

# モーター回転数測定(周期測定の実用)

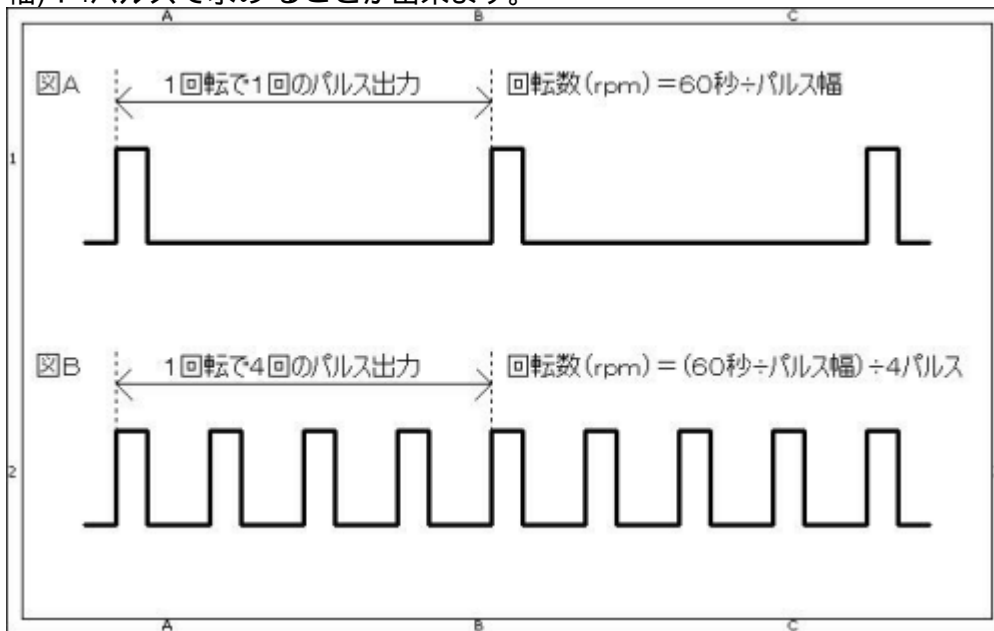
## 概要

ロボコン等にも興味がありますので、必須技術となるモーター制御の一環として、モーターの回転数を測定してみました。

## 動作原理

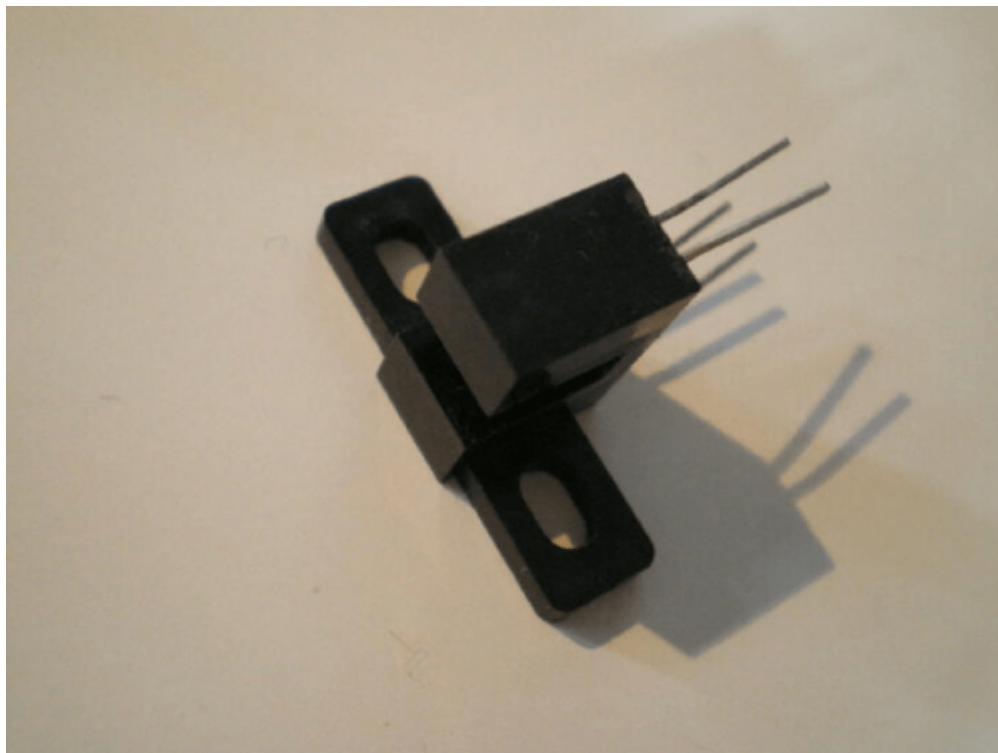
モーターの回転軸にスリット円板を取り付け、それをフォトインタラプタに通すことにより、機械的な回転変位を、電気的なパルスに変換し、そのパルス幅より、回転数rpm(revolution per minute)を求めます。

