



[www.KeeYees.com](http://www.KeeYees.com)



**KeeYees**

**KeeYees Electronic Components Kit**



## 序文

### 当社について

当社 KeeYees Technology Inc. は中国のシリコンバレー・深センにあり、オープンソースのハードウェアを研究開発し、製造し、販売する大規模、プロフェッショナルな電子製品の製造販売業者である。当社の製品は品質が国際標準規格に準拠し、アメリカ、ヨーロッパなどの市場で非常に人気がある。KeeYees は個人用から専門家用まで、様々な電子モジュール、コンポーネントなどを扱い、Arduino と Raspberry Pi のことを勉強でき、あなたのベストチョイスです。さらに、3D プリンターパーツ・アクセサリ、コネクタ、端子セット、DIY 部品なども扱っている。家庭用、学校用、業務用の作業とデザインへの挑戦をサポートする！

US Amazon Store Homepage:

<https://www.amazon.com/shops/A2K4DGCC72N9AG>

UK Amazon Store Homepage:

<https://www.amazon.co.uk/shops/A1F4U6XVWUBG1U>

DE Amazon Store Homepage:

<https://www.amazon.de/shops/A1F4U6XVWUBG1U>

FR Amazon Store Homepage:

<https://www.amazon.fr/shops/A1F4U6XVWUBG1U>

IT Amazon Store Homepage:

<https://www.amazon.it/shops/A1F4U6XVWUBG1U>

ES Amazon Store Homepage:

<https://www.amazon.es/shops/A1F4U6XVWUBG1U>

JP Amazon Store Homepage:

<https://www.amazon.co.jp/shops/A7NY3JX21TGU2>

### 本チュートリアルについて

こちらのチュートリアルは初心者向けのものです。 Arduino ボード、センサー、コンポーネントの使用方法に関する基本知識から体験します。 Arduino のことをさらに勉強したい場合は、Michael Margolis の著作「Arduino Cookbook」を読むことがおすすめです。

このチュートリアルの一部分のコードは、Simon Monk が編集したのです。 Simon Monk は「Programming Arduino」、「30 Arduino Projects for the Evil Genius」、「Programming the Raspberry Pi」などのオープンソースハードウェアに関する作品の著者です。 そちらの作品は Amazon で購入出来ると思います。



## 目次

Lesson 0 Arduino とは.....	4
Lesson 1 Blink.....	12
Lesson 2 LED.....	17
Lesson 3 RGB LED.....	25
Lesson 4 ボタン.....	36
Lesson 5 電子ブザーとポテンショメーター.....	42
Lesson 6 圧電スピーカー.....	47
Lesson 7 8つのLEDと74HC595.....	53
Lesson 8 シリアルモニター.....	61
Lesson 9 フォトレジスタ.....	67
Lesson 10 7セグメントLED表示器.....	73
Lesson 11 PCBはんだ付け.....	81



## Lesson 0 Arduino とは

Arduino はハードウェアのマイコンボードとソフトウェアの IDE で構成される、オープンソース・プロトタイピングプラットフォームです。

Arduino のマイコンボード上に、センサー類やアクチュエータ類(モータなど)などの電子部品を接続し、IDE で開発したプログラムをマイコンボードに書き込み、そのプログラムで記述された動作内容でマイコンボードと電子部品を動かします。

ソフトウェア開発環境は無料でダウンロードして利用できます。そして IDE では、Arduino で動かすソフトウェアを開発したり、開発したソフトをマイコンボードに書き込んだり、更にシリアル通信によって PC との通信も可能です。

### Arduino を選択する理由

Arduino はシンプルで、使いやすいため、何千何万のプロジェクトやアプリケーションで使用されています。初心者にとって、入門がやすいですが、上級ユーザーにとって、十分な柔軟性があります。

Windows、Mac OS、Linux に対応していますため、学校で生徒たちは Arduino を利用して低コストの科学機器を作り、化学と物理の原理を証明し、プログラミングを始めます。アーティストやデザイナー、ホビイスト、そしてインタラクティブな物や環境を作りたいと考える、あらゆる人は気軽に Arduino を利用して自分のアイデアを実現できます。

### 特徴

- ・ 安価-Arduino ボードは、他のマイクロコントローラープラットフォームより、安価で入手できます。
- ・ クロスプラットフォーム-Arduino ソフトウェア（IDE）は「Windows」「Macintosh OSX」「Linux」で動作します。他の多くのマイクロコントローラシステムは Windows に限定されています。
- ・ シンプルで明確なプログラミング環境-IDE は初心者にとって使いやすいだけでなく、上級ユーザーにとって十分な柔軟性があります。また、教師にとって、開発環境は Processing に基づくため、生徒たちは勉強しているうちに慣れてきます。
- ・ オープンソースと拡張可能なソフトウェア-Arduino ソフトウェアは、オ

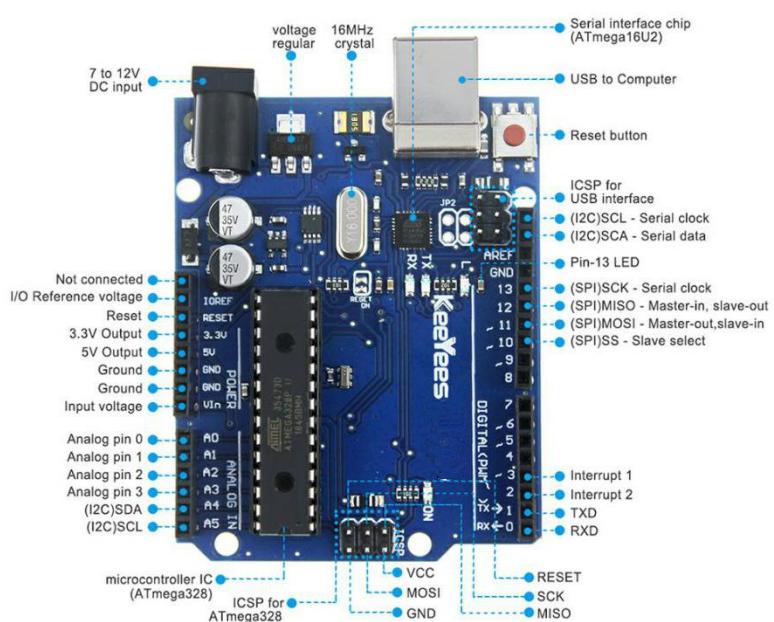


ソースツールとして公開されており、経験豊富なプログラマーによる拡張が可能です。言語はC++ライブラリを通して、拡張できます。

・オープンソースと拡張可能なハードウェア-ユーザーが独自にボードを作成できます。経験豊富な回路設計者は独自のモジュールを作成し、それを拡張して改良できます。経験が浅い人も「モジュールのブレッドボードバージョン」を作って、その仕組みを理解でき、費用も節約できます。

## 開発ボード

本チュートリアルでは、KeeYees Arduino UNOR3 開発ボードを使用してサンプル回路を参考として組みます。別の Arduino UNOR3 開発ボードをお持ちになる場合、使えます。こちらのレッスンでは、本セットにある電子部品を正しく使い方を説明します。



(※こちらの開発ボードはセットに含まれていません)



## Arduino IDE をインストール

Arduino 総合開発環境 IDE は「Windows」「Macintosh OSX」「Linux」で動作できます。

STEP 1: <https://www.arduino.cc/en/Main/Software> こちらのページをアクセスして、以下のページをご注目ください。

The screenshot shows the Arduino Software (IDE) download page. On the left, there's a large Arduino logo with the text "ARDUINO 1.8.9". Below it, a brief description states: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions." On the right, there are download links for different operating systems:

- Windows** Installer, for Windows XP and up
- Windows** ZIP file for non admin install
- Windows app** Requires Win 8.1 or 10  
Get
- Mac OS X** 10.8 Mountain Lion or newer
- Linux** 32 bits
- Linux** 64 bits
- Linux ARM** 32 bits
- Linux ARM** 64 bits

Below these links are links to "Release Notes", "Source Code", and "Checksums (sha512)".

こちらのウェブサイトはいつも最新バージョンを提供しています。今上記のバージョンより新しくなった可能性もあります。

STEP2: コンピューターのシステムと互換性のある開発ソフトウェアをダウンロードします。ここでは、Windows を例としてご参考ください。

The screenshot shows the "Hourly Builds" section of the Arduino website. At the top, it says "HOURLY BUILDS". Below that, a text box says: "Download a **preview of the incoming release** with the most updated features and bugfixes." Underneath, there are download links:

- Windows** (highlighted with a red box)
- Mac OS X (Mac OSX Mountain Lion or later)
- Linux 32 bit, Linux 64 bit, Linux ARM, Linux ARM64



[Windows](#) をクリックして、ZIP ファイルをダウンロードします。

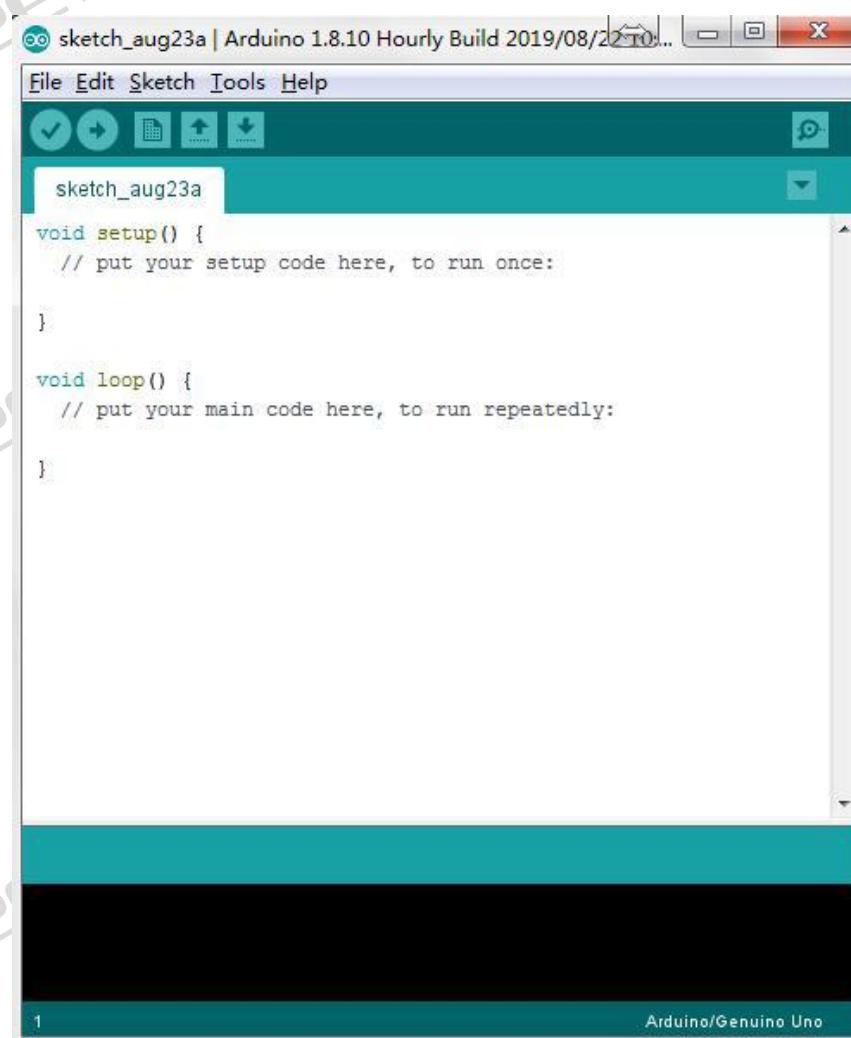
[arduino-nightly-windows.zip](#)

[arduino-nightly](#) ファイルをクリックします。

[arduino-nightly](#)  
 [arduino-nightly-windows.zip](#)

[arduino.exe](#) をクリックします。

drivers	2019/8/23 11:03
examples	2019/8/23 11:03
hardware	2019/8/23 11:03
java	2019/8/23 11:03
lib	2019/8/23 11:03
libraries	2019/8/23 11:03
reference	2019/8/23 11:03
tools	2019/8/23 11:03
tools-builder	2019/8/23 11:03
<b>arduino.exe</b>	2019/8/22 10:33
arduino.l4j.ini	2019/8/22 10:33
arduino_debug.exe	2019/8/22 10:33
arduino_debug.l4j.ini	2019/8/22 10:33
arduino-builder.exe	2019/8/22 10:33

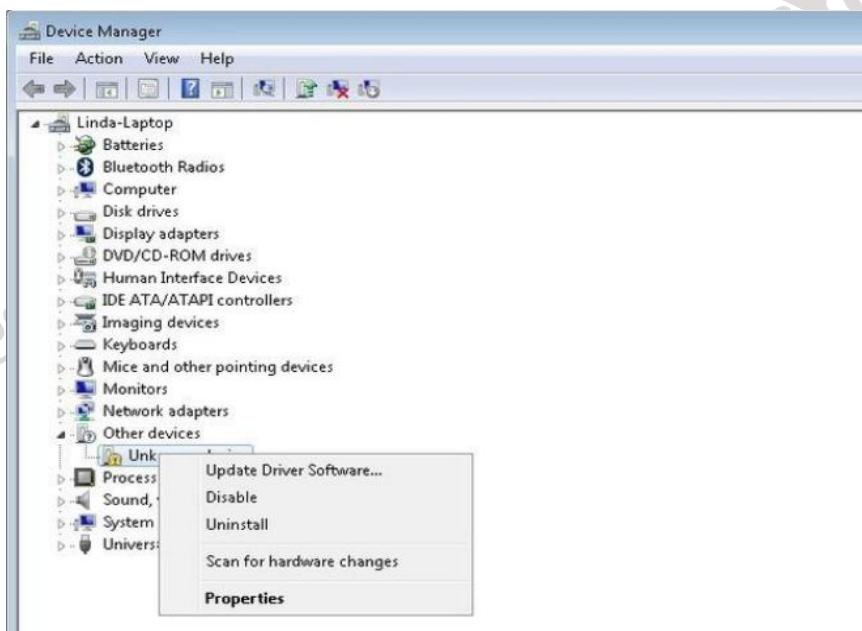




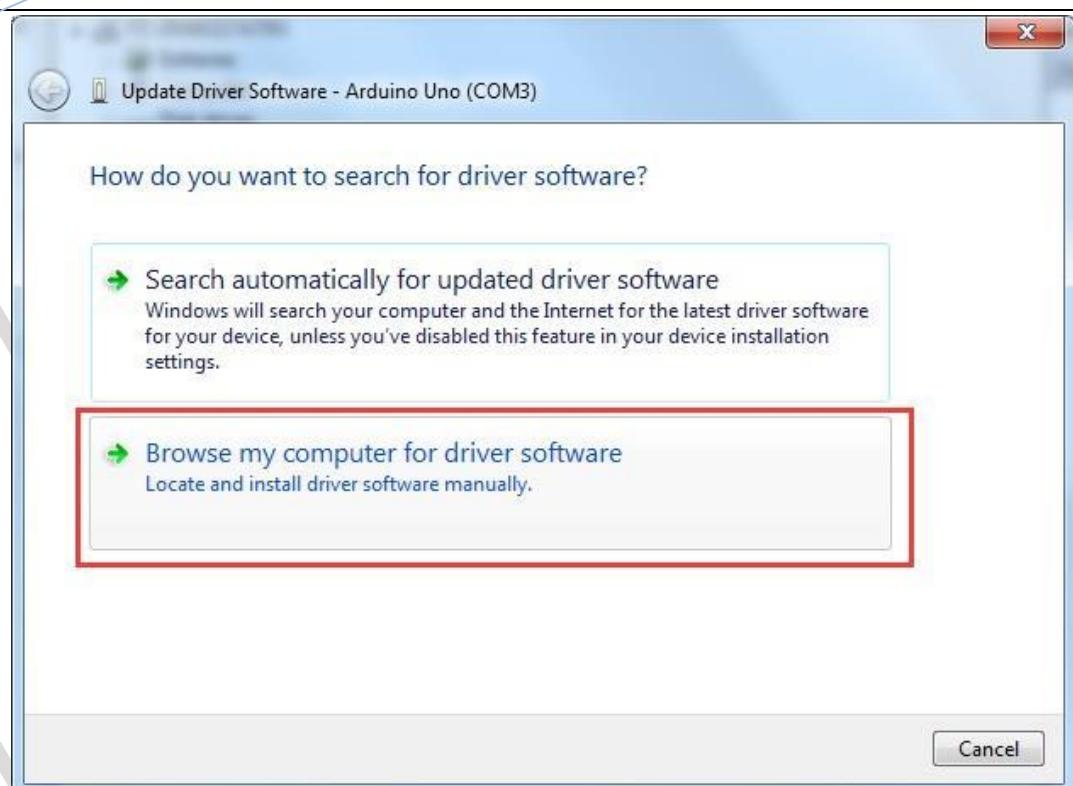
次に、Arduino の USB ドライバーをインストールする必要があります。Arduino ファイルには、Arduino プログラム自体と、USB ケーブルで PC に接続できるようになるドライバーが含まれています。

**Step 1:** Arduino 開発ボードを USB ケーブルで PC に接続します。電源 LED が点灯します。そして、PC から「新しいハードウェアが見つかりました」というメッセージが表示される場合があります。このメッセージを無視してください。PC がドライバーを自動的にインストールしすることもキャンセルしてください。「デバイス マネージャー」を利用して USB ドライバーをインストールすることをおすすめします。

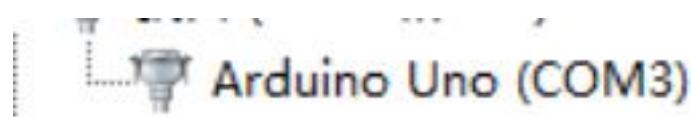
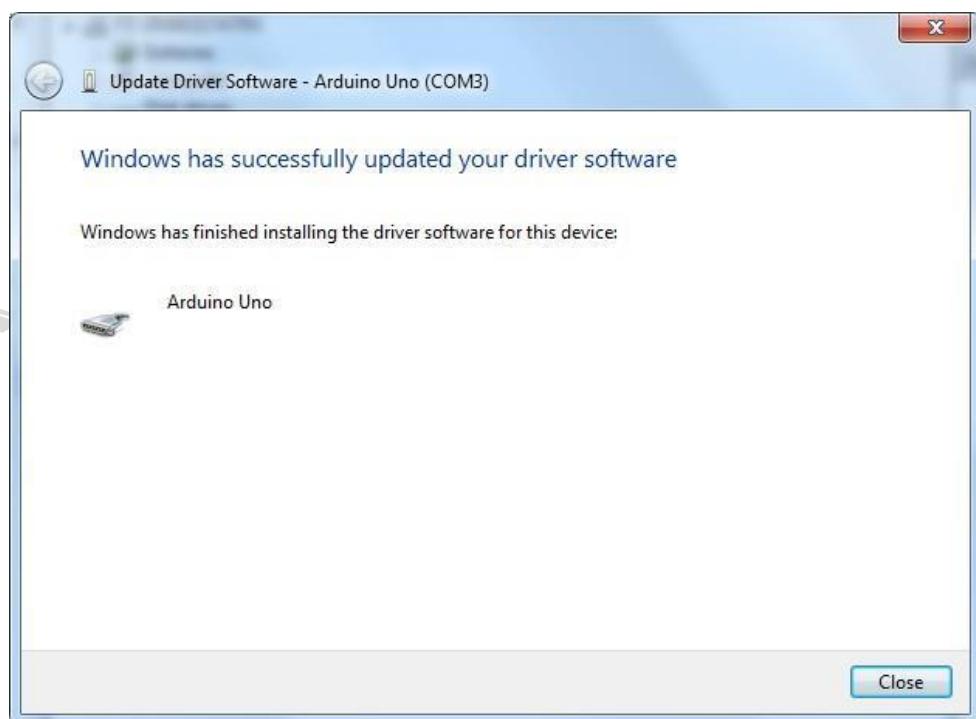
**Step 2:** [デバイス マネージャ]-> [他のデバイス]をクリックします。以下のように、「不明なデバイス」の黄色の警告が表示されると思いますが、これが Arduino 開発ボードです。



**Step 3:** 「ドライバソフトウェアの更新」->「コンピューターを参照してドライバソフトウェアを検索します」をクリックします。



Step 4: [次へ]をクリックすると、セキュリティ画面が表示されましたら、ソフトウェアのインストールを許可してください。しばらくするとインストール完了画面が表示されます。「閉じる」をクリックします。



これで USB ドライバーのインストールが完了し、Arduino が使えるようになります。

コンピューターが Arduino 開発ボード デバイスを認識できる場合は、ドライバーのインストールする手順をスキップしてください。



## Lesson 1 Blink

### 概要

このレッスンでは、Arduino の内蔵 LED が点滅するプログラムと、プログラムをダウンロードする方法を学びます。

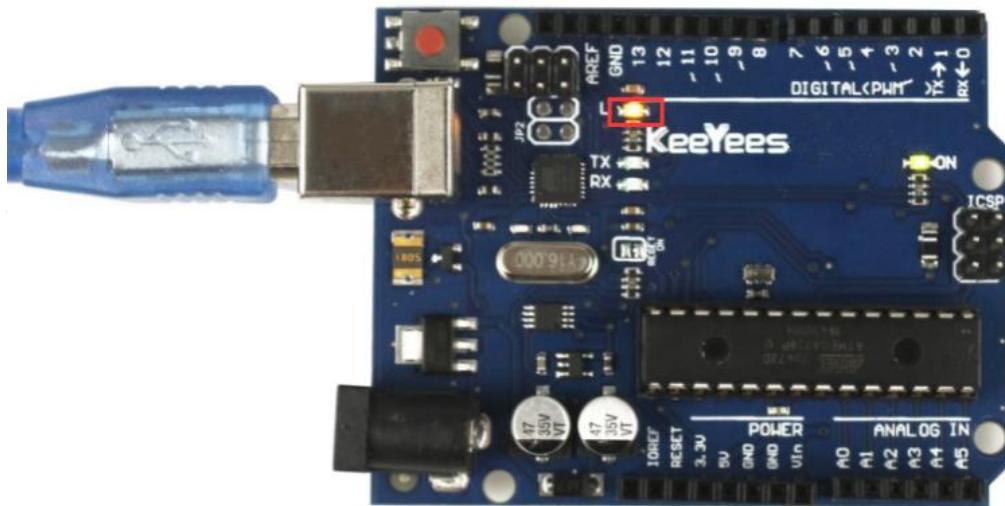
必要なもの：

Arduino UNOR3 開発ボード

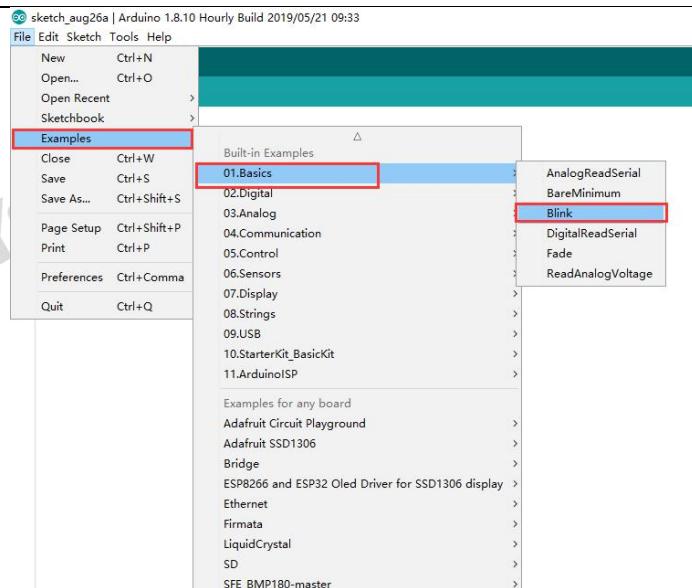
### 原理

開発ボードの両側には複数の I/O ポートがあります。そちらのポートを利用してモジュールとシールドに接続して拡張するのです。

中に内蔵の LED があります。ボード上で「L」と表示されています。この LED はスケッチから制御できる LED です。



Arduino IDE には多くのサンプルプログラムがあります。今回は「Blink」を使用します。以下のように [ファイル]-[スケッチ例]-[01. Basics]-[Blink] をクリックします。

