```
/*
ブラウザからESP32のポートを制御すると共に、入力ポートの状態を読み取る
*/


// Wi-Fiを使うためWiFi.hをインクルード
#include <WiFi.h>

// Wi-Fi接続情報を入力(IPアドレス自動取得)
const char* ssid     = "XXXXX";  ←お使いのルーターに合わせてください
const char* password = "YYYYY";  ←同上

// ウェブサーバーをポート80で開始
WiFiServer server(80);

// HTTPリクエストを保存しておく変数
String header;

// 出力ピンの状態を保存する変数の宣言
bool IO32 = 0;
bool IO33 = 0;

void setup() {
  // シリアル通信ボーレート設定
  Serial.begin(115200);

  // 入力ピン割り当て(プルアップ抵抗付き)
  pinMode(25, INPUT_PULLUP);
  pinMode(26, INPUT_PULLUP);

  // 出力ピン割り当て
  pinMode(32, OUTPUT);
  pinMode(33, OUTPUT);

  // ２つの出力ピンをオフ (Highでオン、Lowでオフ)
  digitalWrite(32, LOW);
  digitalWrite(33, LOW);

  // ssidとpasswordを用いてWi-Fiに接続
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // 取得したIPアドレスをシリアルに出力し、webserverをスタート
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop() {
  WiFiClient client = server.available();   // クライアント（スマホやPCなど）がつながっているかどうかをclientに出力

  if (client) {                             // クライアントが居るとき
    Serial.println("New Client.");          // クライアントが居ることをシリアルモニタに出力
    String currentLine = "";                // クライアントから来るデータを格納する変数
    while (client.connected()) {            // クライアントがつながっている間、以下をループ
      if (client.available()) {             // クライアントからデータが来ているとき
        char c = client.read();             // データを読み込み
        Serial.write(c);                    // 届いたデータをシリアルモニタに出力
        header += c;                        // header変数に積み込み
        if (c == '¥n') {                    // 届いたデータが改行コードだった時
          // もし現在の行が空白ならば、この改行コードのみ受け取る
          // つまりHTTPリクエストの終わりなので、レスポンスを返す
          if (currentLine.length() == 0) {
            // HTTPヘッダは (HTTP/1.1 200 OK)のようなステータスコードから始まる
            // 次にコンテントタイプを送信。今回はhtml形式なので以下のようにする
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html; charset = utf-8;");
            client.println("Connection: close");
            client.println();

            // リクエストに従ってGPIOをスイッチする
            if (header.indexOf("GET /32/on") >= 0) {
              Serial.println("GPIO 32 on");
              IO32 = 1;
              digitalWrite(32, HIGH);
            }
            else if (header.indexOf("GET /32/off") >= 0) {
              Serial.println("GPIO 32 off");
              IO32 = 0;
              digitalWrite(32, LOW);
```

```
            }
            else if (header.indexOf("GET /33/on") >= 0) {
              Serial.println("GPIO 33 on");
              IO33 = 1;
              digitalWrite(33, HIGH);
            }
            else if (header.indexOf("GET /33/off") >= 0) {
              Serial.println("GPIO 33 off");
              IO33 = 0;
              digitalWrite(33, LOW);
            }

            // htmlを表示
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name = ¥"viewport¥" content = ¥"width = device-width, initial-scale = 1¥">");
            client.println("<link rel = ¥"icon¥" href = ¥"data:,¥">");

            // タイトルを表示
            client.println("<title>ESP32_IO_Test</title>");
            // オン/オフボタンのためのCSS
            client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #5555FF; border: none; color: white; padding: 16px 40px;");
            client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
            client.println(".button2 {background-color: #FF5555;}</style></head>");

            // ページ本体 (bodyタグ内)
            client.println("<body><h1>ESP32_IOお試し</h1>");
            // 現在のピンの状態と、オンオフ用のボタンを出力
            if (IO32 == 0) {                                  // 出力オフの場合は、その旨表示すると共にボタンにオンの属性を持たせる
              client.println("<p>GPIO32はオフです</p>");
              client.println("<p><a href = ¥"/32/on¥"><button class = ¥"button¥">オンにする</button></a></p>");
            }
            else {                                            // 違う場合(出力オンの場合)は、その旨表示すると共にボタンにオフの属性を持たせる
              client.println("<p>GPIO32はオンです</p>");
              client.println("<p><a href = ¥"/32/off¥"><button class = ¥"button button2¥">オフにする</button></a></p>");
            }

            if (IO33 == 0) {                                  // 出力オフの場合は、その旨表示すると共にボタンにオンの属性を持たせる
              client.println("<p>GPIO33はオフです</p>");
              client.println("<p><a href = ¥"/33/on¥"><button class = ¥"button¥">オンにする</button></a></p>");
            }
            else {                                            // 違う場合(出力オンの場合)は、その旨表示すると共にボタンにオフの属性を持たせる
              client.println("<p>GPIO33はオンです</p>");
              client.println("<p><a href = ¥"/33/off¥"><button class = ¥"button button2¥">オフにする</button></a></p>");
            }

            // 入力ピン状態表示(Lowでオン)
            if (digitalRead(25) == 0) {
              client.println("<p>GPIO 25は<font color = ¥"red¥">オン</font>です</p>");
            }
            else {
              client.println("<p>GPIO 25はオフです</p>");
            }
            client.println();                                 // 見やすくするため改行をはさむ
            if (digitalRead(26) == 0) {
              client.println("<p>GPIO 26は<font color = ¥"red¥">オン</font>です</p>");
            }
            else {
              client.println("<p>GPIO 26はオフです</p>");
            }
            client.println("</body></html>");

            //  HTTPレスポンスの最後は改行で終了
            client.println();
            // whileループの終了
            break;
          }
          else {                  // 改行コードを取得したら、currentLineをリセット
            currentLine = "";
          }
        }
        else if (c != '¥r') {    // 改行以外の何かしらのコードが来ているとき
          currentLine += c;      // currentLineに追加
        }
      }
    }
    // ヘッダーをリセット
    header = "";

    // 接続をリセット
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```