

IoT Basic Weekend

DAY2: Communication

1. IoT Basic
 - 1-1. 講座の流れ
 - 1-2. DAY2 Communication
2. Learn : 基礎 WiFiに接続
 - 2-1. インターネット
 - 2-2. LANに接続
 - 2-3. 電子回路
 - 2-4. プログラム
 - 2-5. シリアルモニター
 - 2-6. プログラム解説
3. Learn : 基礎 WiFiでLED制御
 - 3-1. サーバーとクライアント
 - 3-2. URI
 - 3-3. ポート
 - 3-4. HTTP通信
リクエストとレスポンス
 - 3-5. 電子回路
 - 3-6. プログラム
 - 3-7. プログラム解説
 - 3-8. HTML
 - 3-9. プログラム改変
 - 3-10. 電池で動かしてみましょう3-11.
演習①
4. Learn : 応用 WiFiでサーボモーター制御
 - 4-1. サーボモーター SG90
 - 4-2. 電子回路
 - 4-3. プログラム
 - 4-4. プログラム実行
 - 4-5. プログラム解説
5. Learn : 応用 環境光センサーの値をブラウザでモニタリング
 - 5-1. 測距 / 環境光センサー VCNL4010
 - 5-2. 電子回路
 - 5-3. プログラム
 - 5-4. プログラム実行
 - 5-5. プログラム解説
6. Learn : 応用 センサーのデータをサーバーに送る
 - 6-1. ThingSpeakを利用する
 - 6-2. ThingSpeak : 概要
 - 6-3. ThingSpeak : アカウント作成
 - 6-4. ThingSpeak : チャンネル作成
 - 6-5. ThingSpeak : API確認
 - 6-6. 電子回路
 - 6-7. プログラム
 - 6-8. プログラム解説
 - 6-9. 演習②
7. Make
8. 参考ウェブサービス



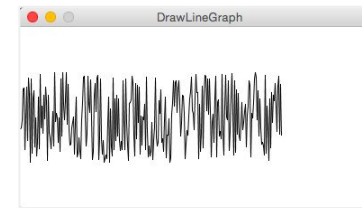
1-1. 講座の流れ

DAY1 : I/O – インput、アウトput –

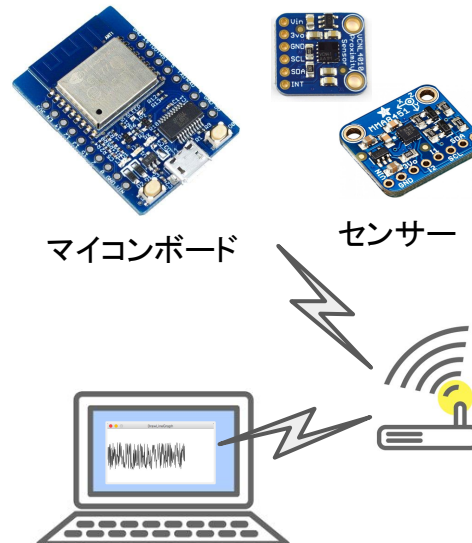
マイコンボードとI/Oデバイス(各種センサー、LED、モーターなど)の扱い方を学びます。

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



プログラムの書き方

```
DrawLineGraph
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);
```

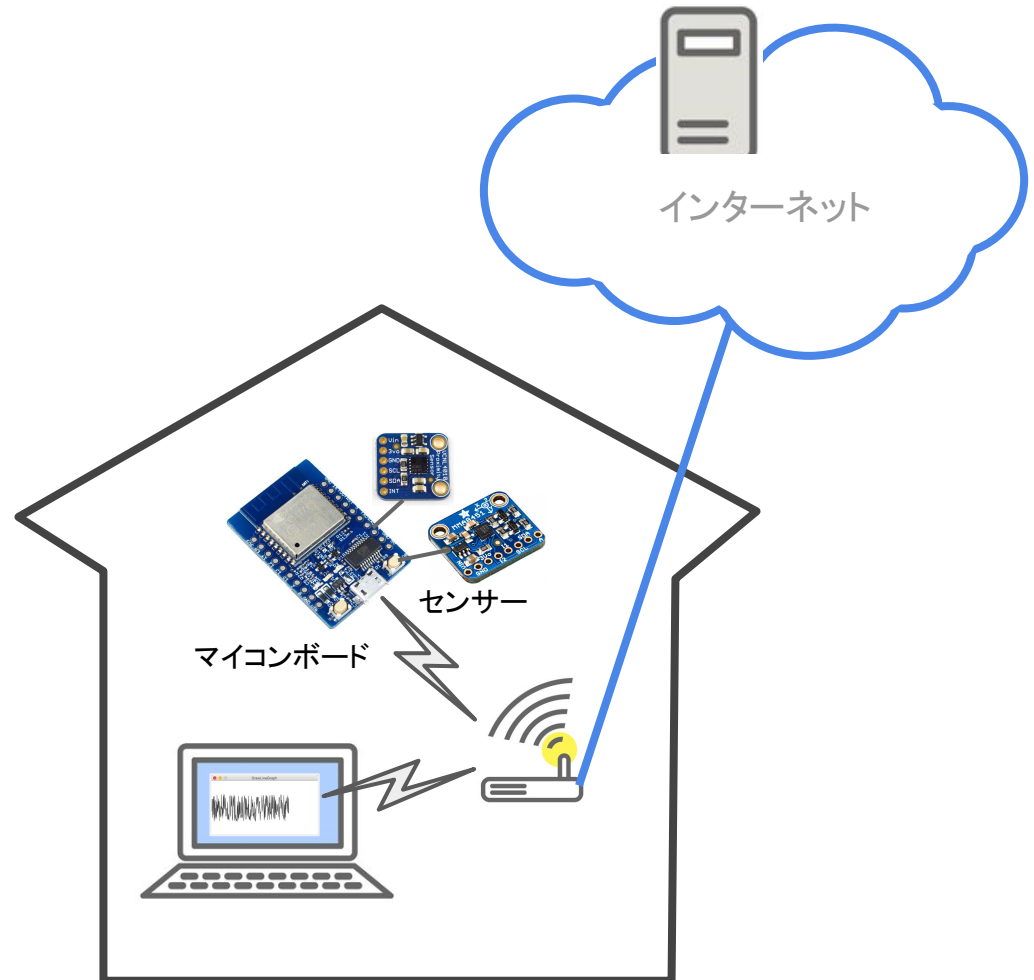
```
LedBlink
1 int spd = 1000;
2 void setup() {
3   // ESPの13番ピンを出力に設定
4   pinMode(13, OUTPUT);
5 }
6
7 void loop() {
8   digitalWrite(13, HIGH);
9   delay(1000);
10  digitalWrite(13, LOW);
11  delay(1000);
12 }
```

1-2. DAY2 Communication - 通信 -

Communication - 通信 - では、センサデバイスとPCを通信させる方法について学びます。

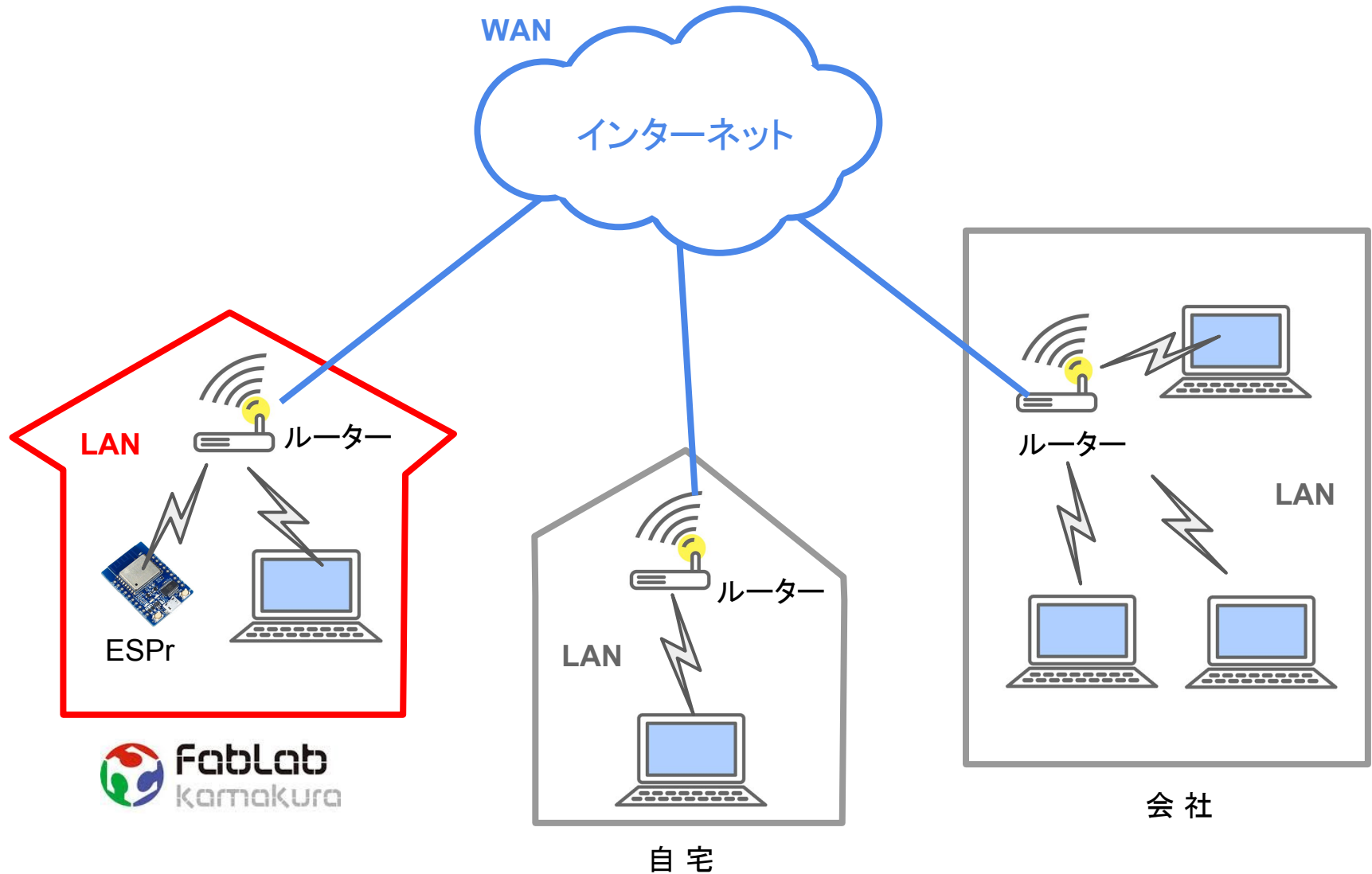
DAY1で学んだProcessing、DAY2で学んだ各種センサを組み合わせ、センサで取得した情報をPCに送信し、画面上でアニメーションさせてみましょう。

