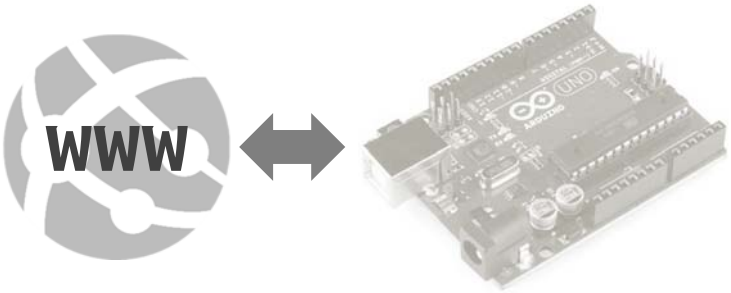
第 8 回

Webからマイコン制御



Arduinoをインターネットにつなぎ、IoTを実現します。
PC、スマホ、マイコン、全てのデバイスがWebにつながり
全てがWebのテクノロジ（HTMLとJavaScript）で動く
次世代Webの基礎を学びましょう！

フィジカルコンピューティング演習
愛知淑徳大学 人間情報学部　　國分 三輝

✈ 8.0 Web からマイコン制御

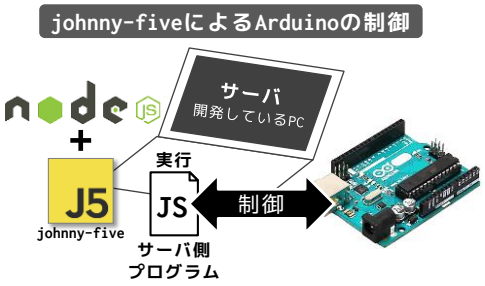
Arduino をインターネットにつなぎ※、IoT を実現します。PC、スマホ、マイコン、全てのデバイスが Web につながり、全てが Web のテクノロジ（HTML と JavaScript）で動く、次世代 Web の基礎を学びましょう！

※ Arduino は直接インターネットにつなげられないので、実際には、サーバ PC 経由でネットにつなげます。

8.1 JavaScript で Arduino

➔ 3 ページ

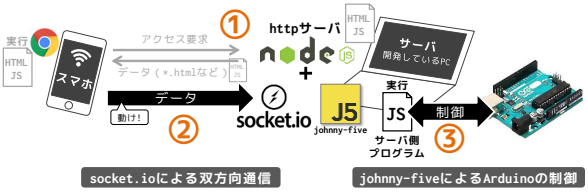
ここまでは専用の開発環境と言語で Arduino を制御しました。ここでは、チャット（双方向通信）プログラミング（第 6 回）で学んだ Node.js を使い、JavaScript で Arduino を制御する方法を学びます。

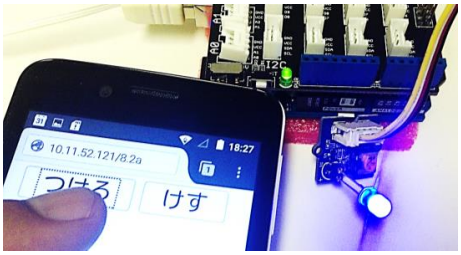


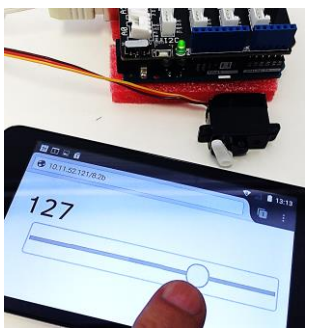
8.2 スマホでコントロール

➔ 11 ページ

Node.js と JavaScript で Arduino が動いたら、第 6 回で学んだ socket.io のプログラムを加えることで、スマホと Arduino の間で双方向通信ができるようになります。ここでは、Arduino につないだアクチュエータをスマホからコントロールする方法を学びます。



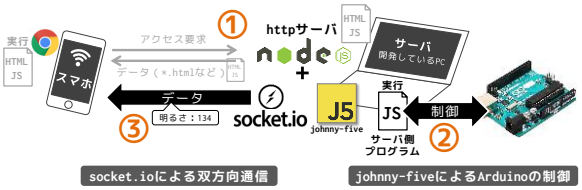


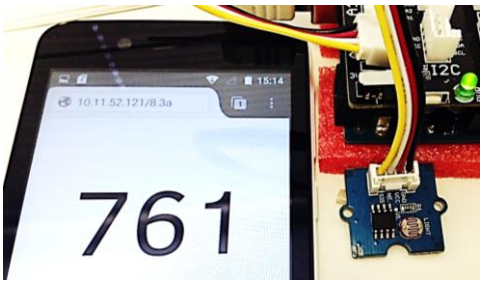


8.3 スマホでセンシング

➔ 21 ページ

Arduino につないだセンサで計測されたデータを、双方向通信を使ってスマホに送ってモニターする方法を学びます。





本日の課題

➔ 28 ページ

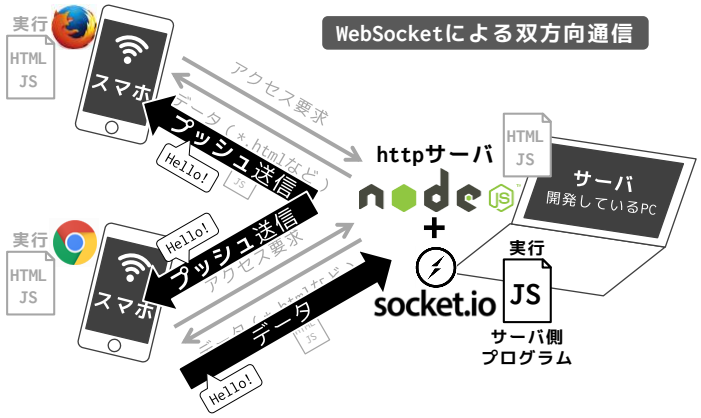
✈ 8.1a JavaScript で L チカ

ここまでは専用の開発環境と言語で Arduino を制御しました。ここでは、チャット（双方向通信）プログラミング（第 6 回）で学んだ Node.js を使い、JavaScript で Arduino を制御する方法を学びます。

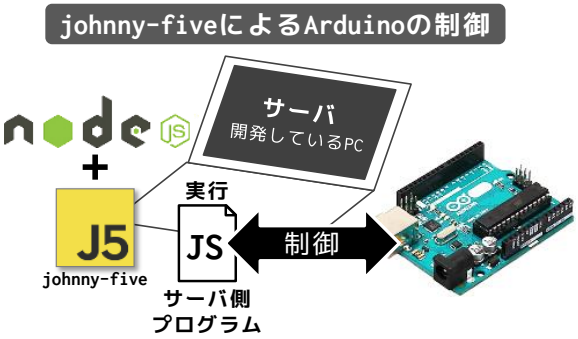
まずは基本中の基本、L チカをやってみましょう！

Learn 始める前に解説 ： Node.js と Arduino の関係

第 6 回で、双方向通信の仕組みを以下のように紹介しました（第 6 回の P.4）。Node.js はサーバ PC 上で JavaScript のプログラムを実行できる環境です。socket.io というライブラリを組み合わせて、双方向通信を実現しました。



