# 創造設計第二 最終報告書
# 10班

2009.2.10

班長：片岡木太郎
会計：上村健人
P.M ：高木拓人
記録：渡部純

# 目 次



# 1 課題１

## 1.1 マシンコンセプト

鉛蓄電池の無い軽量のマシンで、脱線しない安定した速度を保つ。減点 0 かつ全チームの中で最大得点を目標とする。

## 1.2 課題１のマシンの概要

図 1 に示されているようにマシンはチームで作ったオリジナルのマイコン基板とニッケル水素電池 3 本を積んだ金太郎の電気機関車が全体の主軸となっており、A 類で配布された MCU ボードは死重とした。先頭車両には PSD 距離センサをアルミアングルによって設置し、角度調整ができる形とした。マイコン台の台車は EH500 電気機関車の台車を

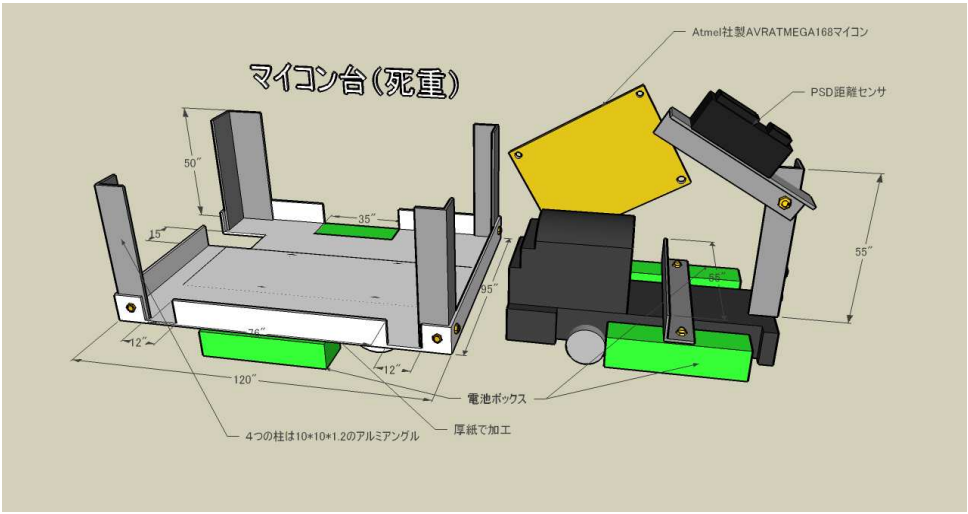



図 1: 課題 1 のマシンの全体像

使用し、MCU ボードを固定する台は厚紙で作られた枠であり、簡単に MCU ボードの取り外しができる。位置の判断は PSD センサによるゲートの検出で行う。

## 1.3 課題 1 のマシンの説明

### 1.3.1 課題 1 のマシンの回路

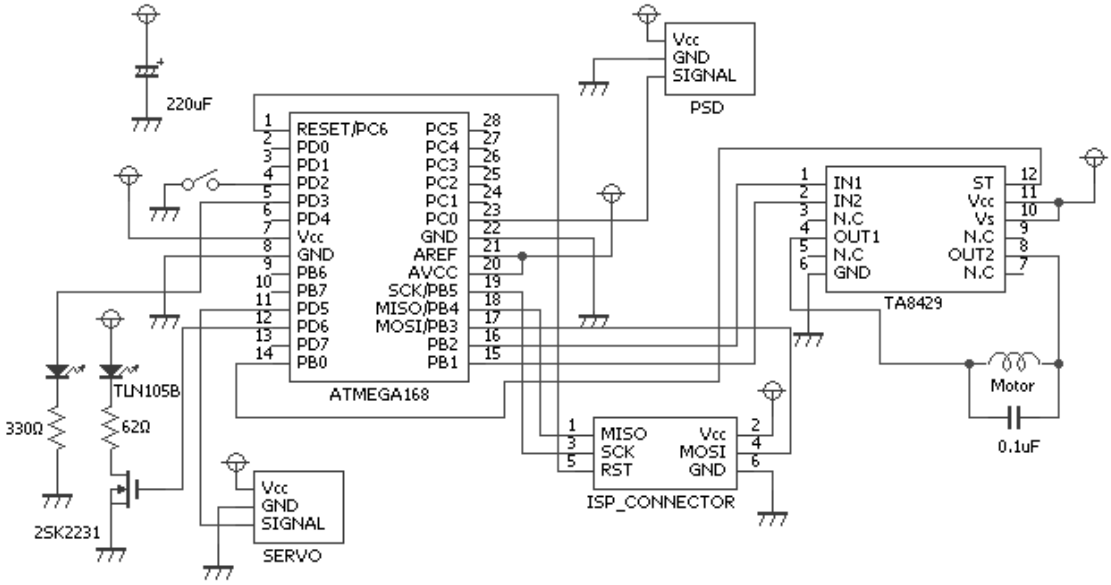



図 2: 課題 1 のマシンの回路図

- 第 2 課題のマシン 1 の先頭車両とほぼ同一である。
- 駆動輪のモータ、サーボ、PSD、スイッチなどの機器を扱える。
- マシン 2 へ信号を送るために赤外線 LED も搭載。(第 2 課題のみ,700Hz 程度で発光)
- MCU ボードを使用せず Atmel 社製 AVR ATMEGA168 マイコンを使用。

– プログラム書き込み用に ISP コネクタが接続されている。

・モータードライバは既製品の TA8429 を使用。

– IN1,IN2 ピンで回転方向を指定し ST ピンで PWM 制御を行なっている。

・電源は Ni-MH 電池のみであるので、モーターノイズがマイコン側に流れ込みリセットされないようにコンデンサがモーター端子間、Vcc-GND 間に挟んである。

### 1.3.2 課題 1 のマシンの動き


図 3: 課題 1 マシンの状態遷移図

課題 1 マシンの入力は PSD センサのみであり、出力は足回り用のモーターのみである。
PSD センサが一定値を下回るとゲートが存在すると判断する。

1. 【RUN】走り出す。
   - 走り出した直後は PSD センサを読み込んでおらずゲート 2 は読み飛ばす。
   - ゲートをくぐった回数を記録している。
   - ゲートをくぐった回数、ゲートを通過してから経過した時間によってモータの速度を調整し脱線しにくくしている。
2. 【RUN_SLOW】ゲートを 2 回くぐると A 区間に帰ってきていることになるので低速走行をはじめる。
3. 【STOP】3 つ目のゲート (ゲート 2) を検出すると停止する。
4. 【WAIT】1 からの経過時間が 60 秒になると 1 へと戻る。
5. 【END】WAIT が 3 回行なわれると終了する。

## 1.4 課題 1 反省

### 1.4.1 課題 1 結果

| 1 周目着 | 2 周目発 | 2 周目着 | 3 周目発 | 3 周目着 | ゲート接触 | 床面接触 | リトライ | 減点 | ポイント |
|---|---|---|---|---|---|---|---|---|---|
| 10.07 | 60.11 | 70.34 | 115.16 | 125.38 | 0 | 1 | 1 | -116 | 184 |

タイムによる加点 300 点、リトライ 1 回 (-60 点)、床面接触 1 回 (-40 点)、早発 4 秒 (-16 点) で合計 184 点であった。

**リトライ前**

一周目は 10 秒弱で走行。
二周目はほぼ 60 秒でスタート。
途中 D 区間 S 字の最中で先頭車両から脱線、床面接触、走行不能でリトライとなった。

**リトライ後**

全体的にプログラムで速度を下げての走行。そのため 10 秒は切れていない。
1 周目、2 周目、3 周目ともに走行自体には問題なく完走。
ただし、3 周目のスタート時刻が 2 分より 5 秒ほど早まっている。

### 1.4.2 脱線

**脱線の原因について**

脱線した理由についてであるが、「速度を出しすぎた」、「先頭車両が軽すぎる」という 2 つの原因によると考えられる。
我々の班は鉛蓄電池を積んでおらず他班よりも軽く速度がでる為、カーブや S 字などで振動しやすい。
また先頭車両が軽いため、重い 2 両目が S 字で振動的になると、先頭車両がそれに引っ張られて脱線しやすい。
第 1 戦でもそのようにして S 字で脱線したのではないかと思われる。

**我々が脱線に対して行なっていた対策**

連結部がプラスチックの連結部のままだと 2 両目からの力が先頭車両に直接伝わってしまい脱線するのでそこを紐の連結に変更した。これによって格段に脱線しにくくなった。
先頭車両に電池を 4 本積んだ場合、2 両目が遠心力で転倒するようになったので 2 両目下部に電池 2 本を搭載し、先頭車両に積む電池を 2 本に減らした。これで 2 両目は転倒しにくくなった。
Ni-MH 電池の充電具合によって最高速度が変化してしまう恐れがあったので Ni-MH 電池よりも電圧の高いアルカリ電池を使用したテストランも行い脱線しないことを確認した。

**他に我々がすべきであった対策**

先頭車両を重くすべきであった。
テストランを様々な条件で何度も行い速度を抑えて安定走行するようにしていたが、先頭車両の軽さは外乱の影響を受けやすくしてしまっていた。先頭車両に死重を乗せるべきであった。

### 1.4.3 55 秒にスタートするバグについて

再現性が無いバグである。atmega168 マイコンにて発生。2 周目もしくは、3 周目のスタート時刻が 5 秒早くなる。2 周目に 5 秒早くなった場合には 3 周目はその後 1 分丁度でスタートする。3,4 回に 1 回ぐらいの確率で発生し、発生原因がわかっていない。早くなるのはいつも 5 秒である。
同じプログラムをマイコンに焼き直すとこのバグが発生しなくなるようである。
プログラムのバグである可能性もあるがライティングソフトや回路の問題である可能性もある。
第 1 課題後このバグが一度も発生していないため原因不明である。

# 2　課題 2

## 2.1　マシンコンセプト

シンプルで確実、というのがチームコンセプトである。

**2 台の動き**　2 つのマシンをそれぞれマシン 1、マシン 2 と呼ぶ。
マシン 1 は登坂をし、ピンポン球を回収、立体交差上からマシン 2 へとピンポン球を受け渡す。
マシン 2 は立体交差下で待機、マシン 1 からピンポン球を受け取りピンポン球を運搬する。

