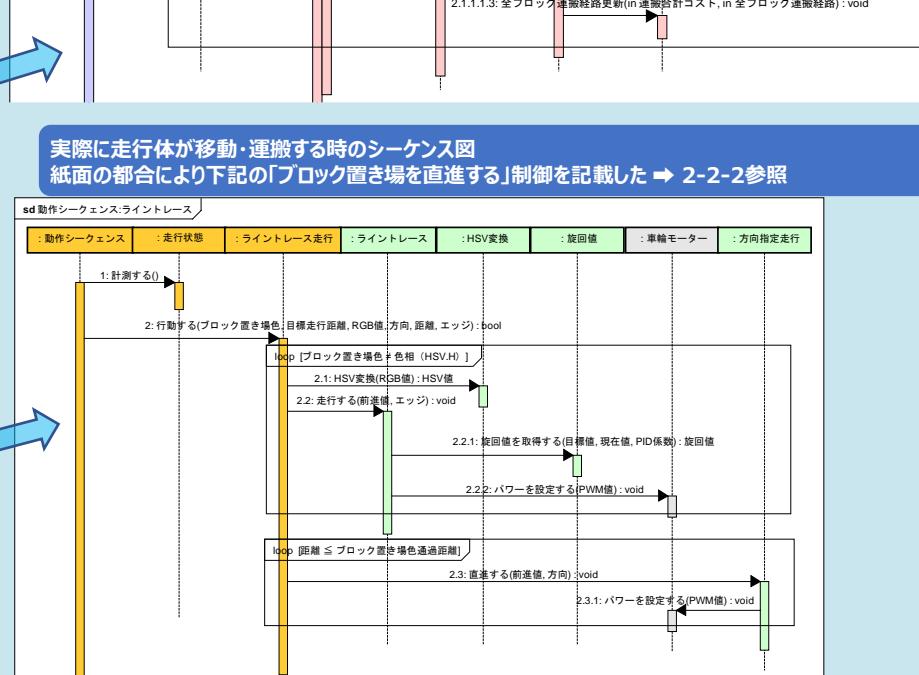
4-1. ターミナルシステム : 状態遷移図



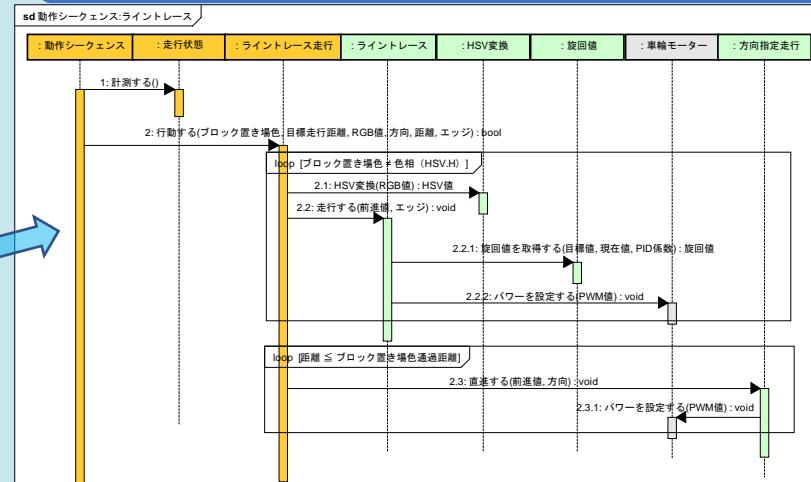
4-3. 走行システム : シーケンス図



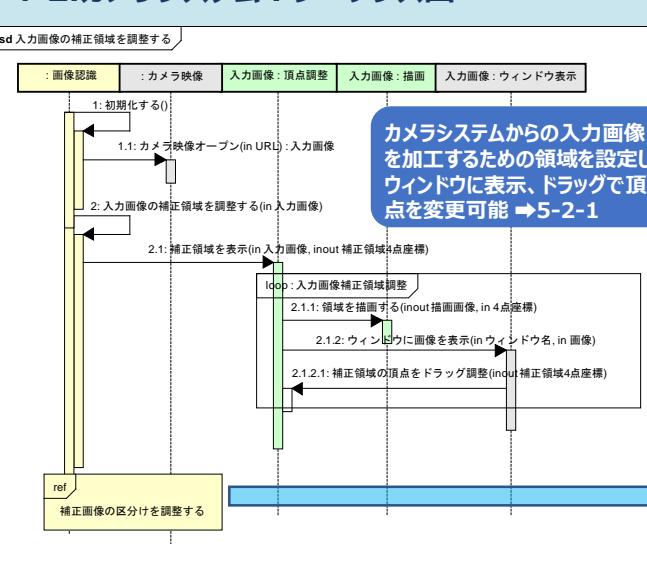
走行競技中に全運搬順序をシミュレートし最適な運搬経路を取得する機能 ➔ 2-3-4



