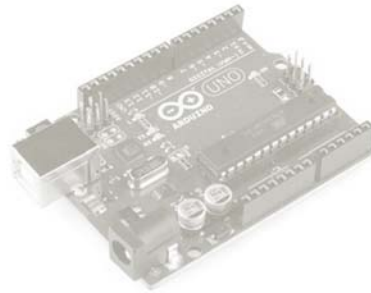




# 第 7 回

## マイコン（Arduino）入門



IoTデバイスの試作やインタラクティブアートでは  
アルドゥイーノ  
Arduinoという小さなコンピュータがよく使われます。  
GUIだけではない新しいインタラク션을創造するために  
マイコンの扱い方とプログラミングに入門しましょう！

## ✈ 7.0 マイコン (Arduino) 入門

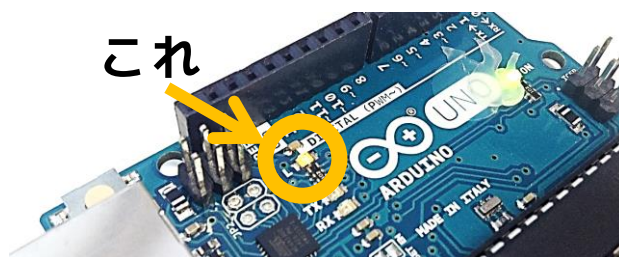
IoT デバイスの試作やインタラクティブアートでは Arduino (アルドゥイーノ) という小さなコンピュータがよく使われます。GUI だけではない新しいインタラクションを創造するためにマイコンの扱い方とプログラミングに入門しましょう！

今回は HTML や JavaScript はいったん離れ、Arduino 単体でマイコンプログラミングの基礎を学びます。次回以降、Arduino を Web から (JavaScript で) 制御します。

### 7.1 LED を点滅させる (通称: L チカ)

→ 3 ページ

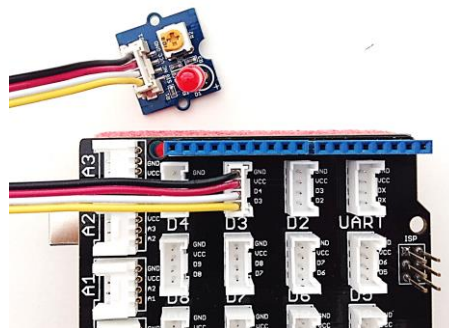
マイコン入門の定番、LED を点滅させるプログラミング (通称: L チカ) を通して Arduino の扱い方の基礎を学びます。



### 7.2 Grove で L チカ

→ 9 ページ

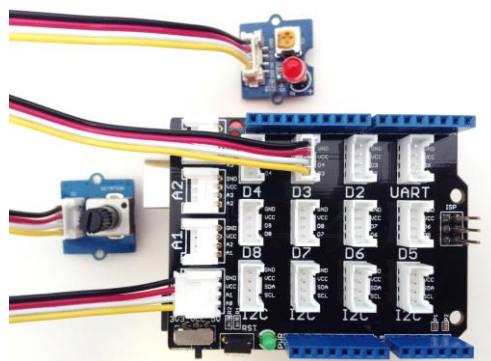
Arduino に装着した Grove (後述) に LED をつないで、より本格的な L チカをします。



### 7.3 センサと LED の組み合わせ

→ 13 ページ

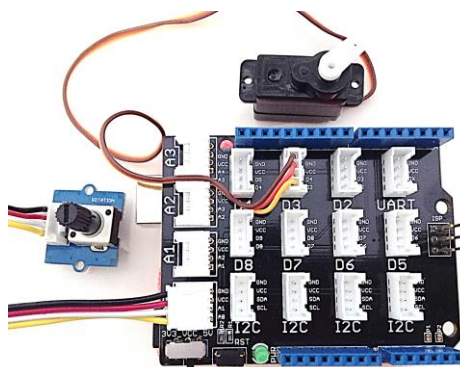
タッチセンサ、角度センサ、光センサ、音センサなどのセンサによる計測と、計測値による LED の制御を学びます。



## 7.4 サーボを動かす

→ 21 ページ

ロボットやラジコンなどに欠かせないアクチュエータであるサーボモーターの制御と、センサの計測値によるサーボの制御を学びます。



## 本日の課題

→ 25 ページ

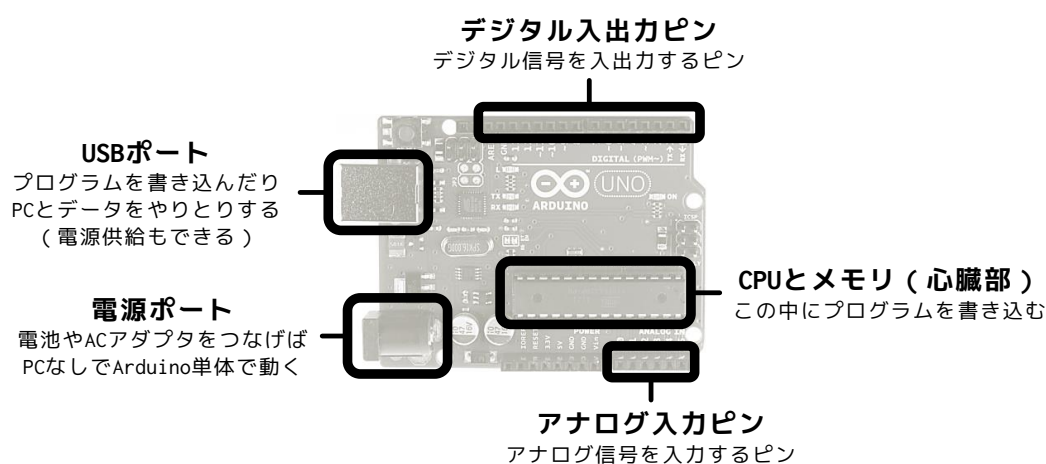
## ✈ 7.1 LED を点滅させる (通称: Lチカ)

マイコン入門の定番、LED を点滅させるプログラミングを通して Arduino の扱い方の基礎を学びます。



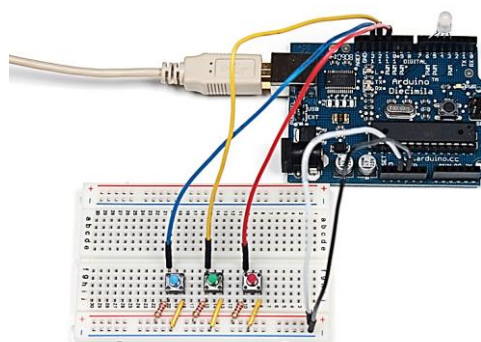
### 始める前に解説 : Arduino って?

