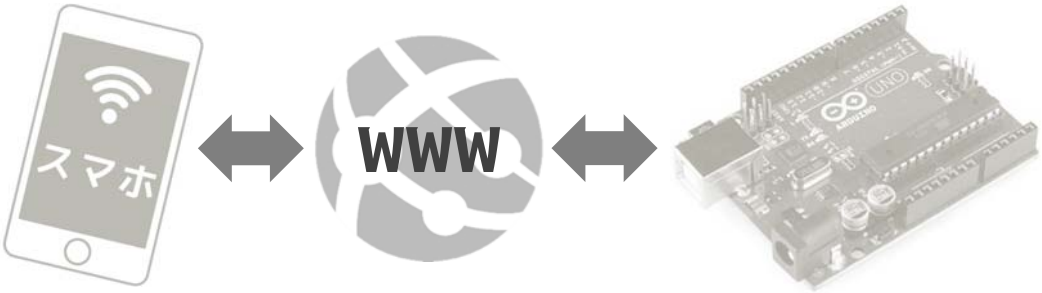
第 9 回

スマホとマイコンで遠隔監視・制御



ここまでに学んだ様々なテクニックを組み合わせて
自宅の様子をスマホで監視して、悪者に警告を発したり
大切なものに触れられたことを知らせてくれる
遠隔監視・制御システムを作ります

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

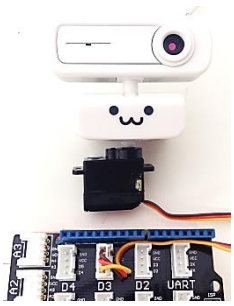
✈ 9.0 スマホとマイコンで遠隔監視・制御

ここまでに学んだ様々なテクニックを組み合わせて、自宅の様子をスマホで監視して、悪者に警告を発したり、大切なものに触れられたことを知らせてくれる、遠隔監視・制御システムを作ります

9.1 スマホで遠隔監視

➔ 3 ページ

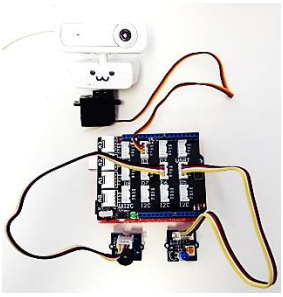
自宅に置いたカメラの映像をスマホで監視するシステムを作ります。Arduino につないだサーボにカメラを搭載して、スマホからカメラの向きをコントロールします。さらに、その画像を保存することもできます。



9.2 遠隔監視と警告

➔ 9 ページ

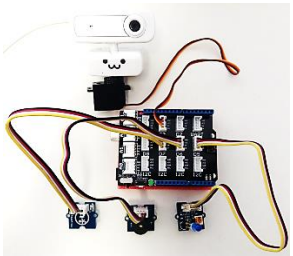
9.1 の遠隔監視システムに、スマホから警告を発する機能を付け加えます。自宅に忍び込んだ悪者を光の点滅とブザーで威嚇します。



