26 点灯

キャンドルLED

デジタル出力

プログラム candle_led.ino

ライブラリ －

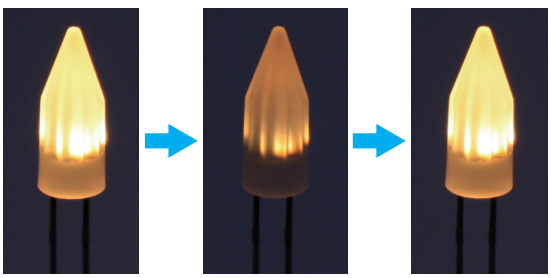
利用パーツ

●キャンドルLED×1（秋：111585） ●トランジスタ×1（秋：106477）
●抵抗器3.3kΩ×1（秋：125332）

ロウソクのように、明るくなったり暗くなったりする形でLEDを点灯したい場合、プログラムでPWMの出力をランダムに大きくしたり小さくしたりすることで実現できます。光を揺らがせるなど、よりリアルに見せたいなら、そうした工夫もプログラムに盛り込んでLEDの点灯制御をし続ける必要があり、手間がかかります。

「キャンドルLED」を利用すると、電源をつなぐだけでロウソクのような光り方でLEDが点灯します（**下図**）。Arduino側からは点灯するか消灯するかを制御するだけで済み、プログラムを単純化できます。

図　ロウソクのように揺らいで光るキャンドルLED



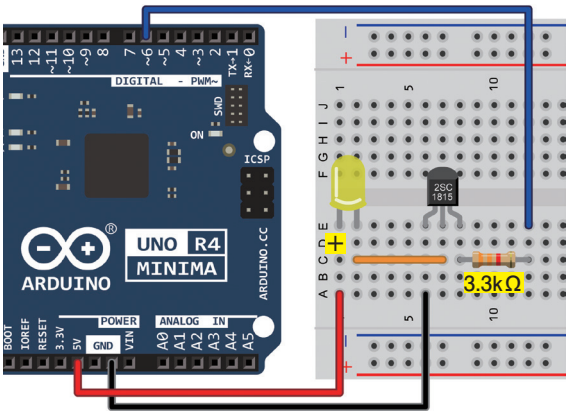
（著者撮影）

ロウソクのように明るくなったり暗くなったりする

キャンドルLEDをArduinoから点灯制御するには**下図**のように接続します。ここで利用しているキャンドルLEDは5Vで動作するため、トランジスタを利用して5Vの電源を接続しています。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　キャンドル LED の接続図



以下に示すプログラム「candle_led.ino」を実行すると、1 分間点灯したあと、消灯します。

ソース　キャンドル LED の点灯、消灯を制御するプログラム

candle_led.ino

```
const int LED_PIN = 6; ── キャンドルLEDを接続したGPIOの番号

void setup() {
    pinMode( LED_PIN, OUTPUT );
}

void loop() {
    digitalWrite( LED_PIN, HIGH );
    delay( 60000 );

    digitalWrite( LED_PIN, LOW );

    while( true ){
      delay( 1000 );
    }
}
```

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

27 動作

振動モーター

デジタル出力

プログラム vib.ino
ライブラリ ―

利用パーツ

●振動モーター×1（秋：106784） ●FET×1（秋：107597） ●ダイオード×1（秋：100934）
●電池ボックス×1（秋：100327） ●コンデンサー 0.1μF×1（秋：110147）
●抵抗器1kΩ×1（秋：125102）、20kΩ×1（秋：103940）

振動モーターは、小さな振動を起こすことができる電子パーツです。スマホのバイブレーション機能のように、音を鳴らしたくない場面において、代わりに振動で通知するといった使い方ができます。

ここで紹介する振動モーター「LA3R5-480DE」は、回転軸に対して重りが偏って取り付けられています（**下図**）。この重りが回転すると、振動を引き起こします。

図　振動モーターの外観と振動する仕組み

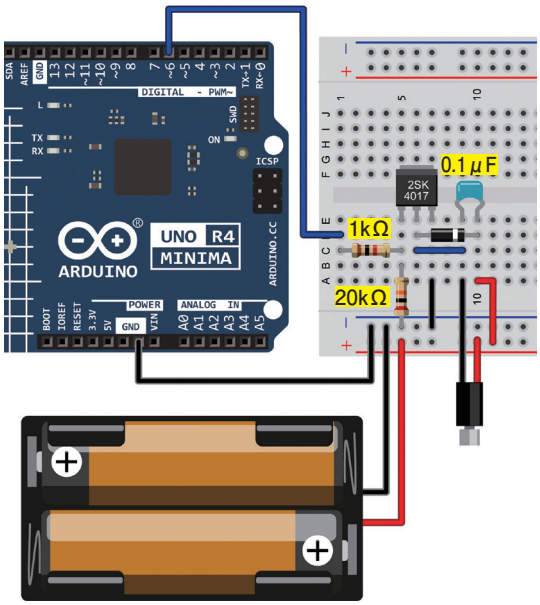


（日経BP撮影）

振動モーターは小型のモーターですが、比較的大きな電流が流れます。LA3R5-480DEの場合、3Vで駆動すると100mAの電流が流れます。このため、DCモーター同様にFETを介して制御します（**下図**）。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



デジタル出力をHighにすると回転し、Lowにすると停止します。スマホのバイブレーション機能と同じように、プログラムによって所定のパターンで振動させることが可能です。以下に示すプログラム「vib.ino」では、PATTERNに設定したパターンで振動します。パターンは振動している時間、停止している時間、と交互に指定します。時間の単位は秒です。

ソース　振動モーターを所定のパターンで振動させるプログラム

vib.ino

```
const int M_PIN = 6;

int PATTERN[] = { 1, 1, 2, 1, 3 };
int pattern_num = sizeof( PATTERN )
 / sizeof( PATTERN[0] );

void setup() {
    pinMode( M_PIN, OUTPUT );
    digitalWrite( M_PIN, LOW );
}

void loop() {
  int output = 1, i;
  for ( i = 0; i < pattern_
num; i++ ){
    digitalWrite( M_PIN, output );
    delay( PATTERN[ i ] * 1000 );
    if( output == 1 ){
      output = 0;
    }else{
      output = 1;
    }
  }
  digitalWrite( M_PIN, LOW );

  while( true ){
    delay( 1000 );
  }
}
```

振動時間(秒)

停止時間(秒)

振動のパターンを設定する

パターンに従って振動させる

28 動作

ステッピングモーター

デジタル出力

プログラム stepping.ino

ライブラリ ー

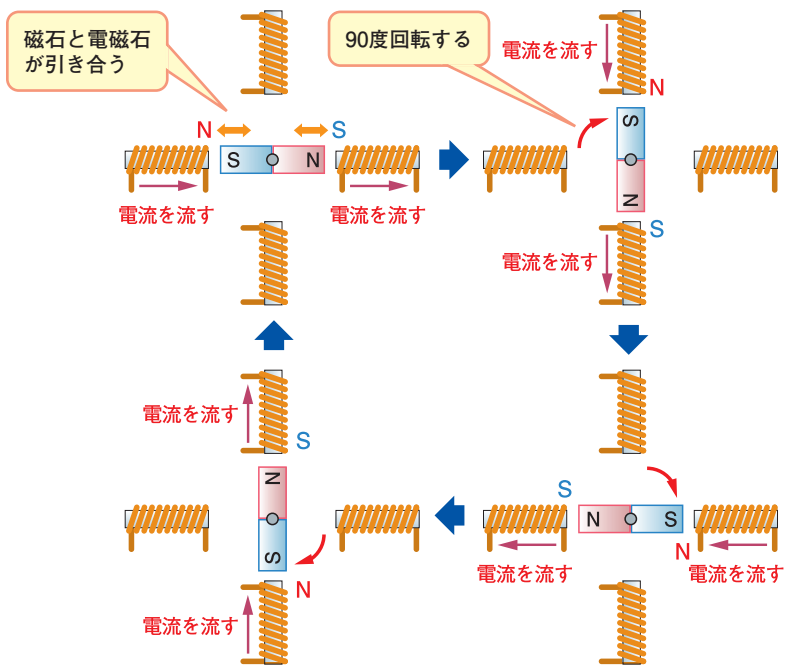
利用パーツ

●ステッピングモーター×1（秋：105372） ●モータードライバー×1（秋：109848）
●ACアダプター 5V×1（秋：111996） ●DCジャック×1（秋：108849）

本書第4章の4-3で紹介したDCモーターは、回転するか停止するかを制御できますが、「指定した回数だけ回転したら停止する」といった正確な制御はできません。正確に回転を制御したい場合には「ステッピングモーター」を利用します。

ステッピングモーターは、**下図**のような原理で動作します。左右に配置した電磁石に電流を流すと磁石が横向きになります。次に上下の電磁石に電流を流すと、磁石が90度回転します。この手順を繰り返すことで、回転数や角度を正確に制御しながら回転させられます。実際の動作はもっと細かく、1回の切り替えで1度程度回転します（製品により異なります）。

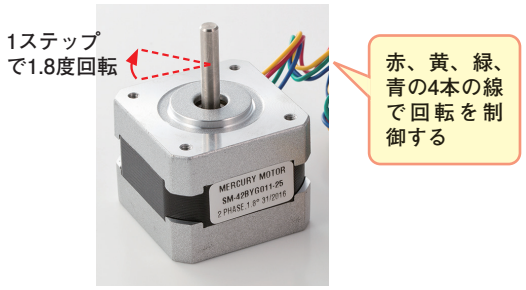
図　ステッピングモーターの動作原理



@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

ステッピングモーターには色違いの複数の線が付いています。ここに電源の＋極と－極を接続することで内部の電磁石の磁極を制御し、軸を回転させられます。ここで紹介するステッピングモーター「SM-42BYG011」は、赤、青、黄、緑の４本の線を備えています（**下図**）。なお、他のステッピングモーターでは配線の色が異なることがあります。この場合は、データシートなどを確認し、接続してください。

図　ステッピングモーターの外観



（日経BP撮影）

SM-42BYG011を回転させるには、各線に**下図**のように電源を接続します。Highは電源の＋極、Lowは－極を表します。時計回りに回転（正転）させるには、左表のようにまず赤、黄をHigh、緑、青をLowにします（①）。次に黄、緑をHigh、赤、青をLowにすると１ステップ分回転します（②）。SM-42BYG011の場合は、1ステップで1.8度回転します。続いて③の状態にするともう１ステップ回転します。このように、①～④の順に接続状態を切り替えることで１ステップずつ回転させられます（④まで達したら①に戻ります）。200回切り替えれば1回転になります（1.8度×200回＝360度）。

一方、右の表のように順次切り替えれば反時計回り（逆転）に回転させられます。

図　回転制御方法

順序	赤	黄	緑	青
①	High	High	Low	Low
②	Low	High	High	Low
③	Low	Low	High	High
④	High	Low	Low	High

時計回り（正転）

順序	赤	黄	緑	青
①	High	High	Low	Low
②	High	Low	Low	High
③	Low	Low	High	High
④	Low	High	High	Low

反時計回り（逆転）

ステッピングモーターには「バイポーラー型」と「ユニポーラー型」の２種類があります（**下図**）。バイポーラー型は、電磁石が一つのコイルで構成されており、電流を流す向きによって磁

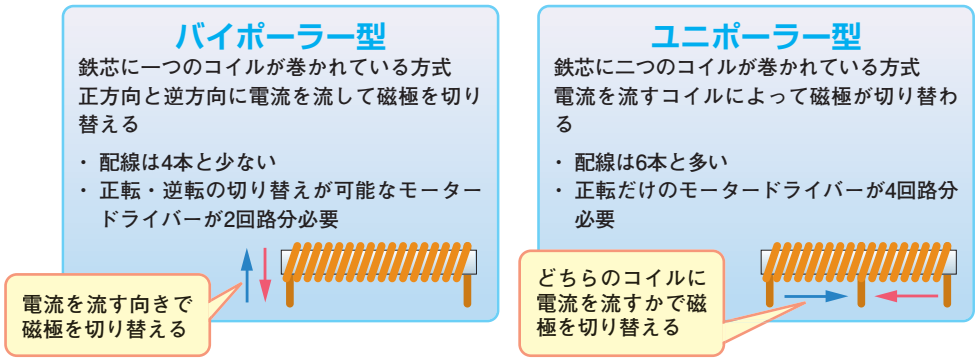
@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

極を切り替えます。このため本書第5章のNo.6で紹介した「フルブリッジドライバー」のような電流の向きを逆にできるドライバーICが必要です。

一方のユニポーラー型は、電磁石が二つのコイルで構成されています。どちらか一方に電流を流すことで磁極を切り替えます。流すか流さないかを切り替えるだけなのでフルブリッジドライバーを使わずに動かせます。半面、動かすには正転だけのモータードライバーが4回路分必要になります。

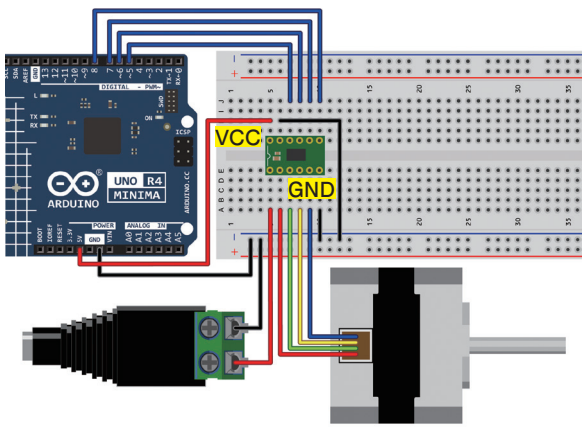
ここで利用するSM-42BYG011はバイポーラー型のステッピングモーターです。ユニポーラー型では回路の接続方法や制御方法が異なるので、購入する際には注意しましょう。

図　ステッピングモーターの種類



SM-42BYG011とArduinoは**下図**のように接続します。このステッピングモーターは配線の赤と緑、青と黄のそれぞれに電磁石が接続されており、それぞれを別に制御します。このため、2系統を備えるモータードライバーのDRV8835を使って制御できます。モーターの制御電源としてACアダプターを利用して5Vを供給するようにしています。

図　接続図



@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

ステッピングモーターを制御するプログラム「stepping.ino」を以下に示します。プログラム中の「STEPS」で指定したステップだけ回転し、その後停止します。「SPEED」の値を増減することで回転速度を変えられます。これは1ステップごとの待機時間（秒）で、値が大きいほど遅く、小さいほど速く回転します。「DIRECT」は回転方向です。1にすると正転、0にすると逆転します。

ソース　指定したスピードと回転方向でステッピングモーターを制御するプログラム

stepping.ino

```
const int ST_MOTOR_A0 = 5;
const int ST_MOTOR_A1 = 6;
const int ST_MOTOR_B0 = 7;
const int ST_MOTOR_B1 = 8;
   モータードライバーに接続したGPIOの番号
const int STEPS = 200;
const int SPEED = 10;
const int DIRECT = 1;
回転方向。1は正転、0は逆転
const uint8_t ST_MOTOR_OUT[ 4 ] ={
    0b1100,     1ステップごとの待機時間
    0b0110,
    0b0011,       回転させるステップ数
    0b1001
};

int all_step = 0;
int finish = 0;

void setup() {
    Serial.begin( 115200 );
    pinMode( ST_MOTOR_A0, OUTPUT );
    pinMode( ST_MOTOR_A1, OUTPUT );
    pinMode( ST_MOTOR_B0, OUTPUT );
    pinMode( ST_MOTOR_B1, OUTPUT );
}

void loop() {
    int i = 0;
    int step_count = 0;
    while ( i < 4 ){
        if ( DIRECT  == 1 ){
            step_count = i;
        } else {
            step_count = 4 - 1 - i;
        }
        int a0 = ST_MOTOR_OUT[ step_cou
nt ] & 0b1000;
        int a1 = ST_MOTOR_OUT[ step_cou
nt ] & 0b0010;
        int b0 = ST_MOTOR_OUT[ step_cou
nt ] & 0b0100;
        int b1 = ST_MOTOR_OUT[ step_cou
nt ] & 0b0001;

        Serial.print( all_step );
        Serial.print( " : " );
        Serial.print( a0 );
        Serial.print( b0 );
        Serial.print( a1 );
        Serial.println( b1 );

        digitalWrite( ST_MOTOR_A0, a0 );
        digitalWrite( ST_MOTOR_A1, a1 );
        digitalWrite( ST_MOTOR_B0, b0 );
        digitalWrite( ST_MOTOR_B1, b1 );

        delay( SPEED );
        i = i + 1;
        all_step = all_step + 1;
        if ( STEPS <= all_step ){
            finish = 1;
            break;
        }
    }

    if ( finish == 1 ){
        while( true ){
            delay( 10000 );
        }
    }

}
```

