
Table of Contents

はじめに	1.1
#102 Buzzer Brick	2.1
#103 Button Brick	2.2
#104 Angle Brick	2.3
#105 Vibrator Brick	2.4
#108 Temperature Brick	2.5
#109 Ambient Light Brick	2.6
#110 Tilt Brick	2.7
#116 Distance Brick	2.8
Propeller Kit	3.1
Motor Shield for Arduino	4.1
高校生向け課題	5.1

FaBo STEM Kit for Arduino 導入マニュアル

本ドキュメントについて

FaBo STEM Kit for ArduinoのSTEM教育向けチュートリアルです。

対象レベル

中学3年生から高校3年生

使用言語

Arduino Programming Language

使用ツール

Arduino IDE

講習実績

2016年2月 宇都宮工業高校 3年生向け授業

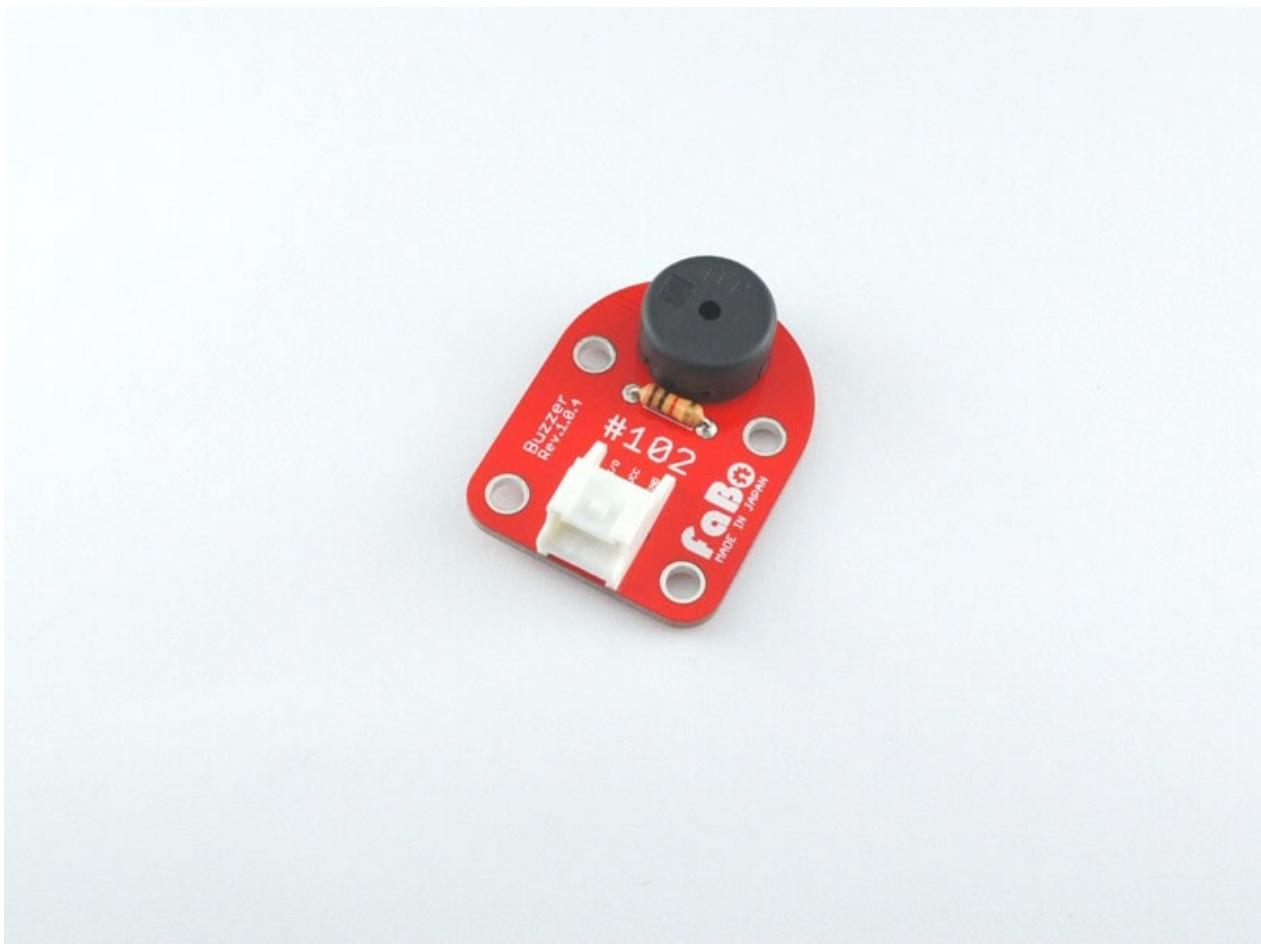
使用ハード

- #102 Buzzer Brick
- #103 Button Brick
- #104 Angle Brick
- #105 Vibrator Brick

- #108 Temperature Brick
- #109 Ambient Light Brick
- #110 Tilt Brick
- #116 Distance Brick
- FaBo Motor Driver
- FaBo Propeller Kit

Powered by FaBo

#102 Buzzer Brick

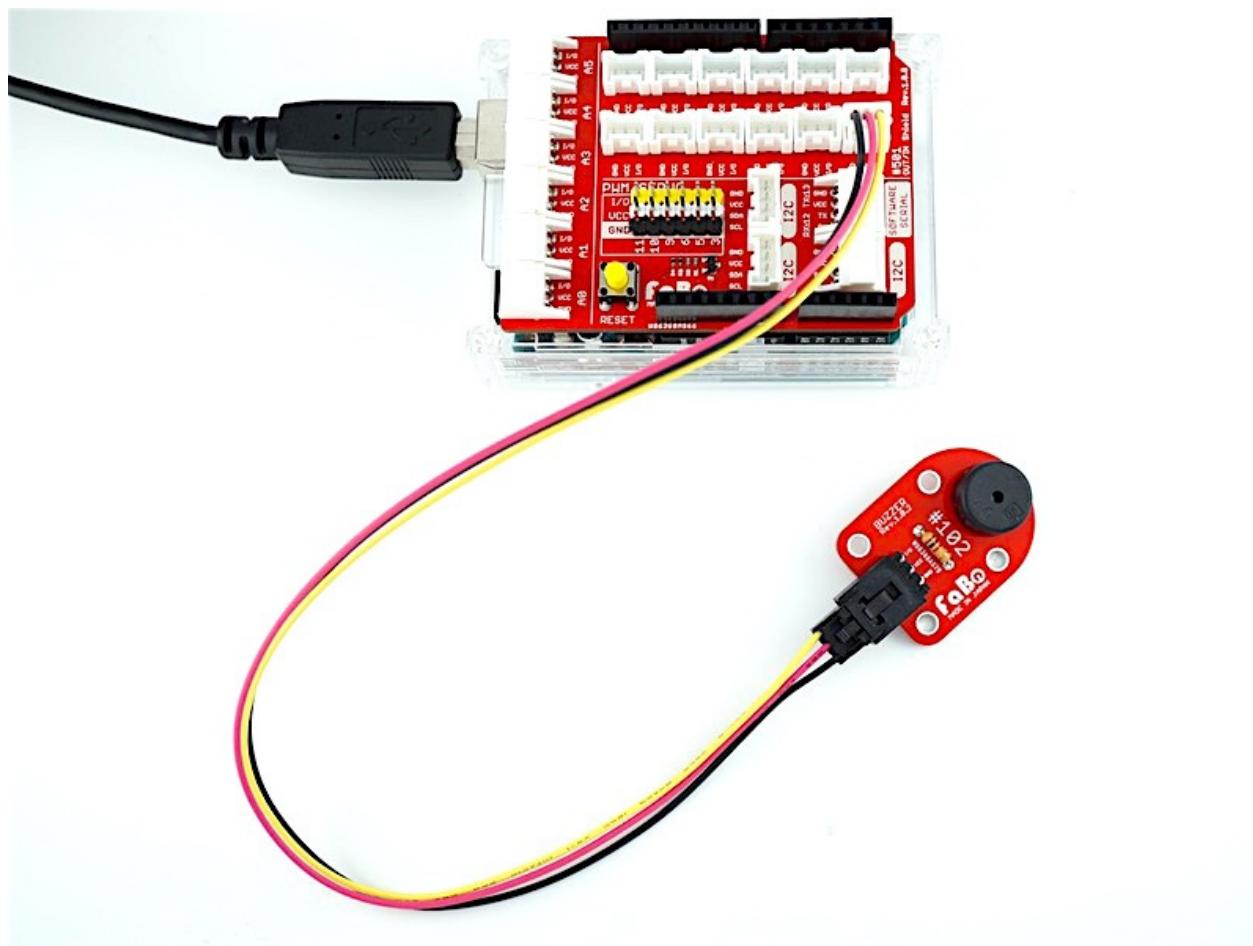


Overview

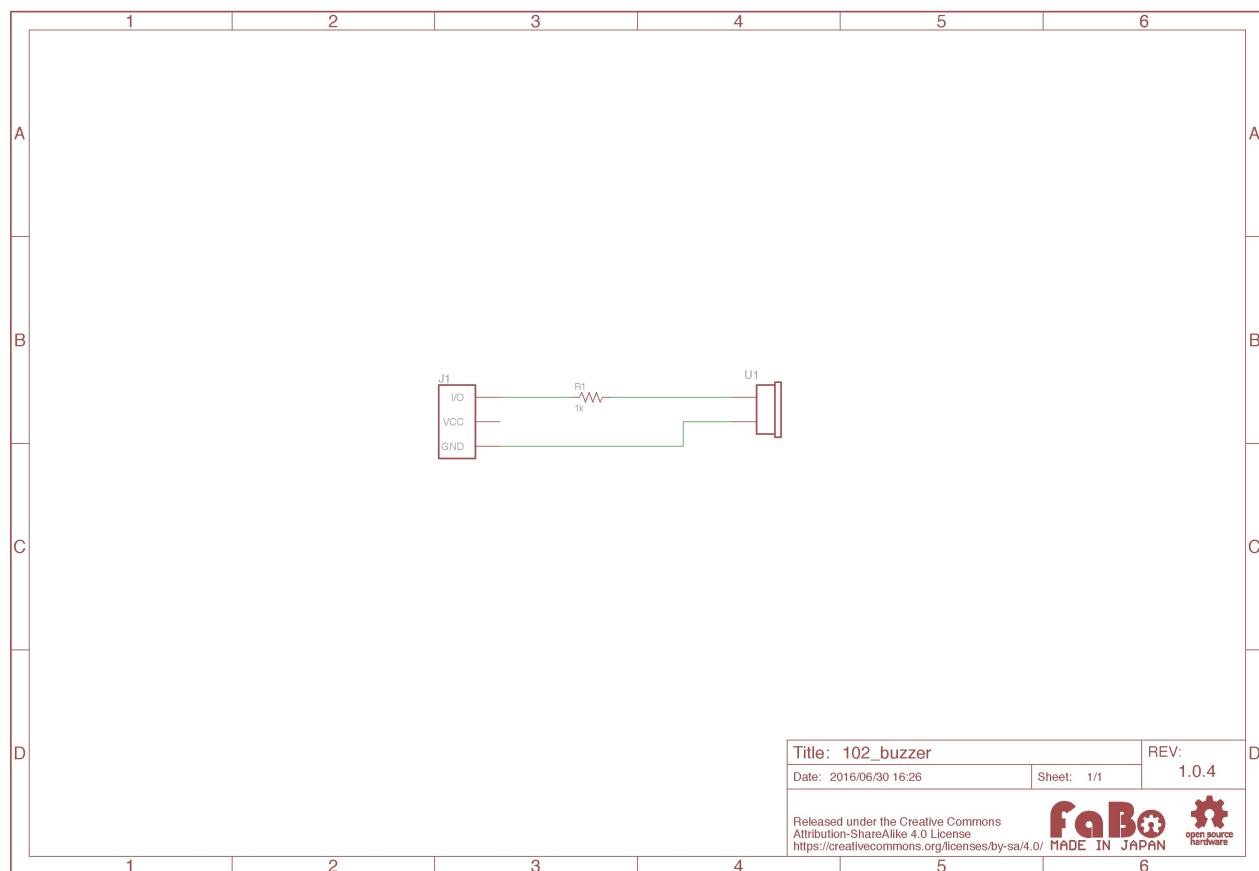
圧電ブザーを使ったBrickです。I/Oピンより、鳴らす音や音の長さを制御することができます。

Connecting

アナログコネクタ(A0～A5)、またはデジタルコネクタ(2～13)のいずれかに接続します。



Schematic



Sample Code

D2コネクタにBuzzer Brickを接続し、ビープ音を鳴らしています。

```

//  

// FaBo Brick Sample  

//  

// #102 Buzzer Brick  

//  
  

#define buzzerPin 2 // ブザーピンの設定  
  

int duration = 500; // 音を鳴らす時間  
  

void setup() {  

    // ブザーピンを出力用に設定  

    pinMode(buzzerPin,OUTPUT);  

}  
  

void loop() {  

    tone(buzzerPin,262,duration); // ド  

    delay(duration);  
  

    tone(buzzerPin,294,duration); // レ  

    delay(duration);  
  

    tone(buzzerPin,330,duration); // ミ  

    delay(duration);  
  

    delay(1000);
}

```

tone関数にて出力できる音階と周波数は下記のようになります。

数値が大きくなるほど音が高くなります。

	ド	ド♯	レ	レ♯	ミ	ファ	ファ♯	ソ	ソ♯
1	131	139	147	156	165	175	185	196	208
2	262	277	294	311	330	349	370	392	415
3	523	554	587	622	659	698	740	784	831

