

機械システム工学実習Ⅱ 最終報告書

実習報告書：2023 年 1 月 30 日

実習 場所：E2 棟 611 号室

班 名：C 班

学生番号：20T1126N

報告者氏名：YANG GUANGZE(楊 広沢)

共同実習者：

板倉春太郎
福永 智樹
石崎 真
山田 尚輝
福田 祐介
藤田 生吹

1. 計画表

C 班は中間報告書と同じく、以下の表 1 と表 2 に示すような計画を立てた。中間発表までに、回路設計を完成し、コース周回と車庫入れのプログラミングを実装する。ボタン押し機構の設計を進めるところまで進めておく。中間発表後に、3D プリンタ、機械加工によるボタン押し機構の作成、ゴール探索プログラミングを実装する。余裕を持って、最後の 3 回を仕上げ、全体チェックに当てている。作業が滞った場合にこの予定を、全体チェックへ繰り下げる。以上のように計画表を作り上げた。

表 1. 計画表 (1)



日程	10/5	10/12	10/19	10/26	11/2	11/9	11/16	11/30	
	ガイダンス等	装置などの説明会 1	アセンブリ3	アセンブリ4	アセンブリ5	アセンブリ6	設計設計発表会	アセンブリ8	
作業	アセンブリ1	アセンブリ2	はんた付け、組立		テスト		アセンブリ7	"中間報告書提出"	
板倉		回路図の設計 ボタン押し機構の構想	回路制作組立	CAD,ボタン		修正、 バウボ作成	CAD 部品作成		
藤田								プログラム (前半部)	プログラム 全体修正
福永									
福田				プログラム回路制作					
山田			ボタン設計	CAD				CAD 部品作成	
梅			回路制作	プログラム (後半部)	ボタン			配線見直し	周回路プログラム
石崎									

表 2. 計画表 (2)

	12/7	12/14	12/21	1/11	1/18		1/25	2/1	最終報告書提出締切 部 品 回 収 作 業
アセンブリ9	アセンブリ10	アセンブリ11	アセンブリ12	アセンブリ13					
組立	組立	全体 チ エ ツ ク							
ボタン									
プログラム									
ボタン									
プログラム									

実際には、中間報告までに回路設計に関して不具合が発生し、回路作成が全体的に遅くなった。そして12/14に解決できたため、ボタン探索のプログラム、車庫入れ後の動

き、ボタン押し機構の仕上げが後ろ倒しになった。次に、H8 と Arduino の通信がうまくいかず、コンテストに間に合うことができなかった。コンテスト終了後にこの通信問題が解決したが、測距モジュールと超音波センサの不具合があり、ゴールエリアまで到達することができなかった。1/27 にセンサを想定通りに動作させることができ、ボタン押しまで完遂することができた（チャンピオン動画）。

2. 役割分担表

C 班の役割分担は、表 3 に示している。ボタン押すアイデアは全員に考えられる。二つグループを分けて電気回路を作成する。周回後のプログラムと周回後のプログラムを分けてプログラミングする。組み立てや実行テストは全員で行う。

表 3. 役割分担表

行程名	担当
回路図の構想	板倉 藤田
ボタン押し機構の構想	全員
回路制作(電源,マイコン)	板倉 藤田 福永 山田
回路制作(モータドライバ,エンコーダ)	福田 楊 石崎
回路制作(フォトリフレクタ)	楊 石崎
CAD,3Dプリントパーツ(ボタン押し機構) 制作, 機械加工	山田 福田 板倉 藤田
周回プログラミング(前半部)	板倉 藤田 福永 山田 福田
周回プログラミング(後半部)	楊 石崎
周回後プログラミング	楊 石崎
組立	全員

しかし、実際の場合では異なったことやお互いへのサポートがある。私は、明確にプログラムとフォトリフレクタ作成の担当になっている。そして、作業については、「全員」、自分の「個人作業」、メンバーたちとの「共同作業」に分けられる。以下は、自分の作業日誌である。

- ① 10/5：リーダーとサブリーダーの決定（全員），計画表の作成（全員），部品チェック（全員）とボタン押す構想の発表会（全員）
- ② 10/12：全員でのボタン押す構想の決定（全員）．H8 と Arduino の環境設定チェック（楊，福田の共同作業）．周回りのプログラムと車庫入れのプログラムの作成（個人作業）．
- ③ 10/19：周回りのプログラムと車庫入れのプログラムの修正（個人作業）．大基板の回路作成（福永，楊，

石崎の共同作業).

- ④ 10/26: 小基板の回路(フォトリフレクタ)作成と電源部の配線作成(楊, 石崎の共同作業). ロボットの組み立て(山田, 藤田, 楊の共同作業). 大基板の回路作成(福永).
- ⑤ 11/2: 周回りのプログラムと車庫入れのプログラムのチェック(個人作業). Arduino で超音波センサと測距モジュールのチェック(個人作業). ゴールまで探索の方法の構想(個人作業). センサ用の Arduino プログラムの作成(個人作業). 回路製作完成(福永, 福田, 石崎, 板倉).
- ⑥ 11/9: デモ機を利用して周回りと車庫入れのプログラムのテスト(全員). 周回りと車庫入れプログラムの速度調整(楊, 福田の共同作業). ボタン押す機構の問題対策(山田, 藤田, 楊の共同作業). ロボット実行テスト(全員). 回路のデバッグ(福永, 楊, 石崎共同作業).
- ⑦ 11/16: 中間発表(全員). Arduino プログラムの作成(個人作業). 回路のデバッグ作業(福永, 石崎).
- ⑧ 11/30: ゴール探索の方法を考え直し, Arduino で超音波センサと RC サーボモータを制御するプログラムの作成(個人作業). 回路のデバック作業(福永). ボタン押す機構の問題対策(山田, 藤田, 楊の共同作業).
- ⑨ 12/7 測距モジュールと超音波センサの役割修正, 新しいゴール探索の方法決定, Arduino プログラムの作成(個人作業). 回路のデバック作業(福永). ボタン押す機構の問題対策(山田, 藤田, 楊の共同作業).
- ⑩ 12/14 Arduino のセンサと RC サーボモータプログラム完成(個人作業). ロボット回路デバック作業完成(福永).
- ⑪ 12/21 ライントレース速度調整と車庫入れプログラム修正(福田, 藤田の共同作業). 車庫入れ後の H8 プログラム(個人作業).
- ⑫ 1/11 ライントレース速度調整と車庫入れプログラムの修正完了(福田, 藤田の共同作業). 探索方法の再確認(楊, 板倉, 福永). Arduino と H8 のプログラムの修正(個人作業). センサとマイコンの回路設計(個人作業). ボタン押す機構と Arduino 回路を組み立て, 実行テスト(全員).
- ⑬ 1/18 実行テスト(全員). H8 と Arduino の通信問題

が生じ, 回路とプログラムのデバック作業(個人作業).

- ⑭ 1/20 Arduino の不具合と H8 と Arduino の通信問題の対策(個人作業). 実行テスト(楊, 山田, 福永, 藤田の共同作業).
- ⑮ 1/25 コンテンツ後, H8 と Arduino の通信問題解決(個人作業). 実行テスト(全員).
- ⑯ 1/27 実行テスト, センサの値の不安定の問題解決, ボタン押すまで実行成功(板倉, 楊, 福永, 藤田, 山田の共同作業).

3. 移動ロボットの概要と概略図



図 1. 移動ロボット

図 1 は, 移動ロボットの概略図を示している. 移動ロボットの概要は, 主に以下の三つの部分に分けられる.

1. 両腕でボタンを掴む構造
2. 磁石を用いた鉄球で押す構造
3. センサとマイコン回路

両腕でボタンを掴む構造では, 3Dprint でできた二つの歯車を両腕と連携している. 基板の下にあるサーボモータ(RC1)は, その二つ歯車を駆動するようになっている.

