

### チーム紹介、目標、意気込み

私たちうぬぼれラビットは同じクラスの志の高いメンバーが揃ってできたチームです。我々は全員が全員なかなか都合が合わず、全員が揃って作業すると言うことが少ない中で設計を行ってきました。しかし、一人一人がお互いを助け合い、自宅でも連携を取り合ってなんとか完成にこぎ着けることができました  
[北海道地区大会優勝を目指し、全国大会でも優勝できる事を目標に、当日はマシンとの友情を深め、マシン自身のモチベーション向上が臨めるように当日も応援を怠らずに参戦したいと思います。](#)

リーダー：栗原佑太  
メンバー：高谷政輝  
中川滉介  
畠山悠介

### モデルの概要

- シーソーダブル攻略を目指し、要求されている要件の分析から走行方法の解説、シナリオの作成を行うことでそれを基にクラス図を作成した。さらにリスク管理を行い、シーソーを走行している際に起こりうるリスクの想定と対策を考えることができ、安定した走行が臨める事がわかる。
- 走行体の振る舞いは構造モデルのクラス図を基にシーケンス図を作成し、また、計算量の部分や尻尾の動きはステートマシン図、アクティビティ図で作成し、それぞれの説明を横に添付してあるのでどのような図なのかを具体的に知ることができる。
- 工夫点では走行させてみた時の課題点に自分たち独自で工夫した点をまとめた

日本工学院北海道専門学校

### モデルの構成

#### 1.機能モデル

- 要求されている条件を明らかにし、それを図に表した
- 要求されている条件を明らかにし、分析することで走行方法を定めることができた。
- 要求されている条件からそれによって起こりうるリスクの判定、分析、対応と走行方法などが記載されたシナリオを作成した。

#### 2.構造モデル

- シーソー攻略の際のプログラムを走行制御、区間制御、進行状態管理、判定、デバイスの5つのパッケージに分けたクラス図を作成した。

#### 3-4.振る舞いモデル

シーソー攻略のための振る舞いの図とその説明を記載した

- シーソーを攻略するため構造モデルのクラス図をもとに、シーソーダブル、競技タスク、シーソー進入、シーソーの通過、傾き判定、衝突判定、ライントレース走行、距離判定、直線制御走法、のシーケンス図を作成した。
- プログラムの計算量が多い区間の全体の処理を分割したタスクの構成と、シーソー全体、ライン検知ステートマシン図を作成した。
- シーソーの攻略の際の尻尾全体の動きと制御のアクティビティ図を作成した。

#### 5.工夫点

うぬぼれラビットが問題点に施した工夫を記載した

- ローパスフィルタによるノイズ除去、輝度幅を調整する光センサ値の正規化、ラインの無い部分でのモーター制御をする直線処理、色の識別方法を書いたRGBライントレースの4つの工夫点を作成した。

## 1. 機能比モデル



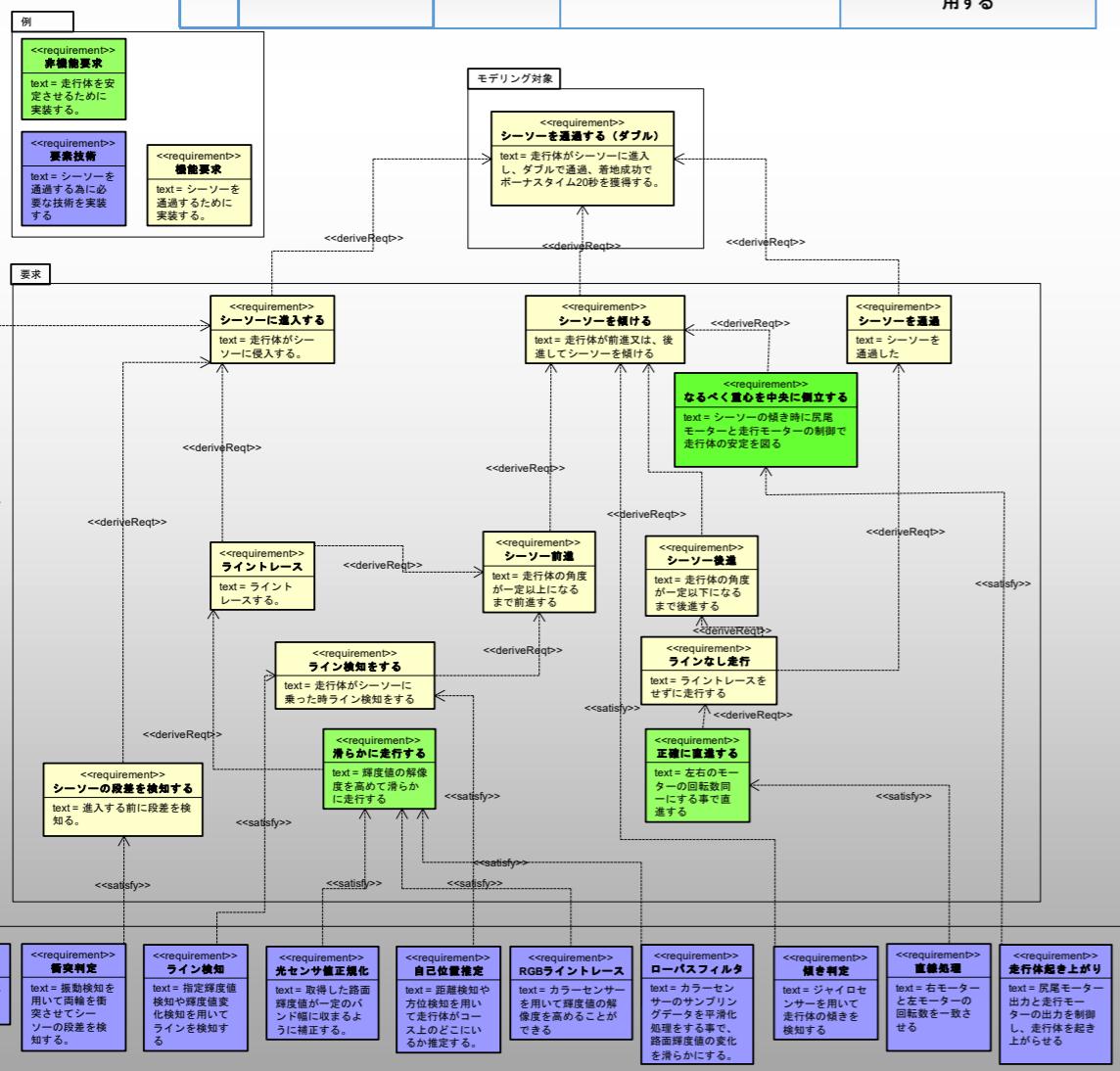
課題

モデリング対象:シーソーを通過する(ダブル)

## 要求分析

**走行方法の説明**

走行方法についての要求分析も行ったが、走行方法はシーソーを通過するまでの全ての動作で行う為、走行要素技術に多くの充足関係の線が繋がってしまい、可視性を保つ事が難しくなる。そのため、要求図には記載せず走行方法の要素技術については、概要と利用状況を含めて右の表に示す事にした