その他の音階については下記をご参照下さい。

<https://www.arduino.cc/en/Tutorial/ToneMelody?from=Tutorial.Tone>

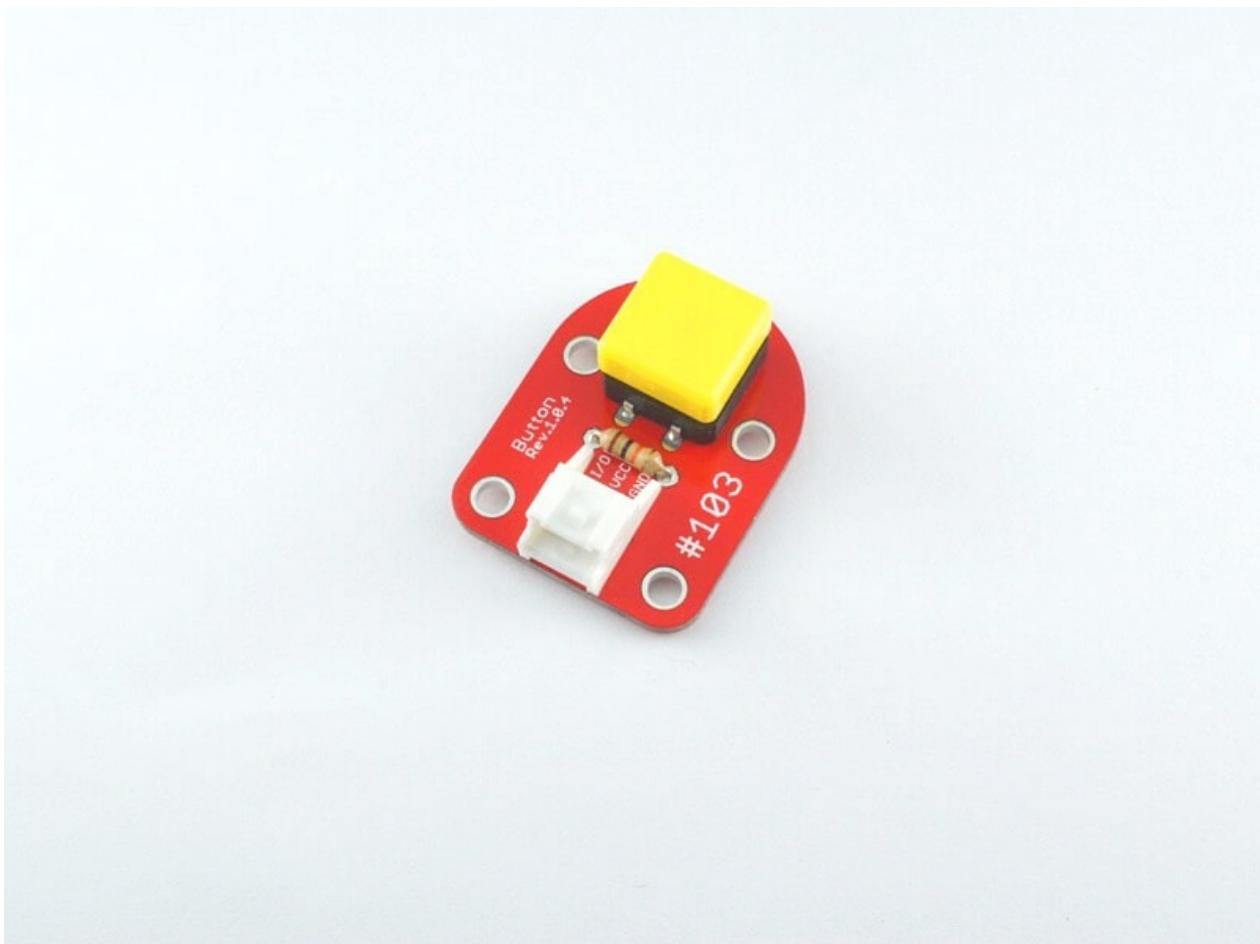
構成Parts

- 圧電ブザー

GitHub

- https://github.com/FaBoPlatform/FaBo/tree/master/102_buzzer

#103 Button Brick



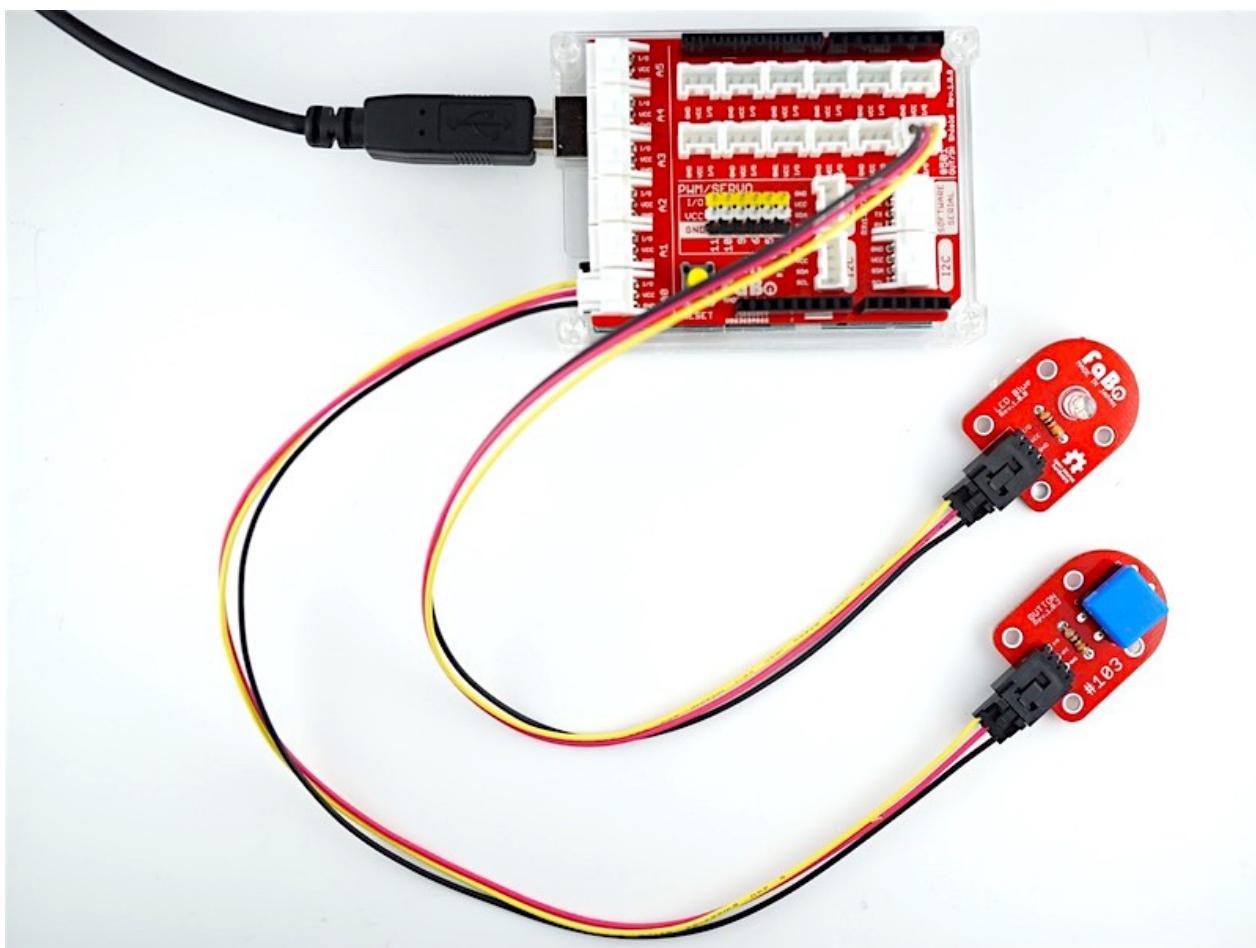
Overview

ボタンを使ったBrickです。I/OピンよりボタンのON/OFFの状態を取得することができます。

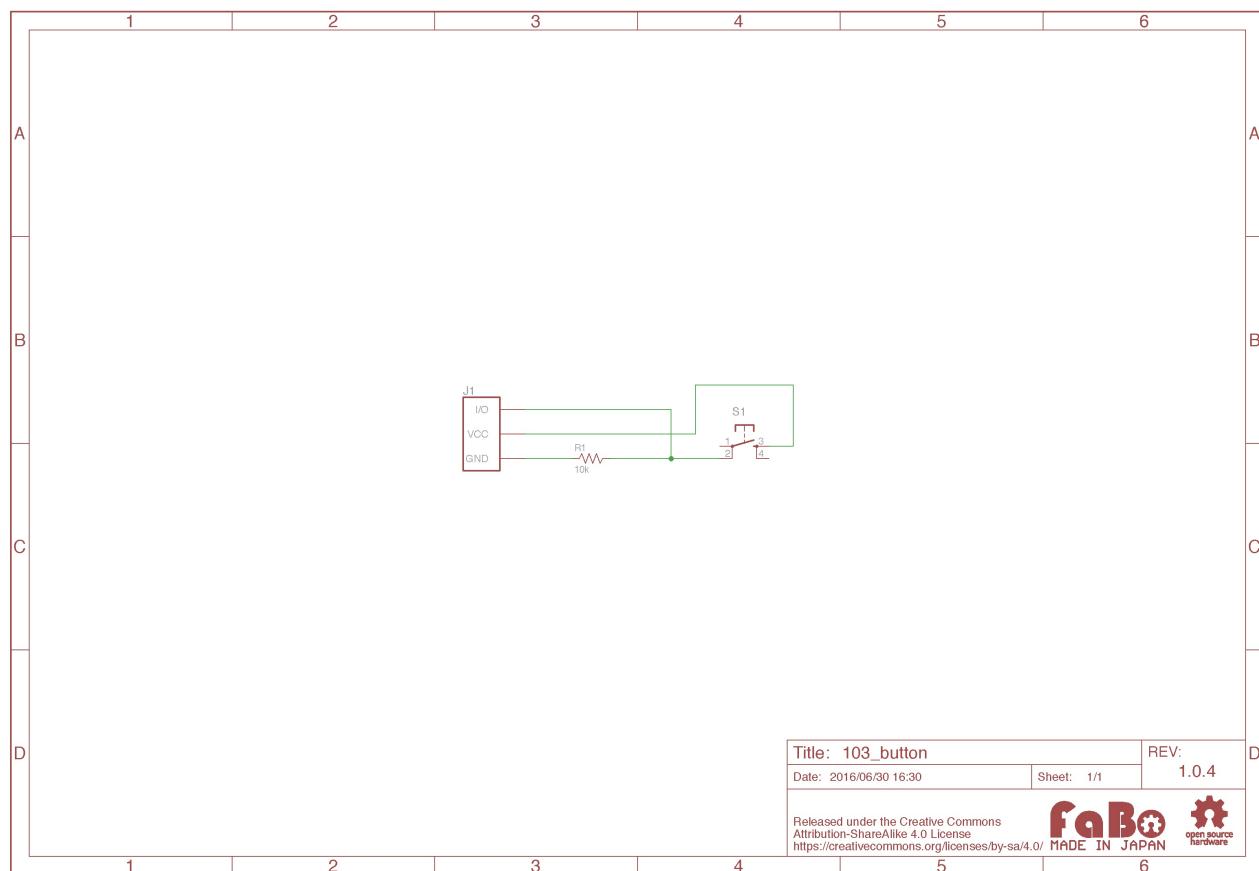
※ボタンカバー部分の色はランダムで送付するため色のご指定はできません。あらかじめご了承ください。

Connecting

アナログコネクタ(A0～A5)、またはデジタルコネクタ(2～13)のいずれかに接続します。



Schematic



Sample Code

A0コネクタに接続したButton Brickの入力により、D2コネクタに接続したLED Brick の点灯/消灯を制御しています。

```

//  

// FaBo Brick Sample  

//  

// #103 Button Brick  

//  

#define buttonPin A0 // ボタンピン  

#define ledPin 2      // LEDピン  

// ボタンの押下状況取得用  

int buttonState = 0;  

void setup() {  

    // ボタンピンを入力用に設定  

    pinMode(buttonPin, INPUT);  

    // LEDピンを出力用に設定  

    pinMode(ledPin, OUTPUT);  

}  

void loop(){  

    // ボタンの押下状況を取得  

    buttonState = digitalRead(buttonPin);  

    // ボタン押下判定  

    if (buttonState == HIGH) {  

        // ボタンが押された場合、LED点灯  

        digitalWrite(ledPin, HIGH);  

    }  

    else {  

        // LED消灯  

        digitalWrite(ledPin, LOW);  

    }  

}

