



### チーム紹介、目標、意気込み

「はれかぜR」は、津山工業高等専門学校の専攻科に通う学生3名と教員1名で構成されたチームです。前大会は「はれかぜ」というチーム名で参加しましたが、満足な結果が得られませんでした。そこで、同じメンバーで「はれかぜR」という名前で再挑戦します。「R」は「Revenge(リベンジ)」の意味を込めて加えました。

前大会の結果よりも好成績が残せるように、反省点を踏まえて、試行錯誤と実験を繰り返してきました。「はれかぜR」が大会で掲げる目標は、地区大会で総合優勝を果たすことです。この目標を達成するために、今までの成果をすべて出し切り、コースの完走を目指します。また、学生3名は今年度で専攻科を修了するので、最高の思い出にしたいと思います。

### モデルの概要

- 「はれかぜR」が選択したモデリング対象は「コースを完走する」です。
- 本モデル図では、リスク分析においてコースの完走を果たすために対処しなければならない脅威を抽出しています。そこから、以下の重要な機能を定義しました。
  - a. リモート指示を受けキャリブレーションを行う
  - b. PID制御をする
  - c. 自己位置推定をする

これら3つの機能の実現を目指し、システムの構造や各機能の振る舞いにおいてUMLを用いた図で表現しています。上記の機能から、前大会で達成できなかったコースの完走を確実に成し遂げることが期待できます。

- インスタンスを管理しやすい構造にするため、Singletonパターンを取り入れたことは**独自性**があり「はれかぜR」のアピールポイントであると自負しています。

### モデルの構成

「はれかぜR」はコースの完走において、以下の目標を達成します。

#### 100%の精度でコースを完走する

次にモデルの構成について説明します。

##### 1. 機能モデル (1ページ目)

- リスク分析において、コースを走行する上で脅威となる要因を抽出し、それらの対策を列挙している
- リスク分析を基に、機能分析として目標達成に必要な機能をユースケース図にまとめ、要となる「コースを安定走行する」機能の詳細をユースケース記述で示している
- アクティビティ図を用いて、ユースケース記述で記した振る舞いの順序を図示する

##### 2. 構造モデル (2ページ目)

- 機能分析により明確にした「コースを安定走行する」機能を実現できるようにクラス設計を行っている
- クラス間において関連が強いグループをパッケージ図で整理している
- 詳細なクラス構成をクラス図で表現する

##### 3. 振る舞いモデル (3, 4ページ目)

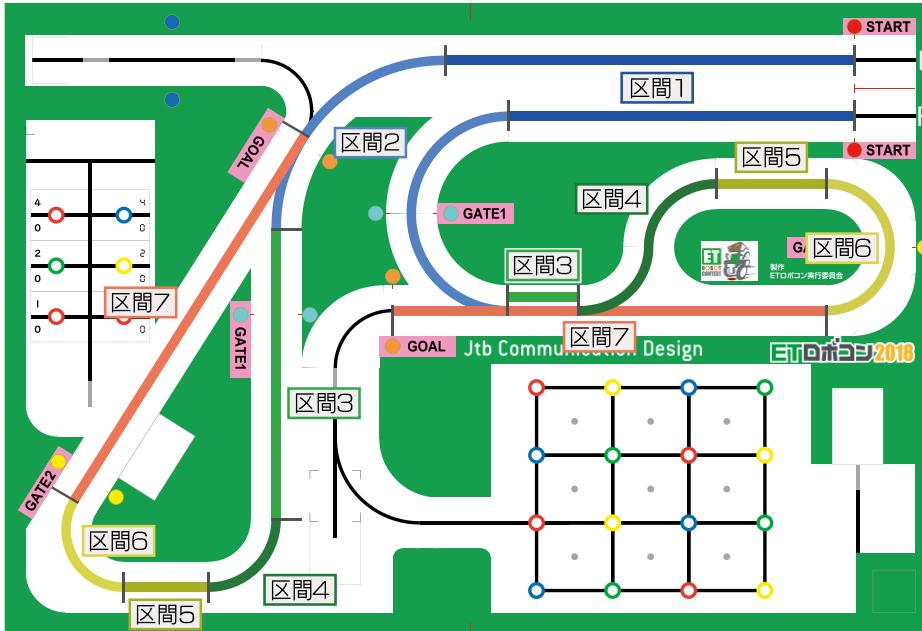
- 機能モデルのアクティビティ図を基に、走行処理に焦点を置いた詳細な状態の変化をステートマシン図に示す
- ステートマシン図の各状態について、シーケンス図により時間軸に沿ってインスタンス間のやりとりを明確にしている

##### 4. 工夫点 (5ページ目)

- 「コースを安定走行する」機能を実現する上で、プログラムの信頼性や安定性の向上を図るため、ケーブル脱落検知を取り上げている
- ケーブル脱落検知における振る舞いをシーケンス図で表す

## 1. リスク分析

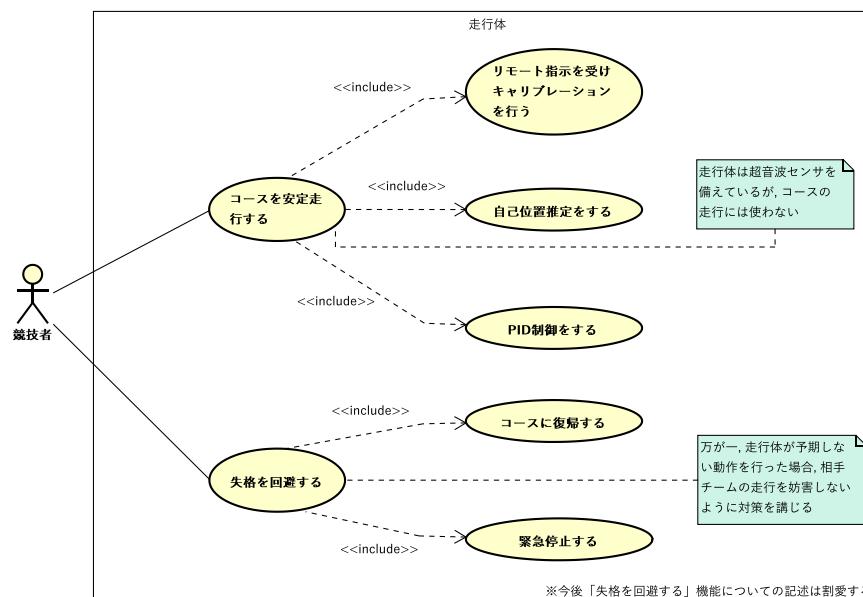
目標を達成する上で、走行上の脅威となる部分の抽出を行う。ここで、リスク分析の円滑化を考慮し、コースを以下のように区切る。