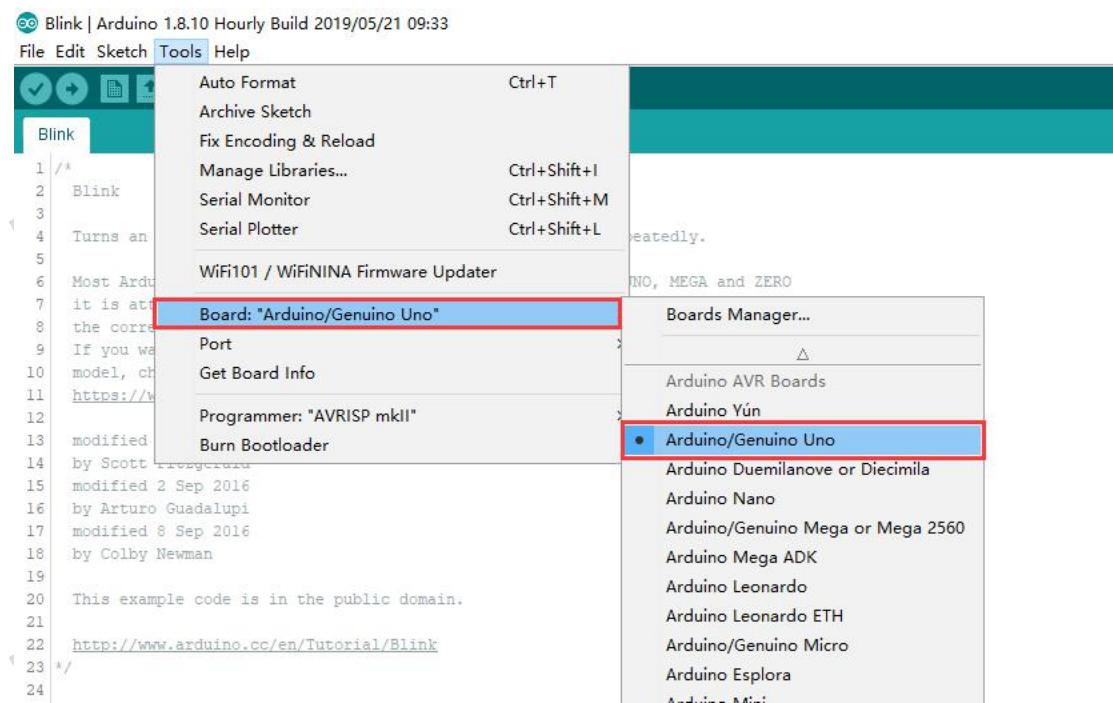


別ウィンドウが開き、Blink のサンプルコードが表示されます。

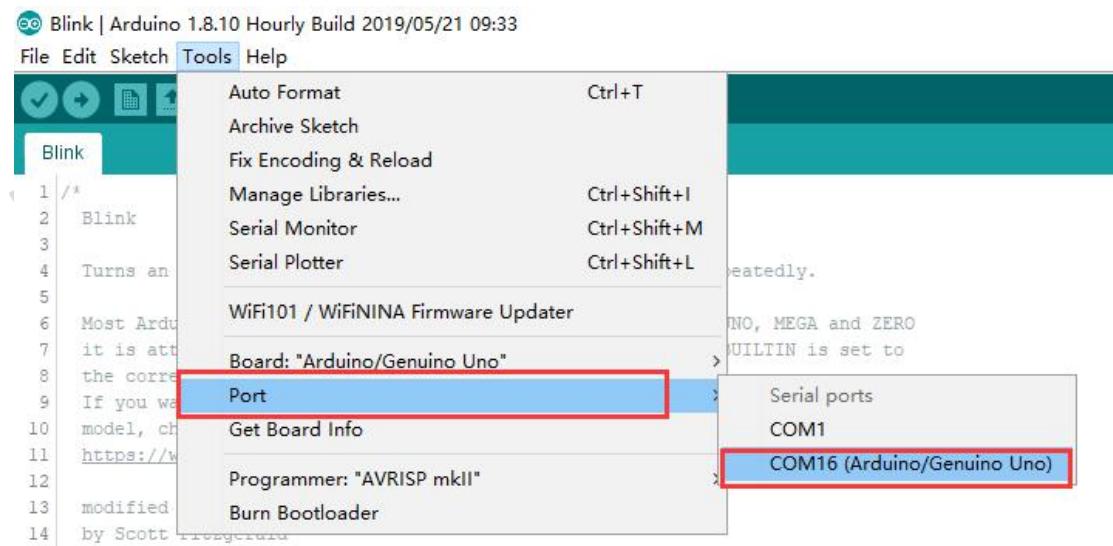
```
/* 
  Blink
  Turns an LED on for one second, then off for one second, repeatedly.
  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products
  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman
  This example code is in the public domain.
  http://www.arduino.cc/en/Tutorial/Blink
*/
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```



次に、Arduino ボードをパソコンに接続し、以下の初期設定を行います。[ツール]-[ボードマネージャ]-[Arduino/Genuino Uno] を選択します。

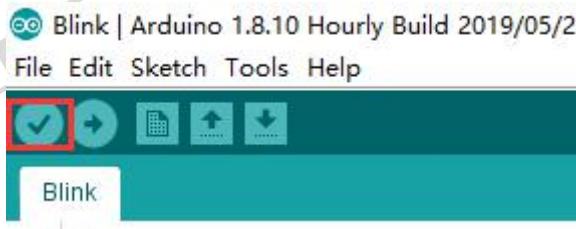


ボードを設定した後、[ツール]-[ポート] から Arduino が接続されている COM ポートを選択します。





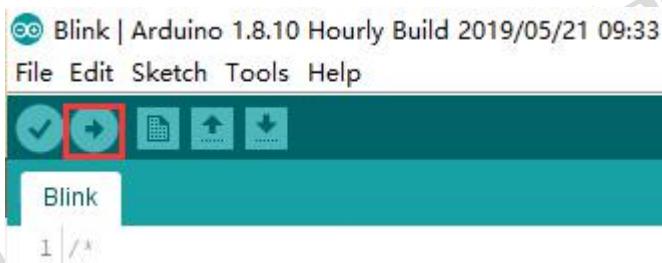
プログラムがきちんとかけているかチェックするコンパイルを行います。



エラーがなければ、Arduino IDE に以下のメッセージが表示されます。もしコンパイルが失敗したら、エラーメッセージが表示されます。

```
Done compiling.  
Sketch uses 930 bytes (2%) of program storage space. Maximum is 32256 bytes.  
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

コンパイルが通れば、Arduino に書き込むアップロードを行います。



コードが無事アップロードすれば、以下のようなメッセージが表示されます。

```
Done uploading.  
Sketch uses 930 bytes (2%) of program storage space. Maximum is 32256 bytes.  
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

それに、Arduino ボード上の LED が 1 秒おきに点滅するはずです。

コードを解読いたします：



## ご注意

このスケッチの大部分はコメントで構成されています。これはプログラム命令ではありません。プログラムがどのように機能するかを説明するだけです。参考できます。

スケッチの最初は何行かはコメントです。/\*と\*/の間はすべて Arduino から無視されます。スケッチを説明する目的で存在します。

1行だけのコメントをつけるときには、//をつけます。//から行末まではコメントです。

はじめのコードは以下:

```
void setup() {  
// initialize the digital pin as an output.  
pinMode(led, OUTPUT);  
}
```

Arduino のスケッチには「setup」関数が必要です。関数名の前後にある文字は、この関数の型とパラメータ(引数)を指定しています。波カッコ{}に挟まれたコードがこの関数の実体であり、実際に仕事をする部分です。

上の例の pinMode() 関数はあるピンを入力か出力のどちらかに設定します。

setup() 関数以外、loop() 関数も必要です。setup() はスケッチがスタートしたときに 1 度だけ呼び出されます。ピンモードやライブラリの初期化といった処理をするのに都合がいい場所です。loop() 関数はスケッチの心臓部であり、繰り返し実行されます。もし一方の関数が不要だとしても、スケッチには両方を書く必要があります。

```
void loop() {  
digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
delay(1000); // wait for a second  
digitalWrite(led, LOW); // turn the LED off by making the voltage LOW  
delay(1000); // wait for a second  
}
```

loop の関数に digitalWrite() 関数はピンに値を出力します。たとえば上の行はまず led を HIGH にセットします。1 秒後、LOW にセットします。LED の点滅を実現させます。

delay() は次の行へ進む前に Arduino を指定したミリ秒だけ待機させます。1 秒は 1000 ミリ秒です。