図Aのように、1回転で1回のパルスが出力される場合の回転数rpmは、 $60\text{秒} \div \text{パルス幅}$ で求めることができます。図Bのように、1回転で4回のパルスが出力される場合の回転数rpmは、 $(60\text{秒} \div \text{パルス幅}) \div 4\text{パルス}$ で求めることができます。

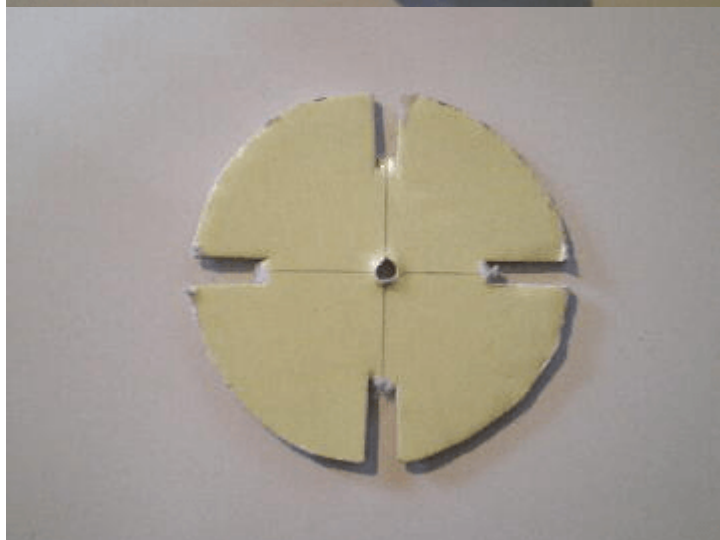


パルスの幅を求める方法は、パルス周期測定ユニットや周期&周波数測定ユニットV2(1Hz~100Hz)を参照してください。尚、測定レンジは、4段階に自動で切り替え、出来るだけ精度を高めるように工夫してあります。

フォトインタラプタは、手持ちの物を使用しました。(実は、型番が良く分かりません)