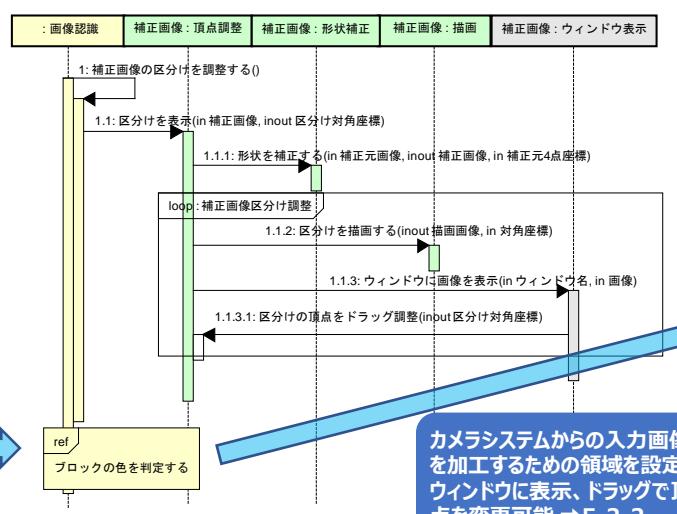
実際に走行体が移動・運搬する時のシーケンス図
紙面の都合により下記の「ブロック置き場を直進する」制御を記載した ➔ 2-2-2参照



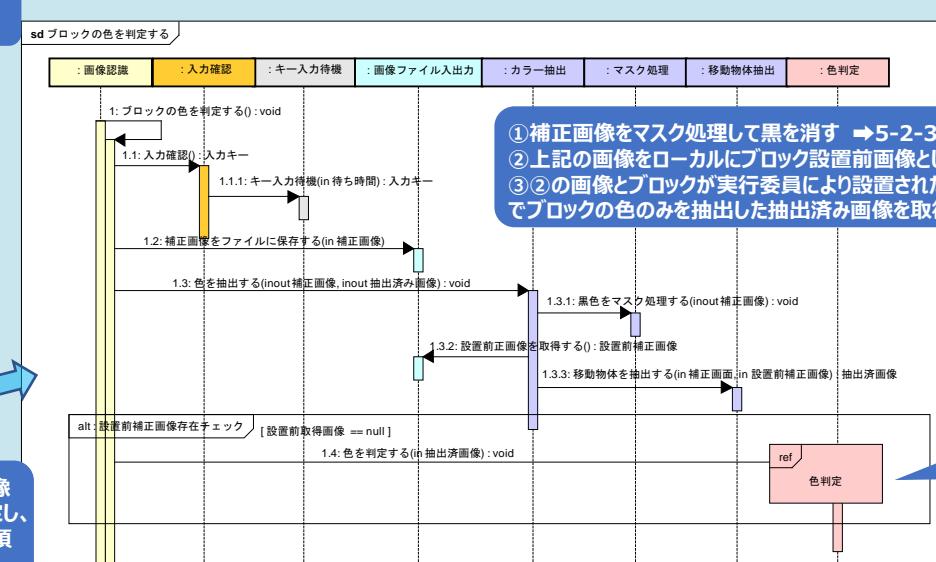
4-2. カメラシステム : シーケンス図



4角形に補正された入力画像 = 補正画像として画面に表示されるが、その補正画像上のブロック置き場を区別する為に、区分描画を表示し調整可能とする



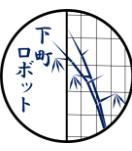
カメラシステムからの入力画像を加工するための領域を設定し、ウィンドウに表示、ドラッグで頂点を変更可能 ➔ 5-2-2