```

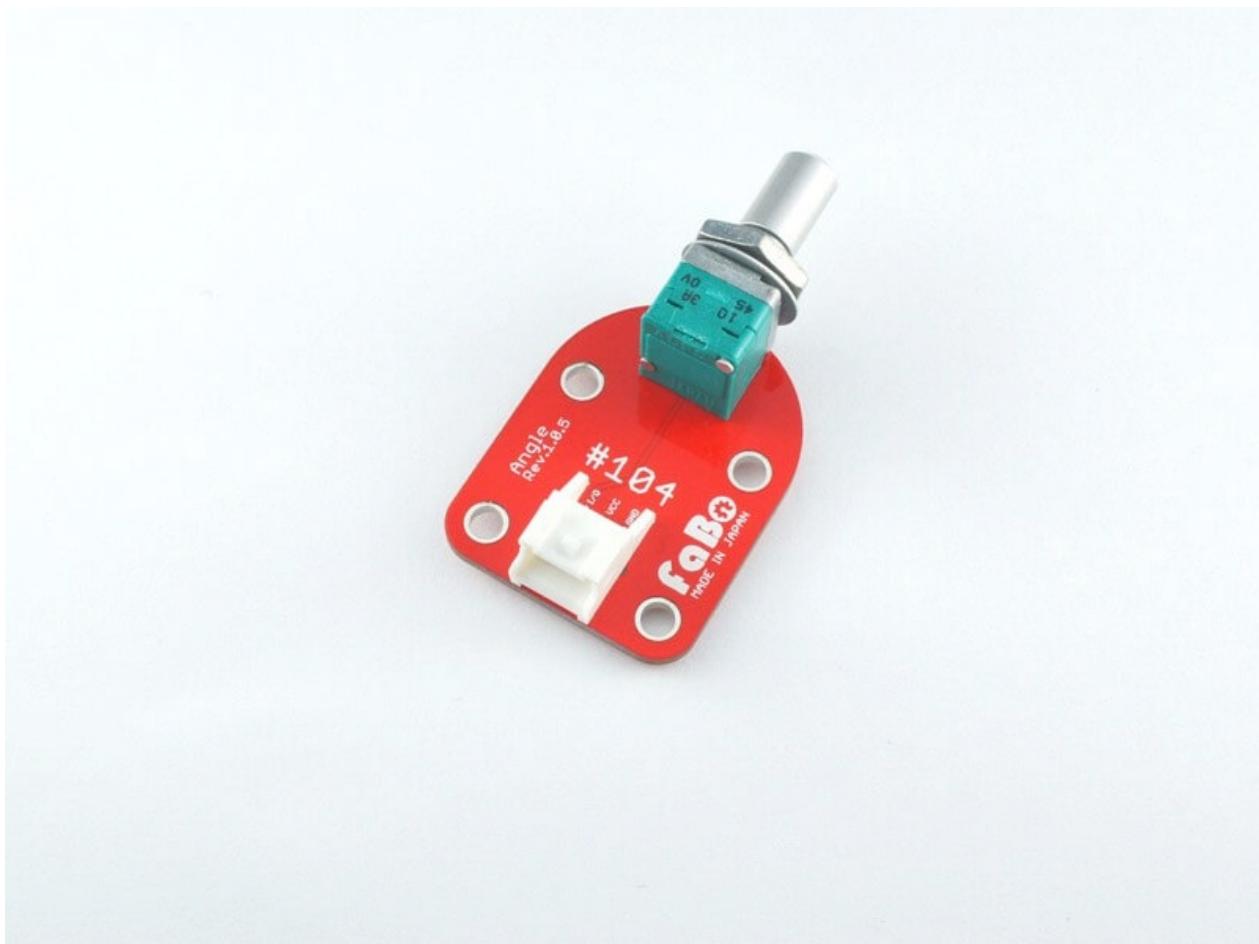
構成Parts

- 12mm角タクトスイッチ

GitHub

- https://github.com/FaBoPlatform/FaBo/tree/master/103_button

#104 Angle Brick



Overview

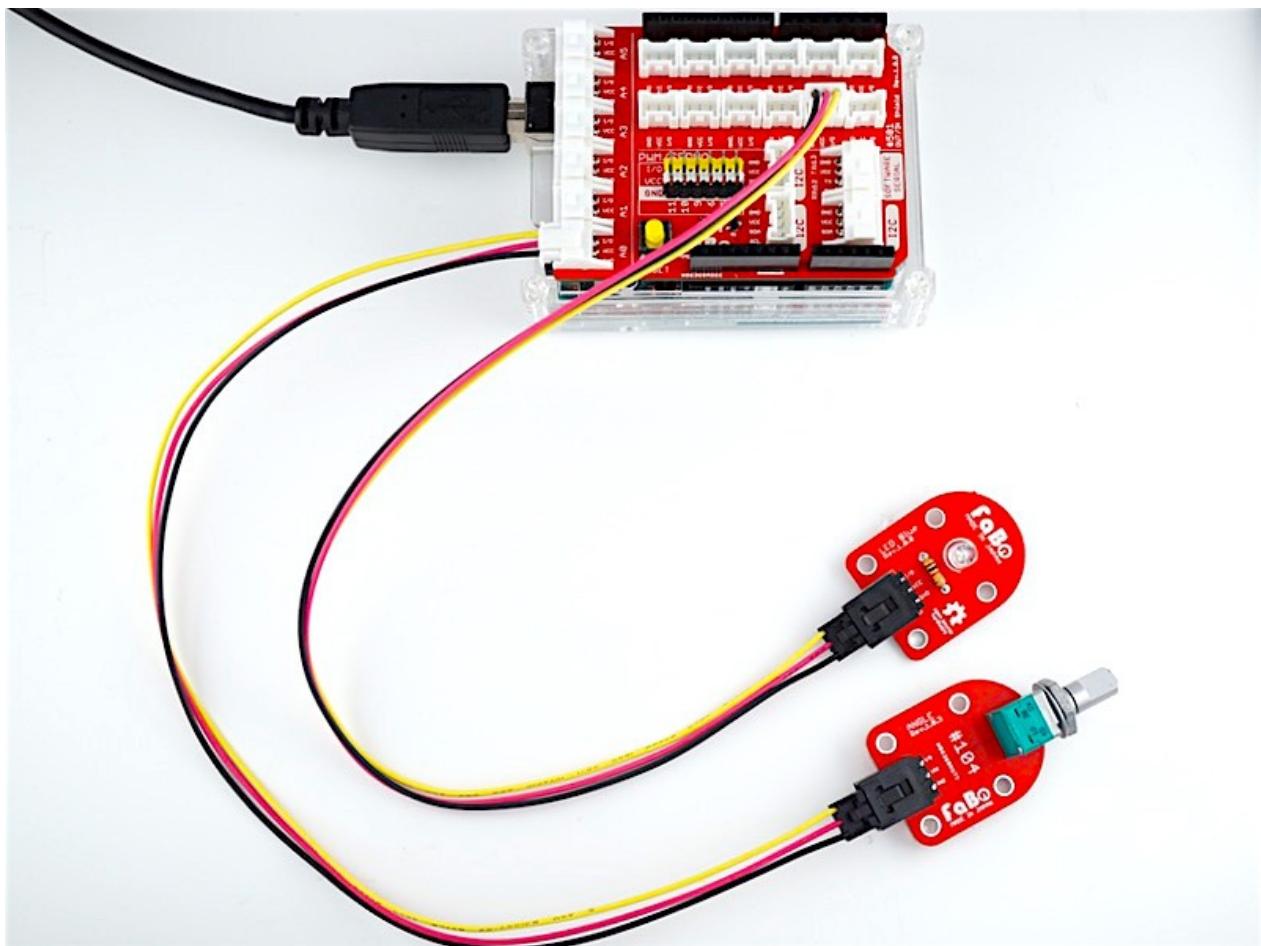
ボリューム抵抗を使ったBrickです。

I/Oピンからアナログ値を取得することができます。

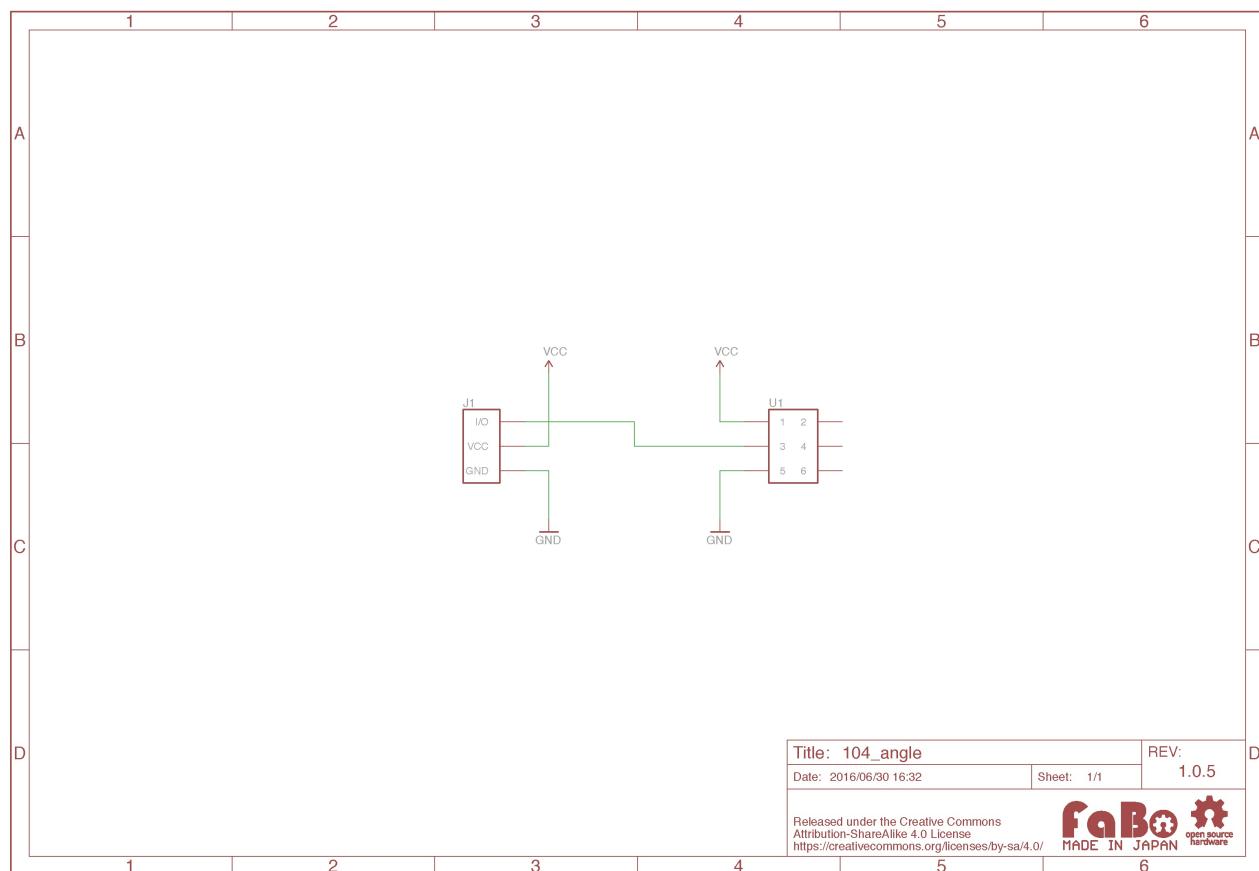
LED Brickの明るさを調節する際などに使用します。

Connecting

アナログコネクタ(A0～A5)のいずれかに接続します。



Schematic



Sample Code

A0コネクタにAngleを接続して、D3コネクタに接続したLED Brickの明るさ調節に使用しています。 LED Birckの色の明るさを変えるには、PWMの端子につなぐ必要があり、今回はD3に接続しています。

なお、FaBo Outin ShieldのPWM対応端子は

- D3
- D5
- D6
- D9
- D10
- D11

になっています。(Arduinoと同じ)

```
//  
// FaBo Brick Sample  
//  
// #104 Angle Brick  
//  
  
#define anglePin A0 // Angleピン  
#define ledPin 3    // LEDピン  
  
int angleValue = 0;  
int outputValue = 0;  
  
void setup() {  
    // Angleピンを入力用に設定  
    pinMode(anglePin, INPUT);  
    // LEDピンを出力用に設定  
    pinMode(ledPin, OUTPUT);  
}  
  
void loop() {  
    // Angleから値を取得(0~1023)  
    angleValue = analogRead(anglePin);  
    // analogWrite用に取得した値を変換  
    outputValue = map(angleValue, 0, 1023, 0, 255);  
    // PWMによりLED点灯  
    analogWrite(ledPin, outputValue);  
}
```

構成Parts

- ボリューム抵抗器A 10k

GitHub

- https://github.com/FaBoPlatform/FaBo/tree/master/104_angle

#105 Vibrator Brick



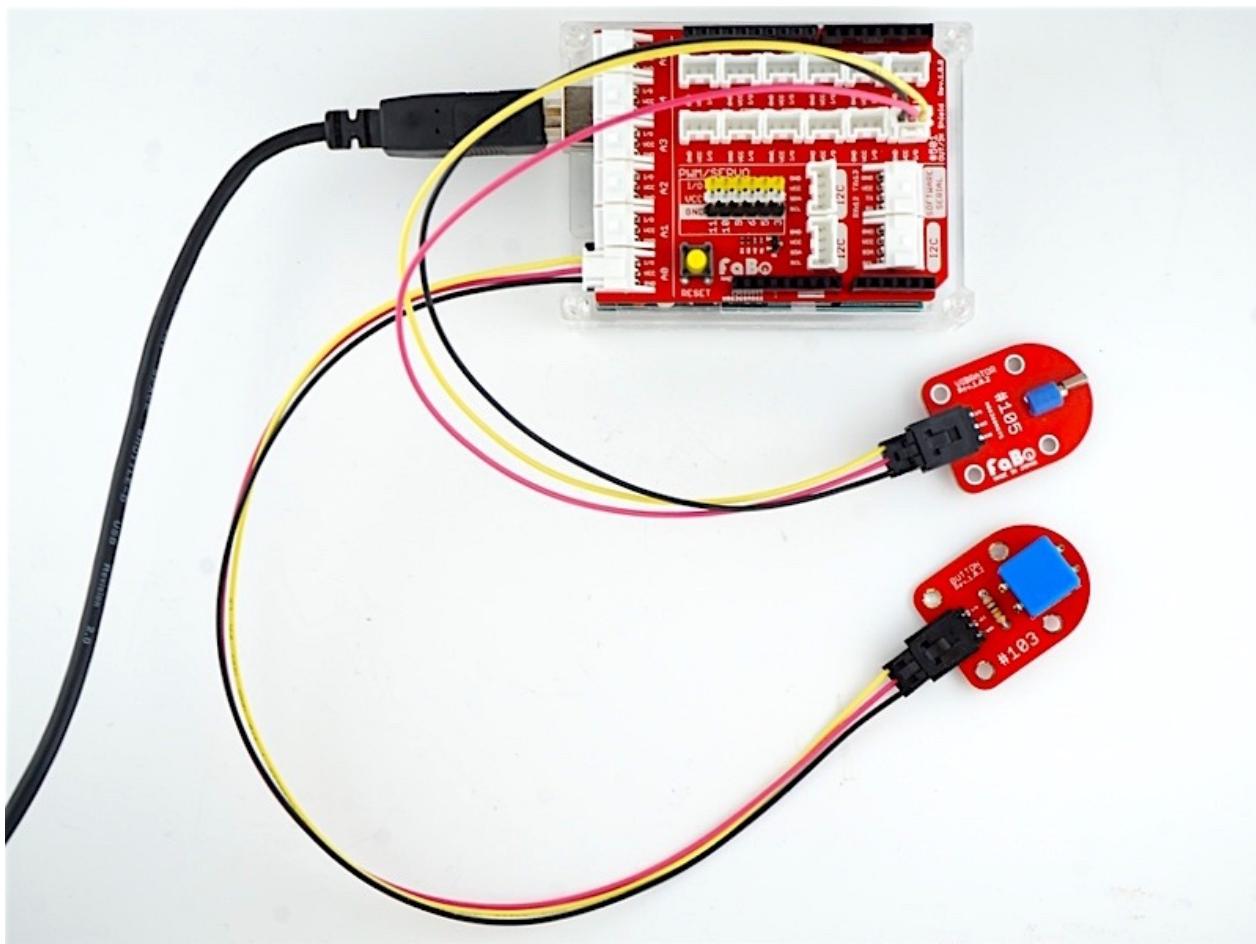
Overview

振動モーターを使用したBrickです。

I/Oピンから振動モーターのON/OFFを制御することができます。

Connecting

アナログコネクタ(A0～A5)、またはデジタルコネクタ(2～13)のいずれかに接続します。

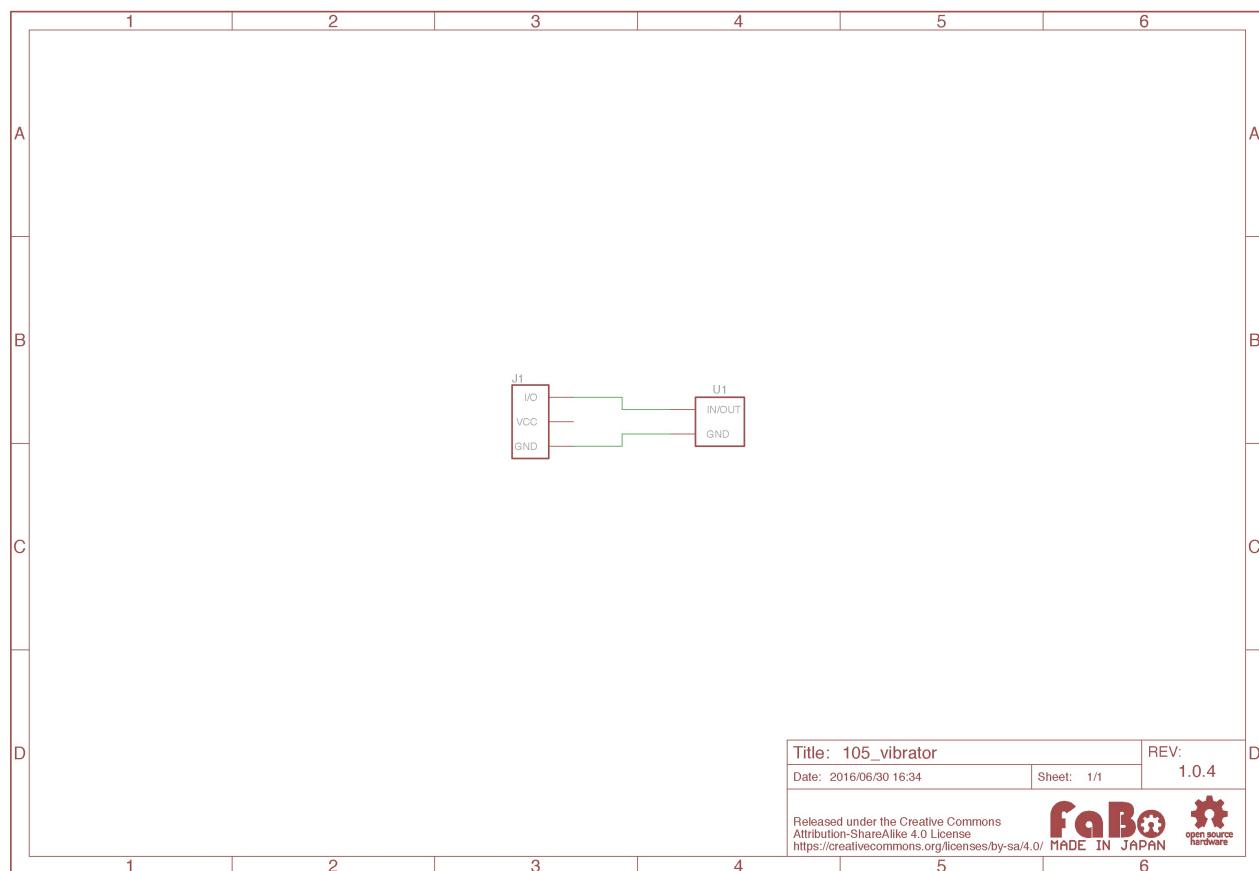


Parts Specification

Document

LA3R5-480AH1

Schematic



Sample Code

A0コネクタに接続したButton Brickの入力により、D2コネクタに接続したVibrator BrickのON/OFFを制御しています。

```

//  

// FaBo Brick Sample  

//  

// #105 Vibrator Brick  

//  

#define vibratorPin 2 // Vibratorピン  

#define buttonPin A0 // ボタンピン  

int buttonState = 0;  

void setup() {  

    // Vibratorピンを出力用に設定  

    pinMode(vibratorPin, OUTPUT);  

    // ボタンピンを入力用に設定  

    pinMode(buttonPin, INPUT);  

}  

void loop(){  

    // ボタンの押下状況を取得  

    buttonState = digitalRead(buttonPin);  

    // ボタン押下判定  

    if (buttonState == HIGH) {  

        // ボタンが押された場合、Vibratorオン  

        digitalWrite(vibratorPin, HIGH);  

    }  

    else {  

        // Vibratorオフ  

        digitalWrite(vibratorPin, LOW);  

    }  

}

