WifiManager.cpp

Documentation :

Cette classe se concentre sur l'initialisation de la classe WIFI Arduino

void initAP();

INITIALISATION ACCESS POINT

Déclare l'ip, la gateway, le masque de sous-réseau, l'adresse MAC et configure le point d'accès Wifi en softAP (= routeur). Nomme le point d'accès ESP8266 + adresse MAC avec le mot de passe "password"

*bool connexion(char *ssid, char *pwd, int expiration);*

Parse les IP, Gateway et DNS de l'Application, configure le wifi avec ces infos et tente la connexion. Tant que la connexion n'a pas réussi ou que le timeout n'est pas atteinte, on fait clignoter la LED pour signifier à l'utilisateur que la connexion est en cours. Si la connexion réussit, on allume définitivement la LED sinon on l'éteint. Enfin on renvoie si oui ou non la connexion a réussi.

String ipToString(IPAddress ip);

Mets en forme texte l'objet IPAddress. Fait office de toString().

void parseBytes(const char str, char sep, byte* bytes, int maxBytes, int base);*

Fait office de parseur d'adresse IP, Gateway, DNS vers un tableau de byte. On passe l'adresse à convertir, le séparateur '.', le tableau de byte qu'on souhaite récupérer par adresse, le nombre max de byte du tableau et enfin la base vers laquelle on souhaite convertir.

Pour i allant de 0 à maxBytes, on convertit dans une certaine base (10 souvent), on relit la chaîne jusqu'au séparateur, si on est à NULL ou '\0' on stop ici sinon on passe au premier caractère après le séparateur.

*WifiManager(Application *c);*

Constructeur de WifiManager, initialise lconfig aux données de l'Application passée en paramètre

void initWifi();

Vérifie la présence d'un fichier de sauvegarde qui certifie que le WIFI a été initialisé. Si non, il appelle InitAP(), puis il convertit en char* toutes les Strings pour permettre la lecture caractère par caractère.

Il teste si connexion(ssid, pwd, 20) renvoie true, si oui il ajoute la sauvegarde. Sinon il demande la reconfiguration du WIFI par l'utilisateur.

String getWiFiInfo();
Renvoie une chaîne de caractère contenant le SSID, l'adresse IP, le WSSID (="`ESP8266`") et le WPASS ("`esppwd`").

void disconnect();
Appelle la méthode static "disconnect()" de Wifi pour mettre hors ligne le réseau.

DIAGRAMME DE CLASSE :

WifiManager
- Application *lconfig;
+ WifiManager(Application *c); - void initAP(); + void initWifi(); + void disconnect(); + String getWiFiInfo(); - void parseBytes(const char* str, char sep, byte* bytes, int maxBytes, int base); - String ipToString(IPAddress ip); - bool connexion(char *ssid, char *pwd, int expiration);

Patrons de conceptions :

- Façade ? Présence d'un Manager qui fait le lien entre la classe Arduino Wifi et l'application pour la connexion. On réduit ainsi le couplage entre les deux.

QUESTIONS :

Ligne 34 -> 38 ??
Quand demande t-on à l'utilisateur de rentrer les infos de connexion ?