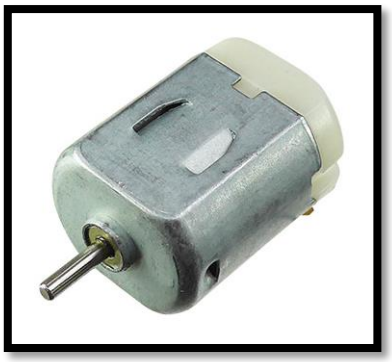
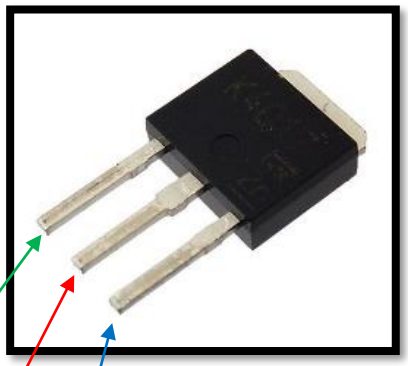
DC モーター（DirectCurrentMotor）



DC モーターとは、直流電圧を印加した電流により発生する電磁力を利用して、一方向に軸を回転させることができる電子部品のこと。
乾電池で駆動させることができ、低価格なことから小型扇風機や電動歯ブラシなど様々な用途で電化製品に使用されている。
今回は MOSFET を使用して回転・停止の切り替えを行う。

MOSFET（MetalOxideSemiconductorFieldEffectTransistor）

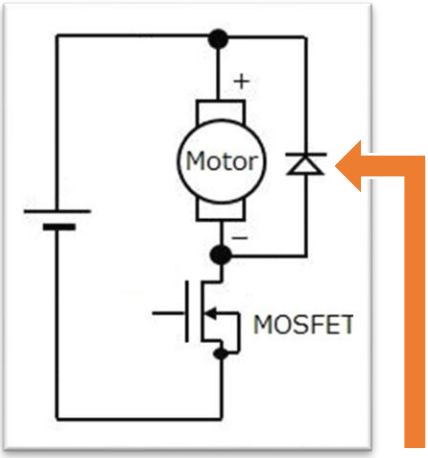


MOSFET とは、電圧を加えることで電子回路の接続と切り離しが行える電子部品のこと。
モーターは駆動する為に必要な電流値が多い為、Raspberry Pi や ESP32 の GPIO からでは駆動させられる電流値を確保できない。
その為、モーターには別の電源から電力を供給し、その回路上に挟んだ MOSFET を制御することで、十分な電流値を確保しつつモーターの制御を行うことができる。

MOSFET（N チャネル型）の制御方法

MOSFET には 3 つのピンが付いており、それぞれゲート・ドレイン・ソースと呼ぶ。
ゲートに電圧を加える（HIGH にする）ことで、ドレイン(+)とソース(-)間が接続されて電流が流れる。
逆に電圧を加えない（LOW にする）ことで、回路が切り離されて電流を止めることができる。

逆起電力とダイオード



モーター内を回転させた際、フレミングの右手の法則により逆方向に電力が生じる現象を逆起電力という。
モーター回路を MOSFET で切り離した際、逆起電力による電位差により、高電圧が一瞬だけ発生する。
その高電圧が MOSFET に加わることによって MOSFET の最大電圧定格値を超えた場合、耐え切れずに壊れてしまう。

