Guide d'utilisation

Projet CodeVSI :

EcoWatt

Béchu Hugo

Epaillard Alexandre

Jolivet Clément

<u>Avertissements :</u>

Ce guide est un guide d'utilisation du code donc il sert pour mieux comprendre comment notre code fonctionne et plus généralement Node-Red, mais avant de le lire il convient d'avoir lu le rapport final que nous avons déjà fourni. Cela permettra de mieux comprendre le sujet puisque dans ce guide seul le code sera expliqué. De plus, bien que ce code fonction, il n'est pas parfait, il est fortement possible qu'il existe à de nombreux moments de meilleures façons de procéder pour gagner en temps en juste pour simplifier le code.

Pour une bonne utilisation du code il convient avant de l'importer de vérifier si toutes nœuds particuliers ont été installés. En effet, dans le code plusieurs nœuds non disponible dans la version de base de Node-Red sont utilisés. Voici les liens pour les télécharger :

Nœud pour l'IHM : https://flows.nodered.org/node/node-red-dashboard

Nœud EcoWatt : https://flows.nodered.org/node/@vital91/node-red-contrib-ecowatt

Noeud Clé et Porte : https://flows.nodered.org/node/node-red-contrib-message-gate

Nœud Linky : https://flows.nodered.org/node/node-red-linky

Table des matières

1 – Récupération des données et mise en place de l'IHM

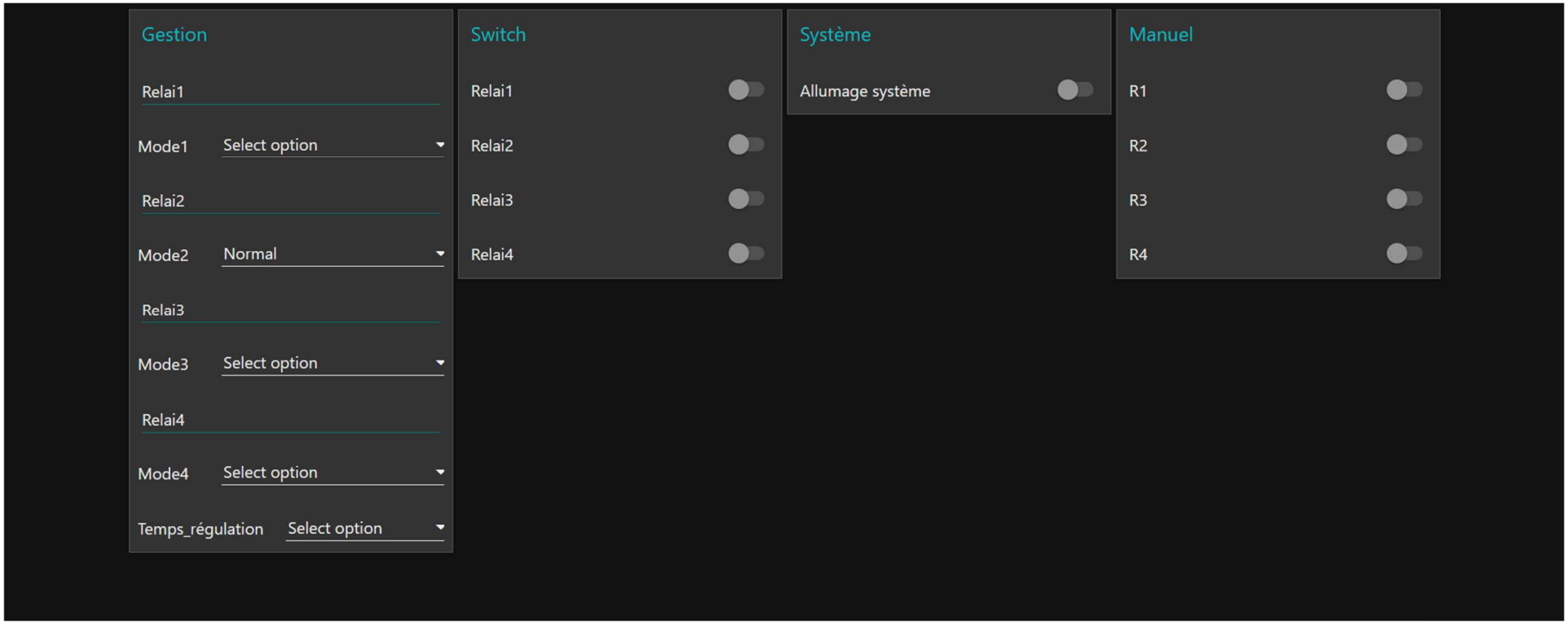

Figure 1 : IHM

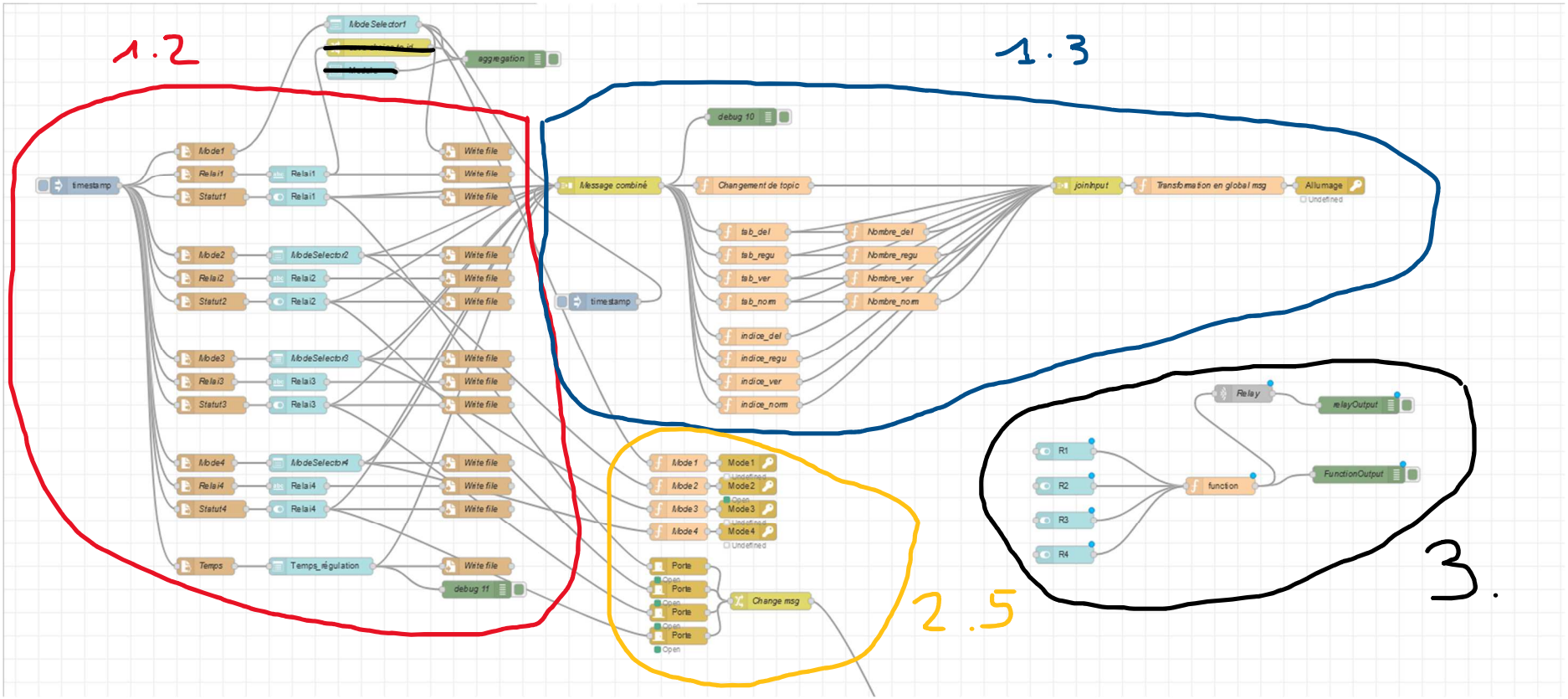

Figure 2 : Récupération des données de l'IHM

La figure 1 représente l'IHM de l'application et la figure 2 représente la 1ère partie du code qui permet de créer l'IHM et de récupérer les données saisies sur l'IHM. (Le code a été découpé en plusieurs parties qui sont affichées sur la figure 2 par des encadrés pour faciliter la compréhension du