今回は、johnny-five※というライブラリを Node.js に組み合わせて、サーバ PC から Arduino を制御します（下図）。



詳しくは後ほど解説しますが、さらに Node.js ＋ johnny-five ＋ socket.io の組み合わせにより、Arduino が JavaScript で制御できるだけでなく、Arduino が Web につながります。

※ **johnny-five:** もともとロボット制御用のライブラリで、Arduino だけでなく様々なマイコンを JavaScript で制御できます。jonny-five のインストール方法は付録 11 で紹介します。また、jonny-five の使い方の詳細は → http://johnny-five.io/

Setup Arduino を PC から（johnny-five で）制御できるように設定する

- Arduino IDE（開発環境）を起動

  スタート　→　すべてのプログラム　→　Arduino

- ファイル → スケッチの例 → Firmata → StandardFirmata を選ぶ※。
- Arduino と PC を USB ケーブルで接続
- Arduino IDE の設定確認
  - ツール → ボード で「Arduino/Genuino Uno」が選択されていることを確認
  - ツール → シリアルポート で「COM ** (Arduino/Genuino Uno)」のように、右側にボードの名前が表示された COM が選択されていることを確認
  - ★ ↑の「COM**」の数字をメモする（後で使います）

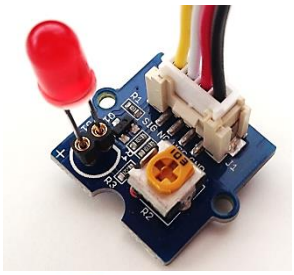
メモ　COM____

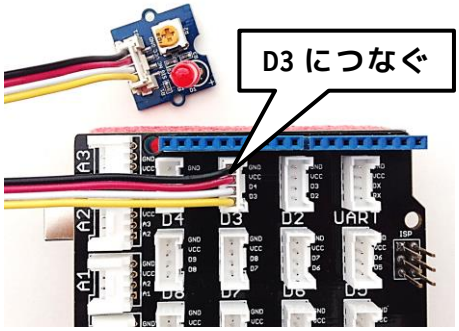
- プログラムを書き込む

※ **Firmata（ファルマータ）:** Arduino に書き込んでおくと、Arduino を PC から操作できるプログラム。Arduino IDE に最初から同梱されている。この作業は最初に 1 回やっておくだけでよい。

Make LED を接続

- Grove の **LED ソケット**を用意します。
- 好きな色の **LED** を選びます。
- LED の**長いほうのピンをソケットの＋**の穴に挿します。
- LED ソケットにワイヤーのコネクタをつなぎます。





- もう一方のコネクタを Grove の **D3** につなぎます。

Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥**の中に **8.1a** というフォルダを作ります。
- 以下のコードを Brackets で入力し、8.1a フォルダ内に **app.js** という名前で保存します。

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

C:￥Users￥（ユーザ名）￥**8.1a**￥ **app.js**

```
// 8.1a JavaScript で L チカ

'use strict';                              // 厳格モードにする

// Johnny-five の準備
const five = require('johnny-five');     // johnny-five モジュールの読み込み                ①
const comPort = 'COM**';                 // ★要書き換え：最初にメモした COM ポート番号     ②
const arduino = new five.Board( {port: comPort} );  // ボードの取得                         ③

// LED の準備と制御
const pinLED = 3;                          // LED を接続したピン番号
let led;                                   // LED 制御用オブジェクトの用意
arduino.on('ready', function() {           // Arduino の準備ができたら                       ④
    led = new five.Led(pinLED);            // LED オブジェクトを取得                         ⑤
    led.blink(100);                        // LED を点滅（間隔を ms で指定）                 ⑥
});
```

Check Arduino で動作確認

- Node.js 用のモジュール（ライブラリ）のコピー
  - CampusSquare から「PHC_08_素材.zip」をダウンロードし、「**8.1 素材.zip**」内にある **node_modules フォルダ（johnny-five ライブラリが入っています）を 8.1a フォルダ内にコピー**
- Arduino と PC を USB ケーブルで接続
- Node.js でサーバ側プログラム（app.js）を起動
  - スタート → すべてのプログラム → Node.js → **Node.js command prompt**
  - プロンプト（>）に続けて順に以下のコマンドを入力して Enter（↲）
    >**cd 8.1a**↲　　… C:￥Users￥（ユーザ名）￥8.1a フォルダに移動
    >**node app.js**↲　… Node.js を使って app.js を実行
- 動作確認
  - 「Connected COM**」および「Repl Initialized」のあとに「>>」が出ればエラー無し
  - Arduino（Grove）につないだ LED が点滅すれば OK

- サーバ側プログラム（app.js）の止め方
  - Node.js command prompt で、キーボードから「**Ctrl + C**」を **2 回**入力

- **動作確認が終わったら USB ケーブルを抜く**

Learn 解　説

① **const five = require('johnny-five');**

johnny-five ライブラリを使うための 1 行です。①～③は johnny-five を使う場合のお決まりの行ですから、この後のプログラムでもコピペして使っていきましょう。

② **const comPort = 'COM**';**

**の部分を、Arduino を USB ケーブルで PC につないだ時の COM ポート番号（P.4 でメモした値）にします。

③ **const arduino = new five.Board({port: comPort});**

johnny-five の Board オブジェクトを作り、②で指定した COM ポートにつながれているマイコンを取得します。

④ **arduino.on('ready', function() { ... });**

プログラムが実行され、Arduino の準備ができると、'ready'イベントが発生します。{ } 内に準備ができた後の処理を書きます。

⑤ **led = new five.Led(pinLED);**

johnny-five の Led オブジェクトを作り、( )内にピン番号を指定します。

⑥ **led.blink(100);**

LED を点滅させる blink(時間[m])メソッドです。100ms 間隔で LED を点滅させています。

Led オブジェクトには他に以下のようなメソッドがあります（主なもののみ）。

メソッド	意味	使用例
on()	LED を点灯させる	led.on();
off()	LED を消灯させる	led.off();
blink()	LED を点滅させる	led.blink();
toggle()	LED が現在 ON なら OFF に、現在 OFF なら ON にする	led.toggle();
brightness()	LED の明るさを変える（PWM: 第 7 回コラム⑱参照）	led.brightness(127);

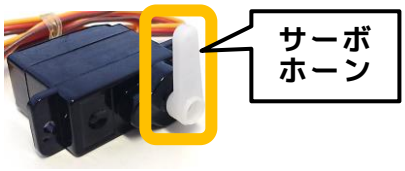
さらに詳しくは → http://johnny-five.io/api/led/

✈ 8.1b JavaScript でサーボ

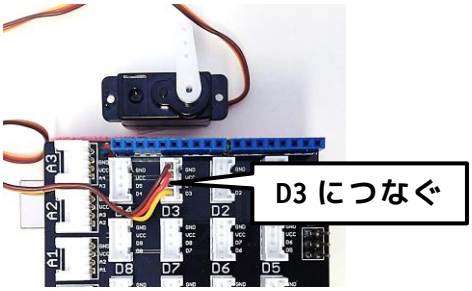
次に、JavaScript でサーボを動かしてみましょう。

Make サーボを接続（他にはなにもつながない）

- **サーボ**を用意します。
- サーボにサーボホーンを装着します。



- ワイヤーを Grove の **D3** につなぎます。



Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.1a をフォルダごと複製**して**フォルダ名を 8.1b に**します。
- app.js を Brackets で開き、以下のようにコードを変更しましょう。**変更は網掛けの部分**です。

