FONCTIONS DOMOTIQUE FABLAB (V1.6~~5~~)

OBJECTIFS DE CE DOCUMENT

Ce document décrit les fonctions de la domotique que le FabLab souhaite implémenter, et la façon dont elles l'ont été.

On verra en détail :

- Détection d'occupation du FabLab,
- Suivi de la consommation électrique globale,
- Suivi de la consommation électrique du chauffage,
- Gestion du chauffage,
- Gestion de l'extraction des fumées des découpes laser,
- Gestion de la VMC,
- Gestion du chauffe-eau,
- Autorisation d'accès aux imprimantes 3D ,
- Envoi et réception de SMS,
- Sécurisation des locaux,
- Gestion de l'éclairage.

Ce document sera complété aux fur et à mesure avec les différents éléments (comme les scripts), les avancées sur les différents points à étudier et les réalisations.

Vous trouverez l'état détaillé d'avancement des différents projets dans le fichier « Installation domotique FabLab »

TABLE DES MATIÈRES

MATÉRIEL COMMANDÉ POUR LES FONCTIONS

En plus des éléments nécessaires au serveur domotique, le matériel suivant a été commandé par le FabLab :

- Un module TIC vers ZigBee 3.0 LINKY + Antenne externe (LIXEE),
- 12 capteurs de température/hydrométrie (Sonoff SNZB-02P),
- 7 sorties de câble file pilote (Legrand 064879),
- 4 prises commandées 240V avec mesure conso (Frient 20201200),
- Une commande d'extraction des fumées (Tongou 25A avec mesure),
- Une commande de la VMC (Tongou 25A avec mesure),
- Une commande du chauffe eau (Tongou 25A avec mesure),
- Un module de gestion de SMS (DIY à base d'ESP8266, modem GSM GA6 et alim 12V),
- 4 modules lecteurs de badges (DIY à base d'ESP8266 et lecteur de badge RC522)

De plus, un module de mesure de consommation électrique ZigBee a été découvert dans le tableau électrique. Il a été installé sur le disjoncteur du chauffage et a eu le bon goût de s'appairer seul sur notre réseau ZigBee. On va donc l'utiliser.

CONVENTIONS

Les conventions suivantes sont utilisées dans ce document :

```
Les zones à saisir sont indiquées de cette façon.
```

En général, on doit les indiquer dans une fenêtre de type « Terminal ».

Les noms de touches sont spécifiées en majuscules entre crochets : par exemple [ENTER].

Les touches de modification sont spécifiées avec leur abréviation suivi d'un tiret et de la lettre à saisir. Par exemple [CTRL-C] ou[ALT-SHIFT-Z].

Les actions à réaliser et les textes variables à insérer sont indiqués comme <<connecter la clef USB>> ou <<mettre ici l'adresse IP du serveur>>.

DÉTECTION D'OCCUPATION DU FABLAB

De nombreuses fonctions se sont actives que pendant la présence de membres dans le FabLab.

Dans un premier temps, on va installer une commande arrêt/marche pour indiquer une présence dans le FabLab à l'entrée. On forcera l'arrêt à 20h30, au cas où on aurait oublié de le couper en sortant. Cette commande étant temporaire, je prêterais un module au FabLab.

Elle pourrait être remplacée par un capteur pris dans le circuit de verrouillage électrique de la porte du FabLab, réalisé par exemple avec un relai de la tension de la ventouse, dont les contacts seraient connectés à un détecteur d'ouverture ou un détecteur d'inondation ZigBee.

On a tenté, sans succès, d'utiliser un contact d'ouverture de porte, qui utilise un ILS (interrupteur à lame souple) activé par un aimant, pour tenter de capter le champs magnétique de la ventouse. L'épaisseur de la porte et sa structure en aluminium sont probablement parmi les raisons de cet échec.

L'ouverture du coffret gérant la ventouse (dans le local radio, à coté de la baie informatique au plafond) montre un relai très accessible. Un test rapide a montré que la porte était verrouillée lorsque le relai était collé. On va donc retester avec un capteur d'ouverture de porte qui détectera la présence de champs magnétique du relai (indiquant donc qu'il est fermé, comme la porte). Si le test est probant (coffret ouvert, on dessoudera l'ILS et on ajoutera des fils afin de pouvoir installer l'ILS contre le relai avec un collier rilsan, et positionner le boitier à l'extérieur du coffret métallique, qui doit être une excellente cage de Faraday, donc peu enclin à laisser passer les trames ZigBee de notre contacteur ;-)

Les détecteurs de présence pourraient également être utilisés pour détecter une absence de verrouillage de la porte lorsqu'aucune présence ne sera détectée après un certain temps.

Le contacteur prêté est un modèle ZigBee. Il sera appairé avec la procédure standard décrite dans la partie « Astuces diverses », puis renommé.

CONDITIONS D'ACTIVATION DES SCRIPTS ET ACTIONS GÉNÉRÉES

Un grand nombre de scripts a besoin de savoir si le FabLab est occupé.

On va donc créer plusieurs indicateurs « virtuels » gérés comme suit :

- L'indicateur « FabLab accessible » indique si la porte est déverrouillée (ou pas). La valeur « On » signifie qu'on peut accéder au FabLab, « Off » l'inverse.

  Dans un premier temps, c'est le bouton « On/Off » prêté qui réalisera cette fonction. Dès que la détection d'activation de la ventouse de la porte sera en place, elle remplacera le bouton prêté.

- L'indicateur « FabLab occupé » indique qu'on a détecté quelqu'un dans le FabLab (ou pas). La valeur « On » signifie qu'on détecte une présence, « Off » l'inverse.

  Dans un premier temps, c'est le bouton « On/Off » prêté qui réalisera cette fonction. détection d'activation de la ventouse de la porte sera en place, elle remplacera le bouton prêté. Par la suite, lorsque que la détection de présence (par infrarouge ou radar) sera en place, elle remplacera l'activation de la ventouse.

La façon de créer un dispositif virtuel est indiqué dans la partie « Astuces diverses » de ce document.

Ici, il faut 2 interrupteurs virtuels nommées « FabLab accessible » et « FabLab occupé ».

La façon de créer un script « LUA classique » est décrite dans la partie « Astuces diverses » de ce document.

SCRIPT « TRACE CHANGEMENT »

Avant d'écrire le premier script, il semble habile d'en écrire un qui va tracer les changements apportés à l'ensemble des dispositifs Domoticz. Il permettra, à partir du log de Domoticz, de voir ce qui se passe en temps réel. Ce script contient :

```
-- Trace changement

function string.starts(String,Start)
   return string.sub(String,1,string.len(Start))==Start
end

commandArray = {}

-- loop through all the changed devices
for deviceName,deviceValue in pairs(devicechanged) do
    -- Ne pas afficher les changements trop fréquents
    if string.starts(deviceName, "Mesure globale chauffage") or string.starts(deviceName, "Consommation électrique globale") then
    else
               -- Ecrire les modifications dans le log de Domoticz
        print ("Dispositif '"..deviceName.."', modifié en '"..tostring(deviceValue).."'");
    end
end

return commandArray
```

Enregistrer le code sous le nom « Trace changement »

SCRIPT « FABLAB ACCESSIBLE »

Le script suivant est utilisé pour implémenter le modèle ci-dessus :

```
-- FabLab accessible
--
--
-- Gère l'indicateur "FabLab accessible":
--
commandArray = {}

-- loop through all the changed devices
for deviceName,deviceValue in pairs(devicechanged) do
    if (deviceName=="Autorisation ouverture porte") then
        if deviceValue == "On" then
            commandArray['Fablab accessible'] = 'Off'
        else
            commandArray['Fablab accessible'] = 'On'
        end
    end
end

return commandArray
```

Il faut l'enregistrer sous le nom « FabLab accessible »

SCRIPT « FABLAB OCCUPÉ »

```
-- FabLab occupé
--
--
-- Gère l'indicateur "FabLab occupé":
--  Pour le moment, on recopie l'état de "Bouton porte"
--
commandArray = {}

-- loop through all the changed devices
```

```
for deviceName,deviceValue in pairs(devicechanged) do
    if (deviceName=="Bouton à l'entrée (Action)_on") then
        commandArray['Fablab occupé'] = 'On'
    end
    if (deviceName=="Bouton à l'entrée (Action)_off") then
        commandArray['Fablab occupé'] = 'Off'
    end
end

return commandArray
```

Il faut l'enregistrer sous le nom « FabLab occupé»

Il faut maintenant les tester. On affiche l'onglet « Interrupteurs », et on vérifie si l'état de « FabLab accessible » et « FabLab occupé » suit bien les actions sur « Bouton Porte ».

PLANNING DE REMISE À ZÉRO DU BOUTON LA NUIT

Reste que dans l'état actuel des choses, il est fort possible que le dernier utilisateur du FabLab oublie d'appuyer sur le bouton en sortant. On va corriger cet oubli en simulant l'appui sur « Off » toutes les heures de 21h à 5h du matin. Si un utilisateur est effectivement présent, il n'aura qu'à ré-appuyer sur « On » toutes les heures. Cette astuce permet de garantir une fermeture effective du FabLab, même si un utilisateur reste tard, et qu'il oublie de désactiver le bouton en sortant à 2h du matin.

Pour ça, plutôt qu'un script, on va simplement associer un planning au bouton placé à l'entrée.

L'exemple de création d'un planning dans la partie « Astuces diverses » détaille les étapes de cette opération.

SUIVI DE LA CONSOMMATION ÉLECTRIQUE GLOBALE

Le suivi de la consommation globale est réalisé par un module ZigBee directement connecté au compteur Linky, sur des broches prévues à cet effet. C'est le compteur qui fournit l'alimentation sans qu'il n'y ait besoin de paramétrer quoi que ce soit.

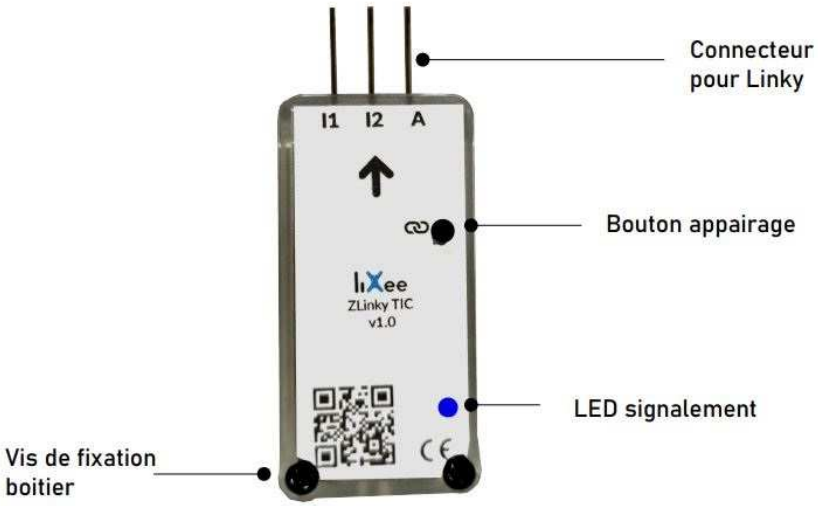
Le module choisi est un modèle LIXEE ZigBee. Le compteur étant à l'extérieur du bâtiment, une version avec antenne a été choisie. Sa description se trouve (entre autres) à https://www.domadoo.fr/fr/domotique/6397-lixee-module-tic-vers-zigbee-30-pour-compteur-linky-antenne-ext.html

La documentation du produit se trouve à https://www.domadoo.fr/fr/index.php?controller=attachment&id_attachment=3098. Une copie a été placée dans le répertoire qui contient ce document, sous le nom manuel-utilisation-lixee-tic-zlinky-zigbee.pdf.

!!! ATTENTION : DANGER DE MORT PAR ÉLECTROCUTION !!!

Le Linky est en permanence sous tension (le disjoncteur est connecté aval du compteur).
Prendre toutes les précautions lors de l'enfichage du module sur le compteur,
notamment en s'isolant électriquement de façon réglementaire.
L'appel à un électricien professionnel est indispensable.

Le module ressemble à ça :



Une fois le module inséré par l'électricien et l'antenne fixée, vérifier que le serveur Zigbee2MQTT est bien actif, avec l'appairage autorisé.

Vérifier que l'intégration des nouveaux dispositifs dans Domoticz est bien interdit.

Demander à l'électricien d'appuyer sur le bouton « appairage ».

L'appairage est terminé lorsque la LED s'allume de façon permanente.

Aller dans l'interface Web de Zigbee2MQTT, et donner un nom au module (par exemple, « Conso électrique globale ».

Vérifier que les données sont bien remontées par le compteur dans l'interface ZigBee2MQTT,

Activez l'intégration des nouveaux dispositifs dans Domoticz

Relancer le matériel « Auto-découverte MQTT » puis désactivez l'intégration dans Domoticz.

Intégrer le dispositif dans Domoticz et vérifier que le compteur est bien mis à jour.

SERVEUR ECOSTRUXURE

Le tableau général électrique du FabLab a été équipé par l'université d'un module ZigBee de mesure de consommation «EcoStruxure Panel Server Universal » de marque Schneider Electric.

D'après la doc (copiée dans le répertoire « Schneider Electric » qui contient ce document), le module serveur est accessible en WiFi et peut également communiquer en ZigBee. Coté ZigBee, .il se comporte en coordinateur, ce qui n'est pas d'une grande aide dans notre cas. On ne pourra donc pas l'utiliser.

Par contre, un module de mesure de consommation électrique ZigBee a été découvert dans le tableau électrique. Il a été installé sur le disjoncteur du chauffage et a eu le bon goût de s'appairer seul sur notre réseau ZigBee. On va donc l'utiliser pour suivre la consommation globale du chauffage.