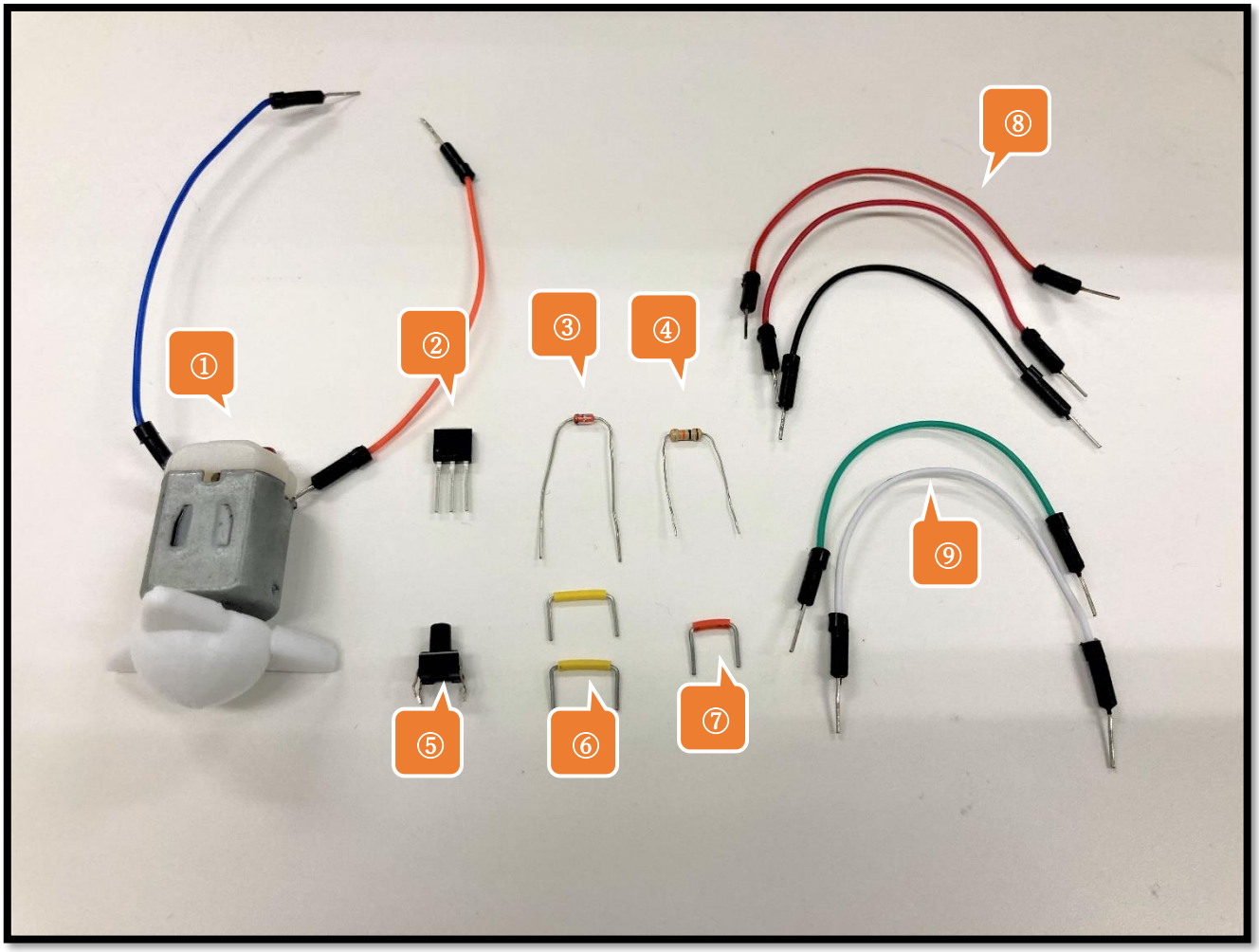
その逆起電力の対策としてダイオードを使用する。
ダイオードをモーター回路に対して並列かつ極性を逆に繋ぎ、モーター内に残った電力をダイオードにショートさせる。
そうすることで逆起電力による高電圧が発生しなくなる為、MOSFET が壊れる心配もなくなる。