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

C:￥Users￥（ユーザ名）￥**8.1b**￥ **app.js**

```
// 8.1b JavaScript でサーボ
'use strict'                              // 厳格モードにする

// Johnny-five の準備
const five = require('johnny-five');     // johnny-five モジュールの読み込み
const comPort = 'COM**';                  // ★要書き換え：Arduino の COM ポート番号
const arduino = new five.Board( {port: comPort} );  // ボードの取得

// サーボの準備と制御
const pinServo = 3;                       // LED を接続したピン番号
let servo;                                // サーボ制御用オブジェクトの用意
arduino.on('ready', function() {          // Arduino の準備ができたら
    servo = new five.Servo(pinServo);     // サーボを取得                    ①
    servo.sweep();                        // サーボを往復させる              ②
});
```

Check Arduino で動作確認

- Node.js 用のモジュール（ライブラリ）のコピー
  - CampusSquare から「PHC_08_素材.zip」をダウンロードし、「**8.1 素材.zip**」内にある **node_modules フォルダ（johnny-five ライブラリが入っています）を 8.1b フォルダ内にコピー**（**8.1a を複製して作業している場合は不要**）
- Arduino と PC を USB ケーブルで接続
- Node.js でサーバ側プログラム（app.js）を起動
  - Node.js command prompt の現在位置が C:￥Users￥（ユーザ名）￥8.1a であることを想定
  - プロンプト（>）に続けて順に以下のコマンドを入力して Enter（↲）

    >**cd..**↲　　… ひとつ上のフォルダに移動（C:￥Users￥（ユーザ名）に移動）

    >**cd 8.1b**↲　　… C:￥Users￥（ユーザ名）￥8.1b フォルダに移動

    >**node app.js**↲　　… Node.js を使って app.js を実行
- 動作確認
  - 「Connected COM**」および「Repl Initialized」のあとに「>>」が出ればエラー無し
  - Arduino（Grove）につないだサーボが繰り返し往復すれば OK
- サーバ側プログラム（app.js）の止め方
  - Node.js command prompt で、キーボードから「**Ctrl + C**」を **2 回**入力
- **動作確認が終わったら USB ケーブルを抜く**

Learn 解　説

① **`servo = new five.Servo(pinServo);`**

johnny-five の Servo オブジェクトを作り、( )内にピン番号を指定します。

② **`servo.sweep();`**

サーボを往復させる sweep()メソッドです。

Servo オブジェクトには他に以下のような命令があります（主なもののみ）。

メソッド	意味	使用例
to()	サーボを指定した角度[deg]に動かす	servo.to(90);
min()	サーボを最小角度に動かす	servo.min();
max()	サーボを最大角度に動かす	servo.max();
center()	サーボを中央地点に動かす	servo.center();
sweep()	サーボを最小角度～最大角度の間で往復させる	servo.sweep();

さらに詳しくは → http://johnny-five.io/api/servo/

✈ 8.1c JavaScript で光センサ

次に、JavaScript で光センサの値を計測します。

Make 光センサを接続（他にはなにもつながない）

- **光センサ**を、Grove の **A0** につなぎます。

←表　裏→

Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.1a をフォルダごと複製**して**フォルダ名を 8.1c に**します。
- app.js を Brackets で開き、以下のようにコードを変更しましょう。**変更は網掛けの部分**です。

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

C:￥Users￥（ユーザ名）￥**8.1c**￥ **app.js**

```
// 8.1c JavaScript で光センサ
'use strict'                              // 厳格モードにする

// Johnny-five の準備
const five = require('johnny-five');     // johnny-five モジュールの読み込み
const comPort = 'COM**';                  // ★要書き換え：Arduino の COM ポート番号
const arduino = new five.Board( {port: comPort} );  // ボードの取得

// 光センサの準備と計測
const pinLight = 'A0';                    // 光センサを接続したピン番号
let light;                                // 光センセ用のオブジェクトを用意
arduino.on('ready', function() {          // Arduino の準備ができたら
    light = new five.Sensor({             // センサを取得                          ①
        pin: pinLight,                    // センサを接続したピン番号
        freq: 250                         // 測定する時間間隔[ms]
    });
    light.on('data', function() {         // データが測定されたら                  ②
        console.log(this.value);          // 値をログに出力する                    ③
    });
});
```

Check Arduino で動作確認

- Node.js 用のモジュール（ライブラリ）のコピー
  - CampusSquare から「PHC_08_素材.zip」をダウンロードし、「**8.1 素材.zip**」内にある **node_modules フォルダ（johnny-five ライブラリが入っています）を 8.1c フォルダ内にコピー**（**8.1a を複製して作業している場合は不要**）
- Arduino と PC を USB ケーブルで接続
- Node.js でサーバ側プログラム（app.js）を起動
  - Node.js command prompt の現在位置が C:￥Users￥（ユーザ名）￥8.1b であることを想定
  - プロンプト（>）に続けて順に以下のコマンドを入力して Enter（↲）

    >**cd..**↲　　… ひとつ上のフォルダに移動（C:￥Users￥（ユーザ名）に移動）

    >**cd 8.1c**↲　　… C:￥Users￥（ユーザ名）￥8.1c フォルダに移動

    >**node app.js**↲　　… Node.js を使って app.js を実行
- 動作確認
  - 「Connected COM**」および「Repl Initialized」のあとに「>>」が出ればエラー無し
  - Node.js command prompt に光センサの計測値が表示されれば OK
- サーバ側プログラム（app.js）の止め方
  - Node.js command prompt で、キーボードから「**Ctrl + C**」を **2 回**入力
- **動作確認が終わったら USB ケーブルを抜く**

Learn 解　説

① `light = new five.Sensor( { ... } );`

johnny-five の Sensor オブジェクトを作ります。{ }内にピン番号や設定値を入れます。

この場合、`pin 属性`にピン番号、`freq 属性`に測定間隔[ms]を入れました。他にも設定可能な属性がありますが詳しくは → http://johnny-five.io/api/sensor/

② `light.on('data', function() { ... });`

センサがデータを取得すると'data'イベントが発生します。{ } 内にデータ取得後の処理を書きます。

③ `console.log(this.value);`

取得されたデータは `this.value` で取り出すことができます。

console.log()でデータをコマンドプロンプトに表示させています。

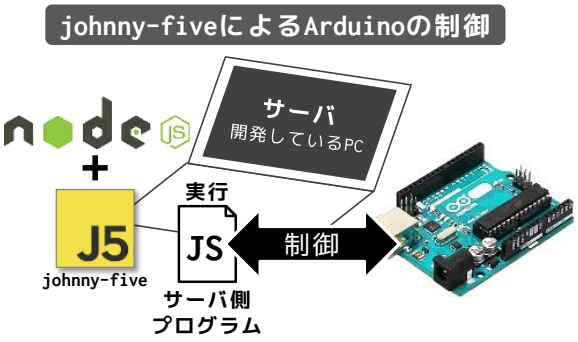
✈ 8.2a スマホで L チカ

Node.js と JavaScript で Arduino が動いたら、第 6 回で学んだ socket.io のプログラムを加えることで、スマホと Arduino の間で双方向通信ができるようになります。