```
// Main program loop
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
    delay(1000);                         // wait for a second
    digitalWrite(LED_BUILTIN, LOW);        // turn the LED off by making the voltage LOW
    delay(1000);                         // wait for a second
}
```

もし、LED の点滅間隔を変えたいなら、`delay()` の値を変えればいいです。例えば、「`delay(1000)`」を「`delay(500)`」に書き換えると、LED の点滅間隔が 0.5 秒おきに変わります。

## Lesson 2 LED

### 概要

抵抗器を使って LED の光の調整を学びます。

### 必要なもの

1 x 開発ボード

1 x ブレッドボード

1 x 5mm LED

1 x 220Ω 抵抗器

1 x 1kΩ 抵抗器

1 x 10kΩ 抵抗器

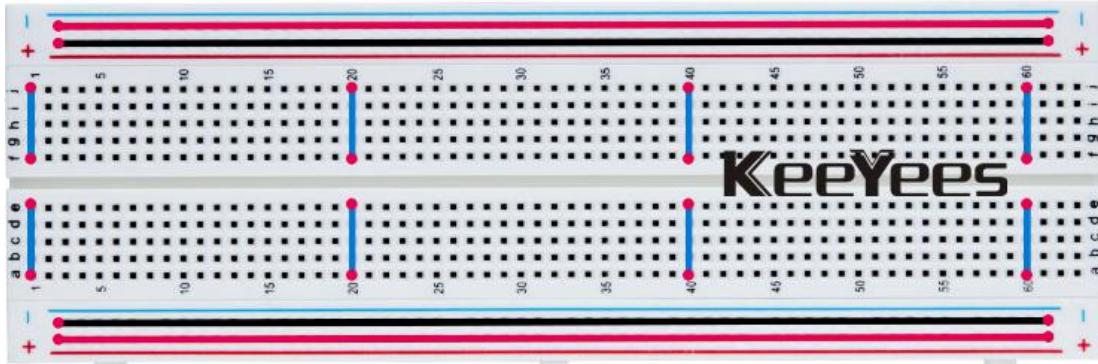
2 x ジャンパーウイヤー オス-オス

### 部品の説明

ブレッドボード

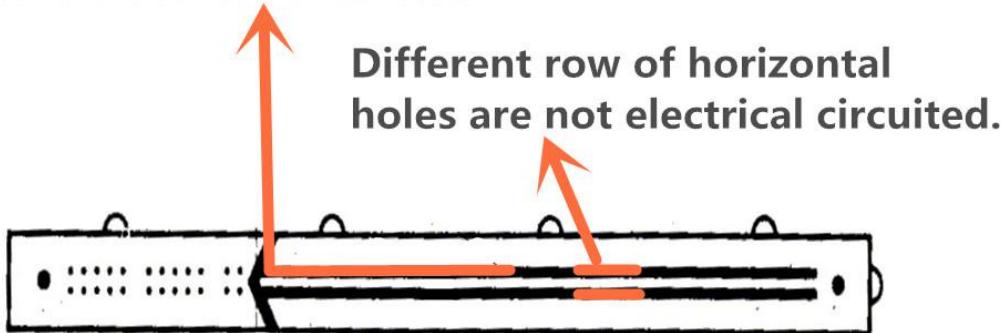


ブレッドボードははんだ付け不要で電子回路の試作や実験ができる便利なものです。



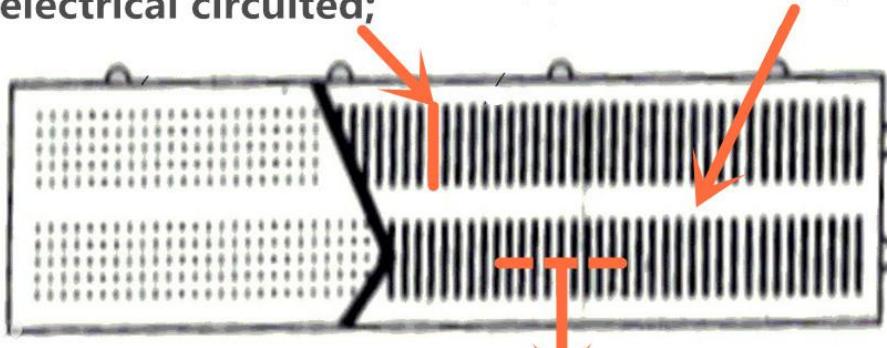
ブレッドボードのソケット（穴）に電子部品のリードを入れると回路が作られます。ソケットは、赤線のようにつながっています。例えば 1 列目の「a～e」はつながっており、同じ 1 列目の「f～j」もつながっています。「a～e」と「f～j」はつながっていません。上下の「+」と「-」は電源用ラインです、横方向につながっています。これを理解して電子部品のリードを穴に入れて回路を作ります。

The same row of horizontal holes is electrical circuited.



Vertical 5 holes are electrical circuited;

The Groove to isolate the upper and lower parts.



All horizontal holes are not electrical circuited.

## LED

よく使われている LED は 3 種類があります。直径 3mm、5mm と 10mm です。このレッスンでは 5mm の青の LED を使っていきます。

LED をバッテリーまたは電源に直接接続することはできません。

注意事項 :

- 1) LED にはプラスとマイナスの極性があり、逆に接続すると、点灯しません。
- 2) 抵抗を使って、流れる電流を制限する必要があります。 そうしないと、過大電流が流れる原因で LED は破壊となります。



極性を見分ける方法：リード線が長いほうはプラス、短いほうはマイナスです。

### 抵抗器

抵抗器は電気回路用部品として、電流の制限や、電圧の分圧、時定数回路などの用途に用いられます。電流は抵抗が大きいと流れにくくなり、小さいと流れやすくなります。抵抗器を使って流れる電流を制限して LED の光を調整します。

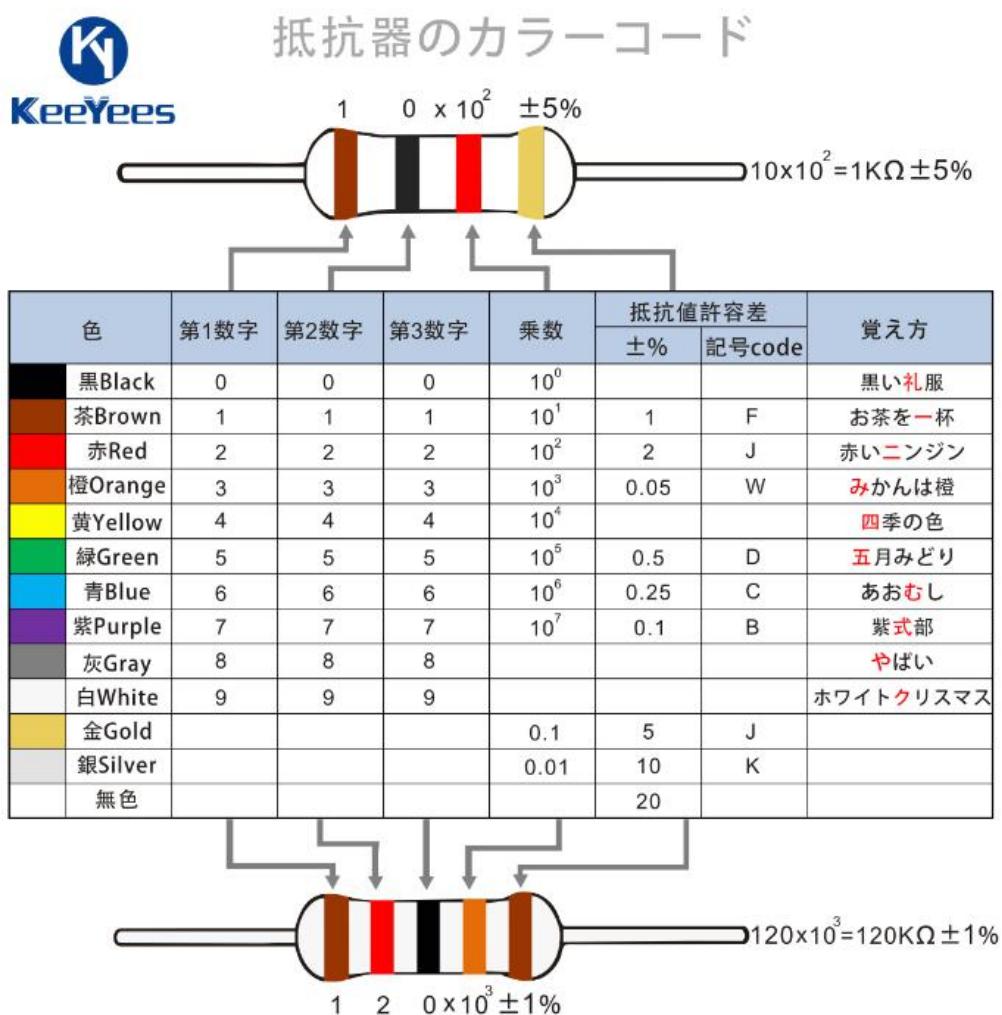
抵抗の単位は  $\Omega$  です。 $k\Omega$  ( $1,000\Omega$ ) と  $M\Omega$  ( $1,000,000\Omega$ ) もあります。「キルオーム」、「メガオーム」と読みます。



このレッスンでは、 $220\Omega$ 、 $1k\Omega$ 、 $10k\Omega$  の 3 つの抵抗器を使用します。形が同じですが、カラーが違います。カラーによって、抵抗値を読み取れます。



抵抗器のカラーコードを以下の図の通りです。ご参考ください。

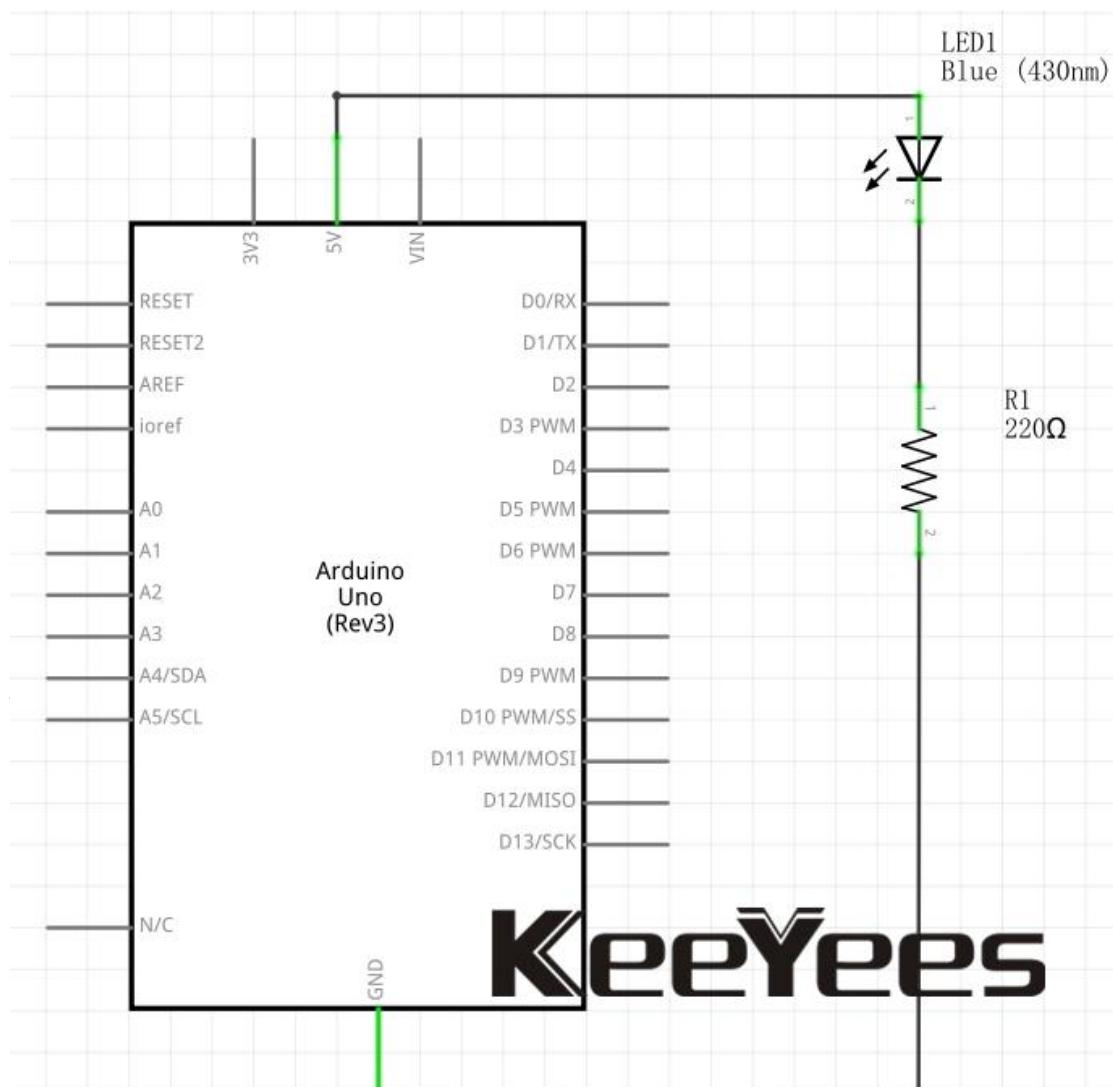


LED と違って、抵抗器には極性がありません。逆接続の心配はありません。

カラーコードで識別には難があると、デジタルマルチメーターを使って直接に計測してもいいです。



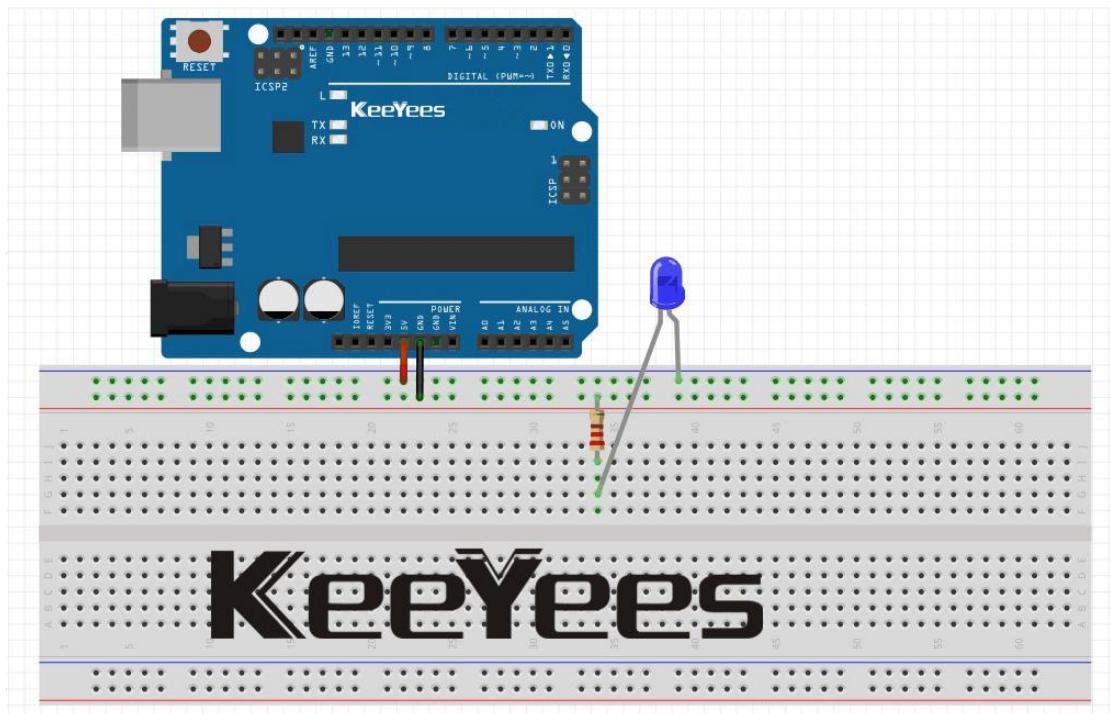
接続回路図



KeeYees



配線図

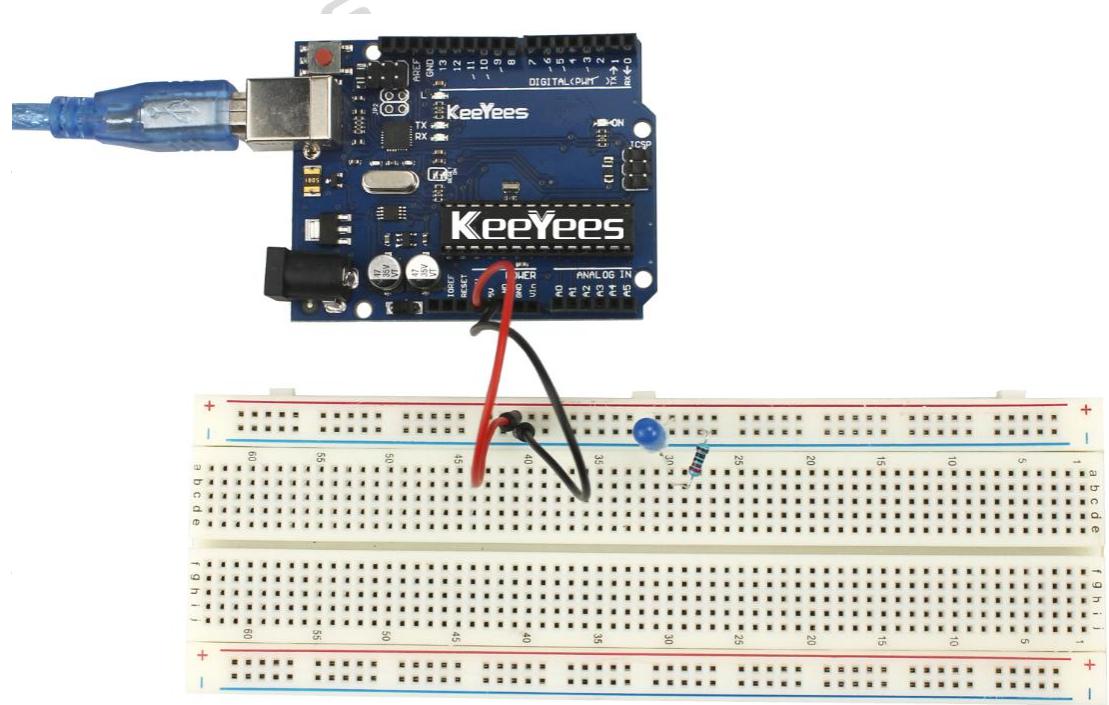




Arduino 開発ボードの 5V 出力から、LED と抵抗器に電源を供給します。  
Arduino 開発ボードを USB ケーブルで電源に接続します。

220Ωの抵抗をつけると、LED はかなり明るくなります。 220Ωの抵抗を 1kΩ の抵抗に変更すると、LED は少し暗くなります。 最後に、10kΩの抵抗をつけると、LED の光がほぼ見えなくなります。

配線についてですが、ジャンパーウイヤーを使って Arduino の 5V 出力ピンを 抵抗器の 1 端のピンに繋げ、もう 1 端のピンを LED の長めのピン(プラス)に繋げ、短めのピン(マイナス)を Arduino の GND に繋げます。電源をオンにすると、LED は点灯するはずです。





## Lesson 3 RGB LED

### 概要

RGB は、Red、Green、Blue の三原色の頭文字です。RGB LED は、赤、緑、青の3つの基本色を混ぜることで好きな色を出すことができる LED です。

### 必要なもの

- 1 x Arduino UNOR3 開発ボード
- 1 x 830 タイピントブレッドボード
- 4 x ジャンパーウイヤー オス-オス
- 1 x RGB LED
- 3 x 220Ω 抵抗器

### 部品の説明

#### RGB

RGB LED は普通の LED に見えますが、普通の LED より2本のピンが多いです。Red、Green、Blue の3本のピンと、共通のGNDピンから構成されています。

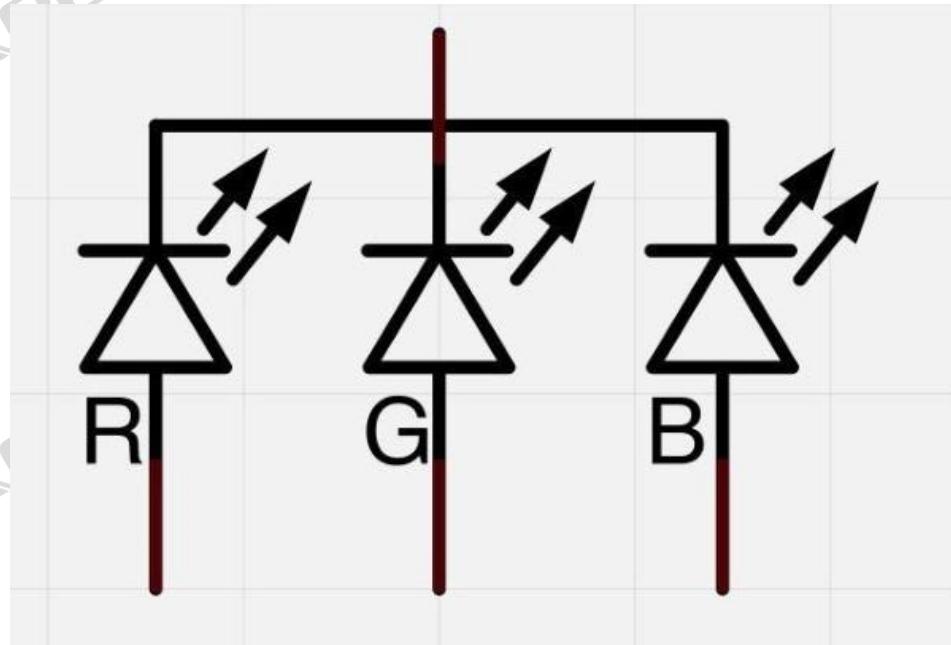
赤、緑、青の3つのLEDの光を調整して好きな色を出します。色を混ぜる方法は、ペイントをパレット上で混ぜる方法と同じです。メインコントロールボードなしで色を調整することはとても難しいです。今回は開発ボードを使用して作業を簡単化します。このボードにはアナログ入出力のピンを利用します。「～」が付いているピンを使用して出力する電力を調整することにより、LEDを制御できます。

このレッスンでは、共通のGNDピンをGNDに接続、他の3つのピンは「～」付きのデジタルピンに接続します。



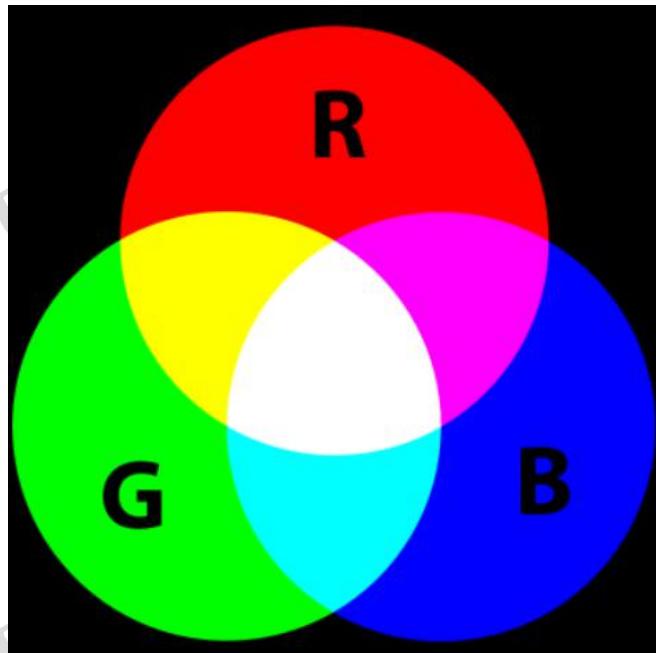


共通の GND ピンは直接に GND に繋げますが、他のピンは電源に繋げるには抵抗器が必要です。上から Blue、Green、GND、Red ピンです。



## カラー

赤、緑、青の 3 つの LED の光を調整することで好きな色を混ぜることができます。理論的には、人間の目には 3 種類の受光器（赤、緑、青）があります。目で見ながら、脳が処理してスペクトルに変換します。カラーテレビ、コンピューター、携帯などの電子製品はいろんな色を表示できます。原理は同じです。

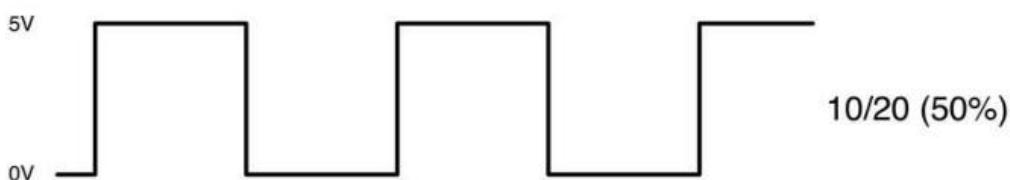
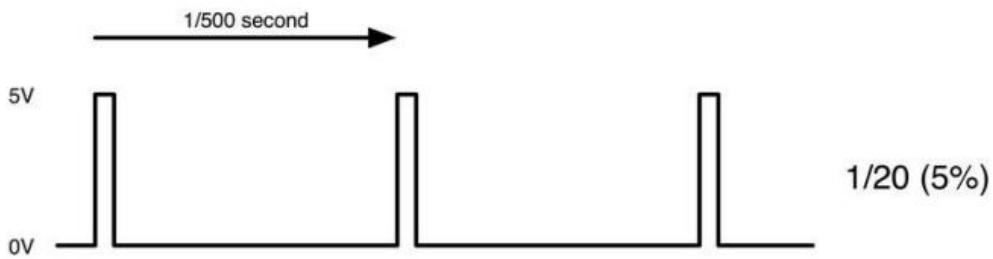


たとえば、3つのLEDを同じ明るさに設定した場合、RGB LEDは白くなります。青いLEDをオフにすると、赤と緑のLEDだけを同じ明るさに設定すると、RGB LEDは黄色になります。黒にしたいなら、3つのLEDの明るさを最低限に調整すればいいです。開発ボードを使用して、各LEDの明るさを簡単に制御できます。従って、様々な色を簡単に制御できます。

### 理論 (PWM)

パルス幅変調 (PWM) は、出力される電力を制御する手段として使用されています。原理は出力信号のデューティ比を変更することです。ここで各 LED の明るさを制御するために使用します。

下の図は開発ボードの某 PWM ピンの信号を示しています。



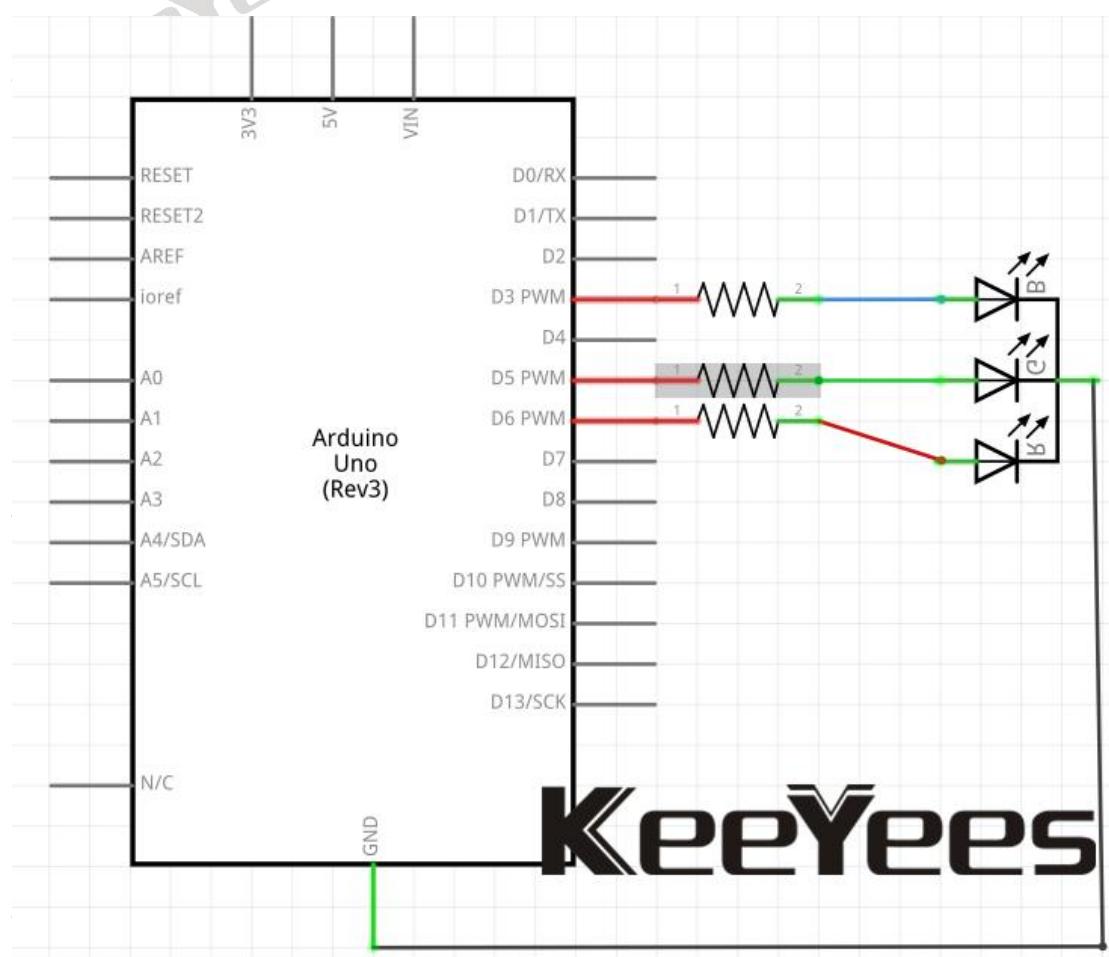
約 1/500 秒ごとに、PWM はパルス信号を生成して出力します。周期は 「`analogWrite`」 関数によって制御されます。 「`analogWrite (0)`」 はパルス信号を生成しません。 「`analogWrite (255)`」 は次のパルスが到着するまでパルスが連続します。この過程では、出力はいつもします。

「`analogWrite`」 関数で 0~255 の値を指定して、パルスが発生します。出力パルスが 5% の時間内だけ High にする場合、本サイクルでは 5% の電力しか得られません。 出力パルスが 90% の時間内に High にする場合、本サイクルで 90% の電力が得られます。

パルスを通して LED のオン/オフの変化は人間の目で見れませんが、LED の色の変化が見えます。

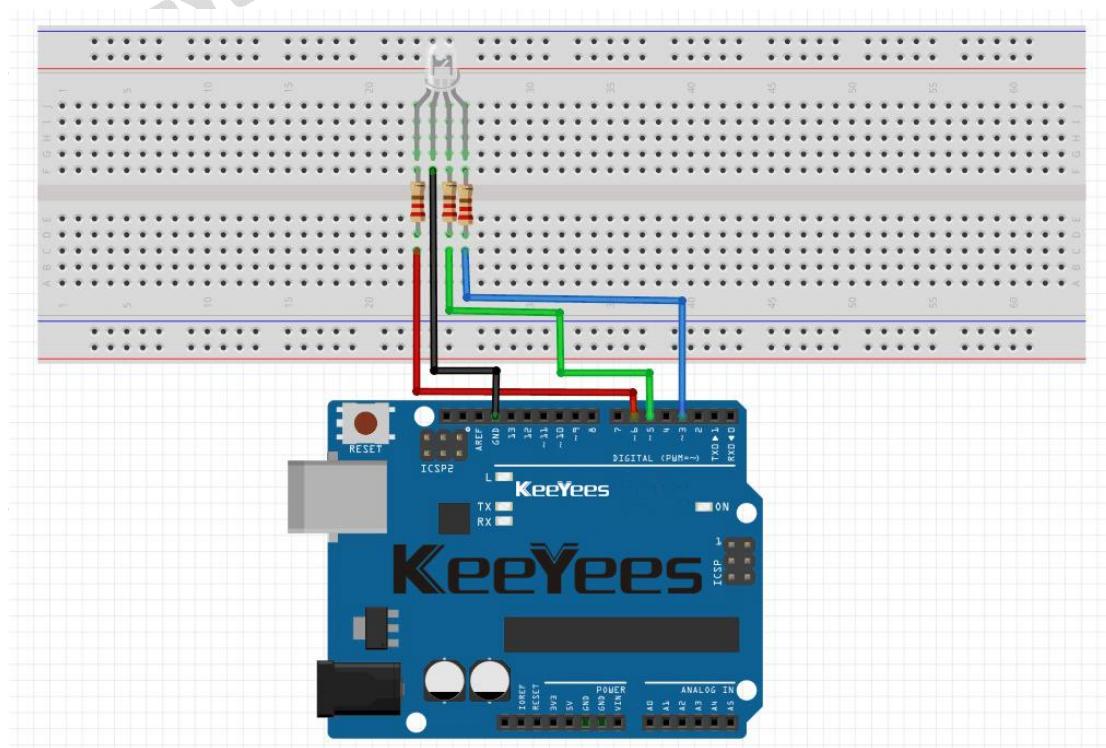


接続回路図





配線図

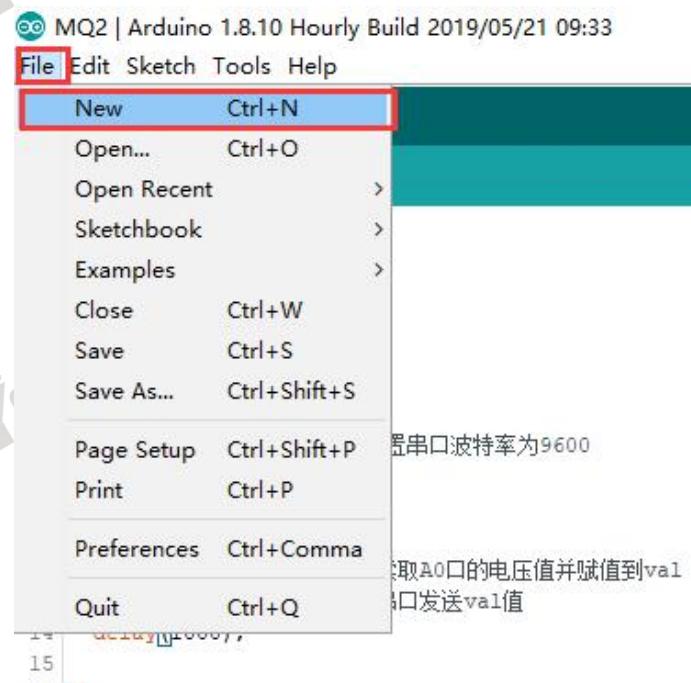




コード

新しいファイルを作成する

Arduino IDEを開いて以下のように新しいファイルを作成します。



The screenshot shows the Arduino IDE with a new sketch titled "sketch\_jul12b". The code in the editor is:

```
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8 }
9 }
```

The status bar at the bottom right shows "Arduino/Genuine Uno on COM10".



以下のコードをコピーし、作成したファイルにある元のコードを置き換えてペーストします。そして、プログラムがきちんとかけているかチェックするコンパイルを行います。コンパイルが通れば、Arduinoに書き込むアップロードを行います。手順はレッスン1と同じです。お忘れの場合、レッスン1をご確認ください。

```
#define BLUE 3
#define GREEN 5
#define RED 6

void setup()
{
pinMode(RED, OUTPUT);
pinMode(GREEN, OUTPUT);
pinMode(BLUE, OUTPUT);
digitalWrite(RED, HIGH);
digitalWrite(GREEN, LOW);
digitalWrite(BLUE, LOW);
}

// define variables
int redValue;
int greenValue;
int blueValue;

// main loop
void loop()
{
#define delayTime 10 // fading time between colors

redValue = 255; // choose a value between 1 and 255 to change the color.
greenValue = 0;
blueValue = 0;

// this is unnecessary as we've either turned on RED in SETUP
// or in the previous loop ... regardless, this turns RED off
// analogWrite(RED, 0);
// delay(1000);

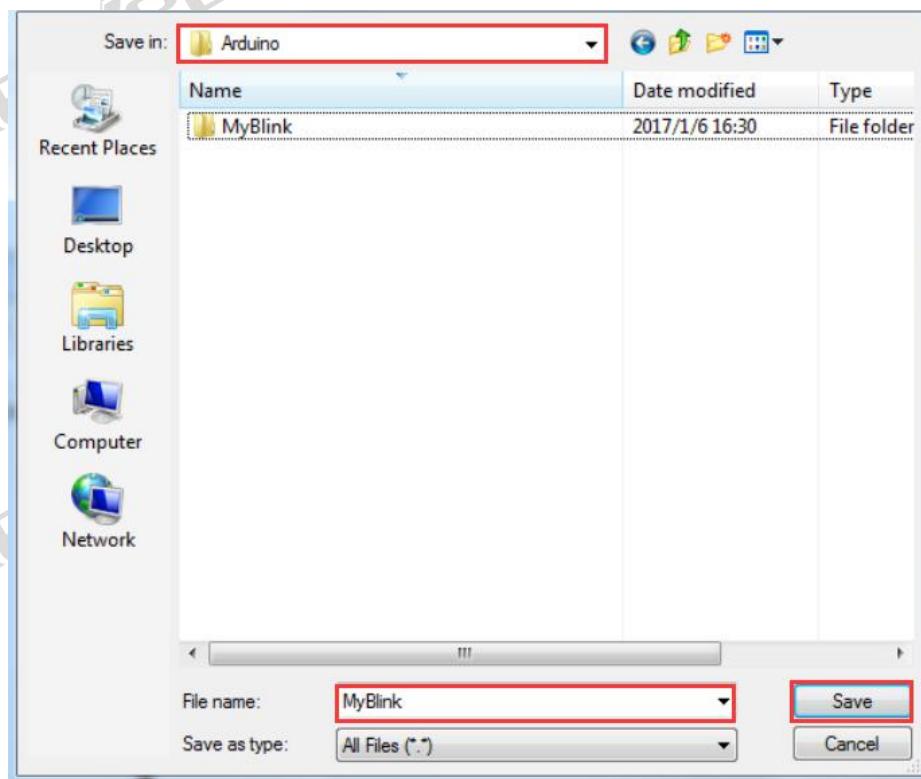
for(int i = 0; i < 255; i += 1) // fades out red bring green full when i=255
{
redValue -= 1;
```



```
greenValue += 1;  
// The following was reversed, counting in the wrong directions  
// analogWrite(RED, 255 - redValue);  
// analogWrite(GREEN, 255 - greenValue);  
analogWrite(RED, redValue);  
analogWrite(GREEN, greenValue);  
delay(delayTime);  
}  
  
redValue = 0;  
greenValue = 255;  
blueValue = 0;  
  
for(int i = 0; i < 255; i += 1) // fades out green bring blue full when i=255  
{  
greenValue -= 1;  
blueValue += 1;  
// The following was reversed, counting in the wrong directions  
// analogWrite(GREEN, 255 - greenValue);  
// analogWrite(BLUE, 255 - blueValue);  
analogWrite(GREEN, greenValue);  
analogWrite(BLUE, blueValue);  
delay(delayTime);  
}  
  
redValue = 0;  
greenValue = 0;  
blueValue = 255;  
  
for(int i = 0; i < 255; i += 1) // fades out blue bring red full when i=255  
{  
// The following code has been rearranged to match the other two similar sections  
blueValue -= 1;  
redValue += 1;  
// The following was reversed, counting in the wrong directions  
// analogWrite(BLUE, 255 - blueValue);  
// analogWrite(RED, 255 - redValue);  
analogWrite(BLUE, blueValue);  
analogWrite(RED, redValue);  
delay(delayTime);  
}  
}
```



コードが実行される際、次のようなポップアップウィンドが出でてきます。ファイルに保管できます。



今回は「FOR Loop」を使用して、LED の色を繰り返して変更します。

最初の「FOR Loop」は赤から緑に変わります。

2番目の「FOR Loop」は緑から青に変わります。

最後の「FOR Loop」が青から赤に変わります。

プログラミングのために、以下のように色をピンに設定する必要があります。  
もちろん、こちらと違うピンに設定することもできますが、「～」付きのピンに設定してください：