Arduino (アルドゥイーノ)<sup>※1</sup> は手のひらサイズの小さなコンピュータ (マイコン) の一種です。



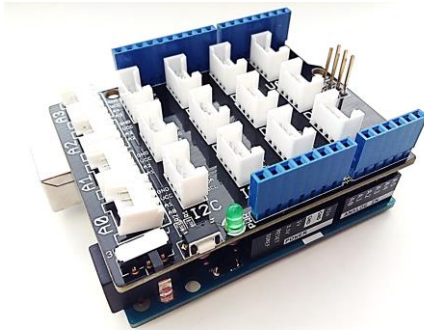
デジタル信号を入出力するピンと、アナログ信号を入力するピンがあり、これらに様々なセンサ<sup>※2</sup>をつないでデータを読み取ったり、様々なアクチュエータ<sup>※3</sup>をつないで動かすことができます。CPU にプログラム<sup>※4</sup>を書き込むことで、動作を自由にデザインできます。

世の中に多くの種類があるマイコンの中でも Arduino は扱いが簡単で、安価で、関連部品や情報も多いため、IoT デバイスの試作やインタラクティブアート、子ども向け教材としてなど、様々な場面で使われています。

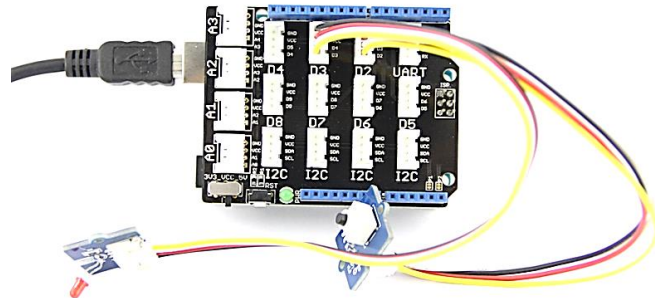


Arduino は通常、上図のように、電子回路を組んだりハンダ付けしたりして使います。オームの法則などの知識が少し必要です。

しかしこの授業では、面倒な作業をできる限り減らすために、**Grove (グローブ)**※<sup>5</sup> という部品を Arduino に装着し、センサやアクチュエータをコネクタでつなぐだけの方式で進めていきます(下図)。電子回路の知識やハンダ付けはいっさい不要です。



Arduino に Grove を装着した状態



Grove にセンサやアクチュエータをつないだ状態

- ※1 **Arduino:** イタリアで開発された世界中で人気のマイコン。Arduino の入手や設定の方法については付録 9 で紹介します。Arduino についてもっと本格的に学びたい場合、最初の 1 冊として以下の書籍をオススメします。

Massimo Banzi, Michael Shiloh 著, 船田巧 訳 (2015)。

**Arduino をはじめよう 第3版**, オライリージャパン。(2,160 円)

- ※2 **センサ:** 光、音、角度、温度、加速度、ジャイロ、GPS、タッチ、スイッチなど、環境やユーザの様々なデータを測ったり、ユーザが操作する部品。

- ※3 **アクチュエータ:** LED、スピーカ、モーター、サーボ、液晶ディスプレイなど、光ったり音を鳴らしたり動いたりして、環境やユーザに働きかける部品。

- ※4 **プログラム:** Arduino IDE という開発環境を使い、C++に準じた言語でプログラミングします。なお、Arduino のプログラムのことはスケッチと呼ぶのが一般的です。Arduino IDE のダウンロードやインストール方法については付録 9 で紹介します。

- ※5 **Grove:** アメリカの Seeed Studio 社が開発した、Arduino をとても簡単にするキット。Arduino 本体にベースとなるボードを装着し、回路組み立て済みのセンサやアクチュエータをコネクタでつなぐだけで、すぐに Arduino を楽しめます。Grove の入手方法や応用については付録 10 で紹介します。






## スケッチ (Arduino のプログラム) を入力

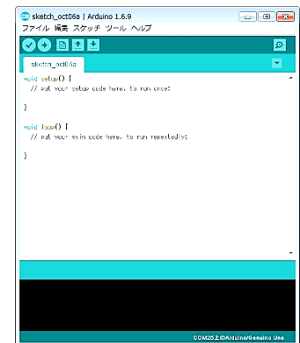
- Arduino IDE (開発環境) を起動します。



スタート → すべてのプログラム →  Arduino

- 以下のコードを入力しましょう。

- 入力し終わったら、**ファイル** → **名前をつけて保存** を選び、「\_7.1」という名前で保存しましょう。場所はどこでも構いません。\_7.1 フォルダの中に **\_7.1.ino** というファイル (Arduino のスケッチ) ができます。



※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません

\_7.1.ino

```
// 7.1 LED を点滅させる (通称: L チカ)
const int pinLED = 13;           // LED のピン番号を指定 ①

// 初期設定をする setup 関数
void setup() {                   ②
    pinMode(pinLED, OUTPUT);     // pinLED で指定したピンを出力モードに ③
}


// メインの処理を行う loop 関数、この中の命令が繰り返し実行される
void loop() {                   ④
    digitalWrite(pinLED, HIGH);  // pinLED に HIGH の信号を出力 (LED が光る) ⑤
    delay(100);                 // 100ms 待つ ⑥
    digitalWrite(pinLED, LOW);  // pinLED に LOW の信号を出力 (LED が消える)
    delay(100);                 // 100ms 待つ
}
```

