プロトタイプ演習 報告書
9 班

4 年 電気情報工学科 樫内蒼太朗
共同実験者 4 年 電気情報工学科 相 方

1. 概要

4 年の前期に行われるプロトタイプ演習とは、2 人 1 班になり、競技会の課題をこなす授業である。この章では、プロトタイプ演習の概要と、競技会に向けた開発スケジュール、結果を示す。

1. 競技会のルール

まず、競技会の課題やルールを説明する。

1.1 課題

① 方眼紙の任意の 100mm 正方形内に渦巻を描く。渦巻の周回数の関数を用いて成績を決定する。

② 方眼紙の任意の 10mm マスに沿った長さ 150mm の線分を描く。線分の長さ, 誤差, 基準線からの最大距離の関数を用いて成績を決定する。

1.2 配布物

- RC サーボモータ
- ブレットボード&ジャンパ線
- A4 サイズアクリル板
- ボールペン

1.3 マシンについて

- アクチュエータには配布された RC サーボモータ以外の使用を禁止する
- アクリル板以外の材料を追加しても良い。ただし高額な物品の使用は自重すること
- 配布したボールペン以外を用いて描いても良い。ただし 2 本以上の筆記具を用いてはならない
- ヒトの手を用いて地面に押し付けるようにマシンを固定することは許可する。ただしマシンを持ち上げるような保持は不可とする
- 渦巻と線分で動作プログラムを変更しても良い
- 渦巻と線分でボールペンの固定位置変更やパーツの交換など、マシンの構造を変更してはならない

1.4 競技会について

- 競技会時に配布する方眼紙に渦巻と線分を描くこと。描きなおしはそれぞれ 1 回までとし、描いた後に 2 つから採点対象を 1 つ学生が指定する。
- 渦巻を描く 100mm 正方形や直線の 150mm の基準線は、 描く前に方眼紙のマスに沿って学生が指定する。
- 方眼紙は折ったり切ったりして使用しても良い。ただし円や線分が描かれている箇所は線の途中で紙が切れていてはならない。また方眼紙を切断して使用した場合は競技終了後にテープ等で接合して提出すること。

2. 使用物品

今回のマシンで使用した物品を以下に示す。

- Arduino
- RC サーボモータ
- ジャンパ線
- A4 アクリル板
- ボールペンの替え芯
- 連結部品
  - M2 ねじ
  - M3 ねじ
  - M3 ナット
- パソコン
- 3D プリンタ
- レーザ加工機

3. ガントチャート

開発の進捗を管理するために、ガントチャートを作成した。図 1-1 は 4 月 22 日に作成した計画で、図 1-2 は 7 月 12 日に作成した実際の進捗である。

モータ精度測定実験などのタスクが急遽増加したが、柔軟に対応することができた。また、私たちのマシンは 3D プリンタを用いている部品が多いため、CAD の詳細な設計や修正は、寮生である私が担当することに変更した。マシン構造検討とマシン詳細設計に時間がかかったため、中間テスト期間にも開発する羽目になったが、比較的計画的に進めることができ、競技会までに間に合わすことができた。

タスク		担当者	所要時間	1週目 4/7	2週目 4/14	3週目 4/21	4週目 4/28	5週目 5/5	6週目 5/12	7週目 5/19	8週目 5/26	9週目 6/2	10週目 6/9	11週目 6/16	12週目 6/23	13週目 6/30	14週目 7/7	15週目 7/14	16週目 7/21	17週目 7/28	18週目 8/4
ハード	マシン構造検討	相方, 樫内	10	2	4	4					中間テスト期間					競技会		期末テスト期間			
	マシン詳細設計(CAD)	相方	30			10	10	10													
	材料加工	相方, 樫内	8						8												
	組み立て	相方	3							3											
回路	回路詳細設計	樫内	3				3														
	回路配線	樫内	3							3											
ソフト	各部品動作確認	樫内	4				4														
	プログラム基本設計	樫内	10					10													
	コーディング	樫内	10						10												
試験	動作確認	相方, 樫内	8										8								
	ハード不具合修正	相方	0																		
	ソフト不具合修正	樫内	0																		
	性能向上・微調整	相方, 樫内	0																		
	タスク数の合計		89	2	4	14	17	20	18	6	0	0	8	0	0	0	0	0	0	0	0

図 1-1 当初の計画

タスク		担当者	所要時間	1週目 4/7	2週目 4/14	3週目 4/21	4週目 4/28	5週目 5/5	6週目 5/12	7週目 5/19	8週目 5/26	9週目 6/2	10週目 6/9	11週目 6/16	12週目 6/23	13週目 6/30	14週目 7/7	15週目 7/14	16週目 7/21	17週目 7/28	18週目 8/4
ハード	マシン構造検討	相方, 樫内	10	2	4	4					中間テスト期間							期末テスト期間			競技会
	マシン概要設計(Blender)	相方	3			3															
	マシン概要設計(CAD)	相方	26				3	2	16	5											
	マシン詳細設計(CAD)	樫内	50								20	15	15								
	材料加工	樫内	10										5	5							
	組み立て	樫内	5										2	3							
回路	回路詳細設計	樫内	2				2														
	回路配線	樫内	1											1							
ソフト	各部品動作確認	樫内	2				2														
	プログラム基本設計	樫内	5					5													
	コーディング	樫内	13						5	8											
モータ精度測定	実験装置設計	樫内	6					6													
	実験装置組み立て	樫内	1						1												
	実験	樫内	3							3											
	データまとめ	相方	3											3							
	分析	樫内	8												8						
試験	動作確認	相方, 樫内	3											1	2						
	ハード不具合修正	樫内	25												25						
	ソフト不具合修正	樫内	10												10						
	性能向上・微調整	相方, 樫内	10												10						
	競技会への対策	樫内	5													5					
タスク数の合計			201	2	4	7	7	13	22	16	20	15	22	13	55	5	0	0	0	0	0

図 1-2 実際の進捗

4. 結果

私たちが作成したマシンを図 1-3, 図 1-4 に示す。

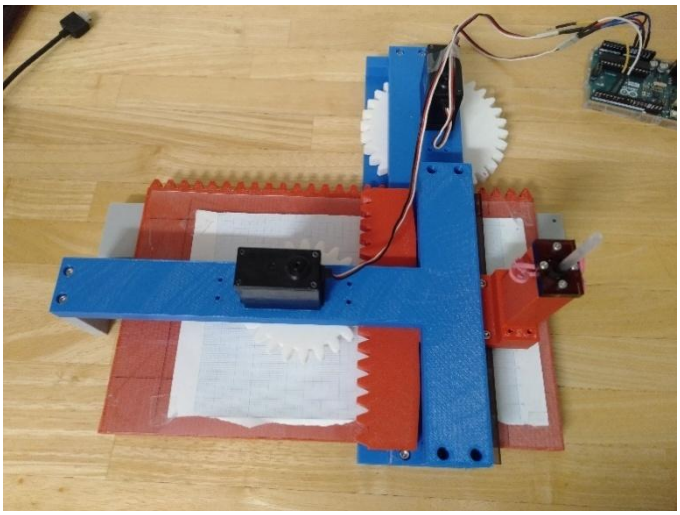


図 1-3 作成したマシンの正面

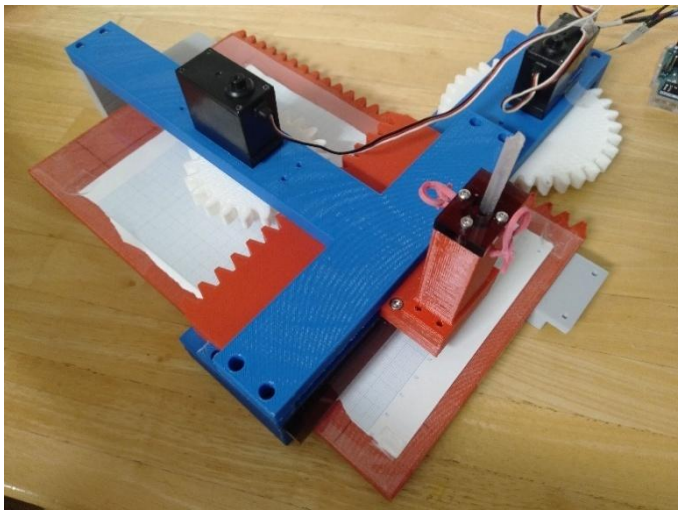


図 1-4 作成したマシンの斜め

私たちの最高記録を表 1-1 にまとめる。また、最高記録の様子を図 1-5, 図 1-6 に示す。

表 1-1 最高記録

課題	**評価項目**	**評価**
渦巻	渦巻の周回数	80[周]
線分	線分の長さ	150.2 [mm]
	基準線からの最大距離	0.1[mm]