インターネットを利用して、外部のサーバーと通信し、遠く離れた場所で情報をやりとりする方法についても学んでいきます。



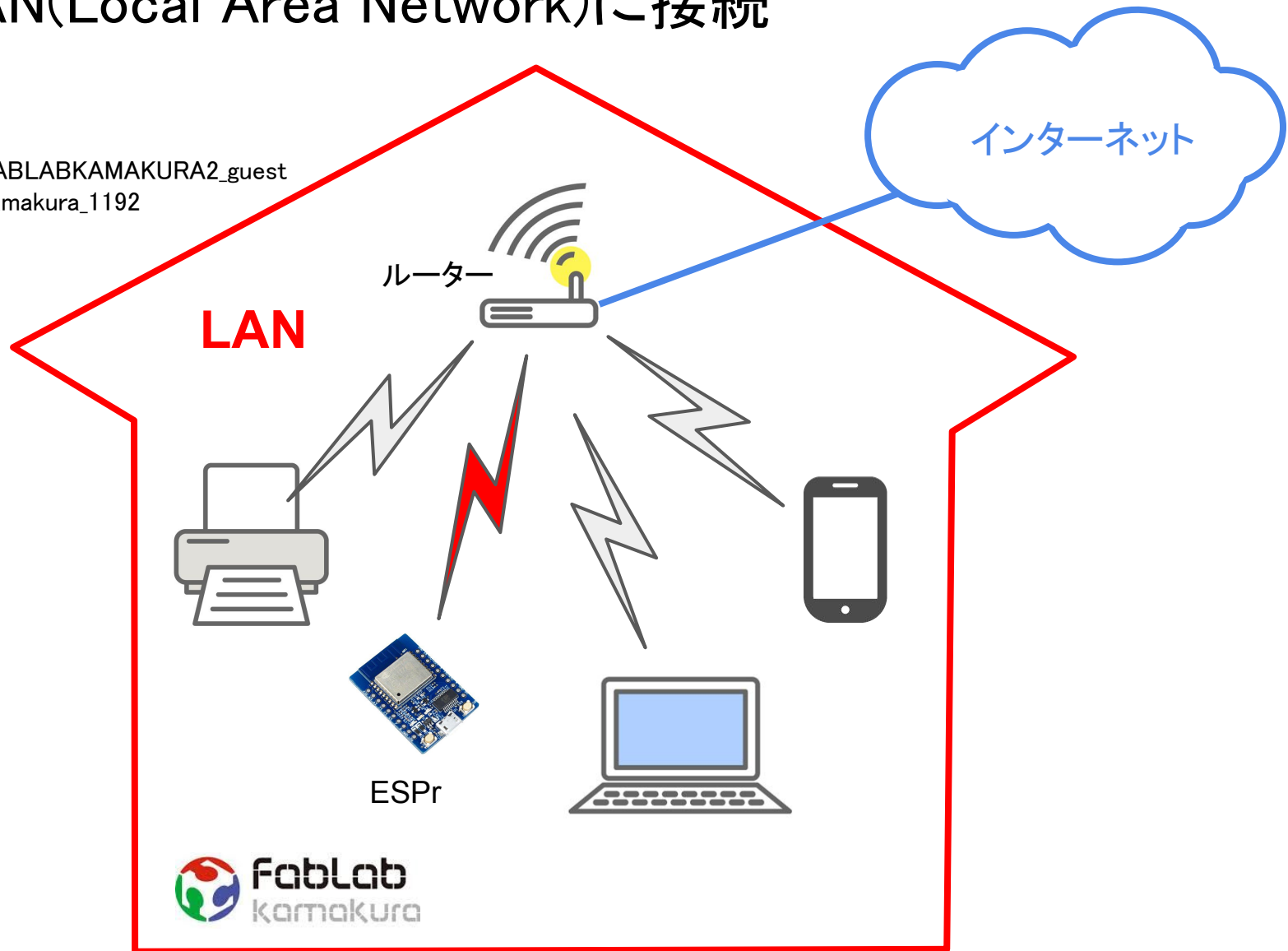
2. **Learn**(基礎) : WiFiに接続

2-1. インターネット



2-2. LAN(Local Area Network)に接続

SSID: FABLABKAMAKURA2_guest
PASS: kamakura_1192

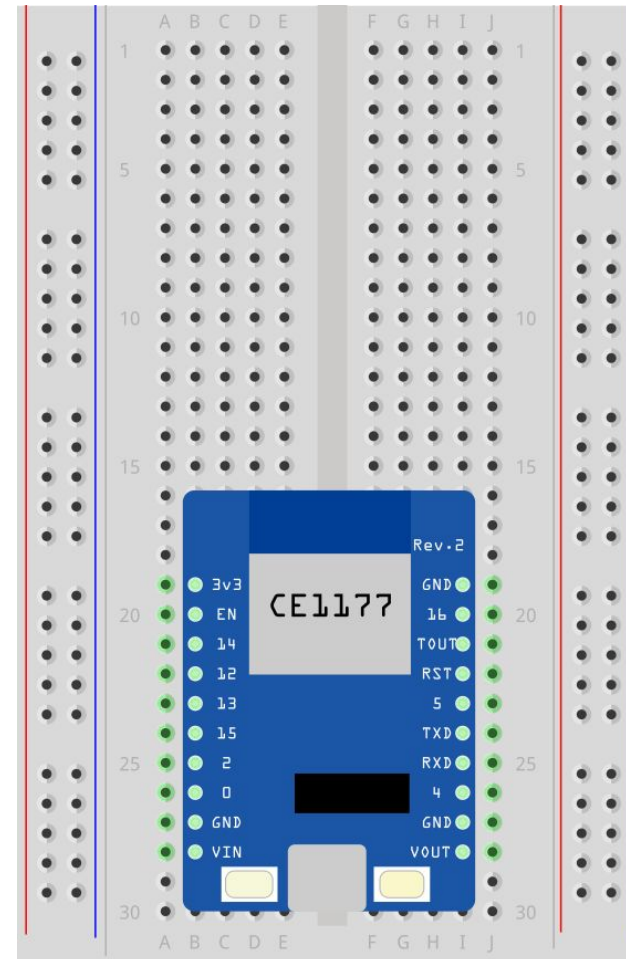
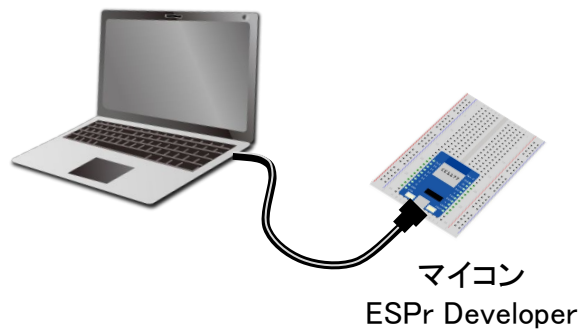


2-3. 電子回路

電子部品を配線する

1. ESPr : VIN -> B28, VOUT -> I28


USBケーブルでマイコンボード ESPr と PC
を接続する




2-4. プログラム

```
1  #include <ESP8266WiFi.h>
2
3  #define SSID "FABLABKAMAKURA2_guest"
4  #define PASS "kamakura_1192"
5
6  void setup() {
7      Serial.begin(9600);
8      WiFi.begin(SSID, PASS);
9      Serial.println();
10     while(WiFi.status() != WL_CONNECTED) {
11         delay(500);
12         Serial.print(".");
13     }
14     Serial.println();
15     Serial.println("WiFi connected");
16     Serial.println(WiFi.localIP());
17 }
18
19 void loop() {
20 }
```

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ESPにアップロードする

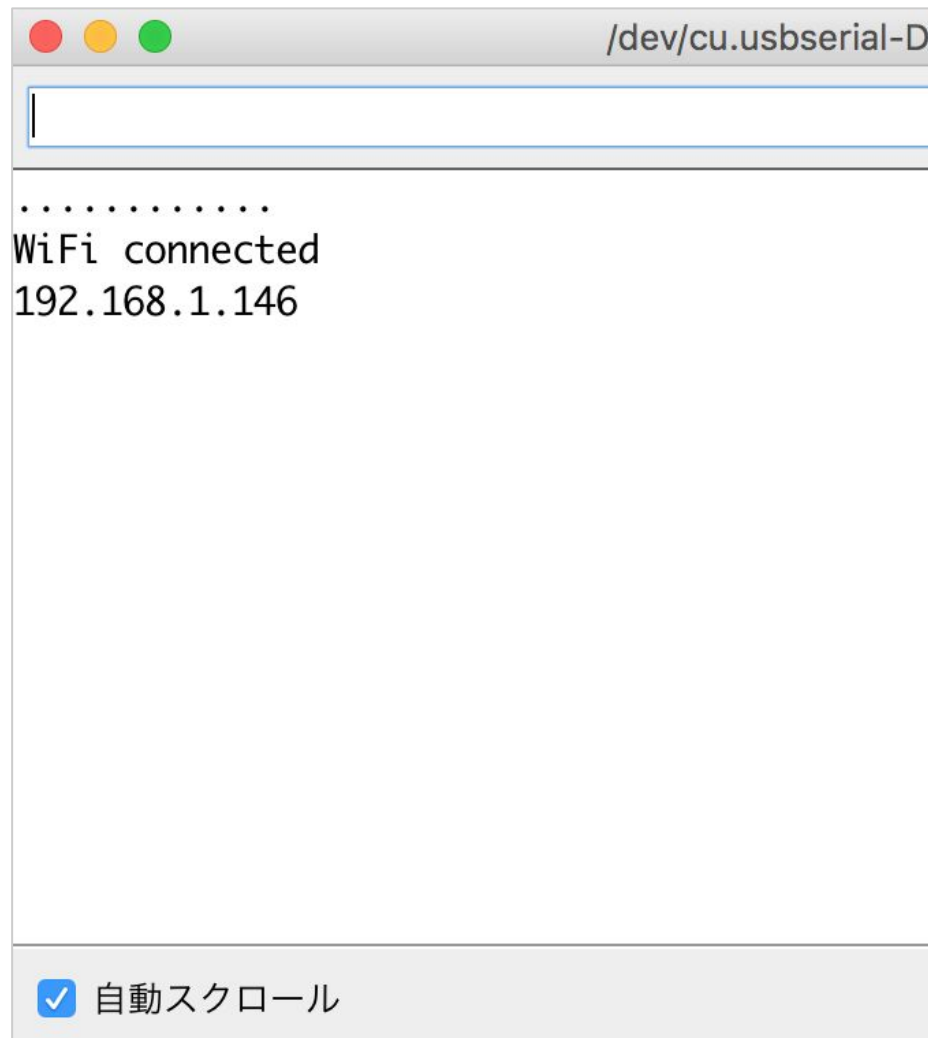
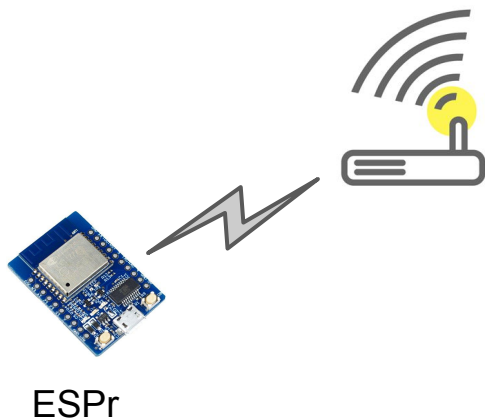
2-5. シリアルモニター



「虫眼鏡アイコン」をクリックして、シリアルモニターを表示しましょう。

「WiFi connected」と IPアドレスが表示されれば成功です。

特に何もしませんが、ESP8266はネットワークに接続されています。



2-6. プログラム解説

```
1 #include <ESP8266WiFi.h>
2
3 #define SSID "FABLABKAMAKURA2_guest"
4 #define PASS "kamakura_1192"
5
6 void setup() {
7     Serial.begin(9600);
8     WiFi.begin(SSID, PASS);
9     Serial.println();
10    while(WiFi.status() != WL_CONNECTED){
11        delay(500);
12        Serial.print(".");
13    }
14    Serial.println();
15    Serial.println("WiFi connected");
16    Serial.println(WiFi.localIP());
17 }
18
19 void loop() {
20 }
```

ESP8266WiFi.hでWiFiを用いた通信を行なうためのライブラリです。

#defineを使うと、以降のプログラム内の1つ目の文字列(例 SSID)は2つ目の文字列(例 FABLABKAMAKURA2_guest)に自動的に置き換えられます。

設定したSSID、パスワードでWiFiに接続処理を開始します。

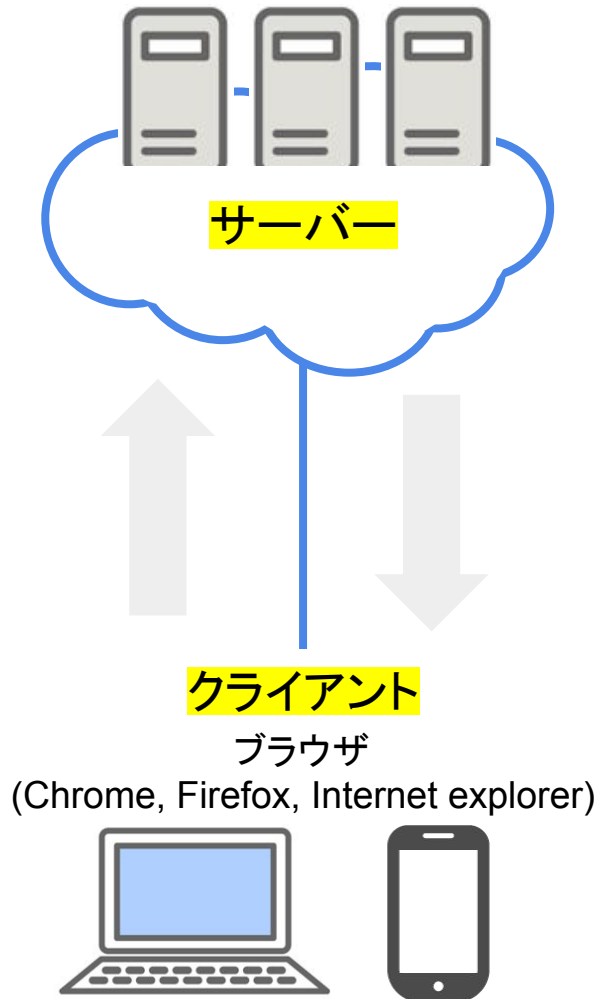
WiFi.status()で現在のWiFi接続の状態を取得し、WiFiに接続されている状態(WL_CONNECTED)ではないことをチェックします。

「!=」は左辺と右辺が異なる場合に「真」となり、接続されていない間はwhile文の中の処理が実行されます。

IPアドレスを返す関数です。

3. **Learn**(基礎) : WiFi経由でLED制御

3-1. サーバーとクライアント



クライアントからの要求に対して
何らかのサービスを提供するもの

例)

