機能

SAGA KUMIKOMI SOFT KENKYUKAI

基本方針

【ＥＴロボコン２０１０の目標】

私達のチームでは以下の事を目標に挙げ、開発を実施した。

（１）高速走行 ⇒ 技術要素1(P3)で解説

開発当初のプログラムはforward値160以上で転倒し、急なカーブでもforward値120以上でコースアウトとなった。そこで、旋回値の算出方法を見直し、forward値175以上でも問題なく走行出来るようにしたい。

（２）走行タイム短縮 ⇒ コース戦略(P5)で解説

走行タイムを出来るだけ短縮させたい為、難所は回避せずポイントを稼ぎたい。また、コースのショートカットを検討する。

（３）３Ｄコースへの対応 ⇒ 技術要素2(P4)で解説

シーソー、階段などの３Ｄコースを安全にクリアしたい。そこで、ジャイロセンサを活用した難所攻略法を検討する。

（４）リカバリー設計 ⇒ コース戦略(P5)で解説

難所通過時にコースアウトした場合のライン復帰など、イレギュラーを想定した設計、コース戦略を行い、安全に難所を通過出来る様にする。

（５）モデルの充実

組込み開発ではモデルの出来が重要視される。「astah」を導入して、記述ルールに沿ったモデリングを行い、バグの少ないプログラム完成を目指したい。

●●チャンピオンシップ大会に向けて●●

チャンピオンシップ大会では、以下に挑戦し、さらなる強化を目指す。

（６）シーソー上停止 ⇒ 技術要素2(P4)で解説

停止時間が1秒に変更された事により、成功の可能性がより高くなった。
⇒ 挑戦してポイントを稼ぎたい。

（７）階段をノンストップで通過 ⇒ 段差検知を技術要素2(P4)で解説

地区大会では、ジャイロオフセット値の調整による段差攻略を行ったが、通過に時間が掛かるのがデメリットだった。チャンピオンシップ大会では、段差検知を用いてノンストップで階段を通過する様に改良する。

（８）ミステリーサークルを華麗に通過 ⇒ 旋回走行を技術要素2(P4)で解説

地区大会では、角度変換を活用したミステリーサークル攻略を行ったが、走行体の停止を伴い、時間が掛かった。チャンピオンシップ大会では、旋回走行を用いてノンストップで華麗に通過出来る様に改良する。

【追加課題】

並行性設計への取り組みについて

ＥＴロボコン２００９ではシングルタスクで設計を行った。今回の追加課題である並行性設計（マルチタスク化）について、実施するか検討を行った結果、以下の理由から今回は**「並行性設計は行わない」**事に決定した。

理由１：
ＥＴロボコン２００９の資産（シングルタスクで設計）をベースに開発し、より品質の高いものに仕上げたい。
実績のある資産なので信頼性が高い**（品質的観点）**
理由２：
資産流用により新規開発をなるべく減らし、作業の効率を上げる**（コスト的観点）**
理由３：
開発効率を上げ、期間内に開発を完了させる**（納期的観点）**

【開発プロジェクトとしてのロボコン】

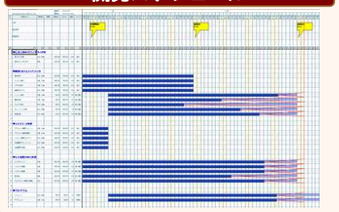
スケジュール表による進捗管理、情報の共有、作業の可視化を積極的に行い、一つの開発プロジェクトとして作業を実施した。