磁石を用いた鉄球で押す構造では, 鉄球の滑り軌道がロボットの上に固定されている. 軌道の後ろに磁石は,

鉄アームに固定されている。磁力で鉄球を維持するため、軌道の尾部が薄い膜に封じされる。また、ロボットの後上にあるもう一つRC サーボモータは、その鉄アームをコントロールする。よって、走行中に磁力による鉄球を落とさず、ボタン押す時、サーボモータの回転によって磁石が離れ、鉄球が軌道に滑り、ボタン押せる。

センサとマイコン回路では、Arduino と H8 マイコンが二つのマイコンである。二つセンサについて、ロボット車体の側面（左側）に設置された測距モジュール（PSD）と、ロボット正面に設置された超音波センサ（SONIC）である。また、フォトリフレクタ、モータ、二つ電池ボックスなどが載っている。

また、以下の図 2, 3, 4, 5, 6 は、移動ロボットの各角度の外観を示している。各部品の位置については、以下のようになっている。

- ① 両腕
- ② 両腕用歯車（3Dprint）
- ③ 両腕駆動用サーボモータ（RC1）
- ④ 超音波センサ
- ⑤ 測距モジュール
- ⑥ 鉄アーム駆動用サーボモータ（RC2）
- ⑦ 磁石
- ⑧ 鉄球の滑り軌道
- ⑨ Arduino

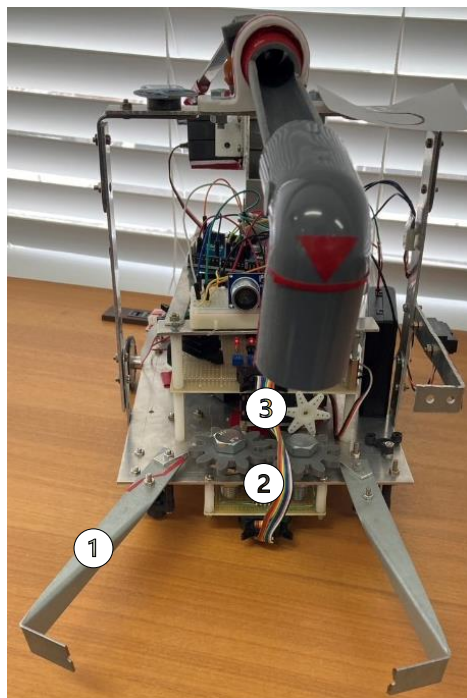


図 2. 移動ロボット（正面）

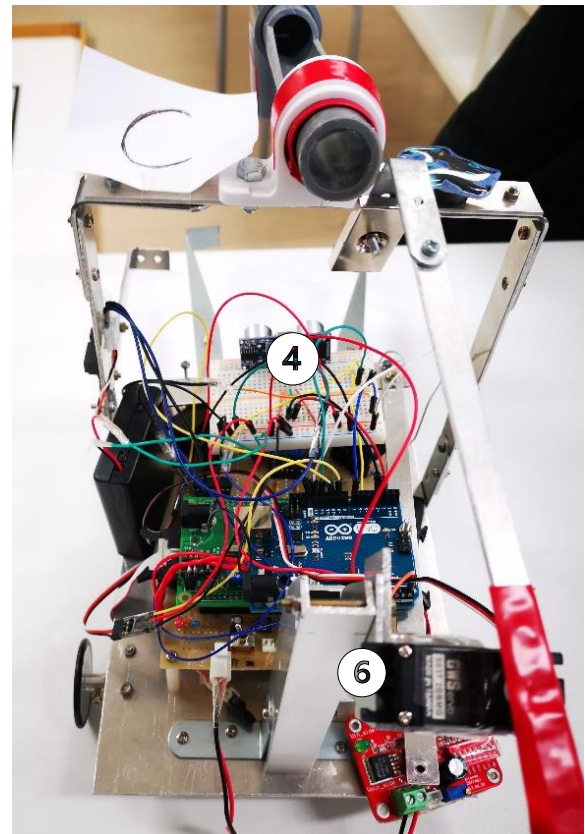


図 3. 移動ロボット（後面）

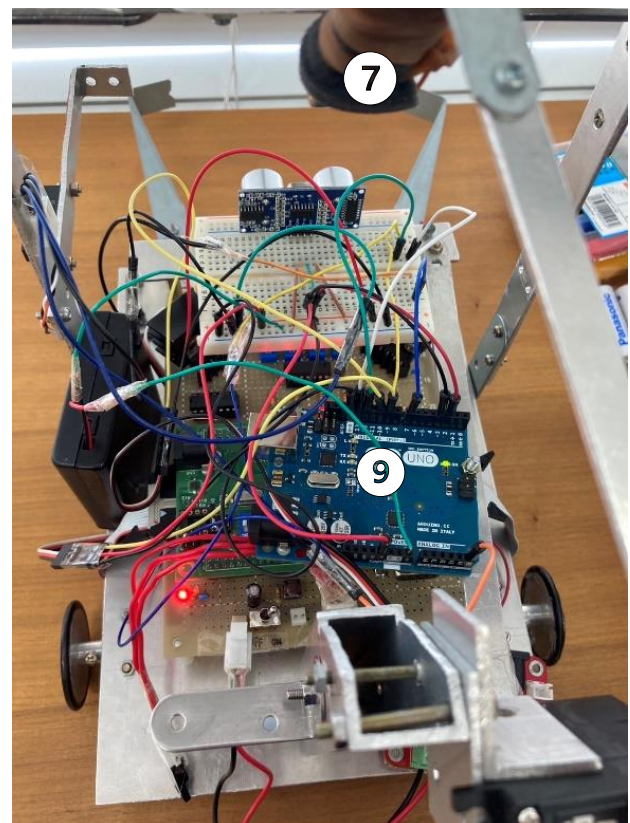


図 4. 移動ロボット（上）

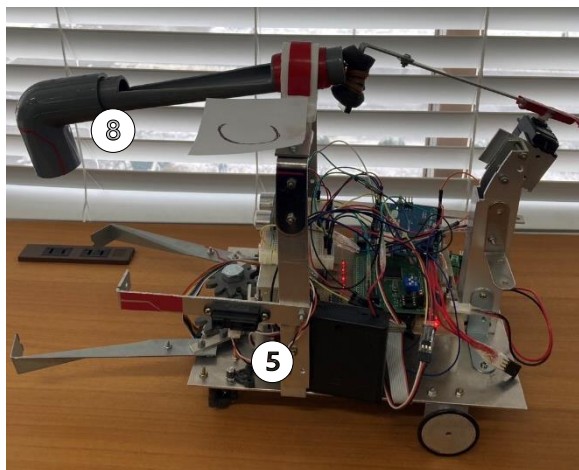


図 5. 移動ロボット（左側面）

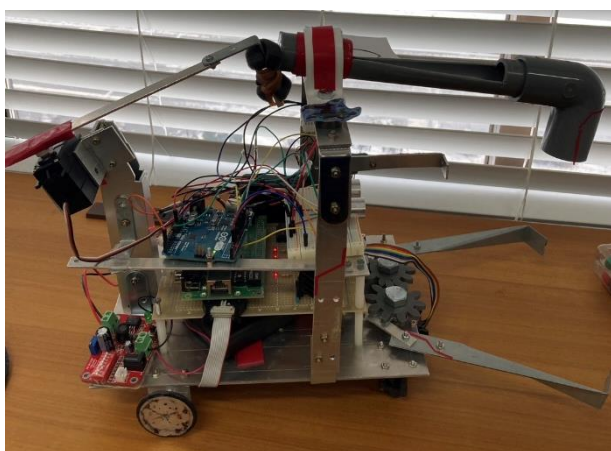


図 6. 移動ロボット（右側面）

4. 各設計図，部品図

以下の図 7～図 15 は，移動ロボット各部品の設計図と実物写真を示している。

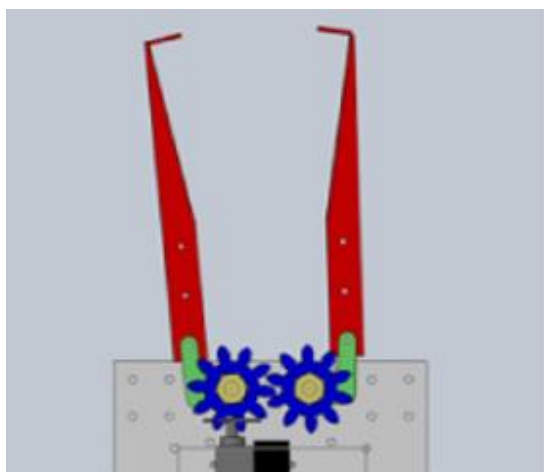


図 7. 両腕構造設計図

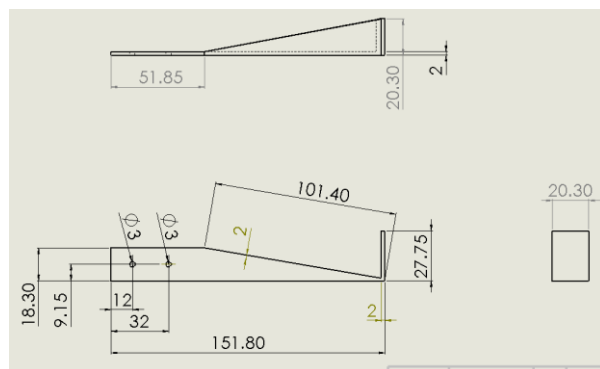


図 8. 両腕構造設計図(左腕)

