Documentation générale informatique – Projet ELFE

Date de mise à jour : 11/10/24

Objectifs de la documentation

- Décrire l'architecture informatique ELFE, avec l'ensemble des composants informatiques
- Permettre une réplication sur d'autres projets et territoires

Table des matières

Présentation du projet ELFE

Le projet ELFE (Expérimentons Localement la Flexibilité Energétique) a été lancé en Novembre 2021 par l'association Anergies Citoyennes en Pays de Vilaine, sur le territoire de Redon Agglomération et de la Communauté de Communes de Pontchateau-Saint-Gildas.

Son objectif est de contribuer à l'équilibre en temps réel entre la production électrique du territoire et sa consommation. Pour cela, des foyers et des acteurs économiques mettent à disposition des flexibilités sur leurs consommations (charge de véhicule électrique, cycle de lave-linge), qui sont ensuite déclenchées au moment optimal par le système informatique.

Ce système est construit entièrement en logiciel libre, afin que les citoyens gardent le contrôle sur le système technique, et également pour permettre un essaimage et une co-construction autour de ces enjeux de flexibilité énergétique citoyenne.

Architecture informatique

Principe de fonctionnement

Le projet ELFE s'articule autour d'un Energy Management System (EMS). Cet outil mathématique prend en compte deux ensembles de données :

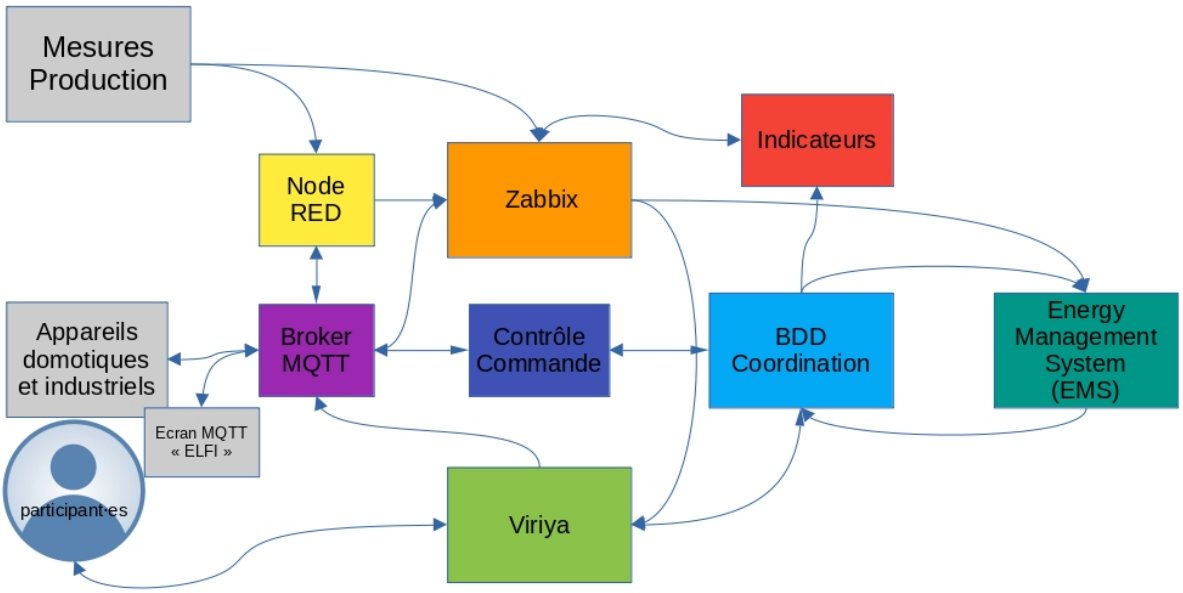
- Les productions et consommations d'énergie passées du territoire, qui permettent d'établir une prévision d'équilibre énergétique futur (en lien avec les prévisions météo),
- Les déclarations de flexibilité énergétiques des participants, sur leurs machines.

Les déclarations sont faites par les participants, soit de manière unique (un chauffe eau a une plage horaire souhatée, un chauffage a un planning de chauffe qui se répète), soit de manière ponctuelle (tel jour, un lave-linge est prêt à partir et il diot être fini à telle heure). Ces contraintes, en lien avec les mesures fines de consommation de chaque équipement piloté, composent les flexibilités sur lesquelles ELFE va pouvoir agir.

