第 6 章

フィードバック制御

本章では, PID 型のフィードバック制御, 特に I-P 制御に関する実験を行う. PID 型のフィードバック制御は, 後期のロボット実験で滑らかな動きを実現するために非常に重要な技術となるので, 応用できるレベルまで理解度を深めてほしい.

6.1 PWM 制御の原理

本節では, PWM 制御を用いてモータを駆動するための基礎的事項の理解とその応用を通して, 省エネルギーを意識した持続可能社会の構築へ向けての必須技術を習得する. 皆さんは, 基礎実験で, すでに PWM 制御について学んでいる. ここでは, PWM 制御のメリット, DC モータ制御に用いる場合の注意点を中心に解説する.

6.1.1 PWM 制御の必要性

PWM(Pulse Width Modulation: パルス幅変調) とは, パルス波の duty 比によってアナログ値を表す変調方式である. PWM 波による LED の明るさ制御 (図 6.1) では, 回路に掛ける電圧を 0[V] ∼ 5[V] までの中間的な電圧に設定する代わりに, 0[V] と 5[V] のディジタル的な電圧を掛け, その duty 比をアナログ的に設定した. このようにすれば, 回路に 5[V] が掛る時間平均によって LED の明るさを制御できることがわかる. アナログ的な電圧でなく, アナログ的な duty 比によって制御するメリットは何であろうか? duty 比 $d \in [0, 1]$ で制御した場合と同じ平均電流が流れるようなつぎの回路 (図 6.2右) を比較対象として考えてみよう. ここで, 図中の Z_L は負荷インピーダンスで, 例えば, モータとコンデンサをその等価回路 (図 6.3) で表したとき,

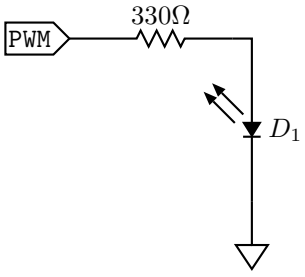

図 6.1 PWM 波による LED の明るさ制御

$$V = Z_L I = \frac{R + Ls}{1 + RCs + LCs^2} I \tag{6.1}$$

となる. PWM 変調された電圧 $V(t)$ は, つぎのように Fourier 級数展開できる.

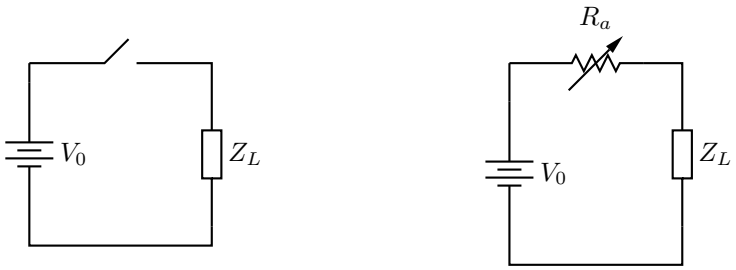

図 6.2　PWM 制御と可変抵抗による制御

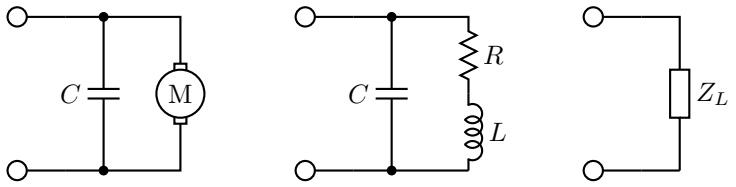

図 6.3　モータの等価回路

$$V(t) = V_0 d + \sum_{n=1}^{\infty} \{a_n \cos n\omega t + b_n \sin n\omega t\} \tag{6.2}$$

ここで, V_0 は電源電圧, d は duty 比, $\omega = 2\pi f$ は PWM 角周波数, a_n, b_n は Fourier 級数展開の係数である. Arduino Uno の 5,6 番ピンのデフォルトの PWM 周波数は約 $f = 977$[Hz] である. モータ電流の平均値 (直流成分) に興味があるので, 以下では, (6.1) 式において $s = 0$ とし, $Z_L = R$ として議論を進める.

