Le fichier de configuration du nœud ConnecSenS

Le comprendre et l'utiliser

Révision 5 – Juin 2018
Applicable à partir de la version 0.7 du firmware du nœud

1 A quoi sert le fichier de configuration ?

Il permet de personnaliser le fonctionnement du nœud ConnecSenS, pour l'adapter à vos besoins, dans la limite des options prévues.
En particulier, il permet de :

- Configurer les paramètres réseaux utilisés par le nœud pour se connecter au réseau sans fil.
- D'informer le nœud des capteurs qui lui sont connectés.
- De configurer les capteurs en question.
- De configurer les périodes de mesure des capteurs et la période d'émission des données sur le réseau sans fil.

2 Où se trouve-t-il ?

Le nœud est équipé d'une mémoire interne, constituée d'une carte SD, sur laquelle le nœud conserve ses mesures, et où figurent également d'autres fichiers nécessaires à son fonctionnement, dont le fichier de configuration. Cette mémoire est accessible au moyen du connecteur USB dont est équipé le nœud.

Si vous connectez le nœud à votre ordinateur au travers d'un câble mini-USB vers le format USB présent sur votre poste de travail, alors vous devriez voir apparaître celui-ci comme un support de stockage de masse (comme une clef USB). Le premier raccordement peut demander un peu de temps, et votre système d'exploitation peut vous afficher différents messages pour vous indiquer la détection d'un matériel jusqu'alors inconnu.

Si tout se passe bien alors vous devriez pouvoir consulter, et modifier, le contenu de la mémoire du nœud, au travers de votre explorateur de fichier par exemple. Le nom du disque monté devrait être du type « ConnecSenS ».

Le fichier de configuration se trouve dans le dossier `env`, à la racine du disque nouvellement monté, et son nom est `config.json`. Comme son extension l'indique, il utilise le format JSON, un format texte d'échange et de stockage de données.

3 A quoi ressemble-t-il ?

Vous trouverez à la page suivante un exemple de fichier de configuration, pour vous donner une idée de ce à quoi il ressemble.

Quelques remarques :

- Un simple éditeur de fichier texte, tel que Notepad par exemple, suffit pour modifier ou créer ce fichier.
- Le format JSON se base sur un système de clef-valeur où la clef est la chaîne de caractères entre guillemets à la gauche du caractère « deux points », et la valeur est à la droite de ce caractère séparateur. La clef doit être unique dans un même « niveau d'accolades ». La valeur peut être une chaîne de caractères (entre guillemets), un nombre entier ou réel, un booléen (`true` ou `false`) , une valeur nulle (`null`), un objet (liste de clefs-valeurs entre accolades, où chaque paire clef-valeur est séparée de la suivante par une virgule) , ou un tableau (une liste de valeurs)(entre crochets, ces valeurs peuvent être simples ou complexes).
- Les clefs utilisées commencent toujours par un caractère minuscule et les mots qui les composent sont séparés par un caractère majuscule. Les clefs sont sensibles à la case, c'est-

à-dire qu'il faut respecter l'utilisation des majuscules et des minuscules, sinon la clef ne sera par reconnue par le nœud. Exemple de clef : `nameOfAParameter`.

Voici un exemple de fichier de configuration :

```
{
    "name": "ConnecSenS_1",
    "verbose": true,
    "led": true,
    "debugSerial": false,

    "interruptions": [{
        "name": "INT1_INT",
        "debounceMs": 1000
    }, {
        "names": ["USART_INT", "SDI12_INT", "SPI_INT"],
        "debounceMs": 10
    }],

    "network": {
        "type":   "LoRaWAN",
        "devEUI": "0203040506070609",
        "appEUI": "A78F1729918331B4",
        "appKey": "D7F66C4B228A7DF609000006A6579FC1",
        "periodSec": 3600,
        "dataRate": 1,
        "useAdapativeDataRate": false
    },

    "sensors": [{
        "name": "Press",
        "type": "LPS25",
        "periodSec": 600,
        "interruptChannel":"LPS25_INT",
        "sendOnInterrupt": true,
        "alarm": {
            "sendOnAlarmSet": true,
            "lowThresholdHPa":  900,
            "highThresholdHPa": 1100
        }
    }, {
        "name": "Lumi",
        "type": "OPT3001",
        "periodSec": 300,
        "interruptChannels": ["OPT3001_INT", "INT1_INT"],
        "sendOnInterrupt": false
    }, {
        "name": "Pluvio1",
        "type": "RainGaugeContact",
        "periodSec": 3600,
        "tickInterrupt":  "INT2_INT",
        "tickDebounceMs": 500,
        "rainMMPerTick":  0.2,
        "interruptChannel": "OPTO1_INT",
        "sendOnInterrupt":  true,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": true,
            "thresholdSetMMPerMinute":  10.0,
            "thresholdClearMMPerMinute": 7.0
        }
    }],

    "time": {
        "syncGPS": false,
        "syncGPSTimeoutSec": 240,
        "syncPeriodSec": 2592000,
        "manualUTC": {
            "year": 2018,
            "month": 6,
            "day": 3,
            "hours": 9,
            "minutes": 50,
            "seconds": 0
        }
    }
}
```

4 Les différentes parties, ou sections, du fichier de configuration

Nous pouvons distinguer quatre parties :

- La configuration générale du nœud, qui correspond aux lignes 2 à 5 du fichier d'exemple donné à la page précédente.
- La configuration des lignes d'interruption, aux lignes 7 à 13.
- La configuration de la communication sans fil, visible aux lignes 15 à 23.
- La configuration des capteurs, aux lignes 25 à 57.
- La configuration de la date et de l'heure, aux lignes 59 à 71.

