

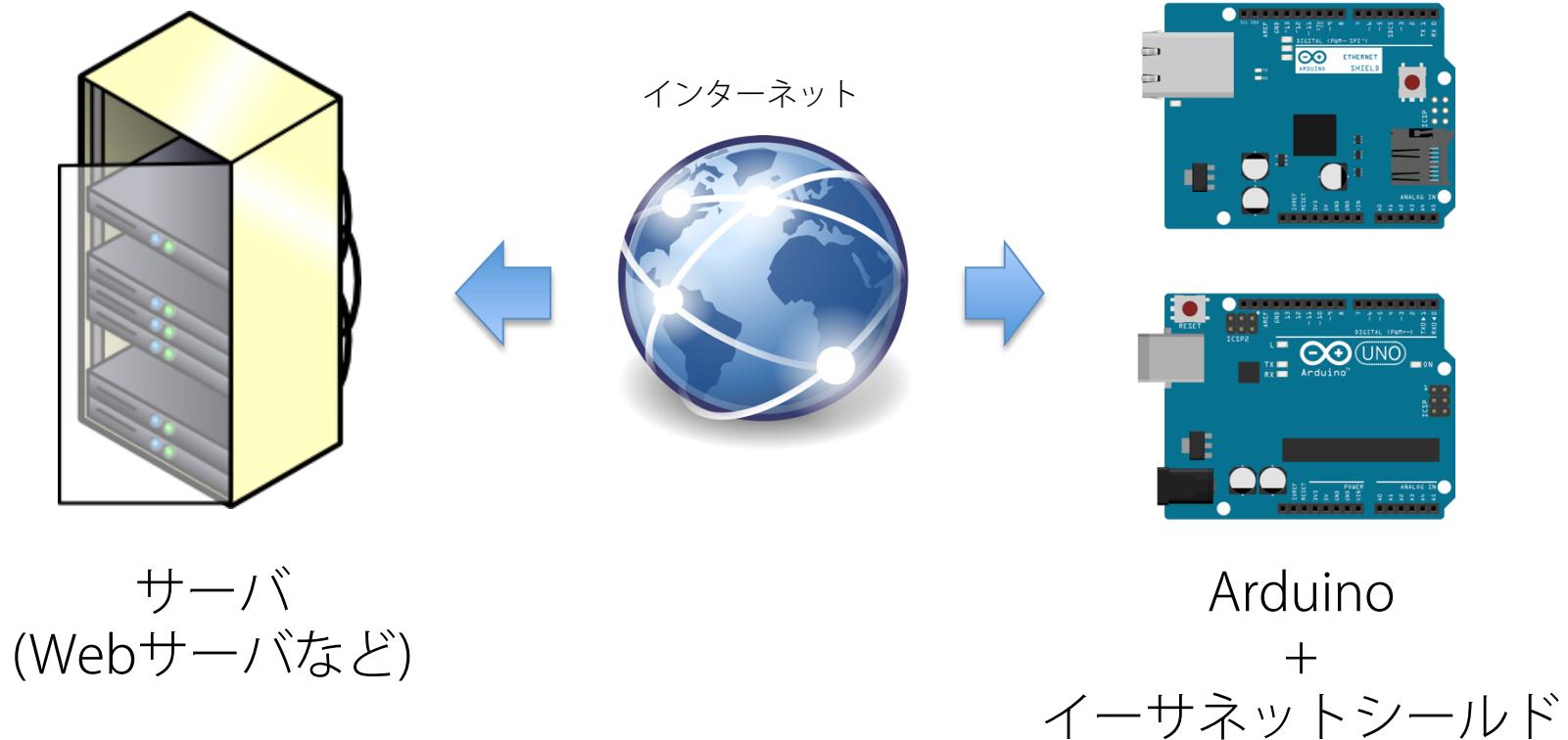
# Arduino入門勉強会 #4

## 【拡張シールドとHTTP通信】

平成27年8月4日  
ソフトピアジャパン ドリーム・コア1F ネクストコア

# Webサーバとの通信

Arduinoにイーサネットシールドを追加することにより、TCP/IP通信によるWebサーバなどとのデータ通信が可能になります。



参考動画 : <https://vimeo.com/53727530>

# 機能拡張ボード：シールド

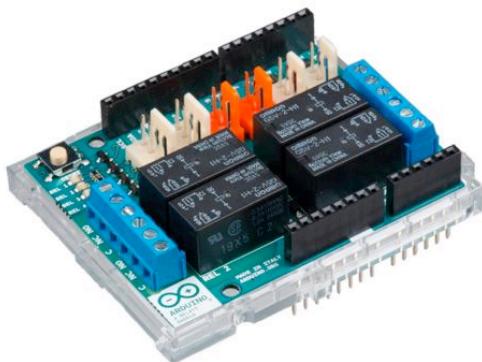
Arduinoにはシールドと呼ばれる拡張基板を追加することにより機能を追加することができます。また、オリジナルのシールドを造ることも可能で、シールド製作用の部品が販売されています。



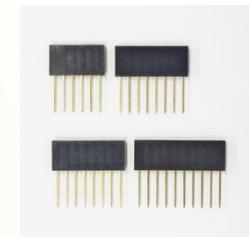
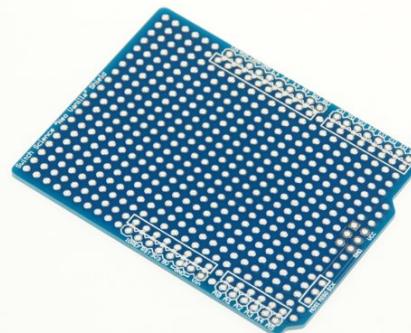
モーターシールド