L'EMS construit un problème d'optimisation sur la base de ces données, puis y apporte une solution en ciblant une « fonction objectif » établie par le projet. Celle-ci est simple : « l'action de ELFE doit minimiser la quantité d'énergie produite sur le territoire et qui n'y est pas consommée. »

Les flexibilités disponibles sont alors « placées » dans le temps, pour construire un nouvel équilibre futur qui améliore la fonction objectif. Enfin, la chaîne de contrôle-commande exécute ce programme de déclenchement des flexibilité, qui est réactualisé toutes les 15 minutes.

Diagramme des composants



Chaque brique est un composant numérique distinct.

Composant	Basé sur	Licence	Fonction
Broker MQTT	Mosquitto		Communication instantanée entre les composants et les équipements
Zabbix	Zabbix		Stockage des données d'énergie
Viriya	Viriya		Affichage de données de consommation, et des indicateurs du projet Déclarations de flexibilités individuelles
BDD Coordination	PostgreSQL	XX (le logiciel) EUPL v1.2 (le schéma)	Stockage des configurations du matériel installé
EMS	Code Python spécifique	EUPL v1.2	Prévision et optimisation de l'équilibre énergétique du territoire
Contrôle Commande	Code Python spécifique	EUPL v1.2	Envoi des ordres de commande aux équipements Gestion des interactions avec l'afficheur ELFI
Indicateurs	Code Python spécifique	EUPL v1.2	Calcul des indicateurs de flexibilité du projet
Node-RED	Node-RED	XX (le logiciel) EUPL v1.2 (les flows)	Calcul d'équilibre mis à l'échelle, calcul affichage et envoi météo énergétique, collecte données PV, prix de la bourse, envoi du mail-flash quotidien
Ecran ELFI	OpenHASP	XX (logiciel) EUPL v1.2 (les écrans)	Déclaration de flexibilités individuelles Affichage du statut de pilotage des équipements pilotés

Fonctionnement spécifique

La chaîne d'information principale de ELFE fonctionne de la façon suivante :

Les équipements domotiques sont installés sur les équipements pilotés, enregistrés dans la base de données coordination, et reliés à un compte utilisateur dans Viriya. Ils se connectent à un broker MQTT central, au moyens d'identifiants individuels, et via le réseau internet du participant (ou une box 4G). Leurs mesures de consommation sont publiées sur le Broker MQTT puis enregistrées dans Zabbix.

Les utilisateurs déclarent des flexibilités (« je remplis ma machine à laver et j'ai besoin qu'elle soit finie au plus tard ce soir à 18h, le cycle dure 2h30 ») via Viriya ou via le ELFI+ Contrôle-Commande. Cela enregistre le nouvel état des flexibilités disponibles dans la Base de Données Coordination.

Toutes les 15min, l'EMS récupère la liste des équipements prêts à être pilotés, prévoit l'équilibre consommation-production du territoire, puis optimise le placement de la consommation des appareils dans le temps et publie une planification dans une base SQL. Cette prévision est faite pour les 24 heures à venir, découpées en 96 créneaux de 15min

Ensuite, le Contrôle-Commande parcourt cette planification et envoie les ordres de déclenchement (ou d'attente) aux équipements domotiques, via le Broker MQTT.

En dehors de cette chaîne de communication :
Node-RED fonctionne sur sollicitation MQTT, ainsi que périodiquement (un envoi de mail quotidien). Les Indicateurs sont calculés de manière asynchrone, toutes les 15 minutes.

Composant	Documentation	Outil d'exploitation
Broker MQTT	Lien	MQTT Explorer
Zabbix		Interface web
Viriya		Interface web
BDD Coordination		Interface web et DBeaver
EMS		Ligne de commande
Contrôle Commande		Ligne de commande
Indicateurs		Ligne de commande
Node-RED		Interface web
OpenHASP	Lien	IDE Arduino

Mise en place des composants

Pour mettre en marche le système de flexibilité ELFE, il faut installer puis configurer l'ensemble des composants de l'architecture décrit ci-avant. Chaque composant nécessite des accès dans d'autres composants, pour obtenir ou publier des données.

Composant	Besoin d'accès* au composant...
Zabbix	Broker MQTT
Viriya	**Broker MQTT**, Zabbix, **BDD Coordo**
Controle-Commande	**Broker MQTT, BDD Coordo**
EMS	**BDD Coordo**, Zabbix
Indicateurs	BDD Coordo, **Zabbix**
Node-RED	**Broker MQTT**, Zabbix, BDD Coordo

* **gras** = accès en écriture

NB : les accès aux données externes (centrales de production, API diverses, logiciels de monitoring...) ne sont pas détaillées ici car dépendent de chaque contexte.