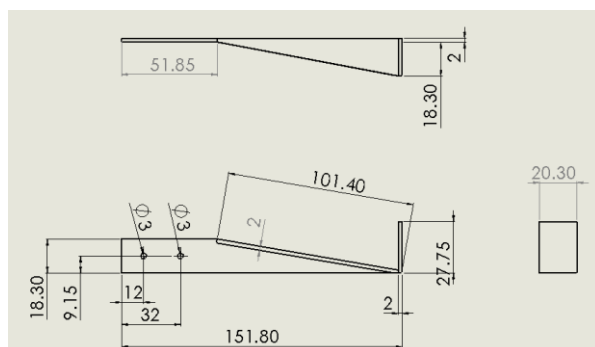


図 9. 両腕構造設計図(右腕)

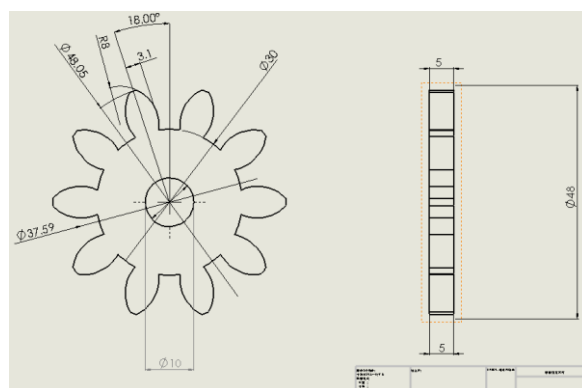


図 10. 歯車(3Dprint)設計図

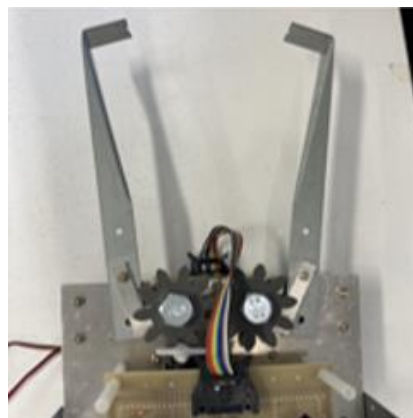


図 11. 両腕構造実物写真

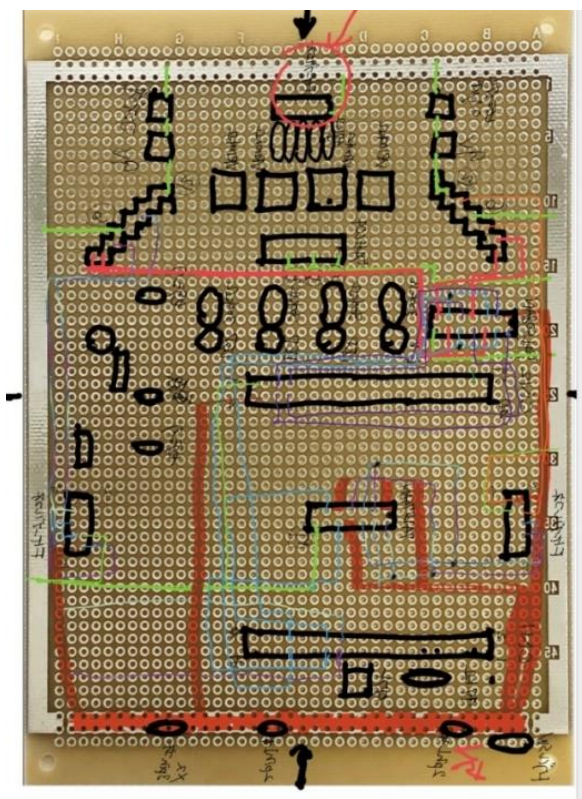


図 12. 基板設計図

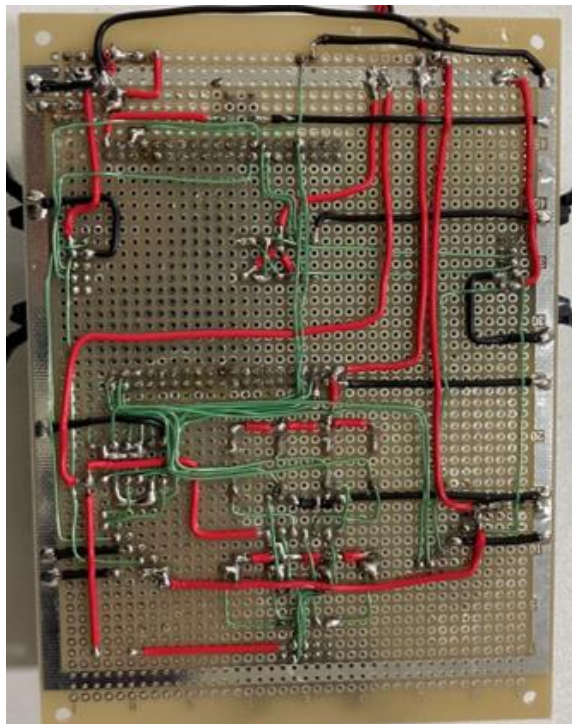


図 13. 基板写真（裏）

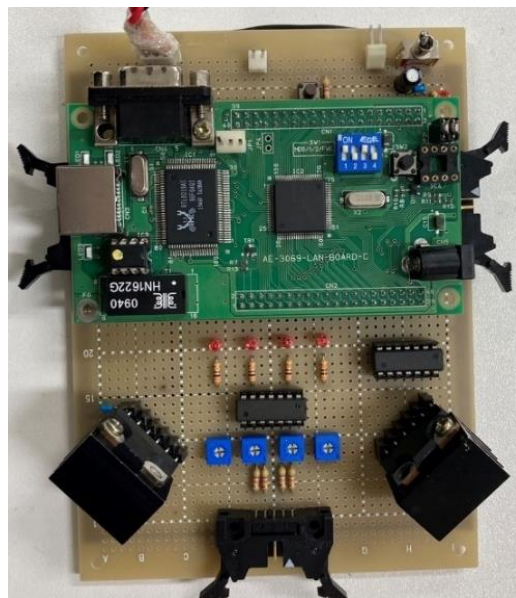


図 14. 基板写真（表）

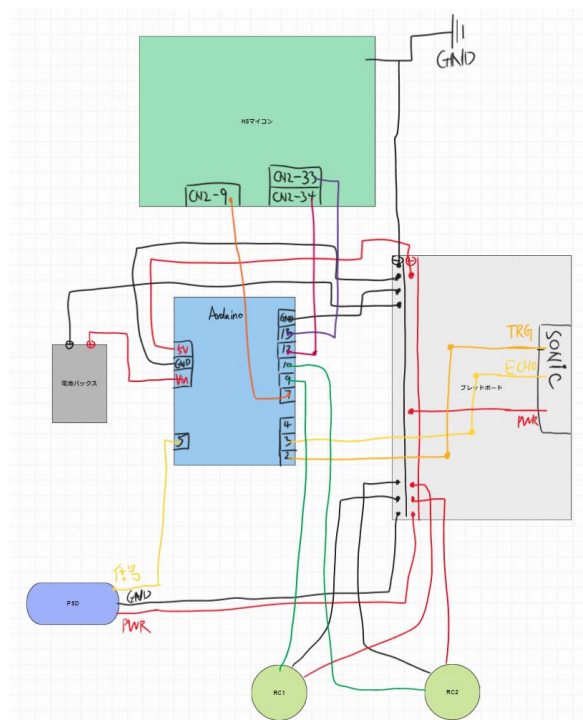


図 15. センサと Arduino 回路設計図

5. オリジナリティ

5.1 ボタン押すアイデアと機構

ボタン押す機構のアイデアについて、メンバーたちはそれぞれ前期の実験テーマ 6 の課題での構想を発表した。

「磁石を用いた鉄球発射構造」、「ボタンを倒して押す構造」、「ペンで押す構造」、「両腕でボタンを掴む構造」、「おもりで押す構造」、「腕の回転で押す構造」などの方案が挙げられる。

私のアイデアとしては、一つ目は、RC サーボモータの回転を直進に変換し、円管内の球体が落ちてボタンが押されることである。二つ目(図 16)は、薄い円管内(非金属)の鉄球を磁石で維持する。RC サーボモータの回転により、磁力の減少によって鉄球が落ちて、ボタンが押されることである。

