

知能制御PBL

第2回 RCR 中間報告

2018年6月6日

西田研究室

13104042 烏谷崇大
14104055 佐々木秀将
15104021 長田駿二郎
15104026 川崎雄太朗
15104050 坂元勇太
15104081 徳野将士
15104113 前田修一
15104117 右田明花
15104134 山福佳
17104311 横田篤紀

目 次

1	目的	2
2	RCR (Robot Car Race) 2018	2
2.1	競技概要	2
2.2	コース	2
2.3	競技ルール	2
3	ROS (Robot Operating System)	3
3.1	ROS とは	3
3.2	ROS ノードとトピックの構成	4
4	機体	5

1 目的

学部3年までに学習した制御理論や電気回路・情報工学の知識を使って、競技場内を自律的に走行するロボットの製作を行う。研究室で一丸となってプロジェクトを行なうことで課題を達成することの難しさや楽しさを学び、エンジニアとして仕事を進めるための素養を身に付ける。

2 RCR (Robot Car Race) 2018

2.1 競技概要

ウレタンパネルを用いてレイアウトされる周回コースにおいて、格子模様のコントロールライン手前から走行開始してコントロールラインを3回通過後に停止するまでの時間を競うロボットカー（ロボカー）を製作する。

2.2 コース

一边50 [cm] の正方形及び扇形の黒色ウレタンパネルと白黒格子模様のコントロールライン付ウレタンパネルを組み合わせ、コースを構成する。なお、競技会当日までコースは公表されない。また、コースフェンスの高さが低いため、コースフェンスのコース側に高さ10 [cm] の壁を設置する。

2.3 競技ルール

- (1) コントロールライン手前からの走行開始からコントロールラインを3回通過後に停止するまでの時間を競う。
- (2) 各チームあたり10分以内に最大3回走行し、最短時間の走行を評価する。
- (3) コース2周以上走行すること。
- (4) 競技会当日のコース試走は認めない。
- (5) ロボカーがコース周囲の壁に接触した場合は失格とする。
- (6) コース及び壁に物を設置したり、手を加えてはいけない。
- (7) コース内に足を踏み入れないこと。

3 ROS (Robot Operating System)

3.1 ROS とは

ROS(Robot Operating System) とは Open Source Robotics Foundation によって管理されているソフトウェア開発者のロボット・アプリケーション作成を支援するフレームワークである。具体的には、ハードウェア抽象化、デバイスドライバ、ライブラリ、視覚化ツール、メッセージ通信、パッケージ管理などが提供されている。つまり ROS は汎用コンピュータ向けの OS ではなく、汎用コンピュータ向け OS 上で動作するメタ OS として捉えることができる [1]。

図 1 に示すように ROS ではプロセス(実行プログラム)はノードという単位で扱い、ノード間の通信はトピックと呼ばれる“Publisher/Subscriber”モデルで実現される [2]。

これにより、プログラミング言語や通信相手さえ意識することなく簡単にプロセス間通信を実現できる。これは各ノード間のインターフェース、即ちトピックの名前と型さえ決定すればノードごとに独立して開発を行うことができるという利点もある。

以上の利点を鑑み、本研究室では ROS がインストール可能なマイコンボードである RaspberryPi3 Model B 上に ROS をインストールして開発を進めていくこととした。