29 動作

連続回転サーボモーター

PWM出力

プログラム rotservo.ino

ライブラリ Servo

利用パーツ

●連続回転サーボモーター ×1（秋：113206）

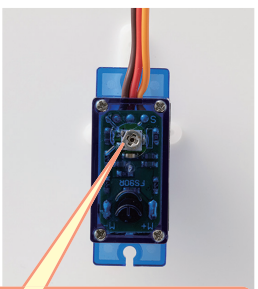
「連続回転サーボモーター（ローテーションサーボモーター）」は、DCモーターのように連続して回転するサーボモーターです。DCモーターの代わりとして、模型の車を動かす目的などで利用できます（**下図**）。

連続回転サーボモーターは、通常のサーボモーターと同じ形状をしており、回転軸にホーンを付けて作品などに取り付けます。ホーンのほかに、回転軸に直接差し込めるタイヤも販売されています（秋：113207）。

図　連続回転サーボモーターの外観



連続回転サーボモーターの外観



パルス幅を1500μ秒にした場合に回転してしまう場合は、ボリュームを調節して停止するようにする

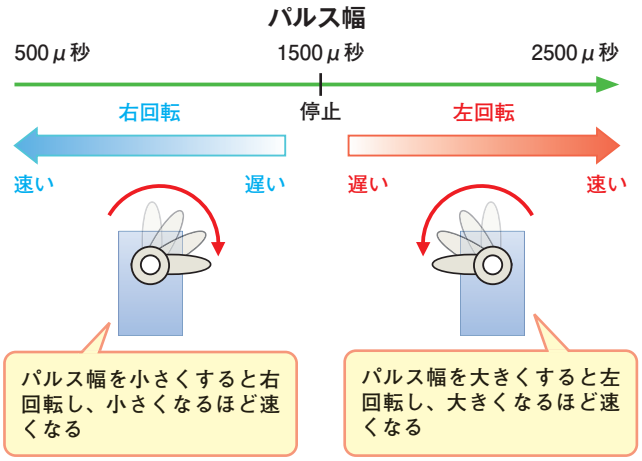
（日経BP撮影）

回転の制御には、通常のサーボモーターと同じPWM出力を使います。ここで利用する「FS90R」は、パルス幅を1500μ秒にすると停止し、それよりもパルス幅を大きくすると左回転、小さくすると右回転します（**下図**）。

回転速度の調節も可能で、1500μ秒に近い値だとゆっくり、1500μ秒より値が離れるほど速くなります。

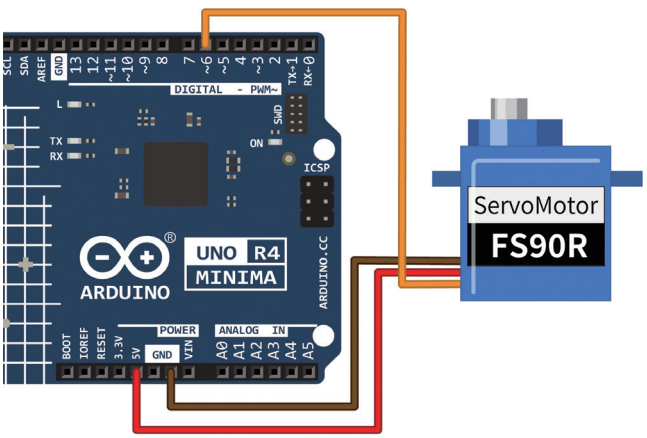
@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　連続回転サーボモーターの制御方法



Arduinoとの接続方法も通常のサーボモーターと変わりません（**下図**）。制御線をArduinoのGPIOに接続し、電源は5V端子に接続します。

図　接続図



連続回転サーボモーターの制御プログラム「rotservo.ino」を以下に示します。一つ前に紹介したサーボモーターと同様に、Servoライブラリを利用し、writeMicroseconds()でPWMのパルス幅をマイクロ秒単位で指定します。

プログラムを実行すると、右にゆっくり回転、右に速く回転、停止、左にゆっくり回転、左に速く回転、停止を繰り返します。

もし、停止（パルス幅が1500μ秒）のところで回転が止まらない場合は、製品の個体差が原因です。底面にあるボリュームを回して停止するように調節してください。

ソース　連続回転サーボモーターの速度と回転方向を制御するプログラム

rotservo.ino

```
#include <Servo.h>

const int SERVO_PIN = 6; ――信号線を接続したGPIOの番号

Servo mservo;

void setup() {
  mservo.attach( SERVO_PIN );
}

void loop() {
  mservo.writeMicroseconds( 1400 ); ――ゆっくり右回転
  delay( 1000 );
  mservo.writeMicroseconds( 1000 ); ――速く右回転
  delay( 1000 );
  mservo.writeMicroseconds( 1500 ); ――0度
  delay( 1000 );
  mservo.writeMicroseconds( 1600 ); ――ゆっくり左回転
  delay( 1000 );
  mservo.writeMicroseconds( 2000 ); ――速く左回転
  delay( 1000 );
  mservo.writeMicroseconds( 1500 ); ――停止
  delay( 1000 );
}
```


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

30 動作

ソレノイド

デジタル出力

プログラム solenoid.ino

ライブラリ ー

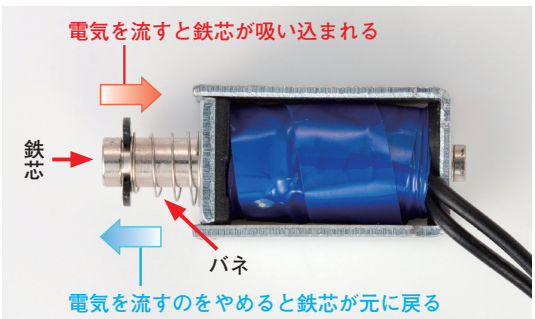
利用パーツ

●ソレノイド×1（秋:110761） ●FET×1（秋:107597） ●ダイオード×1（秋:100934）
●電池ボックス×1（秋:100327） ●コンデンサー 0.1μF×1（秋:110147）
●抵抗器1kΩ×1（秋:125102）、20kΩ×1（秋:103940）

「ソレノイド」は直線的な動作を発生させるための電子パーツです（**下図**）。例えば、ボタンを押したり、ドアをロックしたりする目的で使われます。

コイルの中に鉄芯が入っており、コイルに電気を流すと発生した磁力により鉄芯が引き込まれます。電気を流すのをやめると鉄芯に取り付けられているバネの力で元の位置に戻ります。

図　ソレノイドの外観

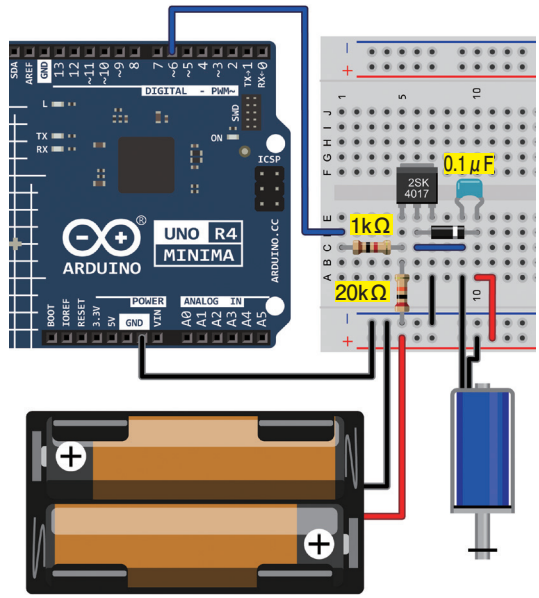


（日経BP撮影）

ソレノイドを動作させるには大きな電流が必要です。ここで紹介する「ZHO-0420S-05A4.5」は、数百mAの電流を必要とします。このため、DCモーター同様にFETを介して制御します。Arduinoも含めた接続方法を**下図**に示します。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



ソレノイドを制御するプログラム「solenoid.ino」を以下に示します。Arduinoのデジタル出力をHighにするとソレノイドの鉄芯が吸い込まれ、Lowにすると元に戻ります。このプログラムでは、鉄芯を10秒間吸い込んだ状態にし、その後元に戻しています。

ソース　ソレノイドの鉄芯を10秒間吸い込むプログラム

solenoid.ino

```
const int S_PIN = 6; ── ソレノイドを接続したGPIOの番号

void setup() {
    pinMode( S_PIN, OUTPUT );
}

void loop() {
    digitalWrite( S_PIN, HIGH );
    delay( 10000 );

    digitalWrite( S_PIN, LOW );

    while( true ){
      delay( 1000 );
    }
}
```


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

31 表示

バーLED

デジタル出力

プログラム barled.ino

ライブラリ －

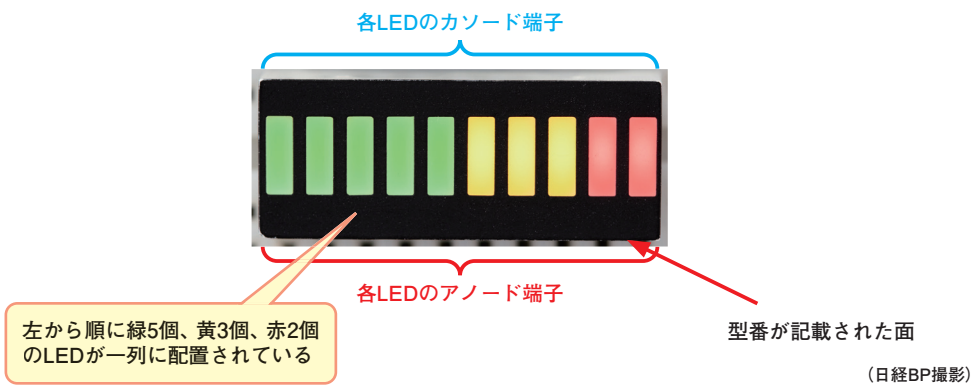
利用パーツ

●バー LED×1（秋：105347） ●抵抗器470Ω×10（秋：125471）

「バーLED」は、複数のLEDを一つの部品として詰め込む形で一列に配置した電子パーツです。音量の表示用インジケーターや、進捗を通知するプログレスバーなどの用途で使われます。ここでは左から順に緑色5個、黄色3個、赤色2個の計10個のLEDが配置された「OSX10201-GYR1」（**下図**）を使う方法を紹介します。

OSX10201-GYR1には、LEDごとにアノードとカソード端子が付いています。型番が記載されている面にある端子がアノードです。

図　10個のLEDが一列に配置されたバーLEDの外観

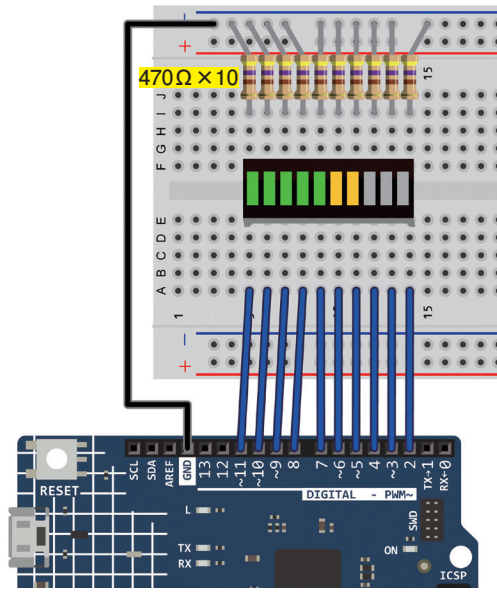


（日経BP撮影）

それぞれのLEDをArduinoのGPIOに接続し、デジタル出力で制御します（**下図**）。この際、制限抵抗を忘れないように接続してください。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



指定した値で点灯するLEDの数を変えられるように制御するプログラム「barled.ino」を以下に示します。プログラム中では、「BAROUT」変数に0～10の数値を指定すると、緑色側のLEDから指定した数のLEDが点灯します。例えば、3の場合は3個の緑のLEDが点灯し、7の場合は緑5個、黄色2個のLEDが点灯します。

ソース　指定した数の LED を点灯するプログラム

barled.ino

```
int BAROUT = 7;
const int LED_PIN[ 10 ] = { 11, 10, 9, 8, 7, 6, 5, 4, 3, 2 };
void setup() {
    for( int pin : LED_PIN ){
        pinMode( pin, OUTPUT );
    }
}

void loop() {
    int count;
    for( int pin : LED_PIN ){
        if( count < BAROUT ){
            digitalWrite( pin, HIGH );
        }else{
            digitalWrite( pin, LOW );
        }
        count++;
    }
    delay( 10000 );
}
```

指定した数だけLEDが点灯する

各LEDに接続したGPIOの番号

OUTPUTに指定した数分のLEDを点灯する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

32 入力

マイクロスイッチ

デジタル入力

プログラム microsw.ino

ライブラリ ―

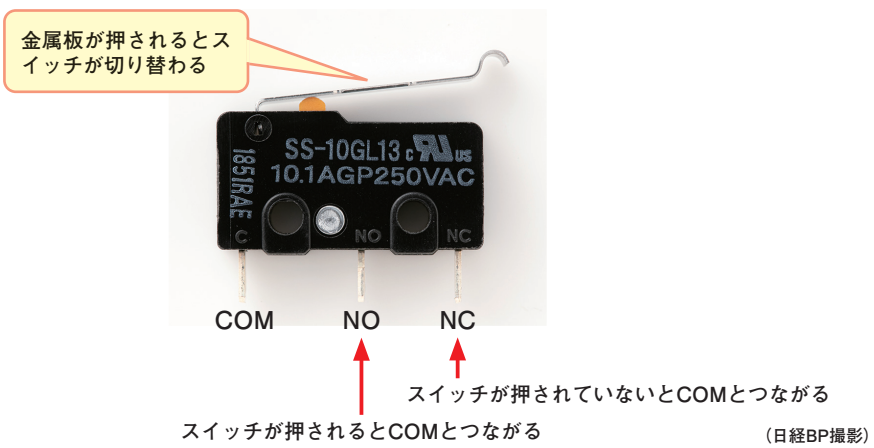
利用パーツ

●マイクロスイッチ×1（秋：114659）

マイクロスイッチは、高い信頼性と耐久性を備えたスイッチです。所定の力が加わると、確実にスイッチが切り替わるようになっています。マウスのボタンや、動く作品が壁に衝突したことを検知する仕組みなどに利用されています。金属製のレバーが付いているタイプのマイクロスイッチを使うと、スイッチとして反応する面が広くなる上、てこの原理により、弱い力でもスイッチを押すことができるようになります。

ここで紹介するマイクロスイッチは「COM」「NO」「NC」という三つの端子を備えています（**下図**）。NO（Normally Open）はスイッチを押していない状態では何も接続されず、スイッチを押すとCOMと接続された状態になります。一方、NC（Normally Close）はNOと逆で、スイッチを押していない状態ではCOMと接続され、スイッチが押されると何も接続しない状態になります。用途に応じてNOとNCのどちらを利用するかを選択します。