9.3 遠隔監視と警告と危険お知らせ

➔ 15 ページ

9.2 のシステムに、大切なものに触れられた時にスマホから音を出したり振動させたりする危険お知らせ機能を付け加えます。



本日の課題

➔ 20 ページ

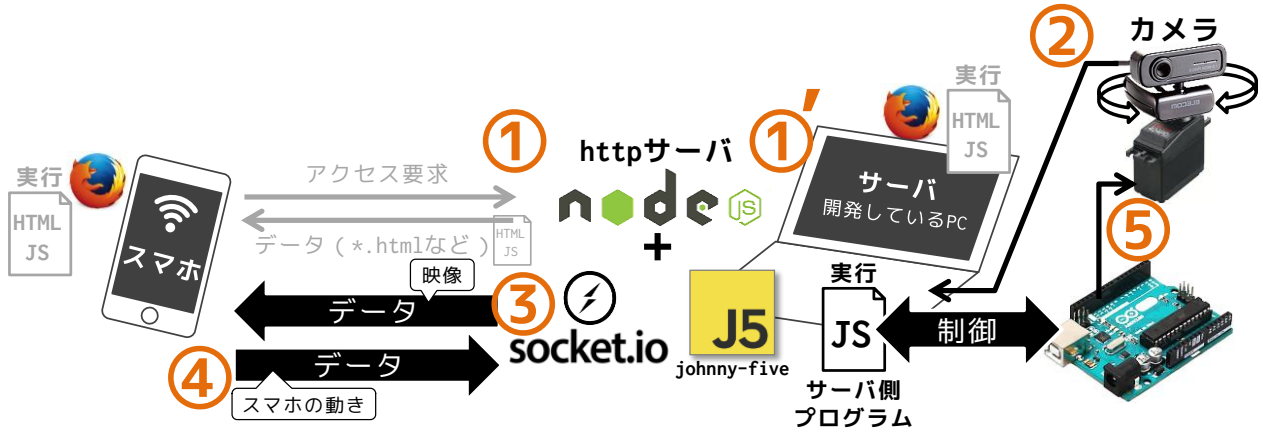
memo

✈ 9.1 スマホで遠隔監視

自宅に置いたカメラの映像をスマホで監視するシステムを作ります。Arduino につないだサーボにカメラを搭載して、スマホからカメラの向きをコントロールします。さらに、その画像を保存することもできます。

Plan 設　計

これから作るシステムの構成を下図に示します。



スマホのブラウザで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て実行します。また、①'サーバ PC 上のブラウザでも同じ index.html を実行します。①'は PC につながれたカメラで映像を撮影して、③socket.io を通じてスマホに画像を送ります。スマホは④socket.io を通じてスマホの動き（加速度やジャイロなど）のデータをサーバに送り、そのデータをもとに⑤johnny-five を通じて、Arduino に接続されたサーボを回転させます。

これまで学んできた、ブラウザ側プログラム（index.html）、カメラ映像の取り扱い、スマホのセンサの取り扱い、Node.js によるサーバ側プログラム、双方向通信（socket.io）、マイコン制御（johnny-five、Arduino）など、様々なテクニックを組み合わせて使います。

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

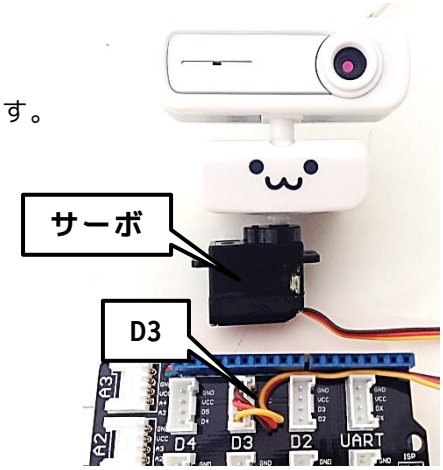
- ファイル → スケッチの例 → Firmata → StandardFirmata を選ぶ。
- Arduino と PC を USB ケーブルで接続
- Arduino IDE の設定確認
  - ツール → ボード で「Arduino/Genuino Uno」が選択されていることを確認
  - ツール → シリアルポート で「COM ** (Arduino/Genuino Uno)」のように、右側にボードの名前が表示された COM が選択されていることを確認
  - ★ ↑の「COM**」の数字をメモする（後で使います）

メモ　COM____

- プログラムを書き込む

Make システムの構築

- **カメラ**の底に丸い**サーボホーン**をテープで固定します。
- カメラの USB ケーブルをサーバ **PC に接続**します。



- カメラに付けたサーボホーンを**サーボ**の回転軸に取り付けます。
- サーボを Grove の **D3** に接続します。

Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥**の中に **9.1** というフォルダを作り、Brackets で以下のコードを入力しましょう。8.2b などの app.js からコピペして変更しながら、効率的に入力しましょう。

※ 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）￥**9.1**￥ **app.js**

```
// 9.1 スマホで遠隔監視
'use strict';                                   // 厳格モードにする

// Johnny-five の準備（8.2b 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
// 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({                    // サーボを取得
        pin: pinServo,                          // サーボのピン番号
        invert: true                            // 回転を逆転させる
    });
    arduinoReady = true;                        // Arduino 準備 OK
});


// ソケット通信（socket.io）の準備（8.2b 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


// ソケット通信による監視・制御
io.on('connection', function(socket) {      // socket 接続があって
    // サーボの制御（8.2b などからコピペしましょう）
    socket.on('servo', function(data) {     // 'servo'というイベント名の socket が来たら
        if (arduinoReady == true) {         // Arduino が準備 OK なら
            servo.to(data.angle);           // サーボを指定角度まで回す
        }
    });
                                    // 次のページに続く
```

```
    // 映像の配信
    socket.on('video', function(data) {         // 'video'というイベントの socket が来たら
        socket.broadcast.emit('video', data);   // 自分以外に映像を配信
    });
});
```