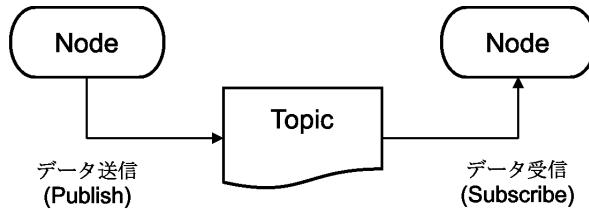


図 1: ROS ノードとトピックの概念

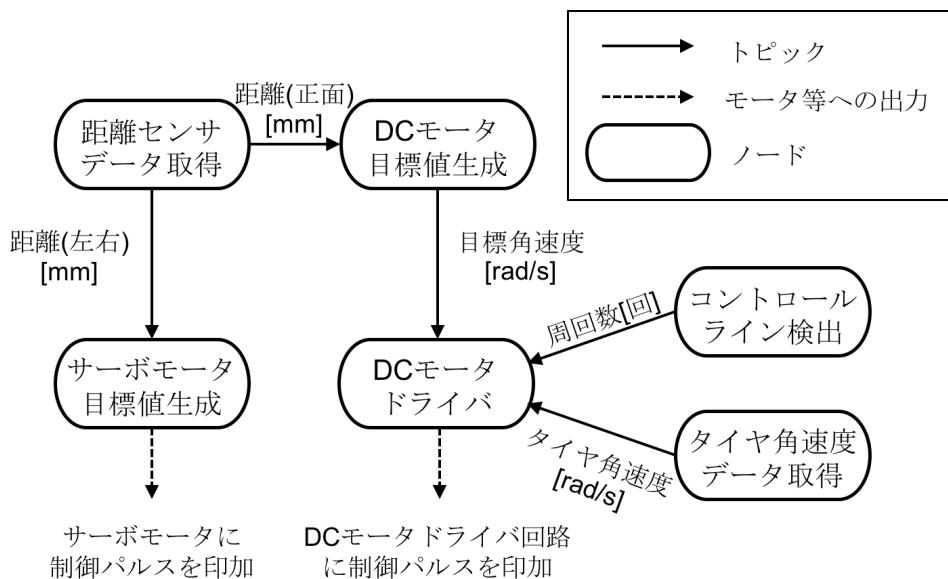


図 2: ROS ノードとトピックの構成

3.2 ROS ノードとトピックの構成

図 2 に開発する ROS ノードとトピックの構成を示す。各ノードの役割は次の通りである。

距離センサデータ取得

機体前方及び両側面に設置した距離センサからシリアルバス規格の一つである I²C を介して距離データを [mm] 単位で取得し外れ値処理や正規化を施した後に Publish する。

コントロールライン検出

機体後方下部に設置したフォトリフレクタによってコントロールラインを通過した回数をカウントし Publish する。

タイヤ角速度データ取得

回転方向は考慮しないため、ロータリーエンコーダの A 相のパルスのみをカウントし、ロータリーエンコーダの一周期あたりの出力パルス数 (500 パルス)、サンプリング周期 (0.01 [s])、ロータリーエンコーダの軸に取り付けたピニオンギアとドライブシャフトに取り付けたギア間のギア比 (2.74) を考慮して k 時点のタイヤの角速度 $\omega(k)$ を [rad/s] 単位で算出する。また、ノイズ対策として算出した角速度を式 (3.1) で表されるデジタルフィルタ (LPF に相当) に通した結果を Publish するようにしている。ただし、 α は 0 ~ 1 の範囲で定める定数である。

$$\omega(k) = \alpha\omega(k) + (1 - \alpha)\omega(k - 1) \quad (3.1)$$

DC モータ目標値生成

機体前方方向の距離データを Subscribe し、それをもとに DC モータに与える目標値を生成し Publish する。

DC モータドライバ

DC モータに与える目標値とタイヤの角速度データを Subscribe し、PI 制御則に基づき DC モータを駆動する。

サーボモータ目標値生成

機体前方の距離データを Subscribe し、それをもとにサーボモータに与える目標値を生成し、その後直接サーボモータを駆動する。サーボモータドライバノードが存在しないのはサーボモータの軸にロータリーエンコーダがついておらず角度フィードバックが出来ないためである。

4 機体

図3に配布されたロボカー本体の写真をしめす。図??より本体の前方には前輪を駆動させ機体を旋回させるためのサーボモータを設置している。そして、本体の後方には後輪を駆動させ機体を動かすためのDCモータを設置し、その上にロータリエンコーダを設置した。

次に、図4に本体の上に設置するパーツの写真を、図5にその上に設置するパーツの写真示す。図4より中央にはDCモータを制御するためのモータドライバ、後方には各部品に電力を供給するためのバッテリ、前方にはRaspberry Pi3 model Bに電力を供給するためのモバイルバッテリを設置した。そして、前方と左右にコースの壁の距離を計測するためのPSDセンサを設置した。図5より、一番上にはRaspberry Pi3 model BとDC-DCコンバータを含めた電気回路を設置している。