① 補正画像をマスク処理して黒を消す ➔ 5-2-3,
② 上記の画像をローカルにブロック設置前画像として保存する
③ ②の画像とブロックが実行委員により設置された後の画像を引き算する事でブロックの色のみを抽出した抽出済み画像を取得する ➔ 5-2-24

色判定のシーケンスは紙面の都合により割愛した ➔ 5-2-5



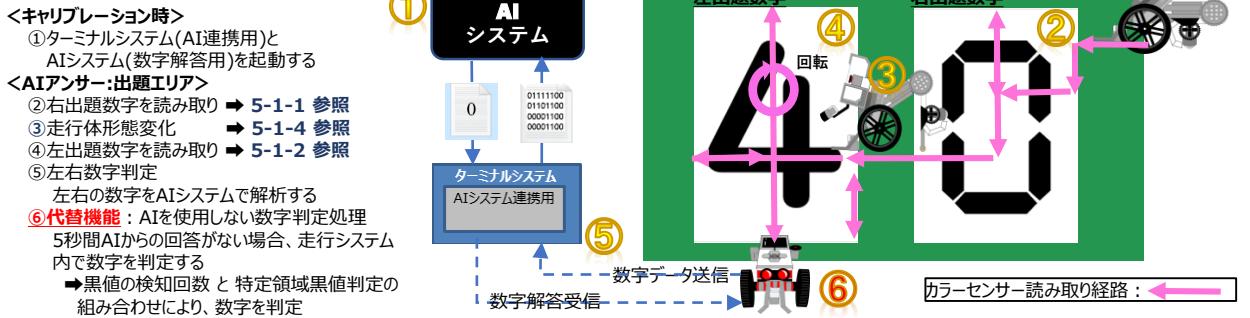


5. 制御モデル (AIアンサー/ブロック並べ)

5-1. AIアンサー数字情報読み取り機能

機能 フォントの異なる2つの出題数字を後続機能で判定/解答するために出題数字の特徴をカラーセンサーで確実に読み取る
予めAIからの回答がないリスクを考慮し、代替機能(AIを使わない数字判定)を実行可能な数字情報を取得する

全体戦略



制御戦略 ~機能実現手順~

5-1-1) デジタル数字読み取り

機能: 右記の読み取り経路を走行し、デジタル数字を一意に決定可能なセンサー値を取得する
※②以外は5-1-3を使用して移動する

- 白を検知後、-90度旋回しPID制御でライントレースを実施する
理由: ライントレース後は正確に下方向を向いている為
自己方向を初期化し5-1-3の精度を向上させる
- 緑を検知するまで直進
- 上方方向を向いたままバックで走行する
理由: 方向転換は走行に誤差が生じる可能性がある為、極力実施しない
- 左方向に向かって直進し前進右出題数字エリアを抜ける5-1-2へ

5-1-2) 通常数字読み取り

機能: 通常数字は特徴領域の幅が少なく誤検知の可能性があるため、5-1-4で一意に決定可能なセンサー値を取得する
※②以外は5-1-3を使用して移動する

- 白を検知後、5-1-4を実施する
- 自己方向を初期化する為、一旦下方向へライントレースし下辺の線を検知後初期化へバック走行で元の位置に戻る
- 左方向へ直進し左辺の線を検知後、中央までバック走行で戻る
- 上方へ直進後、1回転する(位置は読み取り経路参照)
理由: 十字にカラーセンサー値を取得しても「3」と「5」の見分けがつかない為、ココで1回転のセンサー値を取得する
- 再び上方向へ直進し線を検知後、下辺までバック走行する

要素技術

5-1-3) 直進・旋回 制御

課題 非ライントレース状態ではモーターに同じパワーを与えるもモーターの性能の違いやタイヤの摩擦により同一の回転を行わない → 正確に方向転換や直進をしたい

対策 旋回: 左右エンコードの差から自己方向を算出し、その値を基準に目標の方向まで旋回をする

直進: 上記で取得した自己方向と直進方向の差を比例量としPID制御で旋回量を決定する。これにより左右モーター回転にズレ(方向のズレ)が生じた場合でもそのズレを修正しながら直進する

