

目次

第 1 章 序 論	3
1.1 研究の背景	3
1.2 高専ロボコンについて	4
1.3 研究の目的	6
1.4 論文の構成	6
第 2 章 標準制御システムのあるべき姿	8
2.1 要求スペックの検討	8
2.2 Arduino の性能比較	9
2.3 GR シリーズ	10
2.4 GR-SAKURA	10
2.5 今年度の構成	11
2.6 ロボコン shield	12
第 3 章 ログシステムの開発	17
3.1 ログシステムの要求項目	17
3.2 ログシステムのフローチャート	17
3.3 ログシステムのヘッダファイル	18
3.4 ログシステムプログラム	19
第 4 章 結 言	20
4.1 本研究のまとめ	20
4.2 今後の課題	20

参考文献	21
付録 A 制御システムマニュアル	23
A.1 開発環境導入	23
A.2 モータ制御	26
A.3 映像ストリーミングシステム	30
A.4 プログラム	33
A.5 シールド回路図	38

第1章

序論

1.1 研究の背景

本研究室では毎年高専ロボコンに参加していて、去年まで3年連続で全国大会に出場している。今年は地区大会を優勝し、全国大会に出場することを目指とした。試合に勝つための作戦を考え、その作戦を実行できるロボットを設計・製作した。しかし、地区大会初戦で原因不明のマシントラブルに見舞われ敗退した。原因不明というのは、試合後は正常に動作し現象が再現されないこと。そしてロボットのデータログを記録していくなかったため、状況を再現して検証が行えないため。学校での動作試験でも原因不明の不具合に幾度か見舞われたが、現象を再現できず、対応に遅れ結果的にロボット完成が遅れた。これらのことから、ロボットのデータログを記録していれば多くの不具合を修正でき、上位入賞できたのではないかと考える。

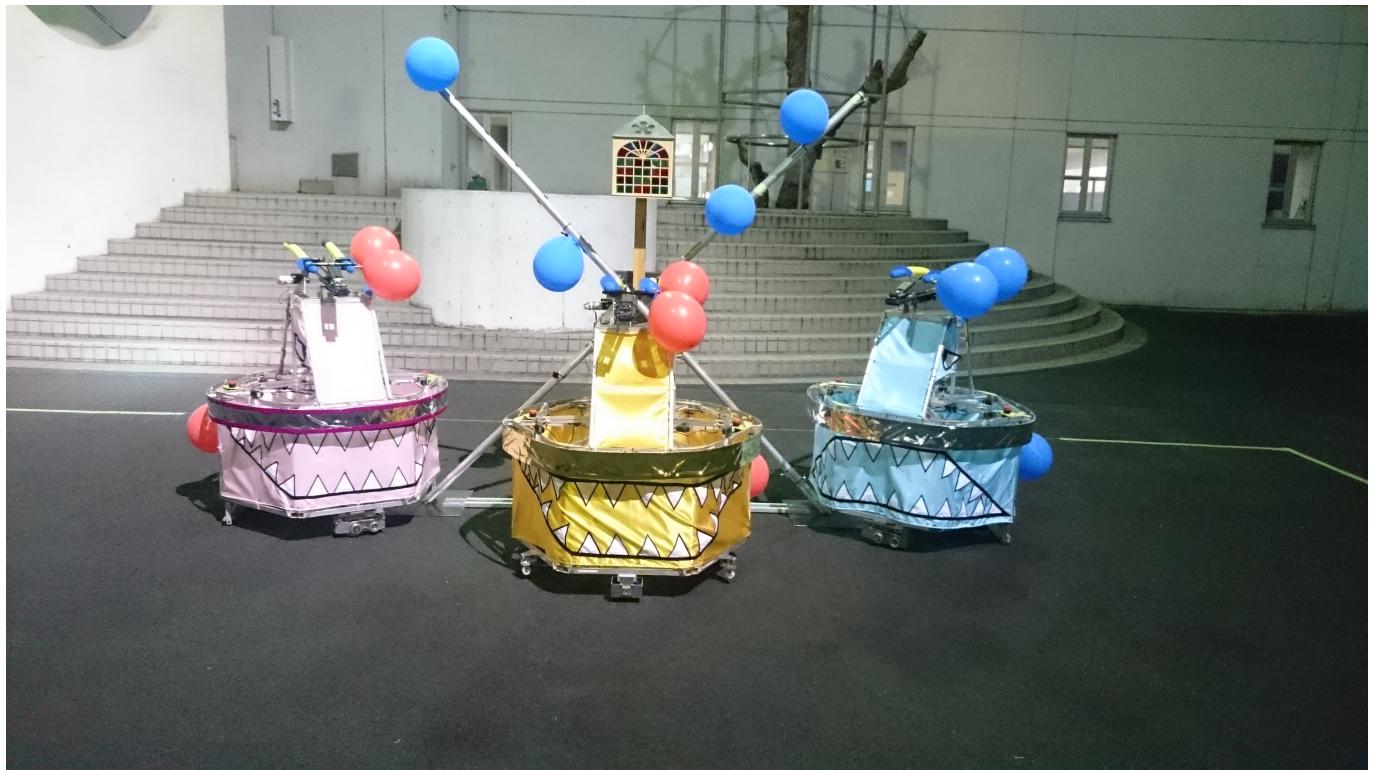


図 1.1 今年度制作したロボット

1.2 高専ロボコンについて

アイデア対決・全国高等専門学校ロボットコンテストの略称であり,NHKが主催するロボットコンテストの1つである。高等専門学生の甲子園といわれていて、ロボコンに憧れて高専へ入学した人も多いだろう。各校A,Bの2チーム参加でき、地区大会の優勝チームと推薦チームが全国大会へ出場できる。本研究室では2014年から2016年まで連続で全国大会に出場していた。本研究室で過去に制作されたロボットたちを下図に示す。