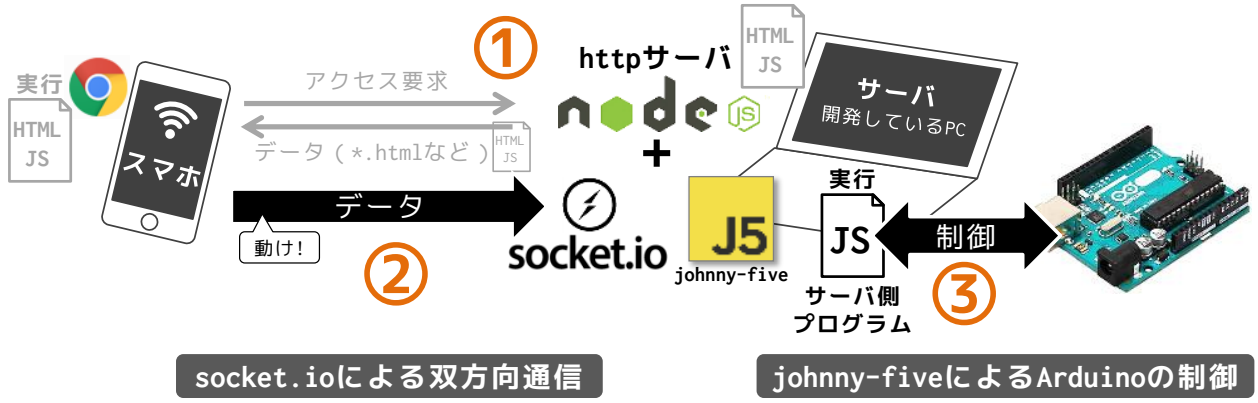
ここでは、Arduino につないだアクチュエータをスマホからコントロールする方法を学びます。まずはスマホから LED を ON/OFF してみましょう。

Learn 始める前に解説 : Node.js + johnny-five + socket.io (コントロールの場合)

8.1 では `Node.js + johnny-five` で Arduino を制御しました (下図)。



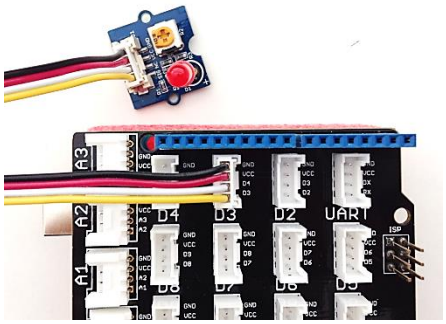
ここではさらに、第 6 回のチャット (双方向通信) プログラミングで用いた socket.io を加えて、`Node.js + johnny-five + socket.io` の組み合わせによりスマホから Arduino につないだアクチュエータをコントロールします (下図)。



スマホはまず①Node.js の http サーバにアクセスして index.html ファイルをゲットします。例えばこの html に置かれたボタンを押すと、②「動け」という命令 (データ) を socket.io でサーバに送信します。データを受け取ったサーバはそのデータの内容に応じて、③johnny-five を使って Arduino をコントロールします。

Make LED を接続（他にはなにもつながない）

- **LED ソケット**を用意します。
- 好きな色の **LED** を選びます。
- LED の**長いほうのピンをソケットの＋**の穴に挿します。
- LED ソケットにワイヤーのコネクタをつなぎます。
- もう一方のコネクタを Grove の **D3** につなぎます。



Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥**の中に **8.2a** というフォルダを作ります。
- **過去のコードからコピペしながら**以下を入力し、8.2a フォルダに **app.js** という名前で保存します。

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

C:￥Users￥（ユーザ名）￥**8.2a**￥ **app.js**

```
// 8.2a スマホで L チカ
'use strict'                              // 厳格モードにする

// Johnny-five の準備（8.1a などからコピペしましょう）
const five = require('johnny-five');     // johnny-five モジュールの読み込み
const comPort = 'COM**';                  // ★要書き換え：Arduino の COM ポート番号
const arduino = new five.Board( {port: comPort} );  // ボードの取得
let arduinoReady = false;                 // Arduino 準備 OK のフラグ（新規）
// LED の準備
const pinLED = 3;                         // LED を接続したピン番号
let led;                                  // LED 制御用オブジェクトの用意
arduino.on('ready', function() {          // Arduino の準備ができたら
    led = new five.Led(pinLED);           // LED を取得
    arduinoReady = true;                  // Arduino 準備 OK（新規）
});

// ソケット通信（socket.io）の準備（6.1 の app.js などからコピペしましょう）
const express = require('express');          // express モジュールを使う
const app = express();                       // express でアプリを作る
app.use(express.static(__dirname));          // ホーム dir にあるファイルを使えるようにする
app.get('/', function (req, res) {           // アクセス要求があったら
    res.sendFile(__dirname + '/index.html');    // index.html を送る
});
const server = require('http').Server(app); // http サーバを起動してアプリを実行
server.listen(80);                           // サーバの 80 番ポートでアクセスを待つ
const io = require('socket.io')(server);     // socket.io モジュールをサーバにつなぐ
                                    // 次のページに続く
```

```
// ソケット通信によるスマホからのコントロール（これ以降はほぼ新規）
io.on('connection', function(socket) {      // socket 接続があって
    socket.on('on', function() {            // 'on'というイベントの socket が来たら      ①
        if (arduinoReady == true) {         // Arduino が準備 OK なら
            led.on();                       // LED を点灯                                ②
        }
    });
    socket.on('off', function() {           // 'off'というイベントの socket が来たら     ③
        if (arduinoReady == true) {         // Arduino が準備 OK なら
            led.off();                      // LED を消灯                                ④
        }
    });
});
```

Input クライアント側プログラム（HTML・JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.2a** の中に 1.1 で作った index.html（テンプレート）をコピー。
- この index.html を Brackets で開き、以下のコードを入力しましょう。

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

C:￥Users￥（ユーザ名）￥**8.2a**￥ **index.html**

```
（略）
        <title>8.2a スマホで L チカ</title>
    </head>
    <body>
        <input type='button' id='btnOn' value='つける'>         <!-- ON ボタン -->
        <input type='button' id='btnOff' value='けす'>          <!-- OFF ボタン -->
        <script src='/socket.io/socket.io.js'></script>         <!-- socket.io -->

        <script>
            const socket = io();                // 双方向通信用のサーバに接続

            const btnOn = document.getElementById('btnOn');     // ON ボタンを取得
            const btnOff = document.getElementById('btnOff');   // OFF ボタンを取得
            btnOn.addEventListener('click', ledOn);             // btnOn のイベント
            btnOff.addEventListener('click', ledOff);           // btnOff のイベント

            function ledOn() {      // ON ボタンがクリックされた時の処理
                socket.emit('on');  // 'on'というイベント名のソケットを送る              ⑤
            }
            function ledOff() {     // OFF ボタンがクリックされた時の処理
                socket.emit('off'); // 'off'というイベント名のソケットを送る             ⑥
            }
        </script>
（以下略）
```