```

構成Parts

- 振動モーター LA3R5-480AH1

GitHub

- https://github.com/FaBoPlatform/FaBo/tree/master/105_vibrator

#108 Temperature Brick



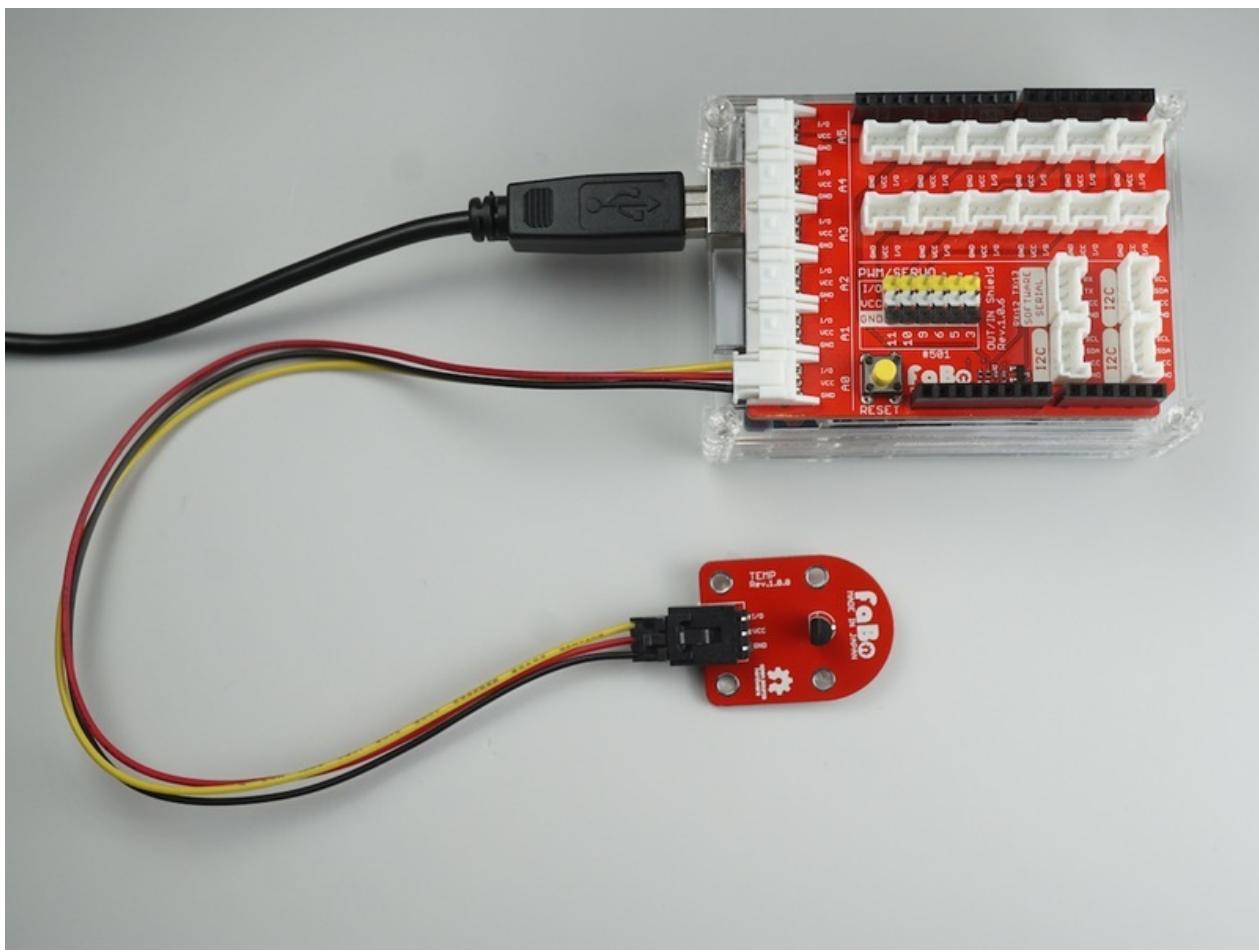
Overview

温度を計測するBrickです。

アナログ値(0～1023)を取得でき、変換することで-30度から100度までの温度を計測することができます。

Connecting

アナログコネクタ(A0～A5)のいずれかに接続します。

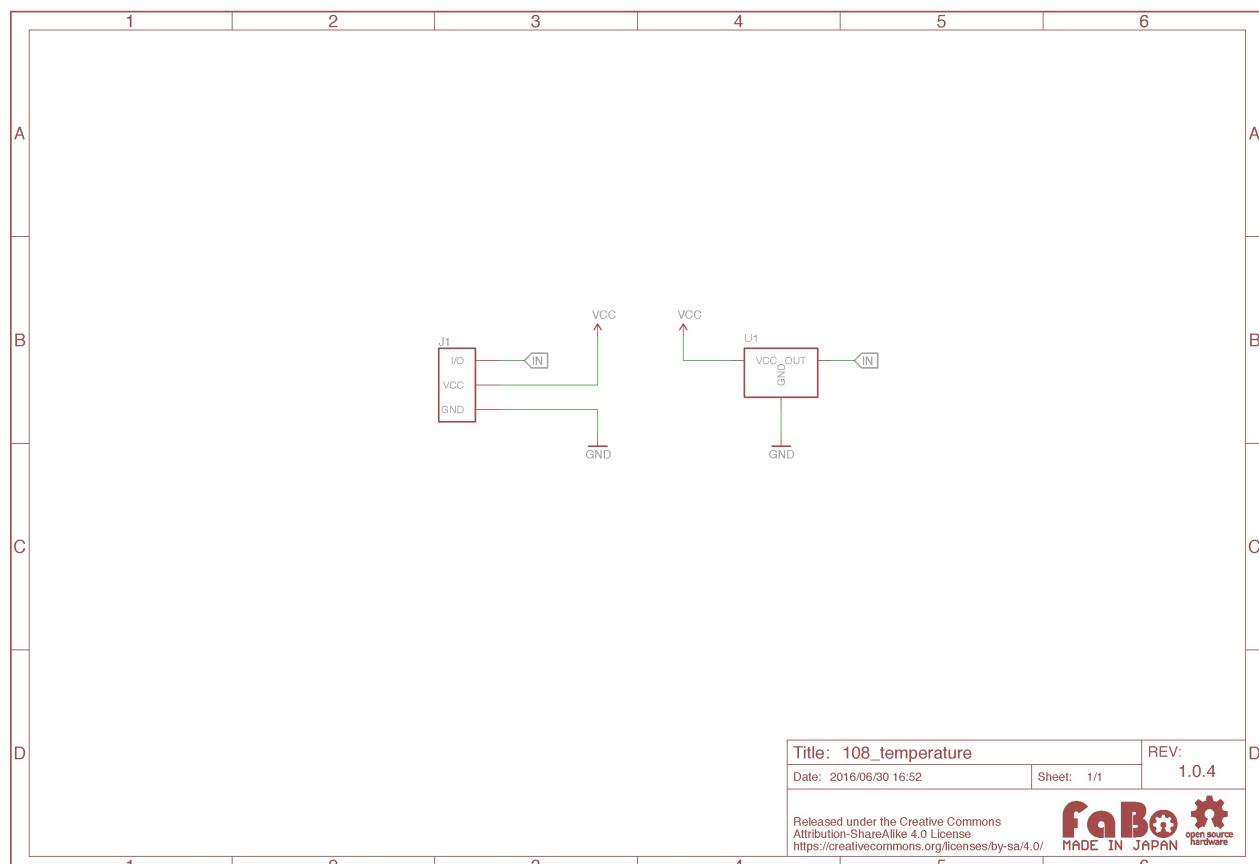


LM61CIZ Datasheet

Document

[LM61CIZ Datasheet](#)

Schematic



Sample Code

A0コネクタにTemperature Brickを接続して、取得した温度をシリアルモニタへ出力します。

```

//  

// FaBo Brick Sample  

//  

// #108 Temperature Brick  

//  

#define tempPin A0 // 温度センサーピン  

int tempValue = 0; // 温度取得用  

void setup() {  

    // 温度センサーピンを入力用に設定  

    pinMode(tempPin, INPUT);  

    // シリアル開始 転送レート：9600bps  

    Serial.begin(9600);  

}  

void loop() {  

    // センサーより値を取得(0~1023)  

    tempValue = analogRead(tempPin);  

    // 取得した値を電圧に変換 (0~5000mV)  

    tempValue = map(tempValue, 0, 1023, 0, 5000);  

    // 変換した電圧を300~1600の値に変換後、温度に変換 (-30~100度)  

    tempValue = map(tempValue, 300, 1600, -30, 100);  

    // 算出した温度を出力  

    Serial.println(tempValue);  

    delay(100);  

}

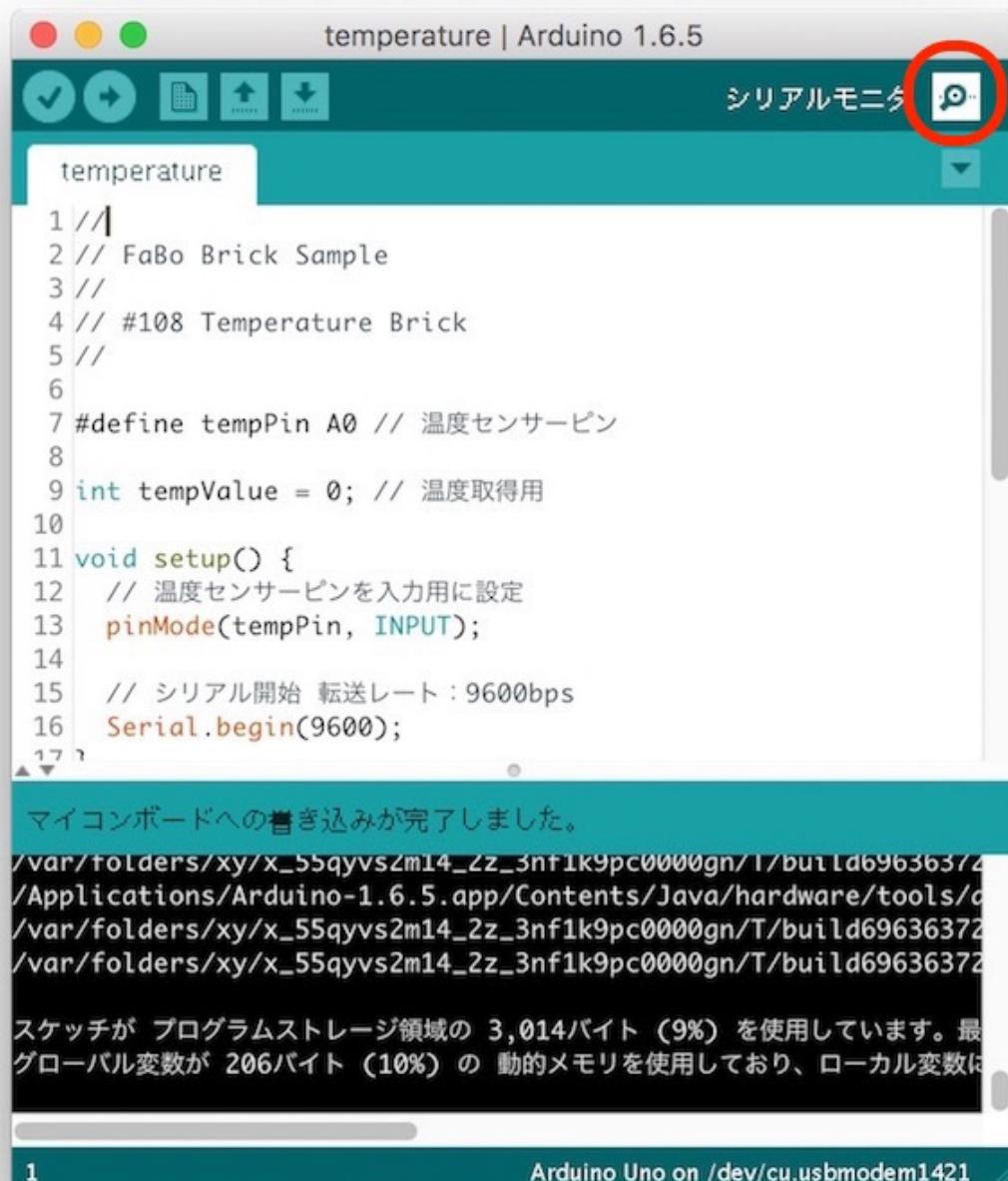
```