キャラクタ液晶シールド



リレーシールド



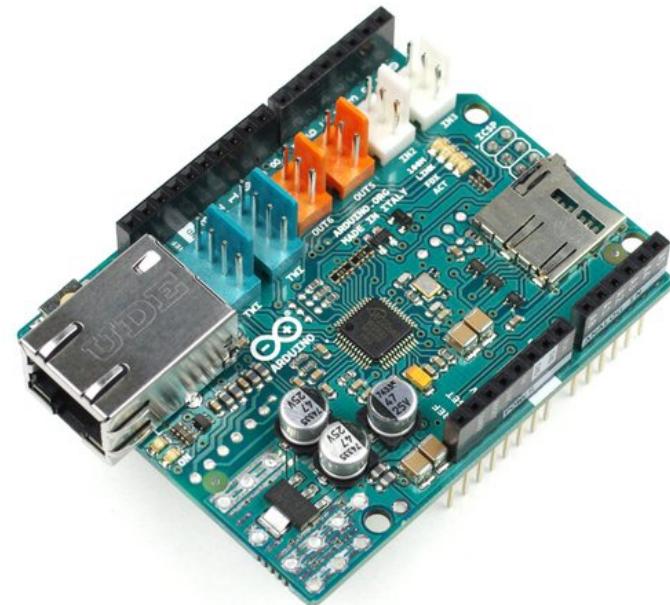
シールド製作用部品

# イーサネットシールド

ArduinoにEthernet接続機能を追加するシールドです。



Arduino Ethernet Shield R3



Arduino Ethernet Shield 2

# 互換品

ハードウェア仕様がオープンソース化されているArduinoでは互換製品が割安で提供されている場合があります。

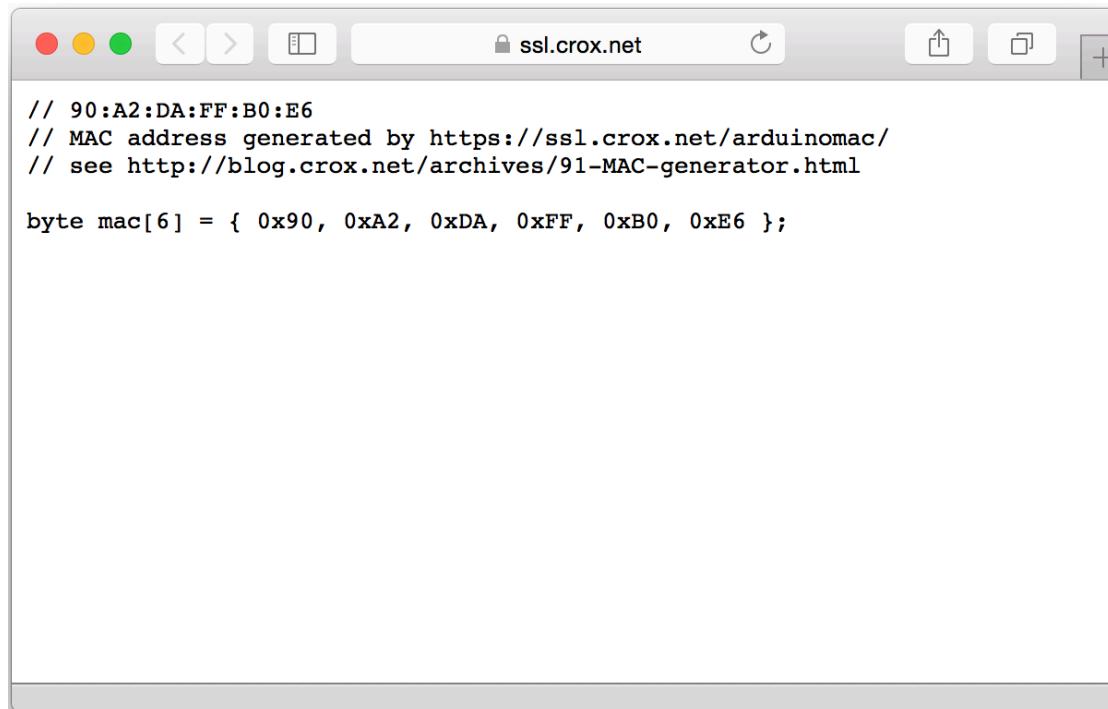


Arduino Ethernet Shield 互換基板

# MACアドレス

今回購入の互換品にはMACアドレスが添付されていません。ローカルネットワーク上でユニークなMACアドレスをソースコード上で設定すれば良いため、利用可能なMACアドレスを生成して利用します。

MACアドレス生成サイト「<https://ssl.crox.net/arduinomac/>」



A screenshot of a web browser window titled "ssl.crox.net". The page displays a block of Arduino C-like code. The code includes a header comment and a single-line assignment statement:

```
// 90:A2:DA:FF:B0:E6
// MAC address generated by https://ssl.crox.net/arduinomac/
// see http://blog.crox.net/archives/91-MAC-generator.html

byte mac[6] = { 0x90, 0xA2, 0xDA, 0xFF, 0xB0, 0xE6 };
```

# イーサネットシールド2の場合

イーサネットシールド2はIDEのver1.7.\*系列でしかサポートされていません。イーサネットシールド2を使用する場合はver1.7.\*系列のインストールもしくはver1.7.\*系列のライブラリをver1.6.\*系列のライブラリに移植する必要があります。

IDEのver1.7.\*系列のダウンロード

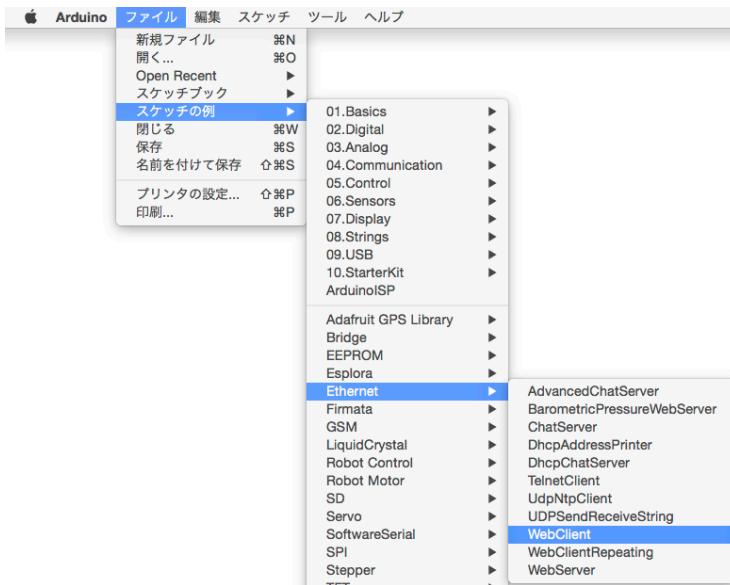
<http://www.arduino.org>

参考：Arduino戦争：グループ分裂、そして新製品の登場

<http://makezine.jp/blog/2015/03/arduino-vs-arduino.html>

# サンプル「Web Client」

Googleにhttpでアクセスした結果を表示させるサンプルコードです。ソースコード上のMACアドレスの部分を自身の環境にあわせて書き換えます。DHCPを使用するので、固定IPの設定をコメントアウトします。