走行方法		ライントレースの有無	概要	利用状況
倒立走行	PID走法	○	PIDを用いて最適な旋回量を算出し、滑らかなライントレースをする	ライン上を滑らかに走行したい時に利用する
	直線制御走法	×	両輪のモーターの回転角度を同一にする事で、直進走行する	ライントレースをせずに直進したい時に利用する
尻尾走行	前進走法	○	ジャイロセンサーを用いて走行体の角度を取得する事により尻尾の角度を制御する	ライントレースをしながらシーソーを前進したい時
	後進走法	×		シーソー上で後進した時、ライントレースができない状況で利用する

シナリオ

シーソーをダブルでを通過する為の走行体が行う動作の流れを以下の表に示す

大区間		シーソー進入				シーソーダブル		
中区間		シーソーの段差を検知する	真っ直ぐにバックする	真っ直ぐに直進する	シーソーに乗り尻尾を下ろす	シーソーのラインを検知する	シーソー上で起き上がる	シーソーが傾くまで少しづつ前進する
説明図								
要素技術名		衝突判定	直線処理・自己位置推定		倒立切り替え	ライン検知	走行体起き上がり	傾き判定
走法	方法	PID走法	直線制御走法		PID走法	前進走法	走行体起き上がり	傾き判定
	姿勢	倒立走行	倒立走行					
検知方法		衝突判定	距離判定	衝突判定	距離判定	ライン検知	尻尾モータエンコード値	傾き判定
リスク判定		段差に当たった際に走行体が倒れるリスクがある	ラインからそれてしまい、走行体がコースからはずれてしまう		尻尾走行に移行する際に転倒してしまう	ラインを検知せずにシーソーから落下してしまう	尻尾を下げても起きあがることできない	シーソーが傾いた際に走行体が倒れてしまう
リスク分析		走行体のスピードが原因で転倒してしまう可能性がある	ライントレースを行わないため、モーターの個体差によって出力に差が出る可能性がある		走行体の体勢が前傾の場合前方に転倒してしまう可能性がある	PID制御が強すぎた場合、シーソーから落下してしまう	後ろに重心があるため、尻尾モーターだけのパワーでは起き上がるための力が不足してしまう	シーソーの傾きの検知が遅ることで走行体が倒れてしまう可能性がある
リスク対応		適切なスピードを実験によって調整する	モーターの回転数が同じになるように制御を行う		後ろに走行体を傾けることで転倒を防ぐ	適切なPID値により検知に猶予を与える	起き上がる際にタイヤを回し、走行体を前傾姿勢にする	シーソーの傾きの角度を正確に検知する

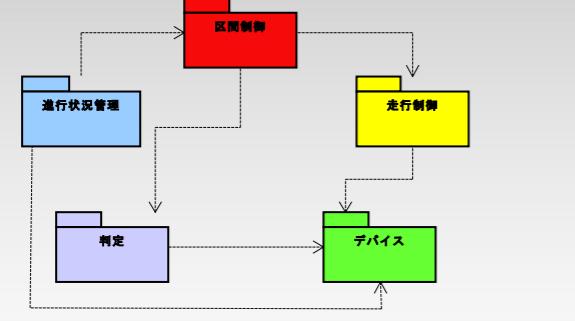
大区間	シーソーダブル						シーソーの通過
中区間	シーソー上で着地させる	シーソー上で起き上がる	シーソーが再び傾くまで後退する	走行体を落ちかせる	シーソー上で起き上がる	少しづつ前進する	シーソーから降りる
説明図							
要素技術名		走行体起き上がり	直進処理		走行体起き上がり	ライン検知	ライン検知・自己位置推定
走法	方法		後進走法			前進走法	前進走法
	姿勢	尻尾走行	尻尾走行	尻尾走行	尻尾走行	尻尾走行	尻尾走行
検知方法	時間	尻尾モーターエンコード値	傾き判定	時間	尻尾モーターエンコード値	傾き判定	距離判定
リスク判定	寝かせる力によって走行体が前に倒れてしまう	前に起こしすぎると倒れてしまう	シーソーが傾いた際に走行体が倒れてしまう	寝かせる力によって走行体が前に倒れてしまう	尻尾を下げても起きあがることができない	シーソーが傾いた際に走行体が倒れてしまう	シーソーが上がる上がることで走行体が飛ばされる
リスク分析	尻尾の力だけで浮いてしまうため走行体が動かなくなってしまう可能性がある	重心が前すぎると、後進の際に前に倒れてしまう可能性がある	シーソーの傾きの検知が遅れることで走行体が倒れてしまう可能性がある	尻尾の力だけで浮いてしまうため走行体が動かなくなってしまう可能性がある	後ろに重心があるため、尻尾モーターだけのパワーでは起き上がるための力が不足してしまう可能性がある	シーソーの傾きの検知が遅れることで走行体が倒れてしまう可能性がある	重心が前になってしまい、さらに上がってくるシーソーが走行体にぶつかることで前に倒れてしまう可能性がある
リスク対応	尻尾の角度を調整して前傾になるのを防ぐ	後進時のスピードを抑え、さらに角度を正確に調整する	シーソーの傾きの角度を正確に検知する	尻尾の角度を調整して前傾になるのを防ぐ	起き上がる際にタイヤを回し、走行体を前傾姿勢にする	シーソーの傾きの角度を正確に検知する	少しスピードをつけることでシーソーが上がってくる前にぶつからない位置まで走行する

# 2.構造モデル

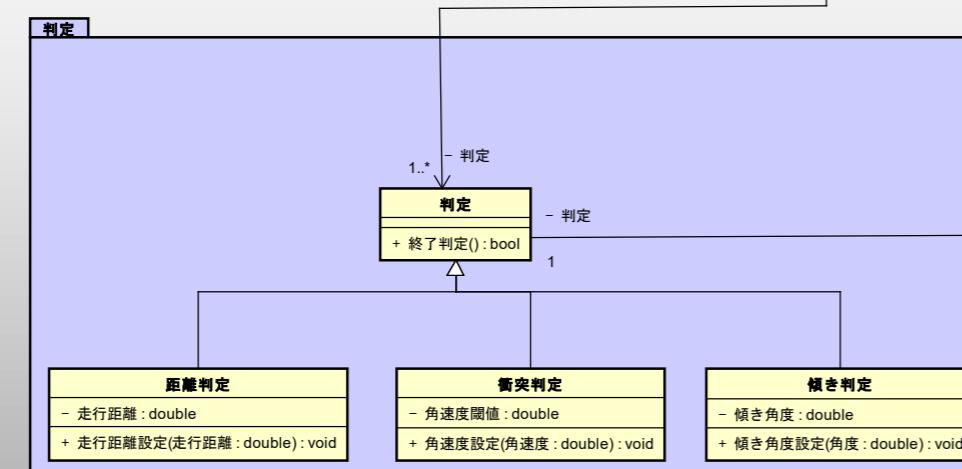
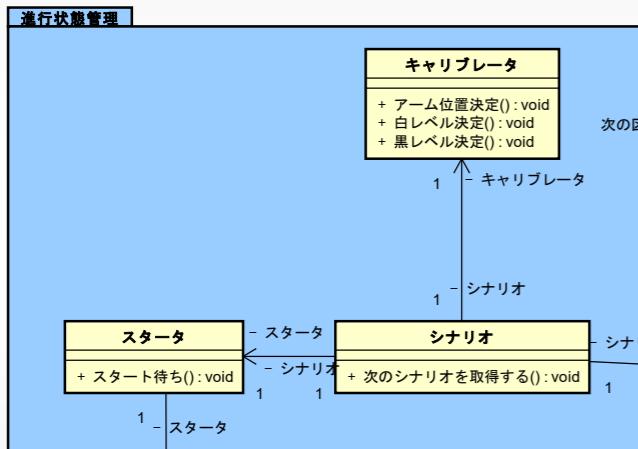