メールサーバー

ウェブサーバー

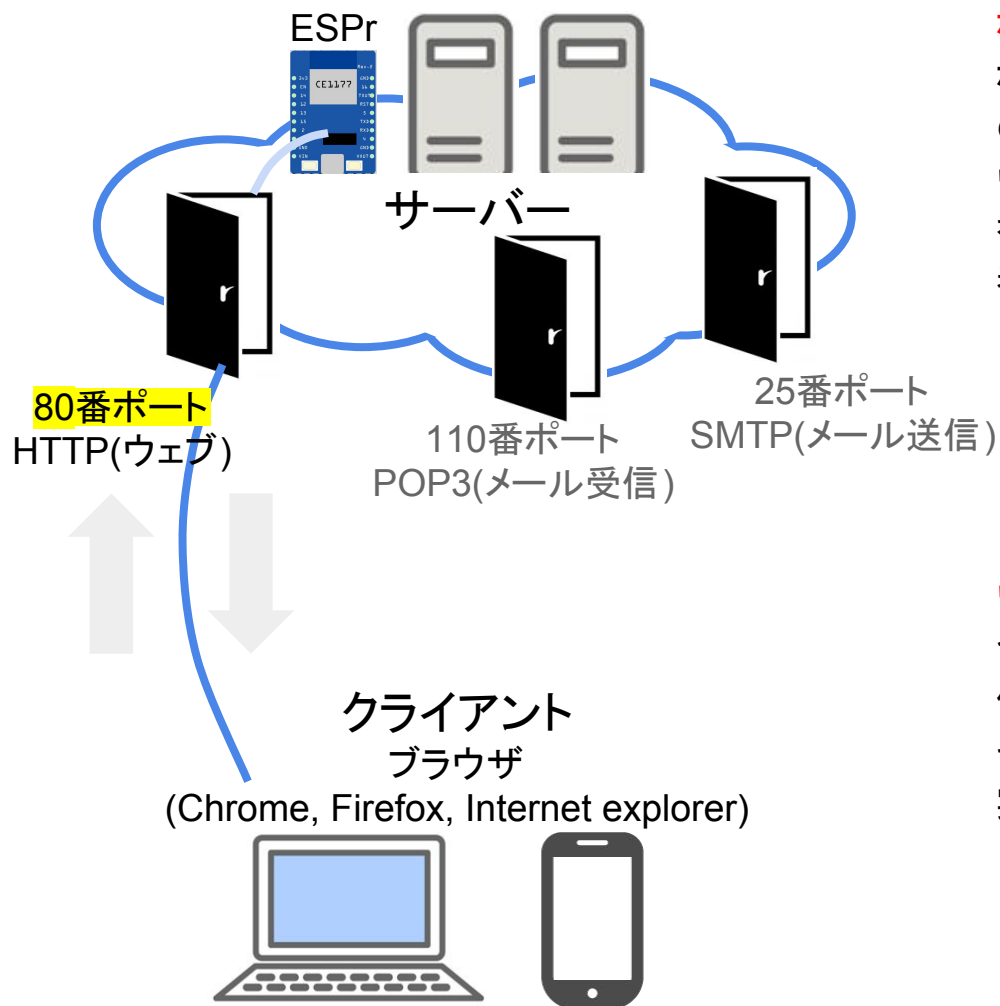
プリントサーバー

サーバーに対して
サービスの依頼を行い、
サービスの提供を受けるもの

3-2. URI



3-3. ポート



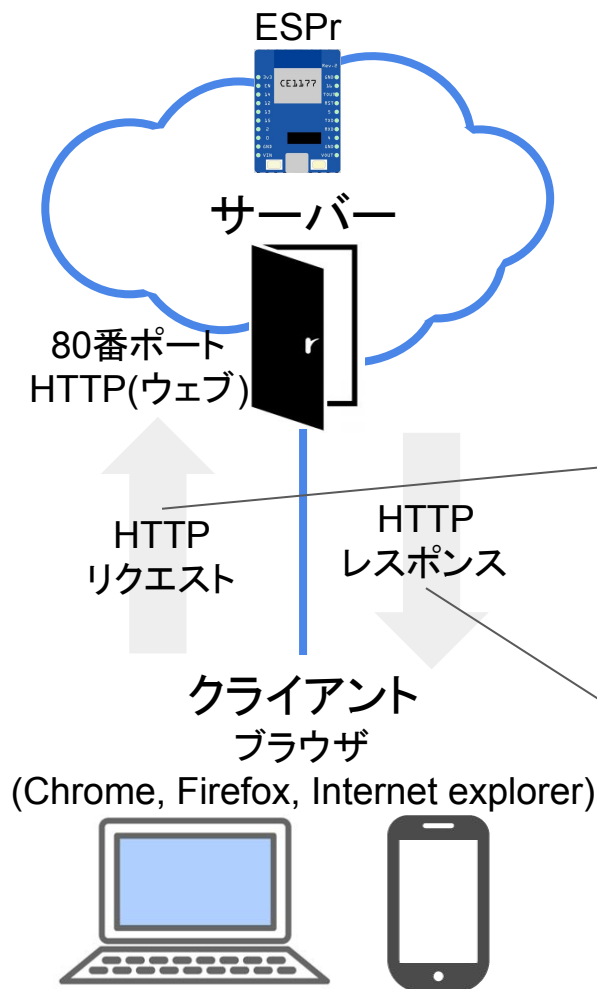
ポート

ポートはサービスごとに取り付けられた扉のようなものです。ウェブサービスであれば 80番のポート(扉)を使いますし、メールを送信するときは 25番ポート(扉)を使用します。

ウェブサービス(80番ポート)

今回はESPrボードをウェブサーバーとして使用します。そのため80番ポートを指定してサービスを実行します。

3-4. HTTP通信 リクエストとレスポンス



HTTP (Hyper Text Transfer Protocol)
ウェブブラウザとウェブサーバーの間でhtmlなどのコンテンツの送受信するためのプロトコルです。

HTTPリクエスト

(例)
<http://google.com>
<http://192.168.1.210:80>
GET / HTTP1.1
Host google.com

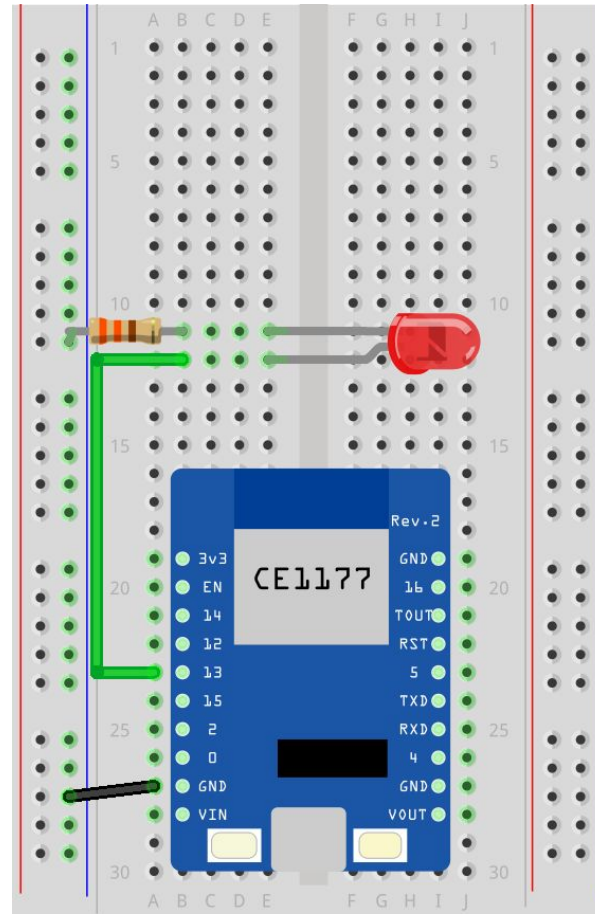
HTTPレスポンス

(例)
HTTP/1.1 200 OK
HTTP/1.1 404 NotFound

3-5.電子回路(LEDをON/OFFする)

電子部品を配線する

1. ESPr : VIN → B28, VOUT → I28
2. LED短い足 → E11
3. LED長い足 → E12
4. 抵抗 : B11 → 青ライン
5. ジャンパー(黒) : A27 → 青ライン
6. ジャンパー(緑) : A23 → B12




3-6.プログラム①(HTTPリクエストに応じてLEDをON/OFFする)

```
1  #include <ESP8266WebServer.h>
2
3  #define LED 13
4  #define SSID "FABLABKAMAKURA2_guest"
5  #define PASS "kamakura_1192"
6
7  ESP8266WebServer server(80);
8
9  void setup() {
10     Serial.begin(9600);
11     pinMode(LED, OUTPUT);
12     digitalWrite(LED, LOW);
13
14     connectWiFi();
15     server.on("/", handleRoot);
16     server.on("/on", switchOnLed);
17     server.on("/off", switchOffLed);
18     server.begin();
19     Serial.println("HTTP server started");
20 }
21
22 void loop() {
23     server.handleClient();
24 }
25
```

```
26 void connectWiFi() {
27     WiFi.begin(SSID, PASS);
28     Serial.println();
29     while(WiFi.status() != WL_CONNECTED) {
30         delay(500);
31         Serial.print(".");
32     }
33     Serial.println();
34     Serial.println("WiFi connected");
35     Serial.println(WiFi.localIP());
36 }
37
38 void handleRoot() {
39     server.send(200, "text/plain", "LED");
40 }
41 void switchOnLed() { // LED ON
42     digitalWrite(LED, HIGH);
43     server.send(200, "text/plain", "ON");
44     Serial.println("switch on led");
45 }
46 void switchOffLed() { // LED OFF
47     digitalWrite(LED, LOW);
48     server.send(200, "text/plain", "OFF");
49     Serial.println("switch off led");
50 }
```

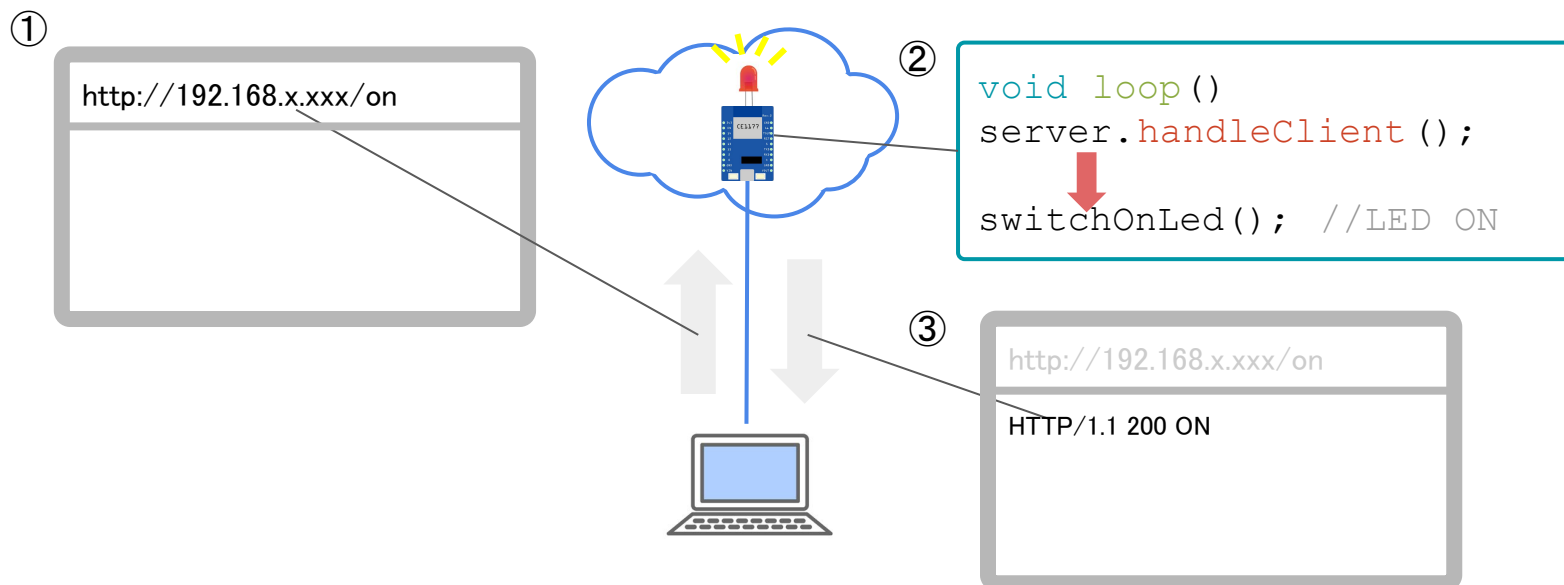
3-6. プログラム②(HTTPリクエストに応じてLEDをON/OFFする)