メニューから  
「ファイル」→「スケッチの例」  
→「Ethernet」→「WebClient」  
と選択します。

```
 WebClient | Arduino 1.6.5

 WebClient

#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on the shield
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// if you don't want to use DNS (and reduce your sketch size)
// use the numeric IP instead of the name for the server:
//IPAddress server(74,125,232,128); // numeric IP for Google (no DNS)
char server[] = "www.google.com"; // name address for Google (using DNS)

// Set the static IP address to use if the DHCP fails to assign
IPAddress ip(192, 168, 0, 177); //コメントアウトします。

// Initialize the Ethernet client library
// with the IP address and port of the server
// that you want to connect to (port 80 is default for HTTP):
EthernetClient client;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    // no point in carrying on, so do nothing forevermore:
    //try to configure using IP address instead of DHCP:
    Ethernet.begin(mac, ip); //コメントアウトします。
  }
  // give the Ethernet shield a second to initialize:
  delay(1000);
}

// ... (rest of the code)
```

MACアドレスを  
書き換えます。

コメントアウト  
します。

コメントアウト  
します。

# 「Web client」コード

ヘッダの宣言

```
#include <SPI.h>           イーサネットシールドはSPIでArduinoと通信をするのでSPI用ヘッダファイル読み込む  
#include <Ethernet.h>       イーサネットを使用するためのヘッダファイル読み込み
```

// Enter a MAC address for your controller below.

// Newer Ethernet shields have a MAC address printed on a sticker on the shield

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };   MACアドレスの指定
```

// if you don't want to use DNS (and reduce your sketch size)

// use the numeric IP instead of the name for the server:

```
//IPAddress server(74,125,232,128); // numeric IP for Google (no DNS)
```

```
char server[] = "www.google.com"; // name address for Google (using DNS)  Googleのアドレスを指定  
DNSを使用しない場合は直接指定する(一行前のコメント化された箇所を使用)
```

// Set the static IP address to use if the DHCP fails to assign

```
//IPAddress ip(192, 168, 0, 177);  IPを直接指定する場合はここに記入  
(今回はコメントアウト)
```

// Initialize the Ethernet client library

// with the IP address and port of the server

// that you want to connect to (port 80 is default for HTTP):

```
EthernetClient client;  接続クライアントの名前宣言
```

void setup()

// Open serial communications and wait for port to open:

```
Serial.begin(9600); シリアルモニタ用にシリアルポートを開く
```

```
while (!Serial) {
```

// wait for serial port to connect. Needed for Leonardo only

```
}
```

// start the Ethernet connection:

```
if (Ethernet.begin(mac) == 0) {  イーサネット通信の開始  
(DHCPによるIPアドレス自動割り当ての使用)
```

```
Serial.println("Failed to configure Ethernet using DHCP");
```

// no point in carrying on, so do nothing forevermore:

// try to configure using IP address instead of DHCP:

```
//Ethernet.begin(mac, ip);  DHCPが使えない場合は直接指定でイーサネット通信の開始  
(今回はコメントアウト)
```

初期化

# 「Web client」コード

```
// give the Ethernet shield a second to initialize:  
delay(1000); // 1000ミリ秒待機  
Serial.println("connecting..."); // シリアルモニタに接続中…と表示  
  
// if you get a connection, report back via serial:  
if (client.connect(server, 80)) { // サーバ(ここではGoogle)にポート80で接続  
    Serial.println("connected"); // 接続成功の表示  
    // Make a HTTP request:  
    client.println("GET /search?q=arduino HTTP/1.1"); // GETリクエストで「arduino」の検索結果を要求  
    client.println("Host: www.google.com"); // バーチャルホスト使用にサイトの場合の決まり文句  
    client.println("Connection: close"); // バーチャルホスト使用にサイトの場合の決まり文句  
    client.println(); // 接続終了  
}  
else { // 改行2回で終了(決まり文句)  
    Serial.println("connection failed");  
}  
}  
  
void loop()  
{  
    // if there are incoming bytes available  
    // from the server, read them and print them:  
    if (client.available()) { // 受信データのある場合  
        char c = client.read(); // 1文字読み込み  
        Serial.print(c);  
    }  
    // 読み込んだ文字をシリアルモニタに表示  
}
```

初期化

接続成功

接続失敗

受信成功の場合

Arduino勉強会 #4

# 「Web client」コード

メイン処理  
の記述

```
// if the server's disconnected, stop the client:  
if (!client.connected()) {  
    Serial.println();  
    Serial.println("disconnecting.");  
    client.stop();  
}  
  