## パッケージ構造

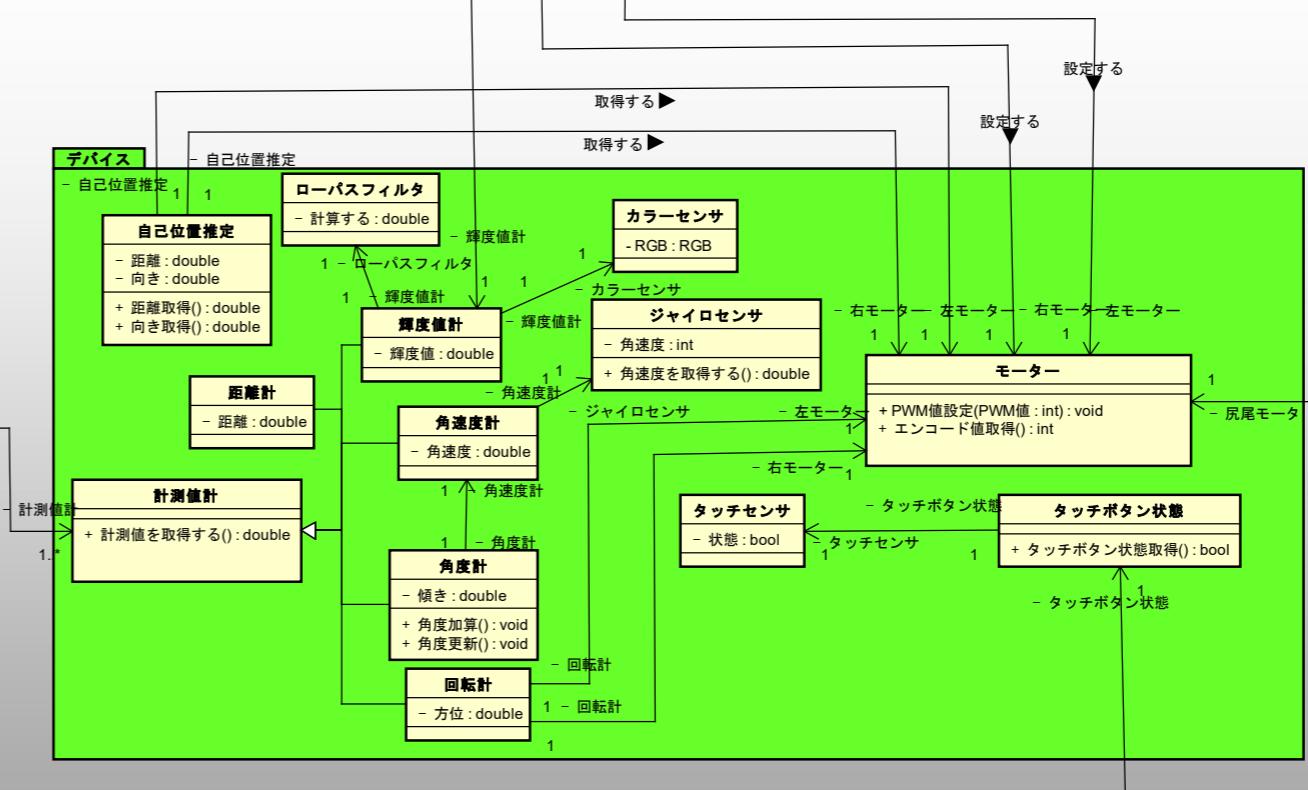
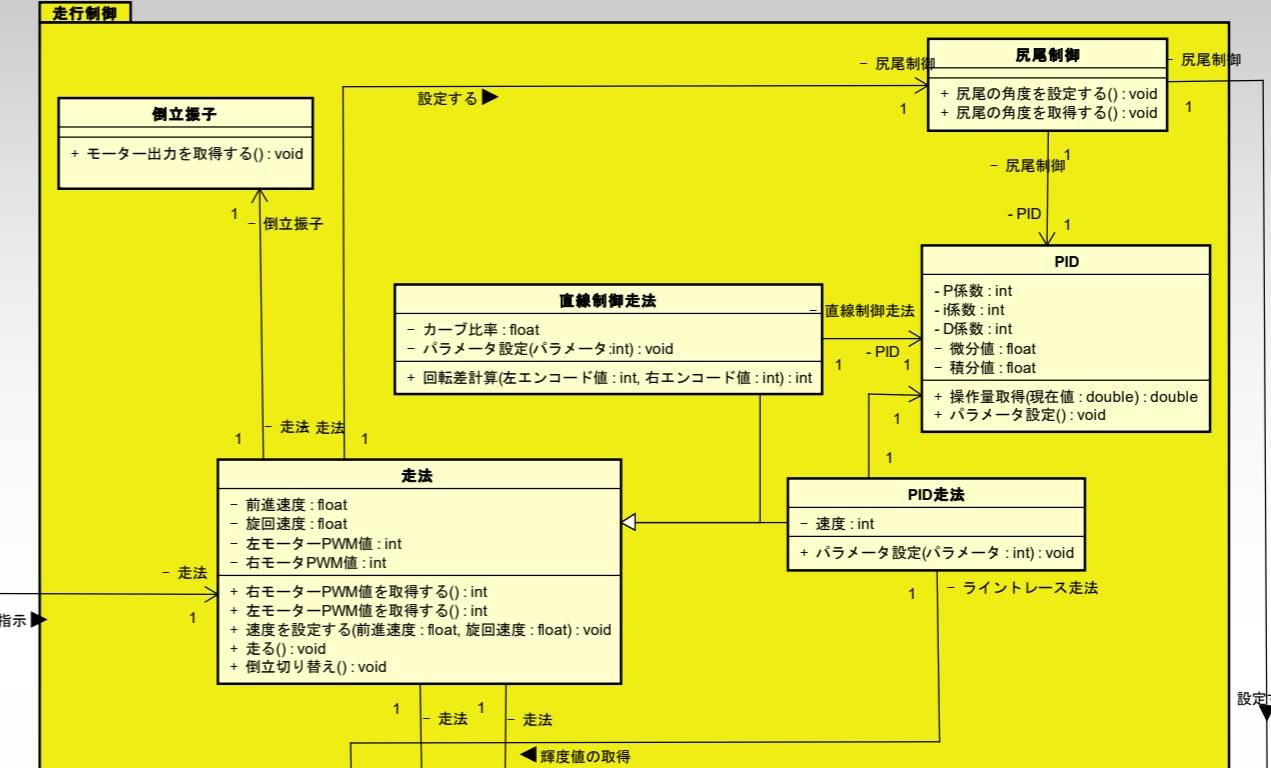
要求分析により導き出した要素技術をもとに、クラスの設計を行った。各パッケージについては右図のパッケージ概要と下記のパッケージ構造の図に示す。



