目次

第 1 章

はじめに

1.1　研究の背景

1.1.1　2016 年度「ロボット・ニューフロンティア」ルール説明

2016 年度に行われた NHK 高専ロボコンは「ロボット･ニューフロンティア」という題目で港町から海を渡り、新大陸を目指すというものである。使用ロボット台数は無制限、港町で灯台を組み上げ、床に接触しないよう海を渡り、丘の上に砦を築く競技である。今年は丘に積み上げた砦の高さを競い合うものであった。図 1.1 はこの競技のフィールドである。

1.2　研究の目的

今回ロボコン競技に使用した積み込みロボットの移動手段となっている全方向移動可能なメカナムホイールにジャイロセンサを加えることで継続的に直進を可能にする。

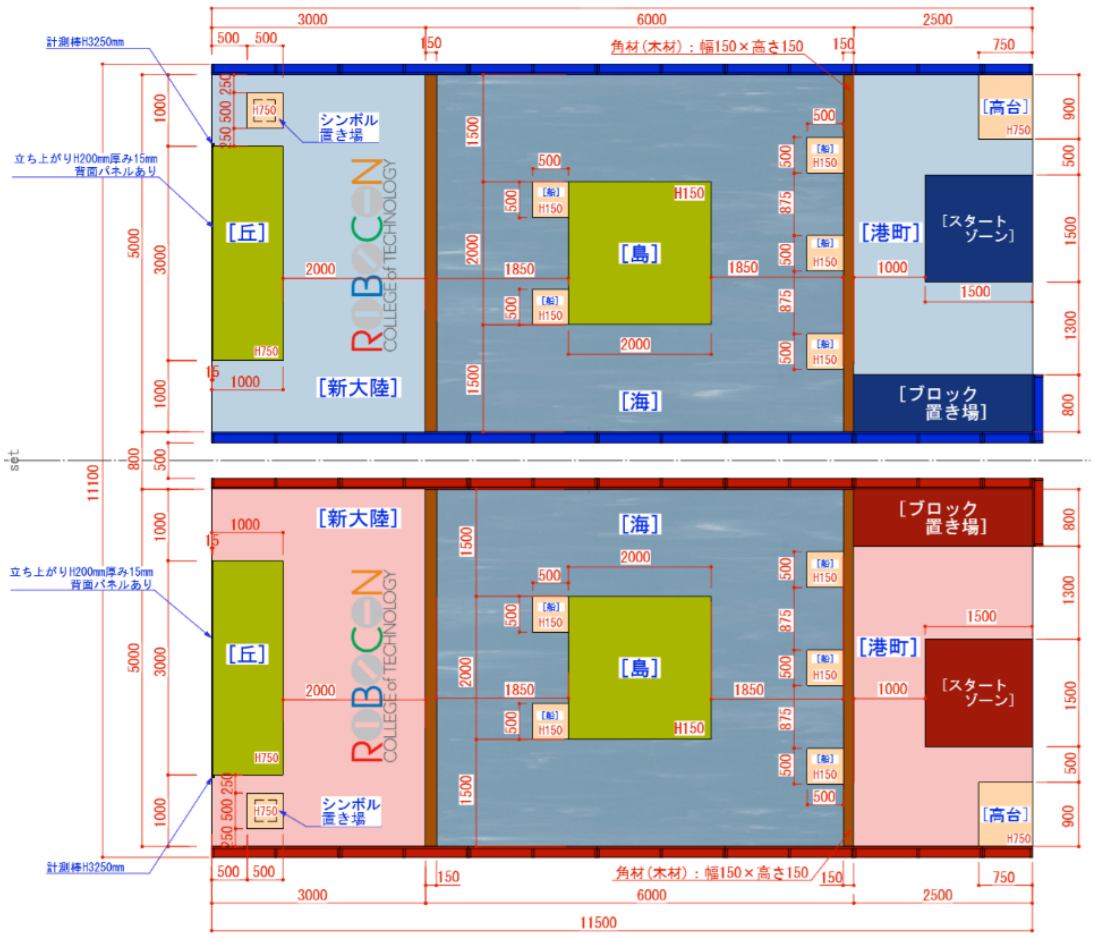

計測棒H3250mm
立ち上がりH200mm厚み15mm
背面パネルあり
角材（木材）：幅150×高さ150
シンボル
置き場
[丘]
[新大陸]
[島]
[海]
[港町]
[スタート
ゾーン]
[高台]
[ブロック
置き場]
H750
H150
ROBOCEN
COLLEGE of TECHNOLOGY
set
11100
11500

図 1.1　競技フィールド

第 2 章

メカナムホイールを採用した理由

何故、今回のロボコンでメカナムホイールを採用したかを以下に示す。

- 出題された多くの課題の中で制限時間 3 分は非常に短く、丘に箱を積み上げるだけでも困難になる。
- メカナムホイールを使用することで 2 輪駆動よりも圧倒的に速く課題をクリアすることができる。

2.1　二輪駆動ロボとメカナムロボの比較

メカナムホイールは 2 輪駆動よりも少ない工程で動作が可能である。例としてロボットの向きを変えずに右に移動させたい場合、2 輪駆動ロボットでは 90° 右旋回してから前進、正面を向き直す為 90° 左旋回と少なくとも 3 工程必要となる。これがメカナムホイールの場合は横に並行移動の 1 工程だけで終えることができる。また、メカナムホイールは移動しながらの旋回も可能で 1 工程の時間の中で 2 工程分の動きも可能なのである。このように、メカナムホイールを用いることで 2 輪駆動よりも少ない工程数で時間の短縮が可能なのである。