予定と実績の管理を行う

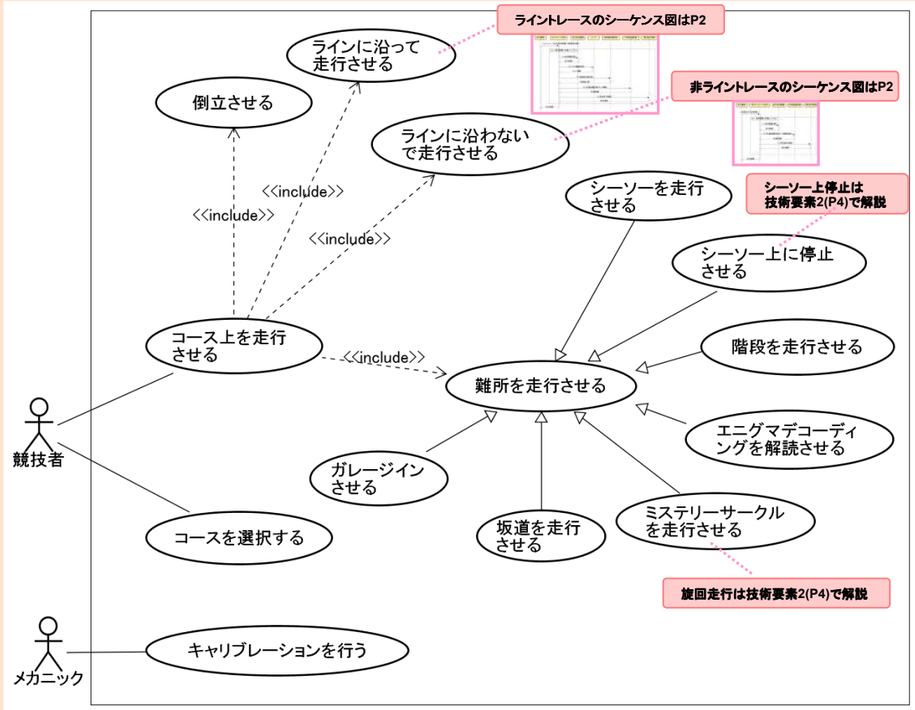
作業項目の可視化を行い、作業の漏れを防ぐ

試走会チェックシート

開発スケジュール



ユースケース図



【ユースケース記述（例）】

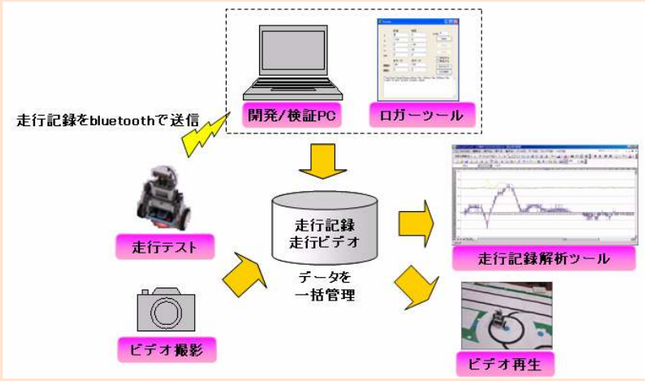
項目	内容
ユースケース	コース上を走行させる
概要	システムにコース上を走行させる
アクター	競技者
事前条件	キャリブレーションが行われていること
	コース選択が行われていること
	走行プログラムの実行が行われていること
事後条件	スタート地点からゴール地点まで完走すること
基本系列	1.アクターはシステムをコース上のスタート地点のライン上に置く
	2.アクターはシステムのタッチセンサを押す
	3.システムは状況に応じて走行方法を切り替えながら競技コースを走行する
	4.システムはゴール地点のゲートを追加する
	5.アクターはシステムをコース上から取り出す
例外系列	1.システムがコースアウトもしくは転倒した場合、アクターはシステムをコース上から取り出す

項目	内容
ユースケース	キャリブレーションを行う
概要	システムのキャリブレーションを行う
アクター	メカニック
事前条件	走行プログラムの実行が行われていること
事後条件	光センサの白値、黒値、グレー値の設定がされていること
基本系列	1.アクターはシステムをコース上に置く
	2.アクターはシステムのタッチセンサを押す
	3.システムはコース上のライン外（白色部分）を走行し、光センサの値を取得する
	4.アクターはシステムをコース上から取り出す
	5.アクターは取得した値を光センサの白値に設定する
	6.アクターはシステムをコース上に置く
	7.アクターはシステムのタッチセンサを押す
	8.システムはコース上のライン上（黒色）を走行し、光センサの値を取得する
	9.アクターはシステムをコース上から取り出す
	10.アクターは取得した値を光センサの黒値に設定する
	11.アクターはシステムをコース上に置く
	12.アクターはシステムのタッチセンサを押す
	13.システムはコース上のライン外（グレー色）を走行し、光センサの値を取得する
	14.アクターはシステムをコースから取り出す
	15.アクターは取得した値を光センサのグレー値に設定する
例外系列	なし

開発・検証環境

【開発環境】

私達のチームでは、bluetoothによる走行情報の収集・グラフ化による解析、およびビデオ再生による走行確認を組み合わせる事により、後戻りの少ない効率の良い作業を実施した。



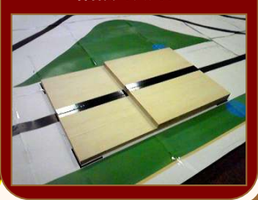
【品質向上への取り組み】

組込み開発はつねに「高品質」が求められる。
私達のチームではテストの重要性についてメンバー間の意識を合わせ、各種テスト用コースの作成を行い、テストの充実を図り品質向上に努めた。

