



**ROBO DESIGNER**  
**Application book**

**TiColla CDE でラジコン用サーボを動かす**

## TiColla CDE でラジコン用サーボを動かす

C 言語はタイルに比べて処理が非常に高速です。

そこで、ソフトウェアで PWM 信号を作り、ラジコン用のサーボを駆動する実験をしてみます。

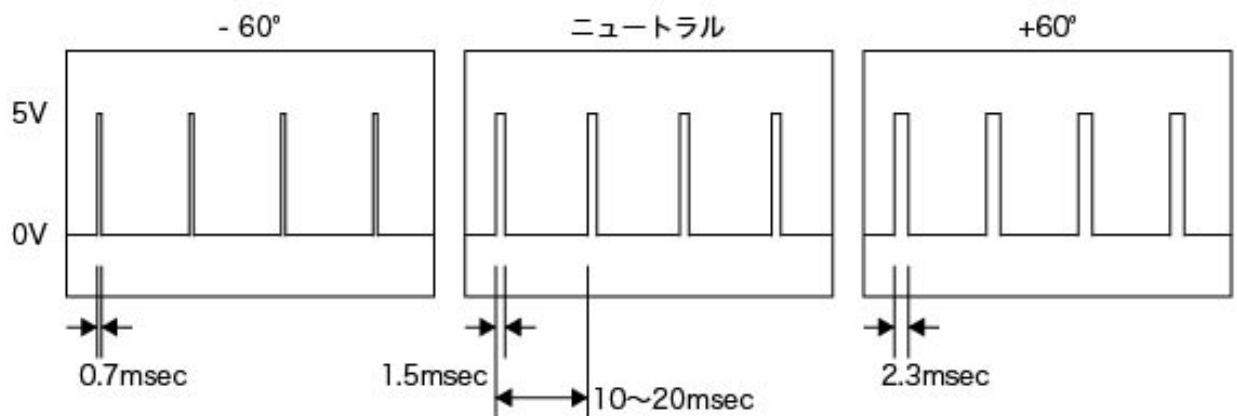


## サーボをコントロールするには？

サーボはラジコンカーやラジコン飛行機の操舵などに使われています。通常のギアボックスと違い、出力軸に連結された可変抵抗でフィードバック制御され、コネクタの信号ラインに入力された PWM 信号に応じて出力軸が一定の位置を保つようになっています。

PWM とは Pulse Width Modulation の略で、一定間隔の出力 on/off の出力 on の幅を変化させて調整用の信号としたり、擬似的に出力電圧を変化させる方法です (RoboDesigner でもモータースピードの調整に使用しています)。

サーボコントロール信号の場合、出力 on の間隔は 10 ~ 20msec、出力 on の幅はニュートラル位置が 1.5msec で、0.7 から 2.3msec 程度 (+60° くらい) の間で動かすことができます。