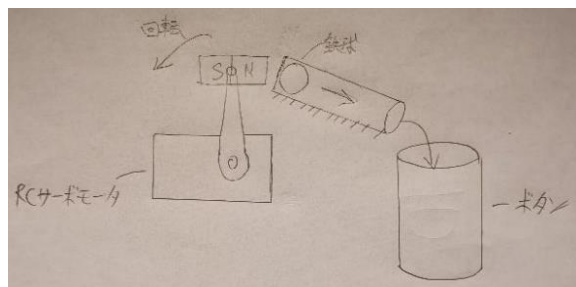


図 16. 磁石を用いた鉄球発射構造

全員の意見を交換した後に、「磁石を用いた鉄球発射構造」と「両腕でボタンを掴む構造」の組み合わせ方案を決定した。そして山田が CAD 製図を担当し、藤田が機械加工を担当し、全員と意見交換しながらアイデアを展開し、図 17, 18 に示すような「磁石を用いた鉄球発射構造」と図 19, 20 に示すような「両腕でボタンを掴む構造」を設計して製作した。

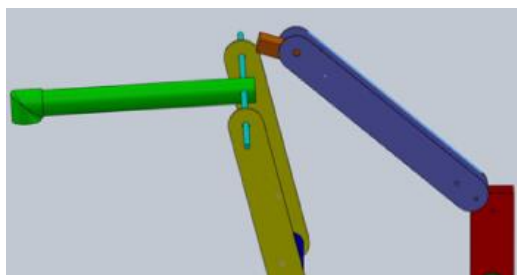


図 17, 磁石を用いた鉄球で押す構造 (1)

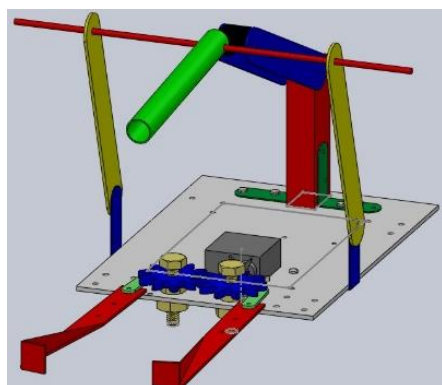


図 18, 磁石を用いた鉄球で押す構造 (2)

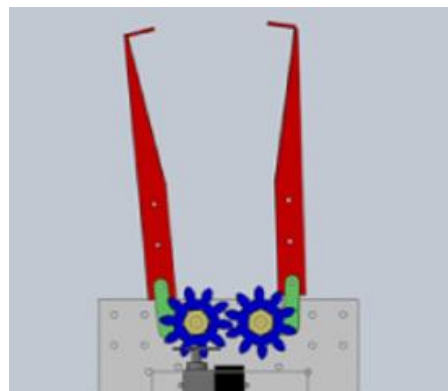


図 19, 両腕でボタンを掴む構造 (CAD モデル上面)

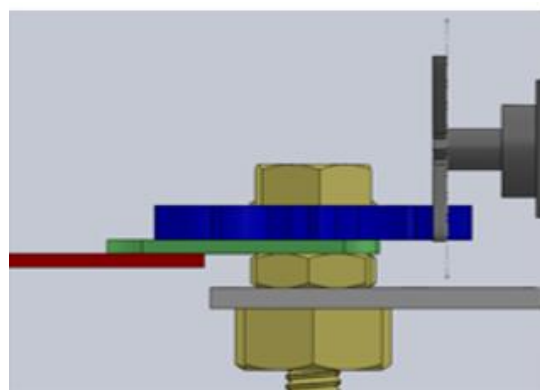


図 20, 両腕でボタンを掴む構造 (CAD モデル側面)

「磁石を用いた鉄球発射構造」については、緑色の円筒内には磁性を持つ大きな鉄球が入っており、走行中は後方のアームに設置されたオレンジ色の磁石によって引きつけられるため、落ちることはない。そしてボタン押しの際には、アームを引き上げ、磁石を鉄球から離すことで、自由となった鉄球が円筒内を通り、ボタンの上へ落ちるという仕組みになっている。円筒の先端にはベンドを取り付け、転がった鉄球が慣性の影響を受けず、真下に落ちるように工夫している。

「両腕でボタンを掴む構造」について、一つ目のポイントはモータに歯車を取り付け、モータの回転にあわせて青色の歯車が回転することにある。その青色の歯車は 3D プリンタで作成した。二つ目のポイントは、歯車の回転により赤色の角の部分が閉じたり開いたりすることにある。ここで、緑色の部分は角を柔軟に動かす関節の役割と歯の長さを延長する役割を果たしている。最後のポイントは真ん中に取り付けたナットにある。ナットは青色の歯車を高い位置に配置し、本体と触れることなくスムーズな回転を行

う補助としての役割がある。

ここまでには、中間報告までに設計した。しかし、ボタンを押すテストを実行した後、二つ問題がある、一つ目は、ボタンを押せるため、鉄球の重量が足りない問題が生じた。元の鉄球の滑り軌道の半径が小さいことで、鉄球のサイズを交換するために、鉄球の軌道も交換する必要がある。二つ目は、磁石の鉄アームを駆動する RC サーボモータ (RC2) の支持構造と回転方向は計画の通りに上手くいかないことである。

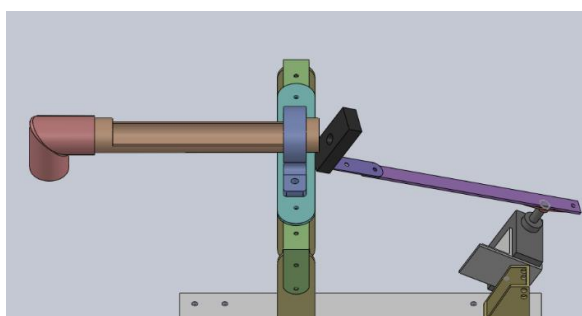


図 21, 磁石を用いた鉄球で押す構造 (3)

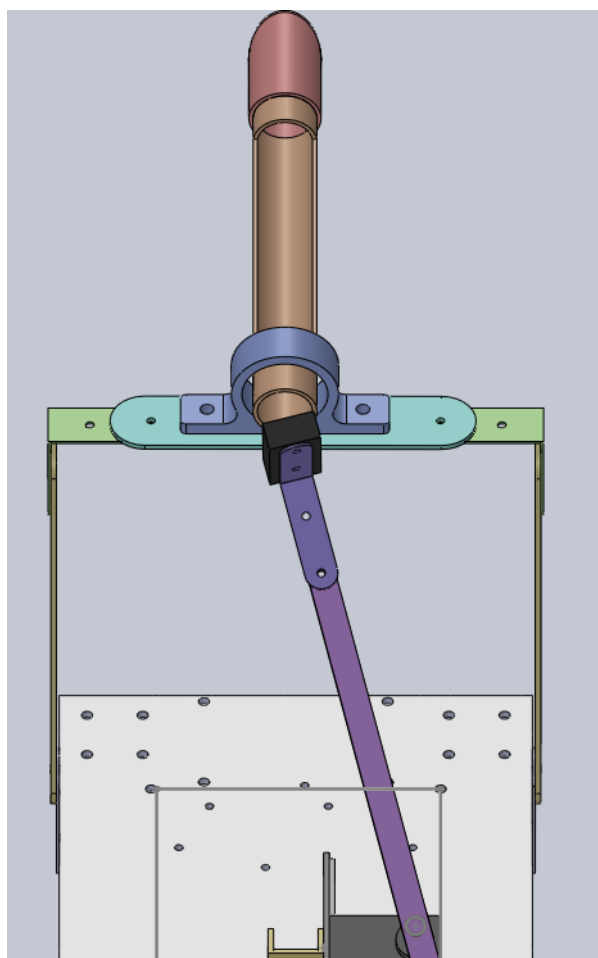


図 22, 磁石を用いた鉄球で押す構造 (4)

それらの問題を解決するため、図 21, 図 22 に示すように、磁石を用いた鉄球で押す構造を改良した。改良した点は、以下の二つがある。一つ目は、新たな鉄球の滑り軌道が元より太く、上側が露天である。よってもっと大きな鉄球を載せ、鉄球の運動を考察できる利点がある。二つ目は、サーボモータ (RC2) の回転方向を水平面に変換し、鉄球のコントロールが上手くなる。

5.2 回路設計と作成

回路作成については、図 12 は板倉と福田が作成した配線設計図を示している。図 13 と図 14 は主に福永が作成した大基板を示している。主に石崎が作成したフォトリフレクタ回路の小基板の写真は乗っていないが、回路作成の方針は、黒の動線が GND、赤が電源部分、緑がその他に繋がるよう動線の色分けをし、ボードの綺麗さとチェックの見易さのために、できるだけ配線の長さをびったりするように作成した。また、私は、図 15 に示すようなセンサと Arudino, H8 マイコンの回路を設計し、作成した。

5.3 プロフラムとゴール探索

私はプログラムの担当になっている。以下の三つの部分を分けて説明する。

1. ライントレース
2. 車庫入れの H8 プログラム
3. センサ配置とゴール探索方法
4. センサと Arduino プログラム
5. 車庫入れ後の H8 プログラム

5.3.1 ライントレース

まず、ライントレースと車庫入れの H8 プログラムでは、前期機械システム工学実験のテーマ 6 とテーマ 7 で作成したプログラムに基づき、今回のプログラムを作成した。テーマ 6 とテーマ 7 プログラムの重複した定義や関数を確認し、一つのプログラムのように組み合わせをする。作成中にプログラムのエラーが出たこともある。例えば、自分のミスで変数の名前を変えたせいで、定義されていない変数や関数のエラーが出た、詳しく確認して修正した。また、新たな「common.h」と「r3069.h」のソースファイルの更新し、最後エラーがなく出来た。

周囲りのプログラムでは、テーマ 6 で作成した 3 周回り

プログラムのライントレース部分を, コード 1 に示すように for 文の「lap_flag<3」と if 文の「lap_flag<2」を書き換え, 2 周回りになる. また, ライントレースの速度と安定を重視し, 表 4 が示すように「line_trace()」のフォトトリフレクタの 10 個パターンの速度調整を行った.