**マシン 1 概要**　マシン 1 は 2 両編成であり先頭車両は課題 1 と同じもの、2 両目はピンポン球を回収運搬するためのアーム A、ピンポン球供給装置を抑えるためのアーム B、そしてカゴを備える。

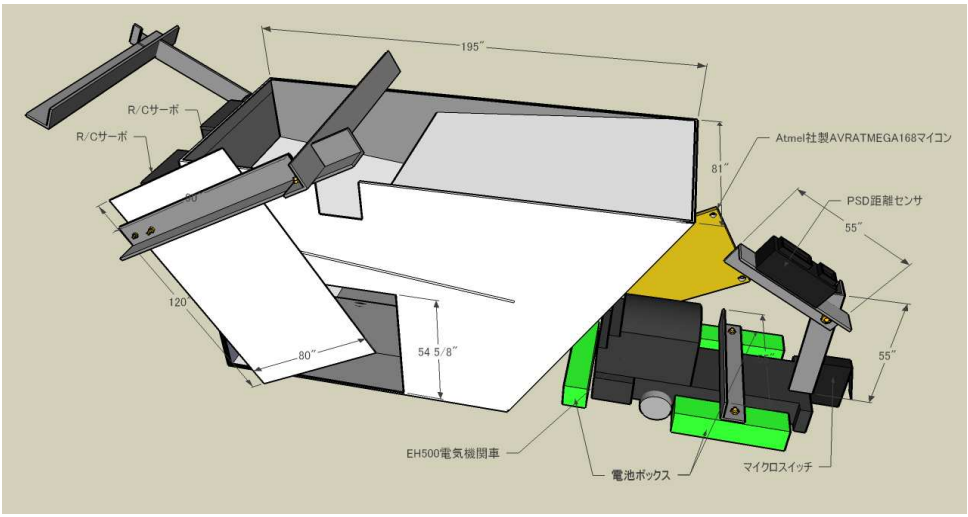



図 4: 課題 2 のマシン 1 の全体像

**マシン 2 概要**　マシン 2 は 3 両編成であり先頭車両にモーターと鉛蓄電池、2 両目に MCU マイコンボード、3 両目にカゴとピンポン球をリリースするためのアームを備える。立体交差下で待機し 1 分 30 秒たつと自動的に立体交差下から出発する。

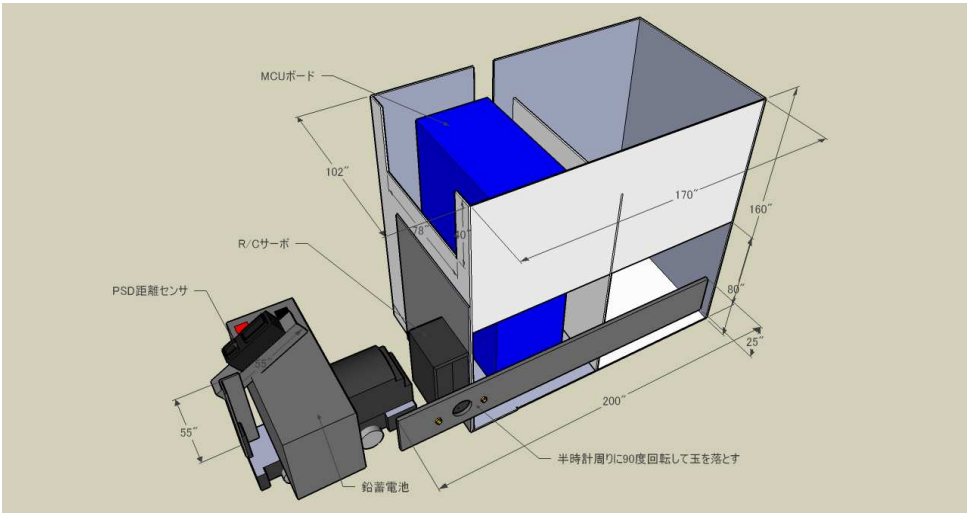



図 5: 課題 2 のマシン 2 の全体像

## 2.2 マシンの説明

### 2.2.1 機械部分

**マシン 1 の機械部分**　先頭車両は課題 1 で用いたものを使用する。2 両目は EH500 電気機関車の台車の上に厚紙で加工したボックス型のカゴとピンポン球を取るためのアーム A、姿勢が崩れないようにピンポン球供給装置を抑えるアーム B を乗せた。ゲートや供給装置下をくぐる際にはアーム A、B ともに低い姿勢にしておく。ピンポン球回収時はアーム B でピンポン球供給装置を押さえアーム A で何度 もピンポン球を押してピンポン球を回収する。
ピンポン球の受け渡し時にはアーム B を傾けることでカゴのフタが開きピンポン球を排出する。
1 両目と 2 両目の間は紐で繋がれている。

**マシン 2 の機械部分**　2 両編成である。先頭車両に鉛蓄電池とモーター、2 両目にカゴとカゴからピンポン球を出すためのアーム、MCU ボードが積まれている。マシン上部からピンポン球を取り込みサーボにアームを動かすことで排出する。

### 2.2.2 回路部分

**マシン 1 の回路**　課題 1 のマシンと基本的に同様の回路であるが ATMEGA168 に車止め検出用のマイクロスイッチが追加されている。

**マシン 2 の回路**

- MCU ボードを使用
- 三端子レギュレーター 7805 を用いて鉛蓄電池の電圧を 5V まで降下させモーター電源としている。
- 動作中、ヒューズをとばす危険性を極力低くするためにモーターに直列に抵抗が入っている。

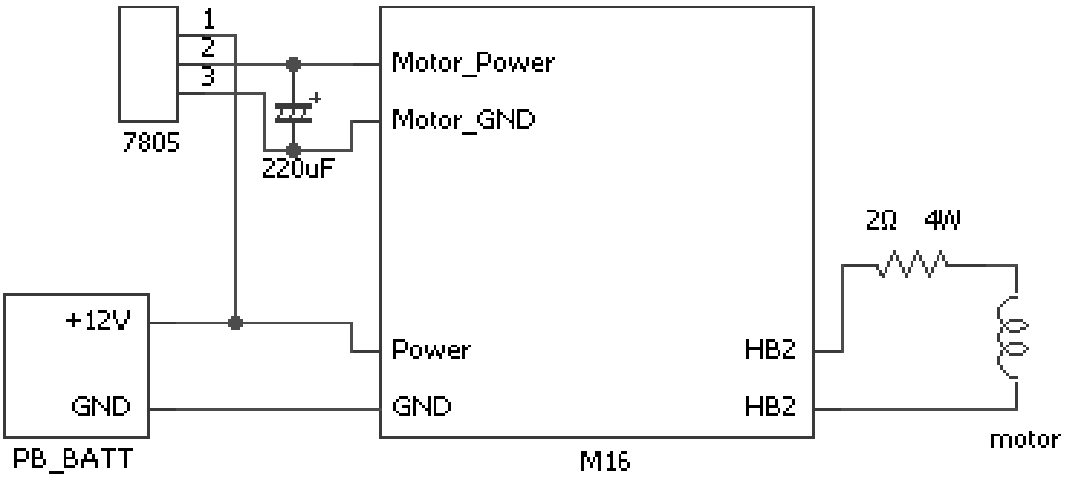



図 6: マシン 2 メイン基板回路図

### 2.2.3 マシンの動き

**マシン 1 について**　アクチュエーターは車輪のモーターとアーム A, アーム B のサーボの 2 つであり、センサは PSD センサ、マイクロスイッチの 2 つである。マシン 1 はH区間でゲート 2 向きでスタンバイする。

1. 【WAIT】スイッチが押されるまで待機する。
2. 【start】スタートして坂の手前まで進む

・一定時間低速走行 (ゲート 2 を通過、登坂へ)

・低速では登坂できないためカーブの途中で停止する。

3. 【climb】登坂動作及び供給装置に対する位置調整

- 1 登坂する
- 2 PSD センサで供給装置までの距離が 300mm まで近づくと停止
- 3 低速で前進
- 4 PSD センサで供給装置までの距離が 140mm まで近づくと停止
- 5 低速で一定時間前進 (ピンポン球供給装置を通過)
- 6 停止
- 7 アーム B を上げる。
- 8 低速で一定時間後進 (ピンポン球供給装置にアーム B が押し付けられる)
- 9 1 両目 (駆動車) と 2 両目 (カゴ車) が接触していると不安定になるので僅かに前進

4. 【capture】ピンポン球の取得

- 1 アーム A を何度か動かしピンポン球を落とし取得

5. 【release】ピンポン球の運搬と放出

- 1 マイクロスイッチで車止めを検出するまで前進
- 2 低速で一定時間後進
- 3 アーム B を動かしピンポン球を放出
- 4 ピンポン球の詰まり解消のためアームを動かしたり前後進を繰り返し機体を揺らす。

6. 【END】停止終了

以上がマシン 1 の動きである。【start】,【climb】,【capture】,【release】はプログラム内の関数と対応している。プログラムはレポートの末に付録。

**マシン 2 について**　アクチュエータは車両駆動用の DC モータとアーム開閉用サーボの 2 つであり、センサはゲートと立体交差検出用の PSD センサ 1 つのみである。
マシン 2 は A 区間でゲート 1 向きにスタンバイする。

1. 【WAIT】スイッチが ON にされるまで待機
2. 【RUN】走行

・PSD でゲート 1、ゲート 3、立体交差を検出

・立体交差検出で遷移

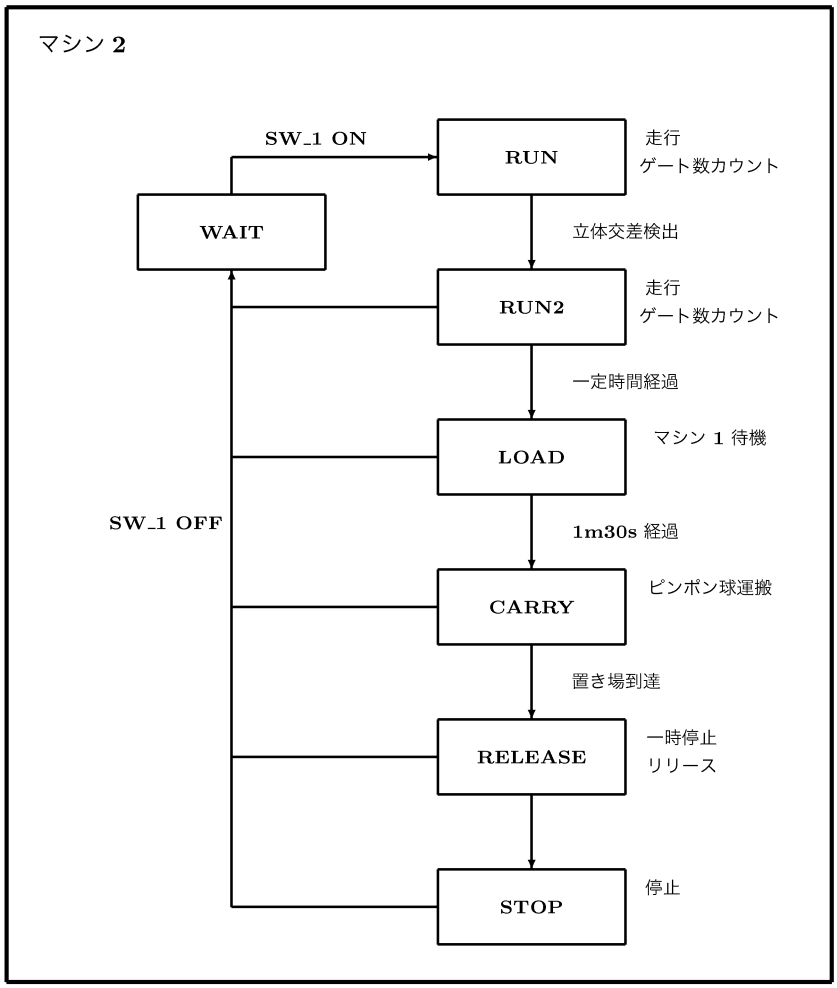
3. 【RUN2】マシン 1 待機場所まで移動
4. 【LOAD】スタートから 1 分 30 秒経過するまで待機
5. 【CARRY】ピンポン球運搬