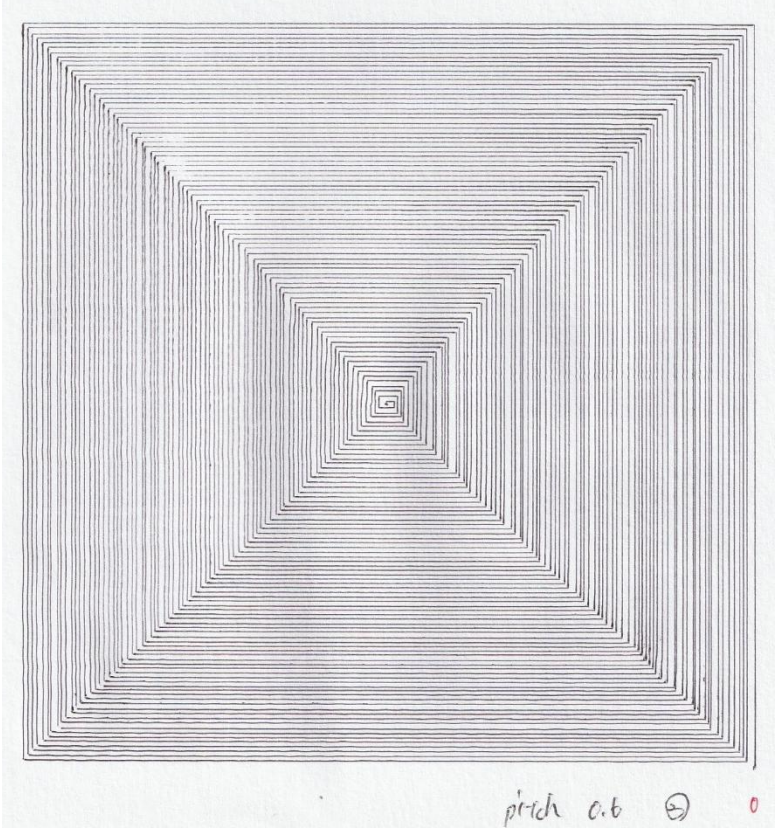


図 1-5 渦巻の最高記録

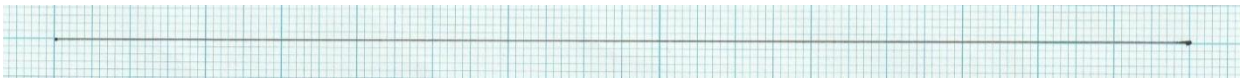


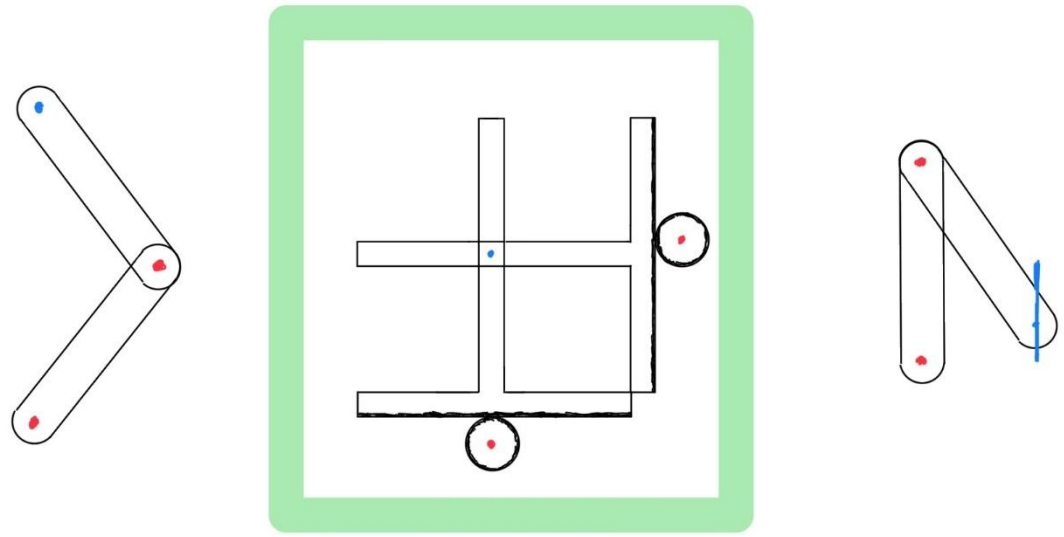
図 1-6 直線の最高記録

2. 機構の選定と設計

記録が残っている過去 8 年間では、細かなルール改定はあったものの、課題は「直線と円を描くこと」であった。しかし今年度からは、「円」の代わりに「渦巻」を描くという大幅な変更が加えられた。そこで本章では、過去の機構を参考にしつつ、新たな課題に対応するための機構の選定と設計を検討する。

1. 主要な機構の選定

初めに、マインドマップを元に様々な機構の案を出した。それらを①アーム型, ②x-y 軸型, ③コンパス型の 3 つに分類した。各機構の概要を図 2-1 に示す。



①アーム型　　②x-y 軸型　　③コンパス型(横から見た図)

図 2-1 主な機構案

今回の課題では、渦巻と直線を描く必要がある。しかし、渦巻の課題では、渦を巻いた回数が得点につながるため、円形の渦ではなく、四角形の渦でも良いことになる。つまり、渦巻も直線を描く課題も、どちらも直線だけで表現することができる。このことから、直線がきれいに描けそうな図 2-1②の x-y 軸型の機構を採用した。

図 2-1①のアーム型を採用しなかった理由は、去年の競技会の資料を見て、あまり精度が出ていなかったためである。その理由としては、2 つのモータの誤差が同時にペン先に現れるからではないかと考えられる。

図 2-1③のコンパス型を採用しなかった理由は、コンパス型には、無限に回転する軸が必要であるからである。今回配布されたサーボモータは 180° しか回転しないので、ラチェット機構などを用いて、無限に回転する機構を作成する必要がある。しかしラチェット機構は、比較的高いハードウェアの知識が必要であると考えられる。そのため、もしコンパス型を採用したが、ラチェット機構が作成できなければ、一から代行の機構を考え直す必要がある。それらのリスクを回避するため、図 2-1③のコンパス型を採用しなかった。

2. 機構の実装方法の選定

次に、図 2-2 に示すように、図 2-1②の x-y 軸型の具体的な実装方法を考えた。図 2-1②x-y 軸型はサーボモータで動作させる部位によって 2 通りの方法が考えられる。具体的には図 2-2①x 軸, y 軸のペン先を動かす方法と、図 2-2②片軸のペン先を、もう片軸の紙を動かす方法である。前者は、x 軸と y 軸を動かし、その交点となる部分にペン先を設置する方法である。後者は、x(y)軸はペン先を、y(x)軸は紙を動かす方法である。

また、各軸を動かす方法として図 2-2(A)ラックとオピニオン, 図 2-2(B)平行リンク機構, 図 2-2(C)スライダークランク機構の 3 通りの方法を考えた。

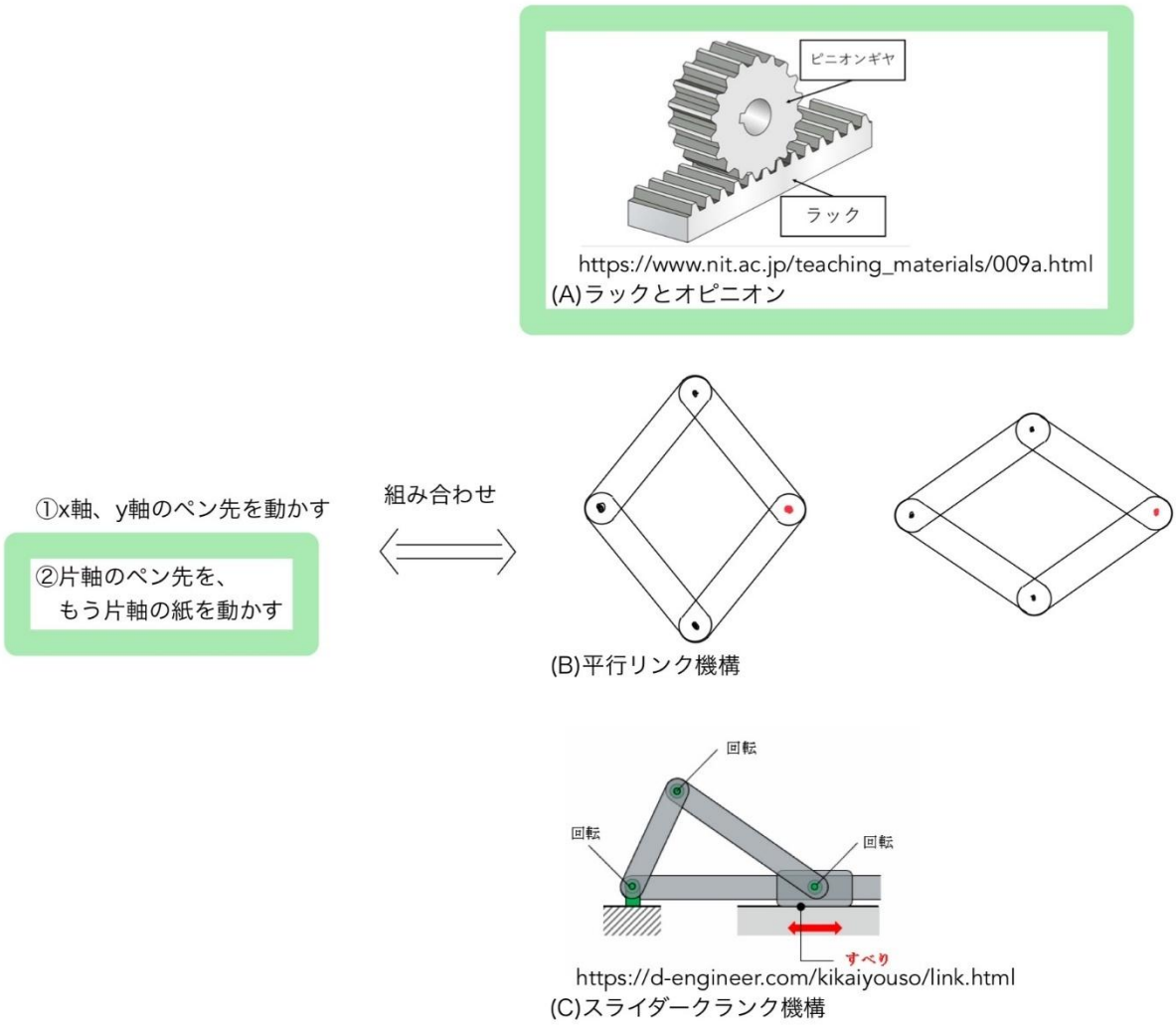


図 2-2 x-y 軸型の実装方法

これらの案のうち、私たちは、図 2-2②片軸のペン先を、もう片軸の紙を動かす方法と、図 2-2(A)ラックとオピニオンの方法を採用することにした。図 2-2①x 軸、y 軸のペン先を動かす方式は、去年採用している班があったが、稼動する部分の交点にペン先を設置する必要があり、ぎくしゃくしている印象があったため、採用しなかった。また、図 2-2(B)平行リンク機構と、図 2-2(C)スライダークランク機構は、サーボモータの角度に対して、移動距離が、一定ではないので、精度が高い部分と、低い部分が出てしまうのではないかと考え、採用しなかった。

3. 機構の設計

次に、マシンの設計を Autodesk 社の Fusion を使用して行った。マシンの概要設計を相方が行い、ギヤの数や、パーツ同士の詳細な固定方法などは私が行った。

作成したマシンの設計の CAD データを図 2-3 に示す。なお図 2-3 では、サーボモータを水色で、x 軸のギヤとラックを赤色で、y 軸のギヤとラックを紫色で、ペンを設置する機構の部分を緑色で表現している。方眼用紙は紫色の板に設置する。

