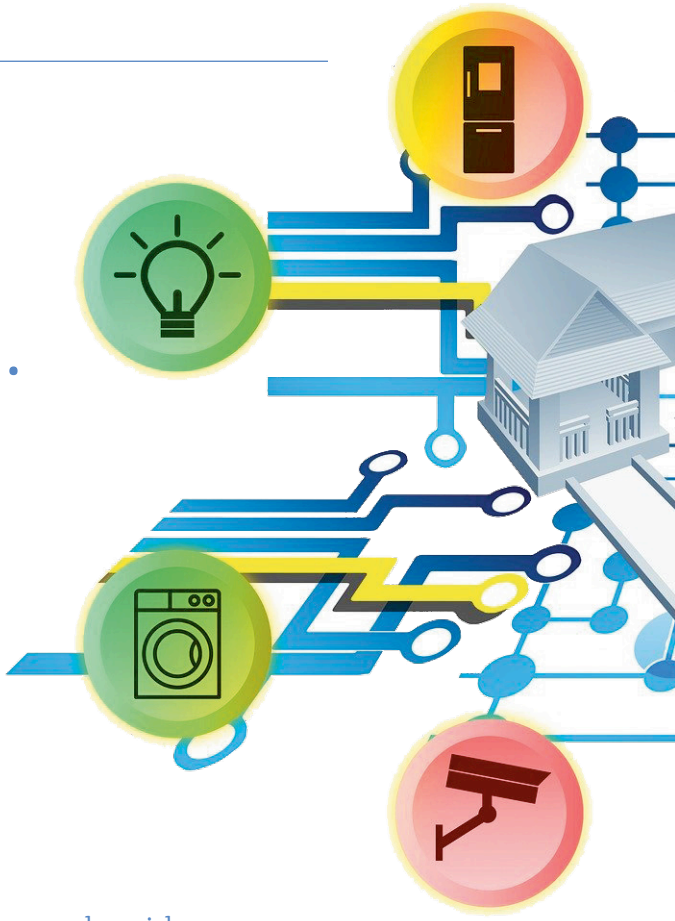
la domotique, c'est facile avec…

ESPHome, Home Assistant et MySensors



Clemens Valens (Elektor Labs)

Qui n'a jamais rêvé d'équiper sa maison avec des éclairages ou des rideaux télécommandés ou des volets roulants qui s'ouvrent ou se ferment automatiquement ? La domotique, voilà comment cela s'appelle !

J'ai fait un rêve

Des rêves, j'en ai fait, mais ça n'est pas allé plus loin. Dès qu'il s'agissait de passer à la réalisation, je me suis invariablement heurté à des obstacles variés et ma motivation a vite molli.

Un capteur de température sans fil sur piles pour mesurer la température du salon, c'est facile à faire, y ajouter un relais télécommandé pour commander un radiateur aussi. Mais après… il faut un contrôleur qui contrôle : *"En semaine, allumer le chauffage à 7 h le matin, mais en fin de semaine pas avant 9 h"*. Et il faut arriver à reprendre la main de telle sorte que tout occupant de la maison puisse, sans avoir suivi d'abord un cours de Python, régler la température à son goût. Tout ça, ce n'est pas encore de la domotique, c'était juste la description d'un thermostat programmable. Pour passer d'une lampe télécommandée à un véritable système domotique extensible, il en faut bien plus. Voilà pourquoi je me suis contenté de rêver.

