TOPPERS 活用アイデア・アプリケーション開発コンテスト

部門　：　アプリケーション開発部門
作品のタイトル　：　ROS 通信による ET ロボコン走行体の制御アプリケーション開発用プラットフォーム

作成者　：　樋山　一樹（南山大学）

共同作業者　：

対象者　：　LEGO SPIKE Prime 使用者, ROS2 プログラミング初学者, ET ロボコン参加者

使用する開発成果物　：　TOPPERS/ASP3，SPIKE-RT，micro-ROS_ASP3

目的・狙い

- SPIKE-RT と ROS 通信を用いた ET ロボコン走行体制御用アプリケーションの開発環境の提供.
- ROS2 プログラミングの学習教材の提供.

アイデア/アプリケーションの概要

本作品は，ET ロボコン走行体(HackSpi)の制御アプリケーション開発のためのソフトウェアプラットフォームである.
SPIKE-RT を用いた SPIKE Prime Hub と RaspberryPi の間を ROS で通信して動作する環境となっている.

1. 背景

本作品の作成の背景に，ROS，SPIKE-RT，ET ロボコンの存在がある．

ROS とはロボットや自動運転向けの分散処理フレームワークである．ROS には研究目的で開発された ROS1 と産業目的で開発された ROS2 の他に，RTOS やベアメタル環境で動作するマイコンを，ROS2 に接続するための機構である micro-ROS（uROS）が存在する．ROS の利用者は近年増加傾向にあり，ロボット開発の分野で注目の技術となっている．uROS の構成を図 1 に示す．

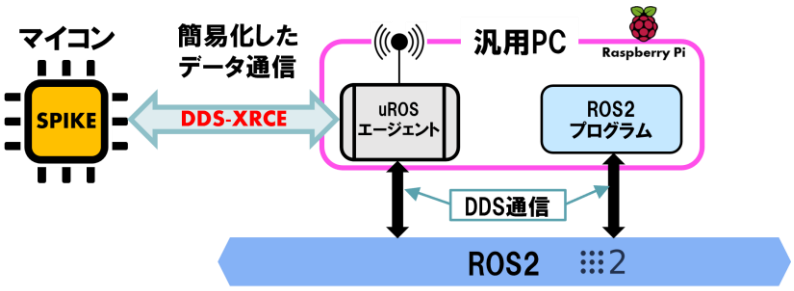

図 1．uROS の構成

SPIKE-RT は，近年開発された LEGO® Education SPIKE™ Prime（SPIKE Prime）向けの RTOS であり，TOPPERS の ASP3 カーネルが使用されている．私が所属する南山大学本田研究室では，この SPIKE-RT と uROS を組み合わせた取り組みを実施しており， SPIKE-RT 向けの uROS ミドルウェアである micro-ROS_ASP3[1]を公開している．

ET ロボコンとは，指定の走行体を制御する組み込みアプリケーションの開発技術を競うロボットコンテストである．ここではもとより RasPike[2]と呼ばれる環境が使用されており，走行体（HackSpi）には SPIKE Prime を使用している．この環境では SPIKE Prime Hub（Hub）と RaspberryPi の 2 台のコンピュータ間でシリアル（UART）通信をして動作するものとなっていて，通信には独自の機構を用いている．Hub と RaspberryPi の画像を図 1 に，HackSpi の画像を図 2 に示す．