出力データの確認方法

Serial.printlnなどで出力した内容はシリアルモニタを使用して確認します。

シリアルモニタはArduinoIDEのメニューより[ツール]->[シリアルモニタ]を選択することで起動できます。

Arduinoのコードを書く画面の右上にある虫メガネマークをクリックしても起動することができます。



起動後、画面右下に転送レートを選択する箇所があるので、その箇所をコードに合わせて変更してください。



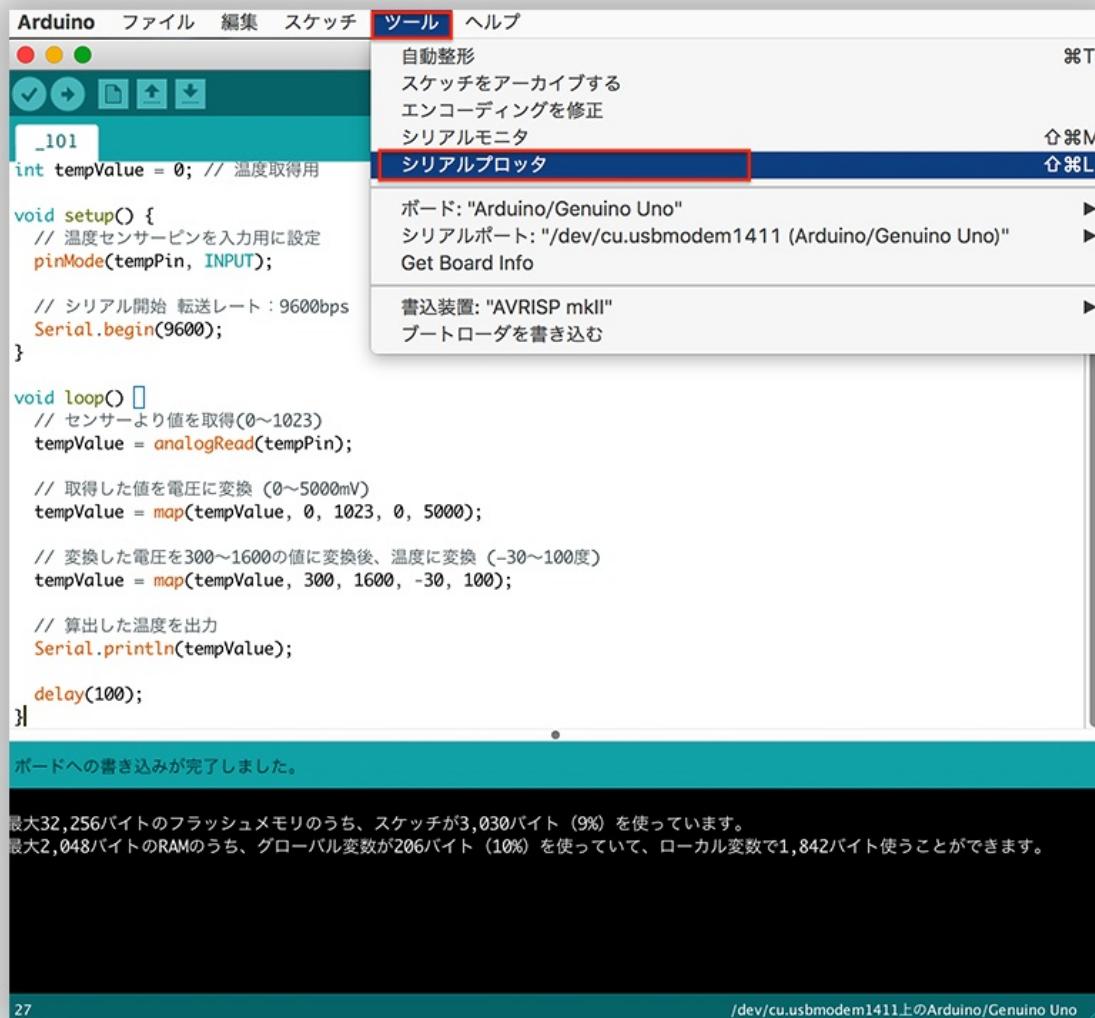
サンプルコードの転送レートを設定している箇所

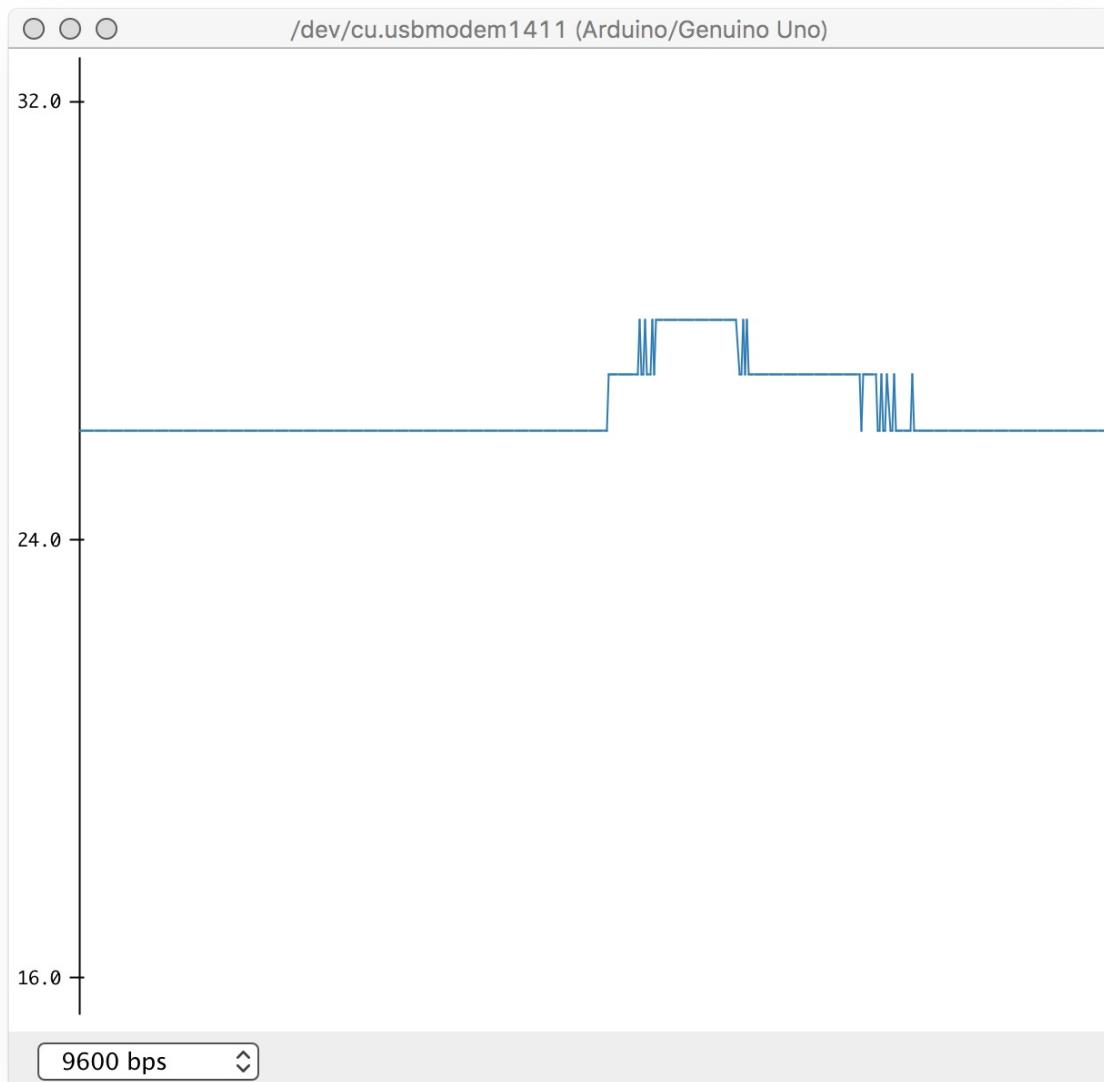
```
Serial.begin(9600);
```

シリアルプロッタ

AndroidIDEではシリアルプロッタでも値を確認することができます。

ArduinolDEのメニューより[ツール]->[シリアルプロッタ]を選択することで起動できます。





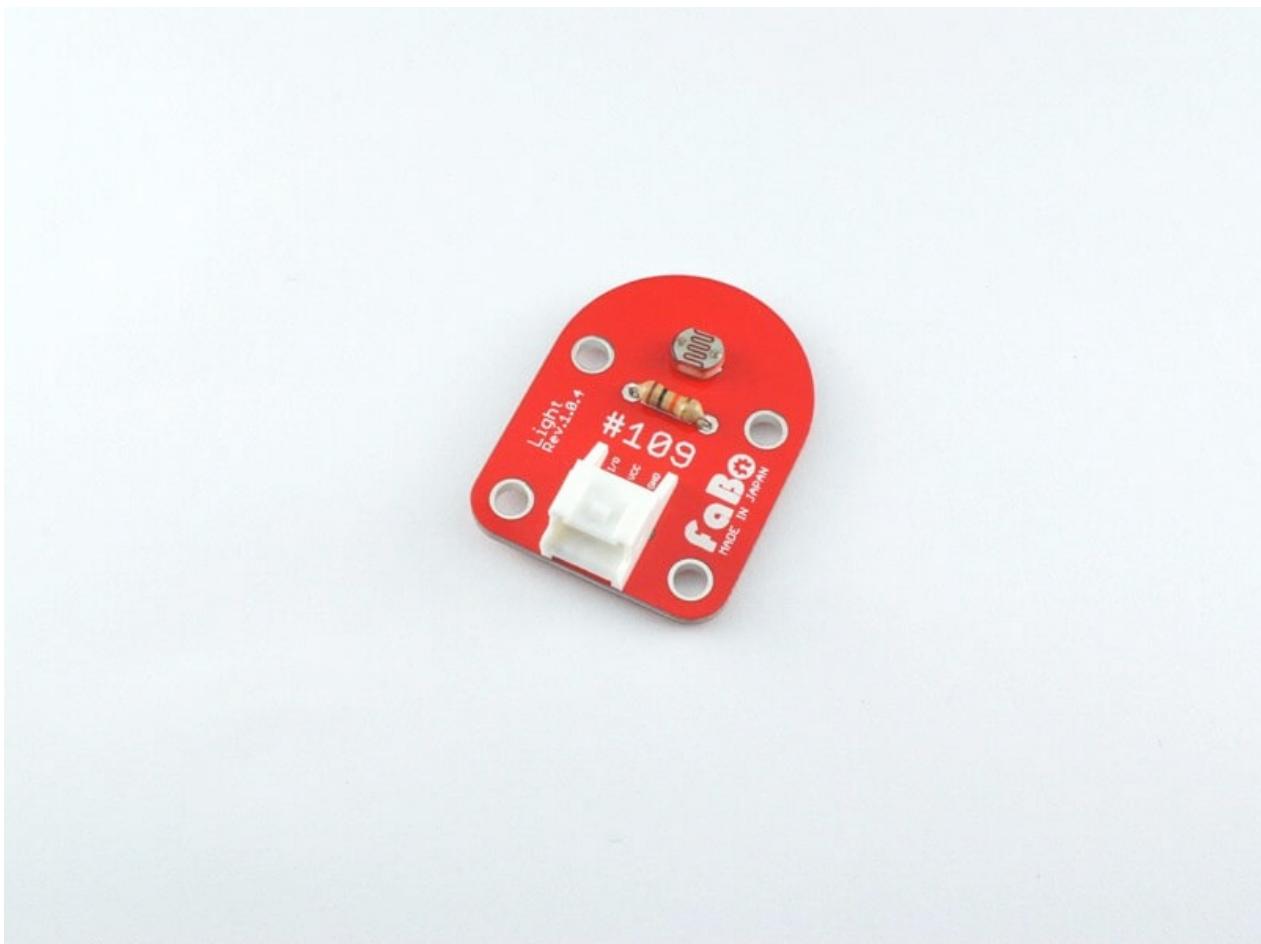
構成Parts

- IC温度センサ LM61CIZ

GitHub

- https://github.com/FaBoPlatform/FaBo/tree/master/108_temperature

#109 Light Brick



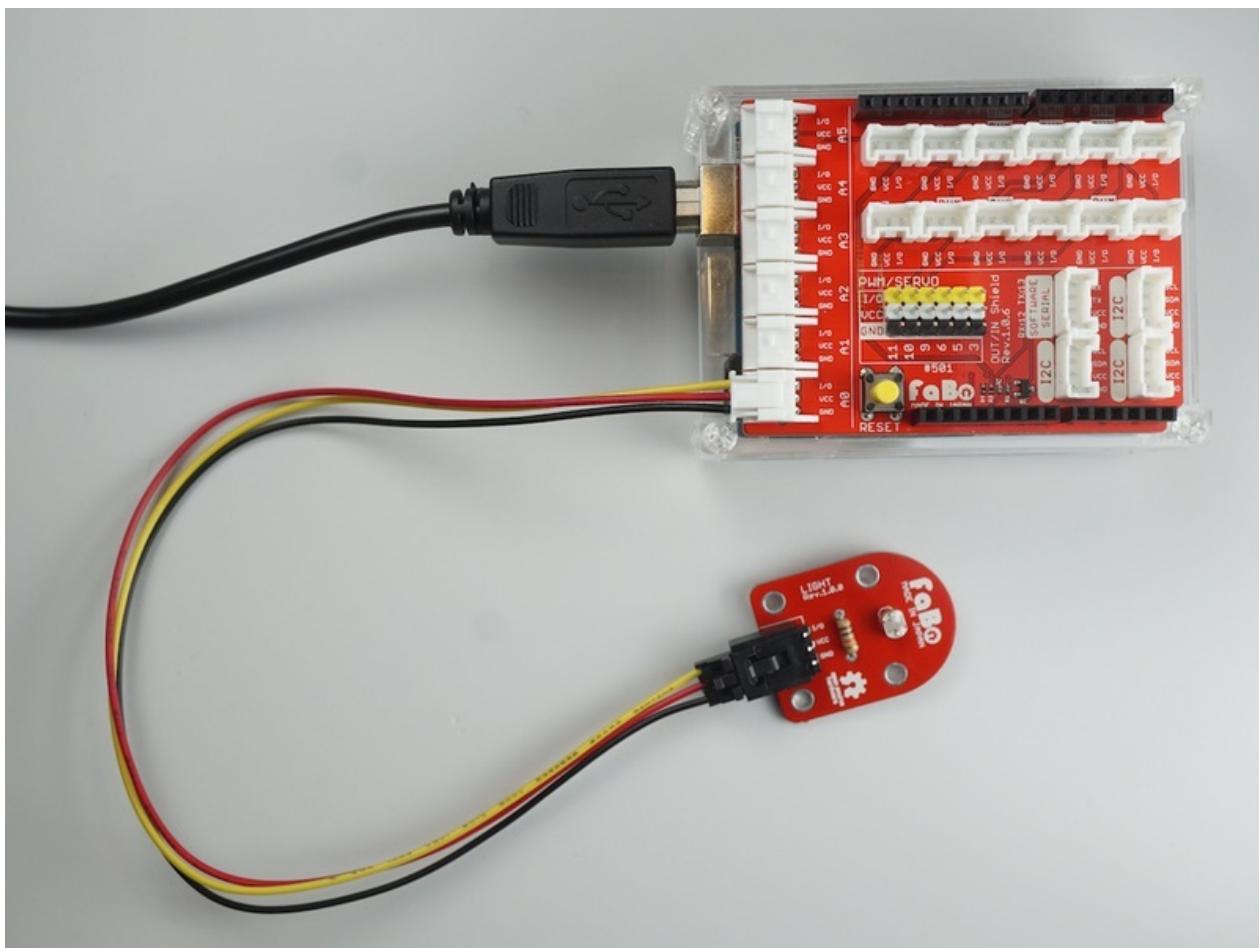
Overview

CDSセルを使用した光センサーBrickです。

周囲の明るさの変化をアナログ値として取得することができます。

Connecting

アナログコネクタ(A0～A5)のいずれかに接続します。

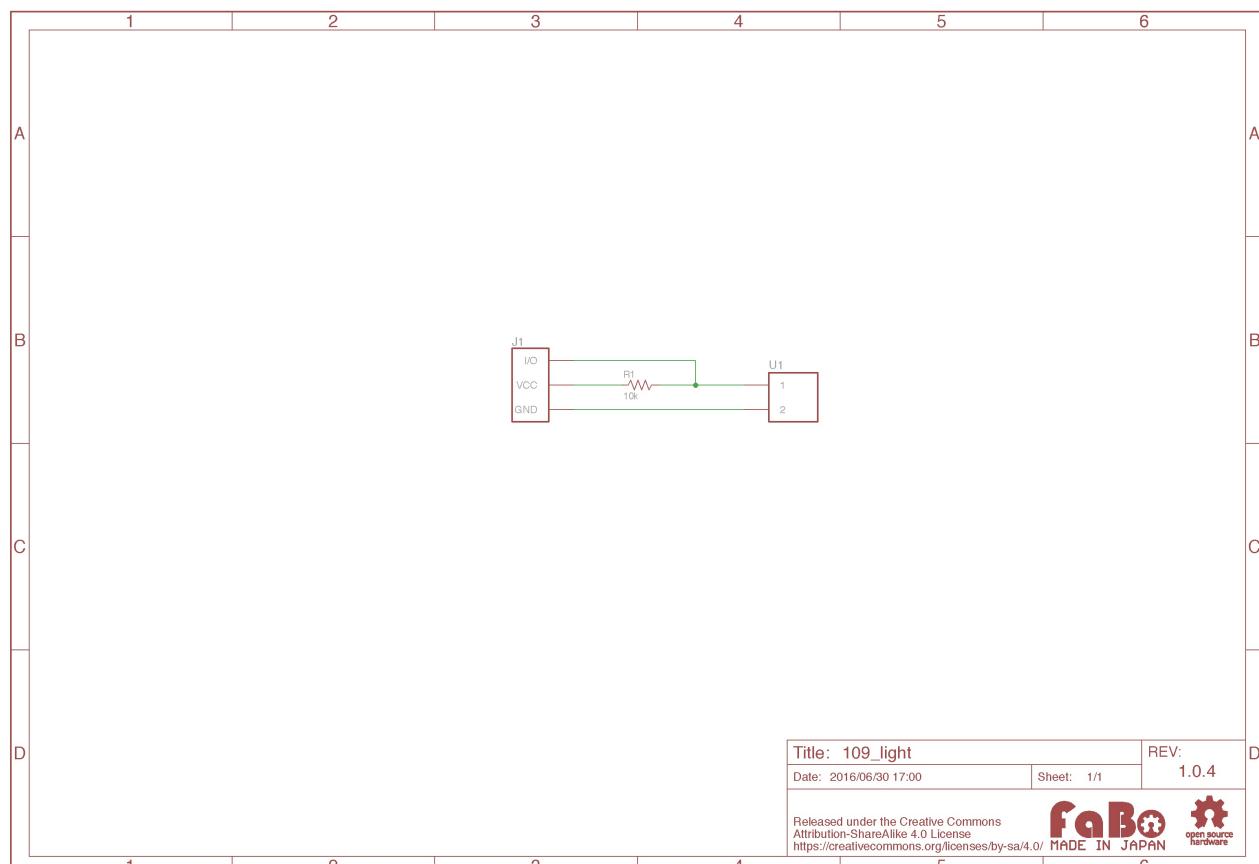


Parts Specification

Document

MI527

Schematic



Sample Code

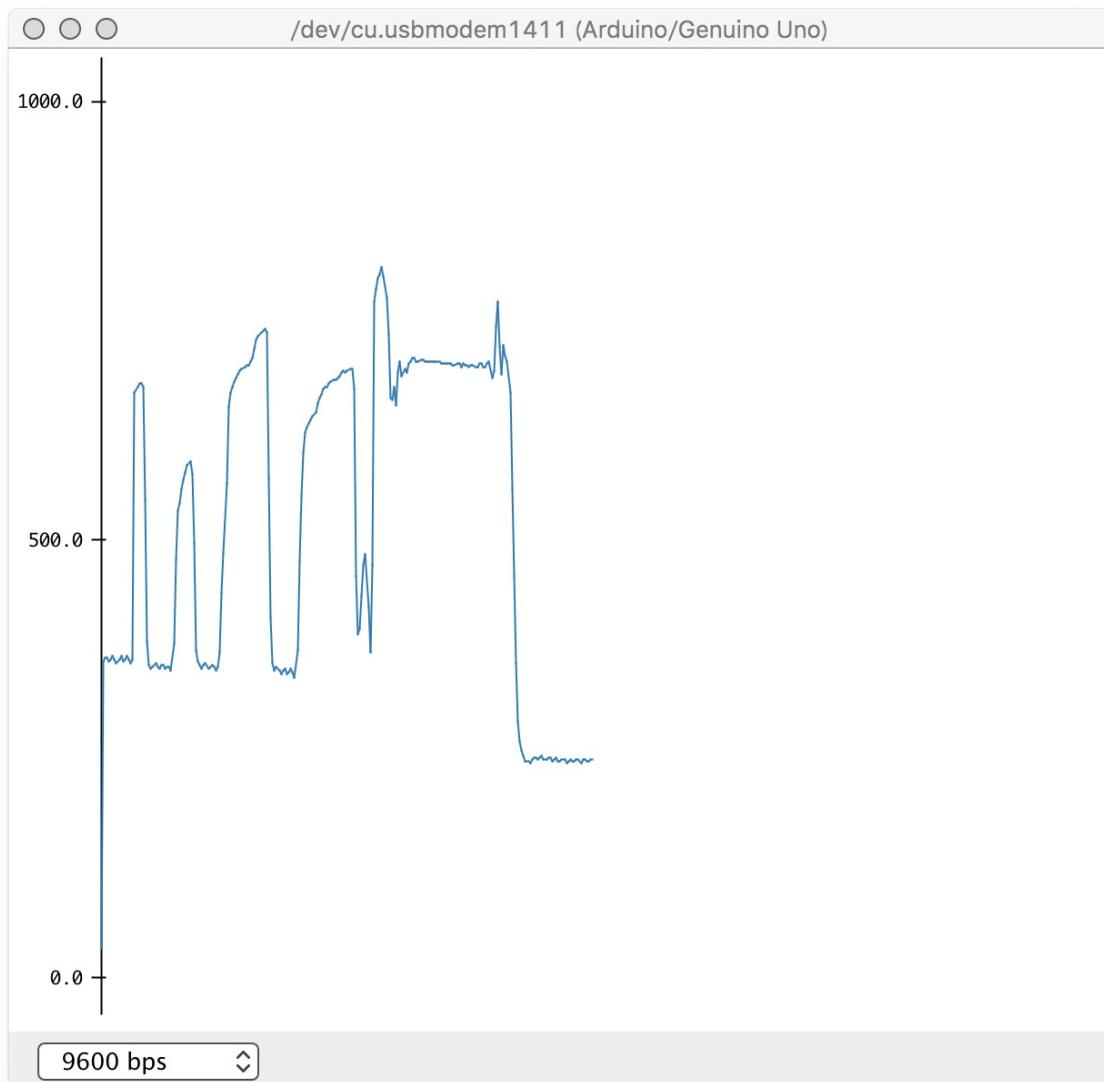
A0コネクタにLight Brickを接続して、明るさに応じたアナログ値をシリアルモニタへ出力します。

```
//  
// FaBo Brick Sample  
//  
// #109 Light Brick  
//  
  
#define lightPin A0  
  
int lightValue = 0;  
  
void setup() {  
    pinMode(lightPin,INPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
    lightValue = analogRead(lightPin);  
    Serial.println(lightValue);  
    delay(100);  
}
```