Étudions-les à présent plus en détails.

5 En cas d'erreur dans le fichier de configuration

Si vous faites une erreur de syntaxe, ou s'il y a une erreur grave avec au moins l'un des paramètres de la configuration, alors le nœud se bloque en état de défaut. Cet état de défaut est signalé par le clignotement rapide et simultané des deux LEDs du nœud. Des messages d'erreur sont également écrits dans le fichier de log, mais peuvent ne pas être visibles sur la ligne série de débogage si vous l'avez activée dans la configuration et si l'erreur se produit avant que le paramètre d'activation de la sortie de débogage série ne soit lu ; c'est le cas par exemple en cas d'erreur syntaxique.

Pour comprendre la source du problème il faut donc relier en USB le nœud à votre ordinateur et appuyer sur le bouton « reset » du nœud (car le nœud ne détecte pas automatiquement la connexion USB comme en temps normal).

6 La configuration générale

`name`: (**obligatoire**) définit le nom du nœud. Sa longueur maximale est de quinze caractères.

`verbose`: (optionnel, valeur par défaut : `false`) valeur `true` ou `false`. Configure la quantité de logs (de messages sur le fonctionnement du nœud) qui est écrite en mémoire interne. Il est conseillé de positionner ce paramètre à `true`, car les logs sont la meilleure chance pour nous de trouver la source des bugs et autres disfonctionnements du nœud. Plus nous avons de logs, plus le support est facile. En revanche, l'écriture de logs consomme de la mémoire interne. Ce dernier point ne devrait toutefois pas être limitant car la mémoire interne est de plusieurs gigaoctets et la quantité de logs nécessaire pour la saturer est par conséquent très élevée.

`led`: (optionnel, valeur par défaut : `false`) valeur `true` ou `false`. Configure l'activation, ou non, des LEDs de diagnostique. Le nœud est équipé de deux LEDs internes pour indiquer son fonctionnement, son état. Puisque ces LEDs ne sont pas visibles un fois le boîtier fermé, il est conseillé de mettre ce paramètre à `false`. Le nœud propose également deux LEDs en façade, visibles par l'utilisateur, mais elles sont pour le moment inutilisées. Il faudrait à l'avenir qu'elles soient au minimum le reflet des LEDs internes.

`watchdog`: (optionnel, valeur par défaut : `true`) valeur `true` ou `false`. Active ou désactive le watchdog (chien de garde). C'est un dispositif qui à pour but de détecter les plantages éventuels du nœud et de remettre ce dernier à zéro le cas échéant. Il est très fortement conseillé, voire il est impératif, de positionner ce paramètre à `true`.

`debugSerial`: (optionnel, valeur par défaut : `false`)valeur `true` ou `false`. Active, ou non, la sortie série de débogage du nœud. Il est optionnel et son absence désactive la sortie de débogage série. Pour l'utiliser il faut relier un adaptateur série 3,3 V vers USB au nœud. Ce type d'adaptateur n'est pas un modèle courant, vendu pour ajouter un port série à un PC, mais un modèle plus orienté développement. Les signaux série ne sortent par sur un connecteur de type DB9 aux niveaux RS232, mais sur des picots individuels avec des tensions comprises entre 0 V et 3,3 V. Si vous êtes équipés d'un tel adaptateur, alors vous pouvez connecter sa ligne de masse à l'un des picots GND du connecteur d'extension interne du nœud et sa ligne Rx sur le picot USART_TX du connecteur d'extension interne. Configurez ensuite le logiciel de communication série que vous utilisez sur votre ordinateur avec les paramètres suivants : 115200 bauds, 8 bits, pas de parité, 1 bits de stop, pas de contrôle de flux matériel. Si le paramètre `debugSerial` est positionné à `true` dans le fichier de configuration du nœud, alors vous verrez apparaître en temps réel sur votre console les messages qui sont écrits dans le fichier de log du nœud. Vous êtes ainsi en mesure de suivre le fonctionnement du nœud en direct.

`workingMode`: (optionnel, valeur par défaut : fonctionnement normal) permet de basculer le nœud dans des modes de fonctionnement alternatifs. Les modes de fonctionnement alternatifs sont :

- `"campaignRange"`: Mode de cartographie des portées LoRaWAN sur le terrain. L'objectif de ce mode de fonctionnement est de faciliter la création des cartes de couverture radio. Dans ce mode, le nœud réalise des synchronisations GPS en permanence, toutes les cinq secondes environ, pour obtenir l'heure et surtout sa position GPS. Régulièrement, selon la période configurée dans la partie réseau qui nous étudierons dans le chapitre suivant, le nœud transmet son heure en UTC, sa position et le datarate LoRaWAN utilisé pour cette transmission. Il alterne le datarate entre deux transmissions consécutives : la première est faite en DR0 (soit celui au débit le plus faible mais à la portée la plus longue), la deuxième en DR5 (soit celui avec le débit le plus important mais à la portée la plus faible), puis DR0 à nouveau, puis DR5, et ainsi de suite… Lorsque le nœud arrive à communiquer avec la passerelle alors il est en mesure d'évaluer la force du signal (RSSI) et son rapport signal sur bruit (SNR). En plus de transmettre ces données, le nœuds les écrits dans un fichier CSV sur sa carte SD interne. Il est ainsi aisé d'exploiter les données à la fin de la campagne de mesure. Il faut toutefois noter qu'à l'heure où j'écris ces lignes la transmission RF n'est pas complètement opérationnelle. Pour une raison qui m'échappe encore le nœud est capable de joindre la passerelle, mais pas de lui envoyer un message de données. Cette communication RF limitée est toutefois suffisante pour récupérer les informations de force et de qualité du signal, qui sont écrites dans le fichier CSV évoqué précédemment. Ce fichier se trouve dans le dossier `campaign/range` de la carte SD et son nom est la date à laquelle la campagne de mesure a eu lieu.

