# Reconnecter ESP32 au WIFI après une connexion perdue

Article original EN

Dans ce tutoriel, nous allons apprendre à reconnecter la carte ESP32 au réseau WIFI lorsqu'elle perd temporairement la connexion. À travers différentes méthodes, nous allons démontrer cette fonctionnalité. Par conséquent, ce guide sera très utile pour éviter la nuisance de perdre constamment la connexion réseau pour de nombreuses raisons telles que le redémarrage du routeur WiFi, l'ESP32 est hors de portée du routeur WiFi, le routeur perd de l'alimentation et redémarre.

### 1ère méthode : fonction WiFi.reconnect()

Dans votre croquis Arduino, utilisez WiFi.connect() fonction pour rétablir la connexion au réseau auquel la carte ESP32 était précédemment connectée.

002.ino

```
WiFi.disconnect()
WiFi.begin(ssid,password):
```

Vous pouvez également utiliser la fonction WiFi.disconnect() puis WiFi.begin(ssid,password) dans un ordre particulier. La fonction WiFi.begin(ssid,password) prendra deux arguments.

- Le nom de votre réseau WIFI appelé SSID est le premier argument.
- Le deuxième argument est le mot de passe de votre réseau WIFI associé. Assurez-vous de spécifier les fonctions dans le bon ordre.

### 2ème méthode : ESP.restart()

L'autre moyen de se reconnecter au WiFi consiste à redémarrer l'appareil ESP32. En utilisantESP.restart()fonction, nous pouvons redémarrer la carte de développement ESP32. Cela peut récupérer la connexion réseau perdue.

Mettre en œuvre le ESP.restart()fonction à l'intérieur de la fonction loop() pour vérifier si votre module ESP est connecté au WiFi. Si la connexion est perdue, il se reconnecte viaWiFi.reconnect() une fonction. Esquisse Arduino

Suivez le croquis Arduino ci-dessous. La fonction boucle infinie () exécutera la fonctionnalité susmentionnée. Après toutes les 20 secondes, il vérifiera si la carte est toujours connectée au WIFI ou non. Sinon, il essaiera de se reconnecter en se déconnectant d'abord, puis en essayant de se reconnecter.

003.ino

```
#include <WiFi.h>

const char* ssid = "Your_SSID";
const char* password = "Your_Password";

unsigned long previous_time = 0;
unsigned long delay = 20000;  // 20 seconds delay

void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WIFI network");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}

void setup() {
  Serial.begin(115200);
  initWiFi();
}

void loop() {
  unsigned long current_time = millis(); // number of milliseconds
since the upload

  // checking for WIFI connection
  if ((WiFi.status() != WL_CONNECTED) && (current_time - previous_time
>=delay)) {
    Serial.print(millis());
    Serial.println("Reconnecting to WIFI network");
    WiFi.disconnect();
    WiFi.reconnect();
    previous_time = current_time;
  }
}
```

Vous pouvez augmenter le temps de revérification en modifiant la valeur du délai dans la ligne suivante :

004.ino

```
unsigned long delay = 20000; // 20 seconds delay
```

Cette méthode est une bonne solution pour reconnecter le WiFi à l'ESP32 en cas de perte accidentelle de la connexion. Mais avec cette méthode, nous devons interroger la if [1]condition à chaque exécution

de loop().

L'alternative à la loop()méthode consiste à utiliser ESP32 WIFI Events. Nous allons montrer en détail une autre fonctionnalité utile connue sous le nom d'événements WIFI. Cela aidera à détecter la connexion réseau perdue. De plus, une fonction sera appelée pour gérer une reconnexion réussie. Ceci est discuté dans la section ci-dessous.

### 3ème méthode : événements WiFi ESP32

La méthode de boucle infinie () indiquée ci-dessus aidera à récupérer la connexion WIFI perdue, mais est légèrement gênante et gaspille les ressources ESP32. Pour une meilleure approche, nous allons essayer d'utiliser les événements ESP32 WIFI à la place. Les événements WIFI détecteront directement l'absence de WIFI. De plus, il appellera simultanément une fonction de gestion pour se reconnecter au réseau.

À des fins de détection et de reconnexion, deux événements WIFI sont très utiles. Ceux-ci sont donnés ci-dessous.

- SYSTEM_EVENT_STA_CONNECTED: L'ESP32 est en mode station et est connecté à un point d'accès (AP).
- SYSTEM_EVENT_STA_DISCONNECTED: L'ESP32 est en mode station et est déconnecté du point d'accès (AP).

Nous utiliserons un croquis Arduino pour expliquer comment les événements WIFI fonctionneront. Grâce à ces événements, l'ESP32 pourra se reconnecter au routeur instantanément après avoir perdu la connexion. Nous utiliserons les deux événements mentionnés ci-dessus à cet égard.

#### Esquisse Arduino

Ouvrez votre IDE Arduino et accédez à Fichier > Nouveau pour ouvrir un nouveau fichier. Copiez le code ci-dessous dans ce fichier. Ce code fonctionnera pour votre carte de développement ESP32. Vous devez remplacer les informations d'identification du réseau.