さて, LED 発光の明るさは LED で消費される電力に比例すると仮定し, 図 6.2の左右の回路図における消費電力を求めてみよう. 図 6.2左の PWM 制御の回路図で duty 比が d のとき, Z_L で消費される電力は,

$$W_L = d\frac{V_0^2}{R}. \tag{6.3}$$

図 6.2右の制限抵抗を用いた回路図で, Z_L が消費する電力 $\dfrac{RV_0^2}{(R+R_a)^2}$ が W_L と等しくなるように R_a を求めると, $R_a = \dfrac{1-\sqrt{d}}{\sqrt{d}}R$. よって, 制限抵抗 R_a で消費される電力 W_a は,

$$W_a = (\sqrt{d} - d)\frac{V_0^2}{R} \geq 0. \tag{6.4}$$

すなわち, 制限抵抗を用いる場合は, 負荷 Z_L で消費される電力とは別に, W_a[W] だけ余分に電力を消費してしまうことになる. 制限抵抗 R_a の代わりにトランジスタを用いても, 結局同じ電力がトランジスタで消費され, 熱となって放出される.

PWM 制御は, ON/OFF の 2 状態だけを使うことによって, **“制御のために無駄なパワーを消費しない制御方式”** とみなすことができる.

消費電力が小さい場合は, あまり問題にならないが, 数 [W] 以上になると, 放熱板などをトランジスタにつけたりして熱の流れについても, 事前に考慮しなければならなくなる. また, 放熱板があると, 小型化はしにくくなる. さらに, バッテリー駆動の回路で無駄に電力を消費すると, バッテリーの持続時間が短くなり, こまめの充電が必要な不便な製品になったり, より大容量のバッテリーが必要になり, 製品の大きさやコストが大きくなったりと, 商品としての競争力を失うことになる. PWM 制御は, エネルギー効率の向上や, 制御回路の小型化にとって, 非常に重要な技術である.

6.1.2 モータの PWM 制御

本項では DC モータの PWM 制御を考える．図 6.1の PWM 制御の回路図において, LED と抵抗を DC モータに変更すればそれでよいであろうか? 答えは否である．DC モータは基本的にはコイルからできているので, 電気的にはコイルと等価である．そこで, インダクタンスの式を思い出すと, 電流 I と電圧 V の関係は,

$$L\dot{I} = V \qquad \Rightarrow \qquad I(t) = \frac{1}{L}\int^{t} V(\tau)\, d\tau \tag{6.5}$$

である．電圧 $V(t)$ が有界な場合, 電流 $I(t)$ は連続的にしか変化しない．もし, 電流が不連続に変わったとすると, デルタ関数のような電圧がかかったことになる．PWM 波形は MOSFET のようなスイッチング素子の ON/OFF によって生成されているので, 図 6.4左のような回路図では, 電流が流れているときに電流を急に OFF にすると, コイル端電圧としてデルタ関数の様な電圧が発生し, これがスパイク状の雑音となる．(“コイル電流は急には止まれない.”) そこで, スイッチング素子が OFF になったときも, 電流を流し続けるために, モータと並列にダイオードを図 6.4右のように接続する．このような使われ方をするダイオードを**フリーホイールダイオード**という．あるいは, DC モータと並列にコンデンサを接続して, モータ電圧を平滑化する．雑音を軽減するためにも, DC モータには必ず並列にコンデンサを接続すること.

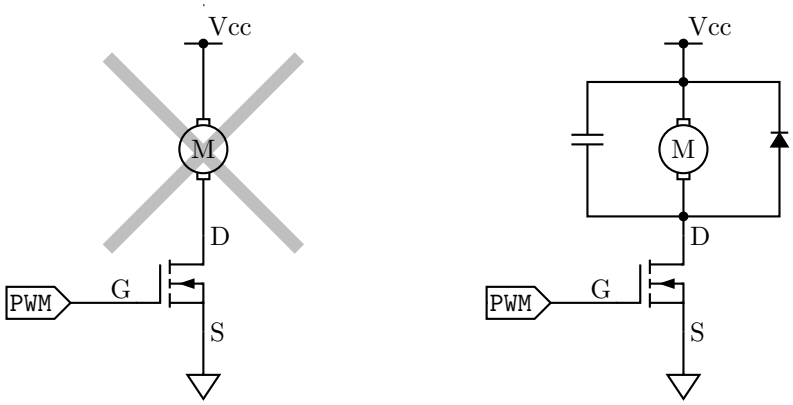

図 6.4 DC モータの PWM 制御 (左の回路ではスパイク状の雑音が発生する.)