コード 1. 周回りプログラムの一部 (main())

```
for ( lap_flag=0 ; lap_flag<3 ;
lap_flag++) //2 周で終了{
    Feed_flag = PHOTO;// フィードバック
関数内での制御フラグ
    while( All_black_flag == 0 ){//コース 1
周内-->line_trace で処理 (while を抜ける
条件は十字マーカー)

        sleep(1);// 1[ms] 値は変更しない}
        if(lap_flag <2 ){//sleep(200)
            while( Photo_1 == Black &&
Photo_2 == Black && Photo_3 == Black
&& Photo_4 == Black){
                motor(LW,250*LD);
                motor(RW, 250*RD);
            }
        }
        All_black_flag = 0; // フラグを戻す
    }
```

表 4. フォトリフレクタの各パターン (line_trace())

	LW	RW
(1) ○●●○	250	250
(2) ●●○○	100	250
(3) ○○○○	0	40
(4) ●●●●	250	250
(5) ●●●○	0	150
(6) ○●●●	150	0
(7) ●○○○	35	250
(8) ○●○○	200	250
(9) ○○●○	150	50
(10) ○○○●	150	30

5.3.2 車庫入れの H8 プログラム

コード 2. 車庫入れプログラム(main())

```
// (YANG) :車庫入れプログラム

Feed_flag = ENCO;
traj_tracking( 40.0*1.57f, 90.0f,
5.0f);
Feed_flag = STOP;
#if 1
save_data(1); // データのセーブ
#endif

Feed_flag = ENCO;
traj_tracking( 50.0f, 0.0f, 3.0f);

Feed_flag = STOP;
#if 1
save_data(2); // データのセーブ
#endif

Feed_flag = ENCO;
//( x_f[cm], tht_f[deg], t_f[s] );
traj_tracking( -40.0*1.57f, -105.0f,
5.0f);
Feed_flag = STOP;
#if 1
save_data(3); // データのセーブ
#endif
```

車庫入れプログラムでは, コード 2 に示すようにテーマ 7 で作成した軌道追従制御部分を書き換えたり追加したりした. 最初のステップ 1 で 2 周回りが終わったロボットは「走行距離を 15 とする」「回転角度を 90 度とする (左 90 度回転)」のようにプログラミングした. 次のステップ 2 からステップ 4 までの部分は, テーマ 7 の通りだが, 距離のパラメータを修正した. ただし, 実際に実行テストの時に, 電池の電量によって電圧が小さくなると, 回転角度や走行距離が足りない場合もあった. そのために, 回転角

度と走行距離を若干増やしたり，減らしたりした．また，車庫入れの精度を高めるために，コード 3 のように「initialize_para()」関数の中にあるハードウェアパラメータを修正した．

コード 3. ハードウェアパラメータのセット

```
// --- ハードウェアパラメータをセッ
ト ---
//(YANG):車輪の半径は 0.0185, 車輪
間の距離は設計図によりおよそ
0.130.
//11/09: 距離:0.1965 半径:0.02115
// ホイールが一回転するとき
(Ki.r)*2*PI [cm] <-> (Ct.Tr)*(128*4)
[pulse]
// 0.0013725 [cm/pulse]

Ki.W= 0.1965f; // ホイール間距
離 [m]

Ki.r= 0.02115f; // ホイール半径
[m]

Ct.WPulse2Rad= 2.0f*PI/512.0f; //
エンコーダのパルスを回転角[rad]に
変換, 512=128[pulse]*4[通倍]

Ct.Tr= 29.0f; // モータ減速比
```

5.3.3 センサ配置とゴール探索

ゴール探索については，中間報告までの計画は以下のようにになっている．図 23 のようにロボット本体の前方に両腕が設置されている．まず，超音波センサをロボットの正面に設置させ，向きはこのオレンジ色矢印のように運動方向と同じである．次に測距モジュールが両腕に設置され，センサの向きをこの青色矢印のように内側に向ける．

そして，ゴール探索では，ステップ 1 として車庫入れが終わったロボットを，その場で左に回転させ，それに伴い，センサにとって壁面までの距離は遠くなるので，超音

波センサの値が少しずつ増えていく．どんどん回転して超音波センサの値が逆に小さくなると，ロボットの向きがボタンの方向と一致すると判断し，回転を停止させる．ボタンの方向を決めた後，次にステップ 2 として直線運転してボタンに近く．そして同時に超音波センサの値がまた小さくてなる．その値が基準値より小さいと，ゴールまで到着したと判断し，運転を停止させる．最後ステップ 3 として RC サーボモータと歯車の動きによって，両腕の間の距離を小さくさせ，測距モジュールの値が減っていく．測距モジュールの値が基準値より小さいと，もうボタンを挟んだと判断し，その回転を停止させ．ボタン押すプログラムを実行する．

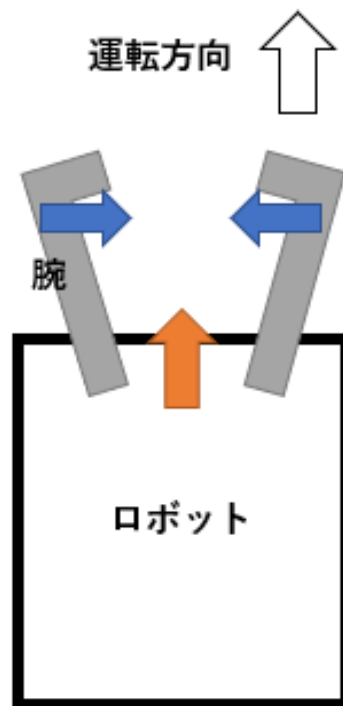


図 23. センサの配置



図 24. ゴール探索の方法

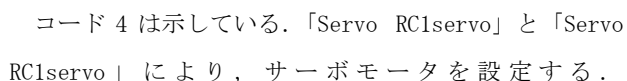
しかし，実際にプログラムの作成中と実行テスト中に，

超音波センサによる直線運動をする。閾値より超音波センサの値が小さいと、運動が停止され、ゴール探索が完了する。ゴール探索が終わった後、ボタン押すプログラムを実行する。サーボモータ（RC1）による両腕でボタンを掴むようになる、そして、このサーボモータ（RC1）の角度が閾値を超えると、もう一つサーボモータ（RC2）の回転による磁石が軌道を離れる。ゆえに、鉄球が軌道に滑り、ボタン押すことになる。



1. define と初期値設定
2. setup 関数
3. loop 関数内の H8 に関する入出力
4. 自作関数
5. loop 関数内のセンサ
6. loop 関数内のサーボモータ

5.3.4.1 define と初期値設定



「PIN_OUTPUT_RC1 9」と「PIN_OUTPUT_RC2 10」により、
Arduino のデジタル 9pin とデジタル 10pin を二つのサー
ボモータ（RC1 と RC2）の信号輸送として用いる。

このようにすると、ゴール探索は、以下の三つのステップで構成されている。まずにステップ1では、車庫入れ後のロボットは、測距モジュールによる直線運動をする。左側面にある測距モジュールがボタンと同じ水平線になると、直線運動が停止される。次に、ステップ2では、単純に左へ 90° を回転する。それで、ロボットの正面がボタンに面している。そして、ステップ3では、ロボットは、