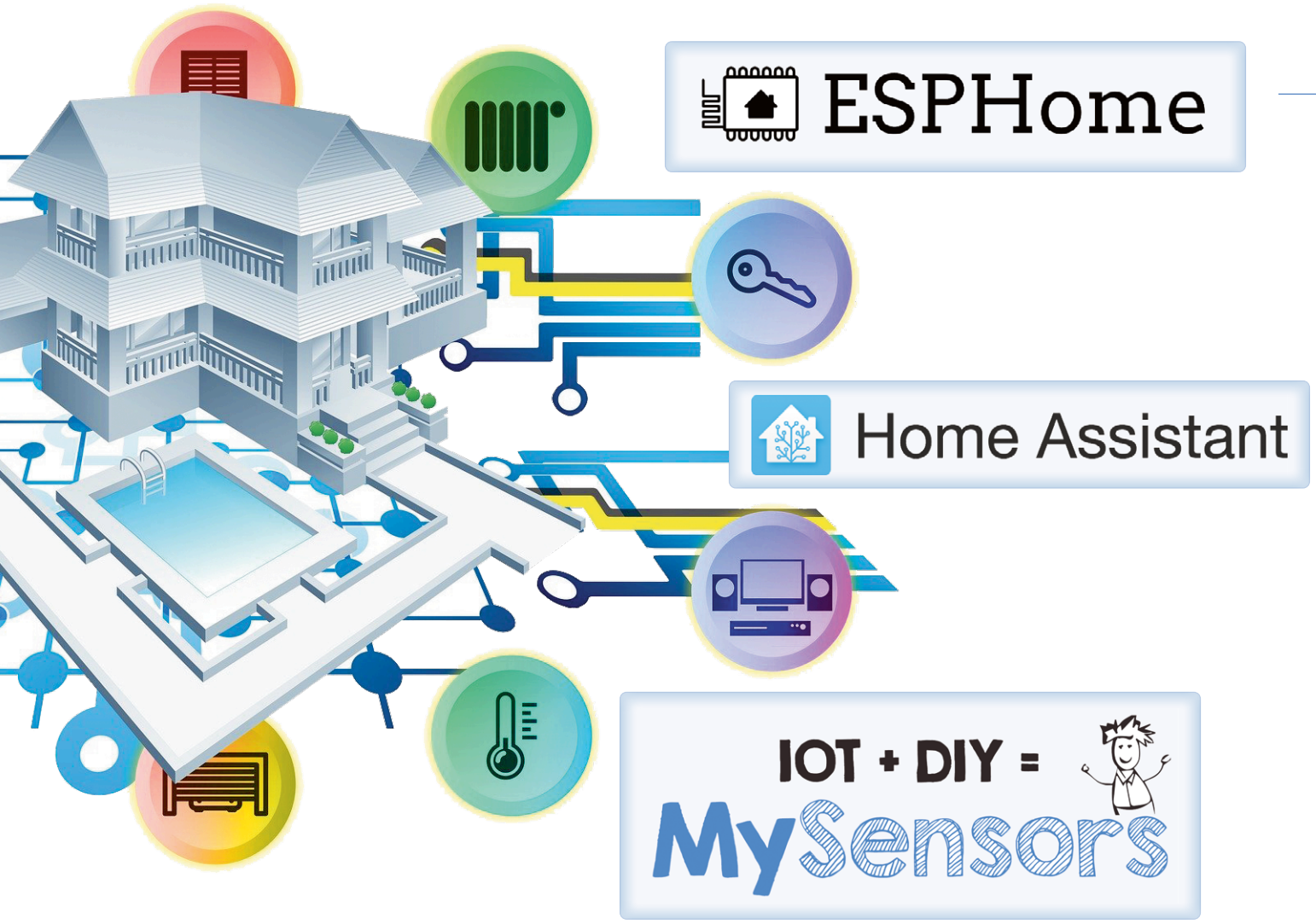
Espurna et ESPHome

Très récemment, en cherchant sur l'internet le manuel de l'utilisateur d'une prise de courant commandée par Wi-Fi achetée il y a quelques années, mais jamais utilisée, je suis tombé sur *Espurna* [1]. Même si finalement je ne m'en suis pas servi, je le mentionne, car cet excellent projet à logiciel ouvert mérite d'être connu. Avec *Espurna*, il est facile de créer des capteurs et des actionneurs Wi-Fi (surtout ESP8266) qui peuvent communiquer entre eux. *ESPHome* [2] (**fig. 1**) est un projet voisin d'*Espurna*.

Figure 1. La page ***ESPHome****, plutôt longue, mentionne tous les composants prêts à l'emploi utilisables par un appareil* ***ESPHome****.*

Et vous trouviez Arduino facile ?

Espurna et *ESPHome* fournissent tous deux un moyen simple de programmer les puces ESP8266 et ESP32 d'*Espressif*. En fait, c'est tellement simple qu'*Arduino* vous paraîtra une jungle inextricable. En un mot, une fois qu'on a installé le logiciel, il n'y a plus qu'à écrire un fichier texte qui précise quel type de capteur est relié à quelle(s) broche(s) du module ESP et, après compilation et flashage du micrologiciel, vous disposez d'un périphérique intelligent avec une interface web, la programmation



à distance (*Over-The-Air*, OTA), la communication MQTT et autres. C'est déjà impressionnant, mais ce n'est pas tout.

Commandez vos prises intelligentes sous Wi-Fi

L'internet déborde de prises intelligentes et pas chères ainsi que de cartes à relais avec interface Wi-Fi incluse ainsi que d'autres gadgets sous Wi-Fi. La plupart utilisent un ESP8266 (**fig. 2**) et ont en commun l'obligation de passer par un service dématérialisé, dans le nuage, c'est-à-dire quelque part sur l'internet. C'est de là que vient l'application de télécommande à installer sur votre portable. C'est généralement inconfortable, et personne ne s'en sert bien souvent. Avec *Espurna* et *ESPHome*, c'est différent, car ils offrent la possibilité de reprogrammer ces appareils avec un micrologiciel modifiable à volonté. Nul besoin de connexion internet, adieu les applications en anglais approximatif et les services douteux dans le nuage. Vous reprogrammez l'appareil comme bon vous semble et vous l'intégrez à votre système de domotique contrôlé par *moi@chezmoi.*

De la télécommande à l'automatisation

ESPHome et *Espurna* comportent des fonctions d'automatisation qui permettent à l'utilisateur de spécifier des règles de commutation en fonction d'événements ou de données de capteurs. Avec l'un ou l'autre, vous réaliserez facilement un système domotique élaboré à partir de composants du commerce à Wi-Fi intégré ou bien réalisés par vous-même.

Cet article se serait arrêté ici si je n'avais pas trouvé exagérément compliquée la domotique avec *Espurna* ou *ESPHome*. Mes capacités intellectuelles sont limitées, certes, mais la documentation de ces projets est médiocre. Copieuse peut-être, mais surtout disparate et plutôt obscure.

