マイコン講習
―第2回―

K.Miyauchi

2021.11.11

目次

- PWMによる電圧疑似変換
- DCモータの回転数制御の基本
- DCモータの速度制御の基本
- DCモータの制御（SMB方式）
- DCモータの制御（LAP方式）
- シリアル通信（UART; 受信）

本講習で使用するもの

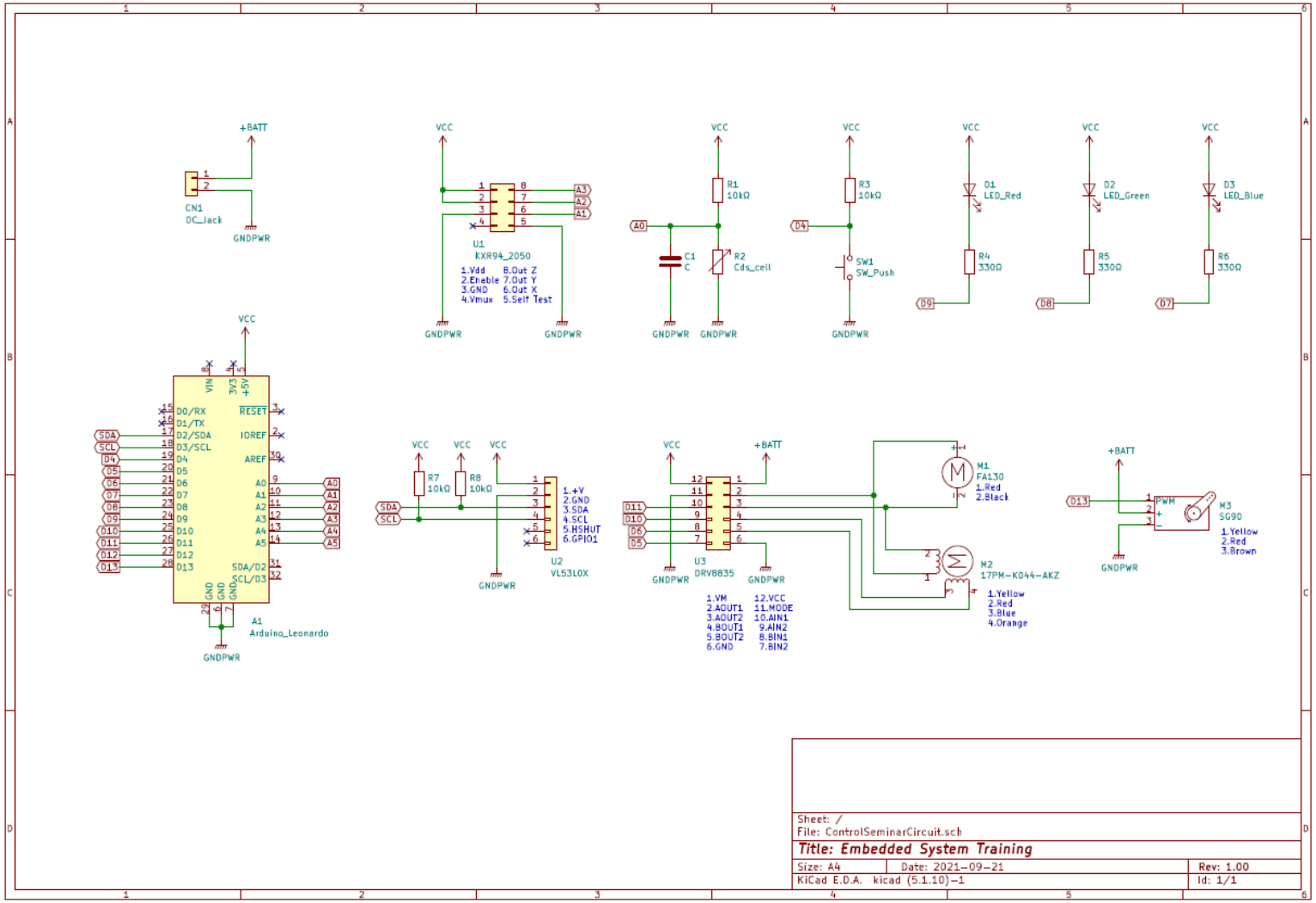
パソコン：Windows, Mac

開発環境：Arduino IDE

マイコン：Arduino Leonardo

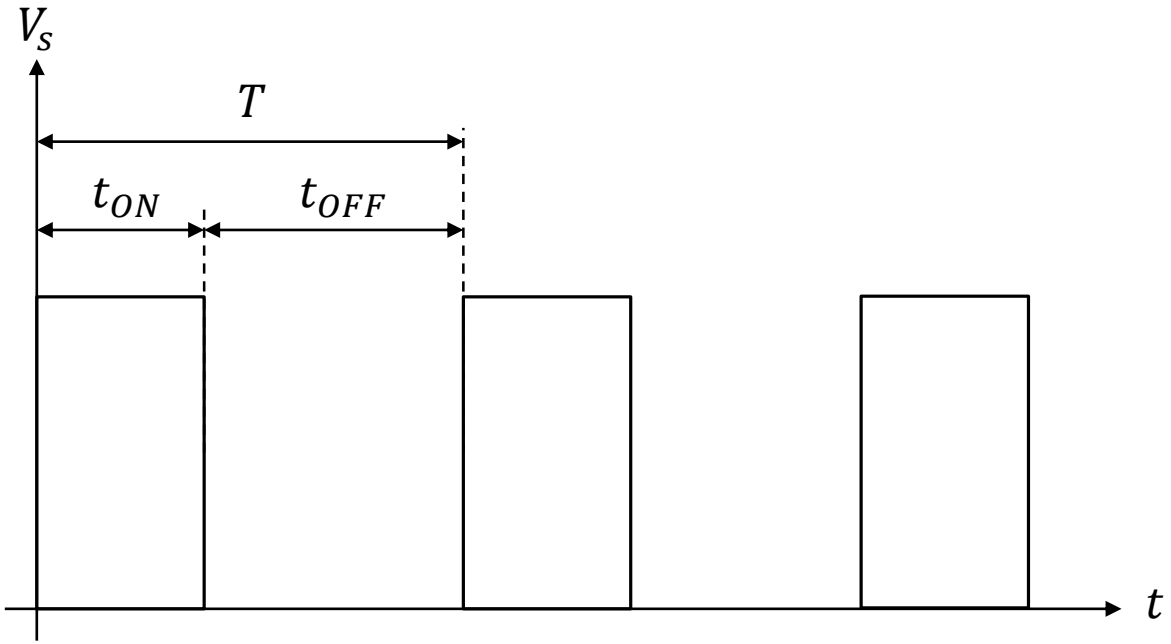
その他，回路周りの物品

回路図


+BATT
CN1
DC_Jack
GNDPWR
VCC
U1
KXR94_2050
1.Vdd 8.Out Z
2.Enable 7.Out Y
3.GND 6.Out X
4.Vmux 5.Self Test
R1
10kΩ
C1
C
R2
Cds_cell
R3
10kΩ
SW1
SW_Push
D1
LED_Red
R4
330Ω
D2
LED_Green
R5
330Ω
D3
LED_Blue
R6
330Ω
A1
Arduino_Leonardo
R7
10kΩ
R8
10kΩ
U2
VL53L0X
1.+V
2.GND
3.SDA
4.SCL
5.HSHUT
6.GPIO1
U3
DRV8835
1.VM 12.VCC
2.AOUT1 11.MODE
3.AOUT2 10.AIN1
4.BOUT1 9.AIN2
5.BOUT2 8.BIN1
6.GND 7.BIN2
M1
FA130
1.Red
2.Black
M2
17PM-K044-AKZ
1.Yellow
2.Red
3.Blue
4.Orange
M3
SG90
1.Yellow
2.Red
3.Brown
PWM
Sheet: /
File: ControlSeminarCircuit.sch
Title: Embedded System Training
Size: A4
Date: 2021-09-21
Rev: 1.00
KiCad E.D.A. kicad (5.1.10)-1
Id: 1/1

PWM信号による電圧疑似変換

パルス幅変調（PWM：Pulse Width Modulation）信号とは、
一定周期のパルス信号において，
ON時間を変調した信号のことである



T：PWM周期
t_{ON}：ON時間
t_{OFF}：OFF時間

t_{ON}とTの比率のことを
デューティ比という.