6.2　非線形システムの線形近似

モータに印加する電圧の duty 比を $U(t)$, モータの回転速度を $Y(t)$[rpm] とする. $U(t)$ を一定値 $\bar{U}$ に保持したとき, $Y(t)$ が一定値 $\bar{Y}$ に収束した状態 (定常状態) でも, $\bar{U}$ と $\bar{Y}$ の比は一定ではない (図 6.5). たとえば, $\bar{U} = 0.3$ における接線は原点を通らないし, $\bar{U}$ が 0.3 から遠ざかると接線からのずれは大きくなる. すなわち, このシステムは非線形システムである.

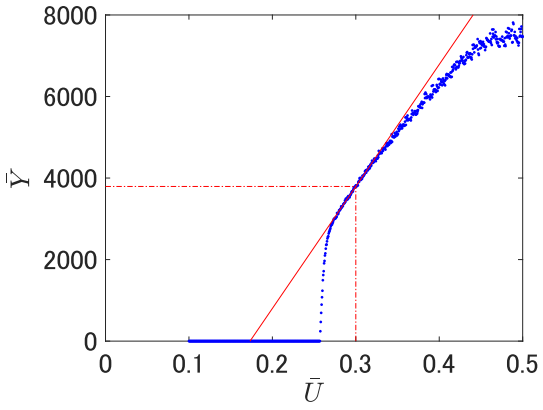

図 6.5　モータの定常応答

制御対象がたとえ非線形システムであっても, ある動作点の近傍でのみ制御する場合, 動作点近傍における線形近似モデルにもとづいて制御することが可能である. このような場合, 具体的にはどのように制御するのであろうか? まず, 線形近似した接線が原点を通らないので, 原点を動作点に移動する. 上の例では, $\bar{U} = U_0 = 0.3$ のとき, $\bar{Y} = Y_0 \sim 3800$ なので,

$$u(t) = U(t) - U_0 \tag{6.6}$$
$$y(t) = Y(t) - Y_0 \tag{6.7}$$

として, 動作点 (U_0, Y_0) の近傍でのみ動かすことにする. このように原点を動作点に移動することによって, 動作点からの偏差 $u(t)$ と $y(t)$ の関係はほぼ線形と考えることができ, その入出力特性は線形近似した伝達関数 $G(s)$ と見なすことができる (図 6.6). 前回の「第 5 章 信号処理」では原点移動についてはあまり深く考えずに実験を行ったかもしれないが, フィードバック制御を施す場合, $(u(t), y(t))$ はあくまでも動作点からの偏差であることを考慮しないと, 制御を実装する場合にミスを犯しやすいので注意が必要である.

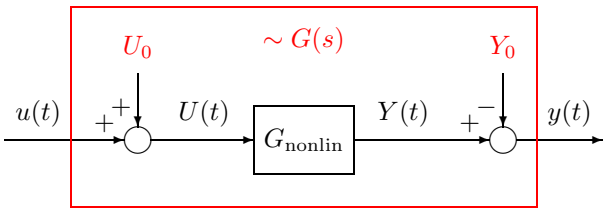

図 6.6　非線形システムの原点移動

cf. モータの目標回転速度 $R(t)$ も動作点に原点移動して, $r(t) = R(t) - Y_0$ としなければならない.

6.3 フィードバック制御

本節では, モータの回転速度を目標値に一致させるための I-P 制御系の設計法を習得する.

前回の「第 5 章 信号処理」の実験で求めた, duty 比からモータ回転速度までの伝達関数は,

$$G(s) = \frac{25000}{1 + 0.5s} \tag{6.8}$$

であった. 本節ではこの制御対象に対する I-P 制御系を設計する. すなわち, 制御入力を

$$u(t) = -K_P y(t) + K_I \int_0^t (r(\tau) - y(\tau))\, d\tau \tag{6.9}$$

で与える. ここで, $r(t)$ は目標回転速度 [rpm], K_P, K_I は制御系の設計パラメータであり, それぞれ, 比例ゲイン, 積分ゲインと呼ばれる. 図 6.7に I-PD 制御系のブロック線図を示す. 今回は, 微分動作 (D 動作) は用いないので, 微分ゲインは $K_D = 0$ である. フィードバックゲインの設計には, 北森の部分モデルマッチング法 [3] を用いることにする. 北森の部分モデルマッチング法とは, 閉ループ伝達関数 $W(s)$ ($r(t)$ から $y(t)$ までの伝達関数) が北森の参照モデル:

$$W^*(s) = \frac{1}{1 + \sigma s + 0.5\sigma^2 s^2 + 0.15\sigma^3 s^3 + 0.03\sigma^4 s^4 + 0.003\sigma^5 s^5} \tag{6.10}$$