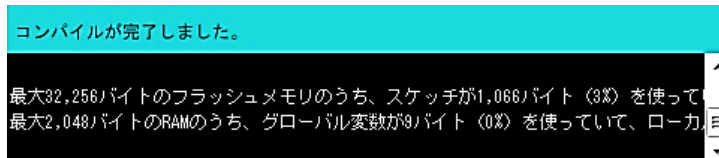


## Arduino で動作確認

- Arduino と PC を USB ケーブルで接続
  - ドライバのインストールが始まった場合はしばらく待つ。
- Arduino IDE の設定確認
  - **ツール** → **ボード** で「Arduino/Genuino Uno」が選択されていることを確認
  - **ツール** → **シリアルポート** で「COM \*\* (Arduino/Genuino Uno)」のように、右側にボードの名前が表示された COM が選択されていることを確認 (何も表示されていないときは Arduino を接続しなおしたり、別の USB ポートにつないでみる)

## ● プログラムを検証


- Arduino IDE の  (検証) をクリック
- 「コンパイルが完了しました。」と出たら、プログラムに問題はありません。

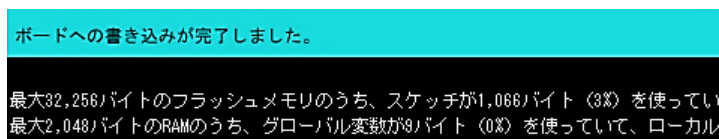


- オレンジ色の背景に何かメッセージが出たら、プログラムに誤りや問題があります。よく見直しましょう。

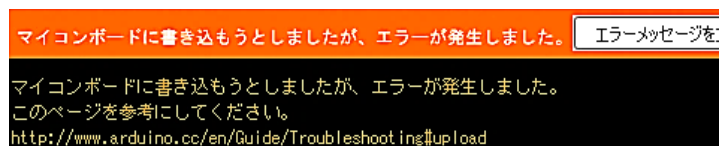


## ● プログラムを Arduino に書き込む

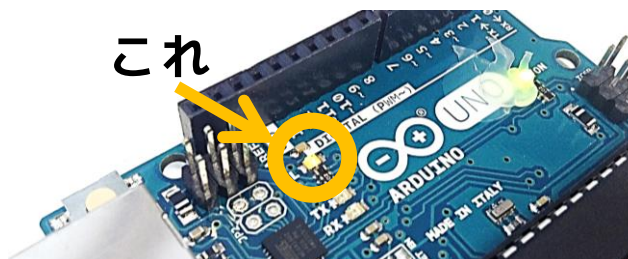
- Arduino IDE の  (マイコンボードに書き込む) をクリック
- 「ボードへの書き込みが完了しました。」と出たら、OK です。



- オレンジ色の背景に何かメッセージが出たら、PC との接続や IDE の設定に問題があります。



## ● Arduino 基板上の LED (オレンジ色) がチカチカ点滅していれば OK



- Grove の下に隠れて見にくいので、のぞき込んで確認
- 緑色に点灯している LED は電源ランプ





## 解説

① `const int pinLED = 13;`

`const` は JavaScript の場合と同様、変数（再代入できない変数）の定義です。`int` はその変数を整数として定義します<sup>※1</sup>。つまり、整数の変数 `pinLED` を定義して 13 という値を入れています。

Arduino にはデジタル入出力ピンが 14 個（0～13）あり、通常は実際にそのピンに LED を配線するのですが… 基板上のオレンジ色の LED はテスト用に最初から 13 番ピンに内部で配線されています。

※1 参考：変数の型（かた）（代表的な型のみ）

宣言	型
<code>int</code>	整数（-32,768～32,767）
<code>unsigned int</code>	符号なし整数（0～65,535）
<code>long</code>	長整数（-2,147,483,648～2,147,483,647）
<code>unsigned long</code>	符号なし長整数（0～4,294,967,295）
<code>float</code>	浮動小数点（32bit）
<code>double</code>	倍精度浮動小数点（64bit）
<code>char</code>	文字（1 文字）
<code>boolean</code>	ブール値（true / false）
<code>byte</code>	バイト型（0～255 の 8bit の数）
<code>word</code>	Word 型（0～65535 の 16bit の数）

② `void setup() { ... }`

Arduino のプログラムにはこの `void setup() { ... }` および④の `void loop() { ... }` という二つの関数を必ず含めます。

`setup()`関数は Arduino の起動（リセット）後に 1 回だけ実行されます。`setup()`関数内で様々な初期設定を行います。

ちなみに頭の `void` は「戻り値の無い関数」という意味です。この `setup()`と④の `loop()`は頭に必ず `void` を付ける、とっておいてください。

③ `pinMode(pinLED, OUTPUT);`

Arduino のデジタル入出力ピンを、入力として使うか / 出力として使うかのモードを設定する `pinMode`(ピン番号, 入出力モード) です。ここでは、`pinLED` (13 番ピン) を出力モード (`OUTPUT`) にしています。

今回は LED を光らせる（LED に電流を流す）ので `OUTPUT` です。センサの入力を受け付ける場合はモードに `INPUT` を指定します。

④ `void loop() { ... }`

Arduino のプログラムにはこの `void loop() { ... }` および②の `void setup() { ... }` という二つの関数を必ず含めます。

`loop()`関数は、`setup()`関数の処理終了後に繰り返し実行されます。Arduino をリセットするか電源を抜くまで永久に繰り返されます。この `loop()`関数内にメインとなる処理を書きます。



⑤ `digitalWrite(pinLED, HIGH);`

指定したピンにデジタル信号を出力する `digitalWrite(ピン番号, 値)` です。指定できる値は `HIGH` または `LOW` です。HIGH は ON の意味で電流を流します。LOW は OFF の意味で電流を止めます。この場合、pinLED (13 番ピン) に電流を流して、LED を点灯させています。

⑥ `delay(100);`

指定した時間、処理を止める `delay(時間[ms])` です。ここでは 100ms の間、処理を止めています。この間、⑤で 13 番ピンに流した電流は流れ続け、LED は点灯し続けます。



## コラム⑦ その他の人気のマイコン

Arduino 以外にも人気のマイコンをいくつか紹介します。もし関心があれば遊んでみましょう。

## ● Raspberry Pi (ラズベリー パイ)

かなり高性能なマイコンです。Arduino にはデジタル入出力とアナログ入力しかありませんが、Raspberry Pi は USB、LAN、HDMI、音声出力なども備えています。Linux 系 OS で動き、ディスプレイ・キーボード・マウスをつなげば PC としても使えます。プログラミングは Python (パイソン), Ruby (ルビー), Scratch (スクラッチ) などで行います。Arduino でもの足りなくなった時のステップアップにおすすめです。



