La carte Arduino MKR Wifi 1010

[Mise à jour le 9/7/2022]

Sources sur le site Arduino

- MKR FAMILY
- Getting started with the MKR WiFi 1010

Lecture connexe

- Wiki matériels - "Capteurs, afficheurs, préactionneurs, etc."
- Wiki Arduino - "Mettre en oeuvre un client MQTT sur un EP8266 (ESP32) Feather Huzzah ou Wifi MKR1010"
- Arduino Library List

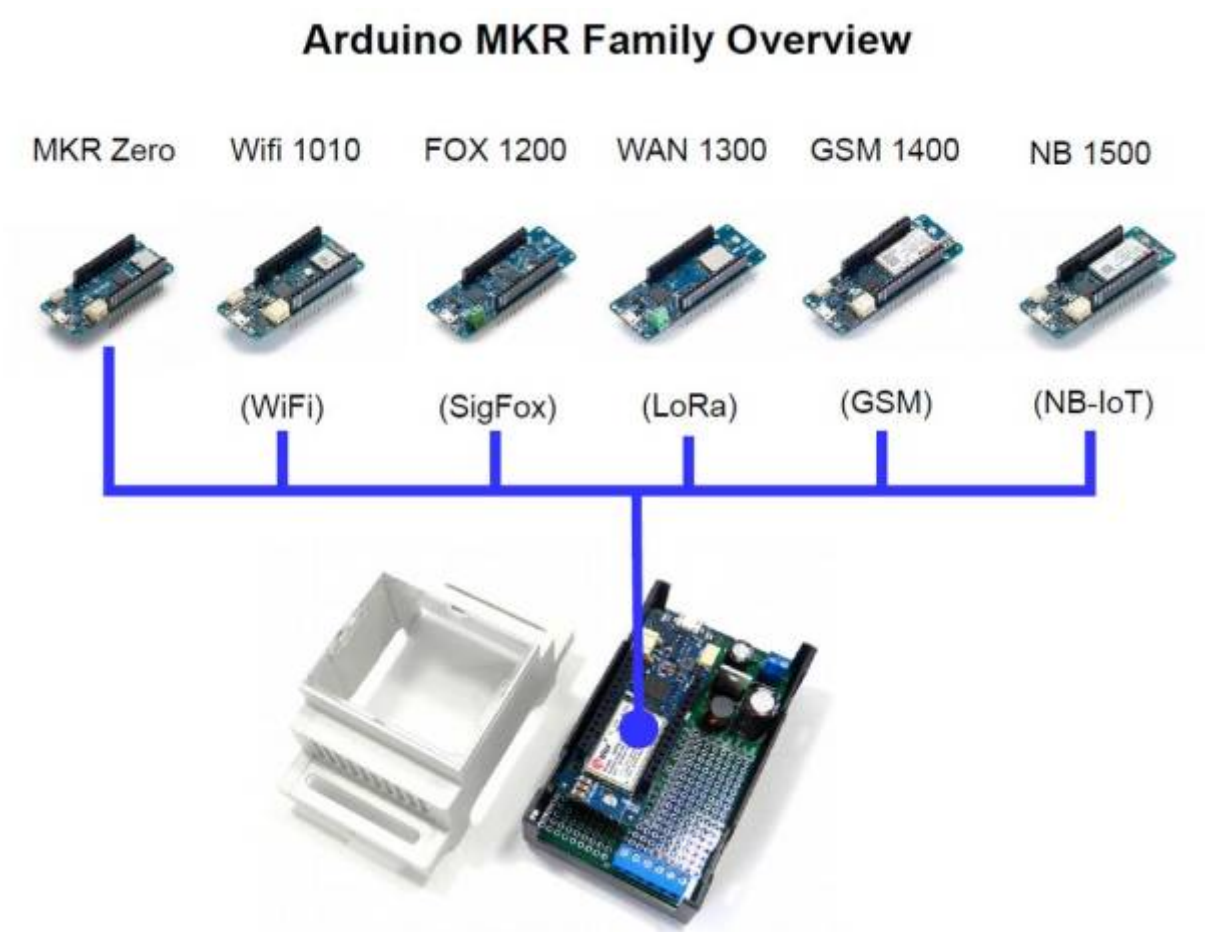
Distributeur

- Mouser

---

1. La gamme MKR

La gamme **MKR** est une référence dans le domaine des cartes de développement **IoT**. Elle regroupe plusieurs cartes dont le tarif varie entre 20 et 60 euros (2019).