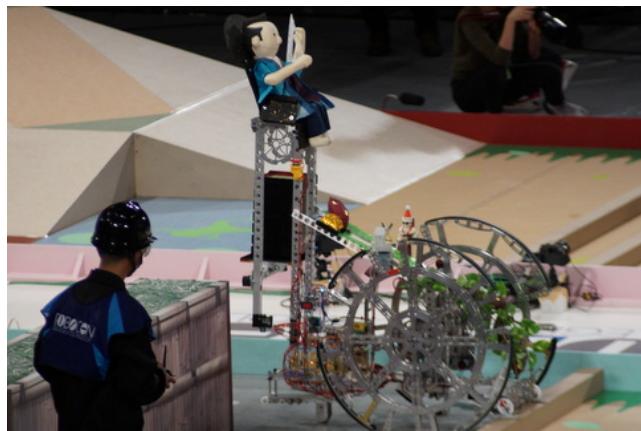


図 1.2 ポテ☆ゴロー

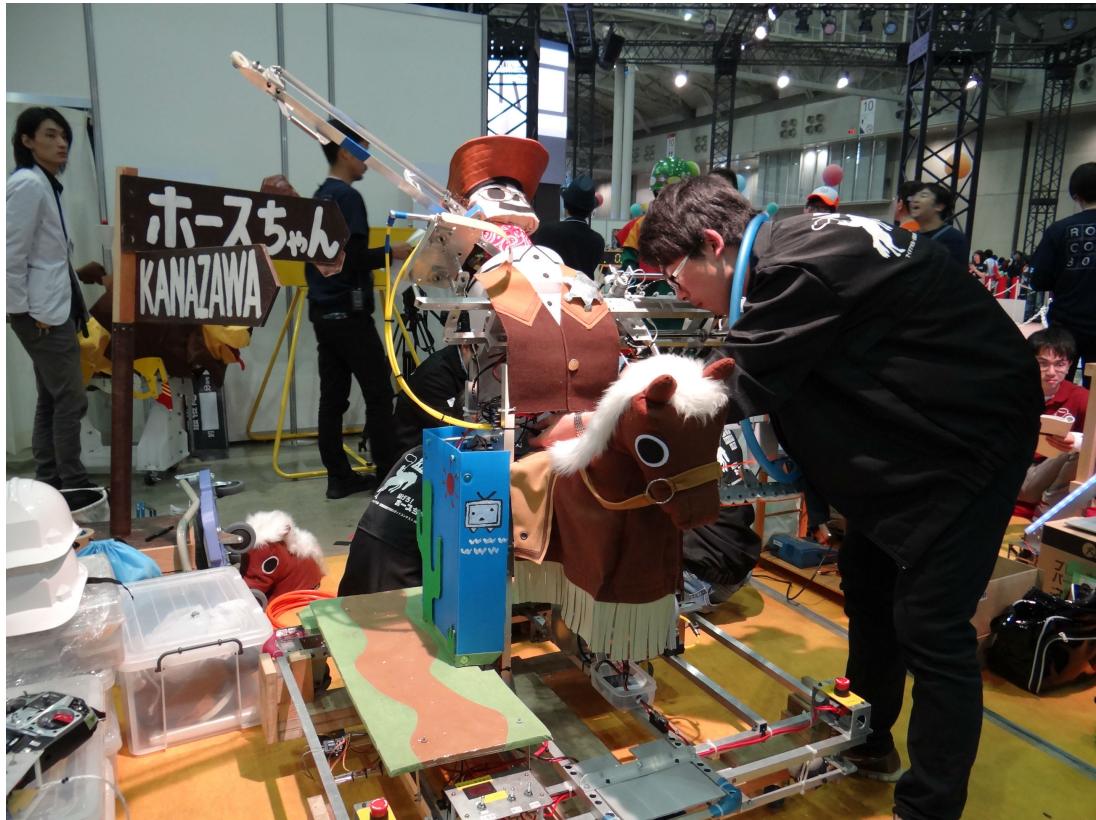


図 1.3 投げろホースちゃん

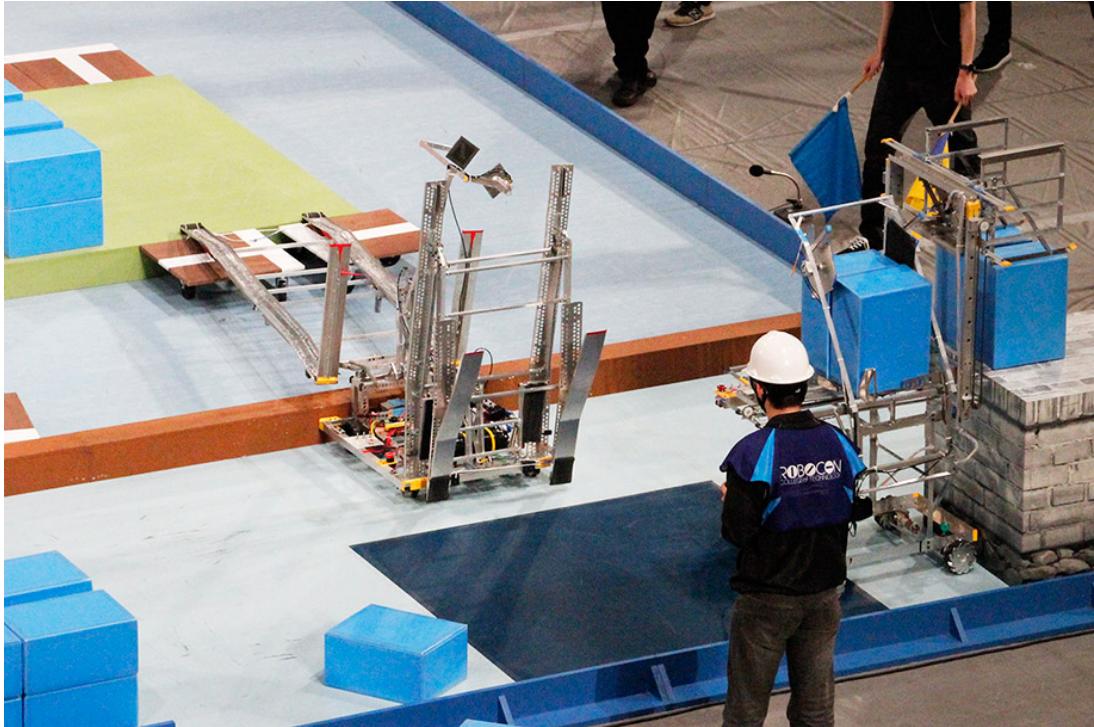


図 1.4 鼓積

1.3 研究の目的

今までではロボットの制御システムから設計を始め、マイコンの選定などを行ってきた。しかし、毎回そこから始めるのは大変で、時間がかかるため効率的ではない。そこで、本研究室のロボコン用標準制御システムを考案し、それをベースとして制御システムの開発をすることにより開発にかかる時間を短縮する。加えて、標準制御システムのデータログシステムを開発することによりトラブルシューティングの効率化を図る。標準制御システムのマニュアルを作成することで、プログラムが苦手な機械科学生でも制御システムの開発を行えるようにし、来年度以降のロボコン活動を支援する。

1.4 論文の構成

この論文では以下のように構成されている。1章では、研究目的や高専ロボコンの説明、今年度の敗因について考察したものである。

2章では、ロボコンの標準制御システム考案とロボコンShieldについて述べる。

3章では、本研究で作成したログシステムについて述べる。

4章では、本研究のまとめを述べる。

第2章

標準制御システムのあるべき姿

2.1 要求スペックの検討

今回は ArduinoMega をメインとして使用した制御システムを構成したが、ロボコン用ロボットに必要なマイコンはどのくらいのスペックが要求されるのか今年度使用したマイコンのスペック表2.1を元に検討した結果表2.1のようになつた。

表2.1 今年度のマイコンスペックと使用用途

項目	数	使用用途、詳細
デジタルIO	54	ソレノイド5つの制御、実装予定だった近接センサ2つ
アナログIO	16	使用しなかった
pwmピン	15	今回は ServoShield を使用したため使用しなかった
シリアルポート	4	タイヤ1つにつき1つで最大3つ使用した
USBホスト	1	USBHostShield で追加してコントローラとの通信に使用した

表2.2 要求スペック

項目	数	理由
デジタルIO	16	ソレノイドやLEDを使うのに多く必要
アナログIO	4	アナログデータが必要なアクチュエータやセンサをする場合に必要
pwmピン	16	サーボモータの制御に使うため
シリアルポート	4	今年度は3輪オムニの制御に3つ使用したが、4輪メカナムや4輪オムニなどを使用するには4つ必要である
SDカードソケット	1	データログを記録するためにSDカード使う
USBホスト	1	Bluetooth ドングルを使い Bluetooth 通信するため

2.2 Arduino の性能比較

今回の高専ロボコンではライブラリが豊富であり比較的開発が簡単であるArduinoシリーズの中でSerialポートが3つあるArduinoMegaを使用したが、Arduinoの性能を調べ比較してみる。各Arduinoの性能をまとめた表2.2を下に示す。

表 2.3 Arduino スペック一覧

名前	ArduinoUNO	Arduino101	ArduinoPRO	ArduinoProMini	ArduinoMega2560
大きさ	68.6×53.4mm	68.6×53.4mm	43×18mm	101.52×53.3mm	56×25mm
重さ	25g		9g		37g
価格	3200	4980	1868	1243	5830
マイコン	ATmega328P	InrelCurie	ATmega328	ATmega328	ATmega2560
動作電圧	5V	3.3V	3.3or5V	5V	3.3or5V
電源電圧	7-20V	7-20V	3.35-12V	3.35-12V	6-20V
デジタルIO	14	14	14	14	54
アナログIO	6	6	6	8	16
pwmピン	6	4	6	6	15
シリアルポート	1	1	1	1	4
フラッシュメモリ	32kB	196kB	32kB	32kB	256kB
SRAM	2kB	24kB	2kB	2kB	8kB
EEPROM	1kB		1kB	1kB	4kB
クロック	16MHz	32MHz	8or16MHz	8or16MHz	16MHz
名前	ArduinoYUN	ArduinoDUE	ArduinoMegaADK	ArduinoEthrnet	ArduinoLeonardo
大きさ	101.52×53.3mm	101.52×53.3mm	68.6×53.3mm	68.6×53.4mm	28×65mm
重さ	36g	36g	28g	20g	9g
価格	9990	6264	8208	7676	3132
マイコン	ATmega32U4	AT91SAM3X8E	ATmega2560	ATmega328	ATmega32U4
動作電圧	5V	3.3V	5V	5V	5V
電源電圧	5V	6-16V	6-20V	6-20V	7-12V
デジタルIO	20	54	54	14	20
アナログIO	12	12	16	6	12
pwmピン	7	12	15	5	7
シリアルポート	1	4	4	1	1
フラッシュメモリ	32kB	256kB	256kB	32kB	32kB
SRAM	2.5kB	96kB	8kB	2kB	2.5kB
EEPROM	1kB	4kB	1kB	1kB	1kB
16MHz	84MHz	16MHz	16MHz	16MHz	16MHz