Check スマホと Arduino で動作確認

- Node.js 用のモジュール（ライブラリ）のコピー
  - CampusSquare からダウンロードした「PHC_08_素材.zip」内の「**8.2 素材.zip**」内にある **node_modules フォルダ(johnny-five、socket.io、express ライブラリが入っています)を 8.2a フォルダ内にコピー**
- Arduino と PC を USB ケーブルで接続
- Node.js でサーバ側プログラム（app.js）を起動
  - Node.js command prompt の現在位置が C:￥Users￥（ユーザ名）￥8.1c であることを想定
  - プロンプト（>）に続けて順に以下のコマンドを入力して Enter（↲）

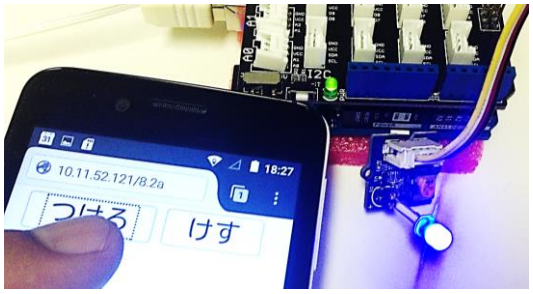
    >**ipconfig**↲　　… サーバ PC の IPv4 アドレスをメモ（★）

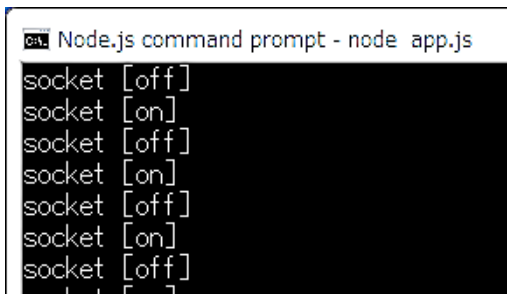
    >**cd..**↲　　… ひとつ上のフォルダに移動（C:￥Users￥（ユーザ名）に移動）

    >**cd 8.2a**↲　　… C:￥Users￥（ユーザ名）￥8.2a フォルダに移動

    >**node app.js**↲　　… Node.js を使って app.js を実行
- 動作確認
  - 「Connected COM**」および「Repl Initialized」のあとに「>>」が出ればエラー無し
  - スマホの Firefox で、メモしたアドレス（★）を入力して Enter

    アドレス例： **10.11.123.456**
  - つけるボタン ➔ LED が点灯、プロンプトに socket [on] と表示
  - けすボタン ➔ LED が消灯、プロンプトに socket [off] と表示 されれば OK





- サーバ側プログラム（app.js）の止め方
  - Node.js command prompt で、キーボードから「**Ctrl + C**」を **2 回**入力
- **動作確認が終わったら USB ケーブルを抜く**

これで、地球の裏側からでも照明が ON/OFF できる！※

※ 現在の仕様・環境だと学内からのみコントロールできます。しかし、原理的には可能です。

Learn 解　説

サーバ側プログラム（app.js）

① `socket.on('on', function() { ... });`

スマホから'on'というイベント名のソケットが届いた時の処理です。
ここでは、以下の②で LED を点灯させています。

② `led.on();`

LED を点灯させる、johnny-five の Led オブジェクトの on()メソッドです（P.6 参照）。

③ `socket.on('off', function() { ... });`

スマホから'off'というイベント名のソケットが届いた時の処理です。
ここでは、以下の④で LED を消灯させています。

④ `led.off();`

LED を消灯させる、johnny-five の Led オブジェクトの off()メソッドです（P.6 参照）。

クライアント側プログラム（index.html）

⑤ `socket.emit('on');`

つける と名付けたボタン btnOn がクリックされたら、'on'というイベントをソケットで送信しています。このソケットにより、サーバ側プログラムの①が動き、LED が点灯します。

⑥ `socket.emit('off');`

けす と名付けたボタン btnOff がクリックされたら、'off'というイベントをソケットで送信しています。このソケットにより、サーバ側プログラムの③が動き、LED が消灯します。

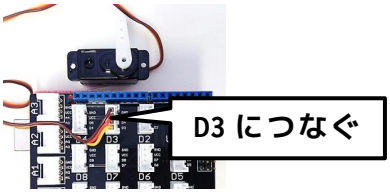
memo

✈ 8.2b スマホでサーボ

次に、スマホからサーボを制御してみましょう。

Make サーボを接続（他にはなにもつながない）

- **サーボ**を Grove の **D3** につなぎます。



Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.2a をフォルダごと複製**して**フォルダ名を 8.2b に**します。
- この app.js を Brackets で開き、**網かけの部分を書き換え**ましょう。

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

C:￥Users￥（ユーザ名）￥**8.2b**￥ **app.js**

```
// 8.2b スマホでサーボ
'use strict'                            // 厳格モードにする

// Johnny-five の準備
const five = require('johnny-five');    // johnny-five モジュールの読み込み
const comPort = 'COM**';                // ★要書き換え：Arduino の COM ポート番号
const arduino = new five.Board( {port: comPort} );  // ボードの取得
let arduinoReady = false;               // Arduino 準備 OK のフラグ（新規）
// サーボの準備
const pinServo = 3;                     // サーボを接続したピン番号
let servo;                              // サーボ制御用オブジェクトの用意
arduino.on('ready', function() {        // Arduino の準備ができたら
    servo = new five.Servo({            // サーボオブジェクトを取得                ①
        pin: pinServo,                  // サーボをつないだピン番号                ②
        invert: true                    // 回転を逆転させる
    });
    arduinoReady = true;                // Arduino 準備 OK（新規）
});

/// ソケット通信（socket.io）の準備
const express = require('express');         // express モジュールを使う
const app = express();                      // express でアプリを作る
app.use(express.static(__dirname));         // ホーム dir にあるファイルを使えるようにする
app.get('/', function (req, res) {          // アクセス要求があったら
    res.sendFile(__dirname + '/index.html');    // index.html を送る
});
                                    // 次のページに続きます
```

```
const server = require('http').Server(app); // http サーバを起動してアプリを実行
server.listen(80);                          // サーバの 80 番ポートでアクセスを待つ
const io = require('socket.io')(server);    // socket.io モジュールをサーバにつなぐ

// ソケット通信によるスマホからのコントロール
io.on('connection', function(socket) {      // socket 接続があって
    socket.on('servo', function(data) {     // 'servo'というイベントの socket が来たら    ③
        if (arduinoReady == true) {         // Arduino が準備 OK なら
            servo.to(data.angle);           // サーボを指定角度に動かす                   ④
        }
    });
});
```

Input クライアント側プログラム（HTML・JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.2b** の中の index.html を Brackets で開き、以下の**網かけの部分を書き換え**ましょう。

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

C:￥Users￥（ユーザ名）￥**8.2b**￥ **index.html**

```
（略）
        <title>8.2b スマホでサーボ</title>
    </head>
    <body>
        <div id='val'>ここに角度を表示</div>
        <input type='range' min='0' max='179' value='90' id='angle'>  <!-- スライダー -->
        <script src='/socket.io/socket.io.js'></script>               <!-- socket.io -->

        <script>
            const socket = io();                // 双方向通信用のサーバに接続

            const slider = document.getElementById('angle');    // スライダーを取得
            slider.addEventListener('change', ctrlServo);       // スライダーのイベント

            function ctrlServo() {              // スライダーが変化した時の処理
                const val = document.getElementById('val');     // div 要素を取得
                val.innerHTML = slider.value;                   // 角度の表示
                socket.emit('servo', {          // servo というイベント名で              ⑤
                    angle: slider.value         // angle 属性に角度を入れて送信          ⑥
                });
            }
        </script>
（以下略）
```

