Sensor Alarm Controller

Voici une solution pour réaliser un système d'activation/désactivation d'une alarme « maison » à partir d'un nodeMCU d'un côté et d'un serveur Domoticz. Le second article expliquera comment contrôler à distance le serveur Domoticz sans ouvrir de port en Nat sur la boxe Internet (solution que je trouve périlleuse en termes de sécurité).

Introduction

Je souhaitais doter ma maison d'un système d'alarme basé sur mon installation Domoticz et facilement activable/désactivable par mes enfants. Le panel de sécurité de Domoticz est bien sympa mais :

- Il faut trouver une tablette, un écran disponible facilement pour désactiver l'alarme avec un code,
- L'usage d'un code n'est pas forcément idéal pour des enfants,
- En plus je n'ai jamais réussi à contourner le problème « Browser Cache refresh failed » de domoticz, et la manip d'effacement du cache n'est pas très user friendly surtout quand une alarme de 120 dB est en train de sonner…

Du coup, j'ai décidé de partir sur une solution basé sur la reconnaissance de clés sans contact RFID. Le lecteur RFID est installé le mur et avec domoticz via le Wifi.

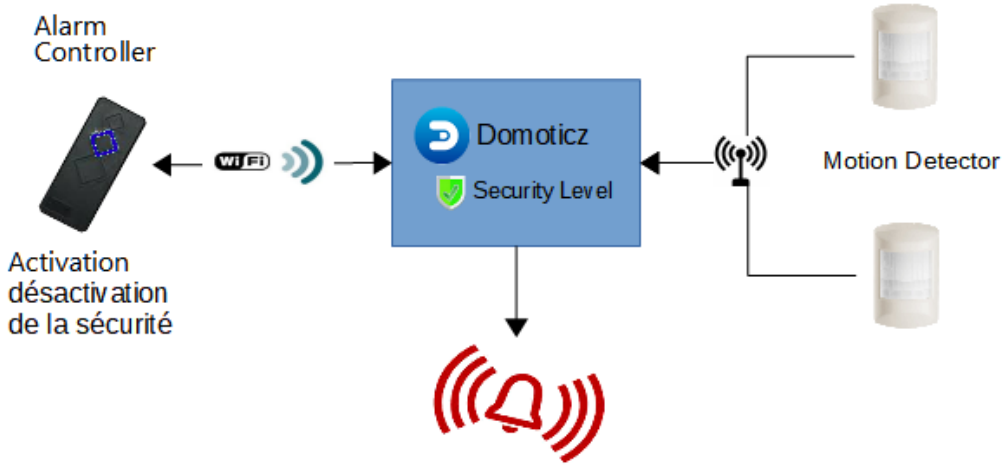
Attention : cette alarme n'a pas la prétention de rivaliser avec un système professionnel. Son objectif est de faire fuir le « rodeur de base » et de prévenir le propriétaire qu'un incident est en cours.

L'ensemble du code est disponible sur github : https://github.com/Hagrou/AlarmController

Principe de la solution

Le cœur de la solution est basé sur un serveur domoticz installé sur un Raspi. Sur ce serveur, un « Virtual Sensor *Security Level* » a été créé afin de définir l'état courant du système de sécurité (désarmé, activé, alarme, etc.). Cet état peut être modifié par :

- Un ensemble de senseurs (détecteurs de mouvement par exemple) qui lorsqu'ils sont activés, modifient par l'intermédiaire d'un script Lua l'état du Security Level
- Un système (Alarm Controller) qui communique en Wifi avec le serveur Domoticz et permet l'activation ou la désactivation d'un Virtual Sensor « Security Level » créé sur Domoticz
- Enfin, Domoticz peut demander l'activation d'une alarme si nécessaire.



Conception du système « Security Controller »

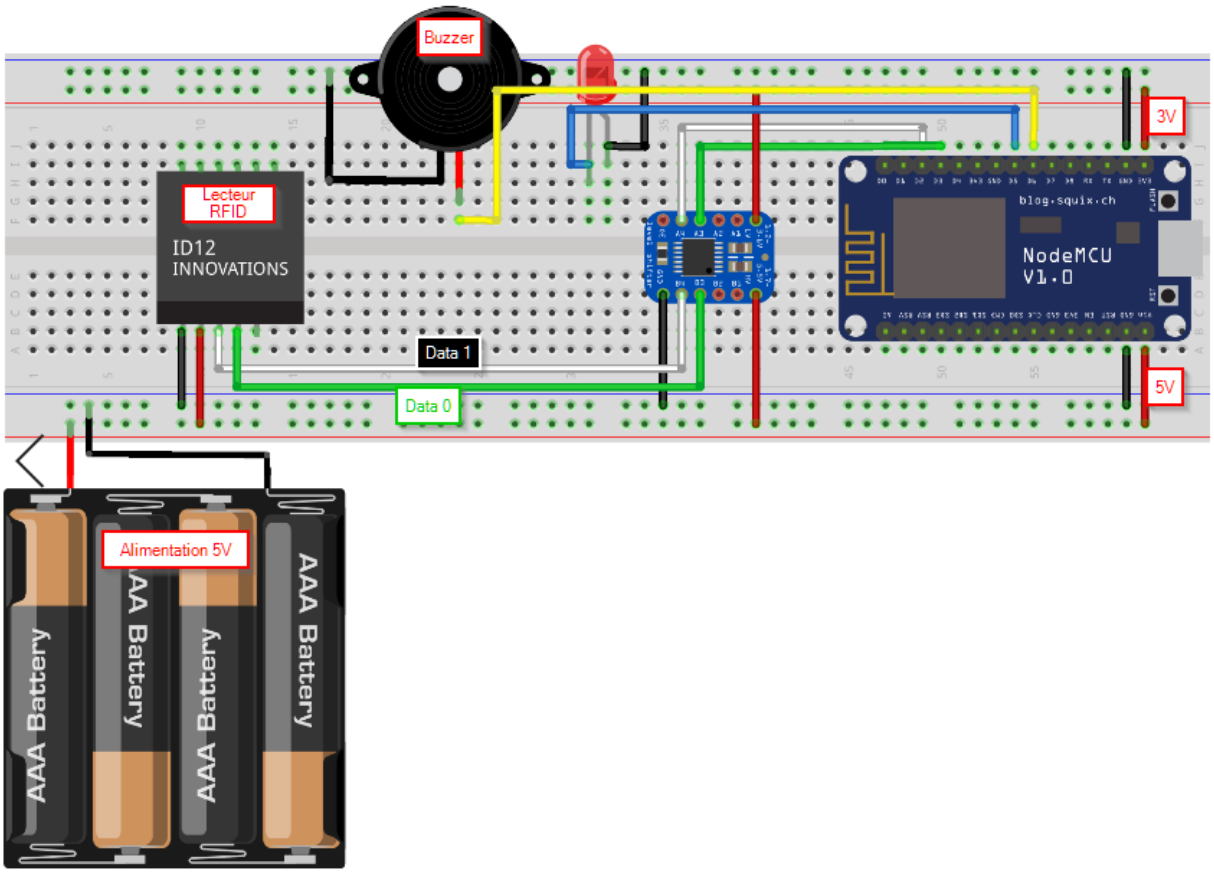
Ce système a pour fonction l'activation et la désactivation de l'alarme sur présentation d'une clé RFID. Il doit également être capable de présenter l'état du système de sécurité ce qui nécessite une synchronisation avec l'état du Security Level. Coté communication, ne connaissant pas bien les différents protocoles radios disponibles, j'ai tout simplement choisi d'utiliser le Wifi ; il est naturellement protégé par le WPA/2 et permet d'interagir directement avec Domoticz via son interface http REST et surtout il est parfaitement supporté avec l'ESP8266 !

