

知能制御PBL

第三回 RCR 中間報告

2017年6月28日

西田研究室

13104042 烏谷崇大
14104043 桑野僚大
14104034 下松八重宏太
14104090 中尾真人
14104111 本田空
14104131 山崎達也
16104313 山下翔

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カーの製作を行う。各研究室でチーム一丸となってプロジェクトを行なう、共同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。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カーの製作を行う。各研究室でチーム一丸となってプロジェクトを行なう、共同で課題を達成することの難しさや楽しさを学び、

2 Robot Car Race(RCR)2017 競技ルール

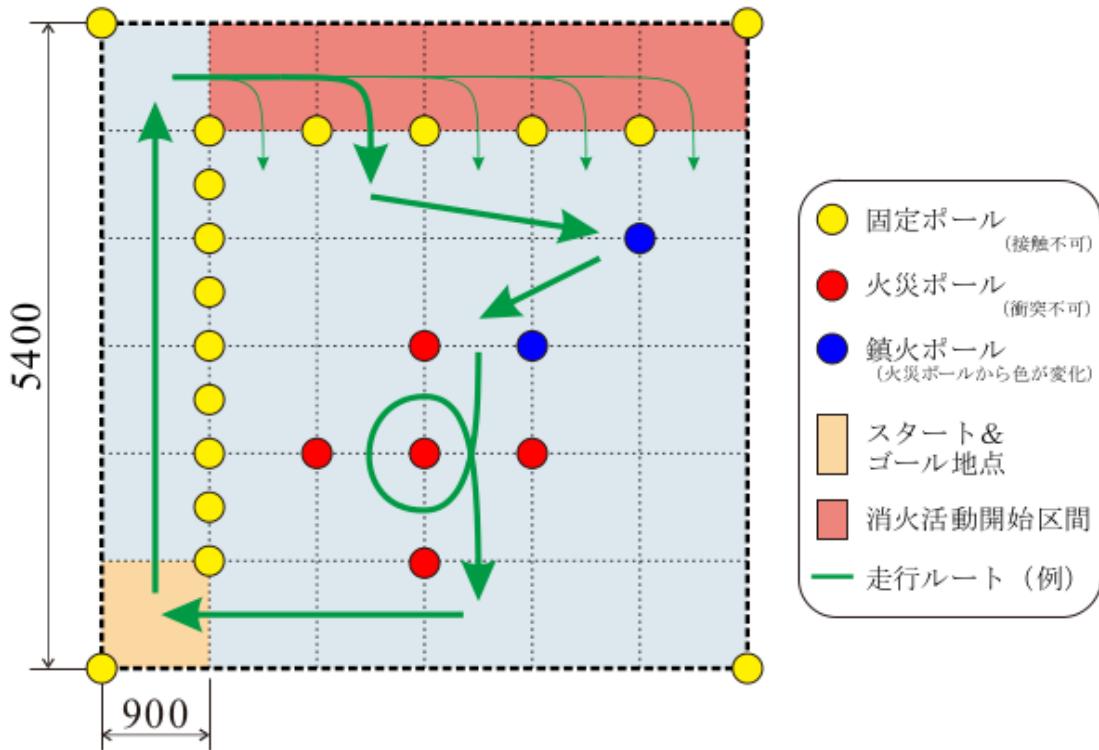
2.1 ルール概要

競技場には黄色のポールや、火災に見立てた複数の赤色のポールが設置されている。ポールに接触せず、できるだけ速やかに火災を鎮火させる消防ロボットカー(ロボカー)を作成する。

2.2 競技場詳細

競技場の全体図を図1に示し、以下に詳細を説明する。

- (1) 競技場は板張りの床であり、縦・横ともに 5400 [mm] である。
- (2) 競技場には黄色の固定ポールと赤色の火災ポールが設置されており、スタートからゴールまで、固定ポールには接触、火災ポールには衝突することなく通過しなければならない。
- (3) 火災ポールは青色の鎮火ポールに赤色の幕を被せたものであり、上部におもりなどを落としたり、幕を剥がしたりすることで、鎮火ポールに変化させる(このポールの製作も行うこと)。
- (4) スタート後は右手に固定ポールを見ながら直進し、消火活動開始区間まで移動しなければならない。消火活動開始区間に進入後は、右折し、火災ポールを発見し次第、消火にあたる。
- (5) すべての火災ポールを消火して、鎮火ポールに変化させたのち、ゴール地点で停止する。
- (6) 火災ポールの配置は競技ごとに異なる。また、鎮火ポールが存在することもある。
- (7) ポールは直径 80 [mm]・高さ 120 [mm] の中空パイプであり、黄・赤・青の色が付けられている。



3 ロボットカーの概要

本年度の RCR での我々の設計コンセプトは、ロボットの安定した運用を可能にすることである、そのために、ロボットカーには比較的構造が単純な独立二輪機構を採用する。火災・鎮火ポールの判別は、ロボット前方に取り付けた単眼カメラの映像を画像処理することで行う、また、PSD(Position Sensitive Detector)センサと赤外線近接センサを用いてポールとロボットの距離を測定することで、適切な軌道の走行とポールの消火を目指す。製作にあたっては、研究室のメンバーをハードウェア、ソフトウェアの担当に分けて進めることとする。

4 消火について

4.1 消火ポール

我々が用いる消火ポールは、以下のように製作する。
まず、塩化ビニル管に青い布を巻きつけて青いポールを作成する。次に、青いポールの上から赤い布を覆うことで赤いポールとする。

4.2 消火方法

用いるポールの構造より、赤い布を青いポールから取り除くことで消火とする。布を取り除く方法は、ロボットアームがポールの上から赤い布を中に押し込んで、青いポールにするというものである。

4.3 消火用機構

消火方法は前述の通りであるが、ここではそのための機構であるアーム機構について述べる。アームは垂直方向に上下する1リンク機構を考える。これは、消火方法が布を押し込むという単純なものであるためである。

アームの動作の概念図・並びに実際に製作したものを図2、図3に示す。概念図のように、モータとリンクを糸状のもので接続し、モータの巻取りによって上下運動を実現する。

実際の製作では、モータを3の左のようにアームと一緒に化させることによって簡略化を図った。また、巻取りのリミットの判定として、アーム上部にマイクロスイッチを設置した。

使用したモータは「TAMIYA ミニモータ ユニバーサルギアボックス」であり、ギア比は269:1とした。また、回転方向の制御のためにモータドライバ「TA7291P」を使用した。

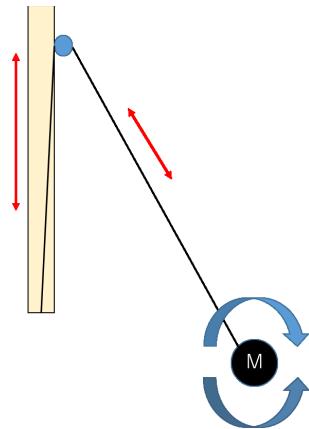
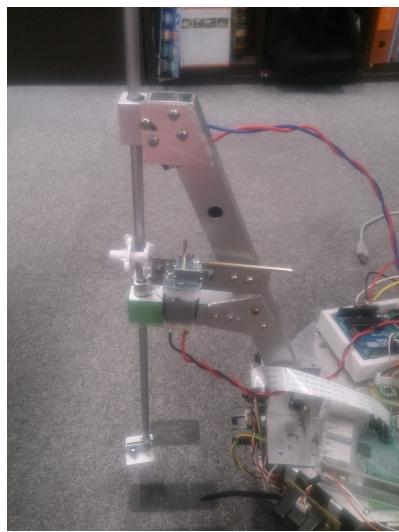
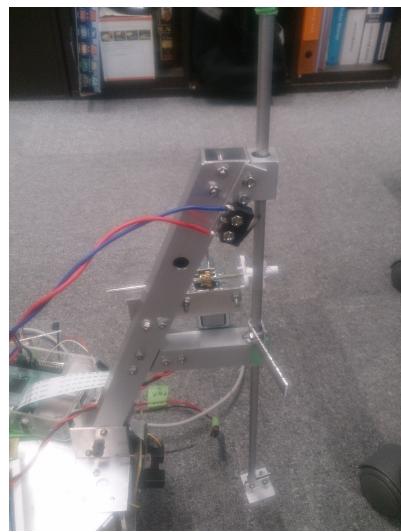


図2: 消火用アーム概念図



[1] アーム 左



[2] アーム 右

図3: 消火用アーム

5 PSD センサの同定実験

前年度までに研究室で購入していた 2 種類の PSD センサについて、その PSD センサの精度を確かめるために同定実験を行った。

5.1 センサの仕様

前年度までに購入していた 2 種類の PSD センサの仕様を以下に示す。また、下記 2 つのセンサを便宜上、順に近距離センサ、遠距離センサと呼ぶこととする。

【シャープ測距モジュール GP2Y0A21YK】

測距範囲：10 ~ 80 [cm]