パッケージ名	パッケージ概要
走行制御	各走法の制御を行いモータを駆動する
区間制御	各走法を利用した区間の進行状況の管理と検知及び制御を行う
進行状況管理	区間走行の全体管理を行う
判定	区間切り替えのための判定処理を行う
デバイス	各センサやモータ等のEV3RTのAPIを呼び出す



## クラス構造

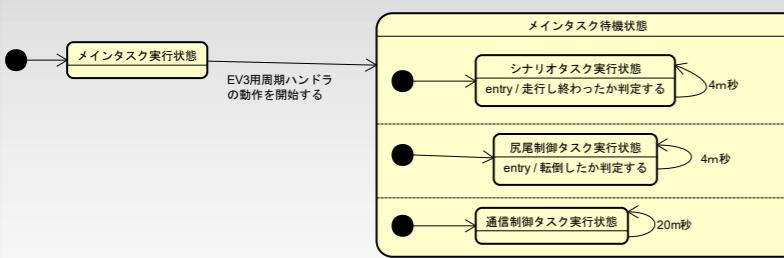


# 3.振る舞いモデル



## 3-1.タスクの分割について

プログラムを実装する上でタスクは1つでも走行可能であるが、計算量が多いが区間では全体の処理が間に合わない可能性がある。倒立制御などの確実に行いたい処理を優先して行い、タスクの分割を行った。それぞれのタスクの状態遷移について以下のステートマシン図にする。



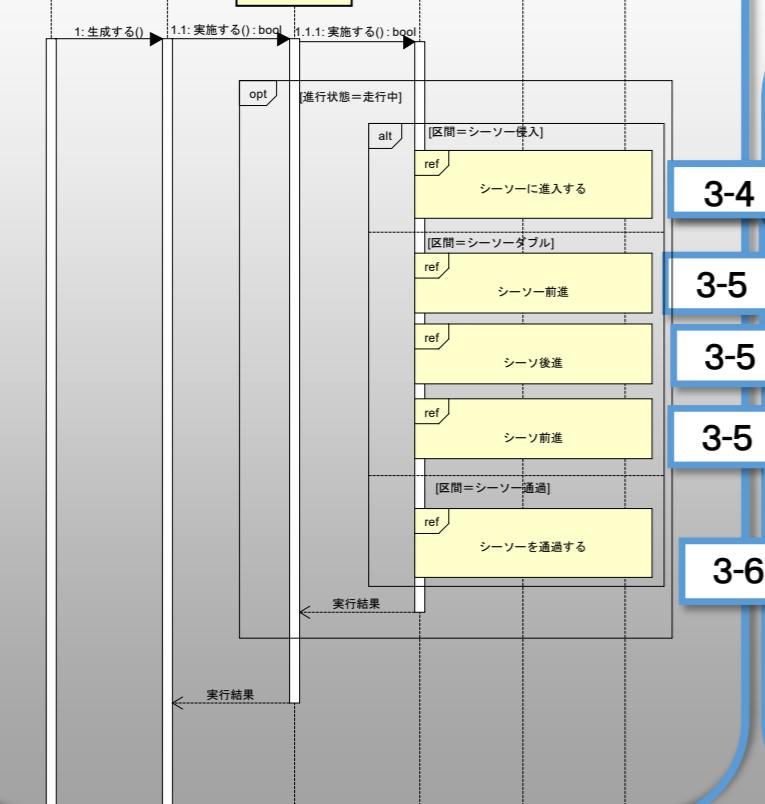
タスクの流れとしては、メインタスクが起動され、実行状態へと遷移し、メインタスクがそれぞれの周期タスクを起動する。起動後、メインタスク自体は待機状態になり、処理は何も行われない。以降はそれぞれの周期タスクが実行され続ける。

以下、それぞれの周期タスクの詳細な振る舞いについて説明する

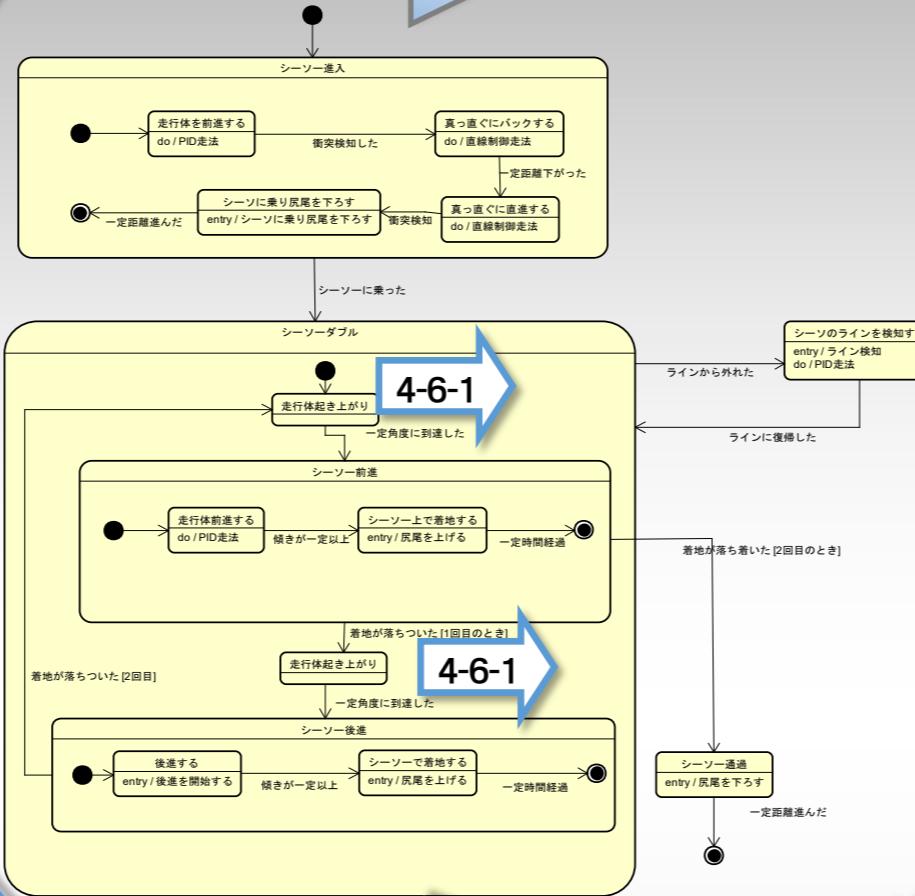
## 3-2.競技タスクの振る舞い

競技タスクでは、計測情報を取得し、走行と判定のための計算を行っている。シナリオが終了した場合、シナリオが終了した場合は、シナリオタスクを終了させるような構造となっている。尚、シナリオの詳細はステートマシン図・シーケンス図・アクティビティ図に示す。