// do nothing forevermore:  
while (true);
```

接続されていない場合

シリアルモニタに「非接続を表示」

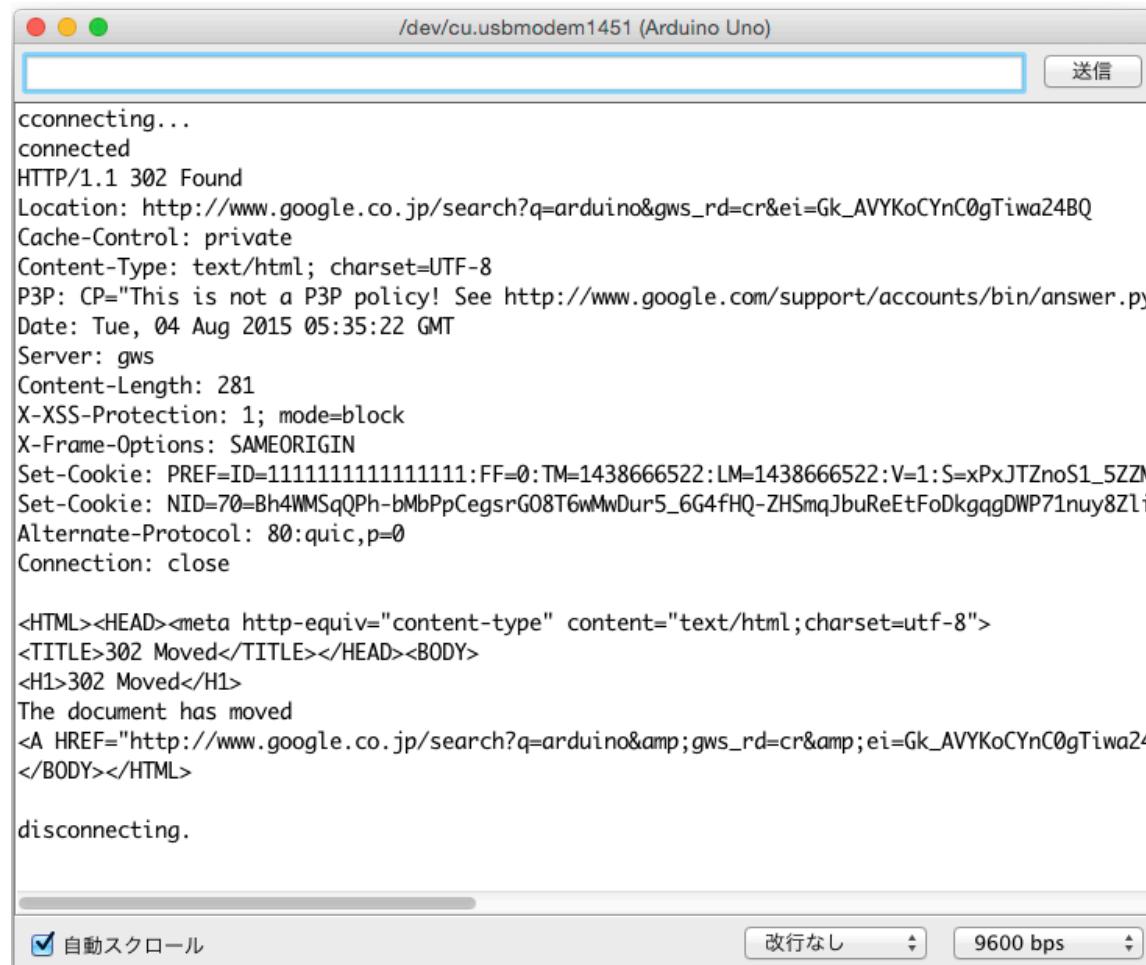
クライアント停止

ずっと何もしない

受信内容の表示が終了した  
場合及び接続失敗の場合

# 「Web client」 結果

Googleからの返信結果が表示されます。



The screenshot shows a terminal window titled '/dev/cu.usbmodem1451 (Arduino Uno)'. The window displays the following text:

```
cconnecting...
connected
HTTP/1.1 302 Found
Location: http://www.google.co.jp/search?q=arduino&gws_rd=cr&ei=Gk_AVYKoCYnC0gTiwa24BQ
Cache-Control: private
Content-Type: text/html; charset=UTF-8
P3P: CP="This is not a P3P policy! See http://www.google.com/support/accounts/bin/answer.py
Date: Tue, 04 Aug 2015 05:35:22 GMT
Server: gws
Content-Length: 281
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: PREF=ID=1111111111111111:FF=0:TM=1438666522:LM=1438666522:V=1:S=xPxJTZnoS1_5ZZM
Set-Cookie: NID=70=Bh4WMSqQPh-bMbPpCegsrG08T6wMwDur5_6G4fHQ-ZHSmqJbuReEtFoDkgqgDWP71nuy8Zli
Alternate-Protocol: 80:quic,p=0
Connection: close