・PSD でゲート 2、ゲート 1 を検出

・ゲート 1 検出後一定時間走行して状態遷移

6. 【RELEASE】アームを開いてピンポン球をリリース
7. 【STOP】停止

ピンポン球運搬は 1 回のみなので、マシン 2 はピンポン球置き場でリリース後、その場で停止する。なお、【WAIT】以外の状態にある時にスイッチを OFF にすると、各変数を初期化し、【WAIT】に戻るようになっている。



## 2.3　マシン製作途上に於ける変更点

### 2.3.1　マシン 1,2:全体の動きの簡素化

**当初想定していた動き**　構想段階では我々のチームは成果報告会で行なった動きよりも複雑な動きを考えていた。具体的には以下の表の通りであるが、端的な特長は

・ピンポン玉の受け渡しの回数が 2 回

・マシン 1 も坂を下って得点に向かう

この 2 点である。

| | マシン 1 | マシン 2 |
|---|---|---|
| 0 | H 区間ゲート 2 向きに待機 | A 区間ゲート 1 向きに待機 |
| | ↓ | ↓ |
| 1 | 坂に向けて発車 | 立体交差下に向けて発車 |
| | ↓ | ↓ |
| 2 | 登坂 | 立体交差下で待機 |
| | ↓ | ↓ |
| 3 | ピンポン球の取得 | |
| | ↓ | ↓ |
| 4 | 立体交差上まで運搬 | |
| | ↓ | ↓ |
| 5 | 立体交差上にてピンポン球放出 | |
| | ↓ | ↓ |
| 6 | 赤外線を発する | 赤外線を受ける |
| | ↓ | ↓ |
| 7 | 供給装置に向けて発車 | ピンポン球置き場にむけて発車 |
| | ↓ | ↓ |
| 8 | | ピンポン球の運搬 |
| | ↓ | ↓ |
| 9 | | ピンポン球置き場にてピンポン球を放出 |
| | ↓ | ↓ |
| 10 | 2〜7 を繰り返す | 2〜9 を繰り返す |
| | ↓ | ↓ |
| 11 | ピンポン球の取得 | 邪魔にならないように立体交差下に移動 |
| | ↓ | ↓ |
| 12 | 坂を下る | 立体交差下で停止。仕事終了。 |
| | ↓ | |
| 13 | ピンポン球置き場まで移動 | |
| | ↓ | |
| 14 | ピンポン球の放出 | |

ただ、作業を進めるにさまざまな問題が現れてきた。まず考えていた以上に坂を下るのが困難だったので、マシン 1 が坂を下って得点するという案が削られた。
次にコースを使用した調整時間がたりず、また大量得点よりもチームコンセプトに沿った確実な得点を目指すべきだと考えたため、受け渡しの回数も 2 度から 1 度に減らすこととなった。

### 2.3.2　マシン 1,2:回路の簡略化

**赤外線 LED による通信の廃止**　1/8(木) の作業終了後に廃止を決定。当初マシン 1 がピンポン球の受け渡しを完了後赤外線 LED を発光させ、マシン 2 がそれを感知しピンポン球の運搬を開始する予定であった。しかし残りの作業時間

と残された仕事内容を考えると仕事量をできるだけ削りたかった為廃止された。これに伴い、マシン 2 は受け渡し場所からの発車を時間制御に変更。複数回の運搬はしないことに。これにより LED 受信機の取り付け位置調整、その部分のプログラムの調整の必要がなくなった。
赤外線 LED の点灯と受信回路の動作確認には成功していたため以下に回路を記す。

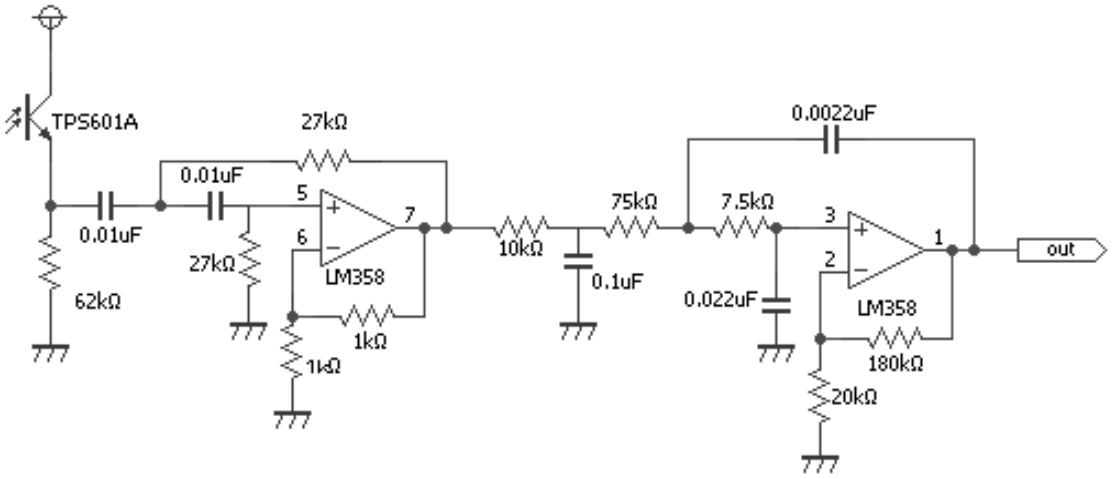



図 7: マシン 2 赤外線 LED 受信基板回路図

**マシン 1:供給装置検出用 200mm センサの廃止**　12/18(木) に廃止を決定。当初マシン 1 のピンポン球搭載車両に 200mm センサをとりつけ、それによってピンポン球供給装置に対するアームの位置を調整しようと考えていた。マシン 1 の供給電圧 (3.6V) のままでは 200mm センサを動作させることができないことに気付いた為、これを諦め、物理的に供給装置にアーム B を押し付ける事によって正確な位置に合わせるようにした。
これにより、200mm センサの固定という作業が減った。

### 2.3.3　マシン 1:アームの構造の変化

当初、図 12 のように R/C サーボ 1 つによるアームでピンポン球を押すことによりピンポン球の取得を試みていたが、12/1 に試行錯誤した結果、ピンポン球取得の際マシンのバランスが崩れてしまうことが分かり、急遽マシン設計の変更が求められた。要求されるポイントとして以下の 3 点が挙げられた。

- ピンポン球を取得する前に、何らかの形でピンポン球供給装置の下のアルミ板を掴む機構。
- ピンポン球を取得する際、固定部分とピンポン球を取得する部分に生じる 2 つのねじれの力に対処できる機構。
- マシン自体が対称性をもつ機構。

この 3 つを満たす機構として図 11 のような設計を採用した。進行方向右側の R/C サーボに付いているアームがピンポン球供給装置の下のアルミ板を捕らえる。ここでの重要点はアルミ板とアームが平行であることである。このことでねじれの力を打ち消すことができ、安定したピンポン球の取得が実現できるようになった。また図から分かるとおり R/C サーボが左右対称に設置されており、登坂するマシンに必要不可欠であるバランスがよくなった。

### 2.3.4　マシン 2:車両数の 3 両から 2 両への減少

当初図 13 のようにマイコンボードを乗せるマイコン台が 2 両目にありピンポン球搭載車が 3 両目であったが、先頭車両の PSD で立体交差やゲートを検知し、走行時間を計って立体交差やピンポン球置き場にピンポン球搭載車の位置をあわせていたので誤差が大きかった。その誤差を小さくするため PSD センサとカゴの距離が近くなるようにピンポン球搭載車に MCU ボートも搭載し 2 両編成に変更した (図 14)。これによりマシンの位置合わせの精度が向上し、また連続カーブの走行性があがった。なお、MCU ボートを搭載しても、ピンポン球は 30 個近く入るので、容量に問題