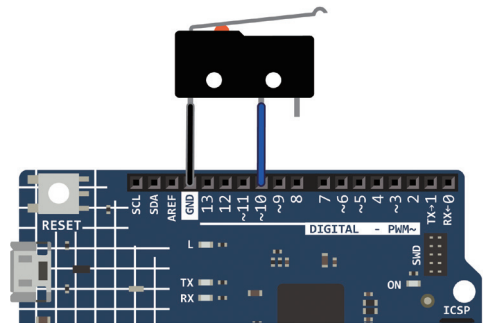
図　マイクロスイッチの外観



（日経BP撮影）

Arduinoとマイクロスイッチは**下図**のように接続します。ここではNO端子を利用し、マイクロスイッチが押されたらCOM（電源）と接続するようにしています。

図　接続図



制御用プログラム「microsw.ino」を以下に示します。このプログラムでは、マイクロスイッチが押されると「Hit」と表示します。また、マイクロスイッチを接続したGPIOについて、プルダウン抵抗が有効になるように設定しています。

ソース　マイクロスイッチが押されたら「Hit」と表示するプログラム

microsw.ino

```
const int SW_PIN = 10; ── マイクロスイッチを接続したGPIOの番号

void setup() {
    pinMode( SW_PIN, INPUT_PULLUP ); ── プルアップ抵抗を有効にする

    Serial.begin( 9600 );
}

void loop() {
    if( digitalRead( SW_PIN ) == 0 ){ ┐
        Serial.println( "Hit" ); ──── ┴── 押されたら「Hit」と表示する
    }

    delay( 500 );
}
```


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

33 入力

リードスイッチ

デジタル入力

プログラム reed.ino

ライブラリ －

利用パーツ

●ドアスイッチ×1（秋：113371）

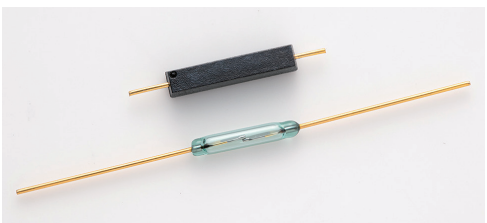
「リードスイッチ」は、磁石を近づけることでスイッチを切り替えることのできる電子パーツです（**下図**）。センサー自体に触れずに感知できるため、動くものの状態を調べたり、密封した空間内の状態を調べたりできます。ドアや窓などの開閉の状態を知りたい場合にも役立ちます。

図の右側に示したように、リードスイッチの内部には、2枚の金属板が離れた状態で配置されています。磁石を近づけると、金属板がN極またはS極になります。N極とS極は引き合うので金属板がくっつき、導通状態になります。図の左側下のガラス管を使った透明のリードスイッチを見ると、内部に2枚の金属版が配置されている様子を確認できます。

図　リードスイッチの外観と仕組み

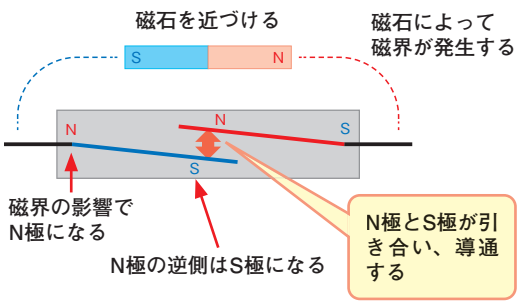
リードスイッチの外観

上はプラケース入り（秋：104209）、下はガラス管入り（秋：103676）のタイプ。



（日経BP撮影）

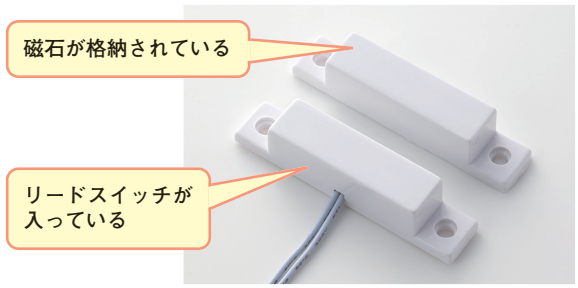
リードスイッチの仕組み



ドアなどの開閉状態を調べる目的で作られた、「ドアスイッチ」と呼ばれるリードスイッチの応用製品も市販されています（**下図**）。ドアスイッチは、二つの部品に分かれています。一つにはリードスイッチが、もう一つには磁石が入っており、二つの部品を近づけることでリードスイッチがオンの状態になる仕組みです。利用する際は、ドアの枠にリードスイッチ側を取り付けて配線し、ドア側に磁石を取り付けておきます。ドアを閉めると磁石がリードスイッチに近づき、スイッチがオンになります。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

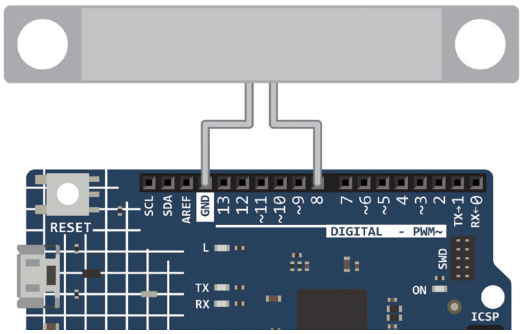
図　ドアの開閉を調べるのに使える「ドアスイッチ」の外観



（日経BP撮影）

リードスイッチとArduinoは**下図**のように接続します。リードスイッチがオンになると、ArduinoのGPIOはLowに接続された状態になります。

図　接続図



制御用プログラム「reed.ino」を以下に示します。ドアスイッチを接続したGPIOはプルアップ抵抗を有効にしておきます。プログラムを実行すると、ドアが開いていれば「Open.」、閉まっていれば「Close.」と表示されます。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

ソース　ドアスイッチを使って扉の状態を確認するプログラム

reed.ino

```
const int SW_PIN = 8; ── ドアスイッチを接続したGPIOの番号

void setup() {
    pinMode( SW_PIN, INPUT_PULLUP ); ── プルアップ抵抗を有効にする

    Serial.begin( 9600 );
}

void loop() {
    if( digitalRead( SW_PIN ) == 0 ){ ── ドアが閉じているかを確認する
        Serial.println( "Close." ); ── ドアが閉じていたら表示
    }else{
        Serial.println( "Open." ); ── ドアが開いたら表示
    }

    delay( 500 );
}
```

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

34 入力

デジタル入力

2進数出力 ロータリースイッチ

プログラム rotlog.ino

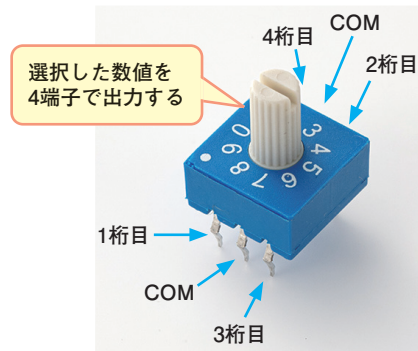
ライブラリ －

利用パーツ

●ロータリースイッチ0～9正論理×1（秋：102274）

ロータリースイッチには、ダイヤル（つまみ）で選択した数値を出力する「正論理」と呼ぶタイプの製品もあります（**下図**）。合わせた数値によって内部のスイッチが切り替わり、各桁の端子の状態が変化します。

図　2進数出力ロータリースイッチの外観



（日経BP撮影）

ここで利用するロータリースイッチは、0～9の数値を選択でき、選んだ数値を4本の端子の出力状態で表現します（**下表**）。表中の「オフ」は何もつながっていない状態、「オン」はCOM端子とつながった状態です。

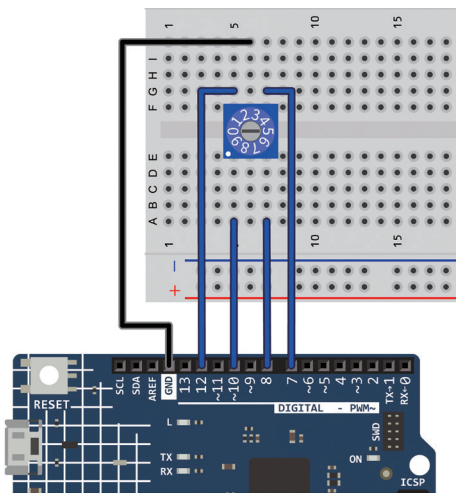
@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

表　各数値に合わせた際のロータリースイッチの端子の状態

数値	4桁目	3桁目	2桁目	1桁目
0	オフ	オフ	オフ	オフ
1	オフ	オフ	オフ	オン
2	オフ	オフ	オン	オフ
3	オフ	オフ	オン	オン
4	オフ	オン	オフ	オフ
5	オフ	オン	オフ	オン
6	オフ	オン	オン	オフ
7	オフ	オン	オン	オン
8	オン	オフ	オフ	オフ
9	オン	オフ	オフ	オン

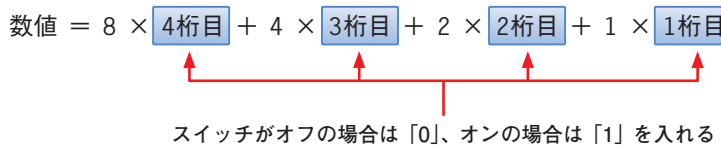
Arduinoで数値を読み取るには、**下図**のようにCOM端子をGND、各桁の端子をGPIOに接続します。各GPIO端子はプルアップ抵抗を有効にしておきます。

図　接続図



プログラムで数値を読み取るには、各桁の端子に接続したArduinoのGPIOの状態を調べ、スイッチがオンの場合は「1」、スイッチがオフの場合は「0」とします。取得した値を**下図**の計算式に当てはめると、どの数値を選択したかが分かります。

図　数値を求める計算式



ただし、回路図のようにCOMにGNDを接続して、GPIOの端子でプルアップ抵抗を有効にすると、スイッチがオフの場合は「1」、オンの場合は「0」と値が反転してしまいます。そこで、取得したGPIOの入力を反転して、オフの場合は「0」、オンの場合は「1」に変換しておきます。1から入力の値を引くことで変換できます。

作成した数値取得用プログラム「rotlog.ino」を以下に示します。「ROT_PIN」には各桁の端子に接続したGPIOの番号を指定しておきます。

ソース　ロータリースイッチで合わせた数値を取得するプログラム

rotlog.ino

```
const int ROT_DIG = 4;
int ROT_PIN[ 4 ] = { 10, 7, 8, 12 };
int data[4] = { 0, 0, 0, 0 };
void setup() {
    int i;
    for( i = 0; i < ROT_DIG; i ++ ){
      pinMode( ROT_PIN[ i ], INPUT_PULLUP );
    }

    Serial.begin( 9600 );
}

void loop() {
  int i, value;
  for( i = 0; i < ROT_DIG; i++ ){
    data[ i ] = 1 - digitalRead( ROT_PIN[ i ] );
  }
  value = data[ 3 ] * 8 + data[ 2 ] * 4 + data[ 1 ] * 2 + data[ 0 ];
  Serial.print( value );
  Serial.print( " (" );
  Serial.print( data[ 3 ] );
  Serial.print( ", " );
  Serial.print( data[ 2 ] );
  Serial.print( ", " );
  Serial.print( data[ 1 ] );
  Serial.print( ", " );
  Serial.print( data[ 0 ] );
  Serial.println( ")" );
  delay( 1000 );
}
```

ロータリースイッチの端子数
各桁に接続したGPIOの番号
各桁の状態を格納しておく配列
接続したGPIOでプルダウン抵抗を有効にする
各桁の状態を読み取る
計算して数値を求める
ロータリースイッチの合わせている数値と各桁の状態を表示する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

35 入力

振動スイッチ

デジタル入力

プログラム fallsw.ino

ライブラリ ー

利用パーツ

●振動スイッチ×1（秋：102349）

「振動スイッチ」は、直立状態か転倒状態かによってオンとオフが切り替わるスイッチです（**下図**）。暖房器具が転倒したことを検知するなどの用途があります。

振動スイッチの内部には金属球が入っており、上下方向に自由に動きます。このため、直立状態では底面に配置された端子に金属球が触れて端子間が導通していますが、転倒すると金属球が端子から離れて導通しなくなります。

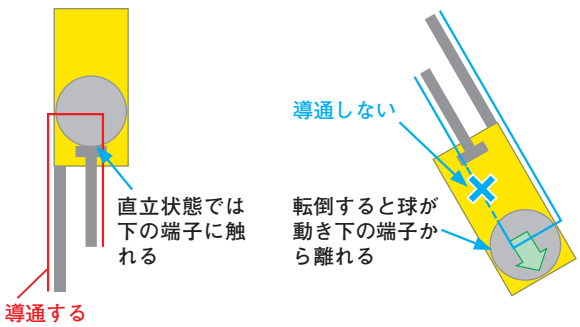
図　振動スイッチの外観と仕組み

振動スイッチの外観



（日経BP撮影）

振動スイッチの仕組み

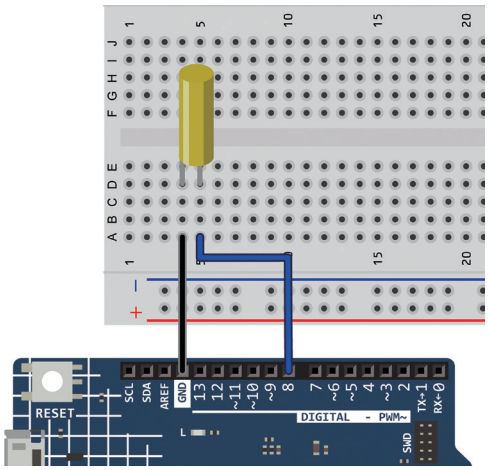


直立の状態　　転倒した状態

振動スイッチとArduinoは、**下図**のように接続します。直立時には振動スイッチがオンの状態となっているため、GPIOはGNDと接続されています。転倒するとオフとなり、GPIOには何も接続されていない状態になります。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



転倒しているかどうかを表示するプログラム「fallsw.ino」を以下に示します。振動スイッチを接続するGPIOは、プログラム中でプルアップ抵抗を有効にしておきます。

実行すると、直立状態であれば「Stability.」、転倒状態では「Fall down.」と表示します。

ソース　転倒しているかどうかを表示するプログラム

fallsw.ino

```
const int SW_PIN = 8; ── 振動スイッチを接続したGPIOの番号

void setup() {
    pinMode( SW_PIN, INPUT_PULLUP ); ── プルアップ抵抗を有効にする

    Serial.begin( 9600 );
}

void loop() {
    if( digitalRead( SW_PIN ) == 1 ){
        Serial.println( "Fall down." ); ── 転倒した場合に表示
    }else{
        Serial.println( "Stability." ); ── 直立状態の場合に表示
    }

    delay( 500 );
}
```


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

36 入力

デジタルジョイスティック

デジタル入力

プログラム d_joystick.ino

ライブラリ ―

利用パーツ

●デジタルジョイスティック×1（秋：115233）

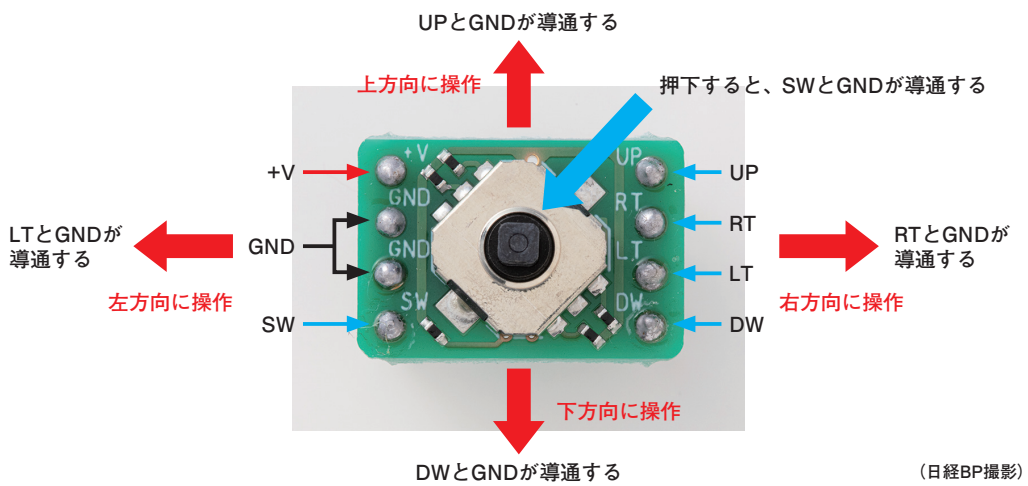
「ジョイスティック」は、搭載するスティックを上下、左右に動かすことで方向の入力ができる電子パーツです。リモコンで動かす模型やロボットアームの方向指示、メニュー画面の操作などに使えます。