シーソー（手作り）

実物大テストコース（手作り）

階段（手作り）

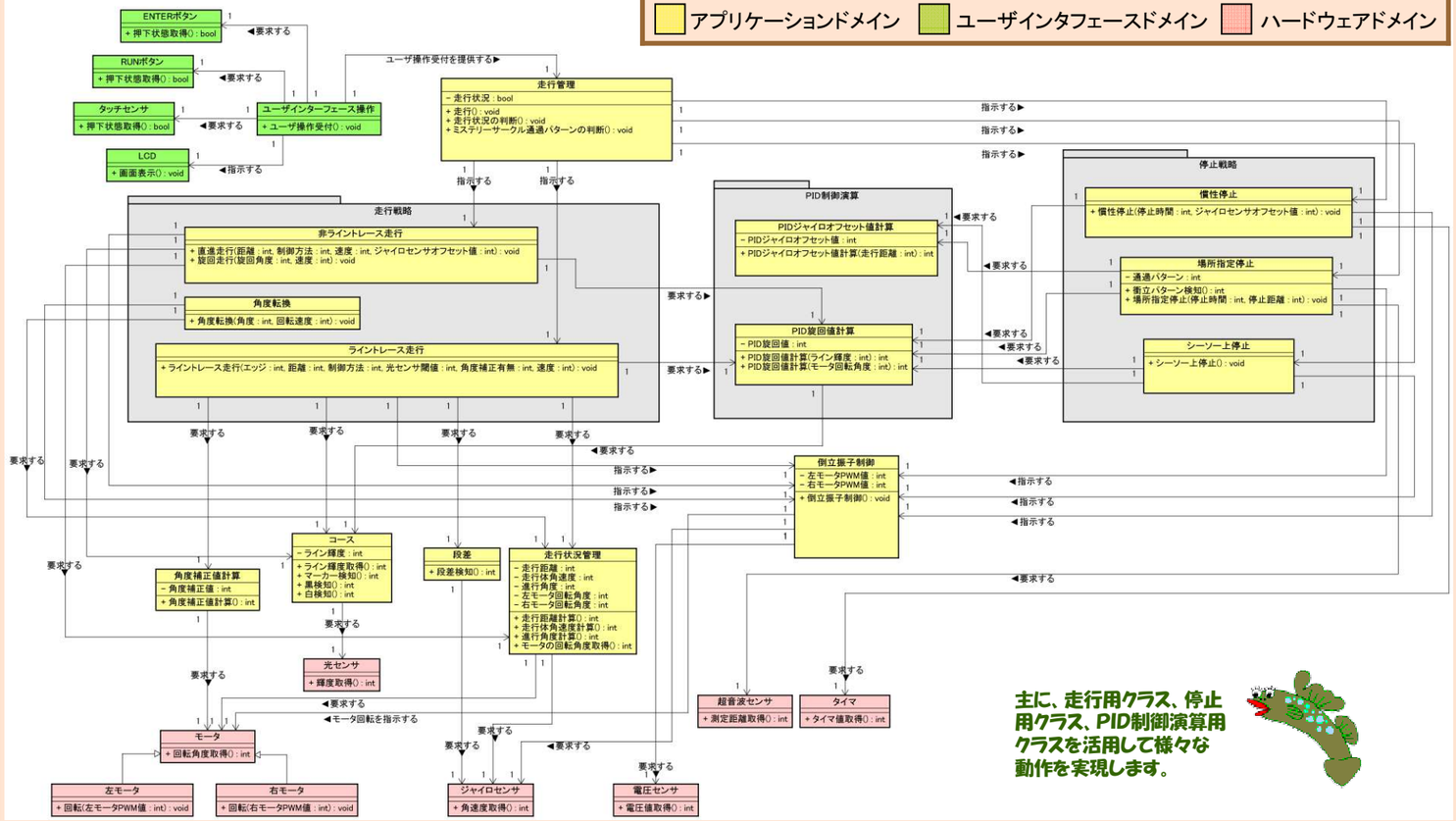


テストコースは紙を何枚もつなげて作ってます。

概要クラス図

クラスのドメイン分類

アプリケーションドメイン　ユーザインタフェースドメイン　ハードウェアドメイン

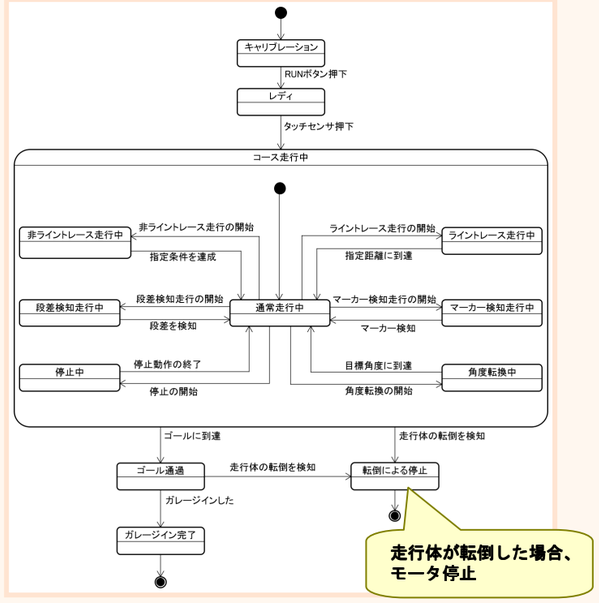


主に、走行用クラス、停止用クラス、PID制御演算用クラスを活用して様々な動作を実現します。

主なクラスの責務

クラス名	説明
走行管理	走行及び停止を指示する。
ライントレース走行	ライントレース走行をする。
非ライントレース走行	ライントレースを行わずに前進(後進)する。
慣性停止	減速しながら前進し、停止する。
場所指定停止	指定された場所で停止する。
シーソー上停止	シーソーの上で停止する。