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

- **C:￥Users￥（ユーザ名）￥9.1** の中に、1.1 の index.html（テンプレート）をコピーして、以下のコードを入力します。長めですが使ったことがあるコードばかりです。頑張りましょう。

※ 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）￥9.1￥ **index.html**

```
（略）
        <title>9.1 スマホで遠隔監視</title>
    </head>
    <body>
        <!-- カメラ映像を表示する video、canvas、img 要素（6.5 を参照） -->
        <video width='144' height='192' autoplay id='video'></video>
        <canvas width='144' height='192' hidden='true' id='local'></canvas>
        <img width='144' height='192' id='remote'>          <!-- 相手の映像 -->
        <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

            // サーボ制御（スマホのセンサを使って）
            window.addEventListener('devicemotion', ctrlServo);     // スマホで加速度計測
            function ctrlServo(event) {                             // サーボの制御処理
                const accX = event.accelerationIncludingGravity.x;  // X 軸加速度を計測
                const servoAngle = 90 + Math.floor(accX) * 10;      // サーボ角度に換算 ①
                socket.emit('servo', {                  // 'servo'とういイベント名で
                    angle: servoAngle                   // angle 属性に角度を入れて送信
                });
            }

            // カメラを使う（3.6 や 6.5 を参照）
            const media = navigator.mediaDevices.getUserMedia({
                video: true,    // カメラを使うか否か
                audio: false    // マイクを使うか否か
            });
            // メディアストリームの取得に成功したら onSuccess、失敗したら onError を実行
            media.then(onSuccess, onError);
            const video = document.getElementById('video');      // video 要素を取得

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

```
            function onSuccess(stream) {                          // 成功時の onSuccess 関数
                video.src = window.URL.createObjectURL(stream);   // video 要素に映像取得
                video.addEventListener('timeupdate', sendVideo); // 更新時 sendVideo 実行
            }
            function onError(err) {                               // エラー時の onError 関数
                window.alert(err);                                // ダイアログに err を表示
            }

            // カメラの映像をキャプチャして送信（6.5 を参照）
            function sendVideo() {
                const canvas = document.getElementById('local');
                const context = canvas.getContext('2d');
                context.drawImage(video, 0, 0, 144, 192);    // 映像を canvas にキャプチャ
                socket.emit('video', {                       // 'video'というイベントで
                    pict: canvas.toDataURL()                 // pict 属性に画像を入れて送信
                });
            }

