2019年度 人と地域共創センター公開講座(春・夏)

AI/IoTセンサのしくみを知ろう(基礎編)
第8回 無線WiFiをつかう2

徳島大学技術支援部
辻　明典　博士(工学)
E-mail: a-tsuji@is.tokushima-u.ac.jp

2019年6月29日(土)

講座内容

- 講師: 辻　　明典(徳島大学技術支援部)
  桑折　範彦(徳島大学名誉教授)
  川上　博　(徳島大学名誉教授)
- 土曜日：10:00～11:30
- 日程：
  ① 5/11 ガイダンス，PC環境設定
  ② 5/18 プログラミングをはじめよう
  ③ 5/25 LED を光らせる(川上先生)
  ④ 6/ 1 温度・湿度をはかる(桑折先生)
  ⑤ 6/ 8 距離をはかる
  ⑥ 6/15 動きをはかる
  ⑦ 6/22 無線Wi-Fiをつかう1
  ⑧ 6/29 無線Wi-Fiをつかう2
  ⑨ 7/ 6 まとめ，振り返り

ネットワーク構成(方式1)

▶方式1・・・LAN，インターネット未接続

クライアント

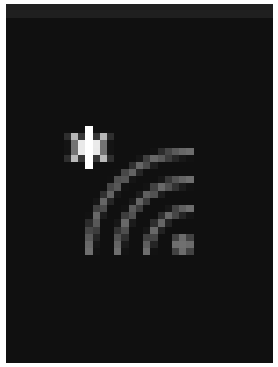
・IPアドレス ／ 192.168.4.2

無線アクセスポイント(無線AP) & ルーター

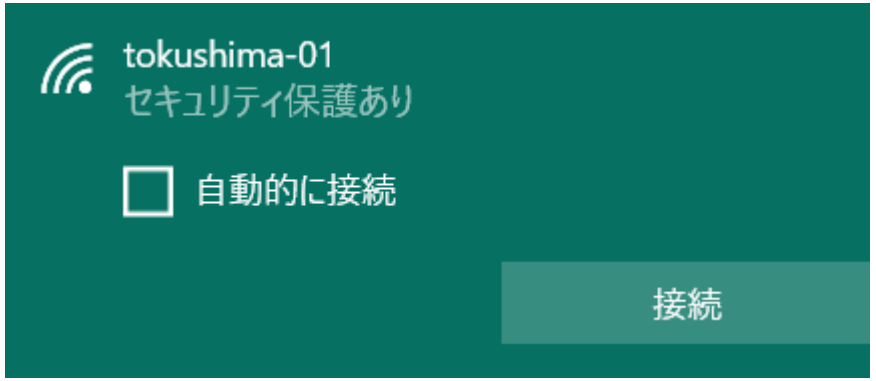
・SSID ／ JJ7-<NN>
・IPアドレス ／ 192.168.4.1
・ウェブサーバー ／ http://192.168.4.1