## ● ESP-WROOM-02

ESP-WROOM-02 は厳密にはマイコンではなく Wi-Fi モジュールです。しかし、デジタル入出力ピンを備えており、Arduino IDE で書いたスケッチで制御できます。Arduino の弱点は単体ではインターネットに接続できないことです。ESP-WROOM-02 は、Arduino と同じように扱え、しかも直接インターネットに接続できて、完全な IoT を実現できます。この授業で使っている Grove には ESP-WROOM-02 用もあり、電子工作なしで遊べるのも良いところです。



## ● Intel Edison

Intel 社が開発した IoT 向けマイコンです。切手サイズの超小型で、そのまま製品に組み込める高信頼のマイコンです。Wi-Fi、Bluetooth、USB なども備えています。取り扱いは少し難しいですが、より本格的に IoT デバイスを開発したいなら、ぜひ勉強してみましょう。



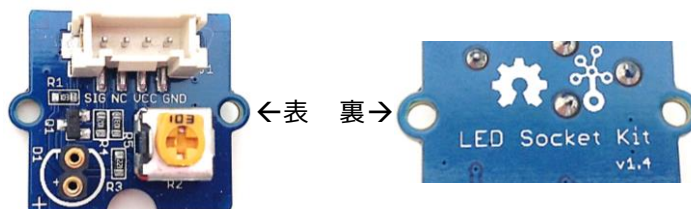
## ✈ 7.2a Grove で L チカ

Arduino の基板上の LED ではなく、Grove を通じて LED を光らせます。



### LED を接続

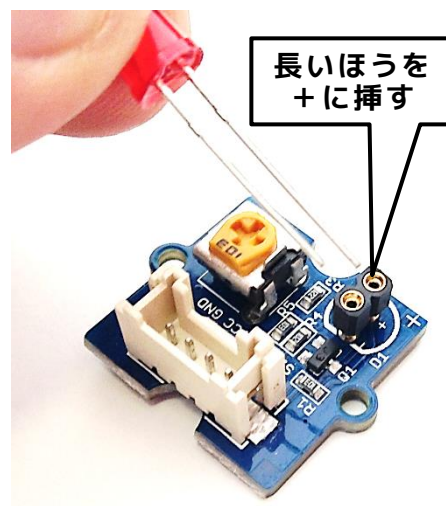
- Grove の LED ソケット を用意します。



- 好きな色の LED を選びます。

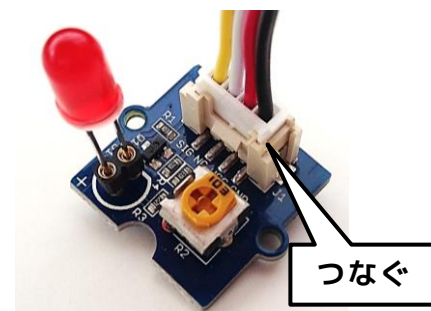


- LED の 長いほうのピン をソケットの + の穴に挿します。



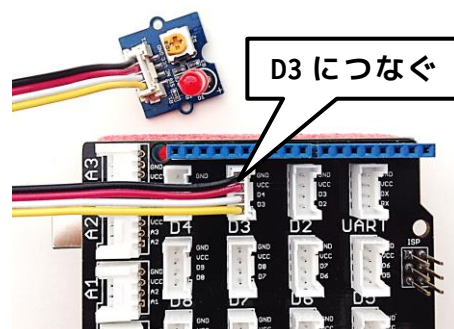
- LED ソケットにワイヤーのコネクタをつなぎます。

※コネクタには出っ張りがあるので、形をあわせてつなぎます。



- もう一方のコネクタを Grove の D3 につなぎます。

※D3 は Arduino のデジタル入出力の 3 番ピンにつながっています。





## スケッチを入力

- Arduino IDE で以下のコードを入力しましょう。
- 入力し終わったら「\_7.2a」という名前で保存しましょう。場所はどこでも構いません。

※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません

**\_7.2a.ino**



```
// 7.2a Grove で L チカ
const int pinLED = 3;           // D3 ピンに LED をつないだので pinLED に 3 を設定

// 以下は 7.1 と全く同じ (7.1 からコピーしましょう)
void setup() {
  pinMode(pinLED, OUTPUT);      // pinLED で指定したピンを出力モードに
}

void loop() {
  digitalWrite(pinLED, HIGH);   // pinLED に HIGH の信号を出力 (LED が光る)
  delay(100);                   // 100ms 待つ
  digitalWrite(pinLED, LOW);    // pinLED に LOW の信号を出力 (LED が消える)
  delay(100);                   // 100ms 待つ
}
```



## Arduino で動作確認

- Arduino と PC を USB ケーブルで接続
- Arduino IDE の設定確認
  - ツール → ボード で「Arduino/Genuino Uno」が選択されていることを確認
  - ツール → シリアルポート で「COM \*\* (Arduino/Genuino Uno)」のように、右側にボードの名前が表示された COM が選択されていることを確認 (何も表示されていないときは Arduino を接続しなおしたり、別の USB ポートにつないでみる)
- プログラムを検証
  - Arduino IDE の  (検証) をクリック
- プログラムを Arduino に書き込む
  - Arduino IDE の  (マイコンボードに書き込む) をクリック
- LED がチカチカ点滅すれば OK

## ✈ 7.2b アナログ L チカ (通称: ほたる)

LED を ON (点灯) / OFF (消灯) するだけでなく、明るさを変化させます。



### スケッチを入力

- Arduino IDE で以下のコードを入力し、「\_7.2b」という名前で保存しましょう。

※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません

\_7.2b.ino



```
// 7.2b アナログ L チカ (通称: ほたる)
const int pinLED = 3;           // D3 ピンに LED をつないだので pinLED に 3 を設定

void setup() {
  pinMode(pinLED, OUTPUT);      // pinLED で指定したピンを出力モードに
}

void loop() {
  for(int i = 0; i < 256; i++) {  // 0~255 まで繰り返す ①
    analogWrite(pinLED, i);      // LED を i の強さで光らせる ②
    delay(5);                   // 少し待つ
  }
  for(int i = 255; i >= 0; i--) { // 255~0 まで繰り返す ③
    analogWrite(pinLED, i);      // LED を i の強さで光らせる
    delay(5);                   // 少し待つ
  }
}
```