シリアルプロッタ

AndroidIDEではシリアルプロッタでも値を確認することができます。

ArduinoIDEのメニューより[ツール]->[シリアルプロッタ]を選択することで起動できます。



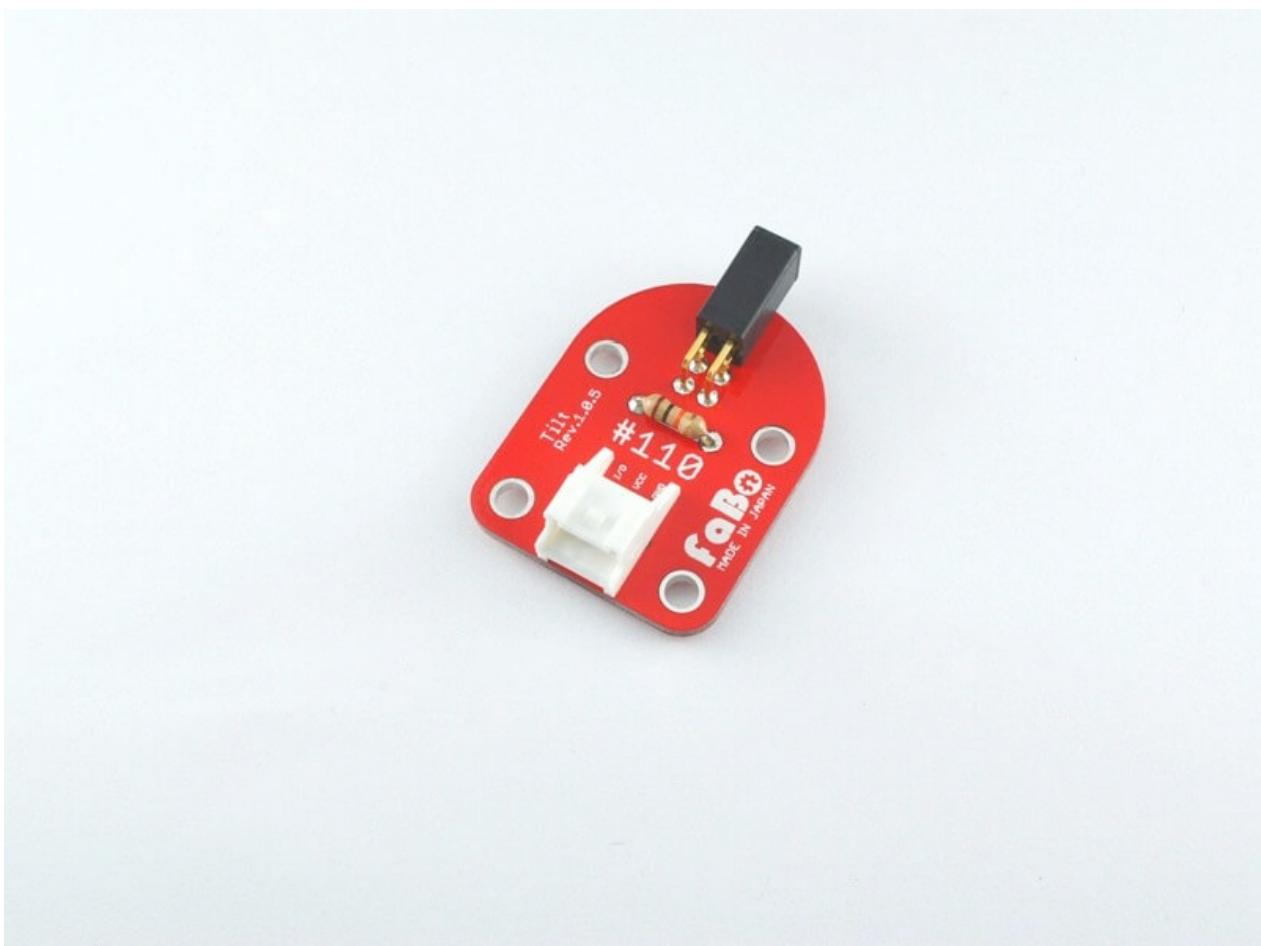
構成Parts

- CDSセル(5mm)

GitHub

- https://github.com/FaBoPlatform/FaBo/tree/master/109_light

#110 Tilt Brick



Overview

傾斜センサーを使用したBrickです。

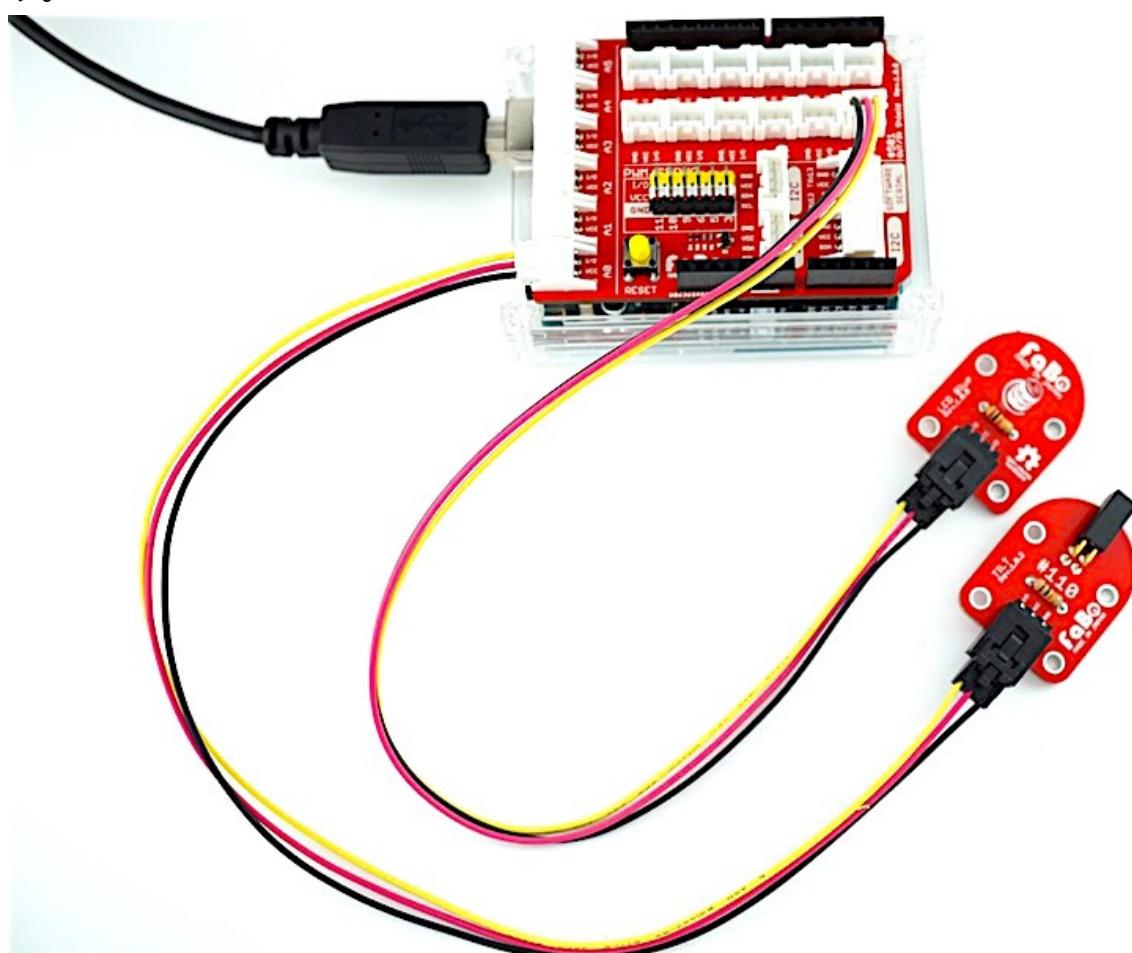
I/Oピンより傾斜センサーの状態をデジタル値(0～1)取得することができます。

黒い部分の中に玉が入っていて傾くとデジタル値が変化します。

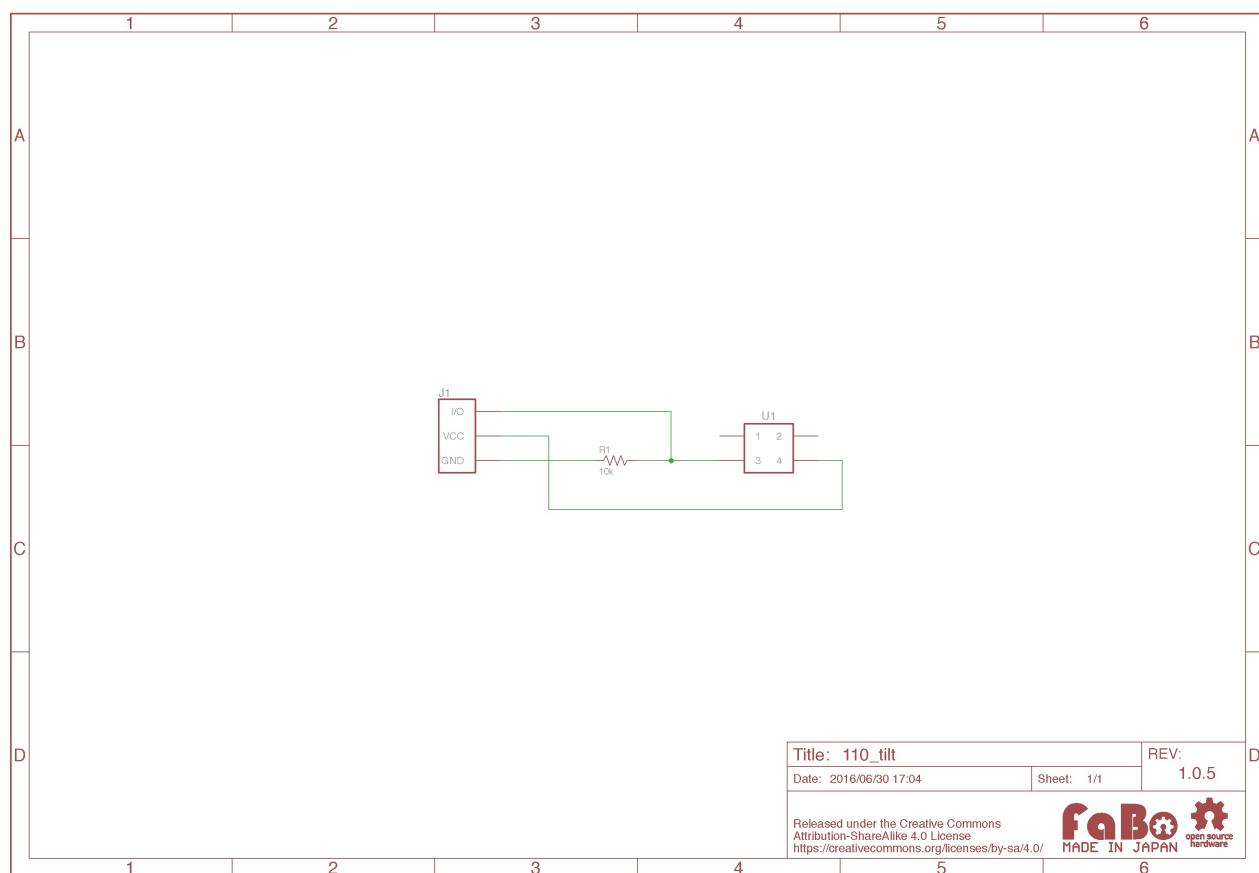
LED Brickを点灯/消灯させる際などに使用します。

Connecting

アナログコネクタ(A0～A5)、またはデジタルコネクタ(2～13)のいずれかに接続します。



Schematic



Sample Code

A0コネクタに接続したTilt Brickの傾きによって、D2コネクタに接続したLED Brickを点灯/消灯させています。

```
//  
// FaBo Brick Sample  
//  
// #110 Tilt Brick  
  
#define buttonPin A0  
#define ledPin 2  
  
int buttonState = 0;  
  
void setup() {  
    pinMode(buttonPin, INPUT);  
    pinMode(ledPin, OUTPUT);  
}  
  
void loop(){  
  
    buttonState = digitalRead(buttonPin);  
  
    if (buttonState == HIGH) {  
        digitalWrite(ledPin, HIGH);  
    }  
    else {  
        digitalWrite(ledPin, LOW);  
    }  
}
```