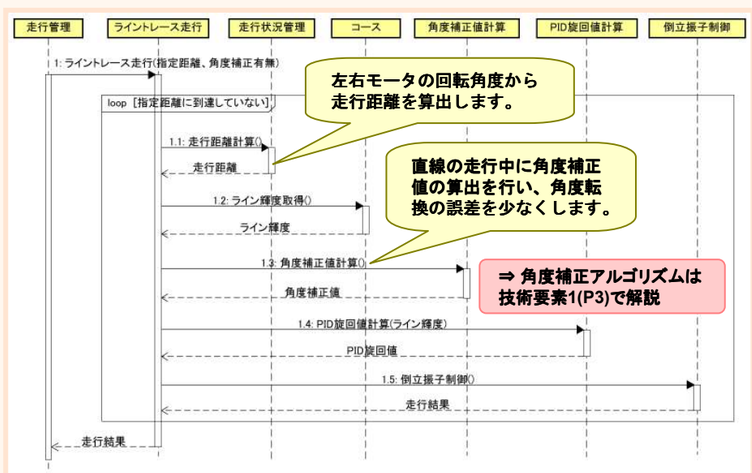
全体ステートマシン図



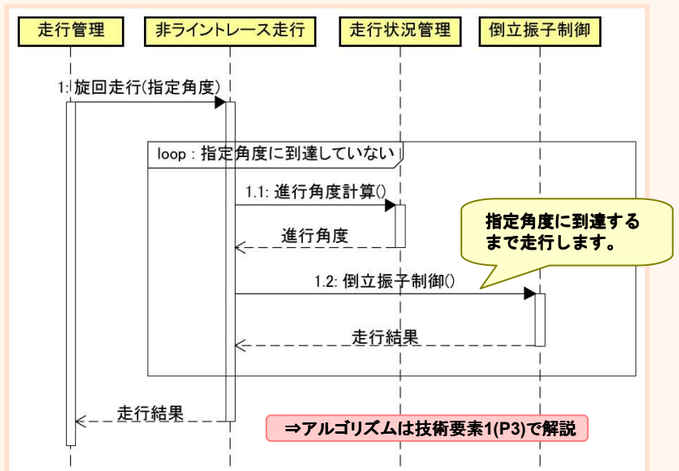
走行体が転倒した場合、モータ停止

シーケンス図

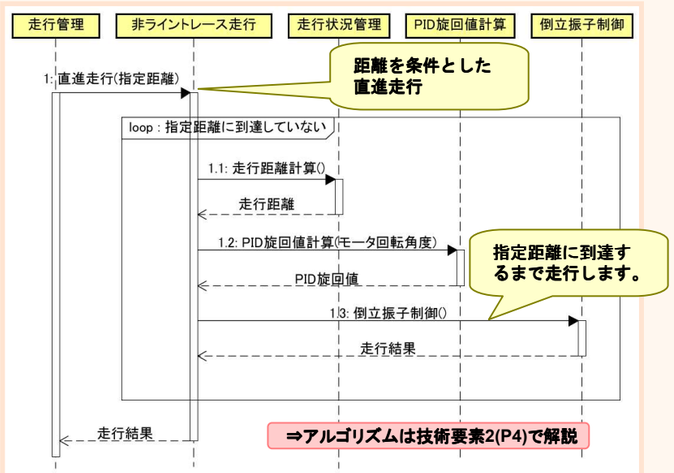
【ライントレースを行いながら、角度補正計算をする】



【角度転換】



【非ライントレース走行】



技術要素1

SAGA KUMIKOMI SOFT KENKYUKAI

PIDライントレース

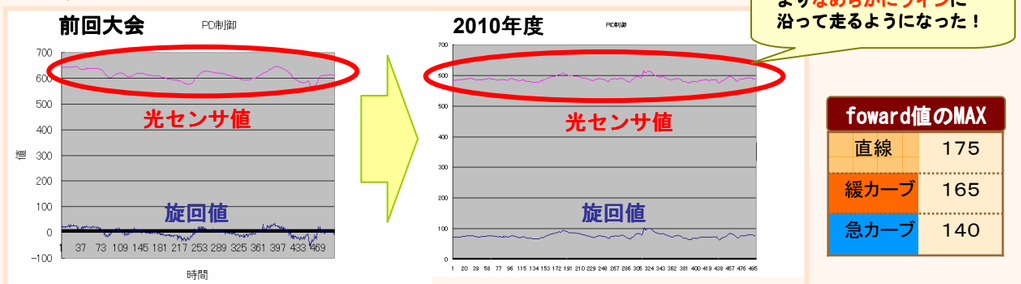
光センサの値から旋回値を求めるＰＩＤ制御を用いて、安定したライントレース走行を実現させる。
ＰＩＤ制御係数は他クラスからの指示により、コース状況（直線、緩やかなカーブ、急カーブ）にあわせて切り替える。
また、**ライントレース中は安定して直進走行が出来るため、直進走行中に「角度補正値の算出」を行うようにする。**

【アルゴリズム】

①偏差の履歴管理
今回の偏差＝（閾値－光センサ値）
偏差履歴２＝前回の偏差履歴１
偏差履歴１＝前回の偏差

②旋回値の計算
Ｐパラメタ＝Ｐ制御係数×（偏差－偏差履歴１）
Ｉパラメタ＝Ｐ制御係数×Ｉ制御係数×今回の偏差
Ｄパラメタ＝Ｐ制御係数×Ｄ制御係数×（今回の偏差－（２×偏差履歴１）＋偏差履歴２）
操作量＝前回の操作量＋（Ｐパラメタ＋Ｉパラメタ＋Ｄパラメタ）×１
※右エッジの場合
操作量＝前回の操作量＋（Ｐパラメタ＋Ｉパラメタ＋Ｄパラメタ）×－１
※左エッジの場合

【実装結果】（光センサ値と旋回値のグラフ）



foward値のMAX	
直線	175
緩カーブ	165
急カーブ	140

超音波センサ

検知方法：
誤検知の可能性を考慮し、一定期間の間40msec周期で前方障害物の検知を繰り返し、検知率を算出する考え方とする。
超音波センサからの結果（距離）が検索指定距離内なら「あり」と判断し、その結果をカウントする。

最終的に、総検知回数と、検知の結果「あり」の回数から検知率を算出し、検知率が基準値以上なら最終的に「障害物あり」と判断する。

【検知率の算出】

検知率(%) ＝ 「あり」の回数 ÷ 総検知回数×100

⇒検知率が、基準値(%)（※５）以上なら「障害物あり」と判断する。

（※５）基準値(%)はキャリブレーション時に決定する。

【実装結果】

上記の方法で検知すると、衝立ａ・ｂの誤検知が無くなった。
そのため、ミステリーサークル通過パターンを衝立ａ・ｂで決定できる為、衝立ｃは検知する必要が無いと判断出来る。

マーカー検知

検知方法：
ライントレース走行中に、急激に黒になった場合にマーカー（マーカーと黒ラインの継ぎ目）と判定する。

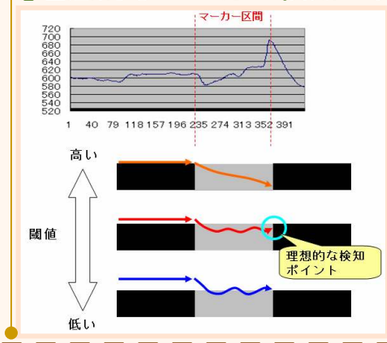
検知までの処理の流れ：
①現在の光センサ値を擬似的なリングバッファに格納

②バッファ上の最低値を求める

③最低値が検知限界値より小さければ、
最低値を検知限界値にする

④現在の光センサ値とバッファ上の最低値との差が
基準値以上ならマーカーありと判断する

【理想的なマーカー検知パターン】



正確なマーカー検知をするための閾値設定は、下記の②のルートをたどるように設定されるのが理想である。

①コースアウトの危険性あり

②理想的な検知パターン

③検知しない可能性あり

角度転換（方向転換）

現在の進行方向（角度）から、目標角度に到達するまで走行体を回転させる。

【角度転換アルゴリズム】

①現在の角度を求める
回転角度差＝（左モータ回転角度－右モータ回転角度）
スタート時の角度を０度とした場合の現在の相対角度（※１）＝

$$\mathrm{MOD}\left(\left[\frac{\text{タイヤ円周率}\times\text{回転角度差}}{360}\right]\div\text{回転円周率}\right)\times\frac{360}{\text{回転円周率}}$$

