

IoT Basic Weekend

DAY1:I/O Devices

1. IoT Basic Weekend
 - 1-1. IoT Basic Weekend
 - 1-2. FabAcademy
 - 1-3. 講座の流れ
 - 1-4. DAY1 I/O Devices
2. 準備
 - 2-1. 部品一覧
 - 2-2. 環境構築
3. Learn : 基礎 LEDを制御
 - 3-1. 部品紹介
 - 3-2. 電子回路
 - 3-3. プログラム作成
 - 3-4. プログラム解説
 - 3-5. 電子回路解説
 - 3-6. 演習①
4. Learn : 応用 タクトスイッチ
 - 4-1. 部品紹介
 - 4-2. 電子回路
 - 4-3. プログラム作成
 - 4-4. シリアルモニター
 - 4-5. 電子回路解説
 - 4-6. プログラム解説
 - 4-7. プログラムの改良
 - 4-8. 論理演算子
 - 4-9. 演習②
5. Learn : 応用 測距 / 環境光センサー
 - 5-1. 部品紹介
 - 5-2. 電子回路
 - 5-3. モジュールとライブラリー
 - 5-4. ライブラリー追加
 - 5-5. プログラム作成
 - 5-6. プログラム解説
 - 5-7. 演習③
6. Learn : 応用 サーボモーター
 - 6-1. 部品紹介
 - 6-2. 電子回路
 - 6-3. プログラム
7. Make
8. 部品の購入先

1-1. IoT Basic Weekend

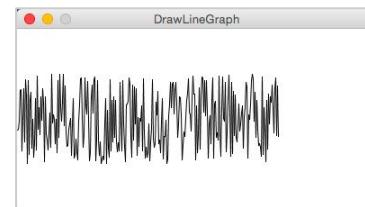
IoTとは、Internet of Thingsの略でさまざまなモノをインターネットに接続して、情報をやり取りさせる仕組みです。

例えば「部屋の温度、湿度」といった基本的な環境情報から、「植物の水がなくなっている」、「猫のエサがなくなっている」といった情報までをさまざまなセンサを使って数値化し、インターネットを介して取得します。そしてその情報をもとに「植物に水をあげる」、「猫にエサをあげる」といったさまざまな処理を行うことができます。

IoT Basicプログラムでは、IoTを実際の生活に利用するための基礎知識として、多様なセンサデバイスの扱い方、プログラムの書き方、情報を映像化する方法を学びます。



情報の映像化



センサー

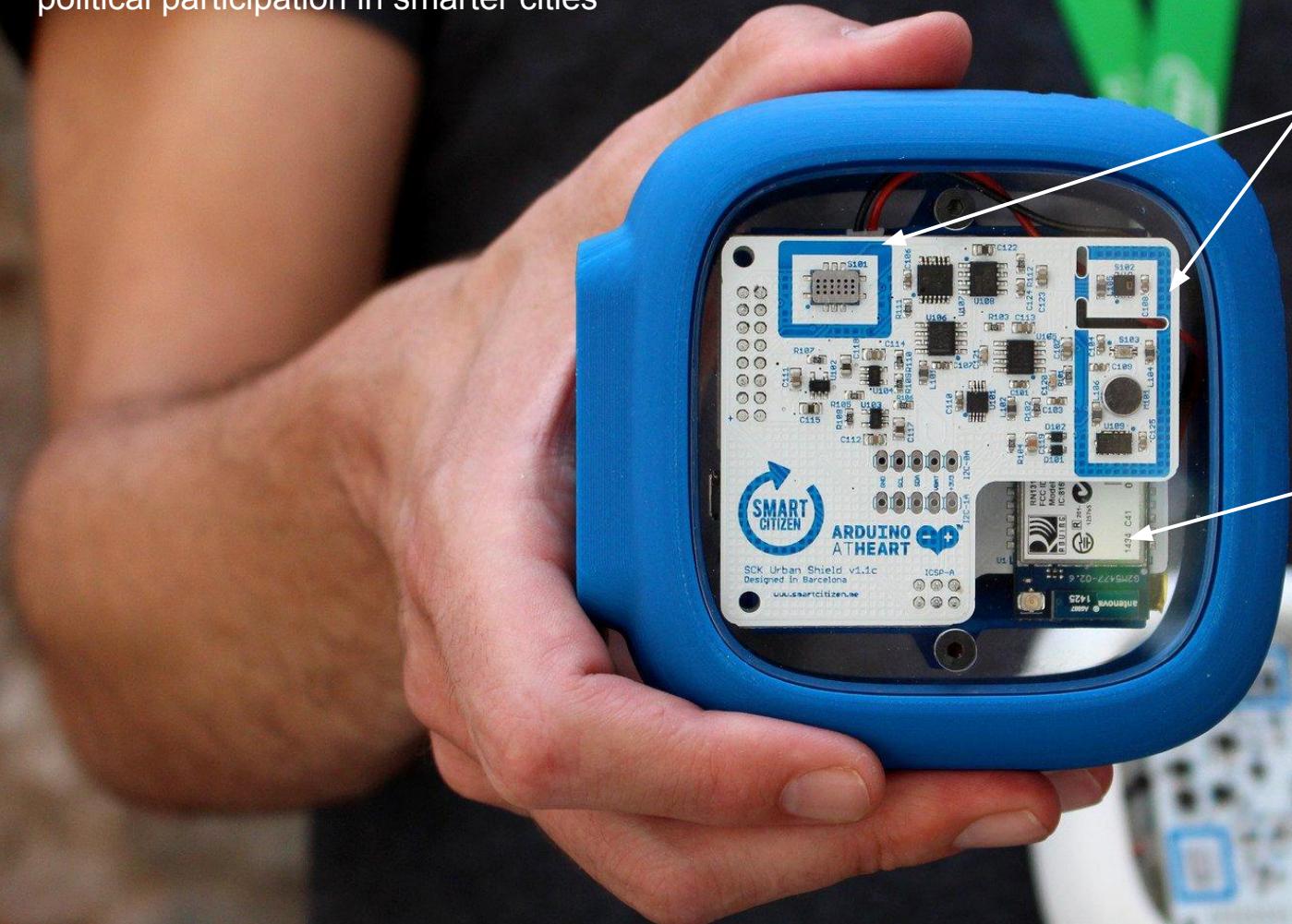


プログラムの書き方

```
1 float x, y, lastX, lastY;
2
3 void setup(){
4     x = 0;
5     y = height/2;
6     lastX = 0;
7     lastY = height/2;
8     size(400, 200);
9     background(255);
10 }
11
12 void draw(){
13
14     y = height/2 + random(-50,50);
15     line(x, y, lastX, lastY);
16
17     lastX = x;
18     lastY = y;
19
20     if(x>width){
```

Smart Citizen

Open source technology for citizens
political participation in smarter cities



センサー

- 温度
- 湿度
- NO2
- CO

インターネット



1-2. FabAcademy

IoT Basicは世界中のファブラボで同時に学ぶ講座、FabAcademyを元にして構成しています。

FabAcademyは6ヶ月間で3Dモデリング、回路設計、プログラミング、機構設計から材料実験までモノをつくるために必要なスキルを総合的に身に付ける講座です。

ファブラボ鎌倉ではIoT Basic、Fab Basicの2つの講座を学ぶことで、モノの外装と中身の両方をつくれるスキルを身につけていただけるように構成しています。



Fab Basic

1. Computer Aided Design
2. Computer-Control Cutting
3. 3DPrinting and Scanning
4. Computer Control Machining
5. Mechanical Design

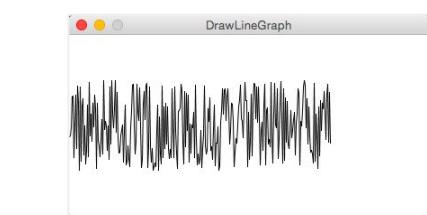
IoT Basic

1. Input Devices
2. Output Devices
3. Network and Communications
4. Electronics Production
5. Embedded Programming

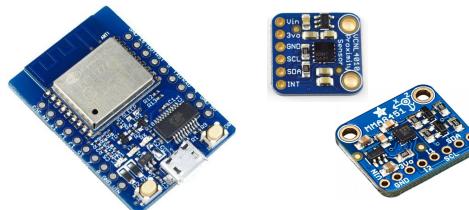
1-3. 講座の流れ

DAY1 : I/O - インプット、アウトプット -
マイコンボードとI/Oデバイス(各種センサー、LED、モーターなど)の扱い方を学びます。

DAY2 : Communication - 通信 -
マイコンボードを使用して情報を送信、受信する方法を学びます。



情報の映像化

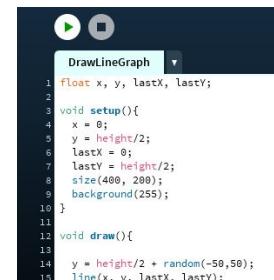


マイコンボード

センサー



プログラムの書き方



```
1 float x, y, lastX, lastY;
2
3 void setup(){
4     x = 0;
5     y = height/2;
6     lastX = 0;
7     lastY = height/2;
8     size(400, 200);
9     background(255);
10 }
11
12 void draw(){
13     y = height/2 + random(-50,50);
14     line(x, v, lastX, lastY);
15 }
```



```
1 int spd = 1000;
2 void setup() {
3     // ESPrの13番ピンを出力に設定
4     pinMode(13, OUTPUT);
5 }
6
7 void loop() {
8     digitalWrite(13, HIGH);
9     delay(1000);
10    digitalWrite(13, LOW);
11    delay(1000);
12 }
```

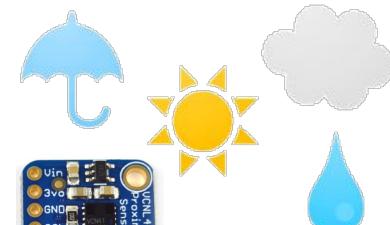
1-4. DAY1 I/Oデバイス - インプット、アウトプット -

Sensing – センシング – では、各種センサーモジュールとマイコンを使って温度、光量といった環境情報を取得します。

ブレッドボードを使って、簡単な電子回路を製作する方法から、Arduinoソフトウェアを使ってマイコンを扱うためのプログラムの書き方まで学んでいきます。



マイコンボード

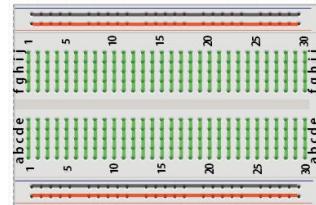


センサー
モジュール

```
LedBlink
1 int spd = 1000;
2 void setup() {
3 // ESP32の13番ピンを出力に設定
4 pinMode(13, OUTPUT);
5 }
6
7 void loop() {
8 digitalWrite(13, HIGH);
9 delay(spd);
10 digitalWrite(13, LOW);
11 delay(spd);
12 }
```