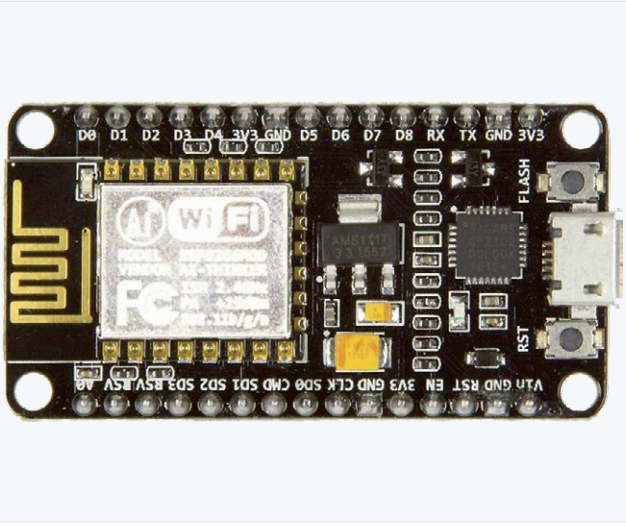


Figure 2. ***ESPHome*** *tourne sur les modules basés ESP8266 tels que* ***NodeMCU****, qui est répandu et peu coûteux. On peut aussi utiliser des modules ESP32.*

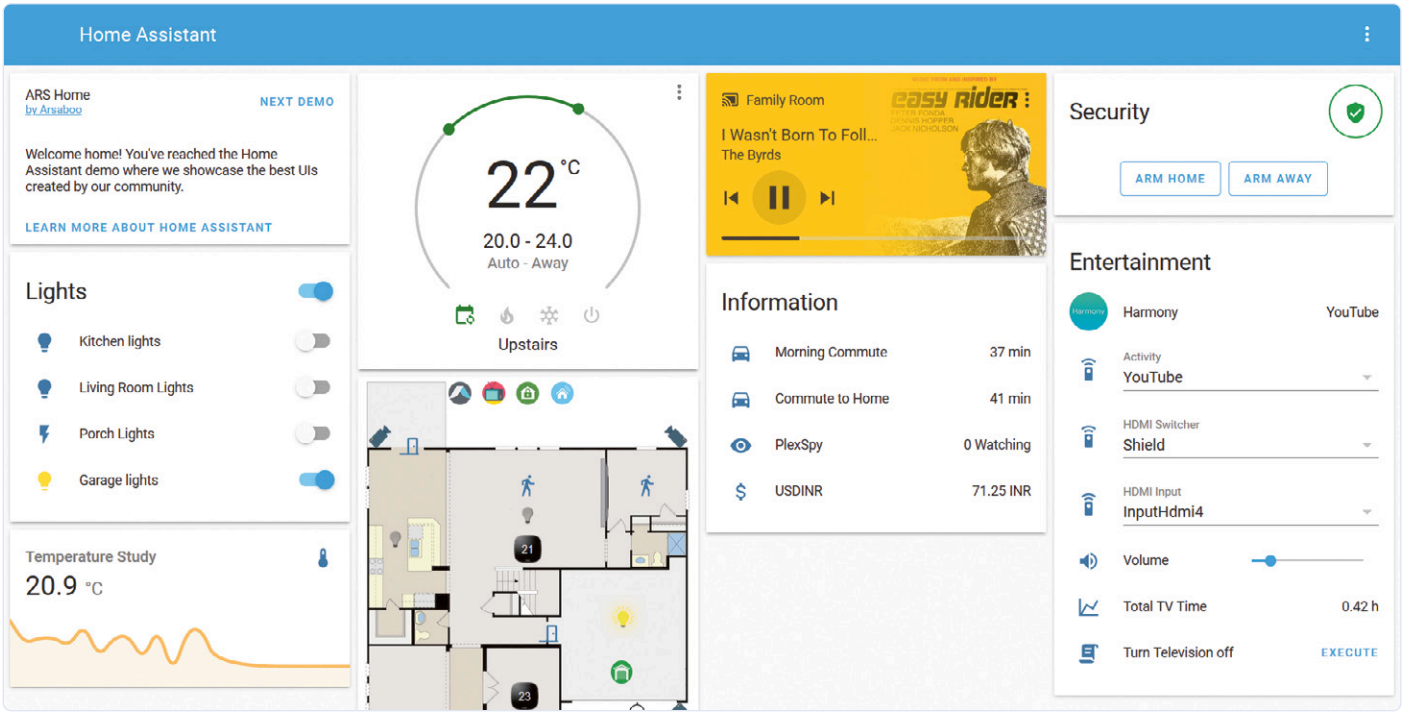


Figure 3. Le tableau de bord de Home Assistant est entièrement personnalisable. Selon votre goût, il sera touffu ou dépouillé.

L'assistant domestique

ESPHome et *Espurna* proposent tous deux une intégration à *Home Assistant* [3], ce qu'on appelle un concentrateur d'automatisation ou *contrôleur*. C'est un appareil qui permet à l'utilisateur de combiner des capteurs et des actionneurs (de fabricants différents) en un système unique. Ici, les mots *capteurs* et *actionneurs* doivent être pris dans un sens très général. Ça va du thermomètre, au GPS ou aux services internet en passant par la commande de moteurs et l'envoi de textos. *Home Assistant* (aussi 'HA', 'Hass' ou 'Hass.io') est compatible avec – pour ne citer que les plus connus - *Alexa* et *OK Google*, le *Trådfri* d'*Ikea*, le *Hue* de *Philips*, la domotique de *Z-Wave* et de *Zigbee*, et *Sonoff* de chez *iTead* [4]. Au moment où j'écris, HA en est à 1574 intégrations et va bien au-delà de ce que proposent *Espurna* et *ESPHome*.