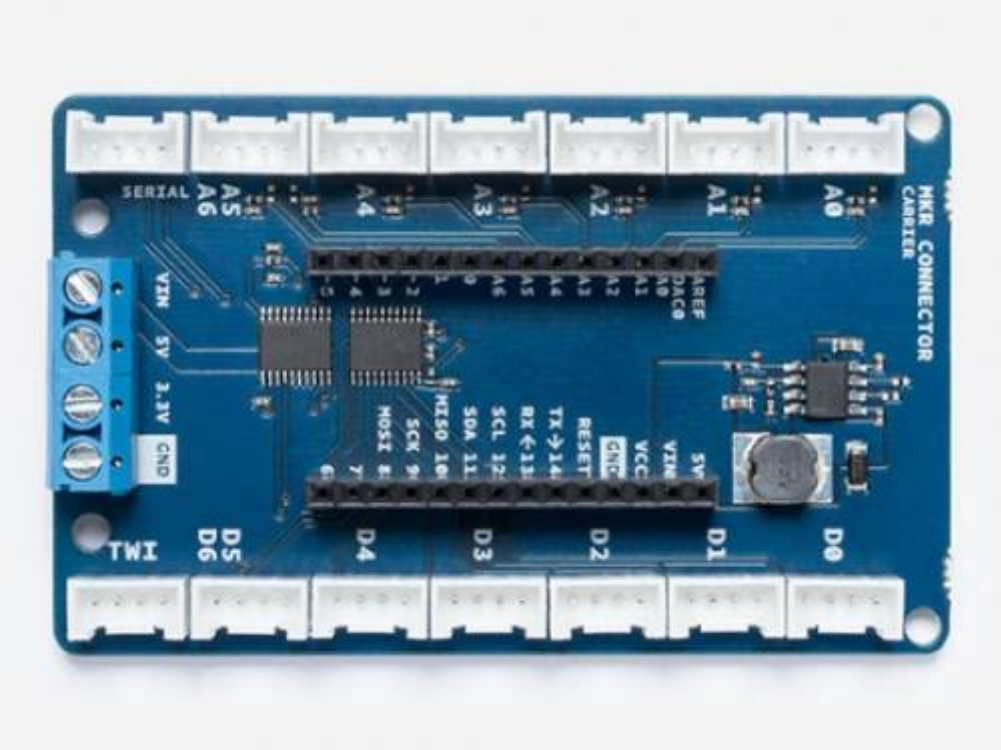


1.1 Connectivité

Toutes les cartes de la gamme MKR possèdent le même nombre d'**E/S**. Elles sont fournies avec un total de **22 broches d'E/S numérique** dont **12 broches PWM**. Elles comprennent également **7 broches d'entrée analogique** et **1 broche de sortie analogique**.

Il existe plusieurs shields MKR, par exemple le blindage MKR Relay Proto Shield qui permet d'utiliser des relais et qui fournit de l'espace pour ajouter d'autres composants grâce à sa zone de prototypage.

Bien qu'elles ne soient pas compatibles avec les shields Arduino Uno, il est possible de connecter facilement des capteurs aux cartes MKR à l'aide de l'adaptateur Arduino MKR Connector Carrier (Grove compatible). [Schéma]



2. La carte Arduino MKR Wifi 1010

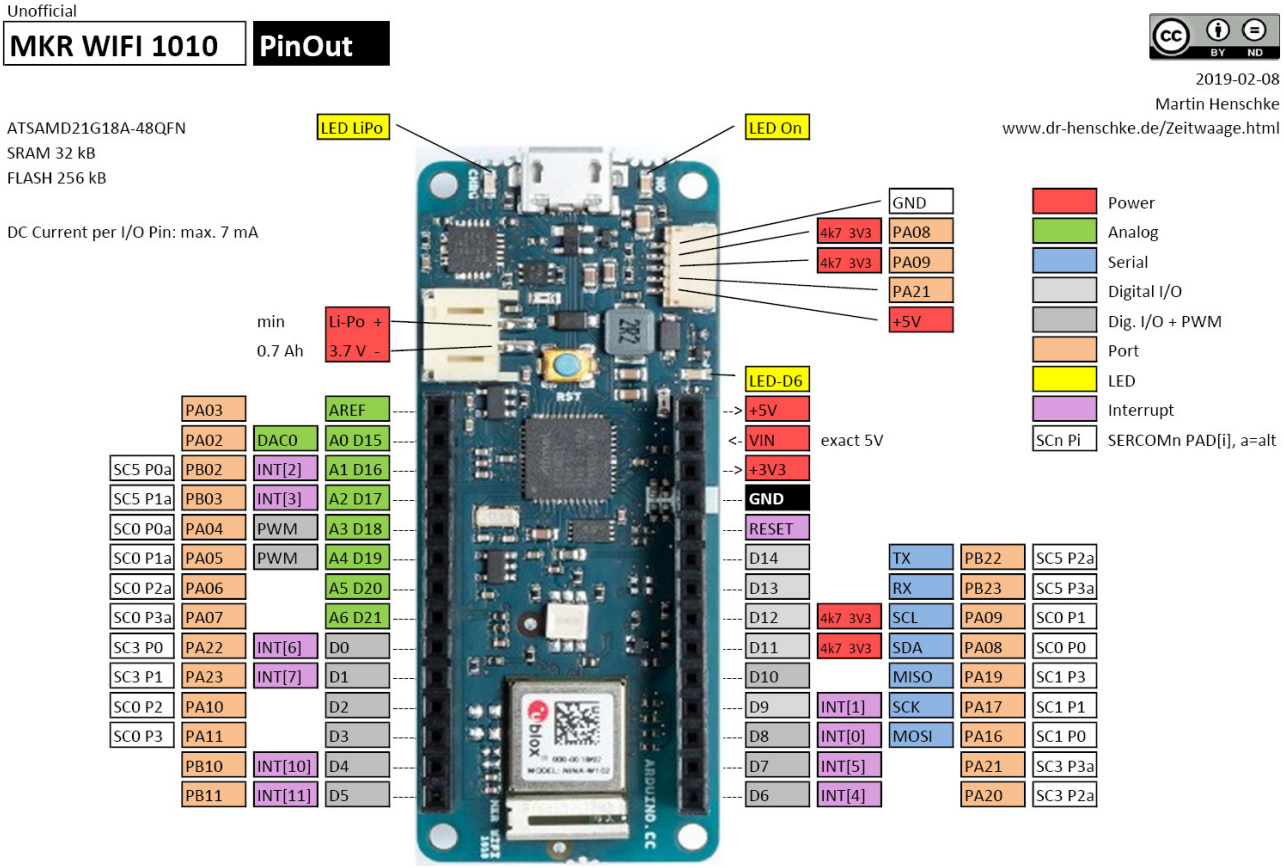
Le MKR WIFI 1010 est équipé d'un module **ESP32** fabriqué par U-BLOX. Cette carte a pour objectif d'accélérer et de simplifier le prototypage des applications IoT basées sur le WiFi grâce à la flexibilité du module ESP32 et à sa faible consommation d'énergie. La carte est composée de **trois principaux blocs** :