無線APの接続

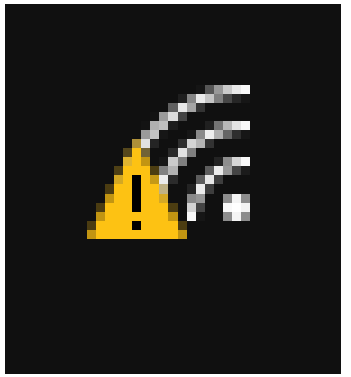
- プログラムを書き込む度に，接続作業が必要



① タスクバーの無線アイコンをクリック



② 各自のSSIDを選択して接続（JJ7-<NN>）



③ 無線接続が完了

＊ インターネットに接続しないため！が表示

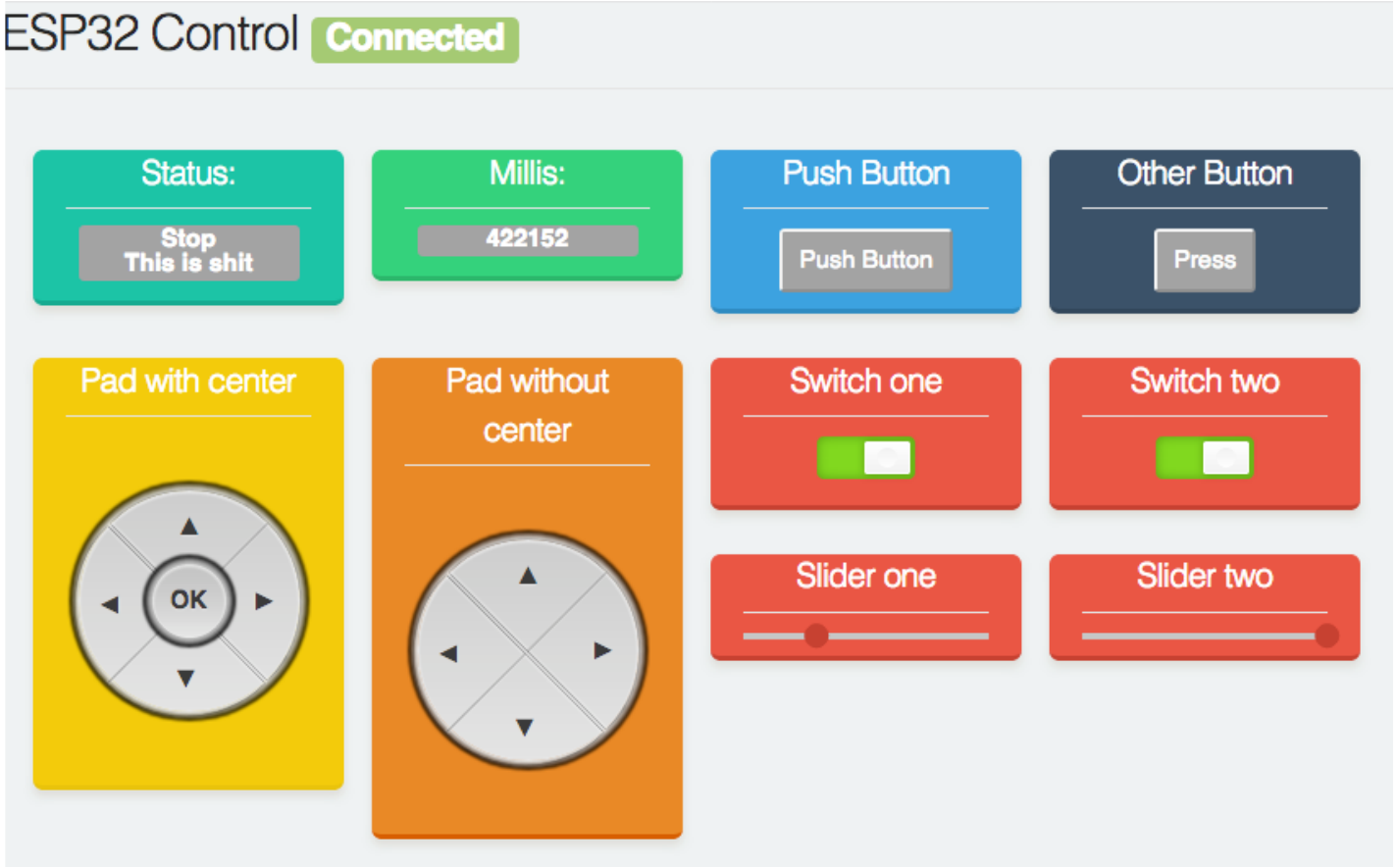
ESPUIライブラリ

- ウェブ上のユーザインタフェース(UI)作成

UIの種類：

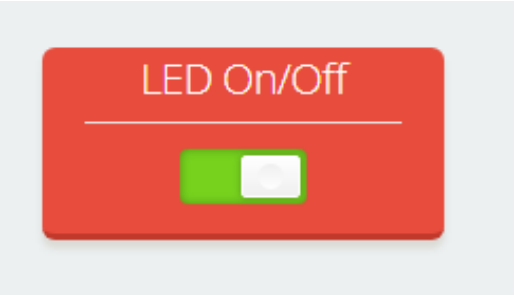
- ラベル
- ボタン
- スイッチ
- コントロールパッド
- コントロールパッド（中央ボタン付き）
- スライダー

- 背景の色や文字を変更できる。
- イベントハンドラーにより動作を指定できる。



イベントハンドラー

- ESPUI.switcher("LED On/Off", true, &Switch, COLOR_NONE);



イベント発生！

スイッチの状態が変化

ON（S_ACTIVE）

または

OFF（S_INACTIVE）

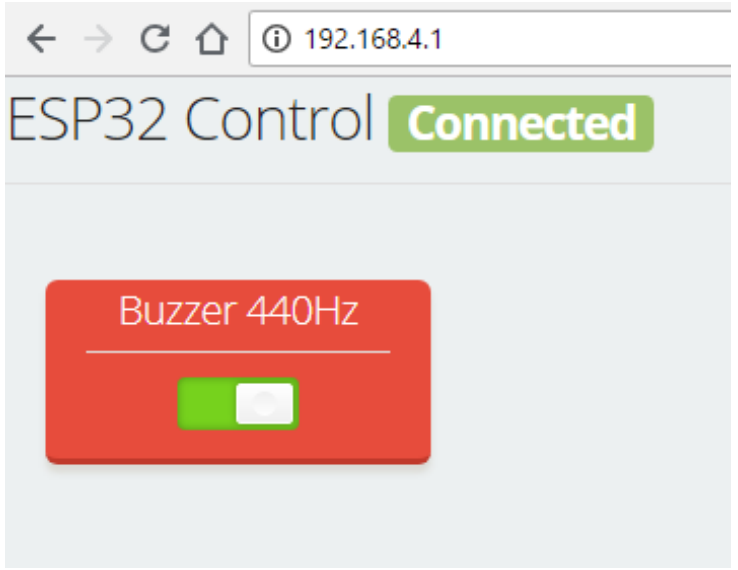
イベントハンドラー

```
void Switch(Control sender, int value) {
  switch (value) {
    case S_ACTIVE:
      digitalWrite(LED_PIN, HIGH);
      break;
    case S_INACTIVE:
      digitalWrite(LED_PIN, LOW);
      break;
  }
}
```