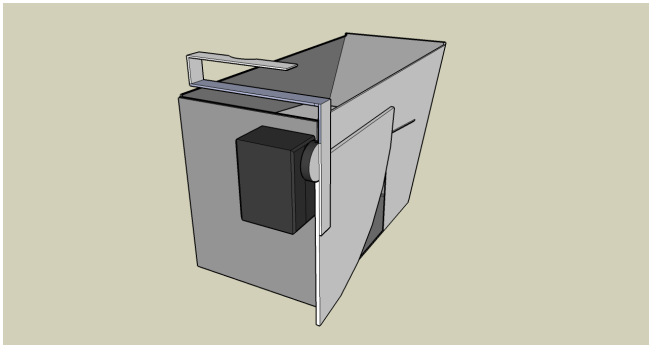


図 8: アーム改善前

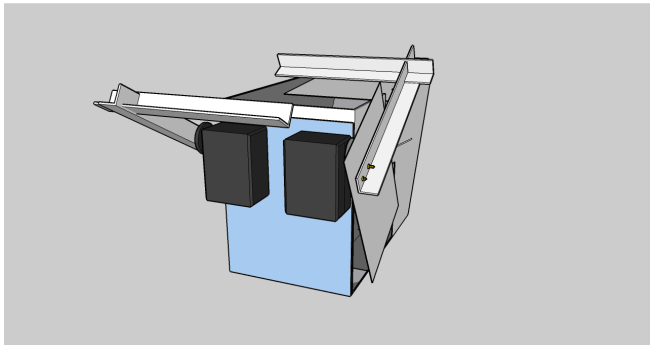


図 9: アーム改善後

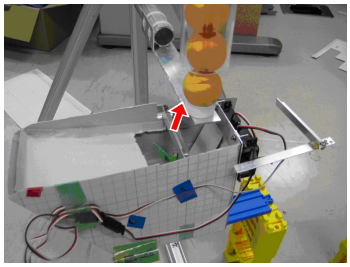


図 10: 初期状態

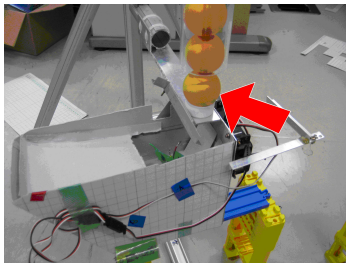


図 11: アーム B による固定

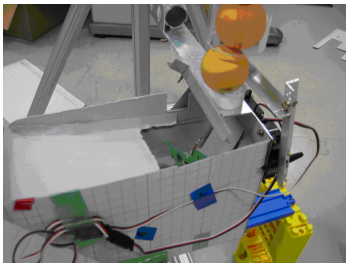


図 12: アーム A による取得

は無し（マシン 1 は 25 個程度収容）。この変更は第 2 成果発表会前の最終日（1/16）に行ったがマシンの構造材の多くが工作用紙だったため、配線を通したりといった細かな加工が容易であり短時間で行なう事ができた。

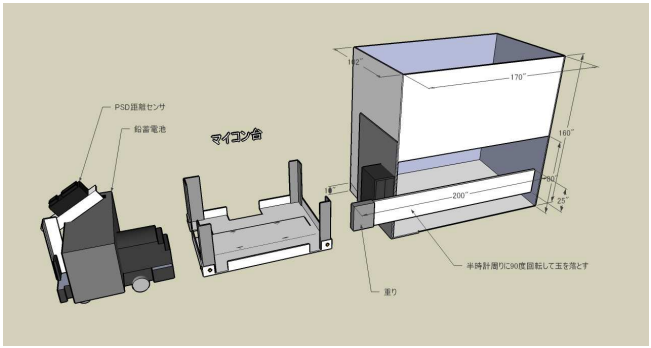



図 13: 車両数減少前

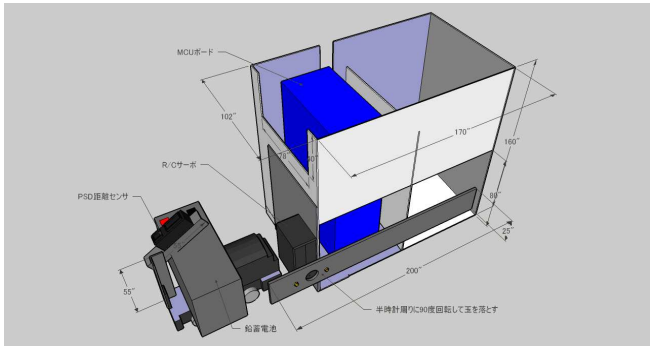



図 14: 車両数減少後

## 2.4 課題 2 反省

### 2.4.1 課題 2 結果

| ゲート不通過 | ゲート接触 | 2 線路外接触 | リトライ | デモ | 加点 | 減点 | 合計 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 210 | 0 | 210 |

全く減点を受けることなくピンポン玉 21 個を運搬、得点をあげることができた。
マシン 1 が 22 個のピンポン玉を取得し、マシン 2 への受け渡し時にはピンポン玉は 1 つもこぼすことがなかったが、最後の得点時に 1 つピンポン玉が枠からこぼれてしまった。

### 2.4.2 反省点及びまとめ

我々の班は一応の成功はおさめたものの、当初の予定をより簡単な構成に変更しての成功であった。
第 2 成果報告会までに 2 台のマシンをつくり、確実に安定して多くのピンポン球を運搬する事ができるようになったの

は、総合的に見て良かったと思う。アームの数の増加や、車両数等の減少といった改良を柔軟に行なう事ができたが、最終的に使用することのなかった赤外線通信の回路やプログラムに作業時間を使ってしまった。その分の作業時間があれば慌しく最終調整をしなくてもすんだと考えられ、総作業量に関して初期構想段階での見通しが甘かった。

## 3　スケジュール表

| 仕事内容 | 担当者 |
|---|---|
| 第一課題プログラム | 上村・片岡 |
| 第二課題プログラム | 上村・片岡 |
| PSDセンサ動作チェック | 片岡 |
| 第一課題支給マイコンの死重化及び鉛蓄電池の除去 | 片岡 |
| 第一課題の得点向上の改良 | 片岡・高木 |
| サーボ動作チェック | 片岡 |
| 車両間赤外線通信システムの製作 | 片岡 |
| マイコン収納車両の製作 | 高木 |
| 登坂車両用ピンポン玉運搬車両の製作 | 高木 |
| 得点車両用ピンポン玉運搬車両の製作 | 高木 |
| PSDセンサ取り付け(2台分) | 渡部 |
| 第二課題用動力車両の製作 | 渡部・片岡 |
| ピンポン玉等取用アームの製作 | 渡部 |
| 第一課題最終調整 | 全員 |
| ピンポン玉等取実験・改良 | 渡部・高木 |
| 赤外線通信実験・改良 | 片岡・上村 |
| ピンポン玉受け渡し実験・改良 | 渡部・高木 |
| 模擬競技・改良 | 全員 |
| 第二課題最終調整 | 全員 |
| 他班の情報収集 | 全員 |

| date | 11/1 | 11/13 | 11/17 | 11/20 | 11/27 | 12/1 | 12/4 | 12/8 | 12/11 | 12/15 | 12/18 | 12/22 | 1/8 | 1/15 | 1/16 | 1/19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Workshop | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 成果発表会1 | 10 | 11 | 12 | 13 | ex | 成果発表会2 |

予定作業期間
実作業時間

プログラム班　上村　片岡
メカ班　高木　渡部
回路班　片岡

| 主な製作スケジュールとの変更点 | 変更日時 | 具体的変更内容 |
|---|---|---|
| ピンポン玉寄取用アームの形状変更 | 第6回 | 1本ではピンポン玉が取れないことから、2本に変更 |
| マイコン間の通信システム | 第10回 | 製作時間的な問題から通信は行わず、下のマシンは時間制御で発進するようにする |
| 200mm光センサの使用中止 | 第10回 | 供給可能電圧の関係から使用できないことが判明し、使用を断念 |
| タッチセンサの導入 | 第10回 | 200mm光センサのかわりにタッチセンサを導入 |

| タスク名改良・改善の仕事中に行ったこと | 変更日時 | 具体的変更内容 |
|---|---|---|
| 第一課題用マシンの電池取り付け | 第7回 | 取り付け位置を低くし、低重心化を達成する |
| ピンポン玉寄取用アームの稼動範囲の改善 | 第11回 | 稼動範囲をずらし、運搬車両部分に食い込むように変更　それにともない運搬車両部分の形状もそれに対応できるように変更 |
| 重りの装着 | 第12回 | 上のマシンの重心を低くするため、ピンポン玉運搬部分 下部に重りを装着 |
| 下のマシンの車両編成を3両から2両に変更 | ex | 下のマシンの安定性の問題からマイコン車両をなくし、ピンポン玉運搬部分にマイコンを挿入できるように改造 |
| 上のマシンのバック走行時の安定性改善 | ex | 上のマシンの動力車後部にピンポン玉運搬部分を安定して押せるよう出っ張りをつける |
| 上のマシンのプログラム | ex | 下のマシンが時間制御により発進することを利用し、ピンポン玉を確実に受け渡すためじたばたする動作を追加 |
| ピンポン玉寄取用アーム供給装置固定側の改善 | ex | 固定用アームのネジが固定の邪魔になっていた部分をアームを厚くすることで改善 |

# 4　会計表

A 類追加注文および B 類購入品

| 日付 | 類 | 部品名 | 形式・型番 | 単価 (税抜) | 個数 | 金額 (税込) | 金額計 (税込) |
|---|---|---|---|---|---|---|---|
| 10/30 | A | EH500 電気機関車 | S-25 | ￥1,600 | 2 | ￥3,360 | ￥3,360 |
| 10/30 | B | 赤外線 LED | TLN105B | ￥40 | 4 | ￥168 | ￥3,528 |
| 10/30 | B | フォト Tr | TPS601A | ￥240 | 2 | ￥504 | ￥4,032 |
| 10/30 | B | フォト Tr | TPS615 | ￥50 | 2 | ￥105 | ￥4,137 |
| 11/06 | A | 電池ボックス | 単三型 | ￥60 | 2 | ￥126 | ￥4,263 |
| 11/06 | B | ビス | M3x20 | ￥400 | 1 | ￥420 | ￥4,683 |
| 11/06 | B | アルミアングル | 10x10x1000x1.2 | ￥140 | 2 | ￥294 | ￥4,977 |
| 11/06 | B | アルミ板 | 400x400x5.0 | ￥800 | 1 | ￥840 | ￥5,817 |
| 11/06 | B | アクリル板 | 300x300x5.0 | ￥1,680 | 1 | ￥1,764 | ￥7,581 |
| 11/10 | B | 工作用紙 | | ￥34 | 10 | ￥357 | ￥7,938 |
| 11/27 | A | バッテリ用ヒューズ | ミゼットガラス管，2A | ￥30 | 4 | ￥126 | ￥8,064 |
| 12/01 | B | マイクロスイッチ | SS5GL | ￥140 | 1 | ￥147 | ￥8,211 |
| 12/01 | B | 積層セラミックコンデンサ | 0.1 μ F | ￥30 | 2 | ￥63 | ￥8,274 |
| 12/01 | B | ユニバーサル基板 | ICB288 | ￥75 | 2 | ￥158 | ￥8,432 |
| 12/01 | B | OP アンプ IC | LM358N | ￥20 | 2 | ￥42 | ￥8,474 |
| 12/01 | B | 緑色 LED | L-513SGT-A-SS | ￥10 | 1 | ￥11 | ￥8,484 |
| 12/01 | B | カーボン抵抗 | ※1 | ￥5 | 12 | ￥63 | ￥8,547 |
| 12/01 | B | セラミックコンデンサ | 0.022uF,0.01uF × 2, 0.0022uF | ￥20 | 4 | ￥84 | ￥8,631 |
| 12/01 | B | MOSFET | FET2SK2231 | ￥100 | 1 | ￥105 | ￥8,736 |
| 12/01 | B | ピンヘッダ | 1x40 | ￥80 | 1 | ￥84 | ￥8,820 |
| 12/01 | B | タクトスイッチ | 高さ 7mm | ￥20 | 1 | ￥21 | ￥8,841 |
| 12/01 | B | 酸化金属皮膜抵抗 | 2W3.9 Ω | ￥30 | 2 | ￥63 | ￥8,904 |
| 12/01 | B | アルミ電解コンデンサ | 耐圧 10V 容量 220uF | ￥20 | 2 | ￥42 | ￥8,946 |
| 12/01 | B | モータドライバ | TA8429HQ | ￥390 | 1 | ￥410 | ￥9,356 |
| 12/01 | B | 三端子レギュレータ | TA7805S | ￥70 | 2 | ￥147 | ￥9,503 |
| 12/01 | B | IC ソケット | 28P スリム 300mil タイプ | ￥20 | 1 | ￥21 | ￥9,524 |
| 12/01 | B | 工作用紙 | | ￥34 | 5 | ￥179 | ￥9,702 |
| 12/01 | B | サラネジ | M3x100 | ￥500 | 1 | ￥525 | ￥10,227 |
| 12/04 | B | サーボホーン | | ￥50 | 2 | ￥105 | ￥10,332 |
| 12/04 | B | ビス | M2x10 | ￥40 | 1 | ￥42 | ￥10,374 |
| 12/18 | A | PSD 距離センサ | GP2D12 | ￥750 | 2 | ￥1,575 | ￥11,949 |
| 12/18 | B | RC サーボ延長コード | | ￥290 | 3 | ￥914 | ￥12,863 |

