PICマイコンによる ラジコンサーボ駆動プログラム

宮城障害者職業能力開発校　新妻 幹也

1.はじめに

「ものづくり」を主体とした訓練を行う現場の多くで、「ロボット」をテーマにした教材や、研究テーマを設定することが、めずらしくなくなっています。その中には、いわゆる「2足歩行ロボット」のような関節を動かすようなものもあります。

この場合、多くの関節を動かす必要が出てくるため、この部分にラジコンで昔からよく使われる「サーボ」を使うことが多くなってきました。

そこで、今回は、このサーボをPICマイコンを使って制御するための基本的な部分について述べたいと思います。その基本的な原理を理解すると、関節タイプのロボットを作成することが大変容易になりますので、参考にしていただければと思います。

2.サーボ機構

①サーボの種類

ラジコン用のサーボは、そのトルクや回転スピードなどによって多くの種類があります。メーカーもフタバ、KO、サンワ、JRの他、海外メーカーのものもあります。しかし、その制御信号はほとんど同じですから、ほぼ、どのメーカーのものでも、トルクやスピード、ケースの大きささえ考慮すれば、使えるはずです。

②ラジコン用サーボ駆動回路

ラジコンのサーボは、もともとラジコン飛行機やラジコンカーなどで、機械的に尾翼を動かしたり、エンジンのスロットルを動かしたり、ステアリングを動かしたりするために作られたものです。その特長は、送信機の各操作に比例したような位置で、サーボの回転位置も止まるということです。これは、大変便利なもので、車のステアリングなら、例えば、平行位置から、10度の角度で保つことも可能だということです。

この便利な機能を使って最近では、2足歩行ロボットの関節に多用されています。これまでは、ラジコンのサーボをこのような用途で使うとは、ラジコンの各メーカーでも想定はしていなかったと思います。

通常、サーボは直接ラジコンレシーバーに付けて使いますが、ロボットでは、レシーバーから出る制御信号と同様のものをマイコンで作ってやることによって動かすことができます。

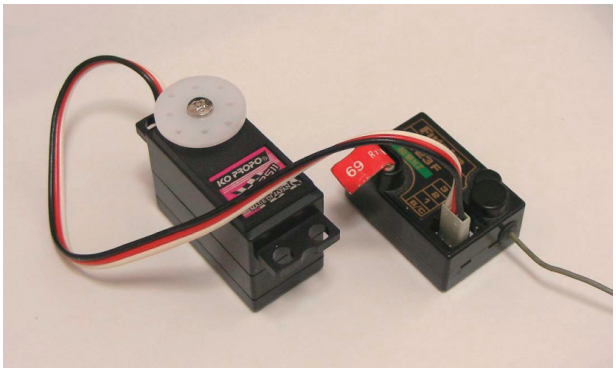

図1 サーボとレシーバー

サーボを動かすためには、前述したようにある一定の周期の矩形波信号を送ればよいということでした。その周期は、図2の解析のように14.26msec、周波数で言えば70Hzぐらいです。

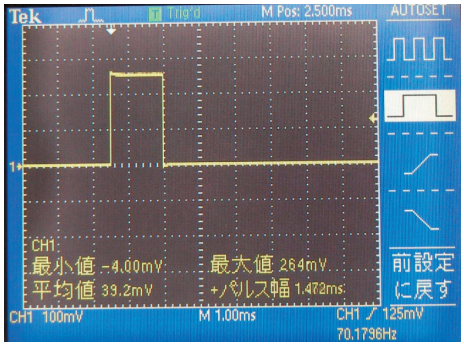

図2 ニュートラルのパルス幅は1.472msec

つまり、14.26ミリ秒で1周期（左エッジから次の左エッジまで）なので、その時間内で、ヤマ部分が何秒あるかによってサーボの角度が決まるということです。この図はニュートラの位置ですので、このヤマ部分が何秒あるかを読み取ります。原始的な方法では、オシロの画面の長さを測って計算する方法もありますが、デジタルオシロスコープでは、図のようにパルス幅も表示してくれます。