図 1．Hub(左)と RaspberryPi（右）．両者は UART で接続されている．

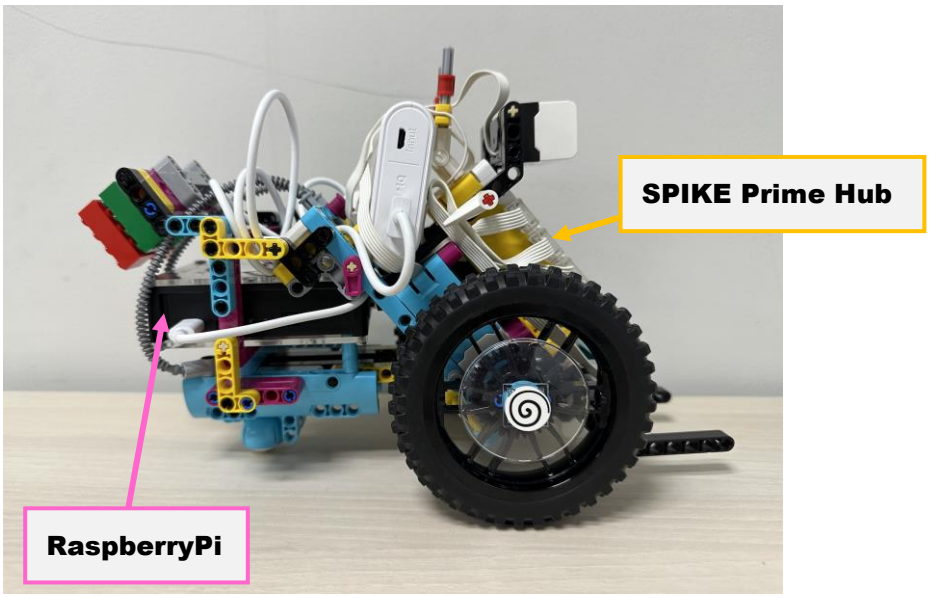

図 2. HackSpi 走行体のイメージ

この，ET ロボコンの環境に SPIKE-RT と ROS を取り入れる事により，応答性の向上や通信の汎用性の向上が期待できることに加え，ET ロボコンの参加者に SPIKE-RT や ROS のプログラミングに触れるきっかけを提供する事ができると考えられる．

また，このことは ET ロボコンの参加者に限らず，ROS のプログラミングを学習したいと考える人に教材を提供するとともに，その人が TOPPERS プロジェクトを知るきっかけになるということも考えられるであろう．

そこで，ROS と SPIKE-RT を用いた，ET ロボコン用走行体の制御アプリケーションを開発するためのソフトウェアプラットフォームを開発した．

2. 作品の概要・構成

今回作成したプラットフォームのソースコードは GitHub[3]で公開している．このプラットフォームは主に以下の 4 つのパッケージから成る．

- raspike_uros_msg
- uros_raspike_rt
- ros2_raspike_rt
- linetrace_sample

それぞれのパッケージ内容の詳細は GitHub 上のドキュメント[4]に記載している．

制御アプリケーションの開発方法として，ROS2 の API や ET ロボコン向けのカスタムメッセージを直接扱って開発を行う方法と，制御アプリケーション開発用の専用 API[5]を介して開発を行う方法の二種類を用意している．前者では ROS2 の API を扱うため ROS2 プログラミングの学習に使用することができ，後者では ROS2 の API を使わずにアプリケーションの開発が可能である．

オリジナルの RasPike の内部構成を図 3 に，今回作成した，ROS 使用のアプリケーション開発用のソフトウェアプラットフォームの内部構成を図 4 に示す．

