Le fichier de configuration du nœud ConnecSenS

Le comprendre et l'utiliser

Versions logicielles `2.x.x`
Révision 12

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 À 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",
    "experimentName": "Campus_Cézeaux",
    "sendConfigPeriodDay": 1,

    "logs": {
        "defaultLevel": "INFO",
        "serial": "EXT_USART"
    },

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

    "network": {
        "type":   "LoRaWAN",
        "devEUI": "0203040506070609",
        "appEUI": "A78F1729918331B4",
        "appKey": "D7F66C4B228A7DF609000006A6579FC1",
        "periodHr": 1
    },

    "sensors": [{
        "name": "Press",
        "type": "LPS25",
        "periodMn": 10,
        "interruptChannel":"LPS25_INT",
        "sendOnInterrupt": true,
        "alarm": {
            "sendOnAlarmSet": true,
            "lowThresholdHPa":  900,
            "highThresholdHPa": 1100
        }
    }, {
        "name": "Lumi",
        "type": "OPT3001",
        "periodMn": 5,
        "interruptChannels": ["OPT3001_INT", "INT1_INT"],
        "sendOnInterrupt": false
    }, {
        "name": "Pluvio1",
        "type": "RainGaugeContact",
        "periodHr": 1,
        "tickInterrupt":  "INT2_INT",
        "tickDebounceMs": 500,
        "rainMMPerTick":  0.2,
        "interruptChannel": "OPTO1_INT",
        "sendOnInterrupt":  true,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": true,
            "periodSec":          60,
            "thresholdSetMMPerMinute":  10.0,
            "thresholdClearMMPerMinute": 7.0
        }
    }],

    "time": {
        "syncMethod": "GPS",
        "GPS": {
            "periodDay": 7,
            "timeoutMn": 5
        },
        "manualUTC": {
            "year": 2018,
            "month": 6,
            "day": 3,
            "hours": 9,
            "minutes": 50,
            "seconds": 0
        }
    }
}
```

4 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 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. Le nœud devrait détecter la connexion, sortir du mode défaut pour retourner dans le bootloader. Si ce n'est pas le cas, appuyez alors sur le bouton reset.

5 La configuration générale

`name`: (**obligatoire**) défini le nom du nœud. Sa longueur maximale est de 31 caractères, tout caractère excédentaire est ignoré.

`experimentName`: (**obligatoire**, disponible à partir de la version `1.2.0` du firmware) défini le nom de l'expérience pour laquelle les données sont produites. Sa longueur maximale est de 31 caractères, tout caractère excédentaire est ignoré.

`uniqueId`: (**optionnel, mais fortement recommandé**) un identifiant unique du nœud, d'une longueur de 31 caractères au maximum. Si vos ne spécifiez pas ce paramètre ou que vous indiquez une valeur vide, alors le nœud tentera de générer automatiquement un identifiant. L'identifiant auto-généré est composé du préfixe `CNSS-NDSTEPAT-` et d'un suffixe obtenu selon deux manières différentes. Si votre nœud a un fichier `proc/board_main` avec le paramètre `SN` défini, et que sa valeur n'est pas trop longue, alors cette valeur sera le suffixe de l'identifiant. Sinon, le suffixe est construit à partir de l'identifiant unique du microcontrôleur qui anime le nœud (programmé en usine).

`batteryLowV` : (optionnel, valeur par défaut : 3,2 V). Indique le seuil, le voltage, en dessous duquel la batterie est considérée comme ayant un niveau bas. La tension de batterie envoyée par ondes radios contient des indicateurs d'état, et si la tension batterie tombe sous le seuil défini par ce paramètre, alors un indicateur de tension batterie basse sera envoyé.

`sendConfigPeriodDay`: (optionnel, valeur par défaut : 0) active ou non l'envoi périodique de la configuration du nœud par ondes radios et spécifie la fréquence d'envoi. Une fréquence de 0 signifie que la configuration n'est pas envoyée périodiquement. L'envoi évènementiel, sur détection d'une modification du fichier de configuration, est actif qu'une période soit spécifiée ou non. Les paramètres `sendConfigPeriodSec`, `sendConfigPeriodMn`, et `sendConfigPeriodHr` fonctionnent également, mais ils ne devraient être d'aucune utilité pratique.