```
// Define Pins
```

```
#define BLUE 3
```

```
#define GREEN 5
```

```
#define RED 6
```

次は「setup」関数です。前も説明しましたが、「pinMode」関数はピンを入力か出力のどちらかに設定します。「digitalWrite」関数はピンの電圧を HIGH か LOW に設定します。こちらはLEDを赤のみに表示することに設定しています。自分で変更することもできます：

```
void setup()
```

```
{
```

```
  pinMode(RED, OUTPUT);
```



```
pinMode(GREEN, OUTPUT);  
pinMode(BLUE, OUTPUT);  
digitalWrite(RED, HIGH);  
digitalWrite(GREEN, LOW);  
digitalWrite(BLUE, LOW);  
}
```

次は PWM の初期変数を設定します：

```
redValue = 255; // choose a value between 1 and 255 to change the color.
```

```
greenValue = 0;
```

```
blueValue = 0;
```

「`analogWrite`」関数は、上記の3つのパラメーターを制御します。1つは赤の明るさ、1つは緑の明るさ、もう1つは青の明るさです。0~255の値を指定して制御します。0はオフ、255は最大輝度を示します。そして「`analogWrite`」関数がパラメーターを設定して、各LEDの輝度を調整します。

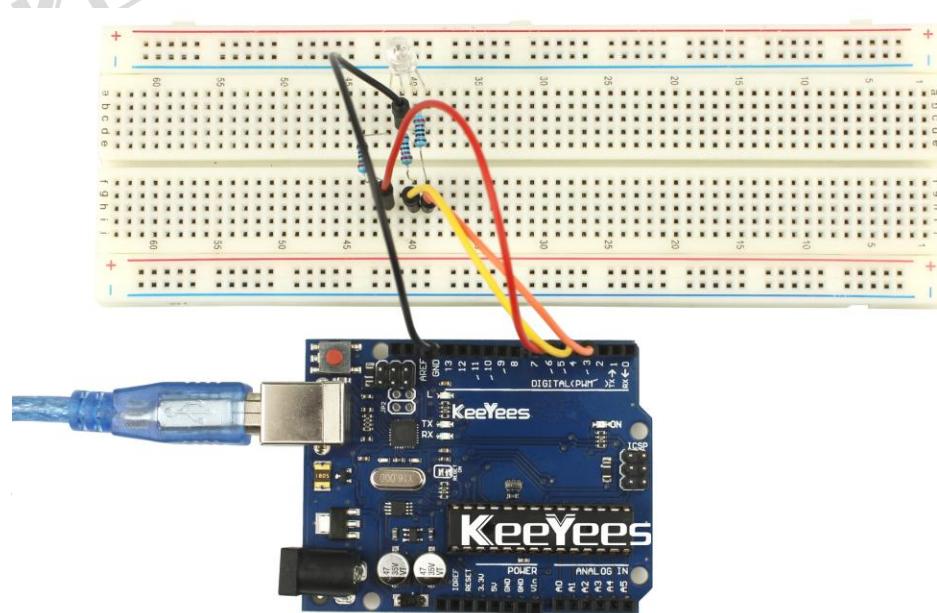
「loop」関数を見れば、始めは赤に設定したことがわかりました。1秒おきに次の色に変わります：

```
#define delayTime 10 // fading time between colors
```

```
Delay(delayTime);
```

プログラムに複数の値を追加してRGB LEDの変化を観察することもおすすめです。

例





## Lesson 4 ボタン

### 概要

このレッスンではボタンを使って LED をオンまたはオフにする方法を学びます。1つのボタンを押すと、LED が点灯します。もう1つのボタンを押すと、LED がオフになります。

### 必要なもの

1 x Arduino UNOR3 開発ボード

1 x 830 タイポイントブレッドボード

7 x ジャンパーウイヤー オス-オス

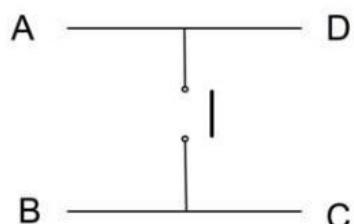
1 x 5mm LED

1 x 220Ω 抵抗器

2 x ボタン

### 部品の説明

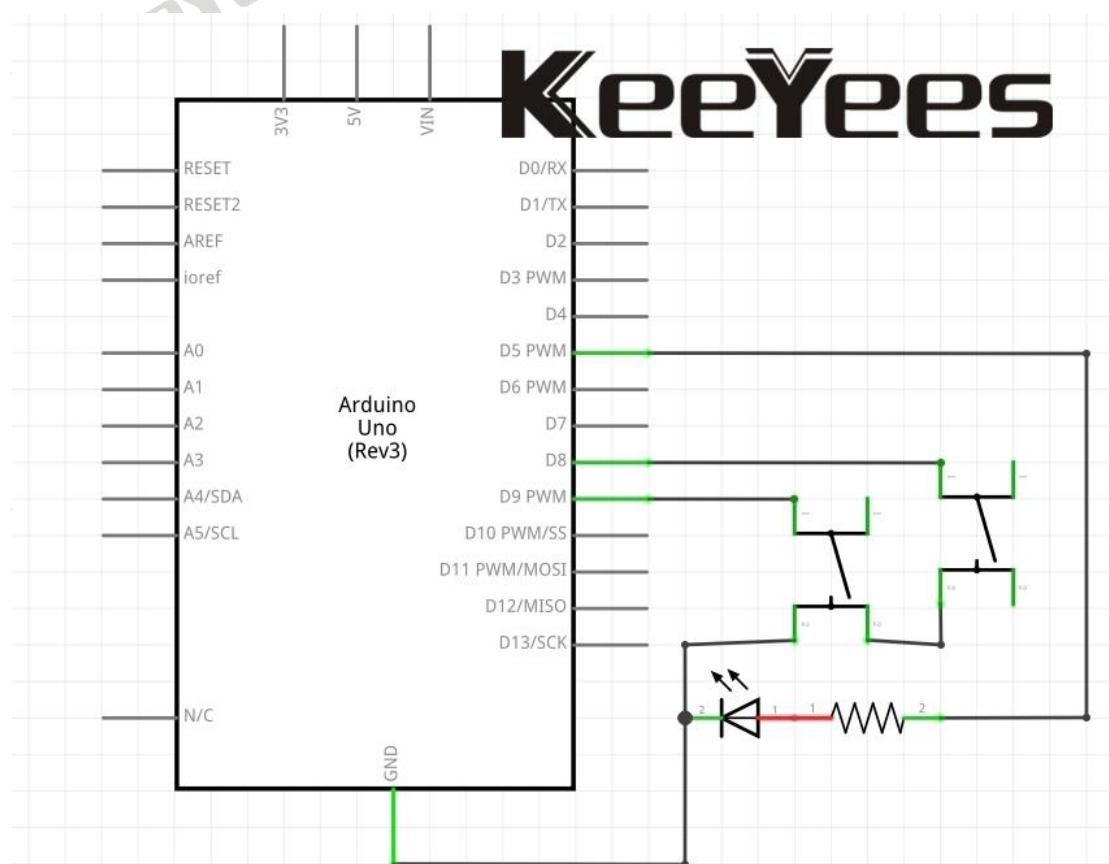
こちらのスイッチはとてもシンプルな部品です。押した時に、2本足リードが接続、導通します。このレッスンではタクトスイッチを使用します。



タクトスイッチは4本足のリードが出ていて、2本ずつが内部でつながっています。AD、BC 内部でつながっています。

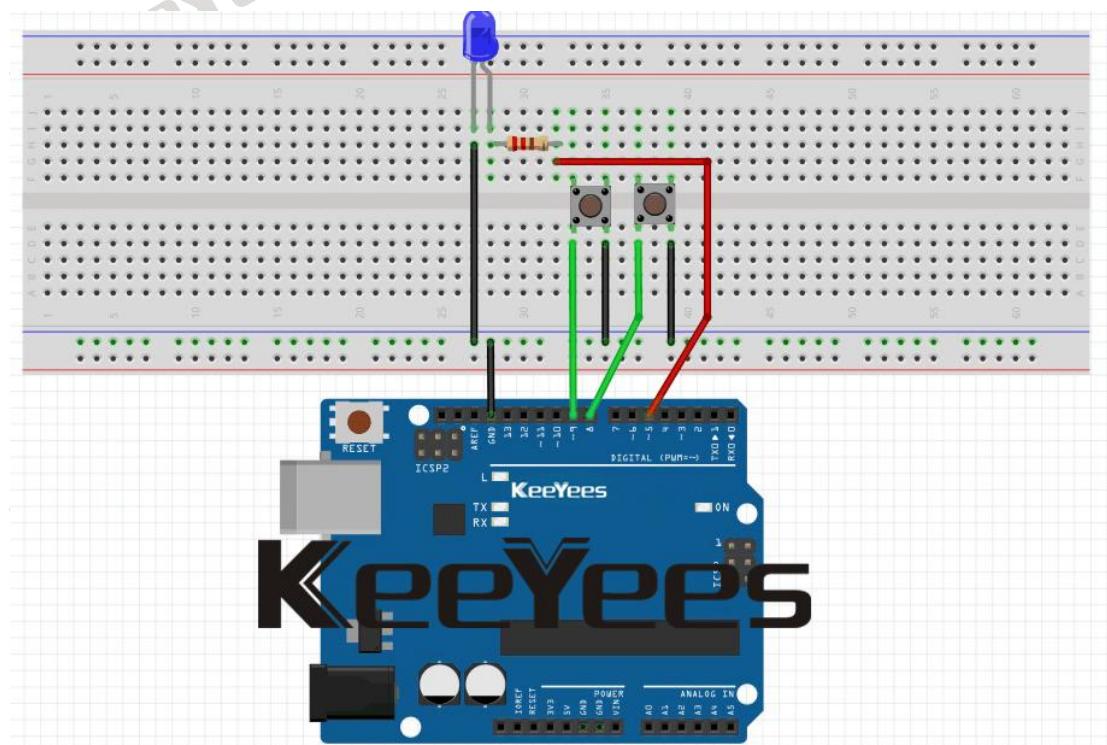


接続図





配線図





スイッチの本体は正方形ですが、2本のリードともう2本のリードの間には距離があり、ブレッドボードに挿して使用には十分です。

注意すべきこと：LEDにはプラスとマイナスの極性があり、リード線が長いほうはプラス、短いほうはマイナスです。そして、過電流を防ぐため、抵抗器も必要です。

### コード

Arduino IDEを開いて ⇒ 新しいファイルを作成します ⇒ 以下のコードをコピーし、作成したファイルにある元のコードを置き換えてペーストします ⇒ プログラムがきちんとかけているかチェックするコンパイルを行います ⇒ コードをファイルに保管できます ⇒ Arduinoに書き込むアップロードを行います。

手順はレッスン1と同じです。お忘れの場合、レッスン1をご確認ください。

```
int ledPin = 5;
int buttonApin = 9;
int buttonBpin = 8;
void setup()
{
    pinMode(ledPin, OUTPUT);
    pinMode(buttonApin, INPUT_PULLUP);
    pinMode(buttonBpin, INPUT_PULLUP);
}

void loop()
{
    if (digitalRead(buttonApin) == LOW)
    {
        digitalWrite(ledPin, HIGH);
    }
    if (digitalRead(buttonBpin) == LOW)
    {
        digitalWrite(ledPin, LOW);
    }
}
```

アップロードした後、左側のボタンを押すと、LEDが点灯します。右側のボタンを押すと、LEDがオフになります。



プログラムの最初の部分は前のレッスンと同じで、まず3つの変数を設定します。そのうちの2つの変数はボタンの入力ピン、もう1つの「ledPin」はLED出力ピンです。「buttonApin」は左側のボタンの入力ピン、「buttonBpin」は右側のボタンの入力ピンです。

「setup」関数は「ledPin」を通常の出力として定義しますが、2つの入力については、次のように「pinMode」を「INPUT\_PULLUP」または「INPUT」に設定して使用します：

```
pinMode(buttonApin, INPUT_PULLUP);
pinMode(buttonBpin, INPUT_PULLUP);
```

INPUT\_PULLUPに設定されたピンは入力として使用されます。入力には何も接続されていない場合、「HIGH」にプルアップします。つまり、入力のデフォルト値は「HIGH」です。

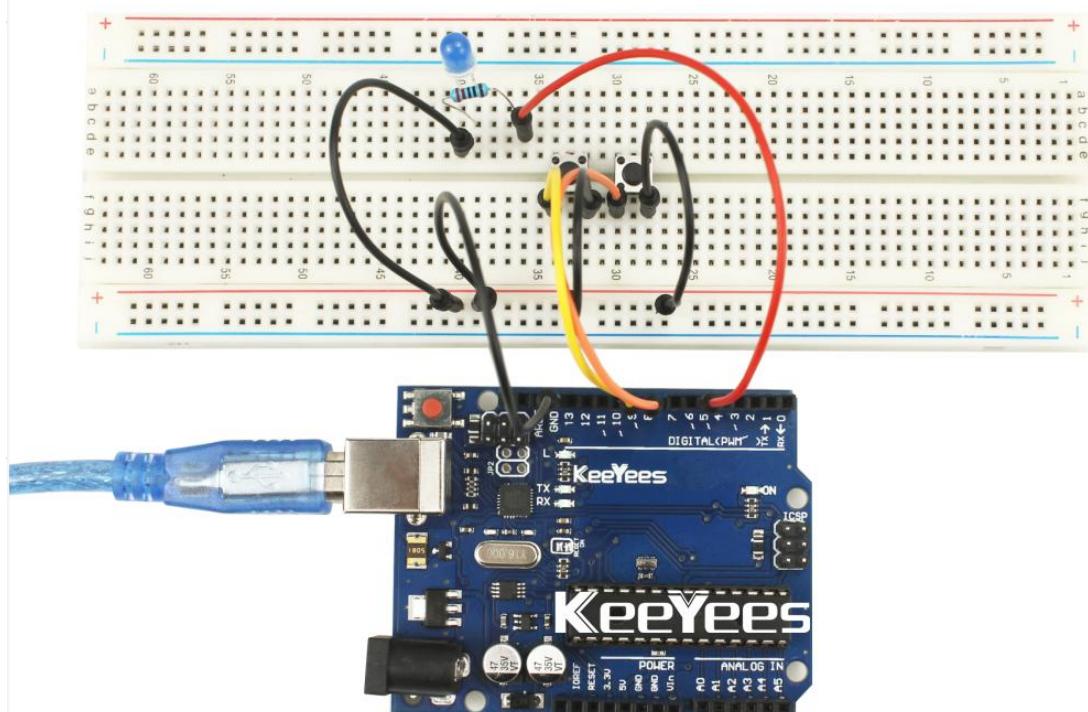
ですので、ボタンがGNDに接続しなければなりません。ボタンを押した場合、ボタンは低レベルのGNDに接続、「LOW」になります。これがボタンを押すという過程です。今回は「loop」関数で実現します。