同様にして、トリガーの引きと押しのパルス幅を測定します。（トリガーの引きと押しは送信機側で変えることができるので、デューティーが長い方が引きとは限らない）

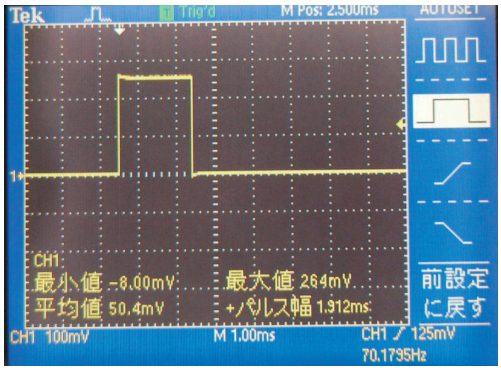

図3 トリガー引きのパルス幅は1.912msec

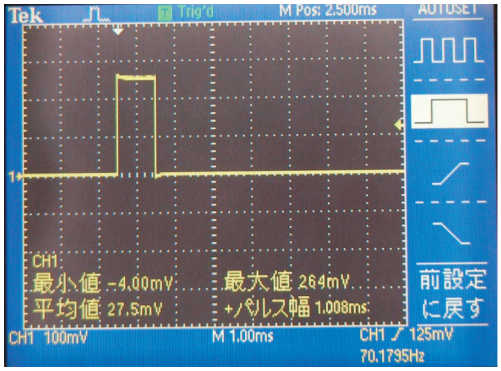

図4 トリガー押しのパルス幅は1.008msec

3.PICマイコンを使ったサーボ制御回路

PICマイコンを使ったサーボ制御回路は図6のとおりです。今回の回路では、8ピンタイプのPIC12F675を使ってみました。このPICは、類似するPIC12F629との違いとして、アナログの入力端子を持っている点があります。回路図にあるようにボリューム（可変抵抗）を使って、ボリュームを回したのと同じような量で、サーボ軸が動くような実験をするために、アナログ（A/Dコンバータ）ポート付きのPICを選択してみました。ですから、PIC16F819などのPICでも、同じように使うことができます。サーボの電源は、PIC用に5Vのレギュレータで作ったものをそのまま使っていますが、6Vまでは可能です。また、サーボはある程度電流を必要としますので、レギュレータの電流容量は500mA以上のものを使った方がよいでしょう。