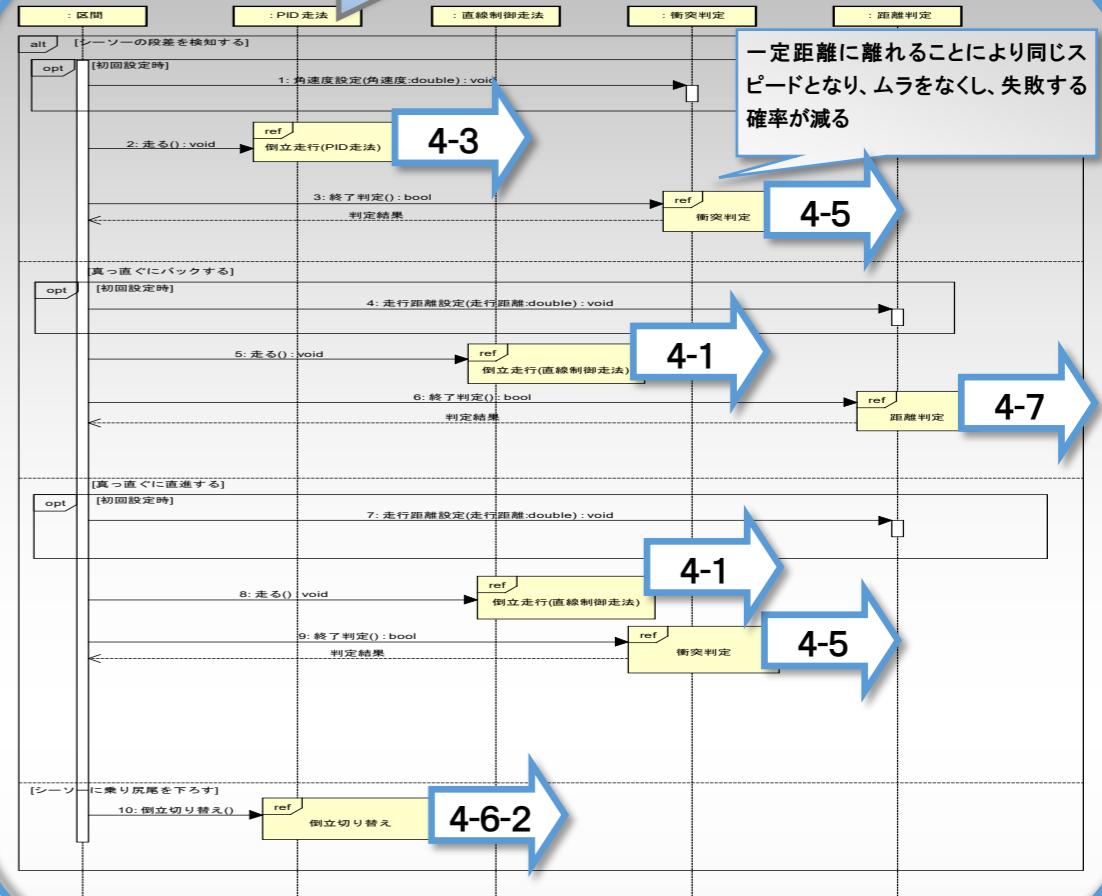
メインタスク シナリオタスク <active Class> : シナリオ : 判定 : 走行



