**NexWiFiClass**

* void setSsid(std::string ssid): função para definir o nome da rede que será utilizada;
* void setPassword(std::string password): função para definir a senha da rede que será utilizada;
* void setCheckConnection(bool checkConnection): função para definir se a rede deve ter conexão com a internet;
* void setAttempt2Find(int attempt2Find): função para definir a quantidade de tentativas máxima para encontrar a rede;
* void setAttempt2Connection(int attempt2Connection): função para definir a quantidade de tentativas máxima de se conectar à rede;
* void setMicroseconds(unsigned int microseconds): função para definir o tempo, em microssegundos, entre tentativas de se conectar à rede;
* void nexDelay(): função que gera um delay otimizado, o valor definido em *setMicroseconds()* será levado em consideração;
* void printDiag(Print& dest): função que imprime na porta *dest* um diagnóstico da rede wifi;
* void checkInternetConnectionCallBack(void (\*sucess)()): função que verifica se uma rede está conectada à internet. Em caso de sucesso executa a função *sucess()*;
* void startStaCallback(fs::FS& store, void (\*sucess)(), void(\*failure)(), void(\*notFound)(), void(\*failureInternet)()): função que verifica se uma rede está conectada à internet. Em caso de sucesso executa a função *sucess(),* em caso de falha de conexão executa a função *failure()*, em caso de não localizar o nome da rede executa a função *notFound()* e em caso de encontrar uma rede que não esteja conectada à internet executa a função *failureInternet()*;
* int checkInternetConnectionDefault(): função que verifica se uma rede está conectada à internet e retorna uma variável que reflete o estado da operação;
* int startStaDefault(fs::FS& store): função que inicializa uma conexão do tipo station e retorna uma variável que reflete o estado da operação. Em caso de sucesso ao se conectar, as informações da rede serão salvas no *store*;
* void startApDefault(): função que inicializa uma conexão do tipo access point com o nome da rede, senha, ip\_local, gateway, subnet definidos respectivamente como: NexIoT\_+MAC, nexsolar, 192.168.4.1, 192.168.4.9 e 255.255.255.0;
* void startStaCallback(fs::FS& store, void (\*sucess)()) função que inicializa uma conexão do tipo station e ,em caso de sucesso, salva as informações da rede no *store* e executa a função *sucess()*;
* void nexReconnect(): função que realiza a reconexão da rede e em caso de múltiplas falhas executa a função ESP.restart();
* void startServer(AsyncWebServer\* server): função que define um servidor HTTP default conforme configurações definidas no Ap;
* void upServerDefault(): função que configura o Ap e em seguido inicializa um servidor;
* #define SSID\_NOT\_FOUND -3, caso esp não encontre o nome da rede;
* #define CONNECTION\_FAILURE -4, caso esp não consiga se conectar à rede existente;
* #define SUCCESSFUL\_INTERNET\_CONNECTION 1, caso esp tenha sucesso em se conectar a uma rede com acesso a internet;
* #define FAILURE\_IN\_INTERNET\_CONNECTION 0, caso esp tenha se conecte a uma rede sem acesso à internet;

\*Obs: É valido lembrar que essa biblioteca deriva de outras bibliotecas WiFi e para seu correto funcionamento, assim todas as demais funções WiFi podem ser usadas normalmente através do objeto NexWiFi. Essa derivação também implica que ao compilar a biblioteca são necessárias as demais bibliotecas WiFi (também a HTTPClient.h, SPIFFS.h, ESPAsyncWebServer.h)