モーター速度(回転数)制御

概要

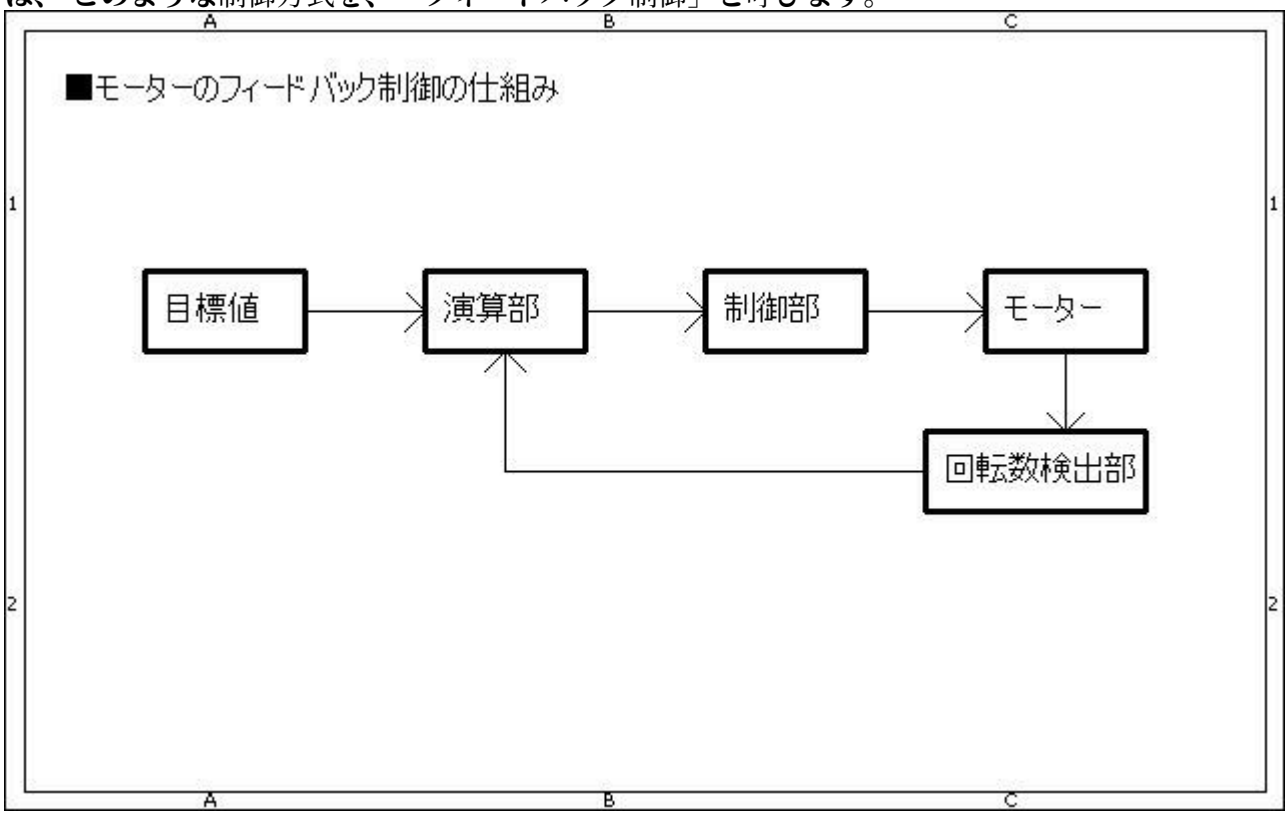
モーターの回転数の測定は、モーター回転数測定(周期測定の応用)で製作しましたので、今回は、回転数を 制御する機能を付加してみました。

<仕様>

- アップ(UP)スイッチで、1000rpm単位で速度(回転)を上げることができる。
- ダウン(DOWN)スイッチで、1000rpm単位で速度(回転)を下げることができる。
- 設定範囲は、1000rpm~10000rpm迄とする。