PWM信号による電圧疑似変換

PWM信号を用いると,
電圧疑似変換をすることができる（疑似降圧DC－DC変換）

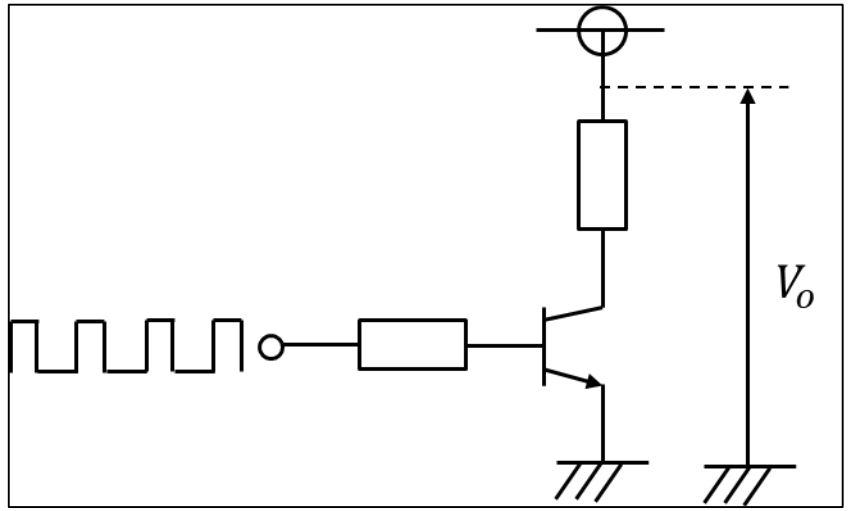
$$V_o \approx E \times \frac{t_{on}}{t_{on} + t_{off}} = E \times \frac{t_{on}}{T}$$

ただし、

V_o : 疑似変換電圧
E : 電源電圧

である.

なぜ, こんなことができるのかは、
Google先生に聞いてください.



PWM信号による電圧疑似変換

メリットとしては、
・手軽に電圧変換を実現することができる

デメリットとしては、
・応答速度が遅い
・デューティ比が小さくなるとさらに応答が悪くなる

といったことがあるので,
この後、モータの速度制御などでも
PWM信号による疑似変換を使用するが,
DACなどに比べると制御性能が落ちることを頭に入れておきましょう.

DCモータの回転数制御の基本

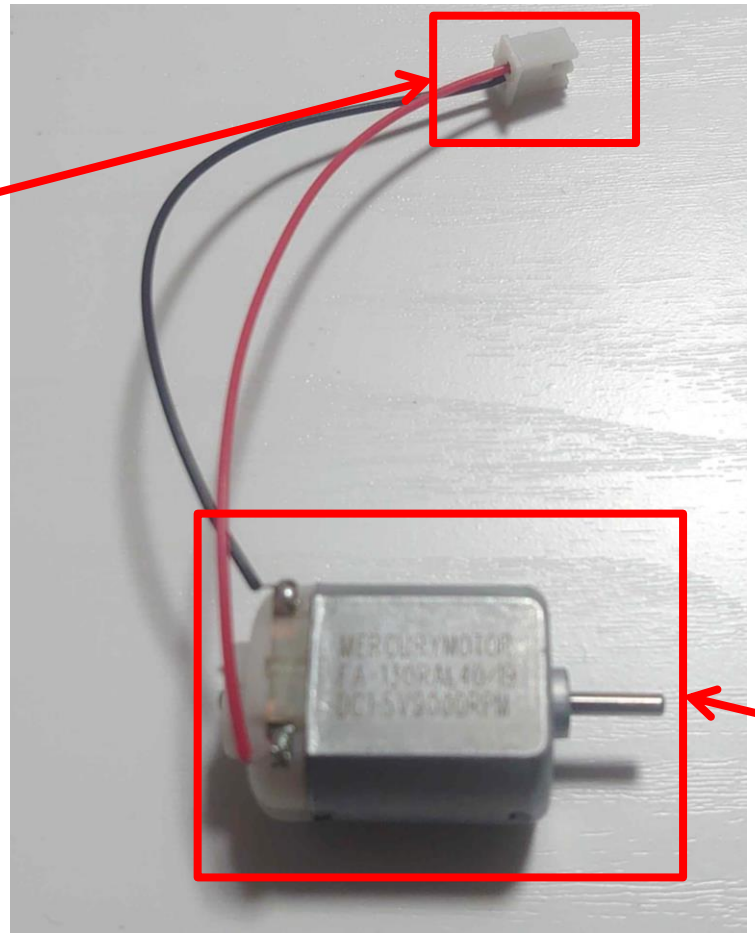
DCモータ（DCブラシモータ）は，
整流子とブラシを用いて構成されているモータ．

小・中学生の理科の実験とかで使われるようなモータは、
DCブラシモータである．

ミニ四駆などで使用されているモータも
このDCブラシモータである．

DCモータの回転数制御の基本

制御端子
2ピンある

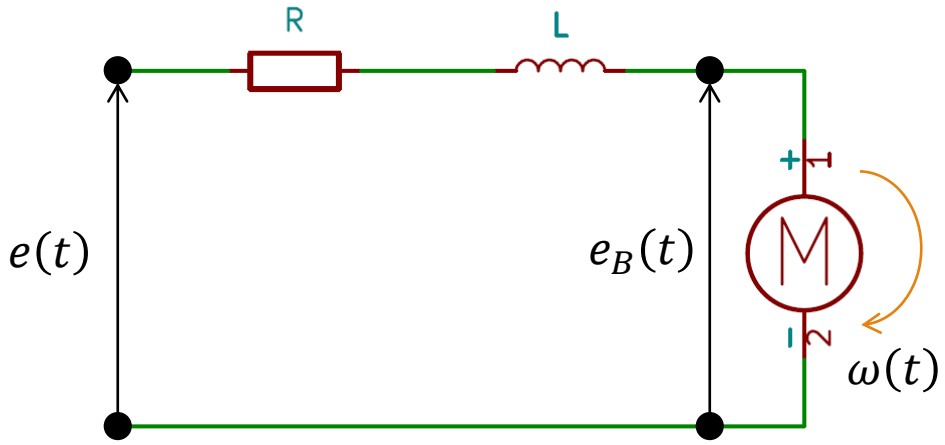


モータ本体

DCモータの回転数制御の基本

・DCモータ（正確にはDCブラシモータ）は以下の4式が成り立つことが知られている.

$$e(t) = L\frac{di(t)}{dt} + Ri(t)$$
$$e_B(t) = K_E\omega(t)$$
$$\tau(t) = K_T i(t)$$
$$\tau(t) = J\frac{d\omega(t)}{dt} + \tau_L + D\omega(t)$$