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

4. ブラウザ(クライアント)から①HTTPリクエストを送信し、②サーバー側 ESPrでリクエストが処理され、③HTTPレスポンスが返ってくる様子を確認する



3-7 プログラム解説①

```
1  #include <ESP8266WebServer.h>
2
3  #define LED 13
4  #define SSID "FABLABKAMAKURA2_guest"
5  #define PASS "kamakura_1192"
6
7  ESP8266WebServer server(80);
8
9  void setup() {
10     Serial.begin(9600);
11     pinMode(LED, OUTPUT);
12     digitalWrite(LED, LOW);
13
14     connectWiFi();
15     server.on("/", handleRoot);
16     server.on("/on", switchOnLed);
17     server.on("/off", switchOffLed);
18     server.begin();
19     Serial.println("HTTP server started");
20 }
21
22 void loop() {
23     server.handleClient();
24 }
25
```

ESP8266ボードをウェブサーバーにするためのライブラリです

80番ポートで待ち受けるウェブサーバーとして設定します

`server.on(アドレス, 関数名);`
第一引数で指定したアドレスにアクセスがあると、第二引数に指定した関数を呼出すように設定されます

`http://192.168.1.xxx/` → `handleRoot`
`http://192.168.1.xxx/on` → `switchOnLed`
`http://192.168.1.xxx/off` → `switchOffLed`

クライアント(ウェブブラウザ)からの入力を待つ待機状態にします

3-7. プログラム解説②

```
26 void connectWiFi() {
27     WiFi.begin(SSID, PASS);
28     Serial.println();
29     while(WiFi.status() != WL_CONNECTED) {
30         delay(500);
31         Serial.print(".");
32     }
33     Serial.println();
34     Serial.println("WiFi connected");
35     Serial.println(WiFi.localIP());
36 }
37 void handleRoot() {
38     server.send(200, "text/plain",
39     "LED");
40 }
41 void switchOnLed() { // LED ON
42     digitalWrite(LED, HIGH);
43     server.send(200, "text/plain", "ON");
44     Serial.println("switch on led");
45 }
46 void switchOffLed() { // LED OFF
47     digitalWrite(LED, LOW);
48     server.send(200, "text/plain",
49     "OFF");
50     Serial.println("switch off led");
51 }
```

クライアント(ウェブブラウザ)
からのリクエスト文字列

`http://192.168.1.xxx/` → `handleRoot`

`http://192.168.1.xxx/on` → `switchOnLed`

`http://192.168.1.xxx/off` → `switchOffLed`

`server.send`(ステータスコード, テキストの種類, テキスト);

どのようなレスポンスを返すかを3つの引数で指定し、クライアントに実際に返送します

「ステータスコード」とは、「404 Not Found」のような状態を表す3桁のコードです

「テキストの種類」`"text/plain"`、`"text/html"`のようなフォーマットを表す文字列です

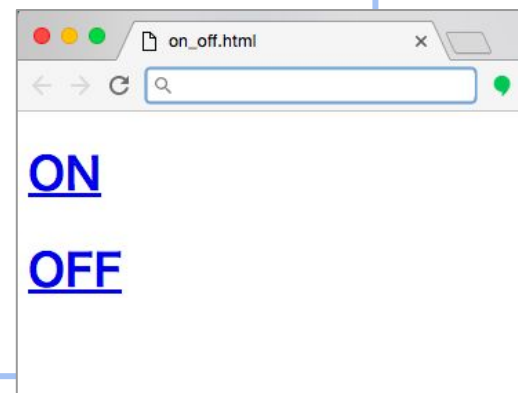
3-8. HTML (ハイパーテキスト マークアップ ランゲージ)

ウェブ上の文書を記述するための言語です。

文章の中に記述することでさまざまな機能を記述設定することができます。

例) LEDをON/OFFするリンクを表示するHTML

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <meta charset='UTF-8'>
6   <meta name='viewport' content='width=device-width'>
7 </head>
8
9 <body>
10  <h1><a href='/on'>ON</a></h1>
11  <h1><a href='/off'>OFF</a></h1>
12 </body>
13
14 </html>
```



ブラウザでウェブサイトを閲覧している時に右クリックして、“検証/Inspect”(chrome)を押すとページのHTML文章が見られます。他のブラウザでも同様の機能があるので試してみましょう。

3-9. プログラムの改変①

前ページと同じHTML形式の文字列 を返すプログラムに書き換えてみましょう。

```
1  #include <ESP8266WebServer.h>
2
3  #define LED 13
4  #define SSID "FABLABKAMAKURA2_guest"
5  #define PASS "kamakura_1192"
6
7  ESP8266WebServer server(80);
8  char* html = "<!DOCTYPE html>\n
  <head>\n
    <meta charset='UTF-8'>\n
    <meta name='viewport' content='width=device-width'>\n
  </head>\n
  <body>\n
    <h1><a href='/on'>ON</a></h1>\n
    <h1><a href='/off'>OFF</a></h1>\n
  </body>\n
  </html>";
9
10 void setup() {
    Serial.begin(9600);
    :
    :
```

char*型は文字列を格納するために用います。

使用する文字を決める

コンテンツのサイズをデバイスの幅に合わせる。スマートフォンでも適切な幅で見ることができます。

'/on'、'/off'へのリンクを作成します。

3-9. プログラムの改変②

```
37 :
38 void handleRoot() {
39     server.send(200, "text/html", html);
40 }
41 void switchOnLed() { // LED ON
42     digitalWrite(LED, HIGH);
43     server.send(200, "text/html", html);
44     Serial.println("switch on led");
45 }
46 void switchOffLed() { // LED OFF
47     digitalWrite(LED, LOW);
48     server.send(200, "text/html", html);
49     Serial.println("switch off led");
50 }
```



1. プログラムを改変する

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

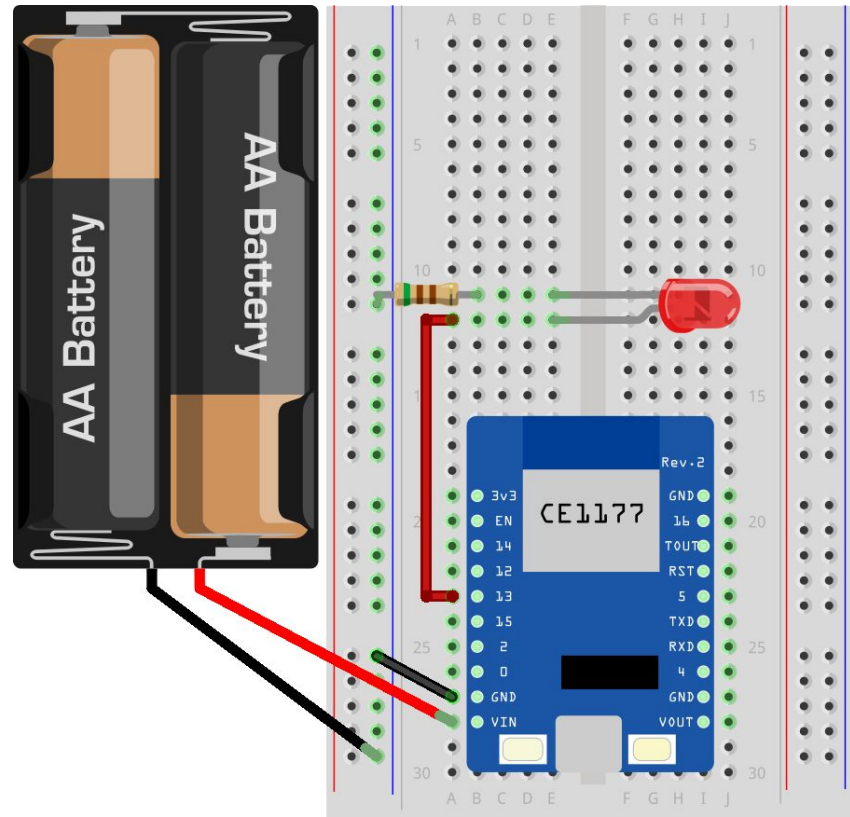
3.  「矢印アイコン」をクリックしてプログラムをESPrにアップロードする

4. ブラウザからHTTPリクエストを送信して、レスポンスが返ってくる様子を確認する

3-10. ESPrを電池で動かしてみよう

バッテリーからの電源回路を追加配線する

1. ESPr : VIN -> B28, VOUT -> I28
2. バッテリー : 赤線 -> A28
3. バッテリー : 黒線 -> 青ライン



3-11. 演習①

2つのLED(LED_AとLED_B)を用意し、以下の4つのスイッチを作って制御しましょう。

1. LED_A、LED_B両方とも点灯。
2. LED_Aのみ点灯。
3. LED_Bのみ点灯。
4. すべて消灯。



Lunch Break

4. **Learn**(応用) : WiFi経由でサーボモーター制御

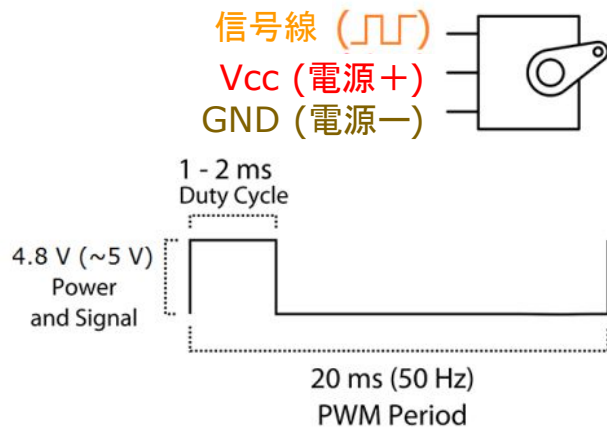
4-1. サーボモーター SG90



物体の位置や方向を目標とし、その目標値に追従するように自動で制御できるモーター

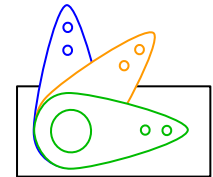
DCモーターやACモーターで応答を早くしたり、制御しやすくしたりすることができる。形状が比較的大きく、位置決めや速度を決めるモーターとして、ロボットアームや3Dプリンターなどによく使用される

動作電圧 4.8V



目標制御位置とPWMパルス幅

-90 (左方向)	0.5ms
0 (中心)	1.5 ms
90 (右方向)	2.4ms

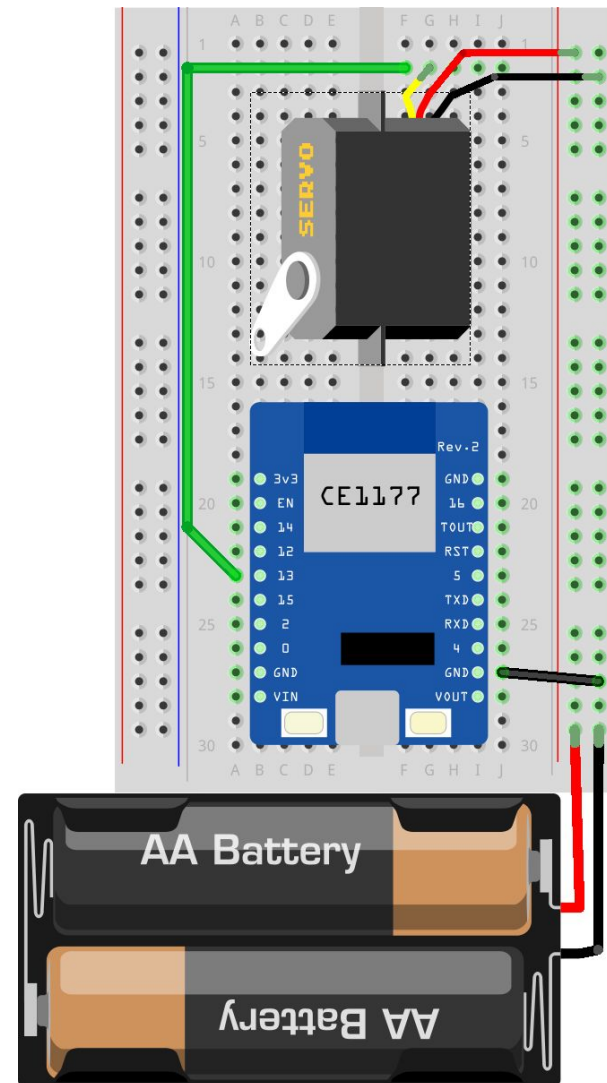


4-2.電子回路

電子部品を配線する

注意 :サーボモーターは別電源にします

1. ESPr : VIN → B28, VOUT → I28
2. サーボモーター(オレンジ) → G2
3. サーボモーター(赤) → 赤ライン
4. サーボモーター(黒) → 青ライン
5. 電池ボックス2(+) → 赤ライン(右)
6. 電池ボックス2(ー) → 青ライン(右)
7. ジャンパー(緑) : F2 → A23





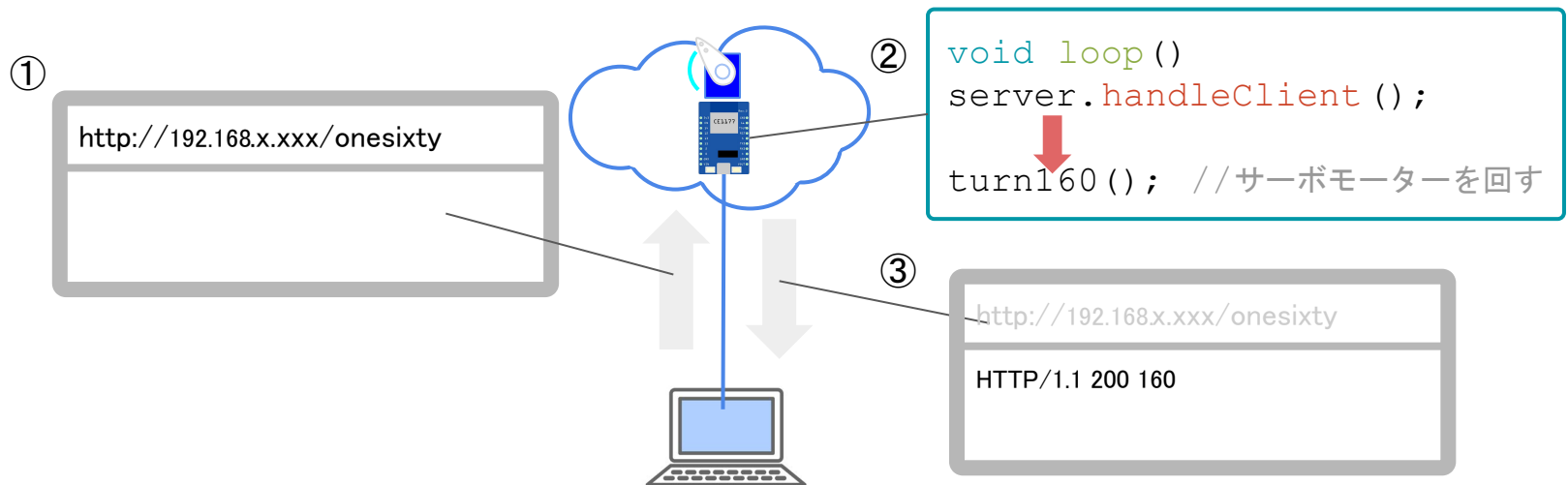
4-3.プログラム(HTTPリクエストに応じてサーボモーターを動かす)