`outputCSVData`: (optionnel, disponible à partir de la version logicielle 1.1.0, valeur par défaut : `true`) active (`true`) ou désactive (`false`) l'écriture des données des capteurs dans un fichier CSV dans le répertoire `output/csv` du nœud. Le nom du fichier contient l'identifiant réseau du nœud

(le devEUI dans le cas d'un réseau LoRaWAN) ainsi qu'une valeur hachée de l'entête CSV en hexadécimal. Ainsi, un nouveau fichier est créé si un ou des capteurs sont ajoutés ou supprimés dans la configuration, ou si les mesures qu'ils produisent changent (l'entête CSV change alors, ce qui produit une valeur hachée très certainement différente).

`addGeoPosToAllReadings` : (optionnel, disponible à partir de la version `1.3.0`, valeur par défaut : `true`). Ajoute (`true`) ou non (`false`) la dernière position géographique obtenue par le nœud dans chaque trame de données. La dernière position géographique est stockée dans la mémoire de l'horloge temps réel, elle est donc conservée tant que le nœud est alimenté. Pour la remettre à zéro (pour la supprimer) il faut donc débrancher toutes les sources d'alimentation de nœud. Cette action aura également pour conséquence de remettre à zéro (au 1er janvier 2000 à minuit) l'heure de l'horloge temps réel. La position n'est pas ajoutée aux trames qui contiennent des données de configuration.

`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 nœud 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.

6 La configuration des logs

Le nœud produit des lignes de log qui permettent de suivre la vie de nœud et de détecter d'éventuels problèmes. Les fichiers de log se trouvent dans le dossier `log/syslog` de la mémoire du nœud (de sa carte SD). Le fichier courant s'appelle `sys.log`. Sa taille maximale est limitée pour éviter qu'il grossisse sans limite en consommant toute la mémoire du nœud et pour que son

téléchargement ne soit pas excessivement long. Lorsqu'il atteint sa taille limite il est renommé en `sys.log.XXXXXX` et un nouveau fichier `sys.log` vide est créé pour recevoir les nouveaux messages à venir. `XXXXXX` est un numéro qui permet de connaître la chronologie entre les différents fichiers sauvegardés. Le nombre de fichiers sauvegardés est limité pour éviter de saturer la mémoire interne. Les choix faits devraient permettre de conserver les logs sur une durée d'au moins un an avec des périodes de mesures de l'ordre de l'heure. La configuration des logs se fait au moyen d'une section **`logs`** optionnelle. Cette section n'est disponible qu'à partir de la version `1.4.0` du logiciel (du firmware) du nœud.

Les messages de log ont un « niveau » de priorité qui permet de les filtrer selon le niveau de verbosité que vous souhaitez avoir. Plus le niveau de filtrage est bas plus la quantité de messages de log produite est importante, plus il est facile de trouver la source de problèmes, mais également plus la consommation du nœud est importante, car il doit écrire ces messages sur la carte SD. Les différents niveaux sont, du plus haut au plus bas (du plus critique au plus trivial) :

- `"CRITICAL"`: ce niveau correspond à une erreur *a priori* irrécupérable. Dans ce cas de figure le nœud redémarrera probablement pour repartir d'une situation saine.
- `"ERROR"`: Niveau des messages d'erreur.
- `"WARN"` (ou `"WARNING"`): Niveau pour signaler une situation qui pourrait être une erreur, mais peut-être pas. Ou pour attirer l'attention sur une situation délicate. Ou pour signaler un fait important qui pourrait avoir un impact sur l'analyse ou sur la production des données.
- `"INFO"`: le niveau des messages ordinaires. Utilisé par le nœud pour indiquer ce qu'il fait, avec suffisamment de détails pour suivre le « fil de ses pensées » et de ses choix les plus importants, mais sans s'attarder sur des détails trop précis.
- `"DEBUG"`: va beaucoup plus dans le détail. Le nombre de message produits peut être important et leur utilité pas forcément justifiée, surtout si l'on ne s'attend pas à des problèmes particuliers. Ce niveau de détail est plutôt utilisé pour mieux cerner un problème identifié.
- `"TRACE"`: écrit absolument tous les messages que peut produire le nœud. Ce niveau est potentiellement très verbeux, mais il n'est pas utilisé à l'heure où j'écris ces lignes.