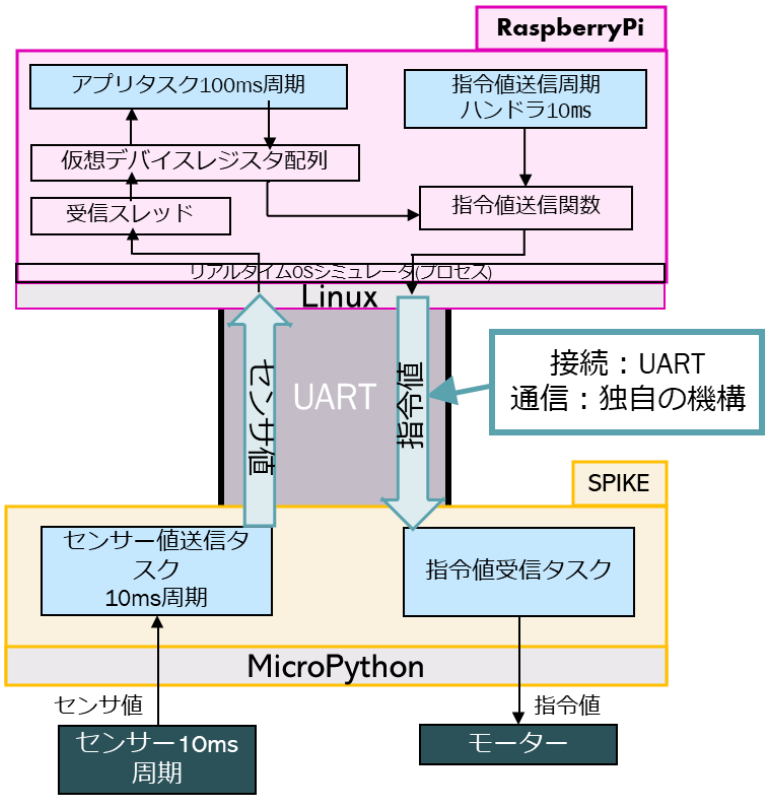

図 3.　RasPike の内部構成

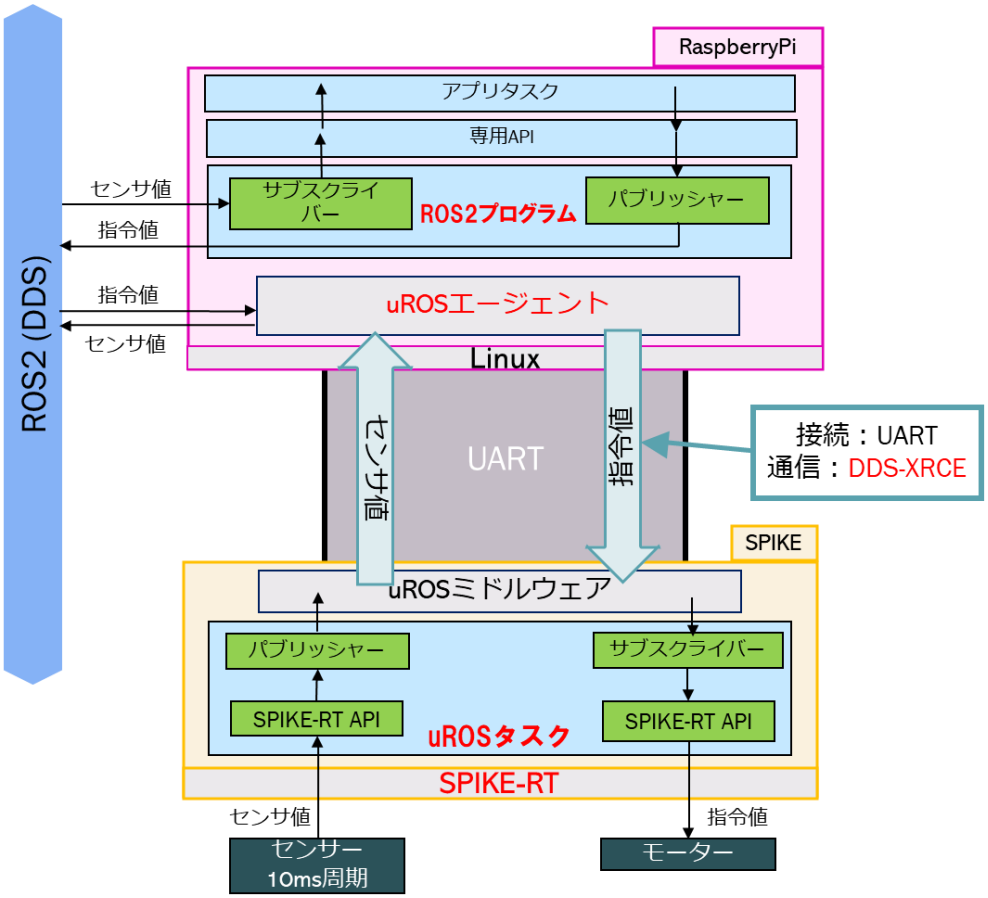

図 4. SPIKE-RT と ROS を使用した，アプリ開発用のソフトウェアプラットフォームの内部構成（この図はアプリ開発専用の API を使用した場合の内部構成図である）

これらの図における，アプリタスクの開発をユーザーが行う．図から見て取れるように，本作品は，SPIKE 側の MicroPython を RTOS である SPIKE-RT に置き換えている．SPIKE-RT の上では先程紹介した uROS ミドルウェアである micro-ROS_ASP3 が動いており，アプリタスクと uROS タスクとの，ROS2 通信を実現している．故に，ROS2 を用いた制御アプリケーションの開発が可能であり，アプリ開発の際に既存の ROS2 のソフトウェア資産を再利用することもできる．

この環境のメリットの一つとして，図 5 に示すようにアプリケーションの開発を外部の PC で行うことが可能である点が挙げられる． これは，ROS2 通信を用いているため，必ずしも RaspberryPi 上でアプリケーションを開発する必要がなく，ROS2 の環境を持つ PC であれば，走行体の外部からのアプリケーション開発が可能であるということである．これにより，任意の PC や走行体を任意の台数だけ接続することができるので，チームで開発を行う場合をはじめ，開発効率を向上させることができると考えられる．