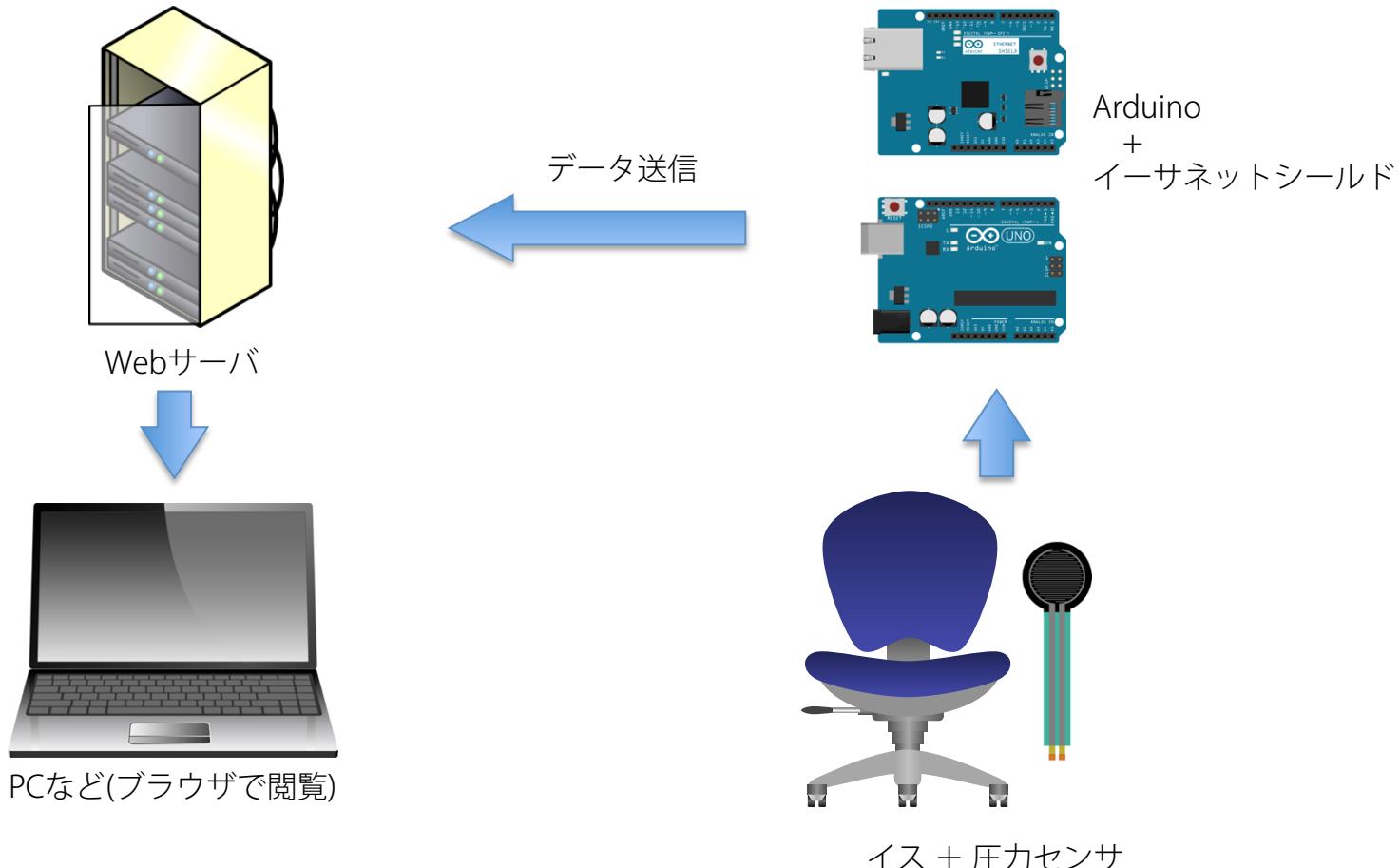
<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.co.jp/search?q=arduino&gws_rd=cr&ei=Gk_AVYKoCYnC0gTiwa24
</BODY></HTML>

disconnecting.
```

At the bottom of the window, there are three buttons: '自動スクロール' (Auto Scroll) with a checked checkbox, '改行なし' (No Line Break), and '9600 bps'.

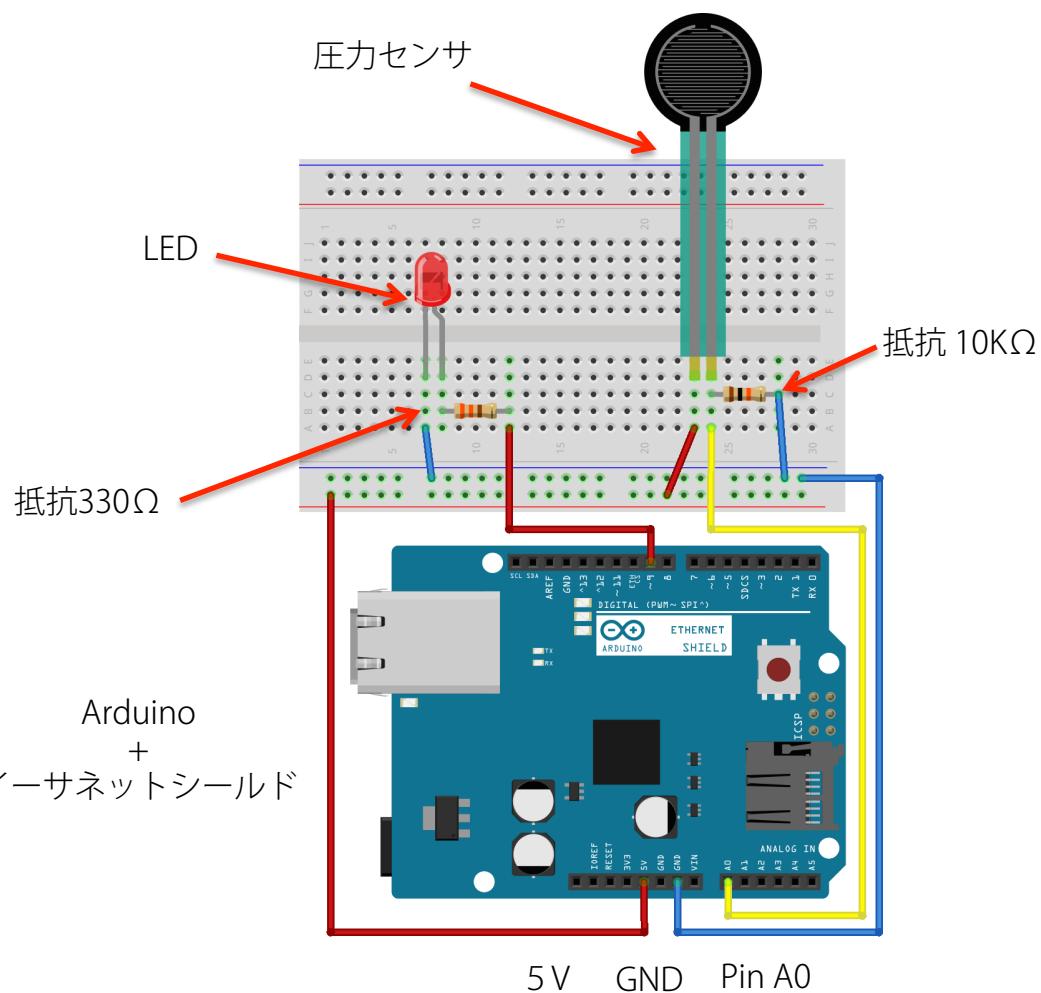
# 座席表サイト

人の着席/離席状態をセンシングして  
表示するWebサイトを構築します。



# 座席表サイト：回路

圧力センサを使って人の着席判断を行います。イーサネットシールドを取り付けたArduinoに図のような回路を作成します。



LED

※脚の長い方が  
+です。

カソード アノード  
+ -

圧力センサ

※極性はありません

抵抗10KΩ



茶黒橙金

抵抗330Ω



橙橙茶金

# 座席表サイト：コード(Arduino)

ヘッダの宣言