`defaultLevel` : (optionnel, disponible à partir de la version `1.4.0`, valeur par défaut : `"INFO"`). Permet de spécifier le niveau des messages à conserver. Tous les messages de log dont le niveau sera supérieur ou égal à celui indiqué par ce paramètre seront conservés, les autres seront ignorés. La valeur est une chaîne de caractère parmi la liste des noms de niveau donnée plus haut, elle n'est pas sensible à la casse. Ainsi, la valeur par défaut `"INFO"` conservera les messages de niveau `"CRITICAL"`, `"ERROR"`, `"WARN"` et `"INFO"`, mais ignorera ceux de niveau `"DEBUG"` et `"TRACE"`.

`serial` : (optionnel, disponible à partir de la version `1.4.0`, valeur par défaut : `""`). En plus d'écrire les messages de log dans un fichier il est possible de sortir ces mêmes messages, en temps réel, sur une sortie série. Cette fonctionnalité ne devrait pas intéresser les utilisateurs courant du nœud, mais plutôt les personnes en charge du développement logiciel du nœud ou de drivers de

capteurs. Cette sortie n'est pas activée par défaut. La valeur n'est pas sensible à la casse et indique le nom du port série du nœud à utiliser. Les valeurs disponibles sont `"EXT_USART"` et `"SigFox_USART"`, qui correspondent aux lignes USART_TX du port d'extension interne du nœud et à l'interface pour le module de communication SigFox (sérigraphié P7 sur la carte mère) respectivement. Il n'est pas conseillé d'utiliser la sortie SigFox, car elle est partagée avec le GPS et il existe donc des risques de conflit. Il est toutefois possible de l'utiliser, par exemple si l'interface série du connecteur d'extension est utilisée par un capteur, auquel cas il est alors fortement conseiller de désactiver l'utilisation du GPS. Pour utiliser l'une de ces lignes il faut la relier à un adaptateur série 3,3 V vers USB. 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 de la ligne série que vous avez sélectionné (très probablement USART_TX du connecteur d'extension interne). Configurez ensuite le logiciel de communication série que vous utilisez sur votre ordinateur avec les paramètres suivants : 115 200 bauds, 8 bits, pas de parité, 1 bit de stop, pas de contrôle de flux matériel.

7 La configuration du debug

Il est possible de configurer les fonctionnalités de débogage du nœud au moyen d'une section `debug` dans le fichier de configuration. Cette section, et tous les paramètres qu'elle comprend, sont optionnels. Les paramètres de cette section sont :

`useInternalLEDs` : (optionnel, valeur par défaut : `false`) valeur `true` ou `false`. Configure l'activation, ou non, des LEDs de diagnostic. Le nœud est équipé de deux LEDs internes pour indiquer son fonctionnement, son état. Puisque ces LEDs ne sont pas visibles une 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, qui elles sont toujours utilisées.

`useBuzzer` : (optionnel, valeur par défaut : `false`) valeur `true` ou `false`. Configure l'utilisation, ou non, d'un buzzer. Pour que ce paramètre soit pris en compte il faut que vous définissiez une section `buzzer` dans votre fichier de configuration, comme c'est expliqué dans le chapitre suivant. Le buzzer est utilisé pour signaler le succès ou l'échec de la transmission des données par ondes radios. Un bip bref indique l'échec, tandis qu'une succession de bips rapides pendant trois secondes signale la réussite de la communication RF. Cette signalétique se veut proche de celle offerte par la LED D1.

8 Configuration d'un buzzer

Il est possible de relier un buzzer auto-oscillant au nœud. Un buzzer auto-oscillant à juste besoin d'être alimenté par une tension continue pour produire un son, contrairement aux buzzers simples qui doivent être excités par une tension alternative. Le buzzer sélectionné doit pouvoir fonctionner avec une tension de 3 V, car il sera alimenté par l'une des broches du connecteur d'extension

interne. Les broches de ce connecteur utilisables pour activer le buzzer sont : `EXT_GPIO1`, `EXT_GPIO2`, `EXT_GPIO3`, `EXT_GPIO4`, `EXT_GPIO5`, `SPI_CS`, `SPI_MISO`, `SPI_MOSI`, `SPI_CLK`, `I2C_SDA`, `I2C_SCL`, `USART_TX` et `USART_RX`. Les buzzers auto-oscillants sont généralement polarisés, c'est-à-dire qu'ils ont une broche positive et une autre négative. Branchez celle négative à l'une des broches de masse du connecteur d'extension interne et celle positive à l'une des broches listées plus haut.