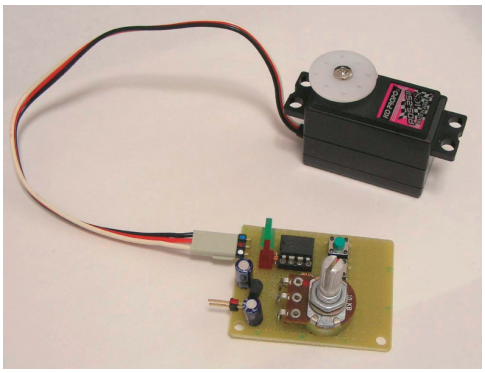

図5 サーボを同じ端子に接続

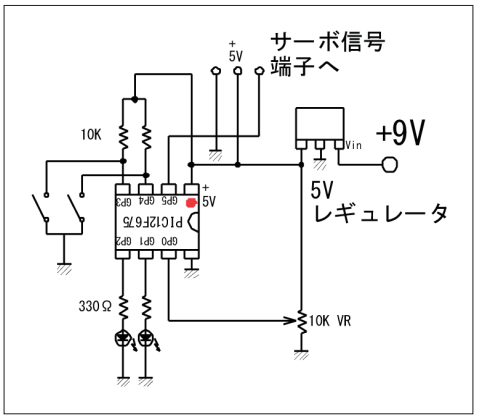

図6 サーボ制御回路図

では、この回路で、まず、スイッチ赤を押したときにサーボ軸が右に回転、スイッチ緑を押したときにサーボ軸が左回転するようにプログラムしてみます。

プログラムの注意点としては、時間待ちをする関数、delay__ms(*)と、delay_us(*)です。delay_ms(*)は、( )内に入れる数値によってそのミリ秒(1/1000秒)間は次へのステップに行きません。delay_us(*)は、( )内に入れる数値によってそのマイクロ秒(1/1000000秒)間は次へのステップに行きません。ただ、delay_us(*)の( )内に14260などと入れると、うまくいかなかったので、プログラム上では、ミリ秒部分とマイクロ秒部分に分けて記述しています。

```
//-----------------------------
// ラジコンサーボ制御実験プログラム
// Programed by Mikiya Niitsuma
//-----------------------------
#include <12F675.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,
NOMCLR,BROWNOUT
#use delay (clock=4000000)
#byte RA = 5
#bit  sig = RA.5 //サーボへの信号
#bit  LED_r= RA.1//赤LED
#bit  LED_g= RA.2//緑LED
#bit  sw_1 = RA.3//赤スイッチ
#bit  sw_2 = RA.4//緑スイッチ
void main()
 {
   //t1=1008,t2=1912,tn=1472,shuki=14250;
   set_tris_a(0x19);
//01,1001 GP1,GP2,GP5が出力ポート、他は入力に
設定
   RA = 0;
 while(1){
    while(!sw_1){//赤スイッチ
      sig=1;//1008
      delay_ms(1);
      delay_us(8);//delay_usはマイクロ秒
      sig=0;//13242
      delay_ms(13);
      delay_us(242);
    }
    while(!sw_2){//緑スイッチ
      sig=1;
      delay_ms(1);
      delay_us(912);
      sig=0;
      delay_ms(12);
      delay_us(338);
    }
    while(sw_1==1 && sw_2==1){ //プロポ操作が
ないときはニュートラルの位置
      sig=1;//1472
      delay_ms(1);
      delay_us(472);
      sig=0;//12778
      delay_ms(12);
      delay_us(778);
   }
 }
}
```

このプログラムでは、ボタンのON･OFFだけである特定の位置に軸が回転します。

次に、PIC12F675のA/Dコンバータ機能を使って、アナログ的にサーボをコントロールする実験もしてみましょう。実験回路に付けてある、ボリュームを動かして、その動きに応じたようにサーボが回転するようにしてみます。

考え方としては、PIC12F675のGP0ポートをアナログ入力に設定し、そこに加える電圧をボリュームで変化させ、その値を読み取り、パルス幅(1008から1912)を変化させるようにします。回路図は、何も変わらず、基板もそのまま使います。変えるのはプログラムだけです。

```
//----------------------------
// ボリュームでサーボを動かすプログラム
//      <×正常動作しない版×>
// programed by Mikiya Niitsuma
//----------------------------
#include <12F675.h>
#device ADC=10
#fuses INTRC_IO,NOWDT,NOPROTECT,NOMCL
R,BROWNOUT
#use delay (clock=4000000)
#byte RA=5
#bit  sig = RA.5 //サーボへの信号
#bit  Analog = RA.0//アナログポートからの信号
#bit  LED_r = RA.1//赤LED
#bit  LED_g = RA.2//緑LED
#bit  sw_1 = RA.3//赤スイッチ
#bit  sw_2 = RA.4//緑スイッチ
void main()
 {
   long int value,onT,offT,shuki=14250;
   byte i,v1,v2,v3,v4;
    set_tris_a(0x19);//01,1001 GP1,GP2,GP5が 出力
ポート、他は入力に設定
   setup_adc_ports(NO_ANALOGS);
   setup_adc_ports(AN0_ANALOG);//ポート0をア
ナログに設定
   setup_adc(ADC_CLOCK_INTERNAL);//ADCの
クロックを内部クロックに設定

   while(1){
       set_adc_channel(0); //ADCを読み込むピンを
指定
      delay_us(40);
      value = read_adc(); //読み込み
      onT=948+value;
      offT=shuki-onT;

      v1=onT/1000; v2=onT % 1000;
      v3=offT/1000;v4=offT % 1000;
      for(i=0;i<16;i++){
          sig=1;
          delay_ms(v1);
          delay_us(v2);
          sig=0;
          delay_ms(v3);
          delay_us(v4);
        }
    }
}
```

