

1 回路設計・作成

設計・作成した回路について仕様や選定の理由について以下に示す。また使用する部品の一覧を表 1.1 に示す。

1.1 マイコンの選定

設計した回路を図 1, 図 2 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、アナログのセンサ類の処理が困難である。そこで、アナログ・ディジタル 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 1.1: 回路用部品表

タイプ	部品名	数	用途
マイコン	Raspberry Pi3 modelB	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
ローテーションモータ	GWS S35 STD	1	アーム用
モータドライバ	MD10C-R3	2	タイヤ用
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
PSD 測距モジュール	GP2Y0A02YK	7	中距離センサ
ToF 近距離センサモジュール	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	信号安定化(ローパスフィルタ回路)
バッテリー	POWER MAX 4000 Ni-MH	1	電源バッテリー 7.2[V]4200[mAh]

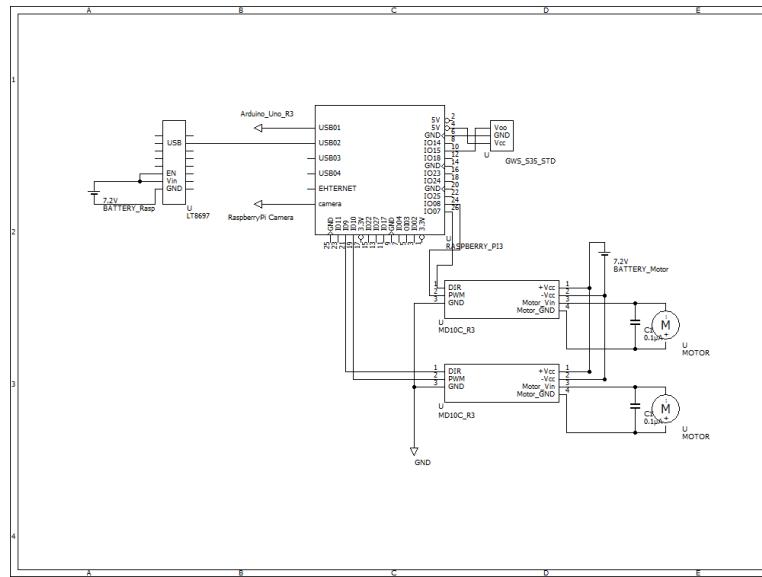


図 1: Raspberry Pi3 接続回路図

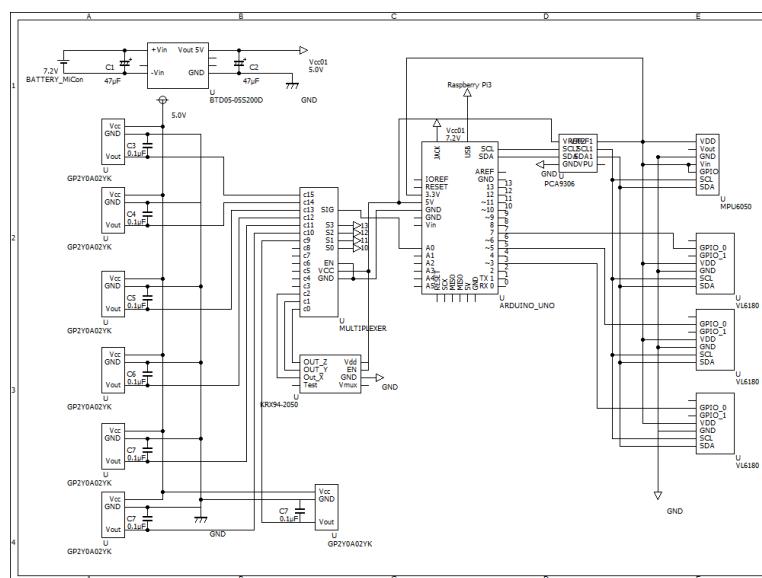


図 2: Arduino unoR3 接続回路図

1.2 モータドライバ

モータドライバは「MD10C R3」(図 3)を両輪駆動用として 2つ使用する。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]