保存しました。

Arduinoソフトウェア



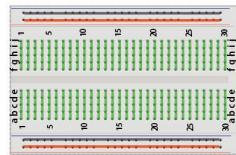
ブレッドボード

2. 準備

2-1. 部品一覧



マイコン
ESP3r Developer



ブレッドボード



マイクロUSBケーブル



抵抗 10kΩ



抵抗 510Ω



LED



タクトスイッチ



測距/環境光センサ
VCNL4010



ジャンパー
ワイヤー



サーボモーター
SG-90



電池ボックス



電池

2-2. 環境構築①

Arduino software(プログラム開発環境 IDE) のインストール

<https://www.arduino.cc/en/Main/Software>

OSに合ったインストールイメージをダウンロードしてインストールします

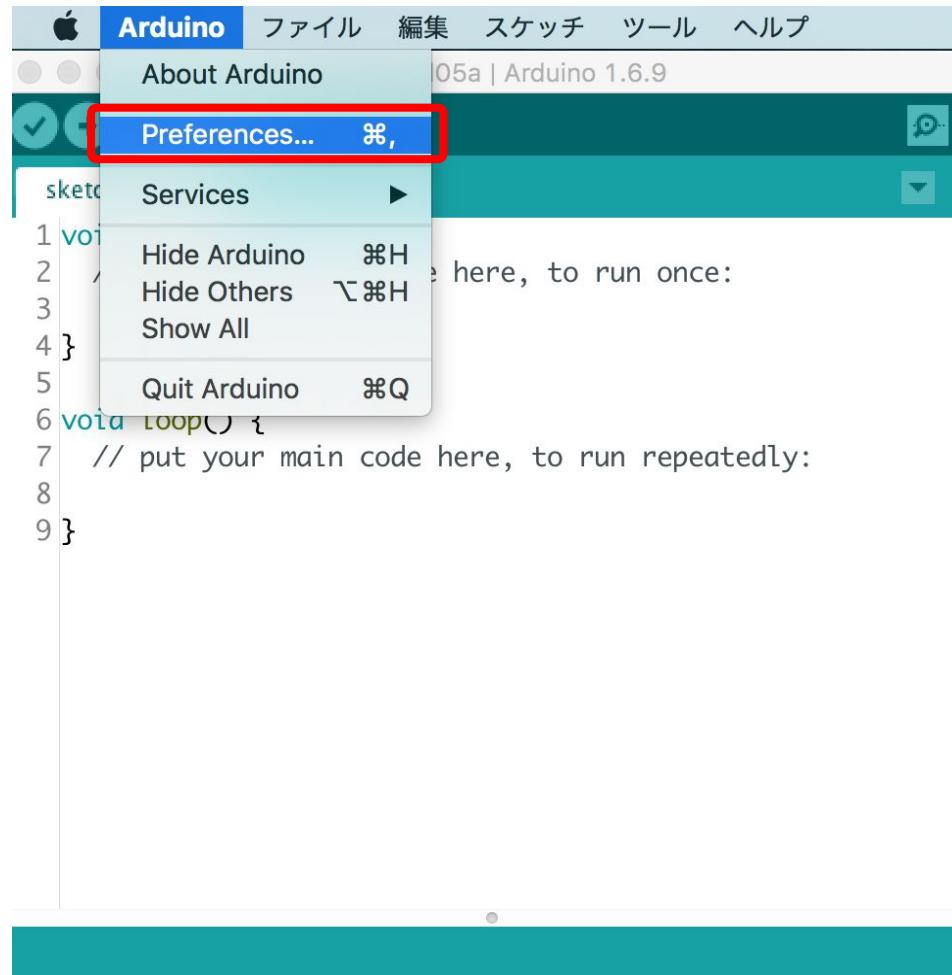


Download the Arduino Software

A screenshot of the Arduino Software download page. It features the Arduino logo and a brief description of the software. A red box highlights the download options for Windows and Mac OS X. The Mac OS X section is specifically labeled "Mac OS X 10.7 Lion or newer". Other download links include Linux 32 bits and Linux 64 bits, along with Release Notes, Source Code, and Checksums links.

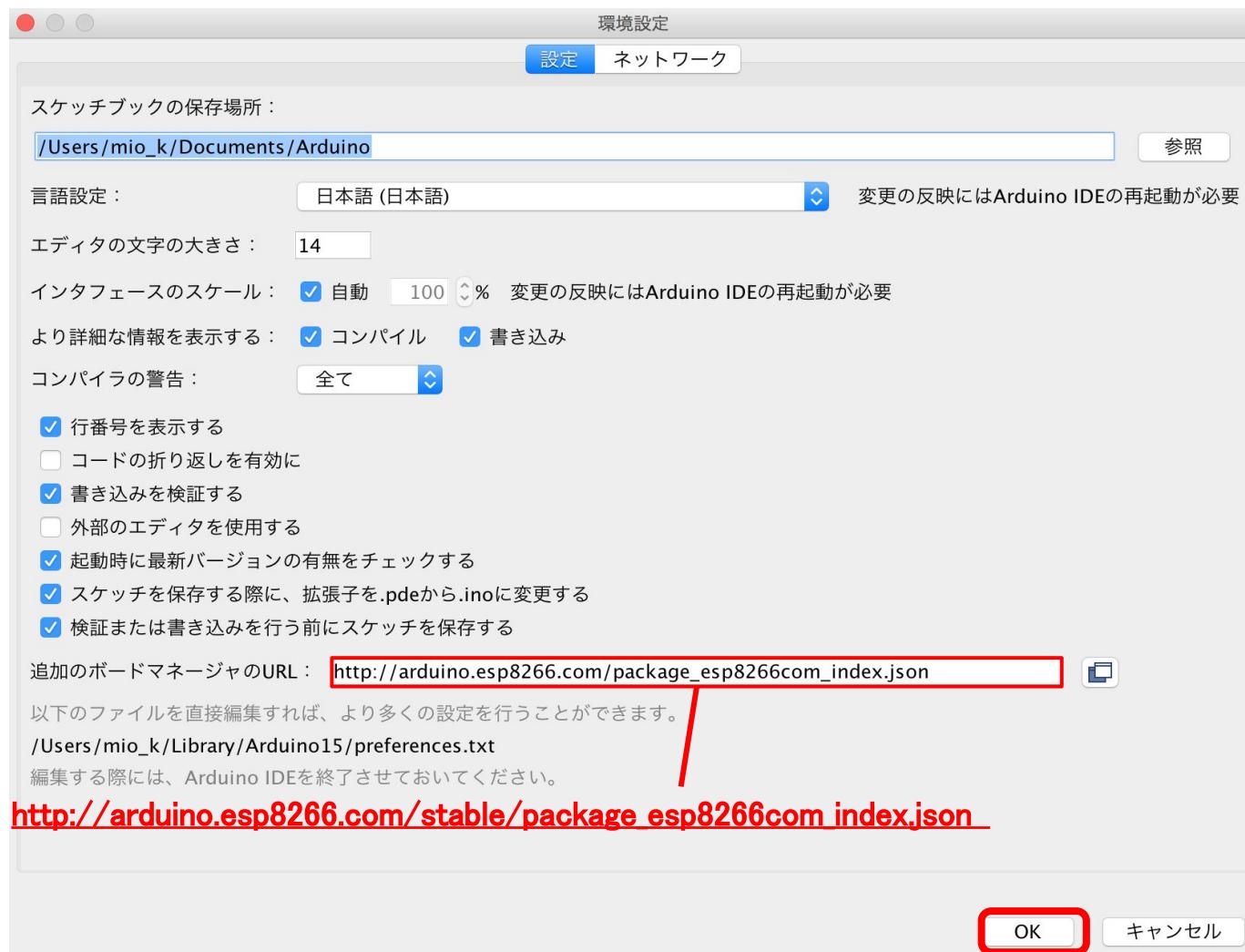
2-2. 環境構築②

Arduinoソフトウェアを立ち上げ、画面上部のメニューバーから「Arduino」>「Preferences」(Windows : 「ファイル」>「環境設定」)を開きます