になるべく一致するように, フィードバックゲイン K_I, K_P, K_D と参照モデルの時定数 σ を決定する方法である. 制御対象の伝達関数が

$$G(s) = \frac{b_0}{a_0 + a_1 s + a_2 s^2 + a_3 s^3 + \cdots} \tag{6.11}$$

のとき, 閉ループ伝達関数は,

$$W(s) = \frac{1}{1 + \dfrac{a_0 + b_0 K_P}{b_0 K_I} s + \dfrac{a_1 + b_0 K_D}{b_0 K_I} s^2 + \dfrac{a_2}{b_0 K_I} s^3 + \dfrac{a_3}{b_0 K_I} s^4 + \cdots} \tag{6.12}$$

となる. (6.8) 式の伝達関数の場合, $a_0 = 1$, $a_1 = 0.5$, $b_0 = 25000$ であり, そのときの $W(s)$ は,

$$W(s) = \frac{1}{1 + \dfrac{1 + 25000 K_P}{25000 K_I} s + \dfrac{0.5}{25000 K_I} s^2} \tag{6.13}$$

となり, これが,

$$W^*(s) = \frac{1}{1 + \sigma s + 0.5\sigma^2 s^2} \tag{6.14}$$

に一致するように, σ, K_I, K_P を決定する. 係数を比較することによって方程式が 2 本立つが, 変数は 3 つあるので, この問題は不定となってしまう. そこで, 参照モデルの時定数 σ を自由に選ぶことにするが, 一般に, ゆっくりな制御対象を応答の速いモデルに一致させようとすると, モデル化されなかった動特性 (unmodeled dynamics) の影響で, 閉ループ系が振動的になったり, 不安定になったりしてしまう. そこで, 参照モデルの時定数 σ は制御対象の時定数と同程度かやや速く設定するのが一般的である. 今回は, 制御対象の時定数 T_{M} と同じ $\sigma = 0.5$ と設定することにする. (6.13) 式と, (6.14) 式 (ただし, $\sigma = 0.5$) が一致するとして方程式を立てると,

$$\frac{1 + 25000 K_P}{25000 K_I} = \sigma = 0.5, \tag{6.15}$$

$$\frac{0.5}{25000 K_I} = 0.5\sigma^2 = 0.125. \tag{6.16}$$

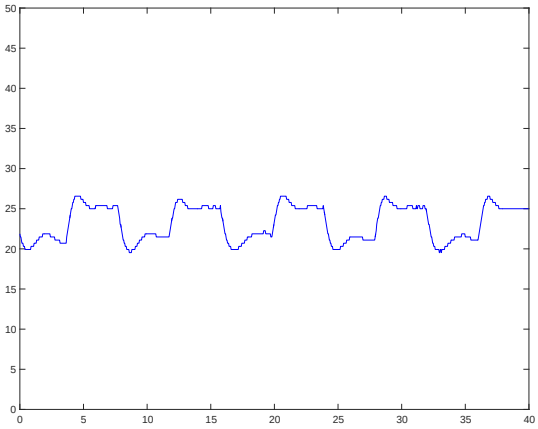

図 6.8　制御入力 $u(t)$,
横軸:t[sec], 縦軸:duty 比 $U(t)$

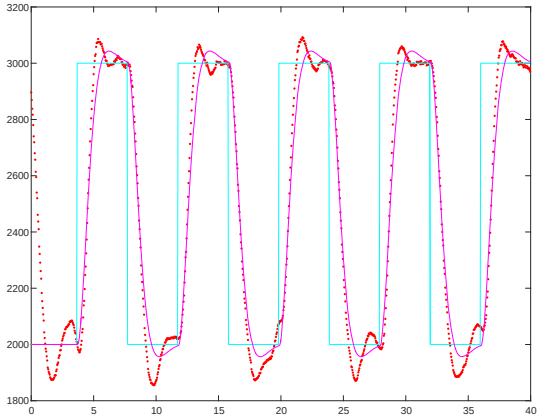

図 6.9　制御結果,
横軸:t[sec], 縦軸:R [rpm](—),
$Y(t)$[rpm](···) $Y^*(t)$[rpm](—)

これより,

$$K_I = 0.00016 \tag{6.17}$$
$$K_P = 0.00004 \tag{6.18}$$

と定まる.

