モータの制御方法

プロペラの動きは直流モータとモータドライバーを用いることで実現している。

モータドライバーとはモータの駆動を制御するための素子であり、モータドライバの 1 ピンと 2 ピンにどの信号を入力したかによって表のようにモータの動作が決定される。

表　モータドライバのピンに入力する信号とその時のモータ動作

1 ピン	2 ピン	モータの動き
1	0	正転
0	1	逆転
1	1	ブレーキ
0	0	停止

モータドライバの 3 ピンと 4 ピンでも同様にもう一つのモータを制御している。

マイコン 2 つに書き込んだプログラム

どちらのプログラムにも LED 点灯を制御する関数、ボタン入力を受け付ける関数が入っており、メインの文章には押したボタンに応じてモータの回転を制御するプログラムがある。

マイコン1では1P のモータを回転させるプログラム、マイコン2では2P のモータを回転させるプログラムを書き込んでいる。

LED 点灯を制御する関数では LED(1p の点灯数 ,2p の点灯数); のように使用することでポート C の下位3ビットに1P の点灯数、4、5、6ビットに2P の点灯数を2進数で代入している。

また、ボタン入力を受け付ける関数では押したボタンに応じて関数からの戻り値が決まっており、UP=1,Stay=2,Down=3,入力なし=0 となっている。ボタンを押された瞬間に break;によって while ループから強制的に抜け出すようになっているため短押し、長押し、連打、同時押しと、どのような押し方をしたとしても最初に押したボタン(同時押しの場合は UP>Stay>Down の優先順)として処理される。また、_delay_ms(10);の後に while ループのカウント値を10ずつ増やしているため、1ループするごとに 10ms 経過するようになっている。そしてこのカウント値が t を超えた瞬間、すなわち t ms 経過すると自動的に while ループから抜け出すようになっており、タイムアウトの処理も実装されている。delay を10ms にした理由としては人間がボタンを押す時間が最短で 0.15 秒であるという点から下図のように 10ms のループで検知することができると判断したためである。

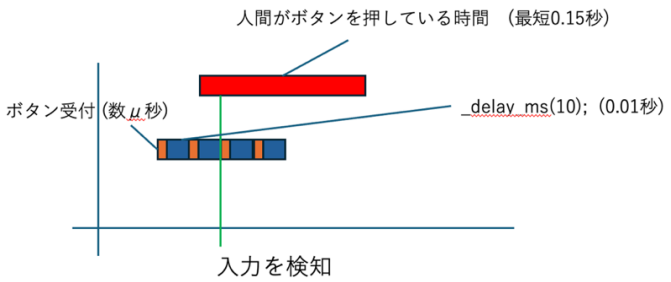
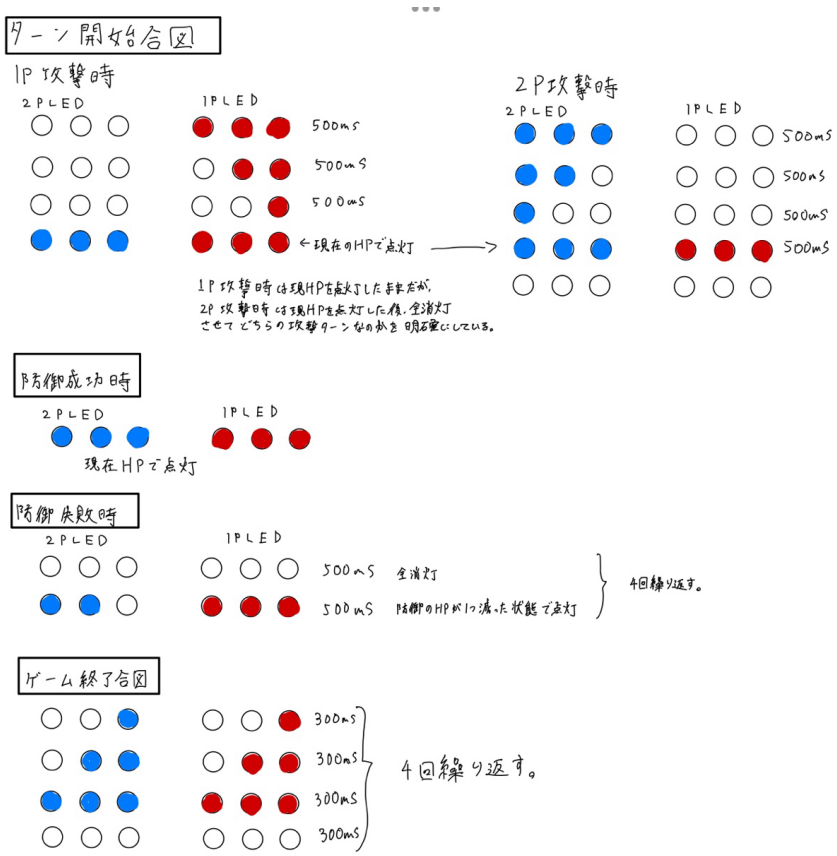


図　ボタン受付タイムアウト処理を実行するための考え方

以上の点から x=button1(3000);のように使用することで 3 秒間のみ1P のボタン入力を受け付け、その戻り値を x に代入することができる。
また、マイコン2では図 2 のように LED を接続していないため LED 関数は必要ないのだが、LED 関数を入れることによってマイコン1で LED を制御している間、マイコン2でも同じ動作をするようにし、2つのマイコン間で時間のずれが生じないようにしている。