```
#include <SPI.h>           イーサネットシールドはSPIでArduinoと通信をするのでSPI用ヘッダファイル読み込む
#include <Ethernet.h>        イーサネットを使用するためのヘッダファイル読み込み
```

```
byte mac[] = { 0x00, 0x50, 0xC2, 0x97, 0x20, 0xAF }; // ここは MAC アドレス
//IPAddress ip(192,168,1, 92); // ここは IP アドレス
byte server[] = { 153,121,53,226 }; // 座席表サイトIPアドレス
```

```
EthernetClient client;          座席表サイトのIPアドレスを指示
```

定数・変数の宣言

```
const int analogInPin = A0; // アナログ入力ピン
const int ledPin01 = 9; // LEDピン01
```

```
int sensorValue = 0; // センサ読み取り値
int outputValue = 0; // 出力値
```

各自の座席番号に変更

```
int seatID= 25; //座席ID
```

```
int threshold = 250; //圧力センサしきい値
```

センサ値が250を未満なら空席とする

```
void setup() {
```

```
Serial.begin(9600);
```

```
pinMode(ledPin01, OUTPUT);
digitalWrite(ledPin01,LOW);
```

LEDピンを出力に設定

初期化

```
if (Ethernet.begin(mac) == 0) {
  Serial.println("Failed to configure Ethernet using DHCP");
```

```
// no point in carrying on, so do nothing forevermore:
// try to conigure using IP address instead of DHCP:
//Ethernet.begin(mac, ip);
```

```
}
```

```
delay(1000);
```

イーサネット通信の開始  
(DHCPによるIPアドレス自動割り当ての使用)

DHCPが使えない場合は直接指定でイーサネット通信の開始  
(今回はコメントアウト)

# 座席表サイト：コード(Arduino)

```

void loop() {
    //センサ値読み取り
    sensorValue = analogRead(analogInPin);    ← センサ読み取り
    Serial.println("sensorValue = ");
    Serial.println(sensorValue);
    //圧力センサの値で着席/離席判断
    if(sensorValue < threshold){           ← センサの値から着席/離席判断
        outputValue = 0; //離席
    }else{
        outputValue = 1; //着席
    }