code. De plus, il est indiqué dans quel section du guide les différentes parties du code sont expliquées.) L'objectif est ainsi de permettre à l'utilisateur de saisir les données liées aux équipements qu'il a branché sur les différents relais ainsi que pour chacun, un mode de fonctionnement qu'il souhaite.

1.1- IMH

Sur l'IHM, représentée par la figure 1 et configuré dans cette version pour 4 relais, l'utilisateur doit alors saisir, dans la 1ère partie « Gestion », pour chaque relai :

- Le nom de l'équipement qui est branché dessus (cette information n'est en fait pas utile puisqu'elle n'intervient pas dans le traitement de la gestion de l'allumage des équipements par application),
- Et le mode à appliquer à l'équipement. Il y a 4 choix possibles, le mode délestage, le mode régulation, le mode verrouillage et le mode normal. Le principe de fonctionnement de ces 4 modes sont détaillés dans le rapport final que nous avons fourni précédemment.

L'utilisateur doit aussi saisir un temps de régulation, une donnée nécessaire pour le mode régulation. La deuxième partie « Switch » correspond à des boutons qui permettent de savoir lesquels des relais sont activés. Si, par exemple, le switch du relai1 n'est pas activé et que toutes les informations de l'équipement branché sur le relai1 sont saisies, alors le relai1 n'est pas pris en compte par application et ne sera jamais allumé. Cela permet de prendre seulement en compte les relais sur lesquels il y a un équipement branché. Les parties 3 et 4 de l'IHM, « Système » et « Manuel » seront expliquées plus tard dans le guide.

Attention, il faut néanmoins pour un bon fonctionnement de l'application que tous les champs de l'IHM soient renseignés (même le nom des équipements).