しかし、上記プログラムでは、正しい動作はしてくれません。まず、注意すべき点は、delay_ms() やdelay_us() などのカッコ内には、変数は使えてもその変数は1バイト型（PICCの場合は、intかbyte）しか使うことができないということです。直値では、delay_ms() は65535、delay_us() は、1000までは、指定できるのですが。

1バイト変数しか使えないということは、符号なしの数値でも、結果的には0から255までしか与えることができません。delay_ms() の方は255を超えない範囲なので問題ないのですが、delay_us() の方は、1～999までの数値の入る可能性があります。プログラムでは、v1～v4までを規定どおりbyte型でとって当てていますが、問題は残ったままです。では、v2,v4をlong型にして当てればいいのかというと、それもだめです。とにかく、（ ）内の変数は1バイト型と決まっているからです。このような問題に直面することはよくあることです。そんな場合の鉄則は、「別な記述で対応することはできないか？」を考えることです。自分で関数を作れば別ですが、用意されていない関数を使うことはできません。従って、用意されている関数は使うが、何らかの工夫をするということです。その工夫したプログラムが次のようなものです。

```
//----------------------------
// ボリュームでサーボを動かすプログラム
//  (正常動作版)
// programed by Mikiya Niitsuma
//----------------------------
```

```
#include <12F675.h>
#device ADC=10
#fuses INTRC_IO,NOWDT,NOPROTECT,NOMCLR,BROWNOUT
#use delay (clock=4000000)
#byte RA=5
#bit  sig = RA.5 //サーボへの信号
#bit  LED_r = RA.1//赤LED
#bit  LED_g = RA.2//緑LED
#bit  sw_1 = RA.3//赤スイッチ
#bit  sw_2 = RA.4//緑スイッチ
void main()
 {
   long int value,onT,offT,shuki=14000;//周波数が70Hzになるようにshukiを調整
  long int v2,v4;//2バイトの変数を追加
  byte i,j,v1,v3,iv2,iv4;
   set_tris_a(0x19);//01,1001 GP1,GP2,GP5が 出力ポート、他は入力に設定
  setup_adc_ports(NO_ANALOGS);
  setup_adc_ports(AN0_ANALOG);//ポート0をアナログに設定
   setup_adc(ADC_CLOCK_INTERNAL);//ADCのクロックを内部クロックに設定


  while(1){
      set_adc_channel(0); //ADCを読み込むピンを指定
     delay_us(40);
     value = read_adc(); //読み込み
     onT=948+value;
     offT=shuki-onT;


     v1=onT/1000; v2=onT % 1000;
     v3=offT/1000;v4=offT % 1000;
      iv2=v2/8;iv4=v4/8;//v2,v4は999を超えないので、8で割れば255以内
     for(i=0;i<8;i++){
       sig=1;
       delay_ms(v1);
       for(j=0;j<iv2;j++){
          delay_us(1);
       }
       sig=0;
       delay_ms(v3);
       for(j=0;j<iv4;j++){
          delay_us(1);
       }
     }
  }
}
```

このプログラムでは、delay_us(1);をforループで指定された回数回るというアイディアです。理論的には100回ループすれば、delay_us(100);に相当するはずです。これなら、ループ変数をlongでとれば、1000回ループしても問題はありません。