Vous utilisez HA pour définir les règles de fonctionnement des appareils à l'intérieur et autour de votre maison. « *Si le soleil se couche dans vingt minutes et si le portable de l'occupant A est présent dans la maison et s'il n'a pas plu les trois dernières semaines, alors ouvrir le troisième gicleur à partir de la droite* ». Voilà une règle typique (et même ordinaire) de HA, en supposant que le matériel en question soit installé et fonctionne comme prévu. HA propose aussi une interface graphique utilisateur (GUI) entièrement personnalisable (**fig. 3**). HA est un logiciel ouvert, en *Python*, sur *Raspberry Pi*, mais qui s'accommode aussi d'autres systèmes d'exploitation. Si je l'ai installé sur un RPi 3, c'est parce que c'était tellement facile. Oui, mon truc, c'est la facilité.

L'intégration dans *Home Assistant*

C'est aussi pour *Home Assistant* que j'ai continué avec *ESPHome* plutôt qu'avec *Espurna*. Pour *ESPHome* il y a un greffon (**fig. 4**) qui l'intègre à HA de telle manière que la détection des appareils basés *ESPHome* soit automatique. « Flashez et jouez ! » Que demander de plus ? L'intégration est telle que vous n'avez même plus besoin d'ordinateur : du fond de votre canapé, vous (re)programmez et (re)configurez vos capteurs et vos actionneurs sur votre portable.

Internet des objets à faible consommation

La solution Wi-Fi, c'est bien pour connecter rapidement des appareils à un réseau, mais gare à la consommation. Elle n'est donc pas la meilleure pour des nœuds alimentés par de l'énergie glanée ou une pile bouton censée tenir des années. Ces appareils sont dormants la plupart du temps et, lorsqu'ils se réveillent, ils expulsent leurs données aussi vite que possible, faute d'énergie suffisante pour de verbeux échanges protocolaires.

MySensors [5] est excellent pour ce genre d'appareils. Ce projet libre de domotique et d'IdO basé sur la bande Radio ISM, en particulier le nRF24 de *Nordic Semiconductor* (**fig. 5**) et le RFM69 de *HopeRF*. On peut utiliser aussi la plateforme nRF5, plus récente, comme sur le *micro:bit* de BBC. Le site de *MySensors* est un peu fouillis, mais une fois familiarisé, vous y découvrirez des choses intéressantes.

MySensors utilise surtout (mais pas seulement) *Arduino* comme plateforme microcontrôleur et il gère lui-même un réseau arborescent (ou en étoile). Comme *ESPHome*, il s'intègre en (pas tout à fait la même) douceur à *Home Assistant*.

Le projet se présente sous la forme d'une bibliothèque *Arduino* incluse dans le gestionnaire de bibliothèques *Arduino*. Une fois installée, vous pouvez créer votre application à partir de l'un des exemples. Très souvent, il suffit de changer les numéros de broches des périphériques connectés.

Les mains dans le cambouis

Fin du préambule, passons à la pratique. Je suggère d'installer *Home Assistant* sur un *Raspberry Pi*. Un bon guide pas-à-pas est disponible [3] (onglet GET STARTED). Il est suggéré d'utiliser un RPi 4,

mais chez moi, ça marche sur un RPi 3. Une carte microSD d'au moins 32 Go est recommandée, mais il faut se rappeler que le RPi ne sait pas gérer les cartes SDXC formatées en exFAT (c'est-à-dire de plus de 32 Go), donc, si vous voulez utiliser une carte de 64 Go ou plus, n'oubliez pas de la reformater en FAT32. Pour améliorer la fiabilité du système, on peut utiliser un disque statique (SSD) au lieu d'une carte microSD fragile.

DNS multipoint

Home Assistant s'appuie sur le protocole DNS multipoint (mDNS) pour découvrir les appareils et communiquer avec eux, mais ce protocole n'est pas très bien supporté par *Android* et *Windows* (cf. encadré). Les appareils *Apple* et le RPi fonctionnent bien et je suppose que c'est le cas d'autres versions de *Linux*. C'est pour cette raison qu'il est bon d'attribuer une adresse IP statique au HA. Mon système HA utilise DHCP et j'ai observé des problèmes de connexion avec l'application HA pour *Android* quand le serveur DHCP attribuait une nouvelle adresse IP à l'ordinateur HA.
Notez que l'installateur HA démarre très tôt un serveur web qui permet de suivre la progression de l'installation ; connecter un écran au RPi n'est guère utile ici. Pour obtenir l'adresse IP de ce serveur, consultez votre routeur de réseau.

Accès au fichier de configuration

Le pas suivant de l'installation consiste à configurer *Home Assistant*. Il y a un assistant pour cela, je ne m'y attarde pas. Intéressons-nous plutôt à l'installation de quelques extensions après la configuration de HA. On l'effectue à partir de l'onglet *Add-on Store* sous le menu *Supervisor*. Pour une raison obscure, il n'est pas possible d'éditer le fichier de configuration principal de HA (*configuration.yaml*) , alors qu'on y est fréquemment obligé, en particulier quand on expérimente avec le système. C'est pourquoi j'ai installé les extensions *File Editor* (précédemment nommée *Configurator*) et *Samba share*. La première permet d'éditer des fichiers de bas niveau directement dans HA, la seconde d'accéder à certains répertoires HA sur le réseau, ce qui vous permet d'éditer des fichiers avec votre éditeur de texte favori. Pour des raisons de sécurité, vous préférerez peut-être installer l'extension *Terminal & SSH*.
Samba vous permet de transformer HA en un serveur de fichiers si vous créez un répertoire 'www' dans le répertoire 'config'. L'installation d'extensions est simple : cliquer sur la fiche de l'extension puis sur *Install*. Une fiche ouverte donne les instructions de configuration de l'extension.