            // サーバからの映像受信（6.5 を参照）
            socket.on('video', function (data) {     // 'video'というイベントで受信したら
                showVideo(data);                     // showVideo を実行
            });
            function showVideo(data) {                           // 画像の表示処理
                const remote = document.getElementById('remote');   // img 要素を取得
                remote.src = data.pict;                          // src 属性に画像を設定
            }
        </script>
（以下略）
```

Check 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_09_素材.zip」をダウンロードし、中にある **node_modules フォルダ（johnny-five、socket.io、express ライブラリが入っている）を 9.1 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（↲）

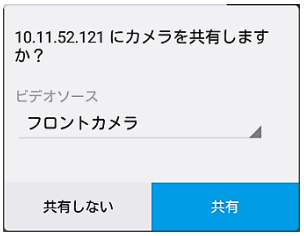
    >**ipconfig**↲　… サーバ PC の IPv4 アドレスをメモ（★）

    >**cd 9.1**↲　… C:￥Users￥（ユーザ名）￥9.1 フォルダに移動

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

- サーバ PC で監視開始
  - PC の Firefox で、メモしたア IP ドレス（★）を入力して Enter
    アドレス例： **10.11.123.456**
  - 「カメラを 10.**.**.**と共有しますか？」と出るので
    接続した（外付けの）カメラを選んで「共有」をクリック
  - カメラの映像が映っていることを確認

- スマホで遠隔監視
  - スマホの Firefox で、メモしたア IP ドレス（★）を入力して Enter
    アドレス例： **10.11.123.456**
  - 「10.**.**.**にカメラを共有しますか？」と出るので
    「フロントカメラ」を選択して「共有」をタップ
    （「共有しない」でも問題ない）
  - **PC につないだカメラの映像が右側に見えていれば OK**
  - **スマホを動かすとサーボ（カメラ）が動けば OK**
  - **右側の映像を長押しすると保存メニューが出れば OK**



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

- **動作確認が終わったら Arduino の USB ケーブルを抜く**

Learn 解　説

① `const servoAngle = 90 + Math.floor(accX) * 10;`

加速度センサのデータは-9.8～9.8[m/s^2]で得られます。`Math.floor()`は小数点以下を切り捨てるメソッドです。つまり、加速度 accX の小数点以下を切り捨てて-9～9 にしたうえで、10 を乗じることで、-90～90 の値にしています。スマホを左右に傾けることで servoAngle の値は 0～180 度の間で変化します。

なお、小数の扱いについては他にも以下のようなメソッドがあります。

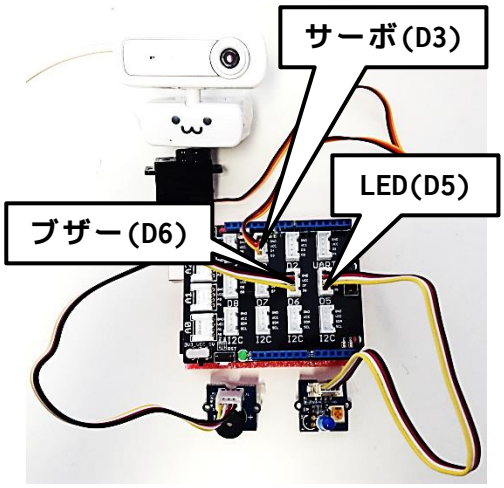
メソッド	意味	使用例	
Math.ceil()	切り上げ	Math.ceil(1.4) → 2	Math.ceil(1.5) → 2
Math.round()	四捨五入	Math.round(1.4) → 1	Math.round(1.5) → 2
Math.floor()	切り捨て	Math.floor(1.4) → 1	Math.floor(1.5) → 1

✈ 9.2 遠隔監視と警告

9.1 の遠隔監視システムに、スマホから警告を発する機能を付け加えます。自宅に忍び込んだ悪者を光の点滅とブザーで威嚇します。

Make システムの構築

- 9.1 のシステムに加えて (サーボとカメラはそのままで)
- **LED** を **D5** に接続します。
- **ブザー** を **D6** に接続します



Input サーバ側プログラム (JavaScript) を入力

- **C:￥Users￥ (ユーザ名) ￥9.1** を **フォルダごと複製** し、フォルダ名を **9.2** にします。
- app.js に以下の **網かけのコードを追加** 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) ￥**9.2**￥ **app.js**

```
// 9.2 遠隔監視と警告
'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
// サーボ、LED 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
const pinLED = 5;                               // LED を接続したピン番号
let led;                                        // LED 用オブジェクトの用意
const pinBuzzer = 6;                            // ブザーを接続したピン番号
let buzzer;                                     // ブザー用オブジェクトの用意
```

// 次のページに続く

```
arduino.on('ready', function() {            // Arduino の準備ができたら
    servo = new five.Servo({                // サーボを取得
        pin: pinServo,                      // サーボのピン番号
        invert: true                        // 回転を逆転させる
    });
    led = new five.Led(pinLED);             // LED を取得
    buzzer = new five.Piezo(pinBuzzer);     // ブザー ( ピエゾ素子 ) を取得                ①
    arduinoReady = true;                    // Arduino 準備 OK
});

// ソケット通信 ( socket.io ) の準備
  ( この部分は全く書き換えないので略記します )