7 La configuration des lignes d'interruption

Une interruption est la détection d'un évènement. Par exemple, il est possible de fixer des seuils d'alarme sur les capteurs internes, et lorsque l'un de ces seuils est franchi alors une interruption est générée. Les capteurs externes peuvent également générer des interruptions aux moyen de lignes dédiées sur le connecteur d'extension interne et qui peuvent être sorties sur les connecteurs M12 du nœud.

La configuration des lignes d'interruption se fait par la section nommée `interruptions`. Cette section est optionnelle et en son absence toutes les lignes d'interruption utilisent les valeurs par défaut. Cette section est un tableau d'objets JSON. Chaque objet représente une configuration spécifique déterminée par les paramètres suivants :

`name` : (**obligatoire**, sauf si `names` est défini) spécifie le nom de la ligne d'interruption à laquelle la configuration s'applique. Le nom n'est pas sensible à la case ; vous pouvez aussi bien l'écrire en minuscules qu'en majuscules.

`names`: (**obligatoire**, sauf si `name` est défini) spécifie la liste des noms des lignes d'interruption auxquelles la configuration s'applique. C'est un tableau JSON. Les noms ne sont pas sensibles à la case. Ce paramètre permet de facilement configurer des lignes de manière identique, et avec concision. Il est possible d'utiliser à la fois les paramètres `name` et `names` si vous le souhaitez, bien que cela revienne simplement à rajouter un élément au tableau `names`.

La liste des noms des lignes d'interruption du nœud ConnecSens est la suivante :

- `SHT35_INT`, l'interruption levée lorsque l'un des seuils d'alarme fixés au capteur interne de température et d'humidité `SHT35` est franchi.
- `LPS25_INT`, celle levée par le capteur interne de pression `LPS25` en cas de dépassement de l'un des seuils d'alarme fixés.
- `OPT3001_INT`, celle générée par le capteur interne de luminosité `OPT3001` en cas de dépassement du seuil d'alarme fixé.
- `LIS3DH_INT`, l'interruption levée par l'accéléromètre interne `LIS3DH` si la détection de mouvement est activée (une alarme) et qu'un mouvement a été détecté.
- `SPI_INT`, l'interruption associée au bus SPI externe. Sensible uniquement à un front montant (passage d'un niveau bas 0 V à un niveau haut de 5 V au maximum).
- `I2C_INT`, celle pour le bus I2C externe. Sensible uniquement à un front montant montant.
- `USART_INT`, l'interruption associée aux lignes UART (séries) externes. Sensible uniquement à un front montant montant.
- `SDI12_INT`, l'interruption associée à l'interface SDI-12 externe. A partir de la version 0.7.1 du firmware l'alias `SDI_INT` est également supporté pour être en accord avec les noms sérigraphiés autour du connecteur d'extension interne. Sensible uniquement à un front montant montant.

- `OPTO1_INT` et `OPTO2_INT`, associées aux entrées opto-isolées 1 et 2 respectivement. A partir de la version 0.7.1 du firmware les alias `OPTO_1` et `OPTO_2` sont également supportés. Sensibles uniquement à un front montant. Ces entrées doivent avoir une résistance montée en série pour limiter le courant circulant au travers des LEDs des optocoupleurs.
- `INT1_INT` et `INT2_INT`, des interruptions externes à usage général. A partie de la version 0.7.1 du firmware les alias `INT_1` et `INT_2` sont également supportés. Sensibles uniquement à un front montant et à une tension maximale de 5 V.

Le fichier de configuration n'est pas sensible à la case du nom des interruptions, il est donc possible de les écrire exactement de la même manière que les noms sérigraphiés autour du connecteur d'extension interne (surtout à partir de la version 0.7.1 du firmware).

Une fois le nom, ou les noms, des interruptions concernées spécifiés, il est alors possible de régler :

`debounceMs`: (optionnel, valeur par défaut : 0) le temps d'anti-rebonds à utiliser. Si une ligne d'interruption est reliée à une interrupteur mécanique, comme un bouton poussoir ou à certains types de pluviomètres à auget par exemple, alors elle peut être victime de « rebonds ». Lorsqu'un interrupteur mécanique entre en contact, ou inversement rompt un contact, la transition marche/arrêt n'est pas nette. Il se produit des oscillations rapides qui peuvent être détectées par le nœud. Ainsi, un contact, ou une rupture de contact, se traduit souvent par plusieurs interruptions au lieu d'une seule. Pour venir à bout de cette problématique il est possible de définir un délai d'anti-rebonds. Avec ce délai en place, la première interruption est prise en compte, mais toutes les suivantes qui se produisent pendant la période d'anti-rebonds sont ignorées. Si vous spécifiez un délai de 500 ms par exemple, alors vous obtiendrez au maximum deux interruptions prises en compte par seconde. Si vous spécifiez un délai de 0 ms, alors le dispositif anti-rebonds est désactivé. Cette valeur de 0 est celle souvent la plus appropriée lorsque la ligne d'interruption est pilotée par un signal électronique ou par un interrupteur mécanique qui incorpore un dispositif anti-rebonds analogique.