コード 4. define と初期値設定

```

/*****
*** ゴール探索でのセンサの通信と
サーボモータコントロールのプログラ
ラム
*** YANG(2023/1/27) バージョン
1.8
*****/
#include <Servo.h>
Servo RC1servo;
Servo RC2servo;
#define PIN_OUTPUT_RC1 9
//腕を挟む用 RC1 への信号線
#define PIN_OUTPUT_RC2 10
//磁石に作用する RC2 への信号線
#define PIN_INPUT_PSD 5
//PSD センサからのアナログ信号一
入力
#define PIN_OUTPUT_B4 13
//処理したの信号を H8 へ送る CN2-
33. PSD
#define PIN_OUTPUT_B5 12
//処理したの信号を H8 へ送る CN2-
34. SONIC
#define PIN_INPUT 7
//H8 から起動信号を受ける
#define PIN_TRG_SONIC 2
//SONIC 超音波センサの TRG 入力
#define PIN_ECHO_SONIC 3
//SONIC 超音波センサの ECHO 出力
//初期値設定
int RC1_pos = 0;
int RC2_pos = 0;
int i_B4 = 0;
int i_B5 = 0;

```

```

void setup() {
    //SONIC の PIN 初期化
    pinMode(PIN_TRG_SONIC,
OUTPUT);
    pinMode(PIN_ECHO_SONIC,
INPUT);
    //OUTPUT and INPUT 初期化
    pinMode(PIN_OUTPUT_B4,
OUTPUT);
    pinMode(PIN_OUTPUT_B5,
OUTPUT);
    pinMode(PIN_INPUT, INPUT);
    pinMode(PIN_INPUT_PSD,
INPUT);
    //RC1 and RC2 初期化
    pinMode(PIN_OUTPUT_RC1,
OUTPUT);
    pinMode(PIN_OUTPUT_RC2,
OUTPUT);
    RC1servo.attach(PIN_OUTPUT_RC
1);

    RC2servo.attach(PIN_OUTPUT_RC
2);

    Serial.begin(115200); //シリアル通
信のポートを開く

    digitalWrite(PIN_TRG_SONIC,LO
W);

    digitalWrite(PIN_OUTPUT_B4,
LOW); //H8 へ 0 を送る
    digitalWrite(PIN_OUTPUT_B5,
LOW); //H8 へ 0 を送る
    //RC の角度の初期化
    RC1servo.write(0);
    RC2servo.write(0);
}

```

5.3.4.2 setup 関数

setup 関数についてコード 5 は、示している。超音波センサ、測距モジュール、サーボモータ、H8 への入出力を設定する。「digitalWrite」により、超音波センサの TRG と H8 への出力を「LOW」で初期化する。また、サーボモータ (RC1 と RC2) の角度を初期化する。

コード 6. loop 関数内の H8 への出力

```
void loop(){
...
...
...
if( i_B4 == 0){

digitalWrite(PIN_OUTPUT_B4,
LOW);//H8 へ 0 を送る
    Serial.print("B4=0¥n");
}else{

digitalWrite(PIN_OUTPUT_B4,
HIGH);//H8 へ 1 を送る
    Serial.print("B4=1¥n");
}
if( i_B5 == 0){

digitalWrite(PIN_OUTPUT_B5,
LOW);//H8 へ 0 を送る
    Serial.print("B5=0¥n");
}else{

digitalWrite(PIN_OUTPUT_B5,
HIGH);//H8 へ 1 を送る
    Serial.print("B5=1¥n");
}
...
...
...
}
```

5.3.4.3 loop 関数内の H8 に関する入出力

loop 関数内の H8 への出力については、コード 6 に示している。初期化された「i_B4」と「i_B5」は、0 であれば、毎回のループに H8 へ「LOW」の信号を送る。0 ではない場合(センサの値と閾値による)に、毎回のループに H8 へ「HIGH」の信号を送る。

そして、loop 関数内の H8 からの入力については、コード 7 に示している。自作関数「Inputread」(コード 8)により、10ms ごとに、二つの H8 からの信号を「int input_1」と「int input_2」に代入する。「int input_1」と「int input_2」が共に「1」である場合のみ、以下のコード 8 のセンサのプログラムが実行できる。Arduino と H8 の通信の不安定を解決するためである。(以下の車庫入れ後の H8 プログラムが示すように最初に H8 から Arduino へ「0」を送る。車庫入れ後に H8 から Arduino へ「1」を送る)それで、車庫入れの前にセンサが実行していない。ゆえに、ライントレースと車庫入れ中に、センサが先に反応しないことを確保できる。

コード 7. loop 関数内の H8 からの入力

```
void loop(){
...
...
...
int input_1 = 0;
int input_2 = 0;
input_1 = Inputread(PIN_INPUT);
delay(10);//時間を持つ
input_2 = Inputread(PIN_INPUT);
delay(10);//時間を持つ

if( (input_1 == HIGH) && (input_2
== HIGH) ){
...
...
...
}

}
```

コード 8. 自作関数 (PSDread, SONICread, Inputread)

```
int PSDread(int PinNo){ //PSD の読み取り
    long ans=0;
    for(int i=0; i<100; i++){
        ans=ans +analogRead(PinNo);
    }
    return ans/100;
}

int SONICread(int distance){
    double sum = 0;
    for(int i=0; i<100; i++){
        sum = sum + distance;
    }
    return sum/100.0;
}

int Inputread(int PinNo){
    int sum = 0;
    for(int i=0; i<100; i++){
        sum=sum +digitalRead(PinNo);
    }
    if(sum < 80) return 0;
    if(sum >= 80) return 1;
}
```

5.3.4.4 自作関数

次に自作関数について、コード 8(PSDread, SONICread, Inputread), コード 9(rc), コード 10(AnaToCm)は、示している。

「PSDread」と「SONICread」の仕組みは、ほぼ同じである。「PSDread」では、PSD の信号 pin 番号を引数として、100 回分のアナログ信号の合計「ans」を求め、「ans」の 100 を割ると、100 回の PSD のアナログ信号の平均になる。これを「return」にする。「SONICread」では、loop 関数内の超音波センサに関するプログラムによる距離「distance」を引数として、これを 100 回分の平均値を取り、「return」にする。「Inputread」では、H8 からの入力信号 pin 番号

を引数として、100 回分の中に 80 以上の「1」がある場合、「1」を「return」にする。逆に 80 未満だと、「0」を「return」にする。以上の三つの関数の目的は、外れ値やミスを削除するために、安定化された値を求める。

コード 9. 自作関数 (rc)

```
int rc(double distance,int sonic_r){
    if( distance >= sonic_r ){
        Serial.print("直線 2 運転指令¥n");
        return 0;
    }else if(distance == 0){
        return 0;
    }else{
        Serial.print("直線 2 運運転停止指令¥n");
        return 1;
    }
}
```

コード 10. 自作関数 (AnaToCm)

```
int AnaToCm(int analogValue){ //PSD の距離示す
    if(analogValue >= 470) return(10);
    if(analogValue >= 330) return(15);
    if(analogValue >= 259) return(20);
    if(analogValue >= 214) return(25);
    if(analogValue >= 172) return(30);
    if(analogValue >= 153) return(35);
    if(analogValue >= 134) return(40);
    if(analogValue >= 105) return(50);
    if(analogValue >= 85)
    return(60);
    return(-1);
}
```

「rc」関数では、超音波センサにより、「SONICread」関数で求められた距離「distance」と超音波センサの閾値

「sonic_r」(sonic_r=13)を引数として、距離が閾値(13cm)より小さいと、「1」を「return」にする。(この「1」は、以下の超音波センサのプログラムにある変数「i_s」に「return」される)。その閾値 13cm については、実際に超音波センサの位置とボタン押せる位置の距離を図ることで得た。

「AnaToCm」関数では、測距モジュールにより、「PSDread」関数で求められたアナログ信号を引数として、距離(cm)を「return」にする。それらの値は、以下の表に示すような前期機械システム工学実験のテーマ 4 の実験 7 測定結果である。

表 5. 測距モジュールによるセンサ値と距離の関係

距離d(cm)	センサ値
10	470
15	330
20	259
25	214
30	172
35	153
40	134

5.3.4.5 loop 関数内のセンサ

Loop 関数内のセンサについて、コード 11(変数定義)、コード 12(PSD)、コード 13(SONIC)は示している。

コード 11 では、測距モジュールの閾値「PSD_r_high」と「PSD_r_low」、超音波センサの閾値「sonic_r」が定義されている(cm)。

コード 12 の測距モジュールでは、「PSDread」関数で「PSD_val」に測距モジュールのセンサ値の平均値を代入する。そして、「AnaToCm」で「PSD_v」をセンサ値からの距離 (cm) を代入する。最後、If 文で以下の式(1)を満たす場合に、「PIN_OUTPUT_B4」に「HIGH」を入れ、「i_B4=1」になる。(i_B4 についてコード 6 は示している)

$$15 < PSD_d \leq 30 \text{ (cm)} \quad (1)$$

コード 13 の超音波センサでは、「PIN_TRG_SONIC」による超音波を発射し、「PIN_ECHO_SONIC」による超音波センサを受ける。そして、以下の式 2 により、発射から受ける

