
TGMGP
HONDA
YOKOHAMA
eparco
A2RL

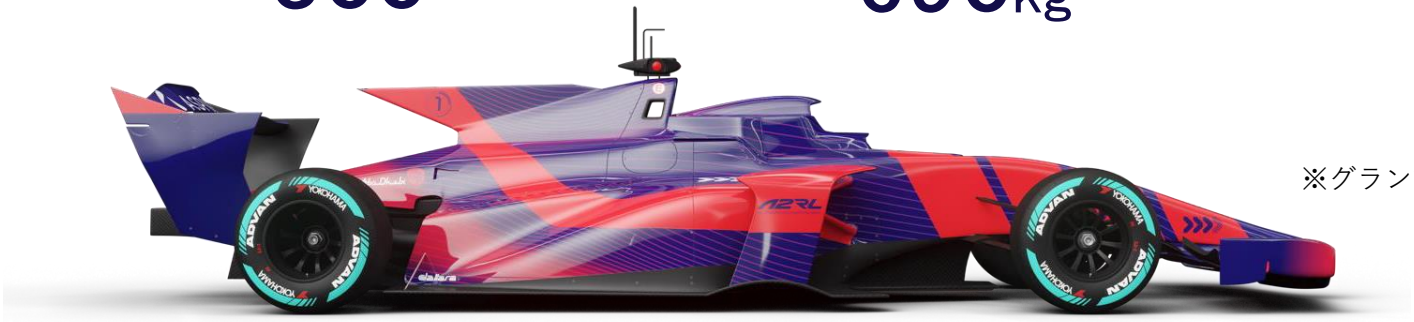
A2RL

■ アブダビで開催される世界初の自律走行フォーミュラカーレース ※

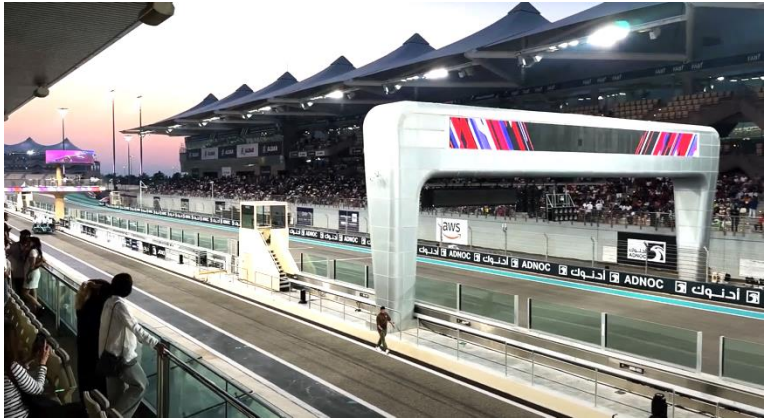
■ ハードウェアは全チーム共通。ソフトウェア（**ROS 2**）の性能のみで勝負。