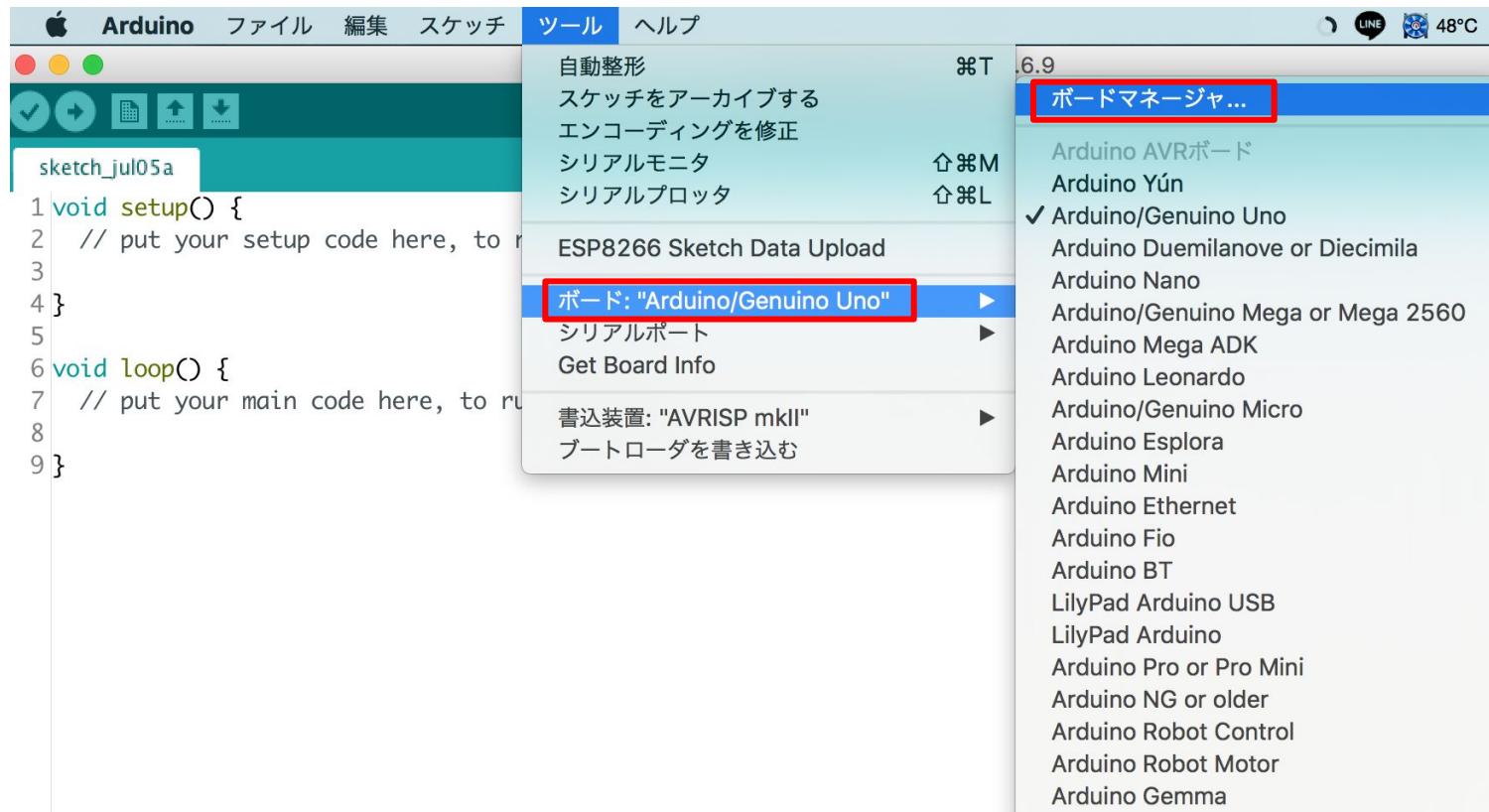
2-2. 環境構築③

「追加のボードマネージャーのURL」に画面下に記載したURLを入力し、「OK」を押します



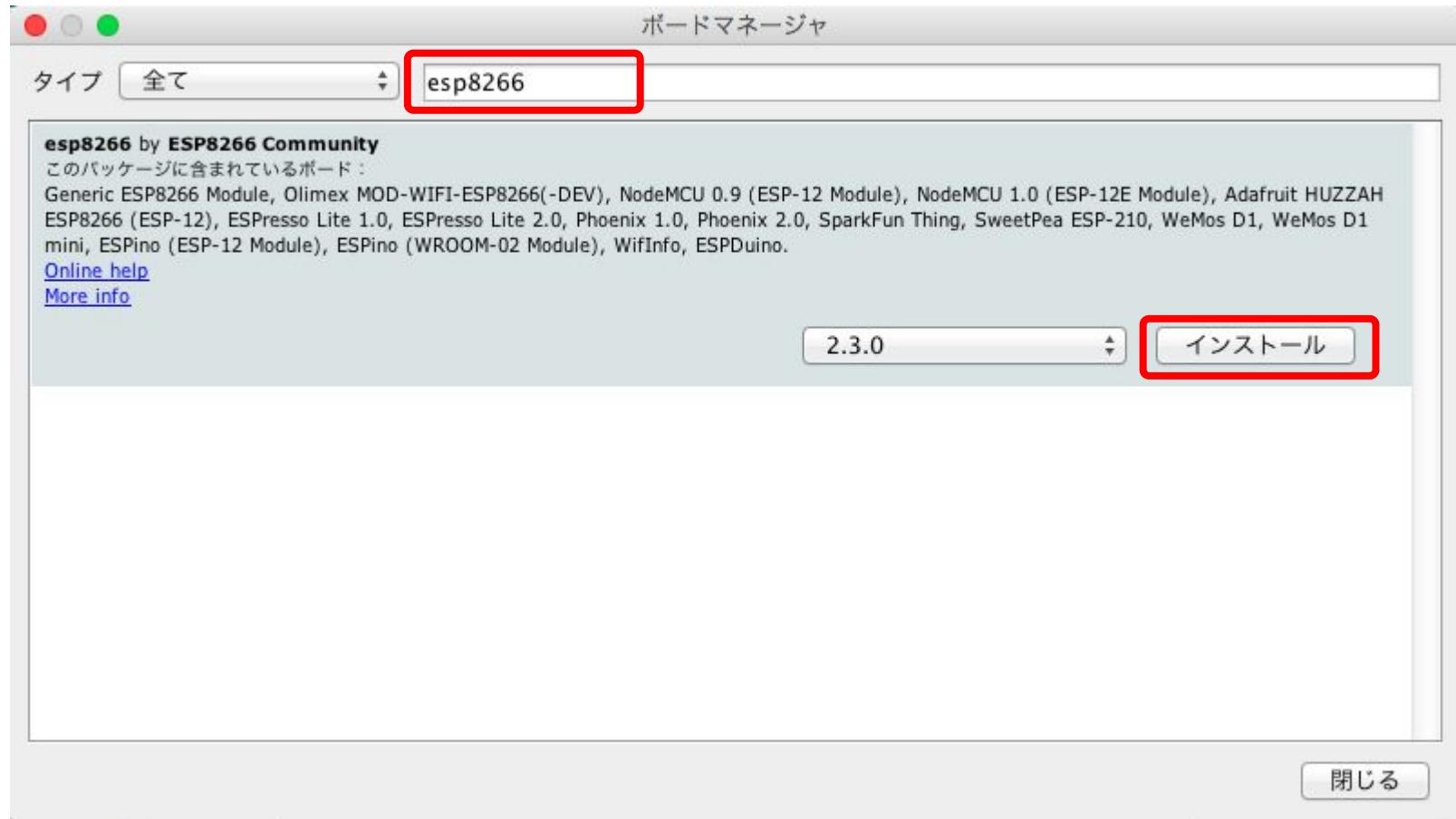
2-2. 環境構築④

「ツール」>「ボード」>「ボードマネージャー」を選択します。



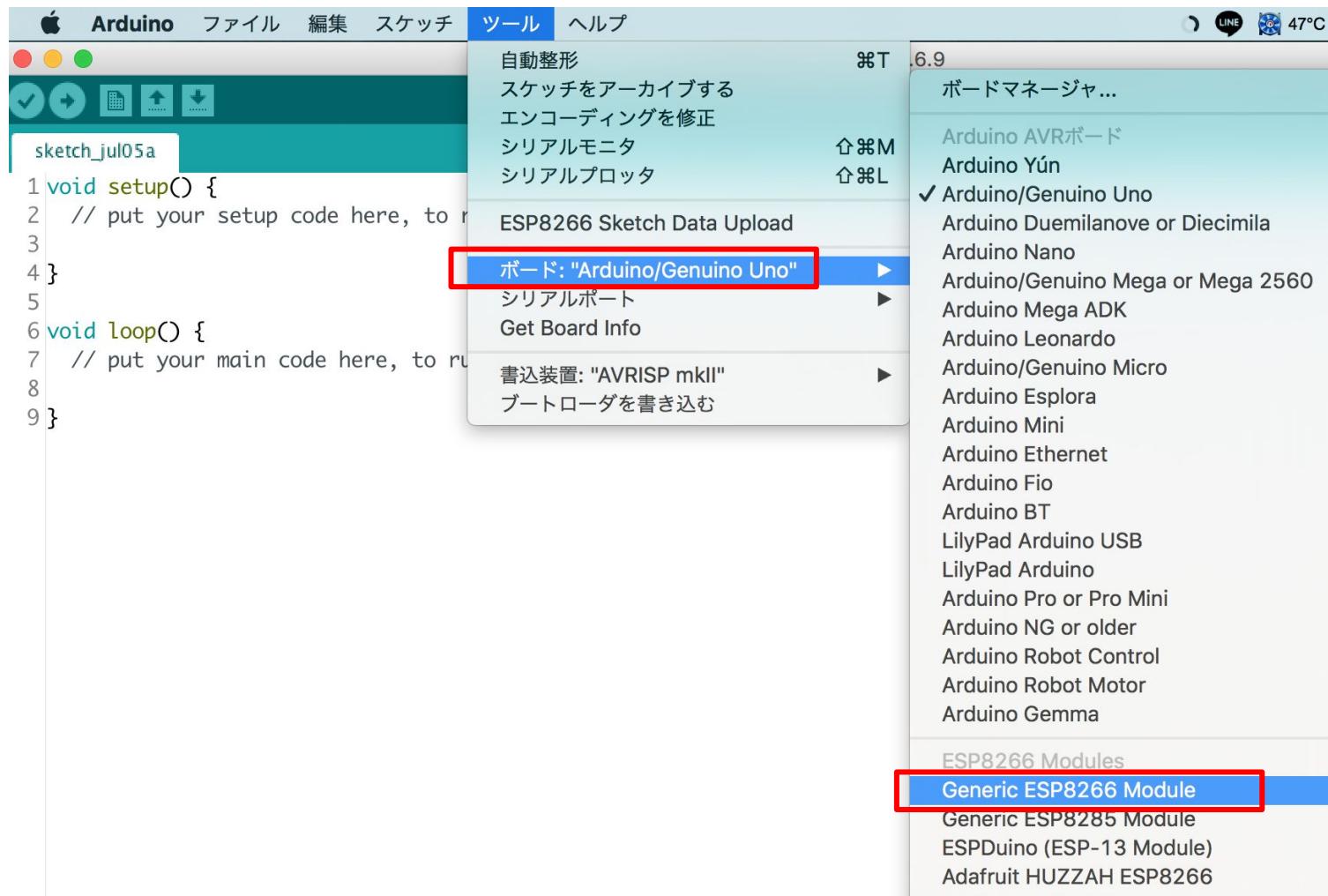
2-2. 環境構築⑤

1. 「esp8266」と検索します。
2. 最新のバージョンを選択して、インストールします。



2-2. 環境構築⑥

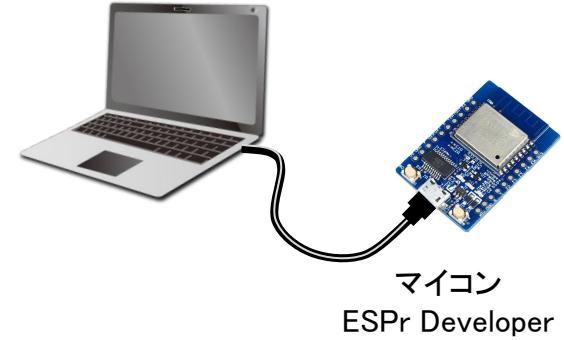
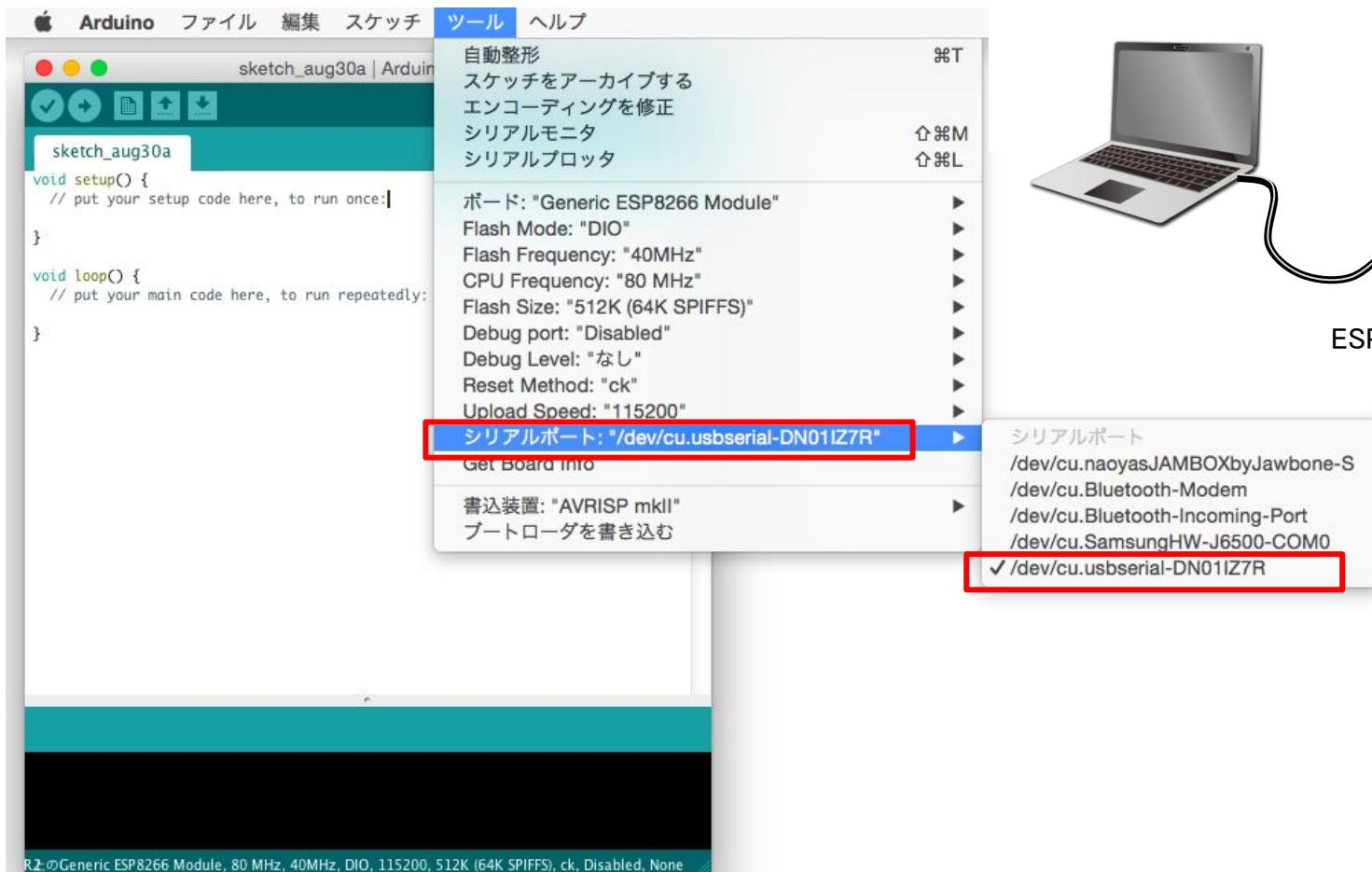
「ツール」>「ボード」>「Generic ESP8266 Module」を選択します