```
void loop()
{
if (digitalRead(buttonApin) == LOW)
{
digitalWrite(ledPin, HIGH);
}
if (digitalRead(buttonBpin) == LOW)
{
digitalWrite(ledPin, LOW);
}
}
```

「loop」関数には2つの「if」があります。1つのボタンに1つの「if」があります。ボタンは「digitalRead」でピンの電圧をHIGHかLOWに設定します。ボタンを押すと、そのボタンの入力はLOWになります。ボタンAはLOWの場合、「ledPin」の「digitalWrite」はLEDを点灯させます。同様に、ボタンBを押すと、「ledPin」にLOWと設定します。



例





## Lesson 5 電子ブザーとポテンショメーター

### 概要

このレッスンでは、ブザー音が鳴る方法、そして、ポテンショメータを使用して音量を調整する方法を学びます。

### 必要なもの

1 x Arduino UNOR3 開発ボード

1 x 電子ブザー

1 x ポテンショメーター

3 x ジャンパーウイヤー メス-オス

### 部品の説明

#### 電子ブザー

ブザーは直流を使用し、集積回路を備えています。コンピューター、プリンター、コピー機、アラーム、電子玩具、自動車用電子機器、電話、タイマー、その他の音声機器の電子音源として使用されています。

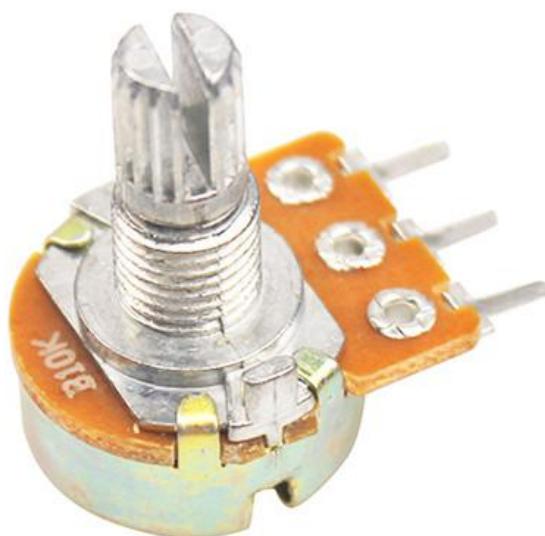
ブザーには電子ブザーと圧電スピーカーがあります。裏に緑の回路基板見えるものが圧電スピーカーです。裏が黒いものが電子ブザーです。

電子ブザーと圧電スピーカーの違いですが、電子ブザーは内部に発振回路が入っていて、決まった直流電圧を掛けるだけで決まった音程のブザー音が鳴ります。圧電スピーカーは普通のスピーカーと同じで、外部から与えた音声信号に従って発音します。従って、音程を変えたり、メロディーを流したりする事も可能です。



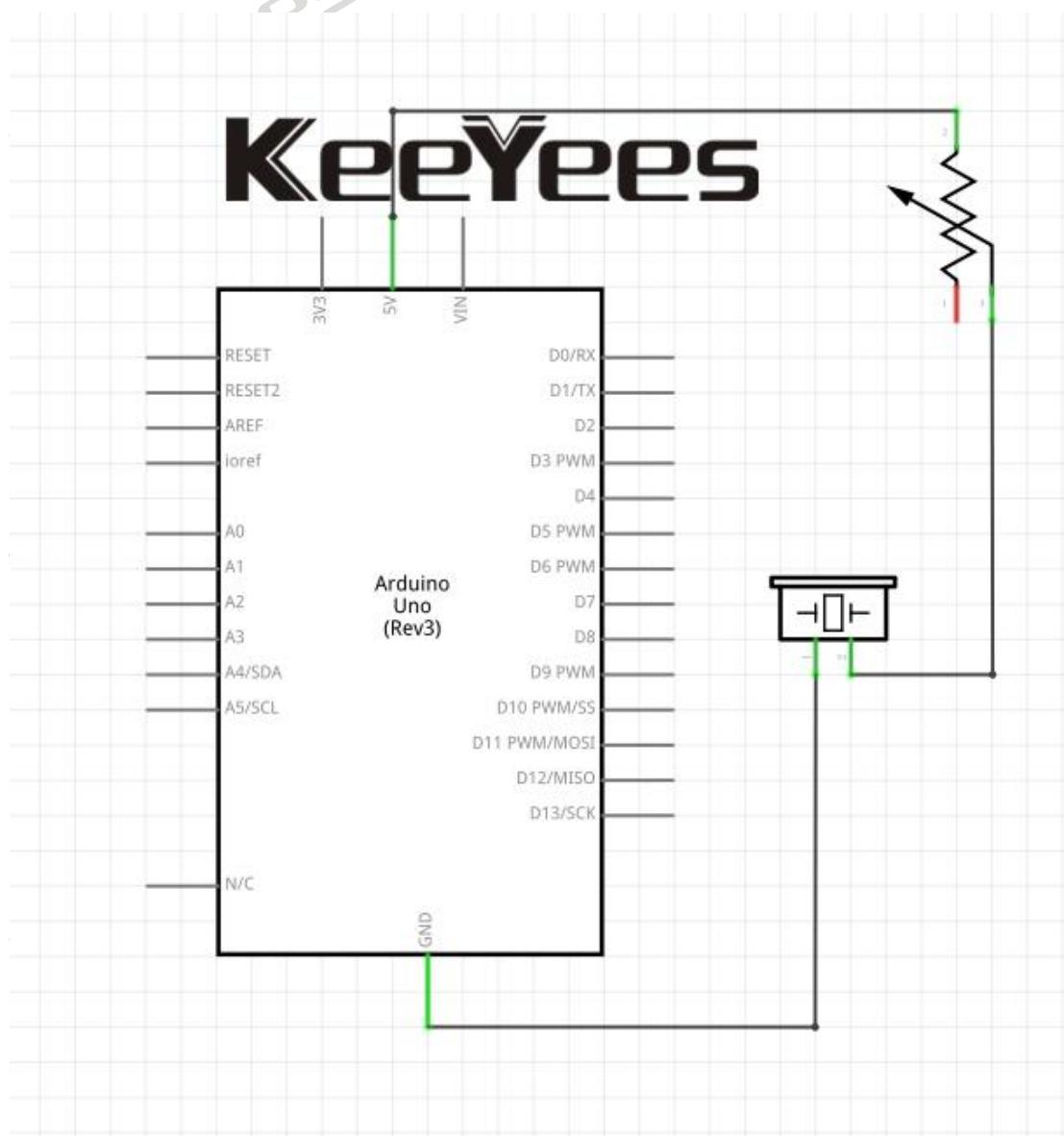
### ポテンショメータ

ポテンショメータは機械的な変位量に比例した電圧出力を得る変位センサです。抵抗体とワイパー（ブラシ）を基本構成として、抵抗体とワイパーの相対的変位量を電圧出力に精度良く変換します。実際には、抵抗体の両端に電圧を加えておき、ワイパーを動かしてその変位量を抵抗体の片側端子とワイパー間の電圧で測定します。ポテンショメータは3ピンがありますが、2ピンで使用も可能です。2ピンで使用する時、可変抵抗器としてと見られます。今回は可変抵抗器として使用します。



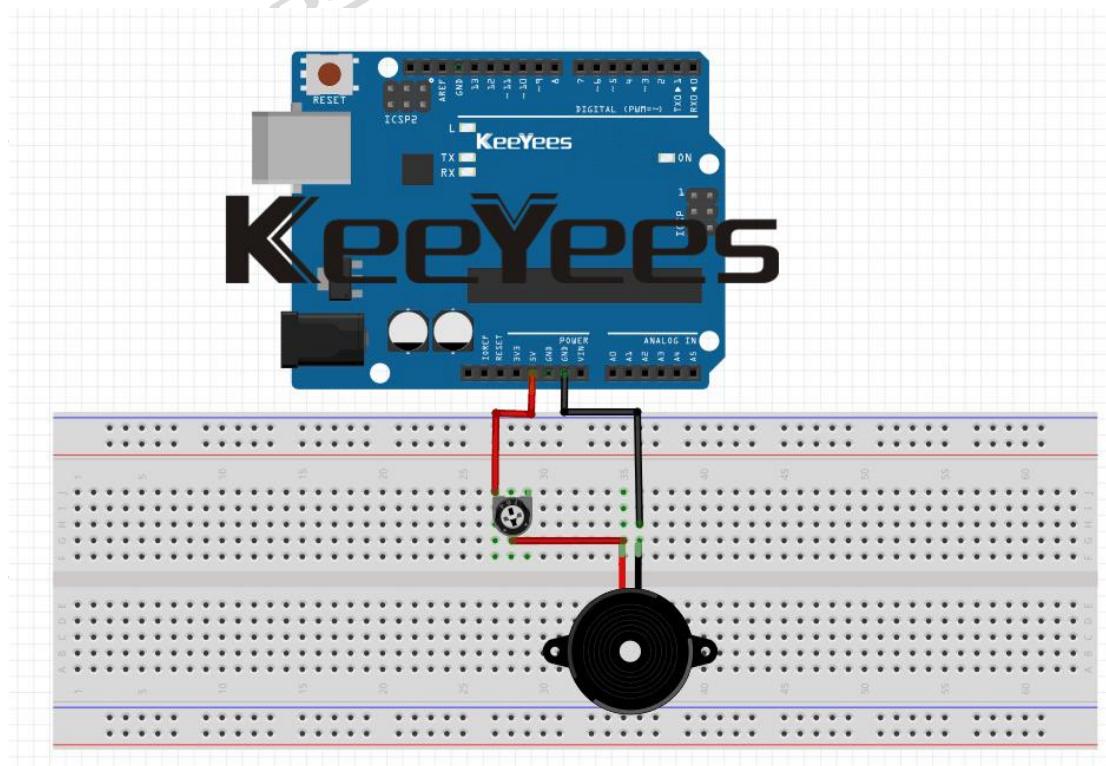


接続図





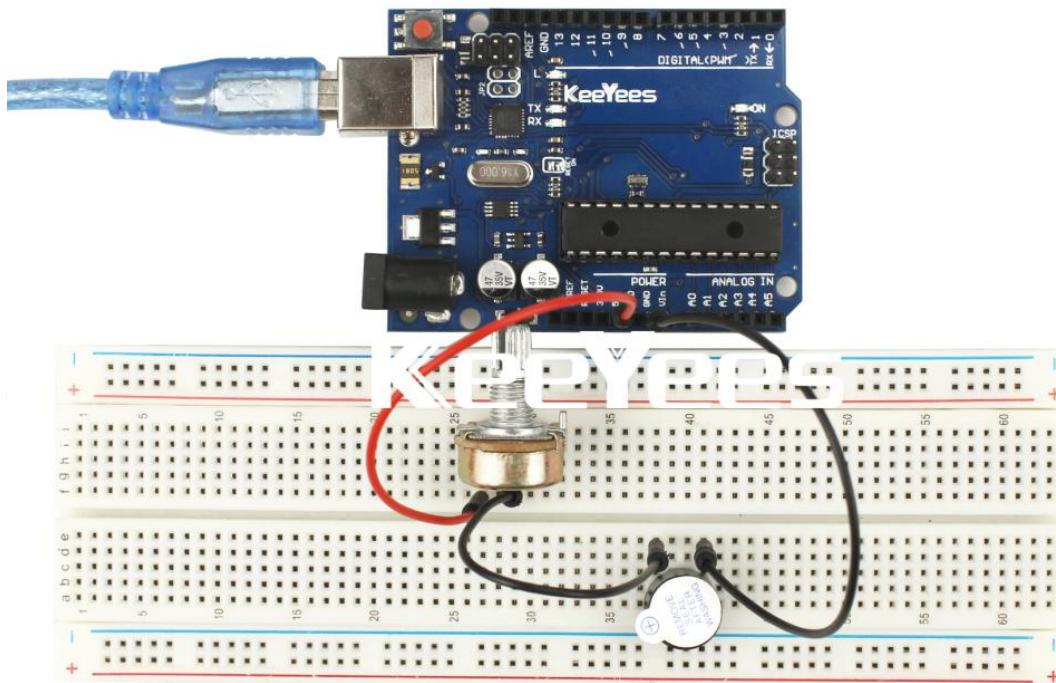
配線図



ポテンショメーターを左から右に回すと、電子ブザーの音量を変更できます。



例





## Lesson 6 圧電スピーカー

### 概要

このレッスンでは、圧電スピーカーの使用方法を学びます。 今回の実験は8つの音を出して、それぞれが0.5秒連続して鳴り続けます：Do (523Hz) 、Re (587Hz) 、Mi (659Hz) 、Fa (698Hz) 、So (784Hz) 、La (880Hz) 、Si (988Hz) 、Do (1047Hz) です。

### 必要なもの

1 x Arduino UNOR3 開発ボード

1 x 圧電スピーカー

2 x ジャンパーウイヤー メス一オス

### 部品説明

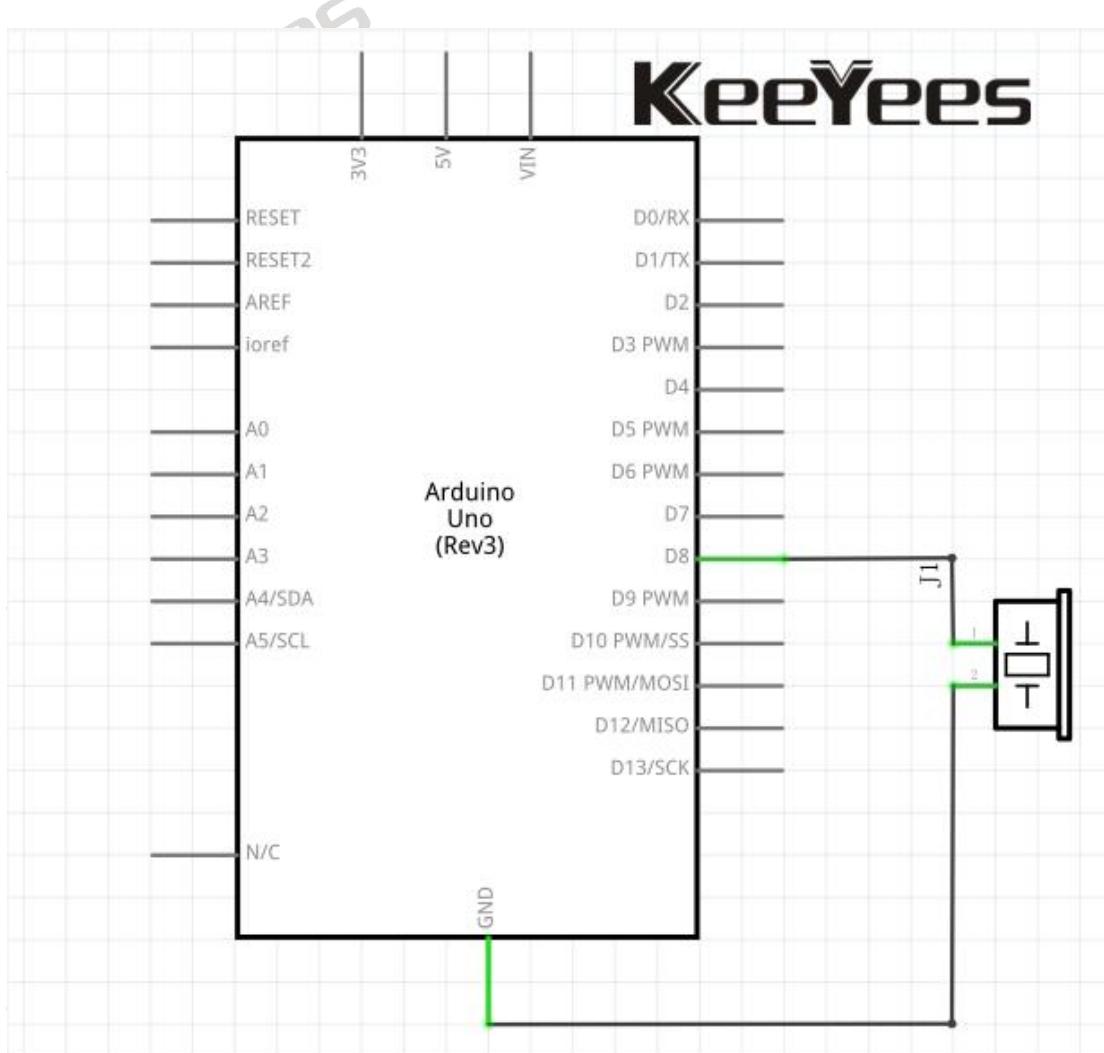
#### 圧電スピーカー

圧電スピーカーは電極に信号電圧を加えることにより圧電体が歪み、その振動を発音体に伝え音（空気の振動）として聞くものです。振動の周波数（電圧）を変えれば、音も変わります。例えば、ドの音は523Hz、レの音は587Hz、ミの音は659Hzです。`analogWrite()` 関数のパルス出力は固定されましたため、NO R3 ボードの `analogWrite()` 関数を使用しないようご注意ください。



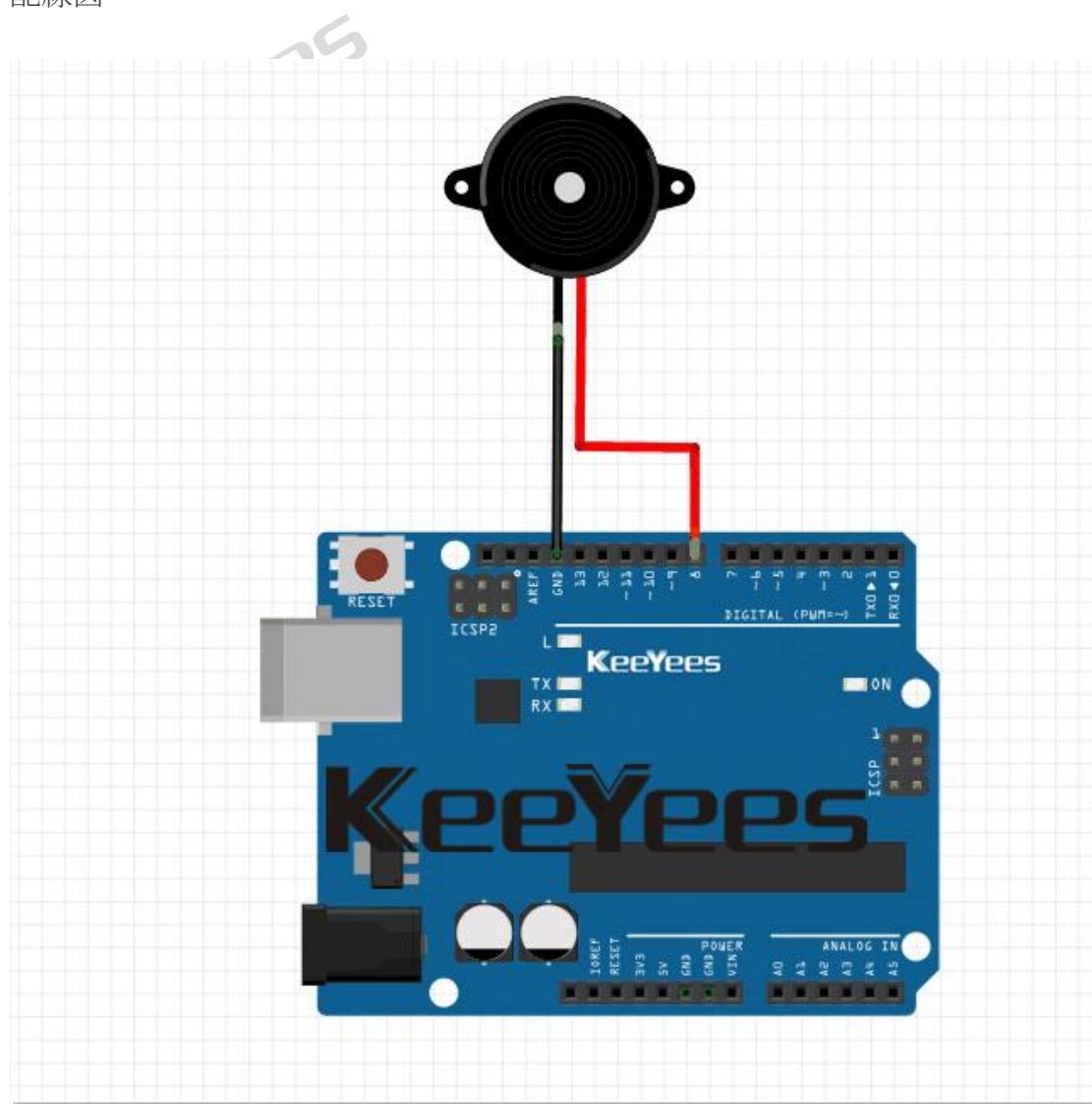


接続図





配線図



圧電スピーカを開発ボードに接続します。「+」の付いた圧電スピーカの片方の足は、開発ボードの8番ピンと接続、もう1つの足をGNDと繋ぎます。



## コード

Aduino IDE を開いて ⇒ 新しいファイルを作成します ⇒ 以下のコードをコピーし、作成したファイルにある元のコードを置き換えてペーストします ⇒ プログラムがきちんとかけているかチェックするコンパイルを行います ⇒ コードをファイルに保管できます ⇒ Arduino に書き込むアップロードを行います。

手順はレッスン1と同じです。お忘れの場合、レッスン1をご確認ください。

```
#define NOTE_C5 523
#define NOTE_D5 587
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_G5 784
#define NOTE_A5 880
#define NOTE_B5 988
#define NOTE_C6 1047
// notes in the melody:
int melody[] = {
    NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_G5, NOTE_A5, NOTE_B5,
    NOTE_C6};
int duration = 500; // 500 miliseconds

void setup() {
}

void loop() {
    for (int thisNote = 0; thisNote < 8; thisNote++) {
        // pin8 output the voice, every scale is 0.5 sencond
        tone(8, melody[thisNote], duration);

        // Output the voice after several minutes
        delay(1000);
    }

    // restart after two seconds
    delay(2000);
}
```



## コード解説

```
#define NOTE_C5 523
#define NOTE_D5 587
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_G5 784
#define NOTE_A5 880
#define NOTE_B5 988
#define NOTE_C6 1047
```

電圧の振動を変えることにより、空気の振動数が変わり、音の違いが生まれます。上記のコードは`#Define` 文で 8 つの振動周波数を定義します。

```
int melody[] = {
    NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_G5, NOTE_A5, NOTE_B5,
    NOTE_C6};
```

上記のコードは、8 つの振動周波数を順番に並べて、後で `for loop` を利用してデータを読み取ります。

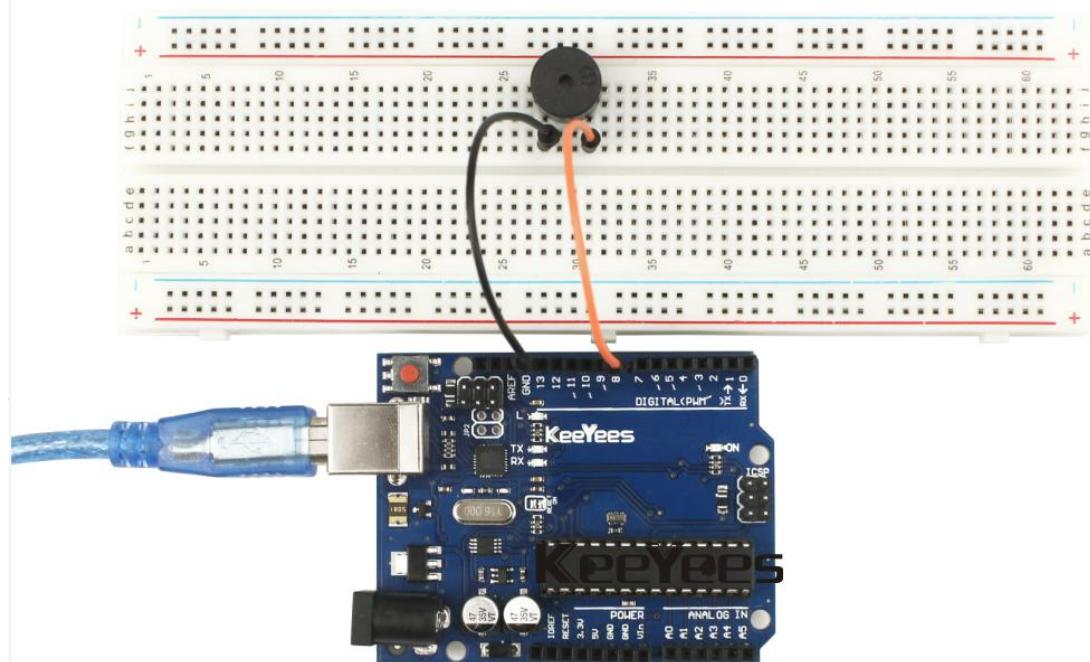
```
void setup() {
}
void loop() {
    for (int thisNote = 0; thisNote < 8; thisNote++) {
        // pin8 output the voice, every scale is 0.5 sencond
        tone(8, melody[thisNote], duration);

        // Output the voice after several minutes
        delay(1000);
    }
    // restart after two seconds
    delay(2000);
}
```

今回は `tone` 関数を使って音を鳴らしています。`tone` 関数は `tone(ピン番号, 出力する周波数, 出力時間)` のように記述します。周波数を順番に変えていくことにより、ドレミファソラシドの音を鳴らしています。



配線図





## Lesson 7 8つのLEDと74HC595

### 概要

このレッスンでは、74HC595を使って開発ボードの出力ピンを増やし、8つのLEDを制御し、流れる点滅の形で表示する方法を学びます。

8つのLEDを点灯させるにはデジタル出力8本が必要です。開発ボードに直接接続できますが、ピンが足りなくなり、センサーヤボタンなど他の電子部品を接続できなくなり、ピン不足になってしまいます。

そこで、「74HC595」のICを使うことでArduino開発ボードのピンを節約します。ICには16本のピンがあり、今回は8つの出力と3つの入力ピンを使用しています。

このチップにより、LEDの駆動は約1秒8,000,000回ではなく、500,000回です。少し遅くなりますが、その違いは人の目で見えません。

### 必要なもの

1 x Arduino UNOR3 開発ボード

1 x 830 タイポイントブレッドボード

8 x LED

8 x 220Ω 抵抗器

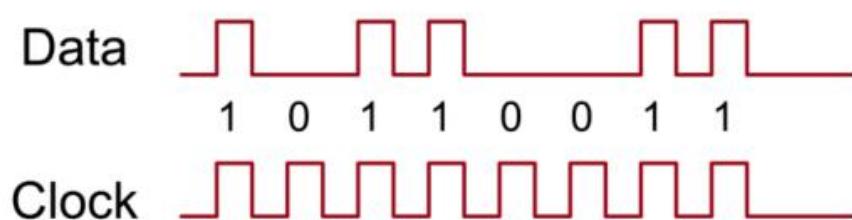
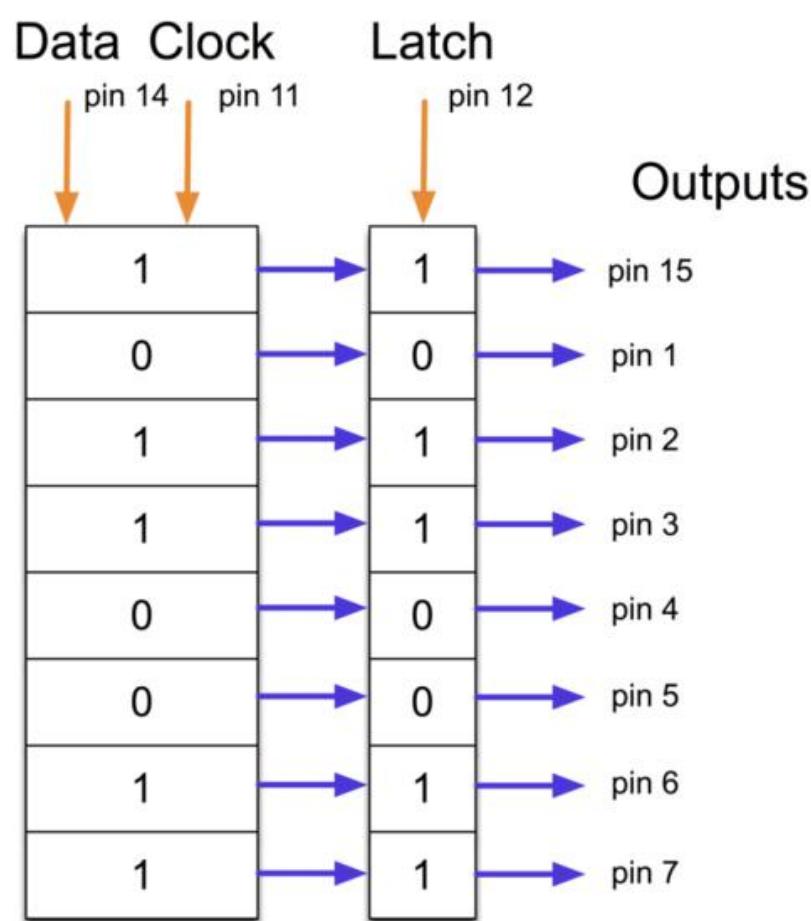
1 x 74hc595 IC

19 x ジャンパーウイヤー オス-オス

### 部品説明

#### 74HC595

ストレージレジスタ付きの8ビットシフトレジスタである74HC595Nには、ストレージレジスタ側にも8つのフリップフロップがあります。各ビットは1または0です。チップの「Data」ピンと「Clock」ピンを使って値をオンするかオフに設定します。



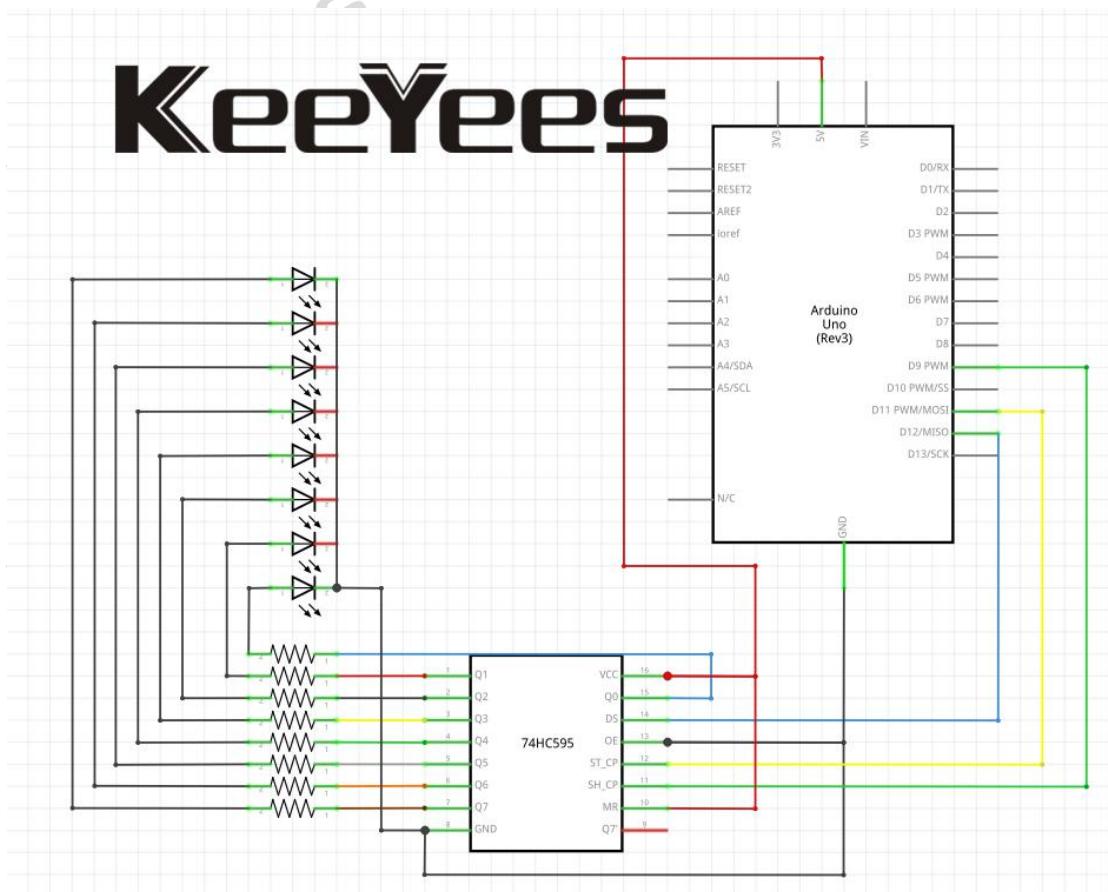


「クロック」ピンは8パルスを受信する必要があります。各パルスで、「データ」ピンがHighの場合、「1」がシフトレジスタに押されます。Lowの場合、「0」となります。

8個のパルスをすべて受信した後、「Latch」ピンを有効にすると、8個の値がストレージレジスタにコピーされます。そうしないと、データがシフトレジスタにロードされたと、LEDが誤って点滅します。

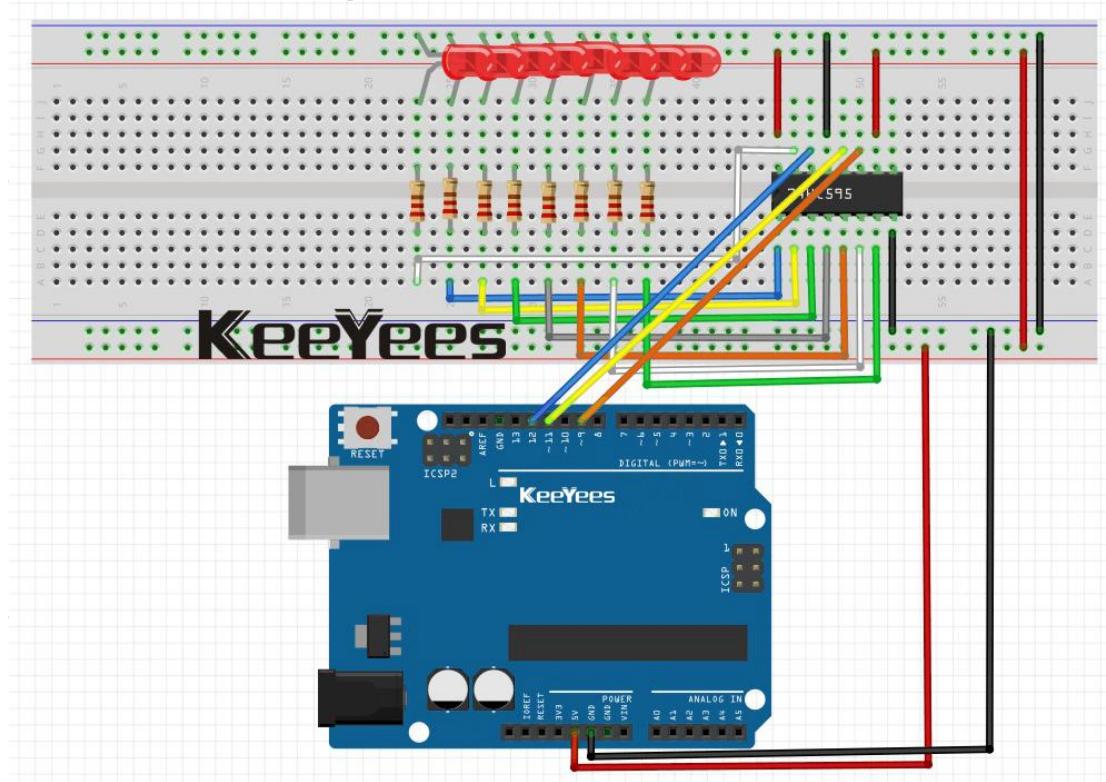
出力ピンの出力制御を行うピン「OE」があります。OEピンにLOWを入力すると、出力が有効となり、HIGHを入力すると、出力が無効となります。PWMに対応する開発ボードのピンに接続して、「analogWrite」関数でLEDの明るさを制御します。

## 接続図





## 配線図





8つのLEDと8つの抵抗を接続しなければなりませんため、接続にはやや難しいです。

74HC595は主な出力であり、ジャンパー線をきれいに配置できるよう、位置を注意しなければなりません。ブレッドボードには数字が付いています。今回はブレッドボードの45～52のところに74HC595を配置します。74HC595の左右のピンがブレッドボードの中央の溝にまたがっており、表面の小さなU字型の開口部がブレッドボードの上部(数字1のところ)に向きます。

開発ボードの12ピンを、チップの#14ピンに挿入します。

開発ボードの11ピンを、チップの#12ピンに挿入します。

開発ボードの9ピンを、チップの#11ピンに挿入します。

チップの出力ピンは全てチップの左側にあります。接続を易くするために、抵抗器をLEDの下に配置したほうがいいです。多くの抵抗器を使っていましため、抵抗器のリード線が接触しないよう、ご注意ください。開発ボードを電源に接続する前に、もう一度チェックしてください。

次に、ブレッドボードにLEDを配置します。リードが長いほうはチップに向きます。

接続図と配線図を見るとわかると思いますが、チップの出力ピン0とピン8のGNDはよく間違われます。出力ピン0がチップの右側にあり、他の出力ピンと違う側です。GNDとVCCもよく間違われて、LEDを損傷します。接続が間違いないと、LEDが順番に1つずつ点灯し、最後に、8つのLEDが一緒に点灯します。その後、全てが消えて繰り返す動作となっています。

## コード

Arduino IDEを開いて ⇒ 新しいファイルを作成します ⇒ 以下のコードをコピーし、作成したファイルにある元のコードを置き換えてペーストします ⇒ プログラムがきちんとかけているかチェックするコンパイルを行います ⇒ コードをファイルに保管できます ⇒ Arduinoに書き込むアップロードを行います。

手順はレッスン1と同じです。お忘れの場合、レッスン1をご確認ください。