CONDITIONS D'ACTIVATION DES SCRIPTS ET ACTIONS GÉNÉRÉES

SCRIPT DE SUIVI DE CONSOMMATION EDF ET DU TARIF

```
-- Consommation EDF
--
--  Charge le dispositif "Consommation EDF" à partir des données Linky
--

commandArray = {}

for deviceName, deviceValue in pairs(devicechanged) do
    -- Est-ce un des dispositifs liés à la consommation ?
    if deviceName == "Consommation électrique globale (PAPP)" or
        deviceName == "Consommation électrique globale (HCHP)" or
        deviceName == "Consommation électrique globale (HCHC)" then
        consoActuelle = otherdevices["Consommation électrique globale (PAPP)"]
        consoHP = otherdevices["Consommation électrique globale (HCHP)"]
        consoHC = otherdevices["Consommation électrique globale (HCHC)"]
        idx = tostring("Consommation EDF")
        -- Définition barbare de la valeur du dispositif contenant la consommation
        commandArray[#commandArray + 1] =
                {['UpdateDevice']=
idx..'|'..tostring(consoHP)..';'..tostring(consoHC)..';0;0;'..tostring(consoActuelle)}
    end
    -- Est-ce le tarif actuel ?
    if deviceName == "Consommation électrique globale (PTEC)" then
        if deviceValue == "HC.." then
            commandArray["Tarif réduit EDF"] = "On"
        else
            commandArray["Tarif réduit EDF"] = "Off"
        end
    end
end

return commandArray
```

A enregistrer sous le nom « Consommation EDF »

SUIVI DE LA CONSOMMATION ÉLECTRIQUE DU CHAUFFAGE

MODULES LEGRAND 064879

Le FabLab a acquis 7 modules Legrand 064879, qui sont des sorties de câble de type « Fil pilote » pour radiateur électrique.

Leur utilisation est prévue dans les pièces suivantes :

- Salle commune : 3 radiateurs de 2 kW
- Salle mécanique : 2 radiateurs de 2 kW
- Salle aéromodélisme : 2 radiateurs de 2 kW
- Salle électronique : 1 radiateur
- Salle radio : 1 radiateur
- Salle laser/impression 3D : 1 radiateur
- WC : 1 radiateur

En plus de la commande du fil pilote sur 6 ordres (éteint, hors-gel, confort, confort – 1°C, confort -2°C, éco), ils ont la capacité de déconnecter électriquement la sortie vers le radiateur, et de mesure la consommation électrique.

On a déjà déterminé que les fils pilotes de chaque radiateur sont remontés au tableau général. On pourra dans tous les cas centraliser les commandes de ce point.

Après vérification sur le schéma électrique, il y a un disjoncteur par pièce qui protège les radiateurs.

L'idée d'origine était de connecter les radiateurs d'une même pièce sur le module de commande de fil pilote, pour récupérer la consommation électrique. Le souci est que les modules Legrand sont limités à 3000 W, et que les radiateurs (à l'exception de celui des WC) font 2000 W.

Pour les 4 pièces qui n'ont qu'un seul radiateur, le montage du module se fera directement au niveau du boîtier de cloison, et que la gestion de la consommation sera possible sans adaptation.

Pour la salle commune et les salles mécanique et aéromodélisme on connectera la sortie de câble sur un des radiateurs, et on reliera ensemble les commandes de fil pilote de la pièce dans le tableau général. De cette façon la commande du fil pilote du radiateur « principal » sera renvoyé vers les autres.

On pourra plus tard ajouter des sorties de câbles si on souhaite avoir une idée plus précise des consommations.

Il faut ensuite intégrer les modules dans Zigbee2MQTT et dans Domoticz (les sélecteurs de commande du mode de fonctionnement seulement) et vérifier que les commandes sont bien reçues et comprises par les radiateurs.

MODULE SCHNEIDER ELECTRIC

L'ouverture physique du tableau électrique a montré un module ZigBee ide mesure du courant installé sur le disjoncteur général du chauffage. La bonne nouvelle est que ce module s'est appairé automatiquement avec notre réseau, et remonte la consommation électrique globale du chauffage.

On va l'utiliser pour mesurer la consommation globale du chauffage.

Il faut le renommer, l'intégrer dans Domoticz et vérifier qu'il remonte bien les données.Gestion du chauffage

La gestion du chauffage peut être réalisée de plusieurs façons.

La partie commande se base sur les boîtiers de commande par fil pilote Legrand 064879, qui sont capables de gérer 5 températures différentes (en plus de l'arrêt des radiateurs).

On utilise également des capteurs de température (et hydrométrie) répartis comme suit :

- Salle commune : 3 capteurs
- Salle mécanique : 2 capteurs
- Salle aéromodélisme : 2 capteurs
- Salle électronique : 1 capteur
- Salle radio : 1 capteur
- Salle laser/impression 3D : 1 capteur
- WC : 1 capteur

On positionnera également un capteur à l'extérieur, à l'abri des intempéries et du soleil.

Les capteurs à l'intérieur doivent être protégés du soleil, et placés à hauteur d'homme (autour d'1,60 m), à 1 mètre minimum de toute source de chaleur (radiateurs et même fenêtres), en évitant les murs extérieurs et les zones de courant d'air (portes par exemple).

L'appairage de ces capteurs se fait de façon classique (s'assurer que l'appairage ZigBee est bien autorisé), et appuyer sur le bouton « Reset » du capteur, puis donner un nom significatif (voir la procédure dans la partie « Astuces diverses » de ce document).

Vérifier la présence des capteurs dans l'onglet « Température » de Domoticz.

Le chauffage sera dépendant de l'indicateur d'occupation du FabLab.

On pourrait également ajouter un planning hebdomadaire pour forcer le chauffage les mardis et jeudis, jours habituels d'activité intense du FabLab.

La consigne de départ sera à 19°C sur chaque radiateur.

On va mesurer le temps mis pour chauffer chaque pièce, en fonction de la différence de température avec la consigne initiale. On notera également la valeur de la température externe, pour estimer ensuite son influence.

On mesurera également la perte de température dans le temps, toujours avec la température extérieure, afin de pouvoir anticiper la coupure du chauffage.

Une fois le temps de chauffage et de refroidissement déterminés, on calculera l'avance à l'allumage et à l'extinction pour que le planning soit respecté.

Lorsque le serveur SMS sera installé, il sera possible d'envoyer une demande de chauffage pour une période donnée.

Ultérieurement, la détection de présence pourrait également être utilisée pour allumer le chauffage.

CONDITIONS D'ACTIVATION DES SCRIPTS ET ACTIONS GÉNÉRÉES

INDICATEUR D'AUTORISATION GLOBALE DU CHAUFFAGE

La première chose à faire concernant le chauffage est de créer un interrupteur indiquant si on doit activer ou non le chauffage. Il permettra de ne pas chauffer en inter-saisons et l'été. Il permettra également de désactiver les automatismes dans le cas ou ils ne conviendraient pas dans une circonstance donnée (un jour férié par exemple).

On va donc créer un interrupteur virtuel « Autorisation chauffage ».

Noter qu'on pourra, dans le cas d'une désactivation partielle (par exemple un four férié), utiliser un planning sur cet indicateur (on explique comment juste en dessous)

PLANNING D'ACTIVATION DU CHAUFFAGE LES MARDIS ET JEUDIS

Dans un premier temps, on va créer un planning pour activer le chauffage les mardis et jeudis.

Voir dans la partie « Astuces diverses » la façon de créer un planning. Créer un interrupteur virtuel « Planning chauffage » qui active l'interrupteur les mardis et jeudis de 10h à 19h.

SCRIPT D'ACTIVATION DU CHAUFFAGE À PARTIR DU PLANNING

Il faut ensuite lier l'état du chauffage au planning, en tenant compte de l'indicateur d'autorisation du chauffage.

On va aussi faire en sorte que l'indicateur de présence dans le FabLab soit utilisé, afin de chauffer en dehors des périodes du planning, si besoin. Chaque appui armera le chauffage pour une heure. En tenant évidement compte de l'indicateur d'autorisation du chauffage.

```
-- Activation chauffage
--
--  Active le chauffage en tenant compte de l'autorisation globale,
--      du planning chauffage et de l'indicateur de présence

-- Active ou désactive le chauffage globalement
function setChauffage(mode)
    -- Définit le nouveau en fonction de l'état
    if mode == "On" then
        valeur = "Set Level 10"
    else
        valeur = "Off"
    end
    commandArray['Chauffage salle commune'] = valeur
    commandArray['Chauffage salle mécanique'] = valeur
    commandArray['Chauffage salle aéromodélisme'] = valeur
    commandArray['Chauffage salle électronique'] = valeur
    commandArray['Chauffage salle radio'] = valeur
    commandArray['Chauffage salle imprimantes'] = valeur
    commandArray['Chauffage salle WC'] = valeur
end

-- Teste les dispositifs modifiés
for quiChange, nouvelleValeur in pairs(devicechanged) do
    -- Est-ce qu'on a changé l'autorisation ?
    if quiChange == "Autorisation chauffage" then
        -- Est-ce qu'on interdit le chauffage?
        if nouvelleValeur == "Off" then
            -- Force le chauffage à off
            setChauffage("Off")
        else
            if otherdevices["Planning chauffage"] == "On" or
               otherdevices["FabLab occupé"] == "On" then
                setChauffage("On")
            end
        end
    end
```

```
    -- Est-ce le planning ou l'occupation ?
    if quiChange == "Planning chauffage" or
       quiChange == "FabLab occupé" then
        -- Chauffage autorisé ?
        if otherdevices["Autorisation chauffage"] == "On" then
            -- Mettre la nouvelle valeur
            setChauffage = nouvelleValeur
        end
    end
end

return commandArray
```

SCRIPT DE CALCUL DE LA TEMPÉRATURE MOYENNE

Même si on n'utilise pas pour le moment la température des pièces, on va écrire un script qui calcule la température et l'humidité moyenne dans les pièces qui comportent plusieurs capteurs.

```
-- Température moyenne
--
--  Calcule la température moyenne d'une pièce en fonction de l'ensemble de ses capteurs
--

function string.starts(String,Start)
   return string.sub(String,1,string.len(Start))==Start
end

function calculeMoyenne(quiChange, debutNom, nombreCapteurs)
    -- Est-ce que le dispositif modifié commence par la racine concernée ?
    if string.starts(quiChange, debutNom) then
        temp = 0
        hum = 0
        -- Lire tous les capteurs et  additionner température et humidité
        for numero = 1, nombreCapteurs do
            temp = temp + otherdevices_temperature[debutNom.." "..tostring(numero)]
            hum = hum + otherdevices_humidity[debutNom.." "..tostring(numero)]
        end
        -- Calcul de la moyenne
        temp = temp / nombreCapteurs
        hum = hum / nombreCapteurs
        -- Récupération du numéro du capteur recevant la moyenne
        idx = tostring(otherdevices_idx[debutNom])
        -- Définition barbare de la valeur du dispositif contenant la moyenne
        commandArray[#commandArray + 1] =
            {['UpdateDevice']= idx..'||'..tostring(temp)..';'..tostring(hum)..';0'}
    end
end

commandArray = {}
-- Teste les dispositifs modifiés
for quiChange, nouvelleValeur in pairs(devicechanged) do
    calculeMoyenne(quiChange, "Température salle commune", 3)
    calculeMoyenne(quiChange, "Température salle mécanique", 2)
    calculeMoyenne(quiChange, "Température salle aéromodélisme", 2)
end

return commandArray
```

A enregistrer sous le nom « Température moyenne »

GESTION DE L'EXTRACTION DES FUMÉES DES DÉCOUPES LASER

Le FabLab est équipé d'un système d'extraction des fumées, séparé de la VMC.

L'idée est de détecter l'utilisation des découpes laser par leur consommation électrique pour commander la mise en route de l'extracteur, et sa coupure après un temps donnée de non utilisation.

De plus, on commandera les deux registres qui autorisent l'accès à l'extraction, en prenant soin de fermer le registre de la découpe non utilisée.

La détection de la consommation électrique se fera par une prise de courant ZigBee (Frient 20201200), qui sera alimentée en permanence. Ces prises seront appairées par la procédure normale et renommées (voir « Astuces diverses » à la fin de ce document).

La commande de l'extraction sera réalisée par un module relai (Tongou 25A avec mesure), qui sera installé dans l'armoire électrique générale par un électricien, après le disjoncteur de l'extraction. Il sera appairé et renommé.

Un script LUA détectera l'activation de la découpe, et lancera l'extraction. Elle coupera cette extraction après une durée fixe (mise initialement à 2 minutes) après la fin de la découpe.

Noter que le script doit activer l'extraction au lancement d'une des 2 découpes, mais la désactiver après arrêt des 2 découpes.

Ce même script commandera l'ouverture du registre de la découpe active, et le fermera à expiration de délai après son extinction.

CONDITIONS D'ACTIVATION DES SCRIPTS ET ACTIONS GÉNÉRÉES

Le script doit surveiller la consommation des 2 découpes, et activer l'extraction dès qu'elle dépasse une valeur donnée, qu'on relèvera sur chaque découpe, lorsque le laser est actif. L'arrêt sera programmé 2 minutes après que les 2 lasers seront sous la limite.

Étant donné que la consommation de courant est remontée à intervalle régulier, on va avoir du mal à déterminer le changement d'état des découpes.

INTERRUPTEURS VIRTUELS

On va donc créer 2 indicateurs virtuels « Utilisation laser CO2 » et « Utilisation laser LED » qui seront chargés en fonction des seuils de consommation.

SCRIPT D'ACTIVATION DE L'EXTRACTION

```
-- Activation extraction
--
--  Active l'extraction tant que la consommation des lasers dépasse une limite
--      et l'éteint 2 minutes après que les 2 lasers soient sous leurs limites.
--
-- Définition des noms utilisés
consommationCo2 = "Consommation*laser CO2"
utilisationCo2 = "Utilisation*laser CO2"
consommationLed = "Consommation*laser LED"
utilisationLed = "Utilisation*laser LED"
commandeExtraction = "Commande extraction"
-- Teste les dispositifs modifiés
for quiChange, nouvelleValeur in pairs(devicechanged) do
```

```
    if quiChange == consommationCo2 then
        -- Détermine si la conso dépasse le seuil
        if nouvelleValeur > 10 then
            laser = "On"
        else
            laser = "Off"
        end
        -- Met à jour l'utilisation si besoin
        if otherdevices[utilisationCo2] ~= laser then
            commandArray[utilisationCo2] = laser
        end
    end
    if quiChange == consommationLed then
        -- Détermine si la conso dépasse le seuil
        if nouvelleValeur > 10 then
            laser = "On"
        else
            laser = "Off"
        end
        -- Met à jour l'utilisation si besoin
        if otherdevices[utilisationLed] ~= laser then
            commandArray[utilisationLed] = laser
        end
    end
    if quiChange == utilisationCo2 or quiChange == utilisationLed then
        -- Active l'extraction si on débute l'utilisation
        if nouvelleValeur > "On" then
            commandArray[commandeExtraction] = "On"
        else
            -- Sinon, couper l'extraction dans 2 minutes si les 2 lasers sont inactifs
            if otherdevices[utilisationCo2] == "Off" and otherdevices[utilisationLed] == "Off"
then
                commandArray[commandeExtraction] "Off AFTER 2"
            end
        end
    end

end

return commandArray
```

GESTION DE LA VMC

Dans un premier temps, la VMC sera asservie par un script à l'indicateur d'occupation du FabLab.