- Microcontrôleur **SAMD21** Cortex-M0+ 32bit Low Power ARM MCU (Datasheet)
- **U-BLOX NINA-W10** Series Low Power 2.4GHz IEEE® 802.11 b/g/n Wi-Fi (Datasheet)
- **ECC508 Crypto Authentication**. (Datasheet)

- **Principales caractéristiques**
  - **Alimentation** de la carte : **5V** (circuit sous **3,3V**)
  - **Batterie** supportée : Li-Po Single Cell, 3.7V, 700mAh Minimum

- **E/S numériques** : 8 (7mA)
- **PWM** : 2 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, A3 - or 18 -, A4 -or 19)
- **UART** : 1
- **SPI** : 1
- **I2C** : 1
- **I2S** : 1
- **Réseau** : Wifi (sécurisé à l'aide du cryptage SHA-256)
- **Entrées analogiques** : 7 (ADC 8/10/12 bit)
- **Sortie analogique** : 1 (DAC 10bits)
- **Interruptions externes** : 8 (0, 1, 4, 5, 6, 7, 8, A1 -or 16-, A2 - or 17)
- **Flash** : 256KB
- **SRAM** : 32KB
- **EEPROM** : Non
- **Fréquence d'horloge** : 48MHz, 32.768 kHz (RTC)

Contrairement à la plupart des cartes Arduino, le MKR WIFI 1010 fonctionne sous 3,3V. La tension maximale tolérée par les broches d'E/S est de 3,3V. **L'application de tensions supérieures à 3,3 V à n'importe quelle broche d'E/S peut endommager la carte**. Bien que la sortie sur des appareils numériques 5V soit possible, la communication bidirectionnelle avec des appareils 5V nécessite un décalage de niveau approprié. **Cette adaptation de niveau est réalisée par la carte Arduino MKR Connector Carrier** ci-dessus.



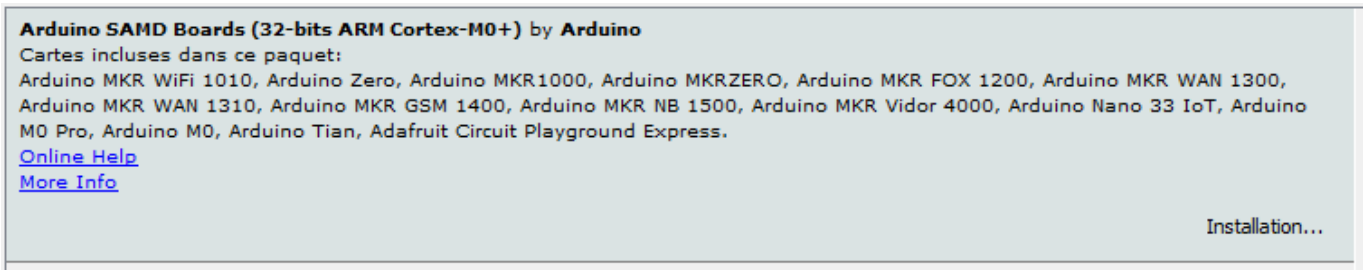
Pour afficher le diagramme de brochage complet au format PDF, cliquer ici ou le **schéma** de la carte MKR Wifi 1010, cliquer ici.