Une fois l'installation et les identifications faites, l'ordre de travail pour déployer les équipements domotiques est le suivant :

- Création des comptes MQTT par participant
- Installation des équipements domotiques chez les participants
- Enregistrement des données publiées sur le broker dans Zabbix
- Enregistrement de la configuration matérielle de chaque participant dans la BDD Coordination
- Création d'un compte utilisateur dans Viriya

  A ce moment-là, le participant peut observer ses consommations sur Viriya, et déclarer ses flexibilités via Viriya ou le ELFI.

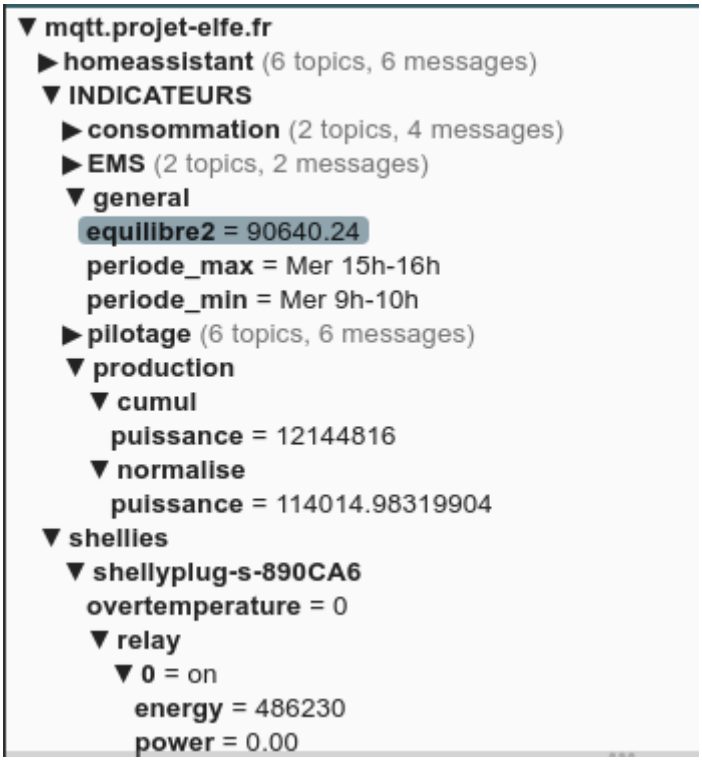
Broker MQTT

Outil de communication avec l'ensemble des équipements domotiques et industriels déployés sur le terrain. Il s'agit du logiciel mosquitto installé sur linux, et qui écoute sur internet.

Chaque équipement domotique y est connecté, avec des identifiants individuels à chaque participant. Cela permet d'éviter d'avoir à ouvrir un tunnel de connexion entre le système ELFE et chaque réseau local.

Les équipements publient leurs valeurs et leurs états, et s'abonnent à certains topics précis pour recevoir des ordres.

Exemple d’arborescence du broker, affichée par MQTT Explorer :



Le topic INDICATEURS/general/equilibre2 a pour valeur 90640.24
La prise shellyplug-s-890CA6 a mesuré 486230 W.min d’énergie

Serveur Zabbix

Cet outil complet de supervision et d'historisation a pour rôle le stockage des valeurs d'énergie publiées sur le broker. Cela permet ensuite leur exploitation par Viriya et l'EMS.

Zabbix enregistre les données au travers *d'éléments* qui sont autant de points de mesure individuels. On peut y ajouter des « tags » pour les identifier (appareil:ECS, client:indicateur) et automatiser certains traitements par d'autres composants ensuite.

Pour ELFE, la plupart des éléments correspondent à un topic du broker MQTT, auquel Zabbix s'abonne, et dont les valeurs sont historisées dans le temps.

Liste des éléments enregistrés, pour chaque équipement domotique :

- puissance instantanée
- énergie totale
- niveau de batterie (pour les doigts robots)
- messages de pilotage reçu ou d'état de commande (au cas par cas, si besoin)

L'outil enregistre également les valeurs des indicateurs opérationnels du projet, calculées périodiquement et qui lui sont envoyées par le script « Indicateurs » au moyen de son API.

Enfin, Zabbix peut récupérer les données de production d'énergie (par API ou des protocoles de communication industriels), qui sont utilisées par Node-RED pour calculer un équilibre instantané et mis à l'échelle du panel actif.