### Arduino で動作確認

- Arduino と PC を USB ケーブルで接続
- プログラムを検証
  - Arduino IDE の  (検証) をクリック
- プログラムを Arduino に書き込む
  - Arduino IDE の  (マイコンボードに書き込む) をクリック
- LED が徐々に明るくなり、徐々に暗くなるのを繰り返せば OK
- 動作確認が終わったら USB ケーブルを抜く



## 解説

① `for(int i = 0; i < 256; i++) { ... }`

{ }内の処理を繰り返す for()です。( )内の意味は…

- `int i = 0` … 整数 `i` を定義し、初期値を 0 にする
- `i < 256` … `i` が 256 未満である間
- `i++` … `i` を 1 ずつ加算しながら { }内の処理を繰り返す

つまり、`i` を 0~255 まで 1 ずつ変えながら { }内の処理を繰り返します。

② `analogWrite(pinLED, i);`

ここまで使ってきた `digitalWrite()` は HIGH ( ON ) か LOW ( OFF ) の二つの値しか出力できません。`analogWrite()` は、0~255 の間の値を出力することができます。①の `for()` により `i` の値が 0~255 に変化するので、LED は消灯(0)から点灯(255)に向かって徐々に明るくなります。

③ `for(int i = 255; i >= 0; i--) { ... }`

①と同様、{ }内の処理を繰り返す for()です。この場合の( )内の意味は…

- `int i = 255` … 整数 `i` を定義し、初期値を 255 にする
- `i >= 0` … `i` が 0 以上である間
- `i--` … `i` を 1 ずつ減算しながら { }内の処理を繰り返す

つまり、`i` を 255~0 まで 1 ずつ変えながら { }内の処理を繰り返します。これにより LED は点灯(255)から消灯(0)に向かって徐々に暗くなります。

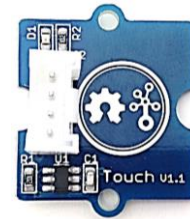
## ✧ 7.3a タッチセンサで L チカ

タッチセンサ、角度センサ、光センサ、音センサなどのセンサによる計測と、計測値による LED の制御を学びます。まずはタッチセンサで L チカします。

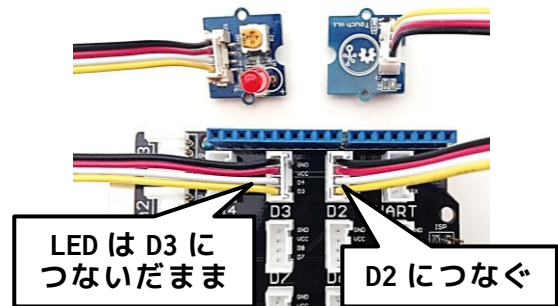


**タッチセンサを接続 (LED は 7.2 までと同じようにつないだまま)**

- Grove の タッチセンサ を用意します。



- ワイヤーで Grove の D2 につなぎます。



**スケッチを入力**

- Arduino IDE で以下のコードを入力し、「\_7.3a」という名前で保存しましょう。

※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません

\_7.3a.ino



```
// 7.3a タッチセンサで L チカ
const int pinTouch = 2;          // Touch センサを Grove の D2 に
const int pinLED = 3;           // LED を Grove の D3 に

void setup() {
  pinMode(pinTouch, INPUT);      // pinTouch で指定したピンを入力モードに ①
  pinMode(pinLED, OUTPUT);      // pinLED で指定したピンを出力モードに
}

void loop() {
  boolean touch = digitalRead(pinTouch); // pinTouch の状態を取得 ②
  if(touch == true) {             ③
    digitalWrite(pinLED, HIGH);
  } else {
    digitalWrite(pinLED, LOW);
  }
}
```



## Arduino で動作確認

- Arduino と PC を USB ケーブルで接続
- プログラムを検証  → 書き込む 
- タッチセンサに手を触れると LED が点灯、手を離すと消灯すれば OK
- 動作確認が終わったら USB ケーブルを抜く



## 解説

### ① `pinMode(pinTouch, INPUT);`

`pinMode` の第一引数にタッチセンサのピン番号(2)、第二引数に `INPUT` を指定して、タッチセンサの値を入力する設定にしています。

### ② `boolean touch = digitalRead(pinTouch);`

`boolean` でブール型 (`true/false`) の変数 `touch` を定義しています (7.1 の解説を参照)。

`digitalRead(ピン番号)` は、指定したピン番号に接続したセンサの値を読み取ります。

これにより、タッチセンサの値を変数 `touch` に代入しています。タッチセンサは、触れている時に `HIGH (true)` の信号を、触れていない時は `LOW (false)` の信号を返します。

### ③ `if(touch == true) { 処理 A } else { 処理 B }`

条件分岐の `if()` です。( )内の式が成り立つ時には処理 A を実行し、成り立たない時には処理 B を実行します。

`touch == true`、つまりタッチセンサに触れている時には `digitalWrite(pinLED, HIGH)` で LED を点灯させ、触れていない時には `digitalWrite(pinLED, LOW)` で LED を消灯させています。