出力：アナログ電圧出力

寸法：29.5 × 13 × 13.5 [mm]

電源：4.5 ~ 5.5 [V]

【シャープ測距モジュール GP2Y0A02YK】

測距範囲：20 ~ 150 [cm]

出力：アナログ電圧出力

寸法：29.5 × 13 × 21.6 [mm]

電源：4.5 ~ 5.5 [V]

5.2 実験装置

PSD センサの実験を行うため、図 4 のような実験装置を製作した。PSD センサは高さ 20 [mm] の位置にセンサの発光部が左、受光部が右になるように箱に水平に装着した。PSD センサを動作せるには Arduino Uno を用い、Arduino IDE のシリアルモニタを用いて出力電圧を測定した。実験時のセンサと Arduino の配線を図 5 に、Arduino のプログラムを付録 1 に示す。配線にはブレッドボードを用いた。また、ポールの代わりに、ポールと同じ形状のスプレー缶を用いた。

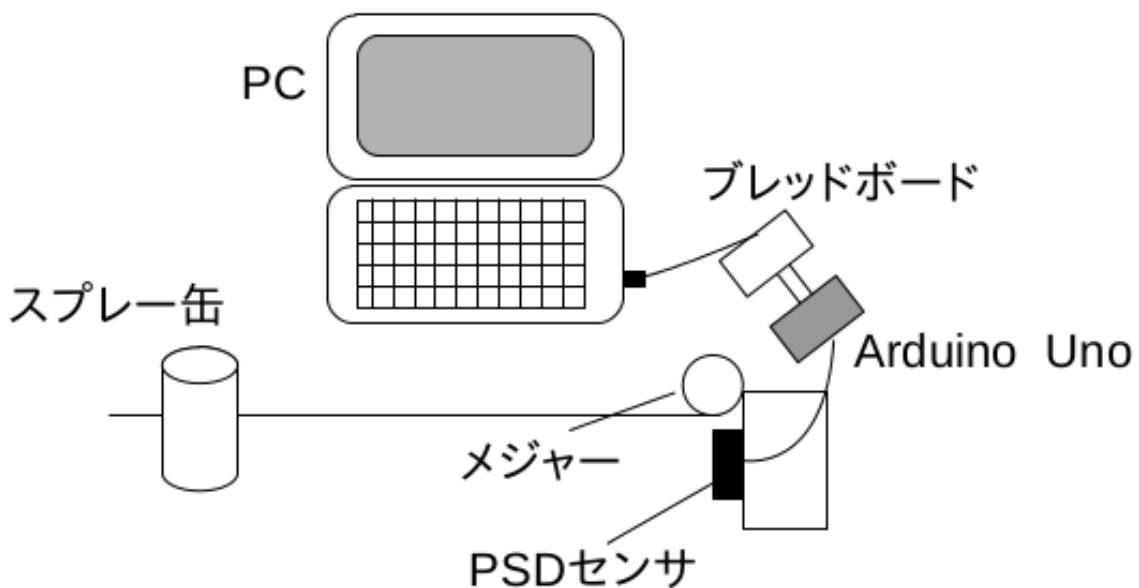


図 4: PSD 実験装置

5.3 実験方法

PSD センサの距離-出力電圧特性を測定するため以下の手順に従い実験を行った .

- (1) PSD センサの発光部・受光部の先端を距離 0 [cm] とし , 近距離センサは 5 [cm] から 100 [cm] まで , 5 [cm] ずつスプレー缶を移動させ出力電圧を記録する . このとき , スプレー缶の中心は PSD センサの中心の正面にくるように置き測定する .
- (2) 先程と同様に , 遠距離センサは 5 [cm] から 170 [cm] まで , 5 [cm] ずつスプレー缶を移動させ出力電圧を記録する .

5.4 実験結果

縦軸を出力電圧 , 横軸を PSD センサ-スプレー缶間の距離とし , 近距離センサの測定結果のグラフを図 6 に , 遠距離センサの測定結果のグラフを図 7 に示す . 図 6 より , 近距離センサは出力電圧が 40 [cm] までは滑らかに減少しており , 40 [cm] からは大きな変化は見られない . それに対して図 7 より , 遠距離センサは測距可能範囲内において出力電圧が 80 [cm] までは滑らかに減少しており , 80 [cm] からは変化に乏しいことがわかる . よって , 近距離センサでは 40 [cm] 以降 , 遠距離センサでは 80 [cm] 以降の距離を算出することが難しくなると考えられる . ここで , 今回のロボットは 40 [cm] 以降も距離を計測する必要がある . 従つて適当なセンサは , 遠距離センサであると考えられる .