```
1 // Servo.ino
2 #include <Servo.h>
3 #include <ESP8266WebServer.h>
4 #define SSID
5 "FABLABKAMAKURA2_guest"
6 #define PASS "kamakura_1192"
7
8 ESP8266WebServer server(80);
9 Servo myservo;
10
11 void setup() {
12     Serial.begin(9600);
13     myservo.attach(13);
14     myservo.write(20);
15     connectWiFi();
16     server.on("/", handleRoot);
17     server.on("/twenty", turn20);
18     server.on("/onesixty", turn160);
19     server.begin();
20 }
21
22 void loop() {
23     server.handleClient();
24 }
```

```
25 void connectWiFi() {
26     WiFi.begin(SSID, PASS);
27     Serial.println();
28     while(WiFi.status() != WL_CONNECTED) {
29         delay(500);
30         Serial.print(".");
31     }
32     Serial.println();
33     Serial.println("WiFi connected");
34     Serial.println(WiFi.localIP());
35 }
36 void handleRoot() {
37     myservo.write(20);
38     server.send(200, "text/plain", "20");
39 }
40 void turn20() {
41     myservo.write(20);
42     server.send(200, "text/plain", "20");
43     Serial.println("Turn on 20");
44 }
45 void turn160() {
46     myservo.write(160);
47     server.send(200, "text/plain", "160");
48     Serial.println("Turn on 160");
49 }
```

4-4.プログラム実行

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. ノートパソコンとESPrをUSBケーブルで接続し  「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
4. ブラウザからHTTPリクエストを送信して、サーボモーターが動作する様子を確認する



4-5.プログラム解説

```
1 // Servo.ino
2 #include <Servo.h>
3 #include <ESP8266WebServer.h>
4 #define SSID
5 "FABLABKAMAKURA2_guest"
6 #define PASS "kamakura_1192"
7
8 ESP8266WebServer server(80);
9 Servo myservo;
10
11 void setup() {
12     Serial.begin(9600);
13     myservo.attach(13); //13番ピン
14     myservo.write(20);
15     connectWiFi();
16     server.on("/", handleRoot);
17     server.on("/twenty", turn20);
18     server.on("/onesixty", turn160);
19     server.begin();
20 }
21
22 void loop() {
23     server.handleClient();
24 }
```

クライアント(ウェブブラウザ)からのリクエスト文字列

リクエスト文字列	対応関数
http://192.168.1.xxx/	handleRoot
http://192.168.1.xxx/turn20	turn20
http://192.168.1.xxx/onesixty	turn160

```
25 void
26 Wi
27 Se
28 wh http://192.168.1.xxx/ → handleRoot
29
30 http://192.168.1.xxx/turn20 → turn20
31 }
32 http://192.168.1.xxx/onesixty → turn160
33 Serial.println("WiFi connected");
34 Serial.println(WiFi.localIP());
35 }
36 void handleRoot() {
37     myservo.write(20);
38     server.send(200, "text/plain", "20");
39 }
40 void turn20() {
41     myservo.write(20);
42     server.send(200, "text/plain", "20");
43     Serial.println("Turn on 20");
44 }
45 void turn160() {
46     myservo.write(160);
47     server.send(200, "text/plain", "160");
48     Serial.println("Turn on 160");
49 }
```


5. **Learn**(応用) :
環境光センサの値をブラウザでモニタリング

5-1. 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。

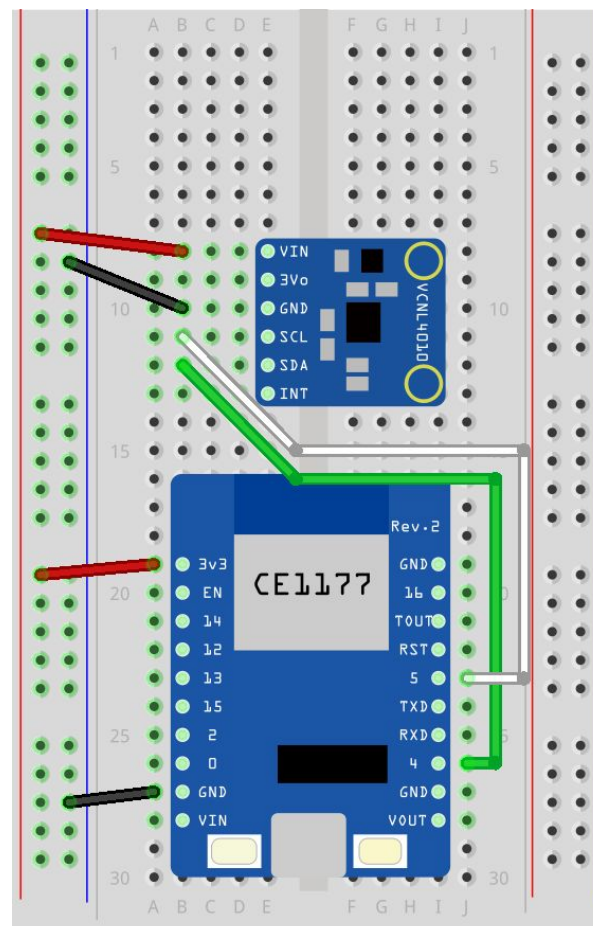
近距離センサだけでなく環境光センサも内蔵しているので、周囲の明るさをLux値で取得することができます。

<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 : B19 -> 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(緑)① : B11 -> J23
8. ジャンパー(緑)② : B12 -> J26




5-3.プログラム(HTTPリクエストに応じて環境光の測定値を返す)

```
1  #include <ESP8266WebServer.h>
2  #include <FaBoProximity_VCNL4010.h>
3
4  ESP8266WebServer server(80);
5  FaBoProximity fabo;
6
7  #define SSID "FABLABKAMAKURA2_guest"
8  #define PASS "kamakura_1192"
9
10 char* html = "<!DOCTYPE html>\n
11 <html>\n
12 <head>\n
13   <meta http-equiv='refresh' content='1'\n
14   charset='UTF-8'>\n
15   <meta name='viewport'\n
16   content='width=device-width'>\n
17 </head>\n
18 <body>\n
19   <h1> Value: %d</h1>\n
20 </body>\n
21 </html>";
22
23 void setup() {
24   Serial.begin(9600);
25   initWiFi();
26   server.on("/", handleRoot);
27   server.begin();
28   fabo.begin();
29 }
```

```
26 void loop() {
27   server.handleClient();
28 }
29
30 void initWiFi() {
31   WiFi.begin(SSID, PASS);
32   if(WiFi.status() != WL_CONNECTED) {
33     delay(500);
34     Serial.print(".");
35   }
36   Serial.println();
37   Serial.println("WiFi connected");
38   Serial.println(WiFi.localIP());
39 }
40
41 void handleRoot() {
42   int value;
43   char temp[400];
44
45   if(fabo.checkAmbiReady()) {
46     value = fabo.readAmbi(); //環境光測定値
47   }
48
49   snprintf(temp, 400, html, value);
50   server.send(200, "text/html", temp);
51 }
```

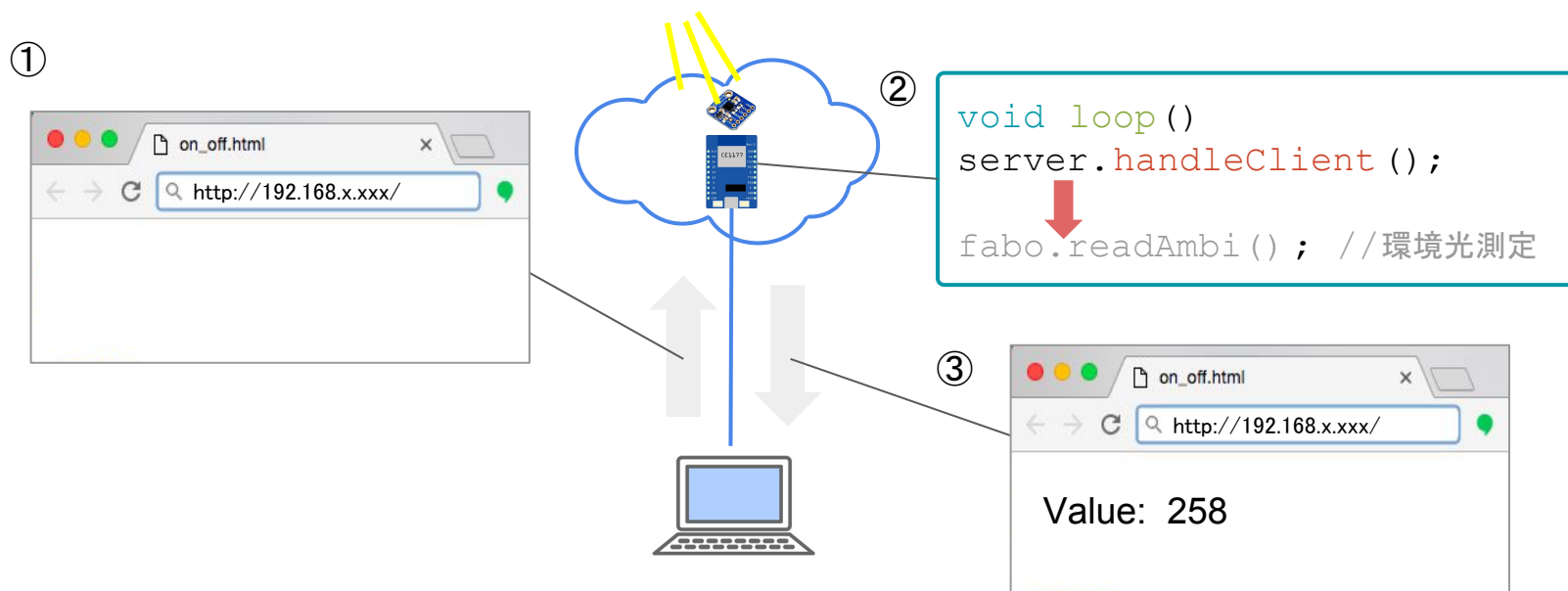
5-4.プログラム実行

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

4. ブラウザからHTTPリクエストを送信して、測距/環境光センサー (VCNL4010) を使って光量の測定値を返す様子を確認する



5-5. プログラム解説①