Installation de l'extension *ESPHome*

Cela nous amène à *ESPHome* parce qu'il nous faut également installer l'extension *ESPHome*. Pour ce faire, commencez par ajouter le lien de son dépôt au store. Ensuite, trouvez le champ *Add new repository by URL* (= Ajoutez l'URL d'un nouveau dépôt) et collez-y l'URL suivant :
`https://github.com/esphome/hassio`
Vous pouvez maintenant installer l'extension *ESPHome*. Il y en a trois versions : ordinaire (= *plain*), beta et dev. Nous utiliserons l'ordinaire. La seule configuration à effectuer pour cette version est l'activation des options *Démarrer à l'initialisation* (= *Start on boot*), *Mise à jour automatique* (= *Auto update*) et *Afficher dans la barre latérale* (= *Show in sidebar*). Cette dernière facilite l'accès à l'extension.

Intégration de *MySensors*

Au moment où j'écris (été 2020), il n'existe pas de *Home Assistant* pour *MySensors*. Pour activer cette intégration il faut ajouter quelques lignes au fichier *configuration.yaml* de HA (voir l'encadré) :

```
mysensors:
  gateways:
  - device: '192.168.1.100'
    persistence_file: 'mysensors/Wi-Fi_gateway.json'
    tcp_port: 5003
    optimistic: false
    persistence: true
    retain: true
    version: '2.0'
```

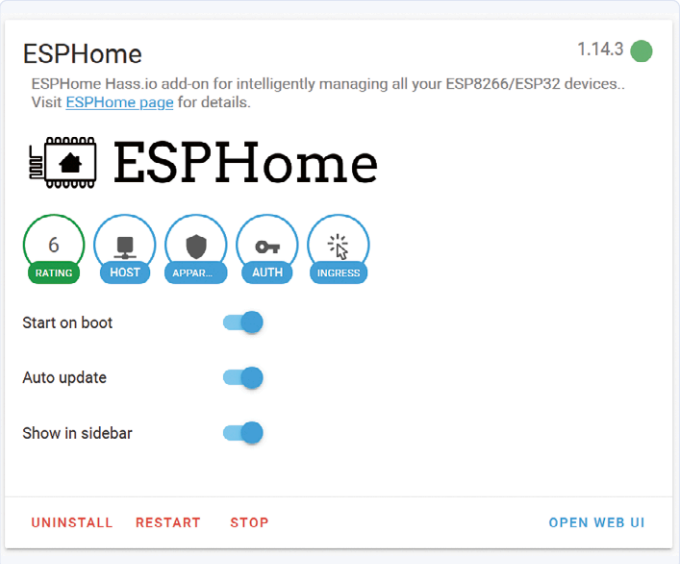


*Figure 4. J'aime bien l'option 'Afficher dans la barre latérale' activée pour l'extension **ESPHome** dans le **Home Assistant**.*

Figure 5. Les modules nRF24 se présentent sous différentes formes, dont la plus commune a 8 broches ; celle à 10 broches n'en diffère que par le brochage.

COMMENT ACTIVER **MDNS** SOUS WINDOWS

Dans une configuration de réseau DHCP (le réseau attribue des adresses IP aux nœuds connectés), les utilisateurs de *Windows* peuvent se heurter à des difficultés pour se connecter à HA parce qu'ils ne disposent pas du protocole DNS multipoint (mDNS) actif. La disponibilité de ce protocole permet de se connecter à HA en utilisant le lien http://hassio.local:8123 (sous [3] on trouve d'autres liens, mais ils n'ont pas fonctionné pour moi). Voici le moyen que j'ai trouvé de régler ce problème :

1. Ouvrez l'éditeur du registre (touches 'Windows'+R, puis tapez 'regedit') et trouvez la clé **Ordinateur\HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\DNSClient**
   Cliquez dessus avec le bouton droit de la souris et choisissez *New* pour créer un DWORD 32-bit nommé *EnableMulticast* et de valeur 0 (zéro).
2. Depuis le site d'*Apple*, téléchargez *iTunes*, mais ne l'installez pas. Ça demande un peu de recherche. La dernière fois où je l'ai fait, j'ai choisi *iTunes for Windows*, mais pas celui du *Microsoft Store*, mais en continuant à dérouler le menu jusqu'à tomber sur *Looking for other versions?*. Ce lien-là vous conduira au fichier désiré. Une fois le téléchargement terminé, changez le type du fichier de .EXE en .ZIP puis extrayez-en le fichier *bonjour.msi* (ou *bonjour64.msi*, selon la version d'*iTunes* téléchargée).
3. Installez *bonjour* et redémarrez votre ordinateur.

https://www.apple.com/itunes/

UN MOT À PROPOS DE L'INDENTATION DE **YAML**

Home Assistant et *ESPHome* utilisent tous deux YAML (*Yet Another Markup Language*) pour leurs fichiers de configuration. Comme *Python* et d'anciens langages assembleur, sa structure est liée à l'indentation (une horreur, si vous voulez mon avis). L'indentation doit être la même pour toutes les lignes de même niveau, mais peut différer d'un niveau à l'autre. Pour les fragments de YAML dans cet article, chaque niveau d'indentation est de deux (2) espaces.