この制御を実装する場合, $e(t) = r(t) - y(t)$ の積分は, 離散的な信号の和によって近似する必要がある. サンプリング周期 T_s [sec] でサンプリング/制御しているとき, 積分を次式で近似する:

$$\int_0^{kT_\mathrm{s}} e(\tau)\, d\tau \sim \sum_{i=1}^{k} e[i] T_\mathrm{s} \tag{6.19}$$

ここで, $e[k]$ は $e(t)$ の時刻 $t = kT_\mathrm{s}$ におけるサンプル値 $e[k] = e(kT_\mathrm{s})$ である.

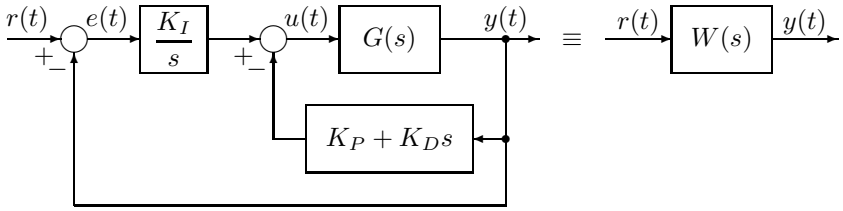

図 6.7　I-PD 制御系

回転速度の目標値 $R(t) = r(t) + Y_0$(—) を 2000[rpm] と 3000[rpm] の矩形波としたときの I-P 制御系の制御入力と出力を, それぞれ, 図 6.8と図 6.9に示す. ここで,

$$Y^*(t) = y^*(t) + Y_0 = W^*(s) r(t) + Y_0 \tag{6.20}$$

は望ましい出力応答である. 実際の応答 $Y(t) = y(t) + Y_0$ (···) は $Y^*(t)$ (—) よりも振動的になっているものの, 時定数とゲインは合っているように見える. 式 (6.9) をプログラムで実装するには, 例えば, リスト 6.1のようにする. リスト中の ... の部分は, リスト 5.1 や演習 5.2 を参照する, あるいは, 各自で考えること. モータ回転速度の原点を `Y0` として比例フィードバックしている.

リスト 6.1 I-P 制御の実装

```
// 大域変数の定義
...
                                          //
float refL = 2000.0;                      // 目標回転速度 1[rpm]
float refH = 3000.0;                      // 目標回転速度 2[rpm]
float ref = refL;                         // 目標回転速度 [rpm]
float U0 = 0.2;                           // 動作点を duty 比 0.2 近傍とする
float Y0 = 1950.0;                        // U0=0.2 のときの定常的な回転速度 [rpm]
int Ts = 40;                              // サンプリング周期 [msec]
float sum_e = 0;
float Kp = 0.00004;
float Ki = 0.00016*((float)Ts/1000.0);  // サンプリング周期をあらかじめ掛けておく
void countUp() {                        // I 外部割込み (D2) の SR
  ...
}
void control() {                        // タイマ割り込み (MsTimer2) の ISR 40[msec] ごと
  float e;
  iteration++
  if (iteration>=100) {                 // 4[sec] ごとに
    // 目標値 ref の設定; 4[sec] ごとに refH, refL に交互に変更
    ...
  }
  // 制御入力の決定
  e = ref-motrpmf;                        // 誤差
  sum_e += e;                             // 誤差の積分 (和分)
  duty = Kp*(Y0-motrpmf)+Ki*sum_e+U0; // I-P 制御
  if (duty<0) {                           // duby 比を [0,0.5] に制限
    duty = 0.0;
  }
  if (duty>0.5) {
    duty = 0.5;
  }
  analogWrite(D6,duty*255.0);             // duty 比の変更 (40[msec] ごとに)
  // データの送信
  ...
}
void setup() {
  ...
}
void loop() {}
```

6.4　フィードバック制御の実験 (I-P 制御)

本節では, I-P 制御によるモータの回転速度制御の実験を行う. 実験の後半で, ストロボフラッシュによるモータ回転速度の可視化を行う. そのために, まず, PWM 信号の周波数を変更する方法に関して演習する.