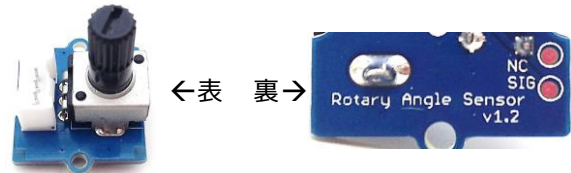
## ✧ 7.3b 明るさ調整ダイヤル

角度センサを使って LED の明るさを調整します。また、センサの計測値をモニターする方法を学びます。

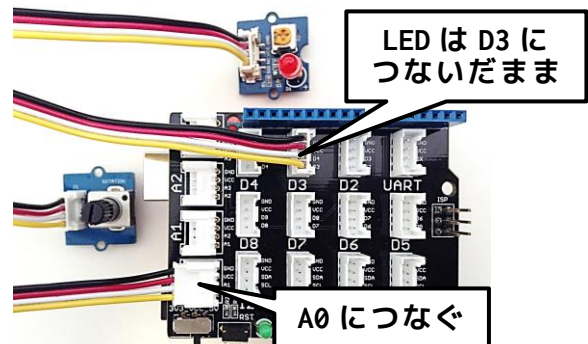


### 角度センサを接続 (LED はつないだまま、タッチセンサは外す)

- Grove の 角度センサ を用意します。



- ワイヤーで Grove の A0 につなぎます。



### スケッチを入力

- Arduino IDE で以下のコードを入力し、「\_7.3b」という名前で保存しましょう。

※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません



\_7.3b.ino

```
// 7.3b 明るさ調整ダイヤル
const int pinAngle = A0;          // 角度センサを Grove の A0 に ①
const int pinLED = 3;             // LED を Grove の D3 に

void setup() {
  pinMode(pinAngle, INPUT);        // pinAngle で指定したピンを入力モードに
  pinMode(pinLED, OUTPUT);         // pinLED で指定したピンを出力モードに
  Serial.begin(9600);              // センサの値をモニターするシリアルモニタの準備 ②
}

void loop() {
  int angle = analogRead(pinAngle); // 角度センサの値を取得 ③
  Serial.println(angle);             // シリアルモニタに値を出力 ④
  int brightness = map(angle, 0, 1023, 0, 255); // 角度の値(0~1023)を 0~255 に調整 ⑤
  analogWrite(pinLED, brightness);  // LED を brightness の強さで光らせる
  delay(100);                       // 少し待ってから次の値を取得
}
```

**Arduino で動作確認**

- Arduino と PC を USB ケーブルで接続
- プログラムを検証  → 書き込む 
- ツール → シリアルモニタ で出るウィンドウに角度センサの値が出るのを確認
- 角度センサのつまみを回すと LED の明るさが変われば OK
- 動作確認が終わったら USB ケーブルを抜く

**解 説**① **const int pinAngle = A0;**

デジタル入出力ピンの場合はピン番号の指定は数値のみですが、アナログ入力ピンの場合は A0 のように数値の前に「A」を付けます。

② **Serial.begin(9600);**

シリアルモニタという機能を使って、計測値を PC でモニターすることができます。この 1 行でそのモニター機能を開始させます。( ) 内の数値はデータ通信速度です。Arduino は 9600 bps (bits/second) でデータを送ってくる仕様なので、9600 を指定します。

③ **int angle = analogRead(pinAngle);**

analogRead(ピン番号)はアナログ入力ピンの値を読み取ります。角度センサの値を読み取って、変数 angle に代入しています。

④ **Serial.println(angle);**

読み取った角度センサの値 angle をシリアルモニタに出力 (プリント) しています。

⑤ **int brightness = map(angle, 0, 1023, 0, 255);**

map(値 0, 値 1, 値 2, 値 3, 値 4)は、値 1~値 2 の範囲の値 0 を、値 3~値 4 の範囲の値に調整します。

今回用いた角度センサは 0~1023 の範囲の値が計測できます (シリアルモニタで確認できます)。いっぽうで、LED の明るさは 0~255 の範囲で指定する必要がありました。そこで、角度センサの値 angle を 0~255 の範囲になるように調整しています。その調整された値を変数 brightness に代入しています。

センサの種類によって、または同じセンサでもメーカーや個体差によって計測値の範囲は異なります。map()関数はこのような計測範囲の違いを調整して処理するための関数です。

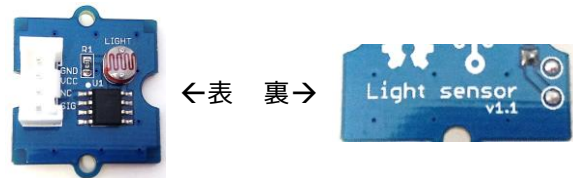
## ✈ 7.3c 自動調光

光センサを使って LED の明るさを調整します。部屋の明るさに応じて LED の明るさを調光するシステムを作ります。

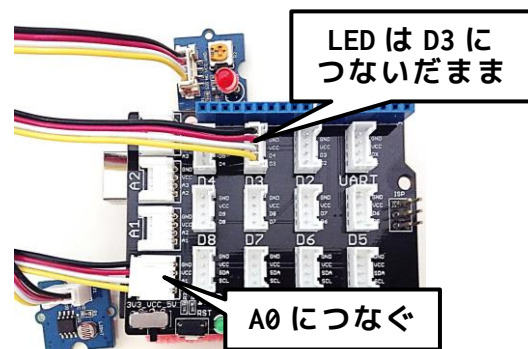


**光センサを接続 (LED はつないだまま、角度センサは外す)**

- Grove の 光センサ を用意します。



- ワイヤーで Grove の A0 につなぎます。



**スケッチを入力**

- Arduino IDE で以下のコードを入力し、「\_7.3c」という名前で保存しましょう。

※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません

\_7.3c.ino



```
// 7.3c 自動調光
const int pinLight = A0;      // 光センサを Grove の A0 に
const int pinLED = 3;         // LED を Grove の D3 に

void setup() {
  pinMode(pinLight, INPUT);    // pinLight で指定したピンを入力モードに
  pinMode(pinLED, OUTPUT);     // pinLED で指定したピンを出力モードに
  Serial.begin(9600);          // シリアルモニタの準備
}

void loop() {
  int light = analogRead(pinLight); // 光センサの値を取得
  Serial.println(light);           // シリアルモニタに値を出力
  int brightness = map(light, 0, 764, 255, 0); // 明るさの値(0~764)を 255~0 に調整 ①
  analogWrite(pinLED, brightness); // LED を brightness の強さで光らせる
  delay(100);                     // 少し待ってから次の値を取得
}
```



## Arduino で動作確認

- Arduino と PC を USB ケーブルで接続
- プログラムを検証  → 書き込む 
- ツール → シリアルモニタ で出るウィンドウに光センサの値が出るのを確認
- 光センサを覆って暗くすると LED が明るく、センサを覆わないと LED が暗くなれば OK
- 動作確認が終わったら USB ケーブルを抜く



