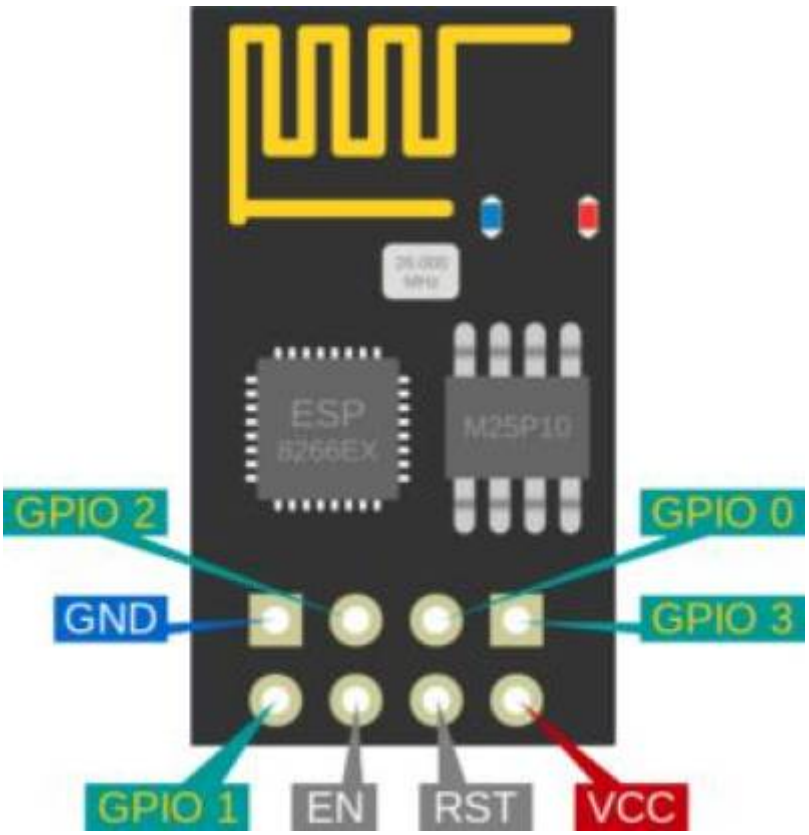
ESP 01 Introduction

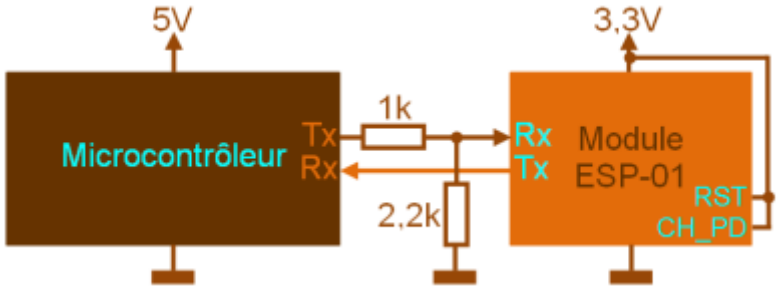
L'ESP-01 est l'une des plus petites unités disponibles. Par rapport aux autres unités ESP, il est alimenté directement et doit donc être alimenté en 3,3 V et non en 5 V. Deux GPIO généraux sont disponibles et si vous avez besoin de plus, vous pouvez utiliser les deux ports série (1,3 / RX,TX).

Matériel



- Version de puce ESP : ESP8266
- Taille du flash : 1M
- Convertisseur USB-TTL intégré : Non
- GPIO éclaté/disponible pour une utilisation gratuite : 0, 2
- Informations sur l'alimentation : 3,3 VDC
- Antenne : antenne PCB intégrée

ESP-01 avec un microcontrôleur fonctionnant en 5V



Câblage/clignotement

Un programmeur est nécessaire pour flasher cet appareil.

- 'Programmateur' 'ESP'
- TX ←→ RX
- RX ←→ TX

- 'Pouvoir'
- 3.3V ←→ VDD
- Terre ←→ Terre

Afin d'obtenir l'unité en mode flash, le GPIO-0 doit être BAS et le CH-EN doit être réglé sur haut. Le CH-EN peut être connecté au VDD, sur les photos un cavalier est soudé entre ceux-ci mais vous pouvez 'utiliser une résistance pour vous assurer qu'aucune surintensité ne blesse l'unité'. Commencez à clignoter et effectuez une réinitialisation une fois afin de lancer le spectacle.

Les unités avec une puce mémoire étiquetée ' PUYA ' doivent être flashées avec les fichiers bin PUYA spécialement construits. Ceci est nécessaire car la puce PUYA doit être traitée d'une autre manière que les puces mem normales.