Le réseau Wi-Fi est à faible consommation d'énergie. Le port USB du MKR1010 peut être utilisé pour alimenter la carte sous 5V. Il possède un circuit de charge Li-Po qui permet à l'Arduino MKR WIFI 1010 de fonctionner sur batterie ou sur une source externe de 5 volts, chargeant la batterie Li-Po tout en utilisant une alimentation externe. Le passage d'une source à l'autre se fait automatiquement.

3. Préparation de l'IDE Arduino

3.1 Installation du support "Arduino SAMD Boards"

- Dans le menu « **Outils** » , « **Type de carte** » et « **Gestionnaire de carte** », rechercher « Arduino SAMD Board » et installer le composant.



3.2 Les bibliothèques

A partir du **gestionnaire de bibliothèque** de l'IDE Arduino, **télécharger** et **installer** :

3.2.1 WifiNINA

- **Sources** sur github
- **Documentation** sur arduino.cc

Active la **connexion réseau** (locale et Internet) des Arduino MKR Wifi 1010, Arduino MKR VIDOR 4000 et Arduino UNO Wifi Rev.2. Avec cette bibliothèque, vous pouvez instancier des serveurs, des clients et envoyer / recevoir des paquets UDP via le wifi. La carte peut se connecter à des réseaux ouverts ou cryptés (WEP, WPA). L'adresse IP peut être attribuée de manière statique ou via un serveur DHCP. La bibliothèque peut aussi gérer le DNS.

- **Mise à jour du firmware du module WIFI NINA**
  1. Mettre à jour la bibliothèque WiFiNINA avec le gestionnaire de bibliothèques.
  2. Vérifier la version du firmware installé en téléchargeant et en exécutant l'exemple ***CheckFirmwareVersion*** (→ Fichier → Exemples → WiFiNINA → Tools). Les informations apparaissent dans le moniteur série.

3. Télécharger et exécuter l'exemple ***FirmwareUpdater*** dans la carte (→ Fichier → Exemples → WiFiNINA → Tools) puis mettre à jour le firmware à l'aide de **WIFI101/WiFiNINA Firmware Updater** (→ Outils).

3.2.2 PubSubClient

- **Sources** et documentation de **Nick O'Leary** sur github

Cette bibliothèque fournit un client pour faire de simples messages de publication / abonnement avec un serveur prenant en charge MQTT.

3.2.3 WifiWebServer

- **Sources** sur github

WifiWebServer est une bibliothèque serveur simple mais complète pour les cartes AVR, Teensy, SAM DUE, **Arduino SAMD21**, Adafruit SAMD21/SAMD51, Adafruit nRF52, ESP32/ESP8266, STM32F/L/H/G/WB/MP1, etc., utilisant les modules/boucliers WiFi (WiFiNINA, WiFi101, U-Blox W101, W102, ESP8266/ESP32-AT, etc.).
Les fonctions sont similaires et compatibles à celles de ESP32 WebServer et des ESP8266WebServer.
A partir de la v1.1.0 cette bibliothèque fournit également un client HTTP et WebSocket de haut niveau dont les fonctions sont similaires et compatibles à celles de la bibliothèque ArduinoHttpClient.

4. Démarrer avec la carte Arduino MKR WiFi 1010

4.1 Blink un premier Programme pour dire "Hello"

*.cpp

```
void setup() {
  pinMode(LED_BUILTIN,OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Télécharger le projet PlatformIO pour VSCode.

4.2 Intégrer une nouvelle carte MKR1010 dans le réseau Wifi

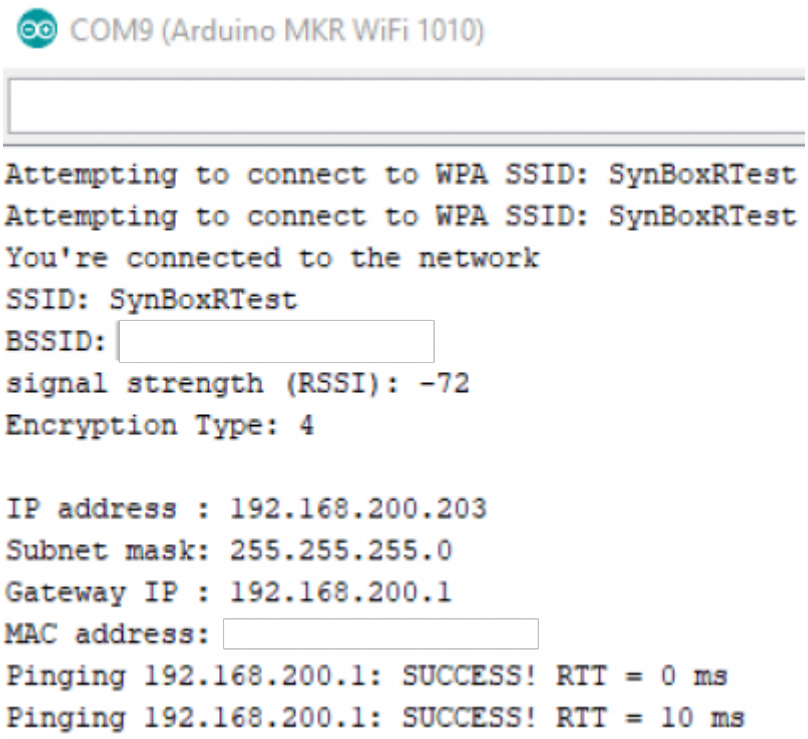
- Ouvrir l'exemple ***WiFiPing*** accessible à partir de (→ Fichier → Exemples → WiFiNINA) dans le logiciel Arduino.
- Compléter le **SSID** et le **mot de passe** du Wifi comme dans l'exemple ci-dessous.

*.cpp