Voici un exemple de configuration :

```
    "debug": {
        "useInternalLEDs": false,
        "useBuzzer": true,
        "verbose": true,
        "serial": ""
    },

    "buzzer": {
        "activationExtPin": "SPI_CS"
    },
```

Les paramètres de la section `buzzer` sont :

`activationExtPin` : (optionnel, valeur par défaut `""`) le nom de la broche du connecteur d'extension interne utilisée pour activer le buzzer (celle reliée à la broche positive du buzzer). La liste des valeurs acceptables a été donnée dans le paragraphe précédent. L'absence de ce paramètre, une valeur vide ou `null`, désactive l'utilisation du buzzer.

9 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 au 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.
- `USART_INT`, l'interruption associée aux lignes UART (séries) externes. Sensible uniquement à un front montant.
- `SDI12_INT`, l'interruption associée à l'interface SDI-12 externe. À 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.
- `OPTO1_INT` et `OPTO2_INT`, associées aux entrées opto-isolées 1 et 2 respectivement. À 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. À partir 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 un 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.

10 La configuration réseau

Elle se décompose en paramètres généraux, applicables à tous les types d'interface réseau et de paramètres spécifique à chaque type d'interface. À l'heure actuelle un seul type d'interface réseau est supporté : LoRaWAN.

10.1 Paramètres généraux

10.1.1 Paramètres obligatoires

`periodSec`, `periodMn`, `periodHr` ou `periodDay`: (**obligatoire**) est la période d'émission des données, en secondes, en minutes, en heures ou en jours respectivement. Un seul de ces paramètres peut être utilisé à la fois. La valeur est un nombre strictement positif entier dans le cas d'une période exprimée en secondes ou réel pour les autres paramètres. Il est 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 à deux minutes pour la mesure des capteurs et pour la transmission des données.

`type`: (**obligatoire**) indique le type de réseau sans fil utilisé. À 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 fonctionnelle.

10.1.2 Paramètres optionnels

`periodWhenAlarmSec`, `periodWhenAlarmMn`, `periodWhenAlarmHr` ou `periodWhenAlarmDay`: (optionnel, disponible à partir de la version `1.4.0`, valeur par défaut : 0) est la période d'émission alternative à utiliser lorsque l'un des capteurs est en alarme, exprimée en secondes, minutes, heures ou jours respectivement. Une valeur inférieure à la période normale sera *a priori* utilisée, mais rien n'empêche d'indiquer une période supérieure. Une valeur de 0 indique qu'il n'y a pas de période d'envoi spécifique au cas où un capteur est en alarme.

`periodSpreadSec`: (optionnel, valeur par défaut : période d'émission / 4) indique l'étalement de la période d'émission à utiliser. Comme les périodes sont absolues par rapport à minuit, alors sans étalement de la période d'émission, tous les nœuds avec des périodes d'émissions identiques ou multiples les unes des autres essayeraient de transmettre au même moment, et risqueraient donc de

se brouiller mutuellement. L'étalement de période permet de répartir la « charge » dans le temps. L'étalement se fait toujours en positif par rapport à la période absolue, c'est-à-dire que l'heure d'émission ne peut qu'être supérieure à celle absolue (sans étalement) et pas inférieure. Ainsi, avec une période d'émission d'une heure et un étalement de 15 minutes, alors l'heure d'émission sera toujours comprise entre H et H + 15mn. Par défaut, l'étalement est activé et sa valeur est égale au quart de la période d'émission. La valeur de l'étalement ne peut dépasser la moitié de la période d'émission. Si le paramètre est spécifié avec une valeur de 0, alors l'étalement de la période d'émission est désactivé. Alternativement, il est possible d'utiliser les paramètres `periodSpreadMn`, `periodSpreadHr` et `periodSpreadDay` pour spécifier l'étalement en minutes, heures ou jours respectivement.

`joinOnSendFailsCount`: (optionnel, disponible à partir de la version `2.0.0`, la valeur est un entier positif, la valeur par défaut est 24). Ce paramètre indique le nombre d'échecs de transmission de données qui déclenche une nouvelle jonction du réseau, pour peu que cette notion ait un sens pour le réseau en question. Une valeur de 0 désactive cette fonctionnalité et aucune tentative de re-jonction du réseau ne sera faite, quel que soit le nombre d'échecs de transmission des données. La valeur par défaut a été choisie pour qu'une tentative de re-jonction du réseau soit faite par jour si toutes les transmissions du nœud échouent et qu'il est configuré avec une période de transmission d'une heure.