2.2　三分間で必要とされる動き

メカナムホイールを使っている積み込みロボットが競技中の三分間で必要とされる動きを以下に示す。

2.2.1　箱の持ち運び

特定の位置に並べられている箱を持ち上げ、距離 4 m ほど運搬する必要がある。この時の課題は箱を拾う際の位置調整である。箱は図 2.1 のように 400mm × 300mm × 200mm で 100mm の等間隔に設置されている。その 100mm の間を図 2.2 のように幅約 72mm のハンドを滑り込ませ、更に側面から見た図 2.3 のように縦は中心で横が先端から約 110mm 前後の位置にハンドの中心がくるように配置しなければならない。そのためメカナムホイールの全方向移動による微調整が 2 輪駆動より有効となる。

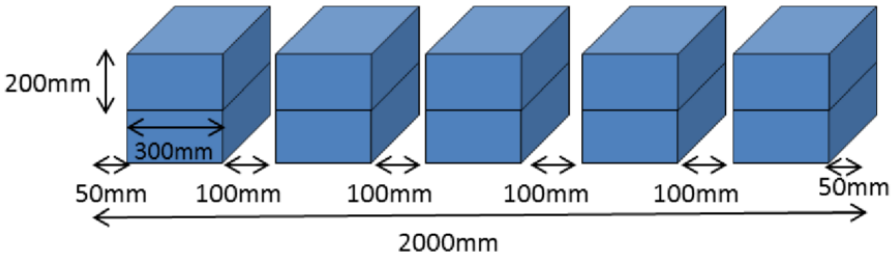


図 2.1　競技中の箱配置

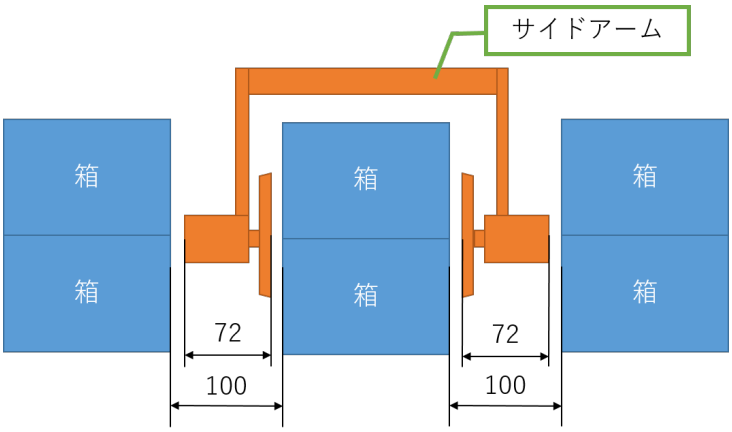


図 2.2　箱の幅とアームの幅

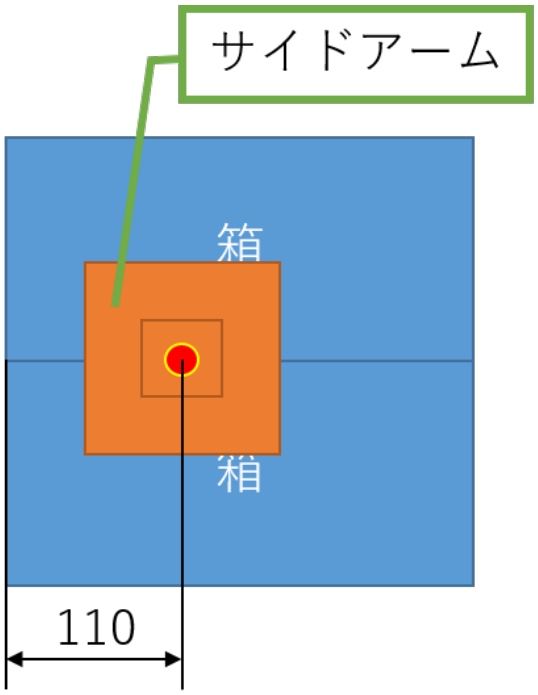


図 2.3　箱回収側面位置

2.2.2　塔建て

運んだ箱を高さ 750mm の台の上に持ち上げて、下段が大きく上段が小さくピラミッド型のように組み立てる。これは「高台」と「丘」によって異なり、「高台」では図 2.4 のように 2 列 1 列 1 列の順に組み立てる。「丘」では図 2.5 のように下段が上段よりも両側 25mm 程大きければいくらでも積んで良いとなっている。製作した積み込みロボットは「丘」であれば動かずに積み上げることができる。だが「高台」の場合は 1 列に 2 段積み上げる必要があるため真横に並行移動できると時間短縮になる。

2.2.3　橋渡り

架橋ロボが設置した橋の上を渡るが、その橋に侵入する際に橋幅約 80mm でタイヤ 2 輪同時侵入の位置調整は難しく、2 輪駆動だと時間が掛かる。そのため、並行移動できるメカナムホイールは 2 輪駆動よりスムーズに位置調整が可能である。