までの時間を距離 (cm) に変換する。

$$\text{距離(cm)} = \text{時間(us)} \times 340(\text{m/sec}) / 2 \quad (2)$$

最後に、「rc」関数で超音波センサの距離を閾値との比較し、「i_s」に代入する。

コード 11. 変数定義

```
int interval = 0;           //pulse
double distance = 0;       //距離
int i_s = 0;               //距離判断のため
int PSD_r_high = 30;      //PSD の閾値の高い
int PSD_r_low = 15;       //PSD の閾値の低い
int sonic_r = 13;         //超音波センサの閾値の値、この値によりRC1 の運転を停止する。
int PSD_val = 0;          //PSD 値
```

コード 12. PSD

```
PSD_val = PSDread( PIN_INPUT_PSD );
int PSD_d = AnaToCm(PSD_val);

Serial.print(AnaToCm(PSD_val));
Serial.println("cm");

if( (PSD_d <= PSD_r_high ) && ( PSD_d > PSD_r_low ) ){

digitalWrite(PIN_OUTPUT_B4,HIGH);

i_B4 = 1;
Serial.print("B4HIGH");
}
```

コード 13. SONIC

```

if( i_B4 == 1 ){
    Serial.println("SONIC に入っ
た");

    digitalWrite(PIN_TRG_SONIC,
HIGH);
    delayMicroseconds(9);

    digitalWrite(PIN_TRG_SONIC,
LOW);

    interval =
pulseIn(PIN_ECHO_SONIC, HIGH,
30000); // 時間計測
    distance = interval * 0.017;

    // 時間 => 距離 に 変 換
[cm]340[m/sec]/2 = 0.017[cm/μs]

    distance=SONICread(distance);//dis
tance の 100 回平均

    // send result
    Serial.print("pulse   width
=");

    Serial.print(interval);
    Serial.print("[us]   distance
=");

    Serial.print(distance);
    Serial.print("[cm]¥n");

    i_s = rc(distance, sonic_r);

```

5.3.4.6 loop 関数内のサーボモータ

loop 関数内のサーボモータについて、コード 14 は示している。「i_s==1」により、超音波センサが閾値より小さいことで、サーボモータの動きを始める。「i_B5=1」と「digitalWrite(PIN_OUTPUT_B5, HIGH);」については、コード 6 の loop 関数内の H8 に関する入出力に説明される

ように、H8 の「P2.DR.BIT.B5」へ「HIGH」を送るようになる。そして、RC1 の角度が 100 度未満の場合、毎回の loop 関数に 10 度ずつ回転すると両腕でボタンを掴む構造 (5.1) が行う。最後に、RC1 の角度が 100 度より大きい場合、RC2 は一気に 180 度回転すると、磁石を用いた鉄球で押す構造 (5.1) が行うようになる。

コード 14. サーボモータ

```

if( i_s == 1 ){//H8 へ信号を送る
    i_B5=1;

    digitalWrite(PIN_OUTPUT_B5,HIGH);

    delay(1000);

    if( RC1_pos < 100 ){

        RC1servo.write(RC1_pos);
        RC1_pos+=10;
        Serial.print("B5HIGH");
        Serial.print("RC1 回転中
¥n");
    }

    delay(50);

    if( RC1_pos >=100 ){
        Serial.print("RC2 回転
中¥n");

        for(int i=0;
RC2_pos<180;RC2_pos+=1 ){

            RC2servo.write(RC2_pos);

            delay(15);
        }
    }
}

```

以上で、Arduino プログラムを説明した。まとめて以下のフローチャート図 27、は示している。

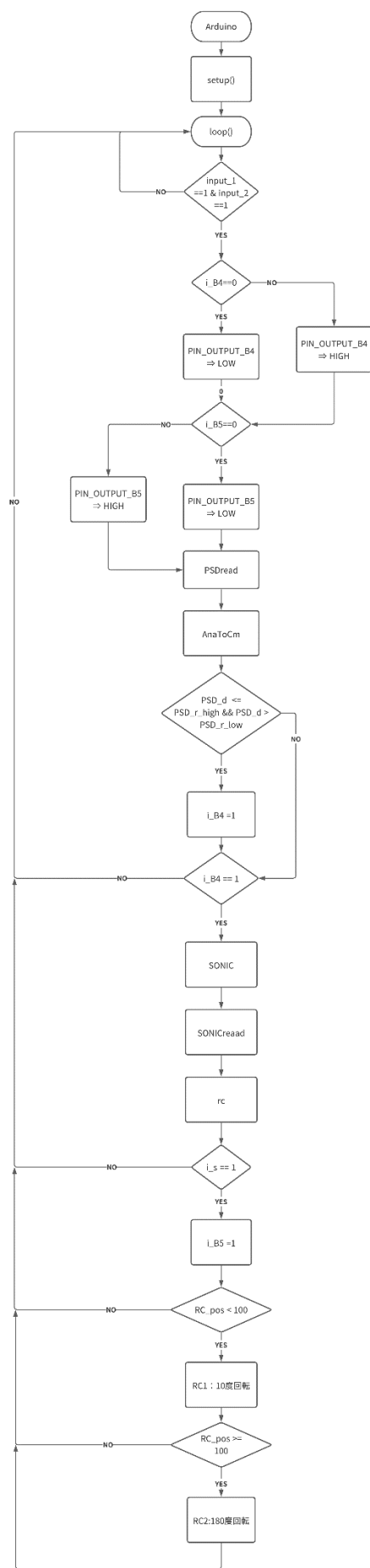


図 27. Arduino プログラムのフローチャート

5.3.5 車庫入れ後の H8 プログラム

車庫入れ後の H8 プログラムを述べる前に、コード 16 に示すように main 文の最初に、ボード初期化し、「0」の場合は「H8 の入力」、「1」の場合は「H8 の出力」である。出力の「P4.DR.BIT.B4」(H8 の CN2-9pin), 入力の「P2.DR.BIT.B4」(H8 の CN2-33pin), 入力「P2.DR.BIT.B5」(H8 の CN2-34pin) を用いる。

コード 16. ボード初期化 (main())

```

P2.DDR=0x0f; //0000 1111
P4.DDR=0x1f; //0001 1111

P4.DR.BIT.B4 = 0;

```

次にコード 15 のゴール探索では、まず、測距モジュールによる一回目直線運動をするため、無限ループ「while(1)」の中に、for 文で、100 回の「P2.DR.BIT.B4」の合計を求める。その合計「sum_P2_B4」が 60 より大きいと、無限ループから「break」し、ボタンと同じ水平線になるはずである。しかし、実際には、測距モジュールの実際の位置がロボット本体の左側面の中心点になっていないので、100 と 60 に 0.8 を掛け、誤差を小さくするためである。この誤差を小さくしないと、後の 90 度回転したら、ロボットの向きはボタンの向きと一致しないことで、ボタンを押せないことがあった。(両腕によるボタンを中央に移動させ、押せることもあった)そして、左 90 度回転する。最後に、一回目直線運動と似ているように超音波センサによる二回目直線運動を行う。(ゴール探索方法については図 16 のところに説明される)

以上で、H8 プログラムを説明した。まとめて以下のフローチャート図 28 は示している。また、H8 と Arduino の信号通信について、以下の表 5 は示している。

表 5. H8 と Arduino の信号通信

H8 の信号	Arduino の信号	機能
P4.DR.BIT.B4	PIN_INPUT	車庫入れ終了判断
P2.DR.BIT.B4	PIN_OUTPUT_B4	PSD による走行
P2.DR.BIT.B5	PIN_OUTPUT_B5	SONIC による走行

コード 15. ゴール探索(main())

```

P4.DR.BIT.B4 = 1;
//一回目直線運動
    int sum_P2_B4 = 0;
    int i_P2_B4;
while(1){
    for
(i_P2_B4=0;i_P2_B4<100*0.8;i_P2_B4++){
        sum_P2_B4 = sum_P2_B4
+ P2.DR.BIT.B4;
        motor(LW, 150*LD);
        motor(RW, 150*RD);
    }
    if(sum_P2_B4 >= 60*0.8){
        break;
    }
}
//左 90 度, 回転
Feed_flag = ENCO;
traj_tracking( 0.0f, 90.0f, 5.0f );
Feed_flag = STOP;
//二回目直線運動
int sum_P2_B5 = 0;
int i_P2_B5;
while(1){
    for
(i_P2_B5=0;i_P2_B5<100;i_P2_B5++){
        sum_P2_B5 = sum_P2_B5
+ P2.DR.BIT.B5;
        motor(LW, 150*LD);
        motor(RW, 150*RD);
    }
    if(sum_P2_B5 >= 60){
        break;
    }
}