脅威	脅威理由	対策
キャリブレーションをタッチセンサで進めていく	競技者の手による影が発生する。(カラーセンサに影響) 競技者の手により走行体が揺れる。(ジャイロセンサに影響)	Bluetooth通信を介したりモード指示を送る。
走行を開始する	開始直後に走行体が転倒するおそれがある。 開始直後に走行体がコースアウトするおそれがある。	キャリブレーションを正確に行う。
サンプルコードに合わせて、ON/OFF制御でコースを走行する	無駄な走行が生じる。	ON/OFF制御より無駄な走行を抑えることができるPID制御を用いる。
PID制御で走行する	コースの直線部分(区間1,3,5,7)とカーブ部分(区間2,4,6)において走行の安定性が変わる。	左の図のとおりに区間分けすることで、各区間に適したパラメータを設定する。そこで、自己位置推定を導入する。

## 2. 機能定義

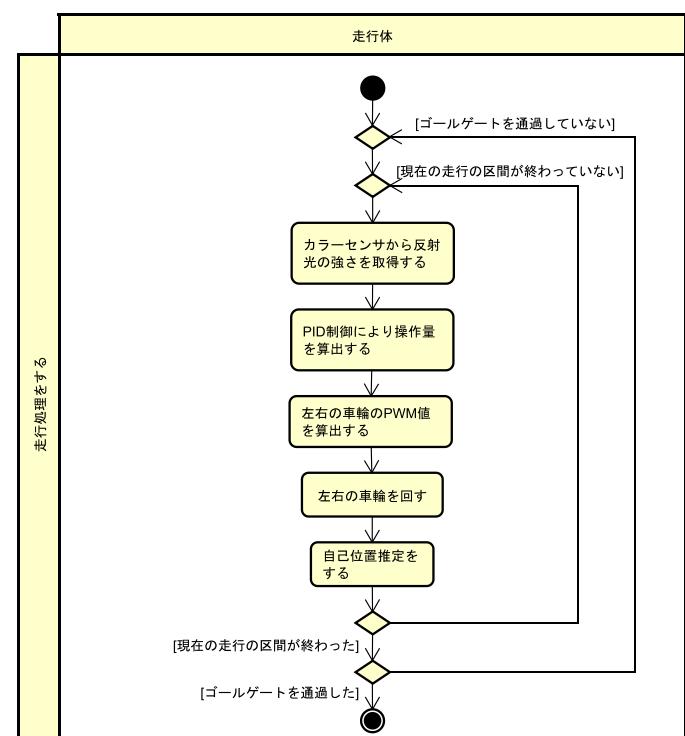
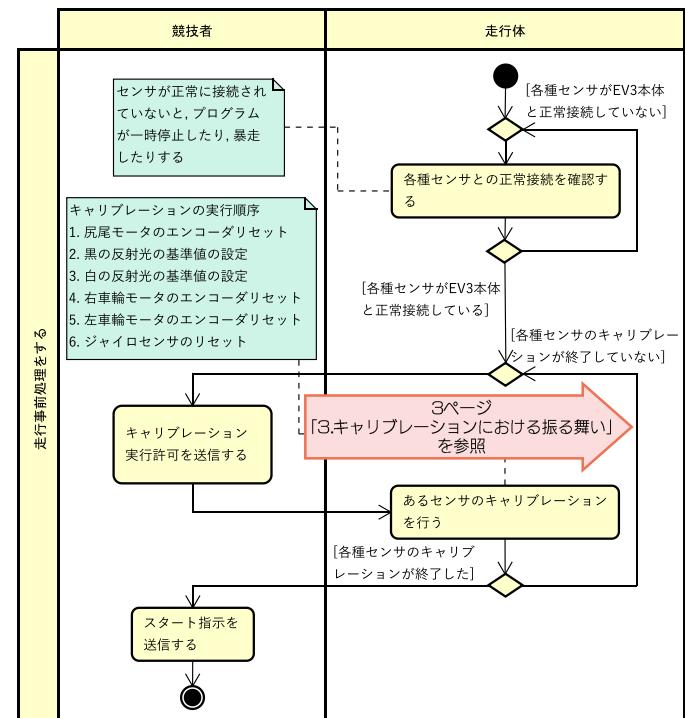
1の結果、「コースを完走する」という目標を達成するために、走行体は競技者に「コースを安定走行する」機能を与える。また、万が一を想定し「失格を回避する」機能も与える。それらを実現するための細かな機能を以下のユースケース図にまとめる。特に「コースを安定走行する」機能について、求められる振る舞いを右のユースケース記述に表す。



項目	内容
ユースケース名	コースを安定走行する
目的	100%の精度でコースを完走すること
アクター	競技者
トリガー	競技者がプログラムを起動すること
正常終了条件	走行体がゴールゲートを通過すること
基本フロー	<ol style="list-style-type: none"> <li>走行体(EV3)が自身と各種センサが正常に接続されているか確認する</li> <li>競技者からのリモート指示を受け、走行体が各種センサをキャリブレーションする</li> <li>競技者がスタート指示を送る</li> <li>走行体が走法のセット(切り替え)を行う</li> <li>走行体が走行する           <ol style="list-style-type: none"> <li>-1. 走行体がPID制御で操作量を算出する</li> <li>-2. 走行体が5-1で算出した操作量を基に倒立走行する</li> <li>-3. 走行体が自己位置推定を行う</li> <li>-4. 走行体が5の動作を繰り返す</li> <li>-5. 走行体が4～6の動作を繰り返す</li> </ol> </li> </ol>