2.3 GR シリーズ

Arduino シリーズでは各種 IO に関しては十分なものもあるが, USBhost や SD カードスロットがなく機能を追加するにはいくつも Shield という基板を重ねなければならない。そこで, USBhost や SD カードスロットが既存で備わっている GR シリーズのスペックをまとめ検討する。GR シリーズの性能をまとめた表 2.3 を下に示す。

表 2.4 GR スペック一覧

名前	GR-SAKURA	GR-KAEDE	GR-PEACH
大きさ	53.34×65.58mm	53.84×91.83mm	53.84×67.58mm
価格	4640	7500	9690
マイコン	RX63N	RX64N	RZ/A1H
動作電圧	3.3V	3.3V	3.3V
電源電圧	5V	5V	5V
デジタル IO	55	29	52
アナログ IO	16	12	6
pwm ピン	9	9	6
シリアルポート	6	4	7
フラッシュメモリ	32kB	64kB	8MB
SRAM	128kB	552kB	10MB
EEPROM	1MB	4MB	8MB
クロック	96MHz	96MHz	400MHz

2.4 GR-SAKURA

GR-SAKURA FULL をメインマイコンに選定した。GR-SAKURA を図 2.1 に示す。

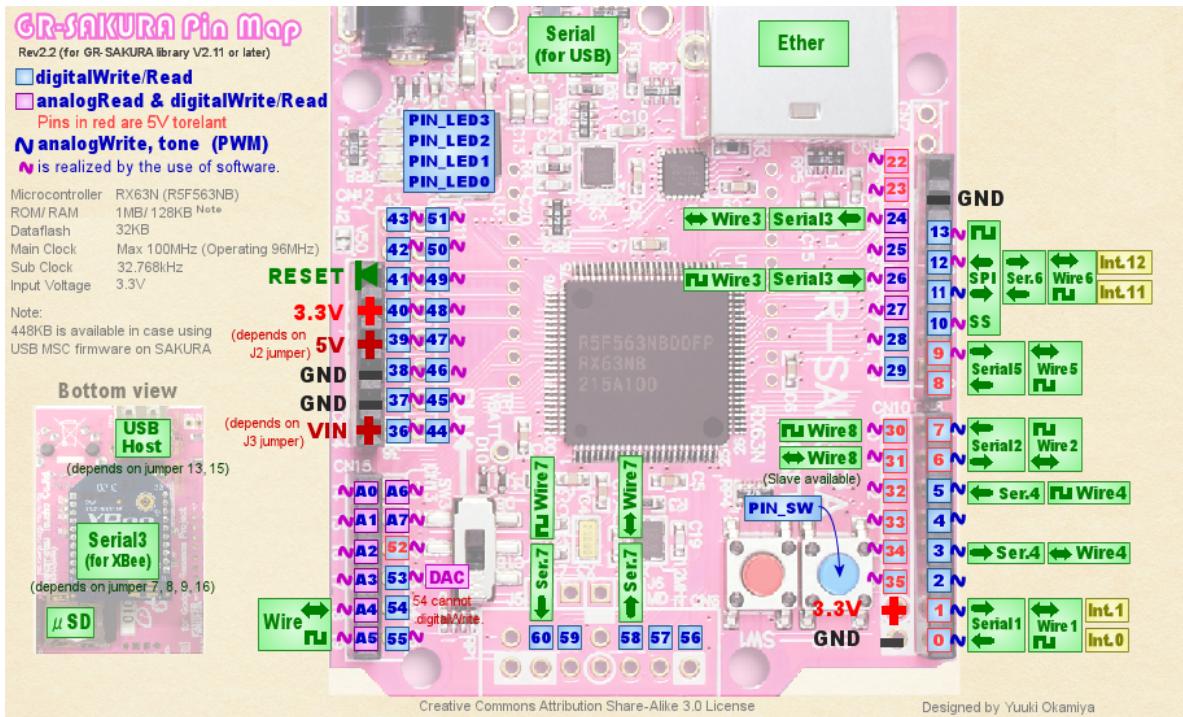


図 2.1 GR-SAKURA

図 2.1 からわかるように GR-SAKURA は 豊富な IO に加えて LAN コネクタ,USB ホストコネクタ,SDMMC カードソケットを備えている。

2.5 今年度の構成

Shield とは Arduino マイコンに機能を追加するための基板であり,モータ制御するためのものや LCD が付いたものなど多くの種類が存在する。それらを Arduino に重ねることにより多くの機能を Arduino に持たせることができ。今年度は ArduinoMega にコントローラーと Bluetooth 通信するための USBhostShield とサーボモータを制御するための Adafruit16-ChannelServoShield を載せ,さらに各種 IO を使いやすくするための自作 Shield も搭載した。今年度のロボット制御基板を図 2.2 に示す。

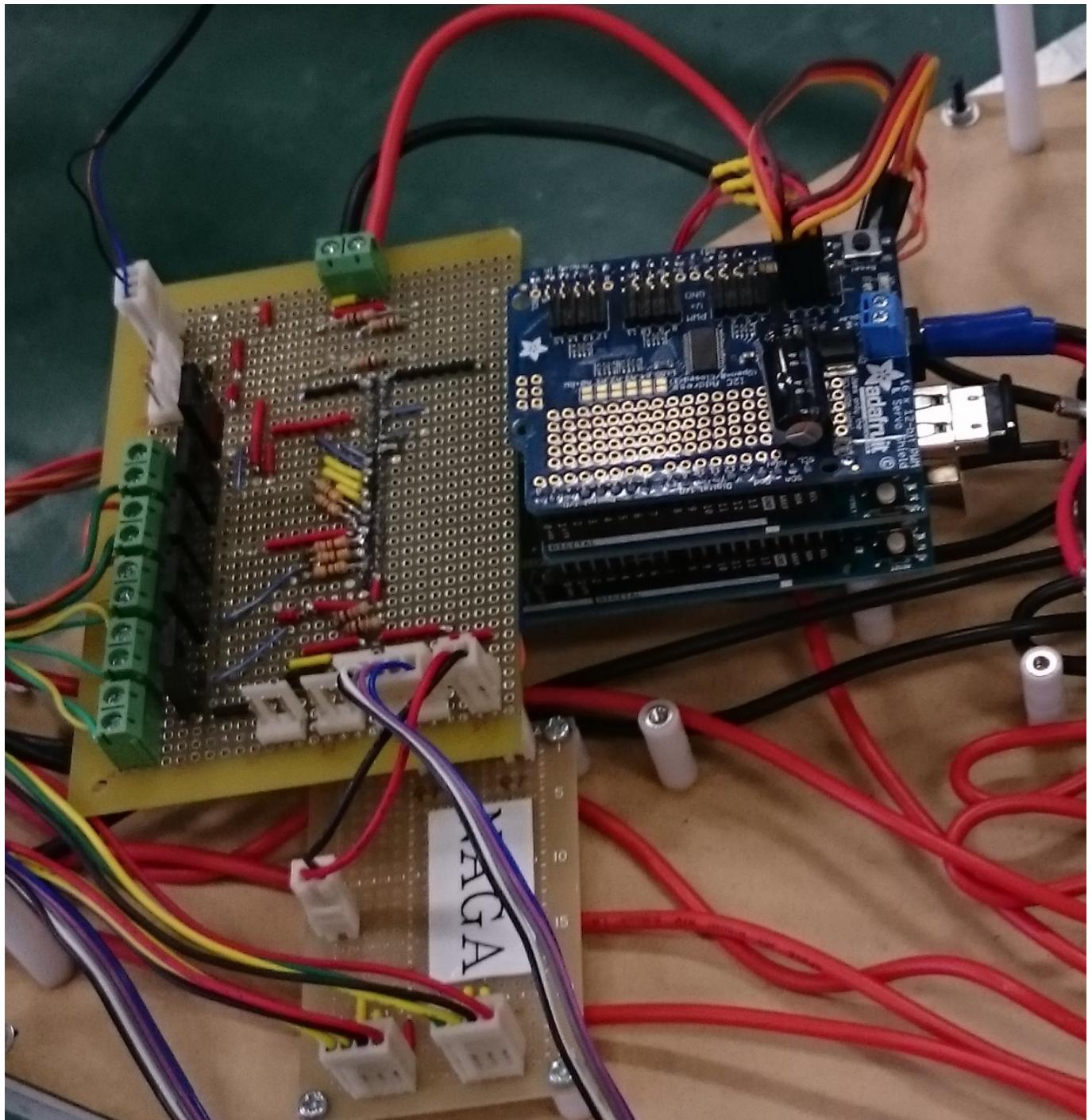


図 2.2 制御基板

2.6 ポボコン shield

GR-SAKURA は USBhost や SD カードスロットが既存であるため、シリアル通信の規格を RS485 に変換するトランシーバと各種 IO を外側に出し接続しやすくするものを作成する。ポボコンシールドを設計するにあた