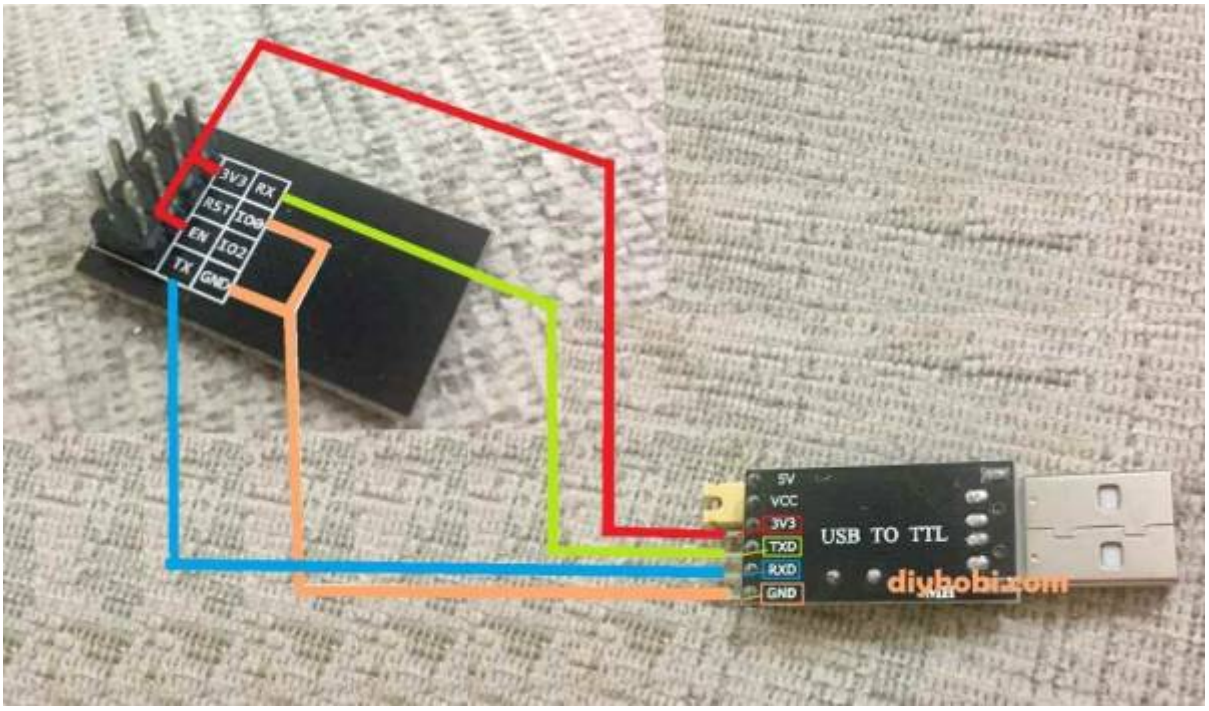
ESP-01 v3

Certains utilisateurs ont signalé que la réinitialisation ne fonctionnait pas sur les versions ultérieures de l'ESP-01. Il y a une résistance supplémentaire de 6k dans la ligne de réinitialisation (la nécessité de cette résistance est inconnue) entre EXT_RSTB et la broche RST du connecteur 8 broches. Après avoir retiré et remis un cavalier à sa place, la fonction RESET fonctionne à nouveau correctement. Donc, essentiellement, vous pontez simplement la résistance, ce qui en fait une ligne droite.