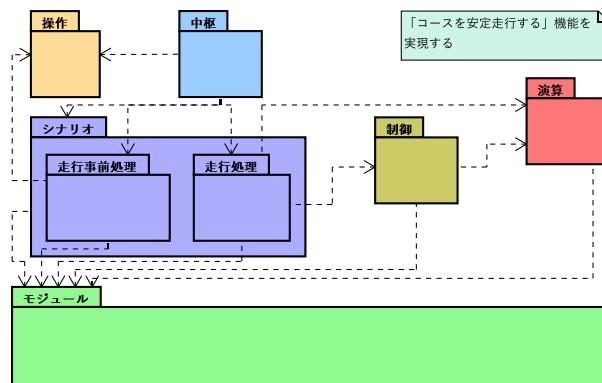
## 3. 基本動作分析

2において、走行体が競技者へ提供する「コースを安定走行する」機能をユースケース記述で文章として記述している。ここでは、「コースを安定走行する」機能における振る舞いの流れをアクティビティ図で表す。



## 1. パッケージ構造

1ページの「1.機能定義」の結果を基に、クラスの設計を行う。クラスの設計において、設計されたクラスは複数のグループに分類することができる。そこで、グループ同士の関連付けを以下のパッケージ図に示す。また、右に各パッケージの役割を示す。



パッケージ名	役割
中枢	プログラム全体の管理を行う
シナリオ	走行事前処理 走行処理
操作	走行処理
制御	走行処理
演算	走行処理
モジュール	走行処理

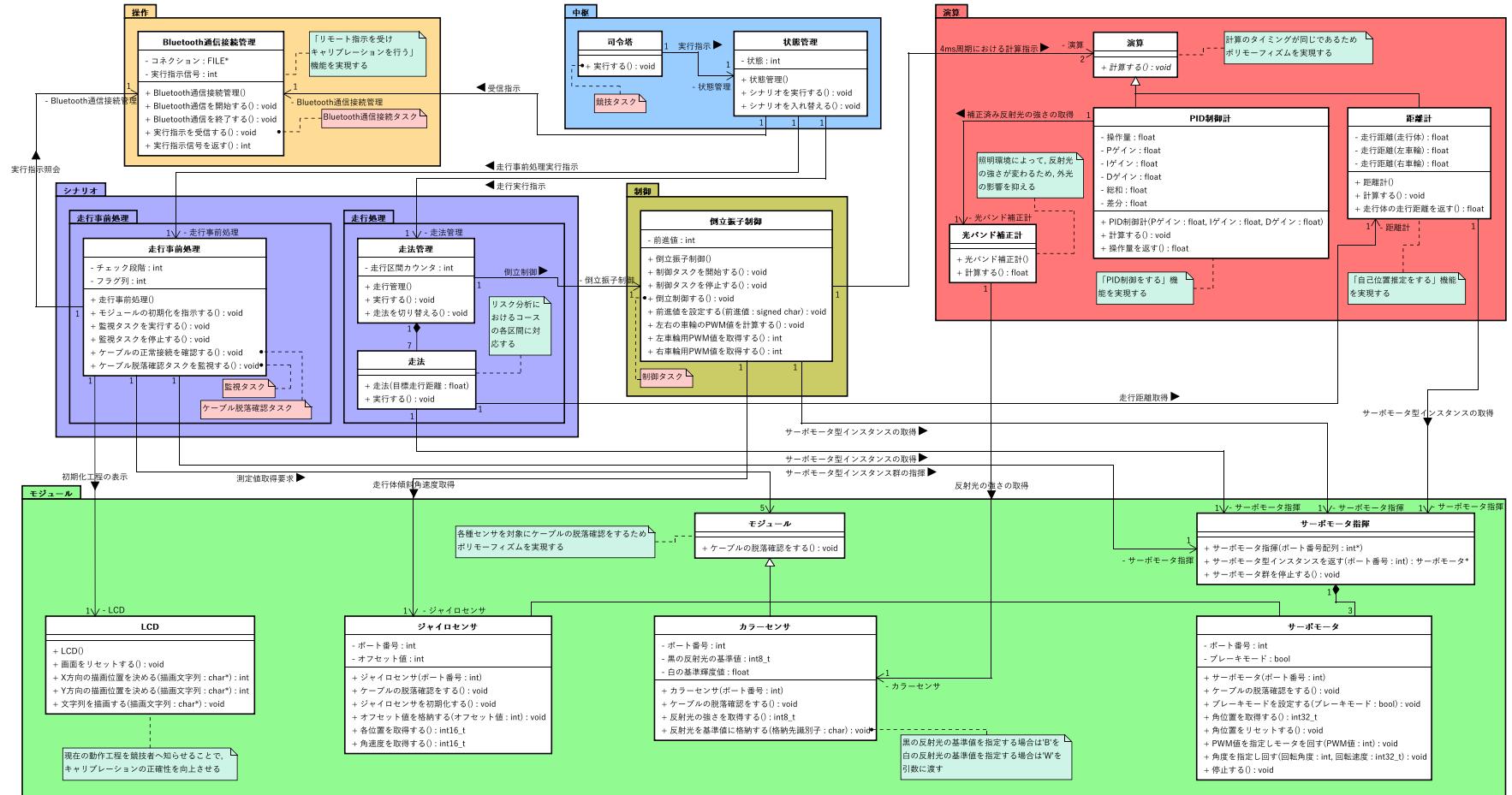
## 2. タスク一覧

EV3RT上で動作するプログラムは、タスクを活用することができる。走行体が提供する機能を実現するため、以下のタスクを用いる。

タスク名	動作内容
競技タスク	エントリーポイントとなるタスク
Bluetooth通信接続タスク	PCとBluetooth通信で接続を行い、隨時PCから送られてくるデータを受信するタスク
ケーブル脱落確認タスク	各種センサへ適当な測定値取得要求を送り、測定値の返答からケーブルの接続状態を確認するタスク
監視タスク [周期タスク]	ケーブル脱落確認タスクの実行状態を監視するタスク
制御タスク [周期タスク]	倒立制御を行うタスク