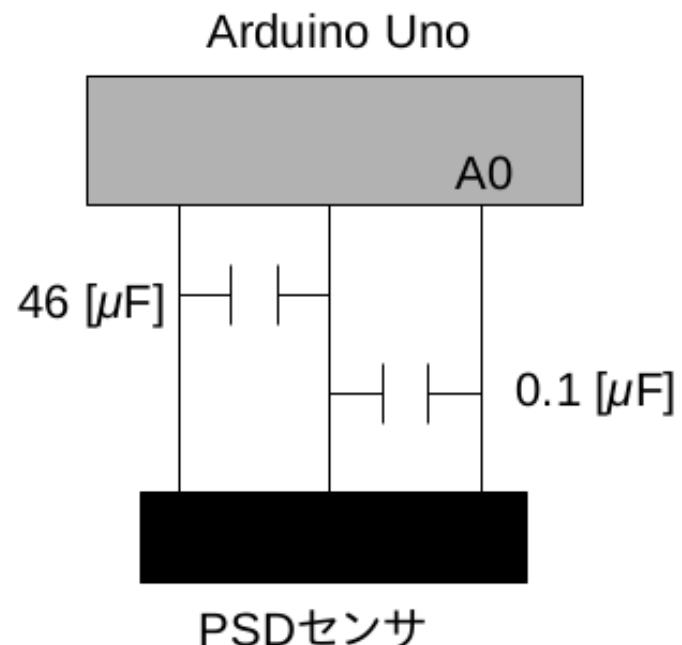


図 5: PSD センサの配線図

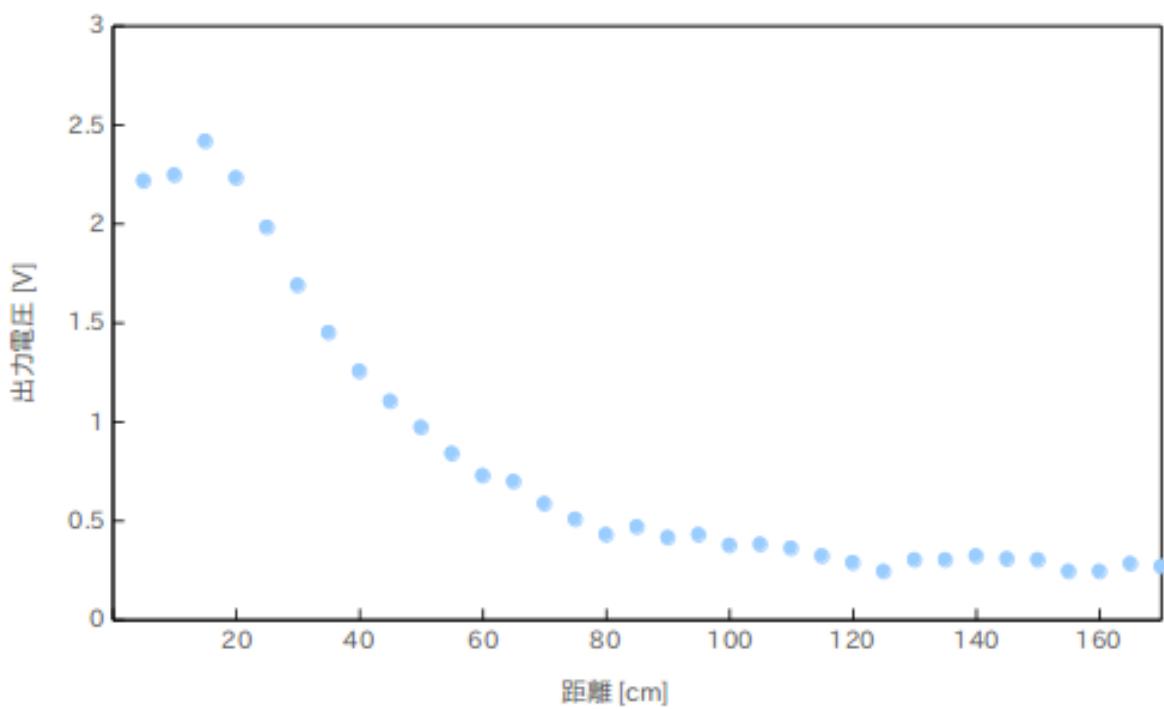


図 7: 遠距離センサ

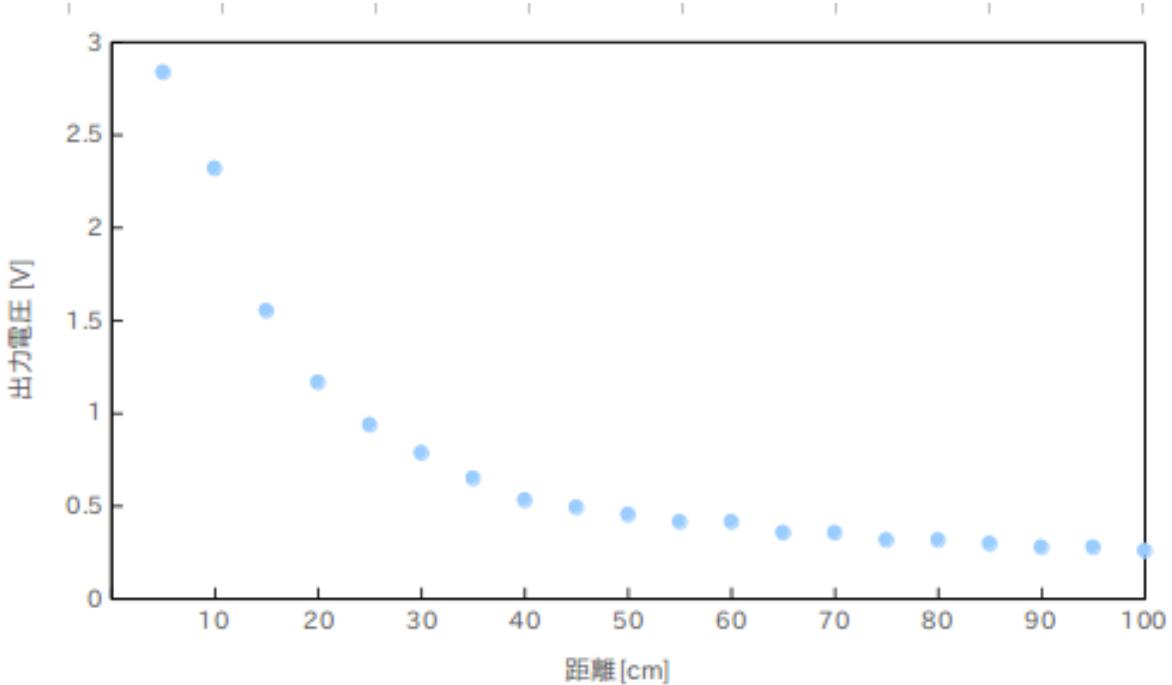


図 6: 近距離センサ

6 機体の再設計

6.1 前回の反省

第1回のレポートで紹介した機体を図8に示す。これを一号機とし、一号機での反省点を元に再設計した機体を二号機とした。二号機の概要を図9に示す。一号機での反省点を以下に述べる。

- 車体に無駄なスペースが多い。
- 車体の幅が大きい。
- 回路のガードがされていない。

西田研究室では、これらの反省点を改善するために、機体の大幅な見直しを行った。改善した点について以下に述べていく。

6.2 モータとエンコーダの配置

一号機の駆動系を図10に示す。一号機では、モータとエンコーダの配置を線対称としていたため、機体の幅が大きくなればならなかった。そこで、二号機ではこのモータとエンコーダの配置を点対象とすることにした。これにより、タイヤのトレッド幅を限りなく小さくでき、機体の幅を小さくすることができた。また、一号機ではモータとエンコーダをアルミ板で下から支えるようになっていた。そのため、モータとエンコーダを支える以外のスペースは無駄なスペースとなっていた。二号機ではこの無駄なスペースを省くために、モータとエンコーダを上から支えることにした。上から支えるにしたがって、モータとエンコーダの重量と機体の荷重が図11に示した部分に集中する恐れがあるため、機体中心部分にモータおよび

エンコーダを支えるパーツを 3D プリンタで作成し、設置した。以上より、下から支えていたアルミ板を廃止することができ、一号機での無駄なスペースを削減することができた。

6.3 回路のガードについて

機体の大部分はアルミでできているため、回路をそのままにしておくと、ショートする可能性がある。したがって、回路をホットボンドで覆うことでショートすることを防ぐようにした。Arduino に関しては専用のケースが存在しないため、3D プリンタでケースを作成した。

6.4 タイヤについて

一号機では図 12 のようなタイヤを使用していたが、このタイヤは中身がスポンジになっており、機体の重量でタイヤの接地面積が増えやすく、また機体の荷重移動によって機体がロールする可能性があると考えた。二号機ではそういう問題を解消するべく、ナロータイヤを使用することにした。使用するナロータイヤを図 13 に示す。使用的なナロータイヤは非常に細いため、地面との接地面積を少なくすることができる。また、ゴムのみでできているので、一号機で使用する予定であったタイヤに比べて沈む量が少なく、機体の荷重移動によるロールも低減できると考えられる。しかし、接地面が小さくなつたことでスリップする可能性があると考えられるので、キャンバー角をつけるなどして対策を考えていく。