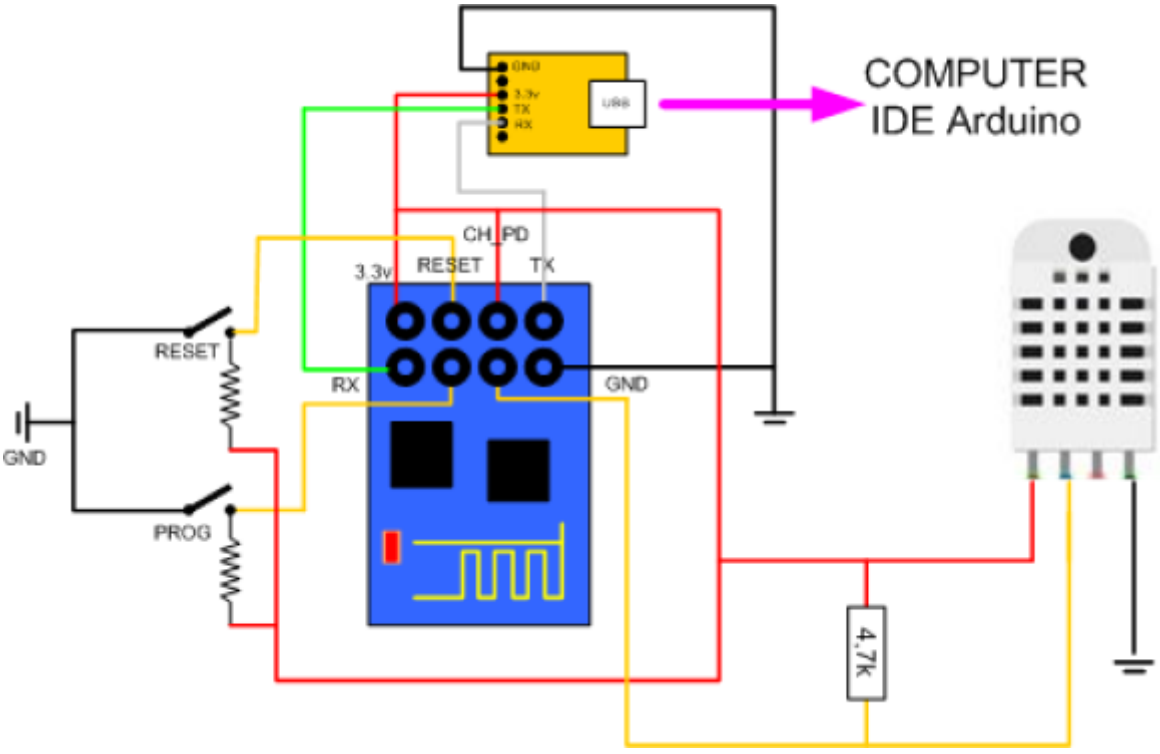
Flashage ESP01

esp01-ota-littlefs

Racordement pour flashage d'un ESP01



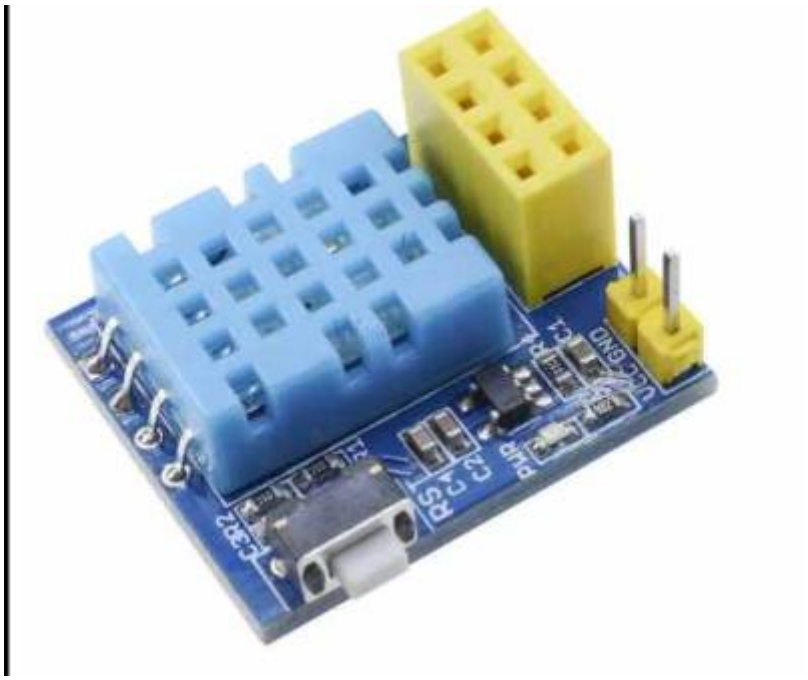
ESP01 et DTH11

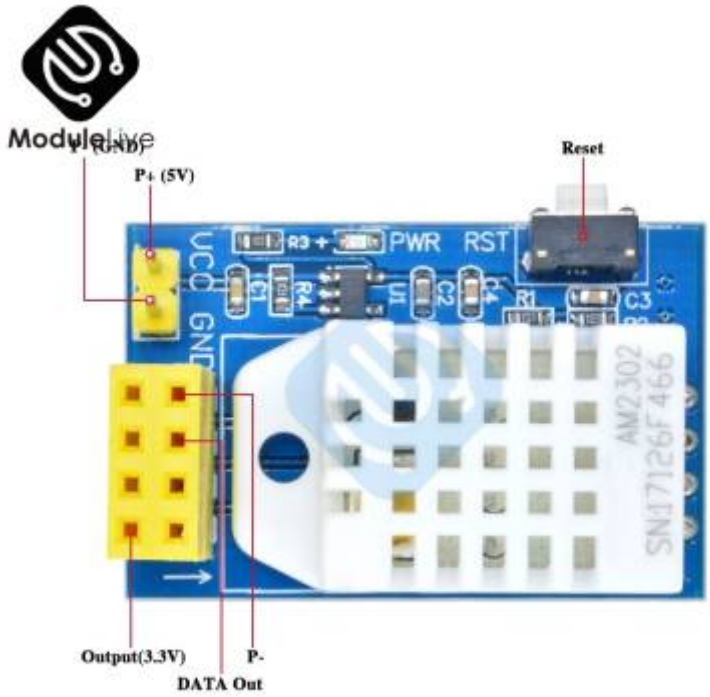


Une video pour expliquer

ESP01 et DHT11 temperature

Ou trouver ce module tout cablé





Pour l'achat du module sur Cdiscount

Pour l'achat du module sur Aliexpress

Code pour esp01 et capteur temperature DHT11 via serveur Web (simplifié)

esp01Tempe001.ino

```
/*
  Objet : Serveur web temperature et humidité avec un esp8266-01
  Nom : phmo_temp_hum_esp_dht_1.ino
  Infos sketch :
  - Créé le 12 jan 2016 sur la base des exemples des bibliotheques
Arduino par PHMO
*/

// ajout des bibs ESP8266 WiFi et sonde dht
#include <ESP8266WiFi.h>
#include "DHT.h"

// la sonde est de type dht22
//#define DHTTYPE DHT22

// la sonde est de type dht11
#define DHTTYPE DHT11

// la sonde dht est connectee sur le port GPIO2 de l esp8266-01
const int DHTPIN=2;

// remplacer par le nom et mot de passe de votre reseau wifi
const char* ssid = "votre nom wifi";
const char* password = "votre mot de passe wifi";

// definition d'un serveur web ecoutant sur le port 80
WiFiServer server(80);

// instanciation objet dht
DHT dht(DHTPIN, DHTTYPE); // Instanciation objet dht

// variables de travail
  static char celsiusTemp[7];
  static char humidityTemp[7];

// traitements init
void setup() {
  // init liaison serie
  Serial.begin(115200);
  delay(10);
  // init catpeur dht
  dht.begin();

  // connexion au wifi
  Serial.println();
  Serial.print("lancement connexion au reseau wifi :  ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("connexion au WiFi OK");

  // demarrage serveur web
  server.begin();
  Serial.println("le serveur web est lancé, attente fourniture ip
locale de l esp ...");
  delay(10000);

  // affichage adresse ip locale
  Serial.println(WiFi.localIP());
}

// boucle de traitement
void loop() {
    // attente client web
    WiFiClient client = server.available();
    if (client) {
        Serial.println("detection dune nouvelle demande client web
.....");
      // bolean to locate when the http request ends
      boolean blank_line = true;
      while (client.connected()) {
          if (client.available()) {
             char c = client.read();
             if (c == '\n' && blank_line) {