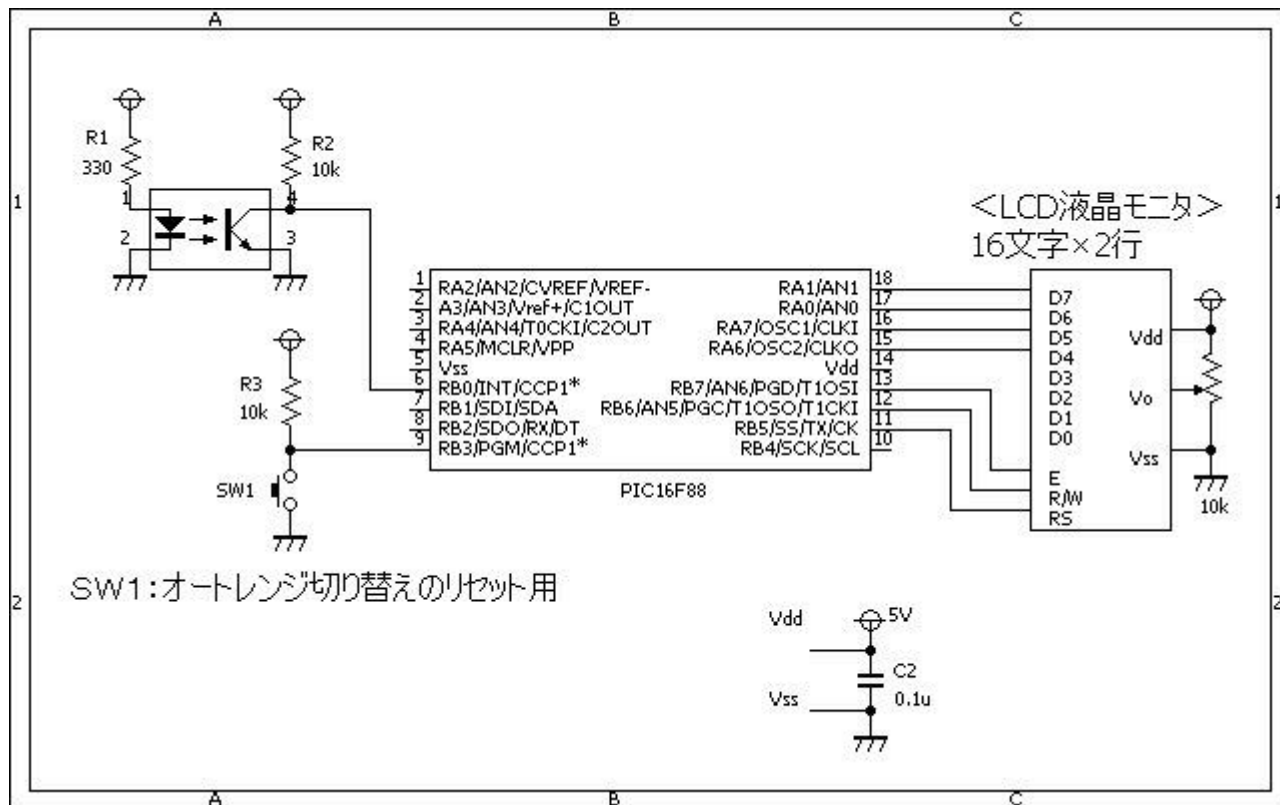


測定の対象としたモーターと厚紙で作ったスリット円板です。





## 回路図



## ソースコード

### RotationMeasure.c

```
//*****
*
#define      CLOCK2MHZ

#ifdef      CLOCK2MHZ          // min1Hz
#define      PS1          2.00    // 2.00 = (1 / 2000000Hz) * 4 * 1 *
1000000 -> ≈8Hz□
#define      PS2          4.00    // 4.00 = (1 / 2000000Hz) * 4 * 2 *
1000000 -> ≈4Hz□
#define      PS4          8.00    // 8.00 = (1 / 2000000Hz) * 4 * 4 *
1000000 -> ≈2Hz□
#define      PS8          16.00   // 16.00 = (1 / 2000000Hz) * 4 * 8 *
1000000 -> ≈1Hz□
/*
□□□□の時の精度□PS9の時) 小数点第4桁
□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□□
□□□□の時の精度□PS1の時) 小数点第3桁
□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□□
```

```

□□□□の時の精度□PS1の時) 小数点第2桁
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□の時の精度□PS1の時) 小数点第1桁
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□の時の精度□PS1の時) 小数点第0桁
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□の時の精度□PS1の時) 小数点第0桁
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
*/
#endif

#ifdef      CLOCK16MHZ      // min8Hz
#define      PS1            0.25      // 0.25 = (1 / 16000000Hz) * 4 * 1 *
1000000 -> ≈62Hz□
#define      PS2            0.50      // 0.50 = (1 / 16000000Hz) * 4 * 2 *
1000000 -> ≈31Hz□
#define      PS4            1.00      // 1.00 = (1 / 16000000Hz) * 4 * 4 *
1000000 -> ≈16Hz□
#define      PS8            2.00      // 2.00 = (1 / 16000000Hz) * 4 * 8 *
1000000 -> ≈8Hz□
/*
最高精度□PS1の時) 小数点第3桁
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□の時の精度□PS8の時) 小数点第3桁
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□の時の精度□PS1の時) 小数点第2桁
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□の時の精度□PS1の時) 小数点第1桁
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□の時の精度□PS1の時) 小数点第0桁
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
*/
#endif