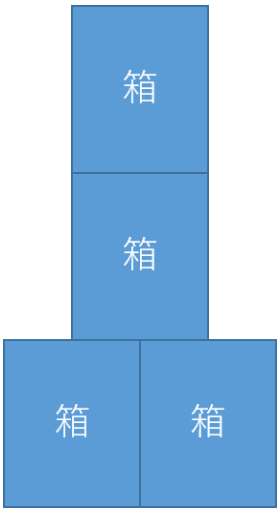


図 2.4　高台の箱積

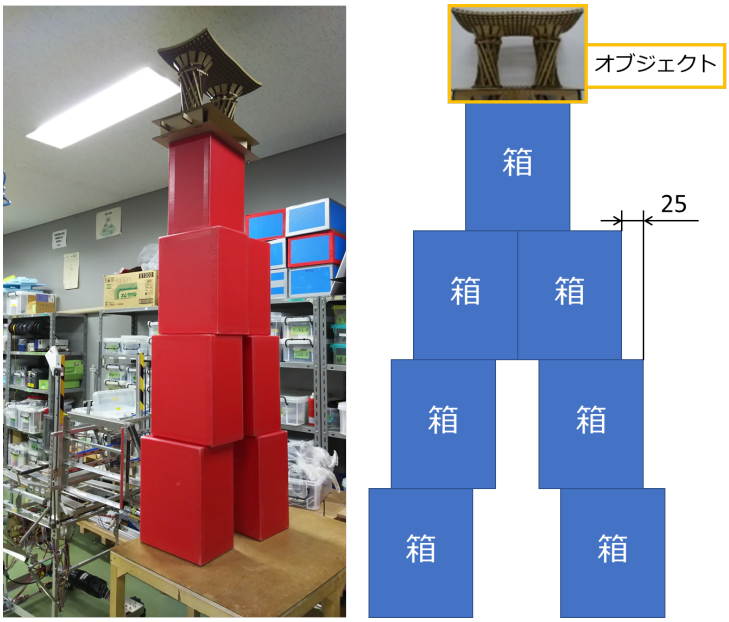


図 2.5　丘の箱積

2.2.4　オブジェクトの持ち運び

丘の左または右に設置されているオブジェクト置き場からオブジェクトの鼓門を持ち運び砦のてっぺんに設置する、この際 2 輪駆動では旋回を何度も必要となるため、並行移動ができるメカナムホイールが有効的である。

第 3 章

メカナムホイールの説明

メカナムホイールとは、一般の自動車と同じように 4 輪 1 セットで使用するが、相違点がいくつか存在する。今回は、形状が特殊であること、向きによって取り付ける場所が決定されていること、各車輪に異なる回転数を与えることで任意の方向に移動可能であることの 3 点を説明する。外見は樽状のタイヤが外周を覆うようにして取り付けられている。また、このタイヤの回転軸は水平・垂直方向から 45 度傾いている。

3.1　メカナムホイールの外見

外見は樽状のタイヤが外周を覆うようにして取り付けられており、このタイヤの回転軸は車輪の水平・垂直方向から図 3.1 のように 45 度傾いている。

3.2　ホイールの向きと組み合わせ

メカナムホイールには向きがあり、これらを正しく取り付けなければ正常に動作しない。図 3.2 のように各車輪に対して縦と横方向に存在する車輪は互いに逆向きの関係であるが、対角に存在する車輪は同じ向きとなる。今回では図 3.1 の向きを順方向として扱い、メカナムホイールごとに番号を振り分けた場合、1 番と 4 番は順方向、2 番と 3 番は逆方向となる。