BDD Coordination

La base de données Coordination (ou BDD Coordo) contient l’ensemble des équipements domotiques installés, des équipements pilotés ou mesurés impliqués dans ELFE, ainsi que leurs états et contraintes déclarés par les participants.

C’est une base PostgreSQL construite pour permettre aux autres blocs du SI qui s’y connectent de remplir leurs fonctions opérationnelles et d’échanger les informations nécessaires. L’ensemble de la base (tables, listes, contraintes d’ajout) est décrite dans un fichier SQL (bdd_coordination.sql), qu’il suffit d’importer pour re-créer une base « vierge » sur un nouveau serveur.

NB : Aucune donnée de consommation n’est historisée dans cette base. Les équipements pilotés contiennent un identifiant qui pointe vers leurs mesures stockées dans Zabbix.

La configuration est faite manuellement (voir la partie « ajout de matériel dans la BDD »), puis Viryia et le Controle-Commande y écrivent et lisent les états des équipements pilotés et équipements domotiques. L’EMS vient y lire la liste des appareils à piloter, mais publie dans une autre base qui lui est dédiée (afin d’éviter des requêtes concurrentes).

Ajout de matériel dans la BDD

Quand des équipements domotiques sont installés chez les participants, il faut ensuite les inscrire dans la BDD au moyen d’un script SQL. Pour le moment, le script doit être rempli manuellement, mais des travaux sont en cours pour le générer automatiquement à partir des données d’inventaire.

EMS

L’EMS est un logiciel en python, qui construit puis résout un problème mathématique d’optimisation. Il se compose de différents modules, qui exécutent chacun une partie des fonctions de façon modulaire. (voir sa documentation exploitant et développeur pour les détails).

Il collecte périodiquement les courbes de consommation de tous les appareils enregistrés sur Zabbix, en les distinguant au moyen des « tags ».
Il collecte également les données de production et de consommation, des prévisions météo sur une source externe (par API) ainsi que la courbe d’équilibre passée, afin de construire une prévision de l’équilibre énergétique futur sur 24h.

Enfin, il récupère la liste des appareils actuellement pilotables, et construit un planning optimisé de pilotage sur les 24h à venir, permettant de minimiser sa fonction objectif, qui est : somme des kWh produits et non-consommés sur le territoire chaque 15min.

Sorties

Nous n’avons rien à piloter sur l’EMS. Il se déclenche périodiquement et publie ses résultats en fonction des contraintes d’entrée, dans la base « sortie_ems », La table « result » contient l’ensemble des machines pilotables, et leur planning de pilotage suohaité pour les 24 prochaines heures.

La table ems_run_info contient le temps de résolution effectif du problème mathématique, ainsi que des points de données significatifs : heure et valeur de la plus grand puissance, heure et valeur de la plus petite puissance.

Ces données d’information sont lues par Node-RED puis publiées sur le broker et remontées dans Zabbix, afin de permettre un monitoring.

Détection des cycles

L’EMS dispose d’une fonction de détection de cycles basique, basé sur 4 paramètres : puissance minimale de départ, durée minimale de l’appel de puissance de départ (détecte le début de cycle), puissance d’extinction, durée minimale de la puissance d’extinction (détecte la fin de cycle). Chaque jour, icette fonction parcourt les courbes de charge de chaque appareil piloté, pour en extraire le dernier cycle, synthétisé au pas 15min. Ce cycle synthétique est ensuite utilisé par l’EMS pour son optimisation des placements de consommation.

Prévisions

La prévision utilisée dans ELFE est un algorithme par persistance, appliqué à la courbe d’équilibre. Les prévisions de la météo climatique ou de consommation ne sont pas utilisées.

Fonction objectif

L’EMS optimise le placement des consommations pour minimiser une expression mathématique, nommée « fonction objectif ». Dans le cadre de l’expérimentation, cette fonction se résume à : la somme de l’énergie importée sur les prochaines 24h, d’après la prévision et le placement des consommations flexibles.

Viriya

Viriya est un logiciel proposé sous forme de « Software as a Service » et utilisable depuis son site web. Il n'est pas exclusif au projet ELFE mais accueille aussi d'autres clients et utilisateurs. EPV y dispose d'une structure « EPV » qui porte un programme ELFE. Des comptes de responsables ont été créés pour les coordinateurs.

Viriya dispose d'un accès à la BDD Coordination et à Zabbix, afin d'obtenir les données de consommation, les valeurs des indicateurs, et de communiquer sur l'état des appareils à piloter.