回転円周率・・・上方より見て、回転した場合の円周

現在角度（※２）＝
スタート時の角度を０度とした場合の現在の相対角度 ＋角度補正値（※３）

②目標までの角度差を求める
目標までの角度差（※４）＝現在角度－目標角度

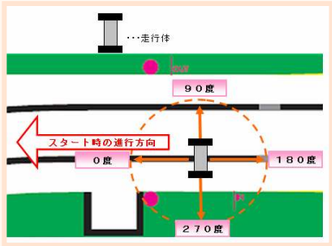
③モータを旋回させる
右旋回のとき：
左モータの出力値＝回転時の速度÷２
右モータの出力値＝回転時の速度÷２×－１

左旋回のとき：
左モータの出力値＝回転時の速度÷２×－１
右モータの出力値＝回転時の速度÷２

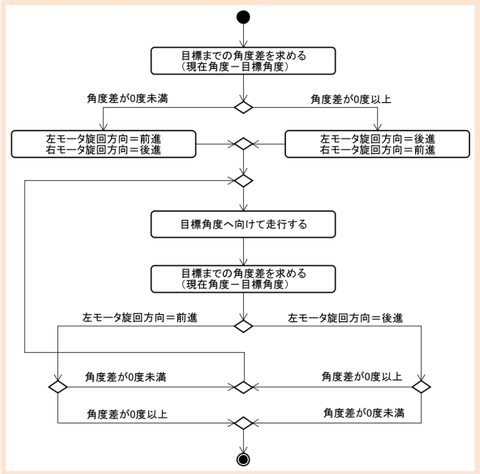
（※１）０度よりマイナスの場合は＋３６０度する
（※２）０度よりマイナスの場合は＋３６０度する
３６０度よりプラスの場合は－３６０度する
（※３）角度補正値の算出式は技術要素１「角度補正値の算出」に記載
（※４）－１８０度よりマイナスの場合は＋３６０度する
＋１８０度よりプラスの場合は－３６０度する

【角度の定義】

スタート時の進行方向を０度として下図の様に定義する。



【アクティビティ図】



角度補正値の算出

通常走行や難所攻略時に発生するタイヤのスリップ等の影響により、
進行方向（角度）の算出に誤差が生じる可能性がある。
そのため、直線を安定して走行している時の、左右モータの正常な回転角度差を求め、角度補正値を算出する。

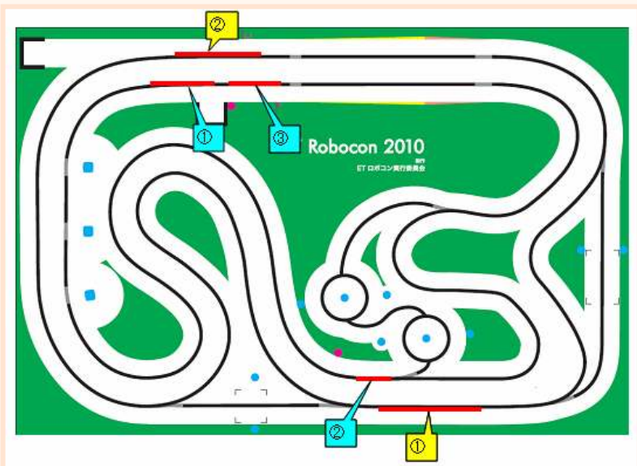
【アルゴリズム】

①本来の角度との誤差を求める
本来の角度との誤差（※６）＝（現在の角度（※７）－本来の角度）
本来の角度との誤差（累積値）＋＝本来の角度との誤差

②補正値を求める
角度補正値＝本来の角度との誤差（累積値）÷補正回数

（※６）
－１８０度よりマイナスの場合は＋３６０度する
１８０度よりプラスの場合は－３６０度する
（※７）
現在の角度の算出式は技術要素１「角度転換」に記載

【理想的な角度補正ポイント】



技術要素2

SAGA KUMIKOMI SOFT KENKYUKAI

傾き制御

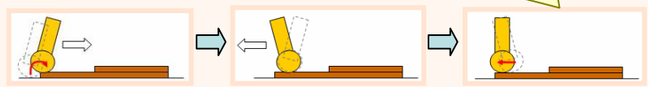
難所攻略において、ジャイロオフセット値を一時的に変更し、傾き制御を利用した走行や停止を行う。
急発進したい時は、ジャイロオフセット値を大きくする。
急ブレーキを掛けたい時は、ジャイロオフセット値を小さくする。
PIDジャイロオフセット値算出により、走行体をより早く停止させる。