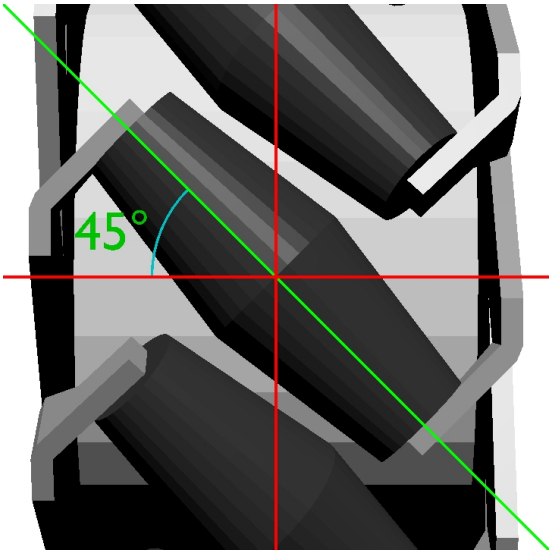


図 3.1　タイヤの角度

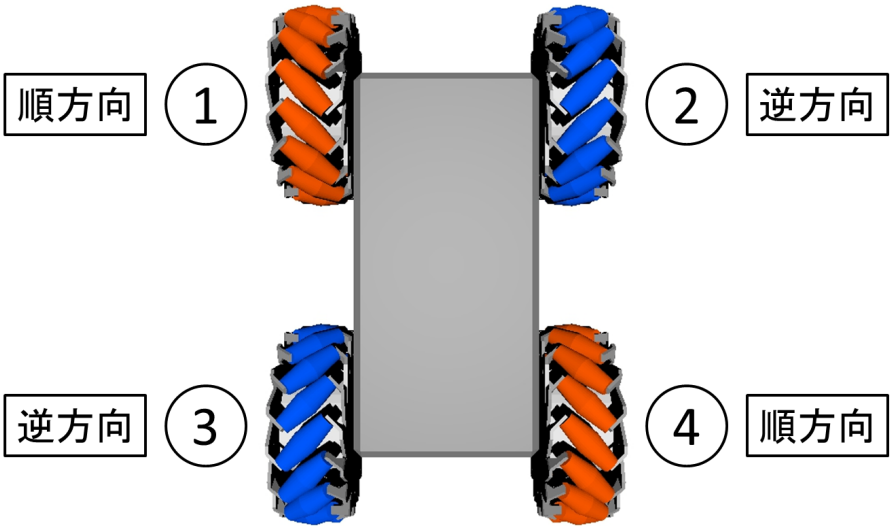


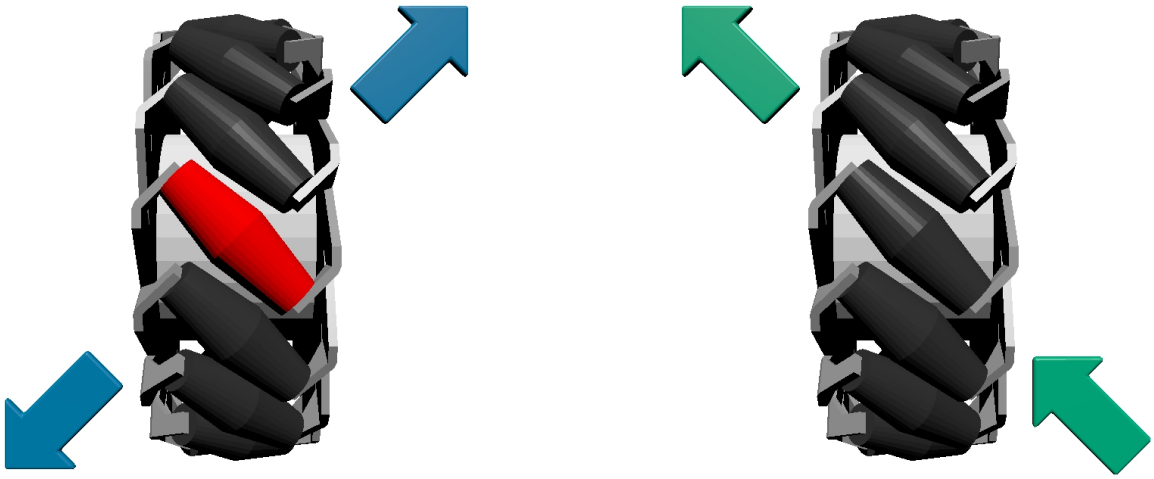
図 3.2　組み合わせ

3.3　各動作

ここではホイール単体の動作と四輪を組み合わせた全体の動作を説明する。

3.3.1　単体の動作

静止した状態では地面に接触しているタイヤの回転方向に進むが、回転時ではその逆方向に移動する。図 3.3 は地面から見たそれぞれの車輪の動きを示している。



(a) 無回転　　(b) 回転

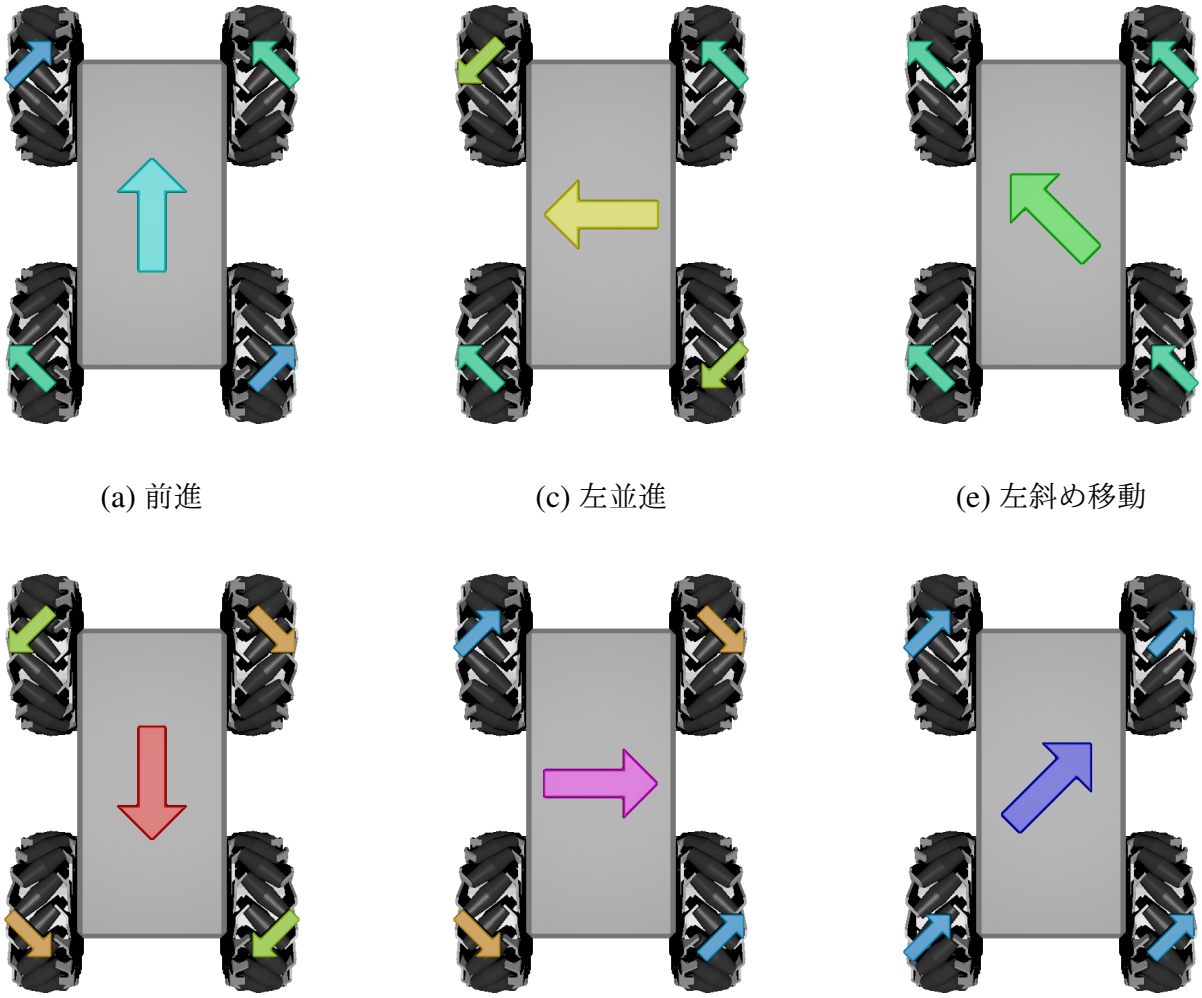
図 3.3　地面から見た車輪の動き

3.3.2　全体の動作

四輪の状態での動作は車輪の回転によって発生する X 方向もしくは Y 方向のベクトルを互いに打ち消し合うことで、図 3.4 に示すように任意の方向に並行移動することができる。

3.4　操作プログラム

ここではメカナムホイールの回転数を制御する計算式や、そのために必要な定義などを説明する。



(a) 前進　(c) 左並進　(e) 左斜め移動

(b) 後進　(d) 右並進　(f) 右斜め移動

図 3.4　車輪のベクトルと本体の進行方向

3.4.1　定義

メカナムホイールを操作するためには縦方向、横方向、回転方向の 3 種類の値が必要である。さらにこれらを計算して PWM 値とするため、アナログ値を出力できるコントローラでなければならない。そのため、これらの基準を満たしている DualShock3 を用いて操作する。また、プログラムの処理には ArduinoMega を使用する。アナログスティックの初期位置を原点とし、そこから X 方向と Y 方向に傾けることで、その量に応じた値が出力される。左スティックは移動用で倒した方向に並行移動することができる。右スティックは旋回用でこちらは X 軸の値しか出力