Circuiterie

Ce système est donc basé autour de :

- Un microcontrôleur capable de faire du Wifi (j'ai choisi un NodeMCU ESP8266 v3)
- Un lecteur de TAG RFID
- Une led et un Buzzer pour permet d'indiquer l'état de l'alarme
- Un Bi-Directional Logic Level Shifter Converter 3.3V 5V pour interfacer la différence de tension entre les signaux du lecteur de TAG RFID (5V) et le NodeMCU (3.3V)
- Une alimentation stabilisée 220V AC -> 5V CC pour alimenter le tout (j'ai recyclé une alimentation de veilleuse Bébé à Led IKEA)

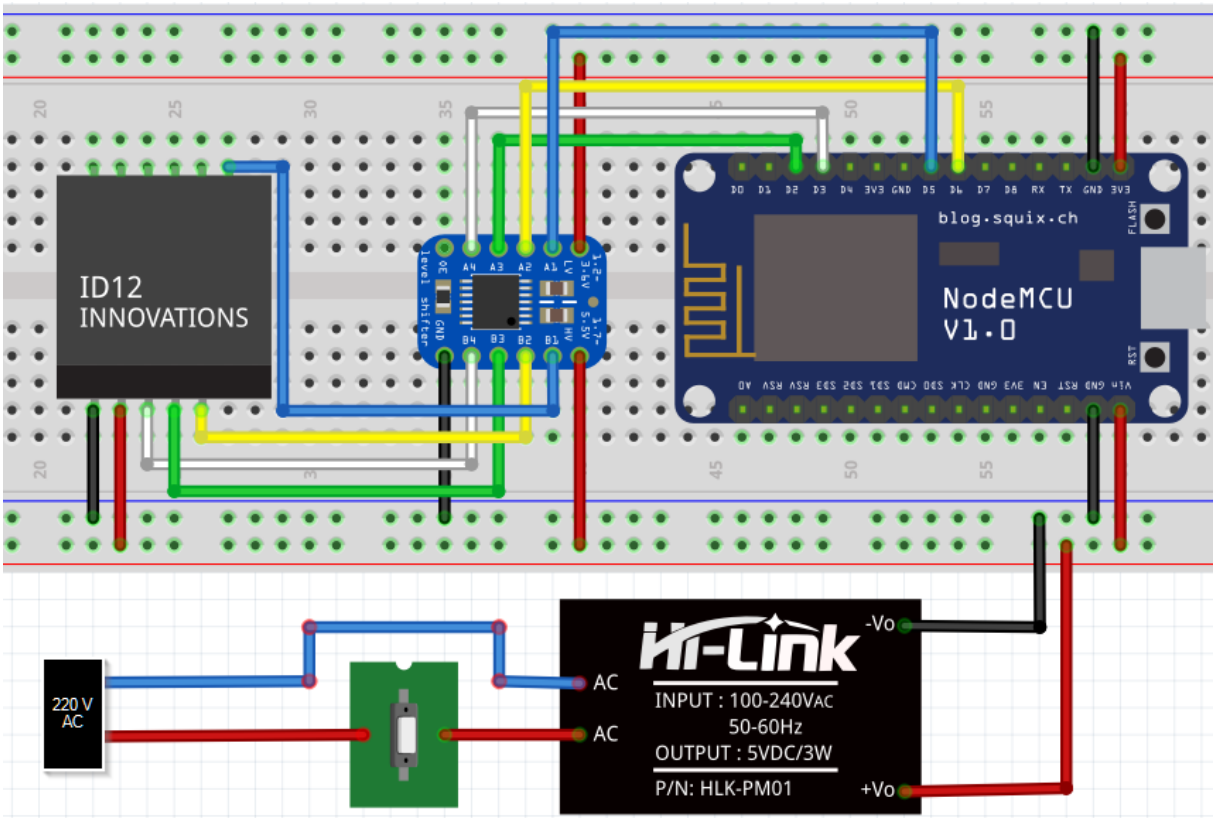
Schématiquement cela donne un truc comme ça :



Voici la liste des composants que j'ai acheté pour réaliser la partie physique du controller :

Composant	Prix indicatif
5x 4-channel Bi-Directional Logic Level Shifter Converter 3.3V 5V Arduino TE291	4,17 €
ID Wiegand 26 Lecteur de Cartes RFID Sécurité Accès Porte Digicode Etanche Noir	10,53 €
NodeMcu ESP8266 v3 Lua CH340 WiFi Internet Development Board modul rP	1,95 €
10 TAG RFID	5,00 €
Une alimentation recyclée d'une veilleuse à Led Ikea	0,00 €
Câble de lampe + Interrupteur recyclés	0,00 €
Porte fusible + fusible 220 mA recyclés	0,00 €