```
// Modifications dans WiFiPing
// Specify IP address or hostname
String hostName = "IP de la box à renseigner";

// Modifications dans arduino_secrets.h
#define SECRET_SSID "à renseigner"
#define SECRET_PASS "à renseigner"
```

Résultat dans le moniteur série



- Programmer la carte.
- Identifier l'adresse **MAC** de la carte avec **Advanced Port Scanner**.
- Entrer l'**@MAC** et attribuer une **@IP** à la carte dans l'onglet "*Réservation DHCP*" de la box du réseau.
- Redémarrer la carte.

Télécharger le projet PlatformIO pour VSCode.

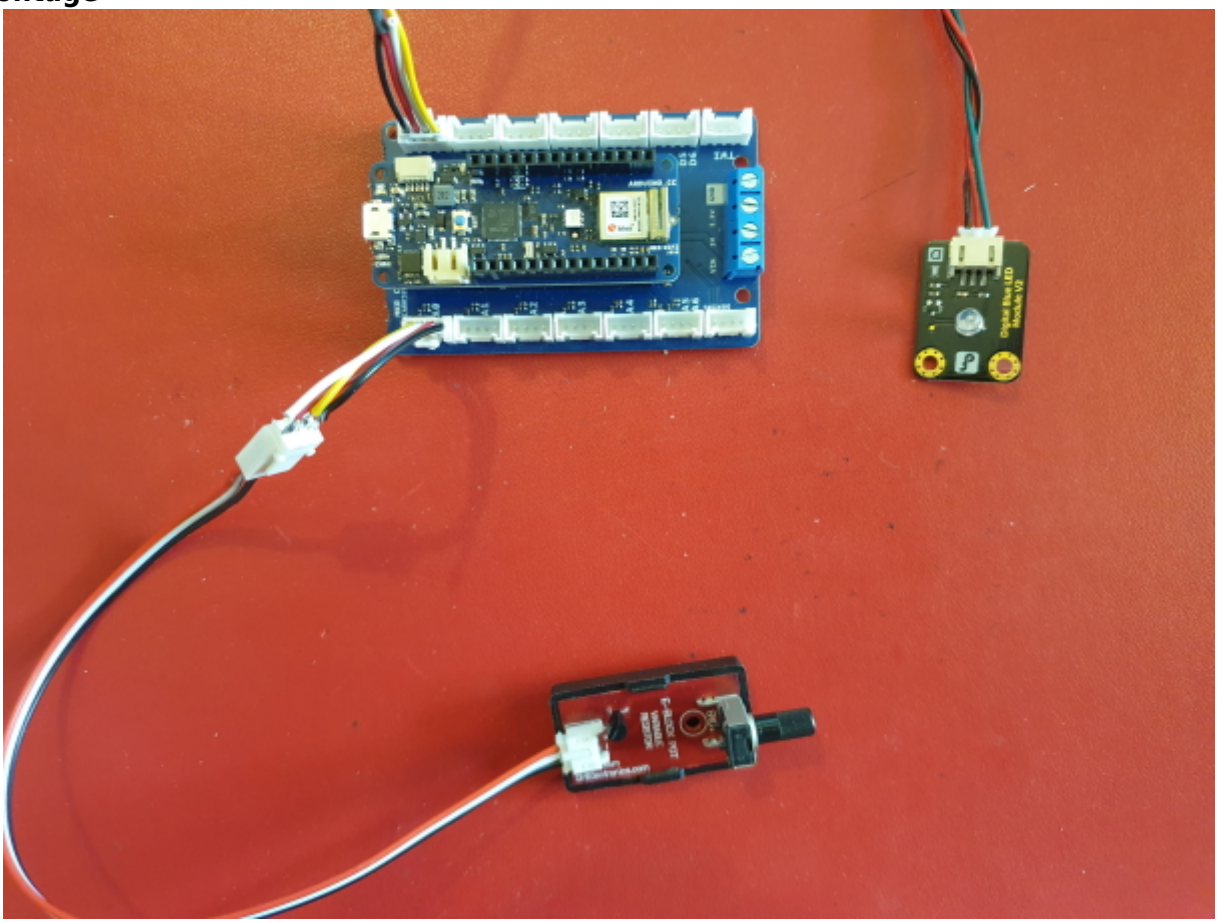
4.4 Serveurs HTTP

- **Ressource** : ESP32 HTTP GET and HTTP POST with Arduino IDE (JSON, URL Encoded, Text)

Ce paragraphe présente **deux** versions d'un serveur HTTP :
- La **Version 1** est une étude de cas pour le cours d'algorithmique
- La **dernière version** est utilisée dans les projets.

4.4.1 VERSION 1 : étude de cas

- **Source** : *SimpleWebServer* (→ Fichier → Exemples → WiFiNINA)
- **Montage**



- **Algorithme**