// ソケット通信による監視・制御
io.on('connection', function(socket) {      // socket 接続があって
    // サーボの制御
    socket.on('servo', function(data) {     // 'servo'というイベントの socket が来たら
        if (arduinoReady == true) {         // Arduino が準備 OK なら
            servo.to(data.angle);           // サーボを指定角度まで回す
        }
    });
    // 映像の配信
    socket.on('video', function(data) {     // 'video'というイベントの socket が来たら
        socket.broadcast.emit('video', data);   // 自分以外に映像を配信
    });
    // 警告の処理
    socket.on('alarm', function(data) {     // 'alarm'というイベントの socket が来たら
        if (arduinoReady == true) {         // Arduino が準備 OK なら
            if (data.light == true) {       // データの light 属性が true なら            ②
                led.blink(100);             // LED を 100ms 間隔で点滅させる
            } else {                        // そうでなければ ( blink 属性が false なら )
                led.stop();                 // LED の点滅を止めて
                led.off();                  // LED を消す
            }
            if (data.sound == true) {       // データの sound 属性が true なら            ②
                buzzer.play({               // ブザー音を鳴らす                           ③
                    song: 'B C B C B C B C B C B C B C ',   // 曲 ( 音程 )             ④
                    beats: 1 / 4,                          // 一拍の長さ
                    tempo: 100                             // テンポ[bpm]
                });
            } else {
                buzzer.off();               // ブザー音を消す                             ⑤
            }
        }
    });
});
```

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

- **C:￥Users￥（ユーザ名）￥9.2** の中の index.html に**網掛けのコードを追加**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）￥9.2￥ index.html

```
（略）
        <title>9.2 遠隔監視と警告</title>
    </head>
    <body>
        <!-- カメラ映像を表示する video、canvas、img 要素（6.5 を参照） -->
        <video width='144' height='192' autoplay id='video'></video>
        <canvas width='144' height='192' hidden='true' id='local'></canvas>
        <img width='144' height='192' id='remote'>          <!-- 相手の映像 -->
        <div id='msg'></div>                                <!-- 警告表示エリア -->

        <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

            // サーボ制御
            window.addEventListener('devicemotion', ctrlServo); // スマホで加速度を計測
            function ctrlServo(event) {                          // サーボの制御処理
                const accX = event.accelerationIncludingGravity.x;  // X 軸加速度を計測
                const servoAngle = 90 + Math.floor(accX) * 10;      // サーボ角度に換算
                socket.emit('servo', {                    // 'servo'とういイベント名で
                    angle: servoAngle                     // angle 属性に角度を入れて送信
                });
            }

            // カメラを使う（6.5 を参照）
            const media = navigator.mediaDevices.getUserMedia({
                video: true,    // カメラを使うか否か
                audio: false    // マイクを使うか否か
            });
            // メディアストリームの取得に成功したら onSuccess、失敗したら onError を実行
            media.then(onSuccess, onError);
            const video = document.getElementById('video');      // video 要素を取得
            function onSuccess(stream) {                         // 成功時の onSuccess 関数
                video.src = window.URL.createObjectURL(stream);  // video 要素に映像取得
                video.addEventListener('timeupdate', sendVideo); // 更新時 sendVideo 実行
            }
            function onError(err) {                              // エラー時の onError 関数
                window.alert(err);                               // ダイアログに err を表示
            }

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

```
            // カメラの映像をキャプチャして送信
            function sendVideo() {
                const canvas = document.getElementById('local');
                const context = canvas.getContext('2d');
                context.drawImage(video, 0, 0, 144, 192);   // 映像を canvas にキャプチャ
                socket.emit('video', {                      // 'video'というイベントで
                    pict: canvas.toDataURL()                // pict 属性に画像を入れて送信
                });
            }

            // サーバからの映像受信
            socket.on('video', function (data) {    // 'video'というイベントで受信したら
                showVideo(data);                    // showVideo を実行
            });
            function showVideo(data) {                      // 画像の表示処理
                const remote = document.getElementById('remote');   // img 要素を取得
                remote.src = data.pict;                     // src 属性に画像を設定
            }

            // 警告を鳴らす処理
            const msg = document.getElementById('msg');
            remote.addEventListener('click', startAlarm);   // img 要素に click イベント
            function startAlarm() {                 // 警報を鳴らす処理
                socket.emit('alarm', {              // 'alarm'イベントを socket 送信      ⑥
                    light: true,                    // LED を点滅させる
                    sound: true
                });
                msg.innerHTML = '警告！';            // 警告文字列の表示
                window.setTimeout(stopAlarm, 2000); // 一定時間後に警告を止める          ⑦
            }
            function stopAlarm() {                  // 警報を止める処理
                socket.emit('alarm', {              // 'alarm'イベントを socket 送信      ⑥
                    light: false,                   // LED を消灯させる
                    sound: false
                });
                msg.innerHTML = '';                 // 警告文字列の消去
            }
        </script>
（以下略）
```