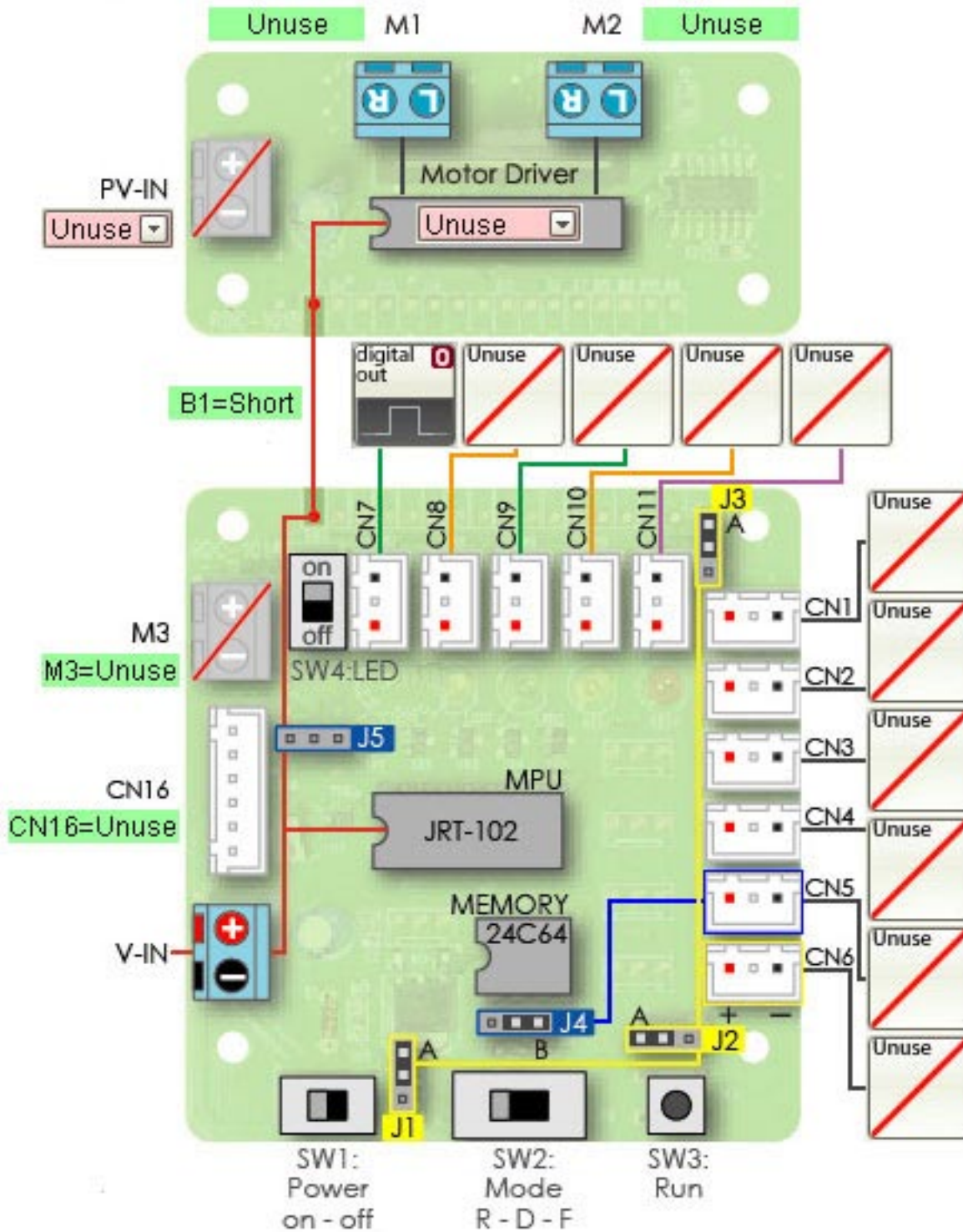




### プログラム (ループテスト)

一定の時間間隔の信号を作るため、まずループで LED(CN7) を点滅させ、おおまかにコントローラーボードの処理速度を測ります。

点灯と消灯のループ回数をそれぞれ変数にして、簡単に点灯の間隔を調整できるようにしておきます。実際に光っている時間を計測し、ループ回数で割れば、処理速度の目安がわかります。



## main.c の変更部分

```

/*-----*/
/* Variable declaration */
/*-----*/

#define LED    CN7    //LED-CN matching

dword i;

/*-----*/
/* Prototype declaration */
/*-----*/

/*-----*/
/* main */
/*-----*/

void main(void) {

    init();          //Initialization

    while(1){
        LED=1; //LED on
        for(i=0;i<=1000000;i++){
            COPCTL = 0;
        }
        LED=0; //LED off
        for(i=0;i<=1000000;i++){
            COPCTL = 0;
        }

        COPCTL = 0;
    }
}