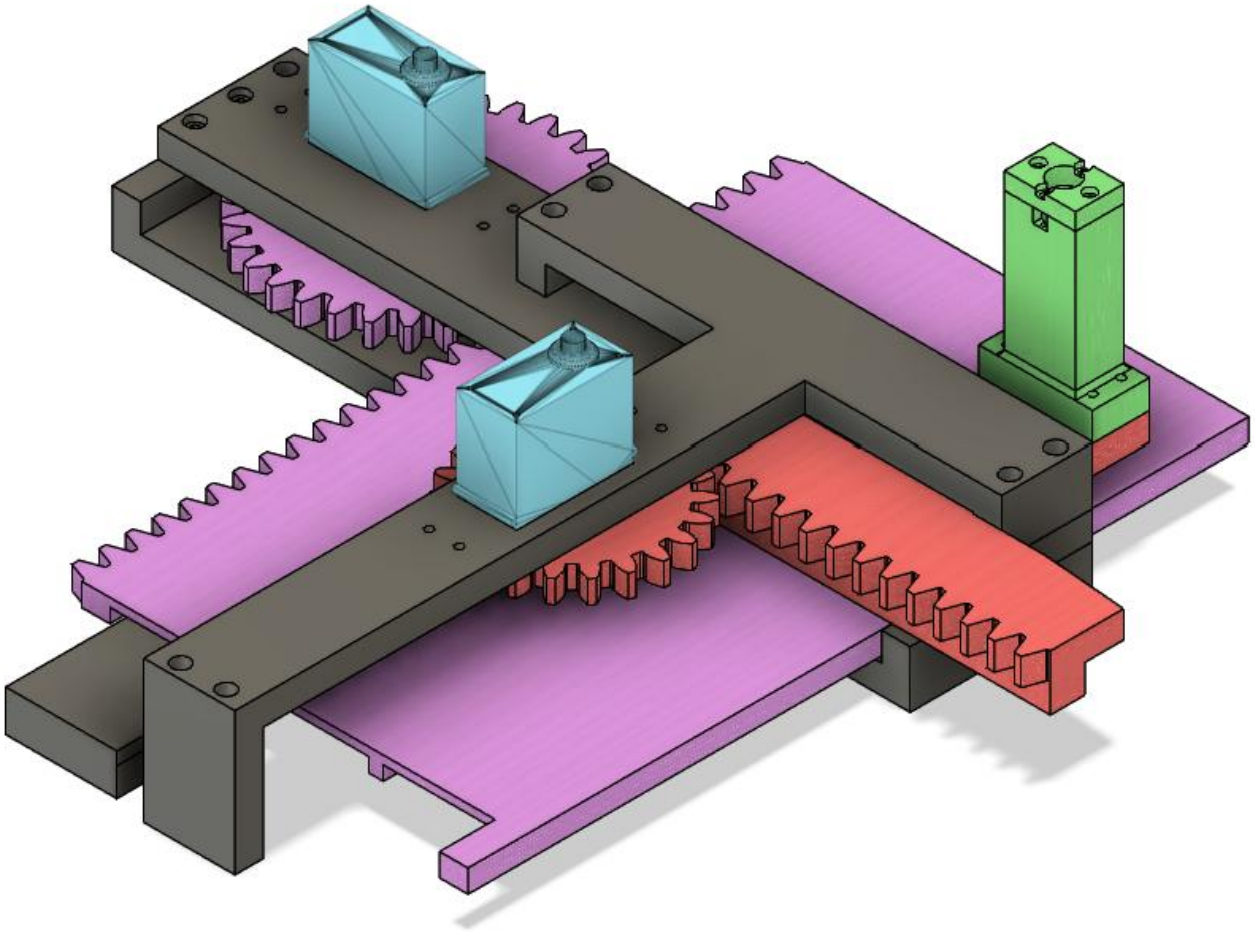


図 2-3 マシンの CAD データ

3. プログラムの作成

この章では、プログラムの詳細な仕様について説明する。なお全てのプログラムは、「7.付録」に記載し、ここでは特に重要と思われる部分のみの説明を行う。

1. 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。サーボモータやシリアル通信の初期化処理等が終了した後、プロンプトを表示、パソコンのキーボードから渦巻モード、直線モード、脱力モードを選択できるようにした。プロンプトと各モードの選択する様子を図 3-2 に示す。また各モードの動作が終了後、再度モード選択できるように無限ループに入れた。こうすることで、直線や渦巻を全く同じプログラムで動作させることができる。

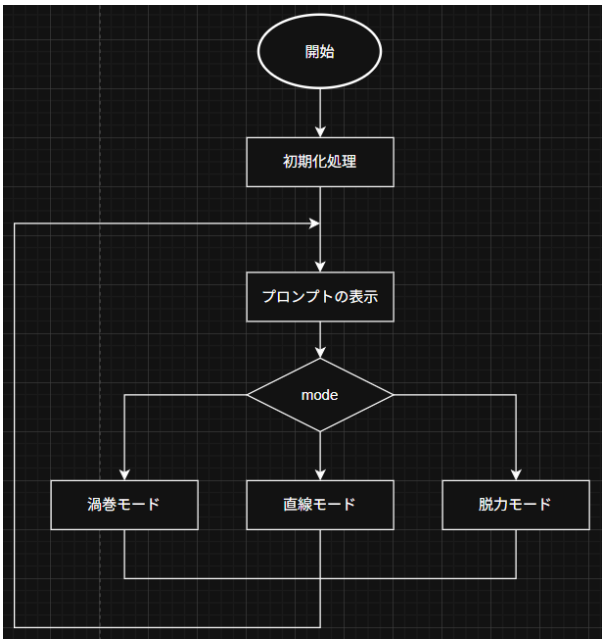


図 3-1 フローチャート

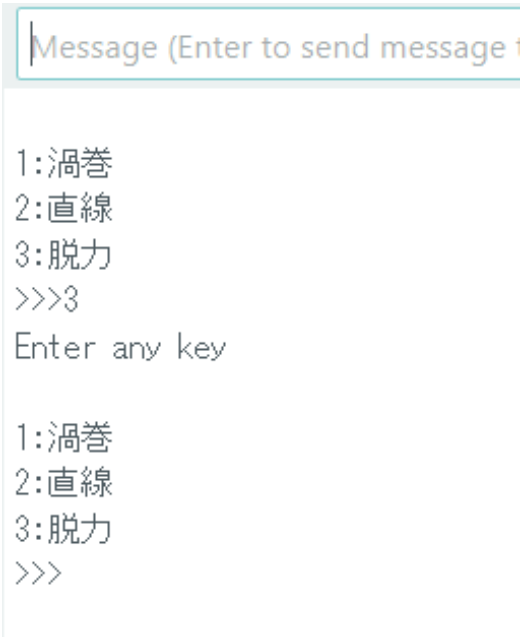


図 3-2 プロンプトの動作の様子

渦巻モードと直線モードは、名前の通り、課題①の渦巻を描くモードと、課題②の直線を描くモードである。脱力モードとは、サーボモータを脱力させるモードで、マシンに紙をセットする際や、マシンのメンテナンスの際に使用する。

2. 最小単位のプログラム

次に各モードの基本となる 3 つの関数を紹介する。なお各関数で使われている x, y の値は、マシンの紙を置く部分の長い方を y 軸、短い方を x 軸としたもので、図 3-3 のような座標系である。

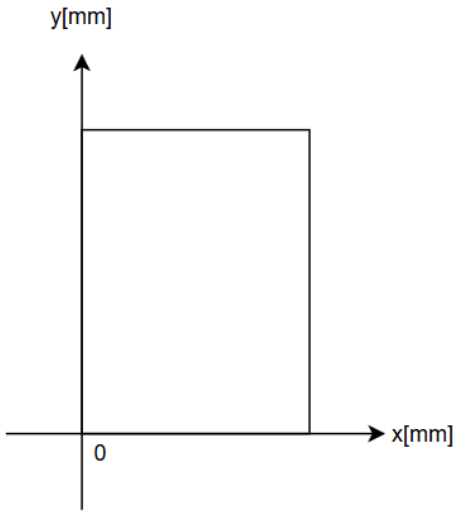


図 3-3 座標系

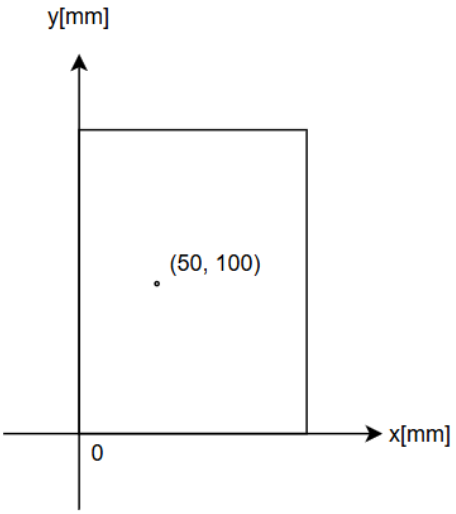


図 3-4 set_position 関数の動作

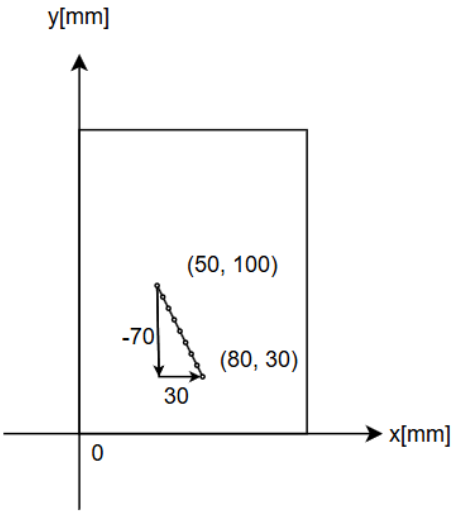


図 3-5 line 関数の動作

void set_position(double x, double y, int delay_time)

引数で指定した座標(x, y)にペン先を移動させる。サーボモータに移動させる命令を送信した後、delay_time ミリ秒だけ待機する。図 3-4 は set_position(50, 100, 300)を呼び出した後のペン先の様子である。

void line(double dx, double dy)

ペン先を(dx, dy)分だけ移動させる。line 関数内では(dx, dy)が作る直線を特定の分解能(0.1[mm]ほど)で分割し、多数の点で近似、その点を通るように set_position 関数を呼び出すことで実装している。図 3-5 は、図 3-4 の状態で line(30, -70)を呼び出した後のペン先の様子である。

void wait_any_key(void)

パソコンのキーボードから何らかの入力があるまで、プログラムを一時停止する。入力は「Enter」だけでも良い。

3. モードの詳細

先ほどの 3 つの関数を用いて、渦巻モード、直線モード、脱力モードのプログラムの詳細を説明する。

渦巻モード

渦巻モードのプログラムを以下に示す。pitch に渦巻の線と線の間隔を[mm]単位で指定する。for 文の中に、line 関数の引数を pitch 分徐々に短くして呼び出すことで実現している。

```
double pitch = 0.6;
set_position(2, 2, 1000);
wait_any_key();

for (double length = 96; 0 < length; length -= pitch * 2) {
  line(length, 0);
  line(0, length);
  line(-(length - pitch), 0);
  line(0, -(length - pitch));
}
```

直線モード

直線モードのプログラムを以下に示す。x 軸側のサーボモータを脱力させることにより、x 軸のペン先を調節できるようにした。x 軸の位置が決まれば、キーボードから適当なキーを押すことで、150[mm]の直線を引くようになっている。

```
set_position(100, 0, 5000);
servo_x.detach();
wait_any_key();
line(0, 150);
```

脱力モード

パソコンの適当なキーを受け付けるまで、サーボモータを脱力させることで実現している。

```
servo_x.detach();
servo_y.detach();
wait_any_key();
servo_x.attach(3);
servo_y.attach(5);
```

4. シミュレーション

相方が「マシン概要設計(CAD 作成)」に取り組んでいる間、作成したプログラムがどのような図形を描画するかシミュレーションを行った。

シミュレーションの方法として、サーボモータの角度を指定している箇所に、シリアル通信でその角度を出力させ、サーボモータの様子を数値としてまとめた。具体的には、set_position 関数内に、シリアル通信にサーボモータの角度を出力させるようにした。

シミュレーションでは、データが多くなりすぎないように、楕円の pitch を 5[mm], 線の分解能を 5[mm]と大きめの値に設定して行った。シミュレーション結果のデータを元にグラフにプロットしたものを図 3-6 に示す。図 3-6 から、100[mm]の正方形の中に渦巻が正しく描けていることが分かる。