`joinTimeoutSec` : (optionnel, disponible à partir de la version `2.0.0`, la valeur est un entier positif, la valeur par défaut est 120). Indique le temps d'attente maximal, en secondes, pour parvenir à joindre le réseau. Il existe également les variantes `joinTimeoutMn` et `joinTimeoutDay`, pour un temps en minutes et en jours respectivement. La première variante pourrait être utile, la seconde semble être peu pratique. Une valeur de 0 désactive cette durée maximale et laisse le processus se poursuivre jusqu'à son terme, quel que soit le temps nécessaire. Pour une plus grande fiabilité il est toutefois conseillé d'indiquer une valeur. La consommation du nœud ne devrait pas augmenter exagérément avec l'augmentation de cette valeur puisqu'il passe le plus clair de son temps dans un mode d'économie d'énergie en l'attente d'une réponse du serveur LoRaWAN. Toutefois, plus le temps laissé est long, plus il aura l'occasion de retenter de joindre le réseau, et chaque tentative représente une consommation d'énergie. Vous êtes donc relativement libre de choisir la valeur qui vous plaît. L'attente de la jonction du réseau n'est pas bloquante, les mesures capteur continuent d'être faites pendant ce temps-là.

`sendTimeoutSec` : (optionnel, disponible à partir de la version `2.0.0`, la valeur est un entier positif, la valeur par défaut est 60). Ce paramètre est comparable à `joinTimeoutSec`, mais pour l'envoie de données. Il fixe un temps maximal, en secondes, pour l'envoie des données. Il existe également les alternatives `sendTimeoutMn` et `sendTimeoutDay`, pour des temps indiqués en minutes et en jours respectivement. Ici aussi, une valeur de 0 laisse le processus se dérouler sans limite de temps, et elle n'est pas une valeur conseillée pour des raisons de fiabilité. Toutes les remarques faites pour le paramètre `joinTimeoutSec` s'appliquent ici aussi.

10.2 Interface réseau LoRaWAN

Elle correspond au `type LoRaWAN`.

10.2.1 Paramètres obligatoires

`devEUI`: (**obligatoire**) l'identifiant unique du nœud ConnecSenS. 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**) l'identifiant d'application. 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**) la clef d'authentification auprès de l'application. 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.

10.2.2 Paramètres optionnels

`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 LoRaWAN 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).

`publicNetwork`: (optionnel, valeur par défaut : `false`) indique si le réseau LoRaWAN auquel se connecter est public (`true`) ou privé (`false`). Le réseau sera normalement privé. En cas de doute, contactez la personne en charge de la gestion du réseau LoRaWAN.

`joinNbTrials`: (optionnel, disponible à partir de la version `2.0.0`, fourchette de valeurs : [1..8], valeur par défaut : 8). Spécifie le nombre d'essais à effectuer au plus lorsque le nœud essaie de joindre le réseau LoRaWAN.

`sendAckNbTrials`: (optionnel, disponible à partir de la version `2.0.0`, fourchette de valeurs : [1..8], valeur par défaut : 2). Spécifie le nombre d'essais maximal pour obtenir un accusé de réception des données envoyées. Plus la valeur est élevée, plus les chances d'obtenir un accusé de réception augmente ; nous n'avons toutefois pas de données pour savoir si cette augmentation de chance est significative, peut-être à vous de voir en pratique. La consommation du nœud augmente avec le nombre de tentatives, puisque le nœud doit procéder à une nouvelle émission RF pour chaque tentative, il vaut donc mieux essayer d'utiliser une valeur faible.

`sendTimeoutSec` : (optionnel, disponible à partir de la version `1.3.0`, supprimé à partir de la version `2.0.0`, la valeur par défaut est 35 secondes). Le nœud attend un accusé de réception après l'envoi de données pour s'assurer que ces données ont bien été reçues, faute de quoi il tentera de les renvoyer plus tard si possible. Ce paramètre fixe la durée maximale du temps d'attente pour la réception de l'accusé de réception, en secondes. La valeur maximale est de 60 secondes. Une valeur de 0 ou supérieure à 60 résulte en l'utilisation de la valeur par défaut.

11 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.