Au départ j'étais parti sur l'achat d'un simple lecteur de tag RFID. Finalement j'ai opté pour un lecteur de carte étanche et intégrant directement une led bi-color (vert et bleu) ainsi qu'un buzzer. Le principe du schéma ne change pas, j'ai juste utilisé l'ensemble des ports du Logic Shifter Converter pour convertir l'ensemble des signaux (Data0, Data1, Buzzer, Led) en 5V depuis le NodeMCU :



J'ai également ajouté un fusible 250mA ainsi qu'un interrupteur ce qui me permet de rebooter le système au cas où. Une fois le circuit soudé et monté ça donne ca :

Programmation

Initialement je pensais utiliser le langage LUA pour programmer le NodeMCU. C'est un langage assez simple que je connais bien et qui et me permettait de faire du code homogène avec les scripts Domoticz. Malheureusement, le composant ainsi programmé était instable et n'arrêtait pas de rebooter !? :/. Du coup je suis parti sur du classique, du C/C++ façon Arduino.

Le fonctionnement attendu du logiciel est :

1. A l'initialisation se connecter au Wifi : pendant cette phase la led clignote rapidement jusqu'à la réalisation de la connexion (ainsi, AlarmController obtient également une adresse IP)

2. Récupérer l'état du status du device SecurityLevel sur domoticz via l'API Rest
3. Lire le device RFID pour voir si une clé n'a pas été lue
4. Recommencer à l'étape 2.

Un micro serveur Web est également implémenté (cf. Figure 1) :

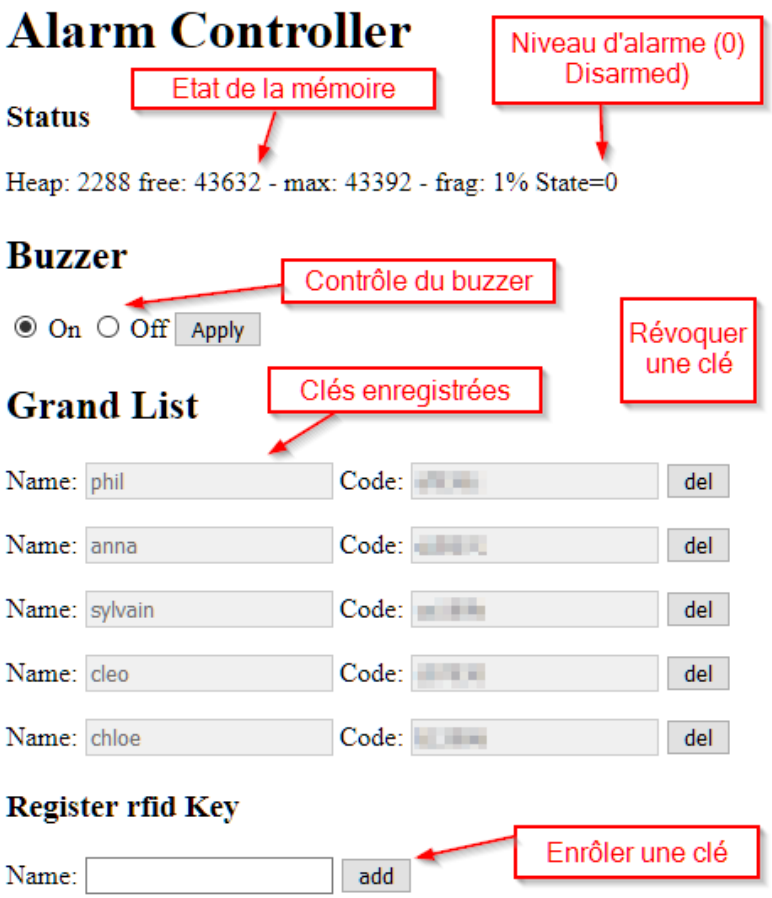


Figure 1- Page Web de Alarm Controller

Il permet de contrôler le comportement du système. On peut ainsi :

- Avoir l'état du composant AlarmController,
- Activer/désactiver le Buzzer
- Voir la liste des clés RFID enrôlées et en supprimer,
- Enrôler une nouvelle clé.

Enrôlement des clés

L'enrôlement d'une clé nécessite d'associer un nom avec la clé puis d'appuyer sur le bouton « add ». A ce moment-là, le buzzer de l'alarm controller se met à biper rapidement. Il faut passer la nouvelle clé sur le lecteur et voila !

Note : Les clés sont enregistrées dans une zone mémoire permanente du nodeMCU. Ainsi, même après une coupure de courant, l'ensemble des clés seront toujours disponibles.

Note : le code est très simpliste… je ne suis pas certain que les caractères accentués soient gérés…

Configuration

Le fichier config.h contient l'ensemble des valeurs à configurer. On y trouve :

- Une partie concernant la connexion entre le lecteur RDFI et le NodeMCU

```
#define PIN_DATA0             D2              // GREEN : DATA0 weigand
#define PIN_DATA1             D3              // WHITE : DATA1 Weigand
#define PIN_BUZZER            D6
#define PIN_LED               D5              // BLUE: LED
#define COM_SPEED             115200          // Serial Com Speed for Debugging
#define GRANT_TAG_SIZE        16               // max key name size
#define GRANT_MAX_SIZE        32               // max registered keys
#define WWW_USERNAME          "#adminName#"   // admin authentication
#define WWW_PASSWORD          "#adminPwd#"
```

- Et une partie concernant l'interface avec Domoticz (nous allons revenir dessus)

```
#define   DOMOTICZ_LONG_DELAY  10*1000 // refresh state delay (ms)
#define   DOMOTICZ_SHORT_DELAY 1*1000

#define WIFI_SSID             "YourWIFISSID"
#define WIFI_PASSWORD         "YouWIFIPassword"

#define DOMOTICZ_URL          "http://192.168.0.10:8080
#define SECURITY_LEVEL_IDX    "17"   // domoticz security level device
#define ALARM_IDX             "18"   // Alarm idx
#define HTTP_SERVER_PORT      80
```

Vous pouvez modifier vous-même ce fichier de configuration, ou utiliser le script python genConfig.py qui configure l'ensemble des codes nécessaires à ce projet depuis un fichier unique Yaml.