2-2. 環境構築⑦

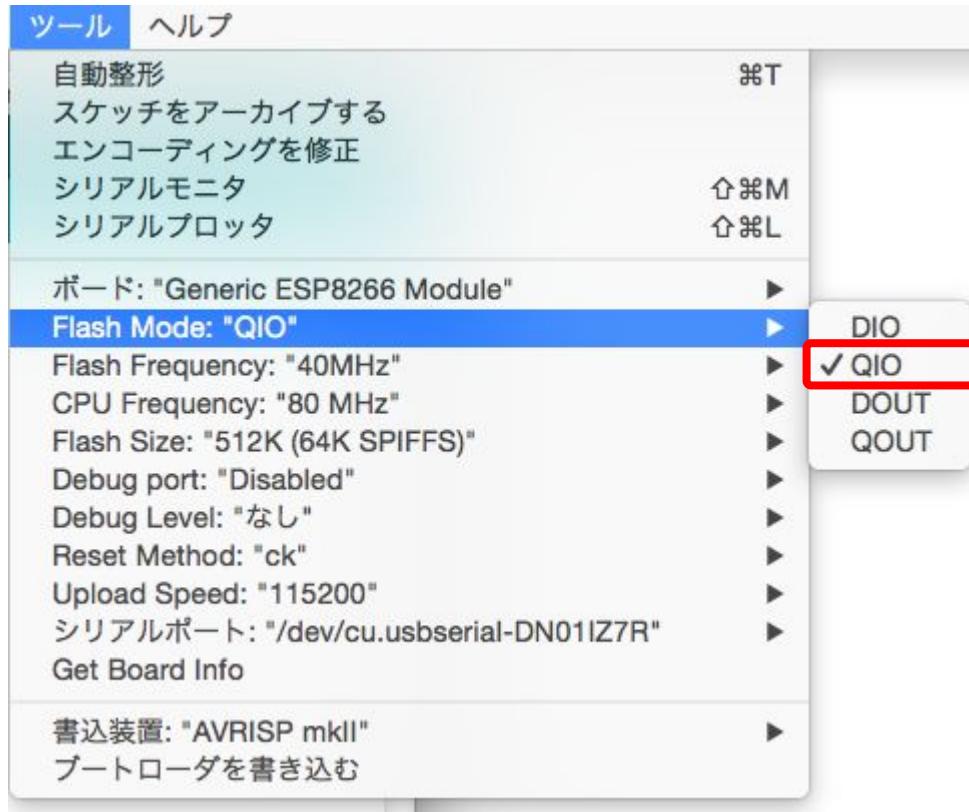
USBケーブルでマイコンボードとPCを接続し、「ツール」>「シリアルポート」> 適切なポート*を選択します。

* Macの場合は「/dev/cu.usbserial-………」、Windの場合は「COMx」



2-2. 環境構築⑧ Flash Mode

「ツール」>「Flash Mode」で「QIO」を選択します



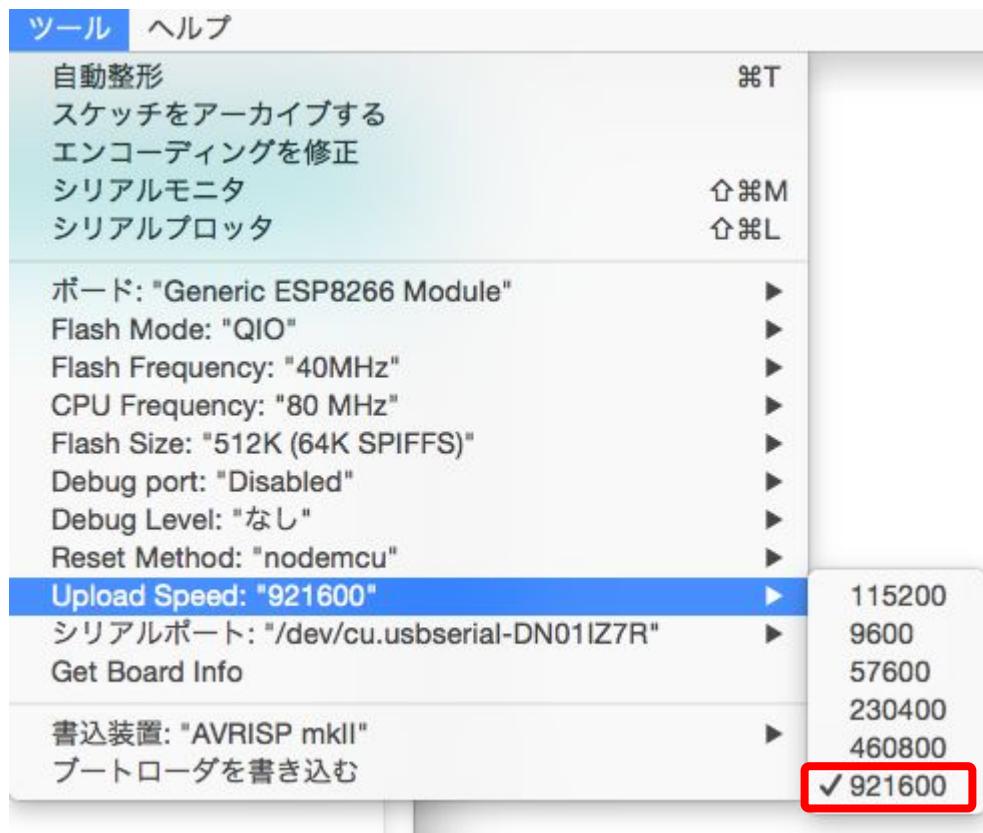
2-2. 環境構築⑨ Reset Method

「ツール」>「Reset Method」で「nodemcu」を選択します



2-2. 環境構築⑩ Upload Speed

「ツール」>「Upload Speed」で「921600」を選択します



3. Learn(基礎)：LEDを制御

3-1. 部品紹介① ESPr Developer

ESPr Developer

WiFiでインターネットに接続することができるマイコンボード。
Arduino環境を用いて開発することができるので、簡単な
プログラムでIoTデバイスを試作することができます。

ピン センサーラやLEDを接続する

リセットボタン 押すと再起動する

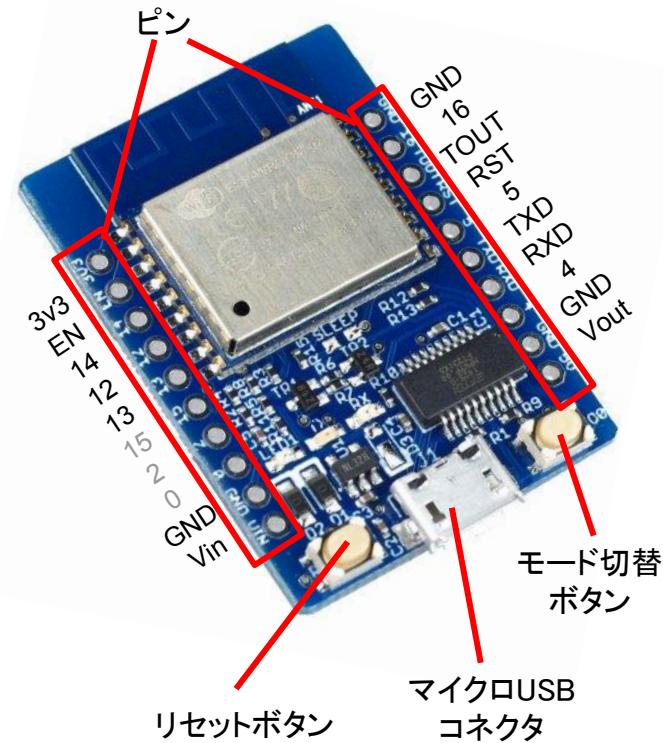
モード切替ボタン 今回は使用しません

マイクロUSB PCと接続。ここから給電も可能

注意事項

0, 2, 15番ピンはモード切替用に使用しているので使いません

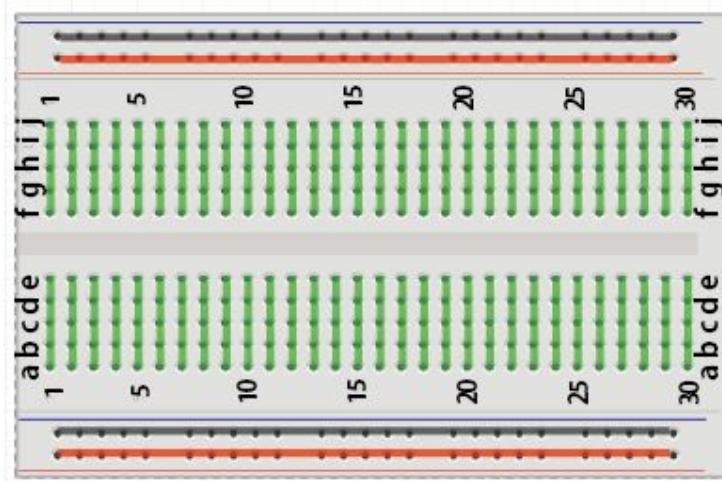
マイコンボード ESPr Developer



3-1. 部品紹介②：ブレッドボード

ブレッドボードの仕組み

ブレッドボードという穴のあいたボードと、ジャンパー線を使うと、半田付けやテープを使って部品の固定をしなくても簡単に回路を組むことができます



ブレッドボード 内部配線図



ジャンパー線（色の使い分け）

3-1. 部品紹介③ : LEDと抵抗



LED

LED (Light Emitting Diode : 発光ダイオード)