Check スマホと Arduino で動作確認

- Node.js 用のモジュール（ライブラリ）のコピー
  - CampusSquare からダウンロードした「PHC_08_素材.zip」内の「**8.2 素材.zip**」内にある **node_modules フォルダ(johnny-five、socket.io、express ライブラリが入っています)を 8.2b フォルダ内にコピー（8.2a を複製して作業している場合は不要）**

- Arduino と PC を USB ケーブルで接続

- Node.js でサーバ側プログラム（app.js）を起動
  - Node.js command prompt の現在位置が C:￥Users￥（ユーザ名）￥8.2a であることを想定
  - プロンプト（>）に続けて順に以下のコマンドを入力して Enter（↲）

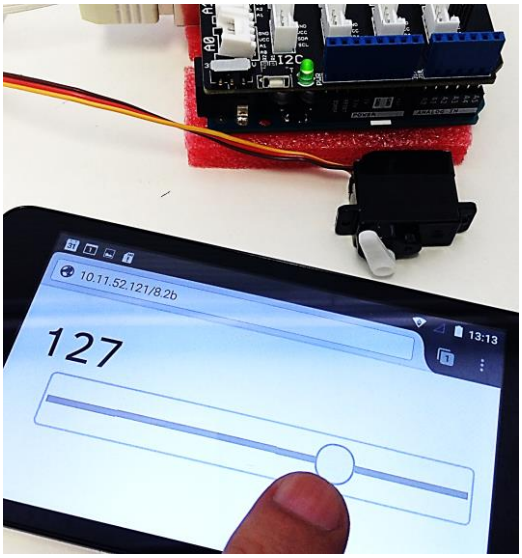
    >**cd..**↲　　… ひとつ上のフォルダに移動（C:￥Users￥（ユーザ名）に移動）

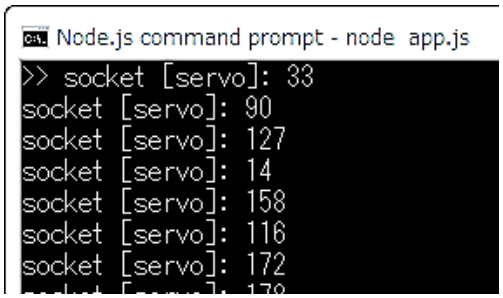
    >**cd 8.2b**↲　　… C:￥Users￥（ユーザ名）￥8.2b フォルダに移動

    >**node app.js**↲　　… Node.js を使って app.js を実行

- 動作確認
  - 「Connected COM**」および「Repl Initialized」のあとに「>>」が出ればエラー無し
  - スマホの Firefox で、サーバのアドレスを入力して Enter

    アドレス例：**10.11.123.456**
  - スライダー操作　➔ サーボが動き、プロンプトに socket [servo]: xx と表示されれば OK





- サーバ側プログラム（app.js）の止め方
  - Node.js command prompt で、キーボードから「**Ctrl + C**」を **2 回**入力

- **動作確認が終わったら USB ケーブルを抜く**

Learn 解　説

サーバ側プログラム（app.js）

① `servo = new five.Servo( {...設定...} );`

8.1b では five.Servo()の( )内にはピン番号のみを指定しました。②のように{ }内に設定値を複数書いて指定することで、サーボの動作を詳細に設定できます。

設定できる値について詳しくは → http://johnny-five.io/api/servo/

② `{ pin: pinServo, invert: true}`

pin: はピン番号の設定、invert: はサーボの回転方向を逆転させる設定（true/false）です。

③ `socket.on('servo', function(data) { ... });`

スマホから'servo'というイベント名のソケットが届いた時の処理です。

ここでは、以下の④でサーボを指定角度に動かしています。

④ `servo.to(data.angle);`

サーボを指定角度に動かす、johnny-five の Servo オブジェクトの to(0～179 の数値)メソッドです（P.8 参照）。スマホから送られてきたデータの angle 属性の値の角度（下記⑤⑥を参照）にしています。

クライアント側プログラム（index.html）

⑤ `socket.emit('servo' { ... });`

スライダーの値が変化したら、'servo'というイベントをソケットで送信しています。また、{ }内のデータを付けて送っています。このソケットによりサーバ側プログラムの③が動き、サーボの角度が変わります。