11.1 Synchronisation du temps par GPS

Pour activer la synchronisation GPS il faut positionner le paramètre `syncMethod` à `GPS` et créer un objet JSON avec le label `GPS`. Les paramètres dont nous allons parler dans ce chapitre sont définis dans cet objet `GPS`.

`periodSec`, `periodMn`, `periodHr` ou `periodDay` : (**obligatoire**) spécifient la période en secondes, en minutes, en heures ou en jours respectivement de resynchronisation de l'horloge du nœud au moyen du GPS. Les valeurs indiquées doivent être strictement positives et les trois derniers paramètres peuvent accepter un nombre réel comme valeur. Il est ainsi possible, par exemple, de donner la valeur 1,5 au paramètre `periodMn`, soit 90 secondes. Si la période est donnée en secondes, alors un nombre entier strictement positif doit être utilisé.

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, voire une fois par mois.

`timeoutSec`: (optionnel, valeur par défaut 120 secondes pour les versions de firmware antérieures à la `1.2.0`, 300 secondes pour les versions postérieures ou égales à la `1.2.0`) règle le temps d'attente maximal pour obtenir une accroche GPS, en secondes, pour les accroches périodiques à partir de la version logicielle `1.2.0` (voir le paramètre `blockingTimeoutSec` pour les accroches bloquantes), pour tous les types d'accroches pour les versions antérieures. La valeur minimale acceptée est de 60 secondes. Alternativement, il est possible d'utiliser le paramètre `timeoutMn` pour spécifier ce temps en minutes, la valeur est alors un nombre réel strictement positif. Dans de bonnes conditions, le temps d'accroche est d'environ 40 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 devrait être suffisant pour la majorité des situations. Néanmoins, dans les cas les plus difficiles, il pourrait être nécessaire d'augmenter cette valeur. La synchronisation par GPS étant très énergivore, il est conseillé de limiter ce temps d'attente.

`blockingTimeoutSec`: (optionnel, disponible à partir de la version logicielle `1.2.0`, valeur par défaut de 600 secondes, soit 10 minutes). Spécifie le temps d'attente maximal pour une accroche GPS bloquante. Les accroches GPS bloquantes empêchent le nœud de faire autre chose tant que l'accroche GPS est en cours. Elles visent à obtenir un bon point de départ, une bonne heure, avant de faire autre chose, avant de produire des données horodatées. Les accroches bloquantes se produisent suite à une mise sous tension du nœud ou suite à un changement de la méthode de mise à l'heure du nœud dans son fichier de configuration vers « GPS ». Sa valeur minimale est de 60 secondes. Il est possible de spécifier une valeur inférieure à celle de `timeoutSec`, au cas où une telle configuration aurait un sens pratique. Alternativement, il est possible d'utiliser la paramètre `blockingTimeoutMn` pour spécifier ce temps en minutes plutôt qu'en secondes, avec une valeur réelle.

11.2 Synchronisation du temps manuelle

Pour que la synchronisation manuelle soit prise en compte il faut positionner le paramètre `syncMethod` à `manual`. La valeur `manualUTC` est équivalente, et est également acceptée pour être plus en accord avec le nom de la section de configuration manuelle du temps.

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`. 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 24 h (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.

Pour obtenir une bonne précision avec cette méthode de mise à l'heure il est conseillé de programmer une heure dans le futur proche, par exemple la minute à suivre, de sauvegarder le fichier de configuration et d'éjecter le disque qui correspond au nœud dans le système d'exploitation. Puis, lorsqu'il est l'heure spécifiée dans le fichier de configuration, débranchez une extrémité du câble USB. L'heure est alors programmée dans le nœud avec une précision de l'ordre de la seconde pour peu que vous ayez été assez rapide.

L'heure indiquée dans le fichier de configuration est prise en compte uniquement si elle a été modifiée. De ce fait, vous pouvez vous retrouver avec une horloge en l'an 2000. C'est un choix volontaire. Si le nœud s'est retrouvé privé d'alimentation, parce que vous avez débranché la batterie par exemple, alors son horloge est remise à 0, et donc au 1er janvier 2000 à minuit. Si vous avez configuré une mise à l'heure manuelle, mais que vous oubliez de changer l'heure indiquée dans le fichier de configuration, alors l'heure de la configuration est ignorée et l'horloge n'est pas modifiée. Vous obtiendrez alors des mesures horodatées en 2000, ce qui vous permettra de facilement les écarter par la suite et vous évitera de prendre en compte des données qui auraient été mal horodatées. En cas de mise à l'heure manuelle pensez donc toujours à mettre à jour la date spécifiée dans le fichier de configuration après avoir changé la batterie, où après une coupure d'alimentation du nœud.