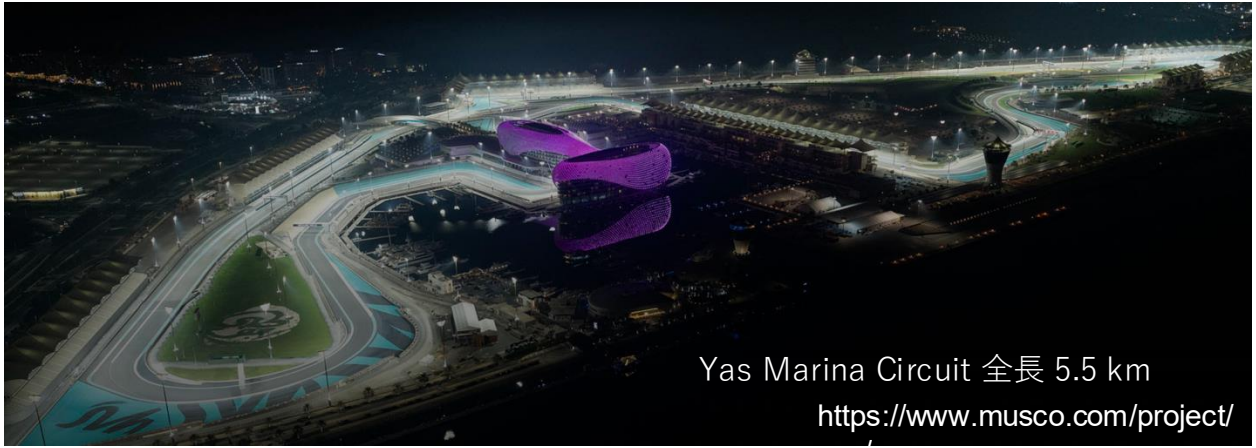
MAX SPEED
300km/h

WEIGHT
690kg



※グランプリ形式において


ADNOC
أدنوك


Yas Marina Circuit 全長 5.5 km
https://www.musco.com/project/

TGMGP はA2RLへ挑戦する世界唯一のレーシングチーム

2024年の A2RL Abu Dhabi Autonomous Racing League 開幕当時、日本チーム不在での戦い目の当たりにし

「**自分たちが今この場にいなければ、日本の技術開発は遅れを取ってしまう**」

そう感じ参戦を決定

TGMGP × **日本の自動運転／ロボティクスのエンジニア／研究者**

- ✓ 車両運動力学
- ✓ タイヤ特性、空力
- ✓ 車両とレースへの深い知見

- ✓ Perception
- ✓ Localization
- ✓ Planning
- ✓ ROS 2
- ✓ …

全国の優秀な技術者・学生に「競争」「チャレンジの場」を提供し

レースを通じて自動運転技術の向上に貢献していく

自動運転の様子 （2025/04時点）

- 120km/hでの単独安定走行を達成 (2025/04時点)
- 高速化＆マルチカーレースに向けて鋭意開発中

1. 車両運動力学／タイヤ特性／空力を踏まえた制御

例）タイヤの摩擦特性は温度や路面状態で大きく変化。 制御失敗はスリップにつながる。

2. 自車状態の推定

- ３００km/hでは1秒で83m進む！高速な推定やセンサ遅延補償が重要。
- 自己位置、速度、スリップ角、ヨーレート、路面摩擦係数などの推定が必要

3. 他車認識

- 高速な認識が必要。100msの認識遅れで8mの誤差。
  前方にスピン車両が現れたら300km/hで回避が必要。
- 走行振動によるセンサ異常を前提とした複数センサ活用

4. 行動計画

- 他車の動きは非決定的。 複雑なレギュレーション、安全、レース（追い抜き）の両立も必要。

5. 多数のセンサの使いこなし

- センサ数が多い

  Camera×7 / Radar×3 / LiDAR×3 / GNSS×2 / IMU×2 / INS×1 / 対地速度 / 車輪速 / TPMS…
- 基本的な使いこなしだけでも大変

  ドライバ / 特性理解 / 異常検知 / センサ間同期

6. 実車に触れる機会が限定的

- アブダビに行かないと触れない。チームに与えられる走行機会も限りあり。
- シミュレータ活用、ＣＩの重要性、日本国内でいかに周到な準備ができるか

7. フェイルセーフ

- 走行振動が大きく、センサ異常のリスクが高い。
- ノードや状態数が多く、異常判定だけでも大変
- そもそも異常時の適切な対応が難しい。急ブレーキは追突やスリップリスクがあり危険。

ROS 2 活用の課題と対応策

カテゴリ	課題内容	現状の解決策
開発環境の統一	ROS 2 に限らず、シミュレータ環境や CUDA、各種依存パッケージ、DevContainer, フォーマッタ、Linter などの環境を共通化したい	• Docker + 環境整備用の IaC のようなリポジトリを用意して運用 • 環境構築コストやコーディング作法の差異などによるロスを削減 • 各ツールや環境構築に不慣れなメンバーでもすぐに開発に着手できる環境を整備 • 実機環境、クラウド環境、ローカル環境とでそれぞれ設定を変える必要があるが、共通的な部分を再利用しつつ、差分を吸収できるように実装
ros2 bag	しばしば記録に失敗し破損してしまう。走行機会が限られるため、ログデータの取得失敗は死活問題。	• デフォルトの db3 だと復元が困難であるため mcap を使用 • サイズおよび記録時間で逐次分割して保存することで破損時に全てのデータが失われるリスクを軽減 • 容量が大きいため zstd 圧縮も活用
ros2 launch	• launch.py の仕様が難解 • 特に、launch arg とデフォルト引数がグローバル的に扱われる仕様が罠	検討中
use_sim_time	• Simulation, 実機切り替え時に必要となる • ROS 1 ではグローバルパラメータサーバに設定するだけで良かったが、ROS 2 ではノード毎に個別に渡す必要がある • 全てのノードに適切に use_sim_time パラメータを渡そうと思うと、ノード実装や launch の書き方で意識が必要。負担が大きい。 • 外部パッケージ実装で ROS_TIME を使用していない場合 (SYSTEM_TIME や chronoなど） には、実装を変更しなければならず手間がかかる （あるいは不可能）	検討中

ROS 2 活用の課題と対応策

カテゴリ	課題内容	現状の解決策
ROS 2 パッケージ作成	• ボイラープレートが多く、新規パッケージの作成・立ち上げに手間がかかる • ros2 pkg create などでも最小限の構成は作成できるが、generate_parameter_library や Component クラスの雛形なども用意したい	テンプレートパッケージを用意し、スクリプトを一回叩くだけで新規パッケージの雛形を作成できるようにした
DDS/ネットワーク－1	• インターネット経由での通信が必要 • サーキット走行時は携帯回線越しのVPNを構築して手元PCと車体を接続するが、ブロードキャスト制限や通信帯域圧迫の問題がある	• DDSではなく websocket_bridge や zenoh_bridge を活用 • 帯域節約のため、モニタリングのために bridge する topicは最小限、最小レートに制限する必要がある • モニタリング用途では foxglove_bridge + Foxglove がパフォーマンスに優れた • 圧縮、再送制御などが他bridgeよりも効率的 • FastDDS Discovery Server の活用も今後検討したい
DDS/ネットワーク－2	Cyclone DDS がうまく機能しない場合があった	• Fast DDS に切り替えるとより安定した • ただし、パフォーマンス改善のためには、少しチューニングが必要であった • 実運用レベルでのDDS利用にはノウハウが必要と痛感

TGMGPのブース：26