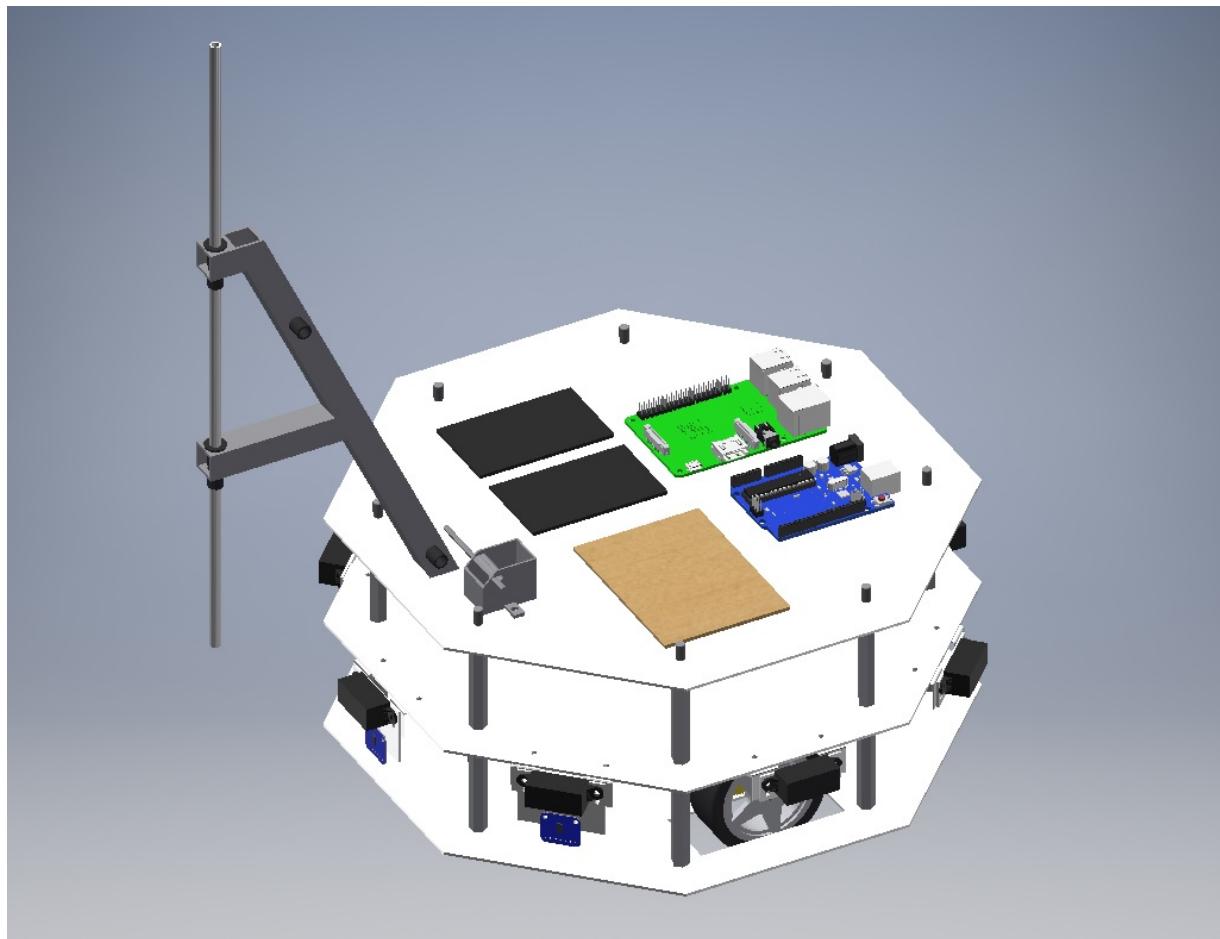


図 8: 一号機

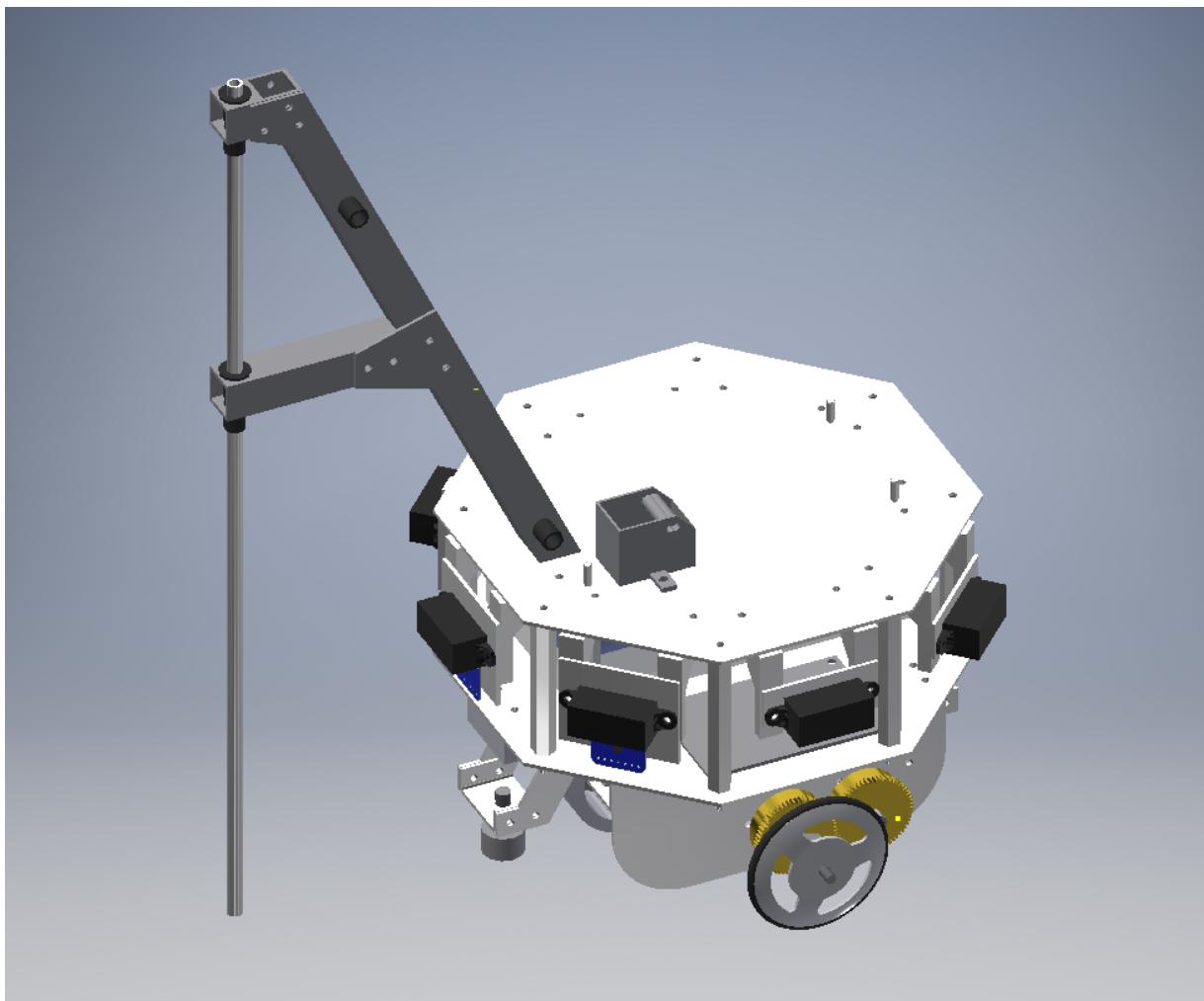


図 9: 二号機

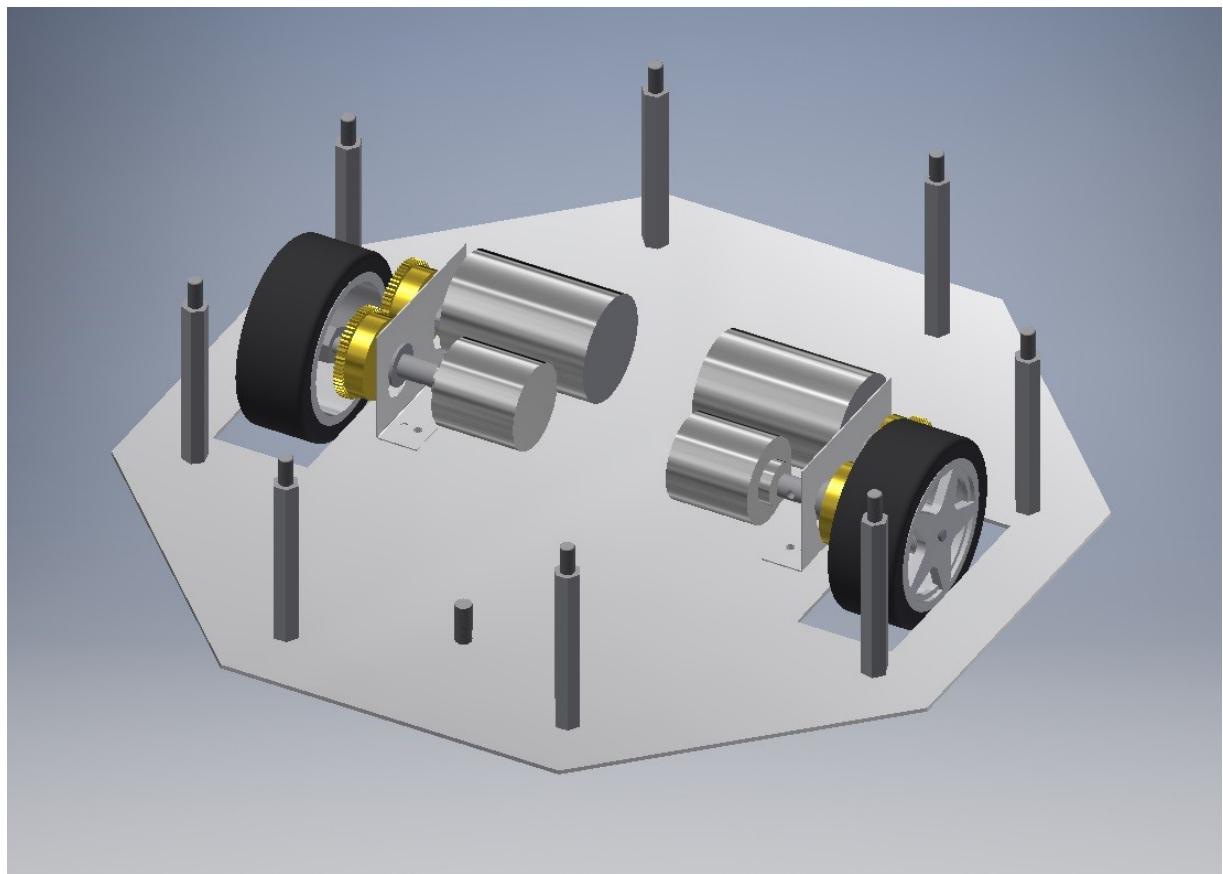


図 10: 一号機の駆動系

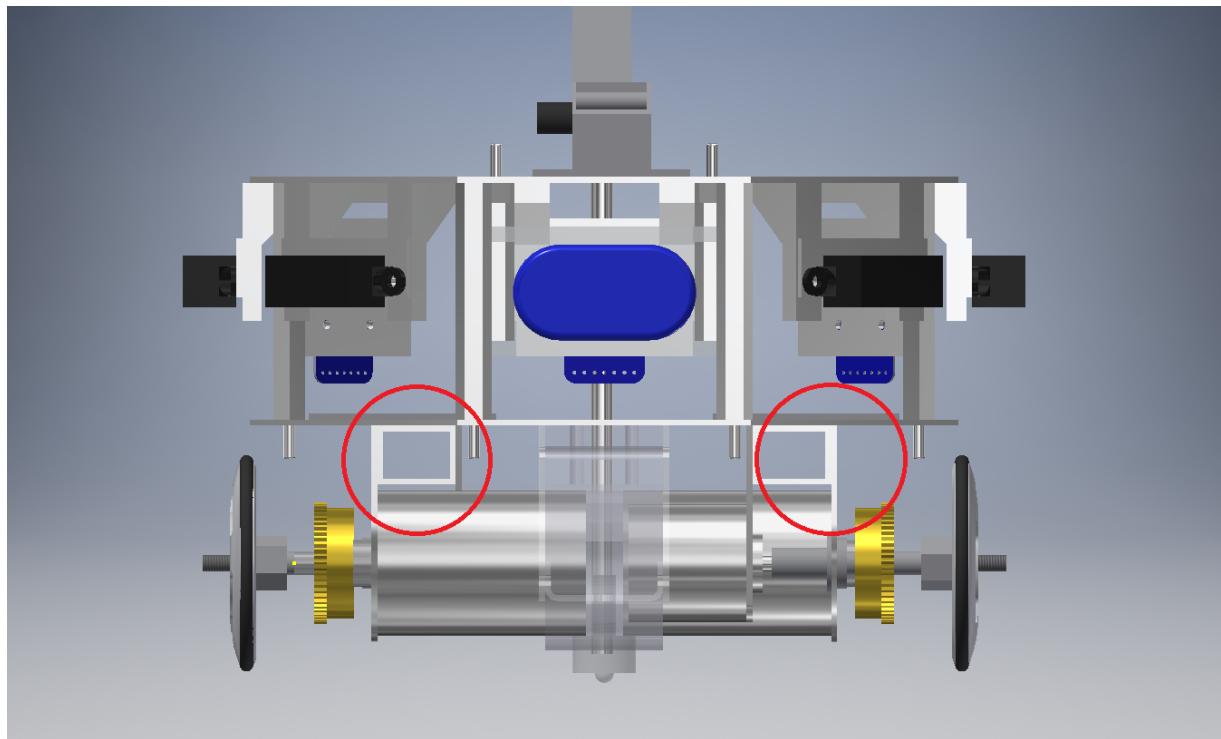


図 11: 最も荷重がかかる位置



図 12: 一号機のタイヤ

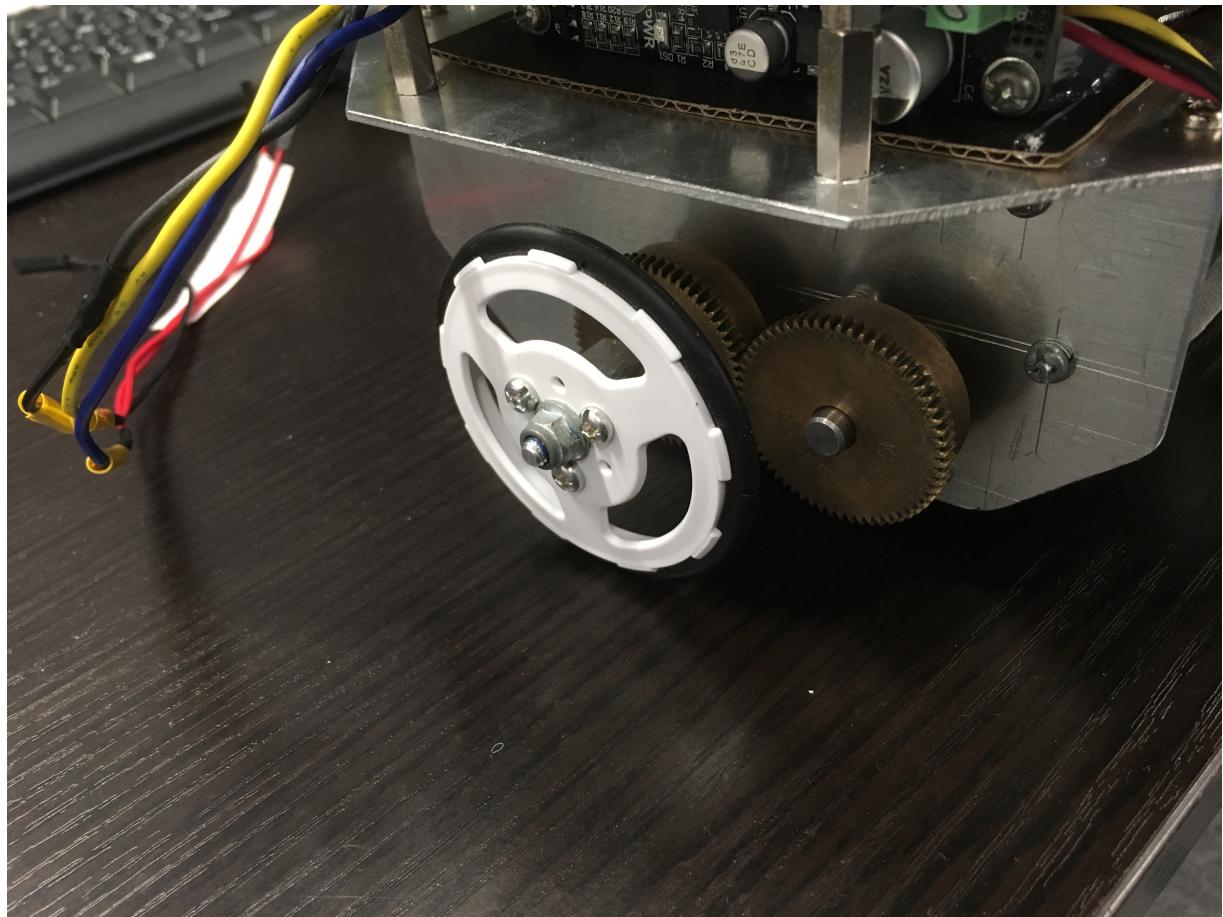


図 13: 二号機のナロータイヤ

6.5 第二回中間発表のデモ走行について

第二回中間発表でデモ走行を行ったが、途中で停止してしまった。本機体は四点で支えられており、路面の凹凸によって三点しか接地できなかった場合に機体が「やじろべえ」と同じような状態になってしまい、駆動力が路面に十分に伝わらず停止したと考えた。

6.5.1 実地調査

この問題を解決する前に、我々は RCR の会場（記念講堂）においても今回と同じような問題が生じるのかを確かめることにした。実地調査をするためには機体を自由に動かせる必要があるため、機体をワイヤレスコントローラ (DUALSHOCK3) を用いて動かせるようにした。記念講堂で動かしてみた結果、途中で機体が停止することはなかったため、機体に修正を加える必要はないと考えた。

7 回路設計

設計した回路について選定の理由や仕様について以下に示す。また使用する部品の一覧を表 7.1 に示す。

7.1 マイコンの選定

設計した回路を図14, 図15に示す。マイコンとして「Raspberry Pi3 Model B(以下RPi)」と「Arduino uno R3(以下Arduino)」を使用する。それぞれが、統合・画像処理・モータ制御、センサ処理、を行う。RPiでは複雑な処理を行う上で、LinuxOSの支援を受けることができ有利である。さらに、処理速度がCPU 1.2[GHz]、メモリ 1[GB]とArduinoの16[MHz]・32[KB]と比べても大きく優れている。これは、並列処理や高速な画像処理に適している。このような理由からRPiを採用した。