Comme pour l'extracteur de fumées, on utilisera un un module relai (Tongou 25A avec mesure), qui sera installé dans l'armoire électrique générale par un électricien, après le disjoncteur de la VMC. Il sera appairé et renommé.Conditions d'activation des scripts et actions générées

Dans un premier temps, on va lier la commande de la VMC à l'indicateur d'occupation du FabLab.

SCRIPT D'ACTIVATION DE LA VMC

```
-- Activation VMC
--
--  Active la VMC en tenant compte de l'indicateur de présence

-- Teste les dispositifs modifiés
for quiChange, nouvelleValeur in pairs(devicechanged) do
    if quiChange == "FabLab occupé" then
       commandArray["Commande VMC"] = nouvelleValeur
    end
end

return commandArray
```

GESTION DU CHAUFFE-EAU

Dans un premier temps, le chauffe eau sera mis en route par appui sur un bouton poussoir et géré par un script.

Noter qu'on pourrait également l'asservir par un script à l'indicateur d'occupation du FabLab, dans le cas où la durée de chauffage de l'eau serait trop longue.

Comme pour l'extracteur de fumées et la VMC, on utilisera un un module relai (Tongou 25A avec mesure), qui sera installé dans l'armoire électrique générale par un électricien, après le disjoncteur de la VMC. Il sera appairé et renommé.

Même chose pour le bouton poussoir qui sera placé à côté du chauffe-eau.

Il pourrait être habile de mettre un dispositif de retro-signalisation afin de visualiser l'activation du chauffage de l'eau.

CONDITIONS D'ACTIVATION DES SCRIPTS ET ACTIONS GÉNÉRÉES

Dans un premier temps, on va écrire un script qui activera le chauffe-eau pendant une heure à chaque appui court sur le bouton, et l'éteint sur un appui long. L'alimentation sera coupée lorsque l'indicateur de présence dans le FabLab sera inactif.

On ajustera la durée de chauffage au temps constaté ensuite.

On pourrait même penser à couper l'alimentation lorsque la consommation retombe à zéro.

Ces modifications seront apportées ultérieurement, afin de ne pas compliquer trop les premières versions.

SCRIPT D'ACTIVATION DU CHAUFFE-EAU

```
-- Activation chauffe-eau
--
--  Active le chauffe-eau après appui sur le bouton du chauffe-eau,
--      et l'éteint au bout d'une heure, ou quand le FabLab est vide.

-- Teste les dispositifs modifiés
for quiChange, nouvelleValeur in pairs(devicechanged) do
    -- Allume le chauffe-eau pour une heure lors d'un appui court sur le bouton,
    --      l'éteint sur appui long
    if quiChange == "Poussoir chauffe-eau (Action)_single" and nouvelleValeur == "On" then
        commandArray["Commande chauffe-eau"] = "On for 60"
    end
    if quiChange == "Poussoir chauffe-eau (Action)_long" and nouvelleValeur == "On" then
        commandArray["Commande chauffe-eau"] = "Off"
    end
    -- Eteint le chauffe-eau lorsque le FabLab devient vide
    if quiChange == "FabLab occupé" then
            if nouvelleValeur == "Off" then
                commandArray["Commande chauffe-eau"] = "Off"
        end
    end
end

return commandArray
```

DÉLESTAGE DES GROS CONSOMMATEURS

L'abonnement électrique du FabLab est de 12 kW.

Le souci est que le besoin du chauffage est de 19,5 kW si l'ensemble des radiateurs est actif.

De plus, le chauffe-eau consomme 1,2 kW.

On pige assez vite qu'avec ça, on risque de satelliser le disjoncteur EDF a intervalle régilier, d'autant plus rapproché qu'il fera froid, aggravé par le fait qu'on ne chauffe qu'occasionnellement, et qu'à l'arrivée dans le FabLab, on aura une pointe de consommation.

Heureusement, on sait mesurer les consommations sur le Linky, et sur les gros consommateurs, et on peut les commander.

L'idée est donc d'implémenter un gestionnaire de délestage des gros consommateurs pour éviter des disjonctions à répétition, défini comme suit :

- On repère les gros consommateurs (radiateurs, chauffe-eau, laser CO2, ...) et on mesure leur consommation,
- On définit une priorité en cas de manque de puissance (par exemple, laser, puis radiateur, puis chauffe-eau),
- On définit une température cible au niveau de chaque pièce (18° dans la salle mécanique, 20° dans la salle électronique, 17° dans les WC et 19° ailleurs)
- On active les radiateurs dans les pièces qui ont le plus besoin de chauffage (différence entre la température cible et la température de la pièce),
- Avec un intervalle court, on mesure la consommation pour chaque phase sur le Linky et on active les dispositifs dans l'ordre décroissant de priorité, jusqu'à ce qu'on épuise la capacité maximale de chaque phase.
- On boucle sur la ligne précédente.

Accessoirement, le Linky ne retourne pas de puissance par phase, mais une puissance globale (pas très utile pour nous), et une intensité instantanée par phase. Petite astuce : Enedis calcule ces intensités sur la base d'une tension de 200 V pour convertir les puissances en intensité, il faudra utiliser le même coefficient. Par exemple, l'abonnement de 12 kW, soit 4 kW par phase, correspond à 20 A par phase (c'est d'ailleurs ce qui est retourné par le Linky). Reste à voir si l'intensité retournée est calculée par mesure réelle du courant, ou par calcul à partir de la puissance.

CONDITIONS D'ACTIVATION DES SCRIPTS ET ACTIONS GÉNÉRÉES

Pour intégrer facilement les modifications du délesteur, sans modifier les scripts qui commandent actuellement directement les gros consommateurs, on va créer des interrupteurs virtuels de nommés « xxx délesté(e)» au lieu de « xxx ». C'est le script de délestage qui copiera (ou pas) le contenu de « xxx » dans « xxx délesté(e) ».

Par exemple « Commande chauffe-eau » aura un pendant « Commande chauffe-eau délestée ».

Lorsque la commande est liée directement au dispositif physique, il faut d'abord renommer le dispositif en y ajoutant « délesté(e) », puis créer un dispositif virtuel du même type, avec le nom initial. Les scripts déjà en place commanderont le dispositif virtuel (sans rien changer), et c'est le délesteur qui commandera le dispositif physique.

On va implémenter la gestion du délestage sur la base d'un script activé) chaque changement de conso par phase :

- Prendre la conso globale, par phase
- Relever les consommations des éléments délestables
- Calculer le puissance restante (globale - délestable)
- (Ré) attribuer la puissance disponible sur les éléments délestables qui sont actifs, jusqu'à ce qu'il n'en reste plus.

RENOMMAGE DES DISPOSITIFS DE COMMANDE EXISTANTS

On doit tout d'abord ajouter « délesté(e) » sur les dispositifs de commande qu'on va délester (les radiateurs et le chauffe eau)

CRÉATION DES DISPOSITIFS INTERMÉDIAIRES

Ensuite on doit créer des dispositifs virtuels avec pour nom les dispositifs qu'on vient de renommer.

CRÉATION DES THERMOSTATS DE TEMPÉRATURE CIBLE PAR PIÈCE

On va créer un thermostat virtuel par pièce, afin de définir sa température de chauffage « normale ». Le type est « setpoint ».

SCRIPT D'ACTIVATION DE GESTION DU DÉLESTAGE

Principe de base :

```
Délestage
  Lors d'un changement de valeur de consommation sur une phase
    Récupérer le numéro de phase modifié
    Appeler la fonction d'analyse sur la phase concernée

Fonction d'analyse par phase :
  Charger les données pour la phase concernée
    Par dispositif concerné (constant)
      Nom de la commande virtuelle
      Nom de la commande réelle (délestée)
      Nom du capteur de température (vide si pas utilisé)
      Nom du thermostat température cible (vide si pas utilisé)
      Puissance nominale du dispositif en W
      Priorité d'origine (0-600)
    Par dispositif concerné (variable)
      Priorité calculée (999)
      Indicateur d'activité (0)
    Pour la phase concernée
      Puissance maximale avant disjonction = ampérage max (lu sur le compteur) * 200
      Puissance consommée actuelle (0)
      Puissance disponible pour le délestage (0)
      Puissance restante après délestage (0)
```

```
  Calcul de la puissance disponible
    Relever la puissance consommée = ampérage instantané de la phase * 200
    Définir la puissance disponible comme la puissance maximale moins la puissance consommée
  Scan des dispositifs
    Ajouter la consommation actuelle à la puissance disponible pour le délestage
    Charger la priorité
      Si l'élément est activable (sur la base de la commande virtuelle)
        Si le capteur de température est défini
          Calculer le delta = température cible - température actuelle
          Si le delta est positif (température non atteinte)
            Définir la priorité calculée comme priorité d'origine plus 10 fois le delta
        Sinon (le capteur de température n'est pas défini)
          Définir la priorité calculée égale à la priorité d'origine
  Dans l'ordre des priorités décroissantes
    Si la puissance disponible est suffisante
      Marquer le dispositif actif
      Déduire la consommation du dispositif  de la puissance disponible
  Scanner les dispositifs
    D'abord désactiver les dispositifs inactifs
    Ensuite activer les autres
```

Voici le script :

```
-- Délestage
--
--  Délestage électrique du FabLab, phase par phase (voir la doc pour les détails)
--

function phaseChanged(phaseNumber, deviceValue, deviceTable)
    print("Phase "..tonumber(phaseNumber).." consomme "..tonumber(deviceValue))
        -- Pointeurs dans la table
        cdeVirtuelle = 1
        cdeDelestee = 2
        consommation = 3
        temperature = 4
        thermostat = 5
        puissance = 6
        priorite = 7
        -- Tables par dispositif
        priorites = {}
        -- Puissance disponible = max - utilisée
        pDispo = (tonumber(otherdevices["Consommation électrique globale souscrite"]) -
deviceValue) * 200
        -- Analyse de la table
        for i, dispositif in pairs(deviceTable) do
                montre("cdeVirtuelle", cdeVirtuelle, dispositif, otherdevices)
                montre("cdeDelestee", cdeDelestee, dispositif, otherdevices)
                montre("consommation", consommation, dispositif, otherdevices)
                montre("température", temperature, dispositif, otherdevices_temperature)
                montre("thermostat", thermostat, dispositif, otherdevices)
                -- Est-ce que le dispositif est actif ?
                actif = 0
                if dispositif[temperature] ~= "" then
                        if otherdevices[dispositif[cdeVirtuelle]] ~= "Arrêt" then
                                actif = 1
                        end
                else
                        if otherdevices[dispositif[cdeVirtuelle]] == "On" then
                                actif = 1
                        end
                end
                print(dispositif[cdeVirtuelle].., actif = "..tostring(actif))
                -- Ajout des consommations au disponible, définition de la priorité
                if actif ~= 0 then
```

```
                        pDispo = pDispo + tonumber(otherdevices[dispositif[consommation]])
                        -- On a un capteur de température, on l'utilise
                        if dispositif[temperature] ~= "" then
                                print("Thermostat: "..dispositif[thermostat])
                                print("Val: "..otherdevices[dispositif[thermostat]])
                                print("Température: "..dispositif[temperature])
                                print("Val:
"..otherdevices_temperature[dispositif[temperature]])
                                deltaTemperature =
tonumber(otherdevices[dispositif[thermostat]]) -
tonumber(otherdevices_temperature[dispositif[temperature]])
                                if deltaTemperature > 0 then
                                        -- On doit encore chauffer
                                        priorites[i] = dispositif[priorite] + (deltaTemperature *
10)
                                else
                                        -- On est à la température cible (ou au dessus)
                                        actif = 0
                                        priorites[i] = -1
                                end
                        else
                                -- Pas de capteur de température, priorité originale
                                priorites[i] = dispositif[priorite]
                        end
                else
                        -- Dispositif pas actif
                        priorites[i] = -1
                end
        end

        -- Analyser les dispositifs par ordre décroissant de priorité
        for i, priorite in spairs(priorites, function(t, a, b) return t[b] < t[a] end) do
                dispositif = table[i]
                if priorite >= 0 then
                        -- Priorité positive, le dispositif est actif
                        if pDispo < dispositif[puissance] then
                                -- Pas assez de puissance dispo, éteindre le dispositif
                                priorites[i] = -1
                        else
                                -- Déduire la puissance du dispositif du disponible
                                pDispo = pDispo - dispositif[puissance]
                        end
                end
        end

        -- Analyser les dispositifs par ordre croissant de priorité (les -1 (off) sortiront en
premier)
        for i, priorite in spairs(priorites, function(t, a, b) return t[b] > t[a] end) do
                dispositif = table[i]
                if priorite >= 0 then
                        -- Priorité positive, le dispositif est actif
                        if dispositif[temperature] ~= "" then
                                changeDispositif(dispositif[cdeDelestee],
otherdevices_svalues[cdeVirtuelle], "Set Level ", otherdevices_svalue)  -- Copier le niveau
d'origine
                        else
                                changeDispositif(dispositif[cdeDelestee], "On", "",
otherdevices)
                        end
                else
                        -- Priorité négative, éteindre le dispositif
                        if dispositif[temperature] ~= "" then
                                changeDispositif(dispositif[cdeDelestee], "30", "Set Level ",
otherdevices_svalue)  -- Level 30 : arrêt
                        else
                                changeDispositif(dispositif[cdeDelestee], "Off", "",
orherdevices)
```

```
                        end
                end
        end
end

-- Tri d'une table
function spairs(t, order)
        -- Générer la liste des clefs
        local keys = {}
        for k in pairs(t) do keys[#keys+1] = k end
        -- Trier la table avec la fonction, ou juste sur les clefs
        if order then
                table.sort(keys, function(a,b) return order(t, a, b) end)
        else
                table.sort(keys)
        end

        -- Retourne l'itérateur
        local i = 0
        return function()
                i = i + 1
                if keys[i] then
                        return keys[i], t[keys[i]]
                end
        end
end

-- Modifie le dispositif délesté si besoin
function changeDispositif(dispositif, valeur, prefix, tableDomoticz)
    print("Devrait mettre "..dispositif.." à "..valeur)
        -- On regarde si on est différent de la valeur courante
        if tableDomoticz[dispositif] =~ valeur then
                commandArray[dispositif] = prefix..valeur
        end
end

commandArray = {}

-- Recherche le déclencheur
for deviceName, deviceValue in pairs(devicechanged) do
    if deviceName == "Consommation électrique globale phase 1" then
        table = {}
        table[1] = {'Chauffage salle mécanique 1', 'Chauffage salle mécanique 1 délesté', 'Consommation salle mécanique 1', 'Température salle mécanique', 'Thermostat salle mécanique', 2000, 0}
        table[2] = {'Chauffage salle radio', 'Chauffage salle radio délesté', 'Consommation salle radio', 'Température salle radio', 'Thermostat salle radio', 1500, 0}
        table[3] = {'Chauffage salle imprimantes', 'Chauffage salle imprimantes délesté', 'Consommation salle imprimantes', 'Température salle imprimantes', 'Thermostat salle imprimantes', 1500, 0}
        phaseChanged(1, deviceValue, table)
    end
    if deviceName == "Consommation électrique globale phase 2" then
        table = {}
        table[1]= {'Chauffage salle électronique', 'Chauffage salle électronique délesté', 'Consommation salle électronique', 'Température salle électronique', 'Thermostat salle électronique', 2000, 0}
        table[2] = {'Chauffage salle commune 2', 'Chauffage salle commune 2 délesté', 'Consommation salle commune 2', 'Température salle commune', 'Thermostat salle commune', 2000, 0}
        table[3] = {'Commande chauffe-eau', 'Commande chauffe-eau délestée', 'Consommation chauffe-eau', '', '', 1200, 300}
        phaseChanged(2, deviceValue, table)
    end
    if deviceName == "Consommation électrique globale phase 3" then
        table = {}
```

```
        table[1] = {'Chauffage salle aéromodélisme 1', 'Chauffage salle aéromodélisme 1 délesté', 'Consommation salle aéromodélisme 1', 'Température salle aéromodélisme', 'Thermostat salle aéromodélisme', 2000, 0}
        table[2] = {'Chauffage WC', 'Chauffage WC délesté', 'Consommation WC', 'Température WC', 'Thermostat WC', 500, 100}
        phaseChanged(3, deviceValue, table)
    end
end

-- Trace des changements
for dispositif, valeur in pairs(commandArray) do
    print("Dispositif "..dispositif.." va changer en "..tostring(valeur))
end

return commandArray
```