しない。アナログスティックの位置を座標化したものを図 3.5 に示す。

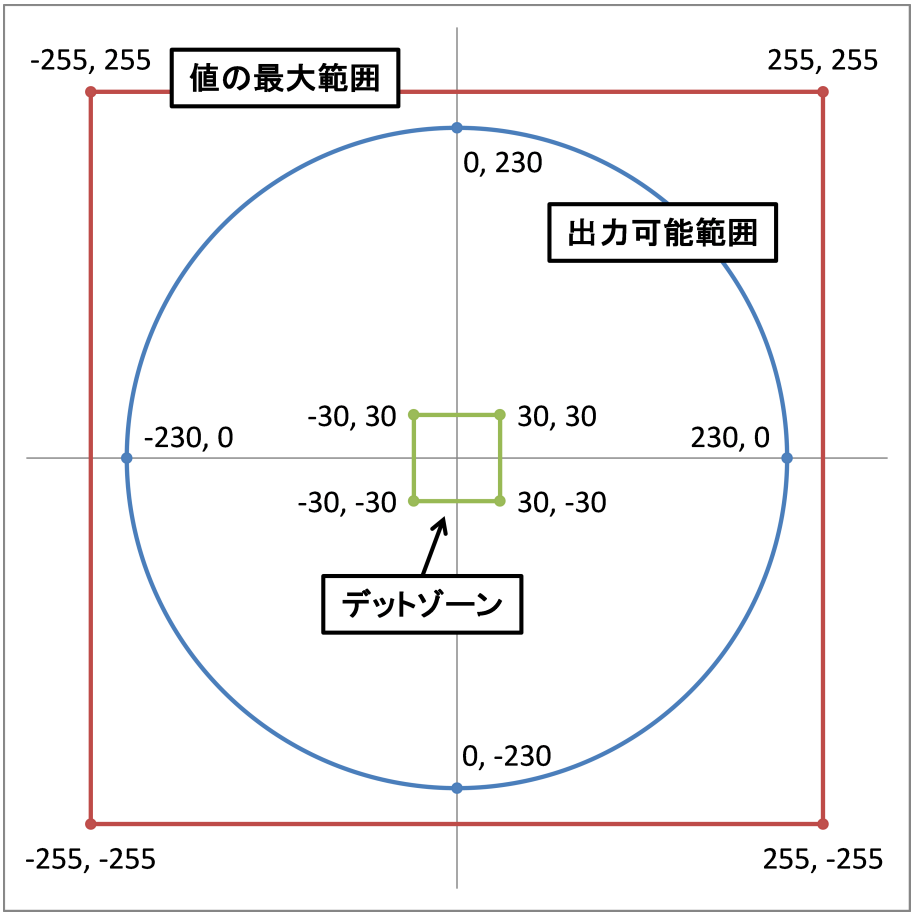


図 3.5　アナログスティックの座標

3.4.2　計算式

左スティックは Y 方向に傾けることで L_y、X 方向に傾けることで L_x を出力する。右スティックは X 方向に傾けることで R_x を出力する。図 3.2 の番号をメカナムホイールに取り付けられたモータの番号とすると、PWM 値の計算式は以下のようになる。

$$M_1 = L_y + L_x + R_x \tag{3.1}$$
$$M_2 = L_y - L_x + R_x \tag{3.2}$$
$$M_3 = L_y - L_x + R_x \tag{3.3}$$
$$M_4 = L_y + L_x + R_x \tag{3.4}$$

第 4 章

今回製作したロボットのメカナムホイールの問題点

4.1 問題点

- 直進中の環境によって進行方向がずれてしまい斜めに進んでしまう。
- 駆動開始時に右方向に大きくずれることがある。
- 停止時に慣性で右方向にずれる。

4.2 原因

4.2.1 項目

以下に 4.1 の問題点が発生すると思われる原因を示す。

1) ロボットの重心
2) 加工精度（タイヤの設置位置）
3) モータ個々の差
4) モータドライバの差

4.2.2　影響を及ぼしている原因とその理由

問題の中で一番影響を及ぼしているのはロボットの重心問題である。理由は、図 4.1 に示すように積み込みロボットには、正面に箱を積み上げるためのラックハンド、片方の側面に箱を回収するためのサイドアームが取り付いているため、重心の位置が中心よりも右前にずれている。これにより左後ろのタイヤが浮くことがあり、4 輪全体が均等に力を床に伝えきれないことから直進中にロボットの進行方向が左右に傾き、ずれてしまうことが問題なのである。