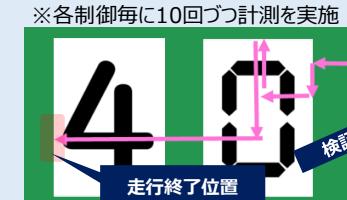
自己方向 = 左右エンコード差 / (車輪直径 × 円周率 × 360)

比例量 = 自己方向 - 直進方向

旋回値 = (P制御係数 × 比例量) + (I制御係数 × 積分量) + (D制御係数 × 微分量) ※積分量と微分量の算出方法は省略

検証 直進制御: 有と直進制御無(直進:pwm:0)

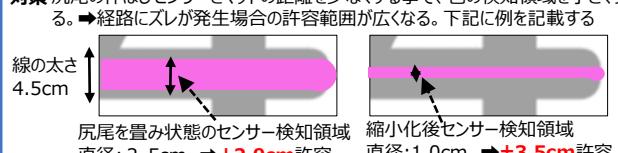
で下記の読み取り経路を走行し、走行終了時の位置バツキを計測した
※各制御毎に10回づつ計測を実施



5-1-4) カラーセンサー検知領域縮小化 制御

課題 尻尾を置んだ状態の走行体はカラーセンサーとマットまでの距離が長い為、検知する領域が広い(2.5cm)。5-1-2では数字の特徴を捉える経路を走行するが経路にズレが生じた場合、想定外の判定してしまうリスクがある

対策 尻尾の伸びセンサーとマットの距離を少なくする事で、色の検知領域を小さくする。→経路にズレが発生場合の許容範囲が広くなる。下記に例を記載する

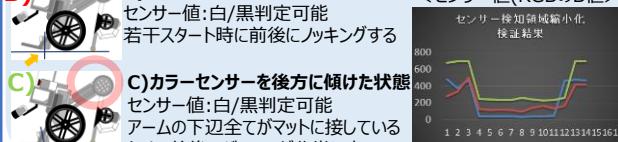


検証 A～Cのアーム角度でセンサー値を取得し、センサー値を確認した

A) カラーセンサーを前に傾けた状態 <検証時の走行した経路>

B) カラーセンサーを真下に向けた状態
センサー値: 白/黒判定可能
若干スタート時に前後にノックしている

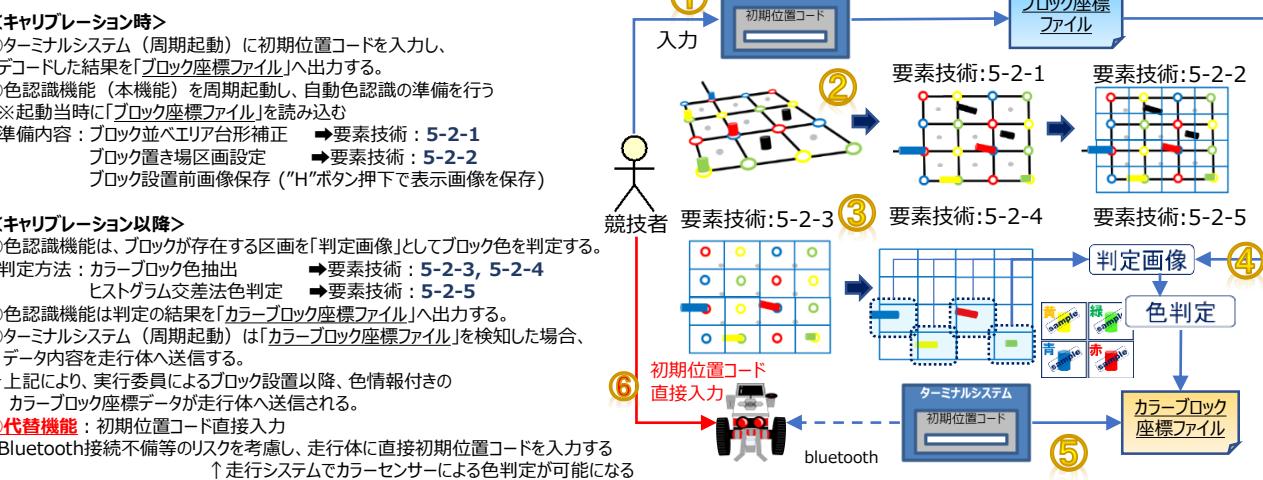
C) カラーセンサーを後方に傾けた状態
センサー値: 白/黒判定可能
アームの下辺全てがマットに接しているため、前後のバランスが非常に良い