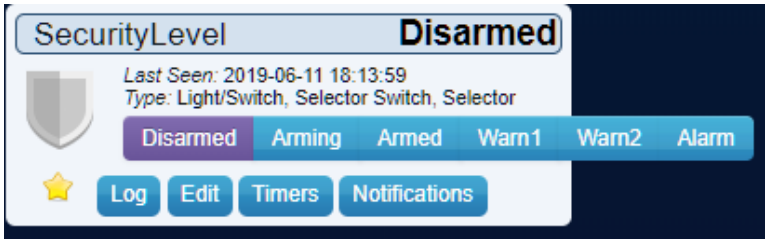
Note : pour les curieux qui vont aller regarder le code, je n'ai pas utilisé de librairie JSON pour parser le résultat du web service Domoticz.. Ce n'est pas par snobisme, mais tout simplement parce que la librairie Arduino JSON demandait trop de mémoire pour parser le résultat (où je n'ai pas trouvé les bonnes options). Du coup, je suis allé extraire les info. intéressantes à la mimine.

Note : pour la lecture RFID, j'ai utilisé la librairie « Wiegand-Protocol-Library-for-Arduino-master » que j'ai directement intégré au code source afin d'éviter d'avoir des problèmes de dépendances en cas de mise à jour de l'environnement Arduino (c'est du vécu).

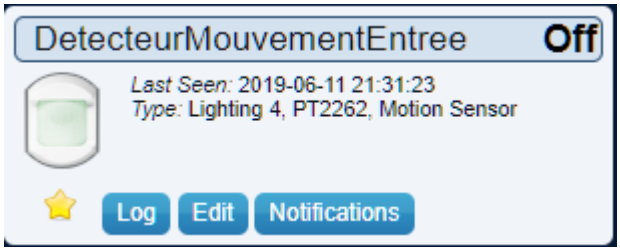
Intégration avec Domoticz

Du coté Domoticz, au moins trois devices permettent de gérer ce système d'alarme :

- Le device principal, « SecurityLevel » gère l'état de l'alarme et est représenté par un Virtual Device « Selector Switch » ce qui permet de gérer autant d'état que je souhaite (16 états maximums)



- Un ou plusieurs device de détection,

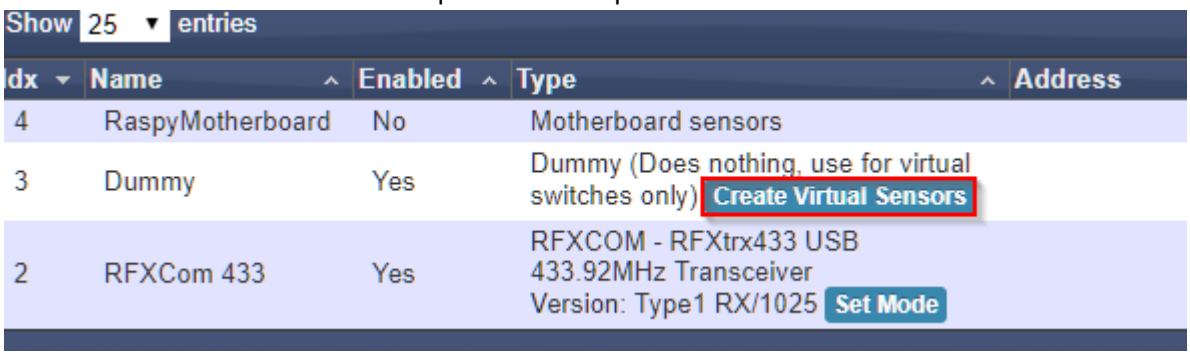


- Un device « Alarme » qui l'état de l'alarme (activé ou non)

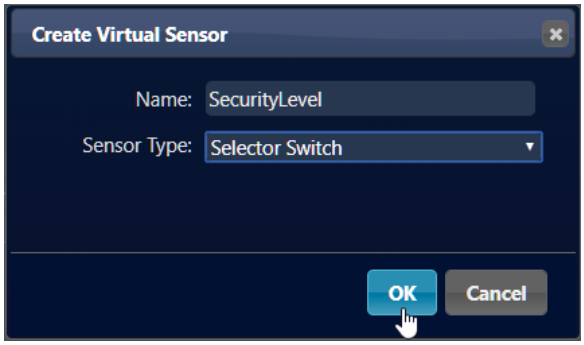
Création du device « Security Level »

Pour créer ce device, il suffit de :

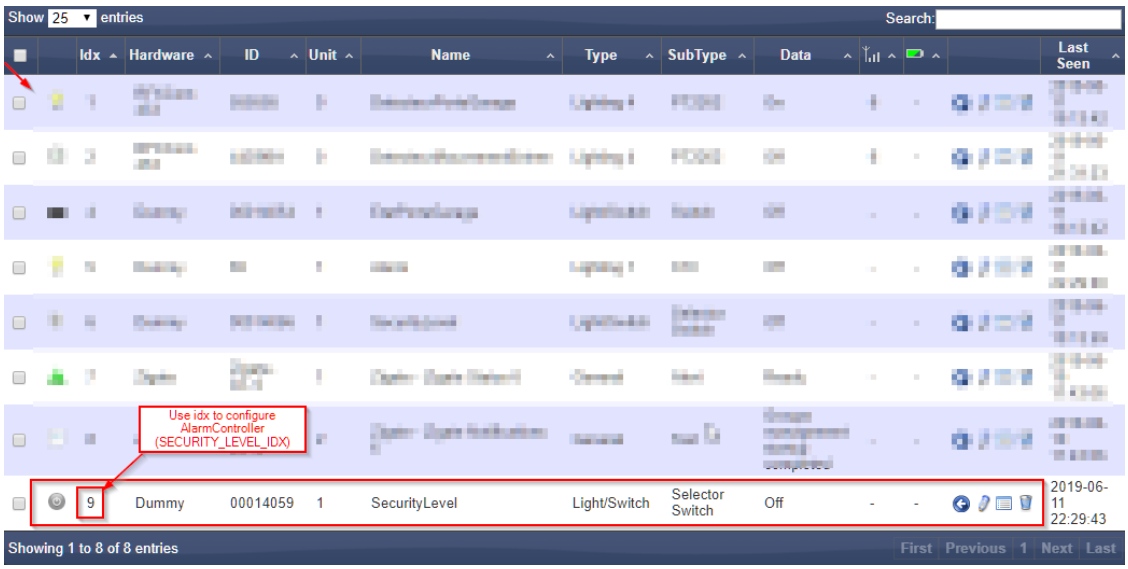
1. Allez dans le menu Domoticz -> Setup -> Hardware
2. Si un device virtuel n'a pas encor été créé, sélectionner « Type » Dummy ainsi qu'un nom de device puis clicker sur Add.
3. Allez dans le menu Domoticz -> Setup -> Hardware puis clickez sur « Create Virtual Sensors »