12 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 le 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 31 caractères, tout caractère excédentaire est ignoré. Il doit être unique, différent du nom de tous les autres capteurs déclarés. Ce nom est choisi par l'utilisateur.

`uniqueId`: (**optionnel, mais fortement recommandé**) un identifiant unique du capteur, d'une longueur de 31 caractères au maximum. Au pire, ce doit être une valeur qui identifie de manière unique le capteur parmi tous les capteurs du même type, mais l'idéal est une valeur qui identifie le capteur de manière unique, quel que soit son type. Pour certains capteurs, le nœud est en mesure de remplir ce champ automatiquement si vous ne spécifiez pas de valeur (référez-vous au chapitre traitant de votre capteur pour savoir si ce dernier offre cette fonctionnalité). Si vous indiquez une valeur, alors cette dernière sera attribuée au capteur, même si celui-ci est capable de la déterminer automatiquement ; votre valeur de configuration est prioritaire.

`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.

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

`periodSec`, `periodMn`, `periodHr` ou `periodDay`: (**obligatoire**) la période de mesure du capteur, en secondes, en minutes, en heures ou en jours respectivement. La valeur est un nombre strictement positif, entier dans les cas d'une période exprimée en secondes, réel dans les autres cas. 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.

À partir de la version `2.0.0` du firmware, il est également possible d'utiliser le paramètre `period` avec la valeur `"sensorFlow"` à la place d'une valeur numérique. Avec cette combinaison, le rythme de mesure n'est plus donné par le nœud mais par le capteur. En effet, une partie des capteurs possèdent une horloge interne propre et produisent de mesures périodiques en fonction de cette

horloge interne. Avec ce paramétrage il n'y a plus à se soucier de trouver une période de mesure adéquate pour ne pas perdre de valeur, ou pour ne pas avoir de valeur en double. Le nœud écoute en permanence le capteur, en attente d'un message de sa part, et enregistre une mesure au moment où le capteur la lui envoie. Ce mode de fonctionnement ne fait pas augmenter significativement la consommation d'énergie du nœud, aussi c'est souvent le mode de fonctionnement à préférer s'il est disponible. Une minorité de capteurs peut fonctionner selon cette relation. La possibilité d'utiliser ce mode de fonctionnement est indiqué dans la documentation de la configuration des capteurs en question.

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 connecteurs 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 est alors tout à fait possible d'associer l'interruption `OPTO1_INT` au capteur interne `LPS25`.

`firmwareVersion`: (**optionnel**) permet de spécifier la version de firmware qui anime un capteur, d'une longueur de 31 caractères au maximum. Tous les capteurs n'ont pas de firmware, ce paramètre n'a alors pas de sens pour eux. Certains capteurs sont en mesure de remplir automatiquement ce paramètre (référez-vous au chapitre traitant de votre capteur pour savoir si celui-ci offre cette fonctionnalité). Si le capteur est en mesure de déterminer tout seul sa version de firmware, alors cette version auto-détectée est prioritaire sur une éventuelle valeur spécifiée dans le fichier de configuration.

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 a 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.

`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.

`useInt3V3WhenActive` : (optionnel, disponible à partir de la version `1.4.0`, valeur par défaut dépendante du capteur mais généralement à `false`). Force l'activation de l'alimentation 3,3 V des interruptions externes lorsque sa valeur est à `true`. Si ce paramètre est absent ou à `false` alors l'utilisation ou non de cette alimentation dépend du driver du capteur en question : si ce capteur en a besoin pour fonctionner lorsque le nœud est actif, alors l'alimentation est activée par le driver, sinon elle est éteinte. Cette alimentation est disponible sur le point de test TP6 qui est équipé d'une broche mâle de connexion depuis la dernière révision matérielle des nœuds. Il est donc possible de s'en servir pour alimenter un capteur, même si ce n'est pas conseillé. En effet, ce 3,3 V est les même que celui qui alimente le cœur du nœud. Un court-circuit de cette alimentation par exemple provoquerait l'arrêt du nœud ; il y survivrait toutefois probablement.

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.

12.1.1 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.