AUTORISATION D'ACCÈS AUX IMPRIMANTES 3D

Le besoin exprimé est d'autoriser l'accès aux imprimantes 3D uniquement aux personnes formées, afin d'éviter des détériorations dues a une mauvaise utilisation sur un équipement coûteux.

L'idée est d'utiliser une prise commandée (Frient 20201200) qui sera activée après lecture du badge d'ouverture de la porte du FabLab de l'utilisateur. La mesure de la consommation de l'imprimante détectera la fin d'utilisation, et coupera l'alimentation après un délai qui assurera le bon refroidissement des différents éléments.

Un module spécifique basé sur un ESP8266 et un lecteur de badge a été développé et enverra l'identifiant du badge lu à Domoticz, qui activera l'alimentation de l'imprimante.

Le module sera connecté à une alimentation 5V de type téléphone ou sur le PC de management de l'équipement lorsqu'il y en a un.

On fera un beau boîtier pour le module … imprimé en 3D ;-)

ENVOI ET RÉCEPTION DE SMS

A l'heure actuelle, le FabLab est connecté au réseau de l'université. L'accès à Internet est possible en WiFi, à partir du moment où on possède un identifiant (et son mot se passe). Le souci est que l'autorisation de la connexion n'est que temporaire, et qu'il faut se reconnecter après un certain temps, ce qui n'est pas envisageable pour un système autonome qui a besoin en permanence d'une connexion à Internet, pour envoyer des mails par exemple.

En attendant de trouver une solution à ce problème (à choisir probablement entre installer une connexion privée pour le FabLab, ou négocier un accès permanent au réseau avec l'université), il a été décidé d'utiliser un serveur GSM pour envoyer/recevoir des messages.

L'envoi de SMS permettra de signaler des évènement graves à une liste de numéros de téléphones.

La réception permettra de gérer certaines fonctions depuis l'extérieur (actions sur des dispositifs, retour d'état, …). Seuls les correspondants figurant dans une liste seront autorisés à communiquer avec le système, les autres seront simplement tracés et ignorés.

Un document séparé décrit l'ensemble des étapes pour réaliser le matériel, charger les micro-logiciels, installer et paramétrer le plug-in.

SÉCURISATION DES LOCAUX

L'idée de base est d'installer 11 détecteurs de présence 3 dans la pièce centrale, 2 dans les sales mécanique et aéromodélisme, 1 dans les salles électronique, radio, imprimantes 3D et les WC.

On installera également une sirène intérieure.

On pourrait utiliser judicieusement l'onduleur donné par Leroy Merlin pour assurer l'alimentation de la domotique et de ses périphériques en cas de coupure de courant 240V.

GESTION DE L'ÉCLAIRAGE

L'idée est de pouvoir commander l'éclairage du FabLab depuis la domotique.

On pourrait :

- l'activer en cas de détection de personnes quand la luminosité ambiante est trop basse (et l'éteindre automatiquement lorsque la pièce est vide),
- l'allumer en cas de détection d'intrusion,
- l'éteindre automatiquement lorsque le FabLab est vide,
- …

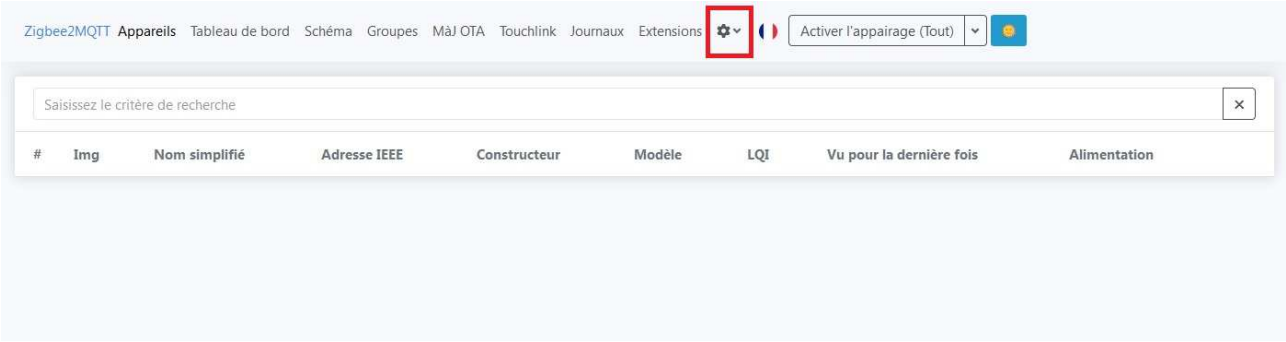
Côté réalisation, il faudrait remplacer le télérupteur par un modèle ZigBee, et les interrupteurs simples par des modèles ZigBee.

ASTUCES DIVERSES

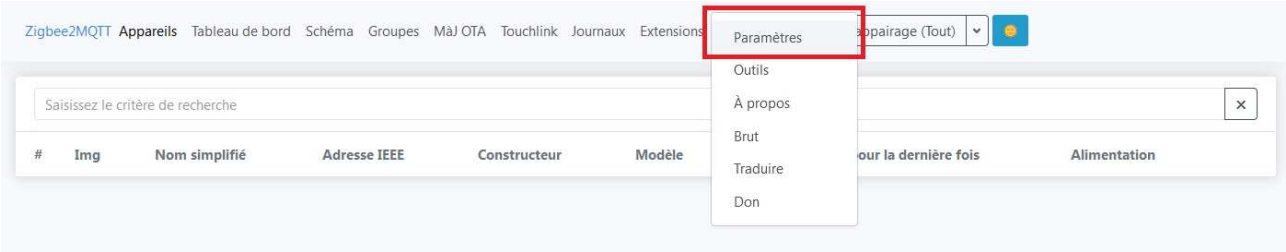
APPAIRAGE DE NOUVEAUX DISPOSITIFS ZIGBEE

Avant d'appairer de nouveaux dispositifs, vérifier que Zigbee2MQTT est en position de les accepter.

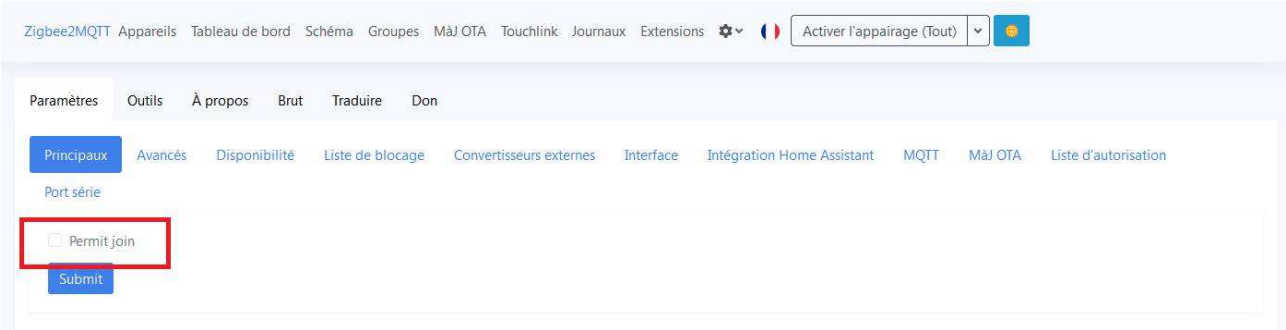
On le fait dans les paramètres, en cliquant sur la roue (encadrée en rouge ci-dessous) ...



... puis sur « Paramètres » :

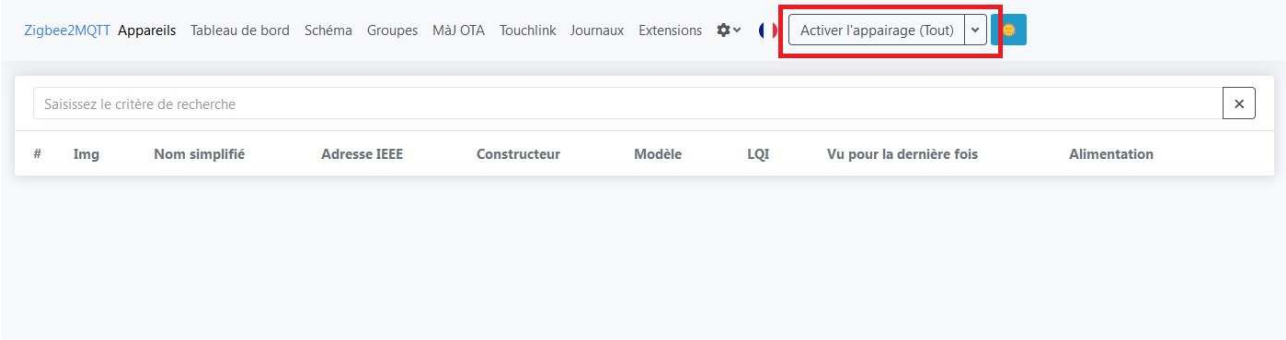


On doit voir une fenêtre comme ça :



Vérifier que la case « Permit join » est bien cochée (sinon, la cocher er cliquer sur « Submit) »

Note : cette façon de faire est didactique, pour montrer l'accès au paramétrage de Zigbee2MQTT. Dans la pratique, on peut vérifier simplement l'état de l'appairage en regardant sur l'écran principal, dans la boîte encadrée en rouge :



Si on trouve « Activer l'appairage (tout) », c'est que l'appairage n'est pas actif. Cliquer dessus pour l'activer. Le texte change en « Désactiver l'appairage (tout) » suivi d'un décompte en minutes/secondes. Ce décompte indique le temps restant pour l'appairage avant que Zigbee2MQTT ne le désactive (le clic l'a activé pour 5 minutes).

Penser également à interdire la création de nouveaux dispositifs dans Domoticz (expliqué juste en dessous).

Une fois l'appairage actif, on peut utiliser la procédure spécifique au dispositif à appairer (voir la doc du dispositif pour plus de détails).

En général, l'idée est d'appuyer sur un bouton « Reset » pendant quelques secondes, ce qui fait clignoter une LED quelques fois, puis s'éteint. Une fois le dispositif appairé, il clignote généralement 3 fois pour dire que la procédure est ok, ou une seule pour signaler un échec.

De plus, si la procédure se déroule correctement, le dispositif apparaît dans la liste des dispositifs de Zigbee2MQTT.

Changer ensuite le nom du dispositif créé dans Zigbee2MQTT (voir plus bas).

Puis autoriser l'intégration de nouveaux dispositifs dans Domoticz (voir encore ci-dessous).

Attendre l'apparition du nouveau dispositif ZigBee, soit en forçant une mise à jour, soit en relançant le matériel « Auto-découverte MQTT » (dans l'onglet « Configuration » / « Matériel », cliquer sur la ligne « Auto-découverte MQTT », puis sur modifier).

Puis interdire l'intégration de nouveaux dispositifs dans Domoticz.

Sélectionner les parties du dispositif à afficher et ajuster leur nom et/ou leur type (comme d'habitude, voir en dessous).

INTERDIRE LA CRÉATION DE NOUVEAUX DISPOSITIFS DOMOTICZ

Le blocage de la création de nouveaux dispositifs dans Domoticz permet d'éviter la création intempestive de … nouveau dispositifs dans Domoticz ;-). Pourquoi faire ?

Si la création est autorisée, Domoticz va créer automatiquement un dispositif s'il n'existe pas déjà. Or, certains matériels (comme ZigBee ou les radios 433/868) scannent leurs réseaux respectifs et détectent tout ce qui passe.