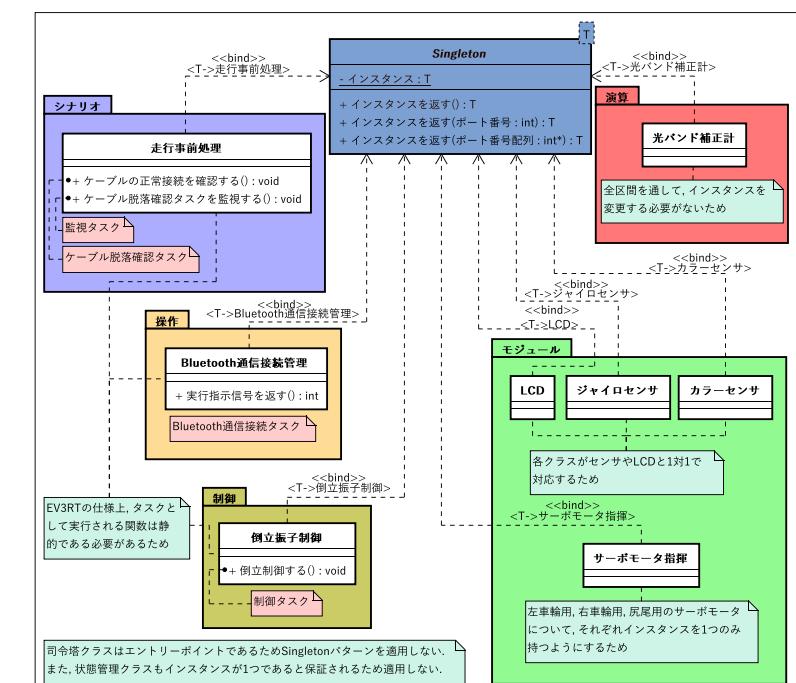
## 3. クラス構造

安定した走行を行いゴールゲートを通過するためのクラスの構成を以下のクラス図に示す。また、後述する振る舞いモデルにおいてインスタンスを円滑に管理するため、Singleton/パターンを適用している。



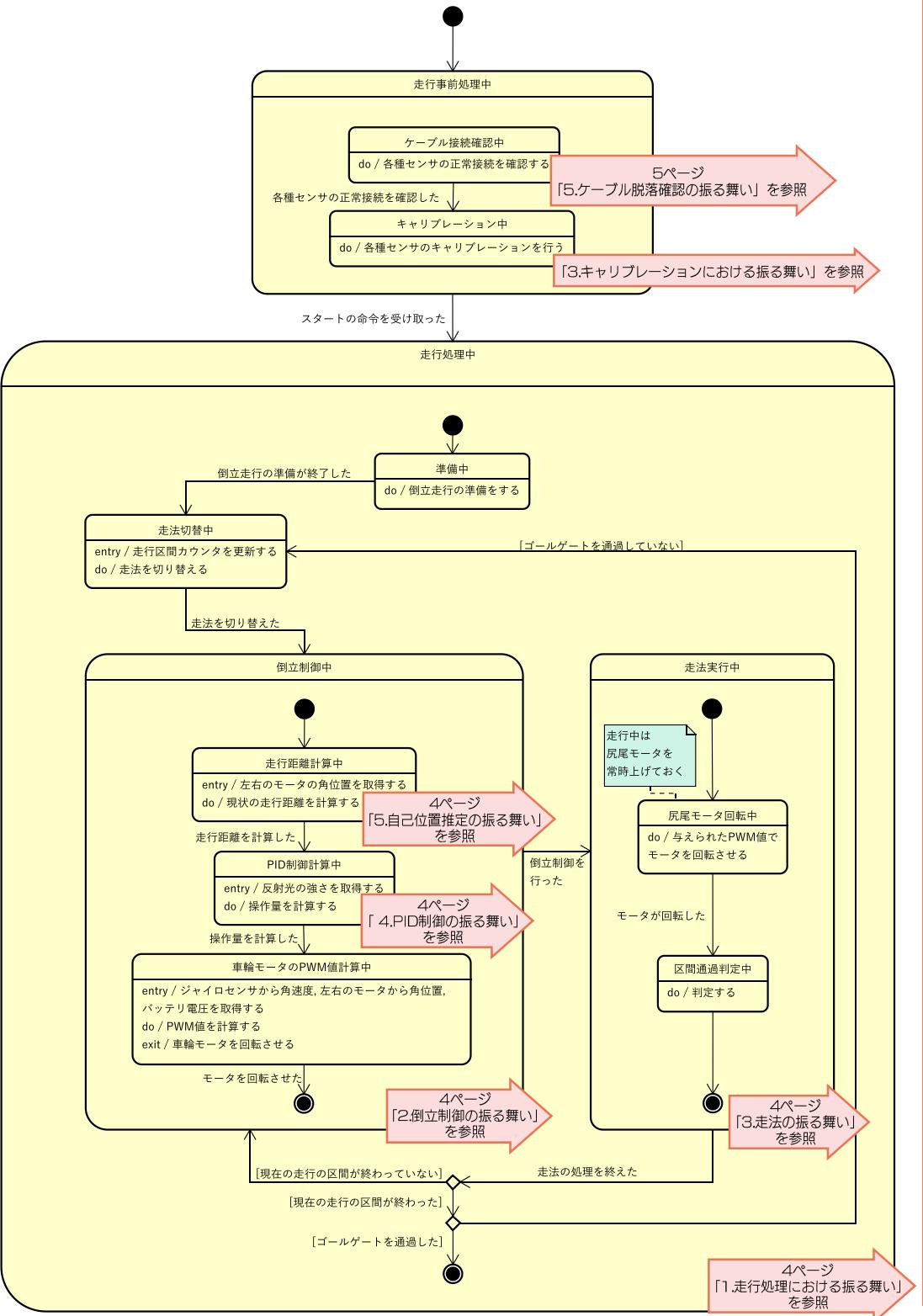
### 3-1. Singletonパターン

設計したクラスのうち、プログラムの開始からゴールゲートの通過までインスタンスの状態を保持させておきたいクラスが存在する。そこで、それらのクラスに「Singletonクラス」を実装し、インスタンスが必ず1つであることを保障させる。以下にクラス図を示し、「Singletonクラス」を実装する事由を添える。