8 La configuration réseau

`periodSec`: (**obligatoire**) est la période d'émission des données, en secondes. Dans le fichier d'exemple donné plus haut cette valeur est fixée à 3600 secondes, les données seront donc transmises toutes les heures. Il est toutefois conseillé d'utiliser une période proche de celle des mesures capteurs car la quantité de données que peut contenir une trame sans fil est limitée à quelques dizaines d'octets et nous ne pouvons occuper les ondes qu'un pourcent du temps au maximum (le nombre de trames consécutives que nous pouvons envoyer est donc limité). De même, cette limitation du temps d'occupation des ondes contraint également la période minimale d'envoi et de mesure des capteurs. Ainsi, il est difficile de descendre en dessous d'une période d'une minute pour la mesure des capteurs et pour la transmission des données.

`type`: (**obligatoire**) indique le type de réseau sans fil utilisé. A l'heure actuelle la seule valeur valide est `LoRaWAN`. Nous nous sommes laissé la possibilité de pouvoir utiliser le réseau Sigfox, mais cette option n'est pas encore fonctionnelle.

`devEUI`: (**obligatoire** pour `LoRaWAN`)est l'identifiant unique du nœud ConnecSenS. Ce paramètre est spécifique au réseau LoRaWAN. Aucun nœud du réseau de capteurs ne doit avoir le même `devEUI`. Nous devrons certainement à l'avenir mettre en place une procédure d'attribution de cet identifiant. Normalement, vous ne devriez donc jamais avoir à changer cette valeur. Pour information, cet identifiant est défini en hexadécimal et est d'une longueur de 8 octets (soit 64 bits). Pour les plus informaticiens d'entre-vous cet identifiant peut-être comparé à une adresse MAC.

`appEUI`: (**obligatoire** pour `LoRaWAN`) est l'identifiant d'application. Ce paramètre est spécifique au réseau LoRaWAN. Il sert à identifier l'application, côté serveur, qui sera en charge de traiter les messages envoyés par le nœud. Cet identifiant doit vous être communiqué par la personne, ou par l'outil informatique, en charge de la gestion de l'application associée à votre nœud. C'est un identifiant définit en hexadécimal et d'une longueur de 8 octets (soit 64 bits).

`appKey`: (**obligatoire** pour `LoRaWAN`) est une clef d'authentification auprès de l'application. Ce paramètre est spécifique au réseau LoRaWAN. Elle permet d'autoriser ou non le nœud à communiquer avec l'application et elle est ensuite utilisée pour dériver les autres clefs permettant de sécuriser les communications une fois que le nœud a joint le réseau LoRaWAN. Cette clef peut être partagée par tous les nœuds en relation avec l'application, ou chaque nœud peut avoir sa propre clef. Cette dernière solution est la plus sécurisée, puisque dans ce cas une clef compromise affecte un seul nœud et non pas tous les nœuds liés à l'application. Cette clef est définie en hexadécimal et est d'une longueur de 16 octets (soit 128 bits). Comme pour l'`appEUI`, celle-ci devrait vous être fournie par la personne, ou par l'outil informatique, en charge de la gestion de l'application avec laquelle communique votre nœud.

`useAdapativeDataRate`: (optionnel, valeur par défaut : `false`) active (`true`) ou non (`false`) l'utilisation d'un datarate LoRaWAN adaptatif. L'ajustement automatique du datarate permet d'optimiser le réseau et la consommation d'énergie des nœuds qui le composent. C'est la passerelle (ou plutôt le LoRa network server qui se cache derrière) qui envoie des commandes individuelles aux nœuds pour optimiser la communication entre la passerelle et chaque nœud, mais également pour optimiser le fonctionnement global du réseau. Il n'est pas conseillé d'utiliser ce mode de fonctionnement pour les nœuds mobiles, ou si la passerelle est mobile, car la vitesse d'adaptation du réseau est très lente.

`dataRate`: (optionnel, valeur par défaut : 5) indique le datarate LoRaWAN à utiliser lors de la transmission. Valeur comprise entre 0 et 5, bornes incluses. Cette valeur est uniquement utilisée si `useAdapativeDataRate` est à `false`. Plus le chiffre est élevé plus la vitesse de transmission est élevée et plus une trame peut contenir de données, mais moins de distance elle peut couvrir. Inversement, une réduction de ce chiffre augmente la portée de la communication sans fil mais réduit la vitesse de transmission et la quantité de données transportées. Il convient de configurer le chiffre le plus élevé utilisable dans votre situation, car vous réduisez ainsi également le temps d'occupation des ondes et la consommation d'énergie de votre nœud (vous augmentez donc son autonomie).

9 La configuration de la date et de l'heure du nœud

Pour que le nœud puisse se réveiller périodiquement pour effectuer des mesures, et pour horodater celles-ci, il a besoin de connaître la date et l'heure. Le nœud travaille en heure universelle UTC.