DCモータ簡易モデル

$e(t)$：起電力
$e_B(t)$：逆起電力
$\omega(t)$：角速度
$i(t)$：電流
$\tau(t)$：発生トルク

R：電機子抵抗[Ω]
L：電機子巻線[H]
K_E：逆起電力定数
K_T：トルク定数
J：ロータの慣性モーメント
τ_L：負荷トルク
D：減衰定数

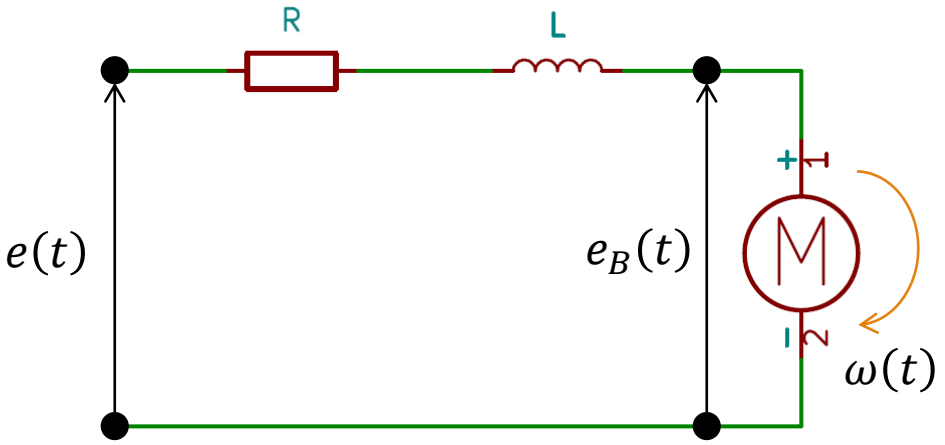
DCモータの回転数制御の基本

・先ほどの4式をラプラス変換して，
（今は，ラプラス変換について知らなくてもよい．）

$$E(s) = LsI(s) + RI(s) + E_B(s)$$
$$E_B(s) = K_E\Omega(s)$$
$$T(s) = K_T I(s)$$
$$T(s) = Js\Omega(s) + T_L + D\Omega(s)$$

整理して，

$$E(s) = LsI(s) + RI(s) + K_E\Omega(s)$$
$$K_T I(s) = Js\Omega(s) + T_L + D\Omega(s)$$



DCモータ簡易モデル

$e(t)$：起電力
$e_B(t)$：逆起電力
$\omega(t)$：角速度
$i(t)$：電流
$\tau(t)$：発生トルク

R：電機子抵抗[Ω]
L：電機子巻線[H]
K_E：逆起電力定数
K_T：トルク定数
J：ロータの慣性モーメント
τ_L：負荷トルク
D：減衰定数

DCモータの回転数制御の基本

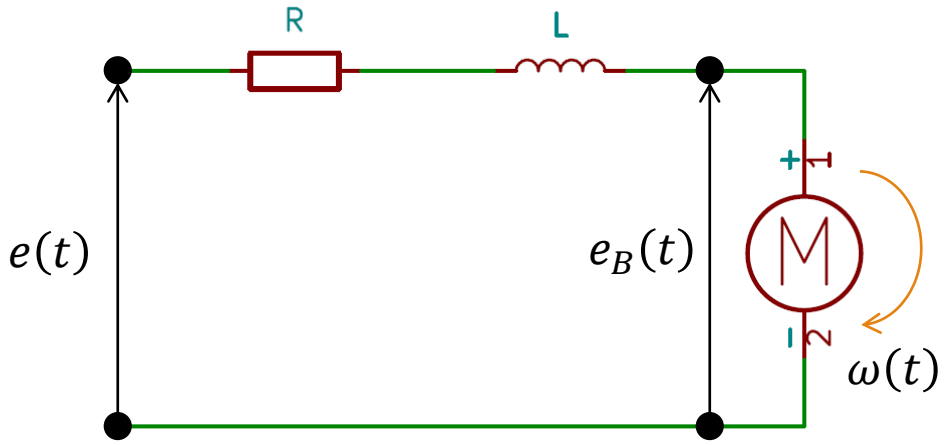
・簡単化のために$T_L = 0, D = 0$として，
整理すれば，

$$I(s) = \frac{E(s) - K_E\Omega(s)}{Ls + R}$$

$$\Omega(s) = \frac{K_T I(s)}{Js}$$

$$\therefore \Omega(s) = \frac{K_T}{Js}\frac{E(s) - K_E\Omega(s)}{Ls + R}$$

$$\therefore \Omega(s) = \frac{K_T}{Js(Ls + R) + K_T K_E}E(s)$$



DCモータ簡易モデル

$e(t)$：起電力
$e_B(t)$：逆起電力
$\omega(t)$：角速度
$i(t)$：電流
$\tau(t)$：発生トルク

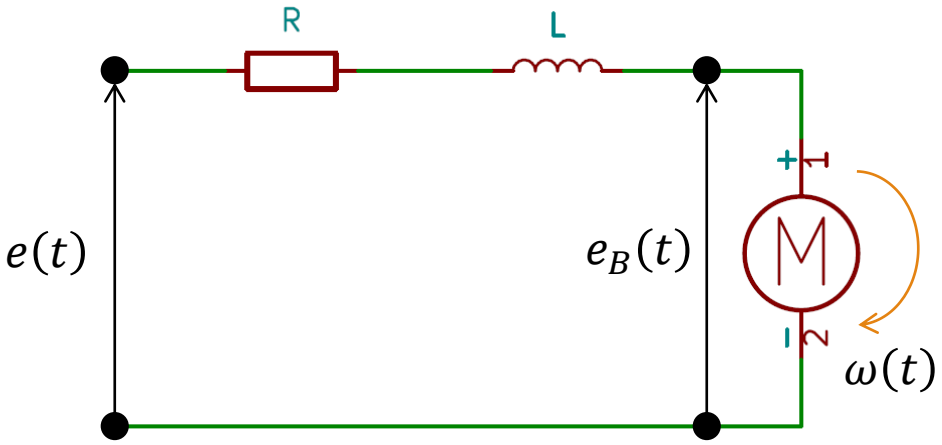
R：電機子抵抗[Ω]
L：電機子巻線[H]
K_E：逆起電力定数
K_T：トルク定数
J：ロータの慣性モーメント
τ_L：負荷トルク
D：減衰定数

DCモータの回転数制御の基本

・さらに整理すれば,

$$\Omega(s) = \frac{\frac{1}{K_E}}{\frac{JL}{K_T K_E}s^2 + \frac{JR}{K_T K_E}s + 1} E(s)$$

$$= \frac{\frac{1}{K_E}}{\frac{JL}{K_T K_E}s^2 + \left(\frac{JR}{K_T K_E}\right)s + 1} E(s)$$



DCモータ簡易モデル

$e(t)$: 起電力
$e_B(t)$: 逆起電力
$\omega(t)$: 角速度
$i(t)$: 電流
$\tau(t)$: 発生トルク

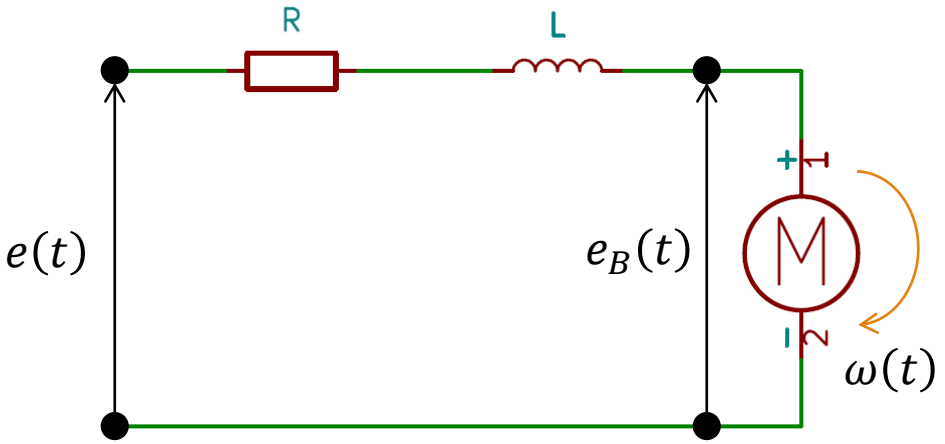
R : 電機子抵抗[Ω]
L : 電機子巻線[H]
K_E : 逆起電力定数
K_T : トルク定数
J : ロータの慣性モーメント
τ_L : 負荷トルク
D : 減衰定数