り、GR-SAKURA の詳細な寸法を調査した。GR-SAKURA の詳細寸法を図 2.3 に示し、ロボコンシールドの回路図を図 A.6 に示す。

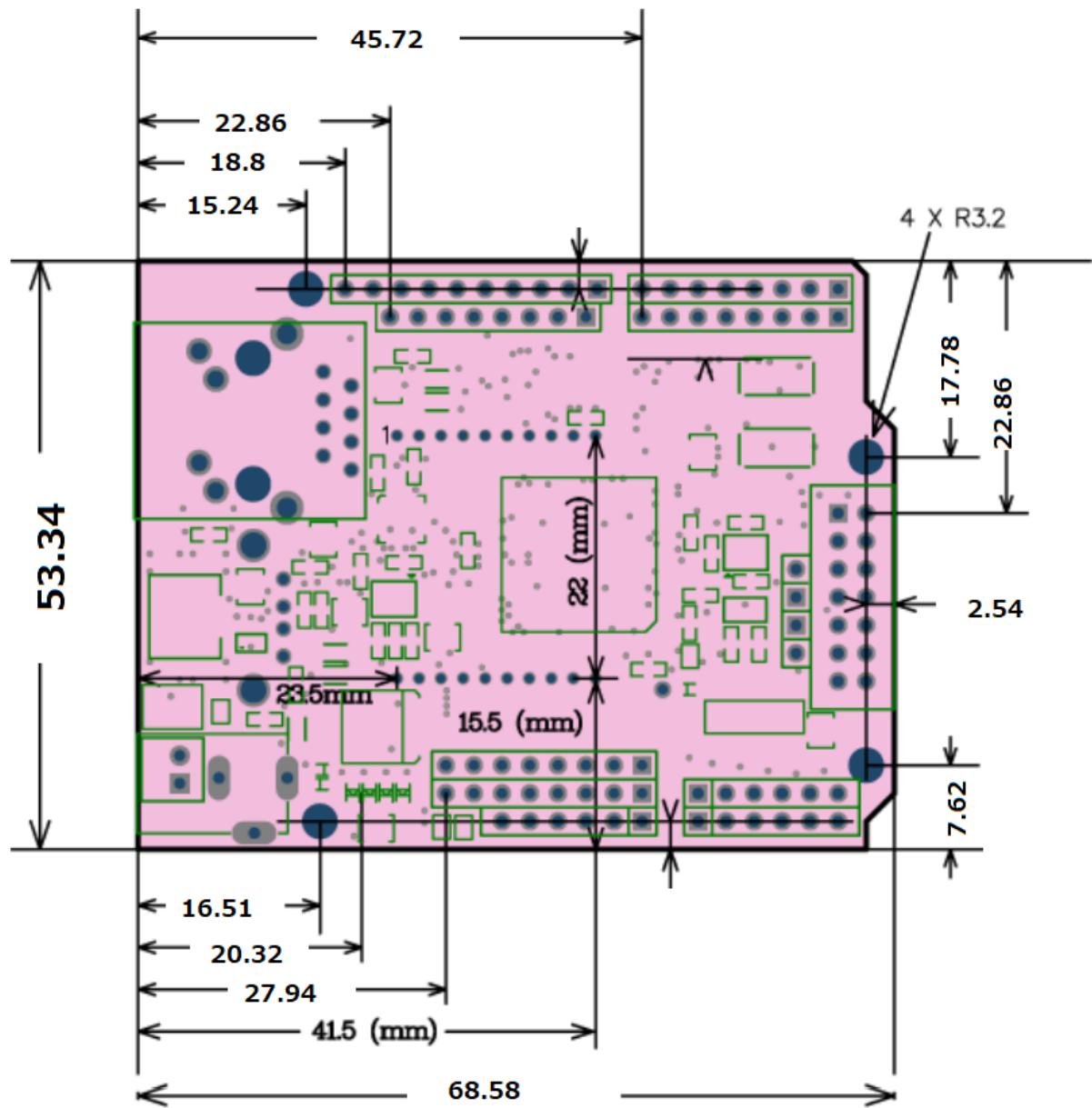


図 2.3 GR-SAKURA 詳細寸法

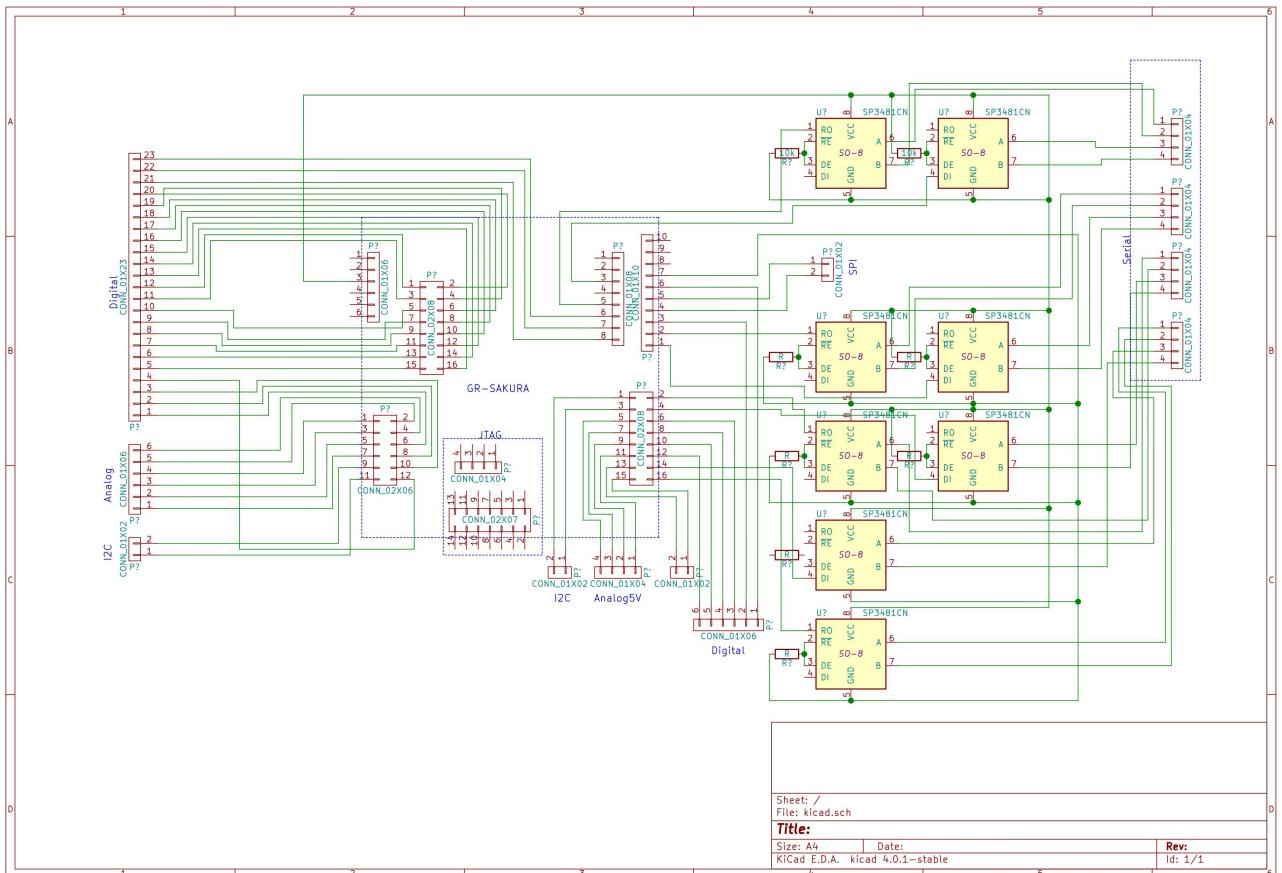


図 2.4 シールド回路図

GR-SAKURA は Serial 通信に使用できるポートが複数あるが、SPI 通信などと兼ねているものがある。そのため使用すると予想される最大数の 4つを Serial 通信に使用できるようにした。また、今年度の Serial 通信の規格は RS232C を使用してノイズに悩まされたため、ノイズ対策として RS485 に変換するトランシーバを搭載した。RS-232C・RS-422A・RS-485 は、EIA (Electronic Industries Association: 米国電子工業会) の通信規格である。中でも RS-232C は、通信規格の中でも用途を問わず多く普及し、パソコンにも標準で搭載されおり、モ뎀やマウスの接続によく利用されている。センサやアクチュエータの中にも、これらのインターフェイスを持ち、通信により制御可能なものが多くの存在する。RS485 は差動信号を採用している。2 線間の電圧の違いによってどんなデータを伝送するか表現している。シリアル