動作原理

次の図のように、目標値と回転数(検出)を比較(演算)し、モーターの速度(回転)を制御します。一般的には、 このような制御方式を、「フィードバック制御」と呼びます。



◎目標値 プッシュスイッチで、1000rpm単位で、アップ(UP)またはダウン(DOWN)させます。

◎演算部 目標値と回転数を比較し、制御部へ指示を出します。

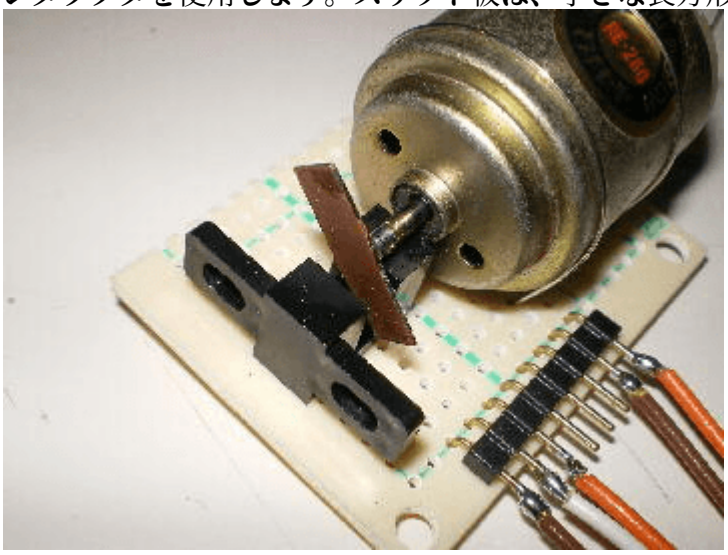
◎制御部 PWM(Pulse Width Modulation)方式を採用し、1024段階の速度(回転)制御を可能とします。