Ex0706｜ボタンが押されたときブザーを鳴らす

- ボタンが押されたことを検知して周波数440Hzでブザーを鳴らす
  - 5kHzのタイマー(CH0)を使用

```
ledcSetup(LEDC_CH0, LEDC_BASE_FREQ, LEDC_TIMER_13BIT);
ledcAttachPin(SPEAKER_PIN, LEDC_CH0);

ESPUI.switcher("Buzzer 440Hz", true, &Switch, COLOR_NONE);

void Switch(Control sender, int value) {
  switch (value) {
    case S_ACTIVE:
      ledcWriteTone(LEDC_CH0, 440); // トーンの開始 440Hz
      break;
    case S_INACTIVE:
      ledcWriteTone(LEDC_CH0, 0); // トーンの停止
      break;
  }
}
```



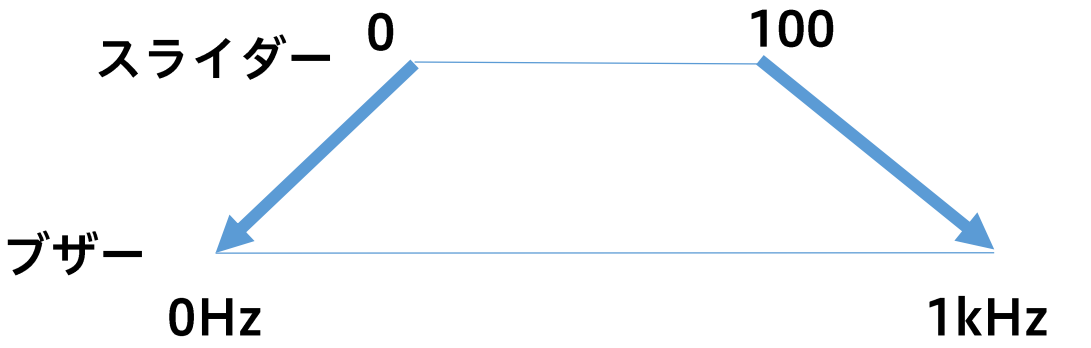
ledcWriteTone(チャネル, 周波数)
指定のチャネルに周波数の波形を生成する関数

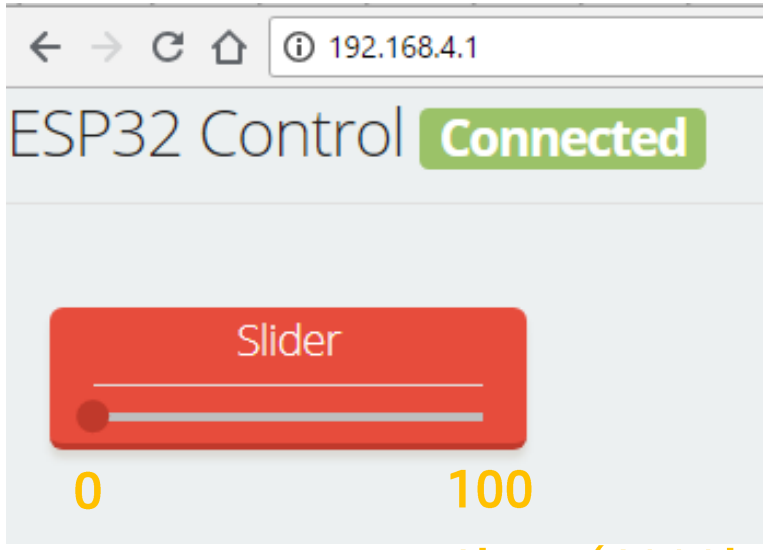
Ex0707｜スライダーによる音階の変化

- スライダーを動かし，音階（トーン）を0Hz～1kHzの範囲で変化

```
ESPUI.slider("Slider", &slider, COLOR_ALIZARIN, "0");

void slider(Control sender, int type) {
  Serial.println(sender.value);
  int n = map(sender.value.toInt(), 0, 100, 0, 1000);
  ledcWriteTone(LEDC_CH0, n); // 0Hz～1000Hz
}
```

map関数：ある数値の範囲を別の数値の範囲に置き換える。





Ex0801｜距離センサの計測結果表示

ラベルの登録

- ESPUI.label("距離(mm)：", COLOR_EMERALD, "0");

ラベルの表示

- ESPUI.print("距離(mm)：", String(d));

＊ ラベル名”距離(mm)”は同じ名前
＊ String関数：文字列に変換

Ex0802｜距離センサの検出表示

ラベルの登録

- ESPUI.label(“検知：", COLOR_EMERALD, "0");

ラベルの表示

- ESPUI.print(“検知：”, “検出中”);
- ESPUI.print(“検知：”, “ー”);