機材準備

今回使用する機材を所定の位置から借りて持って来る。※授業終了時には必ず元の場所に戻すこと。

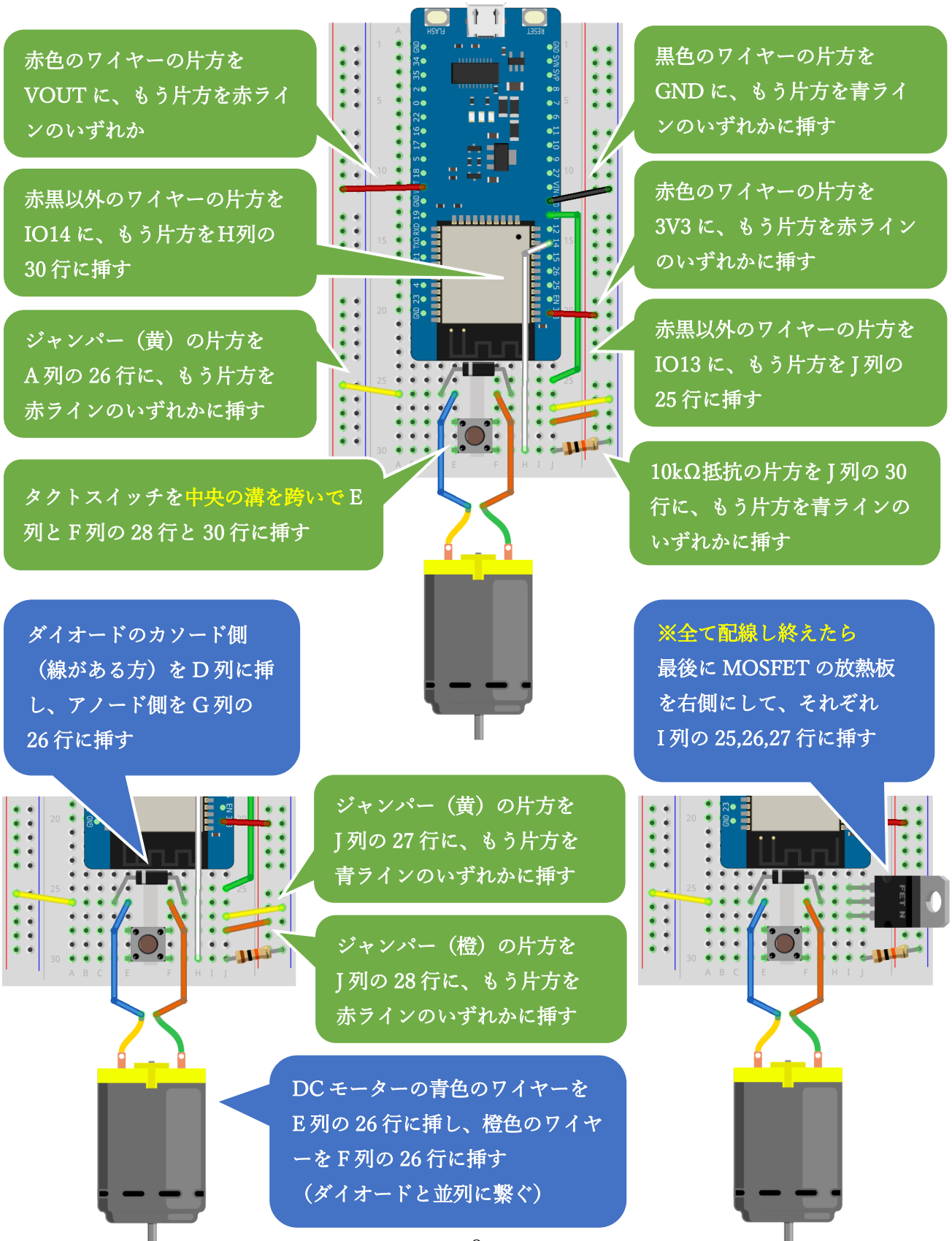
① DC モーター（プロペラ付）
② MOSFET
③ ダイオード
④ 10kΩ抵抗（茶・黒・橙・金）
⑤ タクトスイッチ
⑥ ブレッドボード用ジャンパー（黄）を２本
⑦ ブレッドボード用ジャンパー（橙）を１本
⑧ ジャンパーワイヤー（赤を２本と黒を１本）
⑨ ジャンパーワイヤー（赤・黒以外を２本）