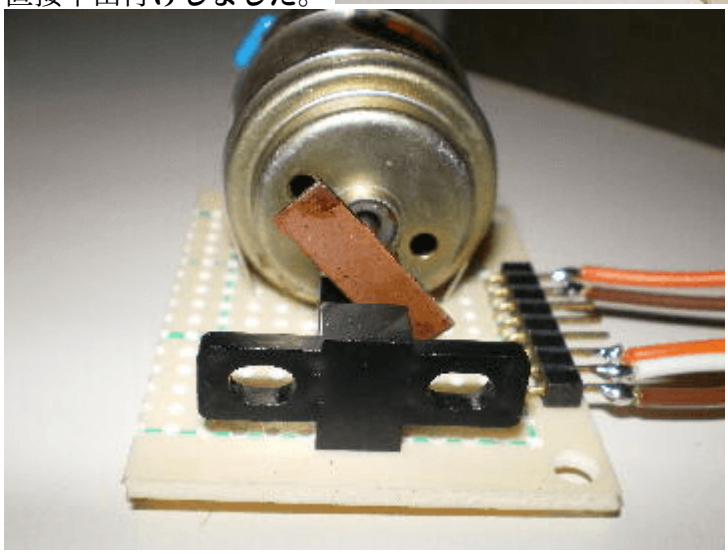
◎モーター 手持ちの、「マブチモータ□RE260□を使用します。

- 限界電圧:1.5~3.0V
- 適正電圧:3.0V

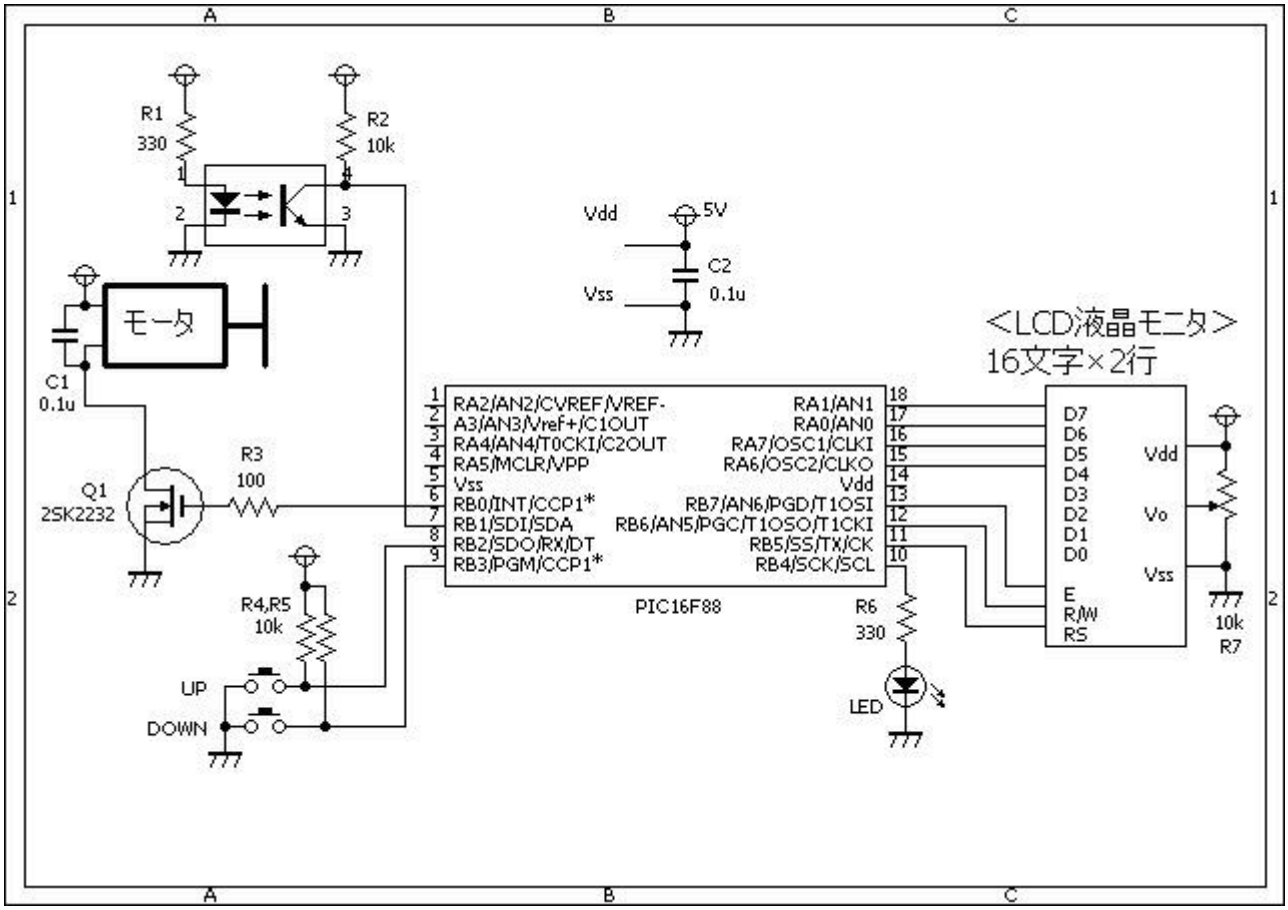
- トルク:10.0g/cm
- 負荷時の回転数:8,900rpm
- 負荷時の消費電流:700mA
- シャフト径:2.0mm
- 重量;30g
- 外形寸法:26.9×23.8mm

◎回転数検出部 フォトインタラプタを使用します。スリット板は、小さな長方形の銅板をモーター軸に直接半田付けしました。





回路図



ソースコード

MotorControl.c

```
//*****************************************************************************
*

#define       TA_SIG        PORTB.F1
#define       CYCLE_DATA    2.0             //エンジン1回転あたりのパルス数

#define       ON            0
#define       OFF            1

#define       STS_HIGH    1
#define       STS_LOW        0

#define       LED             PORTB.F4
#define       SW_UP         PORTB.F2
#define       SW_DOWN         PORTB.F3

//*****************************************************************************
*

static unsigned    short    interrupt_cnt;
```