    //サイトに接続 & データ受け渡し
    Serial.println("connecting...");          ← サーバ(座席表サイト)にポート80で接続
    if (client.connect(server, 80)) {
        digitalWrite(ledPin01,LOW);
        Serial.println("connected");
        client.print("GET /arduino_ws04_01/database/update.php?seat_id=");
        client.print(seatID);
        client.print("&on=");
        client.print(outputValue);
        client.println(" HTTP/1.0");
        client.println();
        delay(100); //100ミリ秒待機
        client.stop();
    } else {
        Serial.println("connection failed");
        digitalWrite(ledPin01,HIGH);
    }
    delay(3000); //サーバ負荷低減のための待機
}

```

メイン処理の記述

接続成功

接続失敗

※ここではclientに以下の文字列を書き込んでいます。  
 "GET /arduino\_ws04\_01/database/update.php?seat\_id=1&on=1 HTTP/1.0"  
 これはWebブラウザで以下のようにアクセスするのと同じです。  
[http://153.121.53.226/arduino\\_ws04\\_01/database/update.php?seat\\_id=1&on=1](http://153.121.53.226/arduino_ws04_01/database/update.php?seat_id=1&on=1)

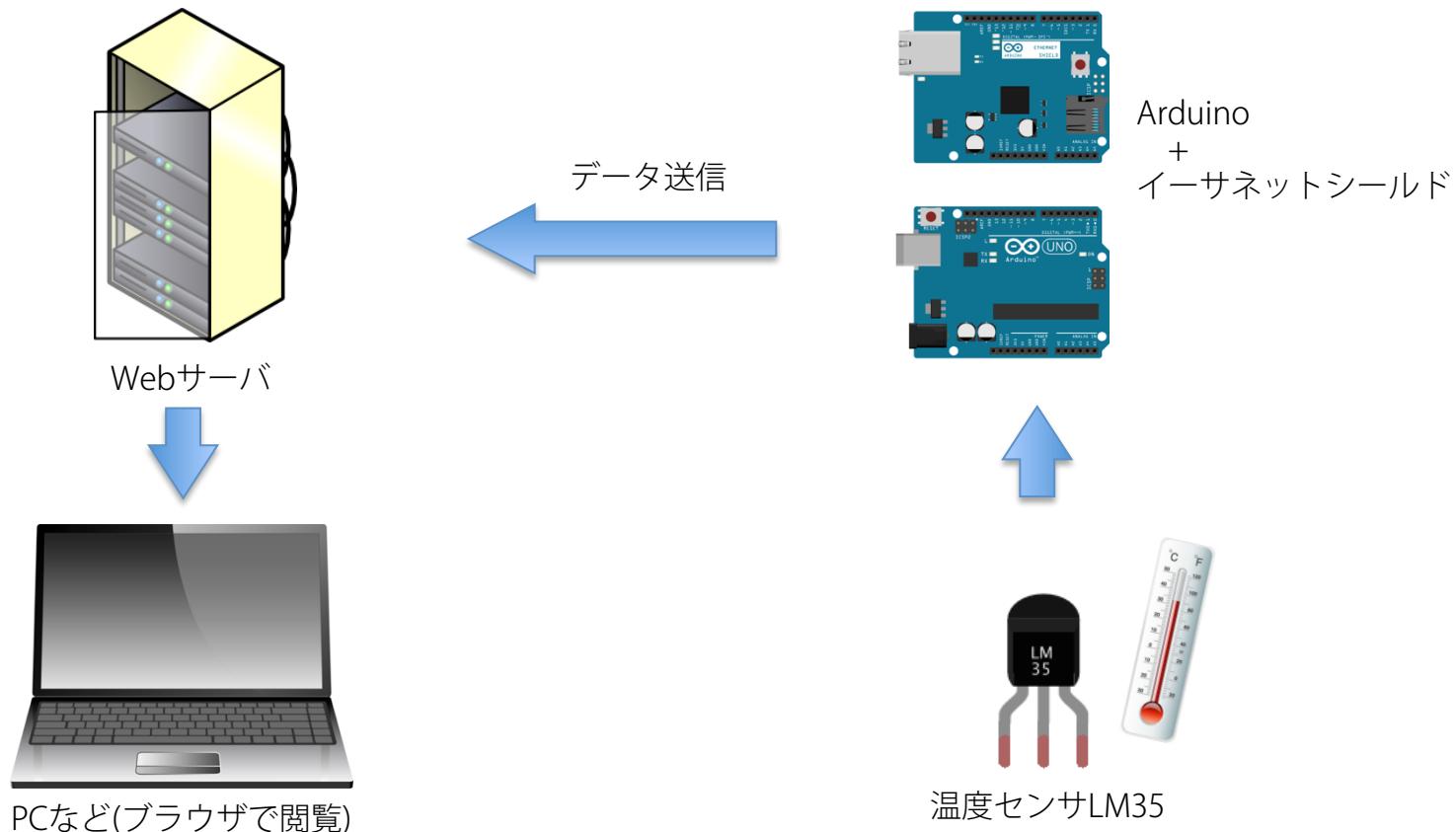
# 座席表サイト：結果

以下の、アドレスにアクセスして結果を表示します。

[http://153.121.53.226/arduino\\_ws04\\_01/](http://153.121.53.226/arduino_ws04_01/)

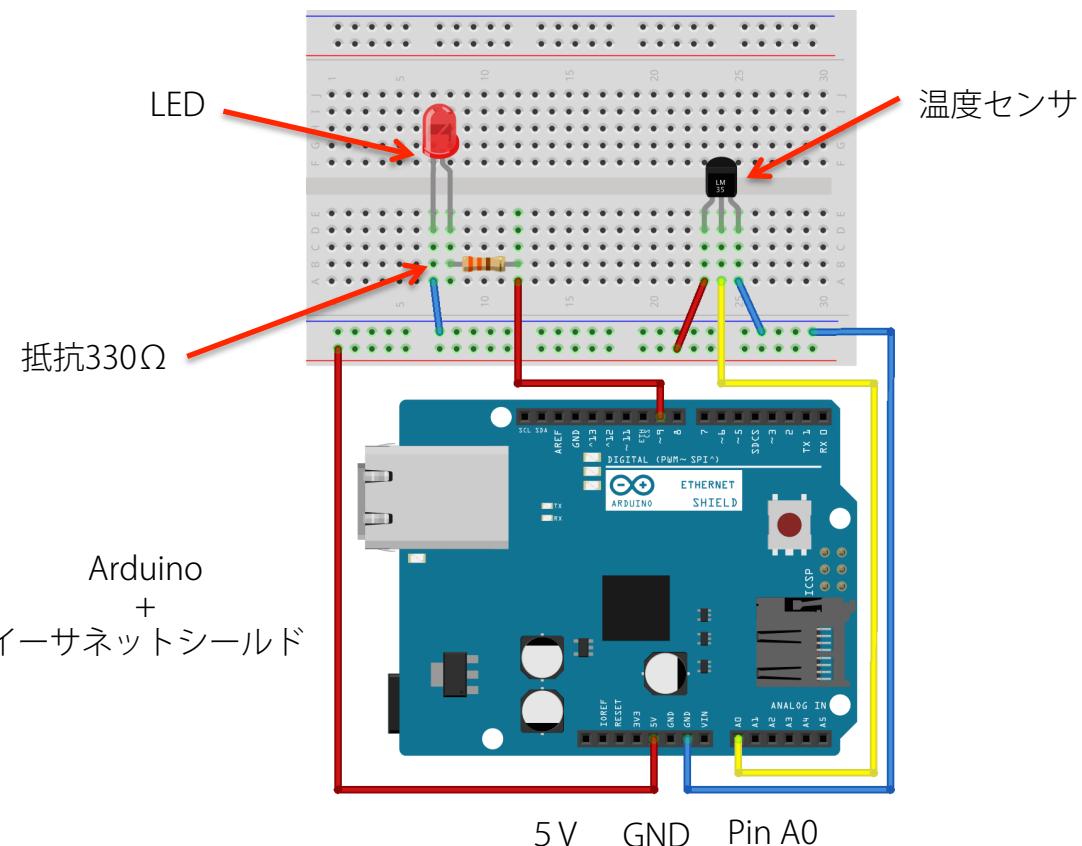
# 温度表示サイト

各座席の温度をセンシングして表示する  
Webサイトを構築します。



# 温度表示サイト：回路

温度センサLM35を用いて温度を計測します。イーサネットシールドを取り付けたArduinoに図のような回路を作成します。



- 1 : 電源(5V)
- 2 : 信号
- 3 : GND



橙橙茶金

# 温度表示サイト：コード(Arduino)

ヘッダの宣言

```
#include <SPI.h>           イーサネットシールドはSPIでArduinoと通信をするのでSPI用ヘッダファイル読み込む
#include <Ethernet.h>        イーサネットを使用するためのヘッダファイル読み込み
```

```
byte mac[] = { 0x00, 0x50, 0xC2, 0x97, 0x20, 0xAF }; // ここは MAC アドレス
//IPAddress ip(192,168,1, 92); // ここは IP アドレス
byte server[] = { 153,121,53,226 }; // 座席表サイトIPアドレス
```

```
EthernetClient client;          温度計サイトのIPアドレスを指示
```

```
const int analogInPin = A0; // アナログ入力ピン
const int ledPin01 = 9; // LEDピン01
int sensorValue = 0; // センサ読み取り値
int outputValue = 0; // 出力値
int seatID= 25; //座席ID
```

各自の座席番号に変更