## 3-3.シーソー全体



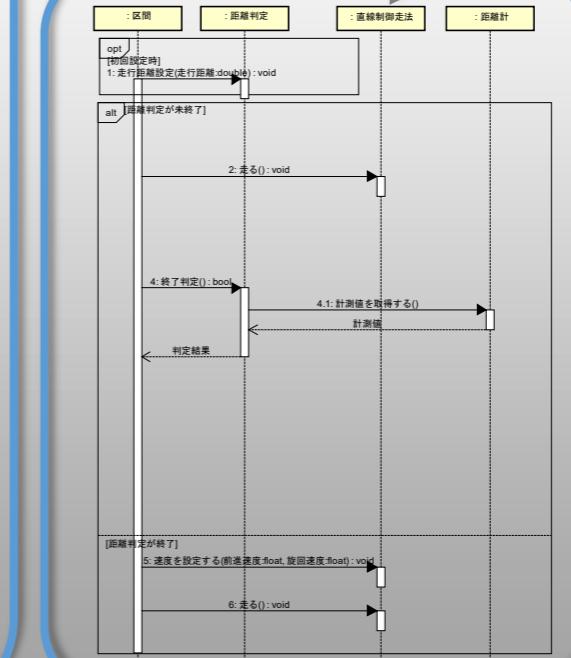
## 3-4.シーソー進入



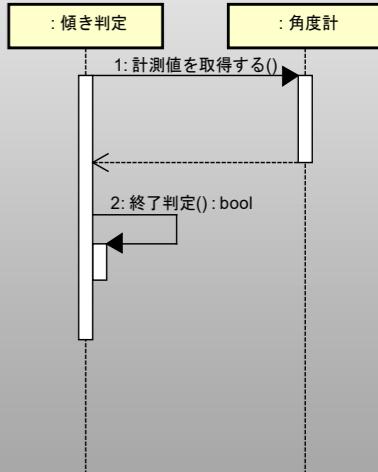
## 3-5.シーソー前進

## 3-5.シーソー後進

## 3-6.シーソーの通過



4-4の角度計算の部分で傾きは随時計算している



3-4

3-5

3-5

3-6

3-7

4-1

4-3

4-1

4-3

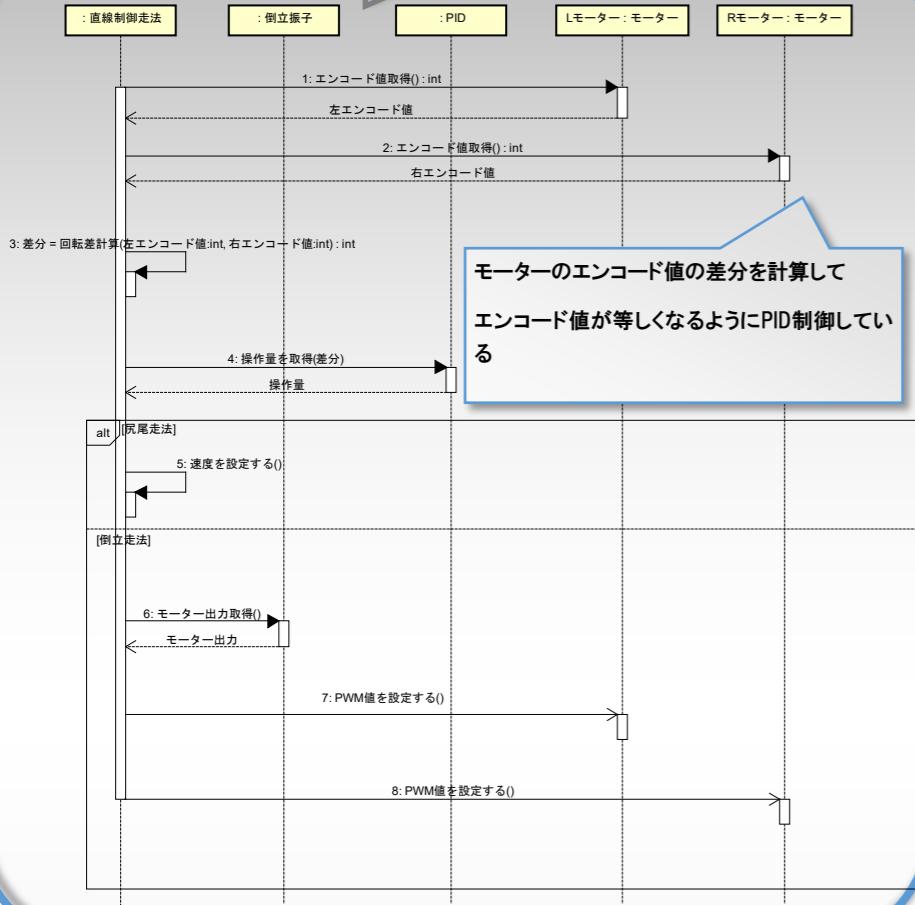
4-2

4-6-2

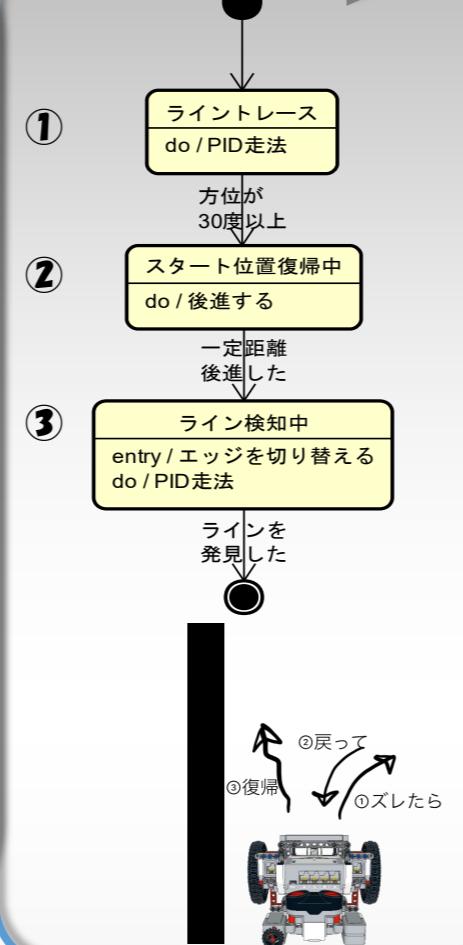
# 4.振る舞いモデル



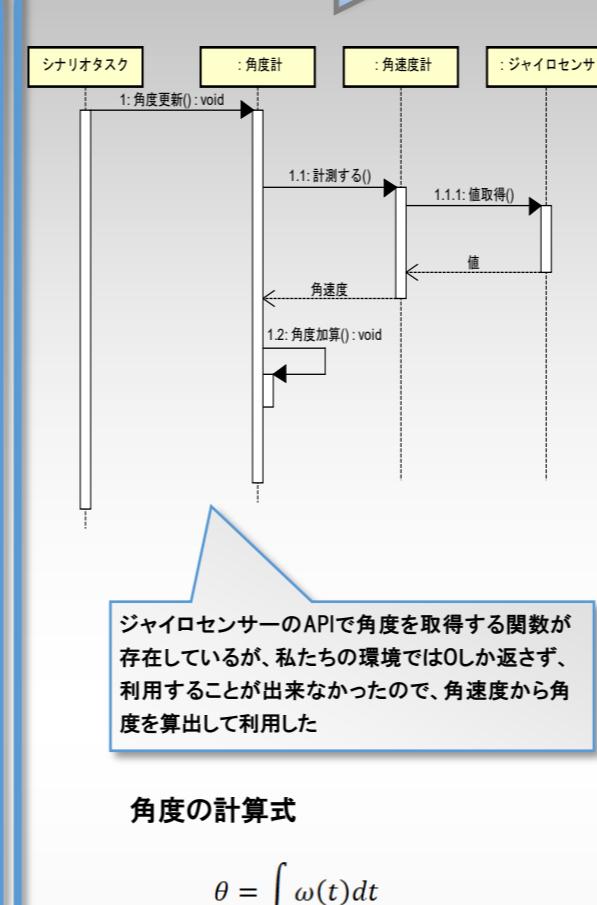
## 4-1.直線制御走法



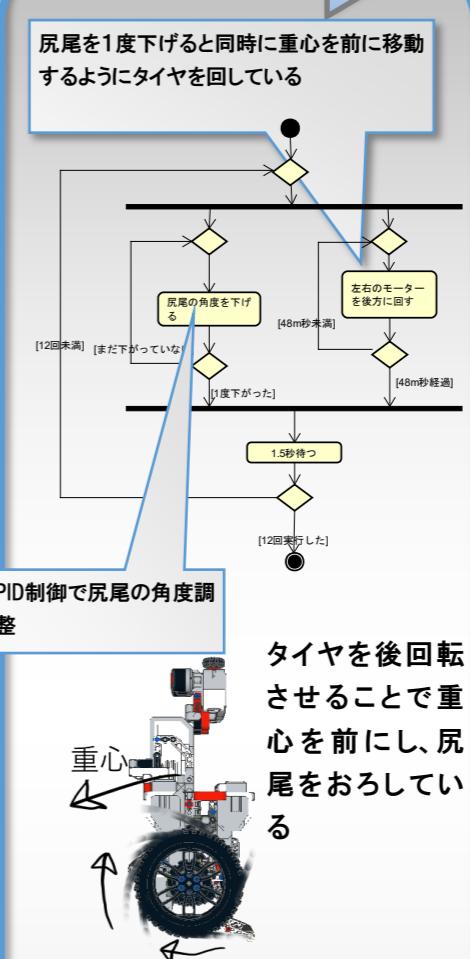
## 4-2.ライン検知



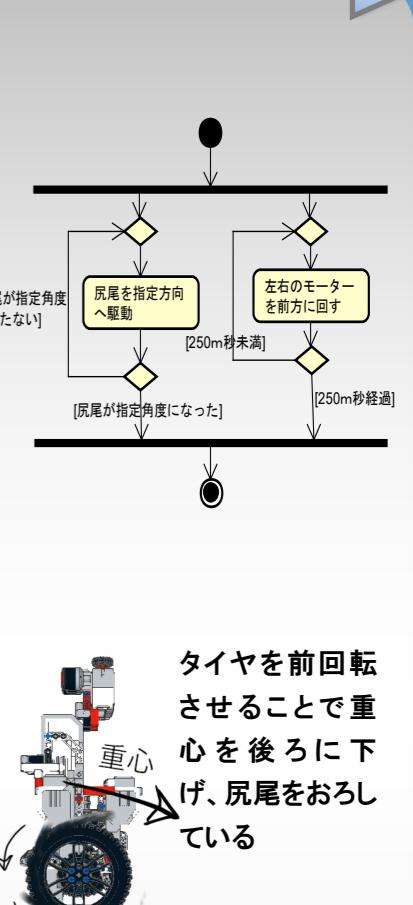
## 4-4.角度計算



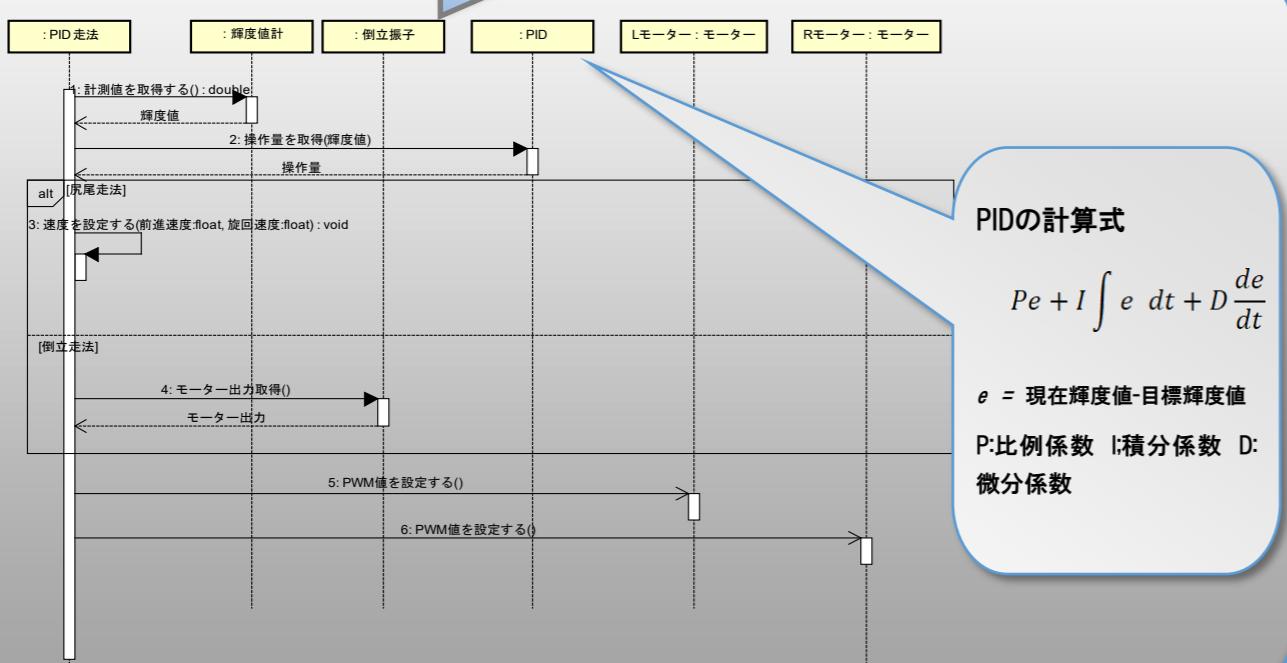
## 4-6-1.走行体起き上がり



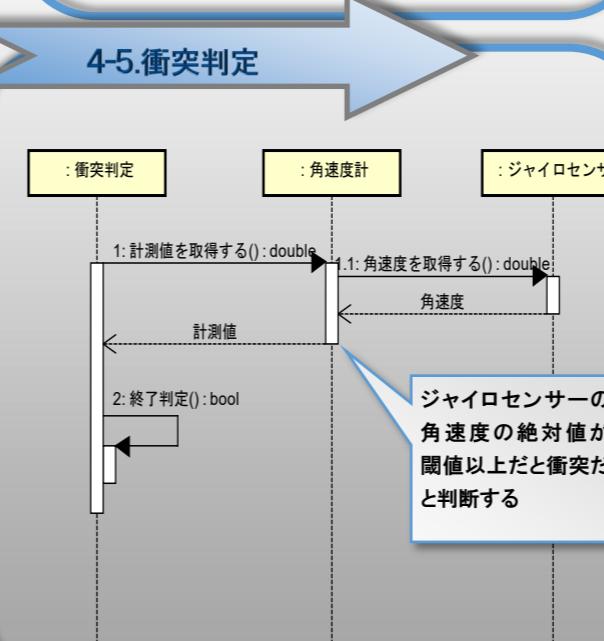