1.2- Implémentation de la récupération des données

Tout d'abord, avant de parler du code propre à notre application, il convient de préciser le fonctionnement de deux nœuds qui apparaissent régulièrement dans le code mais qui ne sont pas vraiment nécessaires pour le fonctionnement de l'application mais qui le sont plus pour le codeur :

- Le nœud en vert « debug ». Il permet, en appuyant sur le petit bouton juste à droite du nœud, d'afficher dans la console de débogage le contenu du message qui passe vers lui. Il permet ainsi de débugger et de vérifier le bon fonctionnement de l'application (on vérifie si les messages qui passent correspondent à nos attentes).
- Le nœud en gris « inject ». Il permet, en appuyant sur le petit bouton juste à gauche du nœud, d'injecter un message dans un flux, soit manuellement, soit à intervalles réguliers (le message peut être de différents types, notamment des chaînes de caractères, des objets JavaScript ou l'heure actuelle). Il est ainsi utile pour actionner des parties du code ou simuler un message et ainsi faire des tests.

Concernant le codage de l'IHM, les nœuds en bleu clair sur la figure 2 correspondent aux différents champs dans l'IHM. On retrouve ainsi les nœuds « text input (abc) » (nommé dans le code 'Relai1', 'Relai2', 'Relai3', 'Relai4') pour la saisie des noms des équipements, « drop down » (nommé dans le code 'ModeSelector1', 'ModeSelector2', 'ModeSelector3', 'ModeSelector4', 'Temps_régulation') pour le choix du mode et les nœuds « switch » (nommé dans le code 'Relai1', 'Relai2', 'Relai3', 'Relai4') pour activer les relais. Le bloc de 4 nœuds tout en haut à gauche sur la figure

2 (2 bleu clair, un jaune et un vert) ne sont pas à prendre en compte, ils servaient pour un test ultérieurement.

Pour éviter d'avoir à chaque fois à ressaisir manuellement toutes les informations dans l'IHM, nous avons mis en place un stockage et de récupération de ces données. Un fois que les données ont été saisies pour la 1ère fois, elles passent dans les nœuds marrons à droite des nœuds bleu clair (qui sont les nœuds des données de l'IMH pour rappel). Ces nœuds sont des nœuds de type « write file », c'est-à-dire qu'ils vont écrire dans un fichier texte local les informations du message qui sont allées à eux. On a alors créé sur mon pc des fichiers « document texte » vides pour chaque champ de l'IHM (4 pour les noms, 4 pour les modes, 4 pour les switchs et 1 pour le temps de régulation) et nous avons indiqué aux nœuds le chemin de ces fichiers sur mon pc. Par exemple, on a indiqué comme chemin pour le fichier associé au nom de l'équipement branché sur le relai 1 :

\Hugo\FISEA1\CodeVSI\Programmation\IHM\Relai1.txt

Il faudra évidement le changer lorsque l'on utilise un autre pc. Ceci forme ainsi la partie stockage.

Pour la récupération, les nœuds utilisés sont des nœuds « read file » en marron aussi et à gauche des nœuds de l'IHM en bleu clair. Ces différents nœuds de lecture permettent une fois que l'on a renseigné le chemin du fichier à lire, de lire le contenu de ces fichiers puis de l'envoyé sous forme d'un message aux nœuds de l'IHM qui alors fonctionnent de la même manière que lors de la saisie manuelle des informations sur l'IHM. C'est le nœud « inject » en gris, tout à gauche qui une fois actionné permet d'activer les nœuds de lecture.

1.3- Transformation des données

Une fois les données saisies sur l'IHM ou par le biais des fichiers de stockage (cf. 1.2), on récupère toutes les informations avec un nœud « join » (nommé dans le code 'Message combiné'), en jaune sur la figure 2. Ce nœud permet de créer un unique message en JSON contenant toutes les données. Le fait que l'on est maintenant un seul message est pour faciliter le traitement de ces données. Par exemple, on obtient un message du type :

{"mode1":"regu","mode3":"regu","mode2":"del","mode4":"ver","temps":4,"statut_relai1":"on","statut_relai2":"on","statut_relai3":"on","statut_relai4":"on"}

Par la suite, ce message va être utiliser pour calculer différentes informations supplémentaires nécessaires pour l'implémentation des 4 modes de fonctionnements. Ainsi, on va pour chacun des modes :

- Construire un tableau contenant le numéro du relai des équipements fonctionnant dans ce même mode.
- Initialiser l'indice du parcours du tableau à 0
- Puis, calculer le nombre d'équipements fonctionnant dans ce même mode

On effectue cette étape grâce à des nœuds « function » (en orange sur la figure 2). Un nœud « function » est une fonction JavaScript à exécuter sur les messages reçus par le nœud. Il suffit ainsi de coder de petites fonctions en JavaScript. Une fois cela fait, on recréer un seul message avec un nœud « join » (*joinput* dans le code). Ce message contient ainsi les données de l'IHM et les nouvelles informations de l'on vient de rajouter.