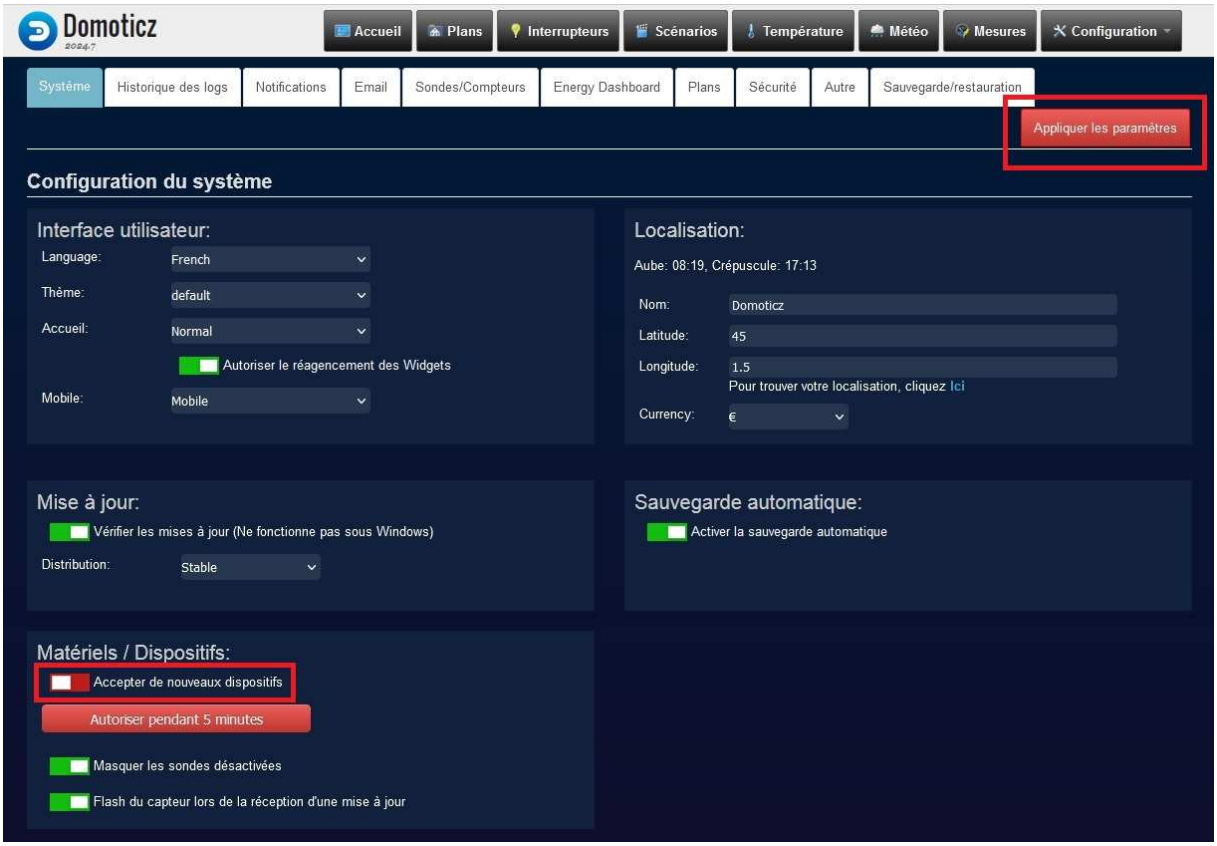
Dans le cas d'une radio 433/868, comme on n'est généralement as le suel à utiliser ce type de technologie dans son quartier, on récupère toutes les commandes des voisins. Ça peut être pratique si on est joueur, mais c'est probablement défendu quelque part dans les lois françaises, et risqué, surtout si le voisin est demi de mêlée, ce qui n'est pas rare chez nous ;-)

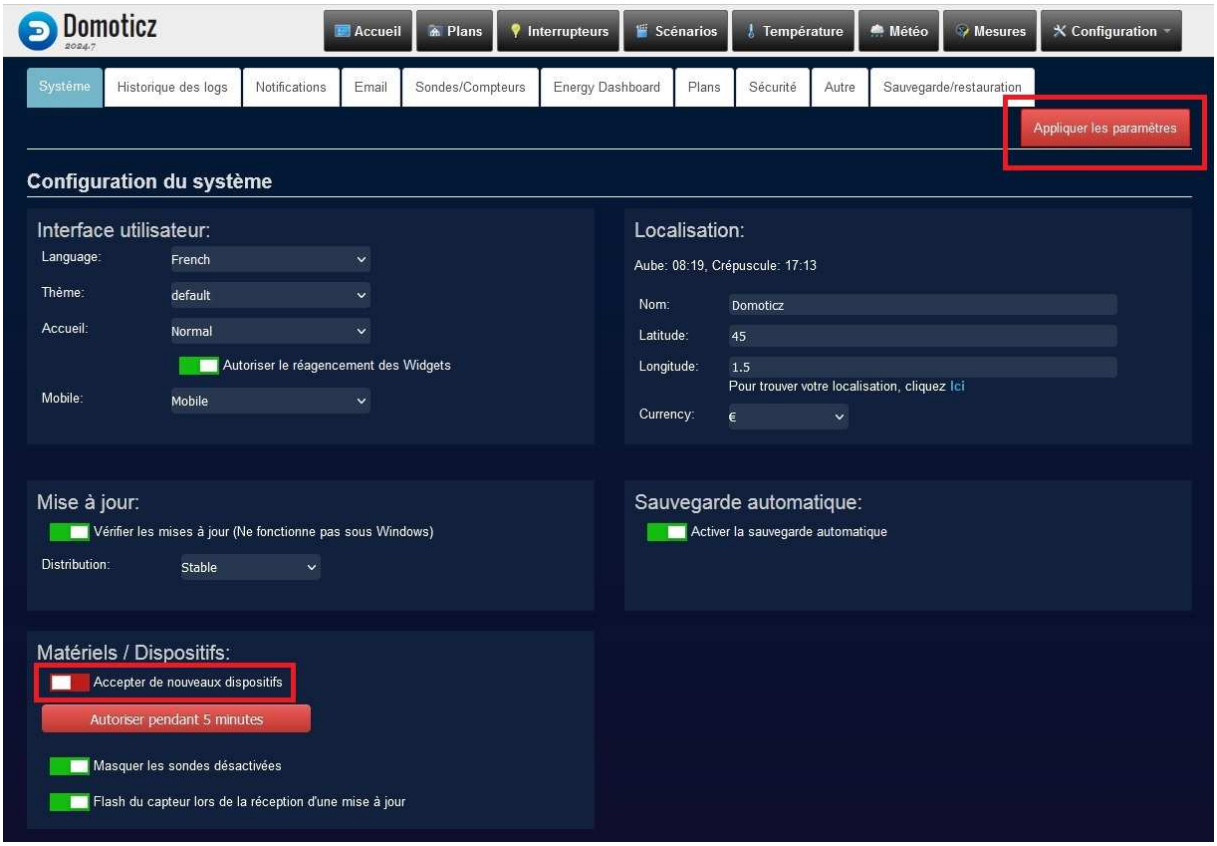
Dans le cas du ZigBee, le souci est différent : tant qu'on n 'a pas changé le nom du module ZigBee dans ZigBee2MQTT, il utilise par défaut l'adresse MAC du module. Dans ce cas, Domoticz reçoit ce nom barbare et crée un capteur avec.

Mais si on a interdit la création, on aura juste un message indiquant que le dispositif n'a pas été créé à cause de cette interdiction.

Du coup, il est habile de laisser la création automatique interdite, d'appairer le dispositif dans Zigbee2MQTT, de changer son nom (toujours dans Zigbee2MQTT), puis d'autoriser la création dans Domoticz, afin de récupérer un nom significatif (plutôt que l'adresse MAC indigeste).

Pour interdire cette création, aller dans le menu « Configuration » / « Paramètres », onglet « Système ».



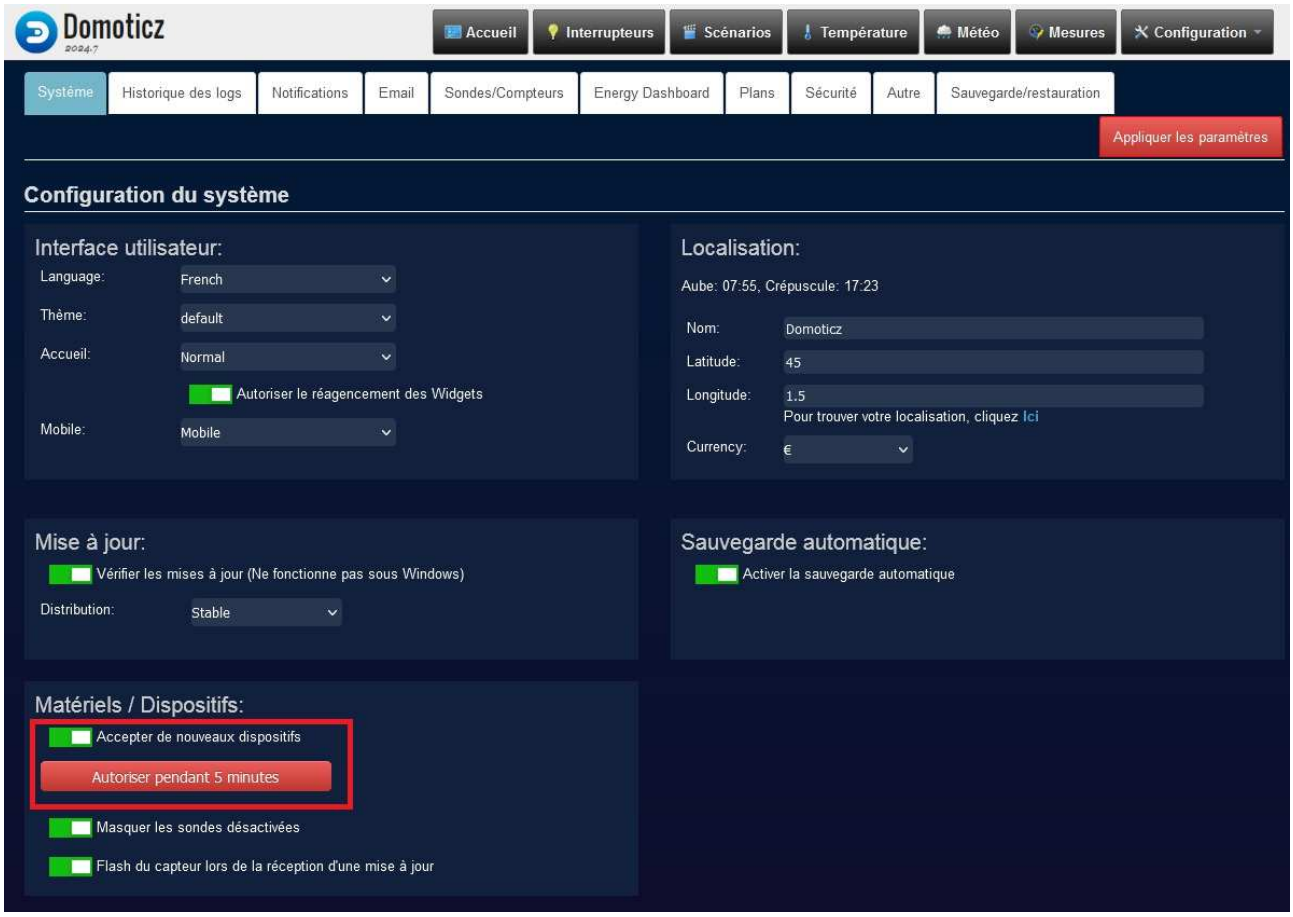


Vérifier que le sélecteur « Accepter de nouveaux dispositifs » soit décoché (rouge). Le modifier si nécessaire.

Penser à valider les changements (s'il y en a eu) en cliquant sur « Appliquer les paramètres »

AUTORISER LA CRÉATION DE NOUVEAUX DISPOSITIFS DOMOTICZ

Aller dans le menu « Configuration » / « Paramètres », onglet « Système ».



Vérifier que le sélecteur « Accepter de nouveaux dispositifs » soit coché (vert), ou qu'on ait appuyé « Autoriser pendant 5 minutes » … il y a moins de 5 minutes ;-).

Penser à valider les changements (s'il y en a eu) en cliquant sur « Appliquer les paramètres »

CHANGER LE NOM D'UN DISPOSITIF ZIGBEE DANS ZIGBEE2MQTT

Il peut être habile de remplacer l'adresse MAC du dispositif par un nom plus significatif, du genre « Température local électronique ».

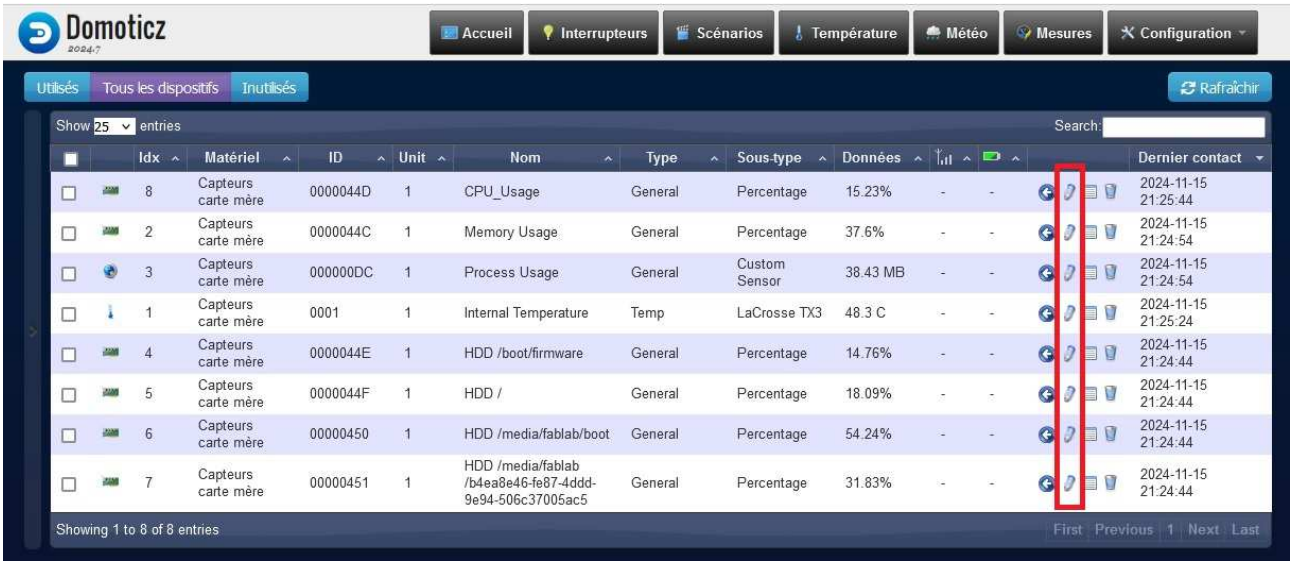
On le fait en cliquant sur l'icône 

Donner ensuite le nom du dispositif et valider.

CHANGER LE NOM D'UN DISPOSITIF DOMOTICZ

On peut modifier le nom d'un dispositif Domoticz de deux façons :

Au travers du menu « Configuration » / « Dispositifs », en cliquant sur l'icône représentant un crayon sur a ligne du dispositif dont on souhaite modifier le nom.