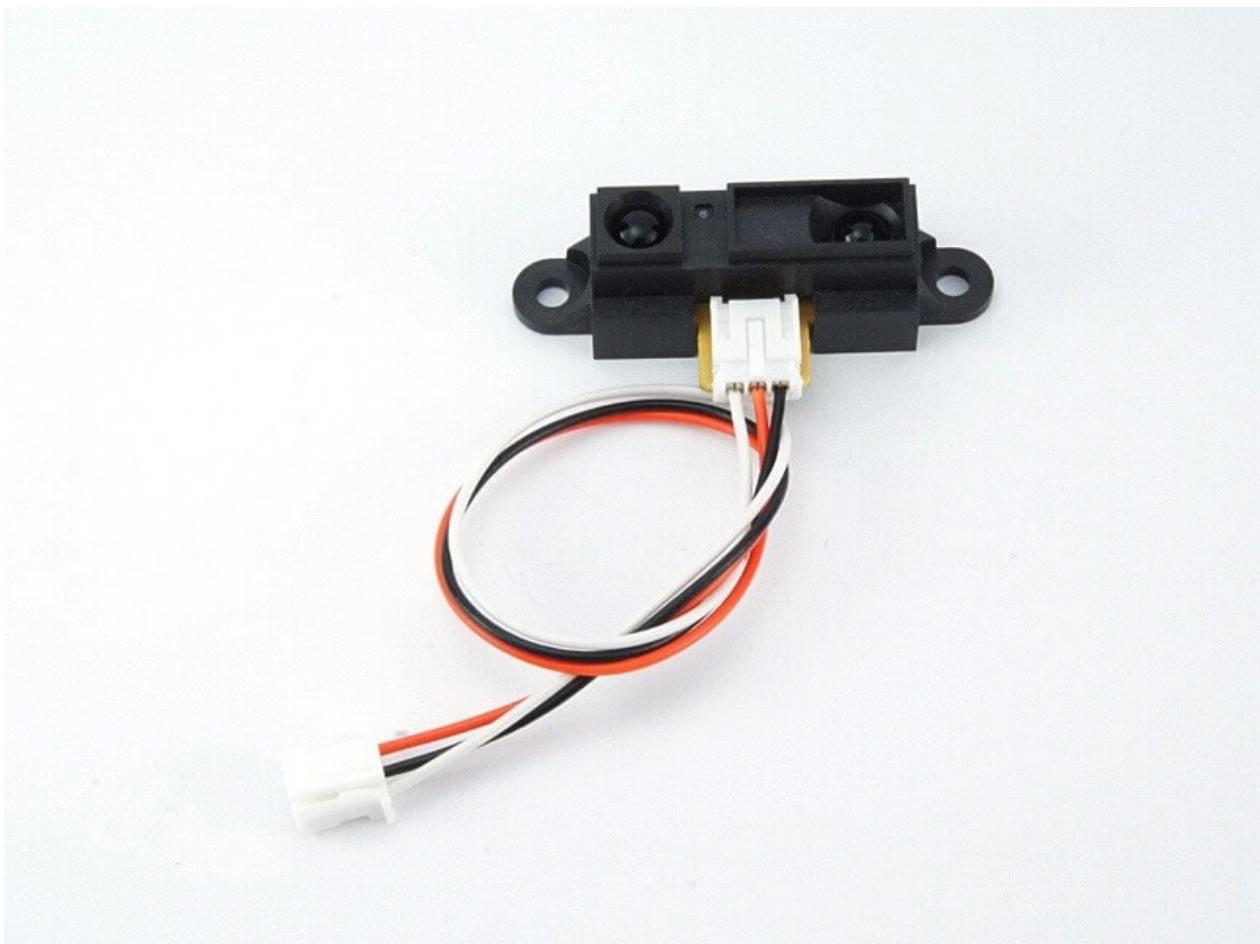
構成Parts

- 傾斜(振動)スイッチ

GitHub

- https://github.com/FaBoPlatform/FaBo/tree/master/110_tilt

#116 Distance Brick



Overview

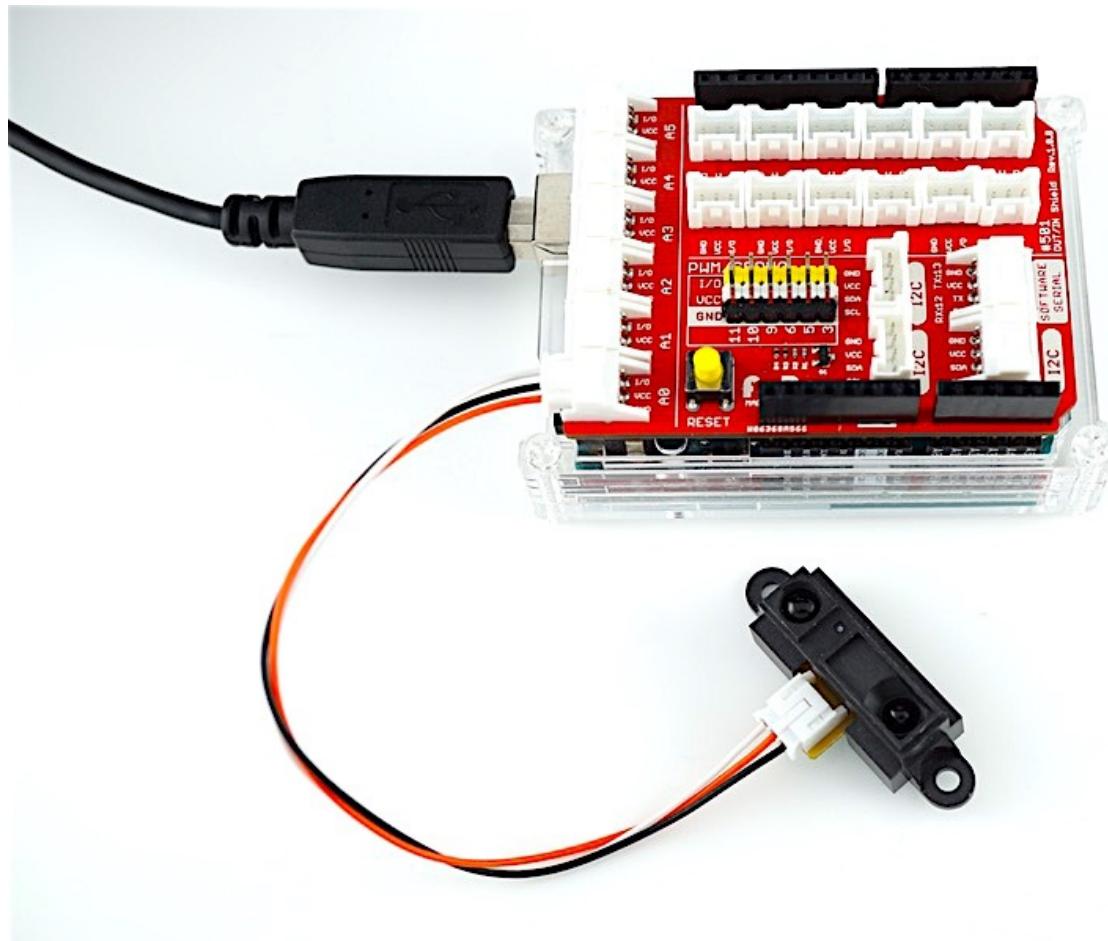
距離センサモジュールを使用したBrickです。

I/Oピンより距離センサーの正面についているレンズから物体までの距離をアナログ値(0~1023)で取得することができます。

測定可能な距離は10~80cmとなっています。

Connecting

アナログコネクタ(A0～A5)のいずれかに接続します。



GP2Y0A21YK Datasheet

Document

[GP2Y0A21YK Datasheet](#)

Sample Code

A0コネクタに接続して、距離を計測します。

```

//  

// FaBo Brick Sample  

//  

// #116 Distance Brick  

//  

#define distancePin A0 // 距離センサーピン  

int distanceValue = 0;  

void setup() {  

    // 距離センサーピンを入力用に設定  

    pinMode(distancePin, INPUT);  

    // シリアル開始 転送レート：9600bps  

    Serial.begin(9600);  

}  

void loop() {  

    // センサーより値を取得(0～1023)  

    distanceValue = analogRead(distancePin);  

    // 取得した値を電圧に変換 (0～5000mV)  

    distanceValue = map(distanceValue, 0, 1023, 0, 5000);  

    // 変換した電圧を3200(3.2v)～500(0.5v)の値に変換後、距離に変換 (10～80cm)  

    distanceValue = map(distanceValue, 3200, 500, 5, 80);  

    // 算出した距離を出力  

    Serial.println(distanceValue);  

    delay(100);  

}

```

構成Parts

- 距離センサーモジュール GP2Y0A21YK

GitHub

- https://github.com/FaBoPlatform/FaBo/tree/master/116_distance

#1501 Propeller Kit



注文を受けてから生産しますので、納期にお時間をいただく場合があります。

Overview

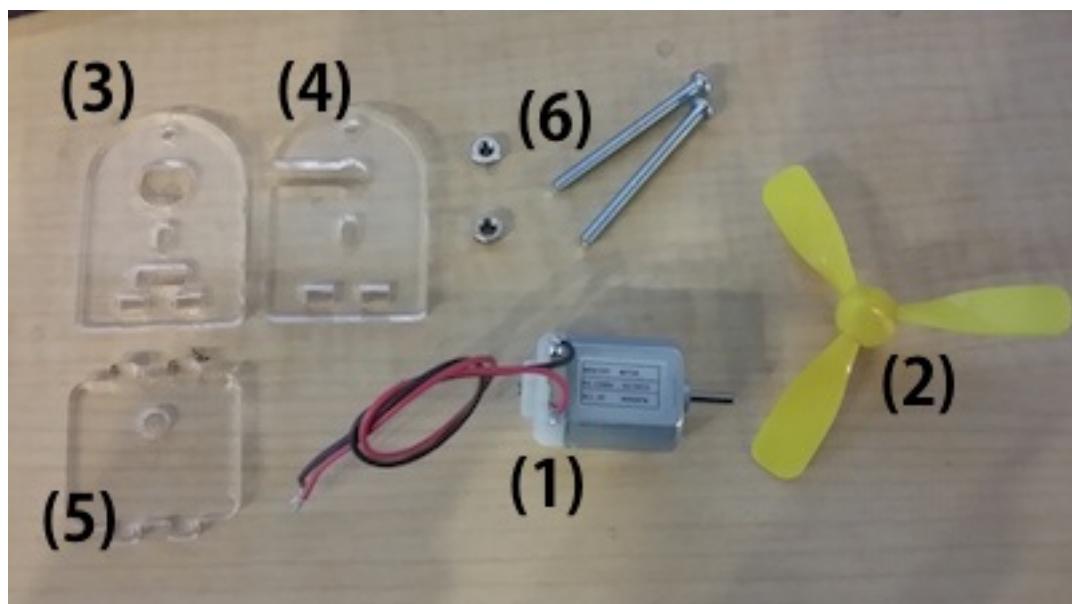
Propeller Kitは、Arduinoなどから制御できる扇風機のキットです。