4. Sélectionnez «Selector Switch » pour le type de sensor, puis validez

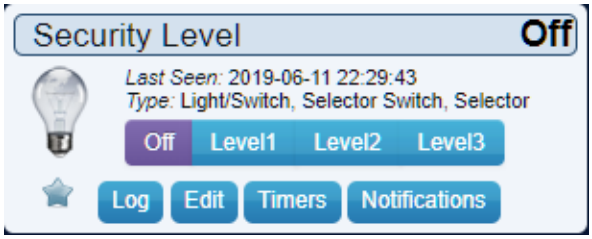


5. Dans le menu Domoticz -> Setup -> Device, vous allez trouver le device ainsi créé :

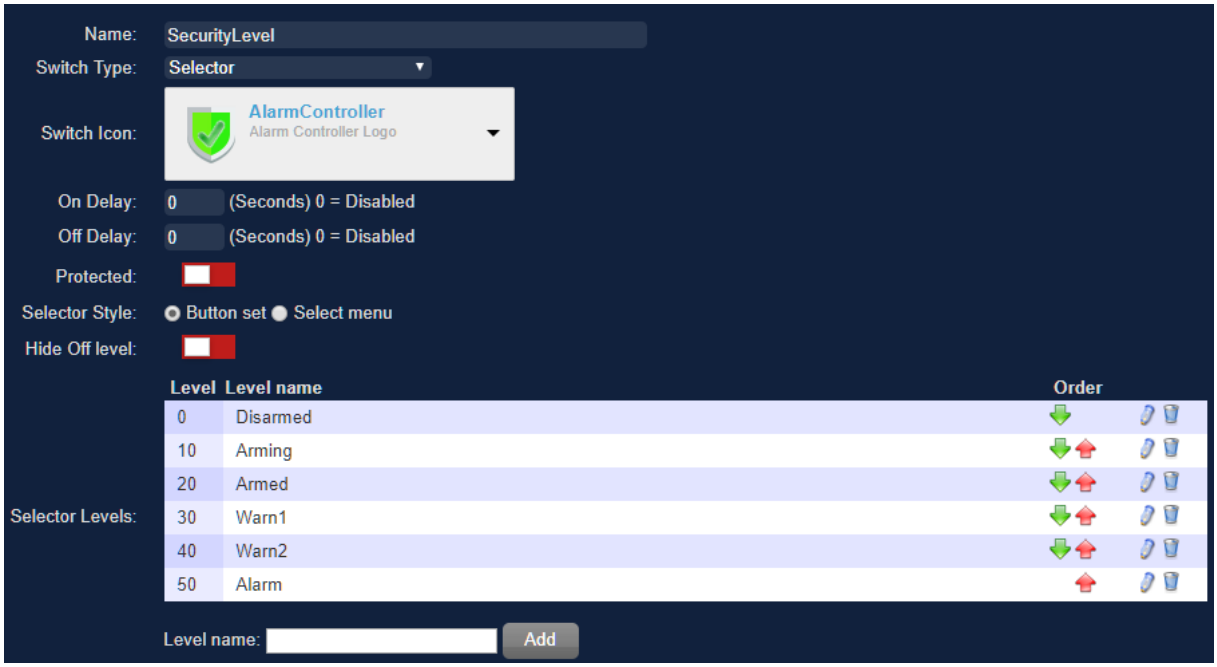


(Notez bien son Idx afin de configurer la variable SECURITY_LEVEL_IDX du sensor AlarmController).

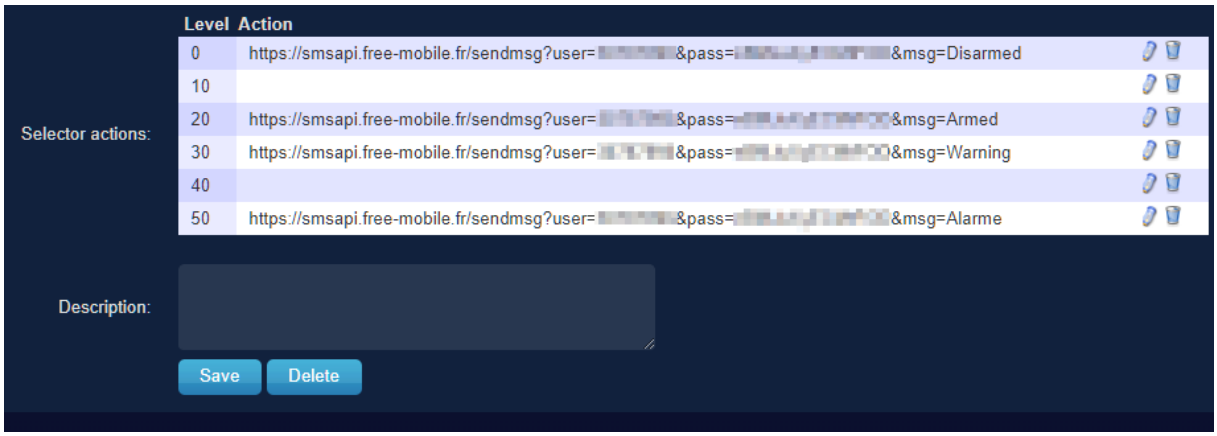
6. Dans le menu Domoticz -> Switches vous pouvez voir votre superbe Switch « Security Level ».



7. Nous allons maintenant le configurer. Pour cela, clickez sur « Edit » puis ajoutez les états « Disarmed », « Arming », « Armed », « Warn1 », « Warn2 » et « Alarm » en respectant bien les valeurs de Level.