ここで紹介するデジタルジョイスティックは、上下左右の4方向にスイッチが入っており、操作すると各スイッチがオンになります（**下図**）。加えて、スティックの根元に押し込むとオンにできる押しボタンスイッチも搭載しています。

各スイッチは一方がGND、もう一方がそれぞれ別々の端子に接続されています。操作すると対象の端子とGNDがつながった状態になります。例えば、上方向に操作するとUP端子とGND端子が導通します。各スイッチの端子をArduinoのGPIOに接続し、デジタル入力がLowになるかどうかを調べることで、スティックの入力方向が分かります。

なお、各スイッチはプルアップ抵抗を搭載しているため、プログラムでArduinoの内蔵プルアップ抵抗を有効にする必要はありません。

図　デジタルジョイスティックの外観

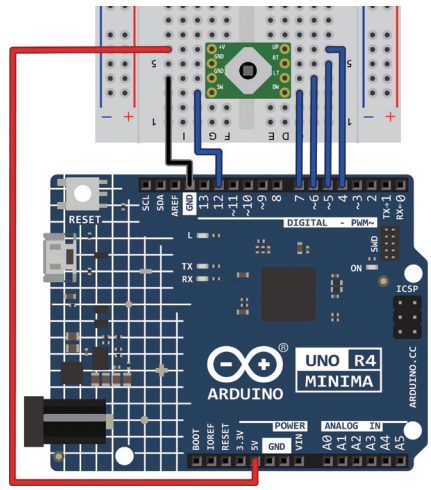


（日経BP撮影）

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

デジタルジョイスティックとArduinoは**下図**のように接続します。各スイッチは、上方向をGPIO 12、下方向をGPIO 15、左方向をGPIO 14、右方向をGPIO 13、押しボタンをGPIO 16に接続しています。

図　接続図



デジタルジョイスティックの操作を読み取るプログラム「d_joystick.ino」を以下に示します。デジタルジョイスティックに接続した各GPIOの状態をチェックし続けて、Lowになったらその方向へのジョイスティックの入力があったと判断し、方向を表示します。

ソース　デジタルジョイスティックの操作を読み取るプログラム

d_joystick.ino

```
const int UP_PIN = 4;
const int DW_PIN = 7;
const int LT_PIN = 6;
const int RT_PIN = 5;
const int SW_PIN = 12;
```

ジョイスティックに接続したGPIOの番号

```
void setup() {
    pinMode( UP_PIN, INPUT );
    pinMode( DW_PIN, INPUT );
    pinMode( LT_PIN, INPUT );
    pinMode( RT_PIN, INPUT );
    pinMode( SW_PIN, INPUT );

    Serial.begin( 9600 );
}
```

```
void loop(){
    if( digitalRead( UP_PIN ) == 0 ){
        Serial.println( "UP" );
    }else if( digitalRead( DW_PIN ) == 0 ){
        Serial.println( "DOWN" );
    }else if( digitalRead( LT_PIN ) == 0 ){
        Serial.println( "LEFT" );
    }else if( digitalRead( RT_PIN ) == 0 ){
        Serial.println( "RIGHT" );
    }else if( digitalRead( SW_PIN ) == 0 ){
        Serial.println( "PUSH" );
    }

    delay( 500 );
}
```


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

37 入力

キーパッド

デジタル入力

プログラム keypad.ino

ライブラリ －

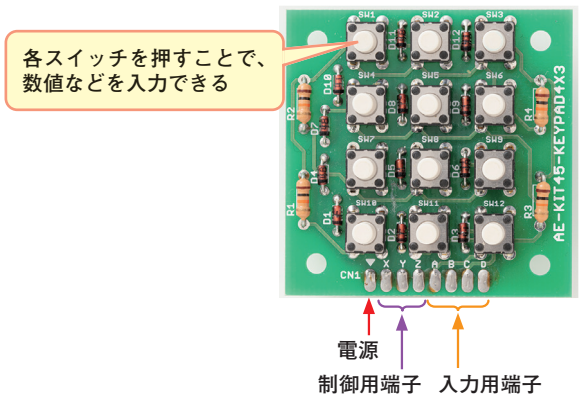
利用パーツ

●キーパッド×1（秋：112229）

「キーパッド」は、複数の押しボタンスイッチ（キー）が配置された電子パーツです。単純に複数個のスイッチとして使えるほか、それぞれのキーに文字などを割り当てることで、PCのキーボードのような文字入力用パーツとしても利用可能です。

ここで紹介するキーパッドは4行3列で合計12個のキーが配置されています（**下図**）。0～9とドット、決定キーを割り当てることで、テンキーのように数字の入力に使えます。

図　複数の押しボタンスイッチを配置したキーパッドの外観



（日経BP撮影）

キーパッドの読み取りには、本書の第5章のNo.8のマトリクスLEDと同じようにダイナミック制御を使います。ここで利用するキーパッドには、読み取り対象列を選択するための3列分の制御用端子（X～Z列）と、押しているキーを判定するための4行分の入力用端子（A～D行）が用意されています（**下図**）。

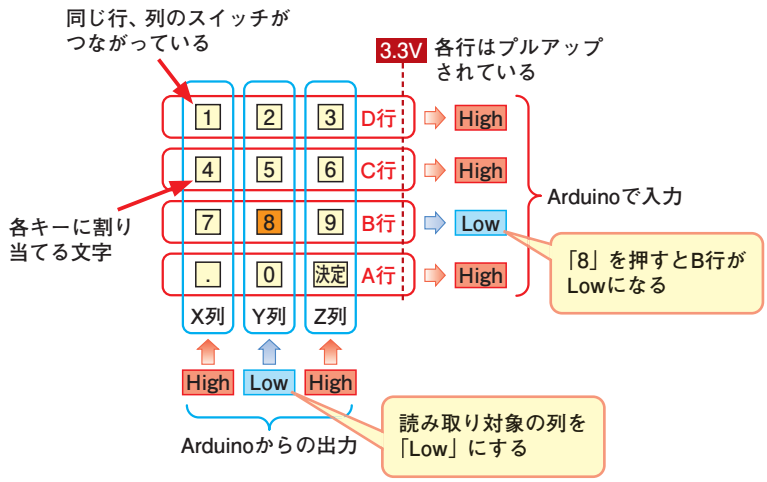
読み取り対象となる列をLow、他の列をHighに設定しておきます。この状態で対象列内のいずれかのキーが押されていると、該当する行の入力用端子がHighからLowに変化します。Arduinoのデジタル入力で各行の状態を読み取ると、どのキーが押されているかが分かります。

この処理をX、Y、Zと順にずらしながら短い時間で切り替えることで、すべてのスイッチの状

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

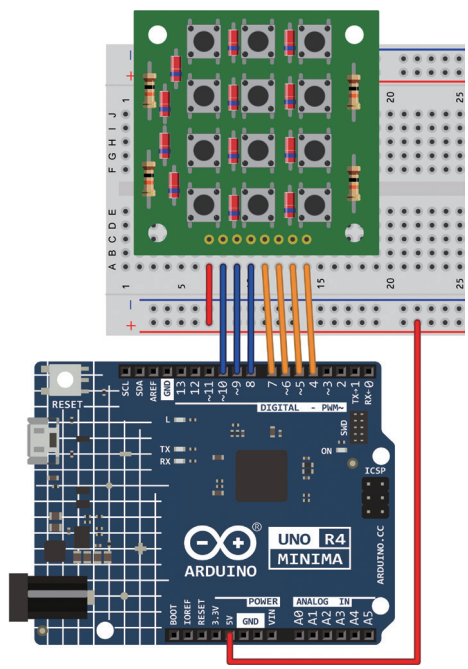
態を確認できます。なお、入力用端子はプルアップ抵抗を備えているため、Arduino側で内蔵プルアップ抵抗を有効にする必要はありません。

図　ダイナミック制御ですべてのキーの状態を読み取る



キーパッドは**下図**のようにArduinoと接続します。行の端子に接続したGPIOはデジタル入力、列の端子に接続したGPIOはデジタル出力として使います。

図　接続図



@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

キーパッドから数値を入力するプログラム「keypad.ino」を以下に示します。プログラム中の「KEY_MAP」で数を割り当てたキーを使って数値を入力し、決定キーを押すと入力した数値が表示されます。ドットは小数点として利用可能です。

ソース　キーパッドから数値を入力するプログラム

keypad.ino

```
const int IN_PIN[] = {7, 6, 5, 4};
const int SEL_PIN[] = {10, 9, 8};

const int NUM_SEL_PINS = 3;
const int NUM_IN_PINS = 4;

const char* KEY_MAP[3][4] = { {".", "7", "4", "1"},
                              {"0", "8", "5", "2"},
                              {"\n", "9", "6", "3"} };

void setup() {
  for (int i = 0; i < NUM_SEL_PINS; i++) {
    pinMode(SEL_PIN[i], OUTPUT);
    digitalWrite(SEL_PIN[i], HIGH);
  }

  for (int i = 0; i < NUM_IN_PINS; i++) {
    pinMode(IN_PIN[i], INPUT);
  }

  Serial.begin(9600);
}

String keybuf = "";

void loop() {
  bool keyPressed = false;

  for (int i = 0; i < NUM_SEL_PINS; i++) {
    digitalWrite(SEL_PIN[i], LOW);
    for (int j = 0; j < NUM_IN_PINS; j++) {
      if (digitalRead(IN_PIN[j]) == LOW) {
        keybuf += KEY_MAP[i][j];
        keyPressed = true;
        while (digitalRead(IN_PIN[j]) == LOW) {
          delay(50);
        }
      }
    }
    digitalWrite(SEL_PIN[i], HIGH);
  }
```

各端子に接続したGPIOの番号

各キーに割り当てる文字

次ページに続く

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

```
  if (keyPressed) {
    if (keybuf.indexOf('\n') > -1) {
      float value = processBuffer();
      Serial.println(value);
      keybuf = "";
    }
  }
}

float processBuffer() {
  String number = "";

  for (unsigned int i = 0; i < keybuf.length(); i++) {
    if (keybuf[i] == '\n') {
      break;
    }
    number += keybuf[i];
  }

  return number.toFloat();
}
```


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

38 センサー

I²C

温湿度センサー

プログラム temphumi.ino

ライブラリ Adafruit SHT31

利用パーツ

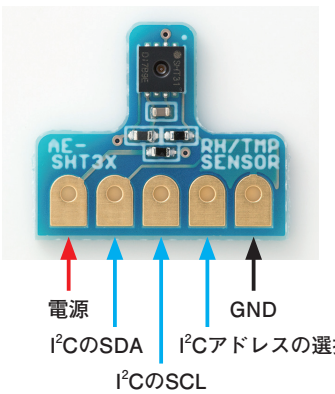
●温湿度センサー ×1（秋：112125）

室内が快適かどうかを知るには、「温湿度センサー」が利用できます。温湿度を計測することで、部屋の快適度を表す「不快指数」が求まります。この指数を基に、エアコンや加湿器を制御して、より快適な状態に近づけるといった応用も可能です。

「SHT31-DIS」は、温度と湿度を同時に計測できるセンサーです（**下図**）。温度は-40～125℃、湿度は0～100%の範囲で計測可能です。精度は、温度が±0.3℃、湿度が±2%となっており、比較的正確と言えるでしょう。なお、SHT31-DISよりも精度が高い「SHT35-DIS」（ス：5337）という製品も販売されています。温湿度の計測可能範囲はSHT31-DISと同じですが、精度が±0.1℃および±1.5%と高くなっています。

SHT31-DISは、計測した温度や湿度をI²C通信を使って取得できます。I²Cアドレスは、0x44、0x45のいずれかを選択可能で、I²Cアドレスの選択端子に何も接続しないと0x45、GNDに接続すると0x44になります。

図　SHT31-DISの外観

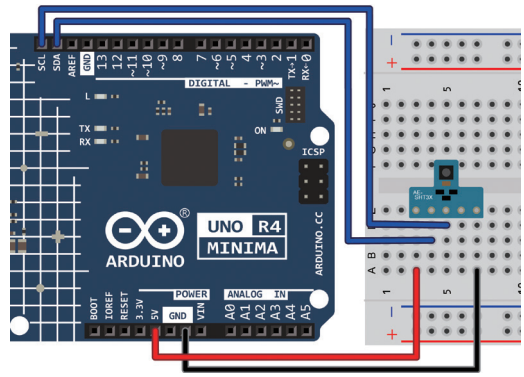


（日経BP撮影）

SHT31-DISとArduinoは**下図**のように接続します。ここでは、I²Cアドレス選択端子に何も接続せず、0x45を使うようにしています。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



SHT31-DISの制御には、Adafruit Industries社が提供するライブラリ「Adafruit SHT31」を利用します。本書第2章の2-2の手順に従ってArduino IDEにライブラリをインストールしておきます。

同ライブラリを使って、SHT31-DISで温度、湿度を定期的に計測する計測プログラム「temphumi.ino」を以下に示します。ライブラリの「sht31.readTemperature()」で温度を、「sht31.readHumidity()」で湿度を取得できます。

ソース　温度、湿度を定期的に計測するプログラム

temphumi.ino

```
#include <Wire.h>
#include "Adafruit_SHT31.h"

Adafruit_SHT31 sht31 = Adafruit_SHT31();

void setup() {
  sht31.begin( 0x45 );
  Serial.begin(9600);
}

void loop() {
    float temp = sht31.readTemperature();
    float humi = sht31.readHumidity();

    Serial.print( "Temp : " );
    Serial.print( temp );
    Serial.print( "C    Humi : " );
    Serial.print( humi );
    Serial.println( "%" );

    delay(1000);
}
```

SHT31-DISを利用できるようにする。I²Cアドレスを指定する。

温度を取得する

湿度を取得する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

39 センサー　SPI

熱電対

プログラム thermocouple.ino
ライブラリ Adafruit MAX31855

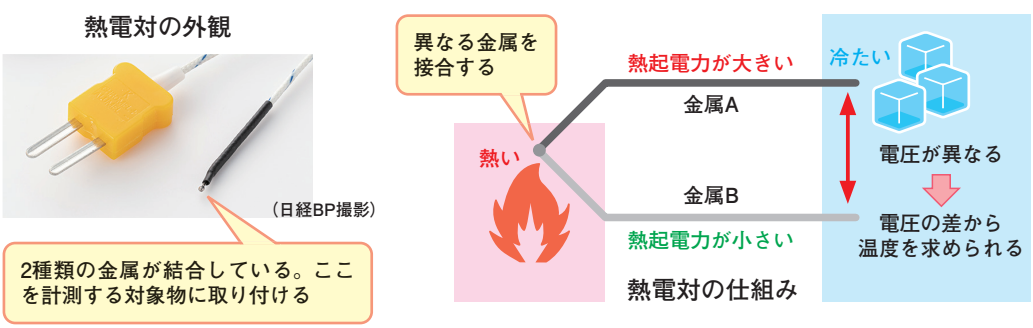
利用パーツ

●熱電対×1（秋：117400）　●熱電対アンプ×1（秋：108218）
●熱電対用コネクタ×1（秋：114971）

「熱電対」は、1000℃を超えるような高温を計測可能な温度センサーです（**下図**）。調理用の鍋や陶芸用の窯、電気炉など、高温かつ温度管理が必要となる場所や器具の温度を計測する目的でよく使われています。

金属は、両端の温度が異なると、内部を電荷が移動します。これを「熱起電力」と呼びます。熱起電力の大きさは金属によって異なります。熱電対は異なる二つの金属を接合したものです。接合した部分を温めると、温度差によりそれぞれの金属で熱起電力が発生し、他方の端との間で電圧が生じます。異なる金属では熱起電力が異なるため、それぞれの金属の端では電圧の差が生じます。この電圧の差を調べることによって温度を求めることができます。