                    float h = dht.readHumidity();          // lecture
humidite
                    float t = dht.readTemperature();       // lecture
temperature
                    if (isnan(h) || isnan(t)) {            // verif si
lecture dht22 ok
                      Serial.println("impossible de lire les donnees du
dht22 !!!!!!!");
                      strcpy(celsiusTemp, "probleme");
                      strcpy(humidityTemp, "probleme");
                    }
                    else {
                      // lecture de la sonde ok, envoi des elements sur
le port serie
                      float hic = dht.computeHeatIndex(t, h, false);
                      dtostrf(hic, 6, 2, celsiusTemp);
                      dtostrf(h, 6, 2, humidityTemp);
                      // envoi des message sur la console serie
                      Serial.print("Humidite : ");
                      Serial.print(humidityTemp);
                      Serial.print(" %\t Temperature : ");
```

```
                    Serial.print(celsiusTemp);
                    Serial.print(" *C ");
                  }
                     // envoi des données au client web
                  client.println("HTTP/1.1 200 OK");
                  client.println("Content-Type: text/html");
                  client.println("Connection: close");
                  client.println();
                  // your actual web page that displays temperature
and humidity
                  client.println("<!DOCTYPE HTML>");
                  client.println("<html>");
                  client.println("<head></head><body><h1>ESP8266 -
Temperature and Humidite</h1><h3>Temperature en Celsius : ");
                  client.println(celsiusTemp);
                  client.println("*C</h3><h3>Humidite : ");
                  client.println(humidityTemp);
                  client.println("%</h3><h3>");
                  client.println("</body></html>");
                  break;
              }
              if (c == '\n') {
                // premiere ligne a blanc
                blank_line = true;
              }
              else if (c != '\r') {
                // lignes suivantes avec du contenu
                blank_line = false;
            }
          }
   }
   // fermeture connexion client
   delay(1);
   client.stop();
   Serial.println("client web deconnecte .........");
 }
}
```