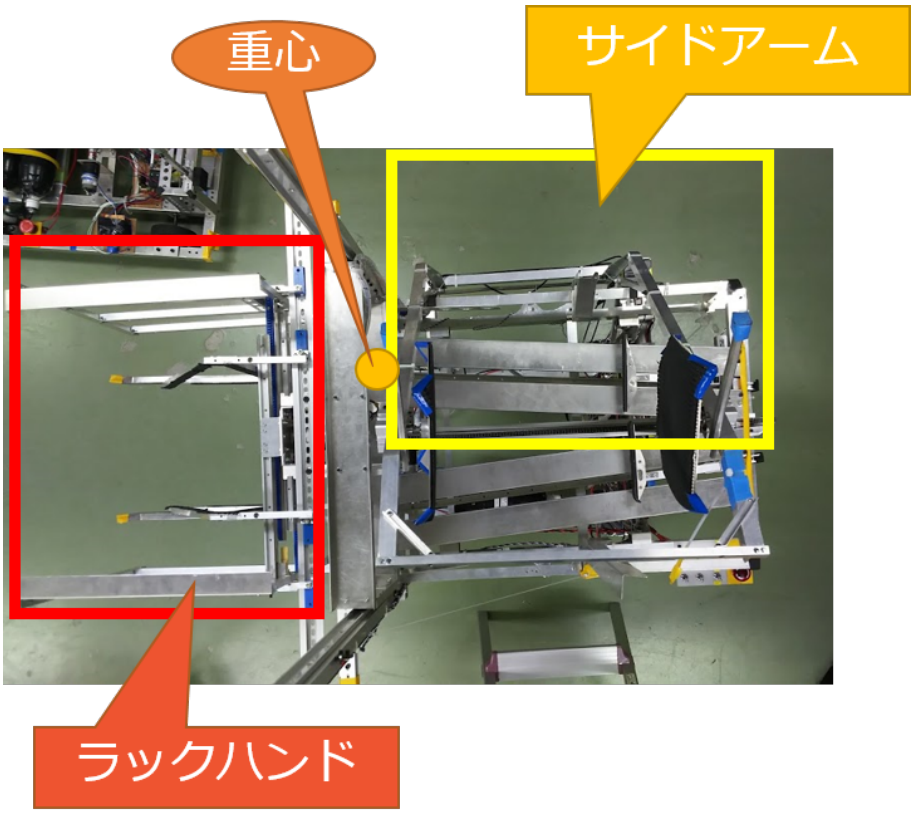


図 4.1　積み込みロボット重心

4.3　方向自動修正

ロボットの進行中の傾きによって直進ができない為、図 4.2 の Arduino 9 軸モーションシールドに搭載されているジャイロセンサを使用し改善を行った。

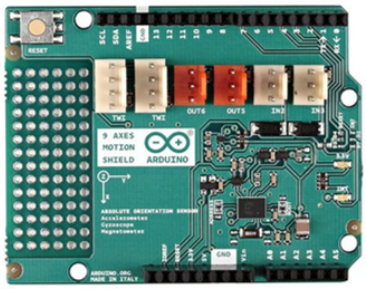


図 4.2　Arduino9 軸モーションシールド図

4.3.1　ジャイロセンサ

ジャイロセンサとは、角速度センサとも呼ばれ、角度が単位時間当たりにどれだけ変化しているのか計測できるセンサである。このセンサは身近にも用いられており、ドローンのバランス調整や PS4 のコントローラに使われている。

4.3.2　補正方法

今回使用したプログラムではジャイロセンサから得られた角度を判定して 0° からずれた分を補正値の PWM 値としてメカナムホイールの動作で使用した式 3.1,3.2,3.3,3.4 に err の形で代入することで以下の式となる。

$$M_1 = L_y + L_x + R_x + err \tag{4.1}$$
$$M_2 = L_y - L_x + R_x + err \tag{4.2}$$
$$M_3 = L_y - L_x + R_x + err \tag{4.3}$$
$$M_4 = L_y + L_x + R_x + err \tag{4.4}$$

こうすることでコントローラからの信号入力中や停止時に慣性で曲がった場合も補正が自動で行われて方向修正が可能となる。

4.4　補正の効果

補正制御あり、なしを比較するため実験を行う。

4.4.1　実験方法

学校の廊下の距離 10m、幅約 2.8m を PWM 値 230、200、150、100、50 の 5 種類で直進させ、そのずれ幅を 5 回測定し平均値で比較する。廊下の壁に接触した場合その時点のずれ幅と進行距離を計測する。

4.4.2　実験結果

補正制御実験の結果から得られた記録を表 4.1、表 4.2 に示す。この時、右方向のずれを正、左方向のずれを負とする。その比較をグラフとして図 4.3 に示す。補正なしの場合、全体が左方向に 1m 前後と大きく傾く結果となった。PWM 値が小さいほど外側にずれて、大きいほど内側にずれた。補正ありの場合、補正なしのようなずれを小さくできたが PWM 値の大きさによって左右にずれが発生する結果となった。PWM 値が大きいほど左にずれ、PWM 値が小さいほど右にずれる。ずれ幅は中心から左右とも 0.4m 以内に収まった。補正ありの時に右方向にずれた理由は、速度が遅く、補正が効きやすくなっていたがロボットの右前に重心がある問題から正面を向いたまま並行に移動してしまった為、傾きが検出できず徐々に右に進んでしまう結果になったと考えられる。速度が速い時に左にずれた理由は直進の PWM 値が大き過ぎて補正の PWM 値が負けてしまい制御なしの時と同様に左方向に引っ張られたと考えられる。

表 4.1　補正なしの表

回数	PWM 値									
	230		200		150		100		50	
	距離	ずれ幅	距離	ずれ幅	距離	ずれ幅	距離	ずれ幅	距離	ずれ幅
1	8.00	-0.52	10.0	-0.66	6.80	-0.72	6.70	-0.72	6.50	-0.70
2	8.50	-0.73	10.0	-0.73	7.20	-0.73	6.20	-0.66	8.80	-0.68
3	7.80	-0.77	10.0	-0.77	7.10	-0.73	5.70	-0.66	6.00	-0.66
4	8.00	-0.72	8.50	-0.80	6.80	-0.66	5.90	-0.72	6.30	-0.73
5	7.80	-0.71	8.90	-0.64	8.80	-0.66	6.10	-0.72	6.00	-0.73
平均	8.02	-0.69	9.48	-0.72	7.34	-0.70	6.12	-0.70	6.72	-0.70

表 4.2　補正ありの表

	PWM 値				
回数	230	200	150	100	50
1	-0.14	-0.40	-0.11	-0.01	0.26
2	-0.10	-0.40	0.07	0.18	0.31
3	-0.32	-0.11	0.13	0.16	0.37
4	-0.40	-0.58	0.16	0.14	0.28
5	-0.56	-0.54	0.05	0.17	0.24
平均	-0.30	-0.40	0.06	0.13	0.29