## 1. 状態の遷移

1ページの「3.基本動作分析」で示したアクティビティ図を基に、走行事前処理と走行処理の各工程における状態の変化を以下のステートマシン図に示す。

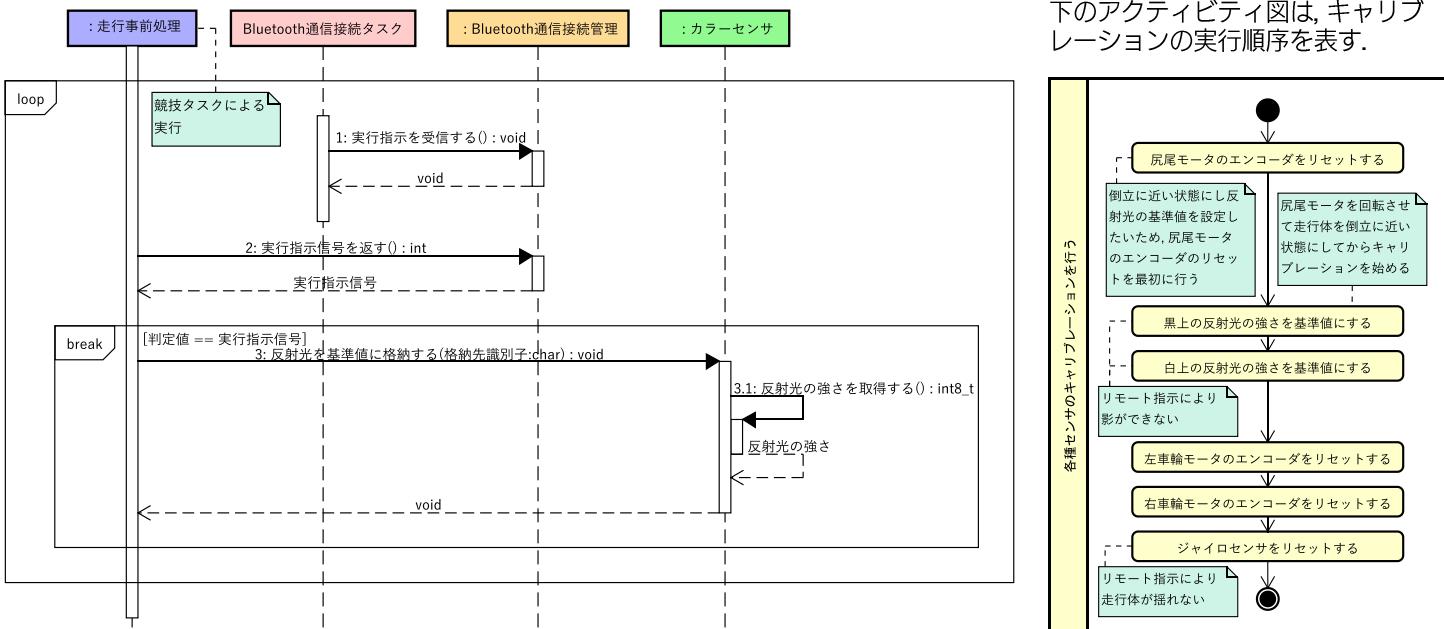


## 2. 状態から振る舞いへ

1のステートマシン図上で表現された状態の中で、機能定義で示した「コースを安定走行する」機能の実現に必要となる振る舞いを説明する。また、「リモート指示を受けキャリブレーションを行う」機能、「自己位置推定をする」機能、「PID制御をする」機能にも焦点を置く。

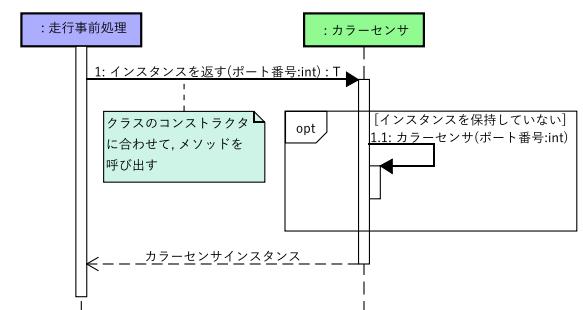
## 3. キャリブレーションにおける振る舞い -「リモート指示を受けキャリブレーションを行う」機能の実現-

キャリブレーションは、コースの安定走行を実現するのに正確性が問われる要素の1つである。1ページの「2.機能定義」で示したユースケース図にあるとおり、タッチセンサを用いてキャリブレーションの工程を進めると、人が走行体に触れる影響が発生してしまう。これは、キャリブレーションの正確性が欠ける要因となり、コースの安定走行の実現を果たすことができなくなる。そこで、以下のシーケンス図に示すとおり、Bluetooth通信を介して送られてきた信号を受信することでキャリブレーションを進めていく。