```

```
//*****
*
unsigned int measurement()
{
    unsigned int dat;
    //
    T1CON.TMR10N = 0; // タイマーオフ
    TMR1H = 0; // タイマー値クリア
    TMR1L = 0; // タイマー値クリア
    PIR1.CCP1IF = 0; // キャプチャフラグクリア
    asm {
loop_001: // キャプチャフラグ確認
(アセンブラで高速処理)
        btfss PIR1, 2
        goto loop_001
    }
// while (PIR1.CCP1IF == 0) // キャプチャフラグ確認
// ;
    T1CON.TMR10N = 1; // タイマーオン
    PIR1.CCP1IF = 0; // キャプチャフラグクリア
    asm {
loop_002: // キャプチャフラグ確認
(アセンブラで高速処理)
        btfss PIR1, 2
        goto loop_002
    }
// while (PIR1.CCP1IF == 0) // キャプチャフラグ確認
// ;
    T1CON.TMR10N = 0; // タイマーオフ
    dat = CCP1RH << 8;
    dat |= CCP1L;
    //
    return (dat);
}

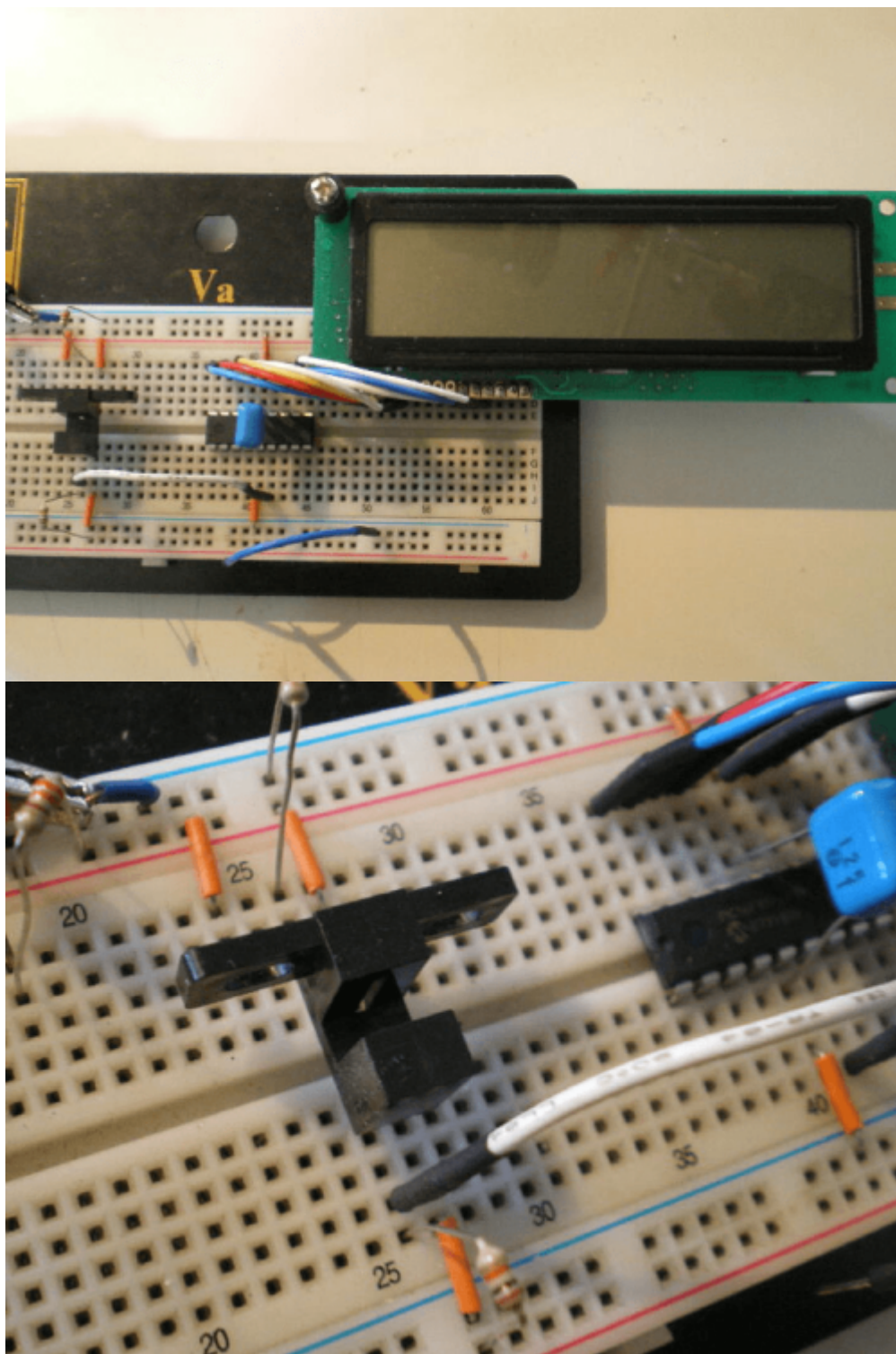
void main()
{
    static unsigned char buf[15];
    static unsigned int dat;
    static unsigned char prescale;
    static double prescaled, cycleTime, rpm;
    //
    OSCCON = 0b01010000; // クロックは2MHz
    CMCON = 0b00000111; // コンパレータは使用しない。
    ANSEL = 0b00000000; // A/D変換は使用しない。
    TRISA = 0b00111100;
    TRISB = 0b00001111;
    OPTION_REG.F7 = 0; // PORTBをプルアップする。
    //
}
```

```
T1CON.TMR1CS = 0;
T1CON.T1CKPS0 = 0;
T1CON.T1CKPS1 = 0;
T1CON.TMR1ON = 0;
TMR1H = 0;
TMR1L = 0;
PIE1.TMR1IE = 0;
PIR1.TMR1IF = 0;
prescale = 1;
prescaled = PS1;
//
CCP1CON.CCP1M3 = 0;
CCP1CON.CCP1M2 = 1;
CCP1CON.CCP1M1 = 0;
CCP1CON.CCP1M0 = 1;
CCPR1H = 0;
CCPR1L = 0;
PIE1.CCP1IE = 0;
PIR1.CCP1IF = 0;
//
Lcd_Custom_Config(&PORTA, 1, 0, 7, 6, &PORTB, 5, 6, 7);
Lcd_Custom_Cmd(LCD_CURSOR_OFF);
Lcd_Custom_Cmd(LCD_CLEAR);
//
while (1) {
    dat = measurement();
    //
    cycleTime = (double)dat * prescaled;    // usec変換
    FloatToStr(cycleTime, buf);
    Lcd_Custom_Out(1, 1, buf);
    Lcd_Custom_Out(1, 13, "usec");
    //
    rpm = ((1000000.0 / cycleTime) / 4.0) * 60.0;    // 回転数変換
    FloatToStr(rpm, buf);
    Lcd_Custom_Out(2, 1, buf);
    Lcd_Custom_Out(2, 13, "rpm");
    //
    if (PIR1.TMR1IF == 0) {                // オーバーフローチェック
        Lcd_Custom_Out(2, 16, "-");
    } else {
        PIR1.TMR1IF = 0;
        Lcd_Custom_Out(2, 16, "*");
        switch (prescale) {                // プリスケアラ自動調整
            case 1:
                T1CON.T1CKPS0 = 1;
                T1CON.T1CKPS1 = 0;
                prescale = 2;
                prescaled = PS2;
                break;
            case 2:
                T1CON.T1CKPS0 = 0;
```