8. Optionnellement, vous pouvez également ajouter des actions en fonction du passage de l'état. Pour ma part, le device m'envoie un SMS sur les moments importants.

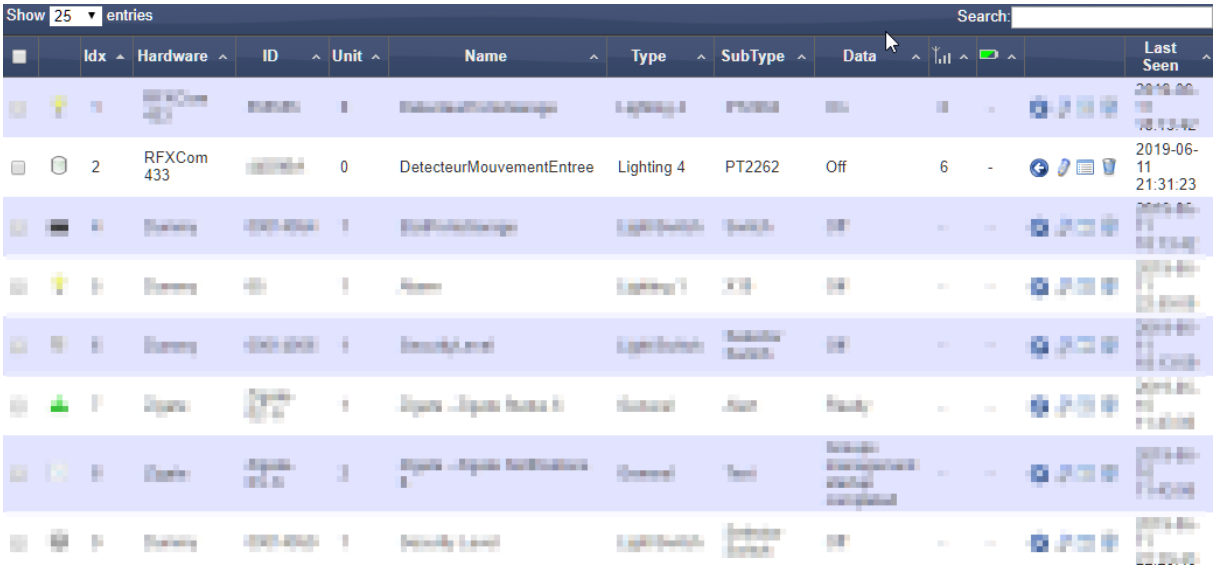


9. Enfin n'oubliez pas de sauvegarder vos modifications (bouton « Save »).

Note : comme vous pouvez le voir sur la capture d'écran, je me suis amusé à créer des icônes pour ce device. Optimiste, j'avais prévu 6 icones différents pour chacun des états possibles, malheureusement je n'ai pas trouvé comme les faire avaler à Domoticz qui ne semble gérer que 2 états possibles. Quoi qu'il en soit, vous pourrez trouver les icones dans le git dans « icones ».

Création des devices « Motion Detection »

Ce sont l'ensemble des devices qui vont permettre la détection d'une intrusion. Créez ces devices à partir de vos récepteurs « Hardwares ». Pour ce tuto, j'ai utilisé un détecteur de mouvement radio en 433 MHz.



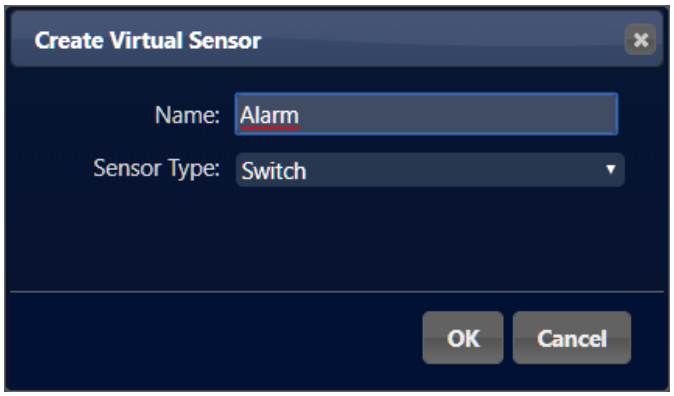
Une fois, ce device créé, allez dans l'onglet Domoticz -> Switches, puis clickez sur le bouton Edit de votre Device.

J'ai choisi de définir un temps d'activation de 1 seconde afin laisser le temps à Domoticz de réagir (je n'ai pas d'avis particulier sur le sujet).

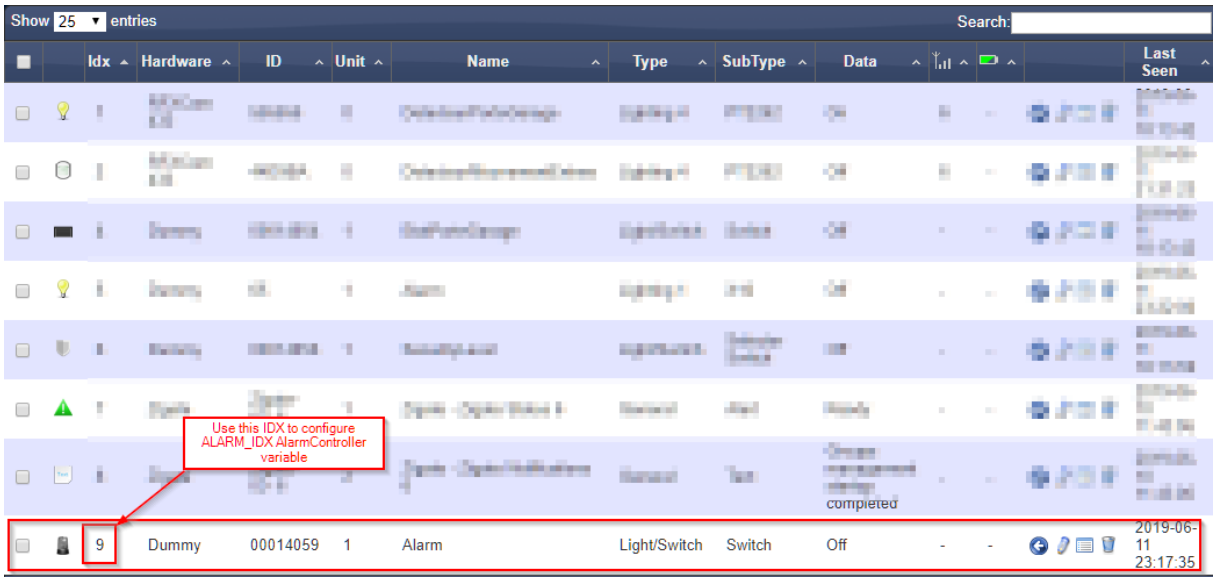
Création des devices « Alarm »