Exemple 2

Esp01-DTH11-002.ino

```
/*********
  Rui Santos
  Complete project details at
https://randomnerdtutorials.com/esp8266-dht11dht22-temperature-and-humi
dity-web-server-with-arduino-ide/
*********/
```

```
// Import required libraries
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Hash.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>

// Replace with your network credentials
const char* ssid = "xxxxxxxxxx";
const char* password = "xxxxxxxxxxxxxxxxxxxx";

#define DHTPIN 2     // Digital pin connected to the DHT sensor

// Uncomment the type of sensor in use:
#define DHTTYPE    DHT11     // DHT 11
//#define DHTTYPE    DHT22     // DHT 22 (AM2302)
//#define DHTTYPE    DHT21     // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()
float t = 0.0;
float h = 0.0;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// Generally, you should use "unsigned long" for variables that hold
time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;    // will store last time DHT was
updated

// Updates DHT readings every 10 seconds
const long interval = 10000;

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
  <style>
    html {
```

```
     font-family: Arial;
     display: inline-block;
     margin: 0px auto;
     text-align: center;
    }
    h2 { font-size: 3.0rem; }
    p { font-size: 3.0rem; }
    .units { font-size: 1.2rem; }
    .dht-labels{
      font-size: 1.5rem;
      vertical-align:middle;
      padding-bottom: 15px;
    }
  </style>
</head>
<body>
  <h2>ESP8266 DHT Server</h2>
  <p>
    <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
    <span class="dht-labels">Temperature</span>
    <span id="temperature">%TEMPERATURE%</span>
    <sup class="units">&deg;C</sup>
  </p>
  <p>
    <i class="fas fa-tint" style="color:#00add6;"></i>
    <span class="dht-labels">Humidity</span>
    <span id="humidity">%HUMIDITY%</span>
    <sup class="units">%</sup>
  </p>
</body>
<script>
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML =
this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML =
this.responseText;
    }
  };
```

```
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 10000 ) ;
</script>
</html>)rawliteral";

// Replaces placeholder with DHT values
String processor(const String& var){
  //Serial.println(var);
  if(var == "TEMPERATURE"){
    return String(t);
  }
  else if(var == "HUMIDITY"){
    return String(h);
  }
  return String();
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);
  dht.begin();

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.println("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println(".");
  }

  // Print ESP8266 Local IP Address
  Serial.println(WiFi.localIP());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest
*request){
    request->send_P(200, "text/plain", String(t).c_str());
  });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(h).c_str());
  });

  // Start server
  server.begin();
}
```

```
void loop(){
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you updated the DHT values
    previousMillis = currentMillis;
    // Read temperature as Celsius (the default)
    float newT = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    //float newT = dht.readTemperature(true);
    // if temperature read failed, don't change t value
    if (isnan(newT)) {
      Serial.println("Failed to read from DHT sensor!");
    }
    else {
      t = newT;
      Serial.println(t);
    }
    // Read Humidity
    float newH = dht.readHumidity();
    // if humidity read failed, don't change h value
    if (isnan(newH)) {
      Serial.println("Failed to read from DHT sensor!");
    }
    else {
      h = newH;
      Serial.println(h);
    }
  }
}
```

Exemple 3

Esp01-DTH11-003.ino

```
#include <ESP8266WiFi.h>
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

const char* ssid     = "tpil";
const char* password = "12345678";
const char* host = "bts2m.free.fr";  // Adresse du serveur
const int httpPort = 80;

int value = 0;
float t,h,pile;
unsigned long t0;
```

```
void setup() {
  dht.begin();
  Serial.begin(115200);
  Serial.println(ssid);
  delay(2000);
  Serial.println();
  Serial.print("Connexion a ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay (500);
  Serial.println(WiFi.localIP());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  delay(500);
}

void loop() {
  t0=millis();
  value++;

  WiFiClient client;
  if (!client.connect(host, httpPort))return;

Serial.println(host);
// Mise en forme et envoi de la requête GET au serveur

t = dht.readTemperature();
h = dht.readHumidity();
if (isnan(t)) t=0;
if (isnan(h)) h=0;
pile=5+5*sin(2*3.1416/20*value);
String url = "/Wifi/sql.php?table=1&M1="+String(t,1)+"&M2="+String(h,1)
    +"&M3="+String(pile);

// This will send the request to the server
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");

while (client.available() == 0) {
    if (millis() - t0 > 5000) {
      client.stop();
      return;
    }}
while (client.available()) {
     String line = client.readStringUntil('\r');
```

```
     Serial.print(line);}
// delay(30000);
while (millis()-t0<30000) delay(10);    // 30s entre 2 mesures
}
```