```

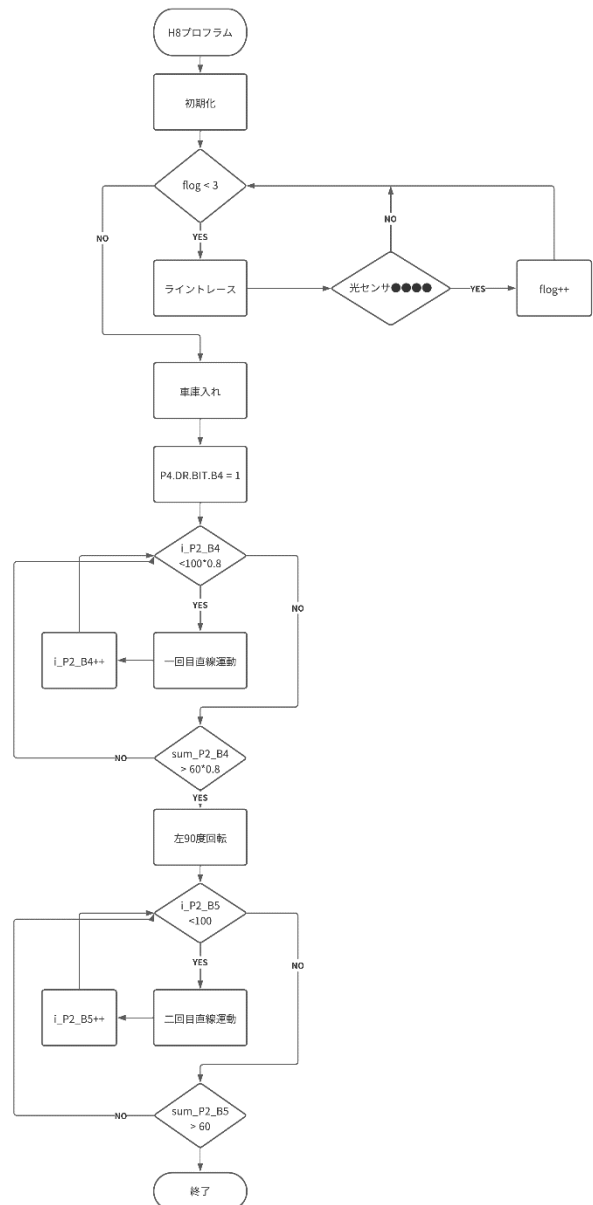


図 28. H8 プログラムのフローチャート

6. 移動ロボットの製作と動作確認の結果

移動ロボットの製作では、最初に大基板の回路設計に問題が生じた。各電気部品を表と裏が逆のままに大基板の回路設計が設計された。これによって回路の半田付けと作成が遅くなり、中間報告の時にロボット本体がまだ走行できない状態であった。また、回路デバッグ作業中に、主な問題点としては、大基板の回路に 5V がなかったことである。これを解決するため、全体的に導通チェックと電気部品のチェックを行うことで、回路を修正し、コネクタによる三端子レギュレータの製作を忘れていたことを発見した。よって、ボタン押す構造とロボット本体の構造、プログラムの作成が順調に進んでいたが、回路の問題により、ライン

トレースと車庫入れの速度調整, ボタン押すテスト, 実行テストが全体的に遅れていた. 実行テスト中に, Arduino の接触不良により, プログラムを書き込みできない問題があった. この理由は, Arduino を固定する鉄アームが Arduino の裏面のいくつかの pin と接触していることである, これを解決するため, 固定用鉄アームと Arduino の間にナットを付けた. さらに, H8 と Arduino の信号通信問題, センサ値の不安定問題があったので, コンテンツ中に, 車庫入れまで上手くいったが, ゴール探索ができない状態になった.

コンテンツ中にうまくいかなかった理由として, コンテンツでは, 一回目に電池の電量が足りなく走行できずに終わった, 電池を交換した後に, 車庫入れまででき, H8 と Arduino の通信問題とセンサ値の不安定問題でゴール探索までうまくいかなかった.

コンテンツ後に, チャンピオン動画としてボタン押すまで動作確認できた. チャンピオン動画でうまくいった理由として, 5.3 の説明のようにプログラムで 100 回分の平均値や連続判断を用いると, 信号とセンサ値の不安定問題を改善できる. さらに Arduino と H8 の GND を繋がることで, H8 と Arduino の信号通信問題, センサ値の不安定問題をできた. また, Arduino がパソコンと繋がっている理由は, 用いられる電池のボックスだけで測距モジュール, 超音波センサと二つのサーボモータに電力の提供が足りないことである.

7. 考察とまとめ

7.1 考察

7.1.1 走行性能

ロボット本体の重量が少ないが, タイヤが安定である. コンテンツの時は, ライントレース速度がかなり早かった (ライントレース約 17 秒). しかし, 電池の電量により, 成功率は若干低い. そして, チャンピオン動画を取る時に, ライントレースの直角になったところの速度を小さくした.

7.1.2 改善と工夫の余地

1. Arduino の電力提供問題については, 電池ボックスで測距モジュール, 超音波センサと二つのサーボモータに電力の提供が足りないことがあった. この問題を解決するために, より電力が安定である電源を用いる. 一方で, 電池

ボックスを変えず, 測距モジュールを廃置すると電力が足りるかもしれない. ただし, ゴール探索とプログラムの修正が必要である.

2. 両腕でボタンを掴む構造については, 二つの歯車と両腕構造の機構が問題ないが, サーボモータ (RC1) と歯車のかみ合いのずれが発生したことがあった. サーボモータの回転部分について, 歯車との規格が同じような回転部を設計し, 組み立てれば, 両腕構造の動きがより良くなる.

3. 測距モジュールの位置については, 測距モジュールをロボットの左側面の中心点 (5.3.3 に説明された) に固定されば, 一回目直線運動より, ロボットの本体の中心点がボタンと同じ水平線になることで, ゴール探索の成功率が上がる.

4. ロボットの外観について, 工夫の余地がある. 特にセンサと Arduino の回路では, ブレッドボードとジャンパー線の繋がりが乱れている. 回路のチェックにも手間がかかる. もっと時間があれば, 小さな基板を用いる回路を改良すると考えている.

7.2 まとめ

C 班では, 両腕でボタンを掴む構造, 磁石を用いた鉄球で押す構造を中心として, 測距モジュールと超音波センサによるゴール探索方法で移動ロボットを製作した. コンテンツ中に残念ながらうまくいかなかったが, チームワークにより, 最後に問題を解決でき, チャンピオン動画のようにボタン押すまでの走行完了をした.

8. ロボットの完成に関して, 貢献した点

私は, C 班のサブリーダーを務めている. プログラム担当として, ライントレースと車庫入れのプログラム, Arduino プログラム, 車庫入れ後の H8 プログラムを個人作業で完成した. 二つのセンサ配置とゴール探索の方法, 磁石によるボタン押すアイデアを提案した. また, センサとマイコン (Arduino) の回路を設計し, 作成した.

次に, サポーターとして, 大基盤, 小基盤 (フォトリフレクタ), 電源部の回路作成をサポートし, ボタン押す機構の問題を解決するために案を考えていた.

また, 実行テスト中の試行錯誤で主な問題点を解決するため, 授業外に多く残業していた. 残業の時にリーダーがよくいなく, ほかのメンバーをまとめて率いていた.

以上, ロボットの完成に関する貢献である.

参考文献

1. 『機械システム工学実験テキスト』 昼間コース C 班, 2022
2. Teams クラス資料 : 「H8 と Arduino 関連操作. ppt」
3. Teams クラス資料 : 「PSD センサ. ppt」
4. Teams クラス資料 : 「arduino 使い方 : 赤外線測距モジュール (GP2Y0A21). pdf」
5. Teams クラス資料 : 「ReadMe. txt」
6. Teams クラス資料 : 「機械システム工学実習 II 【サーボモータ】. pdf」
7. Teams クラス資料 : 「超音波センサ. ppt」
8. Teams クラス資料 : 「超音波センサ関連資料. pdf」
9. Teams クラス資料 : 「sonic_test_z. ino」
10. Teams クラス資料 : 「psd_test_z. ino」
11. 「第 16 回 Arduino (アルディーノ) でパーツやセンサを使ってみよう～超音波モジュール編」
URL: <https://deviceplus.jp/arduino/entry016/>
12. 「Arduino UNO のピン配置と役割【使い方を理解しよう】」
URL: https://miraiworks.org/?p=5655#Arduino_UNO
13. 機械システム工学科実験テーマ 4 の実験 7 の測定結果 (自分のレポート)