Modifier ensuite le nom du dispositif dans la fenêtre suivante, puis cliquer sur « Renommer »

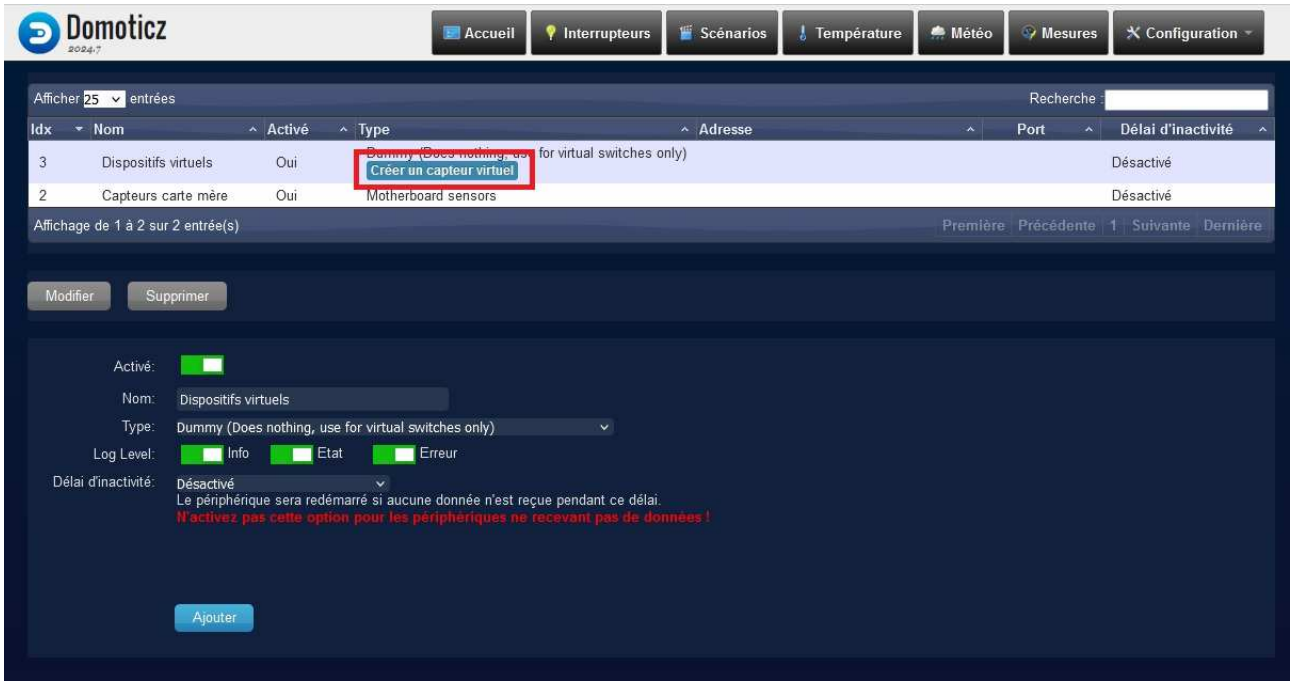
L'autre façon consiste à cliquer sur le bouton « Modifier » présent sur chaque boîte des dispositifs. Par exemple :

La suite est identique au cas précédent.

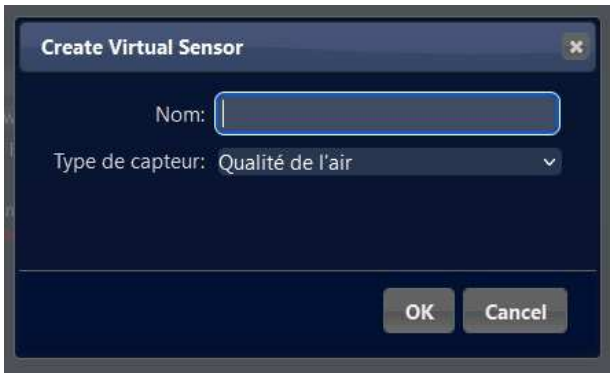
CRÉER UN DISPOSITIF VIRTUEL

La création d'un dispositif virtuel se fait à partir de l'onglet « Configuration » / « Matériel » du menu de Domoticz.

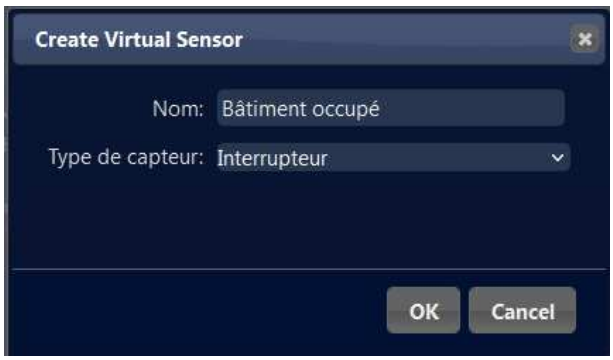
Cliquer sur le bouton « Créer un capteur virtuel » de la ligne « Dispositifs virtuels »:



La fenêtre suivante s'affiche :



Donner « Bâtiment occupé » dans la zone « Nom », et choisir « Interrupteur » dans « Type de capteur »



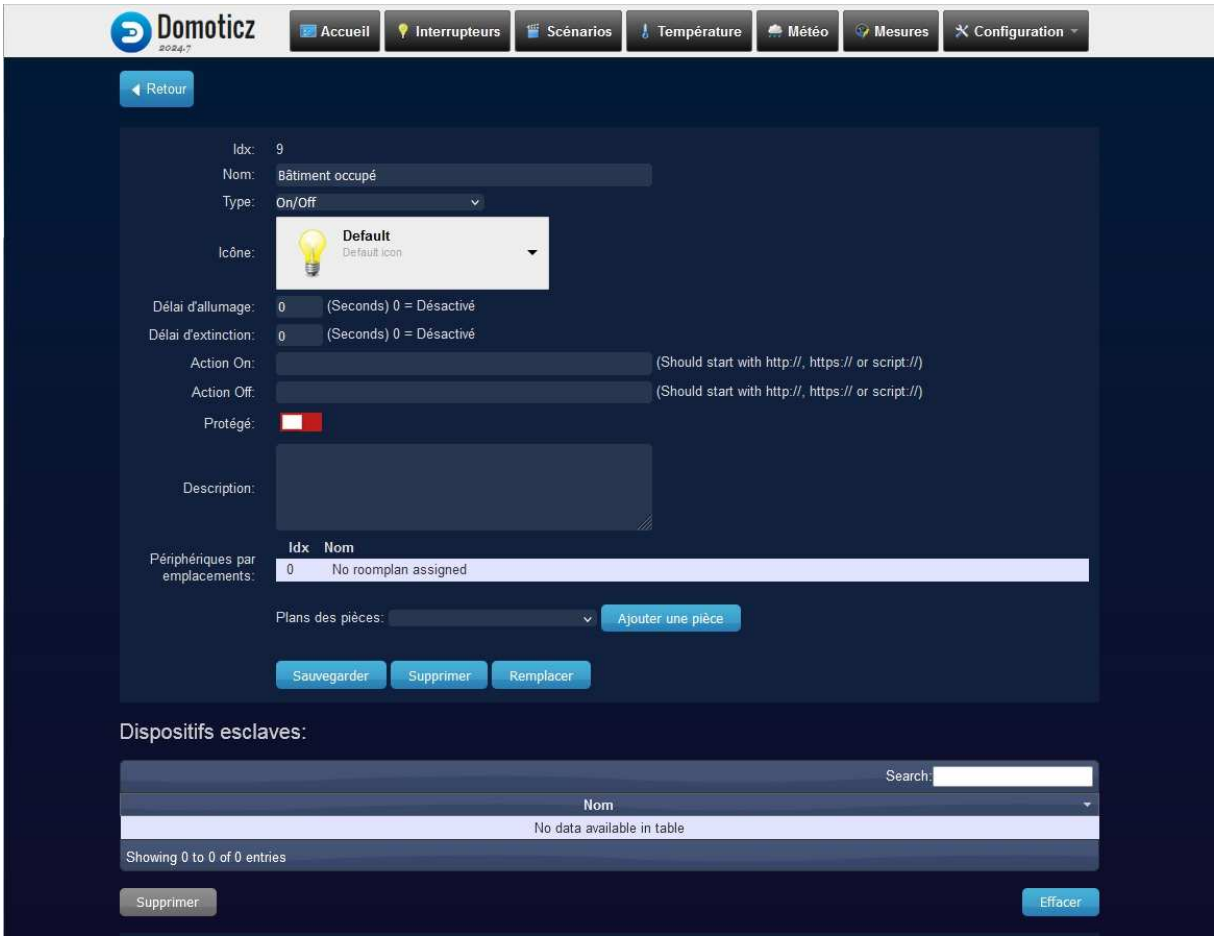
Valider par « Ok ».

On peut retrouver le dispositif créé dans l'onglet « Interrupteurs » :

Bien évidement, il faut adapter le nom et le type du dispositif à votre besoin.

De plus, certains dispositifs permettent de définir plus finement leurs paramètres au besoin de l'utilisateur. Pour le faire, cliquer sur le bouton « Modifier » du dispositif.

La fenêtre qui s'ouvre dépend du type de ce dispositif. Dans le cas d'un interrupteur, on verra :



CRÉER UN PLANNING SUR UN DISPOSITIF

Il est facilement possible de créer un planning sur un dispositif existant. Dans le cas le plus simple, l'action est de type on/off, mais peut pleut également contenir plus de détails (comme une température).

Pour créer, modifier et/ou supprimer un planning, cliquer sur le bouton « Planning » du dispositif concerné :

La fenêtre suivante s'ouvre :

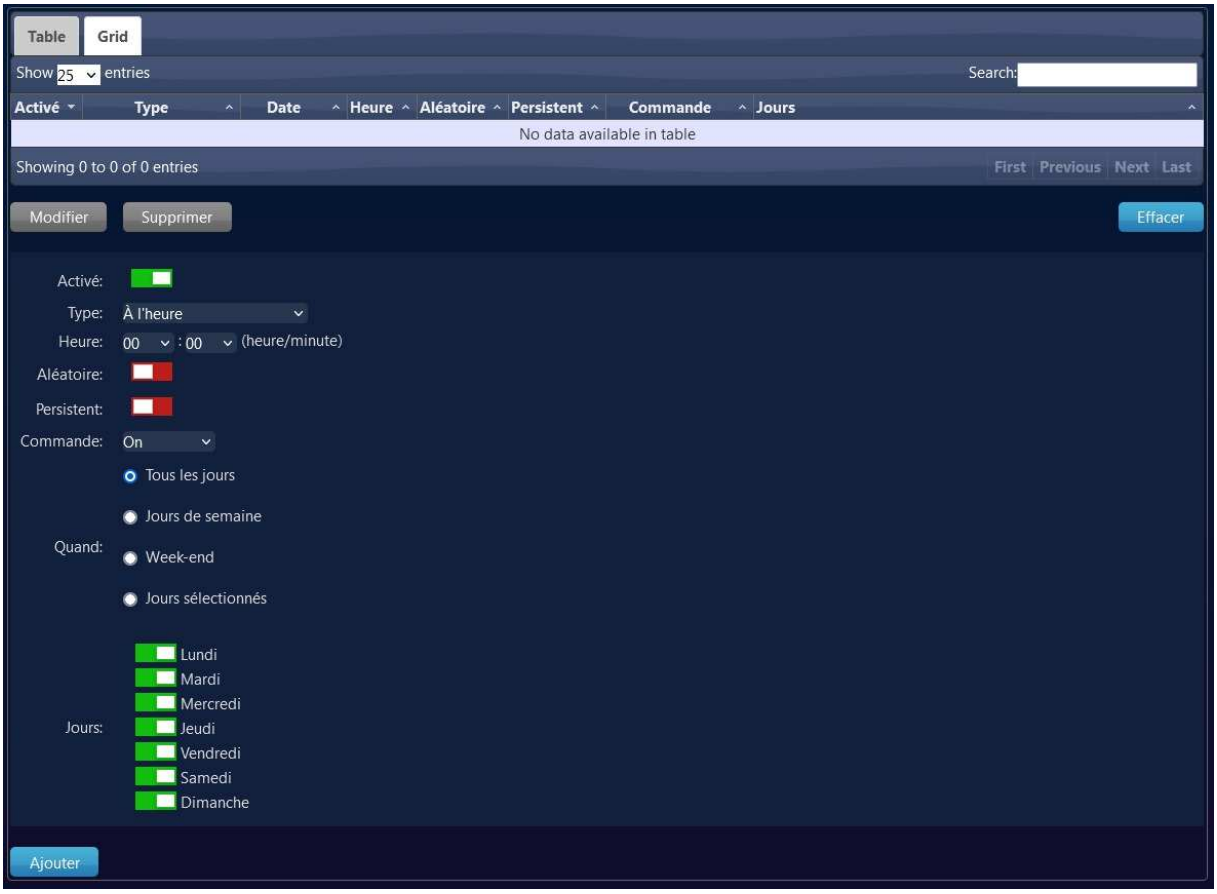


Table
Grid
Show 25 entries
Search:
Activé
Type
Date
Heure
Aléatoire
Persistent
Commande
Jours
No data available in table
Showing 0 to 0 of 0 entries
First
Previous
Next
Last
Modifier
Supprimer
Effacer
Activé:
Type:
À l'heure
Heure:
00
: 00
(heure/minute)
Aléatoire:
Persistent:
Commande:
On
Quand:
Tous les jours
Jours de semaine
Week-end
Jours sélectionnés
Jours:
Lundi
Mardi
Mercredi
Jeudi
Vendredi
Samedi
Dimanche
Ajouter