Les utilisateurs sont autonomes pour obtenir et renouveler leurs mots de passe. En revanche, les invitations expirent et doivent être renvoyées à la main par l'équipe EPV.

Chaque utilisateur est « incubé » par EPV, ce qui permet de prendre la main sur leur compte en tant qu'accompagnant numérique.

1° Ajouter un compte participant

Depuis l'accueil de la Structure EPV, Se rendre dans le menu Paramètres de la structure > menu « Incubés ».

Cliquer sur ajouter et remplir le formulaire :

- référence interne = N° de participant elfe (« R004 »)
- email : bien indiquer l'adresse d'inscription à ELFE
- date de naissance : pas nécessaire
- A CONFIRMER : cocher la case « observateur » pour éviter de donner accès aux paramètres.

Cliquer sur « Enregistrer » : le participant reçoit un mail d'invitation pour se connecter.

2° Ajouter un participant au programme ELFE

Pour que les appareils configurés dans la BDD Coordination soient créés dans le foyer, il faut ajouter ce foyer au « programme ELFE ». Il n'y a pas besoin d'attendre que le participant se connecte/active son compte pour l'ajouter au programme

Pour faire l'ajout, on va dans : Tableau de bord EPV > Programmes > Programme ELFE > Paramètres (en haut à droite) > Participants.

Puis on clique sur le bouton « ajouter » , la liste des comptes incubés s'affiche : on coche uniquement les comptes récemment créés pour les ajouter au programme.

En cliquant sur « enregistrer », les foyers sont ajoutés et leurs appareils sont créés automatiquement.

Consulter ou modifier l’état de pilotage d’un appareil

Il faut d’abord choisir le compte incubé, en tant que structure EPV : dans le menu « EPV » en haut à droite, cliquer sur « incubateur ». Puis cliquer sur le nom du foyer choisi, cela nous fait « rentrer » dans leur rôle.

Ensuite, il faut se rendre dans le menu : foyer > bâtiment > appareils > *l’appareil choisi* > commandes.

Optionnel - Ajouter un appareil à un utilisateur :

Menu appareil > Paramètres > Ajouter

- menu connexion : ajouter un lien à ELFE
    - identifiant elfe = colonne ID de la table epom
- menu connexion : ajouter un lien à zabbix
    - identifiant = colonne « mesure_puissance_elec_id » de la table epom_[machine_generique] (à retrouver via equipement_pilote_specifique_id)

Ajouter un compte « exploitant » :

Ceux qui ont besoin de configurer des choses, avoir accès aux programmes, etc.

Paramètres de la structure > Menu relations > Ajouter

- rôle = « responsable » pour avoir tous les droits.
- Demander à l’administrateur de configurer ce nouveau compte en « assistant numérique » afin de pouvoir gérer des incubés.

Contrôle Commande

Le logiciel se lance seul au démarrage du serveur. Il suit l'état des écrans ELFI et envoie les ordres aux équipements domotiques selon les consignes EMS et les demandes des participants.

Pour savoir ce qu'il a fait/observé, on regarde son journal avec la commande linux (à exécuter sur la machine virtuelle qui l'accueille)(touche « q » pour sortir de la commande) :

- journalctl -u elfecommande -S -15m

Pour connaître l'état du service (démarré, en panne...) :

- systemctl status elfecommande.service

Indicateurs

Les indicateurs opérationnels du projet sont calculés par un script python dédié, exécuté toutes les 15 min (via un service systemd) sur une machine virtuelle.

Les informations nécessaires sont piochées dans Zabbix et les tables de résultat de l'EMS.

Les valeurs calculées sont poussées dans Zabbix, dans des éléments distincts de type « zabbix trapper » et portant l'étiquette « indicateurs ».

Écran ELFI

Réinitialiser le mot de passe wifi de l'écran

Brancher l'écran au PC et ouvrir Arduino.

Choisir le port USB actif (ACM0 ou USB0, pas serial0), puis aller dans « Outils > Moniteur Série. »

Entrer la commande suivante pour remettre l'écran en état initial :

- ["ssid=","pass=","reboot"]

Cliquez sur Envoyer : si l'écran ne reboote pas seul, renvoyer la commande.