```



## サーボとコントローラーボードを接続する

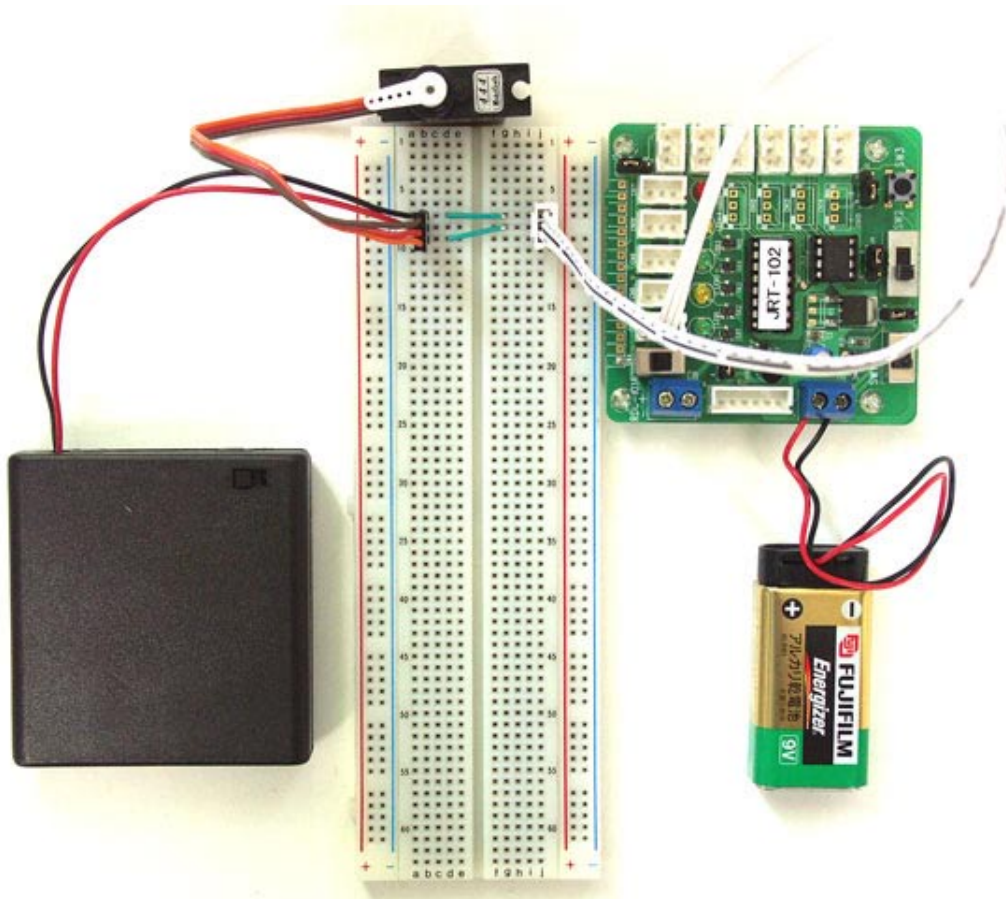
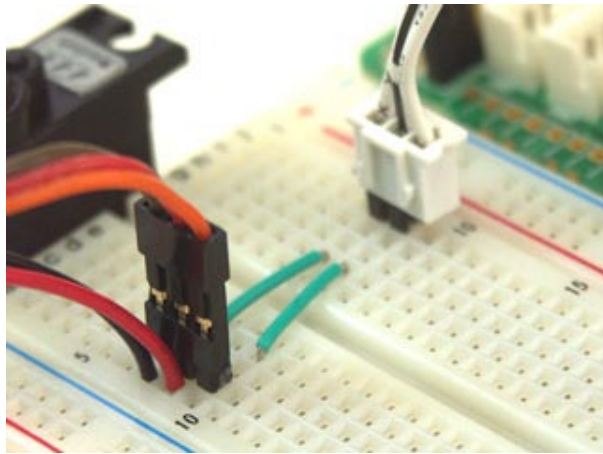
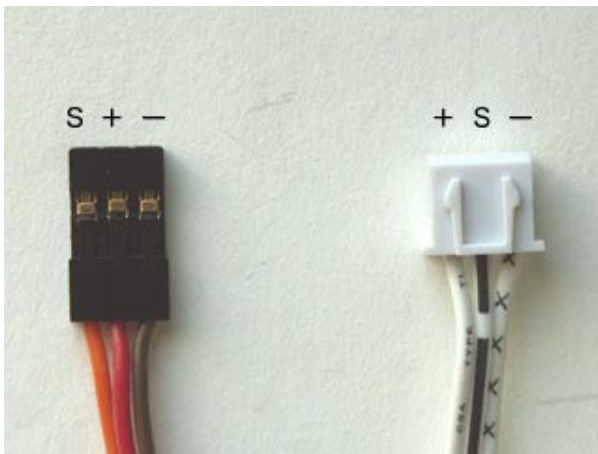
サーボのコネクターは電源 (+)、グランド (-)、信号 (S) の 3 線です。しかし、コントローラーボードのコネクターとは配列が違うので、ブレッドボードで変換します (ユニバーサル基板で変換用のボードを作成してもよいでしょう)。