DCモータの回転数制御の基本

・ここで，$\frac{L}{R} \ll \left(\frac{JR}{K_T K_E}\right)$として，

$$\frac{\Omega(s)}{} = \frac{\frac{1}{K_E}}{\frac{JL}{K_T K_E}s^2 + \left(\frac{JR}{K_T K_E} + \frac{L}{R}\right)s + 1} E(s)$$

$$= \frac{\frac{1}{K_E}}{\left(\frac{JR}{K_T K_E}s + 1\right)\left(\frac{L}{R}s + 1\right)} E(s)$$



DCモータ簡易モデル

$e(t)$：起電力
$e_B(t)$：逆起電力
$\omega(t)$：角速度
$i(t)$：電流
$\tau(t)$：発生トルク

R：電機子抵抗[Ω]
L：電機子巻線[H]
K_E：逆起電力定数
K_T：トルク定数
J：ロータの慣性モーメント
τ_L：負荷トルク
D：減衰定数

DCモータの回転数制御の基本

・$T_M = \dfrac{JR}{K_T K_E}$, $T_E = \dfrac{L}{R}$として,

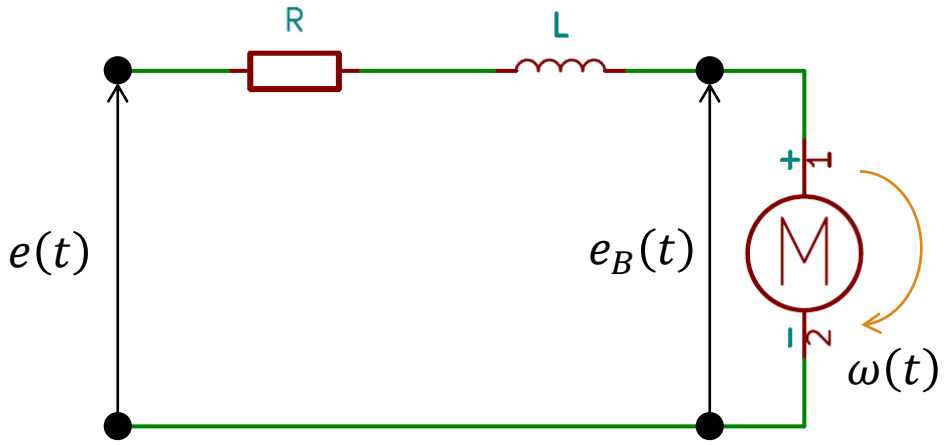
$$\Omega(s) = \frac{\dfrac{1}{K_E}}{(T_M s + 1)(T_E s + 1)} E(s)$$

T_Mを機械的時定数

T_Eを電気的時定数という.

モータメーカーによっては,
この時定数を公開しているところもある.

※ $T_E \ll T_M$である.



DCモータ簡易モデル

$e(t)$:起電力
$e_B(t)$:逆起電力
$\omega(t)$:角速度
$i(t)$:電流
$\tau(t)$:発生トルク

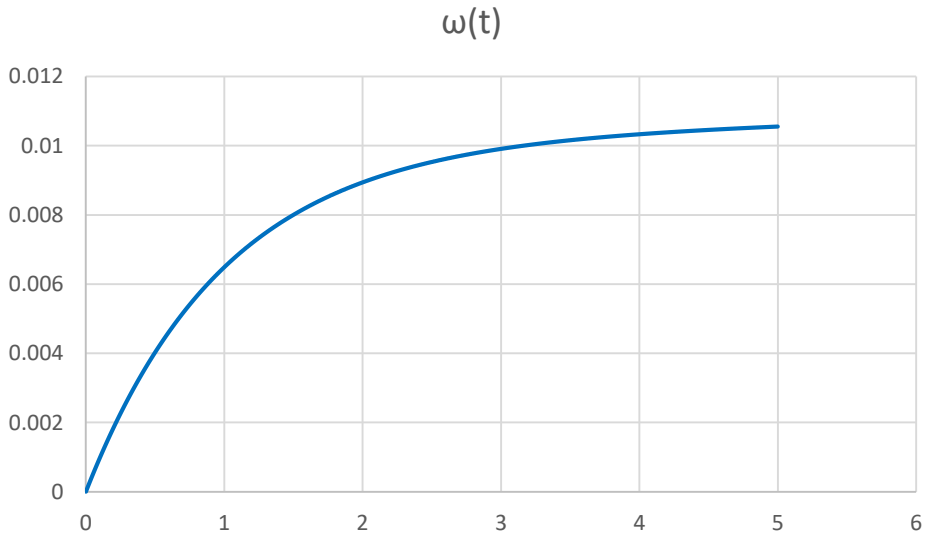
R:電機子抵抗[Ω]
L:電機子巻線[H]
K_E:逆起電力定数
K_T:トルク定数
J:ロータの慣性モーメント
τ_L:負荷トルク
D:減衰定数

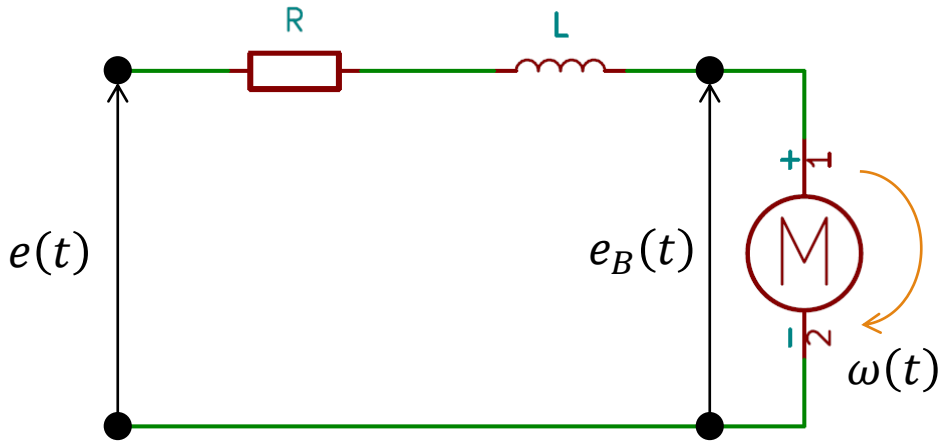
DCモータの回転数制御の基本

・逆ラプラス変換して，

$$\omega(t) = \frac{e^{-\frac{t}{T_E}} - e^{-\frac{t}{T_m}}}{K_E(T_E - T_M)} e(t)$$

以下のような波形になる（ステップ応答）





DCモータ簡易モデル

$e(t)$：起電力
$e_B(t)$：逆起電力
$\omega(t)$：角速度
$i(t)$：電流
$\tau(t)$：発生トルク

R：電機子抵抗[Ω]
L：電機子巻線[H]
K_E：逆起電力定数
K_T：トルク定数
J：ロータの慣性モーメント
τ_L：負荷トルク
D：減衰定数

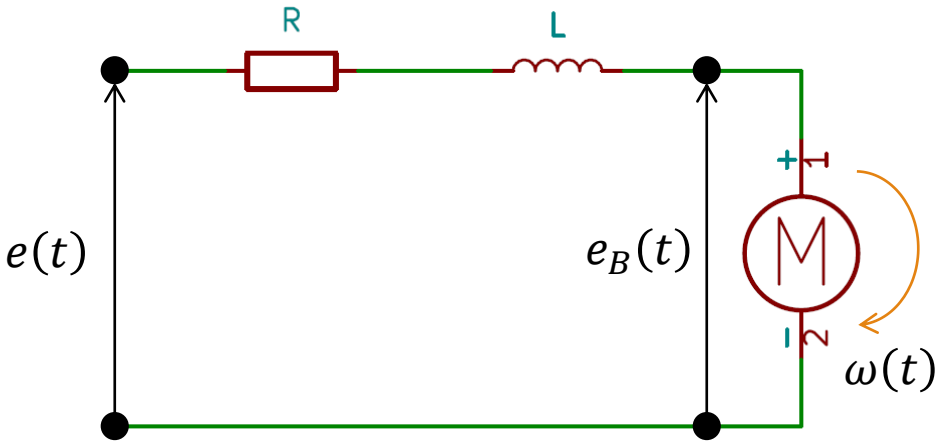
DCモータの回転数制御の基本

先ほどの式より，
印加電圧によって，
最大の回転数が変化することがわかる．

そのため，回転数を制御したい時には，
電圧を制御すればよい．

しかし，
ロボットなどを動かしているときに，
バッテリーを物理的に変えることは困難

PWM（Pulse Width Modulation）を使用して，疑似的に電圧を変化させる．



DCモータ簡易モデル

$e(t)$：起電力
$e_B(t)$：逆起電力
$\omega(t)$：角速度
$i(t)$：電流
$\tau(t)$：発生トルク

R：電機子抵抗[Ω]
L：電機子巻線[H]
K_E：逆起電力定数
K_T：トルク定数
J：ロータの慣性モーメント
τ_L：負荷トルク
D：減衰定数

DCモータの回転数制御の基本

まとめ

・機械時定数は電気的時定数よりものすごく大きい

・モータの角速度と印加電圧は以下の関係である.