Check 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_09_素材.zip」をダウンロードし、中にある **node_modules フォルダ（johnny-five、socket.io、express ライブラリが入っている）を 9.2 フォルダ内にコピー**（**9.1 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）￥9.1 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 9.2**↲　　… C:￥Users￥（ユーザ名）￥9.2 フォルダに移動

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

- サーバ PC で監視開始
  - PC の Firefox で、メモしたア IP ドレス（★）を入力して Enter

    アドレス例：**10.11.123.456**
  - 「カメラを 10.**.**.**と共有しますか？」と出るので

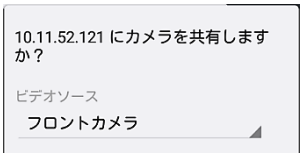
    接続した（外付けの）カメラを選んで「共有」をクリック
  - カメラの映像が映っていることを確認

- スマホで遠隔監視と**警告**
  - スマホの Firefox で、メモしたア IP ドレス（★）を入力して Enter

    アドレス例：**10.11.123.456**
  - 「10.**.**.**にカメラを共有しますか？」と出るので

    「フロントカメラ」を選択して「共有」をタップ

    （「共有しない」でも問題ない）
  - **PC につないだカメラの映像をタップすると…**
    - **ブザーが「シドシド…」と鳴れば OK**
    - **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「**Ctrl + C**」を **2 回**入力

- **動作確認が終わったら Arduino の USB ケーブルを抜く**

Learn 解　説

サーバ側プログラム (app.js)

① `buzzer = new five.Piezo(pinBuzzer);`

johnny-five でブザー (ピエゾ素子・圧電素子) を扱うための Piezo オブジェクトです。

詳しくは → http://johnny-five.io/api/piezo/

② `if (data.light == true) { ... }` および `if (data.sound == true) { ... }`

クライアント側プログラム (index.html) の⑥で、light 属性および sound 属性に true が入って送られてきたら、警告として、LED の点滅とブザーを鳴らしています。

③ `buzzer.play({ 鳴らす音の設定 })`

ブザーを鳴らす Piezo オブジェクトの play()メソッドです。{ }内には、以下④のようにブザーで鳴らす音の設定をします。

④ `{song: 'B C B C B C B C B C B C B C ', beats: 1 / 4, tempo: 100 }`

③でブザーを鳴らす音の設定です。

song 属性には曲の音程を入れます。ドレミファソラシがそれぞれ CDEFGAB に対応します。空白を入れると 1 拍休みます。ここでは、「シ（休）ド（休）シ（休）ド（休)…」と鳴らしています。

beats 属性には 1 拍の長さを指定します。ここでは 1/4、つまり 16 分音符を指定しています。

tempo 属性には曲のテンポ[bpm]を指定します。ここでは毎分 100 拍を指定しています。

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

⑤ `buzzer.off();`

ブザーを止める Piezo オブジェクトの stop()メソッドです。

クライアント側プログラム (index.html)

⑥ `socket.emit('alarm', { light: true, sound: true });`

右側の映像がタップされたら alarm というイベントで{ }のデータを送っています。light および sound 属性のそれぞれに true を指定すると、サーバ側プログラム (app.js) の②で LED を点滅させブザーを鳴らします。false を指定すると②で LED を消灯させブザーを止めます。

⑦ `window.setTimeout(stopAlarm, 2000);`

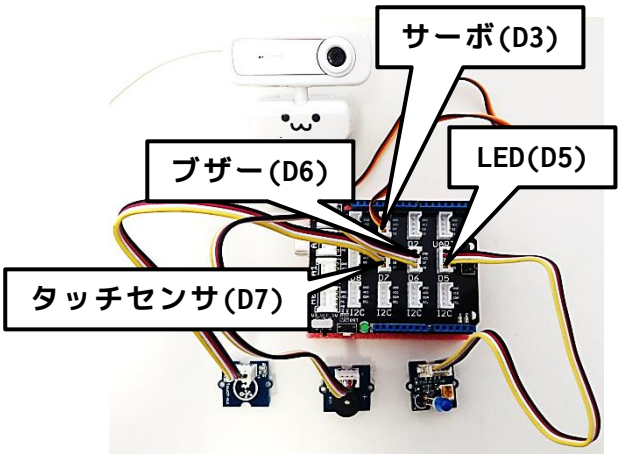
指定時間後に 1 回だけ処理を行う setTimeout()です。ここでは、2000ms (2 秒) 後に stopAlarm を実行しています。つまり右側の映像がタップされたら⑥で LED を点滅させブザーを鳴らし、2 秒後に stopAlarm を実行して LED を消灯させブザーを止めています。