サーボの電源電圧は 4.8 ~ 6V 程度です。ラジコンでは単三電池 4 本またはニッカド 5 本程度のバッテリーで駆動します。

実験ではコントローラーボードのコネクターの 5V を使用します。

ケーブルの色はサーボのメーカー / 機種によって異なります。

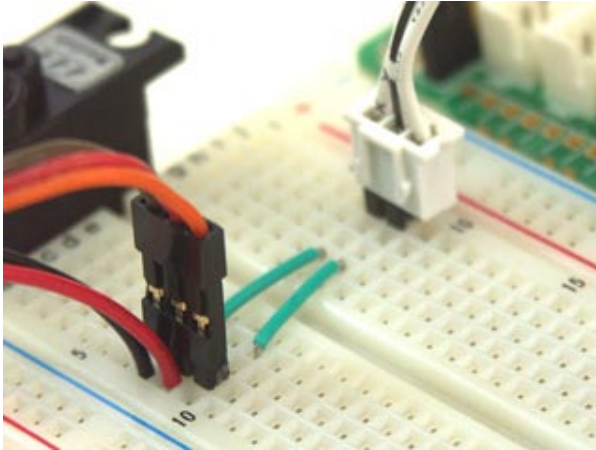
※写真のコントローラーボードはモータードライバー部分を切断してあります。



## [2 電源の場合]

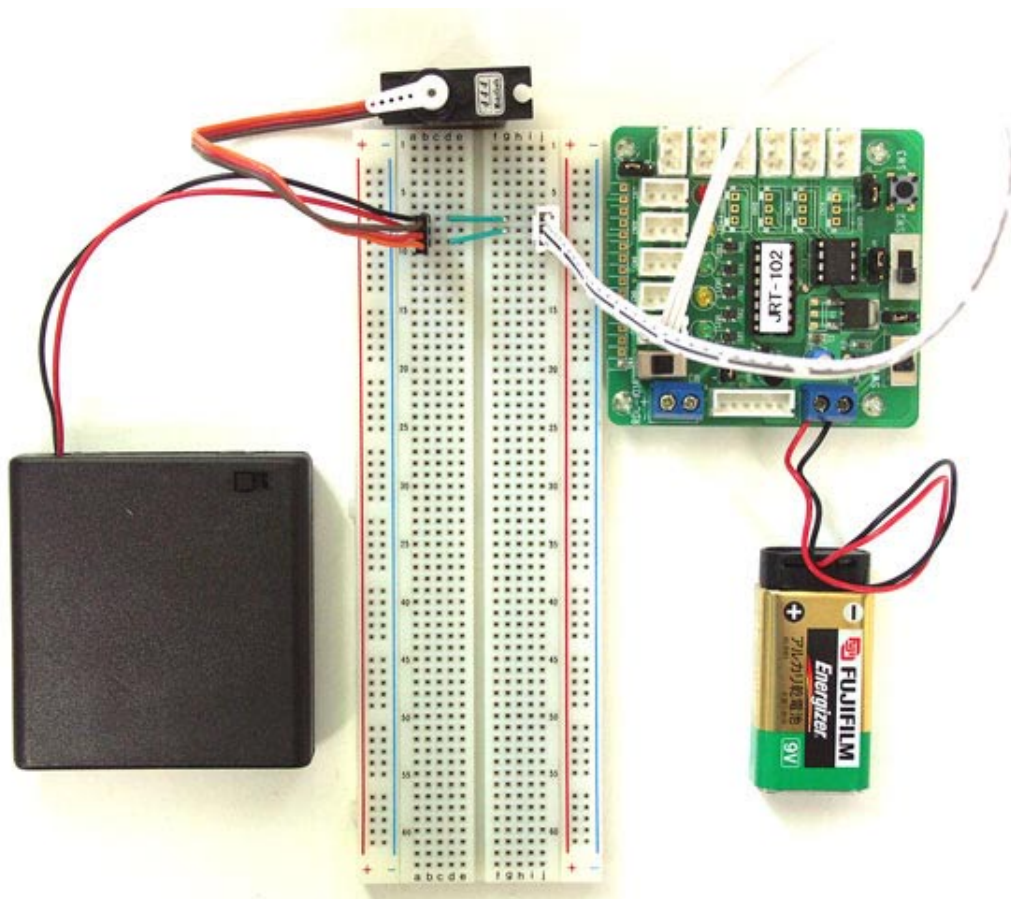
サーボを複数つなげるなど消費電流が増える場合にはコントローラーボードとサーボそれぞれに電源を用意します。