Avec l'exemple si dessous en entrée, on obtient alors le nouveau message :

{"donnees_IHM":{"mode1":"regu","mode3":"regu","mode2":"del","mode4":"ver","temps":4,"statut_relai1":"on","statut_relai2":"on","statut_relai3":"on","statut_relai4":"on"},"tab_del":[3],"tab_regu":[1,2],"tab_ver":[4],"indice_del":0,"indice_regu":0,"indice_ver":0,"tab_norm":[],"indice_norm":0,"nbr_del":1,"nbr_regu":2,"nbr_ver":1,"nbr_norm":0}

Pour finir avec cette partie, on transforme le message précédant contenant toutes les données, en un message global pour faciliter son utilisation future. Pour cela, on utilise encore un nœud « function ».

Une fois cela fait, on fait passer ce message dans un dernier nœud « clé » appelé 'Allumage' sur la figure 1. Ce nœud « clé » n'est pas un nœud disponible de base avec Node-Red et fonctionne de pair avec un nœud « porte » qui n'apparait pas sur la figure 2 mais sur la figure 3 dans la section suivante. Le fonctionnement des nœuds « clé » et « porte » est le suivant :

Si un message en entrée du nœud « clé » contient le champ payload qui est le booléen *true* alors le nœud « clé » est activé et le nœud « porte », auquel on a préalablement renseigné le fait qu'elle soit associée à ce nœud « clé », est ouverte et laisse passer en sortie le message qu'elle avait en entrée. Dans le cas contraire où le booléen est false alors la clé se désactive et la porte se bloque empêchant la propagation du message en sortie.

L'utilisation des nœuds « clé » et « porte » sert donc à bloquer les messages quand on le souhaite pour en autre éviter la propagation de messages inutiles. C'est cette méthode que nous avons trouvée mais il doit en exister bien d'autres.

2 – Implémentation des modes de fonctionnement

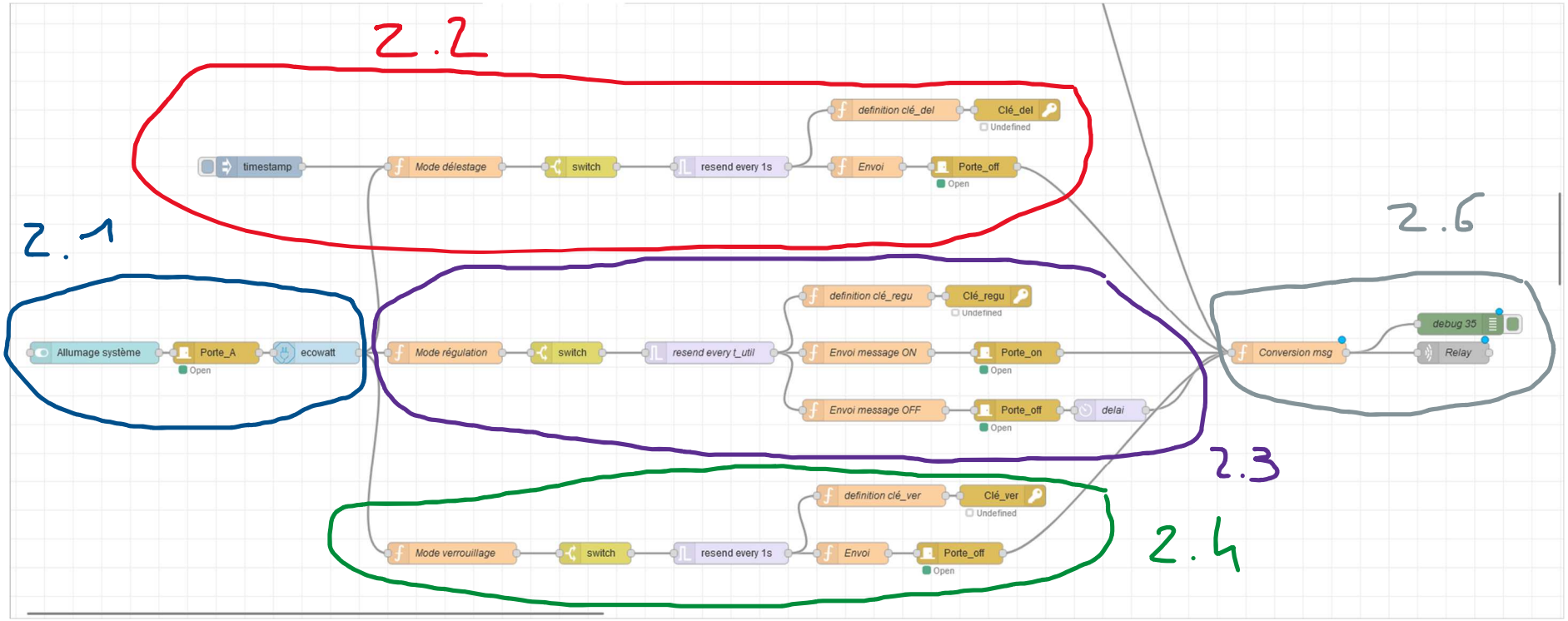

Figure 3 : Implémentation des 3 modes de fonctionnement (délestage, régulation et verrouillage)

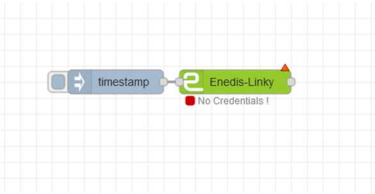

Figure 4 : nœud Linky

2.1 – EcoWatt

Une fois toutes les informations dont nous avons besoin ont été instanciées avec le code figure 1 expliqué dans la partie 1, il nous commander l'allumage de notre application et récupérer aussi les données d'EcoWatt pour agir en conséquence.