＊ ラベル名”検知：”は同じ名前

Ex0803｜ユーザ認証

問題：誰でもサイトにアクセスできる

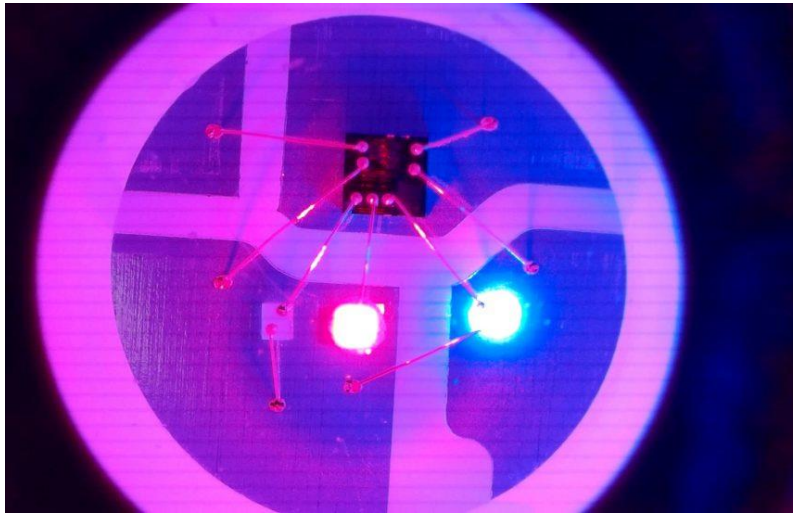
- ユーザ名とパスワードによる認証

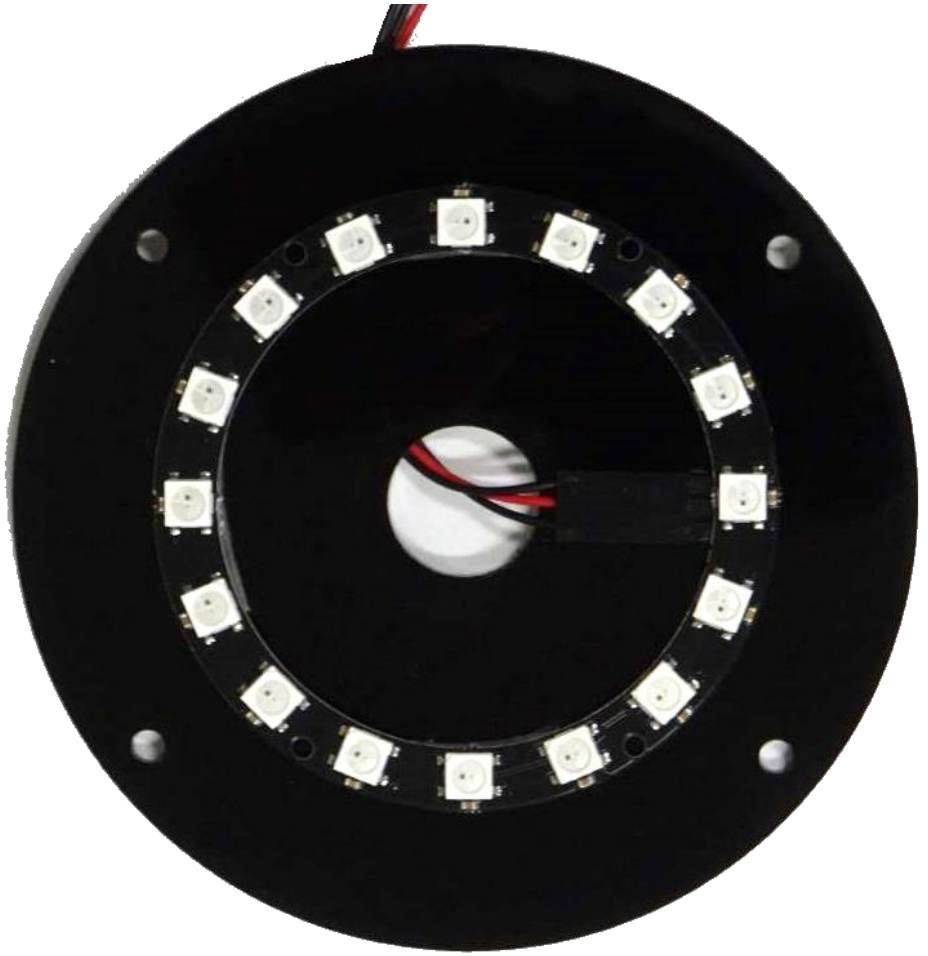
ユーザ認証

- ESPUI.begin(“ESP32 Control”, “ユーザー名”, “パスワード");

リングLED

- リングLED
  - フルカラーLEDが16個
- フルカラーLED
  - 1パッケージに赤，緑，青LED





Ex0804｜リングLEDの点滅

- リングLEDを点滅（FastLEDライブラリ使用）

変更しない

```
#include <FastLED.h>

// リングLED 16個
const int DATA_PIN = 19; // ピン番号
const int NUM_LEDS = 16; // 16個
CRGB leds[NUM_LEDS];    // LEDの配列

void setup() {
  // リングLEDの設定
  FastLED.addLeds<WS2812B, DATA_PIN, GRB>(leds, NUM_LEDS);
  FastLED.setBrightness(32); // 最大輝度：0-255
}
```

Ex0804｜リングLEDの点滅

- LEDの光らせ方をプログラム
- forループ使用
- 配列使用　leds[NUM_LEDS]