## 4-6-2.倒立切り替え



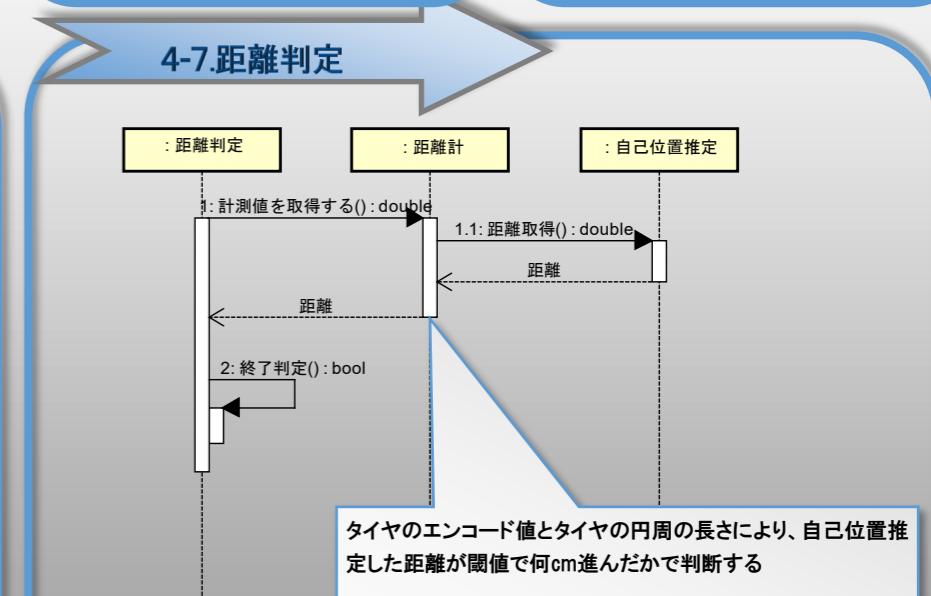
## 4-3.PID走法



## 4-5.衝突判定



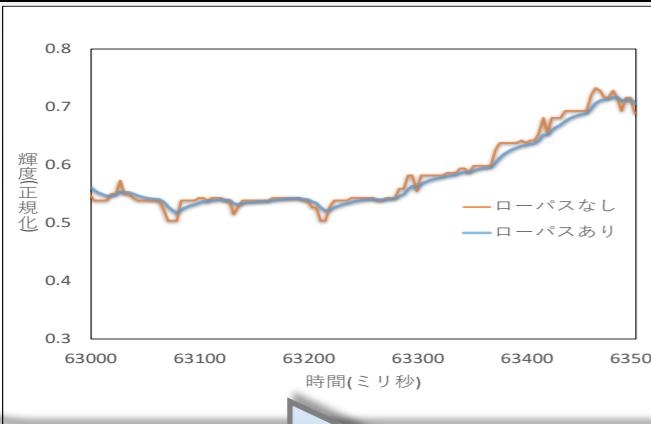
## 4-7.距離判定



# 5.工夫点

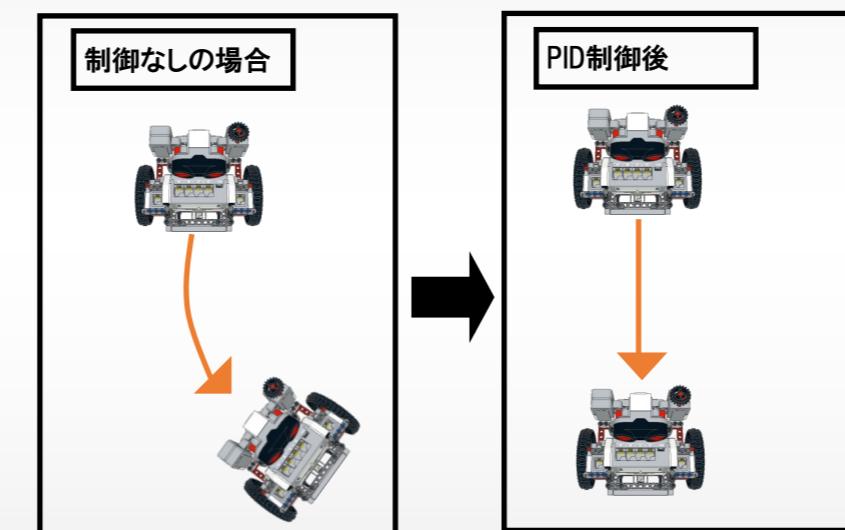
## 1.ローパスフィルタ

課題	外乱光などの環境の変化により光センサにノイズが発生し、走行体が不安定になる
対策	ローパスフィルタを用いてノイズを除去し、走行体の挙動を安定させる
結果	フィルタをかけない場合よりも光センサの変動が滑らかになり、急な走行体の挙動が抑えられた
数式	$x[n] = x[n-1] * \alpha + x[n] * (1-\alpha)$ ※ $\alpha$ には適切な値を入れる



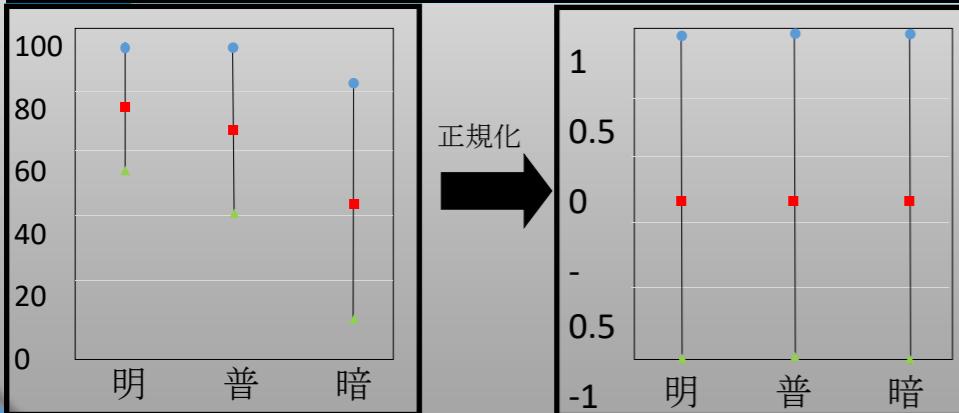
## 3.直線制御走行

課題	モーターの個体差によって同じ出力でも回転数が同じにならないため、左右どちらかの予期せぬ方向に移動してしまう
対策	左右の回転数の差が0になるようにPID制御を行い、モーター
結果	ラインがない部分でも直進することができた
数式	差分=左エンコード値-右エンコード値



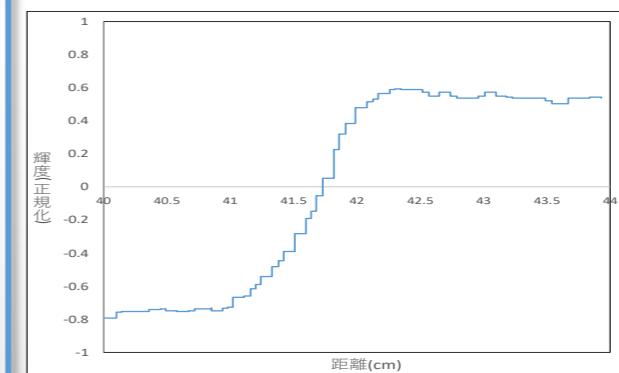