Exemple 4

Esp01-DTH11-004.ino

```
#include <ESP8266WiFi.h>
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

const char* ssid     = "tpil";
const char* password = "12345678";
const char* host = "bts2m.free.fr";  // Adresse du serveur
const int httpPort = 80;

int value = 0;
float t,h,pile;
unsigned long t0;

void setup() {
  dht.begin();
  Serial.begin(115200);
  Serial.println(ssid);
  delay(2000);
  Serial.println();
  Serial.print("Connexion a ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay (500);
  Serial.println(WiFi.localIP());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  delay(500);
}

void loop() {
  t0=millis();
  value++;

  WiFiClient client;
```

```
  if (!client.connect(host, httpPort))return;

Serial.println(host);
// Mise en forme et envoi de la requête GET au serveur

t = dht.readTemperature();
h = dht.readHumidity();
if (isnan(t)) t=0;
if (isnan(h)) h=0;
pile=5+5*sin(2*3.1416/20*value);
String url = "/Wifi/sql.php?table=1&M1="+String(t,1)+"&M2="+String(h,1)
    +"&M3="+String(pile);

// This will send the request to the server
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");

while (client.available() == 0) {
    if (millis() - t0 > 5000) {
      client.stop();
      return;
    }}
while (client.available()) {
     String line = client.readStringUntil('\r');
     Serial.print(line);}
// delay(30000);
while (millis()-t0<30000) delay(10);    // 30s entre 2 mesures
}
```

Exemple 5

Esp01-Dth11-005.ino

```
/* DHTServer - ESP8266 Webserver with a DHT sensor as an input

   Based on ESP8266Webserver, DHTexample, and BlinkWithoutDelay (thank
you)

   Version 1.0  5/3/2014  Version 1.0   Mike Barela for Adafruit
Industries
*/
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#define DHTTYPE DHT11
```

```
#define DHTPIN  2
// Replace with your network details
const char* ssid     = "YourRouterID";
const char* password = "YourRouterPassword";
*/
ESP8266WebServer server(80);

//Initialiser le capteur DHT
// REMARQUE : Pour travailler avec une puce Arduino ATmega328p 16 MHz plus rapide, comme un ESP8266,
// vous devez augmenter le seuil pour les comptages de cycles considérés comme 1 ou 0.
// Vous pouvez le faire en passant un 3ème paramètre pour ce seuil. C'est un peu
// de bidouiller pour trouver la bonne valeur, mais en général plus le CPU est rapide plus
// augmente la valeur. La valeur par défaut pour un AVR 16 MHz est une valeur de 6. Pour un
// Arduino Due qui tourne à 84mhz une valeur de 30 fonctionne.
// C'est pour le processeur ESP8266 sur ESP-01

DHT dht(DHTPIN, DHTTYPE, 11); // 11 works fine for ESP8266

float humidity, temp_c;  // Values read from sensor
String webString="";     // String to display
// Generally, you should use "unsigned long" for variables that hold time
unsigned long previousMillis = 0;        // will store last temp was read
const long interval = 2000;              // interval at which to read sensor

void handle_root() {
  server.send(200, "text/plain", "Hello from the weather esp8266, read from /temp or /humidity");
  delay(100);
}

void setup(void)
{
  // You can open the Arduino IDE Serial Monitor window to see what the code is doing
  Serial.begin(115200);  // Serial connection from ESP-01 via 3.3v console cable
  dht.begin();           // initialize temperature sensor

  // Connect to WiFi network
  WiFi.begin(ssid, password);
  Serial.print("\n\r \n\rWorking to connect");

  // Wait for connection
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("DHT Weather Reading Server");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  server.on("/", handle_root);

  server.on("/temp", [](){  // if you add this subdirectory to your
webserver call, you get text below :)
    gettemperature();       // read sensor
    webString="Temperature: "+String((int)temp_c)+" C";   // Arduino
has a hard time with float to string
    server.send(200, "text/plain", webString);            // send to
someones browser when asked
  });

  server.on("/humidity", [](){  // if you add this subdirectory to your
webserver call, you get text below :)
    gettemperature();           // read sensor
    webString="Humidite: "+String((int)humidity)+"%";
    server.send(200, "text/plain", webString);               // send to
someones browser when asked
  });

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void)
{
  server.handleClient();
}

void gettemperature() {
  // Wait at least 2 seconds seconds between measurements.
  // if the difference between the current time and last time you read
  // the sensor is bigger than the interval you set, read the sensor
  // Works better than delay for things happening elsewhere also
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= interval) {
    // save the last time you read the sensor
    previousMillis = currentMillis;
```

```
    // Reading temperature for humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (it's a very
slow sensor)
    humidity = dht.readHumidity();          // Read humidity (percent)
    /*Fahrenheit est une échelle de température thermodynamique, où le
point de congélation de l'eau est
    à 32 degrés Fahrenheit (°F)et le point d'ébullition à 212 °F (sous
une pression atmosphérique normale).
    Cela sépare les points d'ébullition et de congélation de l'eau
d’exactement 180 degrés. Par conséquent,
    un degré sur l'échelle Fahrenheit représente 1/180 de l'intervalle
entre le point de congélation et
    le point d'ébullition de l'eau. Le zéro absolu est défini comme
égal      à-459,67 °F.

    Celsius :Bien qu'initialement défini comme le point de congélation
de l'eau (et plus tard le point de
    fusion de la glace),  l'échelle Celsius est maintenant
officiellement une échelle dérivée, définie par
    rapport à la l’échelle de température Kelvin .Sur l'échelle Celsius
le zéro (0 °C) est maintenant
    défini comme égal à 273,15 K, avec une différence de température de
1 deg C équivalent à une différence
    de 1 K, c’est-à-dire que la taille de l'unité sur chaque échelle
est la même.Cela signifie que 100 °C,
    préalablement défini comme le point d'ébullition de l'eau, est
maintenant défini comme l'équivalent de
    373,15 K.Une différence de température d’1 °F équivaut à une
différence de température de 0,556 °C.
    /*
    //Pour avoir la temperature en °Celcius à partir de ° Fahrenheit
appliquons la formule :
    // °Celius = (°F -32)/1.800
    temp_c = ((dht.readTemperature(true) -32)/1.800) ;     // Read
temperature as  °Celcius
    // Check if any reads failed and exit early (to try again).
    if (isnan(humidity) || isnan(temp_c)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }
  }
}
```