ル通信の規格を次に示す。

2.6.1 RS-232C

RS-232Cでは、1本の信号線でデータを伝送するシングルエンド伝送を用いており、データを伝送するとき信号線の電圧が-5~-15Vのとき「1」となり、電圧が+5~+15Vのとき「0」となる。電圧レベルで伝送するためノイズに弱く、規格ではケーブルの最大長は15mとなっている。常に外部から電源を供給しているデスクトップパソコンでは高い出力電圧を得られますが、バッテリ等の低い電源で動作するノートパソコンの場合、出力電圧も低くなります。ノートパソコンの中にはRS-232Cのポートが搭載されていても使用しない設定になっているものもある。この場合、RS-232Cポートを使用するよう設定を行う必要がある。

2.6.2 RS-422A

ケーブルには2本の電線をより合わせて対にしたツイストペアケーブルが用いられている。2本の電線をより合わせることで、電流が流れるときに生じる磁気を打ち消すような仕組みになっている。その中でシールドしているものをSTP(Shielded Twisted Pair cable)、していないものをUTP(Unshielded Twisted Pair cable)という。コネクタの形状は規格化されていないが、実際に使用されるものではD-SUB9PやD-SUB25Pが多く、ミニDIN8Pや端子台も使われる。

RS-422では2本の信号線でデータを伝送する差動伝送(ディファレンシャル)を用いており、「+」と「-」の2本の信号線でデータを伝送し、+信号線の電圧が-信号線の電圧より高い場合を「1」、低い場合を「0」と判断する。このように信号線2本の電位差で信号を判断しているため、電線延長による電圧の減衰は「+」と「-」の両方で起こり、「+」と「-」の電位差には影響が少なくなる。0.3Vの電位差で信号と認識するため、比較的ノイズに強く長距離延長が可能となることから、工場のようなノイズが多い環境で

の使用に適している。規格ではケーブルの最大長は1.2kmとなっている。

2.6.3 RS-485

RS-422の上位互換である。RS-422が1:nに対し,RS-485はn:mの接続に対応しており,1対の信号線上に最大32台まで接続できるマルチドロップ方式になっている。複数の機器を接続する場合,回路の終端で信号が反射し,信号波形が乱れてしまうことがある。このような現象が繰り返し行われていると通信速度を上げることができなかったり,正常なデータ伝送ができなくなることがある。そこでRS-485回線には信号線の終端に信号線が持つインピーダンスと等しい抵抗(終端抵抗)を入れることで伝送中の反射を抑え,信号波形の乱れを少なくすることができる。コネクタの形状は規格化されていないが,実際に使用されるものではRJ-45やD-SUB9P,端子台等がある。その他の仕様はRS-422に準拠する。

表 2.5 Serial 通信一覧

パラメータ	RS232C	RS422A	RS485
伝送モード	シンプレックス	マルチポイント シンプレックス	マルチポイント マルチプレックス
最大接続台数	1 ドライバ 1 レシーバ	1 ドライバ 10 レシーバ	32 ドライバ 32 レシーバ
最大伝送速度	20kbps	10Mbps	10Mbps
最大ケーブル長	15m	1200m	1200m
動作モード	シングルエンド(非 平衡型)	ディファレンシャル (平衡型)	ディファレンシャル (平衡型)
特徴	短距離 全2重 1:1の接続	長距離 全2重半2 1:Nの接続	長距離 全2重半2重 N:Mの接続

第3章

ログシステムの開発

3.1 ログシステムの要求項目

ログシステムに必要な項目を挙げる。

1) 指定した値の記録

2) タイムスタンプ

1) 指定した値の記録はログシステムの主となる部分である。製作したロボットの電圧や制御指令値など、様々な値を記録することによりトラブルが発生した時に原因を明らかにする手助けとなる。しかし、ログとして記録する値が変更されたり増えるたびにprint文などのプログラムの中身を書き換えるのは不便である。そこで必要な値のリストをユーザーが用意し、そのリストを読みリストで指定された値を1レコードとして記録する。

2) タイムスタンプはある出来事が発生した時間を示す文字列のことである。ただただ値を記録していくのではただの数字の羅列になり、どこで何が起きたのかわからない。そこで1レコードごとにタイムスタンプをつけることによりその値がいつのものなのか明らかにする。

3.2 ログシステムのフローチャート

プログラムのフローチャートを図3.1に示す。

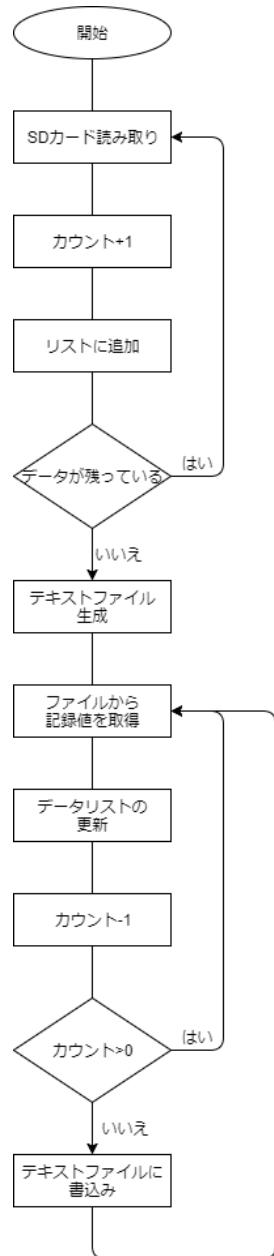


図 3.1 フローチャート

3.3 ログシステムのヘッダファイル

ヘッダファイルはファイル読み取り時、作成されたデータの記号リストの記号にデータの値を代入するものである。データのリストを1レコードとし、タイムスタンプを最後に加えてテキストファイルに書き込みする。その後、リストのデータの値を更新し書き込みを繰り返す仕様になっている。ヘッダファイルを ListingA.7 に示す。

3.4 ログシステムプログラム

ログシステムのプログラムを付録 A.5 に示す。

第4章

結言

4.1 本研究のまとめ

今回製作した標準制御システム及びログシステムを使用すればプログラムまでにかかる時間や、トラブルの対応にかかる時間が短縮される。よってロボットの完成度が上がり、多くの練習時間の得ることができ多くの勝利につながる。本研究室に所属するのは機械工学科の学生であり、ほとんどプログラムに関する知識がない。加えて、少ないプログラムの授業にも多くの学生が苦手意識を持っている。しかし、近年のロボットはコンピュータ上でプログラムを走らせて制御しているものばかりであり、ロボコンとプログラムを切り離すことはできない。そのため、プログラムに苦手意識を持っている機械科学生でもプログラミングができるようなマニュアルを作成する。

4.2 今後の課題

ロボコンシールドに関しては回路設計までしかできておらず、実際に使用するにはプリントパターンなどを作成し、基板を製作する必要がある。また、マニュアルに関しては私が今年度得た知識がほとんどであり、今後ロボコン研究室に所属する学生が新たに得た知識などを書き足してもらいより実用的なマニュアルにしてもらいたい。