LED の点灯パターン

LED の点灯を下のようにし、プレイヤーに何が起きているのか分かり易いようにした。



苦労した所

苦労した点は以下の6つの点であるが、それぞれ以下のようにして解決したことでこの制作物の魅力に繋げることができた。

① ボタン入力に受付時間を設ける

ボタン入力への受付時間はソフト部の図3で解説しているように短い delay のループと break を用いることで解決した。

② モータの動作を関数化すると上手く動作しないことへの対処

モータの動作を関数化しプログラムの簡略化を図ったが、正しく動作しなかった。その原因としてはPORTD=mortar(〇〇); のようにポート D の出力を関数の戻り値（2 ビット）にしていたが、内部的には mortar 関数からの戻り値を演算するところまでは完了するが、その後の PORTD に値を代入するよりも早く次の行に文章が移ってしまったため、回転させられなかったのだと考える。そこでPORTD=0b00000010;のように演算はさせず、代入だけ行わせることでプログラムの量は増えたものの、正しく動作させることができた。

③ ゲームのテンポ感の調節

ゲームのテンポ感は当初、2 人のプレイヤーがボタンを押し合い、LED で判定結果を表示した後、すぐに攻守が入れ替わるというアップテンポのものとなっていた。そこで、攻守交代の間に攻撃開始合図を挟むことによってテンポを抑え、プレイヤーがどちらの攻撃ターンなのかをわかりやすいようにした。一方で、攻守交代に十分な時間ができ、緊張感が弱まってしまったためボタン受付時間を導入することでゲーム自体の緊張感は保たれるように調整した。

④ プレイヤーに何が起きているのか納得させる

LED の点灯パターンをいくつも用意し、LED の様子からゲームフローチャートのどの部分を実行しているのかを明確にすることでプレイヤーがボタンを押すタイミングを間違えないように工夫した。自分の選択したボタンによって動作し、LED の点灯の仕方が変わったのだと納得感を得られるようにすることでゲームとして成り立たせることができた。

⑤ 2つのマイコンの連動

2つのマイコンの連動はボタンを並列に繋ぐことでプログラム動作開始のタイミングを揃えるようにし、途中でずれが生じないように極力同じプログラムを書き、必要な箇所のみ変えるようにした。

⑥　短押し、連打、長押し、同時押しどのような押し方でも正しく動作するようにする
この制作物は短押しによって遊ぶ設計となっているが、ボタン入力に受付時間が設けられているゲームであるため、プレイヤーのテンションによっては連打、長押し、同時押しのようにこちらの想定しない押し方をされることもあり得る。そこでボタンに UP>Stay>Down の優先度を設け、どんな押され方をしてもゲームが止まったり、予期せぬ動作をしないようにプログラムした。

考察

当初作成する予定のゲームではヘリコプターではなくロボットの動きを見て正しいボタンを判断するといったものであった。ロボットではボタンを押すと腕と腰の二箇所を回転させる必要があるため、今回のヘリコプターでのプログラムのモータ動作部分をモータ2つ分にしており、そのほかの LED やターン交代の部分は全く同じものとなっている。
しかし、以下 a～cの3つの問題が発生し、うまく動作させることができなかった。

a．モータが指定した時間通りに回転しない。
b．2つ以上のモータが正しく動作しない。
c．2 つ以上のモータの回転によって LED 点灯数が変化する。

a に関しては目的の角度にするために 200msだけ回転させる部分があり、

```
PORTD=0b00001010;　//0,1 ビットでモータ1、 2,3 ビットでモータ2　を回転
_delay_ms(200);
PORTD=0b111111111;　//2つのモータ停止
```

上記のようにプログラムしていたが、直流モータの始動に十分な時間だけ電圧がかかっていなかった可能性があり、回転させるものを重くすることで回転速度を遅くし、目的の角度に必要な時間を長くすることでモータの始動に十分な時間だけ電圧を加えられるようにすべきだったと考えられる。
b に関しては a のプログラムの回転時間を長くした状態でモータドライバにモータを1つ繋いだ場合は正しく回転するのに対し、モータを2つ接続した場合は同じプログラムであるにも関わらずモータが正しく回転しなかった。モータドライバの仕様によると、2つまでモータを同時に回転させることができるものとして間違いない。したがって可能性としては直流モータを回転させた際にノイズが発生しており、モーター一つ分のノイズでは影響がないが、モータを二つにした際はノイズが重なり合い、IC の誤作動を招いてしまったと考えられる。ブラシつきの DC モータではブラシと整流子の接触によって高周波の電圧ノイズ（数 kHz～数 MHz）が発生することがあるため、各モータとモータド

ライバ間にローパスフィルタを接続し、高周波成分をカットすることで、モータに印加される電圧を一定にするように試す必要があったと考えられる。
c に関してはポート C に LED を接続し、ポート D にモータを接続しているため、ポート D に信号を送り、ポート C には一切の信号を送っていない場面でも LED が消灯するといった現象は回路の接続ミス、または導線同士の接触以外考えられなかった。しかし、実際はモータが1つの時は LED もモータの動きも同じプログラムで正しく動作していたため、回路図は正しいといえる。そのため原因としては先ほどの b と同じように DC モータの回転によって発生する電圧ノイズが IC の誤作動を招いてしまったのだと考える。

以上の点を踏まえると、まず回転させるロボットを回転時間が 400ms 程度と十分な時間になるように重くし、各モータとモータドライバ間にはローパスフィルタを接続することでロボットを使用したゲームを実現できたと考えられる。あるいは、マイコンの数を六個に増やし、一つのマイコンにつき一つのモータを回転させることでコストは大幅に増えるものの、確実に動作すると考えられる。

ロボットによるゲームでは必殺技ボタンも導入することで反射神経だけでなく戦略性もあるゲームにできるようにプログラムを作成しており、それを実現することができなかったのは無念ではあるが、目標であるモータの動きから押すべきボタンを判断するというゲームは実現することができたため、最低限の成果は得られたと考えている。