【活用例①：シーソー／階段攻略】

地区大会での実績：難所通過まで約９秒

ジャイロオフセット値を大きくする　倒立振子制御により急発進する勢いでシーソーを上る

または・・・

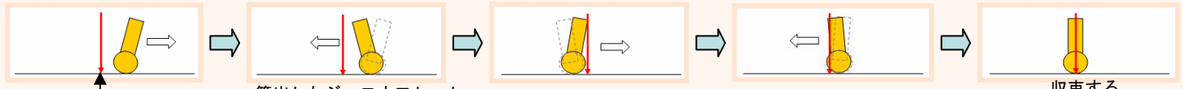


倒立振子制御により急発進する勢いで階段を上る　ジャイロオフセット値を小さくする　倒立振子制御により急ブレーキが掛かる

地区大会での実績：難所通過まで約１３秒

【活用例②：走行体の停止】

正しい位置に停止するための前後進制御をforward値で制御するよりも、ジャイロオフセット値で制御する方が停止までの収束が早い事を利用する。



停止させたい位置　算出したジャイロオフセット値に変更　収束する

地区大会での実績：ガレージイン後停止まで約１秒

【PIDジャイロオフセット値算出アルゴリズム】

①偏差の履歴管理
今回の偏差＝（目標距離－現在の距離）
偏差履歴２＝前回の偏差履歴１
偏差履歴１＝前回の偏差

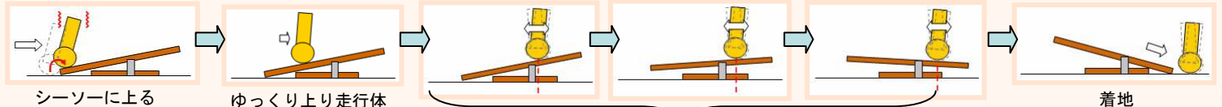
②ジャイロオフセット値の計算
Ｐパラメタ＝Ｐ制御係数×（偏差－偏差履歴１）
Ｉパラメタ＝Ｐ制御係数×Ｉ制御係数×今回の偏差
Ｄパラメタ＝Ｐ制御係数×Ｄ制御係数×（今回の偏差－（２×偏差履歴１）＋偏差履歴２）
今回のジャイロオフセット値＝前回のジャイロオフセット値＋（Ｐパラメタ＋Ｉパラメタ＋Ｄパラメタ）

【活用例③：シーソー上停止】

PID制御による停止ロジックを応用して、以下の方法でシーソー上停止を実現する。

実現の為のポイント：
①左右のモータエンコーダからの回転角度から、支点のおおよその位置まで上る。
②シーソー上でバランスを取りながら、バランスを取る位置を少しずつ前進させ、１秒間の停止条件を満たす様にする。
③エンコーダの回転角度から、現在位置が前方に大きくずれた場合は、シーソーから落下したと判断する。

本方式のメリット：
停止位置を徐々に前方にずらしていく為、シーソー上停止に失敗（１秒未満の停止）しても、必ずシーソーを通過する事ができ、リタイヤを回避できる。



シーソーに上る　ゆっくり上り走行体を安定させる　バランスを取りながら停止位置を前方にずらしていく　着地

直進走行（非ライントレース）

ライントレースを行わずに直進し、指定した距離を走行する。目標距離は左右モータ回転角度から求める。
また、左右モータの回転角度差を閾値に設定し、PID制御を用いて旋回値を算出する。

【アルゴリズム】

①閾値と目標モータ回転角度の算出
・閾値＝左モータ回転角度－右モータ回転角度
・目標モータ回転角度＝

$$\left[\frac{(\text{左モータ回転角度（累計値）}+\text{右モータ回転角度（累計値）})}{2}\right]+\frac{(\text{指定距離}\times 360)}{\text{タイヤ円周率}}$$

②走行
偏差を更新しながら、ＰＩＤ旋回値を求め走行する。
現在の走行距離が目標モータ回転角度に到達したら走行を終了する。
偏差＝（左モータ回転角度－右モータ回転角度）－閾値

【オプション機能の実装】

→コース戦略で黒検知走行と白検知走行を使用(P5)

距離だけでなく、以下の条件を追加し、様々な状況に対応出来る様にする。
①黒検知走行：指定した光センサ値**以上**の値を検知するまで直進走行する。
②白検知走行：指定した光センサ値**未満**の値を検知するまで直進走行する。

旋回走行

走行しながら進行角度を変える事により、走行体を停止させずに角度を変える事が出来る。

【アルゴリズム】

①左右の目標タイヤ回転角度を算出
・旋回目標相対角度(-360～+360)（※）が反時計回りのとき

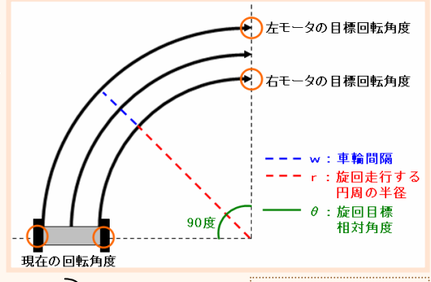
左モータ目標回転角度 ＝

$$\frac{\left[(2\times\pi\times(r+w))\times\frac{ABS(\theta)}{360}\right]\times 360}{\text{タイヤの円周}}$$

右モータ目標回転角度 ＝

$$\frac{\left[(2\times\pi\times r)\times\frac{ABS(\theta)}{360}\right]\times 360}{\text{タイヤの円周}}$$

・旋回目標相対角度(-360～+360)（※）がプラスの時は計算式が左右逆転となる。



②仮想円からみた軌跡の偏差を計算する

$$\text{偏差}=\left[\frac{\text{左モータ回転角度}}{\text{左モータ目標回転角度}}\right]-\left[\frac{\text{右モータ回転角度}}{\text{右モータ目標回転角度}}\right]$$