## 2.光センサ値正規化

課題	照明環境によって黒色と白色の輝度幅が異なるため、照明の明るさの加減により制御量が変動し、環境に依存した挙動になってしまう
対策	光センサ値の正規化を行うことで、本来は1~100の輝度値を-1~1の間で統一できるようにすることで環境に依存しにくい光センサ値を測定することができ、走行体の挙動が安定する
結果	意図的に外乱の輝度を変更し走らせたがその影響を受けずに輝度の判定を行い、安定した走行をすることができた



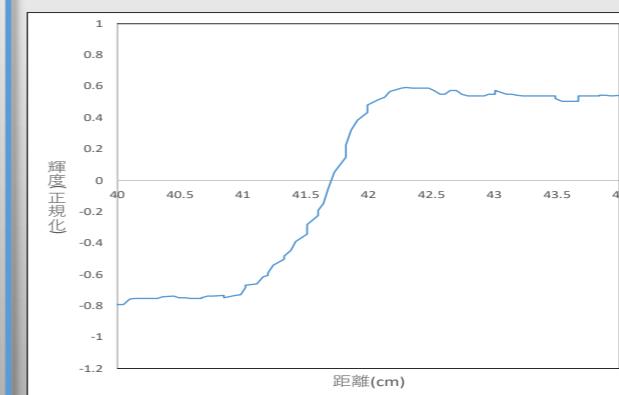
## 4.RGBライントレース

課題	正確にライントレースをせず、走行が安定しない
対策	1. 反射光で色の判定を行う
結果	1. 色の種類を30段階まで識別することができるため、完全に安定しているとはいえないものの、走行することができた 2. 結果2.色の種類を50段階まで識別することができたため、反射光の判定よりもさらに安定した走行をすることができ
結論	上記の結果から、より正確にライントレースができ、かつ解像度が高い手法である手法2を採用することで安定した走行を



○反射光で判定した場合のグラフ

…黒を-1、白を1としてラインを横切る形で移動させた際、色の識別の遷移が階段状になっており、解像度が荒いことがわかる



○カラーセンサーで色の判定を行った場合のグラフ

…反射光と同じ条件で測定した際、色の識別の遷移は反射光で判定した場合のグラフよりも滑らかであり、高解像度になっていることがわかる