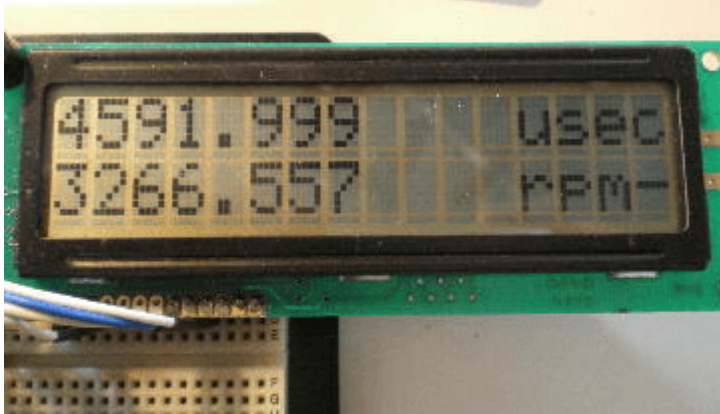
```
        T1CON.T1CKPS1 = 1;
        prescale = 4;
        prescaled = PS4;
        break;
    case 4:
        T1CON.T1CKPS0 = 1;
        T1CON.T1CKPS1 = 1;
        prescale = 8;
        prescaled = PS8;
        break;
    case 8:
        break;
    }
}
//
Delay_ms(500);
//
if (PORTB.F3 == 0) { // プリスケアラリセット (無し: 1 / 1)
    T1CON.T1CKPS0 = 0;
    T1CON.T1CKPS1 = 0;
    prescale = 1;
    prescaled = PS1;
}
}

//*****
*
```