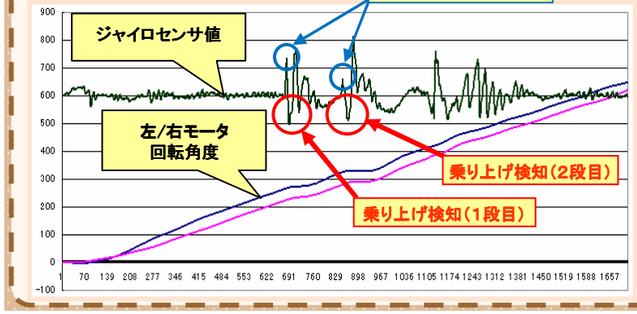
③偏差をもとにPID旋回量を計算する
※計算方法は直進走行時と同じ

（※）＋は時計回り
　　－は反時計回り

段差検知

段差検知の考え方：ライントレース走行中に、急激にジャイロセンサ値が変動(オフセット値に対して一定値以下)した時に、段差に乗り上げたとみなす。
衝突した瞬間はオフセット値に対してセンサ値が増大するが、その後センサ値が下がるタイミングを段差乗り上げとして検知する。

【走行記録による分析】



【理想的な段差検知をする為には】

理想的な段差検知をする為には、以下の調整が必須である。

①適切なスピード調整
速度が速すぎると、乗り上げ後バランスを崩して転倒してしまう。
速度が遅すぎると、勢いが足りずに乗り上げが出来ない。
⇒そのため、適切なスピード調整が必要である。

②接触時の走行体の安定状態
段差に対して、垂直に接触しないと、バランスを崩して転倒する可能性が高い。その為、段差の直前は、ラインに沿って安定して走る事が重要である。
⇒PID制御パラメタと、速度を一定に保ちながら、走行体が安定した状態で段差に接触する様にする。

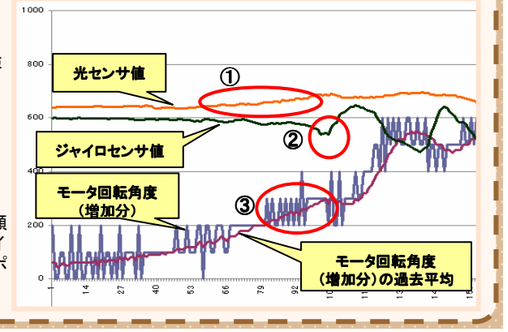
シーソー支点検知への試み

シーソーの支点を検知し、シーソー上停止を行う事が出来ないか、各種センサからの情報を収集して検証を行った。
しかし以下の結果から、**シーソー支点検知は行わずシーソー乗り上げ後からの距離で、支点のおおよその位置を判断する方式が安全**と判断した。

①ジャイロセンサ値
シーソーが傾いたときに、ジャイロセンサが反応する。
しかし、タイミング的にはシーソーが傾いた後の為、支点検知ポイントとしては適切ではない。

②光センサ値
シーソーが傾き始めた時に、光センサ値が変化する。
しかし、急激な変化ではない為、支点検知ポイントとしての判断が難しい。

③左／右モータエンコーダの回転角度増加量
モータエンコーダの一定間隔の回転角度の増加量が加速傾向にあれば、シーソーが傾き始めたと判断出来るが、ジャイロセンサでの検知とほぼ同タイミングである為、支点検知ポイントとしての判断が難しい。