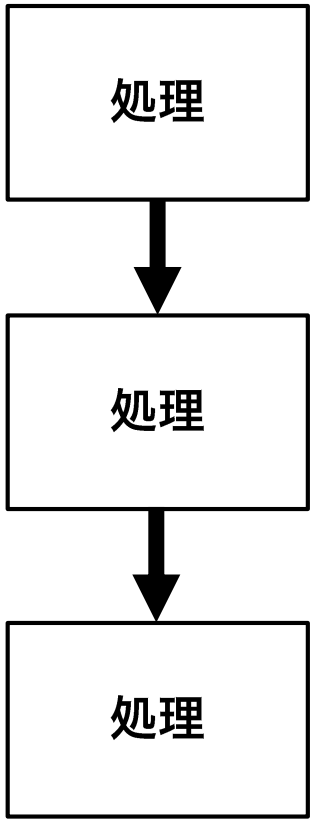
- 配列を使用しない場合

```
leds[0] = CRGB(255,0,0);
leds[1] = CRGB(255,0,0);
leds[2] = CRGB(255,0,0);
        ・・・
leds[13] = CRGB(255,0,0);
leds[14] = CRGB(255,0,0);
leds[15] = CRGB(255,0,0);
```

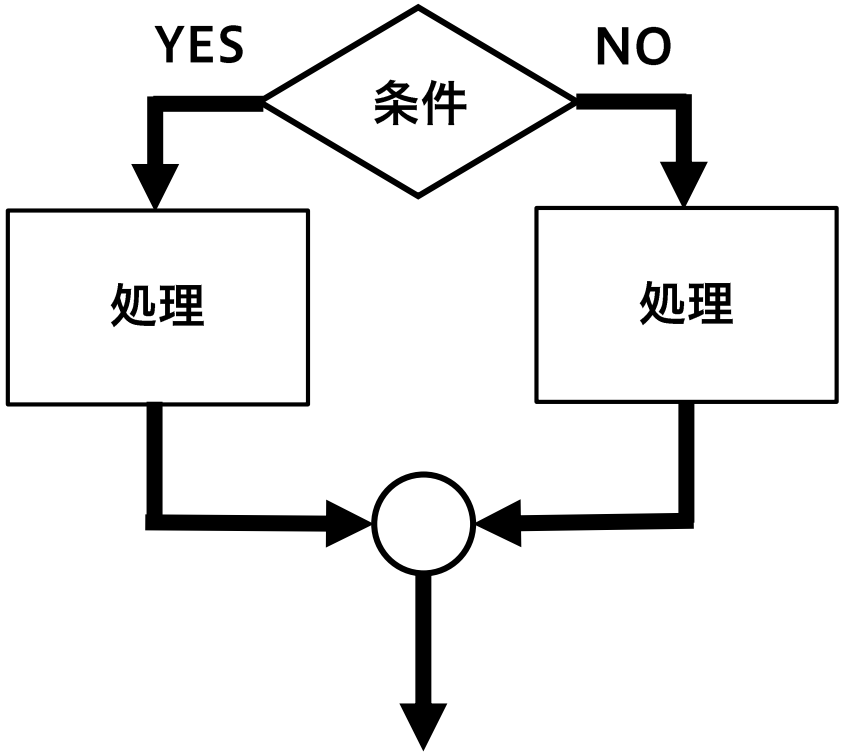
```
void loop() {
 // 赤色で点灯
 for (int i = 0; i < NUM_LEDS; i++) {
  leds[i] = CRGB(255, 0, 0);
 }
 FastLED.show(); // 表示を更新
 delay(1000); // 表示時間

 // 消灯
 for (int i = 0; i < NUM_LEDS; i++) {
  leds[i] = CRGB(0, 0, 0);
 }
 FastLED.show(); // 表示を更新
 delay(1000); // 表示時間
}
```