Comme déjà mentionné, il faut utiliser une adresse IP statique pour la passerelle *MySensors* que vous allez réaliser (si, si, vous allez le faire, voyez ci-dessous) et vous devez choisir un nom et un emplacement pour le fichier JSON où HA pourra stocker l'information réseau *MySensors*.
Important : Chaque fois que le fichier *configuration.yaml* de HA est modifié, le système doit être redémarré pour prendre en compte ces modifications. Ceci peut s'effectuer à partir de l'onglet *Système* du Superviseur. Tant qu'à éditer le fichier de configuration, profitez-en pour ajouter les lignes suivantes (ça vous épargnera un redémarrage) :

```
binary_sensor:
  - platform: workday
    country: [country code]
```

Ces lignes permettent d'écrire des règles d'automatisation actives seulement les jours de semaine ou pendant les week-ends, ce genre de choses. Remplacez **[country code]** par le code du pays dans lequel travaille votre système. Pour trouver ce code (et d'autres informations utiles), reportez-vous à la page d'aide du *Workday Binary Sensor*.

Mon premier appareil *ESPHome*

Si vous avez suivi les instructions ci-dessus, vous êtes prêt à vous mettre à la programmation d'appareils à base d'ESP8266 et d'ESP32. Un nouveau système n'est pas immédiatement compatible avec *ESPHome*, il devra donc être programmé initialement par le port série. Selon l'appareil, il vous faudra peut-être installer un pilote USB-vers-Série sur l'ordinateur HA pour que ça marche. Les cartes *NodeMCU* équipées d'une puce USB CP2102 de *Silabs* (*Silicon Laboratories*) que j'ai utilisées ont fonctionné aussitôt déballées.
Après avoir connecté l'appareil à l'ordinateur hôte de HA, ouvrez le tableau de bord d'*ESPHome*, soit à partir de la barre latérale (si vous avez activé cette option) soit en cliquant sur *Open Web UI* sur la fiche des extensions dans le tableau de bord du superviseur. Vérifiez que le port série est disponible dans la liste déroulante de la fiche de l'extension dans le coin supérieur droit, *OTA (Over-The-Air)* par défaut. Si ce n'est pas le cas, redémarrez l'extension *ESPHome* (en retournant à la fiche *ESPHome* sur le panneau de contrôle du superviseur) ; là, ça devrait marcher.
Ensuite, cliquez sur le bouton rosâtre '+' sur le panneau de commande de l'*ESPHome*, ce qui ouvre un assistant qui va vous guider à travers la première partie. J'avoue n'avoir utilisé aucun mot de passe pour aucun appareil, ce qui témoigne peut-être de mon irresponsabilité. Pour le téléchargement, indiquez le port série auquel l'appareil est connecté.

Édition du fichier YAML

L'assistant a fait son travail et une fiche devrait maintenant exister pour votre appareil. À ce moment, le bouton *Edit* ne fonctionne pas encore (chez moi) ; il s'active quand on recharge la page. Cliquez dessus, vérifiez vos paramètres d'accès au réseau Wi-Fi et assurez-vous de la présence des lignes 'api:' et 'ota:'. Si vous téléchargez cette configuration vers votre appareil, il sera détecté par *Home Assistant* après son redémarrage. On peut maintenant déconnecter l'appareil de l'ordinateur, car la programmation sans fil (*Over-The-Air*) a été activée. Toutefois, l'appareil ne fera rien, car vous n'avez pas encore configuré ses périphériques. Alors, continuez votre lecture avant de télécharger une configuration.
Si vous utilisez un module *NodeMCU*, vous pouvez le démarrer en ajoutant les lignes ci-dessous à la fin du fichier de configuration (en dessous de 'ota:' juste par souci de clarté, car leur position dans le fichier n'a pas d'importance) avant de le télécharger. Cela va donner à HA l'accès à la LED embarquée et au bouton *Flash*.

```
output:
   - platform: gpio
     id: «led»
     pin:
       number: GPIO16
       inverted: True

 light:
   - platform: binary
     name: «LED»
     output: «led»

 binary_sensor:
   - platform: gpio
     name: «Flash pushbutton»
     pin:
       number: GPIO0
       inverted: True
```

Notez que dans ce fragment de code l'indentation de tous les niveaux est de deux (2) espaces, c'est-à-dire que les lignes '**number: GPIOx**' et '**inverted: True**' commencent avec six espaces (car elles sont au troisième niveau d'indentation, voir l'encadré).
Cette configuration peut aussi fonctionner avec un module ESP-01 en remplaçant GPIO16 par GPIO3 et en connectant une LED en série avec une résistance de 470 Ω (environ) entre la broche GPIO3 (RXD) et la masse. Il faut aussi connecter un bouton-poussoir entre la broche GPIO0 et la masse et une résistance de rappel de 10 kΩ (environ) entre GPIO0 et le 3,3 V. N'appuyez pas sur ce bouton au démarrage de l'appareil.
Téléchargez cette configuration vers l'appareil *ESPHome* et attendez qu'il redémarre. *Home Assistant* devrait le voir – le panneau de commande de *ESPHome* devrait indiquer *En service* et, sans intervention de votre part, créer un affichage pour le voyant et un indicateur pour le bouton-poussoir.
Si vous commencez par télécharger la configuration vide, il vous faudra sans doute le chercher dans la liste *Devices* du menu de configuration de HA.
Vous pouvez maintenant ajouter des fiches pour les commandes dans la vue d'ensemble *Overview* de HA (cliquez sur les trois points puis sur *Configure UI*) et créer des automates dans l'éditeur d'automates (*Automations*) du menu de configuration. Tout cela étant à peu près évident, je vous laisse faire sans plus d'explications ; c'est aussi une bonne occasion de farfouiller dans le *Home Assistant*. De plus, *MySensors* nous attend.