6.4.1　PWM 信号の周波数の変更

Arduino は 6 つの PWM 信号を同時に発生できる. これらの PWM は 3 つの独立したタイマカウンタ `TCNT0`, `TCNT1`, `TCNT2` から生成されており, それぞれ, 5,6 番ピン, 9,10 番ピン, 3,11 番ピンに対応している. それぞれの系統の PWM 周波数を変更することが可能であるが, 各系統によって設定方法は異なっている. ここでは, 比較的細かく周波数を変更することが可能な, 9,10 番ピンの系統を使用することにする.

9,10 番ピンの PWM 周波数は, 分周比 `r` とカウントビット `top` という 2 つのパラメータによって変更する. Arduino のクロック周波数を $f_c = 16[\mathrm{MHz}] = 16 \times 10^6[\mathrm{Hz}]$ とするとき, 9,10 番ピンの PWM 周波数 f は次式で与えられる.

$$f = \frac{f_c}{\mathtt{r} \times (2 \times \mathtt{top})}[\mathrm{Hz}] \tag{6.21}$$

分周比 (prescaler) `r` は, `TCCR1B` の下位 3 ビットを使って指定し, $\{1, 8, 64, 256, 1024\}$ の中から選ぶことができる (デフォルト値は 64). カウントビット `top` は `unsigned int` である (デフォルト値は 255). 上式分母の 2 は, phase-correct PWM というモードで動作する際, カウンタのアップカウントとダウンカウントの往復動作を反映するための係数である. すなわち, `top` は半周期分のカウント値である (図 6.10).

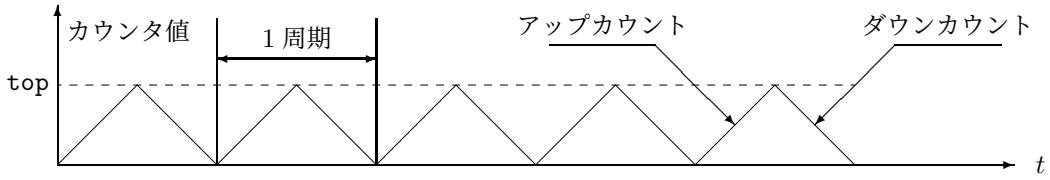

図 6.10　1 周期とカウンタ値の関係

9,10 番ピンのデフォルト PWM 周波数は, 次式により, 約 490[Hz] である.

$$f = 16 \times 10^6/64/2/255 \sim 490.19[\mathrm{Hz}]. \tag{6.22}$$

⟨演習 6.1⟩ LED 点滅周期の変更

超高輝度白色 LED をマイコンの PWM 信号を用いて点滅させる.

1. 前回の実験「第 5 章 信号処理」の回路に追加して, 図 6.11 の LED 発光回路をブレッドボード上に作成し, 信号線をマイコンの出力ポート `D9` に接続せよ (図 6.12).
2. LED を 1Hz で点滅させるプログラム例を, リスト 6.2に示す. リスト 6.2を実行し, 白色 LED が 1[sec] 間隔で点滅しているか確認せよ.
3. LED の点滅周期を, 4 秒ごとに 1Hz → 2Hz → 5Hz → 10Hz → 1Hz → ⋯ と変化させるプログラムを作成し, 実行せよ. 4 秒の計測には `MsTimer2` を用いたタイマ割り込み, または, `millis()` 関数を用いよ.