Deux modes de mise à jour de l'heure sont proposés : par GPS et par configuration manuelle. Il est fortement conseillé d'utiliser le mode GPS, car il est plus précis et limite grandement les risques d'erreur. La solution manuelle est proposée pour faire face aux cas où l'utilisation du GPS serait impossible.

La mise à l'heure du nœud se fait au travers de la section `time` du fichier de configuration. Elle est visible aux lignes 59 à 70 du fichier d'exemple donnée au début de ce document. Dans le fichier d'exemple en question la synchronisation GPS est désactivée au profit du réglage manuel.

9.1 Synchronisation du temps par GPS

Pour activer la synchronisation GPS il faut positionner le paramètre `syncGPS` à `true` et indiquer une période de resynchronisation en secondes au moyen du paramètre `syncPeriodSec`.

L'horloge interne du nœud ConnecSenS est pilotée par un quartz et sa dérive dans le temps devrait être limitée. Toutefois avec le temps, et les variations de températures, un décalage est inévitable. L'utilisation du GPS consomme beaucoup d'énergie, il est donc conseillé d'y recourir avec parcimonie. En tenant compte de ces informations, et probablement en observant votre nœud, ou vos nœuds, vous pourrez déterminer la meilleure période à utiliser. Si vous manquez d'inspiration alors vous pouvez essayer une resynchronisation GPS par semaine (soit toutes les 604800 secondes), voire même une fois par mois (soit 18000000 secondes environ).

`syncGPSTimeoutSec`: (optionnel, valeur par défaut 120 secondes) règle le temps d'attente maximal pour obtenir une accroche GPS. La valeur minimale acceptée est de 60 secondes. Dans de bonnes conditions le temps d'accroche est d'environ 30 secondes. En revanche, lorsque les conditions de réceptions sont plus difficiles, comme dans des bâtiments ou dans des lieux encaissés, ce temps d'accroche peut être beaucoup plus long. Le temps d'attente par défaut, fixé à 120 secondes devrait être suffisant pour la majorité des situations. Néanmoins, dans les cas les plus difficiles, il pourrait être nécessaire de rallonger ce temps d'attente. Il faut toutefois noter que la synchronisation GPS est bloquante, c'est-à-dire qu'il est impossible de faire une lecture capteur ou de réagir à une interruption lors d'une synchronisation GPS. Cette limitation devrait disparaître à l'avenir. En attendant, il est donc conseillé d'utiliser un temps d'attente pas trop long pour éviter de « bloquer » le nœud pour une trop longue période en cas d'accroche GPS difficile. De plus, la synchronisation GPS est très énergivore, il faut donc veiller à réduire autant que possible son activité.

9.2 Synchronisation du temps manuelle

Pour que la synchronisation manuelle soit prise en compte il faut désactiver la synchronisation du temps par GPS en passant le paramètre `syncGPS` à `false`.

Ensuite, il faut indiquer l'heure courante, en heure universelle UTC (donc moins une heure en hivers et moins deux heures en été), à l'intérieur du paramètre `manualUTC` (`manual` pour la version 0.7 du firmware). Sa valeur est un objet où sont indiqués les différents éléments de la date : `year`

pour l'année, `month` pour le mois, `day` pour le jour du mois, `hours` pour l'heure au format 24h (et non pas AM/PM), `minutes` pour les minutes et `seconds` pour les secondes. La nouvelle date sera prise en compte dès que vous débrancherez le câble USB du nœud ou de votre ordinateur. Il est donc difficile d'obtenir une précision de mise à l'heure inférieure à quelques secondes, voire quelques dizaines de secondes.

Il faut par ailleurs noter que le mode de synchronisation manuelle utilisera toujours la date indiquée dans le fichier de configuration après avoir été branché en USB. Il faut donc soit penser à mettre à jour la date à chaque branchement en USB du nœud, ce qui est fastidieux et présente de grands risques d'oubli et donc de données horodatées dans la passé, ou bien supprimer les informations de date du fichier de configuration après la mise à l'heure. Cette dernière méthode est conseillée. Le principe est simple : vous branchez votre nœud en USB une première fois pour configurer sa date manuellement au moyen du fichier de configuration, vous le débranchez ensuite pour qu'il se mette à l'heure, puis vous le branchez à nouveau en USB, vous conservez un mode de fonctionnement de mise à l'heure manuelle mais vous supprimez les paramètres de date ou vous les mettez tous à 0. Dès lors, à chaque débranchement du câble USB, l'heure courante du nœud sera conservée.

10 La configuration des capteurs

Le nœud ConnecSenS contient des capteurs internes qui sont disponibles au cas où ils peuvent vous être utiles. Ils peuvent mesurer les grandeurs suivantes :

- La pression atmosphérique, au travers du capteur nommé `LPS25`.
- Le degré d'humidité relative et la température de l'air, au moyen du capteur `SHT35`.
- La luminosité, avec le capteur `OPT3001`.
- L'accélération et les mouvements du nœud, avec la capteur `LIS3DH`.

A ces capteurs internes il est bien entendu possible d'ajouter des capteurs externes, directement en accord avec vos besoins, par le biais des connecteurs M12 offerts par le nœud ConnecSens.

Voyons les paramètres qui permettent de déclarer les capteurs utilisés par le nœud et de configurer ces derniers. Commençons par les paramètres généraux :

`sensors`: dont la valeur est un tableau qui contient la liste de tous les capteurs utilisés par le nœud et leur configuration spécifique. Chaque capteur correspond à un objet JSON (une série de paires clef-valeur comprise entre accolades), séparé du suivant par une virgule. Dans l'exemple donné plus haut, les lignes 25 à 36 déclarent et configurent un premier capteur, les lignes 36 à 42 un deuxième et les lignes 42 à 57 un troisième.