## 4. Singletonにおける振る舞い

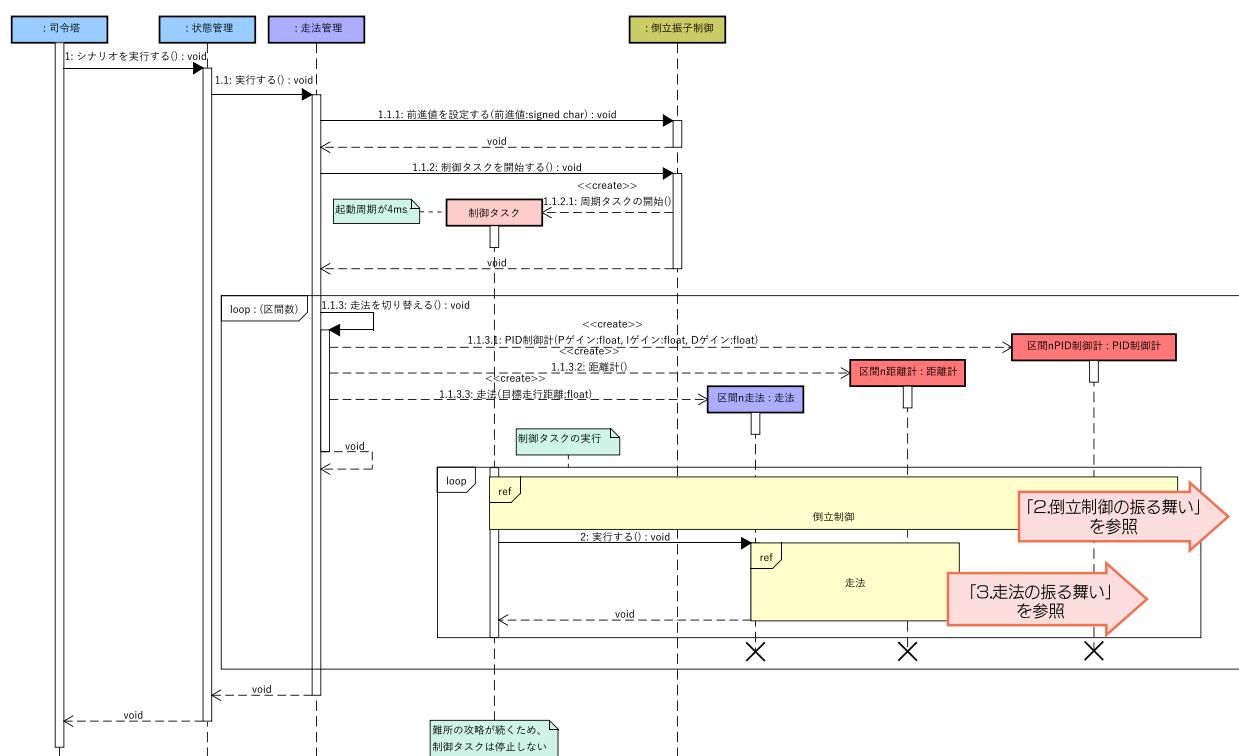
3のシーケンス図では、「カラーセンサ」クラスを用いている。「カラーセンサ」クラスは、2ページの「3-1.Singletonパターン」で示したクラス図により、「Singletonクラス」を実装することから、インスタンスは1つであることが保障される。そのため、インスタンス名を省略する。また、振る舞いについて、「Singletonクラス」を実装するクラスは「インスタンスを返す」メソッドを随时呼び出さなければならない。左下に示すシーケンス図は、上記シーケンス図における「カラーセンサ」インスタンスの「反射光を基準値に格納する」メソッドを呼ぶ際に記述すべきメッセージと複合フラグメントである。省略した理由は、左下の記述によりシーケンス図が複雑になり、可読性の低下につながると考えたためである。同様の理由で、次頁に示す走行処理における振る舞いにおいても以下の記述を省略する。



クラス	呼び出しが必要なメソッド
ジャイロセンサ	インスタンスを返す(ポート番号:int);T
カラーセンサ	インスタンスを保持していない
サーボモータ指揮	インスタンスを返す(ポート番号配列:int*);T
その他	インスタンスを返す();T