```
// Principe
// Le serveur lit les requêtes caractère par caractère et extrait les
données de l'url
// En réponse à la requête, il envoie une valeur ou/et déclenche une
commande
// Exemples
// pas de données dans la requête => envoie de la page d'accueil
// /arduino/digital/led/1 => activation d'une sortie
// /arduino/digital/led/0 => désactivation d'une sortie
// /arduino/analog/val1 => mesure et envoie d'une valeur issue du CAN
```

```
// Initialisation
Créer un serveur HTTP à l'écoute sur le port 80

1.  si le module wifi n'est pas détecté alors
        bloquer le programme
2.  tant que le module n'est pas connecté au wifi faire
        se connecter au wifi avec le SSID et le mot2passe
    fin tant que
3. Démarrer le serveur HTTP

// Programme
Répéter toujours
 début // 4. Traiter les requêtes
    Attente blocante d'un client
    si un client est connecté alors
     |  currentLine <- "" // mémorise la donnée transmise dans la requête
     |  tant que le client est connecté faire
     |    | si il reste des caractères à lire
     |    | alors
     |    |  |  lire le dernier caractère transmis
     |    |  |  si ce caractère est une fin de ligne
     |    |  |  alors
     |    |  |   | si currentLine = "" // on a eu 2 fin de lignes
consécutifs ! => pas de donnée transmise
     |    |  |   | alors
     |    |  |   |    Envoyer la page d'accueil au client et sortir de la
boucle tant que
     |    |  |   | sinon
     |    |  |   |    currentLine <- ""
     |    |  |   | fin si
     |    |  |  sinon