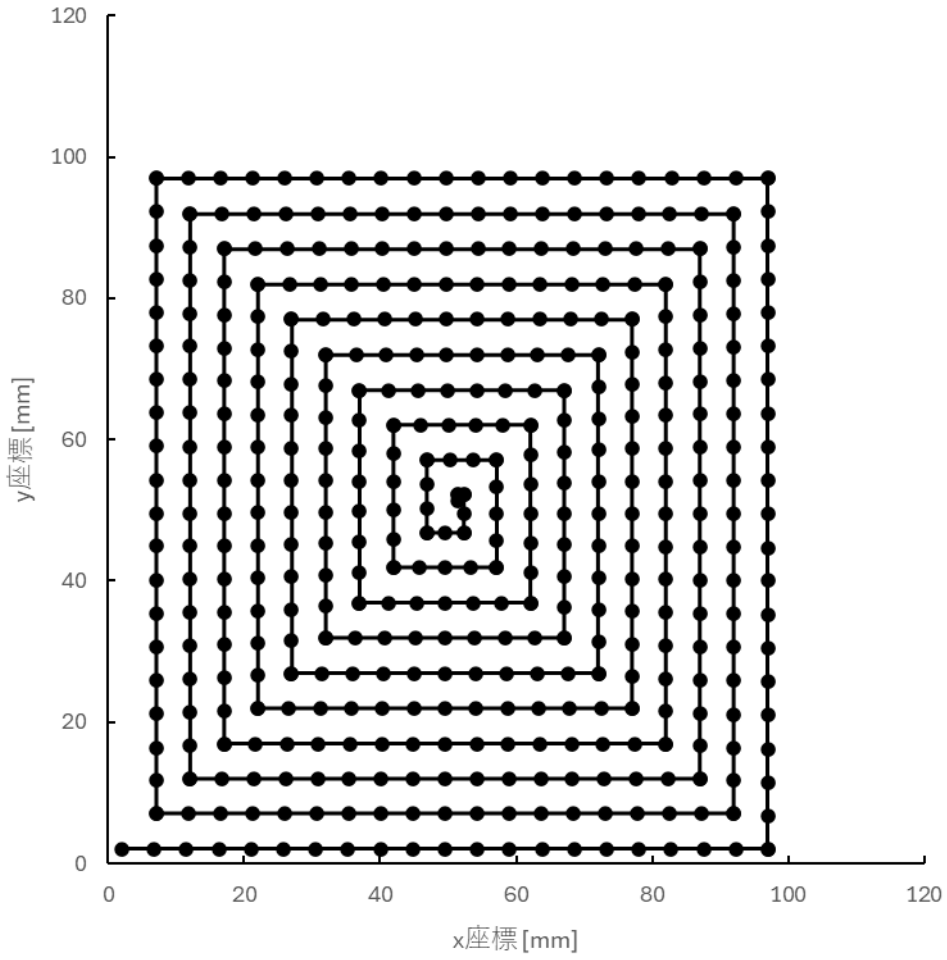


図 3-6 シミュレーション結果

4. 動作確認と諸修正

この章では、精度をあげるための様々な取り組みを紹介する

1. サーボモータ精度測定

今回配布されたサーボモータは、「KRS-786 ICS」である。このサーボモータの詳細な仕様書がインターネット上には存在しない。また、別の班の人たちが、このサーボモータはある特定の角度付近でとても大きな誤差が発生すると言っていた。そこでモータの特性と精度を測定する実験を行うことにした。

1.1 原理

RC サーボモータ（ラジコン用サーボモータ）は、PWM 信号を用いて角度を制御するアクチュエータである。内部には DC モータとギア機構、現在の角度を検出するためのポテンショメータ、そして制御回路などが組み込まれており、外部から与えられる PWM 信号に応じて自動的に所定の角度へと回転する。

一般的なサーボモータの PWM 信号の周期は 20[ms]で、パルス幅が 544[µs]から 2300[µs]の間で動作することが多い。Arduino のハードウェア PWM では簡単に周期 20[ms]のパルスを生成することができない。そこで、Servo ライブラリを用いて、タイマーの割り込み制御を使用して、パルスを生成している。

Servo ライブラリでは write 関数を用いて、整数でサーボモータの角度を指定することができる。より細かな制御を行うには、writeMicroseconds 関数を用いるのが良い。writeMicroseconds 関数では、直接パルス幅を指定することができる。

今回使用するサーボモータを、writeMicroseconds 関数を用いて動作させ、パルス幅の下限と上限を調べた。その結果、周期が 20[ms]でパルス幅が 700[µs]から 2300[µs]までであることが分かった。

1.2 実験装置

実験装置の CAD データを図 4-1 に、方眼用紙とサーボモータを設置後の様子を図 4-2 に示す。

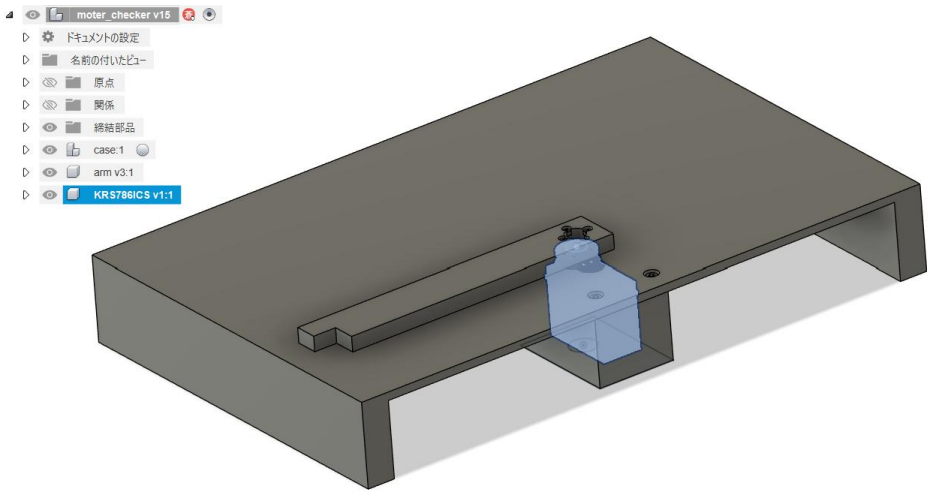


図 4-1 サーボモータ精度測定装置

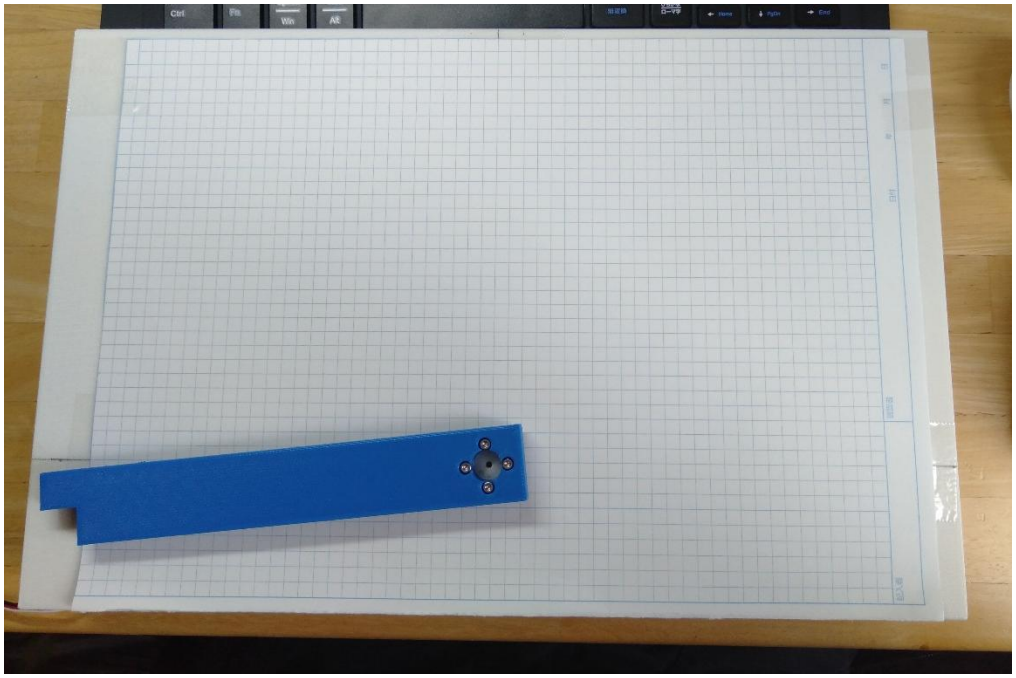


図 4-2 サーボモータと方眼用紙を設置した様子

1.3 実験方法

実験①

① モータを実験装置に取り付け、方眼用紙を設置する。

② パルス幅を 700[μs]から 5[μs]ずつ増加させ、それぞれの場所でアームの先の位置を記録する。

実験②

① モータを実験装置に取り付け、方眼用紙を設置する。

② パルス幅を 700[μs]に設定し、サーボモータを基準まで動かす。

③ パルス幅を 5[μs]増加させた 705[μs]に設定し、アームの先を記録する。

④ 再度パルス幅を 700[μs]に設定し、アームを基準まで戻し、パルス幅を 710[μs]に設定しアームの先を記録する。

⑤ このようにアームを基準まで戻し、そこからパルス幅を 5[μs]ずつ増加させた値を記録していく。

実験①だけではなく、実験②を行う理由として、モータが基準から特定の位置まで回転する方が、パルス幅が 5[μs]分だけ増加するよりも、勢いがあり、誤差が発生すると思ったからである。

1.4 結果

サーボモータ A の実験①と実験②の結果、サーボモータ B の実験①の結果を表 4-1 に示す。実験方法で述べた通り、実験②を行うのは、実験①よりもサーボモータが勢いよく回転し、誤差が大きくなると考えたためである。しかし、サーボモータ A の実験①と実験②の結果から誤差はほとんどないことが分かる。そのため、サーボモータ B の実験②は実験①と同様になることが明らかだったため、実験②を行わなかった。また、サーボモータにパルスを送信した際、特定の角度に収束せず、振動する現象が見られたが、軽く触れることで振動を止め、角度を記録した。

表 4-1 には、実験結果から逆三角関数を用いて求められる角度を測定値とし、サーボモータの最大の角度から算出された、特定のパルス幅に対する理論的な値を理論値としてそれぞれまとめた。また、測定値と理論値との差を誤差としてまとめた。