図　熱電対の外観と仕組み



熱電対は、利用する金属によって計測できる温度が異なります。例えば、クロメル（ニッケルとクロムの合金）とアルメル（ニッケルとアルミニウムの合金）という2種類の金属を使った「K型熱電対」は、-270～1372℃の範囲の温度を計測できます。白金（プラチナ）とその合金である白金ロジウム（13%）を接合した「R型熱電対」なら、-50～1768℃とさらに高い温度まで計測可能です（**下表**）。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

表　主な熱電対と計測可能な温度範囲

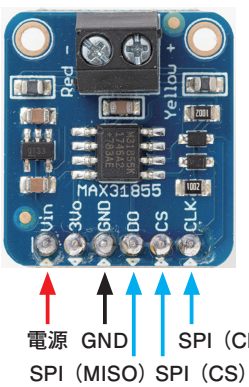
熱電対の型	利用金属（＋側）	利用金属（－側）	計測可能温度範囲*1
K型	クロメル	アルメル	-270～1372℃
N型	ナイクロシル	ナイシル	-270～1300℃
E型	クロメル	コンスタンタン	-270～1000℃
J型	鉄	コンスタンタン	-210～1200℃
T型	銅	コンスタンタン	-270～400℃
B型	白金・ロジウム（30%）	白金・ロジウム（6%）	0～1820℃
R型	白金・ロジウム（13%）	白金	-50～1768℃
S型	白金・ロジウム（10%）	白金	-50～1768℃

＊1　JIS規格で定義されている温度範囲

ここで利用する熱電対（秋月電子通商で販売）は、K型熱電対を用いた製品で、-200～1250℃の温度計測に対応しています。この熱電対は、1℃の温度変化に対して電圧が約40.7 μ V上昇または下降します。この電圧を調べることで温度を計算できます。

ただし、熱電対が出力する電圧は非常に小さいため、そのまま入力したのではArduinoが搭載するADコンバーターでは読み取れません。そこで、熱電対専用のADコンバーターを別途用意して読み取ります。ここでは、Adafruit Industries社の「MAX31855K」を搭載した熱電対用ADコンバーターを利用する方法を紹介します（**下図**）。熱電対から入力された電圧を増幅したあと、14ビットのADコンバーターでデジタル値に変換します。変換した計測値はSPI通信を使ってArduinoで読み取ります。

図　熱電対用ADコンバーター「MAX31855K」搭載モジュール

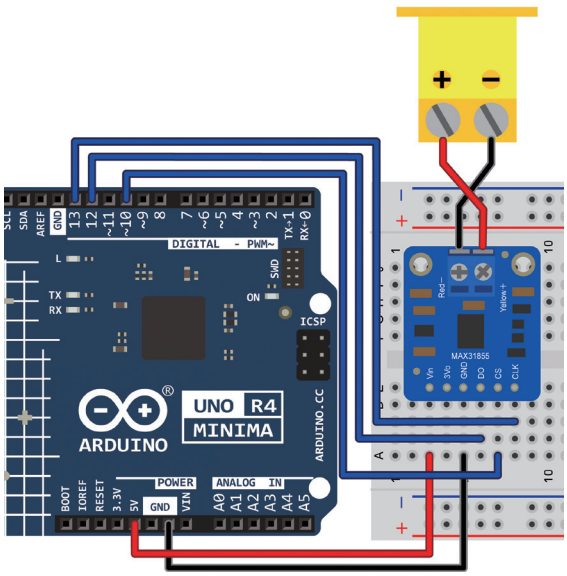


（日経BP撮影）

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

熱電対は**下図**のようにArduinoと接続します。熱電対用のコネクタを熱電対用ADコンバーターと接続します。ADコンバーターの「Yellow＋」と記載されている端子をコネクタの「＋」と記載された端子に接続します。接続したら、熱電対をコネクタに差し込みます。SPI通信をするためにMISOをD12、CLKをD13、CSをD10に接続します。MAX31855Kは出力しかしないため、MOSIを接続する必要はありません。

図　接続図



熱電対で計測した温度を取得するプログラム「thermocouple.ino」を以下に示します。MAX31855Kを制御するには、Adafruit Industries社が提供するライブラリ「Adafruit MAX31855」を利用します。本書第2章の2-2の手順に従ってArduino IDEにインストールしておきます。

熱電対で計測した温度は、thermocouple.readCelsius()関数を呼び出すことで取得できます。

ソース　熱電対で計測した温度を取得するプログラム

thermocouple.ino

```
#include <SPI.h>
#include "Adafruit_MAX31855.h"

const int MISO_PIN = 12;
const int CLK_PIN = 13;
const int CS_PIN = 10;

Adafruit_MAX31855 thermocouple( CLK_PIN, CS_PIN, MISO_PIN );

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

void loop() {
    double temp = thermocouple.readCelsius();

    Serial.print( "Temperature: " );
    Serial.print( temp );
    Serial.println( "C" );

    delay( 1000 );
}
```

SPIの各端子のGPIOの番号

温度を取得する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

40 センサー

フォトインターラプター

デジタル入力

プログラム photo-int.ino

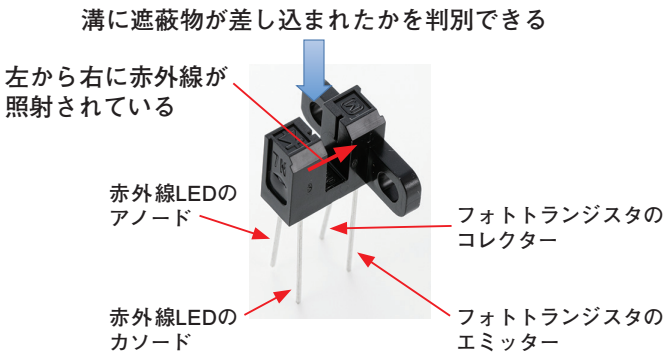
ライブラリ －

利用パーツ

- フォトインターラプター ×1（秋：109668）
- 抵抗器150Ω×1（秋：125151）、750kΩ×1（秋：125754）

「フォトインターラプター」は、赤外線を利用するセンサーモジュールです（**下図**）。モジュール上の溝を挟んで赤外線LEDと光センサー（フォトトランジスタ）が配置されており、この溝を通して赤外線が光センサーまで届くようになっています。溝に遮蔽物を差し込むと、赤外線が届かなくなり、光センサーの出力が変化します。

図　フォトインターラプターの外観



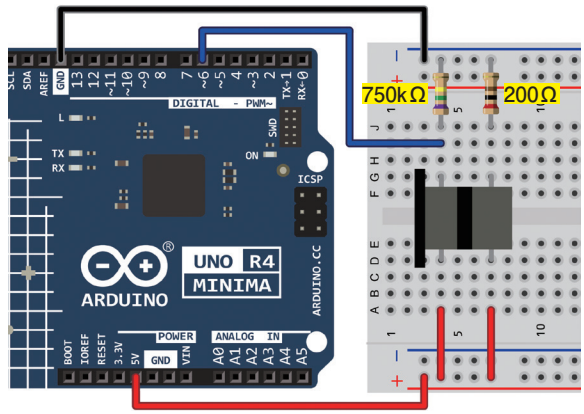
（日経BP撮影）

フォトインターラプターを使うと、例えばモーターが回転した回数を調べられます。モーターに穴が一つ開いた円盤を取り付けておきます。この円盤をフォトインターラプターの溝に差し込んでモーターを回転させると、穴が開いた部分が通過するたびに1回転したことを検出できます。

フォトインターラプターとArduinoは**下図**のように接続します。フォトトランジスタの状態変化を調べるために、一つ前のフォトリフレクターと同じように抵抗器を接続して、フォトトランジスタと抵抗器の間の電圧を読み取ります。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



遮蔽物が差し込まれているかどうかを判断するプログラム「photo-int.ino」を以下に示します。遮蔽物が差し込まれていない場合は、赤外線がフォトトランジスタに届くため入力が「High」となり「None.」と表示します。遮蔽物を差し込むと、赤外線が届かなくなるため入力が「Low」となり「Block.」と表示します。

ソース　遮蔽物が差し込まれているかどうかを判断するプログラム

photo-int.ino

```
const int PREF_PIN = 6;

void setup() {
    pinMode( PREF_PIN, INPUT );

    Serial.begin( 9600 );
}

void loop() {
    if( digitalRead( PREF_PIN ) == 1 ){
        Serial.println( "None." );
    }else{
        Serial.println( "Block." );
    }
    delay( 100 );
}
```

エミッターに接続したGPIOの番号

遮蔽物の有無を判別して表示する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

41 センサー

カラーセンサー

I²C

プログラム color_sensor.py

ライブラリ Adafruit APDS9960 Library

利用パーツ

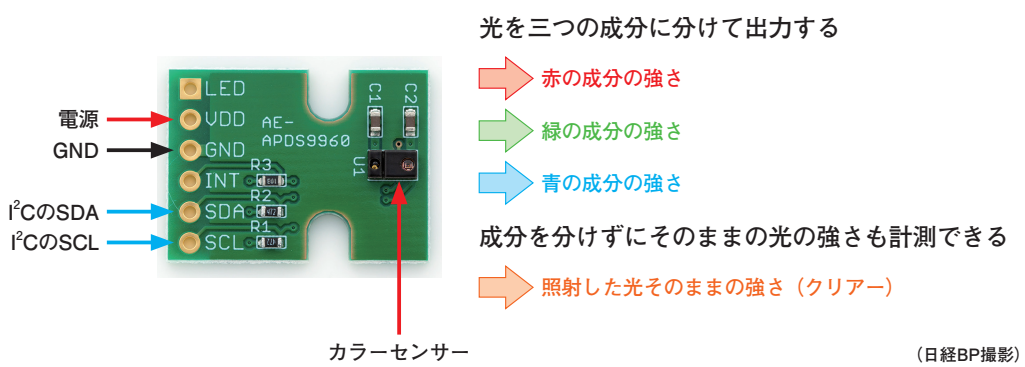
●カラーセンサー（秋:109754）

本書の第5章のNo.17とNo.18で紹介した「光センサー」（CdSセルやフォトトランジスタ）は、光の強さは分かるものの、どのような色の光が照射されているかは識別できません。光の色を知りたい場合には「カラーセンサー」が利用できます（**下図**）。カラーセンサーは、光の3原色である赤、緑、青の各成分を計測できる光センサー素子を搭載しており、受けた光を成分ごとに分けて計測できます。この計測結果から、光がどのような色であるかを判定可能です。例えば、赤と緑の成分が強ければ黄色であると分かります。

ここで紹介するカラーセンサー「APDS-9960」を搭載したモジュールは、中央に搭載したカラーセンサーチップで受けた光を3原色の成分ごとに計測できるほか、3原色に分けずにそのまま光の強さを計測する「クリアー」モードも搭載しています。

なお、APDS-9960は物体の動きを検知することができます。手をセンサーにかざして上下左右などに動かす事で、どちらに手を動かしたかが分かります。ただし、本書ではカラーセンサーとしての使い方を紹介し、動きの検知については説明しません。

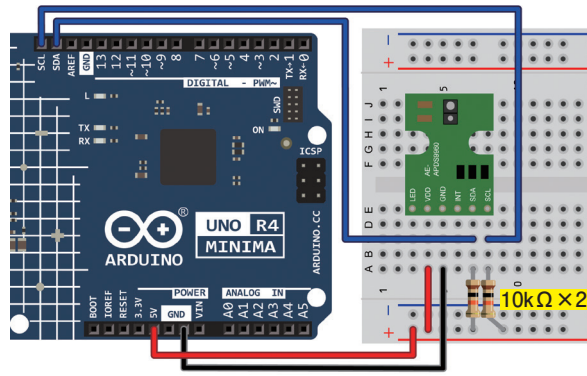
図　カラーセンサーの外観



（日経BP撮影）

APDS-9960は、I²C通信を使ってモジュールから計測結果を取得できます。I²Cアドレスは「0x39」です。Arduinoとは**下図**のように接続します。

図　接続図



APDS9960の制御には、Adafruitが提供するライブラリ「Arduino_APDS9960」を利用します。本書第2章の2-2の手順に従ってArduino IDEにインストールしておきます。

同ライブラリを使って、照射した光の成分を計測するプログラム「color_sensor.ino」を以下に示します。ライブラリの「apds.getColorData()」関数を呼び出すことで、各色成分の計測値を取得できます。

ソース　照射した光の成分を計測するプログラム

color_sensor.ino

```
#include "Adafruit_APDS9960.h"

Adafruit_APDS9960 apds;

void setup(){
    Serial.begin( 9600 );
    apds.begin();
    apds.enableColor( true );
}

void loop(){
    uint16_t red, green, blue, clear;

    while( apds.colorDataReady() == false ){
        delay(5);
    }

    apds.getColorData( &red, &green, &blue, &clear );
    Serial.print( "Red : " );
    Serial.print( red );
    Serial.print( "  Green : " );
    Serial.print( green );
    Serial.print( "  Blue : " );
    Serial.print( blue );
    Serial.print( "  Clear : " );
    Serial.println( clear );

    delay( 1000 );
}
```

光の色を取得できるようにする

センサーから計測値を取得する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

42 センサー

I²C

地磁気センサー

プログラム compass.ino

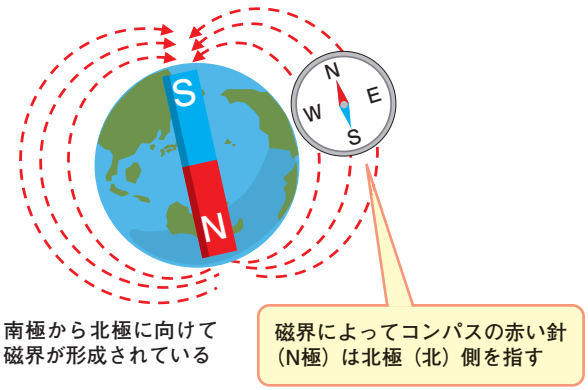
ライブラリ Adafruit LIS3MDL

利用パーツ

●地磁気センサー ×1（ス:6266）

地球は南極をN極、北極をS極とした大きな磁石のようになっており、南極から北極に向けて磁場が形成されています。このためコンパス（方位磁石）の針は、地球の磁場を受けてN極（赤い方）が北極方向、S極が南極方向を指し示し、その結果自分が現在どちらの方角を向いているかが分かります（**下図**）。

図　地球は南極から北極に向けて磁場が形成されている

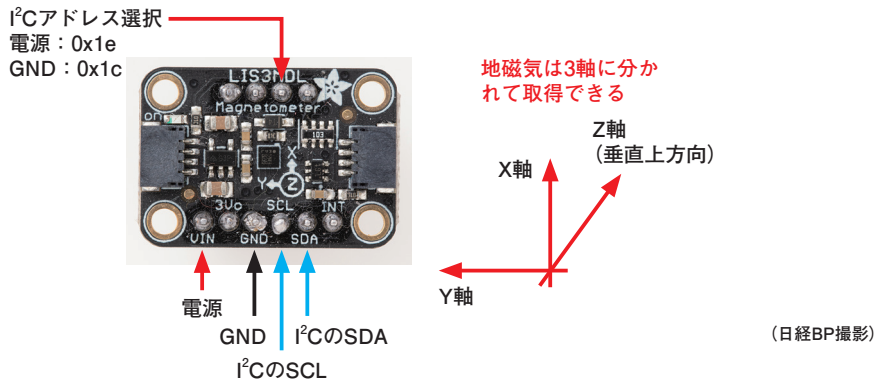


地磁気センサーは、この地球の磁場を計測できるセンサーです。磁場はX軸、Y軸、Z軸の成分に分けて計測されます。ここで紹介する地磁気センサー「LIS3MDL」は、各軸の向きが基板上に記載されています。**下図**のように置いた場合、上側がX軸、左側がY軸、基板表面に垂直な方向がZ軸となっています。