## 解説

### ① `int brightness = map(light, 0, 764, 255, 0);`

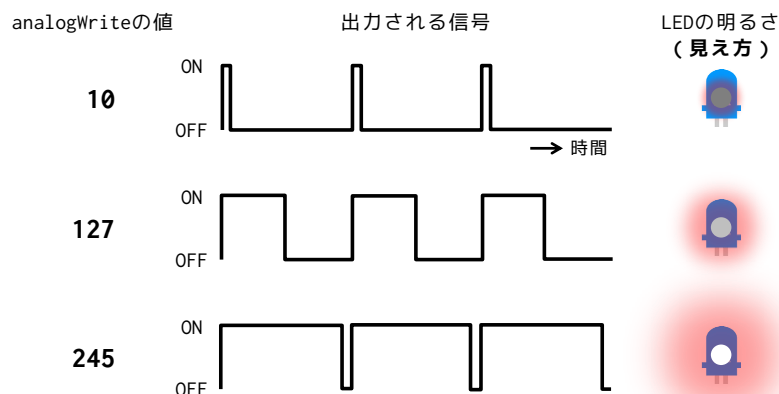
`map()`関数で計測範囲の違いを調整しています。

今回の光センサはおよそ 0 (真っ暗) ~ 764 (明るい) の範囲で変化します。部屋が暗いほど LED を明るく、部屋が明るいほど LED を暗くするので、センサの計測値 (暗い~明るい) を LED の明るさ (明るい~暗い) に反転させる必要があります。`map()`関数はこのように範囲の調整だけでなく、値を反転させることもできます。



## コラム ⑧ LED の明るさは変わらない？

7.2b や 7.3c では `analogWrite()` で LED の明るさを変化させました。しかし実際には LED の明るさは変わっていません。Arduino に アナログ出力ピン はありません。`analogWrite()` は下図のようなやり方で明るさが変わっているように見せているのです。



ON/OFF の時間を変えてアナログ出力を模擬する方法を、パルス幅変調 (Pulse Width Modulation; PWM) と呼びます。Arduino Uno で PWM が使えるピンはデジタルの 3, 5, 6, 9, 10, 11 番のみです。

## ✈ 7.3d サウンド・ライト (光を音に)

音センサを使って LED の明るさを調整します。部屋でもの音がしたら光る警報装置や、音楽によって光の強さを変えるライティングシステムなどに応用できます。

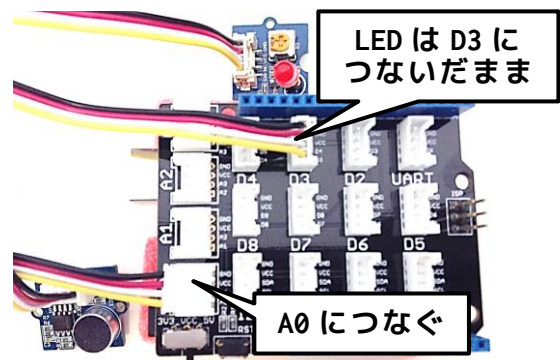


### 音センサを接続 (LED はつないだまま、光センサは外す)

- Grove の 音センサ を用意します。



- ワイヤーで Grove の A0 につなぎます。



### スケッチを入力

- Arduino IDE で以下のコードを入力し、「\_7.3c」という名前で保存しましょう。

※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません

\_7.3c.ino



```
// 7.3d サウンド・ライト (音を光に)
const int pinSound = A0;          //音センサを Grove の A0 に
const int pinLED = 3;             // LED を Grove の D3 に

void setup() {
  pinMode(pinSound, INPUT);       // pinSound で指定したピンを入力モードに
  pinMode(pinLED, OUTPUT);        // pinLED で指定したピンを出力モードに
  Serial.begin(9600);             // シリアルモニタの準備
}

void loop() {
  int sound = analogRead(pinSound); // 音センサの値を取得
  Serial.println(sound);            // シリアルモニタに値を出力
  int brightness = map(sound, 0, 1023, 0, 255); // 音の値 (0~1023) を 0~255 に調整
  analogWrite(pinLED, brightness);  // LED を brightness の強さで光らせる
  delay(15);                       // 少し待ってから次の値を取得 ①
}
```



## Arduino で動作確認

- Arduino と PC を USB ケーブルで接続
- プログラムを検証  → 書き込む 
- ツール → シリアルモニタ で出るウィンドウに音センサの値が出るのを確認
- 音センサに向かって大きな音や声を出したり、センサを指で軽くとんとん叩いた時に LED が光れば OK
- スマホなどで音楽を流し、スピーカをセンサに近づけてみて、音楽に合わせて LED が光れば OK
- 動作確認が終わったら USB ケーブルを抜く



## 解説

### ① `delay(15);`

7.3b や 7.3c では 100ms ごとにセンサの値を計測していましたが、音の大きさは、特に音楽の場合はとても速く変化するので、計測する時間間隔を短めにしました。

## ✈ 7.4a サーボを動かす

ロボットやラジコンなどに欠かせないアクチュエータであるサーボモーターの制御と、センサの計測値によるサーボの制御を学びます。まずは L チカと同じように、サーボを定期的に左右に動かします。



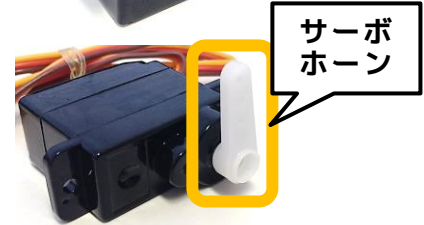
### サーボを接続 (他にはなにもつながない)