※1 ：180k,75k,62k,27k(2 個),20k,10k,7.5k,1k(2 個),330,62 Ω

C 類購入品

| 部品名 | 単価 (税抜) | 個数 | 金額 (税込) | 金額計 (税込) |
|---|---|---|---|---|
| ケーブル | ￥480 | 1 | ￥504 | ￥504 |
| 電池ボックス | ￥40 | 4 | ￥168 | ￥672 |
| AVR マイコン | ￥470 | 1 | ￥494 | ￥1,166 |
| トルクチューンモータ | ￥300 | 1 | ￥315 | ￥1,481 |

# 5 プログラム

## 5.1 マシン1

```
1  //-------------------------------------------
2  // kadai2.c
3  // 創造設計課題 2 上のマシンプログラム for ATmega168
4  // 登坂、ピンポン球の補給、ピンポン球の受け渡しを行なう
5  // 赤外線LED機能は消去
6  // メインセンサはPSDセンサ
7  //-------------------------------------------
8  #include <avr/io.h>
9  #include <avr/interrupt.h>
10 #include "adc.h"  //AD変換に関するヘッダ
11 #include "timer1.h"  //タイマ1の設定に関するヘッダ
12 #include "timer0.h"  //タイマ0の設定に関するヘッダ
13
14 #define ADDATMAX 450
15 unsigned char addat[ADDATMAX];
16
17 #define ICR1_AS_TOP  2499
18 #define VCC  500
19 #define MAXCNT 1023
20 #define push_degreeA 235
21 #define base_degreeA 190
22 int cycle = 0; //モーター PWMの割り込み周期 MAX=
23 unsigned int cnt = 0; //割り込み回数総計
24 int volatile UR_LED = 0;//赤外線 LEDの点灯(1のとき点灯)
25 int volatile PWM_duty = 0;
26 long int volatile wait_time=0;
27 //■■■■■■■■■■■■■■■■■関数いろいろ■■■■■■■■■■■■■■■■■
28 //////////////////その他の関数/////////////////////////////////////////////////
29 void io_init(void);    //ポートの設定、初期化
30 void timer_init(void);   //タイマの設定、初期化
31 void LED(char on);    //on =1 で緑LED点灯   on=0 で消灯
32 int get_meter(void);   //PSDで距離を読む関数 返り値=PSDの距離(mm)
33 //////////////////待機関数いろいろ/////////////////////////////////////////////
34 void wait(volatile int time); //待機ルーチン    引数 =待機時間
35 void wait_sw(void);    //スイッチが押されるのを待つ
36 void wait_nsw(void);   //マイクロスイッチが押されるのを待つ
37 void wait_PSD(int meter);  //PSDセンサが一定値以下になるのを待つ 引数 =目的距離
38 void wait_200(void);   //200mmセンサがが反応するのを待つ    //結局使用せず
39 /////////////////動作関数いろいろ/////////////////////////////////////////////
40 void forward (int speed);  //前へ進む 引数 = スピード (Max16 Min0)
41 void stop (void);    //停止
42 void back (int speed);   //後へ進む 引数 = スピード(Max16 Min0)
43 static void timer1_ic_interrupt(void); //タイマ1割り込み //使用せず
44 static void timer0_ov_interrupt(void); //タイマ0割り込み //駆動モータ用PWMなどの処理に使用
45 /////////////////複雑な動作関数/////////////////////////////////////////////////
46 void start(void);    //スタート動作
47 void climb(void);    //登坂＆補給装置下位置調整
48 void capture(void);    //ピンポン玉回収動作
49 void release(void);    //回収後ピンポン玉解放まで
50
51 //■■■■■■■■■■■■■■■■メイン関数■■■■■■■■■■■■■■■■■■
52 int main(void)
53 {
54     ad_init(0);  //ad変換初期化、PSDセンサを読み込む状態にセット
55     io_init();  //ポートの初期化
56     timer_init(); //タイマの初期化
57     LED(1);   //LEDの点灯
58
59     wait_sw();  //スタートスイッチを待つ
60     start();  //スタート!
61     climb();  //登坂＆補給装置下位置調整
62     capture();  //ピンポン玉回収動作
63     release();  //受け渡し地点まで移動解放動作
64
65     return 0;
66 }
67 //■■■■■■■■■■■■■■■■その他の関数■■■■■■■■■■■■■■■■■■■■
68 ////////////////////////ポートの設定、初期化//////////////////////////////
69 void io_init(void)
70 {
71     //【ポートB使用内訳表】
72     // 駆動モーター用PWMの吐き出し  FB0
```

```
    // サーボの制御 1 PB1
    // サーボの制御 2 PB2
    // 駆動用モータ 1 PB6  //モーターの正転逆転ブレーキフリーを制御するための信号線が 2本ある。
    // 駆動用モータ 2 PB7

    //DDRB は 11000111 下位 3bit と上位 2bit が出力
    DDRB = 0xc7; //0xc7 = 1100 0111
    PORTB = ~0xc6; //入力ポートは全部プルアップ 出力ポートは全てL、PWM吐き出しのポートは H

    //【ポートD 使用内訳表】//スイッチは負論理
    // スイッチ FD2   //スタート等に使うタクトスイッチ
    // 緑LED  FD3   //デバグ用緑LED
    // マイクロsw PD5   //赤い車止め検出用マイクロスイッチ
    // 赤外管理 FD6   //結局使用しなかったが実装はされている。
    // 200mm  FD7   //結局使用しなかったが実装はされている。

    //DDRD 01001000   //LED と赤外線 LED だけ出力
    DDRD = 0x48;    //0x48 = 0100 1000
    PORTD = 0xff;    //入力全てプルアップ、出力全て H


    //【ポートC 使用内訳表】
    // 全てAD 変換に使用できるようにしておく。
    // PSD センサ PC0

    //DDRC 00000000
    DDRC = 0x00;  //全て入力

}
//////////////////////////タイマの設定、初期化////////////////////////////////
void timer_init(void)
{
    //////////タイマ 1関連///////////
    //タイマ 1はサーボの制御に利用
    //16bitPWM
    //以下、設定
    timer1_ic_enable();
    timer1_ic_set_interrupt(timer1_ic_interrupt);
    timer1_set_wave_mode(WGM1_FAST_PWM_ICR1);
    timer1_set_coma_mode(COM_A_HIGH_FIRST);
    timer1_set_comb_mode(COM_B_HIGH_FIRST);
    timer1_set_count(0);
    timer1_set_icr1(ICR1_AS_TOP);

    //アームをそれぞれスタートの角度に設定
    timer1_set_ocr1a(90); //アーム a を 90 の位置に。a が下のコネクタ、a がメインアーム
    timer1_set_ocr1b(182); //アーム b を 182 の位置に。b が上のコネクタ、b がサブアーム
    //タイマ 1をそれぞれスタート
    timer1_start(TIMER1_CK_8); //(8MHz/8)/20000 = 50Hz
    TCCR0A = 0x00;
    TCCR0B = 0x03;

    //////////タイマ 0関連///////////
    //タイマ 0は駆動用モータのPWM や赤外線 LED の点灯に利用
    //一定周期で割り込み
    //以下設定
    timer0_ov_enable();
    timer0_ov_set_interrupt(timer0_ov_interrupt);
    timer0_set_count(0);

    sei();//割り込みを有効に！


}
////////////////////////////////////////////////////////
void LED(char on)
{
    if(on==1){
        PORTD |= 0x08;//LED を強制 H に
    }
    else{
        PORTD &= ~0x08;//LED を強制 L に
    }
}
/////////////////PSD センサの読み込み//////////////////
int get_meter(void)
{
    int adc,adv,meter;
    ad_init(0);        //PC0 を選択
```

```
    adc = ad_get();        //電圧の元を取得
    adv = (int)((VCC * (long)adc) / MAXCNT);//電圧を計算

    if(adv == 0){
        meter=7777;}    //ゼロ割を避けるため適当に大きな値を入れる
    else{
        meter = (25500 / adv) + 14 ; //距離式
    }
    return meter;
}
//■■■■■■■■■■■■■■■待機関数いろいろ■■■■■■■■■■■■■■■■■■■■
//////////////////一定時間待機//////////////////////
void wait(volatile int time)
{
    wait_time = 0;
    while(wait_time <= (time * 3 ))
    {
        wait_time = wait_time;
    }
}
///////////////スタートスイッチを待つ//////////////////
void wait_sw(void)
{
    int volatile stateD = PIND;
    while((stateD & 0x04) != 0)
    {
          stateD = PIND;
    }
}
//////////////マイクロスイッチを待つ//////////////////
void wait_nsw(void)
{
    int volatile oldD = 0;
    int volatile old2D = 0;
    int volatile stateD = PIND;
    while((((stateD & 0x20) != 0)||((oldD & 0x20) != 0))||((old2D & 0x20) != 0))
    {
        old2D= oldD;
        oldD = stateD;
        stateD = PIND;
        wait(1);
    }
}
/////////////PSDが一定値以下になるまで待つ///////////////
void wait_PSD(int meter)
{
    int volatile now_meter,old_meter;

    now_meter = get_meter();
    old_meter = 1000;
    // ノイズがPSDセンサに乗って誤動作するので時間を置いて2回サンプリング
    while( (now_meter >= meter) ||(old_meter >= meter) ){
        old_meter = now_meter;
        now_meter = get_meter();
        wait(1);
    }
    //PSDの値がmeterよりも小さくなると待機終了
}
////////////200mmセンサが反応するまで待つ/////////////// //結局使用せず
void wait_200(void)
{
        int volatile stateD = PIND;
    while((stateD & 0x80) != 0)
    {
        stateD = PIND;
    }
}
//■■■■■■■■■■■■■■■基本動作いろいろ■■■■■■■■■■■■■■■■■■■■
////////////////////前進////////////////////
void forward(int speed)
{
    PWM_duty = speed;
    PORTB |= 0x40; //Hに
    PORTB &= ~0x80; //Lに
}
////////////////////後進////////////////////
void back(int speed)
{
    PWM_duty = speed;
    PORTB |= 0x80; //Hに
```

```
232     PORTB &= ~0x40; //L に
233 }
234 //////////////////ブレーキ//////////////////////
235 void stop(void)