また、RPiはアナログI/Oポートを持っておらず、アナログ電圧出力を行うセンサ類の処理が困難である。そこで、アナログ・デジタルI/Oポートを持つArduinoにセンサ類の処理を担わせることとした。ただし、要求されるアナログI/Oポート数が後述の I^2C 通信を使用しても足りない。そこでArduino用16チャンネル・アナログ・マルチプレクサを使用することで増設を行った。

表 7.1: 回路用部品表

タイプ	部品名	数	用途
マイコン	Raspberry Pi3	1	統括・画像処理・モータ制御
	Arduino uno R3	1	センサ類の処理
DCモータ	AO-8014	2	駆動用
	TAMIYAミニモータ	1	アーム用
モータドライバ	MD10C-R3	2	タイヤ用
	TA7291P	1	アーム用
アナログマルチプレクサ	CD74HC4067	1	ArduinoアナログI/Oピン増設
I^2C 通信用変換モジュール	PCA9306	1	I^2C 通信
赤外線測距センサ	GP2Y0A02YK	6	中距離センサ
ToF近距離センサ	VL6180x	3	近接センサ
カメラモジュール	P5V04A	1	画像処理
3軸加速度センサ	KXR94-2050	1	自己位置推定
3軸ジャイロセンサ	BGD20	1	自己位置推定
DCDCコンバータ	LT8697	1	7.2[V] 5.0[V]2500[mA]降圧レギュレータ
	BTD05-05S200D	1	7.2[V] 5.0[V]2000[mA]降圧レギュレータ
コンデンサ	電解コンデンサ 47[μ F]	2	電源安定化
	セラミックコンデンサ 0.1 μ	9	センサ信号安定化
バッテリー	POWER MAX 4000 Ni-MH	1	電源バッテリー 7.2[V]4200[mAh]