$$\omega(t) = \frac{e^{-\frac{t}{T_E}} - e^{-\frac{t}{T_M}}}{K_E(T_E - T_M)} e(t)$$

⇒印加電圧を変えると回転速度の最大値も変わる.

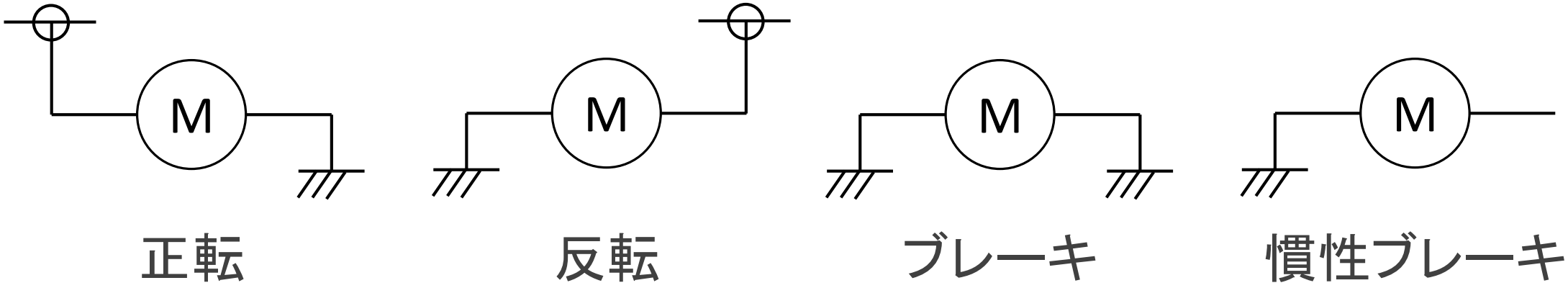
⇒PWMによって，電圧を変化させる

DCモータの速度制御の基本

DCモータは2端子あるため，どちらを高電位，低電位にするかによって，回転する方向を変えることができる.

さきほどの回転数制御と合わせることによって，
DCモータの速度制御を行うことができる.

2端子の接続パターンは，以下の4パターンがよく使用される.

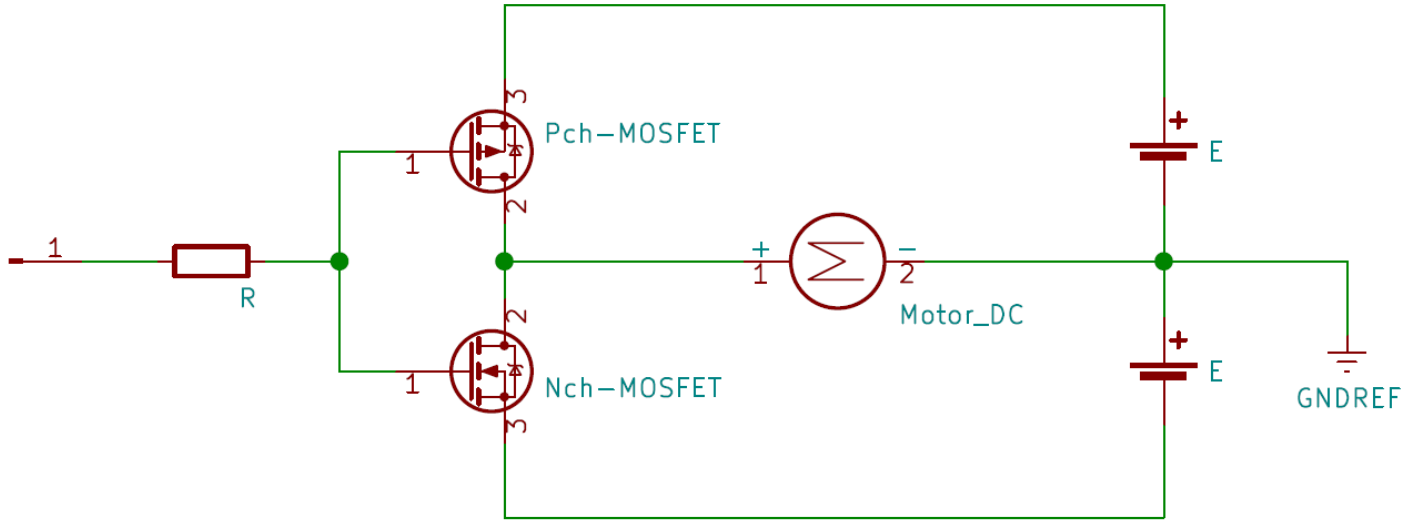


正転　反転　ブレーキ　慣性ブレーキ

DCモータの速度制御の基本

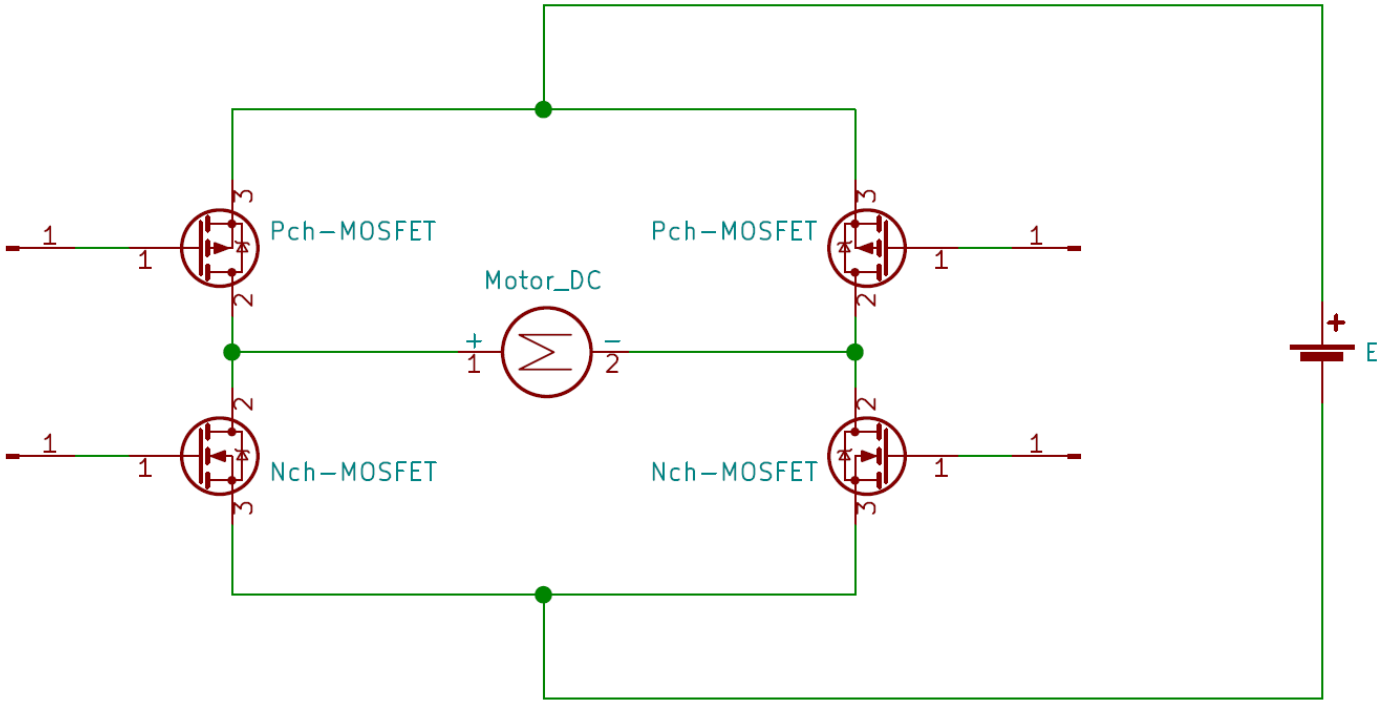
速度を制御するために
2つの電源を用いた方法と1つの電源を用いた方法がある.