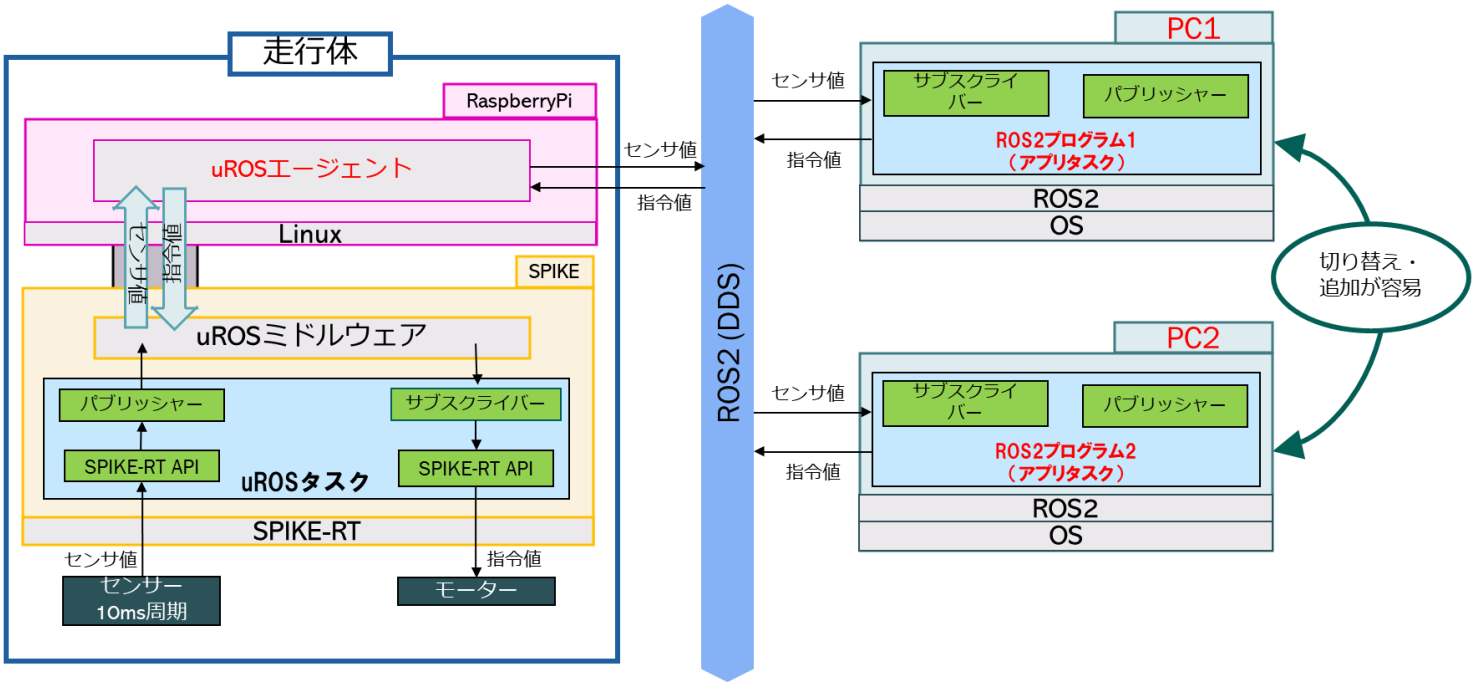

図 5．外部の PC を用いた開発

また，ROS を用いていることにより，rosbag を用いた効率的な開発が可能であるという点もメリットとして挙げることができるであろう．

加えて，このようにローカルの PC でアプリタスクの開発が可能である点は，ROS2 プログラミングの教材として用いる場合においても有益であると考えられる．

3. 本作品の開発に用いた環境

1) マイコン：SPIKE Prime Hub
2) SPIKE 側 OS：SPIKE-RT
3) RaspberryPi 側 OS：RaspberryPi OS 64bit
4) ROS2 バージョン：Humble
5) ROS2 パッケージの開発時に用いた環境：Ubuntu 22.04 LTS

4. 機能・性能評価

今回作成をしたソフトウェアプラットフォームと RasPike 環境との機能・性能評価の比較結果を下記の表 1 に示す.