プログラムの基本構成

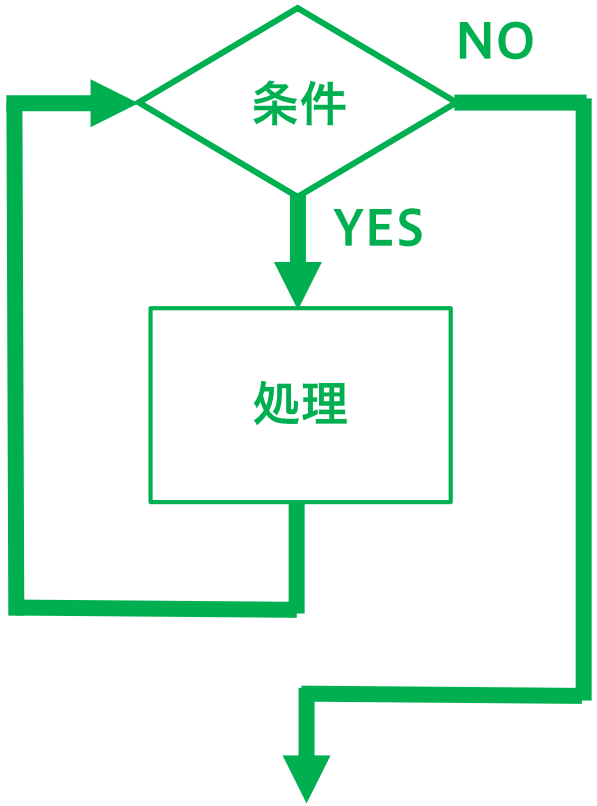
順次



分岐



反復



LEDの表示色の変更

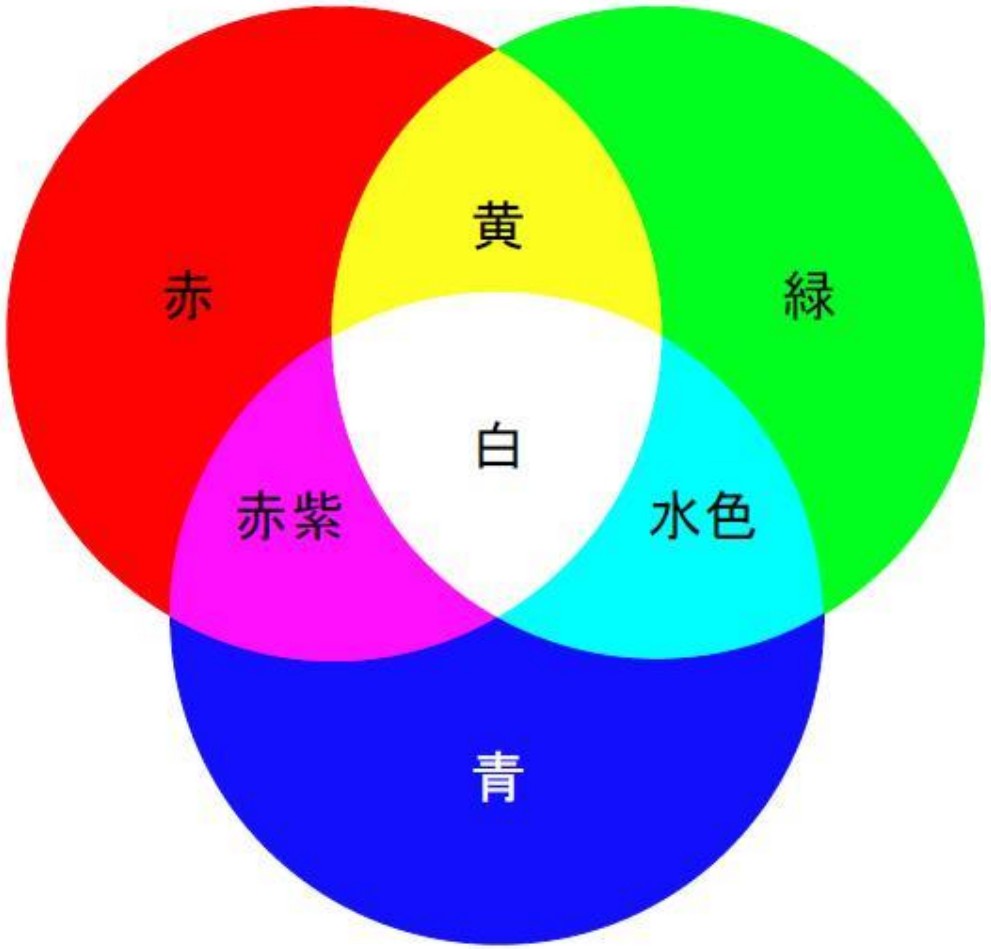
- LEDの色をセット
  - leds[i] = CRGB(R, G, B);
  - leds[i] = CHSV(H, S, V);
- 表示の更新
  - FastLED.show();
- 表示時間
  - delay();

```
// LEDの色をセット
for (int i = 0; i < NUM_LEDS; i++) {
    leds[i] = CRGB(255, 0, 0);
}

// 表示の更新
FastLED.show();

// 表示時間
delay(1000);  // ms
```