Afin de représenter le fonctionnement de l'Alarm, j'ai ajouté un device virtuel « Switch ». Pour créer ce device :

1. Allez dans le menu Domoticz -> Setup -> Hardware, puis clickez sur « Create Virtual Sensors », puis créez un Switch.



2. Allez ensuite dans le menu Domoticz -> Setup -> Device, puis utilisez son IDX pour configurer la variable de ALARM_IDX dans AlarmController.



Note : pour le moment cette Alarm est purement fictive. L'ensemble du système repose sur une alarme silencieuse à base de SMS (cf. SecurityLevel).

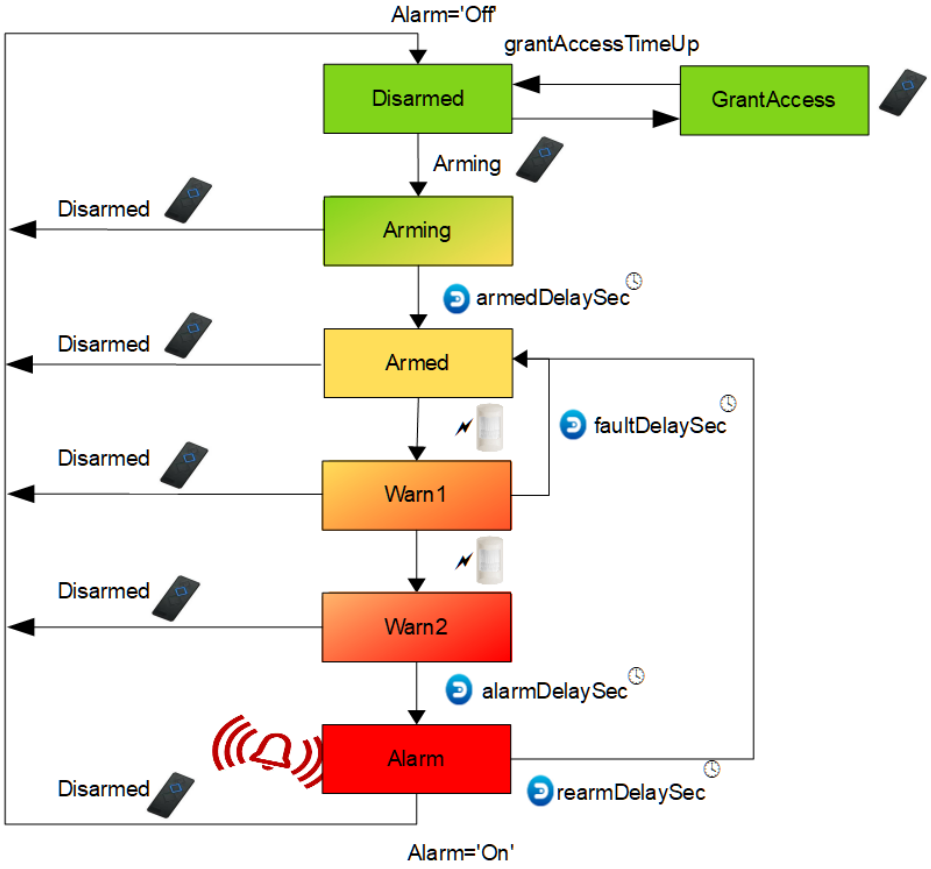
Note : l'utilisation de ALARM_IDX permet de stopper immédiatement l'alarme par un appel au Web Service de Domoticz.

Fonctionnement du l'automate de gestion des alarmes

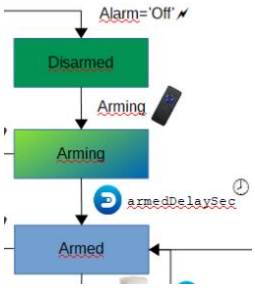
Contrairement au système de sécurité de domoticz, notre système repose sur 7 états différents pour gérer :

- L'état de désarmement (Disarmed)
- L'enrôlement de clés (GrantAccess)
- Le passage de l'armement à un système d'alarme Armé (Arming -> Armed)
- La détection d'une intrusion et le passage à Alarm (Armed -> Warn1 -> Warn2->Alarm)

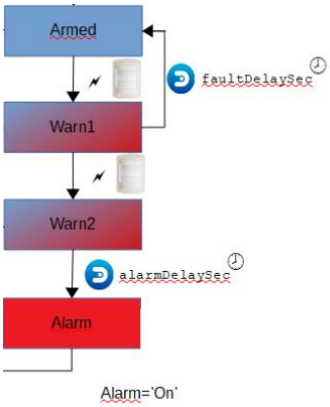
Tout ceci peut être représenté par l'automate suivant :



Cet automate est un peu compliquer du coup voici quelques explications :

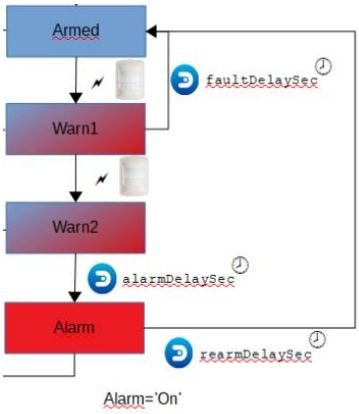


- La première partie de l'automate concerne la phase « Armement ». Initialement, l'automate est désarmé (Disarmed). Si on active le lecteur RFID avec une clé valide, l'état passe en cours d'armement (Arming). Une tempo (armedDelaySec) permet ensuite de quitter le lieu sans déclencher l'alarme. A la fin de cette tempo, le système est armé (Armed).