236 {
237     back(16);
238     wait(10);
239     PWM_duty = 20;
240     PORTB |= 0x80; //H に
241     PORTB |= 0x40; //H に
242 }
243 //■■■■■■■■■■■■■■■割り込み関数■■■■■■■■■■■■■■■■■■■■
244 ////////////////////////////////////////////////////////////////////////////
245 static void timer1_ic_interrupt(void)
246 {
247
248 }
249 ////////////////////////タイマ 0割り込み//////////////////////////////
250 static void timer0_ov_interrupt(void)
251 {
252     TCNT0 = 255-15;
253     cnt++;       //割り込み回数総計+1
254     wait_time++;    //待機時間計測用変数+1
255
256     if((cnt%16) < PWM_duty ){ //PWM_duty の値によって PORTB の下位 1bit が H になるか L になるか決定
257     PORTB |= 0x01;    //cnt を 16 で割った値が PWM_duty より小さければ H
258     }
259     else{
260     PORTB &= ~0x01;    //cnt を 16 で割った値が PWM_duty より大きければ L
261     }
262
263     //if(UR_LED == 1){   //赤外線LED を点滅させる    //結局使用せず
264     //PORTD ^= 0x40;   //PD6 を反転
265     //}
266 }
267 //■■■■■■■■■■■■■■■複雑な動作■■■■■■■■■■■■■■■■■■■■
268 ////////////////////////////////スタート////////////////////////////////
269 void start(void)
270 {
271     LED(0);   //LED 消灯
272     forward(9);
273     wait(1200);  //適当に前進  この速度で進むと登坂できずに坂の入り口付近で停止する。
274     forward(0);
275     wait(100);  //勢いがついていると転倒する可能性があるので一旦停止。
276 }
277 ////////////////////////////////登坂＆供給装置に位置合わせ//////////////////
278 void climb(void)
279 {
280
281     forward(14); //登坂開始！
282     wait(1200);  //登坂中、この時間はかなりアバウト
283
284     /////供給装置に位置合わせ//////
285     wait_PSD(300); //供給装置まで距離 300まで接近したら
286     stop();   //ちょっと停止
287     wait(200);  //
288
289     forward(8);  //ゆっくり前進
290     wait_PSD(140); //供給装置まで距離 140まで接近したら
291     stop();   //停止
292     wait(500);  //
293
294     forward(8);  //ふたたびゆっくり一定時間前進
295     wait(600);  //供給装置を確実に行き過ぎる時間
296     stop();   //供給装置を行き過ぎて停止
297     wait(100);  //ちょっと待つ
298
299     timer1_set_ocr1b(203);//サブアームを 10度ぐらいの位置に //
300     wait(200);  //ちょっと待つ
301     back(7);  //ゆっくりバック
302     wait(380);  //サブアームを供給装置に押し付ける
303     back(6);  //念の為もう少し押し付ける
304     wait(70);
305     stop();   //停止
306     wait(200);
307     forward(8);  //先頭車両が 2両目に接触してると姿勢が良くないので
308     wait(25);  //本当に少しだけ前進
309     stop();
310     wait(100);
```

```
311     LED(1);   //LED 点灯
312
313 }
314 ////////////////////////////ピンポン玉回収動作////////////////////////////////
315 void capture(void)
316 {
317     timer1_set_ocr1a(push_degreeA + 4); //アーム A でピンポン玉を押す 深めに押す
318     wait(125);        //アーム A が動くのを待つ
319     timer1_set_ocr1a(base_degreeA);  //アーム A を元の位置に戻す
320     wait(250);
321     timer1_set_ocr1a(push_degreeA + 4); //2回目 深めに押す
322     wait(125);
323     timer1_set_ocr1a(base_degreeA);  //元の位置に戻す
324     wait(250);
325
326     timer1_set_ocr1a(push_degreeA + 2); //3回目 少し深く押す
327     wait(125);
328     timer1_set_ocr1a(base_degreeA);  //元の位置に戻す
329     wait(250);
330     timer1_set_ocr1a(push_degreeA + 2); //4回目 少し深く押す
331     wait(125);
332     timer1_set_ocr1a(base_degreeA);  //元の位置に戻す
333     wait(250);
334
335     timer1_set_ocr1a(push_degreeA);  //5回目 普通に押す
336     wait(125);
337     timer1_set_ocr1a(base_degreeA);  //元の位置に戻す
338     wait(250);
339     timer1_set_ocr1a(push_degreeA);  //6回目 普通に押す
340     wait(120);
341     timer1_set_ocr1a(base_degreeA);  //元の位置に戻す
342     wait(250);
343     timer1_set_ocr1a(push_degreeA + 2); //7回目 少し深く押す
344     wait(130);        //少しだけ長く押す
345     timer1_set_ocr1a(base_degreeA);  //元の位置に戻す
346     wait(250);
347     timer1_set_ocr1a(push_degreeA);  //8回目 普通に押す
348     wait(125);
349     timer1_set_ocr1a(base_degreeA);  //元の位置に戻す
350     wait(250);
351     timer1_set_ocr1a(push_degreeA + 2); //9回目 少し深く押す
352     wait(125);
353     timer1_set_ocr1a(base_degreeA);  //元の位置に戻す
354     wait(250);
355
356     timer1_set_ocr1a(90); //アーム A がゲートをくぐれる状態に戻す
357     timer1_set_ocr1b(195); //アーム B がゲートをくぐれる状態に戻す(20度)
358     wait(250);
359 }
360 ////////////////////////////回収後ピンポン玉解放まで////////////////////////////
361 void release(void)
362 {
363     forward(10); //適当に前進
364     wait(700);
365     forward(8);  //適当に減速
366     wait(600);
367     forward(7);  //適当に減速
368     wait(400);
369     wait_nsw();  //赤い車止めを検出
370     LED(0);   //LED 消灯
371     stop();   //停止
372     wait(200);
373
374     back(7);  //ゆっくり一定時間バック
375     wait(120);  //
376     stop();   //これでちょうど受け渡しポイントに移動
377
378     ////////////////解放動作//////////////////
379
380     timer1_set_ocr1b(285); //アーム B 全開
381     wait(500);    //しばし待つ
382     timer1_set_ocr1b(230); //アーム B を適当に動かして揺らす
383     wait(150);
384     timer1_set_ocr1b(265); //アーム B 開く
385     wait(250);
386     timer1_set_ocr1b(230); //適当に動かす
387     wait(150);
388     timer1_set_ocr1b(285); //アーム B 全開
389     wait(250);
```

```
390     timer1_set_ocr1b(195); //アーム B を一旦閉じる
391     wait(250);
392
393     forward(10);   //前進  //一旦前進後進して機体を揺らす
394     wait(70);
395     forward(7);
396     wait_nsw();    //車止め検出まで前進
397     stop();    //停止
398     wait(200);
399     back(7);   //バック
400     wait(120);    //ちょうど受け渡しポイントに移動
401     stop();
402
403     timer1_set_ocr1b(285); //アーム B 全開
404     wait(350);    //しばし待つ
405
406     forward(10);   //前進  //アームB を開いたまま前後に機体を揺らす
407     wait(120);
408     stop();
409     wait(200);
410     back(10);
411     wait(100);
412     forward(10);
413     wait(40);
414     stop();
415     wait(350);
416
417     timer1_set_ocr1b(195); //B を 20 度に //一旦アームB を閉じる
418     wait(250);
419
420     forward(7);    //車止めまで前進
421     wait_nsw();
422     stop();
423     wait(200);
424
425     back(7);   //バック
426     wait(120);    //受け渡しポイントまでバック
427     stop();
428
429     timer1_set_ocr1b(285); //B を全開に
430     wait(500);    //しばし待つ
431     timer1_set_ocr1b(195); //B 一旦閉じる
432     wait(250);
433
434     forward(7);    //車止めまで前進
435     wait_nsw();
436     stop();
437     wait(200);
438
439     timer1_set_ocr1b(285); //B を全開に
440     back(7);   //しながらバック
441     wait(120);    //受け渡しポイントまでバック
442     stop();
443     wait(500);    //しばし待つ
444
445     forward(10);   //前進  //機体を前後に揺らす
446     wait(100);
447     stop();
448     wait(200);
449     back(10);    //後進
450     wait(100);
451     forward(10);   //前進
452     wait(100);
453     stop();
454     wait(200);
455     back(10);    //後進
456     wait(100);
457     stop();    //停止
458     wait(200);
459
460     forward(7);    //前進
461     wait_nsw();    //車止めまで前進
462     stop();
463     wait(200);
464     back(7);   //受け渡し地点までバック
465     wait(120);
466     stop();
467     wait(400);
468     timer1_set_ocr1b(195); //アーム B を一旦閉じる
```

```
469     wait(250);
470     timer1_set_ocr1b(230); //アーム B を開け閉めして揺らす
471     wait(150);
472     timer1_set_ocr1b(265);
473     wait(250);
474     timer1_set_ocr1b(230);
475     wait(150);
476     timer1_set_ocr1b(265);
477     wait(250);
478
479     //UR_LED = 1;   //赤外線LED 機能は削除
480     stop();
481     LED(1);
482     wait(200);
483     wait_sw(); //おわり
484 }
```