しかし、これはあくまで理論的な話で、現実はループをすることでのタイムロスは0ではありません。delay_us(1);を100回ループするに必要な時間は1/1万秒とはなりません。もっと多くの時間を要します。これは、マイコンのクロックにも依存します。今回はレゾネータを省略して4MHzで動作していますが、このクロックを速くすれば、タイムロスは減少します。オシロスコープで何回かプログラムのパラメータを変えて（変えたのはv2、v4を割り算する値の8）現実的な値を見つけ出しました。工夫したプログラムによって、若干のタイムロスがいくつか生じて、周波数も70Hzを下回ったので、shukiの値も少し変えました。これで、ほぼ、ボリュームを回したときと同じ感覚でサーボの軸位置が決まります。ボリュームを回してから、ほんのわずかのタイムラグがあってサーボが反応しますがこれは、for(i=0;i<8;i++){8の値のためです。これは、ボリュームで決まった電圧値によって、同一の波形を8回送るためものです。このループを入れないと、電圧チェック（ボリュームの位置チェック）の頻度が上がります。8の値を多くとれば、反応が遅くなります。減らせば、反応は速くなります。実験では、8の値を4にしても問題なく、むしろリアルタイムに動くようです。

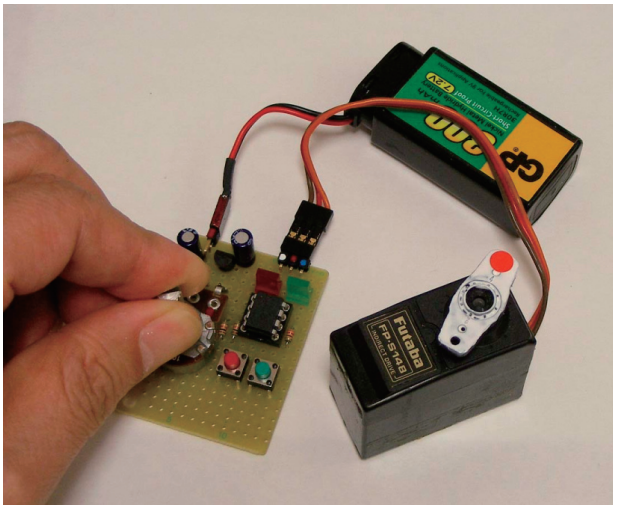

図7 ボリュームを右に回す

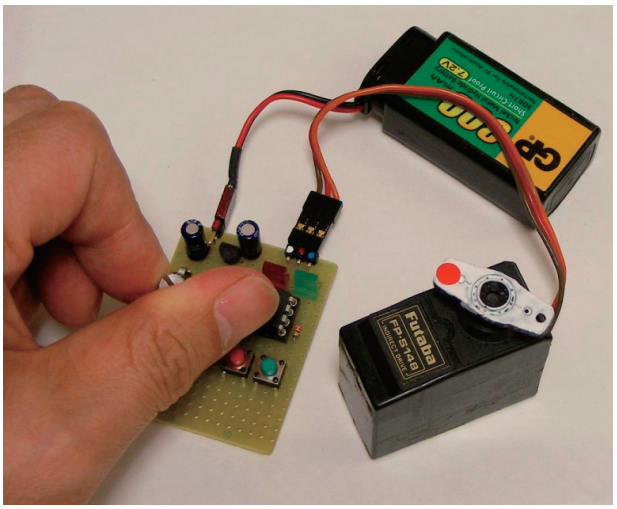

図8 ボリュームを左に回す

4.おわりに

本来ラジコン用のサーボはレシーバーに接続して、無線操縦するものですが、ロボットの分野でも、利用用途が広がってきており、マイコン制御によってさらに広範囲での利用が可能となります。電子回路は至極簡単なものであっても、プログラムによって、驚くほどいろいろなことができるということがおわかりいただけたと思います。今回の例では、マイコンのA/Dコンバータを使うことで、ボリュームによる制御をするためのプログラムを示しましたが、実際のロボットに応用するためには、もっとたくさんのサーボを使い、それをトータル的に制御することが求められます。しかし、その基本は今回示した簡単な回路や、プログラムリストにそのヒントは含まれています。マイコンのプログラムをマスターするということは、ロボットを作るうえで非常に重要なことなので、プログラムをして、1度や2度うまく動作しないからといってもあきらめずに、根気よく自分で考え、自分自らの手でプログラミングをしていく技術者の養成が求められます。それを実現するためにも、今回の記事が訓練カリキュラムにおける教材としての一助になれば幸いです。