表 1．性能評価

評価項目	RasPike	本作品
通信の応答性（処理速度）	× ✓ MicroPython を使用しており，リアルタイム性の保証ができない	○ ✓ RTOS を使用しており応答性が向上
通信の汎用性	× ✓ 独自の通信機構を用いているため汎用性が低い	○ ✓ ROS2 通信を使用しているため汎用性に優れる ✓ ROS のソフトウェア資産の再利用が可能
開発効率	△ ✓ RaspberryPi 上でのみアプリ開発が可能	○ ✓ ローカル PC でのアプリ開発も可能 ✓ rosbag が利用可能 ✓ ROS 通信を用いて Rviz 等のシミュレーター接続することが可能（実施予定）
利用可能なモジュール	△ ✓ スピーカーモジュールが利用不可	○ ✓ スピーカーモジュールが利用可能

5. 動作環境・使用方法

動作環境はドキュメント[4]の「動作環境」に記載している．

以下に本作品の使用方法を記す．

① SPIKE Prime の入手

[6](ET ロボコン SPIKE キット)から購入できる．

② 走行体を組み立てる

- SPIKE Prime で HackSpi を組み立てる．HackSpi の組立図は ET ロボコン参加者のみの公開であったため，ここでの紹介は控える．ET ロボコン走行体の組立図を入手できない場合は以下の条件を満たす走行体を作成する．
  - Hub と RaspberryPi を搭載している．Hub と RaspberryPi はシリアルで接続する．
  - RaspberryPi への給電はモバイルバッテリー等で行う．
  - Powered Up（PUP）デバイスのうち，モーターが 3 つ（左右の車輪用とアーム用）と，カラーセンサー，超音波センサーが取り付けられている．
  - Hub と SPIKE Prime の PUP デバイスの接続ポートがドキュメント[4]に記すポートと等しい．

③ 開発環境の構築

- ドキュメント[4]に従って開発環境を構築する．
  - 「使用方法(uROS プログラムに変更が必要無い場合)」を参照

④ RaspberryPi のターミナルから ROS2 プログラムを起動する．起動方法は[4]を参照する．

6. 実行の様子

今年度は，私が所属する南山大学理工学部 本田研究室では ET ロボコンに参加している．我々のチームでは，今回作成した SPIKE-RT と ROS を用いたソフトウェアプラットフォームを使用して参加する予定であり，実際にこのプラットフォームを使用してアプリケーションの開発を行っている．

下記のリンクに公開している動画は今回作成したソフトウェアプラットフォームを使用して作成した，PID 制御によるライントレースのサンプルプログラムの実行の様子である．

実行の様子(Youtube)：https://www.youtube.com/watch?v=RoaVhumuqcQ

7. 今後の展望

現在の仕様では, Hub に接続する PUP デバイスの種類や接続先のポートが固定化されたものを想定している．そのため，もしセンサーやアクチュエーターの構成を変更して ROS2 のプログラミングを行いたい場合, ユーザーが ROS のカスタムメッセージの構成の変更や，Hub 側の uROS ファームウェアの更新を行う必要がある．これは手間のかかる作業であり，ROS プログラミングの初学者にとっては大きな障壁になり得るであろう．

そこで今後は，ユーザーがセンサーやアクチュエーターの構成をコンフィギュレーションファイルに 記入するだけで，必要な ROS のメッセージをまとめたカスタムメッセージや，uROS 側のファームウェアを自動生成するようなツールを作成する予定である．これにより，より自由度の高い ROS2 プログラミングの教材を提供できるとともに，TOPPERS プロジェクトを知ってもらう機会の拡大に繋げることができると考えている．

8. 参考文献

[1] micro-ROS_ASP3 の GutHub ページ,
https://github.com/exshonda/micro-ROS_ASP3

[2] RasPike の GitHub ページ，https://github.com/ETrobocon/RasPike

[3] ROS 通信による ET ロボコン走行体の制御アプリケーション開発用プラットフォームの GitHub ページ，https://github.com/Hiyama1026/uros_raspike-rt

[4] 本作品の詳細情報をまとめたドキュメント（GitHub），
https://github.com/Hiyama1026/uros_raspike-rt/blob/master/README.md

[5] 専用 API リファレンス（GitHub），
https://github.com/Hiyama1026/uros_raspike-rt/blob/master/ros2_raspike_rt/API_REFERENCE.md

[6] Afrel，ET ロボコン出場者向けセット，https://afrel.co.jp/product/et-set/