Réalisation d'une passerelle Wi-Fi *MySensors*

Pour réaliser un réseau *MySensors*, une passerelle est nécessaire. Vous en aurez aussi besoin pour vous connecter à d'autres réseaux Wi-Fi. Pour cela, j'ai pris un *NodeMCU* basé ESP8266 parce qu'il possède un port SPI dont nous allons avoir besoin. Une autre possibilité est d'utiliser un module ESP32. En fait, il y a de nombreuses possibilités, mais dans cet article, nous nous occupons de Wi-Fi. Connectez un module nRF24L01+ au port SPI (**fig. 6**). Il est recommandé d'ajouter un condensateur électrolytique (4,7 à 47 µF) en parallèle avec un condensateur céramique (de 100 nF environ) entre les broches VCC et masse du module nRF24. C'est tout ce dont vous aurez besoin.
Côté logiciel, il vous faut un ordinateur avec l'EDI *Arduino* installé. Ajoutez à l'EDI le noyau ESP8266 (ou ESP32) pour *Arduino* – vous trouverez tous les détails sous [6] – et la bibliothèque *MySensors*

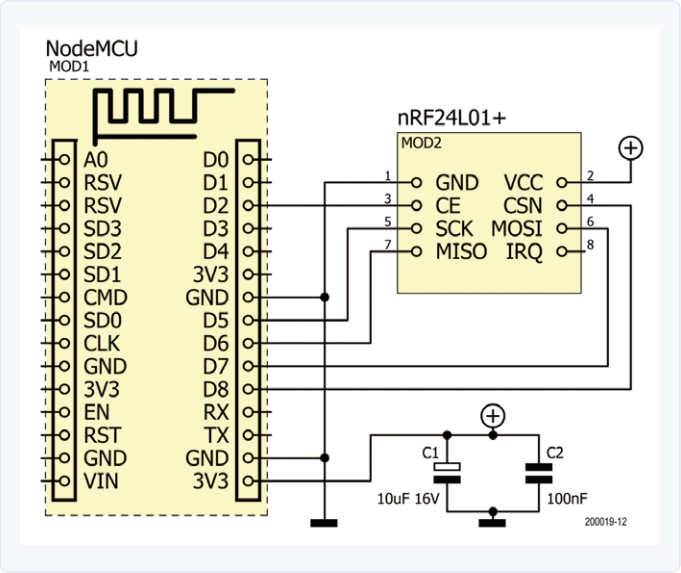


*Figure 6. Une passerelle Wi-Fi **MYSensors** peut être réalisée en utilisant un module **NodeMCU**.*

(Croquis > Inclure une bibliothèque > Gérer les bibliothèques...). Chargez l'exemple passerelle (Fichier > Exemples > MySensors) et saisissez le SSID, le mot de passe et l'adresse IP statique que vous avez indiquée plus tôt dans le fichier *configuration.yaml* de HA. Téléchargez le croquis vers votre appareil et voilà, votre passerelle est prête.
N'oubliez pas qu'une passerelle (ou des nœuds répétiteurs, non traités dans cet article) doivent toujours être alimentés et ne peuvent être mis en sommeil. Les nœuds de capteurs, par contre, peuvent faire ce qu'ils veulent.

Mon premier nœud *MySensors*

La création d'un nœud *MySensors* ressemble à la réalisation d'une passerelle, sauf que le module ESP est remplacé par une carte compatible *Arduino* équipée de capteurs et d'actionneurs. Connectez un module nRF24L01+ au port SPI de la carte et ajoutez les condensateurs mentionnés plus haut (**fig. 7**).
Téléchargez un croquis de la bibliothèque d'exemples de *MySensors*, de préférence un qui ressemble à ce que vous voulez réaliser. Notez qu'il y a d'autres exemples de croquis sur le site web de *MySensors*. Le croquis doit être édité pour se conformer à vos périphériques et numéros de broches, mais c'est à peu près tout ce qu'il y a à faire. Téléchargez le croquis vers votre appareil et laissez-le redémarrer ; il va automatiquement rejoindre le réseau MySensors. Oui, vraiment.