表 4-1 サーボモータ精度測定の実験結果

パルス幅[μs]	サーボモータA実験①					サーボモータA実験②					サーボモータB実験①				
	プロット[mm]		角度[°]			プロット[mm]		角度[°]			プロット[mm]		角度[°]		
	x軸	y軸	測定値	理論値	誤差	x軸	y軸	測定値	理論値	誤差	x軸	y軸	測定値	理論値	誤差
700	-100	-11	0.00	0.00	0.00	-140	-14	0.00	0.00	0.00	-140	-13	0.00	0.00	0.00
750	-140	-4	4.64	5.81	-1.17	-140	-3	4.69	5.86	-1.17	-140	3	6.53	5.74	0.79
800	-100	8	10.85	11.62	-0.77	-140	12	10.61	11.71	-1.10	-140	16	11.82	11.48	0.35
850	-137	24	16.21	17.42	-1.21	-140	26	16.23	17.57	-1.34	-140	31	17.79	17.21	0.58
900	-100	28	21.92	23.23	-1.31	-120	36	22.19	23.43	-1.24	-140	46	23.49	22.95	0.54
950	-135	52	27.34	29.04	-1.70	-130	53	27.89	29.28	-1.39	-130	57	28.98	28.69	0.29
1000	-100	52	33.75	34.85	-1.10	-110	57	33.10	35.14	-2.04	-125	71	34.90	34.43	0.48
1050	-123	78	38.66	40.66	-2.00	-116	75	38.60	41.00	-2.40	-120	86	40.93	40.16	0.77
1100	-95	75	44.57	46.47	-1.90	-100	81	44.72	46.85	-2.13	-110	96	46.42	45.90	0.52
1150	-105	101	50.02	52.27	-2.25	-100	98	50.13	52.71	-2.58	-100	107	52.24	51.64	0.60
1200	-93	110	56.06	58.08	-2.02	-70	84	55.91	58.57	-2.66	-85	111	57.86	57.38	0.48
1250	-85	122	61.41	63.89	-2.48	-82	122	61.80	64.42	-2.62	-76	123	63.59	63.12	0.48
1300	-70	126	67.22	69.70	-2.48	-70	130	67.41	70.28	-2.87	-60	120	68.74	68.85	-0.11
1350	-57	135	73.39	75.51	-2.12	-55	136	73.69	76.13	-2.44	-50	133	74.70	74.59	0.11
1400	-43	140	79.20	81.31	-2.11	-40	136	79.32	81.99	-2.67	-35	130	80.24	80.33	-0.09
1450	-27	145	85.73	87.12	-1.39	-27	145	85.16	87.85	-2.68	-23	141	86.04	86.07	-0.03
1500	-13	140	90.97	92.93	-1.96	-8	130	92.19	93.70	-1.51	-8	140	92.03	91.80	0.23
1550	3	145	97.46	98.74	-1.28	8	146	98.85	99.56	-0.71	7	145	98.07	97.54	0.53
1600	16	140	102.80	104.55	-1.75	15	100	104.24	105.42	-1.18	20	140	103.44	103.28	0.16
1650	33	140	109.54	110.35	-0.81	36	140	110.13	111.27	-1.14	33	135	109.04	109.02	0.02
1700	47	135	115.47	116.16	-0.69	50	134	116.17	117.13	-0.96	45	123	115.40	114.76	0.65
1750	61	128	121.76	121.97	-0.21	65	128	122.63	122.99	-0.35	61	125	121.32	120.49	0.82
1800	73	121	127.38	127.78	-0.40	75	117	128.37	128.84	-0.47	75	122	126.89	126.23	0.66
1850	85	111	133.72	133.59	0.13	89	110	134.69	134.70	-0.01	85	109	133.25	131.97	1.28
1900	100	104	140.15	139.40	0.76	100	100	140.85	140.56	0.30	95	99	139.12	137.71	1.42
1950	105	89	145.99	145.20	0.79	110	87	147.37	146.41	0.96	105	89	145.02	143.44	1.58
2000	118	78	152.81	151.01	1.80	115	75	152.60	152.27	0.33	120	85	149.99	149.18	0.81
2050	121	66	157.67	156.82	0.85	125	63	158.96	158.13	0.84	120	68	155.77	154.92	0.85
2100	126	53	163.46	162.63	0.84	130	50	164.67	163.98	0.69	130	57	161.63	160.66	0.97
2150	132	38	170.22	168.44	1.78	135	35	171.18	169.84	1.34	135	44	167.25	166.39	0.86
2200	135	25	175.79	174.24	1.54	135	22	176.45	175.70	0.76	135	29	173.18	172.13	1.05
2250	137	12	181.27	180.05	1.22	143	9	182.11	181.55	0.56	136	15	179.01	177.87	1.14
2300	138	-1	185.86	185.86	0.00	135	-4	187.41	187.41	0.00	135	4	183.61	183.61	0.00

表 4-1 の理論値を横軸に、測定値を縦軸に取ったグラフを図 4-3 に示す。

グラフからもわかる通り、同じサーボモータであれば実験①と実験②で同じ形、つまり同じ特性を示す。また、サーボモータ A は、角度が 130° 以下では負の方に最大-2.7° 誤差が存在し、以降では正の方に最大 1.8° 程度誤差が存在することが分かる。サーボモータ B は 1.6° 以内の誤差に収まっており、サーボモータ A よりも精度が良いことが分かる。

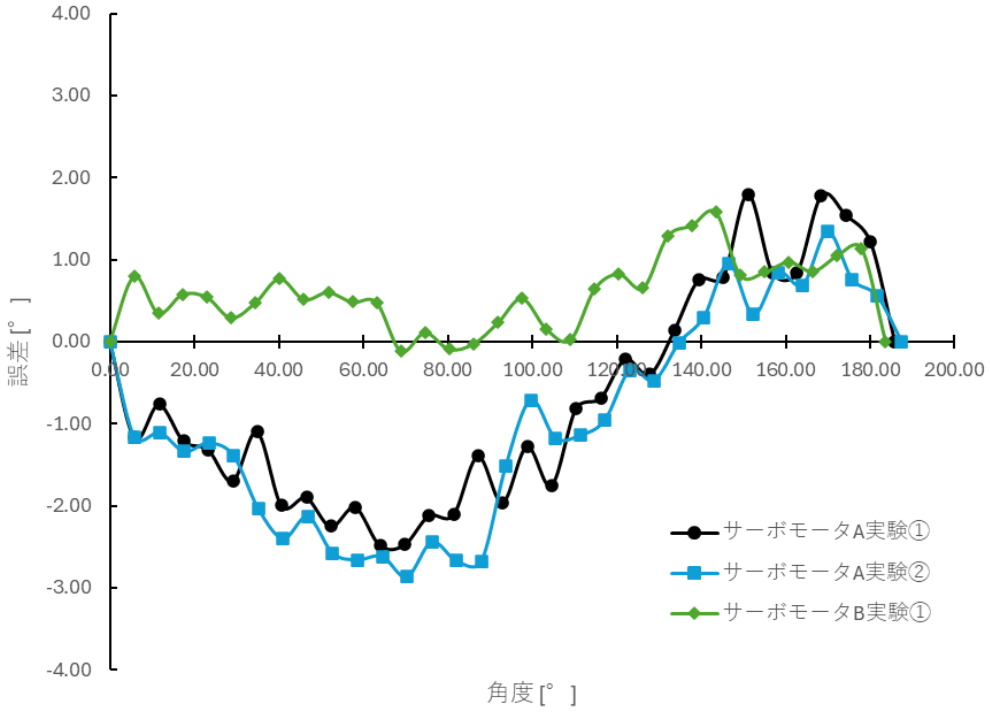


図 4-3 サーボモータの特性

1.5 考察

実験結果から、サーボモータ A とサーボモータ B ともに誤差が±2.7° 以内であることが分かる。またこの誤差には計測機器による誤差も含まれる。このことから、KRS-786 ICS サーボモータの精度は十分であると結論付けた。

また、私たちのマシンでは、x 軸を約 100[mm]、y 軸を約 150[mm]可動できるようにギヤを設計している。そのため、y 軸に使用するサーボモータの方が、サーボモータの誤差を大きくペン先に影響させてしまう。そこで、y 軸側のサーボモータに精度が高かったサーボモータ B を使用した。

図 4-3 から、サーボモータ A の誤差の特性を正弦波と近似して、プログラム上で修正することも検討した。しかし、渦巻を描くと大きなギヤに接続されている方のサーボモータ B による誤差で、失敗することが多く、これ以上サーボモータ A の精度を上げる必要がなかったため、近似して修正することは行わなかった。

実験結果で、サーボモータが静止せず振動したと述べた。これは、サーボモータが特定の角度に収束させようとフィードバック制御を行っているからではないかと考えた。具体的には、PID 制御のハンチングのような現象が生じていると考えられる。この振動を起こさせないようにするためには、ある程度の荷重をサーボモータにかける必要がある。

2. 3D プリント面の影響とその対策

3D プリンタで印刷した土台の上に方眼用紙を置き、渦巻と直線を描くと、震えているような線になる。これは、3D プリンタで印刷した土台の表面がざらざらしており、それが紙にまで伝わっているからだと考えられる。そこで、3D プリンタで印刷した土台の上に、クリアファイルを挟み、紙を置くことにした。クリアファイルがない場合の線を図 4-4、図 4-6（図 4-4 を拡大したもの）、ある場合を図 4-5、図 4-7（図 4-5 を拡大したもの）に示す。クリアファイルを引くことで、きれいな線が描けていることが分かる。