Ensuite, pour chaque capteur, il faut définir les paramètres suivants :

`name`: (**obligatoire**) le nom du capteur dont la longueur maximale est de 15 caractères. Il doit être unique, différent du nom de tous les autres capteurs déclarés. Ce nom est choisi par l'utilisateur.

`type`: (**obligatoire**) le type de capteur. C'est un nom, un identifiant, prédéfini qui permet au nœud de savoir quel est le type du capteur, et donc la façon de communiquer avec lui et le format des données à envoyer par le réseau sans fil. La liste des capteurs actuellement reconnus est:

- `SHT35`, pour le capteur interne d'humidité et de température.
- `LPS25`, pour le capteur interne de pression atmosphérique.
- `OPT3001`, pour le capteur interne de luminosité.
- `LIS3DH`, pour le capteur interne de mouvement.

A l'heure où sont écrites ces lignes aucun capteur externe (connecté aux connecteurs M12 du nœud ConnecSenS) n'est encore supporté.

`periodSec`: (**obligatoire**) la période de mesure, en secondes, du capteur. Comme indiqué précédemment, du fait des faibles débits et des contraintes d'occupation de l'espace Hertzien, il est déconseillé d'utiliser des périodes de mesure inférieures à la minute.

Des mesures capteurs peuvent également être réalisées sur interruptions en plus des mesures périodiques programmées.

Il n'est pas obligatoire d'associer l'interruption d'un nom donné avec le capteur ou l'interface qui lui serait logiquement associé. C'est particulièrement vrai pour les interruptions externes, dont l'association réelle est déterminée par le câblage entre le connecteur d'extension interne et les connecteur M12 externes. Mais c'est également vrai pour les capteurs internes. Si pour votre application l'association d'une interruption externe et d'un capteur interne fait sens, alors cette configuration est possible. Nous pourrions par exemple imaginer une situation où vous voudriez lancer une mesure de la pression atmosphérique au moyen du capteur interne lorsque l'entrée externe opto-isolée 1 passe au niveau haut, il alors est tout à fait possible d'associer l'interruption `OPTO1_INT` au capteur interne `LPS25`.

La configuration des interruptions pour chaque capteur se fait au moyen des paramètres suivants :

`interruptChannel`: (optionnel) dont la valeur est le nom de l'interruption associée au capteur. Le nom doit faire partie de la liste des interruptions donnée plus haut. Lorsque l'interruption désignée se produira alors une mesure du capteur en question sera effectuée. Ce paramètre est optionnel ; si vous ne souhaitez pas réaliser de mesure sur interruption avec ce capteur, alors omettez ce paramètre.

`interruptChannels`: (optionnel) permet de spécifier plusieurs interruptions en mesure de déclencher une mesure du capteur. Sa valeur est un tableau de noms d'interruptions. Ce tableau peut contenir un ou zéro éléments. Si le paramètre `interruptChannel` est également indiqué alors sa valeur est ajouté à la liste des interruptions auxquelles le capteur est sensible. Ces deux paramètres peuvent donc être utilisés simultanément, bien que cela n'est pas vraiment d'utilité ni de sens ; il est préférable de choisir l'un ou l'autre.

`sendOnInterrupt`: (optionnel, valeur par défaut : `false`) valeur `true` ou `false`. Il configure s'il faut envoyer les données après qu'un capteur ai réalisé une mesure pour cause d'interruption (`true`)

ou attendre le prochain envoi périodique programmé (`false`). Si un envoi anticipé est choisi, alors sachez que toutes les données en attentes d'envoi seront également expédiées, et pas uniquement celles de la mesure sur interruption.

`use5VWhenActive`: (optionnel, disponible à partir de la version 0.7.1, valeur par défaut dépendante du capteur). Force l'activation de l'alimentation externe 5V lorsque le nœud est actif. Si ce paramètre est absent ou à `false` alors l'utilisation ou non de l'alimentation 5V dépend du driver du capteur en question : si ce capteur a besoin de l'alimentation 5V pour fonctionner lorsque le nœud est actif, alors cette alimentation est activée par le driver, sinon elle est éteinte. Ce paramètre de « forçage » ne devrait donc probablement jamais être employé par l'utilisateur ; sa fonctionnalité principale est l'aide au débogage.

`use5VWhenAsleep`: (optionnel, disponible à partir de la version 0.7.1, valeur par défaut dépendante du capteur). Force l'activation de l'alimentation externe 5V lorsque le nœud est en sommeil. Si ce paramètre est absent ou à `false` alors l'utilisation ou non de l'alimentation 5V dépend du driver du capteur en question : si ce capteur a besoin de l'alimentation 5V pour fonctionner lorsque le nœud est en sommeil, alors cette alimentation est activée par le driver, sinon elle est éteinte. Ce paramètre de « forçage » ne devrait donc probablement jamais être employé par l'utilisateur ; sa fonctionnalité principale est l'aide au débogage.

Chaque capteur peut ensuite contenir une configuration spécifique à son type. Cette configuration spécifique est requise ou optionnelle selon les capteurs. Étudions ces paramètres spécifiques pour les capteurs actuellement gérés par le nœud ConnecSenS.