参考文献

- [1] e2studio [https://www.renesas.com/ja-jp/products/software-tools/tools/ide/e2studio.html]
- [2] GR-SAKURAe2studio [https://japan.renesasrulz.com/gr_user_forum_japanese/m/mediagallery/144]
- [3] IDE for GR [http://gadget.renesas.com/ja/product/ide4gr.html]
- [4] スケッチリファレンス [http://gadget.renesas.com/ja/reference/sakura/library_sdmmc.html]
- [5] SAKURABOARD [http://sakuraboard.net/index.html]
- [6] 特殊電気回路 RXduino [http://rx.tokudenkairo.co.jp/manual.html]
- [7] Web コンパイラ [http://tool-cloud2.renesas.com/]
- [8] 林 和孝.(2014).RaspberryPi で遊ぼう

謝辞

本論文作成にあたりテーマの決定, 研究の考え方, 方法など長期にわたって厳しくも熱意のあるご指導, ご鞭撻していただいた, 伊藤恒平教授, 林道大教授に厚く御礼申し上げます。

また, 日常の議論を通じて多くの知識や示唆を頂いたロボコン研究室の皆様に感謝します。

NHK 高専ロボコンのスタッフの方々, その他, 助けていただいた多くの皆様に心から感謝しております。ありがとうございました。

付録 A

制御システムマニュアル

A.1 開発環境導入

GR-SAKURA には開発環境が複数用意されている。オンライン開発環境のがじえっとるねさす Web コンパイラ、オンライン環境の e2studio と図 A.2 IDEforGR がある。e2studio は導入方法が複雑だったため、図 A.1 ArduinoIDE のような GUI(Graphical User Interface) で作られている IDEforGR を使うことにした。SD.h や sdmmc.h などのライブラリを入れることはできたが、ライブラリのサンプルプログラムのコンパイルが通らなかった。エラー文を見てみるとどうやらヘッダファイルでエラーが出ている模様。ヘッダファイルのピンの設定をしているところで、Arduino の CPU で分けられて書かれているが、GR-SAKURA の CPU である RX63N はないためエラーで出ていた。GR-SAKURA を Arduino と認識させるための rxduino.h というライブラリが必要である。特殊電気回路というところが FreeRXduino.h というライブラリを配布していた。オンラインの Web コンパイラを使ってプログラミングする。

The screenshot shows the Arduino IDE interface. The title bar reads "sketch_oct28a | Arduino 1.6.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu is a toolbar with icons for save, upload, and search. The main code editor window displays the following sketch:

```
1 void setup() {
2     // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7     // put your main code here, to run repeatedly:
8
9 }
```

The status bar at the bottom right shows "LinkIt ONE on COM24".

図 A.1 ArduinoIDE

The screenshot shows the IDEforGR application window. The title bar reads "sketch_oct30a | IDE for GR 1.02". The menu bar includes "ファイル" (File), "編集" (Edit), "スケッチ" (Sketch), "ツール" (Tools), and "ヘルプ" (Help). The toolbar contains icons for file operations like new, open, save, and upload. The main code editor window displays the following Arduino-style sketch:

```
void setup() {
  // put your setup code here, to run once:
  Serial4.begin(9600);

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Below the code editor is a status bar with a progress bar and the text "コンパイル終了。" (Compile finished.). The terminal window at the bottom shows the compilation results:

```
Binary sketch size: 51,844 bytes (of a 8,388,608 byte maximum) - 0% used
Minimum Memory usage: 24,748 bytes (of a 3,145,728 byte maximum) - 0% used
```

The bottom right corner of the terminal window shows "GR-LYCHEE on COM97".

図 A.2 IDEforGR

A.2 モータ制御

今年度はロボットを3台制作したが、駆動方法は2輪差動と3輪オムニの2種類であり、特に速さを重視した2輪差動の制御に苦労した。ロボットを目的地に素早く移動させるために2輪差動を採用したのだが、操作に用いたDualShock3 図A.3から得られる数値をそのまま使用した場合、モータの回転数が最高値に達するのは直進時ではなく旋回時でありコントローラから受け取った数値を補正する必要があった。また、Arduinoから送っていた制御値がノイズの影響を受けおかしな値になっていたりした。ロボコンのロボットにおいて駆動部のモータ制御は必要不可欠なため、モータ制御を詳しく説明する。



図 A.3 DualShock3

A.2.1 2輪差動制御

今回は DualShock3 のアナログスティックを駆動部の操作に用いた。DualShock3 のアナログスティックの可動域は円形をしているが、数値として出力する値は正方形になっている。そのため最大の値が出力されるのはスティックをまっすぐ前に倒した時ではなく、斜めに倒した時になる。今回はロボットの速度を重視した作戦を採用しており、直進時に最高速が出せないのは問題があった。そこで、コントローラから出力された値にプログラムで正方形からひし形になるように補正をかけた。補正のイメージ図を図 A.4 に示す。これにより直進で最高速ができるようになった。

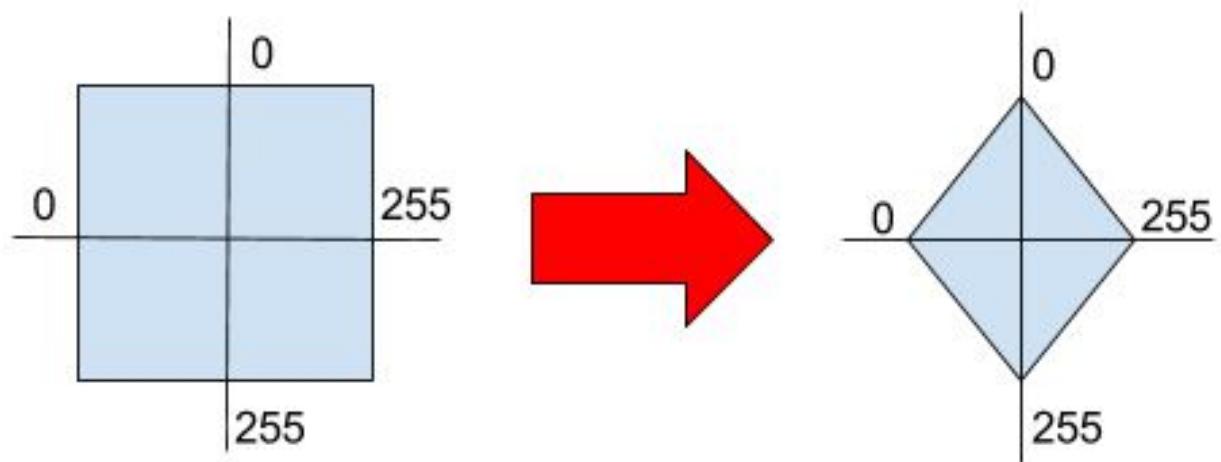


図 A.4 補正イメージ

実際のプログラムを下に示す。

Listing A.1 補正 Program

```
1  float d=1.3;
2  float x,y;
3  int mm;
4
5  ly=(float)PS3.getAnalogHat(LeftHatY)-127.5;
6  y=ly/127.5;
7
8  lx=(float)PS3.getAnalogHat(LeftHatX)-127.5;
9  x=lx/127.5;
10
11 if (y<=0.0 && 0.0<=x) { //右上
12     w=(y+1.0)*x;
```

```

13      }
14  else if (y<=0.0 && x<=0.0) { //左上
15      w=(y+1.0)*x;
16  }
17  else if (0.0<=y && 0.0<=x) { //右下
18      w=-(y-1.0)*x;
19  }
20  else if (0.0<=y && x<=0.0) { //左下
21      w=-(y-1.0)*x;
22  }
23  else{
24      w=0.0;
25  }
26
27  w=w*500.0;
28  w=w*0.8;
29  v=y;
30  v=v*500.0;
31  v=v*0.8;
32
33  lm=(int) ( (v-(d*w)/2.0) * 0.9 );
34  lm=lm+500;
35  mm = motmod( lm , 2 );
36  lm=1000-mm;
37  //左モータ指令確定
38
39  rm=(int) ( (v+(d*w)/2.0) * 0.9 );
40  rm=rm+500;
41  mm = motmod( rm , 1 );
42  rm = mm;
43  //右モータ指令確定

```

A.2.2 3 輪オムニ制御

3 輪 オ ム ニ の 制 御 プ ロ グ ラ ム を 下 記 に 示 す.

Listing A.2 補正 Program

```

1  Vx = (PS3.getAnalogHat(RightHatX))-127;
2      if ((-10<Vx) && (Vx<10)){
3          Vy=0;
4      }
5      if (Vx < -100) {
6          Vx=-100;
7      }
8      if (Vx > 100){
9          Vx= 100;
10     }
11
12     Vy = (PS3.getAnalogHat(RightHatY))-127;
13     if ((-10<Vy) && (Vy<10)){
14         Vy=0;
15     }

```