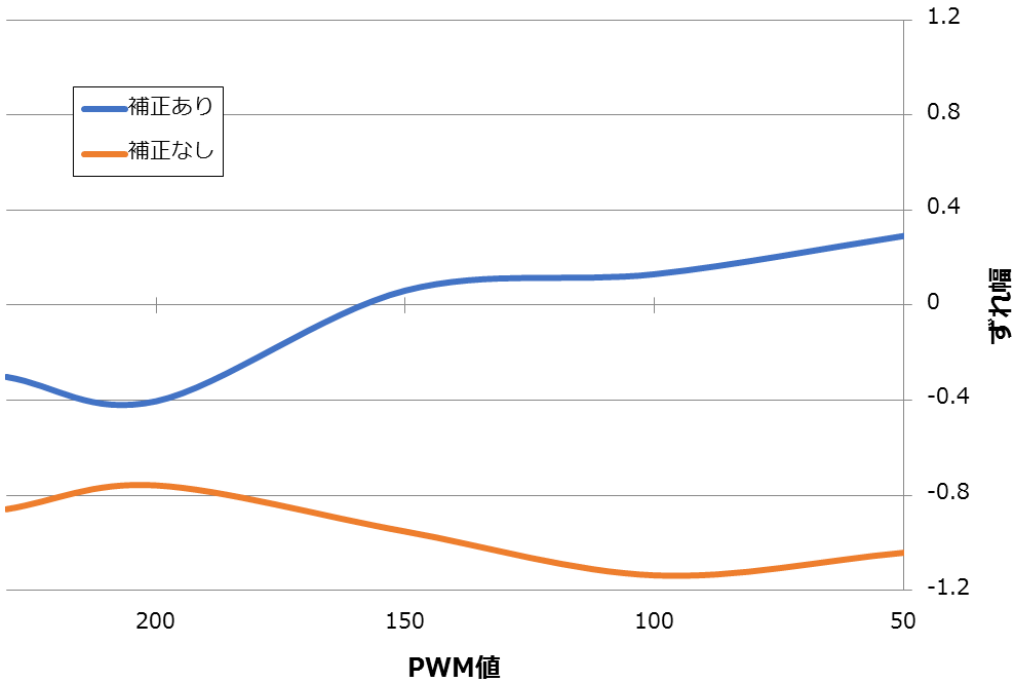


図 4.3　比較のグラフ

第5章

おわりに

実験結果からジャイロセンサでの補正効果は見られたがロボットの傾き情報のみでは、ロボットが正面を向いたまま少しずつ斜めに進行した際に補正の対応ができないことと、直進のPWM値が補正のPWM値を大きく上回る場合、補正効果が薄れてしまうことから完璧な補正はすることができなかった。今後の課題として、ジャイロセンサ以外のセンサを用いて傾き以外の画像や位置といった情報を取り込み、直進のPWM値と比例して補正のPWM値を調整できる制御ができればより良い補正が可能となると考える。

参考文献

[1] Arduino9AxesMotionLibrary,http://www.arduino.org/learning/reference/9-axes-motion,2017/01/16 閲 覧

謝辞

本研究を進めるにあたり、ご指導を頂いた卒業論文指導教員の伊藤恒平先生、伊勢大成先生、また、大会まで多くの応援や差し入れなどをしていただいた皆様へ心から感謝の気持ちと御礼を申し上げたく、謝辞にかえさせていただきます。