ESP01 Temperature et MQTT

Le module ESP8266 ESP01 envoie les données de température à Adafruit MQTT

liens web

ESP01 8266

Presentation du module ESP01

Datasheet ESP01

Mise en route ESP01

ESP01 et un relais

Esp01 et un relais

Fiche technique Relais ESP01

Programme ESP01-Relais

Esp01-Relais-001.ino

```
  /*-----------------------------------------------------
HTTP 1.1 Webserver for ESP8266
for ESP8266 adapted Arduino IDE
http://www.esp8266.com/viewtopic.php?p=65572

-----------------------------------------------------*/

#include <ESP8266WiFi.h>
```

```
const char* ssid     = "FREE";
const char* password = "WUFU";
int ledState = false;
unsigned long ulReqcount;
unsigned long ulReconncount;

byte relON[] = {0xA0, 0x01, 0x01, 0xA2};  //Hex command to send to
serial for open relay
byte relOFF[] = {0xA0, 0x01, 0x00, 0xA1}; //Hex command to send to
serial for close relay

// Create an instance of the server on Port 80
WiFiServer server(80);

void setup()
{
  // setup globals
  ulReqcount=0;
  ulReconncount=0;

  // start serial
  Serial.begin(9600);
  delay(1);

  // inital connect
  WiFi.mode(WIFI_STA);
  WiFiStart();
}

void WiFiStart()
{
  ulReconncount++;

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");
```

```
  // Print the IP address
  Serial.println(WiFi.localIP());
}

void loop()
{
  // check if WLAN is connected
  if (WiFi.status() != WL_CONNECTED)
  {
    WiFiStart();
  }

  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client)
  {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  unsigned long ultimeout = millis()+250;
  while(!client.available() && (millis()<ultimeout) )
  {
    delay(1);
  }
  if(millis()>ultimeout)
  {
    Serial.println("client connection time-out!");
    return;
  }

  // Read the first line of the request
  String sRequest = client.readStringUntil('\r');
  //Serial.println(sRequest);
  client.flush();

  // stop client, if request is empty
  if(sRequest=="")
  {
    Serial.println("empty request! - stopping client");
    client.stop();
    return;
  }

  // get path; end of path is either space or ?
  // Syntax is e.g. GET /?pin=MOTOR1STOP HTTP/1.1
  String sPath="",sParam="", sCmd="";
  String sGetstart="GET ";
```

```
  int iStart,iEndSpace,iEndQuest;
  iStart = sRequest.indexOf(sGetstart);
  if (iStart>=0)
  {
    iStart+=+sGetstart.length();
    iEndSpace = sRequest.indexOf(" ",iStart);
    iEndQuest = sRequest.indexOf("?",iStart);

    // are there parameters?
    if(iEndSpace>0)
    {
      if(iEndQuest>0)
      {
        // there are parameters
        sPath  = sRequest.substring(iStart,iEndQuest);
        sParam = sRequest.substring(iEndQuest,iEndSpace);
      }
      else
      {
        // NO parameters
        sPath  = sRequest.substring(iStart,iEndSpace);
      }
    }
  }

////////////////////////////////////////////////////////////////////////
////////
  // output parameters to serial, you may connect e.g. an Arduino and
react on it