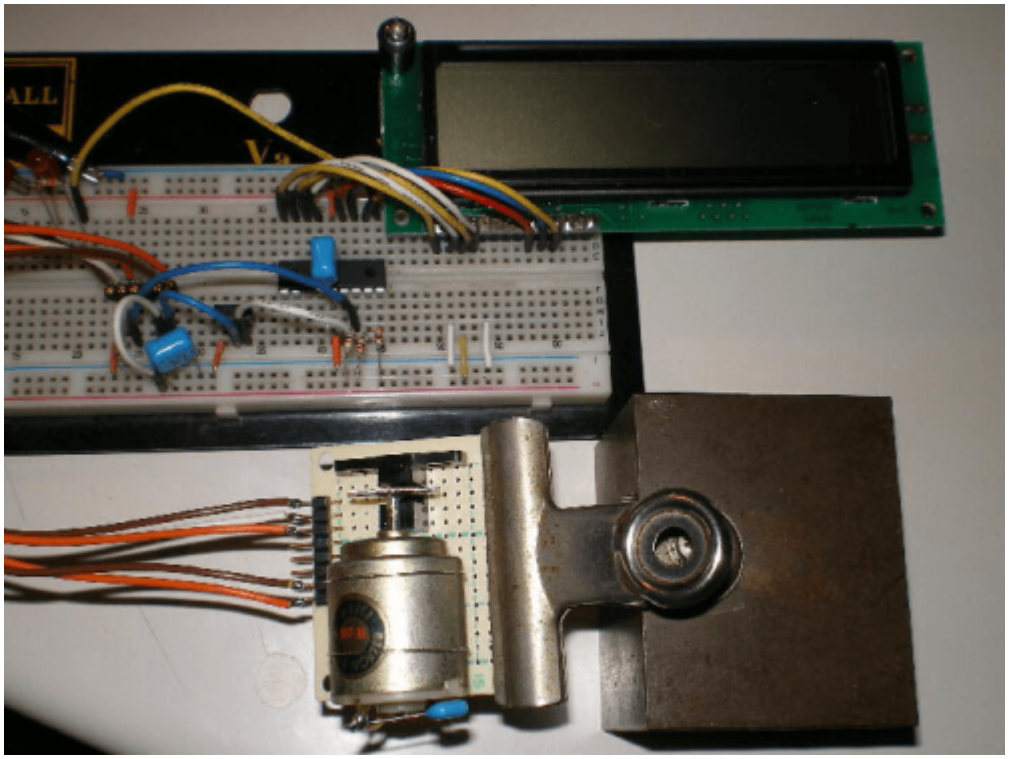
```
void   interrupt()
{
    if (INTCON.T0IF == 1) {
        INTCON.T0IF = 0;
        //
        LED = ~LED;
        //
        if (interrupt_cnt > 0)
            interrupt_cnt--;
    }
}

//*********************************************************************
*

long   measurement()
{
    static    unsigned    int    cnt;
    // TIMER1の設定
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    T1CON.T1RUN = 0;
    T1CON.T1CKPS1 = 0;
    T1CON.T1CKPS0 = 1;
    T1CON.T1OSCEN = 0;
    T1CON.NOT_T1SYNC = 1;
    T1CON.TMR1CS = 0;
    T1CON.TMR1ON = 0;
    TMR1L = 0;
    TMR1H = 0;
    //信号の立ち上がりをチェックする。
    interrupt_cnt = 10;
    while (TA_SIG == STS_HIGH) {
        if (interrupt_cnt == 0)
            return (0);
    }
    interrupt_cnt = 10;
    while (TA_SIG == STS_LOW) {
        if (interrupt_cnt == 0)
            return (0);
    }
    //TIMER1を開始する。
    T1CON.TMR1ON = 1;
    //信号の立ち上がりをチェックする。
    interrupt_cnt = 10;
    while (TA_SIG == STS_HIGH) {
        if (interrupt_cnt == 0)
            return (0);
    }
    interrupt_cnt = 10;
```

```
    while (TA_SIG == STS_LOW) {
        if (interrupt_cnt == 0)
            return (0);
    }
    //TIMER1を停止する。
    T1CON.TMR1ON = 0;
    //オーバーフローをチェックする。
    if (PIR1.TMR1IF == 1)
        return (-1);
    //実時間に変換する。
    cnt = TMR1H << 8;
    cnt = cnt | TMR1L;
    return (cnt);    // 4.0usec=(1/8000000)*4*8*1000000
}

//*********************************************************************
*

void   Pwm_Change_DutyEx(unsigned int duty_ratio)
{
    CCPR1L = duty_ratio >> 2;
    CCP1CON.F6 = duty_ratio & 0b00000001;
    CCP1CON.F7 = (duty_ratio & 0b00000010) >> 1;
}

//*********************************************************************
*

void   main()
{
    static    unsigned    char    buf[16], cnt;
    static    unsigned    long    cycleTime, rpm, targetRpm, tmp;
    static    unsigned    int        duty;
    //
    OSCCON = 0b01110000;          //クロックは8MHz
    CMCON  = 0b00000111;          //コンパレータは使用しない。
    ANSEL  = 0b00000000;          // A/D変換は使用しない。
    TRISA  = 0b00111100;
    TRISB  = 0b00001110;
    OPTION_REG.F7 = 0;              // PORTBをプルアップする。
    //
    duty = 512;
    interrupt_cnt = 0;
    targetRpm = 8000;
    // TIMER0の設定
    INTCON.T0IE = 1;
    INTCON.T0IF = 0;
    OPTION_REG.T0CS = 0;
    OPTION_REG.PSA = 0;
    OPTION_REG.PS2 = 1;
    OPTION_REG.PS1 = 1;
```