光の三原色

- 光の三原色
  - ・赤（Red）
  - ・緑（Green）
  - ・青（Blue）

  の重ね合わせ（加法混色）



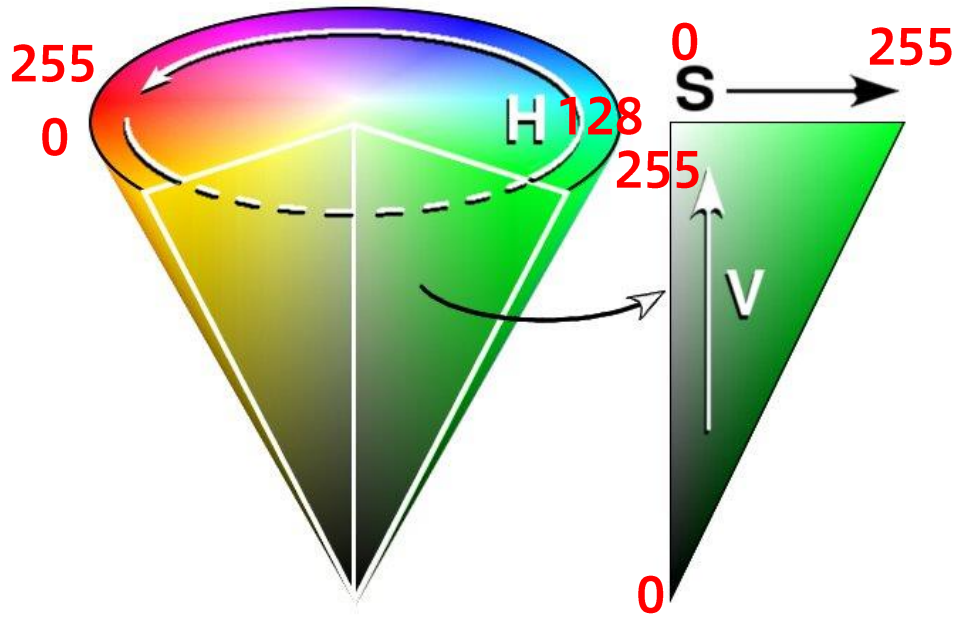
CRGB(R, G, B);

R値：0 – 255 赤の明るさ
G値：0 – 255 緑の明るさ
B値：0 – 255 青の明るさ

HSV表色系

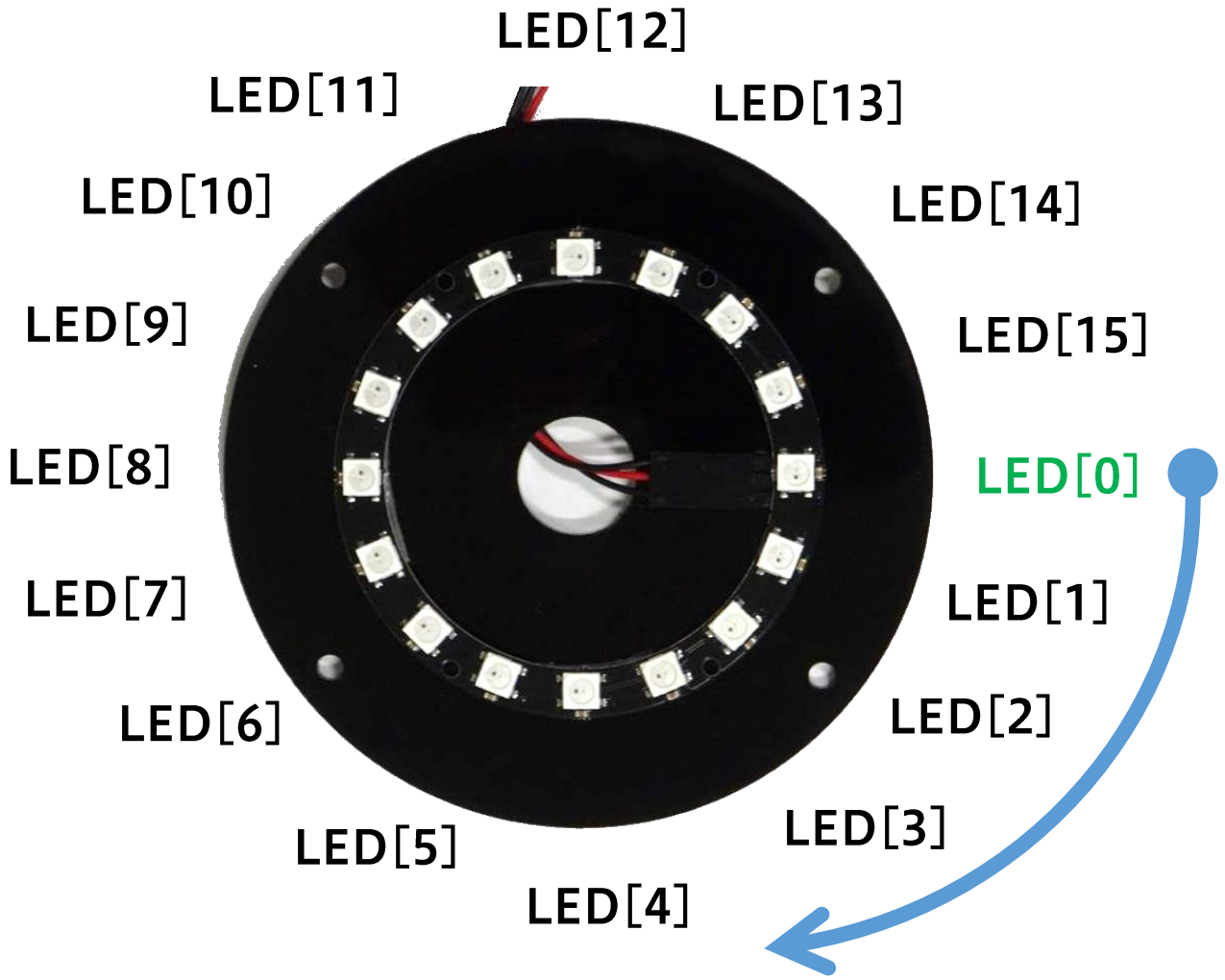
- 色相(H: Hue)
  - 色の種類
- 彩度(S: Saturation)
  - 色の鮮やかさ
- 明度(V: Value, Brightness)
  - 色の明るさ