Pour activer notre application, il faut que le switch « Allumage » sur l'IHM dans la 3ème partie « Système » soit activé.

Néanmoins, avant que l'application se déclenche et attribut l'allumage des relais toute suite, il faut vérifier que toutes les informations de l'IHM ont été saisies et transformées en un message global comme décrit dans la partie 1.3. C'est là qu'intervient le nœud « porte » expliqué à la fin de la partie 1.3. En effet, la clé 'Allumage' ne s'active que si le message global passe par ce nœud « clé » et alors le nœud 'Porte_A' est ouverte et ne bloque pas la suite.

Ainsi l'application peut enfin activer sa partie cœur. Tout d'abord, il faut récupérer les données fournissent par l'API EcoWatt. Ceci est fait très simplement grâce à un nœud qui a déjà été codé : le nœud « EcoWatt ». Pour le fonctionnement du nœud il faut néanmoins fournir une clé authentification STR qui est obtenu en suivant ce guide https://data.rte-france.com/catalog/-/api/doc/user-guide/Ecowatt/4.0. (Il faudra en autre se créer un compte RTE si l'authentification de notre code ne fonctionne plus).

On obtient alors les données sous le format JSON. Voici un exemple :

[{"GenerationFichier":"2022-06-03T07:36:25+02:00","jour":"2022-06-06T00:00:00+02:00","dvalue":1,"message":"Situation normale ","values":[{"pas":0,"hvalue":1},{"pas":1,"hvalue":1},{"pas":2,"hvalue":1},{"pas":3,"hvalue":1},{"pas":4,"hvalue":1},{"pas":5,"hvalue":1},{"pas":6,"hvalue":1},{"pas":7,"hvalue":1},{"pas":8,"hvalue":1},{"pas":9,"hvalue":1},{"pas":10,"hvalue":1},{"pas":11,"hvalue":1},{"pas":12,"hvalue":1},{"pas":13,"hvalue":1},{"pas":14,"hvalue":1},{"pas":15,"hvalue":1},{"pas":16,"hvalue":1},{"pas":17,"hvalue":1},{"pas":18,"hvalue":1},{"pas":19,"hvalue":1},{"pas":20,"hvalue":1},{"pas":21,"hvalue":1},{"pas":22,"hvalue":1},{"pas":23,"hvalue":1}]},

{"GenerationFichier":"2022-06-03T07:36:25+02:00","jour":"2022-06-04T00:00:00+02:00","dvalue":3,"message":"Coupures d'électricité programmées","values":[{"pas":0,"hvalue":1},{"pas":1,"hvalue":1},{"pas":2,"hvalue":1},{"pas":3,"hvalue":1},{"pas":4,"hvalue":1},{"pas":5,"hvalue":2},{"pas":6,"hvalue":2},{"pas":7,"hvalue":3},{"pas":8,"hvalue":3},{"pas":9,"hvalue":3},{"pas":10,"hvalue":3},{"pas":11,"hvalue":3},{"pas":12,"hvalue":3},{"pas":13,"hvalue":2},{"pas":14,"hvalue":2},{"pas":15,"hvalue":2},{"pas":16,"hvalue":2},{"pas":17,"hvalue":3},{"pas":18,"hvalue":3},{"pas":19,"hvalue":3},{"pas":20,"hvalue":2},{"pas":21,"hvalue":2},{"pas":22,"hvalue":2},{"pas":23,"hvalue":2}]},

{"GenerationFichier":"2022-06-03T07:36:25+02:00","jour":"2022-06-05T00:00:00+02:00","dvalue":2,"message":"Risque de coupures d'électricité","values":[{"pas":0,"hvalue":1},{"pas":1,"hvalue":1},{"pas":2,"hvalue":1},{"pas":3,"hvalue":1},{"pas":4,"hvalue":1},{"pas":5,"hvalue":1},{"pas":6,"hvalue":1},{"pas":7,"hvalue":2},{"pas":8,"hvalue":2},{"pas":9,"hvalue":2},{"pas":10,"hvalue":2},{"pas":11,"hvalue":2},{"pas":12,"hvalue":1},{"pas":13,"hvalue":1},{"pas":14,"hvalue":1},{"pas":15,"hvalue":1},{"pas":16,"hvalue":1},{"pas":17,"hvalue":2},{"pas":18,"hvalue":2},{"pas":19,"hvalue":1},{"pas":20,"hvalue":1},{"pas":21,"hvalue":1},{"pas":22,"hvalue":1},{"pas":23,"hvalue":1}]},