```
1  #include <ESP8266WebServer.h>
2  #include <FaBoProximity_VCNL4010.h>
3
4  ESP8266WebServer server(80);
5  FaBoProximity fabo;
6
7  #define SSID "FABLABKAMAKURA2_guest"
8  #define PASS "kamakura_1192"
9
10 char* html = "<!DOCTYPE html>\
11 <html>\
12 <head>\
13   <meta http-equiv='refresh' content='1'\
14   charset='UTF-8'>\
15   <meta name='viewport'\
16   content='width=device-width'>\
17 </head>\
18 <body>\
19   <h1> Value: %d</h1>\
20 </body>\
21 </html>";
22
23 void setup() {
24   Serial.begin(9600);
25   initWiFi();
26   server.on("/", handleRoot);
27   server.begin();
28   fabo.begin();
29 }
```

`<meta http-equiv='refresh' content='1' charset='UTF-8'>`

'http-equiv'に'refresh'を指定すると、'content'に代入した値ごとに画面が更新されます。今回は1秒ごとに画面が切り替わります。

`<h1> Value: %d</h1>`

'%d'と書かれている箇所は指定した変数と置き換わります。ここにセンサーの値が表示されます。

5-5. プログラム解説②

```
26 void loop() {  
27     server.handleClient();  
28 }  
29  
30 void initWiFi() {  
31     WiFi.begin(SSID, PASS);  
32     if(WiFi.status() != WL_CONNECTED) {  
33         delay(500);  
34         Serial.print(".");  
35     }  
36     Serial.println();  
37     Serial.println("WiFi connected");  
38     Serial.println(WiFi.localIP());  
39 }  
40 void handleRoot() {  
41     int value;  
42     char temp[400];  
43  
44     if(fabo.checkAmbiReady()) {  
45         value = fabo.readAmbi(); //環境光測定値  
46     }  
47     snprintf(temp, 400, html, value);  
48     server.send(200, "text/html", temp);  
49 }
```

`char temp[400];`

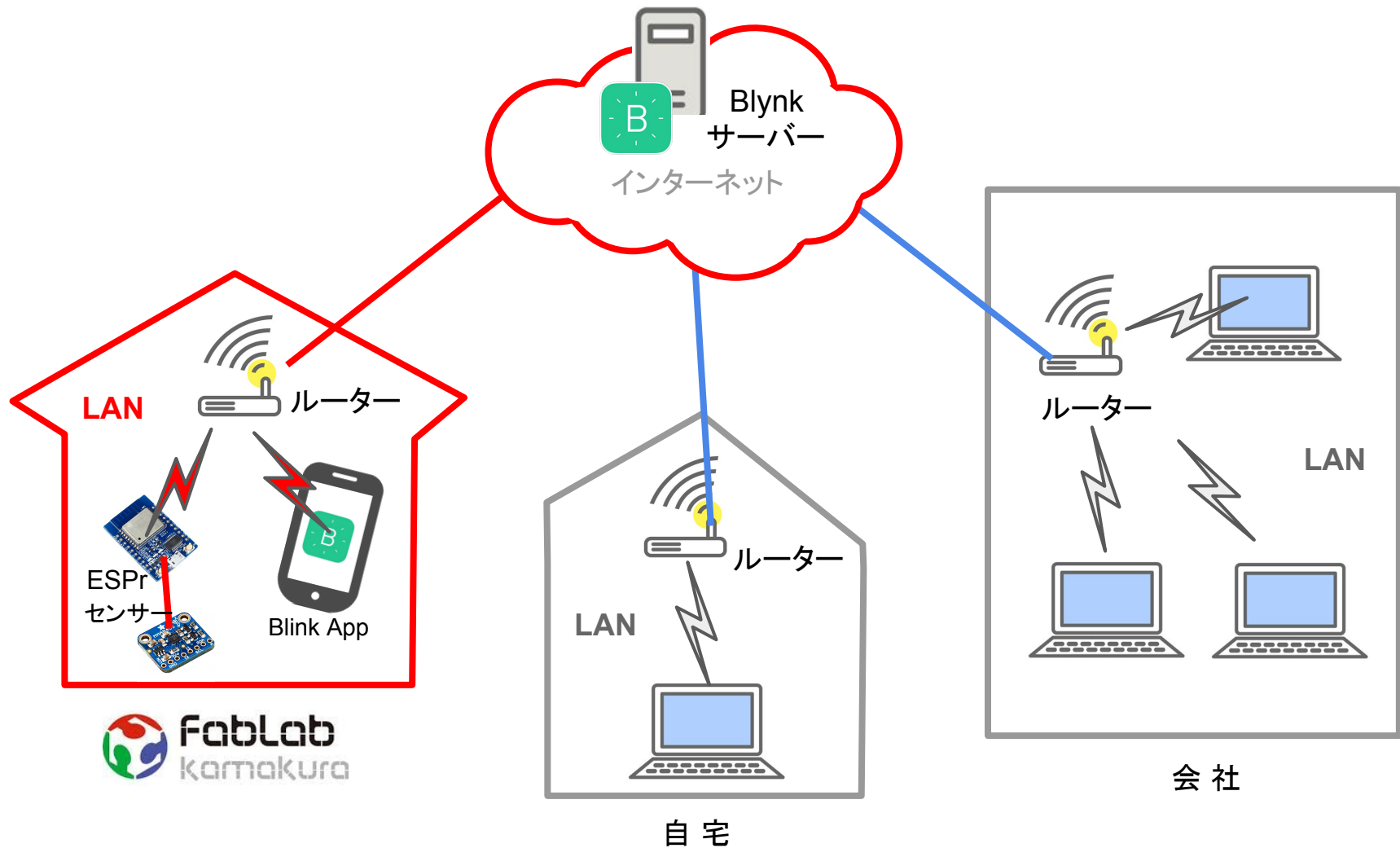
char型の配列tempを作成します。配列は[]の中に指定した数値分の変数を用意します。要するにchar型の変数を400個作成するということです。そして各変数に文字が一つずつ格納されます。

`snprintf(temp, 400, html, value);`

snprintf関数は文字列書式に従って指定文字数分だけ文字配列に書き込みます。第一引数に書き込むための配列を用意し、第二引数に書き込むサイズ、第三引数に書式文字列を渡します。ここで'html'の中で記入されていた'%d'が'value'に置き換わります。

6. **Learn**(応用) : スマートフォンでIoT

3-1. 情報の伝わり方



3-2. Blynk概要

<http://www.blynk.cc/>

Blynk

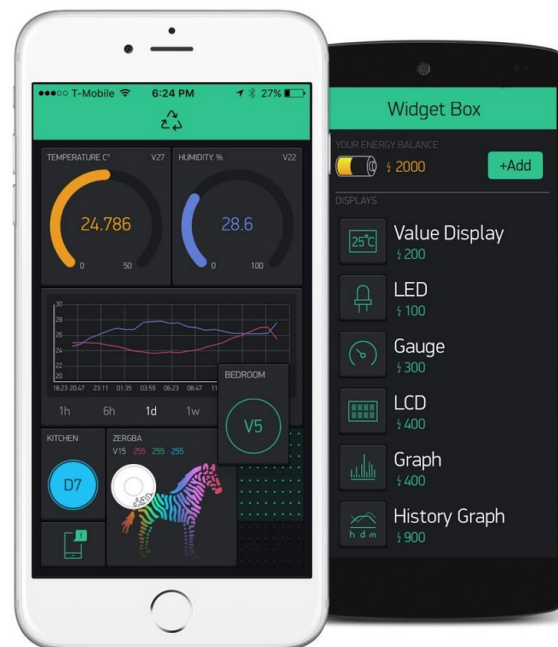
HOME GETTING STARTED DOCS COMMUNITY FOR BUSINESS

Get latest news and updates from us:

SIGN UP FOR BLYNK NEWS

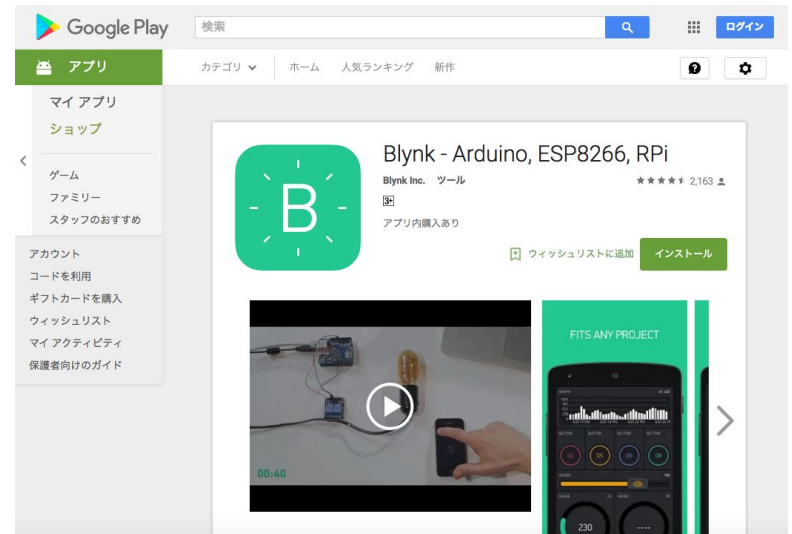
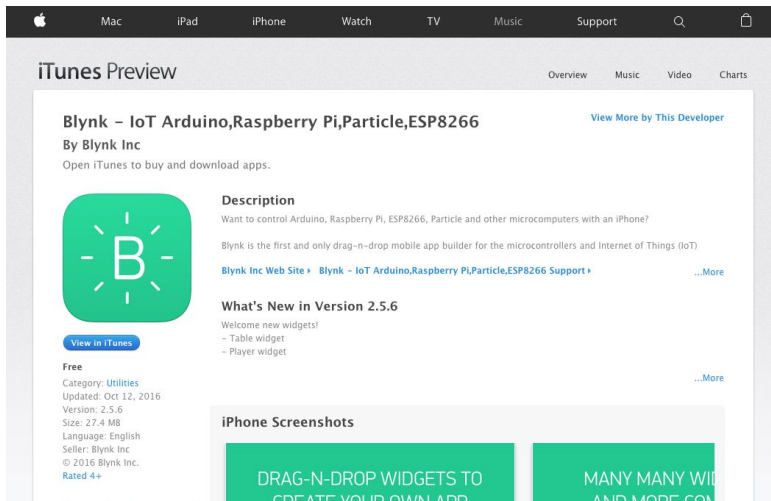
BlynkはiPhoneやアンドロイドなどのスマートフォン端末を利用して、IoTプロダクトをつくることのできるサービスです。

アプリ側のインターフェース構築はドラッグアンドドロップで行なうことが出来るので、アプリ側のコードを書く必要はなく、初学者に門戸が開かれています。



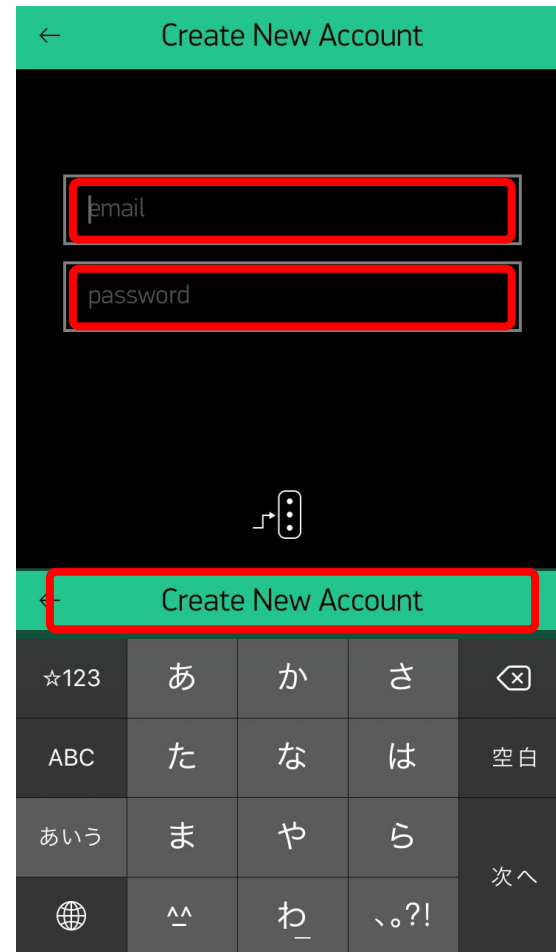
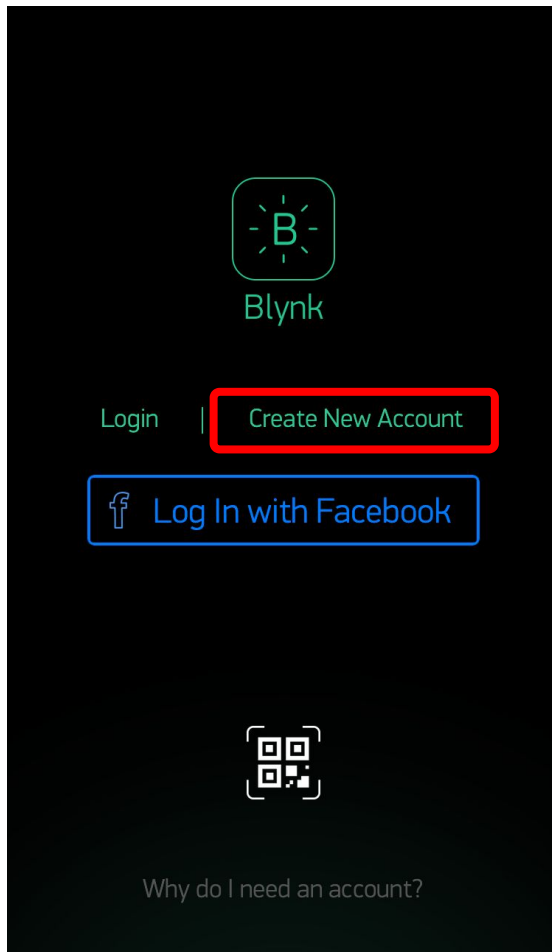
3-3. 環境構築① iOS/Androidアプリのインストール

1. スマートフォンの App Store / Play Store で Blynkを検索し、インストールしてください。



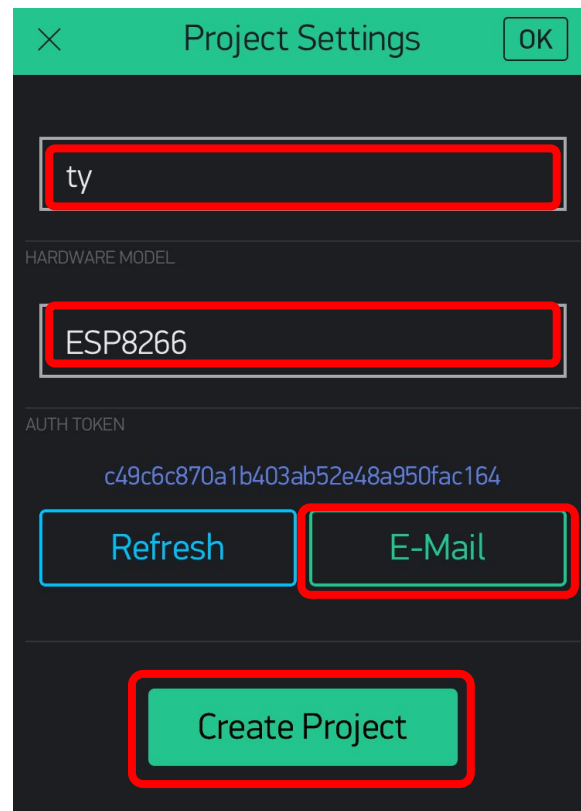
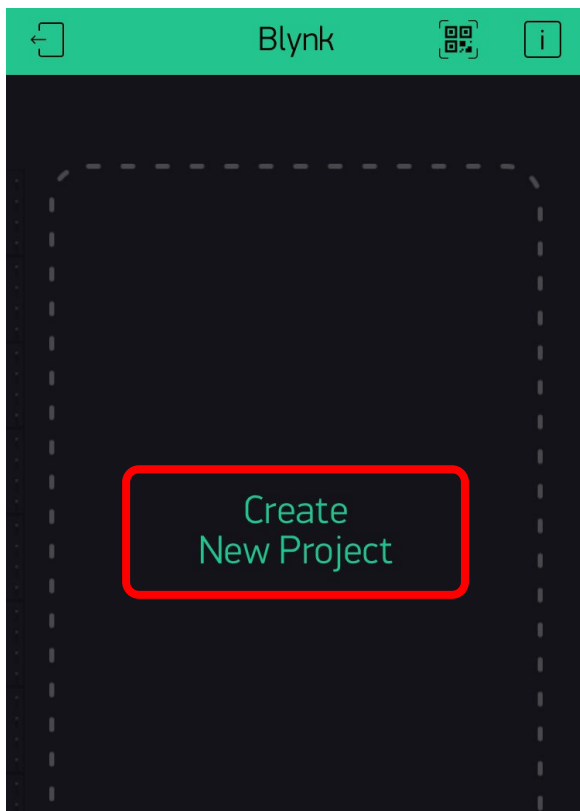
3-3. 環境構築② Blynk使用のためのアカウント作成

2. インストールが完了したらアプリを起動し、メールアドレスとパスワードを入力し、[Create New Account]をクリックして、アカウントを作成します。



3-3. 環境構築③ 新しくプロジェクトを作成

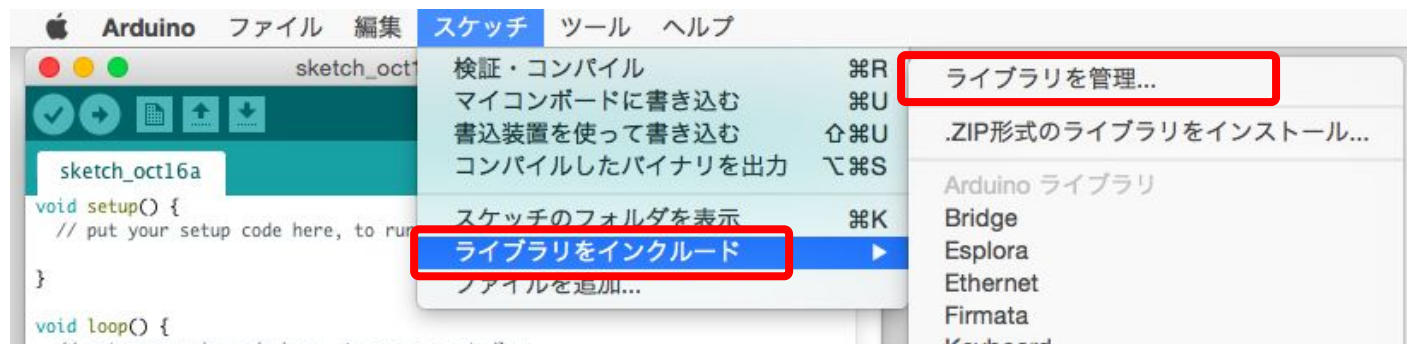
1. 作成したアカウントで Blynk アプリにログインし、[Create New Project]をタップします。



2. プロジェクト名を入力し、[HARDWARE MODEL]は[ESP8266]を選択します。
3. [E-Mail]を押して生成された固有の AUTH TOKENを登録したメールアドレスに送信します。
4. [Create Project]をタップして、プロジェクトを作成します。

3-3. 環境構築④ ArduinoへBlynkライブラリー追加

1. Arduinoソフトウェアで、「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。



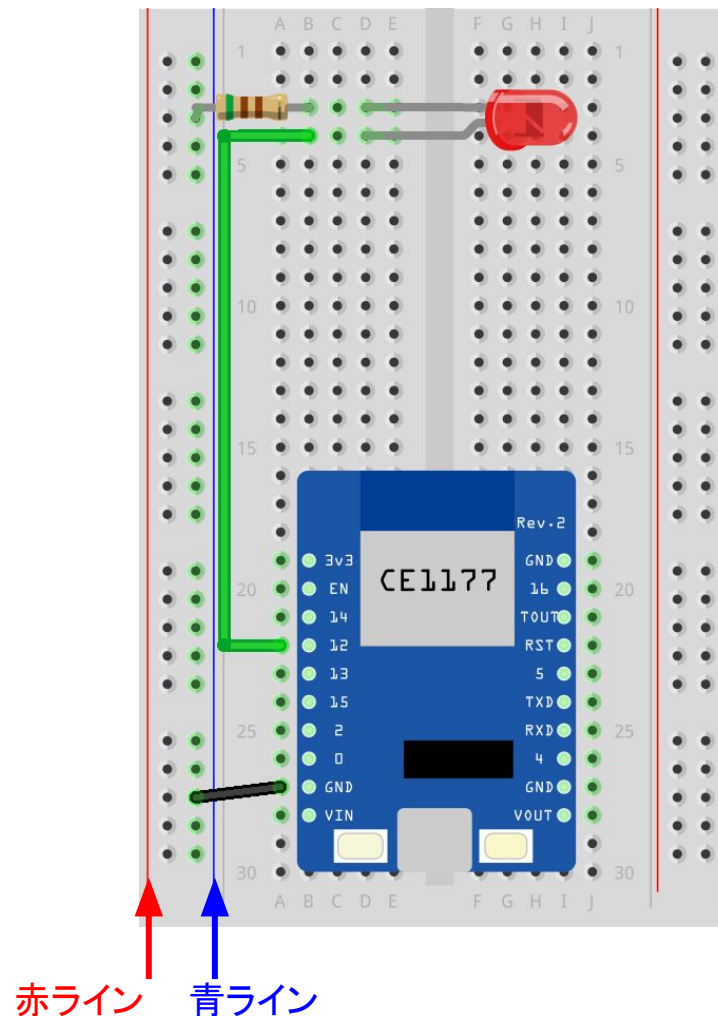
2. 「Blynk」と検索し、「Adafruit MCP9808 Library by Adafruit」というライブラリが出てきますので、選択してインストールします。



3-4. 回路図 : LED

電子部品を配線する

1. ESPr - VIN → B28
2. ESPr - VOUT → I28
3. LED - 短い足 → D4
4. LED - 長い足 → D5
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



3-5. プログラム : LED

Arduinoソフトウェアで以下のソースコードをコーディングし、ESPrへアップロードします。
4行目のAUTHにはメール送信したAUTH TOKENをコピーペーストして入力してください。
(AUTH TOKEN はアプリのプロジェクト画面>右上の設定アイコン から確認することができます)

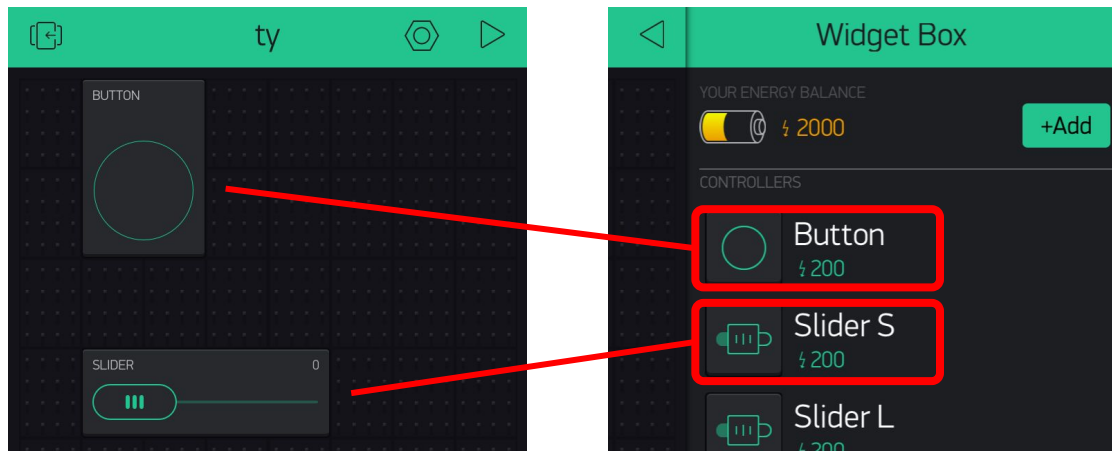
BlynkLED.ino