```

16     if (Vy < -100) {
17         Vy=-100;
18     }
19     if (Vy > 100) {
20         Vy= 100;
21     }
22
23     vf = (-1)*Vx;
24     vl = (int)(0.5*Vx-0.866*Vy);
25     vr = (int)(0.5*Vx+0.866*Vy);
26
27     nvf = (int)(vf*100 / (float)vMAX);
28     nvl = (int)(vl*100 / (float)vMAX);
29     nvr = (int)(vr*100 / (float)vMAX);
30
31     pwm_f = (int)(nvf*pMAX/100);
32     if(pwm_f>pMAX) {
33         pwm_f = pMAX;
34     }
35     pwm_l = (int)(nvl*pMAX/100);
36     if(pwm_l>pMAX) {
37         pwm_l = pMAX;
38     }
39     pwm_r = (int)(nvr*pMAX/100);
40     if(pwm_r>pMAX) {
41         pwm_r = pMAX;
42     }
43
44     v1=(int)((pwm_f*5.55+999)/2);
45     v2=(int)((pwm_l*4.044+999)/2);
46     v3=(int)((pwm_r*4.044+999)/2);
47     if (right==1){
48         v1=v1-mROT;
49         v2=v2-mROT;
50         v3=v3-mROT;
51     }
52     if (left==1){
53         v1=v1+mROT;
54         v2=v2+mROT;
55         v3=v3+mROT;
56     }
57
58     sprintf(m1,"%d%d%d\r\r",v1,v1,v1);
59     sprintf(m2,"%d%d%d\r\r",v2,v2,v2);
60     sprintf(m3,"%d%d%d\r\r",v3,v3,v3);

```

A.2.3 ノイズ対策

今年度のロボット製作で一番悩まされたのはおそらくノイズである。ノイズとは電気信号の無作為な変動であり、全ての電気回路に存在す

る。電子機器が発生するノイズは様々で、その発生原因もいくつもある。信号線をアルミホイルで巻くなど色々な方法でノイズ対策を行ったが、ここではプログラムに関するノイズ対策を説明する。プログラム上で行ったノイズ対策はモータドライバに送る3桁の指令値を3回送り多数決で正しい指令値を判定するというものである。

A.3 映像ストリーミングシステム

カタパルトで紙飛行機を発射する際の照準として搭載したカメラ映像をヘッドアップディスプレイに映し出すもの。システムの概要を図A.5に示す。

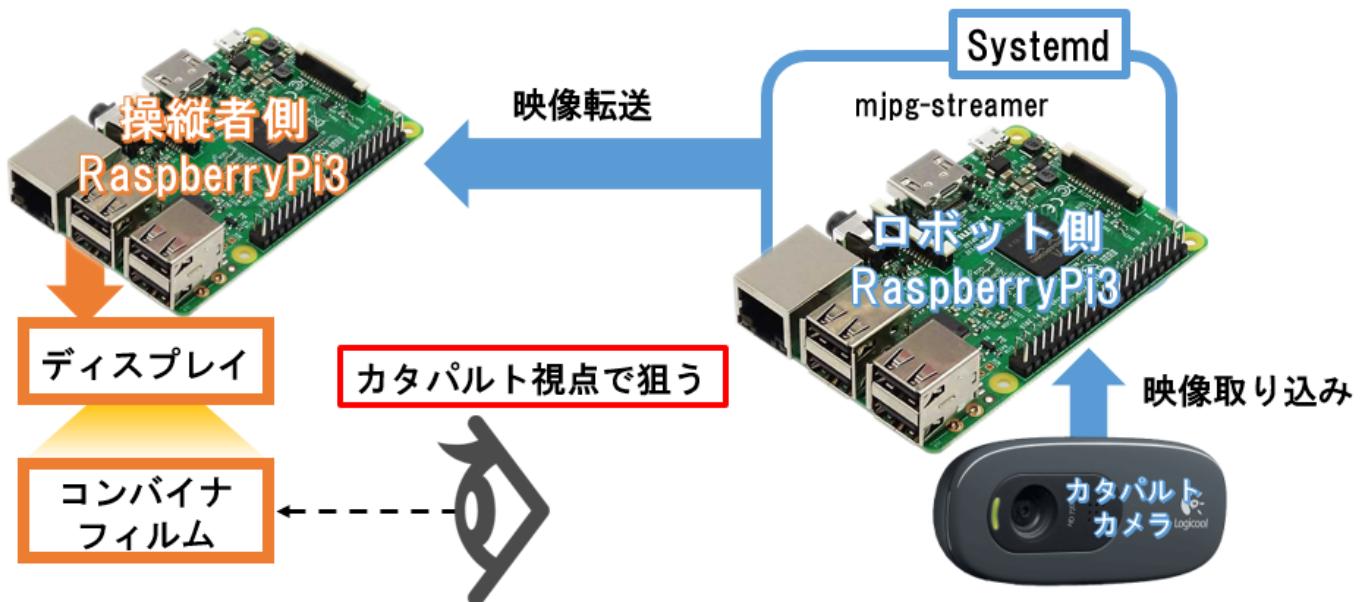


図 A.5 映像ストリーミングシステム

A.3.1 映像ストリーミングソフトウェアのインストール

`mjpg-streamer` というソフトを使用して映像をストリーミングする。まず, `mjpg-streamer` に必要なパッケージである `subversion`、`libjpeg-dev`、`imagemagick`、をインストールする。

<code>sudo apt-get install subversion libjpeg-dev imagemagick</code>	次に, <code>mjpg-streamer</code> をダウンロード
--	--

し ビ ル ド す る.

```
svn co https://svn.code.sf.net/p/mjpg-streamer/code/mjpg-streamer mjpg-streamer
```

```
cd mjpg-streamer
```

```
make
```

mjpg-streamer を起動する. この状態だとネットワーク上に映像をストリーミングしていることになる.

web ブラウザで RaspberryPi のポート 8080 にアクセスすると映像を確認できる.

A.3.2 自動起動

ロボットに搭載した RaspberryPi を外部から操作するのは困難だったため, RaspberryPi が起動したと同時に映像ストリーミングが開始されるようにした. まず,mjpg-streamer を起動するための ListingA.3 シェルスクリプト stream.sh を作成する.

```
nano stream.sh
```

Listing A.3 シェルスクリプト

```
1 #!/bin/sh
2
3 # This is Web—streaming server start up script.for raspi
4 # No warranty.
5
6 # Config
7 PORT="8081"
8 ID="ベーシック認証の ID"
9 PW="ベーシック認証のパスワード"
10 SIZE="640x480"
11 F_RATE="15"
12 MJPG_STREAMER=/usr/local/bin/mjpg_streamer
13
14 export LD_LIBRARY_PATH=/usr/local/lib
15 $MJPEG_STREAMER \
16 -i "input_uvc.so-f$F_RATE-r$SIZE-d/dev/video0-y" \
17 -o "output_http.so-w/usr/local/www-p$PORT-c$ID:$PW" -b
```

ファイルを実行可能なパーティションに変更するコマンドを実行する.