005.ino

```
#include <WiFi.h>
const char* ssid = "Your_SSID";
const char* password = "Your_Password";

void Wifi_connected(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("Successfully connected to Access Point");
}

void Get_IPAddress(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("WIFI is connected!");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
void Wifi_disconnected(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("Disconnected from WIFI access point");
  Serial.print("WiFi lost connection. Reason: ");
  Serial.println(info.disconnected.reason);
  Serial.println("Reconnecting...");
  WiFi.begin(ssid, password);
}

void setup(){
  Serial.begin(115200);
  WiFi.disconnect(true);
  delay(1000);

  WiFi.onEvent(Wifi_connected,SYSTEM_EVENT_STA_CONNECTED);
  WiFi.onEvent(Get_IPAddress, SYSTEM_EVENT_STA_GOT_IP);
  WiFi.onEvent(Wifi_disconnected, SYSTEM_EVENT_STA_DISCONNECTED);
  WiFi.begin(ssid, password);
  Serial.println("Waiting for WIFI network...");
}

void loop(){
  delay(1000);
}
```

**Comment fonctionne le code ?**

Premièrement, nous allons inclure la WiFi.hbibliothèque. Cette bibliothèque aidera à établir la connexion entre notre module ESP32 à un réseau sans fil.

006.ino

```
#include <WiFi.h>
```

Ensuite, nous allons créer deux variables globales, une pour le SSIDet l'autre pour le mot de passe. Ceux-ci contiendront nos identifiants de réseau qui seront utilisés pour se connecter à notre réseau sans fil. Remplacez-les tous les deux par vos informations d'identification pour assurer une connexion réussie.

007.ino

```
const char* ssid = "Your_SSID";
const char* password = "Your_Password";
```

Fonctions définies par l'utilisateur

Ensuite, nous définirons des fonctions que nous devrons appeler plus tard dans le code. Dans un

premier temps, nous définirons leWifi_connecté()une fonction. Cette fonction sera appelée lorsque notre ESP32 se connectera à un point d'accès via leSYSTEM_EVENT_STA_CONNECTEDun événement. Cette fonction imprimera : « Connexion réussie au point d'accès » sur notre moniteur série.

008.ino

```
void Wifi_connected(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("Successfully connected to Access Point");
}
```

Dans un deuxième temps, nous définirons le Get_IPAddress()une fonction. Celui-ci sera appelé lorsque notre carte ESP32 utilisera l'événementSYSTEM_EVENT_STA_GOT_IPpour accéder à son adresse IP. Grâce à cette fonction Get_IPAddress(), nous imprimerons « WIFI est connecté ! » et l'adresse IP de notre module sur le moniteur série.

009.ino

```
void Get_IPAddress(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("WIFI is connected!");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Enfin, nous définirons le Wifi_déconnecté()une fonction. Cette fonction sera appelée lorsque notre ESP32 se déconnectera d'un point d'accès via l'événement SYSTEM_EVENT_STA_DISCONNECTED. Cette fonction imprimera « Déconnecté du point d'accès WIFI ». Il précisera également la raison dans un numéro de code pour la déconnexion. De plus, il essaiera également de se reconnecter au réseau via leWi-Fi.commence() une fonction.

010.ino

```
void Wifi_disconnected(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("Disconnected from WIFI access point");
  Serial.print("WiFi lost connection. Reason: ");
  Serial.println(info.disconnected.reason);
  Serial.println("Reconnecting...");
  WiFi.begin(ssid, password);
}
```

#### Fonction Setup()

À l'intérieur de mettre en place() fonction, nous ouvrirons la communication série à un débit en bauds de 115200.

012.ino

```
Serial.begin(115200);
```

Ensuite, nous appellerons WiFi.déconnecter()fonction avec true comme paramètre à l'intérieur. Cela supprimera toutes les informations d'identification réseau précédentes stockées dans la carte.

013.ino

```
WiFi.disconnect(true);
```

Ensuite, nous appellerons WiFi.onEvent()fonction pour les trois événements WIFI.

Ceux-ci seront SYSTEM_EVENT_STA_CONNECTED, SYSTEM_EVENT_STA_GOT_IP et SYSTEM_EVENT_STA_DISCONNECTED.

Nous les passerons individuellement en tant que deuxièmes paramètres à l'intérieur de la fonction WiFi.onEvent(). Les fonctions précédemment définies par l'utilisateur agiront comme premiers paramètres.

014.ino

```
WiFi.onEvent(Wifi_connected,SYSTEM_EVENT_STA_CONNECTED);
WiFi.onEvent(Get_IPAddress, SYSTEM_EVENT_STA_GOT_IP);
WiFi.onEvent(Wifi_disconnected, SYSTEM_EVENT_STA_DISCONNECTED);
```

Ensuite, nous connecterons notre carte ESP32 au point d'accès via le Wi-Fi.commence() une fonction.

001.ino

```
  WiFi.begin(ssid, password);
  Serial.println("Waiting for WIFI network...");
```

#### Démonstration

Assurez-vous de choisir la bonne carte et le bon port COM avant de télécharger votre code sur la carte.

Allez dans Outils > Carte et sélectionnez ESP32 Dev Module.

Ensuite, allez dans Outils > Port et sélectionnez le port approprié via lequel votre carte est connectée.

Cliquez sur le bouton de téléchargement pour télécharger le code dans la carte de développement ESP32.

Après avoir téléchargé votre code sur la carte de développement, appuyez sur son bouton ACTIVER.

Ouvrez votre Serial Monitor à un débit en bauds de 115200.

Connectez-vous maintenant à votre réseau WIFI local.

Interrompre la connexion. Le code du programme le détecte automatiquement et se reconnecte.

Reconnecter ESP32 à la démo WIFI Moniteur série

[1)]

WiFi.status() != WL_CONNECTED) && (current_time – previous_time >=delay