取得した磁場のX軸成分とY軸成分を利用することで、磁場がどちらの方向に向いているかを導き出せます。取得した値は、N極からS極に向けて正の値となっており、X軸とY軸の成分を合成して取得した方角が北向きとなります。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

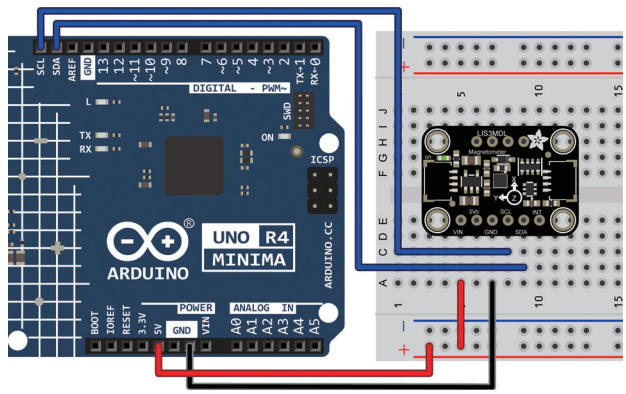
図　地磁気センサーの外観



（日経BP撮影）

LIS3MDLはI²CまたはSPIで通信可能です。ここではI²Cを利用する方法を説明します。I²Cアドレスは「0x1c」を使います。Arduinoとは**下図**のように接続します。

図　接続図



LIS3MDLの制御には、Adafruit Industries社が提供するライブラリ「Adafruit LIS3MDL」を利用します。本書第2章の2-2の手順に従ってArduino IDEにインストールしておきます。

方位を精度高く計測するには、センサーの計測値に対して、現在地に合わせた補正を加える必要があります。そこで、まずは補正値を取得するプログラム「compass_calib.ino」を実行します。すると、CALIB_X、CALIB_Yの補正値が表示されます。この状態で、水平の状態を保ちながらセンサーを数周回転させてください。CALIB_XとCALIB_Yが特定の値で安定したら、その補正値をメモしておきます。

方位の計測プログラム「compass.ino」を以下に示します。compass_calib.inoで取得した補正値をプログラム内の「CALIB_X」と「CALIB_Y」に設定してください。実行すると、方位が北を0度として0～360度の範囲で表示されます。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

ソース　方位を計測するプログラム

compass.ino

```
#include <Wire.h>
#include <Adafruit_LIS3MDL.h>
#include <math.h>

const int CALIB_X = 0;
const int CALIB_Y = 0;

Adafruit_LIS3MDL lis3mdl;

void setup(){
    Serial.begin(9600);
    lis3mdl.begin_I2C();
    lis3mdl.setPerformanceMode(LIS3MDL_MEDIUMMODE);
    lis3mdl.setOperationMode(LIS3MDL_CONTINUOUSMODE);
    lis3mdl.setDataRate(LIS3MDL_DATARATE_155_HZ);
    lis3mdl.setRange(LIS3MDL_RANGE_4_GAUSS);
    lis3mdl.setIntThreshold(500);
    lis3mdl.configInterrupt(false, false, true, true, false, true);
}

void loop(){
    lis3mdl.read();

    int x = lis3mdl.x - CALIB_X;
    int y = lis3mdl.y - CALIB_Y;
    float heading = -1 * ( atan2( x, y ) * 180) / M_PI;

    Serial.print( "Compass :" );
    Serial.print( heading );
    Serial.println( "deg" );

    delay( 500 );
}
```

補正値を設定する

計測値を取得し、補正する

方位を算出する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

43 センサー

I²C

9軸モーションセンサー

プログラム 9axis.ino

ライブラリ Adafruit BNO055

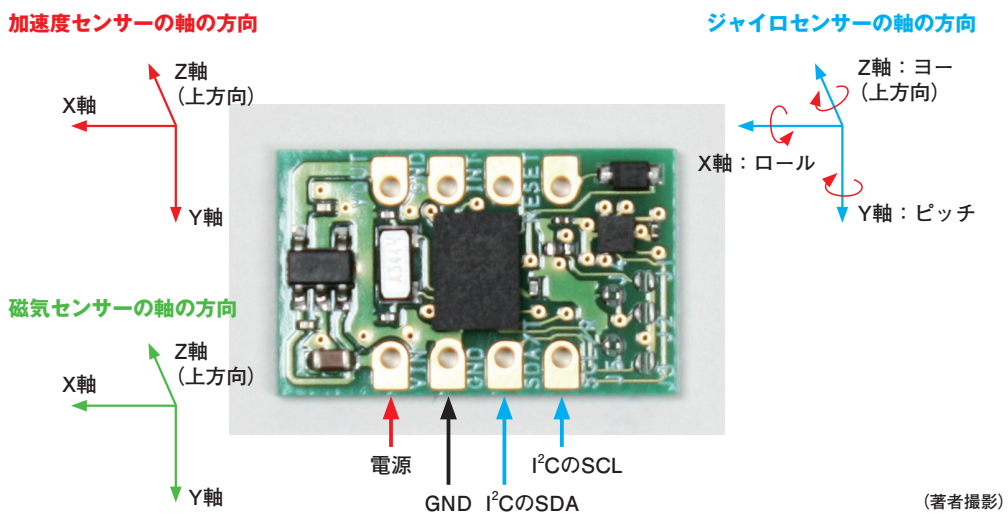
利用パーツ

●9軸モーションセンサー（秋:116996）

「9軸モーションセンサー」は、加速度、ジャイロ（角速度）、地磁気の三つをまとめて測れるセンサーです。ドローンなどの姿勢を調べる際に、この三つのセンサーがよく使われます。加速度センサーでは、機体がどの程度傾いたかや移動速度が分かります。ジャイロセンサーは機体の回転を計測できるので、向きや傾きを調べられます。地磁気センサーは、地球の磁場を計測して向いている方向を取得できます。これらの情報を統合することで、機体の姿勢を正確に把握して制御できます。

ここで紹介する9軸モーションセンサー「BNO055」は、X軸、Y軸、Z軸またはロール、ピッチ、ヨーの要素に分けてそれぞれのセンサーの計測値が出力されます（**下図**）。このように3種類の計測値がそれぞれ3軸／3要素に分けて出力されることから、9軸モーションセンサーあるいは単に9軸センサーと呼ばれています。

図　9軸モーションセンサーの外観

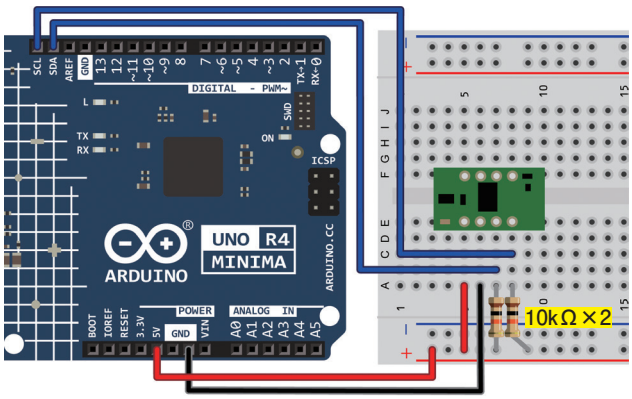


（著者撮影）

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

BNO055から各計測値を取得するには、I^2Cを使って通信します。Arduinoとの接続には**下図**のようにセンサーに付属するコネクタ付きケーブルを用いて接続します。

図　接続図



BNO055の計測には、Adafruit Industries社が提供するライブラリ「Adafruit BNO055」を利用します。本書第2章の2-2の手順に従ってArduino IDEにインストールしておきます。

加速度、磁気、ジャイロの各計測値を表示するプログラム「9axis.ino」を以下に示します。加速度、地磁気、角速度の計測値を取得し、それぞれX、Y、Z軸成分に分けて表示します。

ソース　加速度、地磁気、ジャイロの各計測値を表示するプログラム

9axis.ino

```
#include <Wire.h>
#include <Adafruit_BNO055.h>

Adafruit_BNO055 bno = Adafruit_BNO055( -1, 0x28, &Wire );

void setup(){
    Serial.begin( 9600 );
    bno.begin();
    bno.setExtCrystalUse( true );
}

void loop(){
    sensors_event_t gyro, mag, accel;

    bno.getEvent( &gyro, Adafruit_BNO055::VECTOR_GYROSCOPE );
    bno.getEvent( &mag, Adafruit_BNO055::VECTOR_MAGNETOMETER );
    bno.getEvent( &accel, Adafruit_BNO055::VECTOR_ACCELEROMETER );

    showdata( &accel );
```

各計測値を取得する

次ページに続く

```
    showdata( &mag );
    showdata( &gyro );

    delay( 500 );
}

void showdata( sensors_event_t* event ) {
    double x, y, z;
    if ( event->type == SENSOR_TYPE_ACCELEROMETER ) {
        x = event->acceleration.x;
        y = event->acceleration.y;
        z = event->acceleration.z;

        Serial.print( "Accel X:" );
        Serial.print( x );
        Serial.print( "  Y:" );
        Serial.print( y );
        Serial.print( "  Z:" );
        Serial.println( z );
    } else if( event->type == SENSOR_TYPE_MAGNETIC_FIELD ){
        x = event->magnetic.x;
        y = event->magnetic.y;
        z = event->magnetic.z;

        Serial.print( "Mag X:" );
        Serial.print( x );
        Serial.print( "  Y:" );
        Serial.print( y );
        Serial.print( "  Z:" );
        Serial.println( z );
    } else if ( event->type == SENSOR_TYPE_GYROSCOPE ) {
        x = event->gyro.x;
        y = event->gyro.y;
        z = event->gyro.z;

        Serial.print("Gyro X:");
        Serial.print( x );
        Serial.print("  Y:");
        Serial.print( y );
        Serial.print("  Z:");
        Serial.println( z );
    }
}
```

計測結果を表示する関数

加速度を表示する

地磁気を表示する

角速度を表示する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

44 センサー

曲げセンサー

アナログ入力

プログラム bend.ino

ライブラリ －

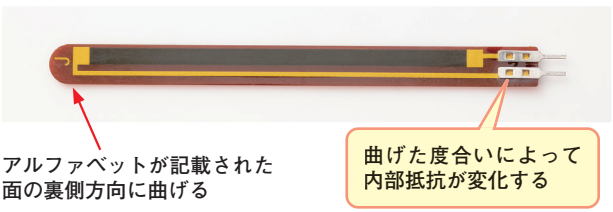
利用パーツ

●曲げセンサー×1（秋：116410） ●抵抗器10kΩ×1（秋：125103）

「曲げセンサー」は、取り付けた物体がどれくらい曲げられたかを計測できるセンサーです。フィルム内に抵抗素子が入っており、曲げることで抵抗値が変化します。この抵抗値を計測することで曲がり度合いを把握できる仕組みです。例えば、ロボットの腕の関節などに曲げセンサーを取り付けておけば、どの程度腕を曲げているかが分かります。

ここで紹介する曲げセンサーは、片面にアルファベットが記載されています（**下図**）。この面とは逆方向に曲げることで内部抵抗値が変化します。伸ばした状態では内部抵抗は小さく、曲げていくことで内部抵抗が徐々に増えていきます。

図　曲げセンサーの外観

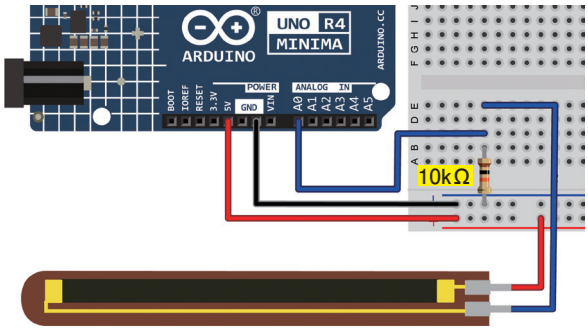


（日経BP撮影）

曲げセンサーとArduinoは**下図**のように接続します。曲げセンサーの内部抵抗値を読み取るには、図中に示したように抵抗器を直列に接続して分圧回路を作り、抵抗器にかかる電圧をArduinoのアナログ入力で読み取ります。ここで示した回路では、伸ばした状態では電圧が高く、曲げることで電圧が低くなります。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



曲げた度合いを調べるプログラム「bend.ino」を以下に示します。このプログラムは、曲げた度合いを割合で表示します。そのために必要となるセンサーの校正情報を、BEND_MIN変数とBEND_MAX変数にあらかじめ設定しておく必要があります。プログラムを実行し、曲げセンサーをまっすぐにした状態と180度曲げた状態における「Value」の値をメモしておきます。そして、BEND_MINにまっすぐの状態の値を、BEND_MAXには180度曲げた状態の値を設定してください。プログラムを再度実行すると、曲げた度合いが正しく表示されるようになります。

なお、180度曲げる際は、センサーに折り目が付く形で折り曲げないよう注意してください。端同士を重ねてループ形状にして計測します（このため校正しても曲げ度合いはおおまかな値となります）。

ソース　曲げた度合いを表示するプログラム

bend.ino

```
const int SENSOR_PIN = A0;
                          伸ばした状態の値
const float BEND_MIN = 300;
const float BEND_MAX = 80;
                        180度曲げた状態の値
void setup() {
    Serial.begin( 9600 );
}

void loop(){
    int value;
    value = analogRead( SENSOR_PIN );
                        曲げた度合を求める
    float ratio = 1 - ( value - ア
BEND_MAX ) / ( BEND_MIN - BEND_MAX );
    ratio = round( ratio * 100 );イ

    if ( ratio < 0 ){
        ratio = 0;
    }
    if ( ratio > 100 ){
        ratio = 100;
    }

    Serial.print( "Ratio : " );
    Serial.print( ratio );
    Serial.print( "%   Value : " );
    Serial.println( value );

    delay( 500 );
}
```


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

45 センサー

UART

ドップラーセンサー

プログラム doppler.ino

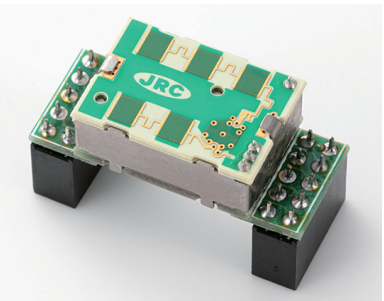
ライブラリ ー

利用パーツ

●ドップラーセンサー ×1（秋：107776）

「ドップラーセンサー」は、人や物体が近づいたり遠ざかったりする動きを検知できるセンサーです（**下図**）。後ろから車が近づいているのを検出したら注意を促すといった使い方ができます。

図　ドップラーセンサーの外観



（日経BP撮影）

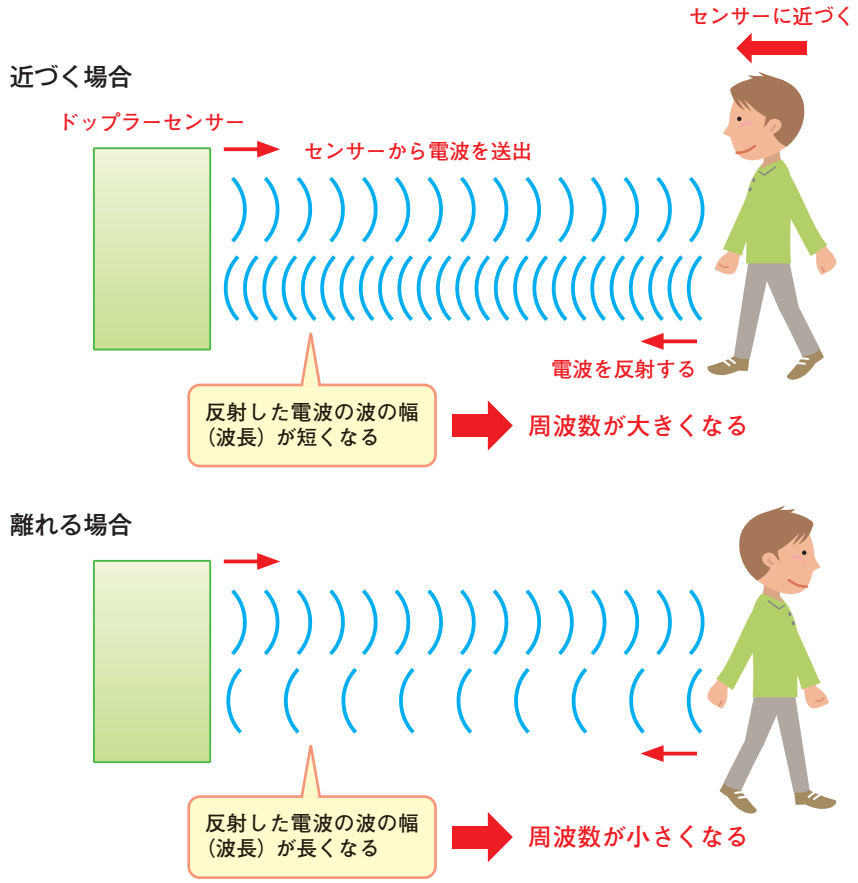
ドップラーセンサーは、電波を送出しています。1秒間に発生する電波の波の数を周波数と呼び、Hz（ヘルツ）という単位で表します。ここで紹介するドップラーセンサーは周波数24GHzの電波を発します。