```
int tDelay = 100;  
int latchPin = 11;      // (11) ST_CP [RCK] on 74HC595  
int clockPin = 9;       // (9) SH_CP [SCK] on 74HC595  
int dataPin = 12;      // (12) DS [S1] on 74HC595  
byte leds = 0;  
void updateShiftRegister()
```



```
        digitalWrite(latchPin, LOW);
        shiftOut(dataPin, clockPin, LSBFIRST, leds);
        digitalWrite(latchPin, HIGH);
    }

void setup()
{
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
}

void loop()
{
    leds = 0;
    updateShiftRegister();
    delay(tDelay);
    for (int i = 0; i < 8; i++)
    {
        bitSet(leds, i);
        updateShiftRegister();
        delay(tDelay);
    }
}
```

### コード解説

まず、使用しようとする 3 つのピンを定義します。開発ボードのデジタル出力ピンを 74HC595 の LATCH、CLOCK、DATA ピンに接続します。

```
int latchPin = 11;
int clockPin = 9;
int dataPin = 12;
```

次に、「leds」という変数を定義します。これは、LED が現在オンまたはオフになっているかどちらのモードを保存します。byte 型のデータは、8 ビットの数を使用します。各ビットともオンまたはオフにすることができます。8 つの LED のどれがオンまたはオフにするかを制御しやすいです。

```
byte leds = 0;
```

setup() 関数は、デジタル出力として使用する 3 つのピンを設定します。



```
void setup()
{
pinMode(latchPin, OUTPUT);
pinMode(dataPin, OUTPUT);
pinMode(clockPin, OUTPUT);
}
```

次は「loop」関数です、まず、変数「leds」に 0 を設定して LED をオフにします。次に、「updateShiftRegister」を呼び出します。「leds」モードをシフトレジスタに送信し、すべての LED をオフにします。「updateShiftRegister」の動作原理については後で説明します。

「loop」関数は 0.5 秒間を停止し、「for」ループと変数「i」を使用して 0 から 7 までカウントをします。「bitSet」関数で変数「leds」の LED のビットを制御するため、設定します。「updateShiftRegister」も呼び出して、変数「leds」の内容が反映できるよう、LED を更新します。「i」が増やす、そして次の LED が点灯するまでに 0.5 秒の遅延があります。

```
void loop()
{
leds = 0;
updateShiftRegister();
delay(500);
for (int i = 0; i < 8; i++)
{
bitSet(leds, i);
updateShiftRegister();
delay(500);
}
}
```

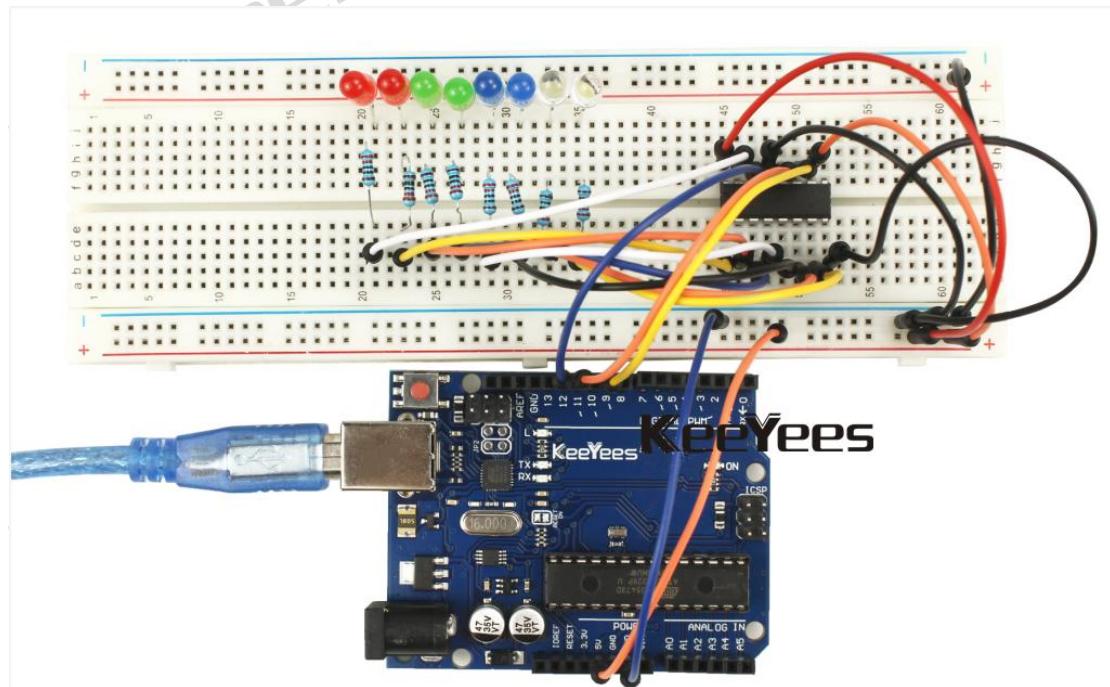
「updateShiftRegister」関数 で、まず、latchPin を LOW に設定し、「shiftOut」関数を呼び出します。「shiftOut」関数は引数として 4 つの値を持ちます。「dataPin」は各ビットを出力するピン、「clockPin」はクロックを出力するピンです。3 番目の値は MSBFIRST または LSBFIRST を指定します。MSBFIRST は最上位ビットから送ること、LSBFIRST は最下位ビットから送ることを示します。今回は「LSBFIRST」に設定します。最後は送信したいデータ (byte) を設定します。今回は「leds」に設定します。そして、latchPin を HIGH に設定します。



```
void updateShiftRegister()
{
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, leds);
    digitalWrite(latchPin, HIGH);
}
```

1つのLEDをオフにしたい場合は、変数「`leds`」でArduinoの「`bitClear`」関数を呼び出して「`leds`」ビットを0に設定します、そして、「`updateShiftRegister`」を呼び出して追跡して更新します。

例





## Lesson 8 シリアルモニター

### 概要

このレッスンでは、レッスン7に基づいて、Arduinoシリアルモニターでコンピュータを通してLEDを制御することを学びます。シリアルモニタとはArduinoとコンピュータや他のデバイスとの通信に使用され、データの送受信やプログラムのデバッグなどに使用されます。例えば、コンピューターからコマンドを送信してLEDの点滅を制御します。このレッスンでは、レッスン7と同じ部品とほぼ同じのブレッドボードレイアウトを使用します。

### コード

Arduino IDEを開いて ⇒ 新しいファイルを作成します ⇒ 以下のコードをコピーし、作成したファイルにある元のコードを置き換えてペーストします ⇒ プログラムがきちんとかけているかチェックするコンパイルを行います ⇒ コードをファイルに保管できます ⇒ Arduinoに書き込むアップロードを行います。

手順はレッスン1と同じです。お忘れの場合、レッスン1をご確認ください。

```
int latchPin = 11;  
int clockPin = 9;  
int dataPin = 12;  
  
byte leds = 0;  
void updateShiftRegister()  
{  
    digitalWrite(latchPin, LOW);  
    shiftOut(dataPin, clockPin, LSBFIRST, leds);  
    digitalWrite(latchPin, HIGH);  
}  
void setup()  
{  
    pinMode(latchPin, OUTPUT);  
    pinMode(dataPin, OUTPUT);  
    pinMode(clockPin, OUTPUT);  
    updateShiftRegister();  
    Serial.begin(9600);  
    while (! Serial); // Wait until Serial is ready - Leonardo  
    Serial.println("Enter LED Number 0 to 7 or 'x' to clear");  
}
```



```
void loop()
{
    if (Serial.available())
    {
        char ch = Serial.read();
        if (ch >= '0' && ch <= '7')
        {
            int led = ch - '0';
            bitSet(leds, led);
            updateShiftRegister();
            Serial.print("Turned on LED ");
            Serial.println(led);
        }
        if (ch == 'x')
        {
            leds = 0;
            updateShiftRegister();
            Serial.println("Cleared");
        }
    }
}
```

### 手順

- 1) コードを開発ボードに書き込んだ後、以下の赤い枠に囲まれたボタンをクリックします。



The screenshot shows the Arduino IDE interface with the sketch titled 'Eight\_LED\_with\_74HC595\_Flash\_LED'. The code is displayed in the editor, and the status bar at the bottom right indicates 'Arduino/Genuino Uno on COM9'. A red box highlights the 'Upload' button in the toolbar.

```
1 int tDelay = 100;
2 int latchPin = 11;      // (11) ST_CP [RCK] on 74HC595
3 int clockPin = 9;       // (9) SH_CP [SCK] on 74HC595
4 int dataPin = 12;       // (12) DS [SI] on 74HC595
5
6 byte leds = 0;
7
8 void updateShiftRegister()
9 {
10    digitalWrite(latchPin, LOW);
11    shiftOut(dataPin, clockPin, LSBFIRST, leds);
12    digitalWrite(latchPin, HIGH);
13 }
14
15 void setup()
16 {
17    pinMode(latchPin, OUTPUT);
18    pinMode(dataPin, OUTPUT);
19    pinMode(clockPin, OUTPUT);
20 }
```

2) シリアルポートのボーレートをコードに従って以下のように設置します。

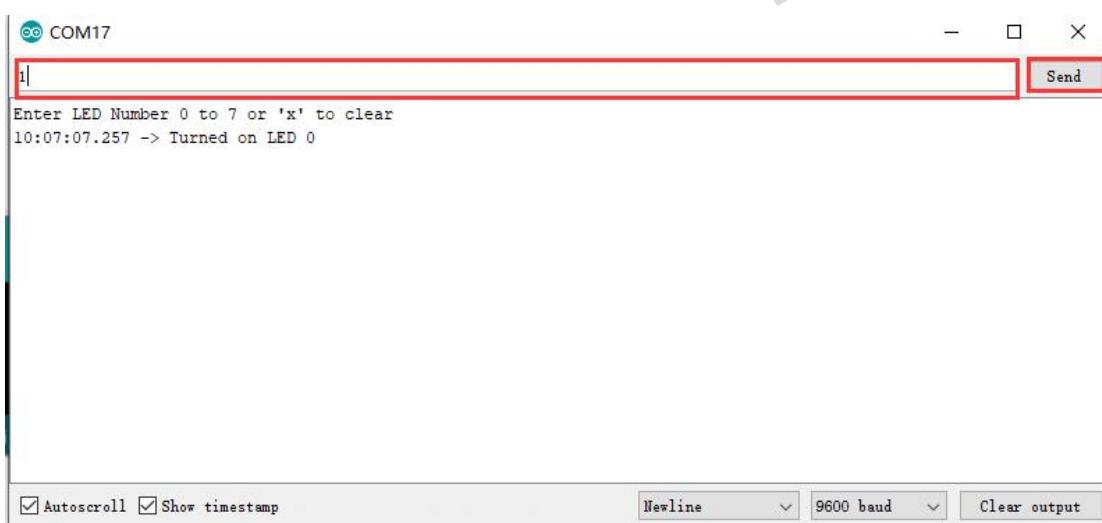
```
void setup()
{
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
    updateShiftRegister();
    Serial.begin(9600);
    while (! Serial); // Wait untilSerial is ready - Leonardo
    Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
}
```

The screenshot shows the Arduino Serial Monitor window. The port is set to 'COM17'. The baud rate is set to '9600 baud', which is highlighted with a red box. The message 'Enter LED Number 0 to 7 or 'x' to clear' is displayed in the text area. The bottom status bar shows 'Autoscroll' and 'Show timestamp' checked.



このウィンドウはシリアルモニターと呼ばれ、Arduino IDE ソフトウェアの一部分です。コンピューターから開発ボードに（USB 経由で）メッセージを送信し、開発ボードからメッセージを受信できます。

Arduino からの「Enter LED Number 0 to 7 or 'x' to clear」というメッセージがシリアルモニターから見れます。「x」を送信すると、全ての LED をオフにします。0 から 7 までの数字を送信すると、相応の LED が点灯します。「0」は下部の LED、「1」は次の LED、「7」は上部の LED です。x、0、3、5 を 1 つずつ送信してみてください。もし、LED が全てオフになっていると、「x」を送信しても、無効です。数字を送信すると、相応の LED が点灯するはずです。そして、開発ボードからの確認メッセージも受信します。



「x」をもう一度送信して全ての LED をオフにします。

## コード解説

「`setup`」関数には最後の 3 行のコードが初めて見ましたかもしれません。

```
void setup()
{
pinMode(latchPin, OUTPUT);
pinMode(dataPin, OUTPUT);
pinMode(clockPin, OUTPUT);
updateShiftRegister();
Serial.begin(9600);
while (! Serial); // Wait until Serial is ready - Leonardo
Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
}
```



まず、コマンド 「Serial.begin (9600)」 があります。これでシリアル通信を開始します。値の 9600 は「ボーレート」と呼ばれます。ボーレートは bps で設定します。シリアルモニターの bps と Serial.begin() のパラメータの bps は同じ値にしないと正しくデータが送信されないのでシリアルモニタ画面右下の部分からパラメータの値と合わせてください。今回は 9600 に設定します。

「while」で始まる行は、シリアルポートがきちんと接続されているかどうかの判断をするのに使われます。

最後の 1 行はシリアルモニターの上部に表示されるメッセージです。

「loop」関数は、すべてのアクションが行われる場所です。

```
void loop()
{
if (Serial.available())
{
char ch = Serial.read();
if (ch >= '0' && ch <= '7')
{
int led = ch - '0';
bitSet(leds, led);
updateShiftRegister();
Serial.print("Turned on LED ");
Serial.println(led);
}
if (ch == 'x')
{
leds = 0;
updateShiftRegister();
Serial.println("Cleared");
}
}
}
```

ループ内で発生することが「if」ステートメントに含まれます。Arduino の関数 「Serial.available ()」 の呼び出しが 「true」 でない限り、何も起こりません。データが開発ボードに送信され、処理の準備ができる場合、Serial.available () は 「true」 に戻します。着信のメッセージはバッファーに保存され、バッファーが空でない場合、Serial.available () は 「true」 に戻します。

メッセージを受信した場合、次の行のコードに移動します。

```
char ch = Serial.read();
```

上記のコードは、バッファから次の文字が読み取られ、削除されます。そして、変数 「ch」 に割り当てます。変数 「ch」 は 「char」 型で、「character」とい



う意味です。1つの文字が含まれています。シリアルモニターの上部にあるヒントの指示通り行う場合、この文字は0~7のどちらの数字か文字「x」となります。

次の行の「`if`」ステートメントは、「`ch`」が文字「0」以上かつ文字「7」以下であるかどうかを確認することにより、それが単一の数字であるかどうかをチェックします。

各文字は、ASCII値と呼ばれる唯一の番号で示されます。ということで、「`<=`」と「`>=`」で文字を比べる時、ASCII値を比べています。

問題なければ、次の行に進みます：

```
int led = ch - '0';
```

上記のコードはターゲットに対して算術演算を行っています。入力した数値から数値「0」を減算します。ということで、「0」と入力すると「0」 - 「0」は0になります。「7」と入力すると「7」 - 「0」は7になります。実際には減算でASCII値を使用しています。点灯するLEDの数がわかっているので、変数「`leds`」にこのビットを設定し、シフトレジスタを更新すればいいです。

```
bitSet(leds, led);
```

```
updateShiftRegister();
```

上記の2行では、確認メッセージです。シリアルモニターに送信します。

```
Serial.print("Turned on LED ");
```

```
Serial.println(led);
```

上記の行は、「`Serial.println`」の代わりに「`Serial.print`」を使用します。その2つの違いは、データを印字した後、「`Serial.println`」は改行します。

「`Serial.print`」は改行しません。メッセージを2部分に分けるため、上の行で「`Serial.print`」を使用します。まずは：「LEDをオンにする」、次、LEDの数です。LEDの数は、テキスト文字列ではなく「`int`」変数に保存されます。「`Serial.print`」は、二重引用符で囲まれたテキスト文字列を使用するか、「`int`」または他の変数を使用できます。

「`if`」ステートメントを処理した後、次の「`if`」ステートメントは「`ch`」変数が文字「x」であるかどうかを確認し、「x」である場合、すべてのLEDをオフにして確認メッセージを送信します。