✈ 9.3 遠隔監視と警告と危険お知らせ

9.2 のシステムに、大切なものに触れられた時にスマホから音を出したり振動させたりする危険お知らせ機能を付け加えます。

Make システムの構築

- 9.2 のシステムに加えて
  （サーボ、カメラ、LED、ブザーはそのままで）
- **タッチセンサ**を **D7** に接続します。



Input サーバ側プログラム（JavaScript）を入力

- **C:￥Users￥（ユーザ名）￥9.2** を**フォルダごと複製**し、フォルダ名を **9.3** にします。
- app.js に以下の**網かけのコードを追加**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）￥**9.3**￥ **app.js**

```
// 9.3 遠隔監視と警告と危険お知らせ
'use strict';                          // 厳格モードにする

// Johnny-five の準備
  （この部分は全く書き換えないので略記します）

// サーボ、LED、タッチセンサの準備
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
const pinLED = 5;                      // LED を接続したピン番号
let led;                               // LED 用オブジェクトの用意
const pinBuzzer = 6;                   // ブザーを接続したピン番号
let buzzer;                            // ブザー用オブジェクトの用意
const pinTouch = 7;                    // タッチセンサを接続したピン番号
let touch;                             // タッチセンサ用オブジェクトの用意

                        // 次のページに続きます
```

```
arduino.on('ready', function() {            // Arduino の準備ができたら
    servo = new five.Servo({                // サーボを取得
        pin: pinServo,                      // サーボのピン番号
        invert: true                        // 回転を逆転させる
    });
    led = new five.Led(pinLED);             // LED を取得
    buzzer = new five.Piezo(pinBuzzer);     // ブザー ( ピエゾ素子 ) を取得
    touch = new five.Button(pinTouch);      // タッチセンサを取得
    arduinoReady = true;                    // Arduino 準備 OK
});

// ソケット通信 ( socket.io ) の準備 ( 6.1 参照 )
  ( この部分は全く書き換えないので略記します )

// ソケット通信による監視・制御
io.on('connection', function(socket) {      // socket 接続があって
    // サーボの制御
    socket.on('servo', function(data) {     // 'servo'というイベントの socket が来たら
      ( この部分は全く書き換えないので略記します )

    // 映像の配信
    socket.on('video', function(data) {     // 'video'というイベントの socket が来たら
      ( この部分は全く書き換えないので略記します )

    // 警告の処理
    socket.on('alarm', function(data) {     // 'alarm'というイベントの socket が来たら
      ( この部分は全く書き換えないので略記します )

    // タッチ注意喚起の処理
    if (arduinoReady == true) {                 // Arduino が準備 OK なら
        touch.on('press', function() {          // ボタンが押されたら
            socket.broadcast.emit('touch', {    // 'touch'というイベント名で          ②
                touched: 'press'                // touched 属性に'press'を入れて送信
            });
        });
        touch.on('hold', function() {           // ボタンが長押しされたら ( 500ms 以上 ) ①
            socket.broadcast.emit('touch', {    // 'touch'というイベント名で          ②
                touched: 'hold'                 // touched 属性に'hold'を入れて送信
            });
        });
    }
});
```