{"GenerationFichier":"2022-06-03T07:36:25+02:00","jour":"2022-06-03T00:00:00+02:00","dvalue":3,"message":"Coupures d'électricité en cours","values":[{"pas":7,"hvalue":3},{"pas":8,"hvalue":3},{"pas":9,"hvalue":1},{"pas":10,"hvalue":1},{"pas":11,"hvalue":1},{"pas":12,"hvalue":1},{"pas":13,"hvalue":1},{"pas":14,"hvalue":3},{"pas":15,"hvalue":3},{"pas":16,"hvalue":3},{"pas":17,"hvalue":3},{"pas":18,"hvalue":3},{"pas":19,"hvalue":3},{"pas":20,"hvalue":3},{"pas":21,"hvalue":2},{"pas":22,"hvalue":2},{"pas":23,"hvalue":2}]}]

Ce message contenant les données d'EcoWatt passe alors dans 3 différents blocks de nœuds correspondant aux 3 modes spécifiques de fonctionnement : mode délestage, mode régulation et mode verrouillage. Il y a des similitudes entre les codes de ces 3 modes, c'est pourquoi les explications pourront être plus brèves.

2.2 – Mode délestage

Parlant tout d'abord du code pour le mode délestage (entouré en rouge sur la figure 3). Rappelons d'abord le principe de ce mode délestage : « il consiste à débrancher automatiquement des équipements secondaires quand la consommation est élevée et dépasse un certain seuil. C'est le cas par exemple de ballon d'eau chaude ou de radiateurs. » Il faut ainsi regarder tout d'abord si les données d'EcoWatt avertissent d'une consommation élevée en France ou que la consommation électricité de l'utilisateur a dépassé un certain seuil puis des messages sont envoyés à la carte relais pour dire d'éteindre les équipements qui sont mis en mode délestage.

On a donc besoin en entrée de deux types de données : les données d'EcoWatt pour vérifier s'il y a un avertissement sur une consommation élevée d'électricité en France et les données de consommation de l'utilisateur (données du compteur Linky par exemple). La récupération des données d'EcoWatt a déjà été expliquée dans la partie précédente (2.1), il faut alors maintenant récupérer les données du compteur Linky.

Sur la figure 3, on peut remarquer que le nœud précédent le mode délestage est un nœud « inject ». En effet, il existe un nœud « Enedis_Linky » déjà codé par un utilisateur de Node-Red, mais pour pouvoir l'utiliser il nous fallait un compte Enedis que nous ne possédons pas. Ainsi nous ne pouvions pas utiliser ce nœud. La méthode que nous avons alors trouvée était, avec le nœud « inject », d'envoyer un message imitant le message qu'aurait envoyé le nœud « Enedis_Linky ». Le message comportait ainsi deux informations : la consommation du l'utilisateur et le seuil de consommation à ne pas dépasser.

Une fois les données récupérées elles passent par un nœud « function » nommé 'Mode délestage'. Ce nœud a été codé de façon à ce que si EcoWatt avertit d'une consommation élevée (dvalue = 2 ou dvalue = 3) ou que si la consommation de l'utilisateur a dépassé le seuil de consommation alors la sortie est le booléen *true* sinon le booléen *false*. Le message contenant le booléen passe alors par un nœud « switch » en jaune. Ce nœud « switch » ne correspond pas au nœud « switch » pour l'IHM précédemment vu dans la partie 1.1. Ce nœud permet de faire passer un message seulement si le message en entrée vérifie certaines propriétés qui sont définies par le codeur. Dans notre cas, on a choisi de faire passer le message si et seulement si le champ payload du message en entrée est le booléen *true*. (En effet, s'il n'y a aucun problème de consommation que ce soit pour l'utilisateur ou au niveau national, alors le mode de délestage ne s'applique pas donc il ne faut pas envoyer des messages permettant son activation).

Maintenant, il convient de créer des messages indiquant quel relai éteindre. Pour cela, nous utilisons dans le code un nœud « trigger » pour permettre entre autres d'envoyer un message toutes les secondes (ce nœud n'est activé que si le nœud « switch » précédent lui a envoyé un message en entrée). La fréquence d'envoi des messages peut être changé mais elle nous semblait assez correcte pour laisser assez de temps à la carte relais de traiter les messages et pour nous de voir l'envoi des messages. Ensuite, ces messages passent, au même instant, à la fois par deux nœuds « function » : 'définition clé_del' et 'Envoi'. C'est à ce moment que le fait qu'on ait créé un message global contenant les informations de l'IHM est important (cf. 1.3).

Le principe de la fonction 'définition clé_del' est qu'elle regarde dans message global les champs 'indice_del' et 'nombre_del' et renvoie *true* au nœud « clé » 'Clé_del' si indice_del est inférieur à nombre_del (et supérieur à 0), sinon *false*. 'Indice_del' correspond à l'indice du parcours du tableau 'tab_del' contenant les numéros de relai sur lesquels sont branchés les équipements fonctionnant en mode délestage, et 'nombre_del' au nombre d'équipements fonctionnant en mode délestage.

La fonction 'Envoi', elle, récupère aussi les données du message global puis parcourt le tableau 'tab_del' en envoyant à chaque fois un message avec le champ payload contenant le chiffre 0 et le champ topic correspondant au numéro du relai. Elle incrémente à chaque fois aussi l'indice de parcours 'indice_del' du tableau.