5-2. カメラシステムによるカラーブロック色認識機能

機能 ブロック並ベエリアの画像とブロックの位置情報から、カラーブロックの色を判定しスタートまでに「カラーブロック座標ファイル」を出力する。

制御戦略 ~機能実現手順~



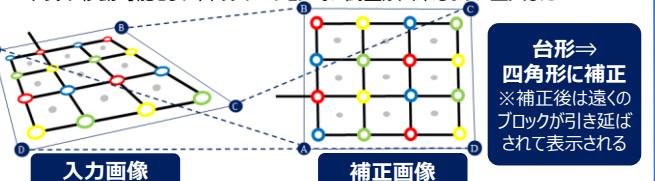
要素技術

5-2-1) ブロック並ベエリア台形補正

課題 ブロック並ベエリアは遠近法により傾いた台形になっており、座標を区切ることが難しく、また遠くにあるブロックの面積が小さい。→四角形に補正したい

対策 画像を射影変換し台形を四角形に補正した「補正画像」を生成する。

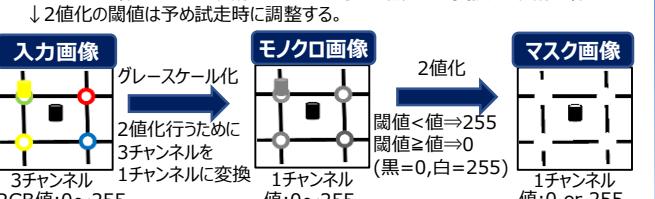
工夫 Webカメラの画角は固定ではない為、補正領域の頂点4つをドラッグ移動可能とし、キャリブレーション時に調整が出来るように工夫した



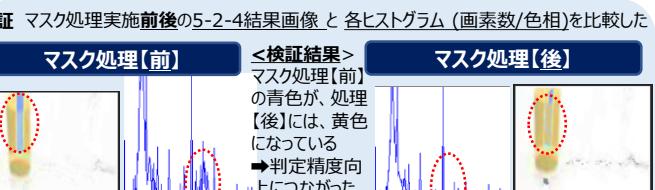
5-2-3) マスク処理 ~黒⇒白に変換~

課題 5-2-4ではブロックと重なる背景に白以外の色(右記の図では黒ラインとブロック置き場)が存在すると、その部分が別の色に変換されてしまう → 黒ラインを白を変えたい

対策 补正画像にマスク処理(値:0のエリアをコピー対象外にするマスク画像を指定)を施し、予め作成した白の画像にコピーする事で、黒を白に変換した画像を作成する。
↓2値化の閾値は予め試走時に調整する。



検証 マスク処理実施前後の5-2-4結果画像と各ヒストグラム(画素数/色相)を比較した



5-2-2) ブロック置き場区画設定

課題 補正画像に複数のカラーブロックが存在する為、ブロック毎に色判定が出来ない
→ブロック置き場毎に区画別けし、座標コードに対応させたい

対策 対角の2点から4角形を作り各辺を4等分した地点から線を引くことで16個区画を生成し、これを座標コードに対応させる

工夫 キャリブレーション時に区画を調整できるように、対角の2点をドラッグ移動させる事で全区画の位置と長さを容易に調整できるように工夫した

座標コードに対応

5-2-4) 移動物体抽出 ~ブロック置き場→白に変換~

課題 5-2-3の処理後の画像にはブロック以外にブロック置き場が存在する。色判定の精度を上げる為、ブロック置き場の色を白に変換したい

対策 ①キャリブレーション時にブロック設置前画像を保存する。
②ブロック設置前画像とブロック設置後画像を引いて絶対値を反転させる
上記により移動物体(設置されたブロック)のみが残り背景は白になる



5-2-5) ヒストグラム交差法色判定

課題 Webカメラへブロックまでの距離や角度によりブロックの形状が変わる。
また判定画像内には、対象のブロック以外にも消し切れない色が存在する。
→形状のバラつきや色の混在があっても、正確にブロックの色を判別したい

対策 ①4色のsample画像と判定画像の類似値をヒストグラム交差法にて算出
②類似値が一番高い(1に近い)sample画像の色を判定画像のブロック色とする