```
void setup() {
```

```
Serial.begin(9600);
pinMode(ledPin01, OUTPUT);
digitalWrite(ledPin01,LOW);
```

LEDピンを出力に設定

初期化

```
if (Ethernet.begin(mac) == 0) {
  Serial.println("Failed to configure Ethernet using DHCP");
  // no point in carrying on, so do nothing forevermore:
  // try to configure using IP address instead of DHCP:
  //Ethernet.begin(mac, ip);
}
```

イーサネット通信の開始  
(DHCPによるIPアドレス自動割り当ての使用)

```
delay(1000);
}
```

DHCPが使えない場合は直接指定でイーサネット通信の開始  
(今回はコメントアウト)

# 温度表示サイト：コード(Arduino)

```

void loop() {
    //センサ値読み取り
    sensorValue = analogRead(analogInPin); センサ読み取り
    Serial.println("sensorValue ="); センサ値をシリアルモニタに表示
    Serial.println(sensorValue);
    //温度センサの値を温度(°C)に変換
    outputValue = sensorValue * 10; 少数点1まで送付できるよう10倍した上で、
    outputValue = sensorValue * 0.48; 係数0.48をかけてセンサ値を温度(°C)に変換
    (実際の温度の10倍の値をサーバに送付する)
    Serial.println("outputValue="); 温度値をシリアルモニタに表示
    Serial.println(outputValue); サーバ(温度計サイト)にポート80で接続
}

```

メイン処理  
の記述

```

//サイトに接続 & データ受け渡し
Serial.println("connecting...");
if (client.connect(server, 80)) {
    digitalWrite(ledPin01,LOW);
    Serial.println("connected");
    client.print("GET /arduino_ws04_02/database/add.php?seat_id=");
    client.print(seatID);
    client.print("&temp=");
    client.print(outputValue);
    client.println(" HTTP/1.0");
    client.println();

    delay(100); //100秒待機
    client.stop();
} else {
    Serial.println("connection failed");
    digitalWrite(ledPin01,HIGH);
}
delay(3000); //サーバ負荷低減のための待機
}

```

GETリクエストで座席IDと温度値を送信

※ここではclientに以下の文字列を書き込んでいます。  
 "GET /arduino\_ws04\_02/database/update.php?seat\_id=1&temp=263 HTTP/1.0"  
 これはWebブラウザで以下のようにアクセスするのと同じです。  
 http://153.121.53.226/arduino\_ws04\_02/database/update.php?seat\_id=1&temp=263

接続成功

接続失敗

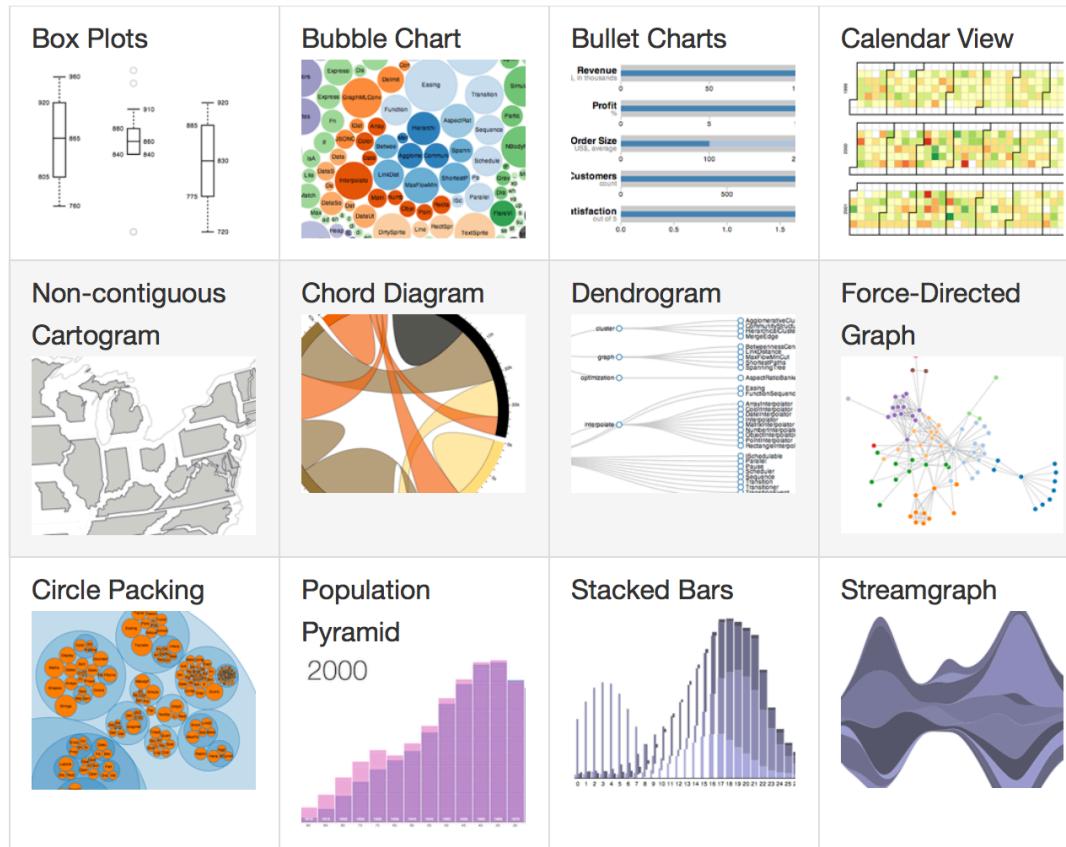
# 温度表示サイト：結果

以下の、アドレスにアクセスして結果を表示します。

[http://153.121.53.226/arduino\\_ws04\\_02/](http://153.121.53.226/arduino_ws04_02/)

# データ可視化ツール 「D3.js」

D3.jsはData-Driven Documentsの略でデータ可視化ツールとして、ウェブブラウザで動的コンテンツを描画するJavaScriptライブラリです。  
<http://ja.d3js.node.ws>



D3.jsでの表示例