```
if (ch == 'x')  
{  
    leds = 0;  
    updateShiftRegister();  
    Serial.println("Cleared");  
}
```



## Lesson 9 フォトレジスタ

### 概要

このレッスンでは、アナログ入力で光強度を測定する方法を学びます。 レッスン 7に基づいて、光の強さを利用して、点灯する LED の数を制御します。

### 必要なもの

1 x Arduino UNO R3 開発ボード

1 x 830 タイピントブレッドボード

8 x LED

8 x 220Ω 抵抗器

1 x 1KΩ 抵抗器

1 x 74hc595 IC

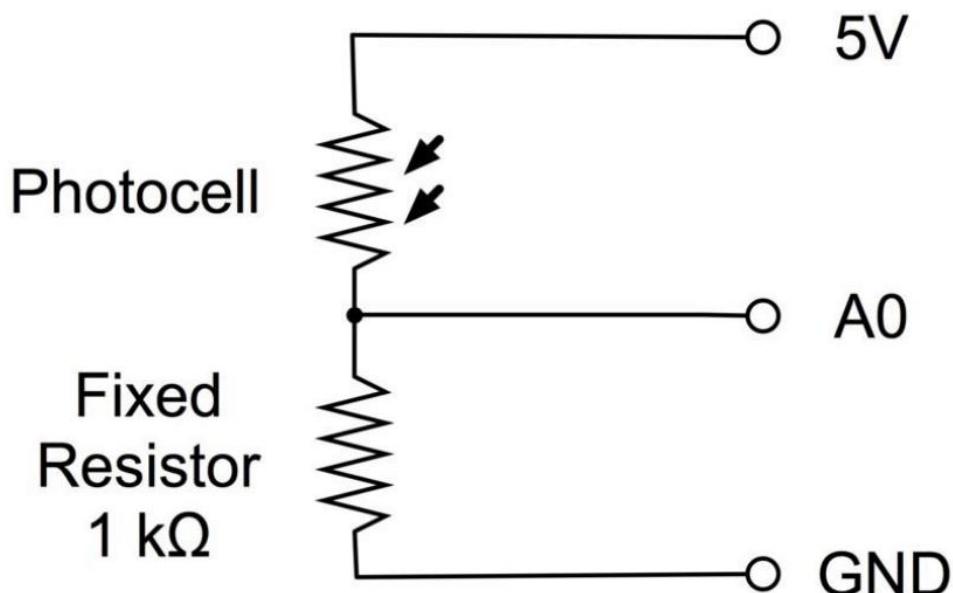
1 x フォトレジスタ

21 x ジャンパーウイヤー オス-オス

### 部品説明

#### フォトレジスタ

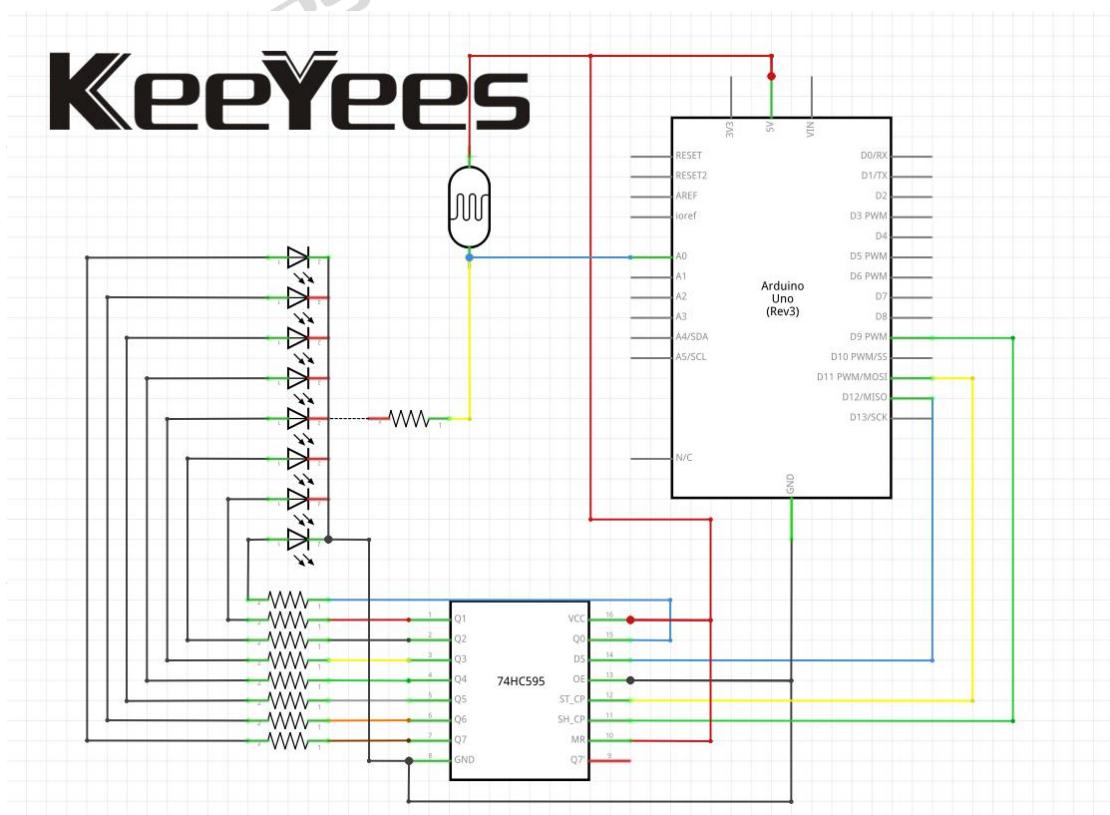
フォトレジスタにはさまざまな種類がありますが、今回使用するものは硫化カドミウム (CdS) を使ったフォトレジスタです。フォトレジスタは入射する光の強度が強いほど抵抗が小さくなる部品です。今回は固定抵抗器と組み合わせて抵抗値の変化を電圧に変換することにより、開発ボードのアナログ入力で測定できる値も変換します。



固定抵抗器とフォトトレジスタを組み合わせて使用します。明るい時、フォトトレジスタの抵抗値は固定抵抗器よりも小さくなります。暗い時、抵抗値は $1\text{ k}\Omega$ の固定抵抗器よりも大きくなります。このレッスンのコードを書き込み、フォトトレジスタを明るい場所に置く時、そして、手で隠す時、LEDの変化をご確認ください。

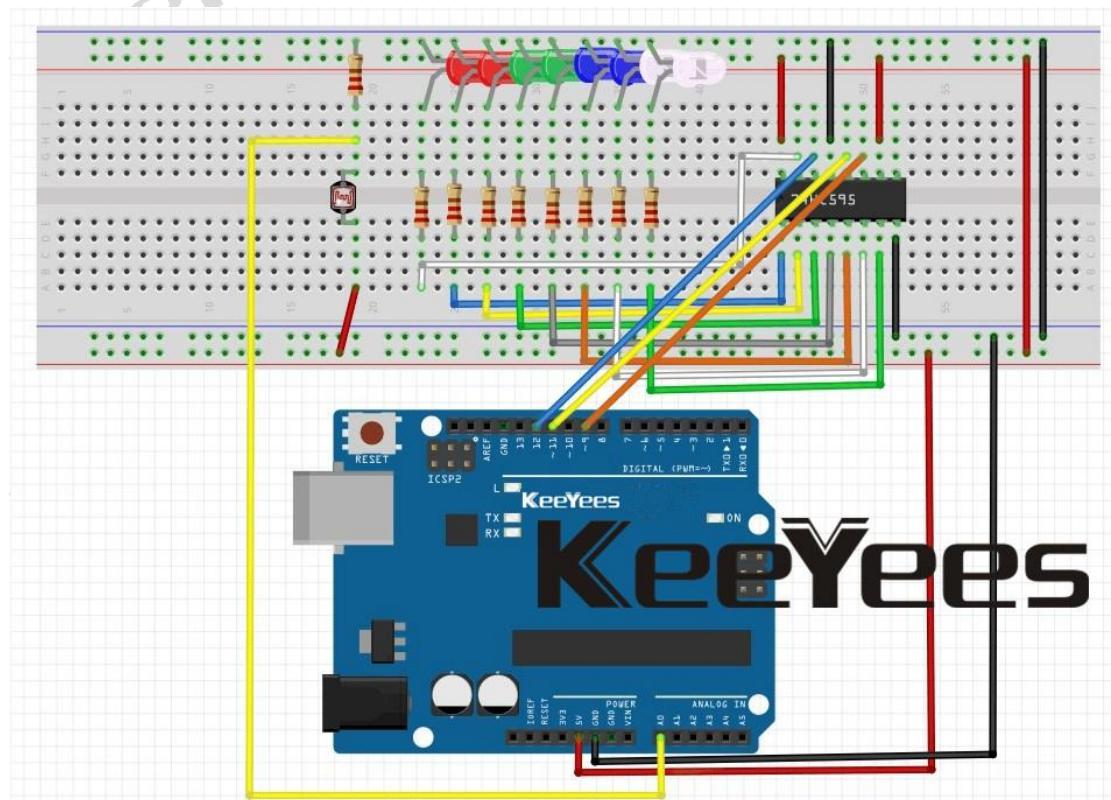


接続図





配線図





## コード

Aduino IDEを開いて ⇒ 新しいファイルを作成します ⇒ 以下のコードをコピーし、作成したファイルにある元のコードを置き換えてペーストします ⇒ プログラムがきちんとかけているかチェックするコンパイルを行います ⇒ コードをファイルに保管できます ⇒ Arduinoに書き込むアップロードを行います。

手順はレッスン1と同じです。お忘れの場合、レッスン1をご確認ください。

```
int lightPin = 0;
int latchPin = 11;
int clockPin = 9;
int dataPin = 12;

int leds = 0;

void setup()
{
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
}

void updateShiftRegister()
{
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, leds);
    digitalWrite(latchPin, HIGH);
}

void loop()
{
    int reading = analogRead(lightPin);
    int numLEDSLit = reading / 57; //1023 / 9 / 2
    if (numLEDSLit > 8) numLEDSLit = 8;
    leds = 0; // no LEDs lit to start
    for (int i = 0; i < numLEDSLit; i++)
    {
        leds = leds + (1 << i); // sets the i'th bit
    }
    updateShiftRegister();
```

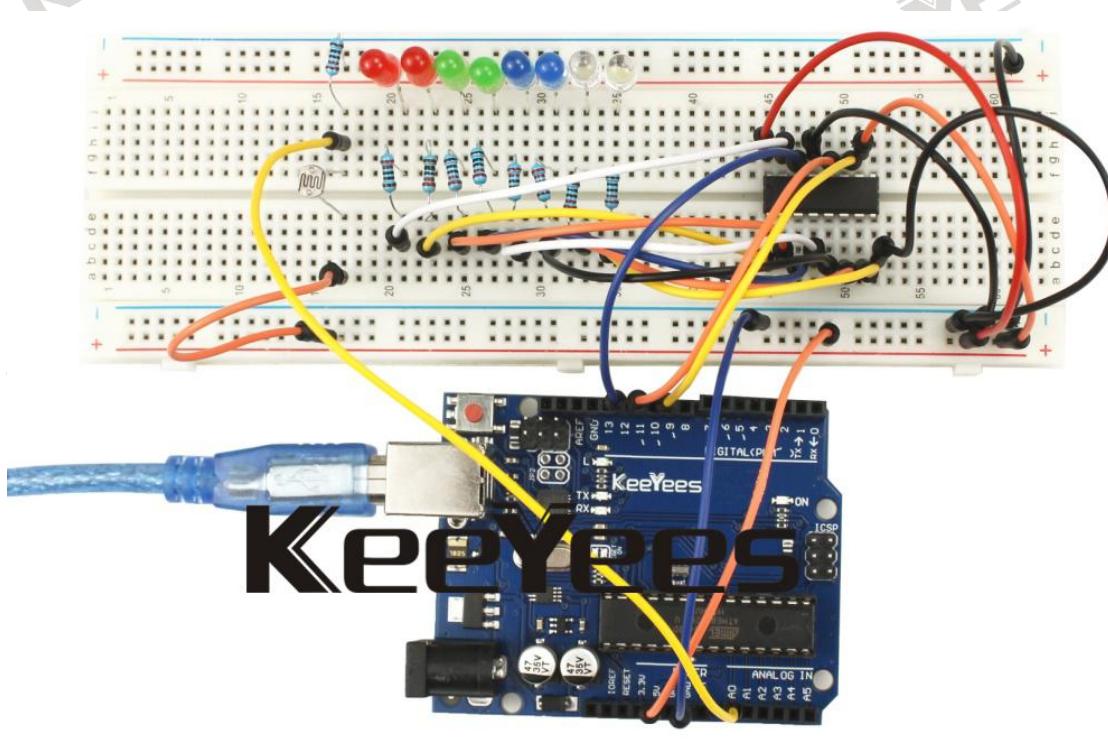


### コード解説

今回のコードではアナログピンを「`lightPin`」に定義します。スケッチに比べて、変更するのは、点灯する LED の数を計算する行です：

```
int numLEDSLit = reading / 57; // all LEDs lit at 1k
```

$1\text{k}\Omega$  の固定抵抗器もあるため、今回は元の値を 114 ではなく 57 で除算します。つまり、データの半分で除算し、LED なしからすべての LED 点灯までの 9 つの領域に分けます。フォトトレジスタの抵抗値が  $1\text{k}\Omega$ （固定抵抗と同じ）の場合、元の値は  $1023/2 = 511$  になり、すべての LED が点灯します、「`numLEDSLit`」の値は 8 になります。





## Lesson 10 7セグメント LED 表示器

### 概要

このレッスンでは、デジタルセグメント LED 表示器の使用方法を学びます。この LED 表示器には多くのピンがあります。開発ボードに直接に接続すると、IO ピンが足りなくなります。今回も 74HC595 を使って開発ボードのピンを節約します。

### 必要なもの

1 x Arduino UNOR3 開発ボード

1 x 7セグメント LED 表示器

1 x 1KΩ 抵抗器

1 x 74hc595 IC

22 x ジャンパーウイヤー オス一オス

### 部品説明

#### 7セグメント LED 表示器

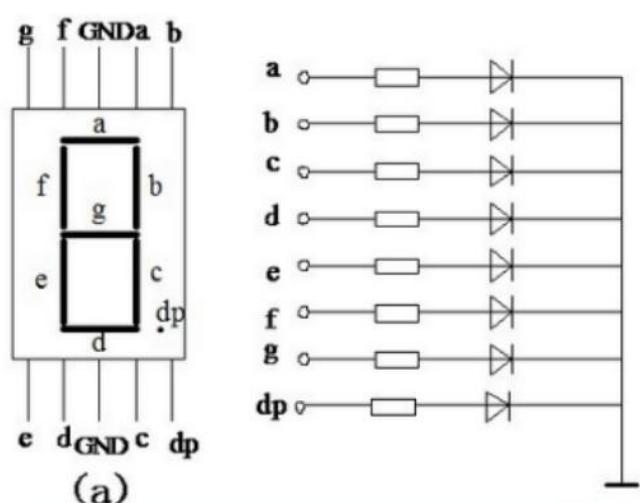
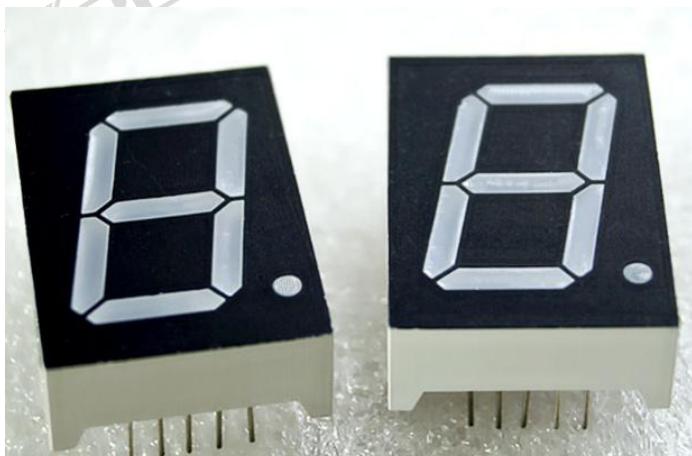
7セグメント LED 表示器は、数字が表現できるように LED を丸から長方形に伸ばした7個と、小数点表示分1個を含む計8個のLEDを組み合わせて1個のモジュールにしたもので。そして、a、b、c、d、e、f、g および dp とセグメントに名前が付いています。これを組み合わせて光らせ、数字を表現します。たとえば、「2」の数字を表現する場合、a、b、g、e、d が点灯し、f、c、dp が消灯します。7セグメント LED 表示器には大きさや表示色が異なる様々な種類があります。LED が 15 個入ってる 14セグメントディスプレイや、17 個入ってる 16セグメントディスプレイもあります。1つの LED の電圧は約 1.8V で、電流は 30mA 以下です。

LED のアノード (+側) が 1つにまとまっているのは「アノードコモン」と呼ばれていて、電源プラスに接続します。LED のカソード (-側) が 1つにまとまっているのは「カソードコモン」と呼ばれていて、電源マイナスに接続します。

例えば「カソードコモン」の場合、共通のカソード (-側) をグランドに接続し、アノード側を抵抗を経由して個々のマイコンの端子に接続すると、マイコン端子の出力に応じて LED の表示を変えることができるようになります。表示

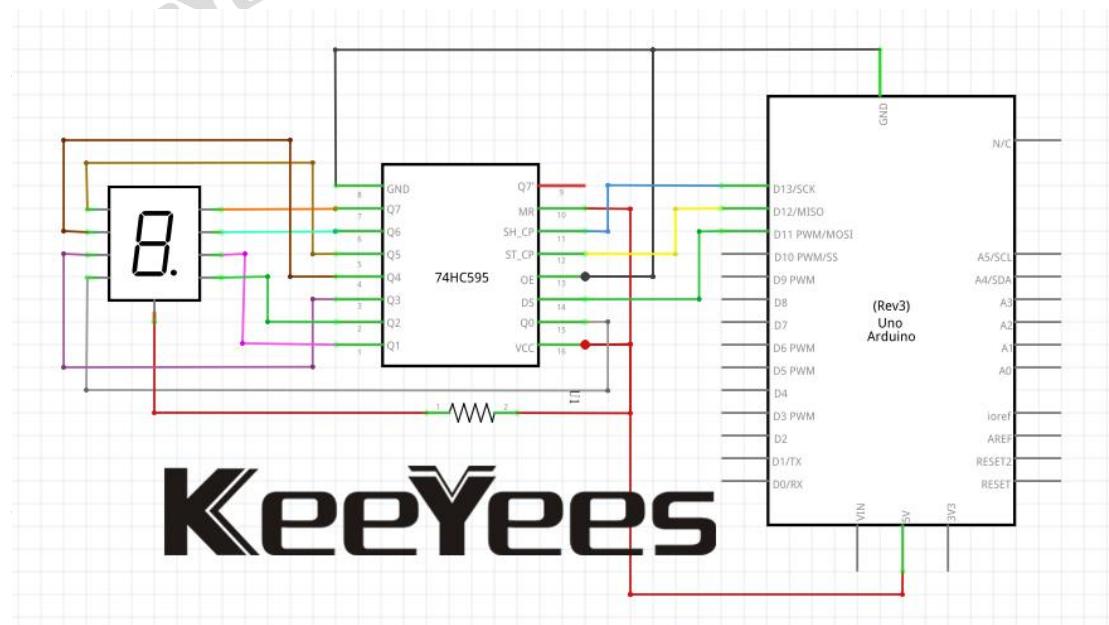


できる数字と英文字は 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F です。



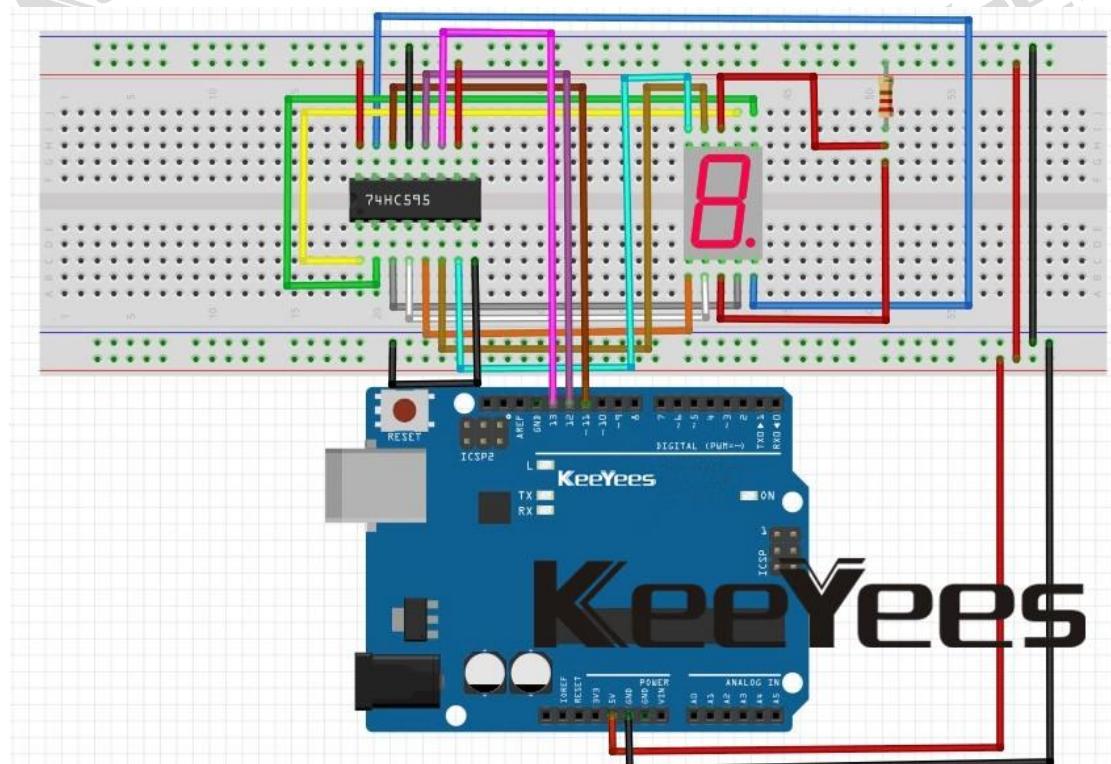


接続図





配線図





## コード

Aduino IDEを開いて ⇒ 新しいファイルを作成します ⇒ 以下のコードをコピーし、作成したファイルにある元のコードを置き換えてペーストします ⇒ プログラムがきちんとかけているかチェックするコンパイルを行います ⇒ コードをファイルに保管できます ⇒ Arduinoに書き込むアップロードを行います。

手順はレッスン1と同じです。お忘れの場合、レッスン1をご確認ください。

```
unsigned char number[10] = {  
    B10000000, // 0  
    B11110010, // 1  
    B01001000, // 2  
    B01100000, // 3  
    B00110010, // 4  
    B00100100, // 5  
    B00000100, // 6  
    B11110000, // 7  
    B00000000, // 8  
    B00100000 // 9  
};  
const byte Pin_DS = 11; //data  
const byte Pin_ST_CP = 12; //latch  
const byte Pin_SH_CP = 13; //clock  
char Buff[1];  
  
void setup() {  
    // put your setup code here, to run once:  
    Serial.begin(9600);  
  
    pinMode(Pin_ST_CP, OUTPUT); //ST_CP  
    pinMode(Pin_DS, OUTPUT); //DS  
    pinMode(Pin_SH_CP, OUTPUT); //SH_CP  
    digitalWrite(Pin_DS, LOW);  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
}
```



```
if (Serial.available())
{
    Serial.readBytes(Buff, sizeof(Buff));
    int comInt = charToInt();
    digitalWrite(Pin_ST_CP, LOW);
    shiftOut(Pin_DS, Pin_SH_CP, MSBFIRST, number[comInt]);
    digitalWrite(Pin_ST_CP, HIGH);
    delay(1000);
}

int charToInt()
{
    int tmp = 0;
    for (int i = 0; i < 1; i++)
    {
        tmp = tmp * 10 + (Buff[i] - 48);
    }
    return tmp;
}
```

コードを書き込んだ後、シリアルモニターを開いて（お忘れの場合、レッスン8をご確認ください）、0～9のどちらの数字を送信すると、LED表示器に表示されます。

### コード解説

```
unsigned char number[10] = {
    B10000000, // 0
    B11110010, // 1
    B01001000, // 2
    B01100000, // 3
    B00110010, // 4
    B00100100, // 5
    B00000100, // 6
    B11110000, // 7
    B00000000, // 8
    B00100000 // 9
};
```

まず、1次元配列を定義します。配列にLED表示器に表示できるコーディングを格納します。上記のコードには10つのコーディングがあります。0～9の10



個の数字です。

```
void loop() {
    // put your main code here, to run repeatedly:

    if (Serial.available())
    {
        Serial.readBytes(Buff, sizeof(Buff));
        int comInt = charToInt();
        digitalWrite(Pin_ST_CP, LOW);
        shiftOut(Pin_DS, Pin_SH_CP, MSBFIRST, number[comInt]);
        digitalWrite(Pin_ST_CP, HIGH);
        delay(1000);
    }
}
```

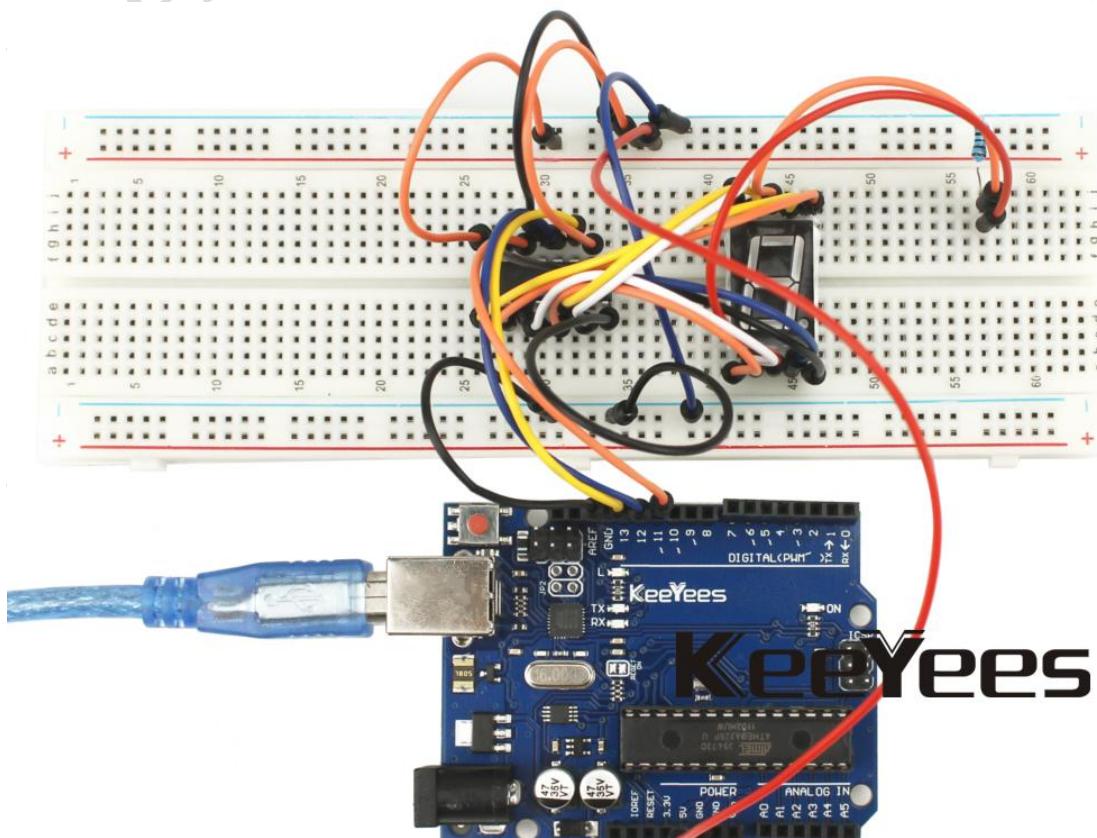
`loop()` 関数を使用して入力したコマンドを連続的に検測できます。入力したコマンドが検測されたら、変数「`comInt`」に保存し、「`shiftOut`」関数を使用してコマンド相応の数字を表示します。もし、入力した番号は 0~9 の範囲外の場合、何も表示しません。