2電源方式の場合は，以下のような回路で制御を行う.
あまり使用する機会がないので説明は省略.



DCモータの速度制御の基本

1電源方式の速度制御では，以下のHブリッジ（フルブリッジ）回路と呼ばれる回路で制御が行われる．



DCモータの速度制御の基本

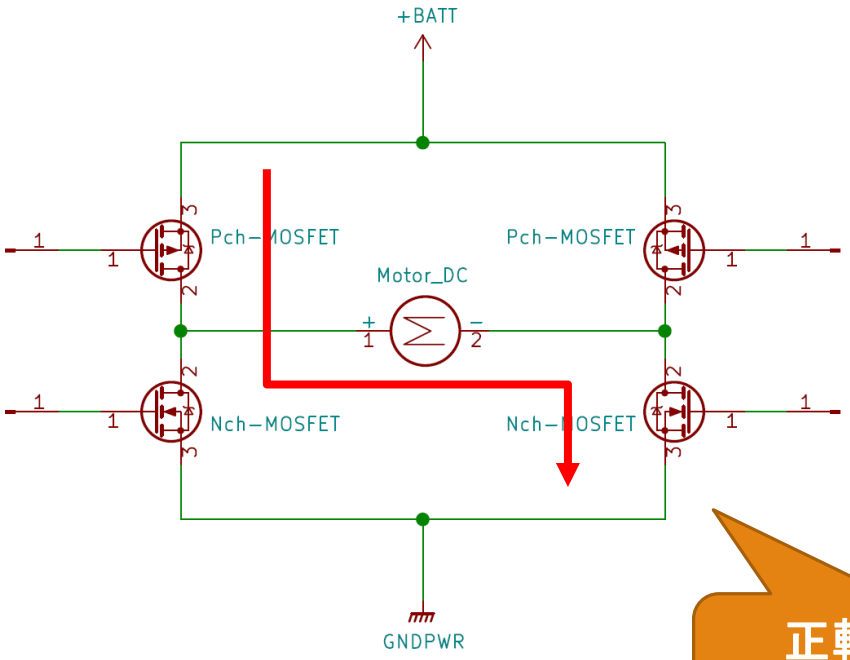
1電源方式の速度制御方法としてSMB方式とLAP方式がある.

SMB（Sign Magnitude Break）方式は,
正転 or 反転の信号とブレーキ信号を交互に送信して
速度を制御する方法.
正転0%，反転0%のときに静止することになる.

LAP（Locked Anti-Phase）方式は,
正転の信号と反転の信号を交互に送信して,
速度を制御する方法.
正転50%，反転50%のときに静止することになる.

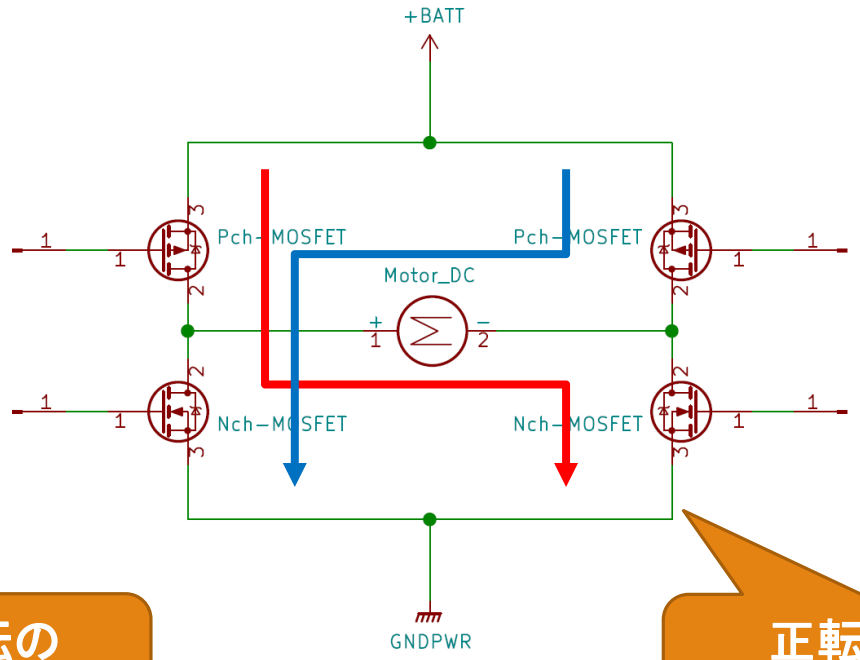
DCモータの速度制御の基本

SMBとLAPのイメージ



SMB方式

正転or反転の
パワーを変える



LAP方式

正転・反転の
パワーの比を変える

DCモータの速度制御の基本

SMBとLAPでは，
それぞれ，メリット／デメリットがある．

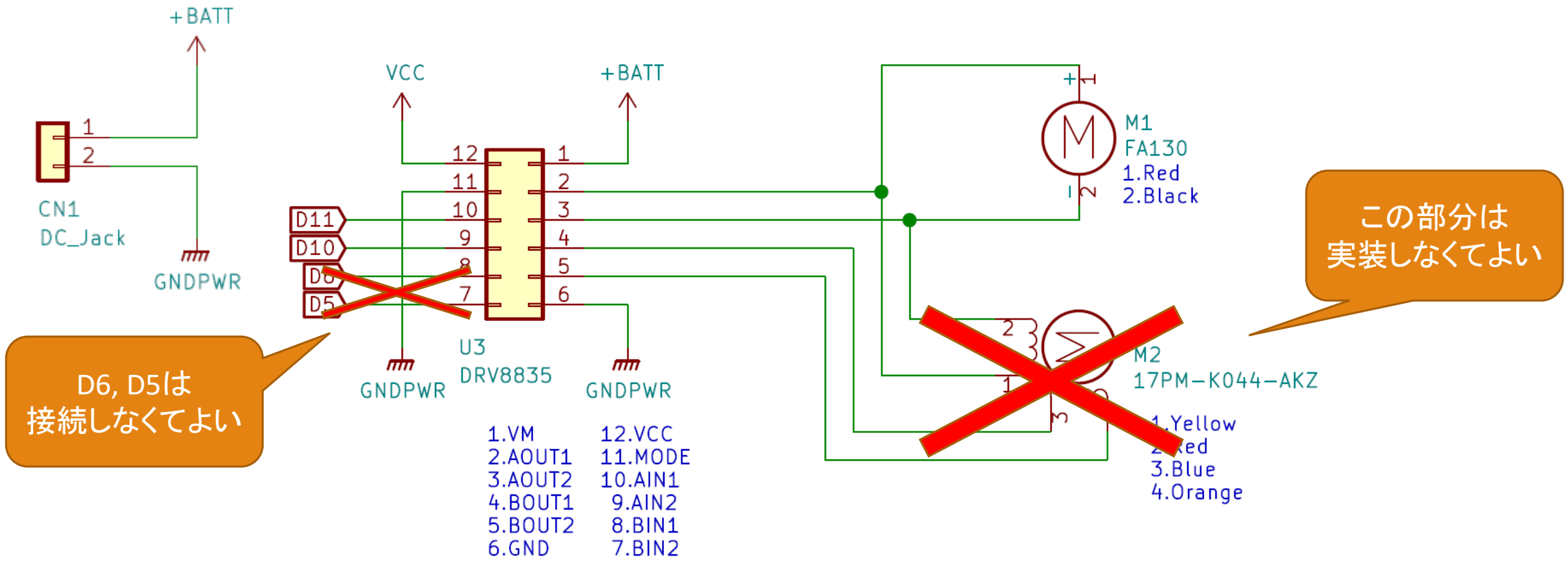
LAP方式では，
DCモータの基本制御式を、ほぼそのまま利用することができる．
しかし，停止時にもSMBに比べて電力を消費する．