Ainsi en résumé, on parcourt la liste d'équipements mis en mode délestage et quand elle est parcourue entièrement, on coupe la transmission des messages grâce à un nœud « clé » qui ferme un nœud « porte » nommé 'Porte_off' dans le code.

Le plus gros du travail de la création des messages pour éteindre les relais est fait. Il reste à les convertir en des messages génériques que la carte relais puisse comprendre. Ceci sera expliqué dans la partie 2.6.

2.3 – Mode régulation

Voyons maintenant le fonctionnement du mode régulation (dont le code est entouré en violet sur la figure 3). Rappelons d'abord le principe de ce mode : « Le mode régulation consiste à lisser la courbe de consommation en alternant le fonctionnement de plusieurs équipements. Ce mode s'applique par exemple pour piloter plusieurs radiateurs allumés en même temps ».

Pour la mise en place du mode régulation il y a donc besoin seulement des données d'EcoWatt (et des données de l'IHM bien sûr). De la même manière que pour le mode délestage, les données d'EcoWatt passent par un nœud « function » 'Mode régulation'. Cette fonction, comme la fonction 'Mode délestage', renvoie un message dont le champ payload est le booléen *true* si EcoWatt avertit d'une consommation élevée (dvalue = 2 ou dvalue = 3) sinon le booléen *false*. Ainsi, contrairement à la fonction 'Mode délestage', on ne prend pas en compte les données du compteur Linky. En revanche, la fonction 'Mode régulation' possède une particularité puisqu'elle regarde en même temps les données de l'IHM via le message global, et plus particulièrement les champs indiquant le nombre d'équipements mis en mode régulation et le temps d'utilisation global pour ces équipements ('Temps_régulation' cf. 1.1) ; et ensuite calcule un temps d'utilisation pour chaque équipement (noté t_{util}). Ce temps t_{util} correspond tout simplement à la division du temps d'utilisation global par le nombre d'équipement mis en mode régulation. La fonction 'Mode régulation' renvoie aussi dans son message ce temps t_{util} à travers le champ delay.

Le message envoyé par la fonction ‘Mode régulation’ passe alors par un nœud « switch » qui, comme pour le mode délestage, laisse passer le message si et seulement si le champ payload contient le booléen *true*, sinon on le coupe.

Pour le fonctionnement du mode régulation, on fait ensuite passer le message par un nœud « trigger » ‘reset every t_{util}’ mais contrairement au nœud « trigger » du mode délestage, on envoie cette fois des messages toutes les t_{util} secondes.

Suivant le même principe que pour le mode délestage, on fait ensuite passer le message par un nœud « function » ‘définition clé_regu’ et en même temps par deux autres nœuds « function » ‘Envoi message ON’ et ‘Envoi message OFF’. Le principe de la fonction ‘définition clé_regu’ est qu’elle regarde dans message global les champs ‘indice_regu’ et ‘nombre_regu’ et renvoie *true* au nœud « clé » ‘Clé_regu’ si indice_regu est inférieur à nombre_regu (et supérieur à 0), sinon *false*. ‘Indice_regu’ correspond à l’indice du parcours du tableau ‘tab_regu’ contenant les numéros de relai sur lesquels sont branchés les équipements fonctionnant en mode régulation, et ‘nombre_regu’ au nombre d’équipements fonctionnant en mode délestage.

La fonction ‘Envoi message ON’, elle, récupère aussi les données du message global puis parcourt le tableau ‘tab_regu’ en envoyant à chaque fois un message avec le champ payload contenant le chiffre 1 et le champ topic correspondant au numéro du relai. Elle incrémente à chaque fois aussi l’indice de parcours ‘indice_regu’ du tableau. Le message passe ensuite par un nœud « porte » ‘Porte on’ qui laisse passer le message si et seulement si la clé ‘Clé_regu’ est activée.

De même, la fonction ‘Envoi message OFF’ récupère aussi les données du message global puis parcourt le tableau ‘tab_regu’ en envoyant à chaque fois un message avec le champ payload contenant le chiffre 0 et le champ topic correspondant au numéro du relai. Néanmoins elle n’incrémente pas à chaque fois l’indice de parcours ‘indice_regu’ du tableau puisque cela est géré par la fonction ‘Envoi message on’. Le message passe ensuite aussi par un nœud « porte » ‘Porte off’ qui laisse passer le message si et seulement si la clé ‘Clé_regu’ est activée. Enfin, le message arrive à un nœud « delay » ‘Délai’ qui, comme son nom l’indique, ajoute un délai à la transmission du message. Ce délai est alors égal à la valeur du champ msg.delay, soit à t_{util} secondes.

Ainsi en résumé, toutes les t_{util} secondes, on lit un élément de la liste des équipements mis en mode régulation et envoie deux messages : un pour allumer le relai correspondant et un pour l’éteindre. En revanche, le deuxième message qui sert à éteindre ce relai est transmis avec un délai de t_{util} secondes pour laisser l’équipement fonctionner pendant exactement t_{util} secondes comme il avait été prévu. Finalement quand la liste est parcourue entièrement, on coupe la transmission des messages grâce à un nœud « clé » qui ferme les nœuds « porte » ‘Porte_on’ et ‘Porte_off’.