⑥ `{ angle: slider.value }`

⑤で'servo'というイベント名で送るデータです。angle という属性にスライダーの値（0～179）を代入しています。

これで、地球の裏側から何かを動かすことができる！※

※ 現在の仕様・環境だと学内からのみコントロールできます。しかし、原理的には可能です。

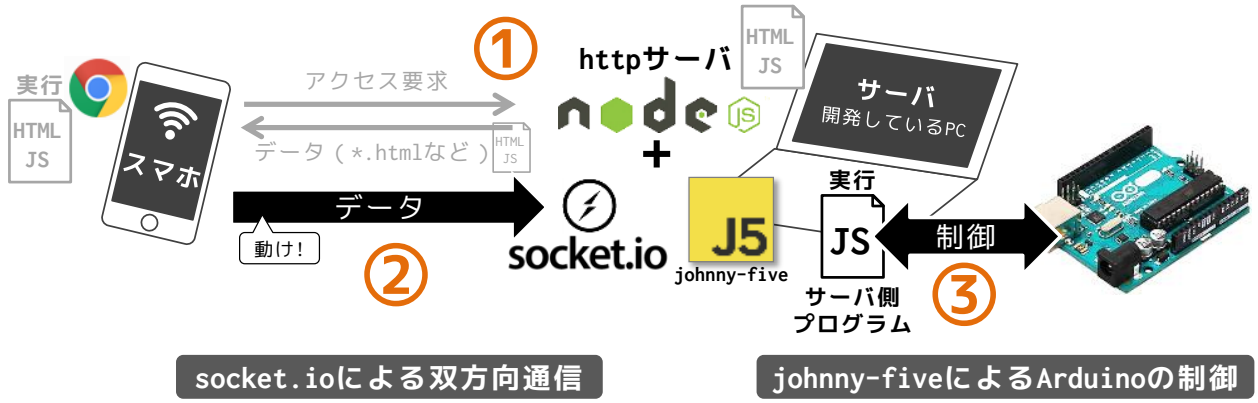
✈ 8.3a スマホで光センサ

Arduino につないだセンサで計測されたデータを、双方向通信を使ってスマホに送ってモニターする方法を学びます。

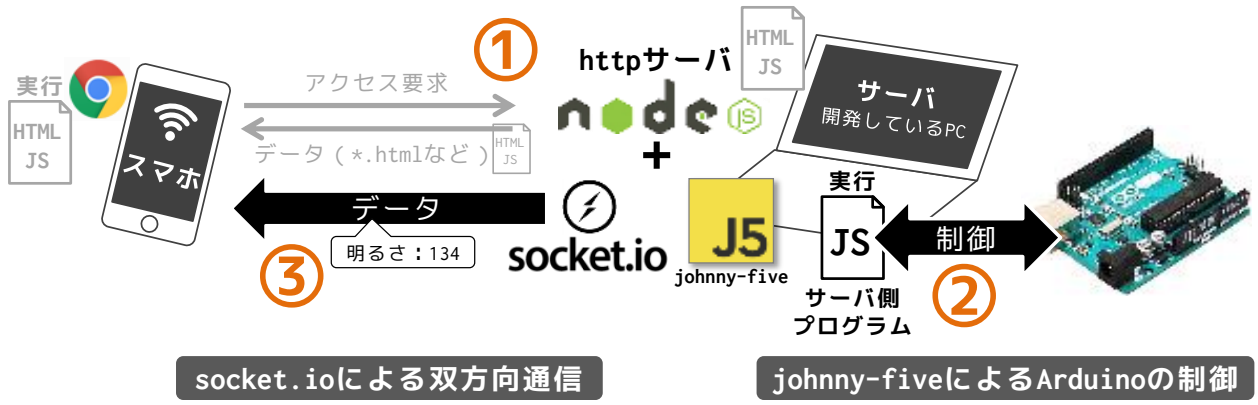
ここでは、Arduino につないだ光センサの計測値をスマホでモニターしてみましょう。

Learn 始める前に解説 : Node.js + johnny-five + socket.io (センシングの場合)

8.2 では `Node.js + johnny-five + socket.io` の組み合わせによりスマホから Arduino につないだアクチュエータをコントロールしました (下図)。



ここではさらに、Arduino につないだセンサの計測値をスマホでモニターします (下図)。



スマホはまず①Node.js の http サーバにアクセスして index.html ファイルをゲットします。サーバ側プログラムは②johnny-five を使って Arduino につながれた例えば光センサで明るさを計測し、③明るさの値 socket.io でスマホに送信します。

Make 光センサを接続（他にはなにもつながない）

- **光センサ**を、Grove の **A0** につなぎます。

←表　裏→

Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.2b をフォルダごと複製**して**フォルダ名を 8.3a に**します。
- この app.js を Brackets で開き、**網かけの部分を書き換え**ましょう。

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

C:￥Users￥（ユーザ名）￥**8.3a**￥ **app.js**

```
// 8.3a スマホでセンシング（光）
'use strict'                            // 厳格モードにする

// Johnny-five の準備
const five = require('johnny-five');    // johnny-five モジュールの読み込み
const comPort = 'COM**';                // ★要書き換え：Arduino の COM ポート番号
const arduino = new five.Board( {port: comPort} );  // ボードの取得
let arduinoReady = false;               // Arduino 準備 OK のフラグ
// 光センサの準備
const pinLight = 'A0';                  // 光センサを接続したピン番号
let light;                              // 光センセ用のオブジェクトを用意
arduino.on('ready', function() {        // Arduino の準備ができたら
    light = new five.Sensor({           // センサオブジェクトを作る
        pin: pinLight,                  // センサを接続したピン番号
        freq: 250                       // 測定する時間間隔[ms]
    });
    arduinoReady = true;                // Arduino 準備 OK
});

// ソケット通信（socket.io）の準備
const express = require('express');         // express モジュールを使う
const app = express();                      // express でアプリを作る
app.use(express.static(__dirname));         // ホーム dir にあるファイルを使えるようにする
app.get('/', function (req, res) {          // アクセス要求があったら
    res.sendFile(__dirname + '/index.html');    // index.html を送る
});
const server = require('http').Server(app); // http サーバを起動してアプリを実行
server.listen(80);                          // サーバの 80 番ポートでアクセスを待つ
const io = require('socket.io')(server);    // socket.io モジュールをサーバにつなぐ

                                    // 次のページに続く
```

```
// ソケット通信によるスマホでのセンシング
io.on('connection', function(socket) {      // socket 接続があって
    if (arduinoReady == true) {             // Arduino が準備 OK なら
        light.on('data', function() {       // データが測定されたら
            console.log(this.value);        // console に出力する
            io.sockets.emit('sensor', {     // sensor というイベント名でソケット配信      ①
                brightness: this.value      // brightness 属性に値を入れて
            });
        });
    }
});
```

Input クライアント側プログラム（HTML・JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.3a** の中の index.html を Brackets で開き、以下の**網かけの部分を書き換え**ましょう。

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

C:￥Users￥（ユーザ名）￥**8.3a**￥ **index.html**

```
（略）
        <title>8.3a スマホでセンシング（光）</title>
    </head>
    <body>
        <div id='val'>ここに測定値を表示</div>
        <script src='/socket.io/socket.io.js'></script>              <!-- socket.io -->

        <script>
            const socket = io();    // 双方向通信用のサーバに接続

            socket.on('sensor', function(data) {    // 'sensor'というイベントが来たら    ②
                const val = document.getElementById('val'); // div 要素の取得
                val.innerHTML = data.brightness;            // データの表示
            });
        </script>
（以下略）
```

Check スマホと Arduino で動作確認

- Node.js 用のモジュール（ライブラリ）のコピー
  - CampusSquare からダウンロードした「PHC_08_素材.zip」内の「**8.2 素材.zip**」内にある **node_modules フォルダ(johnny-five、socket.io、express ライブラリが入っています)を 8.3a フォルダ内にコピー**
- Arduino と PC を USB ケーブルで接続

- Node.js でサーバ側プログラム（app.js）を起動
  - Node.js command prompt の現在位置が C:￥Users￥（ユーザ名）￥8.2b であることを想定
  - プロンプト（>）に続けて順に以下のコマンドを入力して Enter（↲）

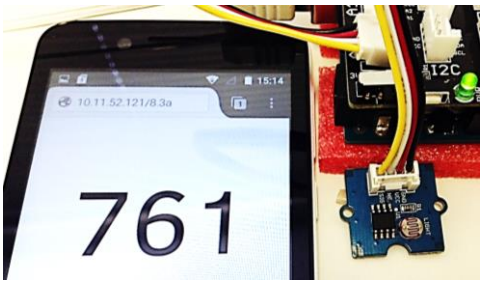
    >cd..↲　　… ひとつ上のフォルダに移動（C:￥Users￥（ユーザ名）に移動）

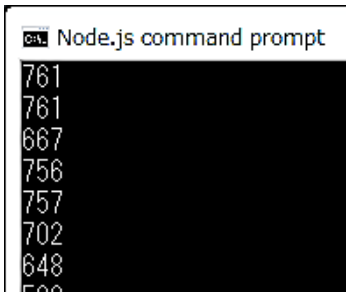
    >cd 8.3a↲　　… C:￥Users￥（ユーザ名）￥8.3a フォルダに移動

    >node app.js↲　　… Node.js を使って app.js を実行

- 動作確認
  - 「Connected COM**」および「Repl Initialized」のあとに「>>」が出ればエラー無し
  - スマホの Firefox で、サーバのアドレスを入力して Enter

    アドレス例：**10.11.123.456**
  - スマホに計測値が表示され、プロンプトにも計測値が流れれば OK





- サーバ側プログラム（app.js）の止め方
  - Node.js command prompt で、キーボードから「**Ctrl + C**」を **2 回**入力