デザインは予告なく変更される場合があります。

マニュアルで使われている写真のパーツと、実際のパーツの色や形が異なる場合があります。あらかじめご了承ください。

注意：ネジを必要以上にキツく締めるとパーツが破損する可能性があります。

パーツ構成 プロペラキット部分



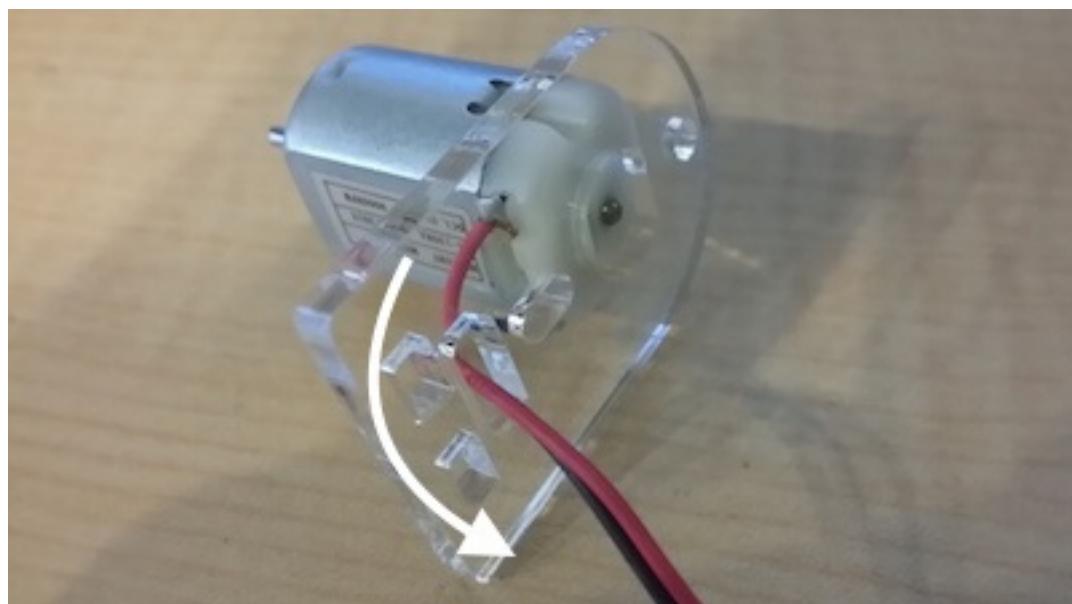
(1)DCモーター

(2)プロペラ

(3)～(5)アクリルパーツ

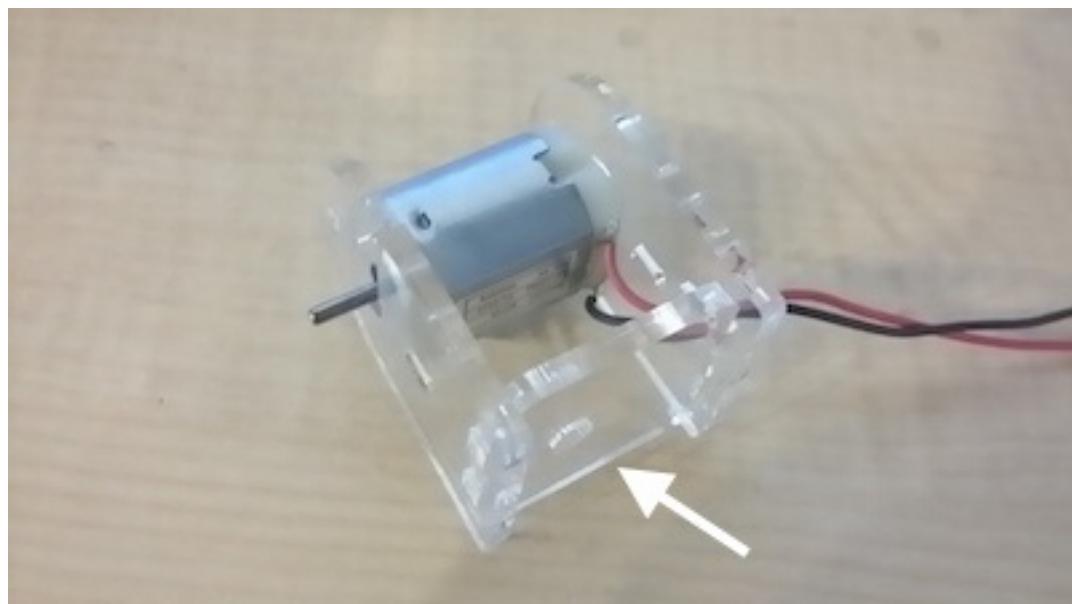
(6)M3-35mmネジ2本、M3ナット（鉄）2個

1.アクリルパーツ(3)と(4)でDCモーターとアクリルパーツ(5)を挟みます。



アクリルパーツ(3)をDCモーターの後ろからセットします。※DCモーターのリード線が下向きになります。

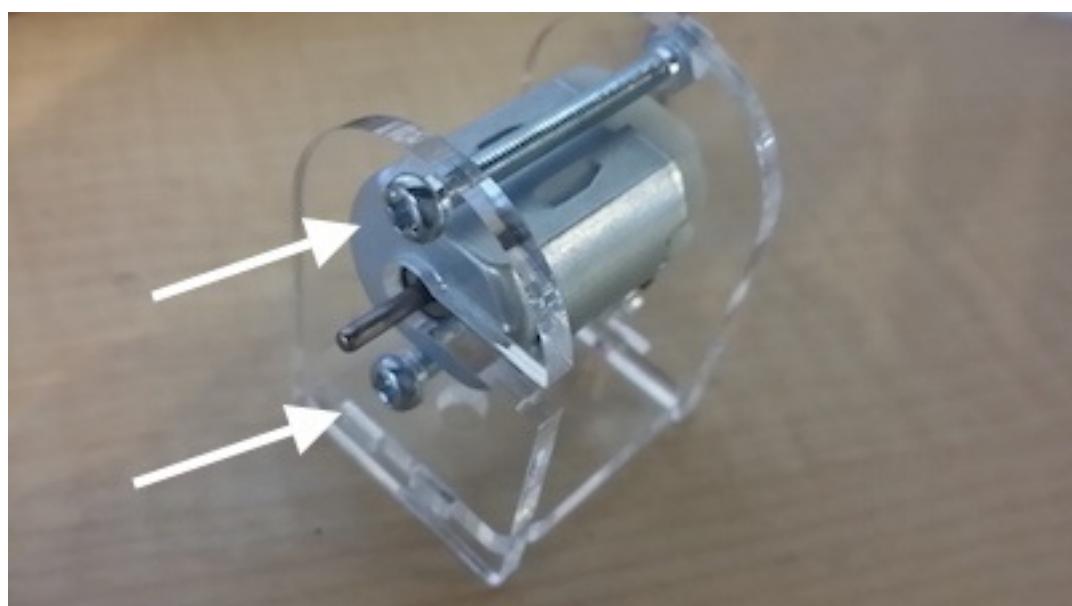
矢印のようにリード線を穴に通します。



アクリルパーツ(5)をセットし、アクリルパーツ(4)で挟みます。

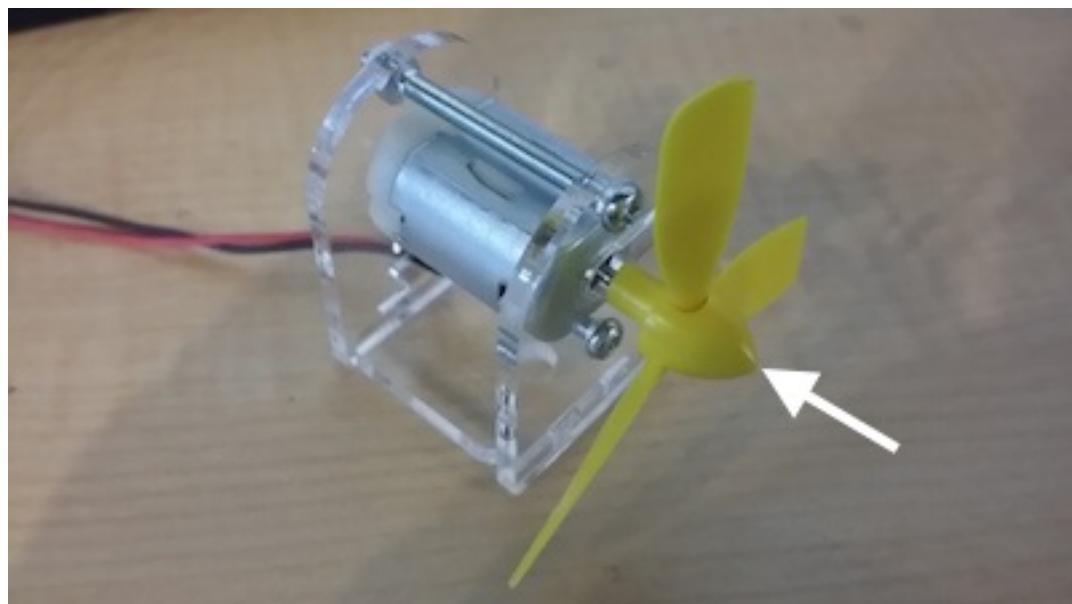
アクリルパーツ(5)にあるサーボ穴が前になるようにしてください。

2.ネジで固定します。



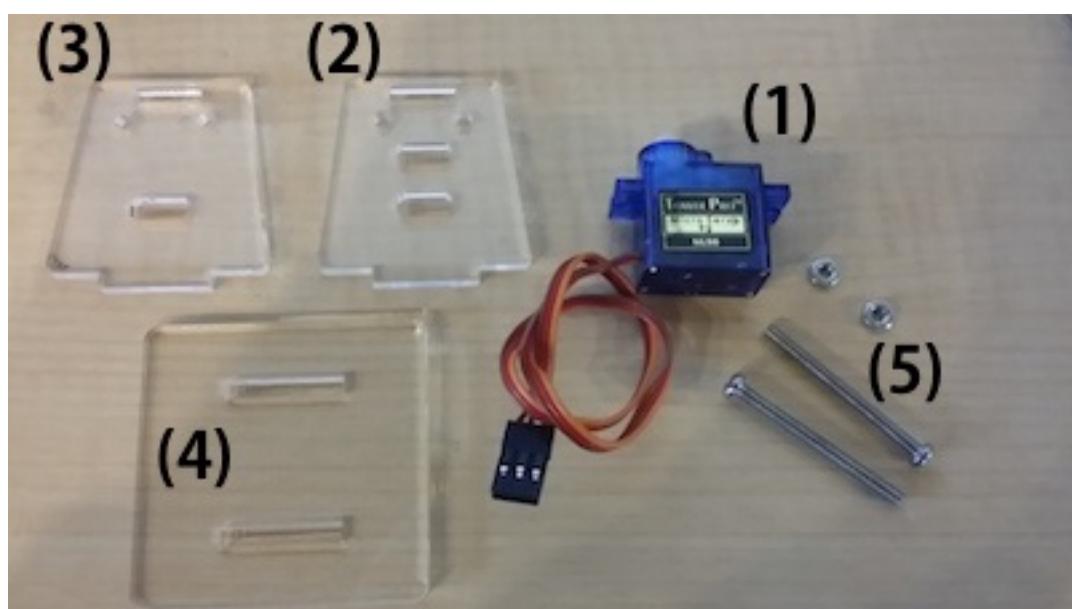
ネジ（6）で固定します。

3.プロペラを取り付けます。



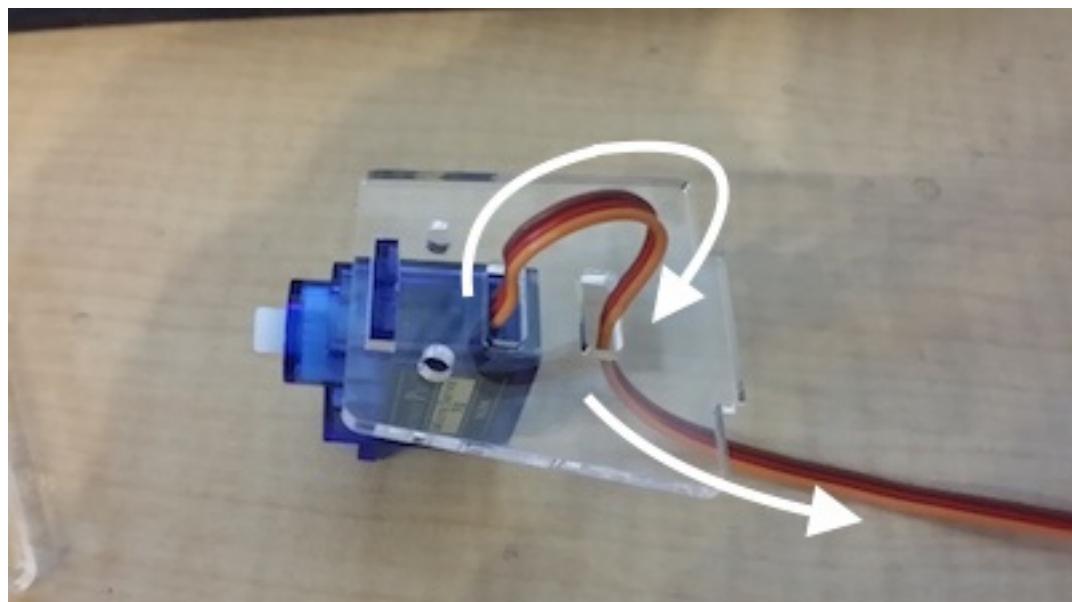
土台部分

パーツ構成 Kadenキット土台部分

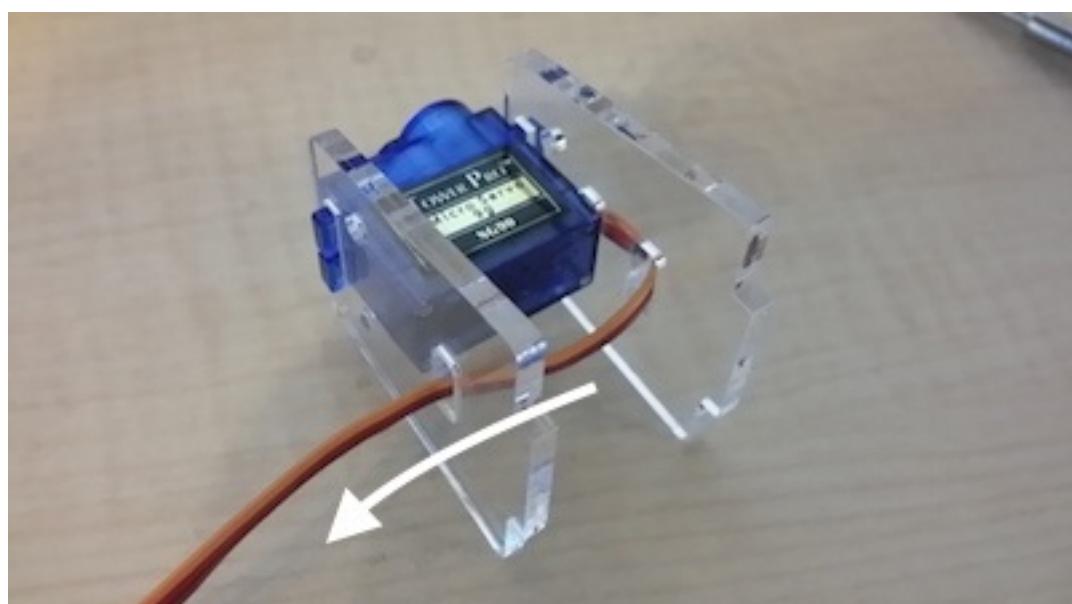


(1) サーボ、(2)~(4)アクリルパーツ、(5) M3-35mmネジ2本、M3ナット2個
(鉄)

1.アクリルパーツ(2)と(3)でサーボを挟みます。



アクリルパーツ(2)をサーボギア寄り面にセットします。サーボのリード線を写真のように穴に通します。

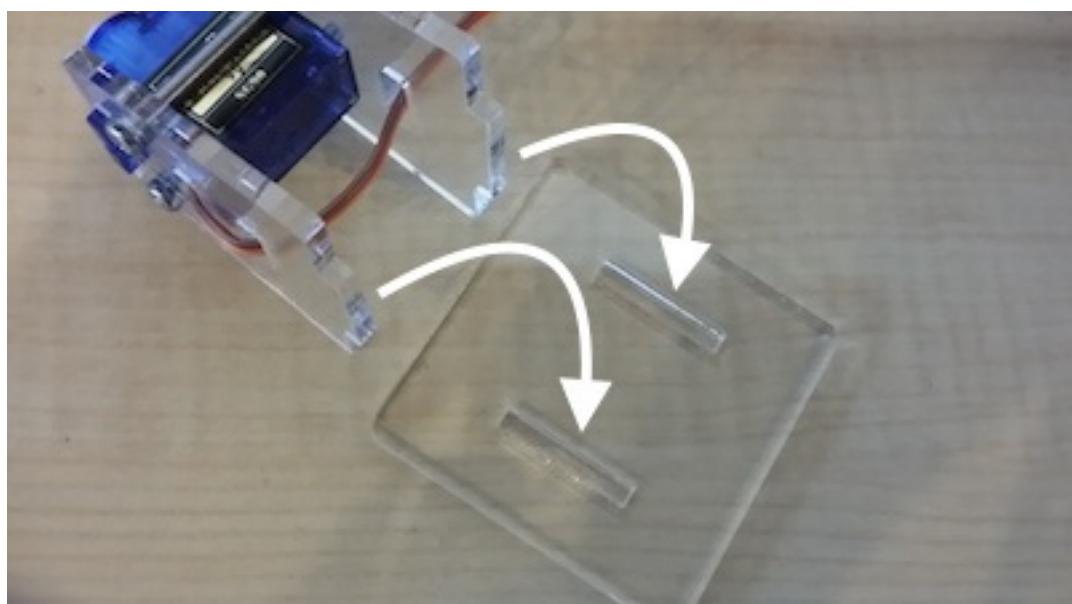


アクリルパーツ(3)をサーボギアの反対面にセットします。サーボのリード線を写真のように穴に通します。

2.ネジで固定します。

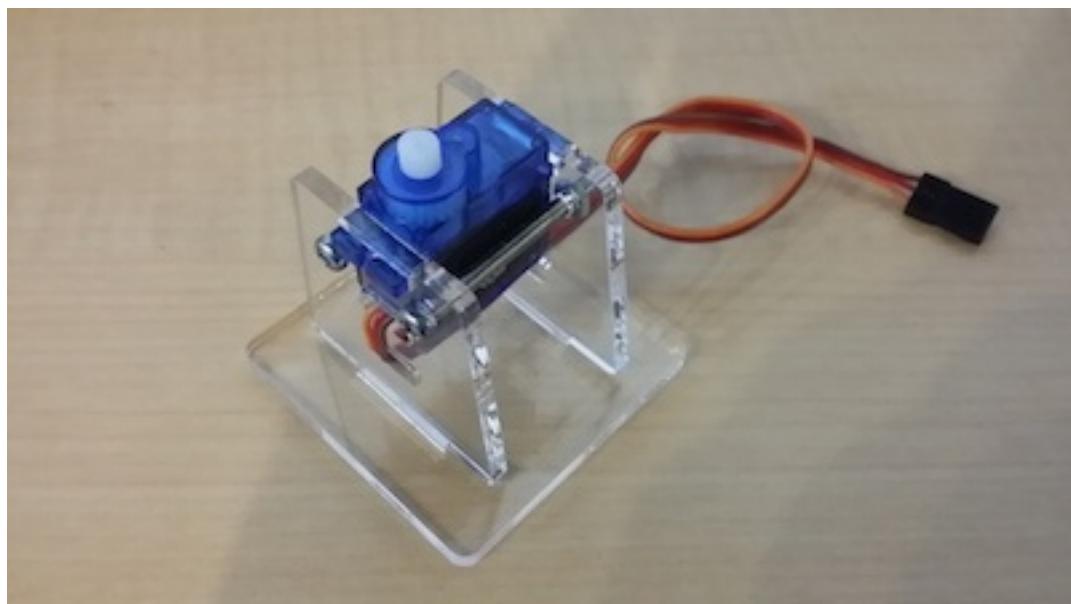


3.底版をセットします。



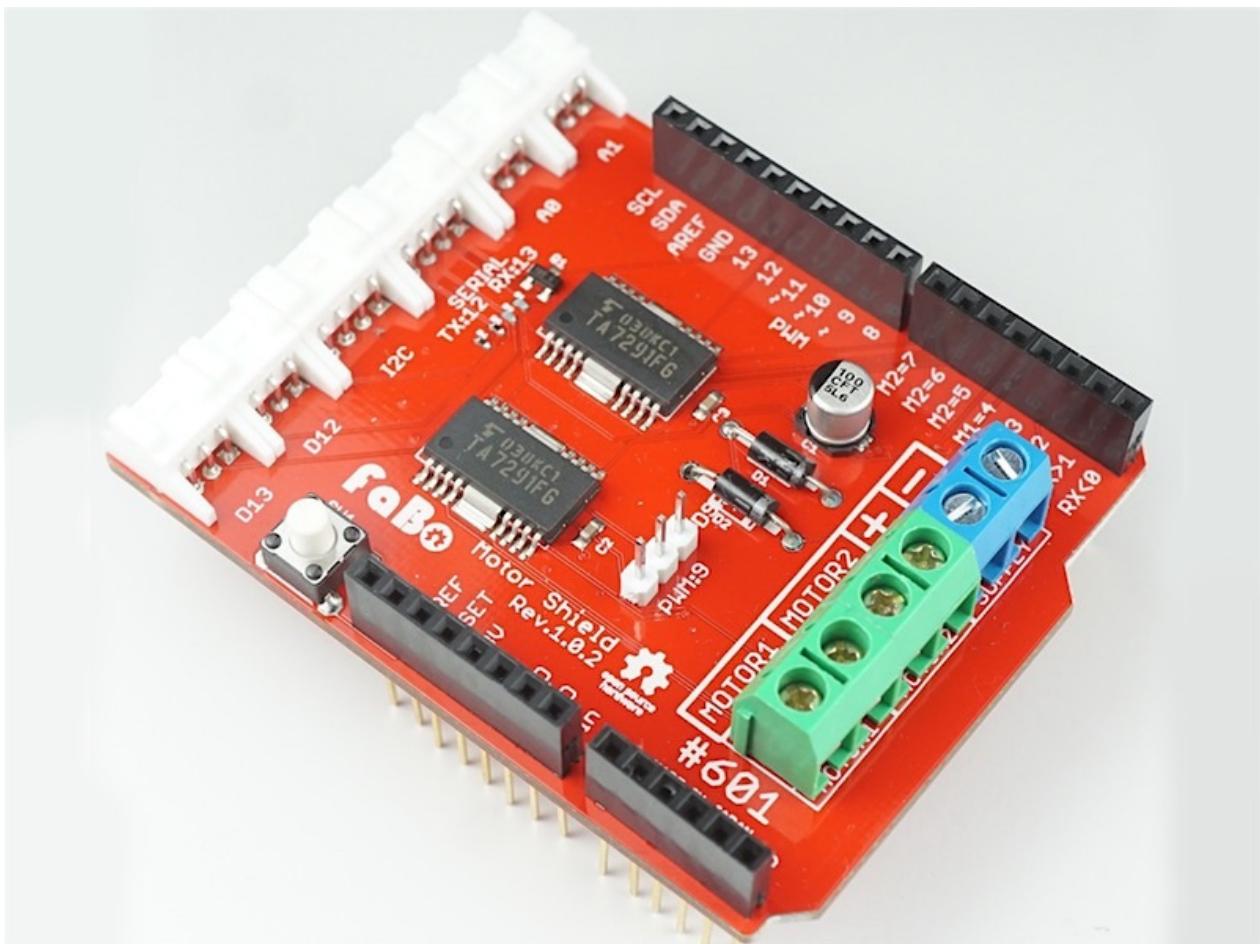
はめるだけですが、お好みで接着固定してください。

土台完成



前者で作成したプロペラ本体を土台のサーボギアに取り付けて完成です。

#601 Motor Shield for Arduino



Overview

2個のDCモーターを制御することができるシールドです。

モーターを動かすには外部からの電源供給が必要になります。

コネクタ

DCモータ用コネクタ

- Moter1用コネクタ
 - D2 (信号1)
 - D4 (信号2)

- Motor2用コネクタ
 - D5 (信号1)
 - D7 (信号2)
- 外部電源(DCモータ用)

アナログコネクタ

- A0
- A1

デジタルコネクタ

- D12
- D13

PWM/Servoコネクタ

- サーボモータ接続用コネクタ
 - PWMに対応するD9

シリアルコネクタ

SoftwareSerialとして使用するため、RX,TXはそれぞれ、D12,D13になります

I2Cコネクタ

Arduino MEGAではR3以降から対応になります。 Arduino UNO R3/R2では使用可能で
す。

PIN配置について

モーターシールドのピンは以下のようになっています。

Pin	モーターNo	説明
D2	1	信号1
D3	1	出力値設定
D4	1	信号2

Pin	モーターNo	説明
D5	2	信号1
D6	2	出力値設定
D7	2	信号2

動作方法について

モーター（モータードライバ）に対して2つの信号を送り、その組み合わせによってモーターを制御することができます。

信号1	信号2	動作
HIGH	LOW	前進
LOW	HIGH	後退
LOW	LOW	静止

信号1、信号2の両方をHIGHにすると、部品が壊れる可能性があるので設定しないようにして下さい。

for Arduino

```
void setup()
{
    // DCモーター1
    pinMode(2, OUTPUT); // モーター1のピン1(前進用)
    pinMode(3, OUTPUT); // モーター1の出力設定用
    pinMode(4, OUTPUT); // モーター1のピン2(後退用)

    // DCモーター2
    pinMode(5, OUTPUT); // モーター2のピン1(前進用)
    pinMode(6, OUTPUT); // モーター2の出力設定用
    pinMode(7, OUTPUT); // モーター2のピン2(後退用)

}

void loop()
{
    // モーター1の設定(HIGH/LOW:前進)
    digitalWrite(2, HIGH);
    digitalWrite(4, LOW);
    analogWrite(3, 255); // 0-255 強さ

    // モーター2の設定(LOW/HIGH:後退)
    digitalWrite(5, LOW);
    digitalWrite(7, HIGH);
    analogWrite(6, 255); // 0-255 強さ

    delay(10);
}
```

課題

- [Level1] Angle Brickをつかって回転の速度を調節してみよう
- [Level2] 温度センサーで、温度が高くなったら回転させよう
- [Level3] 距離センサーで、手が近づいたら回転させよう