SMB方式では，
消費電力が少なく済む．
しかし，DCモータのLAPに比べて基本制御式が成り立ちません．

	SMB	LAP
制御性	△	○
消費電力	○	△

DCモータの制御（SMB方式）

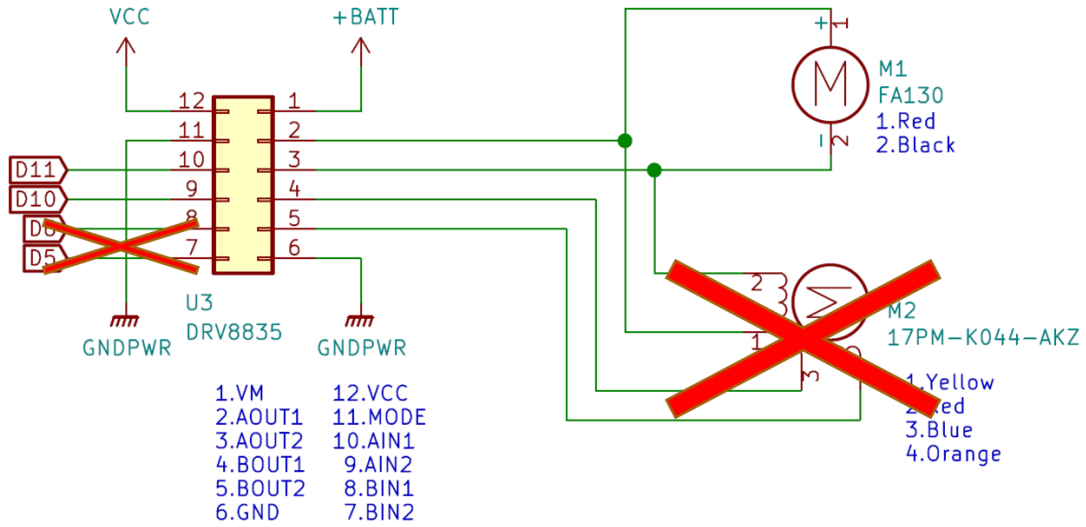
・PWMを使用して，モータの回転方向と速度制御をしてみましょう.



DCモータの制御（SMB方式）

・右の回路を実装して，SMB方式の速度制御をしてみましょう．また，analogWriteの数値を変えて，速度がどう変化するか確かめましょう．

```
#define MOTOR_RED   11    // DCモータ赤線に対応したピン
#define MOTOR_BLACK 10    // DCモータ黒線に対応したピン

void setup() {
  pinMode(MOTOR_RED, OUTPUT);
  pinMode(MOTOR_BLACK, OUTPUT);
}

void loop() {
  // 正転
  analogWrite(MOTOR_RED, 128);     // duty比 50%
  analogWrite(MOTOR_BLACK, 0);     // duty比  0%
  delay(2000);

  // 反転
  analogWrite(MOTOR_RED, 0);       // duty比  0%
  analogWrite(MOTOR_BLACK, 128);   // duty比 50%
  delay(2000);
}
```



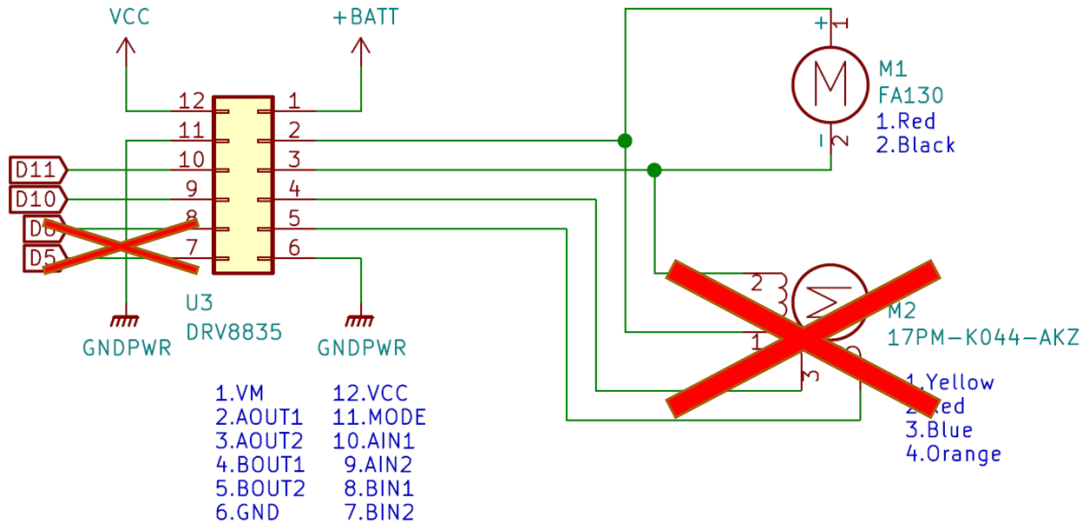
DCモータの制御（SMB方式）

・analogWrite(pinNum, duty)
出力ピンからPWMを出力する関数
pinNum　　: ピン番号
duty　　: duty比（0～255）

DCモータの制御（LAP方式）

・右の回路を実装して，LAP方式の速度制御をしてみましょう．また，dutyの数値を変えて，速度がどう変化するか確かめましょう．

```
#define PWM_PERIOD  1000  // PWM周期

#define MOTOR_RED   11    // DCモータ赤線に対応したピン
#define MOTOR_BLACK 10    // DCモータ黒線に対応したピン

void setup() {
  pinMode(MOTOR_RED, OUTPUT);
  pinMode(MOTOR_BLACK, OUTPUT);
}

void loop() {
  int duty = 500;
  digitalWrite(MOTOR_RED, HIGH);
  digitalWrite(MOTOR_BLACK, LOW);
  delayMicroseconds(duty);                  // 赤線のON時間
  digitalWrite(MOTOR_RED, LOW);
  digitalWrite(MOTOR_BLACK, HIGH);
  delayMicroseconds(PWM_PERIOD - duty); // 黒線のON時間
}
```



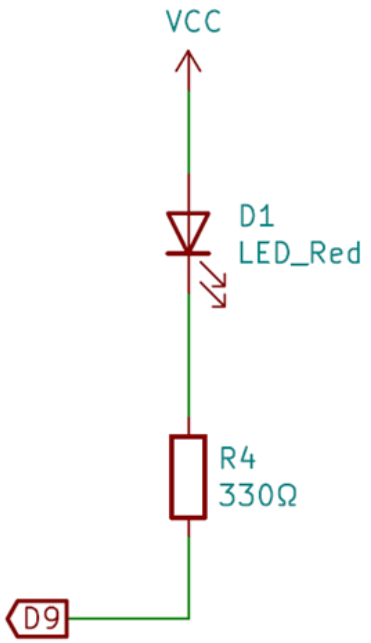
DCモータの制御（LAP方式）

・delayMicroseconds(waittime);
処理停止関数
waittime　：待機時間[μs]

