

チームID：28

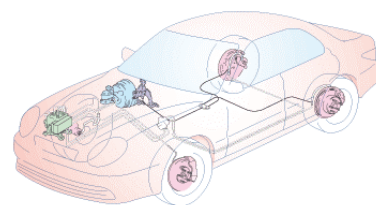
チーム名：HELIOS



所属：株式会社 アドヴィックス

☆チーム紹介

世界屈指の輸送機用ブレーキ部品サプライヤ

株式会社 アドヴィックス (HP <http://www.advics.co.jp/jp/>)

☆組込み、そしてモデリングの未来へ一言

SysML/UMLモデルと MATLAB/Simulink モデルによるモデルベース開発を推進して、機能安全に対応した量産品を世界に提供する。

☆コンテストにかける意気込み、アピール

アドヴィックスのETロボコンへの挑戦は、社内有志5名の個人参加によって、2006年に始まった。2007年からは会社のバックアップを受けて企業参加に切り替え、若手エンジニアの技術教育として取り組んできた。

2008年には東海地区大会でモデル部門、走行部門、総合部門の3冠を達成(ロボコン史上初)し、同年のチャンピオンシップ大会においても総合優勝に輝いた。

2009年は東海地区大会、チャンピオンシップ大会においてエクセレントモデルを獲得してモデル部門を制覇するものの、あと一歩及ばず、総合準優勝となり惜しくもV2を逃している。

今年は入社2~3年目の若手4名(内2名は昨年からの継続メンバ)でチームを構成し、全国制覇を経験したメンバがバックアップする体制で、2010年チャンピオンシップ大会での総合優勝奪還に挑む。

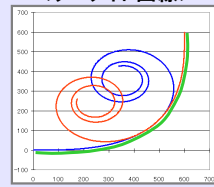
『ADoniSからHELIOSに受け継がれる伝統は“技術屋魂”!』

☆ここに注目!

■ポイント1:クロソイド曲線の適用

直線と円弧の間に、クロソイド曲線と呼ばれる緩和曲線を設けてスムーズに走行する。これにより、ミステリー・サークル(難所)をスムーズに攻略。

<クロソイド曲線>



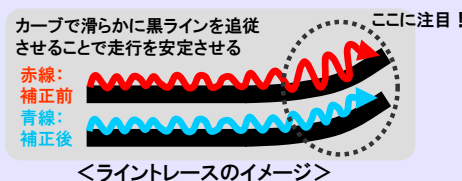
<走行経路マップ用の座標データ>

ポイント	X座標	Y座標	左輪	右輪
0	0	0	0	0
1	501.597	0.129478	48.5923	50.7277
2	100.3104	1.035766	98.0231	102.6169
3	151.5441	3.574729	146.3531	156.6869

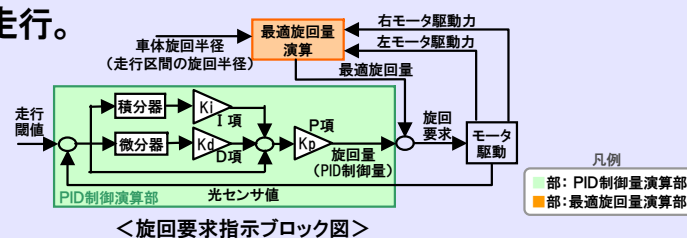
—:補間開始点からのクロソイド曲線
—:補間終了点からのクロソイド曲線
—:目標走行軌跡(合体したクロソイド曲線)

■ポイント2:曲率の異なるカーブを滑らかに走行

曲線でPID制御の基準値をカーブの旋回半径に適した旋回量に補正して、コース全体を滑らかに走行。

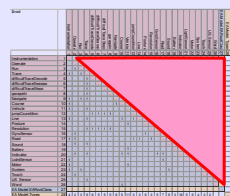


<ライトレースのイメージ>



<旋回要求指示ブロック図>

■ポイント3:構造解析(Dependency Structure Matrix)による品質分析
ソフトウェアの構造解析を通してモジュール(クラス)の独立性を検証し、保守性、移植性の高い構造設計を維持する。また、この検証結果の活用を今後の派生開発に役立てる。



<構造解析結果>

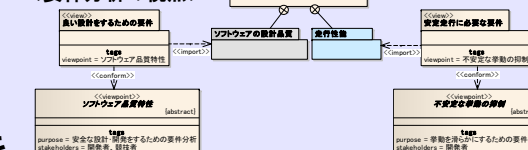
☆設計思想

■基本コンセプト

『コース全域(難所を含む)を安全に走破する』

1. 全ての難所を安全に走行するために必要な要件を分析
2. 安全に走行するために必要な非機能要件の分析を「走行性能の要件」と「ソフトウェア設計品質の要件」の2面から分析アプローチ(SysML活用)
⇒抽出した非機能要件や制約事項を“機能要件”、“設計方針”、“実装方針”、“開発環境”の各方面で対応

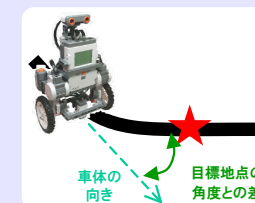
<要件分析の視点>



<機能要件>

◇安全性確保のために必要な機能要件を追加

- ・必要な補正機能追加(旋回半径に応じたPIDの基準値の補正)
- ・特性のバラツキによる影響を補正
- ・例外ケース時の対応シナリオ追加(各難所の走行ロジックでのリスク分析)



<設計方針>

- ◇アーキテクチャ要件への対応(独立性の確保、再利用部品の活用など)
- ◇タスク設計での処理時間制約への対応

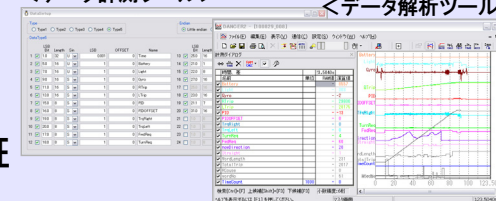
<実装方針>

- ◇命名規約(社内規約準拠)やコーディングルール(MISRA-C準拠)の工夫と展開
- ◇UMLツールEnterprise Architect / Ageにてスケルトンコード生成

<開発環境>

- ◇データ計測・解析ツールの活用(自作ツール)
- ◇Visual Source Safeによる構成管理&差分管理
- ◇Lattix & Understandによるアーキテクチャ設計検証

<データ計測ツール>



<データ解析ツール>

☆追加課題への取り組み

■並行性の設計方針

- ◇RTOSのオーバーヘッド時間により処理時間が増加するので、タスクの数を増やしすぎたくない
- ◇同期実行させたい処理は同じタスクに割り付ける
- ◇時間が掛かる処理または長い演算周期でよい処理は別タスクにして制御部の処理負荷を軽減する
- ◇操作装置の走行前処理は走行中処理の負荷にはならないので制御部と同じでよい
- ◇本来の機能以外の負荷機能(開発時だけ使用して競技時に使用しない機能)を別タスクにして、負荷機能使用時においても制御部の処理負荷を開発時と競技時で同等にする。これにより、負荷機能の実行が本来の機能に影響を及ぼさないようにできる

<タスク間のやりとり>