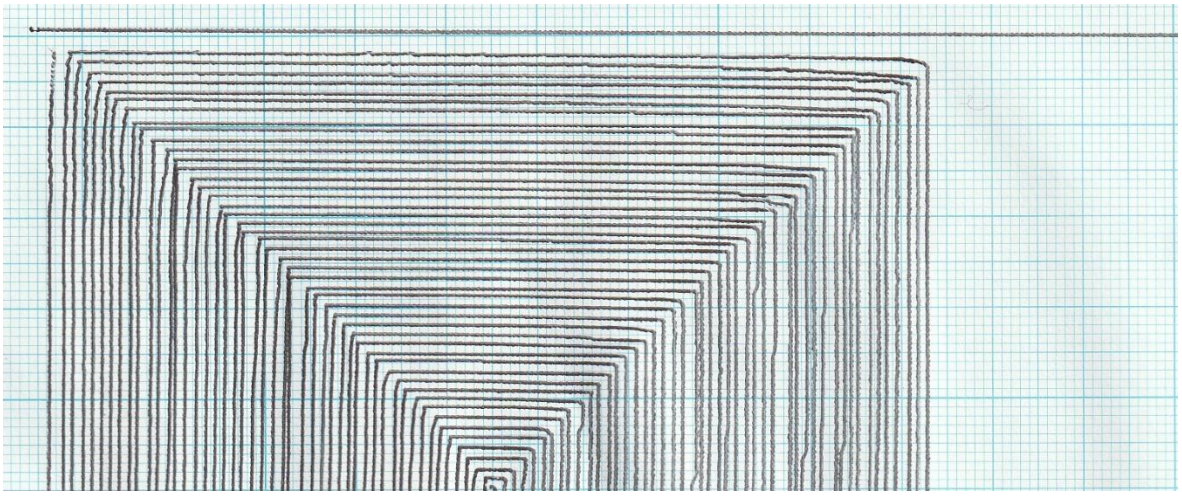


図 4-4 クリアファイルなしの渦巻

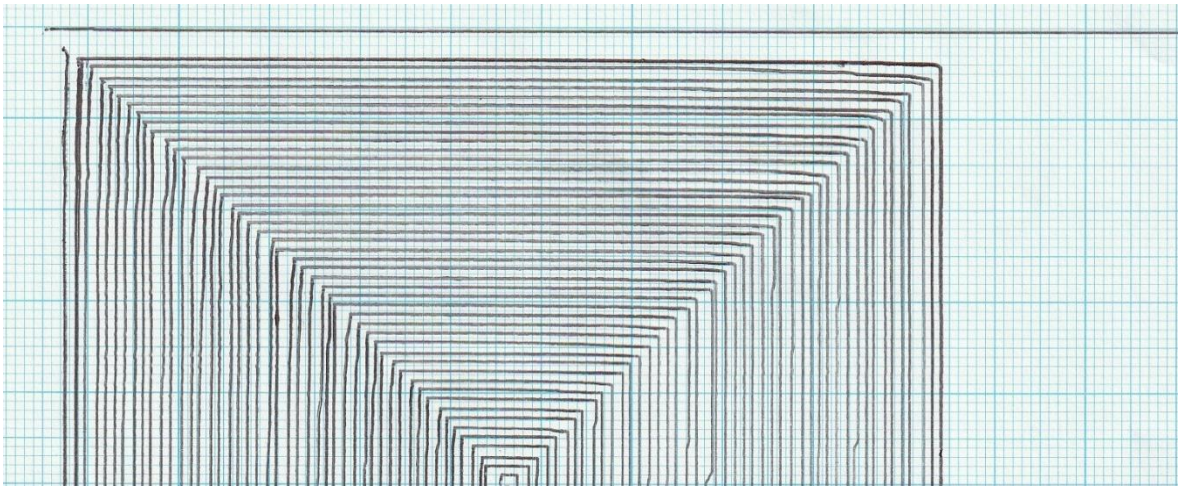


図 4-5 クリアファイルありの渦巻

図 4-6 クリアファイルなしの渦巻（拡大）

図 4-7 クリアファイルありの渦巻（拡大）

3. ペン固定機構

ペンを固定する部分の工夫点を 2 つ紹介する。

1 つ目は、輪ゴムを使用することで、一定の筆圧を加えることができる点である。図 4-8 に、輪ゴムとペンを取り付けた様子を示す。このように輪ゴムを用いることで、かすれることなく安定して線を描くことができる。

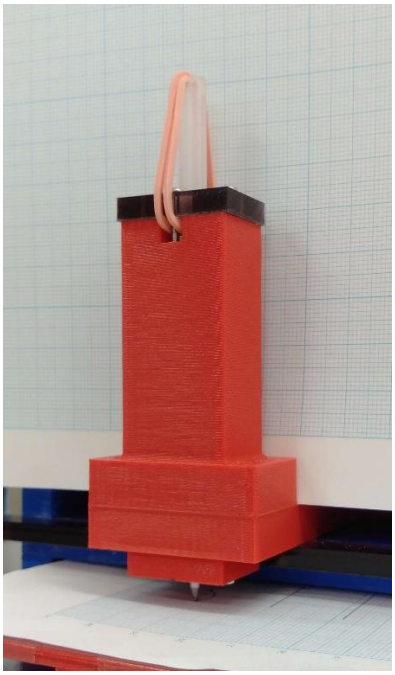


図 4-8 輪ゴムを用いたペン固定機構

2 つ目は、どのようなペンであっても、最小の変更で対応できる点である。私たちの班は、ハードウェアの設計段階で、いくつかのペンの種類を試すつもりでいた。そのため、どのようなペンであっても簡単に対応できるようにする必要があった。そこで、ペンのノズル部分の円の半径のみを変更することで、対応できるようにした。図 4-9 と図 4-10 にペンの固定部分の CAD データを示す。赤色で囲まれた 2 つの部品のみを交換することで、ペンの種類を変更することができる。

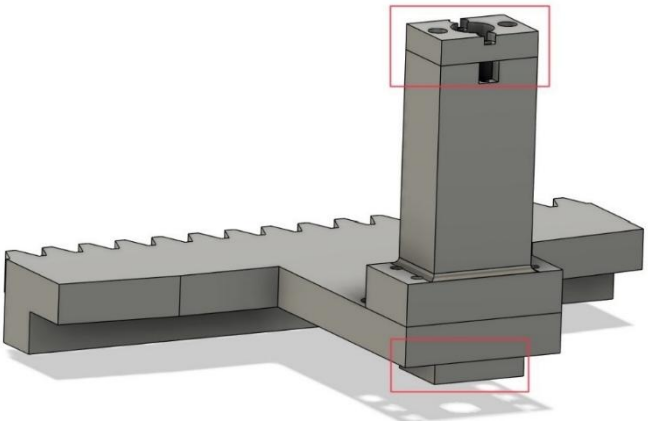


図 4-9 ペンの固定部分の CAD データ

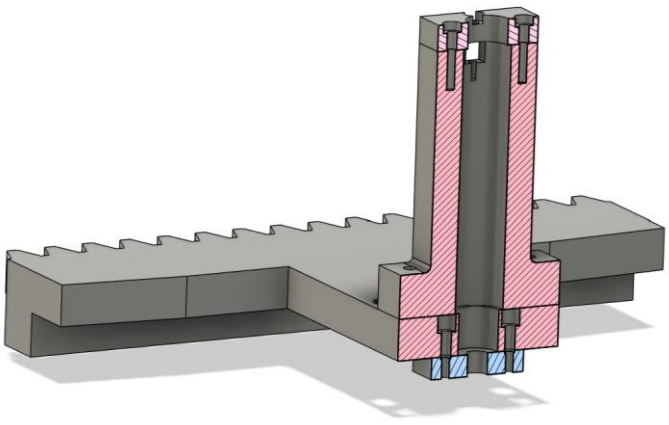


図 4-10 ペンの固定部分の CAD データ（断面図）

4. ペンの選定

ペンの芯が細ければ細いほど、渦巻を描くときに重なる確率が下がる。当初は、学校側から配布されたペンを使用しており、このペンの芯の細さは約 1.0[mm]だと推測される。そこで ZEBRA のジェルボールペンの芯が 0.3[mm]である「RJF3-BK」に変更した。学校から配布されたペンを用いた場合を図 4-11 に、RJF3-BK に変更した場合を図 4-12 に示す。図 4-12 の方が、図 4-11 よりも、細くなっていることが分かる。しかし図 4-12 では、渦巻の角の部分がにじんでいる。これは角の部分でペン先が一度止まることにより発生するものと考えられる。更に、図 4-11 では角にはにじみがないことに気づいた。このことからインクの種類によって、にじみを軽減できるのではないかと考えた。

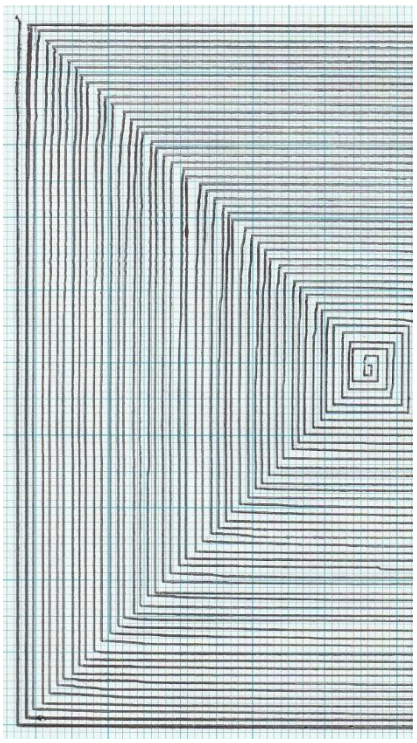


図 4-11 学校から配布されたペンの渦巻

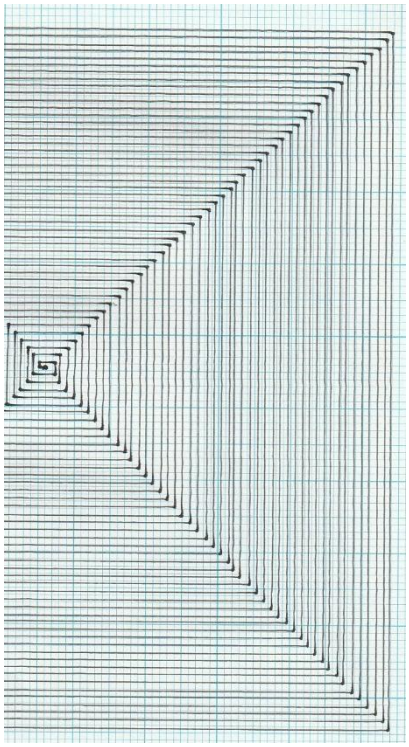


図 4-12 RJF3-BK ペンの渦巻

ボールペンのインクの種類には主に油性、水性、ジェルインク（ゲルインク）がある。それぞれの特性を図 4-13 にまとめた。ジェルインクは、水性に近いに特性を示す。そのため、粘土が低くにじみやすい。油性ペンは粘土が高くにじみにくい。しかし、粘土が高すぎて、ペン先が詰まりやすいので、そもそも細い芯が存在しない。そこで、油性に近い特性を持ち、滑らかな書き心地も持つ、三菱鉛筆のジェットストリームの「SXR-38」に変更することにした。SXR-38 の芯の太さは、0.38[mm]である。