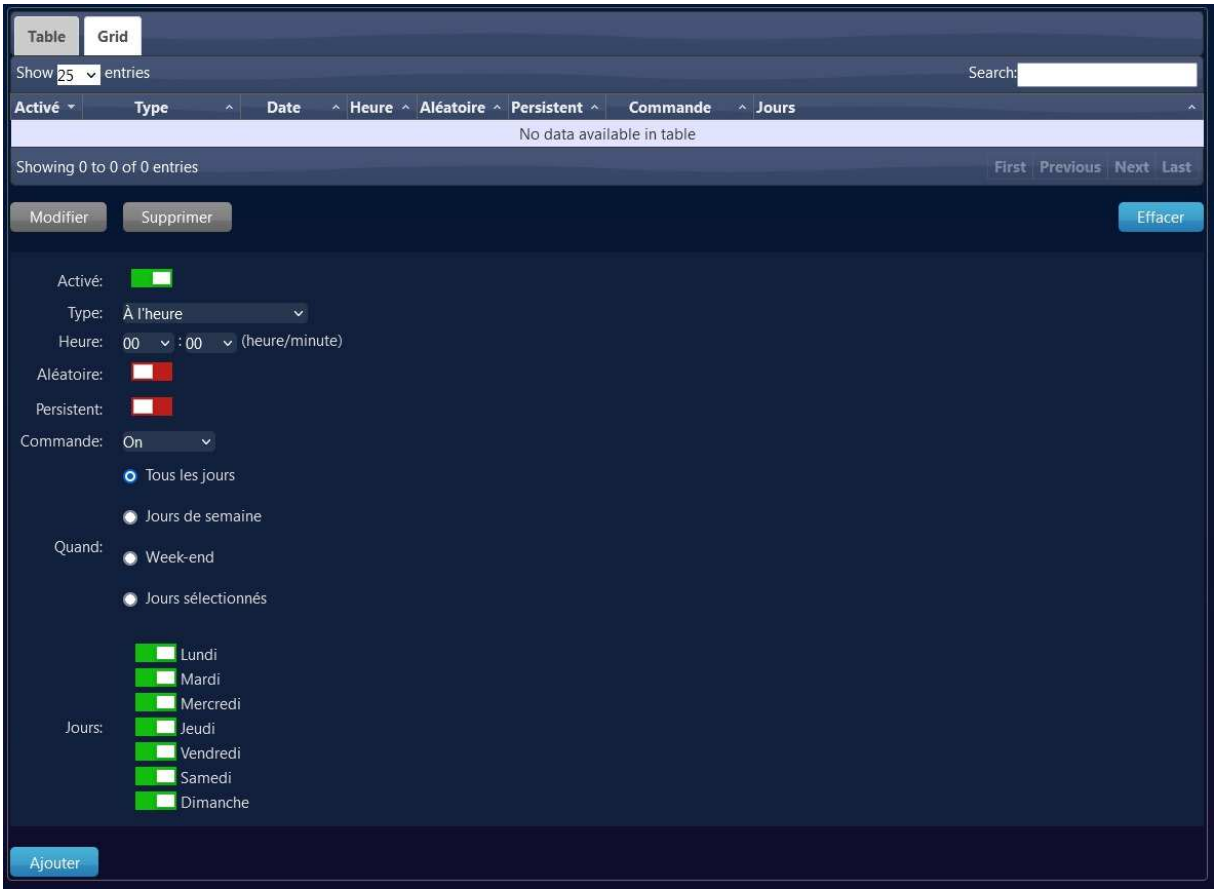
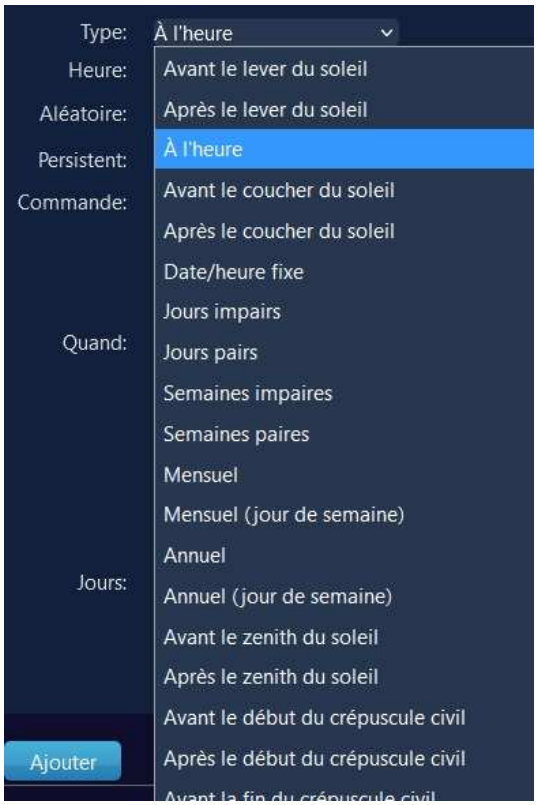


Table
Grid
Show 25 entries
Search:
Activé
Type
Date
Heure
Aléatoire
Persistent
Commande
Jours
No data available in table
Showing 0 to 0 of 0 entries
First
Previous
Next
Last
Modifier
Supprimer
Effacer
Activé:
Type:
À l'heure
Heure:
00
: 00
(heure/minute)
Aléatoire:
Persistent:
Commande:
On
Quand:
Tous les jours
Jours de semaine
Week-end
Jours sélectionnés
Jours:
Lundi
Mardi
Mercredi
Jeudi
Vendredi
Samedi
Dimanche
Ajouter

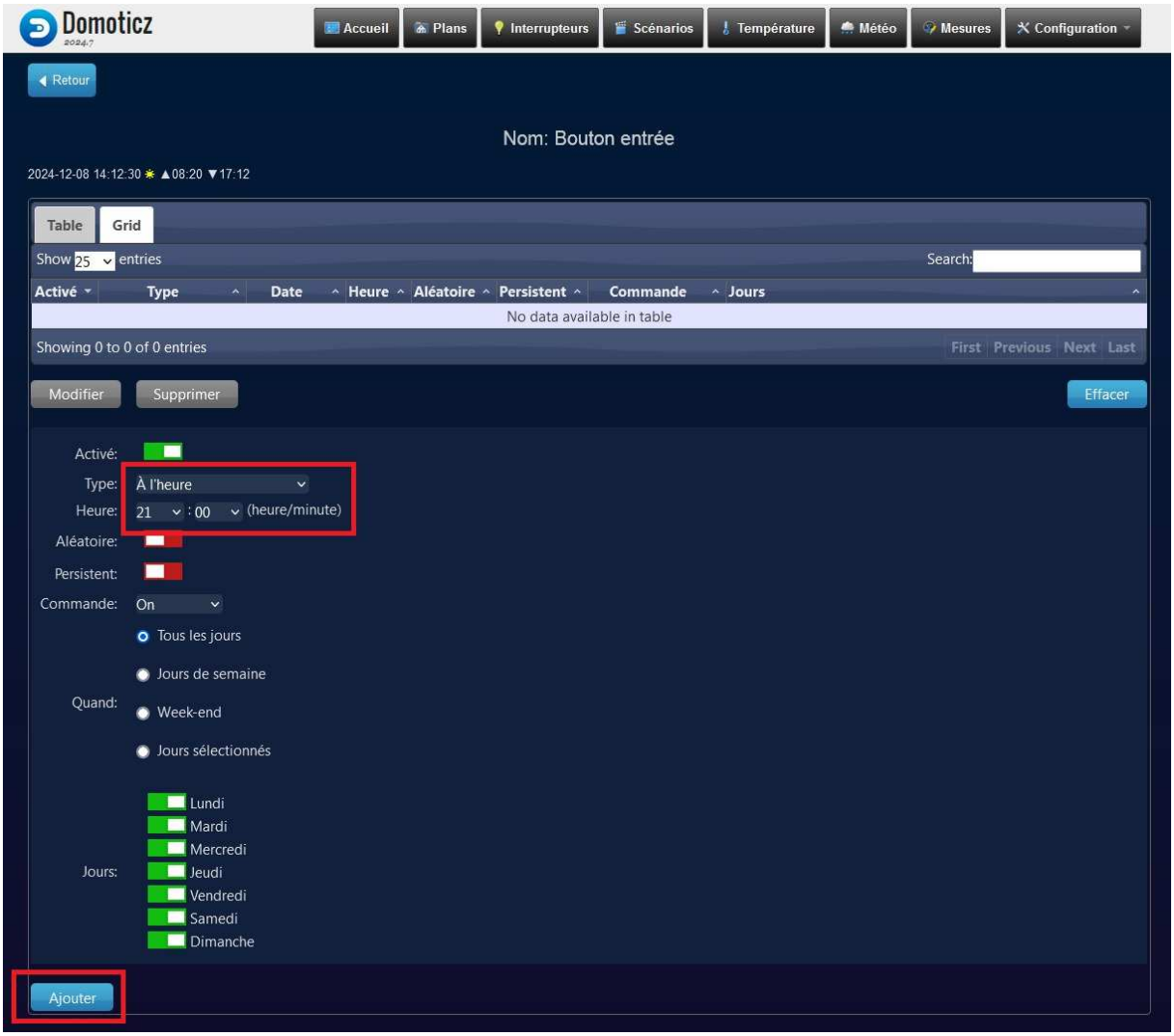
CRÉATION D'UN PLANNING AVEC UNE INTERFACE TEXTE

On peut l'utiliser pour spécifier les modifications de l'état du dispositif manuellement. Dans ce cas, on peut choisir le type d'évènement parmi la liste (partielle) suivante :

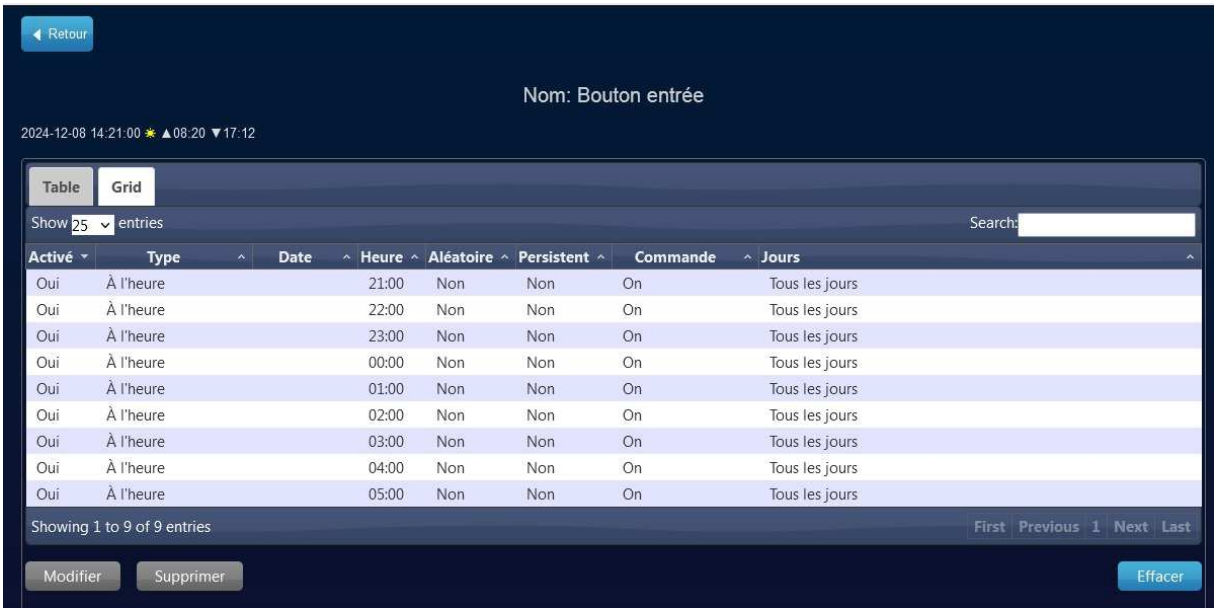


Dans l'exemple suivant, on va générer une commande « Off » chaque heure, entre 21h et 5h du matin.

Pour le faire, on va choisir le type « A l'heure », choisir « 21 » comme heure, et cliquer sur « Ajouter » :



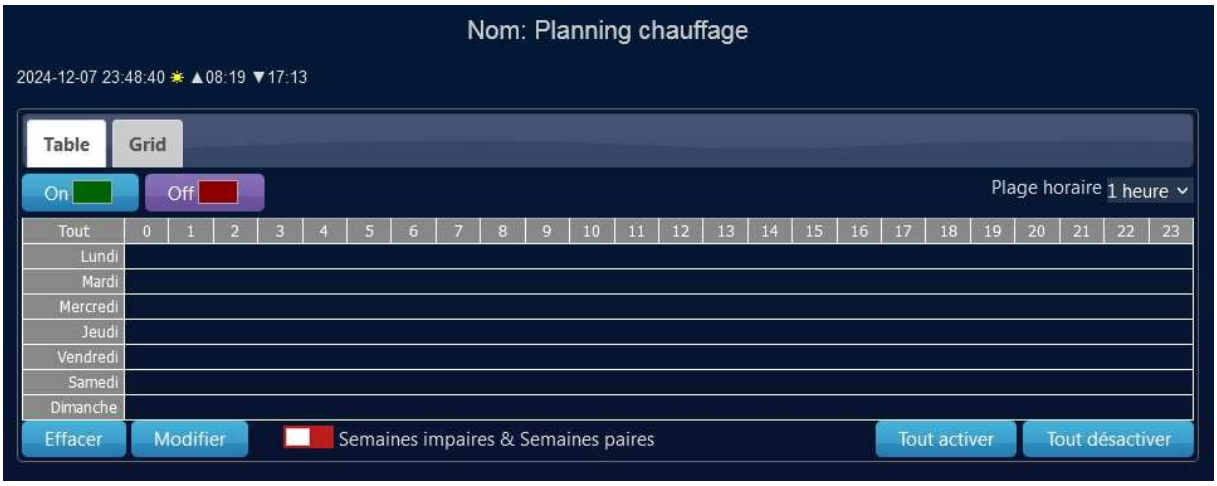
Répéter l'opération avec 22, 23, 00 …, 05. On doit arriver à :



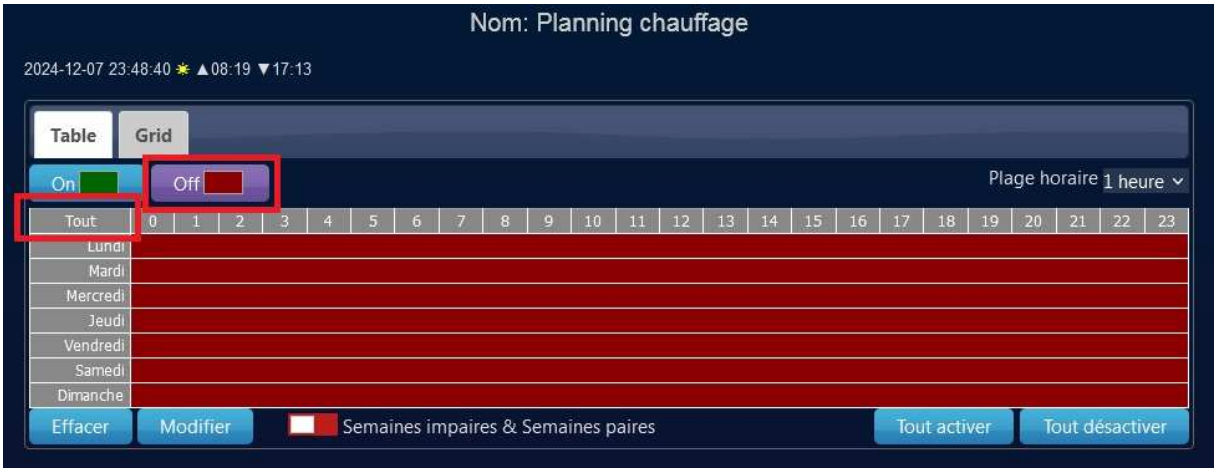
A partir de maintenant, le bouton sera remis à zéro toutes les heures de 21h à 5h du matin !