電圧を加えると発光する部品

電子工作では通電確認によく使われる

赤、青、緑、白の他、フルカラーLEDもある



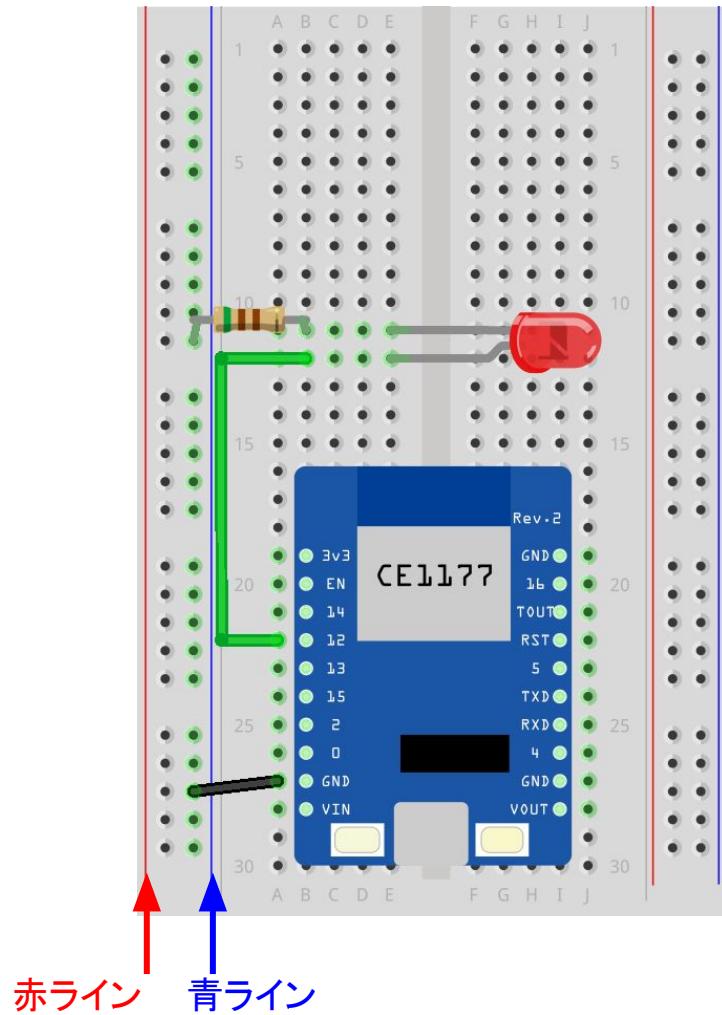
抵抗510Ω

抵抗 (Register)

電流を制限するために用いる

電極をそのままLEDにつなげると電流が流れすぎてLEDが壊れることがある。このとき、適切な抵抗をLEDと電極の間に入れることで、適切な電流をLEDに流すことができる。

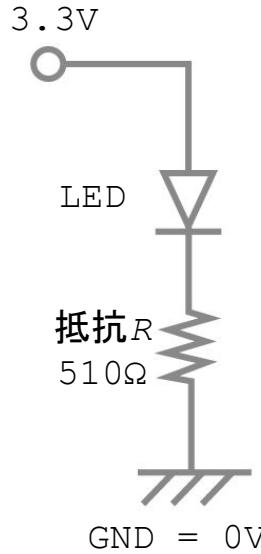
3-2. 電子回路を組む



電子部品を配線する

1. ESPr - VIN → B28
2. ESPr - VOUT → I28
3. LED - 短い足 → E11
4. LED - 長い足 → E12
5. 抵抗 510Ω : B11 → 青ライン
6. ジャンパー(黒) : A27 → 青ライン
7. ジャンパー(緑) : A22 → B12

電子回路図



3-3. プログラム作成

```
1 // LedBlink.ino
2
3 int spd = 1000;
4
5 void setup() {
6     // ESPrの12番ピンを出力に設定
7     pinMode(12, OUTPUT);
8 }
9
10 void loop() {
11     digitalWrite(12, HIGH); // 光る
12     delay(spd);           // 1秒待つ
13     digitalWrite(12, LOW); // 消える
14     delay(spd);           // 1秒待つ
15 }
```

1. プログラムを記述する
2.  「チェックアイコン」をクリックしてプログラムをコンパイルする
3.  「矢印アイコン」をクリックしてプログラムをESPrにアップロードする

注意事項

※ コンパイルが失敗すると、画面下のバーがオレンジ色に変化して黒いスペースにエラーメッセージが表示されます。エラーメッセージをよく読んでエラーに対処しましょう。

<ヒント> 文字の半角/全角、カッコ{}の位置や数が適切か、セミコロン;のつけ忘れ etc。

3-4. プログラム解説①

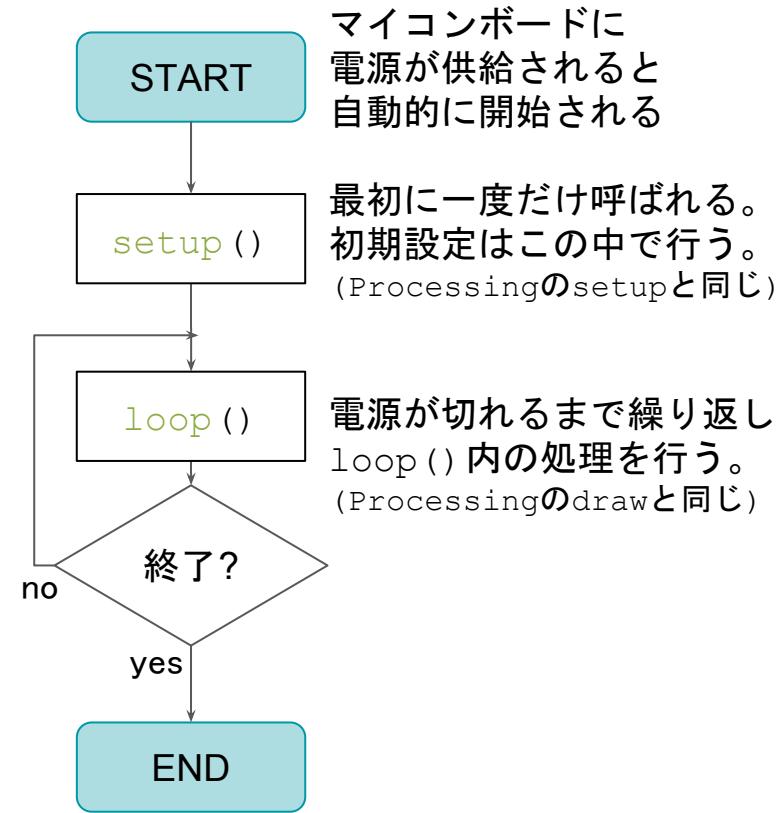
```
1 // LedBlink.ino
2
3 int spd = 1000;
4
5 void setup() {
6     // ESPrの12番ピンを出力に設定
7     pinMode(12, OUTPUT);
8 }
9
10 void loop() {
11     digitalWrite(12, HIGH); // 光る
12     delay(spd);           // 1秒待つ
13     digitalWrite(12, LOW); // 消える
14     delay(spd);           // 1秒待つ
15 }
```

delay()

何もしないで待つための関数。
引数に入力する値はミリ秒単位。

例)

```
delay(100);
100ミリ秒(0.1秒)待つ
```



3-4. プログラム解説②

```
1 // LedBlink.ino
2
3 int spd = 1000;
4
5 void setup() {
6     // ESPrの12番ピンを出力に設定
7     pinMode(12, OUTPUT);
8 }
9
10 void loop() {
11     digitalWrite(12, HIGH); // 光る
12     delay(spd);           // 1秒待つ
13     digitalWrite(12, LOW); // 消える
14     delay(spd);           // 1秒待つ
15 }
```

pinMode ()

ESPrのピンの設定を INPUT, OUTPUT どちらかに設定する。

1つ目の引数 指定するピンの番号

2つ目の引数 入力なら INPUT、出力なら OUTPUT を指定

例)

pinMode(12, OUTPUT);

12番ピンを出力に設定する

digitalWrite ()

pinMode () で OUTPUT に指定したピンに電圧をかけるか、かけないか設定する。

1つ目の引数 指定するピンの番号

2つ目の引数 電圧をかける HIGH か、かけない LOW かを指定

例)

digitalWrite(12, HIGH);

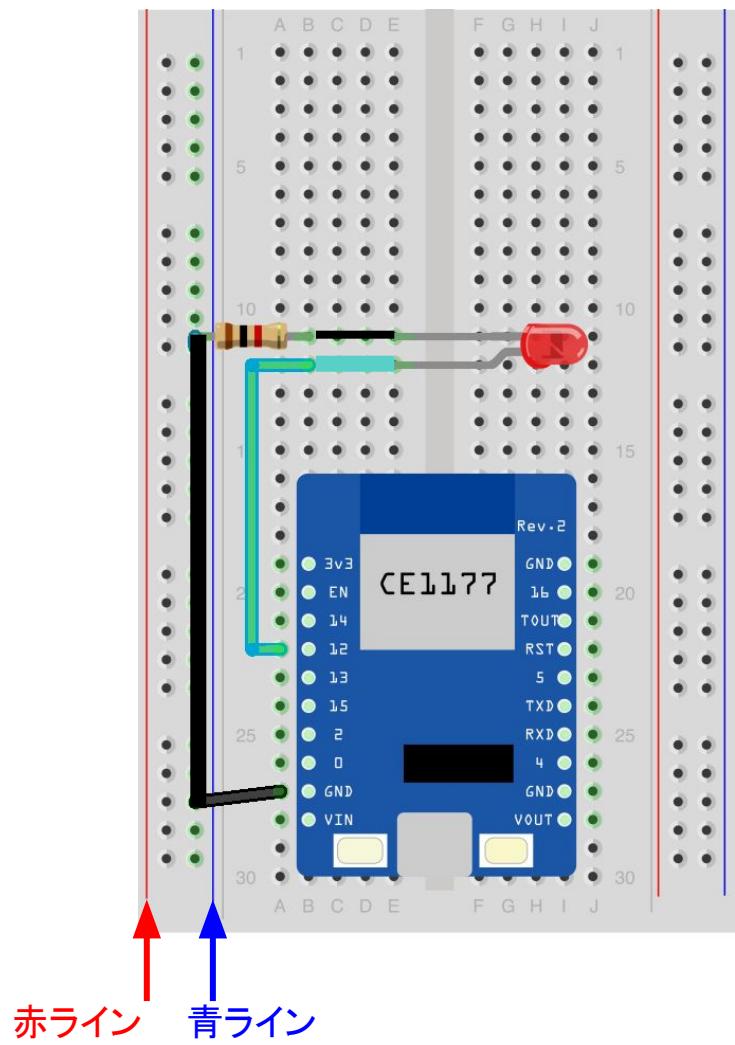
出力に設定した 12 番ピンに電圧をかける

Tips : Arduinoリファレンス

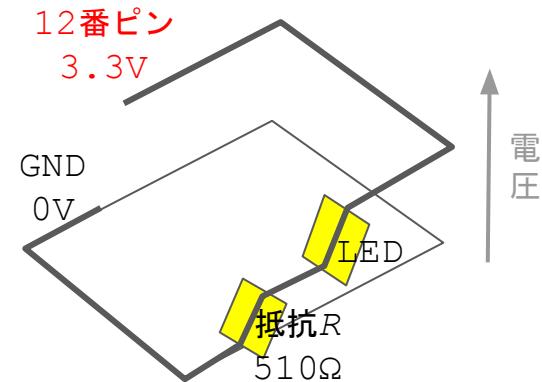
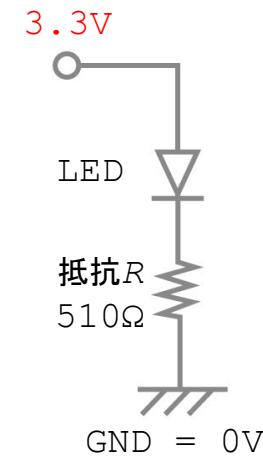
[公式リファレンス](#) (英語)

[日本語リファレンス](#) (2008年訳)