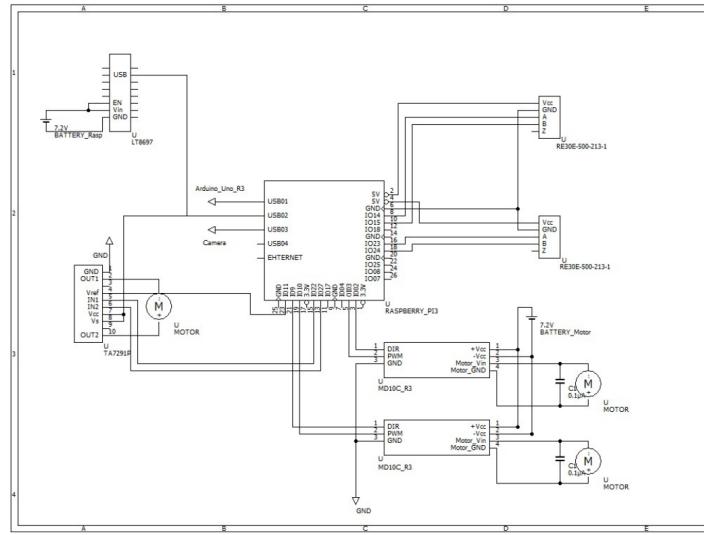


図 14: Raspberry Pi3 接続回路図

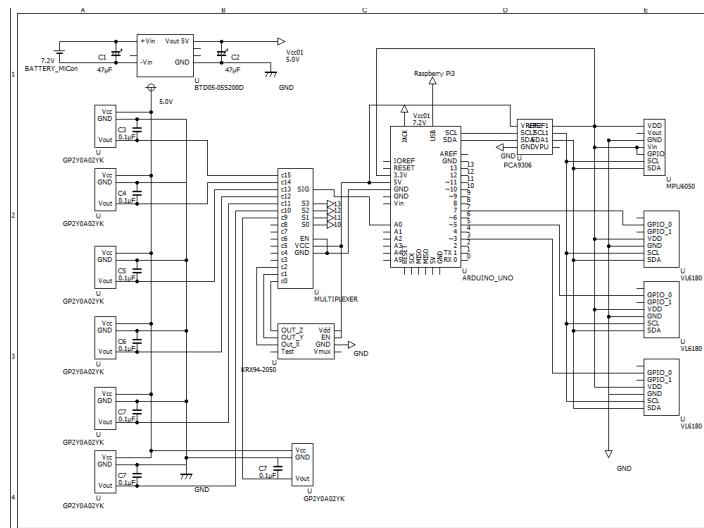


図 15: Arduino unoR3 接続回路図

7.2 モータドライバ

モータドライバは「MD10C R3」(図 16)を両輪駆動用として2つ使用し、「TA7291P」(図 17)をアーム用として使用する。各仕様を下に示す。

[MD10C R3](駆動用)

- モータ電源電圧 : DC 5[V] ~ 25[V]
- モータ最大電流 : 13[A]
- ロジック用電源 : モータ用より供給
- ロジック電圧 : DC 5[V] or 3.3[V]

[TA7291P](アーム用)

- モータ電源電圧 : DC 0[V] ~ 20[V]
- モータ最大電流 : 1.0[A]
- ロジック電圧 : DC 4.5[V] ~ 20[V]

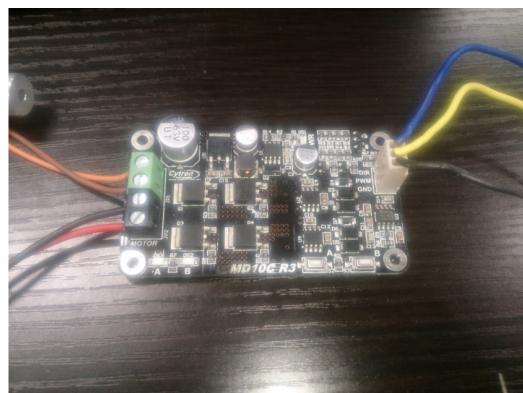


図 16: MD10C R3

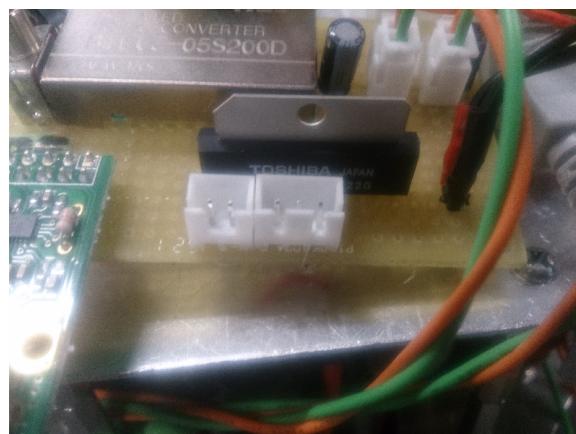


図 17: TA7291P

7.3 センサ仕様

ロボットに搭載されているセンサは以下である.

- 赤外線測距センサ : GP2Y0A025YK : 有効距離 20 ~ 150[cm]
- ToF 近距離センサ : VL6180x : 有効距離 5 ~ 20[cm]
- 3 軸加速度センサ : KXR94-2050 : [x, y, z] 軸 加速度出力
- ジャイロセンサ : L3GD20 : [x, y, z] 軸 角加速度出力
- カメラモジュール : P5V04A : RPi 用カメラ

7.3.1 測距センサ

測距センサは本体周囲に PSD センサを 7 つ , 前方に ToF 近接センサを 3 つ搭載する . (図 18) これは自律行動の際に , 周辺環境 , 特に各種ポールを把握するために用いる . このとき , 近接センサは I^2C 通信によって使用する . センサの仕様については実験を行ったので 5 に示す .

また , 各測距センサには信号のノイズを吸収し安定化させるために $0.1[\mu F]$ のセラミックコンデンサを接続する . これは , コンデンサの持つ交流成分のみを吸収し , 直流成分を通すというローパスフィルタ的特徴を利用したものである .

7.3.2 3 軸加速度・ジャイロセンサモジュール

加速度センサは [x, y, z] 軸におけるロボットの加速度を測定するものである .

ジャイロセンサは [x, y, z] 軸まわりの角加速度を測定するものである . (図 19)

我々はこれらをロボットの自己位置推定に用いる . 特にジャイロセンサについては , ロボット本体の直進走行制御に使用する . 詳細は ??において説明する .

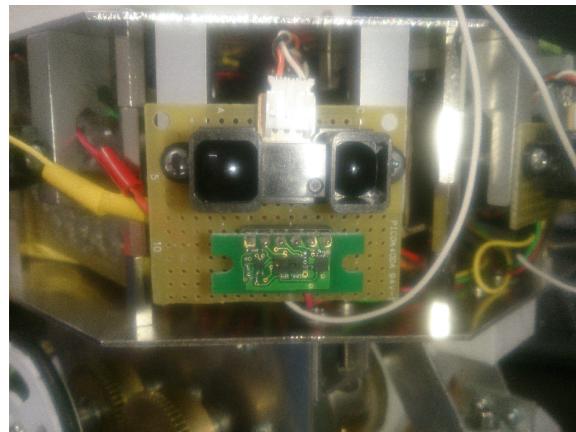


図 18: 上:PSD センサ 下:ToF センサ

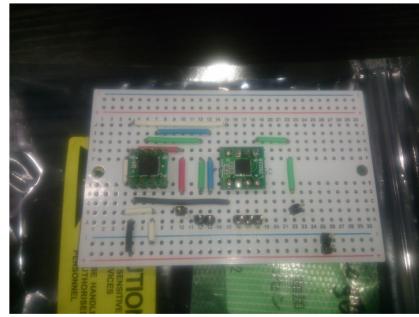


図 19: 3 軸加速度・ジャイロセンサモジュール



図 20: 電源回路モジュール

7.4 電源回路

電源回路は各回路図の左上に示している。

バッテリーはひとつしか搭載しないが、RPi と Arduino では定格電流値が異なるために同一の電源は使用できない。そこで、それぞれに降圧レギュレータとして DCDC コンバータを用いてバッテリーからの供給電源を分電することとした。各仕様を下に示す。また、実際に作成した電源回路を図 20 に示す。

[LR8697](RPi・モータ用)

- 電源電圧：DC 6.0[V] ~ 42.0[V]
- 出力電圧：DC 5.0[V]
- 出力電流：2.5[A]

[BTD05-05S200D](Arduino・センサ用)