2.4 – Mode verrouillage

Concernant le principe du mode verrouillage (entouré en vert sur la figure 3), « Le mode verrouillage s’applique quand l’utilisateur souhaite s’interdire l’usage d’un équipement quand le système électrique est tendu ou très tendu. C’est le cas par exemple du four ou du sèche-linge. »

Le code de ce mode est très similaire au code du mode de délestage. On commence de la même façon, par un nœud « function » ‘Mode verrouillage’ qui, plus simplement que le nœud ‘Mode délestage’, vérifie si EcoWatt avertit d’une consommation élevée (dvalue = 2 ou dvalue = 3). S’il y a un avertissement alors on laisse passer le message avec un nœud « switch » puis on envoie aussi des

messages toutes les secondes et on parcourt la liste d'équipements mis en mode verrouillage avec le nœud 'Envoi' (même s'il a le même nom que le nœud 'Envoi' du mode délestage, les deux ne différent par le fait qu'ils ne traitent pas les mêmes tableaux). Une fois qu'elle est parcourue entièrement, on coupe la transmission des messages grâce à un nœud « clé » 'Clé_ver' qui ferme un nœud « porte » nommé 'Porte_off' dans le code. Les messages transmis aussi à chaque fois avec le champ payload contenant le chiffre 0 et le champ topic correspondant au numéro du relai.

2.5 – Mode normal

Finissons l'explication des modes de fonctionnement par le mode normal qui permet d'allumer et d'éteindre normalement les relais. Le code apparait entouré en jaune sur la figure 2). Le principe est que l'on regarde la valeur des nœuds « dropdown » ('ModeSelector1', 'ModeSelector2', 'ModeSelector3' et 'ModeSelector4') qui correspondent au mode de fonctionnement du relai et avec des nœuds « function » ('Mode1', …) on vérifie pour chacun des relais si le mode sélectionné par l'utilisateur est le mode normal ou non. Si un relai est en mode normal alors cela active la clé qui lui est associé (Mode1 ou Mode2 ou …).

Lorsqu'une clé est activée, elle ouvre le nœud porte associé à cette clé. Le message provenant du nœud « switch » associé au relai qui a été mis en mode normal ('Relai1' ou 'Relai2' ou …), passe alors par un nœud « change » 'Change msg'. Ce nœud « change » est un nœud qui sert à définir, modifier, supprimer ou déplacer les propriétés d'un message. Dans notre cas, on l'utilise pour pouvoir changer le message provenant des nœuds « switch » ('Relai1', …) en un message dont le champ payload est 0 si le switch est désactivé et 1 s'il est activé, et dont le champ topic est le numéro du relai associé au switch (soit 1 ou 2 ou 3 ou 4).

La sortie de ce nœud 'change msg' est alors relié au nœud « function » 'Conversion msg' dont le fonctionnement est expliqué à la partie suivante.

2.6 – Allumage des relais

Cette dernière partie de la section 2 correspond à la transmission des messages d'allumage et de coupures d'alimentation des différents relais. On a vu que, à la fin du code de chaque mode de fonctionnement, un message est transmis et il est de la forme :

Msg.payload = 0 ou 1 et msg.topic = « numéro de relai associé »

La valeur 0 correspond au fait d'éteindre le relai et la valeur 1 au fait d'allumer le relai.

Le nœud « function » 'Conversion msg' est le nœud qui va assurer la conversion de ces messages en un message compris par la carte relais. Les messages deviennent ainsi des messages de la forme :

"A00"+msg.topic+"0"+msg.payload+"A"+msg.value

Avec msg.value= msg.topic + msg.payload

(msg.value sert à vérifier qu'il n'y a pas eu de perte de bit lors de la transmission du message)

Pour finir, ces messages passent par un dernier nœud ‘Relay’. Ce nœud est un nœud « TCP request » qui envoie le message au port tcp d'un serveur et donc à la carte relais qui comprend le message précédent et allume ou éteint les relais en fonction de ces messages. On ne peut envoyer des messages à la carte relais que si l’on est connecté à son wifi.

3 – Mode Manuel

Les sections précédentes ont décrit tout le processus de fonctionnement de notre code pour pouvoir satisfaire la problématique de la gestion des modes de fonctionnement. Cependant, nous nous sommes dit qu’il était préférable que nous application contienne aussi un mode manuel qui en cas de problème pourrait s’avérer très utile.

Sur la figure 2 est entouré en noir le code permettant le fonctionnement du mode manuel. Il s’agit simplement de 4 nœuds « switch » de l’IHM. Lorsqu’un est activé alors il envoie un message contenant dans le champ payload, le chiffre 1 et dans le champ topic, son numéro de relai associé. Ensuite, le nœud « function » fait exactement la même chose que le nœud ‘Conversion msg’ (on ne l’a juste pas renommé de la même manière), c’est-à-dire, renvoie au nœud ‘Relay’ le message "A00"+msg.topic+"0"+msg.payload+"A"+msg.value. Enfin le nœud ‘Relay’ envoie le message à la carte relais.