```
1  #include <ESP8266WiFi.h>
2  #include <BlynkSimpleEsp8266.h>
3
4  #define AUTH "xxxxxxxx"
5
6  #define SSID "FABLABKAMAKURA2_guest"
7  #define PASS "kamakura_1192"
8
9  void setup() {
10     Serial.begin(9600);
11     Blynk.begin(AUTH, SSID, PASS);
12 }
13
14 void loop() {
15     Blynk.run();
16 }
```


3-6. アプリ作成 : LED①

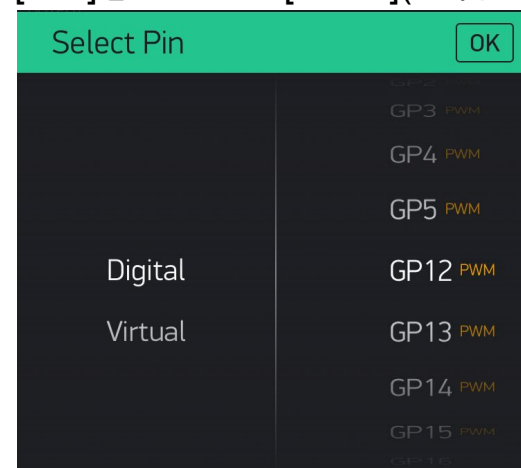
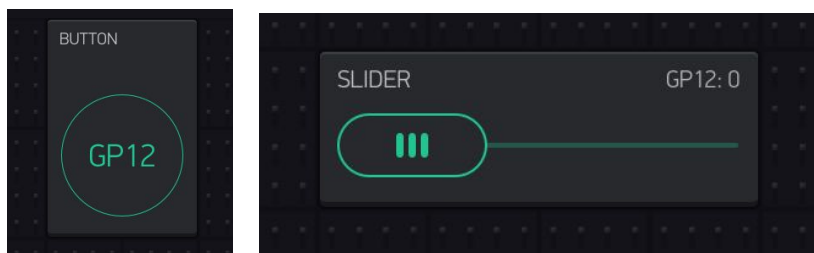
5. 空っぽのプロジェクト画面をタップすると Widget Boxと書かれたウィジェット (部品) のリストが表示されるので、以下のいずれかを選択して配置します。

- Button
- Slider S



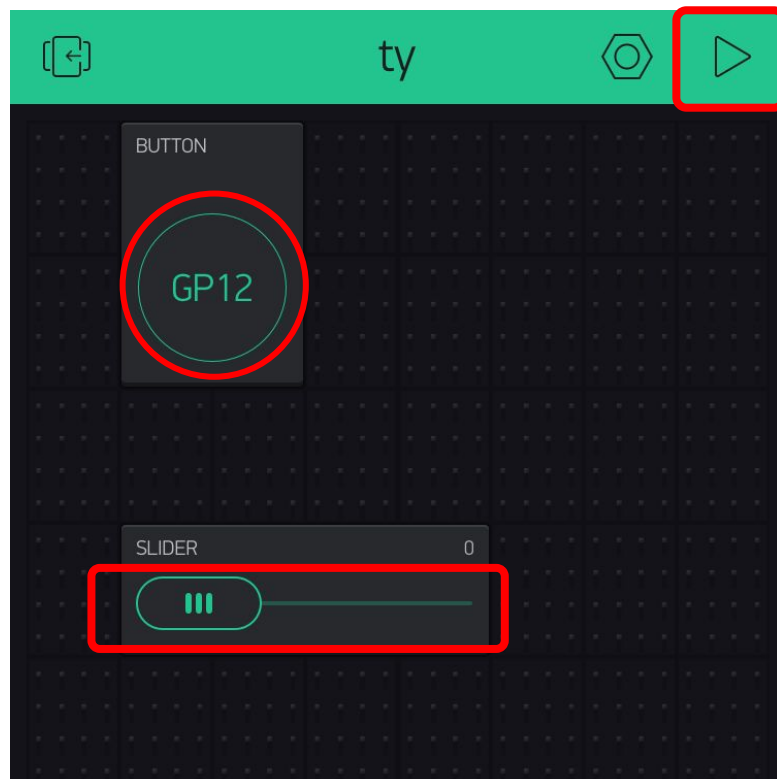
6. 選んだウィジェットをタップすると設定画面が表示されるので、[PIN]をタップして[GP12](12番ピン)を紐付け、[OK]をタップして戻ります。

7. プロジェクト画面でウィジットに GP12と表示されます。



3-6. 動作確認：LED②

1. スマートフォンのアプリ画面右上の [▶] をタップして、操作用アプリを実行します。

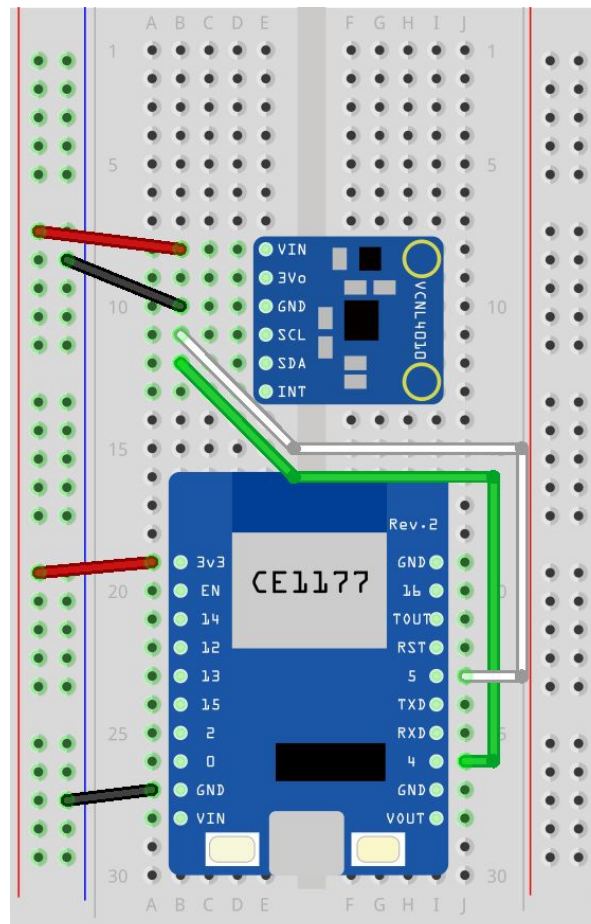


2. 作成したウィジェットをタップ／スライドして、LEDの動作を確認しましょう。
3. (Option) プロジェクト編集画面から各ウィジェットの設定を変更して、試してみましょう。

3-7.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 : B19 -> 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(緑)① : B11 -> J23
8. ジャンパー(緑)② : B12 -> J26

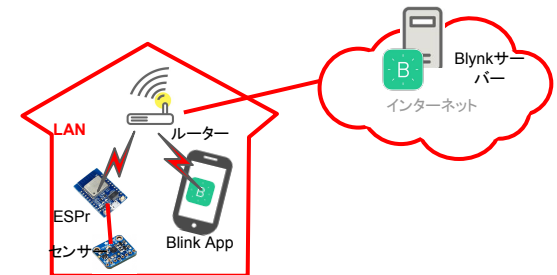


3-8. プログラム：センサー

ESPr2Blynk.ino