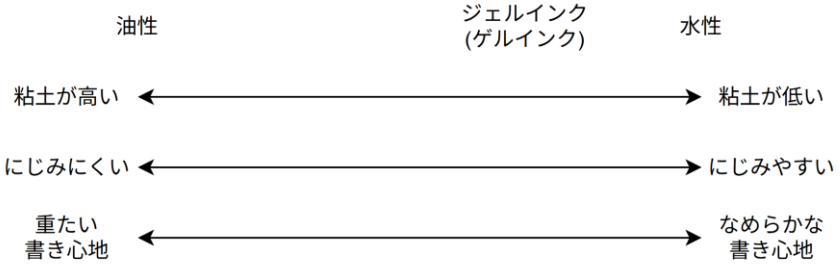


図 4-13 ボールペンの種類と特性

ジェットストリームに変えた場合を図 4-14 に示す。角の部分ににじみも少なく。細いペン先を実現できている。

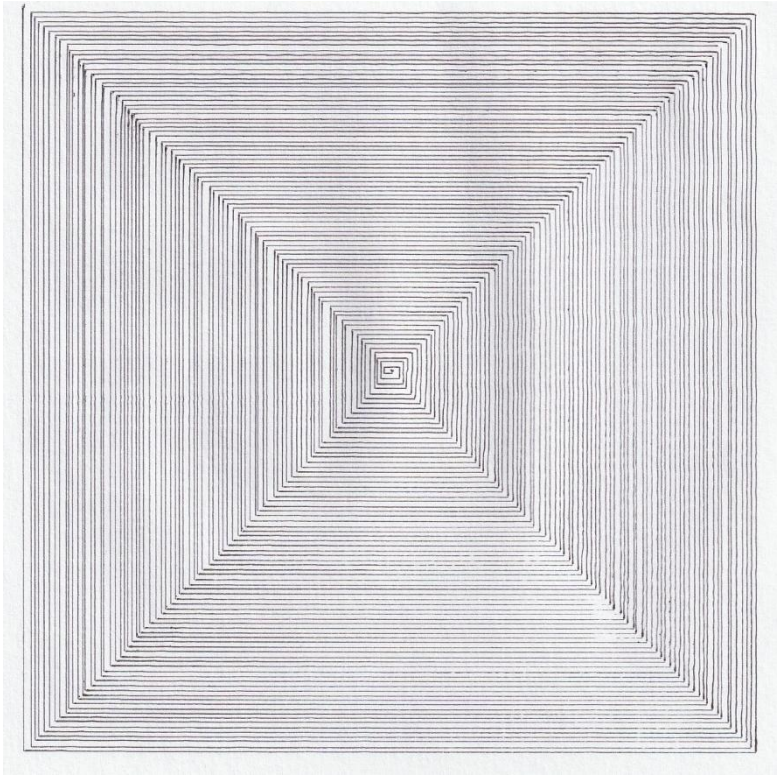


図 4-14 SXR-38 ペンの渦巻

5. 脱力

渦巻を描く際、図 4-15 のように、直線の途中でずれてしまい、となりの線と重なってしまう。また、この時のずれは動かしていない方のサーボモータよって生じている。具体的に説明すると、x 軸に平行な線を描くために、y サーボモータ固定で、x サーボモータを動かしているとする。y サーボモータは固定しているはずなのに、微妙に動作し、ずれが生じている。これは、「4.動作確認と諸修正の 1.サーボモータ精度測定」の結果・考察で述べた、サーボモータのフィードバック制御による振動が原因だと考えられる。サーボモータは指定された角度を維持しようとしている。そのため、動作させている方のサーボモータの微小な振動や、力に対して過剰に反応し、フィードバックしようとしてしまっているのではないかと考えた。そこで、動作させていない方のサーボモータのフィードバックを停止させ、サーボモータの制御をおこなわない方法はないのか検討した。

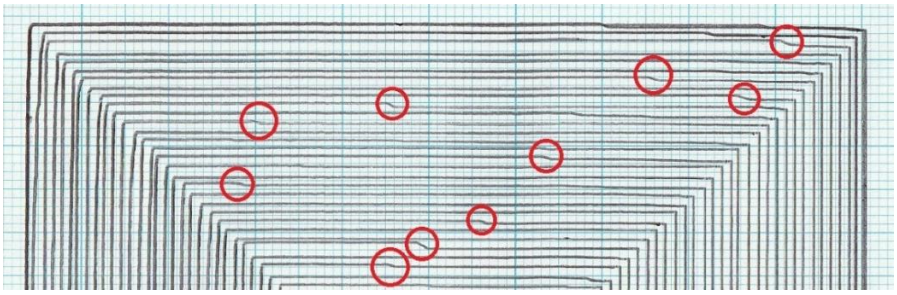


図 4-15 隣の線と重なる様子

サーボモータには Arudino の電源のプラスとマイナス、制御用の線の 3 つが接続されている。試しにサーボモータの角度が指定されている状態で、サーボモータの角度を手で変えてみようとした。しかし、フィードバック制御がかかっているため、元の角度を維持しようと逆方向の力が働いた。そこで、電源のプラス線を抜いてみた。すると脱力することが分かった。また制御用の線も抜いてみると、こちらも同様に脱力した。しかも電源のプラスを抜くよりも、適度に力が加わった状態で脱力した。これは、サーボモータ内の DC モータがショートブレーキを起こしているためではないかと考えられる。

電源のプラス側をソフトウェア的に制御するとなると、GPIO ピンに直接接続したくなるが、サーボモータには大きな電流が流れるため、大変危険である。そのためトランジスタを用いて、スイッチングする必要がある。その点、制御線を用いると、GPIO ピンから直接接続して、制御できる。そこで後者の制御線を制御する方式をとることにした。制御線を用いて制御する場合、パルスを生成せずに、LOW 出力させればよいことが分かった。

これらの工夫を凝らす前の様子を図 4-16 に、制御線を制御し脱力するプログラムを取り入れた時の様子を図 4-17 に示す。図からもわかる通り、線を引く途中でずれが生じていないことが分かる。

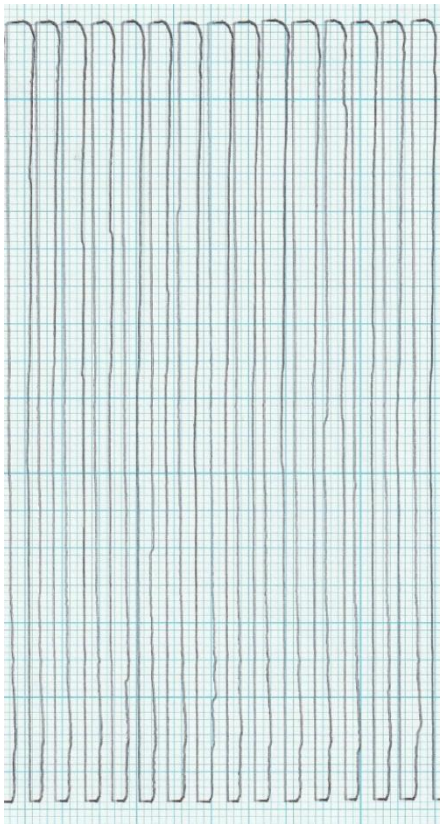


図 4-16 脱力なしの直線

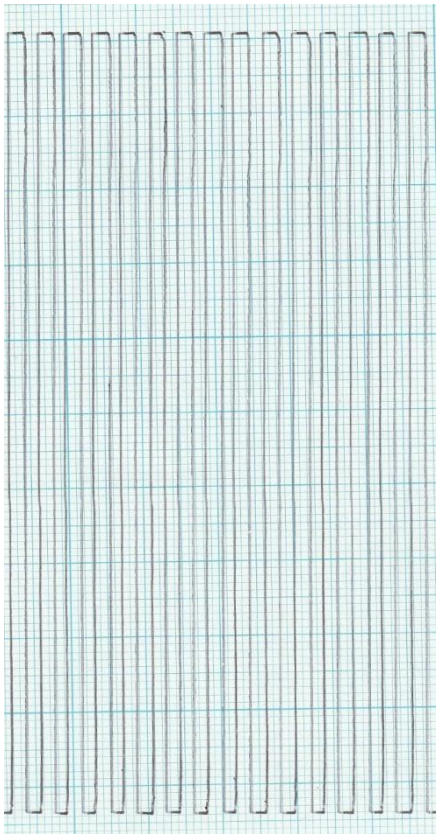


図 4-17 脱力ありの直線

5. 競技会に向けて

前章では、マシンの様々な工夫点を紹介した。この章では、それらの工夫を適応し、競技会当日、最大限のパフォーマンスを発揮できるようにする方法を検討する。

1. データの取得

渦巻モードにおいて、どれぐらいの pitch の細さまで描くことができるのか検証した。5 回連続で成功する pitch の値から 0.1[mm]ずつ減らし検証した。その結果を表 5-1 にまとめる。

この結果から、0.7 [mm]では安定して動作していることがうかがえる。

表 5-1 pitch と成功率の関係

pitch[mm]	渦巻の回数[回]	実施回数[回]	成功回数[回]	成功率[%]
0.8	60	5	5	100%
0.7	69	5	4	80%
0.6	80	5	1	20%

2. フローチャート

競技会では、2 回まで渦巻を描きなおすことができる。そこで、前項の結果をもとに、当日どのような pitch を採用するかフローチャートにまとめた。

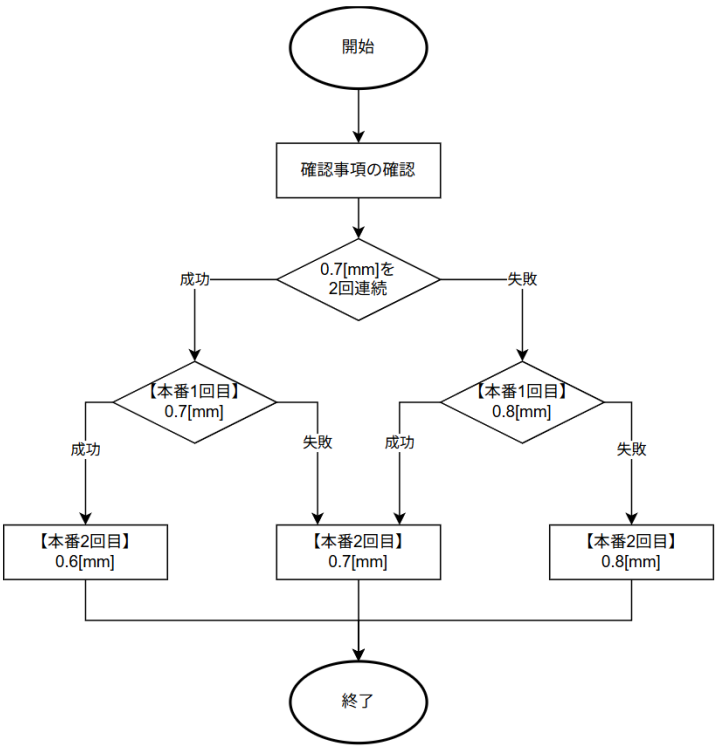


図 5-1 当日の流れ

3. 当日の確認事項

これらに加え、当日成功させるために、チェックリストを作成した。

準備物

- 予備のクリアファイル
- 予備の輪ゴム
- 予備の替え芯
- セロハンテープ

確認事項

- インクが正常に出ているか確認する
- y 軸のサポータが浮いていないか確認する

6. まとめ

この章では、マシンやプロトタイプ演習全体を振り返る。競技会についての振り返りがないのは、この報告書の提出日が、競技会当日であるからだ。

1. マシンを振り返る

「2.機構の選定と設計」で述べた通り、私たちのマシンは x-y 軸型を採用した。予想通り、直線が非常に安定して描くことができるマシンとなった。また、たくさんの周回数描くことができた。また、他のチームの機構を見ていると、アーム型やコンパス型でも非常に精度の高い結果を残しているチームがいた。私たちのチームは、去年の報告書から精度がでないと考えたが、実際は上手く設計することで、高い精度を出せるのだと感じた。

ペンの種類を簡単に変更できるような機構にしたのは、非常に便利でよかった。しかし、実際はペンの替え芯に対応したマシンにすれば十分だと感じた。ペンの替え芯に対応させると、ペンよりもサイズが小さいためコンパクトになる。また、どのような種類のペンでも替え芯はほぼ同じ形であるため、対応がしやすいからである。

2. プロトタイプ演習を振り返る

プロトタイプ演習を通して、良かった点は、サーボモータの精度を測定したり、成功率の確立を求めたりと、さまざまな実験を行い、根拠のもと、改良を行えた点である。また、少しでも渦巻の回数が多くなるように、ペンの細さや、種類までにこだわることができた点が良かった。