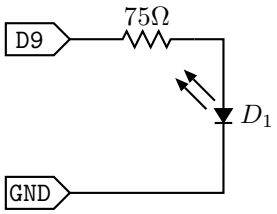

図 6.11　白色 LED 発光回路

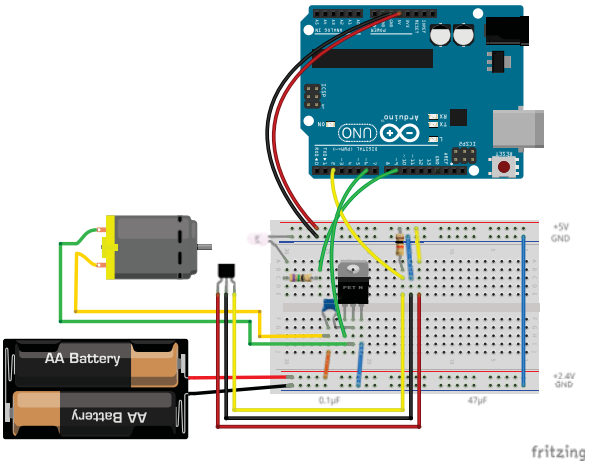

図 6.12　図 6.11 の回路のブレッドボード上での実装

6.4.2　I-P 制御

第 6.3 節で説明した I-P 制御によって, モータの回転速度を制御する. 北森の部分モデルマッチング法によって, あらかじめ, K_P, K_I を求めておくこと. その際, 制御対象のモデルは, (6.8) 式ではなく, 前回の実験「第 5 章 信号処理」で求めた伝達関数を使うこと.

⟨課題 6.1⟩ I-P 制御

1. リスト 5.1, および, リスト 6.1を参考にして, I-P 制御器 (6.9) を実装せよ.
   - Duty 比は 0 以上, 0.5 以下となるように飽和させてから印加すること.
   - 制御周期は, $T_\mathrm{s} = 40$[msec] とせよ.
   - 参照信号 $R(t)$ は, 4[sec] ごとに値 2000[rpm] と 3000[rpm] を交互に切り換える矩形波とせよ.
   - LPF に通したモータ回転速度 $Y(t)$, duty 比の 255 倍 $(255 \times U(t))$, 参照信号 $R(t)$ の三つ組みを, `unsigned int` 型変数にキャストし, `'H'` に続く 6 バイト (各 2 バイト) のバイナリデータとして, 40[msec] ごとにパソコンに送信するようにせよ.
2. `rpmmon.pde` を用いて, 回転速度 $Y(t)$, duty 比 $U(t)$, 参照信号 $R(t)$ を観測せよ. 意図した通りの動作になっているか, 確認せよ.
3. K_I の値は変更せず, K_P の値を 2 倍, 0.5 倍に変更したとき, どのような応答が得られるか? 確認せよ.
4. K_P の値は変更せず, K_I の値を 2 倍, 0.5 倍に変更したとき, どのような応答が得られるか? 確認せよ.

ソースコード作成の際, PDF ファイルから Copy&Paste でコピーしないこと. 文字化けしてコンパイルエラーまたは実行時エラーの原因となる.

リスト 6.2　タイマー 1 の初期設定

```
const int D9 = 9;
float top = 31250;
void initPWM1() { // PWM freq = 16MHz / r (prescaler) / 2 (phase-correct) / (top)
// when r=256, 1Hz  -> top = 31250 (60rpm)
//             10Hz  -> top = 3125 (600rpm)
TCCR1A = 0b10100000;
// TCCR1B = 0b00010001; //r=1
// TCCR1B = 0b00010010; //r=8
// TCCR1B = 0b00010011; //r=64
TCCR1B = 0b00010100;    //r=256
// TCCR1B = 0b00010101; //r=1024
ICR1 = top;
}
void setup() {
    initPWM1();
    analogWrite(D9, 255 * 0.03);
}
void loop() {}
```

6.4.3　ストロボフラッシュによるモータ回転速度の可視化

モータに取り付けられた回転板には白黒のパターンが描かれている．モータの回転速度に合わせて, ストロボ光を回転板に当てれば回転板は止まって見えるはずである．

⟨課題 6.2⟩ ストロボフラッシュによるモータ回転速度の可視化

LED の点滅周波数をモータの目標回転速度に設定することによって, 回転板の白黒パターンが止まって見えるようにする．

1. 課題 6.1のプログラムに, 白色 LED を 3000[rpm] で点滅させるプログラムを追加して実行せよ．
   注. 白色 LED の光がモータの回転板に当たるように配置せよ．また, ストロボ光らしくするために可能な限り duty 比を小さくせよ．