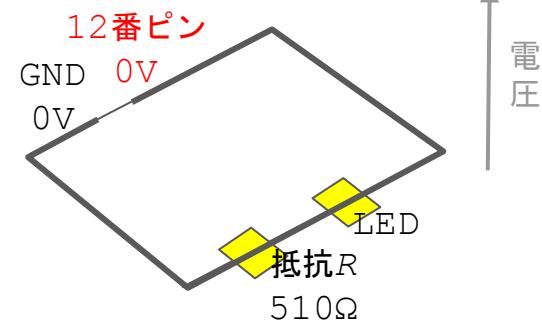
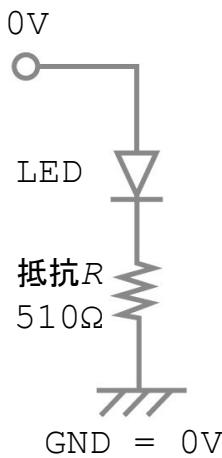
3-6. 電子回路解説



12番ピンがHIGHのとき



12番ピンがLOWのとき



3-6. 演習①

1. 13番ピンにLEDをつなぎかえて点滅させてみましょう
2. 点滅の周期を1000ミリ秒から100ミリ秒に改変して動かしてみましょう

4. Learn(応用) : タクトスイッチ

4-1. 部品紹介：タクトスイッチ

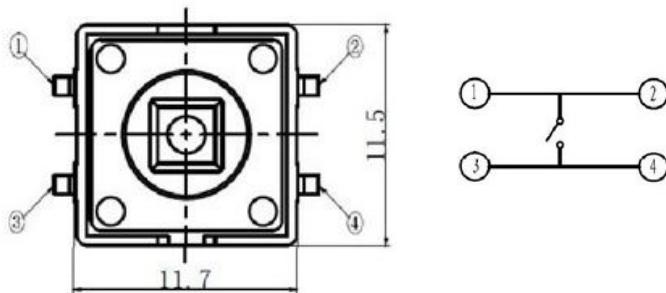


タクトスイッチ

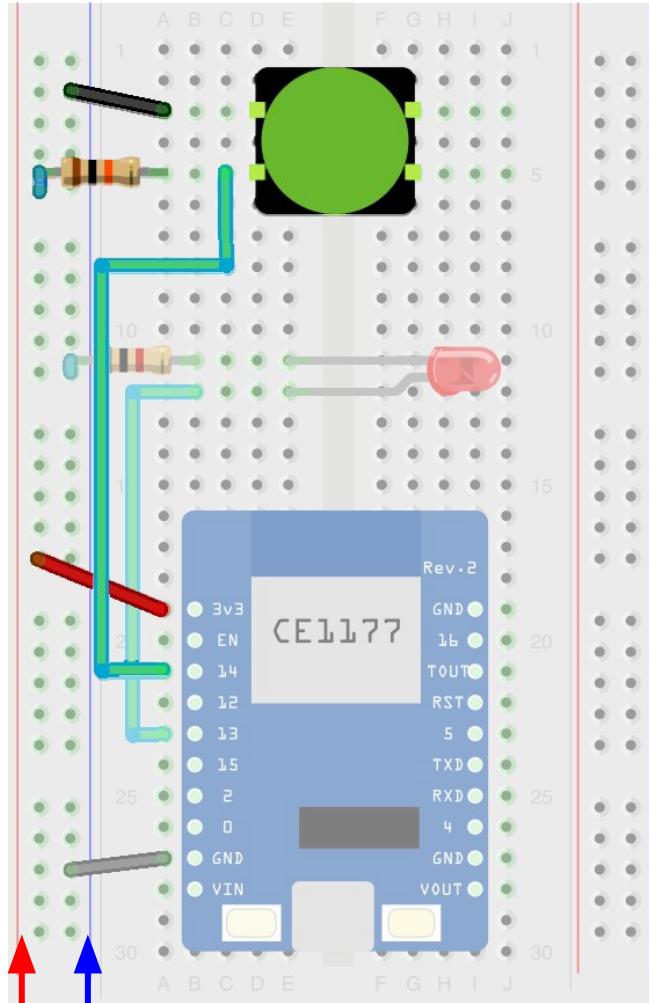
タクトスイッチは押すことで中の金属片がつながるスイッチです。

裏側の溝を隔てて、両端のピンは通常絶縁されており、スイッチを押すことでつながります。

IoT的な用途としては、ボタンを押すとTwitterにつぶやく、入退室を告げて、ウェブブラウザ上で認識できるなどが考えられます。



4-2. 電子回路



電子部品を配線する

1. タクトスイッチ : D3, D5, G3, G5
2. 抵抗10KΩ : A5 → 赤ライン
3. ジャンパー(赤) : A19 → 赤ライン
4. ジャンパー(黒) : A3 → 青ライン
5. ジャンパー(緑) : A21 → C5
6. ESPr : VIN → B28, VOUT → I28
7. LED : 長い足 → E12
8. LED : 短い足 → E11
9. 抵抗1K : B11 → 青ライン
10. ジャンパー(黒) : A27 → 青ライン
11. ジャンパー(緑) : A12 → A23

4-3. プログラム作成

```
1 // File : Button.ino
2 #define BTN 14
3 #define LED 13
4
5 void setup() {
6     pinMode(LED, OUTPUT);
7     pinMode(BTN, INPUT);
8     Serial.begin(9600);
9     digitalWrite(LED, LOW);
10 }
11
12 void loop() {
13     int state = digitalRead(BTN);
14     if(state==LOW) {
15         Serial.println("Turn on LED");
16         digitalWrite(LED, HIGH);
17     } else {
18         digitalWrite(LED, LOW);
19     }
20     delay(100);
21 }
```

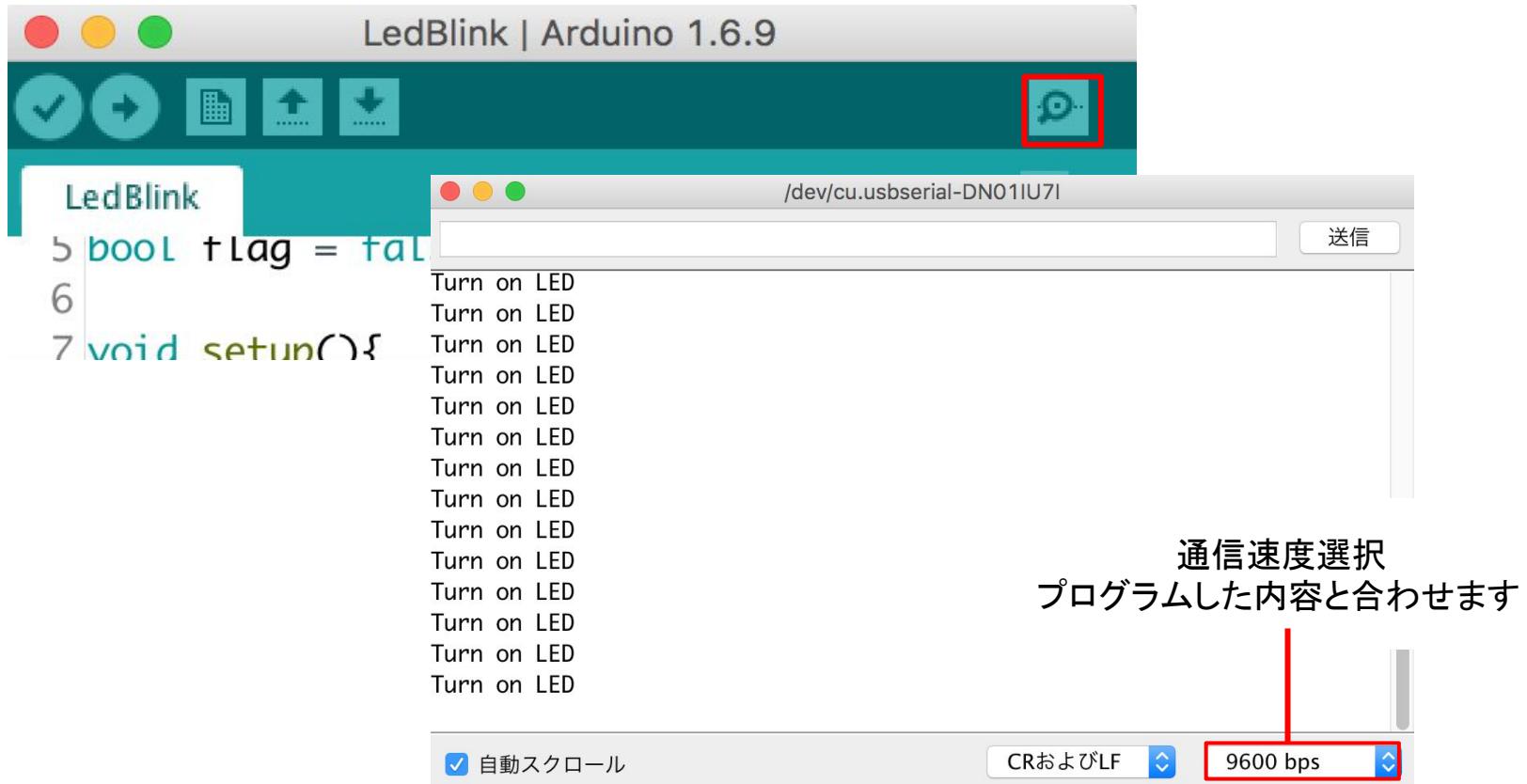
1. プログラムを記述する
2.  「チェックアイコン」をクリックしてプログラムをコンパイルする
3.  「矢印アイコン」をクリックしてプログラムをESP8266にアップロードする