サーボとの接続は信号とグランド (-) だけにします。ヘッダーピンを工夫するか、必要のないケーブルをカットしてください。



コントローラーボードは消費電流が少ないので、006P 電池などで十分です。

サーボは電源と接続しておくとも常に電流を消費しますので、電源スイッチがあると便利です。





## プログラム（サーボ駆動）

ループで LED を点滅させるプログラムをベースに、出力 on の幅 (= サーボの角度) を設定する変数と、その位置を保持する時間を引数とする関数「servo」を定義します。

しかし、一つの位置を指定しただけでは正しく動いているのかわかりませんので、main 関数で関数 servo を呼び出し、2 つの位置で交互に止まるようにします。

ハードウェア設定はループテストと同じです。

※サーボの特性は機種ごとに異なりますので、詳しくはご自分で使用されるサーボのスペックシートを参照してください。

※サーボが電源に接続されていると電力を消費し続けるので、ボードとの接続ケーブルをこまめに抜いてください。

※サーボは慎重に扱わないと、内部のギアが破損する場合があります。また、今回の実験で使用しているような小さなタイプでは底部がネジで固定されていないので、取り扱いに気をつけてください。

※プログラムの PWM の設定に注意してください。無理な位置まで動かそうとすると破損する場合があります。

## main.c の変更部分

```

/*-----*/
/* Variable declaration */
/*-----*/

//pulse to pulse:10-20msec/pulse width:0.7-2.3msec center=1.5msec
#define T_BASE 100 //base for motion interval adjustment
#define PWM_BASE 9 //base for PWM adjustment

#define SERVO_1 CN7 //servo-CN matching

/*-----*/
/* Prototype declaration */
/*-----*/

void servo(byte deg, byte motion_time);

/*-----*/
/* main */
/*-----*/

void main(void) {

    init(); //Initialization

    while(1){
        servo(14,1);
        servo(16,2);

        COPCTL = 0;
    }
}

/*-----*/
/* function */
/*-----*/

void servo(byte deg, byte motion_time){
    word i;
    word m;

    motion_time*=T_BASE;
    for(m=0;m<=motion_time;m++){
        SERVO_1=1;
        for(i=0;i<=PWM_BASE*deg;i++){
            COPCTL = 0;
        }
        SERVO_1=0;
        for(i=0;i<=PWM_BASE*120;i++){
            COPCTL = 0;
        }
        COPCTL = 0;
    }
}

```