////////////////////////////////////////////////////////////////////////
////////
  if(sParam.length()>0)
  {
    int iEqu=sParam.indexOf("=");
    if(iEqu>=0)
    {
      sCmd = sParam.substring(iEqu+1,sParam.length());
      Serial.println(sCmd);
    }
  }


  ////////////////////////////
  // format the html response
  ////////////////////////////
  String sResponse,sHeader;

  ////////////////////////////
  // 404 for non-matching path
  ////////////////////////////
  if(sPath!="/")
```

```
  {
    sResponse="<html><head><title>404 Not
Found</title></head><body><h1>Not Found</h1><p>The requested URL was
not found on this server.</p></body></html>";

    sHeader  = "HTTP/1.1 404 Not found\r\n";
    sHeader += "Content-Length: ";
    sHeader += sResponse.length();
    sHeader += "\r\n";
    sHeader += "Content-Type: text/html\r\n";
    sHeader += "Connection: close\r\n";
    sHeader += "\r\n";
  }
  ////////////////////
  // format the html page
  ////////////////////
  else
  {
    ulReqcount++;
    sResponse  = "<html><head><title>Demo pour ESP8266 version
ESP-01</title></head><body>";
    sResponse += "<font color=\"#000000\"><body bgcolor=\"#d0d0f0\">";
    sResponse += "<meta name=\"viewport\" content=\"width=device-width,
initial-scale=1.0, user-scalable=yes\">";
    sResponse += "<h1>Demo pour ESP8266 version ESP-01</h1>";
    sResponse += "Allumez en cliquant sur le bouton.<BR>";
    sResponse += "<FONT SIZE=+1>";
    sResponse += "<p>Funktion 1 <a
href=\"?pin=FUNCTION1ON\"><button>Allumer</button></a> <a
href=\"?pin=FUNCTION1OFF\"><button>Eteindre</button></a></p>";

    ////////////////////
    // react on parameters

    ////////////////////
    if (sCmd.length()>0)
    {
      // write received command to html page
      sResponse += "Kommando:" + sCmd + "<BR>";

      // switch GPIO
      if(sCmd.indexOf("FUNCTION1ON")>=0)
      {
      Serial.write(relON, sizeof(relON));     // turns the relay ON
      ledState = false;
      }
      else if(sCmd.indexOf("FUNCTION1OFF")>=0)
      {
      Serial.write(relOFF, sizeof(relOFF));   // turns the relay OFF
      ledState = true;
```

```
      }
    }

    sResponse += "<FONT SIZE=-2>";
    sResponse += "<BR>Aufrufz&auml;hler=";
    sResponse += ulReqcount;
    sResponse += " - Verbindungsz&auml;hler=";
    sResponse += ulReconncount;
    sResponse += "<BR>";
    sResponse += "</body></html>";

    sHeader  = "HTTP/1.1 200 OK\r\n";
    sHeader += "Content-Length: ";
    sHeader += sResponse.length();
    sHeader += "\r\n";
    sHeader += "Content-Type: text/html\r\n";
    sHeader += "Connection: close\r\n";
    sHeader += "\r\n";
  }

  // Send the response to the client
  client.print(sHeader);
  client.print(sResponse);

  // and stop the client
  client.stop();
  Serial.println("Client disonnected");
}
/*
test a faire
#define RX_PIN 3   // GPIO3
#define TX_PIN 1   // GPIO1

void setup() {
// dont Serial.begin(74880)!!!!!!!!!!!!!!!!!!
  pinMode(RX_PIN, INPUT);
  pinMode(TX_PIN, INPUT);
}
*/
```