## 5.2 マシン 2

### 5.2.1 メイン

```
1  //=================================================================
2  // ファイル名： SS2kadai02a.c   (09/01/16)
3  // 内容：   課題２下マシン用プログラム
4  //       スイッチ１ははじめは下に。
5  //
6  //   ゲート１手前より時計回りに走行
7  //   ゲート１→ゲート３→立体交差(積み込み)→ゲート２→置き場
8  //=================================================================
9  // 配線
10 //  P100：PSD 出力
11 //  HB2：DC モータ（外側に黒線をつなぐ）
12 //  RCS0：サーボ PWM 出力
13 //
14 // 初期設定
15 //  HB2 ジャンパは下にする
16 //  SW1 は下にセット
17 //=================================================================
18 //
19 // 創造設計第二試作検討  kadai0101.c を変更
20 //sprintf()を消去､代理関数を利用
21 //=================================================================
22 #include <string.h>    //strcpy()を使うため
23 #include "sfr26.h"    // CAKSmini 用定義ファイル
24 #include "LCDfunc.h"    // LCD 表示
25 #include "nosprintf.h"    //sprintf()代理関数
26 #include "RCservo.h"    //RC サーボを使う
27
28
29 // プロトタイプ宣言
30 //使わない関数についても記述する必要がある
31 void ta0int( void );    //タイマー A0 割込み関数
32 void ta1int( void );    //タイマー A1 割込み関数
33 void ta2int( void );    //タイマー A2 割込み関数
34 void ta3int( void );    //タイマー A3 割込み関数
35 void ta4int( void );    //タイマー A4 割込み関数
36 void tb0int( void );    //タイマー B0 割込み関数
37 void tb1int( void );    //タイマー B1 割込み関数
38 void tb2int( void );    //タイマー B2 割込み関数
39 void adint( void );     //AD 変換割込み関数
40 void int0int( void );    //外部割込み 0関数
41 void int1int( void );    //外部割込み 1関数
42 void int3int( void );    //外部割込み 3関数
43 void int4int( void );    //外部割込み 4関数
44 void int5int( void );    //外部割込み 5関数
45 void uart0traint( void );   //uart0 送信完了割り込み
46 void uart0recint( void );   //uart0 受信完了割り込み
47 void uart1traint( void );   //uart1 送信完了割り込み
48 void uart1recint( void );   //uart1 受信完了割り込み
49 void uart2traint( void );   //uart2 送信完了割り込み
50 void uart2recint( void );   //uart2 受信完了割り込み
51
52
53 //割り込みの宣言
```

```
54 //使わない割り込みも記述する必要がある
55 #pragma INTERRUPT ta0int
56 #pragma INTERRUPT ta1int
57 #pragma INTERRUPT ta2int
58 #pragma INTERRUPT ta3int
59 #pragma INTERRUPT ta4int
60 #pragma INTERRUPT tb0int
61 #pragma INTERRUPT tb1int
62 #pragma INTERRUPT tb2int
63 #pragma INTERRUPT adint
64 #pragma INTERRUPT int0int
65 #pragma INTERRUPT int1int
66 #pragma INTERRUPT int3int
67 #pragma INTERRUPT int4int
68 #pragma INTERRUPT int5int
69 #pragma INTERRUPT uart0traint
70 #pragma INTERRUPT uart0recint
71 #pragma INTERRUPT uart2traint
72 #pragma INTERRUPT uart1recint
73 #pragma INTERRUPT uart2traint
74 #pragma INTERRUPT uart2recint
75
76
77 // マクロ定義
78 #define cnt_ta1 31250-1   // タイマA1カウンタ値
79 #define ta2_max 0xd000   //モータへのPWM入力最大値（0x0000~0xfffe）
80 #define PSD_D 180    //PSD検知距離(mm)
81
82 //STATE定義
83 #define sWAIT 0    //初期待機
84 #define sRUN 10    //走行
85 #define sRUN2 20   //PSDセンサが交差感知～上マシン待機場間
86 #define sLOAD 30   //待機中...
87 #define sCARRY 40   //運搬
88 #define sRELEASE 50   //ピンポン球解放
89 #define sRESET 60
90 #define sSTOP 70
91
92 //関数定義
93 float AD0_read( int average_count ); //AD変換
94 void AD_init(void);      //AD変換初期化
95
96 // 変数の宣言
97 float psdout;      //PSDセンサ出力値
98 int ta2_tmp;      //モータPWM一時出力値
99 unsigned long int timer_count;  //タイマ（スタート～）
100 unsigned int gate_count;   //ゲート数
101 unsigned char LCD_buff[33];   //LCD用バッファ
102
103
104 //=================================================================
105 // 関数名 :main()
106 // 機能  : 50msec周期で割り込み関数を呼ぶ設定
107 //    : タイマA0のカウントソースをf32とする
108 //    : 1カウントは1/(20MHz/32)=1.6μsec
109 //    : タイマ値の初期値 = 0.05sec/1.6μsec = 31250
110 //=================================================================
111 void main( void ) {
112    unsigned int state;      //状態
113    unsigned char rcs_errcode;    //RCサーボのエラーコード
114
115    unsigned long int wait_start_time;  //センサを読まない時間始め
116    unsigned int wait_time;     //センサを読まない時間（50msec × wait_time）
117    unsigned long int posi_start_time;  //時間制御用
118    unsigned int posi1_time;    //ゲート１～ピンポン球置き場にかかる時間
119    unsigned int posi2_time;    //橋の下～ロード場所間にかかる時間
120    unsigned long int release_start_time; //リリース時間制御用
121    unsigned int release_time;    //リリース時間
122
123
124    //変数の初期化
125    psdout = 0.0;
126    timer_count = 0;
127    gate_count = 0;
128    wait_start_time = 0;
129    posi_start_time = 0;
130    state = sWAIT;
131
132    //各時間制御用パラメータ
```

```
133     wait_time = 30;     //30× 50ms = 1500ms(1.5s) センサを読まない
134     posi1_time = 34;    //34× 50ms = 1700ms(1.7s) ゲート1から置き場間
135     posi2_time = 4;     //4× 50ms = 200ms(0.2s)  橋の下→上マシン待機場所間
136     release_time = 100;   //100× 50ms = 5000ms(5.0s) 置き場へのリリースにかける時間
137
138     LCD_init();      // LCD初期設定
139     LCD_cls();      // 全て消去
140     AD_init();     // AD変換の初期化
141
142     // I/Oポート初期化
143     pd1 = 0x00;      // ポート 1方向レジスタ  入力
144     pd6 = pd6 & 0xef;    // P6-4: SW1入力
145     pd7 = 0xff;      // ポート 7出力設定
146     p7 = ~0x00;      // ポート 7出力L(LED消灯)
147
148     //PWM波形出力準備(TA0初期化)
149     ta0mr = 0xa7;
150     //bit7,6.....10(カウントソースはf32(20MHzを 32分周))
151     //bit5........1(8bitタイマーとして使用)
152     //bit4........0(トリガ選択・ソフトウェアトリガ)
153     //bit3........どちらでも良い (ソフトウェアトリガ)
154     //bit2........1(PWMモードのときは 1)
155     //bit1,0.....11(PWMモード)
156
157     udf &= ~0x01;      //タイマ A0カウントダウンモード
158
159     ta0l = (unsigned char)RCS_ROUND_COUNT;  //タイマ A0の周期の設定
160
161     // タイマ割り込み初期化
162     ta1mr = 0x80;      // タイマモード クロック : 1/32
163     ta1 = cnt_ta1;      // タイマ値の初期化
164     ta1ic = 0x06;      // 割り込みレベルの設定
165     udf &= ~0x02;      //ダウンカウント設定
166
167
168     trgsr = 0x00;
169     onsf = 0x00;
170     //DCモータ駆動用 ta2
171     ta2ic =0x00;
172     ta2mr = 0x0f;
173     ta2 = 0;
174
175     tabsr = 0x06;      // A1,A2カウント開始
176     _asm( "\tFSET␣I");     // 割り込み許可
177
178
179     rcs_errcode = RCservo(0,-11);  //サーボを初期位置に
180
181     while(1){
182         switch(state){
183           case sWAIT:
184             ta2_tmp = 0x0000;
185             timer_count = 0;
186             LCD_locate(0,1,0,0);
187             LCD_print('W');
188             LCD_print('A');
189             LCD_print('I');
190             LCD_print('T');
191             LCD_print('␣');
192             break;
193
194           case sRUN:
195             ta2_tmp = ta2_max;
196
197         //ゲート数のカウント
198         //一度ゲートを認識したらそこから wait_time の間はゲート数を追加せず
199             if(((wait_start_time + wait_time) < timer_count) && (psdout < PSD_D)) {
200                 gate_count ++;
201                 wait_start_time = timer_count;
202             }
203             LCD_locate(0,1,0,0);
204             LCD_print('R');
205             LCD_print('U');
206             LCD_print('N');
207             LCD_print('␣');
208             LCD_print('␣');
209             break;
210
211           case sRUN2:
```

```
212             ta2_tmp = ta2_max;
213             LCD_locate(0,1,0,0);
214             LCD_print('R');
215             LCD_print('U');
216             LCD_print('N');
217             LCD_print('2');
218             LCD_print('␣');
219             break;
220
221           case sLOAD:
222             ta2_tmp = 0x0000;
223             LCD_locate(0,1,0,0);
224             LCD_print('L');
225             LCD_print('O');
226             LCD_print('A');
227             LCD_print('D');
228             LCD_print('␣');
229             break;
230
231           case sCARRY:    //sRUN とほぼ同じ
232             ta2_tmp = ta2_max;
233             if(((wait_start_time + wait_time) < timer_count) && (psdout < PSD_D)) {
234                 gate_count ++;
235                 wait_start_time = timer_count;
236                 if(gate_count == 5) { posi_start_time = timer_count; }
237             }
238             LCD_locate(0,1,0,0);
239             LCD_print('C');
240             LCD_print('A');
241             LCD_print('R');
242             LCD_print('R');
243             LCD_print('Y');
244             break;
245
246           case sRELEASE:
247             ta2_tmp = 0x0000;
248             LCD_locate(0,1,0,0);
249             LCD_print('R');
250             LCD_print('E');
251             LCD_print('L');
252             LCD_print('E');
253             LCD_print('S');
254             break;
255
256           case sRESET:    //もう一回ピンポン球を運ぶとき用
257             timer_count = 0;
258             gate_count = 1;
259             rcs_errcode = RCservo(0,-11);
260             LCD_locate(0,1,0,0);
261             LCD_print('R');
262             LCD_print('E');
263             LCD_print('S');
264             LCD_print('E');
265             LCD_print('T');
266             break;
267
268           case sSTOP:    //活動停止
269                 ta2_tmp = 0x0000;
270             break;
271
272           default:
273             LCD_locate(0,1,0,0);
274             LCD_print('E');
275             LCD_print('R');
276             LCD_print('R');
277             LCD_print('0');
278             LCD_print('1');
279             while(1){;}
280             break;
281         }
282
283         switch(state){    //遷移用   sw1 下で sWAIT に戻れる
284
285           case sWAIT:    //sw1 でスタート
286             if(p6_4 == 0) { state = sRUN; }
287             break;
288
289           case sRUN:    //立体交差感知で遷移
290             if(gate_count == 3) {
291                 state = sRUN2;
292                 posi_start_time = timer_count;
```

```
293             }
294             if(p6_4 == 1) {
295                 state = sWAIT;
296                 gate_count = 0;
297                 wait_start_time = 0;
298                 posi_start_time = 0;
299                 rcs_errcode = RCservo(0,-11);
300             }
301             break;
302 
303           case sRUN2:    //一定時間後に遷移
304             if((posi_start_time + posi2_time) < timer_count) {
305                 state = sLOAD;
306             }
307             if(p6_4 == 1) {
308                 state = sWAIT;
309                 gate_count = 0;
310                 wait_start_time = 0;
311                 posi_start_time = 0;
312                 rcs_errcode = RCservo(0,-11);
313             }
314             break;
315 
316           case sLOAD:    //スタート時から 1分 30秒後(50ms × 1800=90000ms=90s) にスタート
317             if(timer_count > 1800){ state = sCARRY; }
318             if(p6_4 == 1) {
319                 state = sWAIT;
320                 gate_count = 0;
321                 wait_start_time = 0;
322                 posi_start_time = 0;
323                 rcs_errcode = RCservo(0,-11);
324             }
325             break;
326 
327           case sCARRY:    //ゲート 1 認識から一定時間後に遷移
328             if((gate_count >= 5) && ((posi_start_time + posi1_time) < timer_count)) {
329                 state = sRELEASE;
330                 release_start_time = timer_count;
331                 rcs_errcode = RCservo(0,60);   //ピンポン球リリース
332             }
333             if(p6_4 == 1) {
334                 state = sWAIT;
335                 gate_count = 0;
336                 wait_start_time = 0;
337                 posi_start_time = 0;
338                 rcs_errcode = RCservo(0,-11);
339             }
340             break;
341 
342           case sRELEASE:   //一定時間後に遷移
343             if((release_start_time + release_time) < timer_count) {
344                 state = sRESET;
345             }
346             if(p6_4 == 1) {
347                 state = sWAIT;
348                 gate_count = 0;
349                 wait_start_time = 0;
350                 posi_start_time = 0;
351                 rcs_errcode = RCservo(0,-11);
352             }
353             break;
354 
355           case sRESET:    //繰り返したいなら sRUN へ
356             state = sSTOP;   //繰り返しはしないのでストップへ
357             if(p6_4 == 1) {
358                 state = sWAIT;
359                 gate_count = 0;
360                 wait_start_time = 0;
361                 posi_start_time = 0;
362                 rcs_errcode = RCservo(0,-11);
363             }
364             break;
365 
366           case sSTOP:
367             if(p6_4 == 1) {
368                 state = sWAIT;
369                 gate_count = 0;
370                 wait_start_time = 0;
371                 posi_start_time = 0;
372                 rcs_errcode = RCservo(0,-11);
373             }
```

```
374             break;
375
376           default:
377             LCD_locate(0,1,0,0);
378             LCD_print('E');
379             LCD_print('R');
380             LCD_print('R');
381             LCD_print('0');
382             LCD_print('2');
383             while(1){;}
384             break;
385         }
386     }
387 }
388
389 //=====================================================================
390 // 関数名 :ta1int()
391 // 機能   :ta1 割り込み関数
392 //  PSD センサ値->距離変換
393 //  LED 受信値：0~1023
394 //=====================================================================
395 void ta1int( void ){
396     timer_count ++;    //タイマのインクリメント
397
398     //AD 変換値を距離に
399     psdout = ((255*1024) / (5*AD0_read(5))) + 14;
400
401     //モータ駆動
402     p7_5 = 1;           //回転方向
403     if ( ta2_tmp >= 0xffff ) {ta2_tmp = 0xfffe;} // TA2 設定可能範囲に補正
404     ta2 = ta2_tmp;          // PWM 信号出力
405
406     //PSD 出力を LCD に表示
407     strcpy(LCD_buff, "P=");
408     itoa_cat(LCD_buff, psdout,4);
409     strcat(LCD_buff, "␣␣");
410
411     strcat(LCD_buff, "g=");
412     itoa_cat(LCD_buff, gate_count,2);
413     strcat(LCD_buff, "␣␣␣␣");
414
415     LCD_locate(0, 0, 0, 0);
416     LCD_print_str(LCD_buff);
417 }
418
419 //=====================================================================
420 // AD0 から電圧値を読み出す。
421 // 入力 :int average_count : この回数分だけ読み出して平均する。
422 // 出力 :AD 変換結果(float 型)0.0～1023.0
423 // 速度 : 1 回の読み出しで 3.3[usec]。
424 //=====================================================================
425 float AD0_read( int average_count )
426 {
427     float result = 0.0;     // AD 変換結果積算
428     int i;
429     if ( average_count < 1 ) { average_count = 1; }
430     if ( average_count > 100 ) { average_count = 100; }
431
432     adcon0 = 0x80;      // アナログ入力端子AN0 を選択
433     for ( i = 0; i < average_count; i++ ) {
434         adst = 1;      // AD 変換開始
435         while ( ir_adic == 0 ) {} // AD 変換終了を待つ(割り込み要求ビットをチェック)
436         ir_adic = 0;     // 割り込み要求ビットをクリア
437         result += ad0;     // AD 変換結果を積算
438     }
439     return ( result / (float)average_count );
440 }
441
442 //=====================================================================
443 // AD 変換の初期設定
444 //=====================================================================
445 void AD_init(void)
446 {
447     // 10bit 分解能/SH あり
448     // φAD = 10MHz
449     // 単発モード/ソフトウェアトリガ
450     // 変換時間 = 100nsec × 33 = 3.3usec
451
452     adst = 0; // AD 変換停止
453     adic = 0x00; // AD 割り込みレベル = 0(割り込みを使わない)
```

```
454
455     adcon0 = 0x80;
456     // bit2,1,0: 000: アナログ入力端子ANOを選択
457     // bit4,3: 00: アナログ入力モード＝単発モード
458     // bit5: 0: ソフトウエアトリガ選択
459     // bit6: 0: AD変換停止
460     // bit7: 1: AD変換動作周波数 fAD/2を選択
461
462     adcon1 = 0x28;
463     // bit1,0: 00: AD掃引端子 AN0,AN1 を選択(単発モード:無効)
464     // bit2: 0: AD動作モード選択繰り返し掃引モード1以外
465     // bit3: 1: 分解能選択 10ビット選択
466     // bit4: 0: AD変換動作周波数 fAD/2またはfAD/4を選択
467     // bit5: 1: Vref 接続：接続
468     // bit7,6: 00: ANEX0,ANEX1 は使用しない
469
470     adcon2 = 0x01; // bit0: 1: サンプル&ホールド有り
471 }
472
473 //使わない関数についても空の関数を記述する必要がある
474 void ta0int( void ) {}
475 void ta2int( void ) {}
476 void ta3int( void ) {}
477 void ta4int( void ) {}
478 void tb0int( void ) {}
479 void tb1int( void ) {}
480 void tb2int( void ) {}
481 void adint( void ) {}
482 void int0int( void ) {}
483 void int1int( void ) {}
484 void int3int( void ) {}
485 void int4int( void ) {}
486 void int5int( void ) {}
487 void uart0traint( void ){}
488 void uart0recint( void ){}
489 void uart1traint( void ){}
490 void uart1recint( void ){}
491 void uart2traint( void ){}
492 void uart2recint( void ){}
```