Attention à la version du protocole

Arrivé là, ça a coincé : Home Assistant ne reconnaissait pas mon appareil, un relais distant. J'ai fini par découvrir que c'était en rapport avec la version de l'interface de programmation (API) utilisée. En inspectant le fichier JSON de persistance (voir la section 'Intégration de *MySensors*'), j'ai remarqué l'indication que la passerelle utilisait l'API ou le protocole version 2.3.2. Ma page *MySensors* [7] présente un exemple avec une indentation spécifique à l'utilisation de l'API version 2.x. Quand je l'ai essayé, mon nœud est apparu dans la liste *Entities* du menu de configuration de HA et j'ai pu créer des fiches UI pour lui. Selon [7], pour qu'un nœud basé V2.x fonctionne sous HA, il

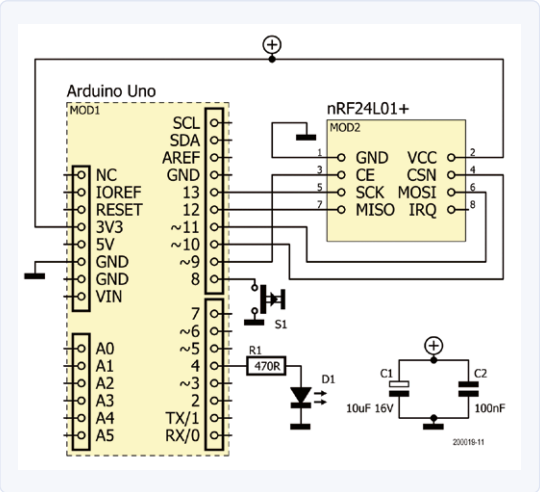


*Figure 7. Schéma d'un nœud **MySensors** avec un bouton-poussoir pour un capteur binaire et une LED pour un actionneur.*

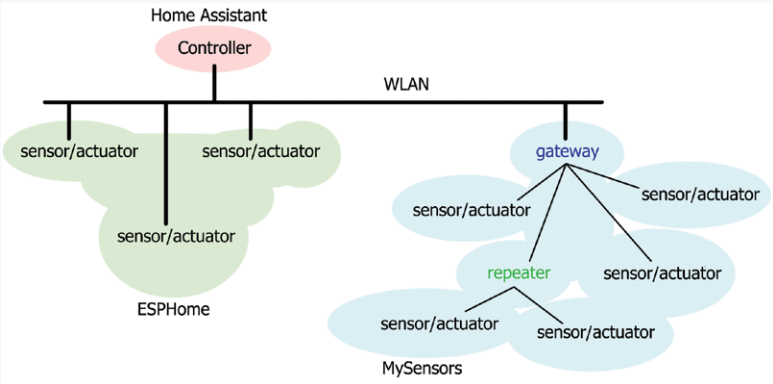


Figure 8. Si vous combinez tous les éléments décrits dans cet article, vous aurez réalisé un système domotique tel que celui-ci.

doit envoyer une valeur initiale depuis la fonction **loop()**, mais seulement pendant la phase de démarrage. Les croquis exemples ne le font pas. Mais en approfondissant, j'ai conjecturé qu'un nœud ***actionneur*** doit requérir de HA (et peut-être aussi lui envoyer) une valeur initiale pour être reconnu. Un nœud ***capteur*** est repéré aussitôt qu'il envoie des données. La demande n'a pas besoin de provenir de la fonction **loop()**, elle doit simplement être effectuée à un moment quelconque de la phase de démarrage.

Fonctions spéciales : *before()* et *presentation()*

Pour distinguer les croquis V2.x de types plus anciens, cherchez la fonction **presentation()**. Si le croquis en contient une, c'est un V2 ou plus récent. L'inverse n'est toutefois pas vrai, car un croquis récent peut ne pas la contenir. Les instructions **present** habituellement contenues dans cette fonction ne peuvent pas être omises et doivent être déplacées vers une autre fonction, par exemple **setup()**.

Les croquis *MySensors* peuvent comporter une fonction **before()**. Les deux fonctions **before()** et **presentation()** sont appelées avant **setup()** et dans cet ordre, c'est-à-dire **before(), presentation(), setup()**, et enfin **loop()**.

Adapter du code existant à votre matériel est devenu très facile. Il y a beaucoup d'exemples pour toutes sortes de capteurs dans la bibliothèque *Arduino* aussi bien que sur l'internet, prenez le temps de chercher.

J'espère vous avoir intéressé

Voici la fin de mon topo sur la domotique facile (**fig. 8**). Il est incomplet et on trouvera sans doute plus simple ou mieux. Je suis encore en phase d'apprentissage moi-même.

Il existe beaucoup de projets de domotique qui font des choses similaires, probablement aussi de meilleurs ou de plus beaux ; tous ont des points forts et des faiblesses.

Comme ils sont innombrables, le choix est difficile. J'en ai présenté ici quelques-uns que j'ai testés puis gardés pour réaliser mon système domotique. Je les utilise encore et leurs possibilités m'émerveillent par leur variété. Je n'ai pas d'ailleurs l'intention de m'en tenir là.

Sur le site Elektor Labs, vous trouverez des exemples de configurations pour *ESPHome* [8] et des composants *MySensors* [9], et quelques vidéos.

200019-02

LIENS

[1] **Espurna** : https://github.com/xoseperez/espurna

[2] **ESPHome** : https://esphome.io/

[3] **Home Assistant** : www.home-assistant.io/

[4] **Sonoff** : www.itead.cc/

[5] **MySensors** : https://www.mysensors.org/

[6] **ESP8266 core pour Arduino** : https://github.com/esp8266/Arduino

[7] **MySensors in Home Assistant** : www.home-assistant.io/integrations/mysensors

[8] **ESPHome sur le site Elektor Labs** : www.elektormagazine.com/labs/how-to-home-assistant-esphome

[9] **MySensors sur le site Elektor Labs** : www.elektormagazine.com/labs/mysensors-home-assistant-howto