     |    |  |   | si le caractère lu n'est pas un retour chariot
     |    |  |   | alors
     |    |  |   |    on l'ajoute à la fin de currentLine
     |    |  |   | fin si
     |    |  |  fin si
     |    |  |  // Traitement les autres requête(s) (une url suit l'@IP)
     |    |  |  Traiter la requête 1 et sortir de la boucle tant que
     |    |  |  Traiter la requête 2 et sortir de la boucle tant que
     |    |  |  ...
     |    |  |  Traiter la requête n et sortir de la boucle tant que
     |    | fin si
     |  fin tant que
    fin si
    Fermer la connexion
 fin
```

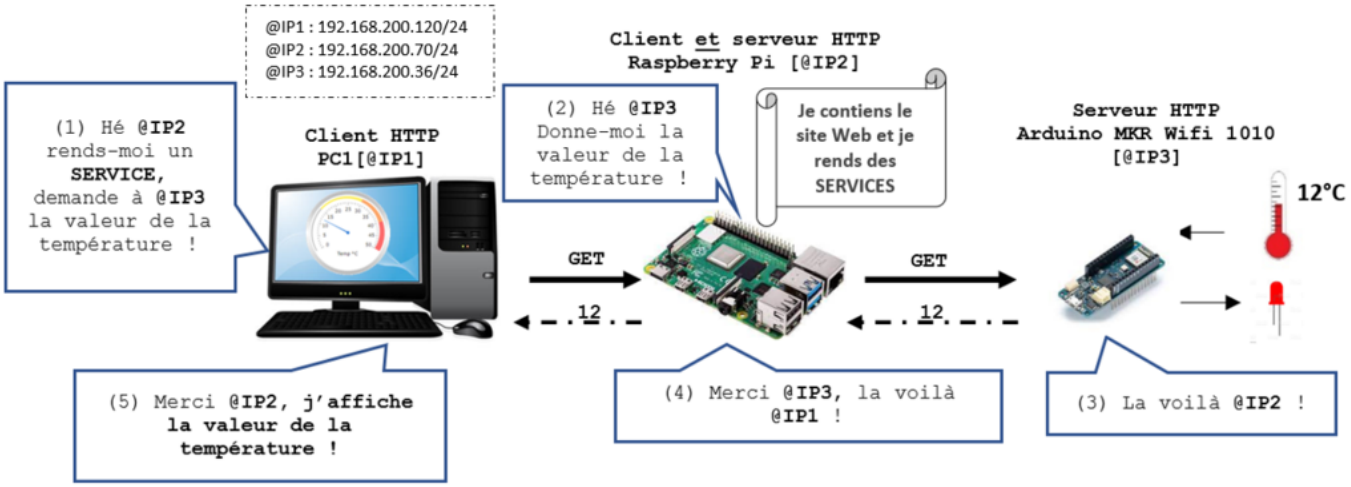
- **Code** (adaptation de l'exemple SimpleWebServer)

Télécharger le projet Arduino.

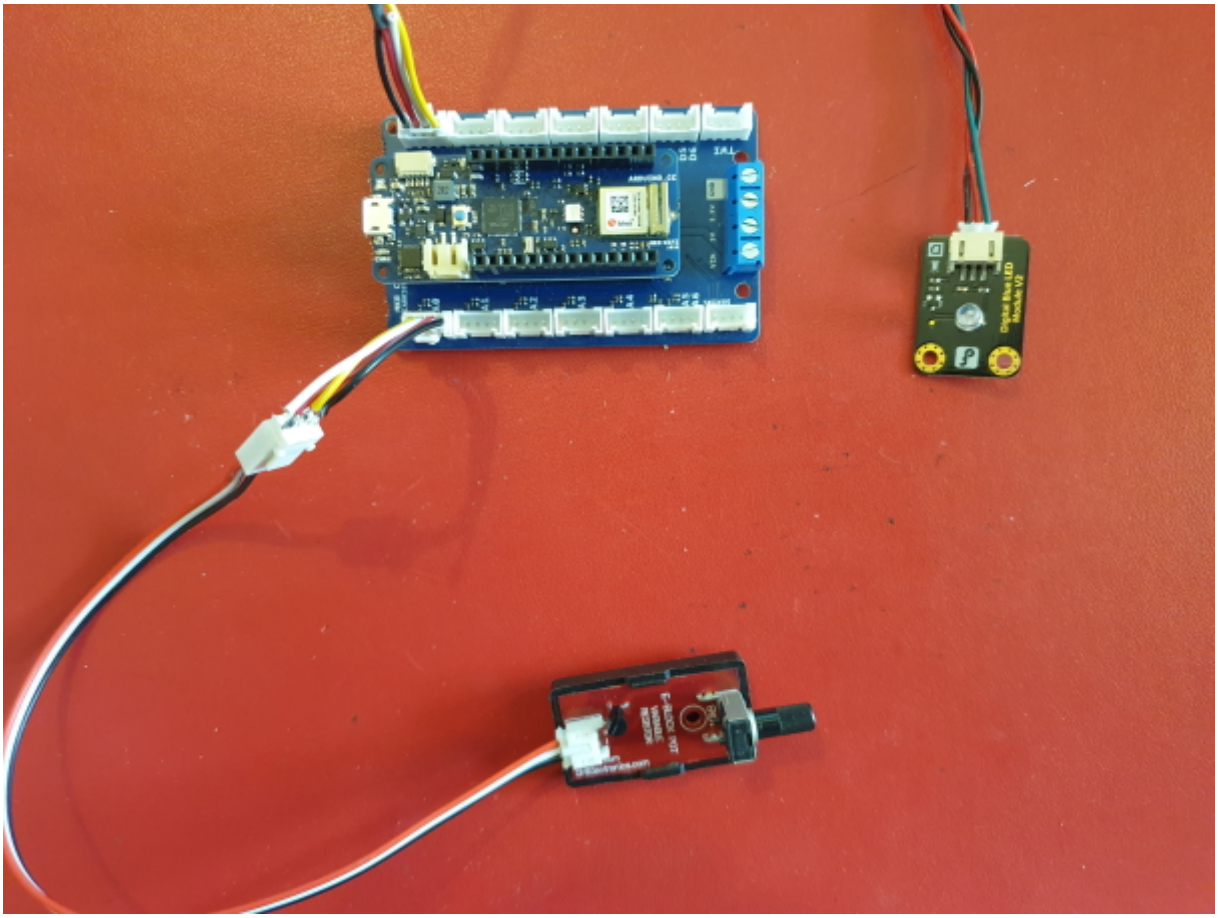
- **Tests**

4.4.2 VERSION utilisée dans les projets

- **Principe retenu**



- **Source** : *AdvancedWebServer* (→ Fichier → Exemples → WiFiWebServer)
- **Montage**



- **Algorithme**

```
// Serveur HTTP V4
// Le serveur répond à une requête en envoyant une ou des valeurs lues sur
ses entrées
// et/ou en déclenchant une commande sur ses sorties
// Exemples
// pas de données dans la requête => envoie de la page d'accueil
// /ecrire/valA?val=1 => activation de la sortie valA
// /ecrire/valA?val=0 => désactivation de la sortie valA
// /lire/val1 => mesure et envoie la valeur présente sur l'entrée val1
(issue du CAN)
// ------------------------------------------------------------------------
-----------
// Variables
   chaîne de caractères <- page HTML (accueil)
   tableaux de caractères <- SSID et mot2passe