コース戦略

SAGA KUMIKOMI SOFT KENKYUKAI

シーソー

【攻略ポイント】

●●地区大会●●

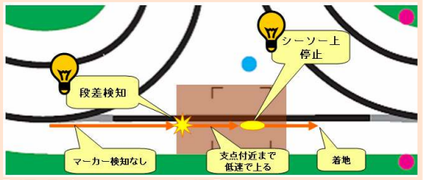
①マーカー検知
②傾き制御を利用した乗り上げ
③コース着地後のリカバリー走行

新戦略により、シーソー到着から通過までの時間が約１秒短縮！

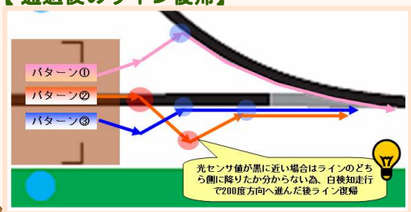
●●チャンピオンシップ大会●●

①マーカー検知の廃止&段差検知によるシーソー乗り上げの判断
②シーソー上停止
③コース着地後のリカバリー走行

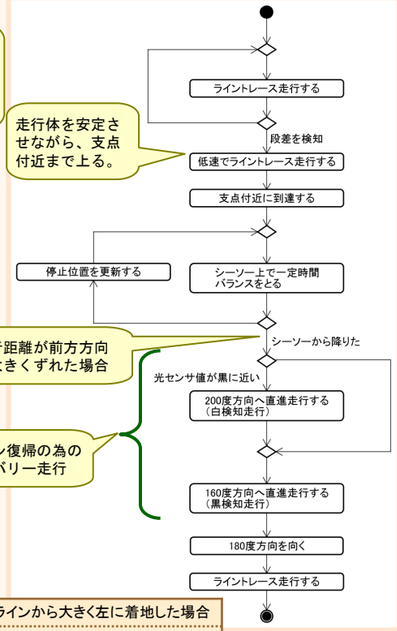
【攻略手順】



【通過後のライン復帰】



アクティビティ図



パターン① ラインから大きく左に着地した場合
パターン② ライン周辺に着地した場合
パターン③ ラインから大きく右に着地した場合

エニグマ・デコーディング

【攻略ポイント】

●●地区大会●●

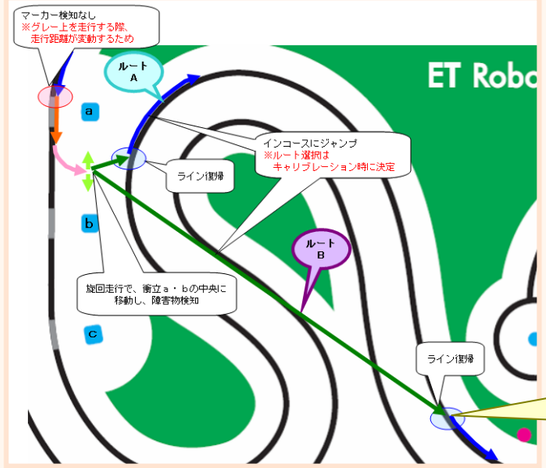
①角度転換を用いて、衝立ａ・ｂの間へ走行
②ショートカット（ルートＡ）

●●チャンピオンシップ大会●●

①衝立ａ・ｂの間までの走行方法を改善（マーカー検知の廃止。旋回走行使用）
②ショートカット（ルートBを目指す）

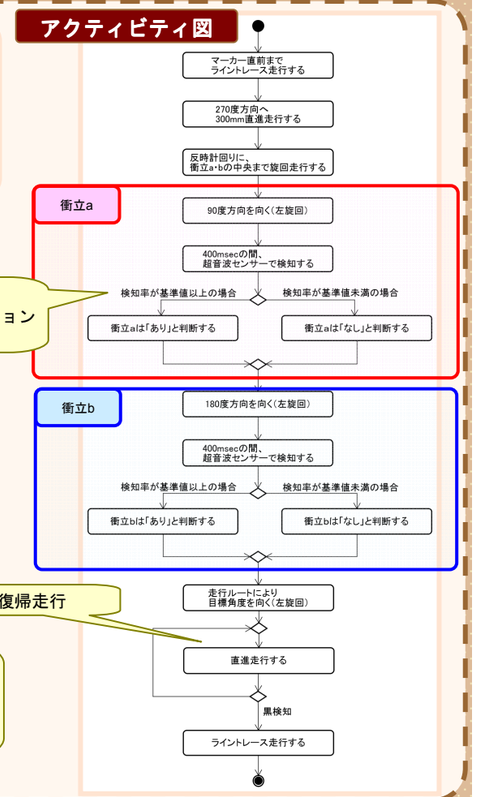
ライントレース走行
直進走行
直進走行（ライン復帰を行う）
旋回走行
超音波検知

【攻略手順】



新戦略により、スタートからライン復帰までの時間が約２秒短縮！

アクティビティ図



※基準値：0%～100%
値はキャリブレーション時に決定

ライン復帰走行

階段

【攻略ポイント】

●●地区大会●●

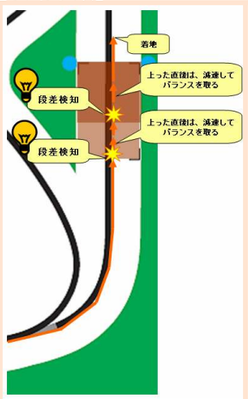
①マーカー検知
②傾き制御を利用した乗り上げ
③コース着地後のリカバリー走行

新戦略により、階段到着から通過までの時間が約７秒短縮！

●●チャンピオンシップ大会●●

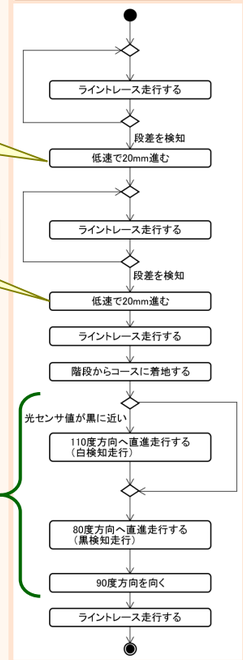
①マーカー検知の廃止と段差検知による階段乗り上げの判断
②速度調整をしながら階段を通過
③コース着地後のリカバリー走行

【攻略手順】



【通過後のライン復帰】

アクティビティ図



ミステリーサークル

【攻略ポイント】

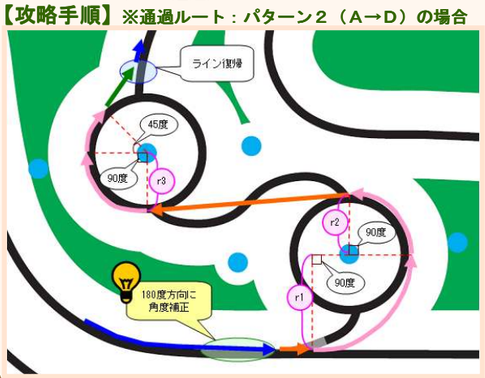
●●地区大会●●

①ミステリーサークル通過前の角度補正
②**角度転換**と直進走行によるゲート通過

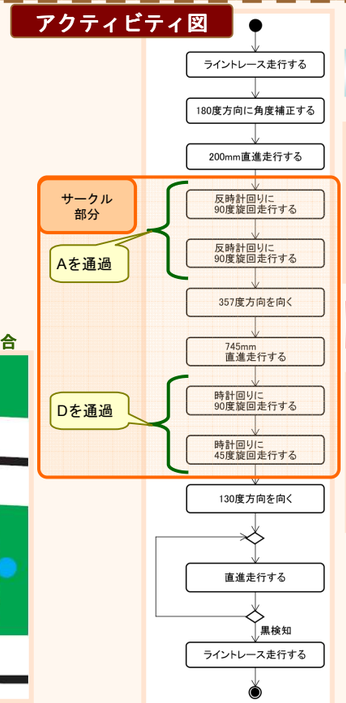
●●チャンピオンシップ大会●●

①ミステリーサークル通過前の角度補正
②**旋回走行**と直進走行によるゲート通過
（ノンストップで通過）

【攻略手順】※通過ルート：パターン２（A→D）の場合



アクティビティ図



【パターンごとの通過ルート】

●●地区大会●●

■パターン１、４

■パターン２、３

●●チャンピオンシップ大会●●

■パターン１、４

■パターン２、３

パターンごとの走行タイム（秒）

	1	2	3	4
旧	12	13	12	13
新	8	9	8	9