```
int charToInt()
{
    int tmp = 0;
    for (int i = 0; i < 1; i++)
    {
        tmp = tmp * 10 + (Buff[i] - 48);
    }
    return tmp;
}
```

「`charToInt`」関数は入力した数字をバイナリコードに変換して元に戻します。最後に「`int comInt = charToInt ()`」を通して変数「`comInt`」に保存します。



例





## Lesson 11 PCB はんだ付け

ユニバーサル基板を利用して回路とコンポーネントをはんだ付けします。このレッスンを学んだ後、独自の回路を設計でき、構築できればと思います。はんだ付けには必要な道具を用意しなければなりません。このセットにはユニバーサル基板、基板用はんだ、ピンヘッダー、ピンソケットがあります。はんだごてをご使用している時、やけどしないようにお気をつけてください。

### Part 1

#### 電子ブザーが鳴る回路をはんだ付けする

##### 概要

このレッスンでボタンで電子ブザーを制御する回路をはんだ付けします。ボタンを押している時、電子ブザーが鳴ります。この回路にはプログラミングが必要がなくて、電源に接続すればいいです。自転車や、寝室のドアなど、ご必要なところに設置できます。

##### 必要なもの

1 x 2 ピン ピンヘッダー

1 x LED 赤

1 x  $220\Omega$  抵抗器

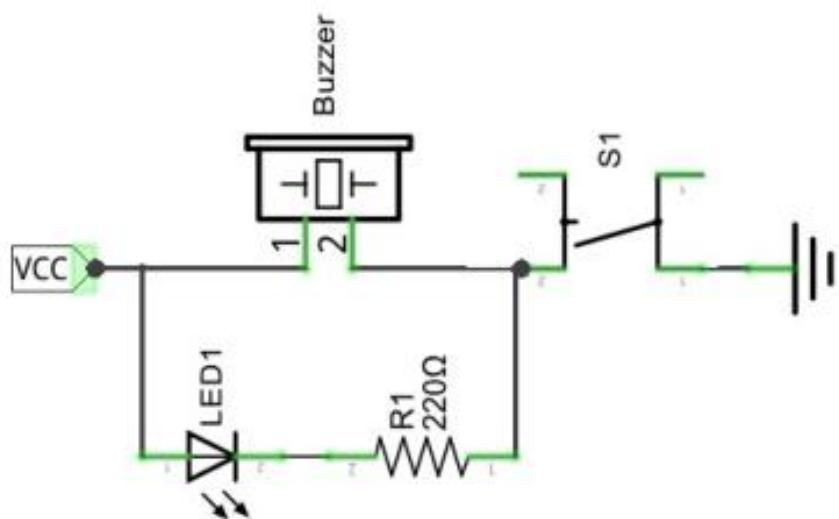
1 x 電子ブザー

1 x タクトスイッチ



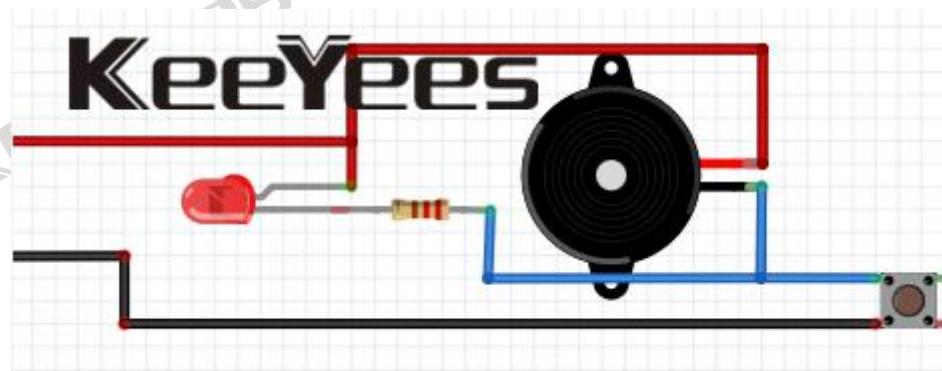
はんだ付けのコツ

回路図



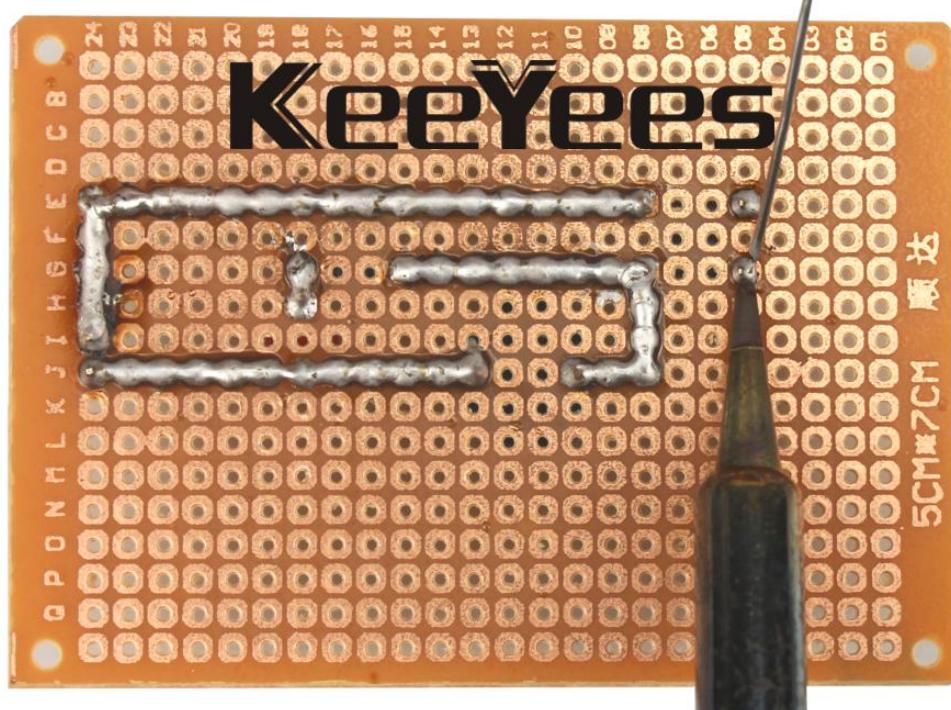


接続図



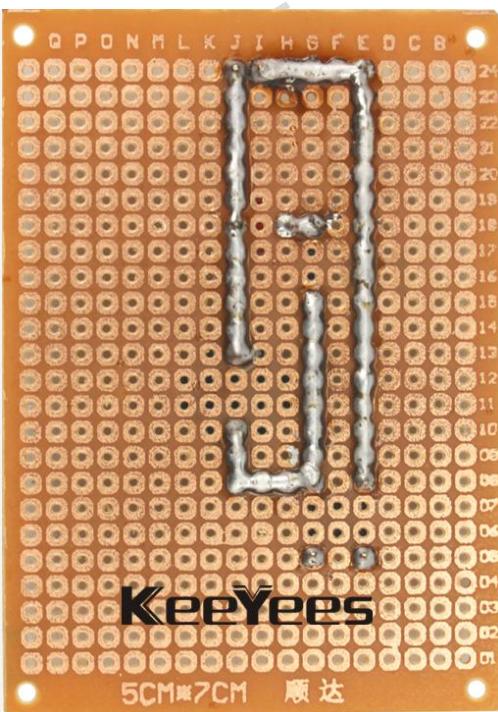
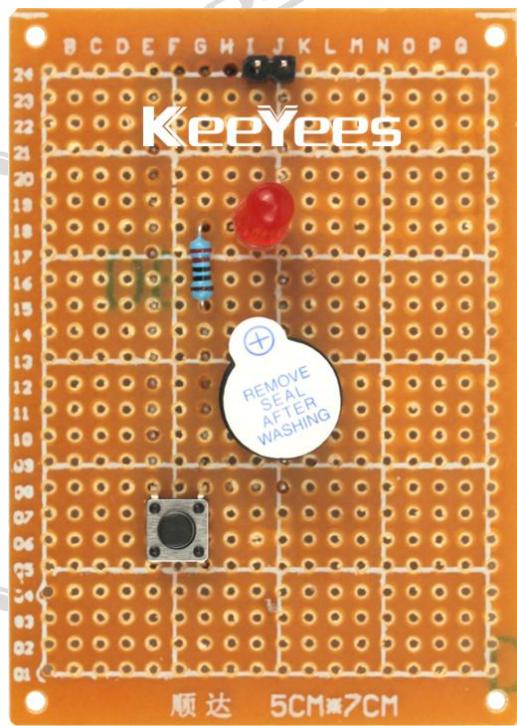
はんだ付け回路

コンポーネントをユニバーサルボードに挿入し、ピンを反面側(銅箔付かない側)から挿入し、銅箔つき側にはんだ付けします。以下の通りです。





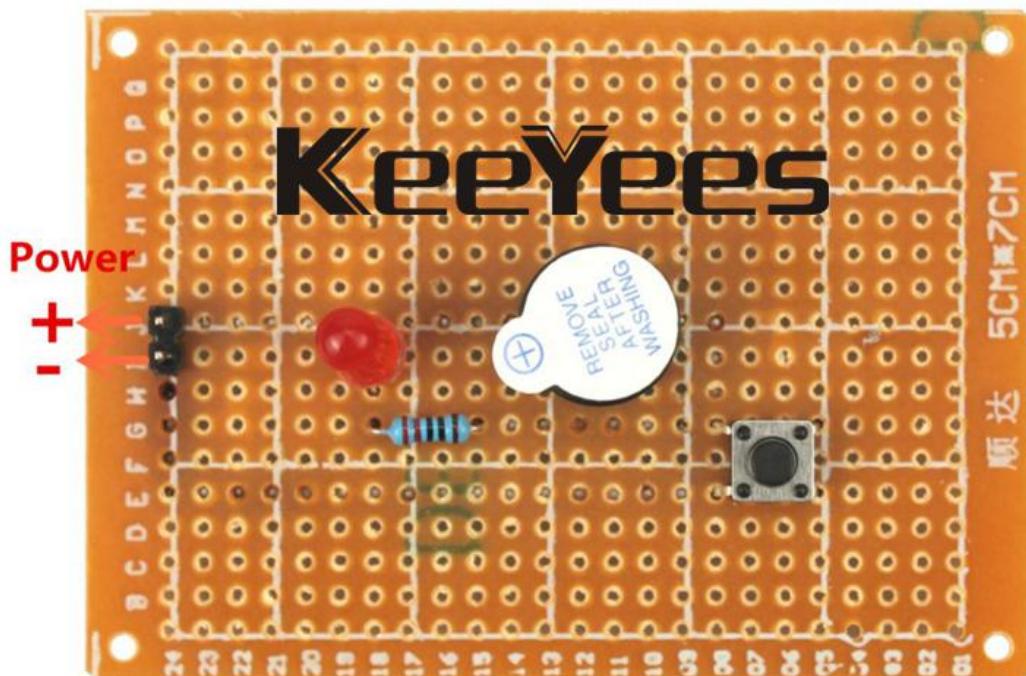
はんだ付け済みのものは以下の図の通りはずです。





うまくはんだ付けできますかをチェック

2つのピンを電源（3.3V～5V）に接続します。電源はバッテリー、Arduino開発ボード、Raspberry Piなどから給電できます。電源に接続した後、ボタンを押すと、LEDが点灯し、電子ブザーが鳴りましたら、回路がちゃんとはんだ付けされましたということです。失敗したら、回路をよくご確認ください。





## Part 2

### LED ストリームライトの回路をはんだ付けする

#### 概要

このレッスンでは、LED ストリームライト(流れる点滅)の回路のはんだ付けを学びます。8 個の LED があるため、今回も 74HC595 IC を使います。

#### 必要なもの

1 x 7 ピン ピンヘッダー

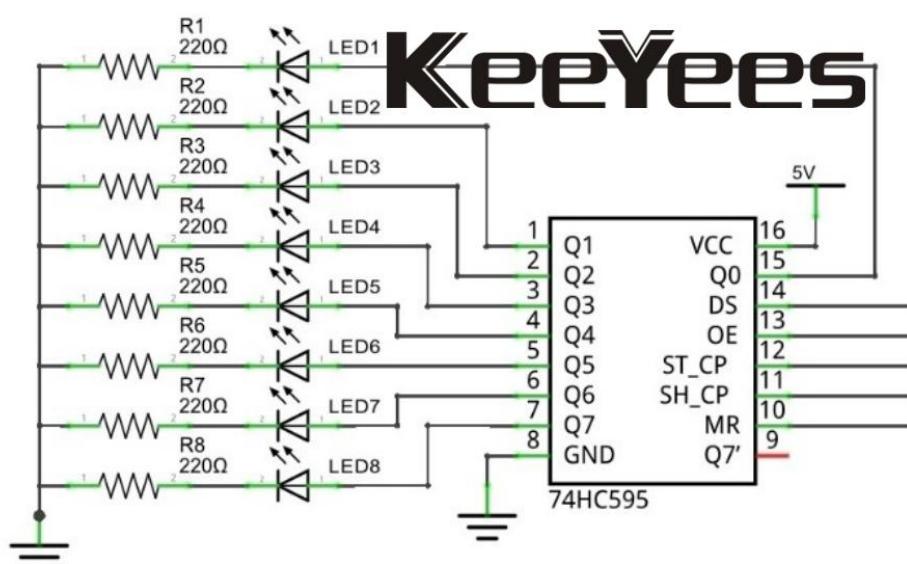
8 x LED

8 x 220Ω 抵抗器

1 x 74HC595

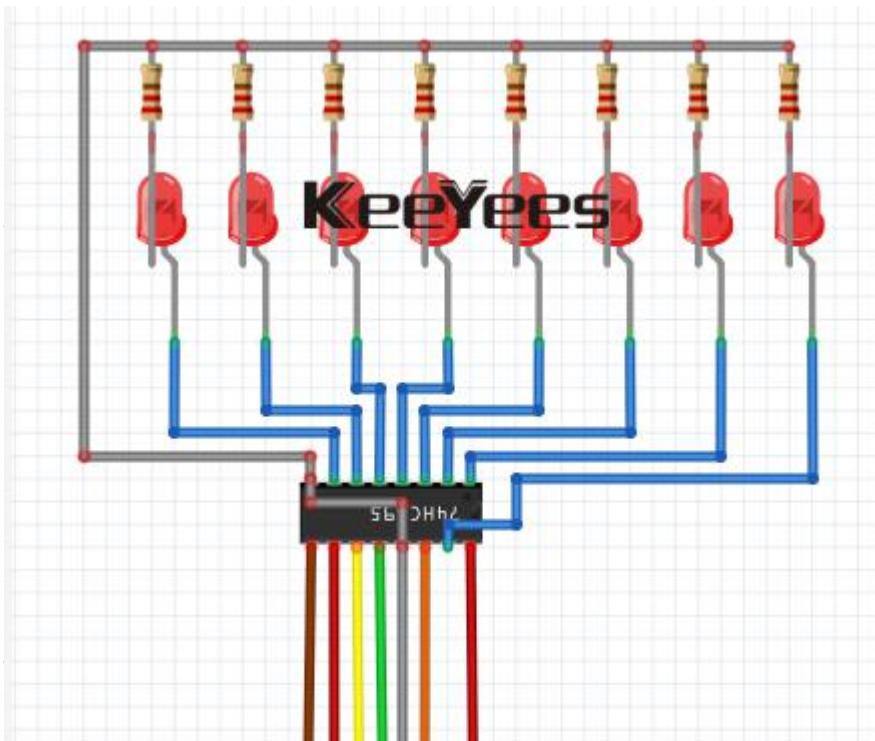
#### はんだ付けのコツ

#### 回路図



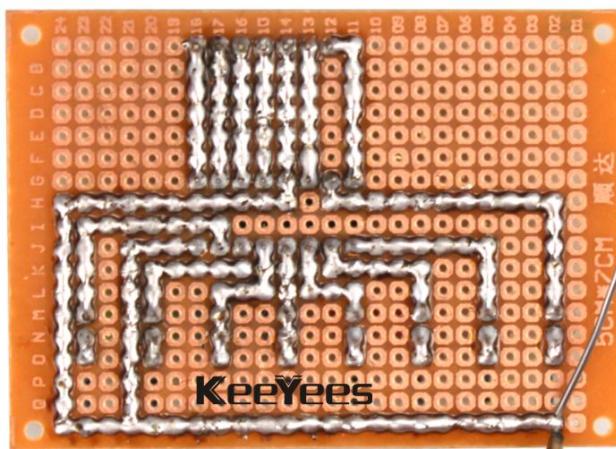


接続図



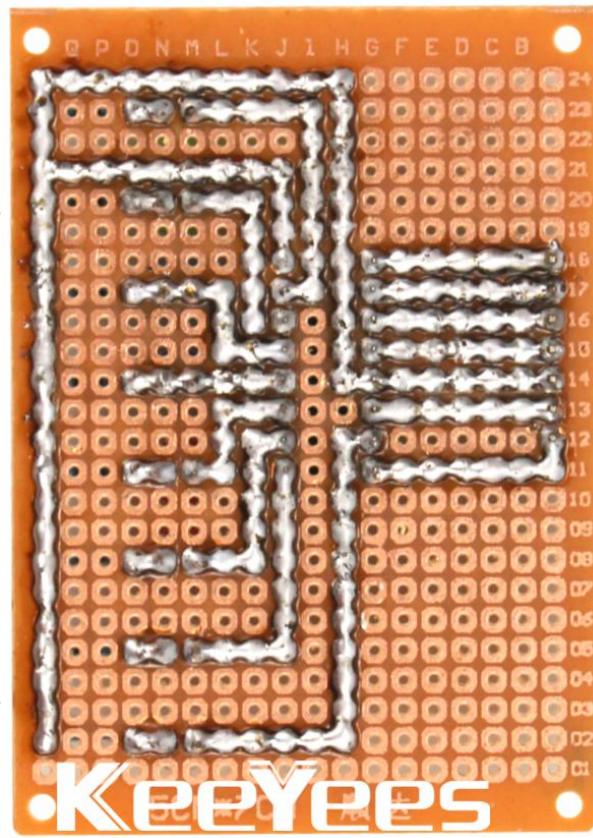
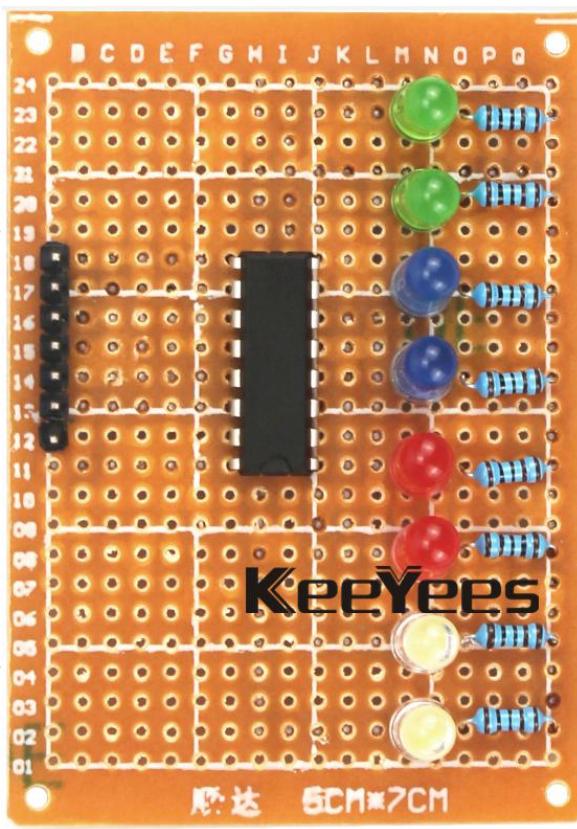
### はんだ付け回路

コンポーネントをユニバーサルボードに挿入し、ピンを反面側(銅箔付かない側)から挿入し、銅箔つき側にはんだ付けします。以下の通りです。





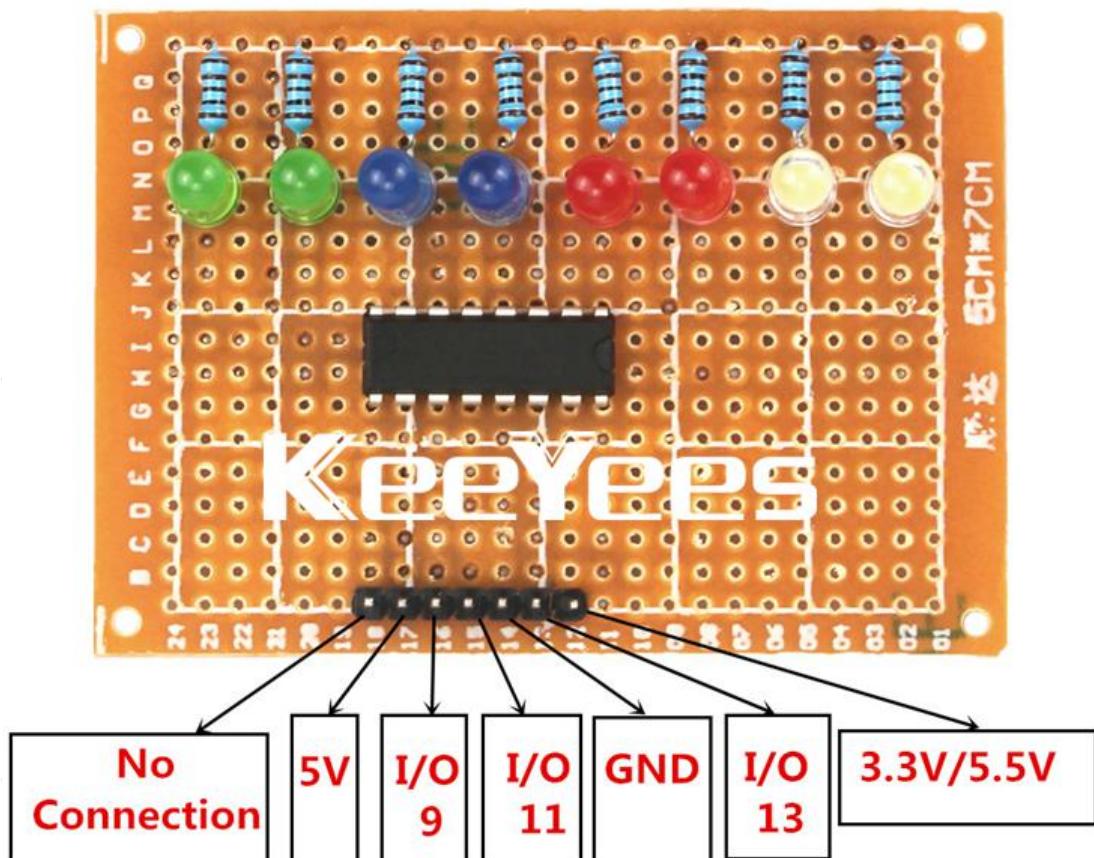
はんだ付け済みのものは以下の図の通りはずです。





うまくはんだ付けできますかをチェック

- 1) Arduino に接続します。



- 2) 下記のコードを Arduino 開発ボードに書き込みます。 (手順やコード解説は、レッスン7をごチェックください)。

```
int tDelay = 100;
int latchPin = 11;      // (11) ST_CP [RCK] on 74HC595
int clockPin = 9;       // (9) SH_CP [SCK] on 74HC595
int dataPin = 12;        // (12) DS [S1] on 74HC595
byte leds = 0;
void updateShiftRegister()
```



```
        digitalWrite(latchPin, LOW);
        shiftOut(dataPin, clockPin, LSBFIRST, leds);
        digitalWrite(latchPin, HIGH);
    }

void setup()
{
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
}

void loop()
{
    leds = 0;
    updateShiftRegister();
    delay(tDelay);
    for (int i = 0; i < 8; i++)
    {
        bitSet(leds, i);
        updateShiftRegister();
        delay(tDelay);
    }
}
```