```
1  #include <ESP8266WiFi.h>
2  #include <FaBoProximity_VCNL4010.h>
3  #include <BlynkSimpleEsp8266.h>
4
5  #define AUTH "xxxxxxxx"
6  #define SSID "FABLABKAMAKURA"
7  #define PASS "kamakura_1192"
8  FaBoProximity fabo;
9
10 BLYNK_READ(V0) {
11   if(fabo.checkProxReady()) {
12     int value = fabo.readProx();
13     Blynk.virtualWrite(V0, value);
14   }
15 }
16
17 void setup() {
18   Serial.begin(9600);
19   Blynk.begin(AUTH, SSID, PASS);
20   fabo.begin();
21 }
22
23 void loop() {
24   Blynk.run();
25 }
26
27
28
29
```

1. Arduinoソフトウェアでソースコードをコーディングし、ESPrへアップロードします。
2. スマートフォンのBlynkアプリ画面右上の[>]をタップして、操作アプリを実行します。
3. ブレッドボードごと加速度センサーを動かして、作成したウィジェットの表示の変化を確認しましょう。



4. (Option) プロジェクト編集画面から各ウィジェットの設定を変更して、様々な表示方法を試してみましょう。

3-9. アプリ作成(センサー)①

1. プロジェクト画面の余白部分をタップしてウィジェット (部品)のリストを表示させ、[Graph]を選択します。同様の操作を繰り返して、3つの [Graph]を表示させましょう。
(注) [Labeled Value M]や[Gauge]などを使用することも出来ます。
2. 一つ目のウィジェットをタップすると設定画面が表示されるので、[PIN]をタップして[Virtual]の[V0]を紐付け、値を-5000から5000に変更します。[OK]をタップして戻ります。プロジェクト画面でウィジットに V0と表示されます。二つ目、三つ目も同様の操作で [PIN]を[Virtual]の[V1]、[V2]にそれぞれ紐付けます。



3-9. アプリ作成(センサー)② 動作確認

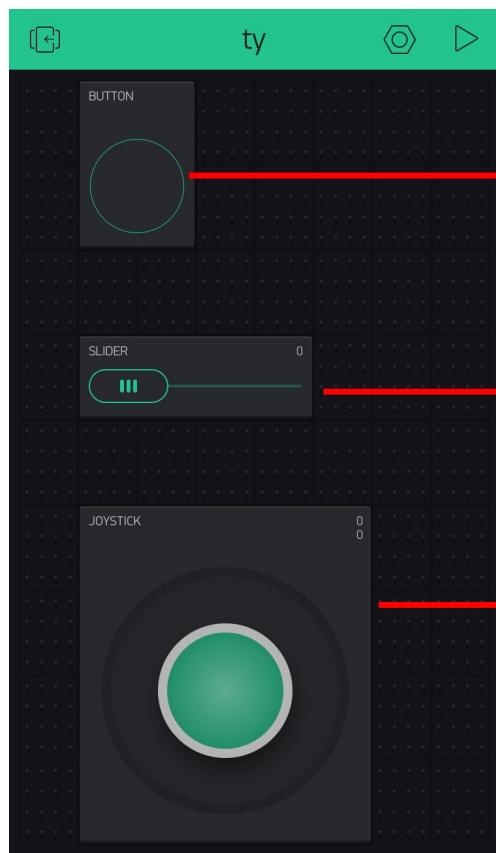
1. 右上にある三角マークをタップするとアプリが動作します。右のような折れ線グラフが表示されます。



3-10. iOS/Androidアプリ画面例① アウトプット

スマートフォンアプリのインターフェース構築はドラッグアンドドロップで簡単に行うことができます。あらかじめプログラムを書いて動作を制御する代わりに、このインターフェースを使って逐次 ESPrを介して操作するためのOUTPUT値をコントロールすることができます。

Blynkアプリでは、各種アウトプットデバイスの操作に適した様々なインターフェースが提供されています。



BUTTON

押すことで状態を変化させることが出来るスイッチです。

SLIDER

位置によって送る値を変化させることが出来るスライダースイッチです。LEDの調光などに利用できます。

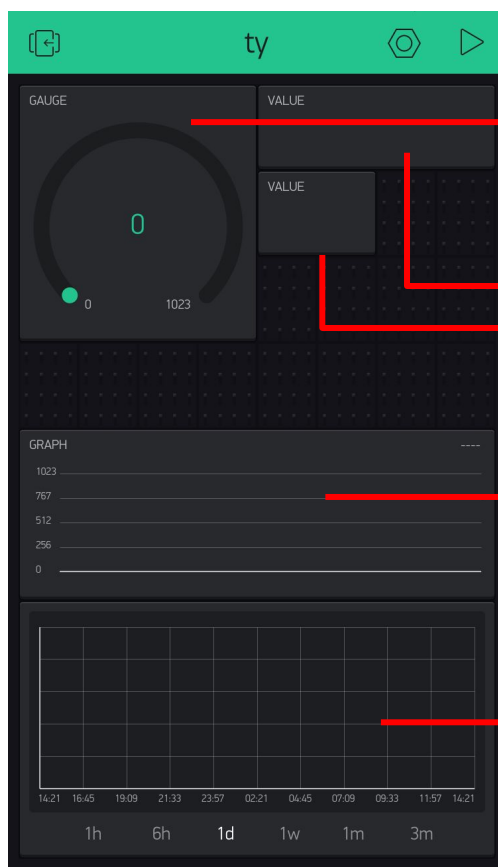
JOYSTICK

X,Yの2つの値を操作することが出来るパーツです。

3-10. iOS/Androidアプリ画面例② インプット

各種センサーデバイスから ESPrへのINPUTデータが、Blynkサーバーを経由してスマートフォンアプリに送信されます。

Blynkアプリでは、それらを表示する様々なタイプのインターフェースが提供されています。



GAUGE

ゲージタイプのディスプレイです。
値に応じて円弧上に色が変わります。

VALUE

値を表示するだけのシンプルなディスプレイです。

GRAPH

時間とセンサから送られてくる値の2つの軸で表示するディスプレイです。

HYSTORY

時間とセンサから送られてくる値の2つの軸で表示するディスプレイです。
GRAPHとの違いは過去に取得した値をすべて保存していて、それらを表示することができる部分です。

7. Make

1日目、2日目の**Learn**のセッションで学んだ内容を組み合わせてプロトタイプを作ってみましょう。

8. 参考ウェブサービス

[MDN : Web入門](#) : mozillaが作っているウェブを学ぶためのプラットフォーム

[ドットインストール](#) : 3minウェブ動画。html、css、javascriptなどウェブを構成する技術を学べます。

The image shows two web pages. The top page is the MDN (Mozilla Developer Network) website, which has a grey header with the MDN logo, navigation links like 'WEB プラットフォーム', 'MOZILLA DOCS', and '開発者ツール', and a search bar. Below the header, there's a breadcrumb trail 'MDN > Webを学ぶ > Web入門' and a large heading 'Web入門'. The bottom page is the 'ドットインストール' (Dots Install) website, featuring a yellow header with the site's logo and navigation links. The main content area has a large orange and white banner for '3分動画でマスターするプログラミング学習サイト' (Master programming with 3-minute videos), a green button to 'すべてのレッスンを見る' (View all lessons), and a sidebar with login options including '新規登録 (無料)' (Sign up for free), 'ログイン' (Login), and 'Sign in with Google'. A small inset image shows a smartphone displaying a lesson from the Dots Install app.