CHSV(h, s, v);
h値：0 – 255　色相
s値：0 – 255　彩度
v値：0 – 255　明度



HSVの関係

LEDの配置



Ex0805：LEDリモコン

- コントロールパッド（中央ボタン付き）
- ESPUI.pad("操作パネル", true, &padEvent, COLOR_SUNFLOWER);

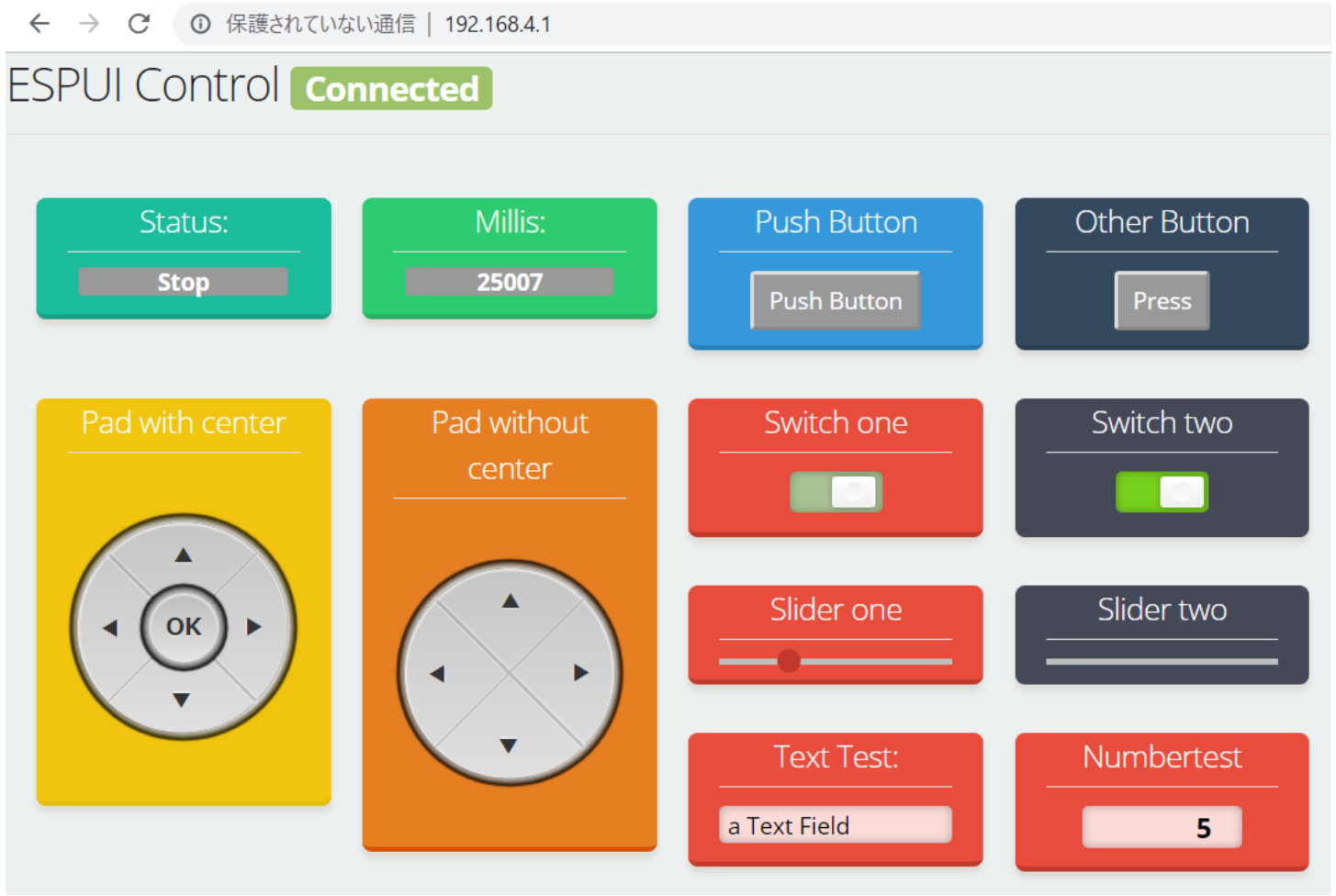
- 上，下ボタン｜明るさ調整
- 左，右ボタン｜色相調整
- 中央ボタン　｜消灯／点灯

考えてみよう

- Ex0801｜cm単位で計測に変更
- Ex0802｜距離の検出範囲を変更
- Ex0803｜他のプログラムにもユーザ認証を追加
- Ex0804｜LEDの色を変更，点灯時間を変更
- Ex0805｜リモコンをスライダとボタンに変更

付録：使用例(espgui)

- ESPGUIのすべての使用例



付録：接続エラー

- 正常にサーバに接続できた場合，
  ESP32 Control : Connected
- エラーが発生した場合，
  Error/No Connection や Offline
  その場合，
  ・ブラウザをリロード
  ・無線AP（SSID）に再接続
- プログラムに誤りがある場合，
  ・修正をして再度書き込み