ドップラーセンサーから発した電波は、対象物に当たると反射してセンサーに戻ってきます。この際、対象物がセンサーに近づく方向に動いていると、センサーが受け取る1秒当たりの波の数が多くなる、つまり、周波数が高くなります（**下図**）。

一方、センサーから離れる方向に動くと、1秒当たりの波の数が少なくなり、周波数が低くなります。この周波数の変化を見ることで、近づいているか離れているかを判断できます。これがいわゆる「ドップラー効果」で、救急車が近づいてくる場合にサイレンが高い音（周波数が高い）で聞こえ、遠ざかると低い音（周波数が低い）で聞こえるのと同じ原理です。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

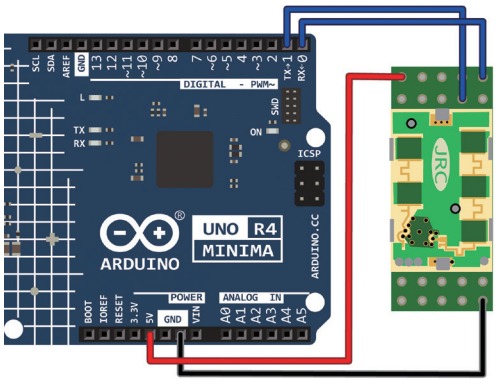
図　ドップラー効果を用いて物体が近づいているか遠ざかっているかを判別する



ドップラーセンサーとArduinoは**下図**のように接続します。ここで利用するドップラーセンサーには基板とピンソケットが付属しており、あらかじめはんだ付けしておく必要があります。特にセンサー本体と基板のはんだ付けについては端子間の間隔が狭いため、はんだでつながってしまう「ブリッジ」にならないよう注意してください。はんだ付けが終わったら、オス-オス型ジャンパー線をドップラーセンサーのピンソケットに直接差し込んで接続します。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



物体が近づいているか遠ざかっているかを判別するプログラム「doppler.ino」を以下に示します。ここで利用するドップラーセンサーとは、UARTで通信して計測結果を取得できます。取得した値が「10」の場合は物体がセンサーに近づいており、「78」の場合は遠ざかっていると判別できます。

ソース　物体が近づいているか遠ざかっているかを判別するプログラム

doppler.ino

```
void setup() {
    Serial.begin( 9600 );
    Serial1.begin( 9600, SERIAL_8O1 );
}

void loop() {
    if ( Serial1.available() > 0 ) {
        uint8_t buf[2];
        int len = Serial1.readBytes( buf, 2 );

        if ( len > 1 ) {
            int ans = buf[1];
            if ( ans == 10 ) {
                Serial.println( "Approach." );
            } else if ( ans == 78 ) {
                Serial.println( "Move away." );
            }
        }
    }
    delay( 100 );
}
```

値が67（近づいている）の場合に表示する

値が76（遠ざかっている）の場合に表示する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

46 センサー

赤外線距離センサー

I²C

プログラム ir_length.ino

ライブラリ －

利用パーツ

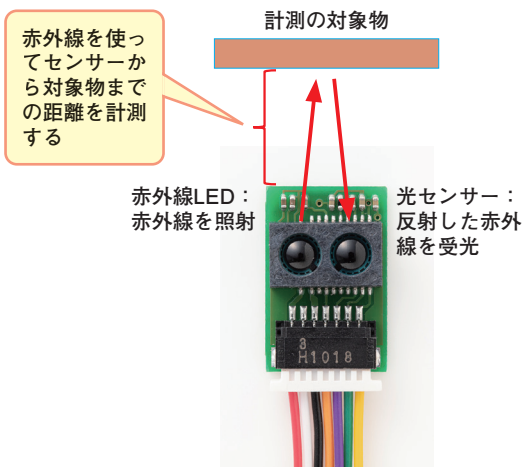
●赤外線距離センサー ×1（秋：107547）

「赤外線距離センサー」は、センサーから対象物までの距離を赤外線を使って計測するセンサーです。一つ前で紹介した超音波距離センサーと同じような使い方ができます。サイズが小さな対象物でも距離を測りやすいことが、超音波距離センサーよりも優れている点です。

赤外線距離センサーは、対象物に照射した赤外線の反射光を受光して距離を計測します（**下図**）。ここで紹介する赤外線距離センサー「GP2Y0E03」は、4～50cmの距離の計測が可能です。二つのレンズが取り付けられており、一方が赤外線を照射し、もう一方が反射した赤外線を受光します。

GP2Y0E03が計測した距離データをArduinoで取得するにはI²C通信を使います。I²Cアドレスは「0x40」です。このセンサーは、I²Cで計測値を送るだけでなく、計測した距離に対応する電圧を2番端子から出力します。この電圧をArduinoのADコンバーターで読み取って計算することで距離を求める方法もありますが、ここではI²C通信で取得する方法を説明します。

図　赤外線距離センサーの外観



（日経BP撮影）

端子の用途

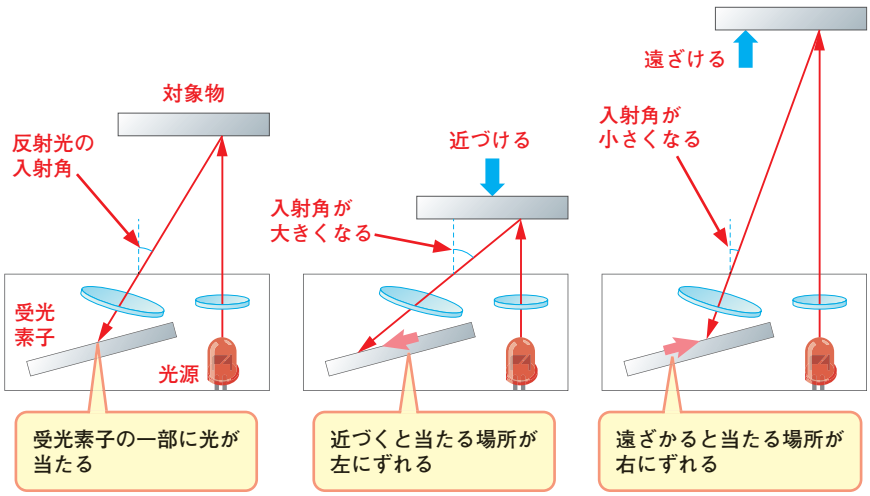
番号	線色	用途
1	赤	電源
2	白	出力（電圧変化）
3	黒	GND
4	橙	I/O電圧
5	紫	アクティブの切り替え
6	緑	I²CのSCL
7	黄	I²CのSDA

I²Cアドレス：0x40

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

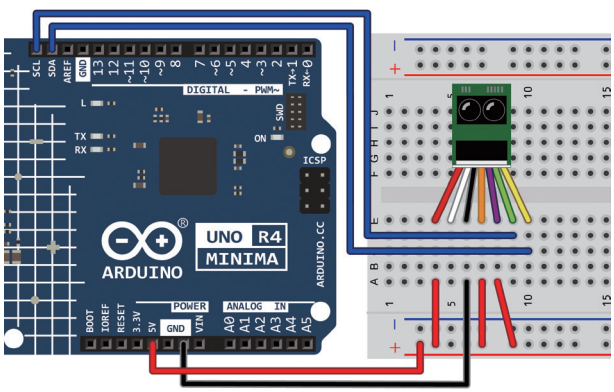
GP2Y0E03は、「PSD（Position Sensing Detector）」と呼ぶ方式で距離を計測します（**下図**）。PSD方式は、対象物との距離によって、反射光のレンズへの入射角度が変わることを利用します。このとき、内部に配置された光センサーにレンズで集束された光が当たる位置も変わります。図に示した配置では、対象物が近いほどセンサーの左寄りに、遠いほど右寄りに光が当たります。この当たった位置を計測して、対象物までの距離を求める仕組みです。

図　赤外線距離センサーの仕組み



GP2Y0E03とArduinoは**下図**のように配線します。GP2Y0E03は、付属するコネクタ付きケーブルを使ってブレッドボードに取り付けます。

図　接続図



赤外線距離センサーで対象物までの距離を計測するプログラム「ir_length.ino」を以下に示します。「measure()」関数を呼び出すことで、対象物までの距離をcm単位で取得します。

ソース　赤外線距離センサーで対象物までの距離を計測するプログラム

ir_length.ino

```
#include <Wire.h>

const int GP2Y0E03_ADDR = 0x40;

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

float measure( uint8_t addr ) {
    Wire.beginTransmission( addr );
    Wire.write( 0x35 );
    Wire.endTransmission( false );
    Wire.requestFrom( addr, (uint8_t)1);
    uint8_t shift = Wire.read();

    Wire.beginTransmission(addr);
    Wire.write(0x5E);
    Wire.endTransmission(false);
    Wire.requestFrom(addr, (uint8_t)1);
    uint8_t d_h = Wire.read();

    Wire.beginTransmission(addr);
    Wire.write(0x5F);
    Wire.endTransmission(false);
    Wire.requestFrom(addr, (uint8_t)1);
    uint8_t d_l = Wire.read();

    int d = (d_h << 4) + d_l;
    float dist = d / (16.0 * pow(2, shift));

    return dist;
}

void loop() {
    float distance = measure( GP2Y0E03_ADDR );
    Serial.print( "Distance: " );
    Serial.print( distance );
    Serial.println( " cm" );
    delay( 500 );
}
```

I²Cアドレス

距離を計測する関数

計測した距離を取得する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

47 センサー

近接センサー

I²C

プログラム approach.ino

ライブラリ Adafruit_VCNL4010

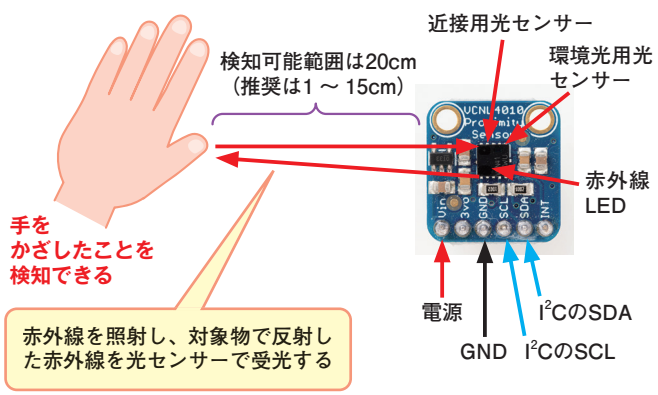
利用パーツ

●近接センサー ×1（ス：2640、マ：M0082-9735）

「近接センサー」は、手などをかざしたかどうかを検知できるセンサーです。十数cmの距離以内の範囲にかざした場合のみ反応します。例えば、手をかざすだけで水や石けん水を流す装置などに利用できます。蛇口のレバーに触れずに済むため衛生的に使えます。

ここで紹介する近接センサー「VCNL4010」は、約20cm以内に対象物があるかどうかを調べられます（**下図**）。赤外線を照射し、対象物で反射した赤外線を光センサーで受けることで、手がかざされたかどうかを検知します。環境光を計測する光センサーも搭載しており、周囲の明るさも計測できます。

図　近接センサーの外観

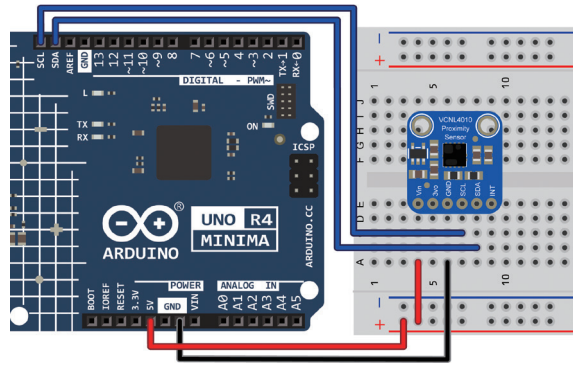


（日経BP撮影）

ArduinoでVCNL4010の計測値を受信するには、I²C通信を使います。I²Cアドレスは「0x13」です。Arduinoとは**下図**のように接続します。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

図　接続図



VCNL4010を制御するには、Adafruit Industries社が提供するライブラリ「Adafruit_VCNL4010」を利用します。本書第2章の2-2の手順に従ってArduino IDEにインストールしておきます。

同ライブラリを使って、手などをかざしたかどうかを検知するプログラム「approach.ino」を以下に示します。このプログラムは、センサーからの入力値が「TH」変数で設定した値以上になったら手が近づいたとみなし、「Approach.」と表示します。THの初期値として4000を代入していますが、各自の環境に合わせて調節可能です。まず1回プログラムを実行し、手をかざして近づいたと検知させたい位置まで近づけていき、表示された値をTHに設定してください。

ソース　手などをかざしたかどうかを検知するプログラム

approach.ino

```
#include <Wire.h>
#include "Adafruit_VCNL4010.h"

const int TH = 4000;
Adafruit_VCNL4010 vcnl;

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

    vcnl.begin();
}

void loop() {
    int value = vcnl.readProximity();
    if( value >= TH ){
        Serial.print( "Approach.  " );
    }
    Serial.print( "Value: " );
    Serial.println( value );

    delay( 500 );
}
```

近接したかを判断するしきい値

計測値を取得する

近接したかを判断する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

48 センサー

ロードセル

シリアル通信

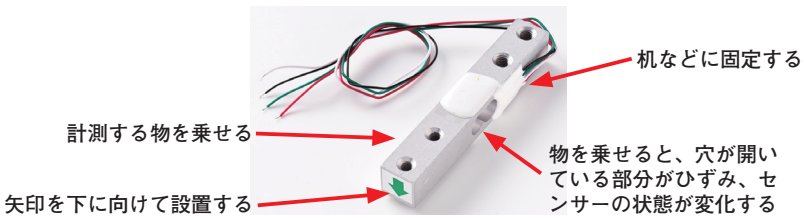
プログラム weight.ino
ライブラリ Adafruit HX711

利用パーツ

●ロードセル×1（秋:113042） ●ロードセル用ADコンバーター ×1（秋:112370）
●タクトスイッチ×1（秋:103647）

「ロードセル」は、重さを計測できるセンサーモジュールです（**下図**）。中央付近にひずみセンサーが貼り付けてあります。物体をロードセルに乗せると、中央の穴の部分でひずみが生じ、ひずみセンサーの内部抵抗値が変化します。この内部抵抗値の変化を読み取ることで重さを計測できます。ロードセルに取り付けられた赤と黒の配線を電源に接続すると、白と緑の配線に電圧が生じます。この電圧を読み取ることで重さが分かります。