Entrer la commande suivante pour paramétrer directement le Wifi d'un foyer :

["ssid=nom_du_reseau","pass=mot_de_passe_du_reseau","reboot"]

Equipements domotiques connectés

Fréquence de publication

Matériel	Fréquence de publication	Testament
Shelly plug	30s	
Shelly Plus	1min	Online = false
Prise Nous	30s	
Carte TIC	5min	
Doigt Switchbot	10min	
Thermomètre	6 h	
Socorel	30s	

Prise Shelly

Les premiers modèles reçus (de 2022) sont bloqués lors de la mise à jour du firmware. Il faut forcer la mise à jour avec une version plus récente, par exemple avec la commande : http://192.168.1.133/ota?url=http://archive.shelly-tools.de/version/v1.9.4/SHPLG-S.zip

Le topic de public de publication d'une prise Shelly n'est pas configurable avec les anciens firmwares : c'est son identifiant matériel (*ex : A8BC25*). Pour retrouver une prise sur le broker MQTT, il faut aller chercher cette référence dans l'inventaire matériel ELFE.

Automate Socorel

L'automate physique dispose de 8 sorties indépendantes : elles sont enregistrés dans ELFE comme 8 équipements domotiques indépendants. Cependant, ces 8 équipements publieront sur le même topic MQTT, dans des sous-topics distincts (EtatSortie1 à EtatSortie8).

Passage en mode forcé avec reprise du contrôle de puissance depuis l'écran du Socorel : le contrôle commande n'est plus opérationnel, on passe l'appareil en mode désactivé 70.dans la BDD Coordo.

Annexes

- Tableau des topics MQTT pour les différents équipements domotiques : « topics_et_ordres_mqtt.ods »

Type Item	equipement_domotique_type_id	Exemple racine	Valeur 1	Valeur 2	Valeur 3	topic_mqtt_commande_text
A – shelly	111	shellies/shellyplug-s-ABCDEF	relay/0/energy	relay/0/power		relay/0/command
A – Nous A1T	112	tele/A123	SENSOR			POWER
B	211	tele/B001	SENSOR			POWER
C	311	switchbot/C002/bot/C003	battery	rssi		set
D	411	D003	status/switch:0			rpc
F	611	shellies/F002	sensor/temperature	sensor/humidity	sensor/battery	rpc
G	711	hasp/g002	1 topic par item	state/sensor		command
SOCOREL		Socorel/121345678/	PA1MN			sortie1 (à8)

Type Item	Topic de commande complet	Forme du message de pilotage
A – shelly	shellies/shellyplug-s-ABCDEF/relay/0/command	on
A – Nous A1T	cmnd/A123/POWER	ON
B	cmnd/B001/POWER	ON
C	switchbot/C002/bot/C003/set	PRESS
D	D003/rpc	`{"id":123, "src":"controle/D003","method":"Switch.Set", "params":{"id":0,"on":true}}`
F	shellies/F002/rpc	-
G	hasp/g002/command	jsonl={"page":9,"id":8,"obj":"label","x":50,"y":50,"w":160,"h":32,"mode":"break","align":"center","text":"Tout va bien :)"}
SOCOREL		1

Type Item	topic_mqtt_controle_json	topic_mqtt_lwt	Commande config
A – shelly	relay/0	online	
A – Nous A1T	STATE	LWT	
B	STATE	LWT	
C	status	/switchbot/C002/lastwill	
D	/controle/D003	online	{ "id": 123, "src": "controle/D014", "method": "Switch.SetConfig", "params": { "id": 0, "config":{ "id":0, "initial_state":"on", "auto_on":true, "auto_on_delay":"1800" } } }
F	-	online	
G	-	LWT	["ssid=","pass=","reboot"]
SOCOREL	Socorel/031636408854/EtatSorties	à déterminer par HttP	

Type Item	Lien documentation		
A – shelly	https://shelly-api-docs.shelly.cloud/gen1/#shelly-plug-plugs-mqtt		
A – Nous A1T	https://tasmota.github.io/docs/Commands/#with-mqtt		
B	https://tasmota.github.io/docs/Commands/#with-mqtt		
C			

D	https://shelly-api-docs.shelly.cloud/gen2/ComponentsAndServices/Switch/#mqtt-control	{
		"id": 123,
		"src": "controle/D063",
		"method": "Mqtt.SetConfig",
		"params": {
		"config": {"status_ntf":true}
		}
		}
			Si pas de publication du status : Mqtt.GetConfig puis Mqtt.SetConfig	{
				"id": 123,
				"src": "controle/D063",
				"method": "Shelly.CheckForUpdate",
				"params": {

F			
G	https://openhasp.haswitchplate.com/0.6.3/commands/		
SOCOREL			