- 電源電圧：4.5-9.0[V]
- 出力電圧：5.0[V]
- 出力電流：2000[mA]

7.5 I^2C 通信

今回、我々のロボットには測距センサを始めとする複数のセンサが搭載されている。これらの殆どがアナログ出力であるが、Arduino のアナログ I/O ポートは 6 つしかなく、要求を満たしていない。

そこで、 I^2C 通信を用いることとする。これは、 I^2C 通信がパーティライン構成が可能となっており、1 つのマスタで複数のスレーブデバイスと通信することが可能であるからである。概要を以下に示す。

- (1) マスタ側 (Arduino) とスレーブ側 (n 個のセンサ等) を明確に分け、各スレーブに異なるアドレスを割り振る。
- (2) マスタ側が、Start Condition を出力し続いてアドレスと Read/Write 要求を出力する。
- (3) 全スレーブがこの時の SCL のクロックを元に SDA のデータを受信し、SSPADD レジスタにセットされたアドレスと一致したデバイスだけが、その後の送受信を継続する。
- (4) 受信した側がデータを受信完了すると自動的に ACK ビットを返送し、同時に SSP 割込みを発生する。
- (5) これをマスタが Stop Condition を出力するまで続ける。

本ロボットでは、近接センサ・ジャイロセンサについて I^2C 通信を行うこととする。また、Arduino と各デバイスは Arduino の SDA・SCL ポートを使用することで通信が可能となる。これを実現するために Arduino 用 I^2C バス用双方向電圧レベル変換モジュール (図 21) を使用して接続した。接続の方法は回路図に示している。

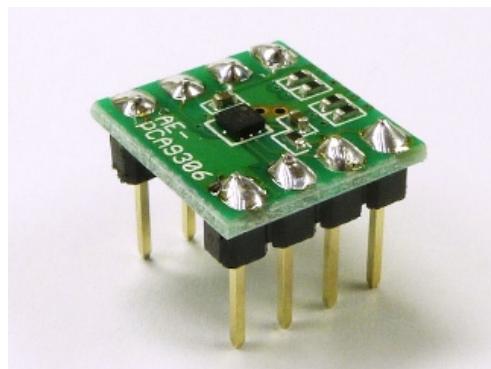


図 21: レベル変換モジュール

8 ソフトウェア

8.1 モデル化

プログラム上でセンサから読み取った値を扱えるようにするために、モデル化を行った。ここで距離、加速度、電圧の単位はそれぞれ cm, m/s², V である。

8.1.1 PSD センサ GP2Y0A21

前回求めた PSD センサの性能から距離-電圧特性は以下のようになる。

$$(距離) = 45.514 \times (\text{電圧})^{-0.822} \quad (8.1)$$

8.1.2 近接センサ VL6180X

前回の PSD センサと同様な実験で、近接センサの距離と出力値の関係を求めた。そのときの関係を表した式を以下に示す。また、出力値とは arduino で 0 ~ 255 までの値で出力された値であり、対象物との距離が近いほど、値が小さくなるようにプログラムで処理している。

$$(距離) = 0.09999 \times (\text{出力値}) + 0.4477 \quad (8.2)$$

8.1.3 加速度センサ KXR-94

この加速度センサのモデル化するために必要な仕様を以下に示す。

- 感度 : 1 [V/g]
- オフセット : 2.5 [V]

これらより、加速度と電圧の関係を以下に示す。

$$(加速度) = 9.8 \times [(\text{電圧}) - 2.5] \quad (8.3)$$

8.2 フィルタ処理

8.2.1 ローパスフィルタ

加速度センサやジャイロセンサで読み取った値は、ノイズが入り大きく上下していた。そこでセンサ値の平滑化を行うため、ローパスフィルタを用いた。ローパスフィルタの式は以下に示す。

$$y[i] = py[i - 1] + (1 - p)x[i] \quad (8.4)$$

ここで、 $y[i]$ は出力値、 $y[i - 1]$ は前回の出力値、 $x[i]$ は現在のセンサ値であり、 p ($0 < p < 1$) はパラメータである。この式の特徴として、パラメータ p を大きくすれば滑らかになるが、位相が遅れることがわかっている。

8.2.2 ハイパスフィルタ

センサ値をローパスフィルタで平滑化を行い、移動距離や回転角度を求めるためにその値の積分を行った。ここでの積分は数値積分であったため、積分誤差が生じ、時間とともに値がずれていた。これを解決するためにハイパスフィルタを用いた。ハイパスフィルタの式は以下に示す。

$$\dot{y}[i] = x[i] - y[i] \quad (8.5)$$

ここで、 $\dot{y}[i]$ は出力値、 $x[i]$ は現在の積分したセンサ値、 $y[i]$ は積分したセンサ値をローパスフィルタに通した出力値である。

8.3 自己位置推定

8.3.1 アフィン変換

加速度センサで得られる加速度はロボットのローカル座標系における加速度である。そこで、ジャイロセンサから得られた角度を用いて三次元アフィン変換を適用しグローバル座標系における加速度を求める。アフィン変換において、回転移動は次の行列で表される。 x 軸、 y 軸、 z 軸まわりの回転行列をそれぞれ R_x, R_y, R_z とおくと、

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_x & \sin \theta_x & 0 \\ 0 & -\sin \theta_x & \cos \theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.6)$$

$$R_y = \begin{bmatrix} \cos \theta_y & 0 & -\sin \theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta_y & 0 & \cos \theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.7)$$

$$R_z = \begin{bmatrix} \cos \theta_z & \sin \theta_z & 0 & 0 \\ -\sin \theta_z & \cos \theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.8)$$

また、ローカル座標系の原点のグローバル座標系での座標を (T_x, T_y, T_z) とすると、アフィン変換における平行移動は次の行列で表される。

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix} \quad (8.9)$$

ここで、ローカル座標系を $(x, y, z, 1)$ 、グローバル座標系を $(x_g, y_g, z_g, 1)$ とすると

$$(x_g \ y_g \ z_g \ 1) = (x \ y \ z \ 1) R_z R_x R_y T \quad (8.10)$$

となる。これによってローカル座標系の加速度センサで得られた加速度をグローバル座標系に変換することができた。

8.3.2 ジャイロセンサ

ジャイロセンサ(角速度センサ)を用いて、車体の角度を求める。角速度センサと同様に考える。角度を $\theta(t)[\text{deg}]$ 、角速度を $\omega(t)[\text{deg}/\text{s}]$ とすると

$$\theta(t) = \frac{t_2 - t_0}{6} (\omega(t_0) + 4\omega(t_1) + \omega(t_2)) \quad (8.11)$$

となる。

8.3.3 加速度センサ

加速度センサを用いて、移動距離を求める。そのために加速度センサから、加速度とそのときの時間を得る。今回は加速度を積分するのに、数値積分のシンプソン公式を用いた。シンプソン公式による区間 (a, b) の $f(t)$ の積分値は以下のようになる。

$$\int_a^b f(t) dt = \frac{b-a}{6} \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) \quad (8.12)$$

これを用いて、 (t_0, t_2) の区間で積分する(ただし $t_1 = \frac{t_0+t_2}{2}$)。
 $a(t)[\text{m}/\text{s}^2]$ を加速度、 $v(t)[\text{m}/\text{s}]$ を速度とすると

$$v(t) = \frac{t_2 - t_0}{6} (a(t_0) + 4a(t_1) + a(t_2)) \quad (8.13)$$

となる。よって移動距離を $x(t)[\text{m}]$ とすると以下のようになる。

$$x(t) = \frac{t_2 - t_0}{6} (v(t_0) + 4v(t_1) + v(t_2)) \quad (8.14)$$

8.4 画像認識

炎上ポールの認識には配布された Raspberry Pi NoIR Camera V2(以下、カメラモジュール)を使用する。画像撮影から最も近いポールと思われる物体へのベクトルを出力する一連の手順を、簡単に以下に示す。画像処理にはOpenCVライブラリを用いており、各処理で使用した主要なライブラリ関数を併記する。

画像取得

カメラモジュールへのアクセスにはOpenCVとは異なる既成ライブラリ[1]を利用した。撮影によりピクセル値の2次元配列(`cv::Mat`)が出力として得られる。

カラーモデル変換

得られた画像のカラーモデルをRGBからHSVに変更する。

使用関数：`cv::cvtColor`

2値画像化

HSV画像データに対し赤色マスクをかけて2値画像に変換する。

使用関数：`cv::inRange`

ノイズ除去

モルフォロジー処理によりノイズを除去する。

使用関数 : cv::morphologyEx

構造解析

2値画像中の輪郭線を検出した後、それを矩形で囲む。囲んだ矩形を縦横比で解析し、ポールの縦横比に対して $\pm 20\%$ 以上の差があるものを除外する。

使用関数 : cv::findContours, cv::boundingRect

ベクトル作成

除外されず残った矩形の重心点を求め、カメラの画角 (62.2×48.8 [2]) を元に機体中心から重心点へ向かうベクトルを作る。

使用関数 : cv::moments

8.5 アルゴリズム

機体の進行方向及び速度は人工ポテンシャル法により制御を行う。

目標方向として、赤い物体へ向かうベクトル v_{red} または順路に沿ってゴールへ向かうベクトル v_{goal} を与える。また、障害物回避のためポテンシャル場を設定する。