付録

mecanum 1　メカナムホイールの操作とジャイロによる方向修正プログラム

```
1  #include <PS3BT.h>
2  #include <usbhub.h>
3  #ifdef dobogusinclude
4  #include <spi4teensy3.h>
5  #include <SPI.h>
6  #endif
7
8
9
10 #include "NAxisMotion.h"
11 #include <Wire.h>
12
13 int M1 = 8;                  //メカナムホイール用
14 int M2 = 11;               //軸ジャイロにピンは使えない34
15 int M3 = 12;
16 int M4 = 3;
17 int E1 = 22;
18 int Q1 = 26;
19 int E2 = 23;
20 int Q2 = 27;
21 int E3 = 24;
22 int Q3 = 28;
23 int E4 = 32;
24 int Q4 = 33;
25 int LeftY = 0;
26 int LeftX = 0;
27 int RightX = 0;
28 int vector1 = 0;
29 int vector2 = 0;
30 int vector3 = 0;
31 int vector4 = 0;
32 int absolute1 = 0;
33 int absolute2 = 0;
34 int absolute3 = 0;
35 int absolute4 = 0;
36 int range1 = 0;
37 int range2 = 0;
38 int range3 = 0;
39 int range4 = 0;
40 int value1 = 0;
```

```
41 int value2 = 0;
42 int value3 = 0;
43 int value4 = 0;
44
45
46
47
48 int Rf=180;
49 int rf=0;
50 int sp=0;
51 int err=0;
52 int err2=0;
53
54
55
56
57 USB Usb;
58 BTD Btd(&Usb);
59 PS3BT PS3(&Btd);
60
61
62
63 //軸ジャイロ3
64 NAxisMotion mySensor;
65 unsigned long lastStreamTime = 0;
66 const int streamPeriod = 20;
67
68
69
70
71
72 void setup() {
73
74
75   pinMode(E1, OUTPUT);                                    //メカナムホイール用入出力
76   pinMode(Q1, OUTPUT);
77   pinMode(E2, OUTPUT);
78   pinMode(Q2, OUTPUT);
79   pinMode(E3, OUTPUT);
80   pinMode(Q3, OUTPUT);
81   pinMode(E4, OUTPUT);
82   pinMode(Q4, OUTPUT);
83   analogWrite(M1, value1);
84   analogWrite(M2, value2);
85   analogWrite(M3, value3);
86   analogWrite(M4, value4);
87
88
89
90  Serial.begin(115200);
91   while (!Serial);
92   if (Usb.Init() == -1) {
93     Serial.print(F("\r\nOSC did not start"));
94     while (1);
```

```
95      }
96      Serial.print(F("\r\nPS3 Bluetooth Library Started"));
97
98
99
100
101      I2C.begin();
102     //Sensor Initialization
103     mySensor.initSensor();
104     mySensor.setOperationMode(OPERATION_MODE_NDOF);
105     mySensor.setUpdateMode(MANUAL);
106
107
108
109
110
111
112   }
113
114
115   void loop() {
116     Usb.Task();
117
118   //メカナムホイールプログラム開始
119
120     static int mokuhyouti = 0;
121     static int kaiten = 0;
122     static int RS0 = 0;
123     int RS = (float)(mySensor.readEulerHeading());
124     if(RS > 360-20 && RS0 < 0+20) {
125       kaiten = kaiten - 1;
126     }
127     if(RS0 > 360-20 && RS < 0+20) {
128       kaiten = kaiten + 1;
129     }
130     rf = 360 * kaiten + RS;
131     RS0 = RS;
132
133     if (PS3.PS3Connected){
134       if (PS3.getAnalogHat(LeftHatY) > 135 || PS3.getAnalogHat(LeftHatY) < 120){
135         LeftY = (int)((((float)PS3.getAnalogHat(LeftHatY) - 127.5f) / 127.5f) * -255.0f);
136       }
137       else{
138         LeftY = 0;
139       }
140
141
142       if (PS3.getAnalogHat(LeftHatX) > 135 || PS3.getAnalogHat(LeftHatX) < 120){
143         LeftX = (int)((((float)PS3.getAnalogHat(LeftHatX) - 127.5f) / 127.5f) * 255.0f);
144       }
145       else{
146         LeftX = 0;
147       }
148       if (PS3.getAnalogHat(RightHatX) > 135 || PS3.getAnalogHat(RightHatX) < 120){
```

```
149         RightX = (int)((((float)PS3.getAnalogHat(RightHatX) - 127.5f) / 127.5f) * 255.0f);
150       }
151       else{
152
153         mokuhyouti = rf;
154         RightX = 20 * (0 - rf);
155         RightX = constrain(RightX, -20, 20);
156       //Serial.print(rf);
157       //Serial.print(' ');
158       Serial.println(RightX);
159       }
160       if (LeftY != 0 || LeftX != 0 || RightX != 0){
161         vector1 = LeftY + LeftX + RightX;
162         vector2 = -(LeftY - LeftX - RightX);
163         vector3 = LeftY - LeftX + RightX;
164         vector4 = -(LeftY + LeftX - RightX);
165         absolute1 = abs(vector1);
166         absolute2 = abs(vector2);
167         absolute3 = abs(vector3);
168         absolute4 = abs(vector4);
169         range1 = constrain(absolute1, 0, 255);
170         range2 = constrain(absolute2, 0, 255);
171         range3 = constrain(absolute3, 0, 255);
172         range4 = constrain(absolute4, 0, 255);
173         value1 = range1 * (vector1 / absolute1);
174         value2 = range2 * (vector2 / absolute2);
175         value3 = range3 * (vector3 / absolute3);
176         value4 = range4 * (vector4 / absolute4);
177       }
178       else{
179         value1 = 0;
180         value2 = 0;
181         value3 = 0;
182         value4 = 0;
183       }
184
185
186
187       if (value1 > 0){
188         digitalWrite(E1,HIGH);
189         digitalWrite(Q1,LOW);
190       }
191       else if (value1 < 0){
192         digitalWrite(E1,LOW);
193         digitalWrite(Q1,HIGH);
194         value1 = abs(value1);
195       }
196       else {
197         digitalWrite(E1,LOW);
198         digitalWrite(Q1,LOW);
199       }
200
201       if (value2 > 0){
202         digitalWrite(E2,HIGH);
```

```
203         digitalWrite(Q2,LOW);
204       }
205       else if (value2 < 0){
206         digitalWrite(E2,LOW);
207         digitalWrite(Q2,HIGH);
208         value2 = abs(value2);
209       }
210       else {
211         digitalWrite(E2,LOW);
212         digitalWrite(Q2,LOW);
213       }
214
215       if (value3 > 0){
216         digitalWrite(E3,HIGH);
217         digitalWrite(Q3,LOW);
218       }
219       else if (value3 < 0){
220         digitalWrite(E3,LOW);
221         digitalWrite(Q3,HIGH);
222         value3 = abs(value3);
223       }
224       else {
225         digitalWrite(E3,LOW);
226         digitalWrite(Q3,LOW);
227       }
228
229       if (value4 > 0){
230         digitalWrite(E4,HIGH);
231         digitalWrite(Q4,LOW);
232       }
233       else if (value4 < 0){
234         digitalWrite(E4,LOW);
235         digitalWrite(Q4,HIGH);
236         value4 = abs(value4);
237       }
238       else {
239         digitalWrite(E4,LOW);
240         digitalWrite(Q4,LOW);
241       }
242
243       analogWrite(M1, value1);
244       analogWrite(M2, value2);
245       analogWrite(M3, value3);
246       analogWrite(M4, value4);
247
248       if (PS3.getButtonClick(PS)){
249         Serial.print(F("\r\nPS"));
250         PS3.disconnect();
251       }
252
253     }
254
255   //ここまでメカナムホイール
256
```

```
257
258       if (PS3.getButtonClick(PS)) {                          //ボタンPS
259         Serial.print(F("\r\nPS"));
260         PS3.disconnect();
261       }
262
263
264
265
266      if ((millis() - lastStreamTime) >= streamPeriod)
267     {
268       lastStreamTime = millis();
269       mySensor.updateEuler();
270       mySensor.updateCalibStatus();
271     }
272
273
274           if (PS3.getButtonClick(CIRCLE)) {                  //○ボタン
275            Serial.print(F("\r\nCircle"));
276   //          (mySensor.readEulerHeading())=0
277         }
278
279
280  }
```