※数が多いので落とさない＆無くさないように注意

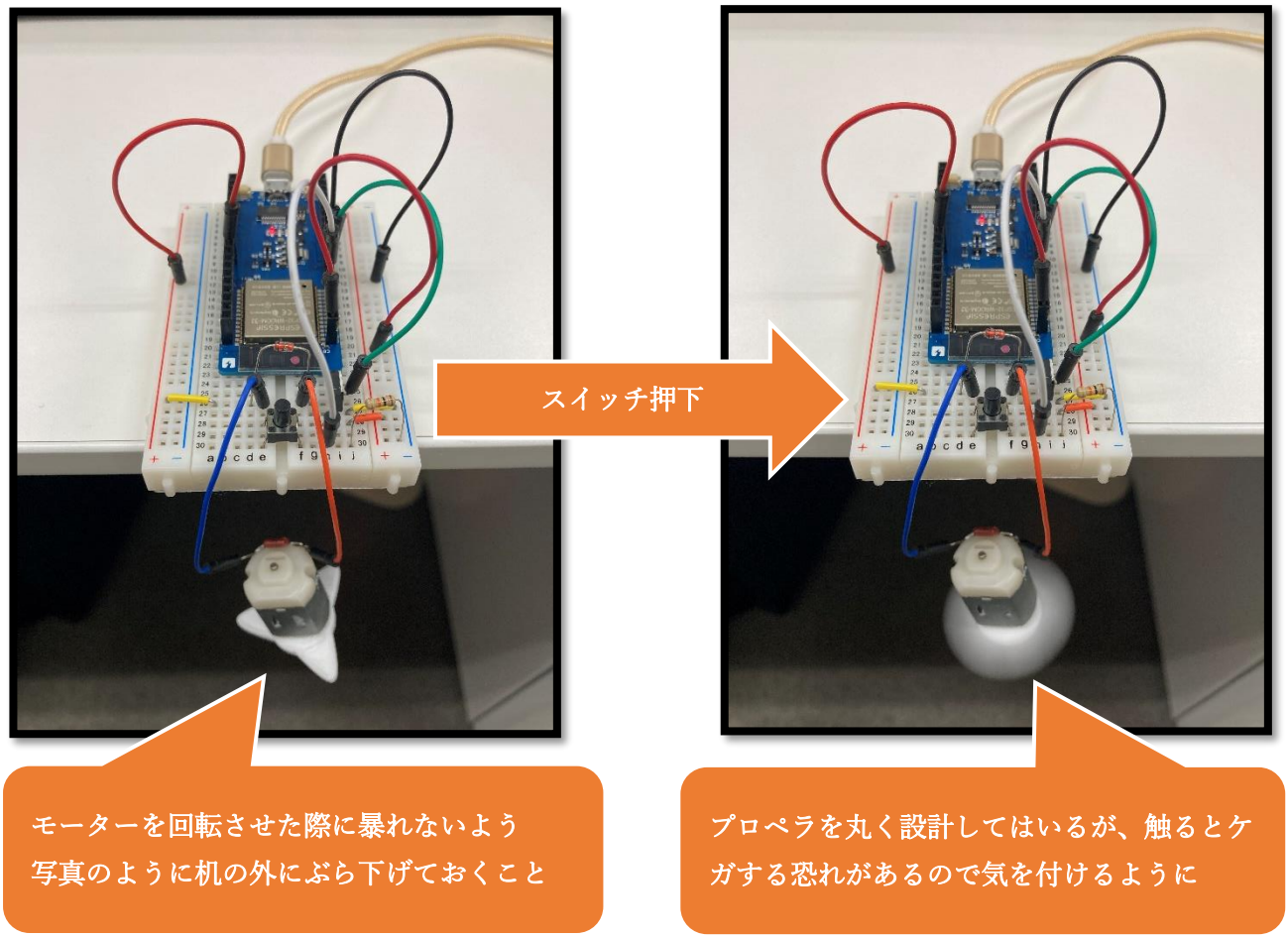


練習（Motor）

ESP32 に Micro-B USB ケーブルを差し込まずに以下の図のとおりに配線を行いなさい。



配線を行った後、Micro-B USB ケーブルを ESP32 と実習機に繋ぎ通電する。
その後、以下のサンプルコードを記述し、ESP32 に書き込みを行う。
スイッチを押す度に 切 → 弱 → 中 → 強 → 切 と扇風機のようにモーターの回転速度が変更され、現在の状態がシリアルモニタ上に表示されるか確かめる。



```
/*
  Motor
  Date : 2022/01/01
  Author : IE1A 99 K.Murakami
*/

// ピン番号をマクロで定義
#define SW_PIN 14
#define MOTOR_PIN 〜 省略 〜

#define LEDC_CHANNEL_1   1    // LEDC の PWM チャネル 0 から 15
#define LEDC_TIMER_8_BIT 8    // LEDC タイマーの精度 8 ビット
#define LEDC_BASE_FREQ   100  // LEDC のベース周波数 100Hz

boolean swData;              // SW の状態を保持する変数
boolean swDataOld = false;  // 前回の SW の状態を保持する変数（初期値は false）


// モーターの回転速度
int motorSpeed[] = {0, 30, 50, 70};
String speedMsg[] = {"切", "弱", "中", "強"}; // 速度文字
int speedIndex = 0; // 回転速度配列要素番号

// 起動時に一度だけ呼び出されるメソッド
void setup() {
  Serial.begin(115200); // シリアル通信の転送レート（bps）を設定

  // ピンの入出力設定
  pinMode(SW_PIN, INPUT);

  // モーターチャネル設定
  ledcSetup(LEDC_CHANNEL_1, LEDC_BASE_FREQ, LEDC_TIMER_8_BIT);