- Grove の サーボ を用意します。

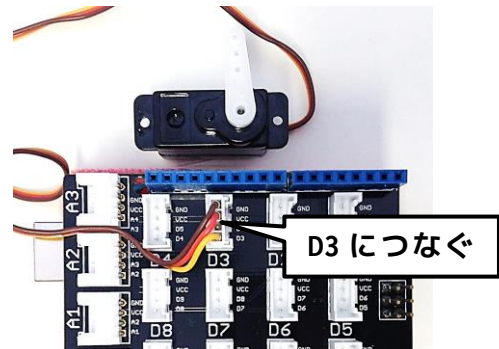


- サーボにサーボホーン<sup>※</sup>を装着します。

※サーボ (モーター) の回転を機械に伝えるための腕のような部品



- ワイヤーを Grove の D3 につなぎます。



### スケッチを入力

- Arduino IDE で以下のコードを入力し、「\_7.4a」という名前で保存しましょう。

※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません

\_7.4a.ino

```
// 7.4a サーボを動かす
const int pinServo = 3;           // サーボを Grove の D3 に
#include <Servo.h>                 // サーボを使うためのライブラリの読み込み ①
Servo servo;                      // サーボオブジェクトを宣言 ②

void setup() {
  servo.attach(pinServo);         // サーボのピン番号を指定 (サーボは pinMode を使わない) ③
}
```





```

void loop() {
  for(int i = 0; i < 180; i++) {    // 0~179 まで繰り返す
    servo.write(i);               // サーボの角度を i にする      ④
    delay(5);                     // 少し待つ
  }
  for(int i = 179; i >= 0; i--) {  // 179~0 まで繰り返す
    servo.write(i);               // サーボの角度を i にする
    delay(5);                     // 少し待つ
  }
}

```



### Arduino で動作確認

- Arduino と PC を USB ケーブルで接続
- プログラムを検証  → 書き込む 
- サーボが左右に行ったり来たり動けば OK
- 動作確認が終わったら USB ケーブルを抜く



### 解説

#### ① `#include <Servo.h>`

サーボを制御するには Servo.h という外部ライブラリを使います。この行はその宣言です。

#### ② `Servo servo;`

Arduino でサーボを扱うには、Servo オブジェクト (Servo.h に入っている部品) を使います。ここでは、servo という名前で Servo オブジェクトを宣言しています。

#### ③ `servo.attach(pinServo);`

これまでは pinMode(ピン番号, 入出力モード) で入出力の準備をしましたが、サーボの場合は Servo オブジェクトの attach(ピン番号) メソッドで準備をします。

#### ④ `servo.write(i);`

指定した角度にサーボを動かす servo.write(角度) です。

ここでは、最初の for 文で 0 度から 179 度に向かって 1 度刻みでサーボを動かし、2 回目の for 文で 179 度から 0 度に向かって 1 度刻みでサーボを動かしています。Grove 付属のサーボは 0~179 度の間で動きます。

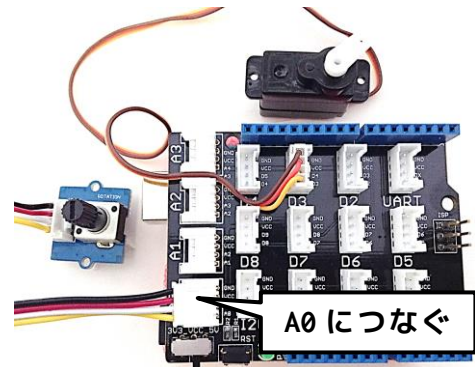
## ✈ 7.4b サーボをセンサで動かす

角度センサの値に応じてサーボを動かします。



角度センサを接続 (サーボはそのまま)

- 角度センサを Grove の A0 につなぎます。



スケッチを入力

- Arduino IDE で以下のコードを入力し、「\_7.4b」という名前で保存しましょう。

※ ①や②などの丸数字は打ち込まないこと  
コメント文は打ち込み必須ではありません

\_7.4b.ino

```
// 7.4b サーボをセンサで動かす
const int pinAngle = A0;           // 角度センサを Grove の A0 に
const int pinServo = 3;             // サーボを Grove の D3 に
#include <Servo.h>                   // サーボを使うためのライブラリの読み込み
Servo servo;                        // サーボオブジェクトを宣言



void setup() {
  pinMode(pinAngle, INPUT);         // pinAngle で指定したピンを入力モードに
  servo.attach(pinServo);           // サーボのピン番号を指定
  Serial.begin(9600);               // シリアルモニタの準備
}

void loop() {
  int angle = analogRead(pinAngle); // 角度センサの値を取得
  Serial.println(angle);             // シリアルモニタに値を出力
  // 角度センサの値 (0~1023) をサーボの角度 (0~179) に調整
  int servoAngle = map(angle, 0, 1023, 0, 179);
  servo.write(servoAngle);           // サーボの角度を設定
  delay(15);                         // 少し待ってから次の値を取得
}
```

①



## Arduino で動作確認

- Arduino と PC を USB ケーブルで接続
- プログラムを検証  → 書き込む 
- ツール → シリアルモニタ で出るウィンドウに角度センサの値が出るのを確認
- 角度センサを回したとおりにサーボが動けば OK
- 動作確認が終わったら USB ケーブルを抜く



## 解説

- ① `int servoAngle = map(angle, 0, 1023, 0, 179);`

角度センサの値 `angle` (0~1023 の範囲) をサーボの角度の範囲 (0~179) に調整しています。  
もし、角度センサを回す方向とサーボが回る方向が反対だったら、方向が合うように書き換えてみましょう。



## データを持ち帰る

- 本日作成したデータは全て USB メモリなどにコピーして持ち帰りましょう
  - 任意の場所に作成した以下のフォルダ (9 個)
  - \_7.1    \_7.2a/b    \_7.3a/b/c/d    \_7.4a/b



## 本日の課題

- ① **\_7.1、\_7.2a、\_7.3a、\_7.4a のコードを完成させる**
  - ・ 資料のコードをもとに自分なりのアレンジを加えても構いません。その際は②の readme.txt の所感にアレンジ内容を書いてください。効果的なアレンジであれば成績評価に加点します。
- ② **readme.txt というテキストファイルを作り、以下を書く**
  - ・ 学籍番号 と 氏名
  - ・ 所感 (考えたこと、感想、応用アイデアなど。字数不問だが数行は書いてほしい。)
  - ・ 質問 (無ければ不要)

### 【提出方法】

- ・ ①のフォルダ 4 個と、②の「readme.txt」1 個、全てをまとめて zip 圧縮
- ・ zip のファイル名は「学籍番号.zip」とする (例: 12345nhu.zip)
- ・ CampusSquare のレポート「フィジカルコンピューティング **07**」から提出

**【提出締切】**    **11 月 24 日 (木) 15:00** (遅れてしまった場合は担当教員に相談のこと)