シリアル通信（UART;受信）

PCから一文字を取得して，”a”だとLEDを点灯，”b”だとLEDを消灯させてみましょう.

```
#define LED_PIN 9

void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  char receiveData;
  // 受信文字があるときループ
  while(Serial.available()){
    // 受信データ 1文字読み込み
    receiveData = Serial.read();
    if(receiveData == 'a'){
      digitalWrite(LED_PIN, LOW);
    }
    else if(receiveData == 'b') {
      digitalWrite(LED_PIN, HIGH);
    }
  }
}
```



シリアル通信（UART;受信）

・Serial.available()
受信したバイト数（文字数）を取得する関数
返り値　　：受信バイト数（文字数）

・Serial.read()
受信データを1バイト読み込み
返り値　　：受信した1バイト（1文字）

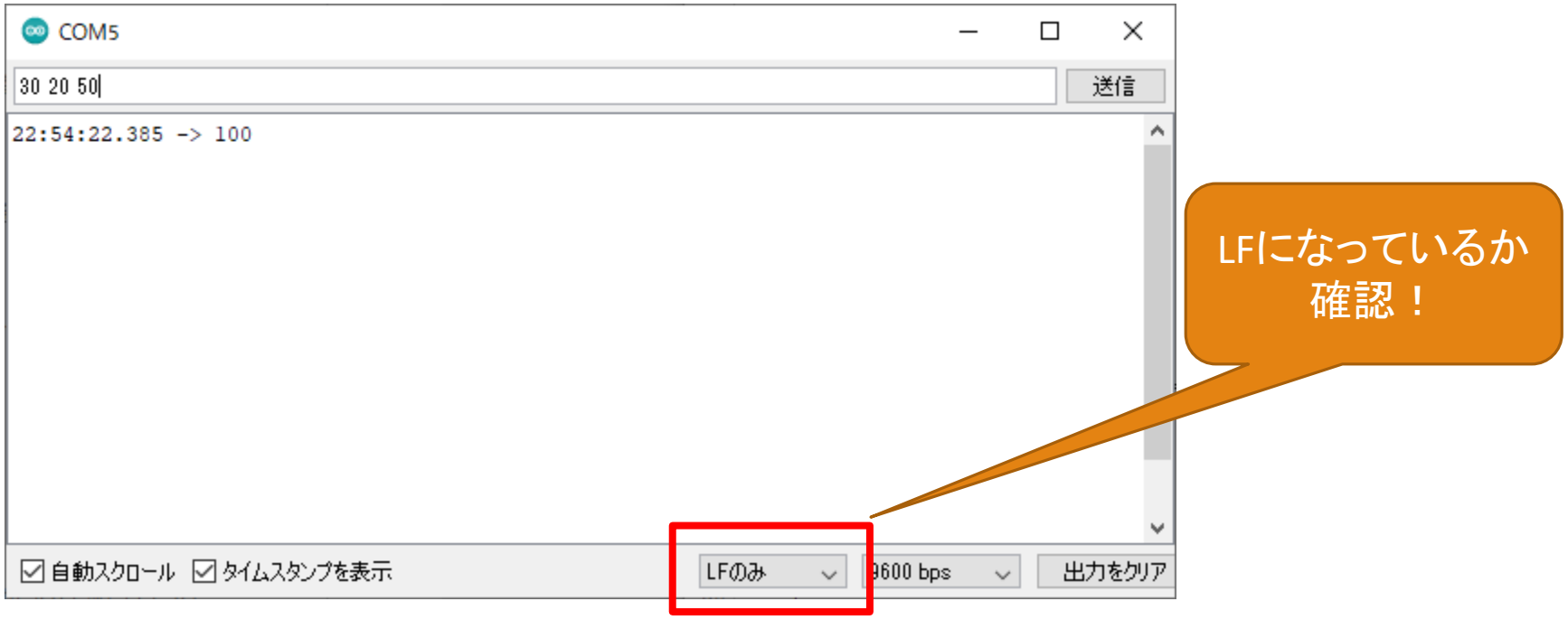
シリアル通信（UART;受信）

3つの数値を空白区切りで受信して，その加算結果を送信してみましょう.

```
int recvIndex = 0;          // 受信データ格納インデックス
char recvData[32] = "";     // 受信データ格納用

void setup() {
  Serial.begin(9600);
}

void loop() {
  int a, b, c;

  // 受信文字があるときループ
  while(Serial.available()){
    // 受信データ取得
    char key = Serial.read();

    // 改行文字受信
    if(key == '\n'){
      // 受信データを数値に変換
      sscanf(recvData, "%d %d %d", &a, &b, &c);
      Serial.println(a+b+c);
      memset(recvData, 0, sizeof(recvData));
      recvIndex = 0;
    }
    else{
      // 受信データ保存
      recvData[recvIndex] = key;
      recvIndex++;              // 次の受信データは次の要素
    }
  }
}
```

シリアル通信（UART;受信）

以下のようになれば，OK.



シリアル通信（UART;受信）

＜C言語の機能（標準関数）＞

・sscanf(str, “format”, *val1, *val2, …)
文字列を入力としたscanf関数
str　　：文字列
format　　：入力のフォーマット
val　　：データを格納する変数のアドレス

・memset(*buf, set, size)
メモリセット関数（setを配列bufの要素に格納する関数）
*buf　　：保存先の先頭アドレス
set　　：格納するデータ
size　　：バイト数（要素数）

シリアル通信（UART;受信）

＜C言語の機能（予約語）＞

・sizeof(*val)
変数のバイト数を取得する
val　：サイズを取得したい変数もしくは型

シリアル通信（UART;受信）

PCから文字列を取得して，”LED”の受信（改行文字込み）後に
”ON”を受信するとLEDを点灯，”OFF”だとLEDを消灯させてみましょう.

```
#define LED_PIN 4            // LEDピン

int phase = 0;               // 受信データ判定用
int recvIndex = 0;           // 受信データ格納インデックス
char recvData[32] = "";      // 受信データ格納用

void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // 受信文字があるときループ
  while(Serial.available()){
    // 受信データ取得
    char key = Serial.read();

    // 改行文字受信
    if(key == '\n'){
      switch(phase){
      case 0:  // コマンド取得;
         if(strcmp(recvData, "LED") == 0) phase = 1;  // 次の文字列はLED制御コード
         break;

      case 1:  // LED制御コード取得
        if(strcmp(recvData, "ON") == 0) digitalWrite(LED_PIN, LOW);
        if(strcmp(recvData, "OFF") == 0) digitalWrite(LED_PIN, HIGH);
        phase = 0;                                    // コマンド取得に戻す
      }

      // 受信データ初期化
      memset(recvData, 0, sizeof(recvData));
      recvIndex = 0;
    }
    else{
      // 受信データ保存
      recvData[recvIndex] = key;
      recvIndex++;              // 次の受信データは次の要素
    }
  }
}
```

シリアル通信（UART;受信）

＜C言語の機能（標準関数）＞

・strcmp(str1, str2)
文字列比較関数
返り値　　: 0=一致, 正値=str1 > str2, 負値=str1 < str2
str1, str2　: 文字列

演習

・PCからDCモータを制御したい．回転方向とduty比を改行区切りで送信し，回転方向コードを，”CW”のとき正転，”CCW”のとき反転とし，duty比コードを0～255として制御せよ．

入力例）　CW 55　　⇒ DCモータ正転 duty=55

　　CCW 10　　⇒ DCモータ反転 duty=10

・DCモータとLEDを同時に制御したい．LEDの制御コードを0だと消灯，1だと点灯とし，モータの制御コードを-255～255で送信し，負であれば反転，正であれば正転で数値の絶対値のduty比で制御せよ．

入力例）　0 -100　　⇒ LED消灯　DCモータ反転 duty=100

　　1 25　　⇒ LED点灯　DCモータ正転 duty=25