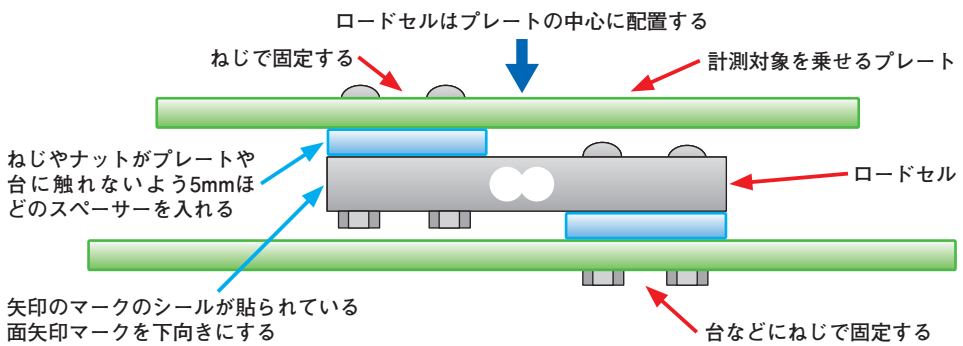
図　ロードセルの外観



（日経BP撮影）

ロードセルは、記載されている矢印が下向きになるように配置し、底面を台座などに固定します。この状態で天面に物を乗せることで、重さを計測できます。設置例を**下図**に示します。

図　ロードセルの設置例

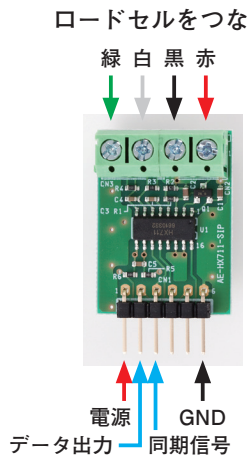


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

ロードセルは製品によって計測できる重さの範囲が異なるため、計測したい対象物に合わせて利用するロードセルを選択する必要があります。ここでは、10kgまで計測可能なロードセルを使う場合を紹介します。

ロードセルは、出力する電圧の変化が非常に小さいため、Arduinoが搭載するADコンバーターでは変化を十分には読み取れません。そこで、別途ロードセル用のADコンバーターを用意し、これを介してロードセルが出力した電圧を読み取ります。ここで利用するロードセル用ADコンバーター「HX711」は分解能が24ビットと高く、非常に小さな電圧の変化も読み取り可能です（**下図**）。

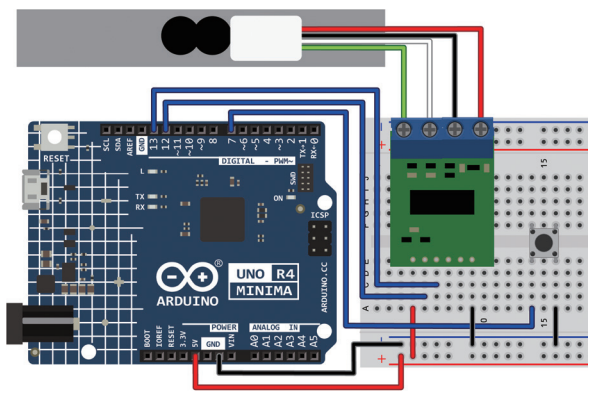
図　ロードセル用ADコンバーターの外観



（日経BP撮影）

ロードセルおよびロードセル用ADコンバーターとArduinoは**下図**のように接続します。計測値を0にリセットするためのタクトスイッチも接続しておきます。

図　接続図



@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

HX711から計測値を取得するには、Adafruit Industries社が提供するライブラリ「Adafruit HX711」を利用します。本書第2章の2-2の手順に従ってArduino IDEにインストールしておきます。

同ライブラリを利用し、重さを計測するプログラム「weight.ino」を以下に示します。プログラムを実行したら、まずタクトスイッチを押してください。すると、重さが0にリセットされます。その後、ロードセルに計測対象物を乗せると重さが表示されます。

ソース　重さを計測するプログラム

weight.ino

```
#include "Adafruit_HX711.h"
                データ出力端子のGPIO番号
const int DOUT_PIN = 12;
const int SCK_PIN = 13;    同期信号端子のGPIO番号
const int SW_PIN = 7;
                タクトスイッチのGPIO番号
const float OUT_VOL = 0.001;
const float LOAD = 20000.0;

Adafruit_HX711 scale( DOUT_PIN, SCK_PIN );

void scaletare( int avgtime ){
    long sum = 0;
    for ( int i = 0; i < avgtime; i++ ){
        sum = sum + (long)scale.readChan
nelRaw( CHAN_A_GAIN_128 );
        delay( 100 );
    }

    float tareavg = (float)sum / avgtime;
    Serial.print( "Tare : " );
    Serial.println( tareavg );

    scale.tareA( (long)tareavg );
}

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

    pinMode( SW_PIN, INPUT_PULLUP );

    scaletare( 10 );
}

                ADコンバーターから値を取得して重さを算出する
void loop() {
    if (!scale.isBusy()) {
        long sum_data = 0;
        for ( int i = 0; i < 10; i++ ){
            sum_data = sum_data + sca
le.readChannelBlocking( CHAN_A_GAIN_1
28 );
            delay( 10 );
        }
        long data = (long)( sum_data
/ 10 );
        float volt = (float)data * 4.
2987 / 16777216.0 / 128;
        float weight = volt / ( OUT_V
OL * 4.2987 / LOAD );

        Serial.print( "   Weight: " );
        Serial.print( weight );
        Serial.println( "g" );

    } else {
        Serial.println("HX711 not ready");
        delay( 1000 );
    }
                タクトスイッチを押したら重さを0にリセットする
    if( digitalRead( SW_PIN ) == 0 ){
        scaletare( 10 );
    }

    delay( 100 );
}
```

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

49 センサー

ガスセンサー

アナログ入力

プログラム smell.ino

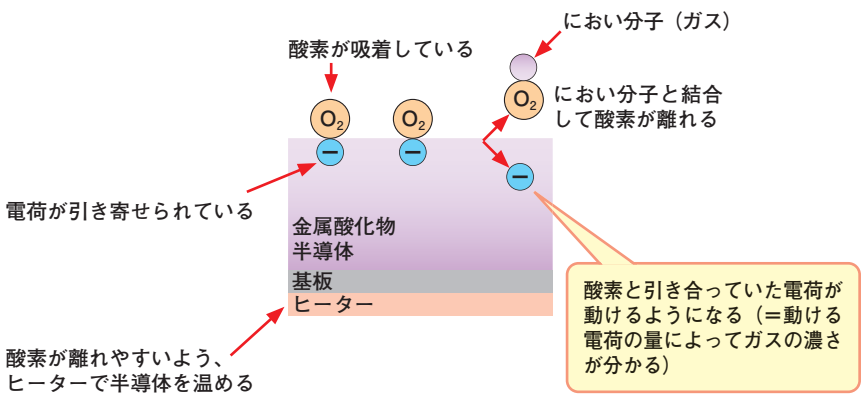
ライブラリ ―

利用パーツ

●においセンサー×1（秋：100989） ●トランジスタ×1（秋：102612）
●抵抗器27Ω（1W）×1（秋：107960）、1kΩ×2（秋：125102）、10kΩ×1（秋：125103）

「ガスセンサー」を使うと、空気中に含まれる特定の種類のガスを検出できます。ここで利用するガスセンサー「TGS2450」は、メチルメルカプタンや硫化水素といった口臭の主な原因となるガス成分（硫黄化合物系ガス）を検出可能な金属酸化物半導体を使ったガスセンサーです（**下図**）。ガスセンサーの周りにこれら対応するガスがあると、金属酸化物半導体に吸着している酸素がガスによってはぎ取られます。その結果、酸素に引き寄せられていた金属酸化物半導体内の電荷が自由に動けるようになって内部抵抗値が下がります。この変化を計測することでガスを検出できる仕組みです。

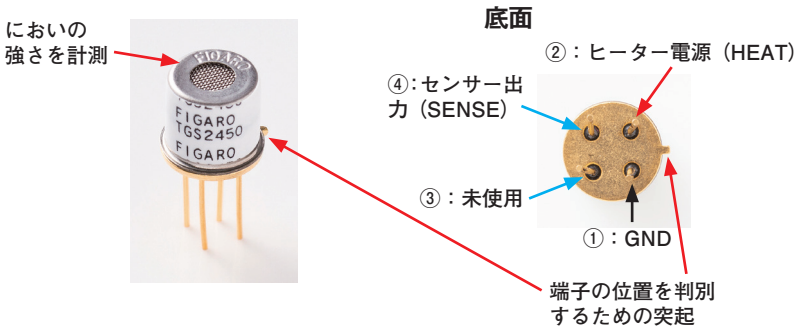
図　ガスセンサーの仕組み



TGS2450は下面に四つの端子を備えています（**下図**）。①番端子がGND、②番端子が金属酸化物半導体を温めるためのヒーター用電源端子、④番端子がセンサー出力端子となっています（③番端子は未使用）。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

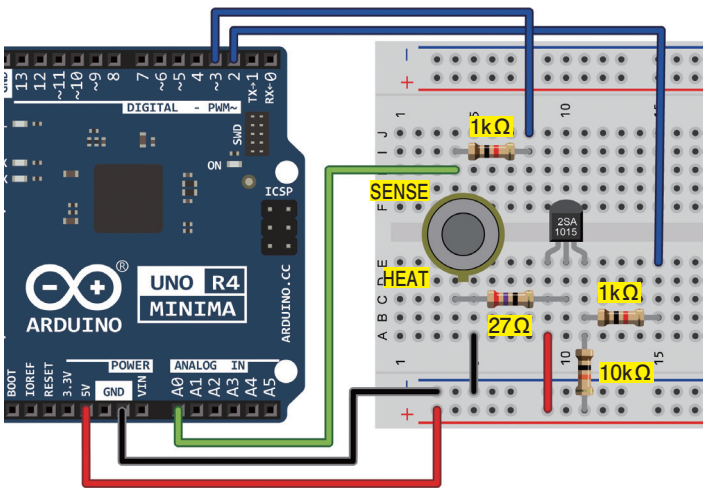
図　TGS2450の外観と各端子の役割



（日経BP撮影）

TGS2450とArduinoは**下図**のように接続します。検出に当たっては、金属酸化物半導体から酸素が離れやすくなるように事前にヒーターで加熱する処理が必要です。このヒーターの温め処理と計測処理は同時には実行できません。そこで、トランジスタを利用してヒーター温め処理と計測処理を切り替えられるようにしています。

図　接続図



TGS2450を使ってにおいを計測するプログラム「smell.ino」を以下に示します。ガスセンサーの計測結果はアナログ入力への電圧値として取得できるので、これを「volt」変数に格納しています。この値が所定の値（TH）よりも小さい場合は、においが強いと判断して「BAD.」と表示します。

THをどの程度にするかは、検出対象のガスがない状態でプログラムを実行して通常時の電圧を記録しておきます。続いて、口臭がすると感じる息をガスセンサーに吹きかけて電圧を記録します。通常時の電圧と息を吹きかけたときの電圧の中央値をTHに設定してください。

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

ソース　においを計測するプログラム

smell.ino

```
const float TH = 2.0; ── においが強いと判断するしきい値
const float VREF = 5.0;

const int HEATER_PIN = 2;  ┐
const int SENSOR_PIN = 3;  ┘── センサー制御用のGPIO番号
const int OUTPUT_PIN = A0; ── センサーの計測値を取得するアナログ入力の番号

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

    pinMode( HEATER_PIN, OUTPUT );
    pinMode( SENSOR_PIN, OUTPUT );
    digitalWrite( HEATER_PIN, HIGH );
    digitalWrite( SENSOR_PIN, LOW );
}

void loop() {
    int value = 0;

    delay(237);
    digitalWrite( SENSOR_PIN, HIGH ); ── センサーの状態を読み取れるようにする
    delay(3);
    value = analogRead( OUTPUT_PIN ); ── センサーの計測値を取得する
    delay(2);
    digitalWrite( SENSOR_PIN, LOW ); ── センサーの読み取りを無効にする

    digitalWrite( HEATER_PIN, LOW );  ┐
    delay(8);                         ├── ヒーターでセンサーを温める
    digitalWrite( HEATER_PIN, HIGH ); ┘

    float volt = value / 1024.0 * VREF;

    if ( volt > TH ) {                ┐
        Serial.print( "OK. Volt: " );  │
        Serial.println( volt );       │
    } else {                          ├── においが強いかを判別する
        Serial.print( "BAD. Volt: " ); │
        Serial.println( volt );       │
    }                                 ┘
}
```


@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

50 センサー

UART

CO2センサー

プログラム co2.ino

ライブラリ ー

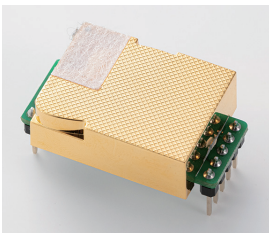
利用パーツ

●CO2センサー ×1（秋：116142）

「CO2センサー」は、二酸化炭素（CO2）の濃度を計測できるセンサーです（**下図**）。冬場は窓を開けることが少なくなるため、室内のCO2濃度が高くなりがちです。そこで、CO2センサーでその濃度を計測し、所定の濃度を超えたら換気を促すといった使い方ができます。

ここで紹介するCO2センサー「MH-Z19C」は、400～5000ppmのCO2濃度を計測可能です。Arduinoで計測値を読み取るにはUARTを使います。

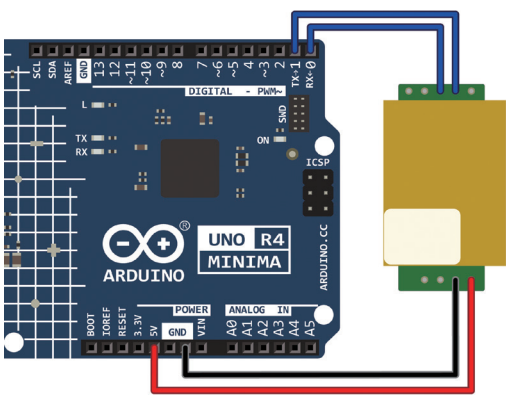
図　CO2センサーの外観



（日経BP撮影）

MH-Z19Cは、一般的なブレッドボードと端子間隔が異なっているため直接差し込めません。そこで、**下図**のようにオス-メス型ジャンパー線を使ってブレッドボードと接続します。

図　接続図



@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc

CO2濃度の計測プログラム「co2.ino」を以下に示します。ArduinoとMH-Z19CはUARTの0チャンネルを使って通信します。MH-Z19Cに対してプログラムで所定の文字列を送り込むことでセンサーの設定や計測値を取得できます。計測値を取得するには、16進数で「ff」「01」「86」「00」「00」「00」「00」「79」と文字列を送ることで値が返ってきます。取得した値の3文字目（s[2]）と4文字目（s[3]）が計測値です。3文字目を256倍して4文字目を足すことで、CO2濃度を求められます。

ソース　二酸化炭素の濃度を計測するプログラム

co2.ino

```
const bool AUTO_CALIB = true;

void setup() {
    Serial.begin( 9600 );
    Serial1.begin( 9600, SERIAL_8N1, RX_PIN, TX_PIN );

    if (AUTO_CALIB) {
        Serial1.write((const uint8_t*)"\xFF\x01\x79\xA0\x00\x00\x00\x00\xE6", 9);
    } else {
        Serial1.write((const uint8_t*)"\xFF\x01\x79\x00\x00\x00\x00\x00\x86", 9);
    }
    delay(1000);
}

void loop() {
    Serial1.write((const uint8_t*)"\xFF\x01\x86\x00\x00\x00\x00\x00\x79", 9);
    delay(1000);

    if (Serial1.available()) {
        uint8_t buf[9];
        Serial1.readBytes(buf, 9);
        if (buf[0] == 0xFF && buf[1] == 0x86) {
            int co2 = buf[2] * 256 + buf[3];
            Serial.print( co2 );
            Serial.println( " ppm" );
        } else {
            Serial.println( "Failed to read CO2 concentration" );
        }
    } else {
        Serial.println( "No data received" );
    }
}
```

計測値取得の命令を送る

計測値を取得する

計測値を表示する

@2025 Kazuhiro Fukuda, Nikkei Business Publications, Inc