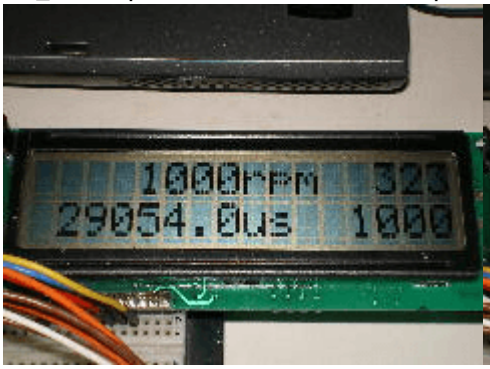
```
    OPTION_REG.PS0 = 1;
    TMR0 = 0;
    //
    Lcd_Custom_Config(&PORTA, 1, 0, 7, 6, &PORTB, 5, 6, 7);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Cmd(LCD_CLEAR);
    Lcd_Custom_Out(1, 9, "rpm");
    Lcd_Custom_Out(2, 9, "us");
    //
    Pwm_Init(10000);
    PR2 = 0xFF;
    Pwm_Change_DutyEx(duty);
    Pwm_Start();
    //割り込みを許可する。
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    while (1) {
        //パルス間隔の時間を測定する。
        cycleTime = 0;
        for (cnt = 0; cnt < 100; cnt++) {
            tmp = measurement();
            if (tmp == 0) {
                Lcd_Custom_Out(1, 1, "error! 1");
                cycleTime = 0;
                break;
            }
            if (tmp == -1) {    //オーバーフローでいればエラー表示する。
                Lcd_Custom_Out(1, 1, "error! 2");
                cycleTime = 0;
                break;
            }
            cycleTime += tmp;
        }
        if (cycleTime == 0)
            continue;
        cycleTime = cycleTime / 10;
        //1分間の回転数を求め表示する。
        rpm = (600000000 / cycleTime) / CYCLE_DATA;
        tmp = rpm / 100;
        if ((rpm - (tmp * 100)) >= 50) {
            tmp++;
        }
        tmp = tmp * 100;
        LongToStr(tmp, buf);
        Lcd_Custom_Out(1, 1, &buf[3]);
        //パルスの間隔を求め表示する。
        LongToStr(cycleTime, buf);
        buf[12] = 0x00;
        buf[11] = buf[10];
```