そして、現在、配線が図4、図5よりロボカー本体に他の部品を設置するためにユニバーサルプレートを採用した。その理由は最初に、加工しやすいからである。ユニバーサルプレートは素材がプラスチックで出来ているので切断しやすいという特徴がある。次に、様々な部品の取り外しや設置がやりやすいからである。ユニバーサルプレートには無数の穴が開いているので簡単にボルトとナットに部品を設置できる。また、プラスチックは電気を通しにくいので電気回路の絶縁にもなるからである。

最後に、図6にロボカーの背面の写真を示す。図6より、現在はまだ設置していないがロボカーの背面にゴールラインを読み取るためのフォトリフレクタを設置する予定である。

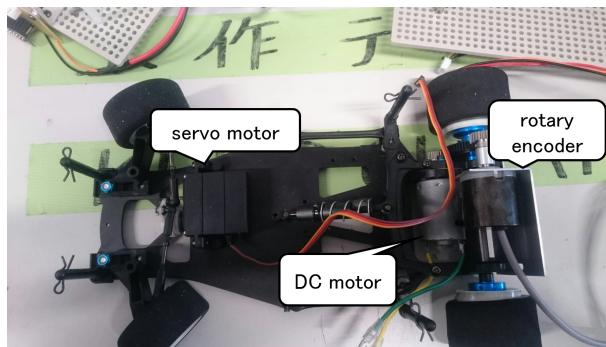


図3: ロボカー本体

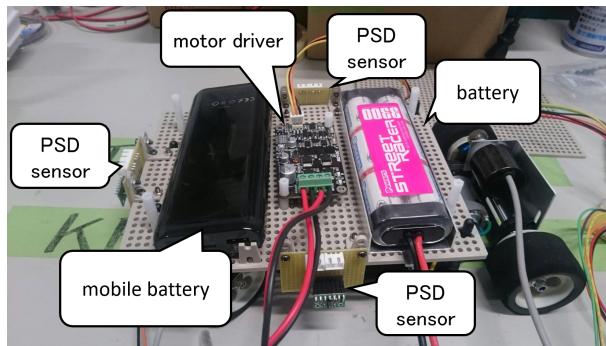


図4: ロボカー 1段目

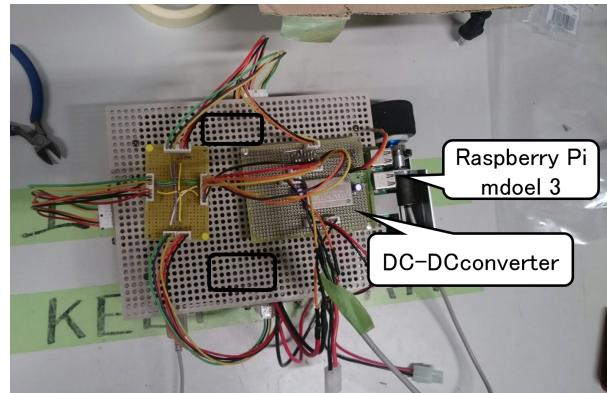


図 5: 口ボカ－ 2段目

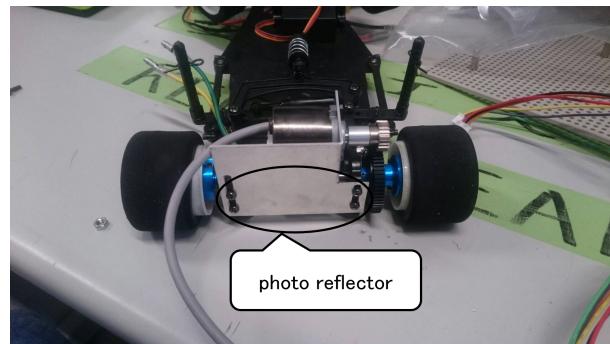


図 6: 口ボカ－背面

参考文献

- [1] 表允, 倉爪亮, 渡邊裕太, ”詳説 ROS ロボットプログラミング-導入から SLAM・Gazebo・MoveIt まで-”, Kurazume Laboratory, pp.15-18, (2015).
- [2] 小倉崇, ”ROS ではじめるロボットプログラミング”, 工学社, pp.8-10, (2015).