二次元平面上の円筒状の障害物（ポール） p_i について、中心軸を原点にスカラー場 f_i を設定する。

$$f_i(x, y) = -\tanh^{-1}(x^2 + y^2 - 1) \quad (8.15)$$

次に障害物 p_i のスカラー場の勾配として表されるベクトル場は次のようになる。ただし、 \hat{x}, \hat{y} はそれぞれ x, y 軸方向の単位ベクトルである。

$$\frac{\partial f_i}{\partial x} \hat{x} + \frac{\partial f_i}{\partial y} \hat{y} = \nabla f_i(x, y) \quad (8.16)$$

しかし、実際の機体はコース上の全障害物の位置を常に把握しておくことは不可能であり、測距センサの検出距離にも限りがあることから、機体から半径 $d_{max} = 0.9$ [m] の範囲内のポールについてのみベクトル場を考慮している。

測距センサにより得られたポール p_i へのベクトルを $\mathbf{d}_i = [d_{ix}, d_{iy}]^T$ とすると、機体がポールから受ける斥力 \mathbf{F}_i は次のように表せる

$$\mathbf{F}_i = \nabla f_i \left(\frac{d_{ix}}{d_{max}}, \frac{d_{iy}}{d_{max}} \right) \quad (8.17)$$

以上より、範囲内に n 個ポールが存在する場合の機体の進行方向 \mathbf{v} は次のように表せる。

$$\mathbf{v} = \frac{\mathbf{v}_{target}}{\|\mathbf{v}_{target}\|} + \frac{\sum_{i=0}^{n-1} \mathbf{F}_i}{\|\sum_{i=0}^{n-1} \mathbf{F}_i\|} \quad (8.18)$$

\mathbf{v}_{target} には、 v_{red} と v_{goal} のどちらかが設定される。機体正面のカメラにより撮影された画像を元に v_{red} が生成されている場合はそれが、ない場合は v_{goal} が設定される。

なお， v_{target} に v_{red} が設定された場合，機体は赤い物体の直前で $v = [0, 0]^T$ となり停止する．これを消火作業を行うための条件とし，消火作業により機体正面のカメラから赤色の物体が消えることで消火作業が終了する．

8.6 今後の予定

図 22 に大まかな今後の予定を示す．

9 使用物品

今回の RCR で使用する予定の物品を表 9.3, 9.4 に示す．5 月 31 日時点での合計金額は 79248 円である．

		ソフト				ハード		
	week	大目標	タスク			大目標	タスク	
5月	1	予備実験	画像解析	6軸センサ	アルゴリズム開発	機体製作	本体・駆動系	基幹電装
	2					センサ類実装	センサ電装	
	3	仮走行試験	モータドライバ	距離センサ		調整	走行試験の結果を踏まえて調整	
	4	自己位置推定	オドメトリ解析					
6月	1	実環境走行試験			アルゴリズムの実装・調整			
	2							
	3							
	4	試運転			模擬コース上での走行試験			
7月	1				調整期間			
	2							
	3							
	4				競技会			

図 22: 今後の予定

表 9.2: 配布物品

メーカー	商品名	単価	数量	価格
秋月電子通商	ピンヘッダ (1X40)	35 円	1	35 円
秋月電子通商	ピンヘッダ (2X40)	50 円	1	50 円
秋月電子通商	ソケット (2X40)	95 円	1	95 円
秋月電子通商	スミチューブ (黒)	40 円	1	40 円
秋月電子通商	スミチューブ (赤)	40 円	1	40 円
秋月電子通商	スミチューブ (クリア)	40 円	1	40 円
秋月電子通商	ハンダ	280 円	1	280 円
秋月電子通商	ハンダ吸い取り線	190 円	1	190 円
日本圧着端子	XH ベース 2 極	10 円	2	20 円
日本圧着端子	XH ベース 3 極	10 円	7	70 円
日本圧着端子	XH ベース 4 極	10 円	6	60 円
日本圧着端子	XH コネクタ 2 極	10 円	2	20 円
日本圧着端子	XH コネクタ 3 極	10 円	7	70 円
日本圧着端子	XH コネクタ 4 極	10 円	6	60 円
日本圧着端子	圧着端子 (XH)	4 円	15	60 円
光	アルミ板	651 円	1	651 円
			合計	1130 円

表 9.3: 引き継ぎ品

メーカー	商品名	単価	数量	価格
TAMIYA	タミヤギヤードモータ	4860 円	2	9720 円
TAMIYA	ギヤードモータハブシャフトセット	540 円	1	9720 円
TAMIYA	ミディアムナローレーシンググラジアルタイヤ	540 円	1	540 円
TAMIYA	ミディアムナロー 5 本スパークホイール	540 円	1	540 円
TAMIYA	SP.106 7.2V コネクタ	270 円	2	540 円
RS コンポーネンツ株式会社	カメラモジュール	2740 円	1	2740 円
RS コンポーネンツ株式会社	Raspberry Pi 3 ケース	1080 円	1	1080 円
RS コンポーネンツ株式会社	Raspberry Pi 3 Model B	6200 円	1	6200 円
MISUMI	トグルスイッチ	80 円	2	160 円
MISUMI	キャスター ホイール	990 円	1	990 円
KHK	歯車 (モジュール 0.5 歯数 60)	1360 円	4	5440 円
KHK	プラスチックナット+連結スペーサーセット	100 円	1	100 円
秋月電子通商	3 軸加速度センサモジュール	850 円	1	850 円
秋月電子通商	3 軸ジャイロセンサモジュール	750 円	1	750 円
Cytron	DC ブラシモータ ドライバ	2160 円	2	4320 円
Powers	POWER MAX 4000 Ni-MH	4000 円	2	8000 円
シャープ株式会社	シャープ測距モジュール	400 円	6	2400 円
株式会社矢島製作所	ユニバーサル基盤	100 円	2	200 円
日本電産コバル電子	ロータリエンコーダ	7500 円	2	15000 円
Arduino Srl	Arduino UNO	3240 円	1	3240 円
TOSHIBA	microSDHC メモリーカード 8GB	1058 円	1	1058 円
OptoSupply	スペーサー	45 円	4	180 円
GWS	サーボモータ	1000 円	1	1000 円
ルビコン株式会社	電解コンデンサ (47 μ F)	10 円	2	20 円
			合計	65608 円

表 9.4: 新規購入品

取引先	商品名	型番または商品コード	単価	数量	価格
Strawberry Linux	近距離センサモジュール	VL6180X	1296 円	3	3888 円
Strawberry Linux	降圧型 DC-DC コンバータ	LT8697	1404 円	1	1404 円
スイッチサイエンス	アナログマルチプレクサ	767 円	1	767 円	
秋月電子通商	シャープ測距モジュール	GP2Y0E03	760 円	1	760 円
秋月電子通商	I2C バス用双方向レベル変換モジュール	PCA9306	150 円	1	150 円
misumi	小径玉軸受	FL686ZZ	440 円	4	1760 円
misumi	イグリデュール G フランジ型軸受	GFM-0608-25	75 円	5	375 円
misumi	アルミ 角パイプ	HFHQ1515-1.5-500	200 円	1	200 円
秋月電子通商	モータドライバ	TA7291P	300 円	1	300 円
横山テクノ	アルミ 丸パイプ 6 × 1.0 × 400		310 円	1	310 円
ヨドバシカメラ	ミニモータ低速ギアボックス	70189	716 円	1	716 円
				合計	11610 円

参考文献

- [1] "RaspiCam: C++ API for using Raspberry camera with/without OpenCV" ,
["https://www.uco.es/investiga/grupos/ava/node/40"](https://www.uco.es/investiga/grupos/ava/node/40) ,
 2017 年 5 月 31 日最終確認 .
- [2] "RPi Camera Module - eLinux.org" ,
["http://elinux.org/RPi_Camera_Module#Technical_Parameters_.28v.2_board.29"](http://elinux.org/RPi_Camera_Module#Technical_Parameters_.28v.2_board.29) ,
 2017 年 5 月 31 日最終確認 .

```
void setup() {  
    Serial.begin(9600) ;  
}  
void loop() {  
    int ans ;  
  
    ans = analogRead(0) ;  
    Serial.println(ans) ;  
    delay(500) ;  
}
```