  // モーターのチャネルにピンを接続
  ledcAttachPin(MOTOR_PIN, LEDC_CHANNEL_1);
}
```

モーターを制御する MOSFET のゲートの IO 番号を指定

出力を上げすぎると ESP32 が壊れる可能性があるので
回転速度はこの値から絶対に変更しないこと！

```
// メインループメソッド
void loop() {
  // 現在の SW の状態を読み取る
  swData = digitalRead(SW_PIN);
  // 現在の SW の状態が、前回の SW の状態と違う場合
  if (swData != swDataOld) {
    // 現在の SW の状態が ON(true)の場合
    if (swData) {
      // 要素番号を次へ更新（0→1→2→3→0）
```

この部分の処理は自分で考えて記述すること

```
      ledcWrite(LEDC_CHANNEL_1, motorSpeed[speedIndex]); // モーター回転
      Serial.println(speedMsg[speedIndex]);
    }
  }

  // 次回の為に現在の SW の状態を保存する
  swDataOld = swData;

  delay(10);  // チャタリング防止
}
```

課題（MotorFan）

前述の練習を別名保存し、サーミスタから 1 秒間隔で温度を取得し、その温度が 28℃以上の場合は強、28℃未満~25℃以上の場合は中、25℃未満~22℃以上の場合は小、22℃未満の場合は切に風量が自動的に切り替わる疑似的な風量調節機能付き小型扇風機のプログラムに変更しなさい。
サーミスタの回路を追加する必要がある為、追加で必要な部品を考えて所定の位置から持って来ること。
代わりにスイッチの回路は不要になる為、タクトスイッチは取り除いて良い。
配線を行う際は Micro-B USB ケーブルを ESP32 に差し込まずに行うこと。
また、以下の 2 つの条件を満たすこと。
① DC モーターの回路は練習のまま変更しないこと
② 温度を取得する getTemperature メソッドを作成し、メインループから呼び出して使用すること。

※課題が完成した人は ESP32 にプログラムを書き込んだ状態で講師を呼び、チェックしてもらうこと。