## 動作原理



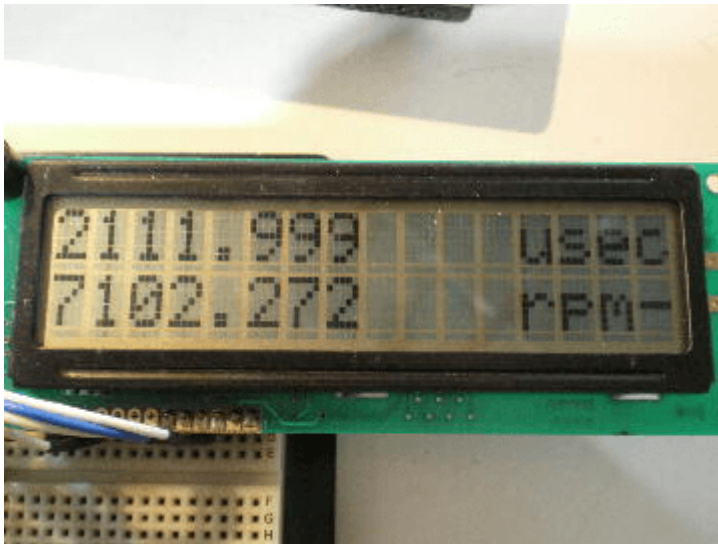
モーターを回しながら、フォトインタラプタに近づけて測定しました。左側:モーターを1Vで廻した時のパルス幅と回転数です。右側:モーターを2Vで廻した時のパルス幅と回転数です。



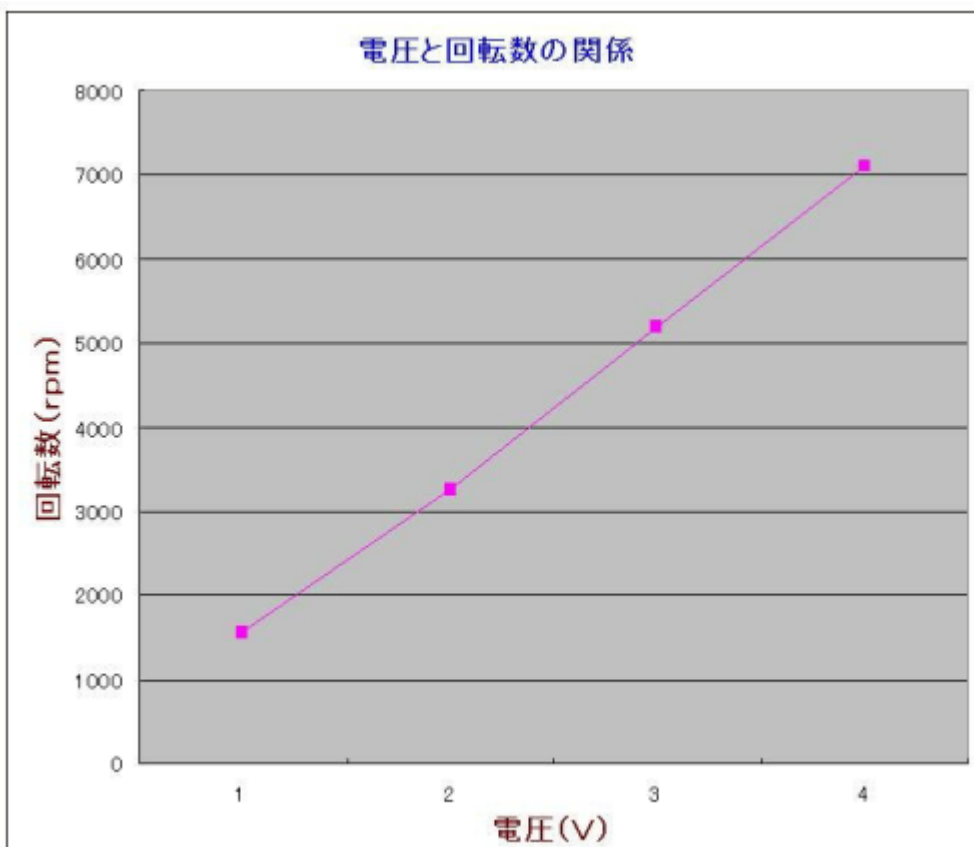
左側:モーターを3Vで廻した時のパルス幅と回転数です。 右側:モーターを4Vで廻した時のパルス幅と回



転数です。



モーターへの印加電圧と回転数の関係をグラフにしてみました。



無負荷状態のモーターの回転数ですが、結構直線性は良いようです。手作りのスリット円板にしては上出来ですね! 😊

From: <http://www.deepsky.jp/wiki/> - うごくといいな

Permanent link: <http://www.deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:106&rev=1588206272>

Last update: 2025/10/17 14:27