良くなかった点は、機構の選定に時間がかかってしまい、テスト期間などにも開発をすることになった点である。スケジュールを立てる段階で、余裕を持たせることが大切だと改めて感じた。また、3D プリンタを使いすぎた点もよくなかった。3D プリンタの印刷に時間がかかるのと、3D プリンタが空いていなければ開発がストップしてしまうからだ。3D プリンタの造形物は、想像以上に強度が高いので、薄型にしたり、余分な部分を削ったりして、造形物の体積を減らしていきたい。

3. 後輩へのメッセージ

やる気のあるうちに、できるだけ早い段階で開発をどんどん進めていくのがよいと思います。意外にも、ある程度動くものができてからの改良が楽いので、まずは早めに「動くもの」を作ることを目指してみてください。また、周りの班が高い記録を出し始めると焦るかもしれませんが、冷静に。自分たちにできること、自分たちの開発を着実に進めていけば大丈夫です。頑張ってください！

（最重要）

3D プリンタを使いすぎると、周りからブーイングが大量に飛んできます。ほどほどに。

4. 後輩へのお土産

連結部分や、サーボモータの固定など、来年度も引き継いで使えそうな設計のコツを紹介する。上手くいかなくても自己責任でお願いします。

図 6-1 のオレンジと紫の部品を M2 か M3 のねじで固定する場合を考える。このとき、僕が考える（3D プリンタでの）最適な直径のサイズを表 6-1 に示す。また M2 ねじ用のタップを作る道具がないので、力ずくで留められるぐらいのサイズを示している。

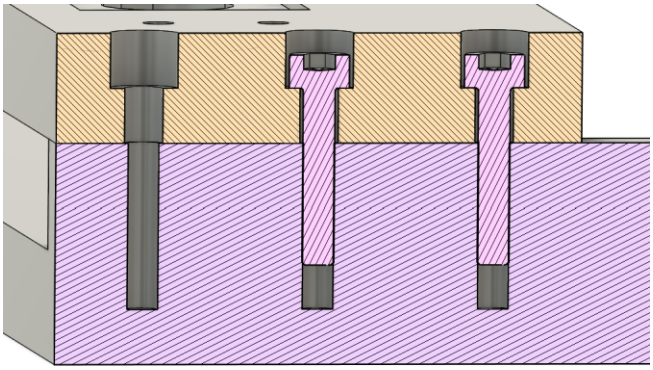


図 6-1 連結部分の設計

表 6-1 ねじ穴の最適なサイズ

場所	M3 ねじ	M2 ねじ
頭部	6.2mm	4.5mm
ねじ部	3.5mm	2.5mm
ねじ部(タップ用)	2.8mm	1.9mm

次に、サーボモータを板で通して固定する場合の設計書を図 6-2 に示す。

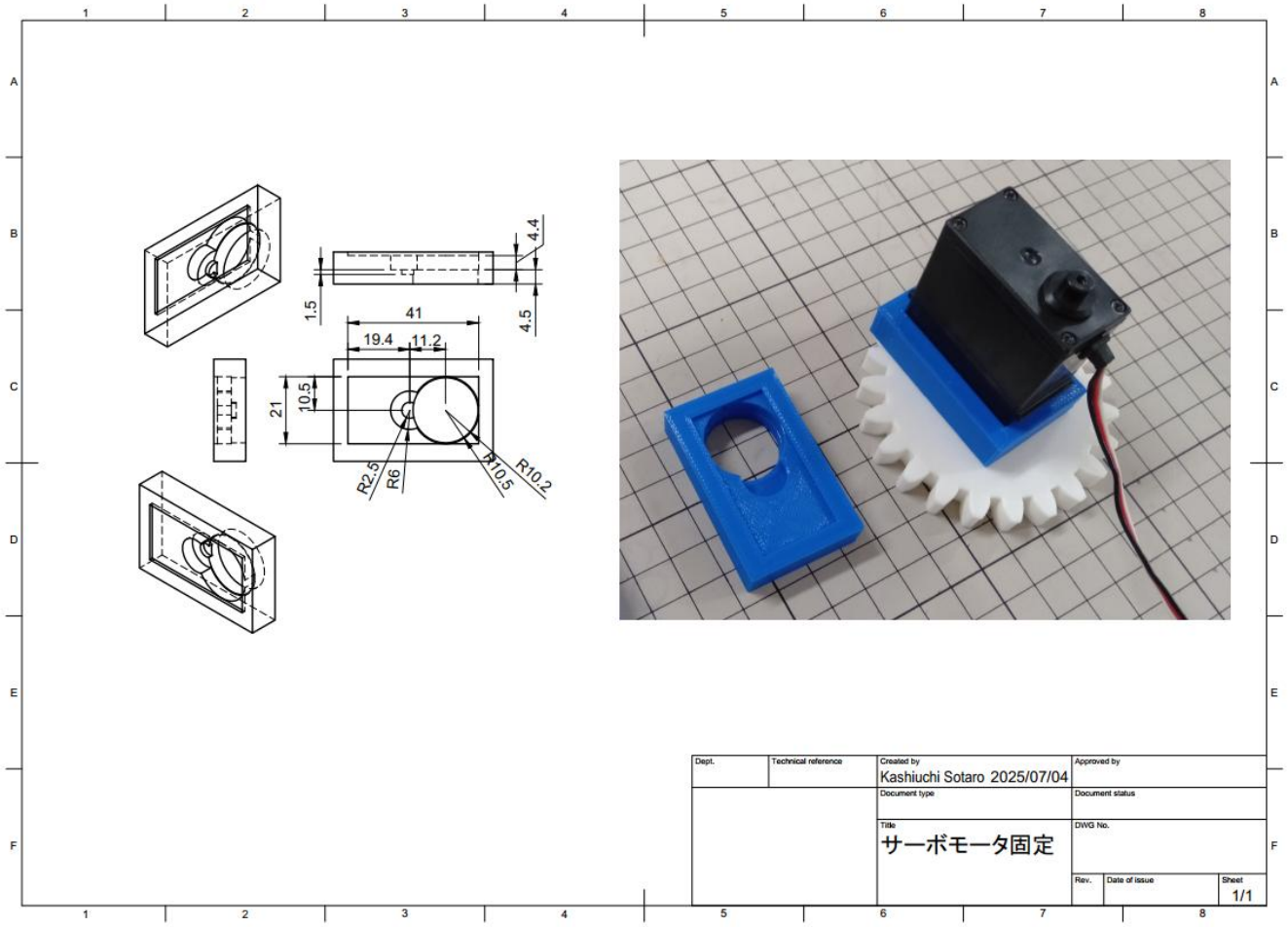


図 6-2 サーボモータ固定の設計

7. 付録

1. 付録 A メインプログラム

```
#include <Servo.h>
#include <math.h>

// ギヤの円ピッチ
#define X_CP (3 * 22)
#define Y_CP (3 * 32)

// 距離の分解能
#define DISTANCE_RESOLUTION 0.1

Servo servo_x;
Servo servo_y;

struct Position_t {
  double x;
  double y;
};

Position_t position_now = { 0, 0 };

void set_position(double x, double y, int delay_time) {
  if (!((0 <= x && x <= X_CP * M_PI * 0.5) && (0 <= y && y <= Y_CP * M_PI * 0.5))) {
    Serial.print("Error: larege size. ");
    Serial.print(x);
    Serial.print(" ");
    Serial.print(y);
    Serial.println("");
    return 0;
  }

  servo_x.writeMicroseconds((x * 1600 * 2) / (X_CP * M_PI) + 700);
  servo_y.writeMicroseconds((y * 1600 * 2) / (Y_CP * M_PI) + 700);
  position_now.x = x;
  position_now.y = y;

  Serial.print("x:");
  Serial.print((x * 1600 * 2) / (X_CP * M_PI) + 700);
  Serial.print(" y:");
  Serial.print((y * 1600 * 2) / (Y_CP * M_PI) + 700);
  Serial.println("");

  delay(delay_time);
}

void wait_any_key(void) {
  Serial.println("Enter any key");
  while (Serial.available() == 0);
  while (Serial.read() != -1);
}
```

```
void line(double dx, double dy) {
  double distance = sqrt(dx * dx + dy * dy);
  double count = distance / DISTANCE_RESOLUTION + 1;
  Position_t position_start = position_now;
  servo_x.detach();
  servo_y.detach();

  if (dx == 0 || dy == 0) {
    if (dx == 0) {
      set_position(position_start.x, position_start.y + dy, 0);
      servo_y.attach(5);
    }
    if (dy == 0) {
      set_position(position_start.x + dx, position_start.y, 0);
      servo_x.attach(3);
    }
    delay((abs(dx) + abs(dy)) * 10 + 1000);
  } else {
    for (int i = 0; i < count; i++) {
      set_position(
        position_start.x + dx / count * i,
        position_start.y + dy / count * i,
        0);
      servo_x.attach(3);
      servo_y.attach(5);
    }
    delay(300);
  }
}

void setup() {
  // 初期化処理
  Serial.begin(250000);
  servo_x.attach(3);
  servo_y.attach(5);
}

void loop() {
  // プロンプトの表示・選択
  Serial.println("");
  Serial.println("1:渦巻");
  Serial.println("2:直線");
  Serial.println("3:脱力");
  Serial.print(">>>");
  while (Serial.available() == 0);
  int mode = Serial.parseInt();
  while (Serial.read() != -1);
  Serial.println(mode);

  if (mode == 1) {
    // 渦巻モード
    double pitch = 0.6;
    set_position(2, 2, 1000);
    wait_any_key();

    for (double length = 96; 0 < length; length -= pitch * 2) {
      line(length, 0);
      line(0, length);
      line(-(length - pitch), 0);
      line(0, -(length - pitch));
    }
  }

  if (mode == 2) {
    // 直線モード
    set_position(100, 0, 5000);
    servo_x.detach();
    wait_any_key();
    line(0, 150);
```

```
  }

  if (mode == 3) {
    // 脱力モード
    servo_x.detach();
    servo_y.detach();
    wait_any_key();
    servo_x.attach(3);
    servo_y.attach(5);
  }
}
```

2. 付録 B サーボモータ精度測定プログラム

```
#include <Servo.h>
#define START_POSITION 700
#define RESOLUTION 5

Servo servo;

void setup() {
  Serial.begin(250000);
  Serial.setTimeout(10);
  servo.attach(3);
}

int set_position(int default_position){
  while(Serial.available()==0);
  int target_position = Serial.parseInt();
  while(Serial.read()!=-1);
  if(target_position==0){
    target_position=default_position;
  }
  Serial.print(target_position);
  Serial.println(" [ms]");
  return target_position;
}

void loop() {
  // 実験 1
  for(int target_position=START_POSITION; target_position<20000;
target_position+=RESOLUTION){
    target_position = set_position(target_position);
    servo.writeMicroseconds(target_position);
  }

  // 実験 2
  for(int target_position=START_POSITION; target_position<20000;
target_position+=RESOLUTION){
    target_position = set_position(target_position);
    servo.writeMicroseconds(START_POSITION);
    delay(1000);
    servo.writeMicroseconds(target_position);
  }
}
```