4-4. シリアルモニター

マイクロコントローラーからシリアル通信で送られてくるデータを表示するためのモニター。

 ドットを押すと別のウィンドウが開きます。

今回のプログラムでは「Turn on LED」という文字列が表示されます。



LedBlink | Arduino 1.6.9

LedBlink

```
5 bool tFlag = false;
6
7 void setup() {

```

/dev/cu.usbserial-DN01IU71

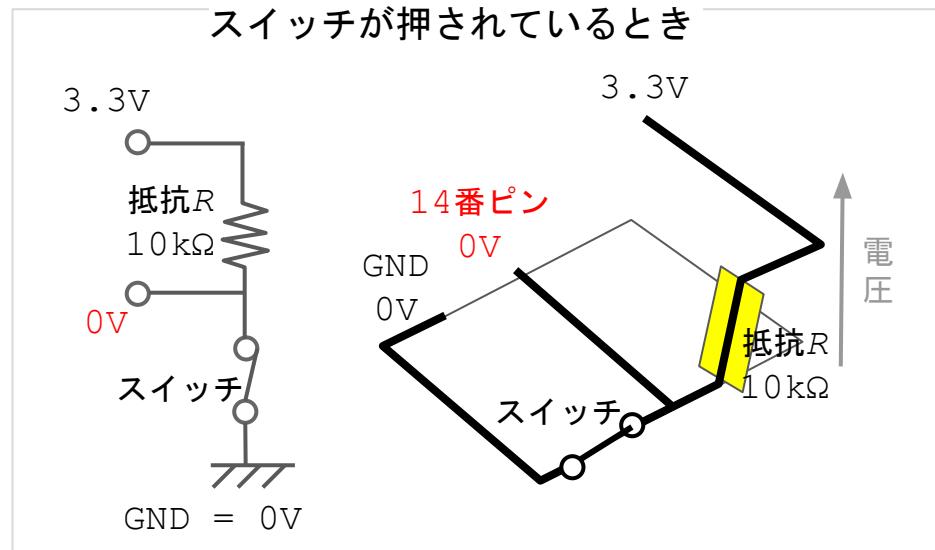
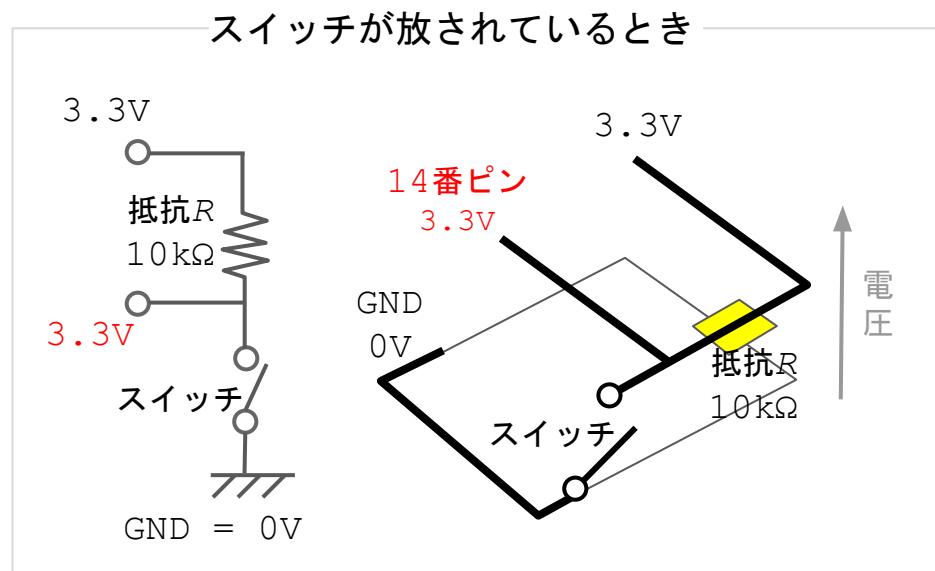
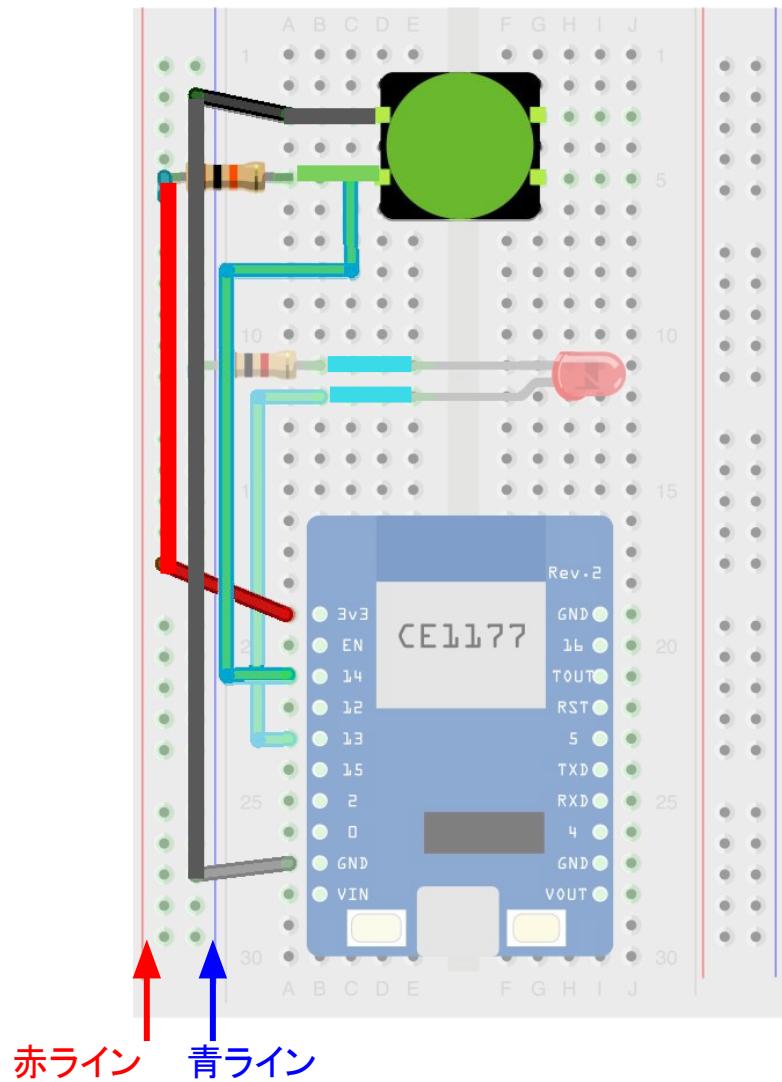
Turn on LED
Turn on LED

通信速度選択
プログラムした内容と合わせます

9600 bps

自動スクロール

4-5. 電子回路解説



4-6. プログラム解説①

```
1 // File : Button.ino
2 #define BTN 14
3 #define LED 13
4
5 void setup() {
6     pinMode(LED, OUTPUT);
7     pinMode(BTN, INPUT);
8     Serial.begin(9600);
9     digitalWrite(LED, LOW);
10 }
11
12 void loop() {
13     int state = digitalRead(BTN);
14     if(state==LOW) {
15         Serial.println("Turn on LED");
16         digitalWrite(LED, HIGH);
17     } else {
18         digitalWrite(LED, LOW);
19     }
20     delay(100);
21 }
```

Serial.begin(int value);
指定した通信速度でシリアル通信が行えるようになります。
通信速度はいくつか選択することができ、9600, 115200 など決まった値を第一引数に入力します。

Serial.println(String);
指定した文字列をシリアル通信で送信することができます。
Arduino環境では、文字列はダブルクオーテーションで囲みます。

==
プログラムでイコールを2つ書くと左辺と右辺が同じという意味になります。
イコールが1つだと代入になるので気をつけましょう。

4-6. プログラム解説②

```
1 // File : Button.ino
2 #define BTN 14
3 #define LED 13
4
5 void setup() {
6     pinMode(LED, OUTPUT);
7     pinMode(BTN, INPUT);
8     Serial.begin(9600);
9     digitalWrite(LED, LOW);
10 }
11
12 void loop() {
13     int state = digitalRead(BTN);
14     if(state==LOW) {
15         Serial.println("Turn on LED");
16         digitalWrite(LED, HIGH);
17     } else {
18         digitalWrite(LED, LOW);
19     }
20     delay(100);
21 }
```

digitalRead()

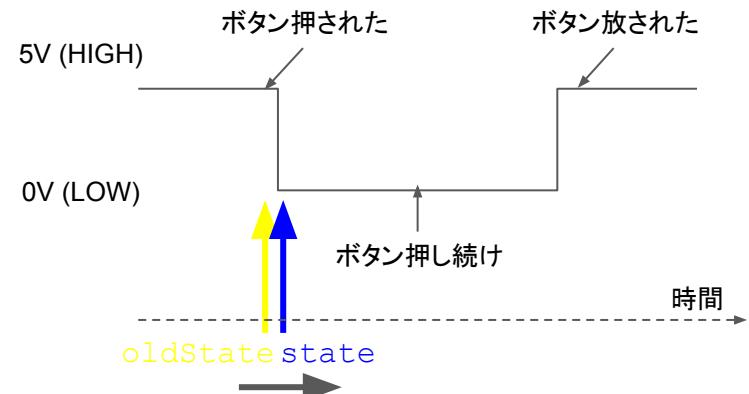
pinMode()でインプットに指定したピンの状態を読みます。

例)

```
int state =
digitalRead(ledPin);
ピン(ledPin)の状態(HIGH or
LOW)を取得し、変数stateに入れ
ます。
```

4-7. プログラム改良(スイッチを押した瞬間だけLEDが点灯)

```
1 // File : Button2.ino
2 #define BTN 14
3 #define LED 13
4 int oldState = HIGH;
5
6 void setup() {
7     pinMode(LED, OUTPUT);
8     pinMode(BTN, INPUT);
9     Serial.begin(9600);
10    digitalWrite(LED, LOW);
11 }
12
13 void loop() {
14     int state = digitalRead(BTN);
15     if(state==LOW && oldState==HIGH) {
16         Serial.println("Turn on LED");
17         digitalWrite(LED, HIGH);
18     } else {
19         digitalWrite(LED, LOW);
20     }
21     oldState = state;
22     delay(100);
23 }
```



| | | oldState | |
|-------|------|----------------|--------|
| | | HIGH | LOW |
| | | HIGH | 何もしない |
| state | HIGH | | 放された瞬間 |
| | LOW | HIGH 押された瞬間 | 押し続け |

4-8. 論理演算子

&& AND演算子

左辺・右辺の2つの値がどちらもtrueのときにtrueとなる。

例)

(1==1 && 2==2) → 真 (true)

(1==0 && 2==2) → 偽 (false)

|| OR演算子

左辺・右辺の2つの値のどちらか一方でもtrueならばtrueとなる。

例)

(1==1 || 2==2) → 真 (true)

(1==0 || 2==2) → 真 (true)

4-9. 演習②

2つのLED(LED_A, LED_B)とスイッチを用意します。

スイッチを押しているときは、LED_Aが点灯、LED_Bは消灯

スイッチを離すとLED_Aが消灯、LED_Bが点灯

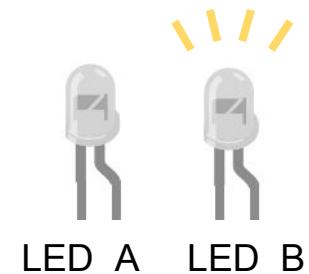
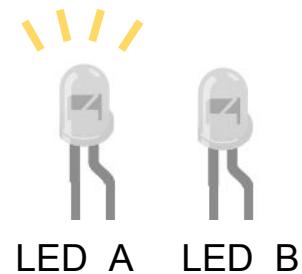
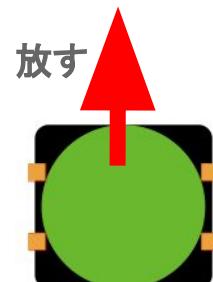
このような動きをするプログラムと回路を作成してみましょう。

用意するもの

- LED x 2
- 抵抗(510Ω) x 2
- スイッチ x 1

ヒント

P.32のプログラムが参考になります。



Lunch Break

5. Learn(応用)：測距/環境光センサー

5-1. 部品紹介 測距 / 環境光センサー (VCNL4010)



仕様

電圧 : 3.3v ~ 5.0v

測定距離範囲 : 1 ~ 200 mm

測定光度範囲 : 10 ~ 16383 Lux

ピンアサイン

Vin : 電源 +

GND : 電源 -

SDA : I2C信号線

SCL : I2Cクロック線

VCNL4010を搭載したセンサモジュールです。距離を測定する機能と環境光を測定する機能を持っています。マイコンとの通信は、I2Cで行います。

測距センサは10~150 mmほどの距離の測定が可能です。比較的近距離の測定に向いているので、手をかざしたかどうかの判定やロボットの壁との衝突判定などに向いています。