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

- **C:￥Users￥（ユーザ名）￥9.3** の中の index.html に**網掛けのコードを追加**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）￥9.3￥ index.html

```
（略）
        <title>9.3 遠隔監視と警告と危険お知らせ</title>
    </head>
    <body>
      （この部分は全く書き換えないので略記します）

        <script>
            // 双方向通信用のサーバに接続
            const socket = io();
            // サーボ制御
              （この部分は全く書き換えないので略記します）
            // カメラを使う（3.6 や 6.5 を参照）
              （この部分は全く書き換えないので略記します）
            // カメラの映像をキャプチャして送信（6.5 を参照）
              （この部分は全く書き換えないので略記します）
            // サーバからの映像受信（6.5 を参照）
              （この部分は全く書き換えないので略記します）
            // 警告を鳴らす処理
              （この部分は全く書き換えないので略記します）

            // 注意喚起の処理
            socket.on('touch', function(data) {   // 'touch'というイベントを受信したら　③
                attention(data);                  // attention を実行
            });
            function attention(data) {            // 注意喚起する処理
                const sound = new Audio();              // Audio オブジェクト
                if (data.touched == 'press') {          // タッチセンサに触れられたら
                    sound.src = 'aa.mp3';               // 音声「あ！」を読み込み
                    sound.play();                       // 鳴らす
                    navigator.vibrate(200);             // バイブレータを短めに振動
                } else if (data.touched == 'hold') {    // タッチセンサが長押しされたら
                    sound.src = 'kyaa.mp3';             // 音声「きゃー！」を読み込み
                    sound.play();                       // 鳴らす
                    navigator.vibrate(2000);            // バイブレータを長めに振動
                }
            }
        </script>
（以下略）
```

Check 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_09_素材.zip」をダウンロードし、中にある **node_modules フォルダ（johnny-five、socket.io、express ライブラリが入っている）を 9.3 フォルダ内にコピー（9.2 を複製して作業している場合は不要）**
  - 「PHC_09_素材.zip」の中にある**'aa.mp3'と'kyaa.mp3'を 9.3 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）￥9.2 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（↲）

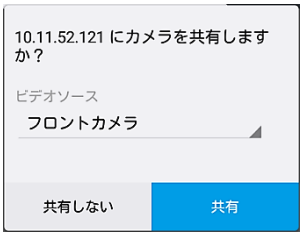
    >**cd..**↲　　… ひとつ上のフォルダに移動（C:￥Users￥（ユーザ名）に移動）

    >**cd 9.3**↲　　… C:￥Users￥（ユーザ名）￥9.3 フォルダに移動

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

- サーバ PC で監視開始
  - PC の Firefox で、メモしたア IP ドレス（★）を入力して Enter
  - 「カメラを 10.**.**.**と共有しますか？」と出るので
    接続した（外付けの）カメラを選んで「共有」をクリック
  - カメラの映像が映っていることを確認

- スマホで遠隔監視と**警告**
  - スマホの Firefox で、メモしたア IP ドレス（★）を入力して Enter
  - 「10.**.**.**にカメラを共有しますか？」と出るので
    「フロントカメラ」を選択して「共有」をタップ
    （「共有しない」でも問題ない）
  - **Arduino につないだタッチセンサに触れると…**
    - **少し触ると、「あ！」という音声と短く振動**
    - **長く触ると、「きゃー！」という音声と長く振動**



警告！

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

- **動作確認が終わったら Arduino の USB ケーブルを抜く**

Learn 解　説

サーバ側プログラム（app.js）

① **touch.on('hold', function() { 処理 });**

8.3b ではタッチセンサのイベントとして'press'（触れられた）と'release'（離された）を使いました。Button オブジェクトにはもう一つ'hold'（長押し）イベントがあります。

② **socket.broadcast.emit('touch', { データ });**

タッチセンサに触れられたら'touch'というイベント名のソケットを送っています。この際送るデータには touched 属性を設けて、'press'（触れられた）かまたは'hold'（長押しされた）かを設定しています。

クライアント側プログラム（index.html）

③ **socket.on('touch', function(data) { attention(data); });**

上記のサーバ側プログラム（app.js）②によって'touch'というイベント名のソケットが送られてきたら、関数 attention を実行しています。

関数 attention では、ソケットで送られてきたデータの touched 属性が'press'（触れられた）であれば'aa.mp3'を再生し、スマホを短く振動させています。また、touch 属性が'hold'（長押し）であれば'kyaa.mp3'を再生し、スマホを長めに振動させています。

Copy 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（3 個）
  - 9.1　9.2　9.3

Work 本日の課題

① 9.1、9.2 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 2 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 **09**」から提出

【提出締切】　12 月 8 日（木）　15:00（遅れてしまった場合は担当教員に相談のこと）