⟨発展課題 6.1⟩ 原点移動の影響

動作点 (U_0, Y_0) の近傍で制御するため, 原点を移動して制御していた．原点移動にはどのような効果があるか調べてみよう．

1. U_0, Y_0 の片方だけまたは両方を 0 に設定し直して, 課題 6.1 または 6.2と同様の実験を行ってみよ．どのような違いが見られるか観測せよ．
2. 立上りの状況を繰り返し再現するため, 目標回転速度を 4 秒ごとに 0rpm → 2000rpm → 3000rpm → 2000rpm → 3000rpm → 0rpm → $\cdots$ と変化させるようにプログラムを修正せよ．また, 目標回転速度が 0rpm のときは, `duty` 比を 0 とし, 積分補償器の状態 (`sum_e`) を 0 にリセットせよ．U_0, Y_0 をいろいろと変更して応答を比較せよ．

6.5 考察とレポート

報告は, グラフやプログラムを添付するだけでなく, 結果や考察を必ず日本語または英語で説明すること.

[レポート 6.1] I-P 制御

1. 課題 6.1の目的は何か? 90 字以上で答えよ.
2. フィードバックゲイン K_P, K_I および (U_0, Y_0) を報告せよ. 値を報告するだけでなく, どのように求めたか? どのように定めたか? 説明すること.
3. 課題 6.1-2 の結果を報告せよ. その際, モータの回転速度, duty 比, 目標回転速度などのデータを, `Scilab` を用いてグラフにすること.
4. 課題 6.1-3 の結果を報告せよ. K_P の値の違いによる応答の違いをグラフを用いて説明せよ. K_P の役割や調整の仕方について考察せよ. Scilab によるグラフの作成では, 3 つの応答の時間軸を調整して 1 つのグラフに描き, 時定数などを比較できるようにせよ.
5. 課題 6.1-4 の結果を報告せよ. K_I の値の違いによる応答の違いをグラフを用いて説明せよ. K_I の役割や調整の仕方について考察せよ. Scilab によるグラフの作成では, 3 つの応答の時間軸を調整して 1 つのグラフに描き, 時定数などを比較できるようにせよ.

[レポート 6.2] ストロボフラッシュ

1. 課題 6.2の目的は何か? 50 字以上で答えよ.
2. 課題 6.2の結果を報告せよ. その際, モータの回転速度, duty 比, 目標回転速度などのデータを `Scilab` を用いてグラフにすること. 回転板がどのような状況のときにどのように見えたか説明せよ.

[発展課題レポート 6.1] 原点移動の影響

1. 発展課題 6.1の目的は何か? 60 字以上で答えよ.
2. 発展課題 6.1の結果を報告せよ. その際, Scilab を用いて, モータの回転速度, duty 比, 目標回転速度などのデータを比較が容易になるように工夫してグラフにすること. (U_0, Y_0) の値の違いにより, 応答にどのような違いが表れたか説明せよ. また, その違いの原因について考察せよ.

表 6.1　使用部品

部品名	規　格	個数	備考
Arduino Uno		1	
USB ケーブル	A オス-B オス 1.5m A-B	1	
ブレッドボード	EIC-801	1	
ブレッドボード・ジャンパーワイヤ	EIC-J-L	1	
モータ・ホールセンサセット	FA-130RA[1] DRV5023A[2] 回転板 ネオジム磁石 ($\phi 3 \times 4$mm)	1	貸出 次回回収
FET	2SK2232[4]	1	貸出 次回回収
白色超高輝度 LED		1	貸出 次回回収
コンデンサ	積層セラミック, 0.1μF	1	貸出 次回回収
小型クリップ付コード		2	貸出 次回回収
電池ボックス	2 個用	1	貸出 次回回収
電池	NiH 充電池	2	貸出 本日回収
抵抗	75Ω	1	
抵抗	10kΩ	1	

参考文献

[1] MABUCHI MOTOR. *FA-130RA Metal-brush motors*, 3 2005.

[2] Texas Instruments. *DRV5023 Digital-Switch Hall Effect Sensor*, 5 2016.

[3] 北森俊行. 制御対象の部分的知識に基づく制御系の設計法. **計測自動制御学会論文集**, 15:549–555, 1979.

[4] 東芝. **東芝電界効果トランジスタシリコン** *N* **チャネル** *MOS* **形** *2SK2232*, 11 2006.