```
chmod 755 stream.sh
```

次に,systemd を使ってシェルスクリプトを自動起動させるための service

ファイル stream.service を/etc/systemd/system に作成する。

Listing A.4 service ファイル

```
1 [Unit]
2 Description=Movie Streaming
3
4 [Service]
5 ExecStart=/home/pi/systemd/stream.sh
6 Restart=always
7 Type=simple
8
9 [Install]
10 WantedBy=multi-user.target
```

自動起動を有効化するためのコマンドを実行する。

`sudo systemctl enable stream.service` 有効化されたか確認するには,

`sudo systemctl status stream.service` というコマンドを実行し,enabled と表示されれば有効化されている。

A.3.3 アドホック通信

mjpg-streamer をそのまま使ってもネットワーク環境がないと映像を見ることができない。そのため 2 台の RaspberryPi を 1 対 1 でアドホック通信させ、ネットワーク環境がない状況でも映像ストリーミングを可能にさせる。アドホック通信をするために RaspberryPi の/etc/network/interfaces を ListingsA.5 のように書き換える。

Listing A.5 interfaces

```
1 auto wlan0
2 iface wlan0 inet static
3 address 192.168.12.1
4 netmask 255.255.255.0
5 wireless-channel 1
6 wireless-mode ad-hoc
7 wireless-essid pi
8 wireless-key 01234567890123456789012345
```

受信側も同じように interfaces を書き換えるが, address の下一桁を違う数字に変える必要がある。wireless-key は任意の数字でいいが, 26 桁の 16 進数である必要がある。アドホック通信ができているか確認するには ping コマンドを使用する。

```
ping 192.168.12.1
```

このように ping のあとに相手の address を入れる。

A.4 プログラム

記録したいデータを読み取り,SD カードに書き込むプログラムを Listing A.6 に示す。

Listing A.6 ログ Program

```
1 #include<Arduino.h> //
2 #include<SD.h> //
3 #include<sdmmc.h> //
4 #include<data.h> //記録するデータリスト
5
6 unsigned long time=0;
7 int i,j;
8 float data;
9
10
11 void loging(void){
12     file = SD.open("log.txt", FILE_WRITE)           ;//書き込み用のファイル生成
13
14     while(data_seq[i] >= countA){                  //記録するデータの種類だけループ
15         while(1){
16             if(dataB_seq[j] == data_seq[i]){
17                 check();
18                 file.print(dataB_seq[j]); //データの値を書き込み
19                 break;
20             }
21             else{ j=j++;
22                 }
23         }
24         i=i++;
25     }
26     file.println(time); //タイムスタンプを書き込み改行
27 }
28
29
30 void setup() {
31     File file = SD.open("data.txt", FILE_READ)      ;//テキストファイルを開く
32     while(file.available()){ //dataの中身の数だけループ txt
33         data_seq[i]=file.read()                      ;//記録したいデータの種類がいくつあるか
34         読む
35         countA=countA++; //記録するデータの種類のカウント
36         i=i++          ;//
37     }
38     time = millis();
39 }
```

```
41
42 void loop(){
43     logging(); //関数呼び出し
44 }
```

各種 IO を数字に置き換え、指定されたデータの値を代入するためのヘッダファイル。

Listing A.7 ヘッダファイル data.h

```
1 //データ表
2 RXD0 = 0 //PIN_P21
3 TXD0 = 1 //PIN_P20
4 Temp = 2 //PIN_P22
5 Voltage = 3 //PIN_P23
6 PIN_P24 = 4
7 PIN_P25 = 5
8 PIN_P32 = 6
9 PIN_P33 = 7
10 PIN_PC2 = 8
11 PIN_PC3 = 9
12 PIN_PC4 = 10
13 PIN_PC6 = 11
14 PIN_PC7 = 12
15 PIN_PC5 = 13
16 PIN_P40 = 14
17 PIN_P41 = 15
18 PIN_P42 = 16
19 PIN_P43 = 17
20 PIN_P44 = 18
21 PIN_P45 = 19
22 PIN_P46 = 20
23 PIN_P47 = 21
25 PIN_PC0 = 22
26 PIN_PC1 = 23
27 PIN_P50 = 24
28 PIN_P51 = 25
29 PIN_P52 = 26
30 PIN_P53 = 27
31 PIN_P54 = 28
32 PIN_P55 = 29
33 PIN_P12 = 30
34 PIN_P13 = 31
35 PIN_P14 = 32
36 PIN_P15 = 33
37 PIN_P16 = 34
38 PIN_P17 = 35
39 PIN_PD0 = 36
40 PIN_PD1 = 37
41 PIN_PD2 = 38
42 PIN_PD3 = 39
43 PIN_PD4 = 40
```

```

44 PIN_PD5 = 41
45 PIN_PD6 = 42
46 PIN_PD7 = 43
47 PIN_PE0 = 44
48 PIN_PE1 = 45
49 PIN_PE2 = 46
50 PIN_PE3 = 47
51 PIN_PE4 = 48
52 PIN_PE5 = 49
53 PIN_PE6 = 50
54 PIN_PE7 = 51
55 PIN_P07 = 52
56 PIN_P05 = 53
57 PIN_P35 = 54
58 PIN_PJ3 = 55
59
60
61
62 //データの値代入
63 void check(void){
64     if(j==0){
65         data=PIN_P21 //RXD0
66     }
67     else if(j==1){
68         data=PIN_P20 //TXD0
69     }
70     else if(j==2){
71         data=PIN_P22
72     }
73     else if(j==3){
74         data=PIN_P23
75     }
76     else if(j==4){
77         data=PIN_P24
78     }
79     else if(j==5){
80         data=PIN_P25
81     }
82     else if(j==6){
83         data=PIN_P32 //TXD6
84     }
85     else if(j==7){
86         data=PIN_P33 //RXD6
87     }
88     else if(j==8){
89         data=PIN_PC2 //SPI_CS3
90     }
91     else if(j==9){
92         data=PIN_PC3
93     }
94     else if(j==10){
95         data=PIN_PC4 //SPI_CS0
96     }
97     else if(j==11){

```

```

98           data=PIN_PC6 //SPI_MOSI
99       }
100      else if(j==12){
101          data=PIN_PC7 //SPI_MISO
102      }
103      else if(j==13){
104          data=PIN_PC5 //SPI_CLK
105      }
106      else if(j==14){
107          data=PIN_P40 //dataD0
108      }
109      else if(j==15){
110          data=PIN_P41 //dataD1
111      }
112      else if(j==16){
113          data=PIN_P42 //dataD2
114      }
115      else if(j==17){
116          data=PIN_P43 //dataD3
117      }
118      else if(j==18){
119          data=PIN_P44 //dataD4
120      }
121      else if(j==19){
122          data=PIN_P45 //dataD5
123      }
124      else if(j==20){
125          data=PIN_P46 //dataD6
126      }
127      else if(j==21){
128          data=PIN_P47 //dataD7
129      }
130      else if(j==22){
131          data=PIN_PC0 //SPI_CS1
132      }
133      else if(j==23){
134          data=PIN_PC1 //SPI_SC2
135      }
136      else if(j==24){
137          data=PIN_P50 //TXD2
138      }
139      else if(j==25){
140          data=PIN_P51
141      }
142      else if(j==26){
143          data=PIN_P52 //RXD2
144      }
145      else if(j==27){
146          data=PIN_P53
147      }
148      else if(j==28){
149          data=PIN_P54
150      }
151      else if(j==29){

```

```

152           data=PIN_P55
153       }
154   else if(j==30){
155       data=PIN_P12 //IRQ2 / RXD2
156   }
157   else if(j==31){
158       data=PIN_P13 //IRQ3
159   }
160   else if(j==32){
161       data=PIN_P14 //IRQ4 / DPUPE
162   }
163   else if(j==33){
164       data=PIN_P15 //IRQ5 / SDcdatard
165   }
166   else if(j==34){
167       data=PIN_P16 //IRQ6 / VBUS
168   }
169   else if(j==35){
170       data=PIN_P17 //IRQ7
171   }
172   else if(j==36){
173       data=PIN_PD0
174   }
175   else if(j==37){
176       data=PIN_PD1
177   }
178   else if(j==38){
179       data=PIN_PD2
180   }
181   else if(j==39){
182       data=PIN_PD3
183   }
184   else if(j==40){
185       data=PIN_PD4
186   }
187   else if(j==41){
188       data=PIN_PD5
189   }
190   else if(j==42){
191       data=PIN_PD6
192   }
193   else if(j==43){
194       data=PIN_PD7
195   }
196   else if(j==44){
197       data=PIN_PE0
198   }
199   else if(j==45){
200       data=PIN_PE1
201   }
202   else if(j==46){
203       data=PIN_PE2
204   }
205   else if(j==47){

```

```
206             data=PIN_PE3
207         }
208     else if(j==48){
209         data=PIN_PE4
210     }
211     else if(j==49){
212         data=PIN_PE5
213     }
214     else if(j==50){
215         data=PIN_PE6
216     }
217     else if(j==51){
218         data=PIN_PE7
219     }
220     else if(j==52){
221         data=PIN_P07
222     }
223     else if(j==53){
224         data=PIN_P05 //DdataI
225     }
226     else if(j==54){
227         data=PIN_P35 //NMI
228     }
229     else if(j==55){
230         data=PIN_PJ3
231     }
232     else {
233         Seridatal.print("Loging↓Error");
234     }
```

A.5 シールド回路図

図 A.6 シールド回路