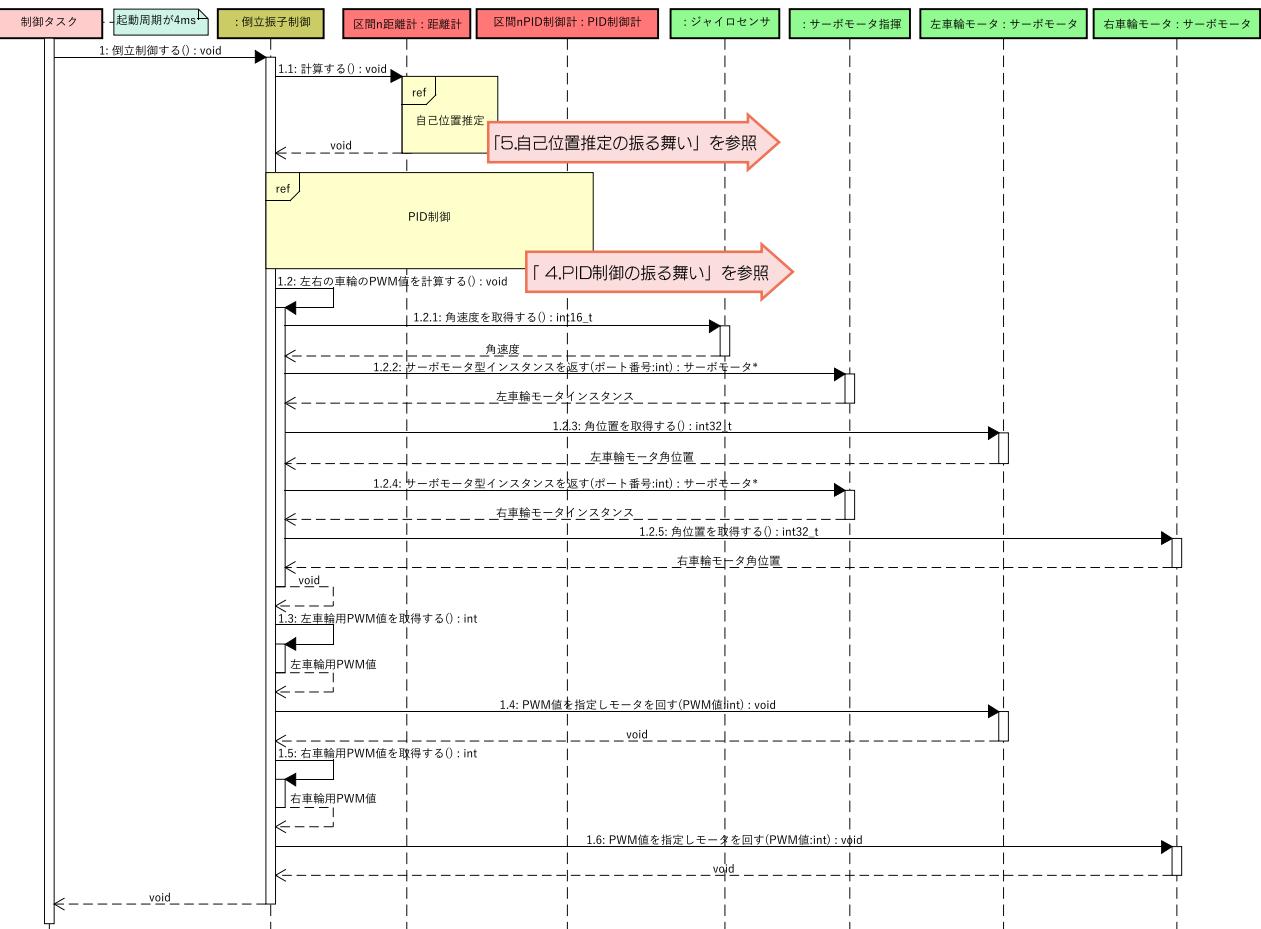
## 1.走行処理における振る舞い

スタート指示を受けた走行体がゴールゲートを通過するには、1ページの「1.リスク分析」で区間分けしたとおり、LコースとRコースを問わず複数の区間を完走する必要がある。そのため、区間ごとのコースの特徴を基に安定走行の実現を図る。コースの特徴は「シナリオ」パッケージの「区間n走法」(目標走行距離)と「演算」パッケージの「PID制御計」(比例係数、微分係数、積分係数)が保持する。走行の開始からゴールゲート通過までの振る舞いを以下にシーケンス図に示す。ここでは、すでに「状態管理」のシナリオが走行処理を指しているとする。



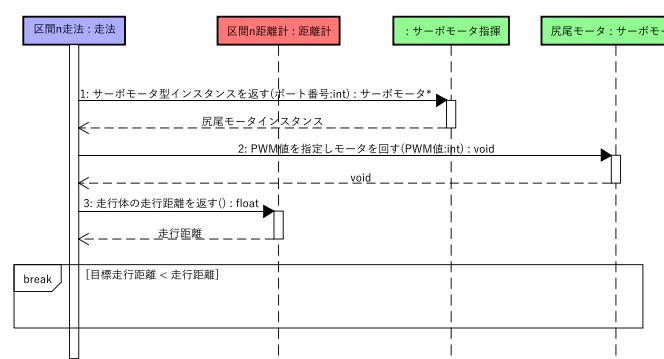
## 2.倒立制御の振る舞い

倒立制御は、4ms周期で実行しなければならない制約があるため、制御タスクを4ms周期で実行し走行体の倒立を実現する。それに伴い、走行距離(「距離計」と操作量(「PID制御計」)の更新も行う。以下に倒立制御のシーケンス図を示す。



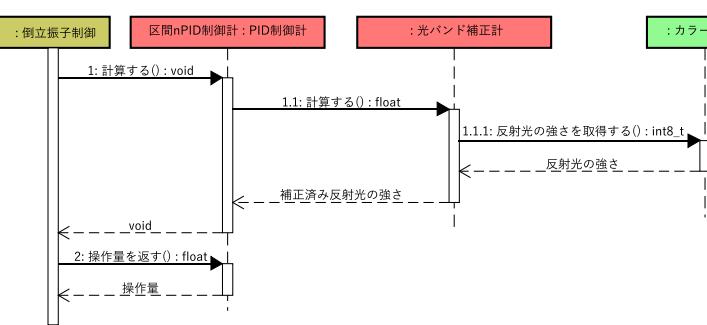
## 3.走法の振る舞い

走法は、倒立制御の4ms周期の実行に生じる隙間時間を利用し処理される。ここでは、主に現在走行中の区間の終了判定を行っている。以下に走法のシーケンス図を示す。



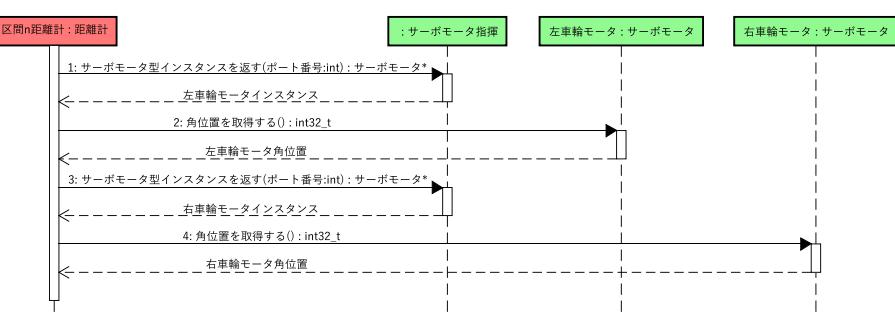
## 4.PID制御の振る舞い -「PID制御をする」機能の実現-

「PID制御計」は、コンストラクタの呼び出しで渡される比例係数、微分係数、積分係数と、「光バンド補正計」から得られる補正済み反射光の強さを用いて操作量を計算する。以下にPID制御のシーケンス図を示す。



## 5.自己位置推定の振る舞い -「自己位置推定をする」機能の実現-

自己位置推定は、左右の車輪モータから角位置を取得し変化量を調べることで、走行距離の推定を可能にする。ここでは、走行距離の計算のみを実行する。実際に計算値を取得する処理は、走法で行われる。以下に自己位置推定のシーケンス図を示す。



## 1.工夫点

走行体(EV3)と各種センサをつなぐケーブルの脱落検知を自動で行う。

## 2.背景

昨年、前大会に向けてプログラムを開発し走行体を試走させているときに、ケーブルの脱落に気づかず走行体が暴走する場面を何度も目にした。さらに、脱落により起こる不具合の原因がソフトウェア側にあると思い込むことが多々あった。そこで、本大会ではケーブルの脱落検知を自動で行い、走行体の暴走を事前に防ぎ、コースの完走精度を向上させる。

## 3.期待できる成果

ソフトウェア側でケーブルの脱落検知を行うことで、走行体を走行させる前に暴走の要因を取り除くことができる。また、プログラムの不具合の特定が容易になる。加えて、競技者はケーブル脱落を警戒する必要がなくなる。