CRÉATION D'UN PLANNING AVEC UNE INTERFACE GRAPHIQUE

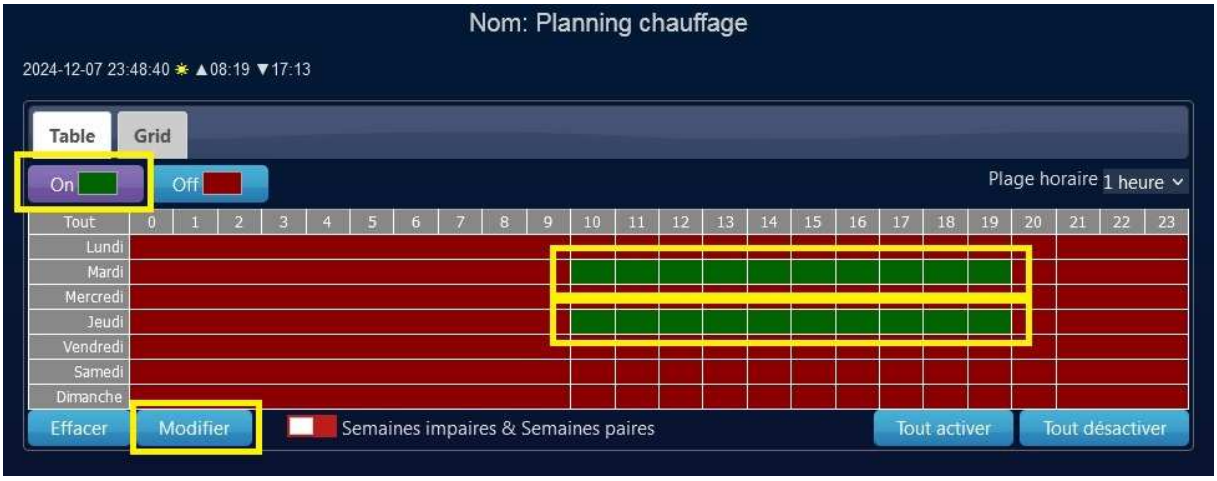
, en partant de la fenêtre initiale, on peut créer le planning de façon graphique en cliquant sur « Grid » en haut à gauche, on obtient la fenêtre suivante :



On commence par définir un mode « off » sur la globalité du planning, en cliquant sur « Off » puis « Tout ». Ça donne ça :



Dans cette exemple, on veut définir le dispositif sur « on » de 10h à 19h les mardis et jeudis. Pour ça, on clique sur « On » et on sélectionne de 10h à 19h le mardi, et de 10h à 19h le jeudi. On valide ces modifs en cliquant sur « Modifier ».



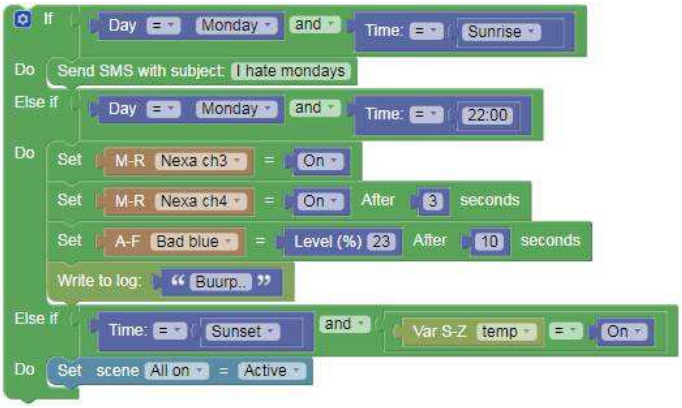
On peut (mais ce n'est pas obligé) voir ce qui a été généré en cliquant sur « Table » :

On peut revenir à la page initiale en cliquant sur « Retour ».

CRÉER UN SCRIPT LUA DOMOTICZ

Domoticz possède 3 types de scripts :

- Les scripts de type « Blocky », dans laquelle on n'écrit pas de code, mais une programmation graphique dans laquelle on déplace des boîtes qui réalisent certaines fonctions. L'interface ressemble à ça :



- Les scripts de type « LUA classique », où l'échange de données avec Domoticz se fait au travers de quelques tables. Le code ressemble à :

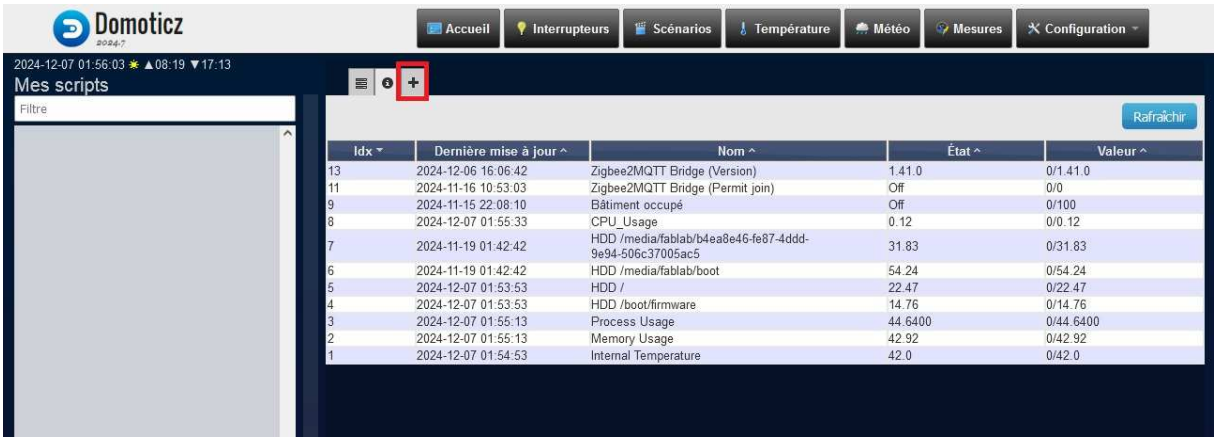
```
-- Trace device changes to Domoticz's log
commandArray = {}
-- loop through all the changed devices
for deviceName,deviceValue in pairs(devicechanged) do
    if otherdevices[deviceName] ~= nil then
        print ("Event fired on '"..deviceName.."', value '"..tostring(deviceValue).."'");
    end
end
return commandArray
```

- Les scripts de type « dzVents », également basés sur LUA, où Domoticz est très intégré au langage LUA, mais probablement un peu plus compliqué à utiliser, en raison du plus grand nombre de fonctions disponibles. Le code ressemble à :

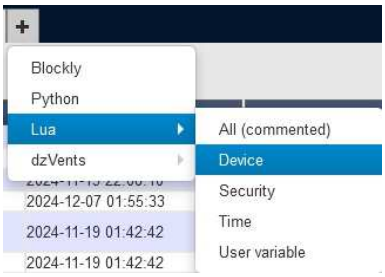
```
-- Compute pressure delta over one hour
return {
    on={
        devices={"Pression atmosphérique"}
    },
    data={
        pressureHistory={history=true, maxHour=1} -- Pressure history (max 1 hour)
        },
    execute = function(domoticz, item)
        local pressure, status = item.sValue:match("([^;]+);([^;]+)") -- Split using ';'
        domoticz.data.pressureHistory.add(tonumber(pressure))
        local deltaPressure = domoticz.data.pressureHistory.deltaSinceOrOldest("01:00:00")
        domoticz.devices("Delta pression 1 heure").setValues(0,tostring(deltaPressure))
    end
}
```

Dans un premier temps, nous allons utiliser des scripts LUA « classiques », plus simples à comprendre et à utiliser. On utilisera plus tard des scripts « dzVents », quand le besoin de fonctions avancées sera là.

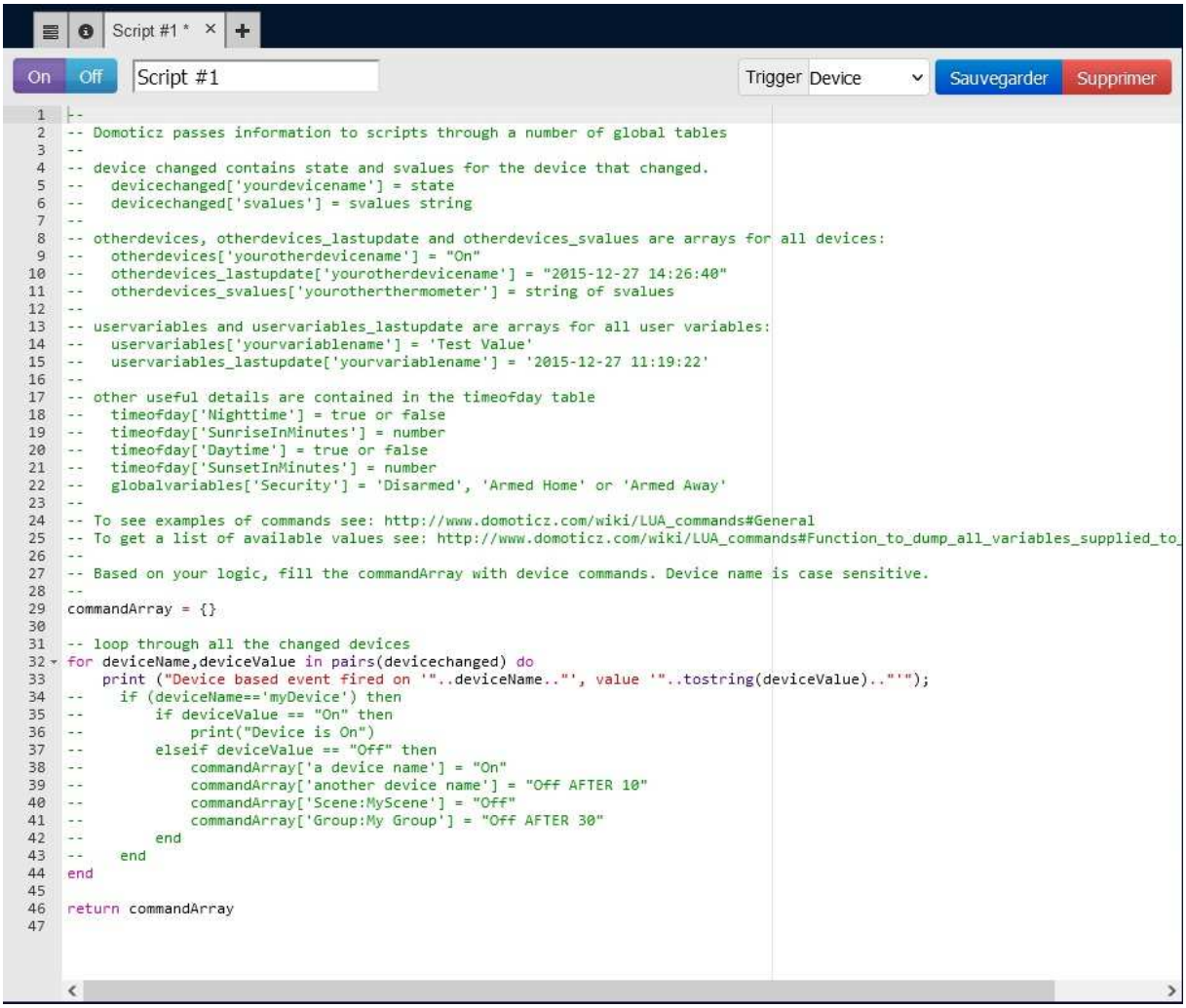
Pour créer un nouveau script LUA « classique », on utilise le menu « Configuration » / « Plus d'options » / « Évènements », qui ouvre une feêtre comme :



Cliquer sur le bouton « + » (en rouge sur l »image précédente. La liste déroulante suivante s'ouvre :



Domoticz ouvre une fenêtre avec un exemple commenté :



La doc complète sur LUA (en Anglais) se trouve à https://wiki.domoticz.com/LUA_commands

Domoticz va activer les scripts sur des évènements (changement d'un dispositif ou d'une variable, timer, …). En indiquant « Device » lors de la création, on indique qu'on s'intéresse aux dispositifs uniquement. On sera donc activé à chaque changement d'état d'un dispositif.

On comprend rapidement que les commentaires sont précédés de « -- ». La première ligne est affichée en cas d'erreur, il est donc habile de mettre un truc significatif afin de pouvoir retrouver le script. Le nom du script est une bonne idée ...

Le principe est de charger la table « commandArray » avec les commandes à passer : par exemple, pour allumer la « Lampe du séjour », on écrit :

```
commandArray['Lampe du séjour'] = 'On'
```

On obtient la liste des dispositifs qui ont déclenchés l'appel du script dans « deviceschanged » par une boucle comme :

```
for deviceName,deviceValue in pairs(devicechanged) do
       <<ici, on dispose de « deviceName » qui contient le nom du dispositif
       et « deviceValue » sa valeur>>
end
```

On peut par exemple écrire dans le log de Domoticz chaque modification de dispositif avec le code suivant :

```
for deviceName,deviceValue in pairs(devicechanged) do
 print ("Changement : '"..deviceName.."' =  '"..tostring(deviceValue).."'");
end
```

Pour enregistrer le script, lui donner un nom significatif (par exemple « Trace changements ») à la place de « Script #1 », et cliquer sur « Sauvegarder ».

On voit apparaître le nom du script dans la liste sur la gauche.

Cliquer dessus pour l'afficher dans la fenêtre de droite.

Ne pas oublier de cliquer sur « Sauvegarder » à la fin de la modif, à moins qu'on ait vraiment envie de refaire le boulot une seconde fois ...

Voilà, vous savez créer un script LUA

Note : il peut être habile de conserver cet exemple, assez partique pour savoir ce qui s'est passé, et voir quelles sont les valeurs qui sont envoyées lors d'un changement d'état.

TESTER DU CODE LUA

On peut vouloir tester une partie de code LUA.

Une façon simple de le faire est de l'inclure dans un script « Test » dont le déclencheur est l'appui sur un bouton virtuel « Test ».

Inclure dans ce script la partie de code qu'on souhaite tester.

Par exemple :

```
commandArray = {}

for deviceName,deviceValue in pairs(devicechanged) do
    if (deviceName=='Test') then
        <<Mettre ici le code à tester ou les actions à exécuter>>
    end
end

return commandArray
```

On peut aussi utiliser cette technique pour modifier/définir l'état ou le contenu de dispositifs ou de variables.