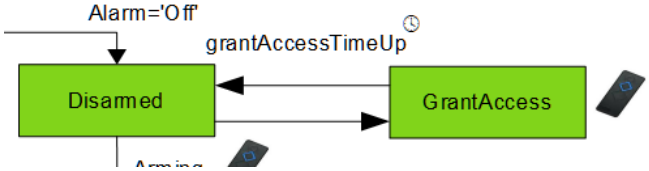
- Afin de filtrer d'éventuelles fausses détections (parasite radio, changement de température, etc.) le système nécessite 2 détections successives pour passer à l'état « Alarm ». Ainsi :
  - Sur une première détection, le système passe ne « Warn1 ». Si au bout d'un temps « faultDelaySec » une seconde détection n'a pas eu lieu, alors le système repasse en « Armed »
  - Si par contre, une deuxième détection a lieu, alors le système passe en « Warn2 ». Il reste encore un petit laps de temps « alarmDelaySec » pour désactiver l'alarme et repasser en « Disarmed ».
  - Dans le cas contraire, l'alarme est donnée.

- Si l'alarme a été déclenchée, alors le système repasse en mode « Armed » au bout d'un temps « rearmDelaySec ».



- à chaque instant il est possible de désactiver l'alarme, c'est pourquoi chaque état possède une transition vers l'état « Disarmed »

- Enfin, l'état d'enrôlement qui est automatiquement abandonné après une période de temps « grandAcessTimeUp »



Scripts

Deux scripts Lua permettent d'exécuter les transistions de l'automate :

- Script_time_securityStatus.lua : ce script est appelé toutes les 60 secondes par Domoticz. Il permet de gérer toutes les transitions temporelles :
  - Arming -> Armed
  - Warn1 -> Armed
  - Warn2 -> Alarm

- Script_device_security2Warning.lua : ce script est appelé à chaque changement de valeur d'un device. Il contient une fonction « isSomethingAppening() » qui permet de filtrer les devices concernant la sécurité. C'est cette méthode qu'il faudra modifier pour y ajouter vos déctecteurs.

Il faut installer ces 2 scripts sur Domoticz dans le répertoire /home/pi/domoticz/script/lua

Note : avoir une activation de l'ensemble des script device_* par Domoticz à chaque changement d'état de device quel qu'il soit ne facilite pas la configuration et l'écriture d'un service générique de gestion des alertes de sécurité. En effet, si on passe en Warn1 suite à une détection et que dans le même intervalle de temps un autre device s'active, le script « Script_device_security2Warning.lua » se réactive, voit que le device « Motion Detector » est à « On » (il n'a pas eu le temps de se désactivé), alors on passe directement en Warn2 puis en Alarm *_* .Il aurait été plus simple de pouvoir appeler les scripts Lua depuis une action « On Action ». Malheureusement, j'ai l'impression qu'il n'est pas possible de faire du Lua dans ce champ. Pour les développeurs de Domoticz, il pourrait être intéressant de proposer un tag « luascript:/// » permettant d'appeler directement un script lua dans le contexte de Domoticz…

Configuration

Vous trouverez à la racine du dépôt git :

- Un fichier de configuration « config_test.yaml »
- Un script python « genConfig.py »

Le script python permet de propager l'ensemble des valeurs de configuration dans les différents codes sources. Il comprend :

```
{
    #AlarmController Sensor
        # Wifi configuration
        'wifiSSID'         : 'myWifiName',
        'wifiPassword'     : 'MyWifiPassword',

        # Other The Air update Configuration
        'otaHost'          : 'esp8266-alarmController',
        'otaPassword'      : 'MyOTAPassword',
        'otaPort'          : '8266',

        # Web Controller
        'adminName'        : 'admin',
        'adminPwd'         : 'MyHttpPassword',

        # Domoticz setting
        'domoticzURL'      : 'http://192.168.0.128:8080',
        'controllerServerPort': '80',
        'securityLevelIDX': '6',
        'alarmIDX'         : '5',

    #Domoticz Scripts
        'securityCtrlUrl' : 'http://192.168.0.130',
        'armedDelaySec'    : '30',  # delay for transition Arming->Armed
        'faultDelaySec'    : '30',  # delay to return Warn1->Armed
        'alarmDelaySec'    : '10',  # delay for transition Warn2->Alarm
        'rearmDelaySec'    : '300', # delay after Alarm->Armed
        'blindDelaySec'    : '10',  # delay of not taking into account the activation of
the same device
}
```

L'application de cette configuration sur le code source est réalisée en appelant le script python :

```
python.exe genConfig.py --conf=config_test.yaml --src_dir=. --build_dir=build

build_dir already exists, clean it

Copy ./alarmController/alarmController.ino -> build/alarmController/alarmController.ino

Copy ./alarmController/Buzzer.cpp -> build/alarmController/Buzzer.cpp

Copy ./alarmController/Buzzer.h -> build/alarmController/Buzzer.h

Patch File build/alarmController/config.h

Copy ./alarmController/Vigil.cpp -> build/alarmController/Vigil.cpp

Copy ./alarmController/Vigil.h -> build/alarmController/Vigil.h

Copy ./alarmController/Wiegand.cpp -> build/alarmController/Wiegand.cpp

Copy ./alarmController/Wiegand.h -> build/alarmController/Wiegand.h

Copy ./alarmController/Wiegand_README.md -> build/alarmController/Wiegand_README.md

Copy ./domoticz/domoticz -> build/domoticz/domoticz

Copy ./domoticz/Readme.txt -> build/domoticz/Readme.txt

Patch File build/domoticz/script_device_security2Warning.lua

Patch File build/domoticz/script_time_securityStatus.lua

Process finished with exit code 0
```

L'exécution génère l'ensemble du code à déployer dans un répertoire build :

- Build/alarmController : code pour l'IDE Arduino
- Build/domoticz : code pour les scripts domoticz

Mise à jour

Afin de pouvoir facilement mettre à jour le code dans ce système qui en définitive ne sera plus facilement accessible (dans mon mur en ce qui me concerne), la mise à jour OTA (Other The Air) a été activé dans le code afin de pouvoir pousser une nouvelle version via le wifi.

Conclusion

Fin de la première étape. Le système est en production chez moi depuis 2 mois et fonctionne correctement. La prochaine étape consiste à réaliser un accès sécurisé distant à notre Domoticz sans l'exposer directement sur Internet !