## 4.実現方法

### 4-1.EV3RTの仕様

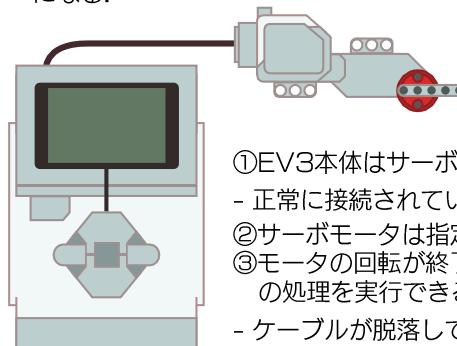
EV3と接続された各種センサへ測定値の取得を要求した際に、センサがEV3と正しく接続されていない場合、プログラムはセンサから値が返されるまで次の処理を実行することができない。

### 4-2.手順

ケーブル脱落確認タスクと監視タスクを用いて実現する。

ケーブル脱落確認タスクは各種センサへ測定値の取得要求を行う。これにより、センサから測定値を取得できた場合は、チェックフラグ列の更新を行う。

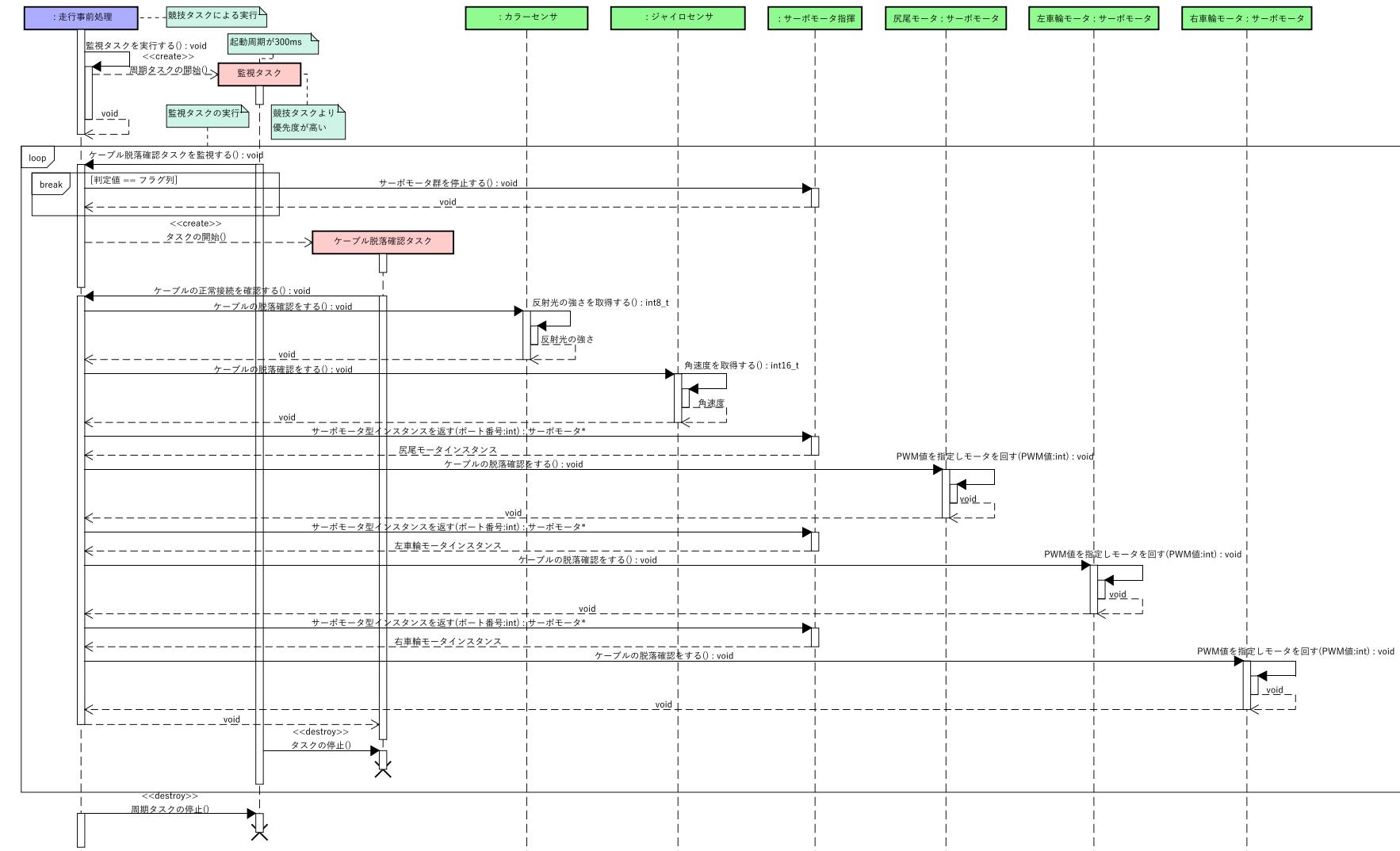
周期タスクである監視タスクが起動すると、チェックフラグ列を確認し、判定値と等しくなれば、ケーブル脱落確認処理を終了する。等しくない場合、正常に接続されていないセンサが存在することになる。



- ①EV3本体はサーボモータへ回転命令を送る。
  - 正常に接続されていた場合 -
  - ②サーボモータは指定されたPWM値で回転する。
  - ③モータの回転が終了すると、プログラムは次の処理を実行できる。
    - ケーブルが脱落している場合 -
    - ②モータへ回転命令を送れないため、モータは回転しない。
    - ③モータが回転しないため、プログラムは次の処理を実行できない。

## 5.ケーブル脱落確認の振る舞い

ケーブルの脱落確認における振る舞いを以下のシーケンス図に示す。



## 6.導入前と導入後の状況比較

以上の振る舞いを行うように実装し、昨年度の動作状況と比較した。下図に、「操作性」、「信頼性」、「安全性」についての比較結果を示す。

	操作性面	信頼性面	安全性面
導入前	 キャリブレーションのみ。	走行処理が正常に動作せず走行体が転倒  ケーブルの脱落により、次回のプログラム実行時に走行体が暴走する可能性が増す。	不具合発生  ソフトが原因?  ハードが原因? 原因の断定ができない。
導入後	 Check OK!  ケーブル脱落検知後にキャリブレーション。	走行処理が正常に動作せず走行体が転倒  プログラム実行時にケーブル脱落検知を行うため、走行体が暴走する可能性に変化なし。	不具合発生  ソフトが原因! 原因が全てソフトウェア側だと断定できるようになった。