10.1.1 Configuration des alarmes

Certains capteurs sont capables de produire des alarmes en mesure de réveiller le nœud pour que celui-ci puisse faire une mesure du capteur en question, et éventuellement pour envoyer les données les plus récentes. La configuration de ces alarmes se fait au moyen d'une section nommée `alarm`. C'est un objet JSON qui contient des paramètres qui peuvent être applicables à tous types de capteurs et des paramètres spécifiques au capteur configuré.

Étudions ici les paramètres généraux, applicables à tout capteur susceptible de générer une alarme :

`sendOnAlarmSet`: (optionnel, valeur par défaut : `false`) indique si un envoi est déclenché lorsque l'alarme est activée.

`sendOnAlarmCleared`: (optionnel, valeur par défaut : `false`) indique si un envoi est déclenché lorsque l'alarme est désactivée, lorsque la ou les valeurs suivies repassent sous le seuil d'alarme. Cette option n'est pas supportée par tous les capteurs.

10.2 Configuration spécifique du capteur d'humidité et de température SHT35

Cette configuration est optionnelle. Elle est uniquement nécessaire si vous souhaitez configurer l'alarme offerte par ce capteur. Tous les seuils doivent être renseignés pour que l'alarme soit activée. Ces paramètres se trouvent dans la section `alarm` du capteur. Ce capteur ne supporte pas le

paramètre `sendOnAlarmCleared` et son utilisation est donc sans effet. Les paramètres à positionner sont les suivants :

`humidityHighSetPercent`: le taux d'humidité relative, en pourcents (mais sans le caractère « % »), de déclenchement de l'alarme d'humidité haute.

`humidityHighClearPercent`: le taux d'humidité relative qui met fin à l'alarme d'humidité haute. En coordination avec le paramètre précédent il permet de mettre en œuvre une hystérésis.

`humidityLowSetPercent`: le taux d'humidité relative, en pourcents, en dessous duquel une alarme se déclenche.

`humidityLowClearPercent`: le taux d'humidité au dessus duquel l'alarme pour humidité basse est arrêtée. Comme pour l'alarme haute ce paramètre permet d'introduire une hystérésis.

`temperatureHighSetDegC`: la température, en degrés Celsius, de déclenchement de l'alarme de température haute.

`temperatureHighClearDegC`: la température, en degré Celsius, qui coupe la voix à l'alarme de température haute. Encore une fois il est possible d'introduire une hystérésis sur l'alarme avec ce paramètre.

`temperatureLowSetDegC`: la température, en degrés Celsius, sous laquelle se déclenche l'alarme de température basse.

`temperatureLowClearDegC`: la température, en degré Celsius, au-dessus de laquelle l'alarme de température basse est rendue muette. Permet d'introduire une hystérésis.

Voici un exemple de configuration :

```
    "sensors": [{
        "name": "TempHumi",
        "type": "SHT35",
        "periodSec": 1800,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": false,
            "humidityHighSetPercent":   75,
            "humidityHighClearPercent": 70,
            "humidityLowClearPercent":  15,
            "humidityLowSetPercent":    10,
            "temperatureHighSetDegC":   30,
            "temperatureHighClearDegC": 25,
            "temperatureLowClearDegC":  5,
            "temperatureLowSetDegC":    0
        }
    }],
```

10.3 Configuration spécifique du capteur de pression LPS25

Cette configuration est optionnelle et n'est indispensable que pour utiliser l'alarme proposée par ce capteur. Les paramètres en question se trouvent dans la section `alarm` de la configuration du capteur. Ce capteur ne supporte pas le paramètre `sendOnAlarmCleared` et son utilisation est donc sans effet. Pour configurer cette alarme il est nécessaire de positionner tous les paramètres suivants :

`highThresholdHPa`: la pression, en hectopascals, au dessus de laquelle se déclenche l'alarme.

`lowThresholdHPa`: la pression, en hectopascals, en-dessous de laquelle se déclenche l'alarme.

Voici un exemple de configuration :

```
    "sensors": [{
        "name": "Pressure",
        "type": "LPS25",
        "periodSec": 900,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": false,
            "lowThresholdHPa":  1000,
            "highThresholdHPa": 1024
        }
    }],
```

10.4 Configuration spécifique du capteur de luminosité OPT3001

La seule configuration spécifique concerne l'alarme ; au travers de la section `alarm` de la configuration du capteur. Ce capteur ne supporte pas le paramètre `sendOnAlarmCleared` et son utilisation est donc sans effet. Pour que l'alarme soit fonctionnelle il faut définir tous les paramètres suivants :

`highLimitLux`: la luminosité, en lux, au dessus de laquelle l'alarme se déclenche.

`lowLimitLux`: la luminosité, en lux, en-dessous de laquelle l'alarme se déclenche.

Voici un exemple de fichier de configuration :

```
    "sensors": [{
        "name": "Lumi",
        "type": "OPT3001",
        "periodSec": 600,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": false,
            "highLimitLux": 5000,
            "lowLimitLux":  3
        }
    }],
```

10.5 Configuration spécifique du capteur de mouvement (l'accéléromètre) LIS3DH

La seule configuration spécifique concerne l'alarme ; au travers de la section `alarm` de la configuration du capteur. Ce capteur ne supporte pas le paramètre `sendOnAlarmCleared` et son utilisation est donc sans effet. Pour que l'alarme soit fonctionnelle il faut définir le paramètre suivant :

`motionDetection`: valeur `true` pour activer l'alarme, `false` sinon. Il n'est pas possible de régler le seuil de sensibilité. Ce serait certainement une amélioration à apporter. La seule utilisation possible à l'heure actuelle est le déclenchement d'une lecture et d'un envoi (si l'envoi est activé) lorsqu'un mouvement du nœud est détecté.