// Liste des gestionnaires (fonctions) répondant à une requête
   g1. Envoie de la page d'accueil
   g2. Commande de la led L de la carte et de la sortie D0
   g3. Commande de la sortie D1
   g4. Mesure et transmission de la valeur analogique présente sur A0
   g5. Mesure et transmission de la valeur analogique présente sur A1
   g6. Mesure et transmission des valeurs analogiques présentes sur A0 et A1
au format JSON
   g7. Traitement des requêtes non prises en charge
   ...
```

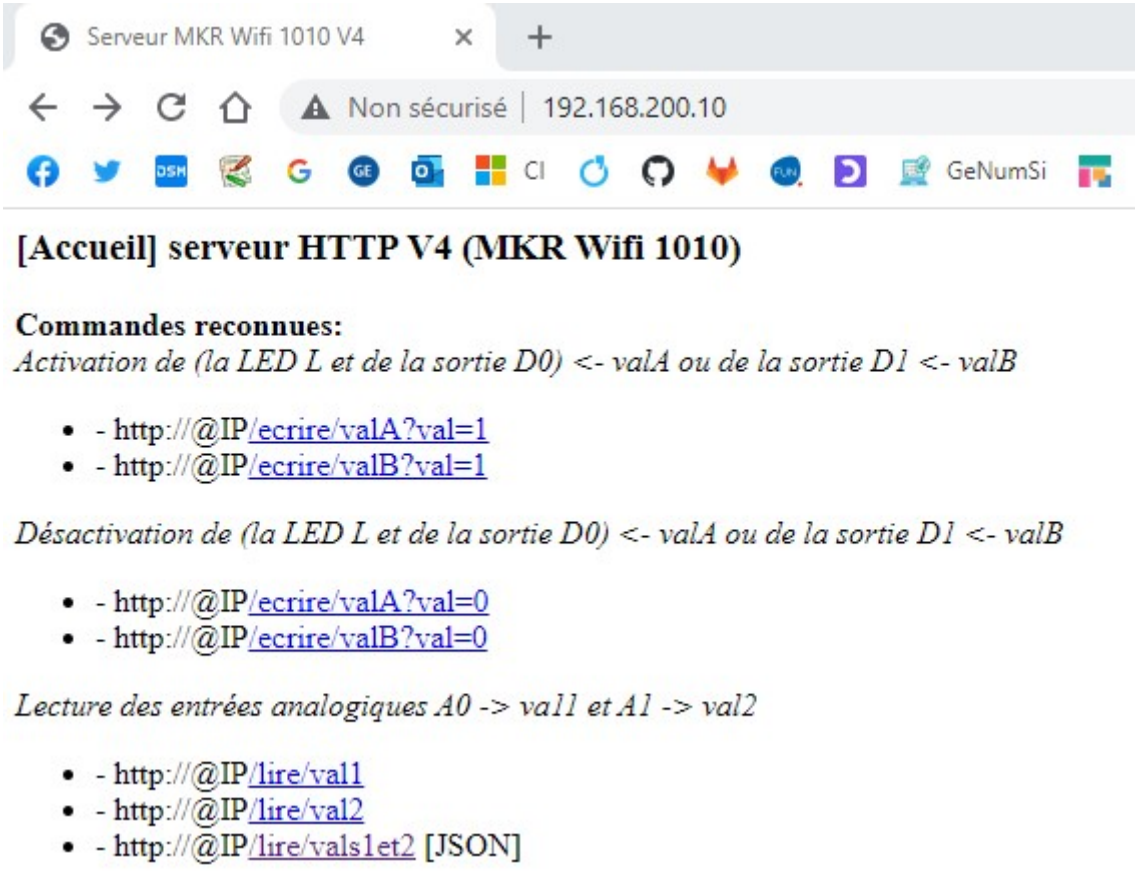
```
   gn ...

// Initialisation
   a. Configuration des E/S
   b. Si le module wifi n'est pas détecté, alors
        bloquer le programme !!!!
   c. tant que le module n'est pas connecté au réseau wifi
        faire
          se connecter au réseau Wifi avec le SSID et le mot2passe
      fin tant que
   d. Inscription les gestionnaires de requête
   e. Démarrage du serveur HTTP
Répéter toujours
   Traiter une requête
```

- **Code**


Télécharger le projet PlatformIO pour VSCode ou le projet Arduino.

- **Tests**



4.5 Client MQTT

- Voir la page "Mettre en oeuvre un client MQTT sur un EP8266 (ESP32) Feather Huzzah ou MKR Wifi 1010"

5. Tutoriels

Des liens vers des tutoriels sont accessibles sur la page webographie.