- **動作確認が終わったら USB ケーブルを抜く**

これで、地球の裏側からでも自室の明るさがわかる！※

※ 現在の仕様・環境だと学内でのみモニターできます。しかし、原理的には可能です。

Learn 解　説

① `io.sockets.emit('sensor', { brightness: this.value });`

計測されたデータを、'sensor'というイベント名で、brightness 属性に計測値を入れて、全てのクライアントにソケット送信しています。

② `socket.on('sensor', function(data) { ... });`

①のデータが届いたら div 要素内に brightness 属性の値を表示しています。

✈ 8.3b スマホでタッチセンサ

次に、Arduino につないだタッチセンサの計測値をスマホでモニターしてみましょう。

Make タッチセンサを接続（他にはなにもつながない）

- **タッチセンサ**を、Grove の **D2** につなぎます。

Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.3a をフォルダごと複製**して**フォルダ名を 8.3b に**します。
- app.js を Brackets で開き、**網かけの部分を書き換え**ましょう。

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

C:￥Users￥（ユーザ名）￥**8.3b**￥ **app.js**

```
// 8.3b スマホでセンシング（タッチ）
'use strict'                              // 厳格モードにする

// Johnny-five の準備
const five = require('johnny-five');     // johnny-five モジュールの読み込み
const comPort = 'COM**';                  // ★要書き換え：Arduino の COM ポート番号
const arduino = new five.Board( {port: comPort} );  // ボードの取得
let arduinoReady = false;                 // Arduino 準備 OK のフラグ
// タッチセンサの準備
const pinTouch = 2;                       // タッチセンサを接続したピン番号
let touch;                                // タッチセンセ用のオブジェクトを用意
arduino.on('ready', function() {          // Arduino の準備ができたら
    touch = new five.Button(pinTouch);    // タッチセンサ（ボタン）を取得          ①
    arduinoReady = true;                  // Arduino 準備 OK
});


// ソケット通信（socket.io）の準備
const express = require('express');          // express モジュールを使う
const app = express();                       // express でアプリを作る
app.use(express.static(__dirname));          // ホーム dir にあるファイルを使えるようにする
app.get('/', function (req, res) {           // アクセス要求があったら
    res.sendFile(__dirname + '/index.html');    // index.html を送る
});
const server = require('http').Server(app); // http サーバを起動してアプリを実行
server.listen(80);                           // サーバの 80 番ポートでアクセスを待つ
const io = require('socket.io')(server);     // socket.io モジュールをサーバにつなぐ

                                  // 次のページに続く
```

```
// ソケット通信によるスマホでのセンシング
io.on('connection', function(socket) {      // socket 接続があって
    if (arduinoReady == true) {             // Arduino が準備 OK なら
        touch.on('press', function() {      // タッチセンサ（ボタン）が押されたら        ②
            console.log('pressed');         // console に出力する
            io.sockets.emit('sensor', {     // sensor というイベント名でソケット配信
                touched: true               // touched 属性に true を入れて
            });
        });
        touch.on('release', function() {    // タッチセンサ（ボタン）が離されたら        ③
            console.log('released');        // console に出力する
            io.sockets.emit('sensor', {     // sensor というイベント名でソケット配信
                touched: false              // touched 属性に false を入れて
            });
        });
    }
});
```

Input クライアント側プログラム（HTML・JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥8.3b** の中の index.html の**網かけの部分を書き換え**ましょう。

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

C:￥Users￥（ユーザ名）￥**8.3b**￥ **index.html**

```
（略）
        <title>8.3b スマホでセンシング（タッチ）</title>
    </head>
    <body>
        <div id='val'>ここに測定値を表示</div>
        <!-- 双方向通信（socket 通信）のための socket.io ライブラリの読み込み -->
        <script src='/socket.io/socket.io.js'></script>

        <script>
            const socket = io();                    // 双方向通信用のサーバに接続

            socket.on('sensor', function(data) {    // 'sensor'というイベントが来たら
                const val = document.getElementById('val'); // div 要素の取得
                val.innerHTML = data.touched;               // データの表示
            });
        </script>
（以下略）
```

Check スマホと Arduino で動作確認

- Node.js 用のモジュール（ライブラリ）のコピー
  - CampusSquare からダウンロードした「PHC_08_素材.zip」内の「**8.2 素材.zip**」内にある **node_modules フォルダ(johnny-five、socket.io、express ライブラリが入っています)を 8.3b フォルダ内にコピー（8.3a を複製して作業している場合は不要）**
- Arduino と PC を USB ケーブルで接続
- Node.js でサーバ側プログラム（app.js）を起動
  - Node.js command prompt の現在位置が C:￥Users￥（ユーザ名）￥8.3a であることを想定
  - プロンプト（>）に続けて順に以下のコマンドを入力して Enter（↲）

    >**cd..**↲　　… ひとつ上のフォルダに移動（C:￥Users￥（ユーザ名）に移動）

    >**cd 8.3b**↲　　… C:￥Users￥（ユーザ名）￥8.3b フォルダに移動

    >**node app.js**↲　… Node.js を使って app.js を実行
- 動作確認
  - 「Connected COM**」および「Repl Initialized」のあとに「>>」が出ればエラー無し
  - スマホの Firefox で、サーバのアドレスを入力して Enter

    アドレス例： **10.11.123.456**
  - スマホに計測値（true/false）が表示され、プロンプトにも計測値が出れば OK
- サーバ側プログラム（app.js）の止め方
  - Node.js command prompt で、キーボードから「**Ctrl + C**」を **2 回**入力
- **動作確認が終わったら USB ケーブルを抜く**

Learn 解　説

① `touch = new five.Button(pinTouch);`

タッチセンサのように ON/OFF のみを測るセンサは johnny-five の Button オブジェクトを使います。( )にはピン番号を指定します。

② `touch.on('press', function() { ... });`

Button オブジェクトが ON になると（タッチセンサに触れると）'press'イベントが発生します。ここでは、センサに触れたら、touched 属性に true という値を入れてソケットで送信しています。

③ `touch.on('release', function() { ... });`

Button オブジェクトが OFF になると（タッチセンサから指を離すと）'release'イベントが発生します。ここでは、センサから指が離れたら、touched 属性に false という値を入れてソケットで送信しています。

データを持ち帰る

- 本日作成したデータは全て USB メモリなどにコピーして持ち帰りましょう
  - 任意の場所に作成した以下のフォルダ（9 個）
  - 8.1a/b/c　8.2a/b　8.3a/b

本日の課題

① 8.1a、8.2a、8.3a のコードを完成させる

・ 資料のコードをもとに自分なりのアレンジを加えても構いません。その際は②の readme.txt の所感にアレンジ内容を書いてください。効果的なアレンジであれば成績評価に加点します。

② readme.txt というテキストファイルを作り、以下を書く

・ 学籍番号 と 氏名
・ 所感　（考えたこと、感想、応用アイデアなど。字数不問だが数行は書いてほしい。）
・ 質問　（無ければ不要）

【提出方法】

・ ①のフォルダ 3 個と、②の「readme.txt」1 個、全てをまとめて zip 圧縮
・ zip のファイル名は「学籍番号.zip」とする　（例：12345nhu.zip）
・ CampusSquare のレポート「フィジカルコンピューティング **08**」から提出

【提出締切】　12 月 1 日（木）　15:00（遅れてしまった場合は担当教員に相談のこと）