Voici un exemple de fichier de configuration :

```
    "sensors": [{
        "name": "Accelero",
        "type": "LIS3DH",
        "periodSec": 3600,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": false,
            "motionDetection":    true
        }
    }],
```

10.6 Configuration de pluviomètres de type à augets

Pour ces capteurs il faut utiliser le type `RainGaugeContact`. Les pluviomètres à augets génèrent un « tick » à chaque basculement qui correspond à une quantité de pluie donnée. Le nœud se chargera donc de compter le nombre de ticks entre deux demandes de lectures ainsi que le temps écoulé entre deux ticks consécutifs pour déterminer une vitesse de précipitations qu'il pourra comparer aux niveaux d'alarme configurés.

Pour fonctionner, ce type de capteur nécessite des paramètres complémentaires. Voici un exemple de configuration d'un capteur :

```
    }, {
        "name": "Pluvio1",
        "type": "RainGaugeContact",
        "periodSec": 3600,
        "tickInterrupt":  "INT2_INT",
        "tickDebounceMs": 500,
        "rainMMPerTick":  0.2,
        "interruptChannel": "OPTO1_INT",
        "sendOnInterrupt":  true,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": true,
            "thresholdSetMMPerMinute":  10.0,
            "thresholdClearMMPerMinute": 7.0
        }
    }],
```

Dans l'exemple donné ci-dessus le capteur enverra des données lorsque l'alarme se déclenchera ou s'arrêtera ou encore lorsque l'interruption `OPTO1_INT` se produira. Si vous souhaitez qu'il n'envoie qu'en cas d'alarme, alors il vous faut conserver le paramètre `sendOnAlarmSet` à `true` (et éventuellement `sendOnAlarmCleared` selon vos besoins) et supprimer le paramètre `interruptChannel` (ou lui donner une valeur vide `""`).

10.6.1 Les paramètres obligatoires

`tickInterrupt`: (**obligatoire**) indique le nom de l'interruption utilisée pour compter les ticks. Il faut rappeler que le nœud n'est sensible qu'aux fronts montants des lignes d'interruption physiques. Cependant, dans le cas des pluviomètres à augets ce facteur ne devrait pas être problématique, car à chaque basculement l'interrupteur dont il est équipé devrait générer un contact puis une rupture de contact (ou inversement), et donc un front montant est assuré à chaque basculement.

`tickDebounceMs`: (**obligatoire**) le temps d'anti-rebonds, en millisecondes, à utiliser. Si votre pluviomètre utilise un contact « propre », comme un interrupteur électronique ou un interrupteur mécanique filtré par un système d'anti-rebonds analogique par exemple, alors vous pouvez positionner ce paramètre à 0 pour désactiver l'anti-rebonds logiciel. En revanche, s'il emploie un interrupteur mécanique basique, ce paramètre sera certainement nécessaire. Un interrupteur mécanique n'établit, ni ne rompt, un contact en une fois ; il se produit des « rebonds » mécaniques qui se traduisent par des ouvertures et des fermetures multiples et très rapides du contact. Le nœud peut parfois détecter ces contacts intempestifs et compter plusieurs interruptions lorsqu'il faudrait qu'il n'en voie qu'une. Lors de mes essais sans anti-rebonds chaque basculement se traduisait par la détection de deux interruptions : l'une lorsque l'interrupteur entrait en contact, et l'autre lorsque le contact se rompait. Car, comme indiqué plus haut, le contact ne se rompt pas simplement, des rebonds se produisent, et donc un front montant était généré. Les autres rebonds sont filtrés par le temps de réveil du nœud et par le temps nécessaire au traitement des interruptions. Le temps d'anti-rebonds doit être assez long pour éviter de détecter les rebonds, mais pas trop pour ne pas passer à côté de ticks. Une valeur de quelques centaines de millisecondes est en général appropriée.

`rainMMPerTick`: (**obligatoire**) la quantité de pluie, en millimètres, correspondante à un « tick », à une interruption.

10.6.2 Les paramètres optionnels

Ils servent à configurer l'alarme et se trouvent donc dans la section `alarm` du capteur.

`thresholdSetMMPerMinute` ou `thresholdSetMMPerHour`: (optionnel) définit la quantité de pluie par unité de temps, la « vitesse des précipitations », qui déclenche l'alarme. La valeur associée au premier paramètre est exprimée en millimètres par minute tandis que celle du second est en millimètres par heure. A vous de choisir celui qui vous convient le mieux. Si les deux sont spécifiés alors seul `thresholdSetMMPerHour` sera pris en compte. Si ce paramètre est indiqué, alors il faut obligatoirement également spécifier le paramètre `thresholdClearMMPerMinute` ou `thresholdClearMMPerHour`. Ce paramètre complémentaire peut utiliser un unité différente de celle de déclenchement de l'alarme.

`thresholdClearMMPerMinute` ou `thresholdClearMMPerHour`: (optionnel), est le paramètre complémentaire de celui étudié précédemment. Il définit le seuil qui permet d'arrêter une alarme en cours. Il permet d'introduire un hystérésis.