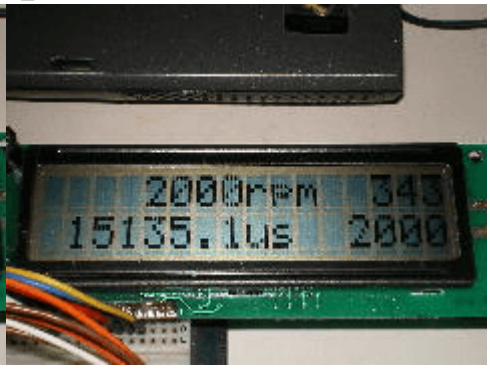
```
        buf[10] = '.';
        Lcd_Custom_Out(2, 1, &buf[4]);
        //□□□の□□□□を表示する。
        WordToStr(duty, buf);
        Lcd_Custom_Out(1, 13, &buf[1]);
        //□目標値を表示する。
        WordToStr(targetRpm, buf);
        Lcd_Custom_Out(2, 12, buf);
        //速度の制御を行う。
        if (rpm > targetRpm) {
            duty--;
        } else {
            duty++;
        }
        Pwm_Change_DutyEx(duty);
        //
        if (SW_UP == ON) {
            targetRpm += 1000;
        }
        if (SW_DOWN == ON) {
            targetRpm -= 1000;
        }
    }
}

//*****************************************************************************
*
```