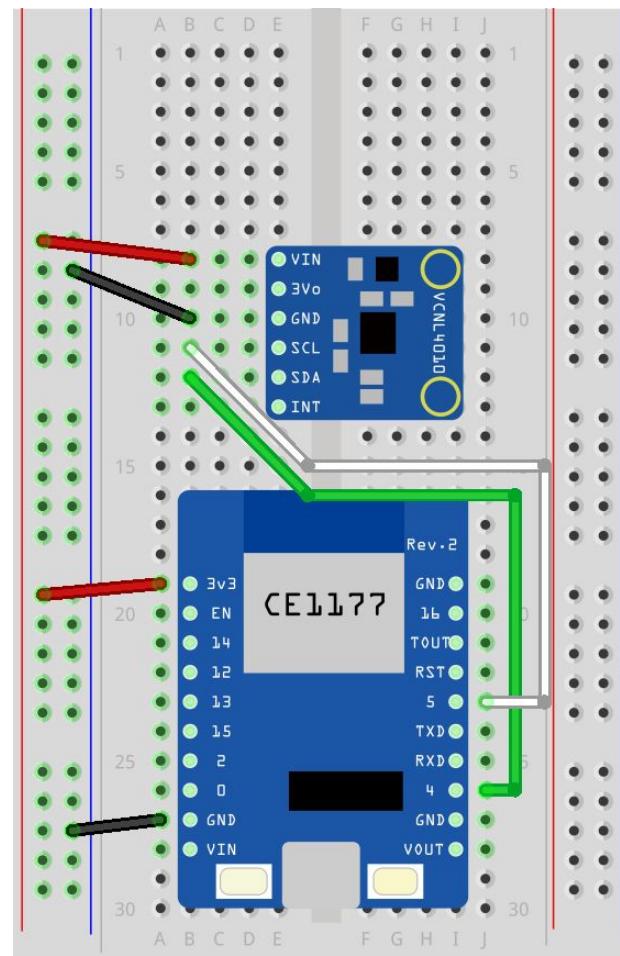
近距離センサだけでなく環境光センサも内蔵しているので、周囲の明るさをLux値で取得することができます。

<https://www.switch-science.com/catalog/2640/>

5-2. 電子回路

電子部品を配線する

1. ESPr : VIN -> B28, VOUT -> I28
2. VCNL4010 -> VIN -> E8, INT -> E13
3. ジャンパー(赤)① : B8 -> 赤ライン
4. ジャンパー(赤)② : A19 -> 赤ライン
5. ジャンパー(黒)① : B10 -> 青ライン
6. ジャンパー(黒)② : A27 -> 青ライン
7. ジャンパー(白)① : B11 -> J23
8. ジャンパー(緑)② : B12 -> J26



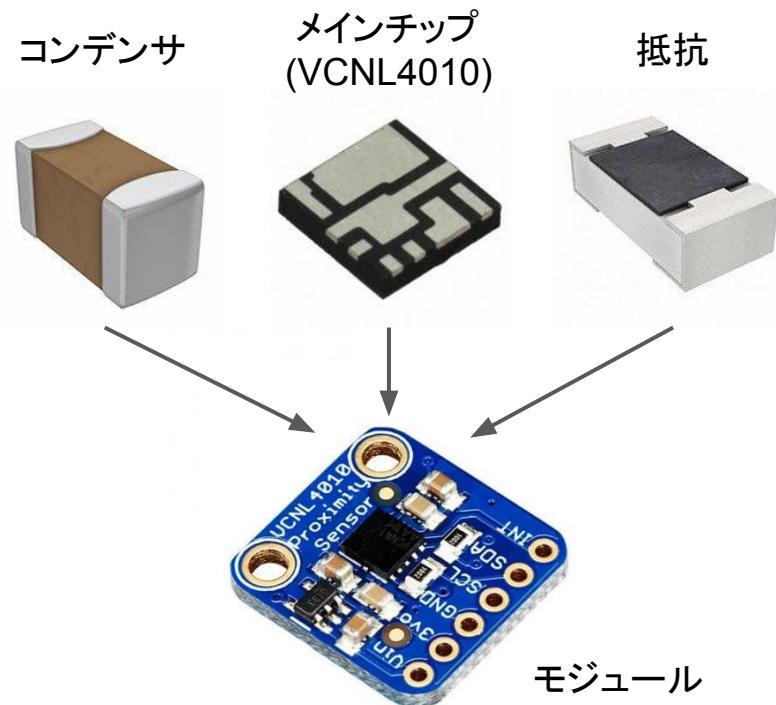
5-3. モジュールとライブラリー

モジュール（ハードウェア）

中心となるチップと、抵抗、コンデンサ等の部品をまとめて、扱いやすくしたもの。

ライブラリー（ソフトウェア）

機能や用途で分類し、集められた汎用性のあるプログラム群。



Adafruit

<https://www.adafruit.com>

電子工作を手軽に始めるためのモジュール等を販売している米国の会社。

Adafruitのモジュールは扱いやすいライブラリーがセットになっています。

直接輸入も出来ますが、日本では主にスイッチサイエンスで扱っています。

スイッチサイエンス

<https://www.switch-science.com>

5-4. ライブラリー追加

- 「スケッチ」>「ライブラリをインクルード」>「ライブラリを管理」を選択します。



- 「VCNL」と検索し、「FaBo 205 Proximity VCNL4010 by FaBo」というライブラリを選択して、インストールします。



5-5. プログラム作成

```
1 // Prox.ino
2 #include <Wire.h>
3 #include <FaBoProximity_VCNL4010.h>
4
5 FaBoProximity fabo;
6
7 void setup() {
8     Serial.begin(9600);
9
10    fabo.begin();
11 }
12
13 void loop() {
14     // 距離を測定
15     if(fabo.checkProxReady()) {
16         int value = fabo.readProx();
17         Serial.print("prox:");
18         Serial.println(value);
19     }
20     delay(1000);
21 }
```

5-6. プログラム解説① 距離を測定

```
1 // Prox.ino
2 #include <Wire.h>
3 #include <FaBoProximity_VCNL4010.h>
4
5 FaBoProximity fabo;
6
7 void setup() {
8     Serial.begin(9600);
9
10    fabo.begin();
11}
12
13 void loop() {
14     // 距離を測定
15     if(fabo.checkProxReady()) {
16         int value = fabo.readProx();
17         Serial.print("prox:");
18         Serial.println(value);
19     }
20     delay(1000);
21 }
```

FaboProximity fabo;

距離センサを扱うためのライブラリ
FaboProximity_VCNL4010.h を利用する
ために fabo というオブジェクトを生成します。

fabo は別の名前でもかまいません。

fabo.begin();

距離センサを使うための初期設定をします。

FaBoRroximity_

fabo.checkProxReady()

距離センサからの値を読める状態かどうか
チェックします。

fabo.readProx()

距離センサの値を取得します。

5-6. プログラム解説② 環境光を測定

```
1 // Ambi.ino
2 #include <Wire.h>
3 #include <FaBoProximity_VCNL4010.h>
4
5 FaBoProximity fabo;
6
7 void setup() {
8     Serial.begin(9600);
9
10    fabo.begin();
11 }
12
13 void loop() {
14     // 環境光を測定
15     if(fabo.checkAmbiReady()) {
16         int value = fabo.readAmbi();
17         Serial.print("Ambi:");
18         Serial.println(value);
19     }
20     delay(1000);
21 }
```

fabo.checkAmbiReady()

距離センサからの値を読める状態かどうか
チェックします。

fabo.readAmbi()

環境光を測定する関数です。
測定して表示される値はルクスになります。

5-7. 演習③

明るさ、距離のどちらかが、ある一定の値（自分で値を決めましょう）を超えたらシリアルモニターにお知らせを表示するプログラムを作ってみましょう。

ヒント：if文を使う

6. **Learn**(応用) : サーボモーター

6-1. 部品紹介 サーボモーター SG90



小型のデジタルマイクロサーボモーターです。

[アマゾン](#)で購入可能です。

仕様

トルク : 1.8kg/cm (4.8v)

回転速度 : 100msec/60度 (4.8v)

ピンアサイン

赤 : 電源 +

茶 : 電源 -

橙 : 信号線

6-2. 電子回路

電子部品を配線する

ESPr : VIN → B28, VOUT → I28

サーボモーター(オレンジ) → A23

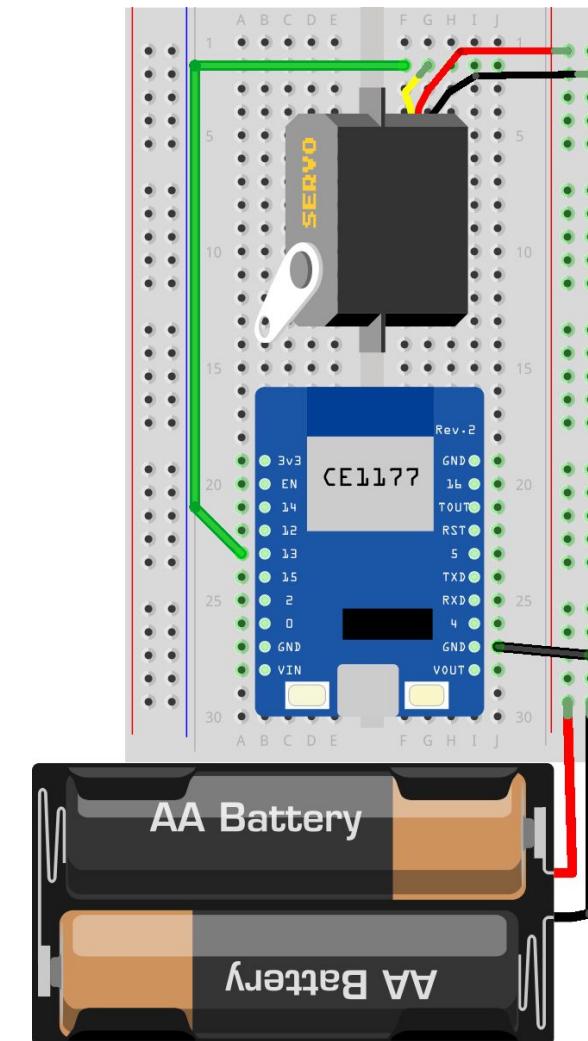
サーボモーター(赤) → 赤ライン

サーボモーター(黒) → 青ライン

電池ボックス(赤) → 赤ライン

電池ボックス(黒) → 青ライン

ジャンパー(黒) : J27 → 青ライン



6-3. プログラム

```
1 // Servo.ino
2 #include <Servo.h>
3
4 Servo myservo;
5
6 void setup() {
7     Serial.begin(9600);
8     myservo.attach(13);
9     myservo.write(90);
10 }
11
12 void loop() {
13     char s = Serial.read();
14     if(s=='a'){
15         myservo.write(20);
16     } else if(s=='s'){
17         myservo.write(90);
18     } else if(s=='d'){
19         myservo.write(160);
20     }
21 }
22
23
24
25 }
```

Servo.h

Arduinoでサーボモータを扱うための
ライブラリ

Servo myservo

Servo.hライブラリを利用するための
オブジェクト

myservo.attach(13);

指定したピンをサーボモータ用に割り
あてます。

myservo.write(90);

指定した角度にサーボモータを動かし
ます。SG90は20~160度の範囲で使用
してください。

Serial.read();

シリアル通信で信号を受信する関数。
今回はパソコンからの入力を受け付け
ます。

7. Make

[Learn](#)のセッションで学んだLED、スイッチ、環境光/測距センサ、サーボモーターを組み合わせて、プロトタイプを作ってみましょう。

8. 部品の購入先

国内

スイッチサイエンス

<https://www.switch-science.com>

千石電商

<http://www.sengoku.co.jp>

秋月電子

<http://akizukidenshi.com/catalog/default.aspx>

国外

Adafruit

<https://www.adafruit.com>

Digikey

<http://www.digikey.jp>

Mouser

<http://www.mouser.jp/?gclid=CODS4cLH2c4CFUMAvAodaJQAhQ>