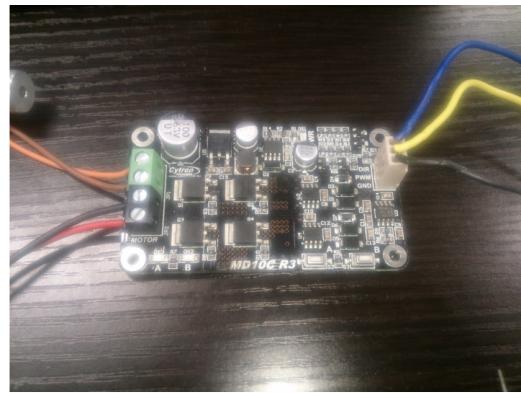


図 3: MD10C R3

1.3 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 : BGD20 : [x, y, z] 軸 角加速度出力
- カメラモジュール : P5V04A : RPi 用カメラ

1.3.1 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 4) 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??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。

1.3.2 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 5)

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。

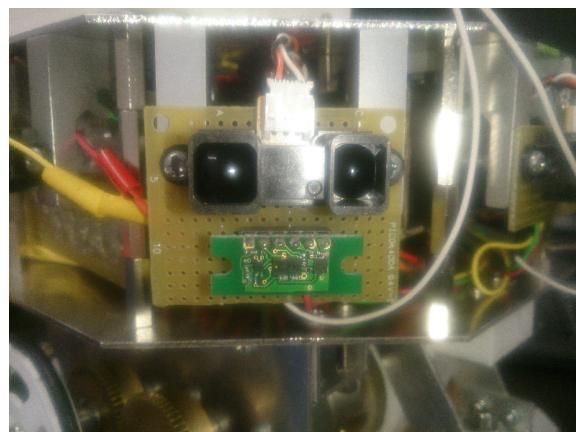


図 4: 上:PSD センサ 下:ToF センサ

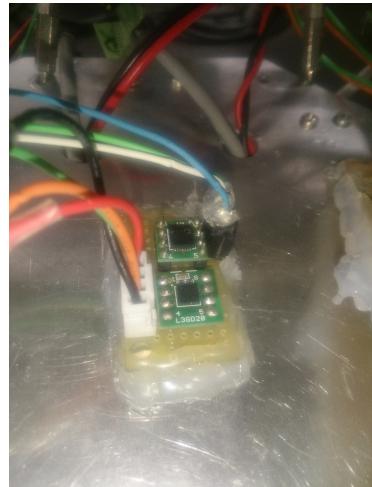


図 5: 3 軸加速度・ジャイロセンサモジュール



図 6: 電源回路モジュール

1.4 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 6 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 : 2500[mA]

[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]

1.5 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 7) を使用して接続した。接続の方法は回路図に示している。

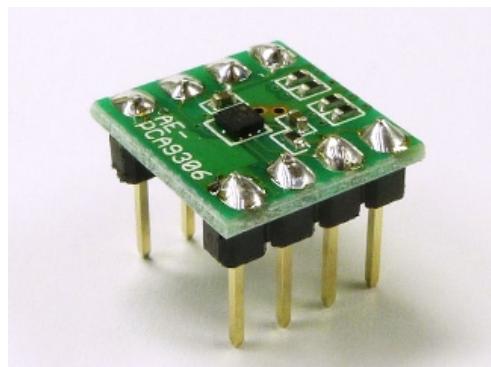


図 7: レベル変換モジュール

1.6 回路作成

本ロボットはセンサを計 13 個搭載している。そのため、それぞれの配線や配置は困難であった。電源管理では、当初センサへの給電を Arduino からしていたが、電流の不足から RPi とのシリアル通信が不安定になるという問題が生じた。そのため、電源回路からきちんと別電源を用意することとした。

また、最大の問題となったのが配線である。全方位に搭載した各種センサのために機体内部の配線が煩雑となってしまい、故障の原因となっていた。

そこで、センサ配線のハブモジュールの作成や各種ピンのコネクタによる一元化を行うことで整理された配線となるように工夫した。それでもメンテナンスの困難さは解消されなかったことが反省点であり、センサの数の最適化や回路仕様の工夫を検討すべきだったと考える。