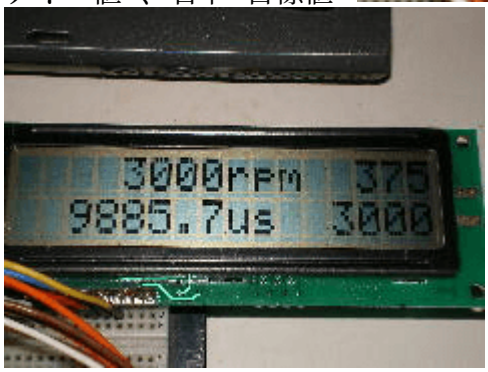
動作確認

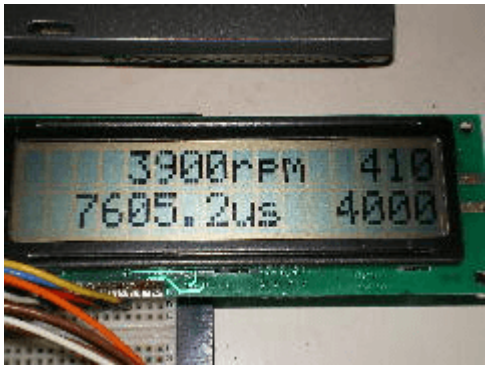


左側から□1000rpm□2000rpm□3000rpm表示内容は、左上<rpm>□左下<パルス幅>、右上<PWMのデューティー値>、右下<目標値>

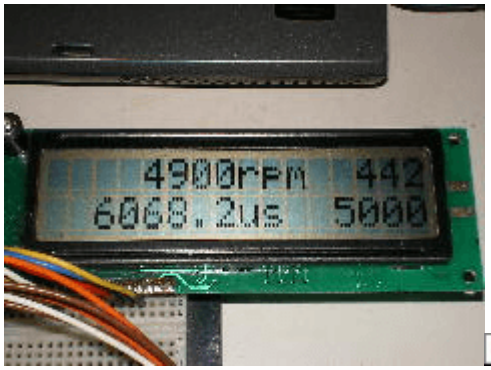








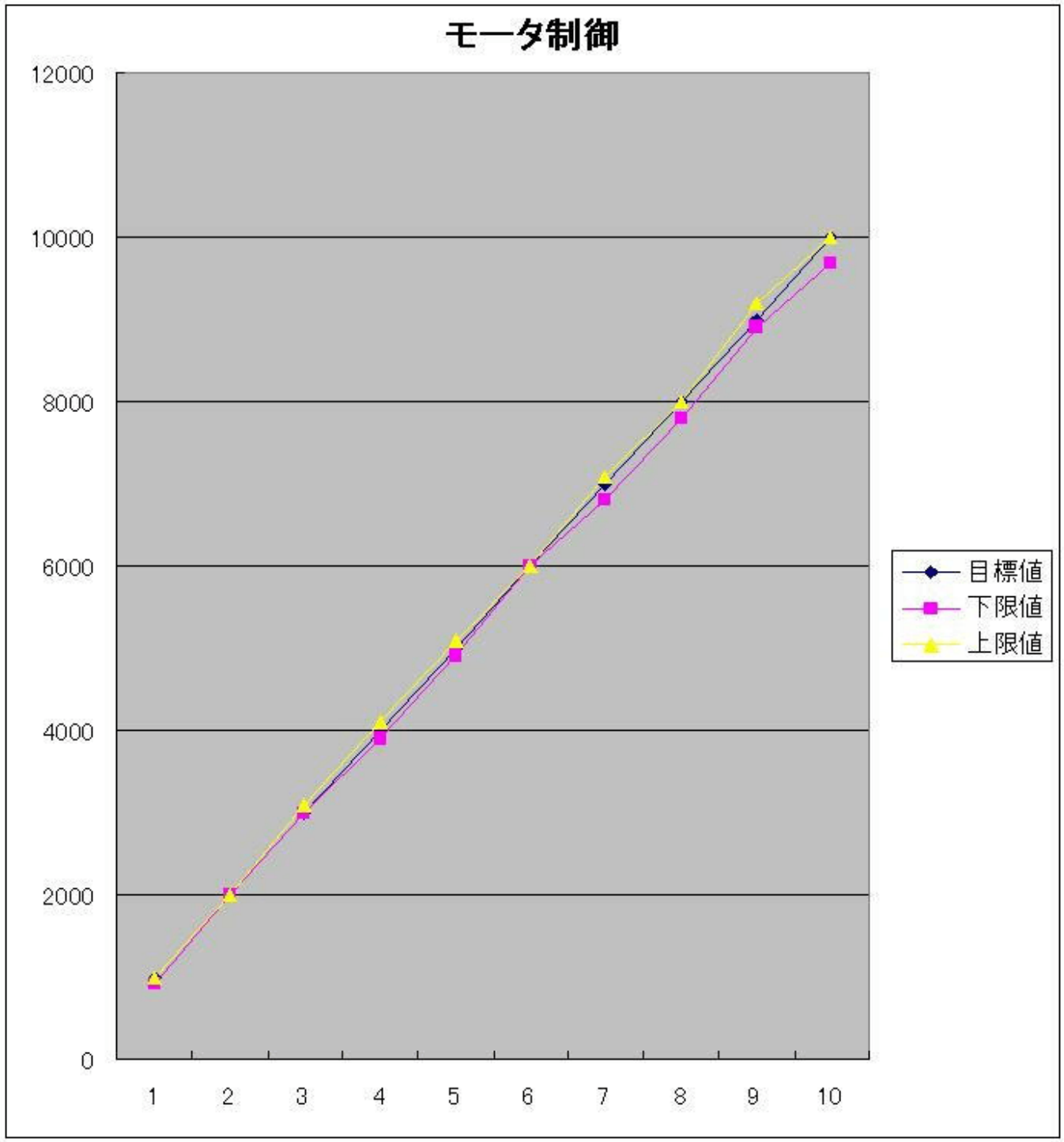
左側から□4000rpm□5000rpm□6000rpm



左側から7000rpm、8000rpm、9000rpm

10000rpm

目標値と実際の速度の下限値と上限値をグラフにしてみました。



その時の、PWMのデューティー値をグラフにしてみました。 今回、使用した「マブチモータ□RE260□の回転数は、8900rpmなので□10000rpmでは、デューティー値が極端に 高くなっています□2000rpm~8000rpmが、実用的な範囲だと考えられます。