### 5.2.2 RCservo.h

```
1  #ifndef __RCSERVO_H__
2  #define __RCSERVO_H__
3
4  int RCservo(int port, int deg);
5
6  #define RCS_ROUND_COUNT ((15.0 * 625.0 / 255.0) - 1.0)
7  #define RCS_MAX_DEGREE (60.0)
8  #define RCS_MIN_DEGREE (-60.0)
9  #define RCS_F_32 (625000.0)
10 #define RCS_NEUTRAL (0.00152)
11 #define RCS_MAX_DEG_SEC (0.0006)
12
13 //エラーコード
14 #define RCS_PORT_ERROR 1 //ポートの設定が悪い
15 #define RCS_DEG_ERROR 2 //角度が指定範囲外
16
17
18
19 #endif
```

### 5.2.3 RCservo.c

```
1  /************************************
2  RCservo.c
3  RCサーボの角度を変える
4  2006年 11月 1日
5  @author:創造設計第二 2IA 張 辰樹
6  ************************************/
7  #include "RCservo.h"
8  #include "sfr26.h"      //CAKSmini 用定義ファイル
9
```

```
/*******************************************
int RCservo(int port, int deg)
機能:指定したport につながっている RC サーボの角度を
指定したdeg にする
引数:port:何番ポートにつながっているサーボに指令を出すか
deg :目標角度 [deg]
返り値:0:正常終了
それ以外:エラーコード
【注意！!】必ず初期化を行ってから使うこと
*******************************************/
int RCservo(int port, int deg)
{
    if (deg < RCS_MIN_DEGREE || deg > RCS_MAX_DEGREE) return RCS_DEG_ERROR;

    switch(port){
      case 0:
        tabsr &= ~0x01;  //いったんタイマーを停止

        ta0h = (1.52 + 0.01*deg) * 625 / (RCS_ROUND_COUNT + 1);
        //'H'幅の設定
        //周期 RCS_ROUND_COUNT, 角度deg で与えられている

        tabsr |= 0x01;  //タイマー再開
        break;

      default:
        return RCS_PORT_ERROR;
        break;
    }
    return 0;
}
```