L’architecture du serveur python se structure autour d’un système multi-threads.

Nous aurons deux threads d’écoute (listeners) destinés à recevoir les trames envoyées par la passerelle EnOcean d’une part et par l’application Web (tablette) d’autre part.

A la réception d’une trame, ces threads utiliseront le parser pour trier les informations, rejetteront immédiatement les trames incorrectes puis feront passer ces informations dans la base d’informations (architecture plus détaillée de celle-ci à demander à V et A) avant d’envoyer un signal « s1 » au thread Command.

Le thread Command crée est à l’origine d’un Timer qui renvoi à chaque cycle un signal « s1 » défini qui a faire réagir le thread Command (par un handler) et qui le force à faire appel au moteur de règles.

Le moteur de règle examine la base d’information où sont stockées les informations contenues dans les trames reçues (éventuellement un historique de ces informations si cela est nécessaire à l’interprétation d’une règle) et à partir d’un ensemble de pré-conditions définies statiquement, choisi une ou plusieurs commandes (ie une information à encoder au sein d’une trame) et le thread Command crée alors autant de threads Sender que de commandes à envoyer (celles-ci peuvent être envoyées à la passerelle ou à l’application Web).

A la réception d’une commande, le Sender encode la commande (ie l’information dans une trame compréhensible par le système EnOcean ou selon le formalisme choisit pour l’application Web (à définir avec M) puis envoie l’information sur le port de communication.

Note : Les ports des différentes machines, les identifiants des capteurs, les règles de décision des commandes doivent être définies statiquement dans le programme ou dans un fichier.

Note : Penser à réarmer le Timer à chaque fin de boucle.