`periodSec`, `periodMn`, `periodHr` ou `periodDay` : (optionnel, valeur par défaut : 0) indique la périodicité de mesure à utiliser lorsque le capteur est en alarme. Au déclenchement de l'alarme, la périodicité est changée à celle indiquée par ce paramètre, et la périodicité normale est rétablie lorsque l'alarme s'éteint. Cette périodicité d'alarme peut être inférieure, ou supérieure, ou même égale (ce qui est équivalent à une valeur 0), à la périodicité normale. Une valeur de 0 désactive cette fonctionnalité et la périodicité normale est alors utilisée en toutes circonstances.

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

Ce capteur est en mesure de remplir automatiquement le paramètre `uniqueId`. La valeur automatiquement générée suit le format suivant : `CNSS-XXXXXXX-SHT35-YYYYYYYY`, où `XXXXXXX` est le numéro de série de la carte capteur interne du nœud (visible sur l'étiquette collée à cette carte interne). Si le nœud n'est pas en mesure de générer automatiquement cette valeur, alors cela veut probablement dire que le fichier `proc/board_sensors` n'est pas présent dans la mémoire interne (carte SD) du nœud ; auquel cas rapprochez-vous d'une personne compétente pour créer ce fichier. `YYYYYYYY` est une représentation hexadécimale petit-boutiste (les deux premiers caractères représentent donc l'octet de poids faible) du numéro de série du capteur ; un numéro obtenu par interrogation du capteur.

Ce capteur offre une alarme. 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
        }
    }],
```

12.3 Configuration spécifique du capteur de pression LPS25

Ce capteur est en mesure de remplir automatiquement le paramètre `uniqueId`. La valeur automatiquement générée suit le format suivant : `CNSS-XXXXXXX-LPS25`, où `XXXXXXX` est le numéro de série de la carte capteur interne du nœud (visible sur l'étiquette collée à cette carte interne). Si le nœud n'est pas en mesure de générer automatiquement cette valeur, alors cela veut probablement dire que le fichier `proc/board_sensors` n'est pas présent dans la mémoire interne (carte SD) du nœud ; auquel cas rapprochez-vous d'une personne compétente pour créer ce fichier.

Il est possible de configurer une alarme avec 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
        }
    }],
```

12.4 Configuration spécifique du capteur de luminosité OPT3001

Ce capteur est en mesure de remplir automatiquement le paramètre `uniqueId`. La valeur automatiquement générée suit le format suivant : `CNSS-XXXXXXX-OPT3001`, où `XXXXXXX` est le numéro de série de la carte capteur interne du nœud (visible sur l'étiquette collée à cette carte interne). Si le nœud n'est pas en mesure de générer automatiquement cette valeur, alors cela veut probablement dire que le fichier `proc/board_sensors` n'est pas présent dans la mémoire interne (carte SD) du nœud ; auquel cas rapprochez-vous d'une personne compétente pour créer ce fichier.

La seule configuration spécifique concerne l'alarme ; au travers de la section `alarm` de la configuration du capteur. Il est possible de fixer deux seuils d'alarme : un bas et un haut. Il est possible d'utiliser ces seuils simultanément, pour déclencher une alarme lorsque la luminosité descend sous le seuil bas ou dépasse le seuil haut, ou bien d'utiliser uniquement l'un de ces deux seuils, ou encore aucun d'eux. Chaque seuil est défini par deux paramètres, l'un de déclenchement de l'alarme et l'autre d'arrêt de l'alarme. Si un paramètre de seuil est défini alors son paramètre complémentaire doit également être défini. Il est ainsi possible d'introduire une hystérésis sur les seuils. Si vous ne souhaitez pas utiliser d'hystérésis, alors donnez la même valeur aux deux paramètres complémentaires. Ce capteur supporte le paramètre `sendOnAlarmCleared`.

Le seuil bas est défini par les paramètres suivants :

`lowSetLux` : la luminosité, un nombre réel, en lux, en dessous de laquelle l'alarme se déclenche.

`lowClearLux` : la luminosité, un nombre réel, en lux, au-dessus de laquelle l'alarme s'arrête.

Le seuil haut est défini par les paramètres suivants :

`highSetLux` : la luminosité, un nombre réel, en lux, au-dessus de laquelle l'alarme se déclenche.

`highClearLux` : la luminosité, un nombre réel, en lux, en dessous de laquelle l'alarme s'arrête.

Autres paramètres d'alarme :

`filterNb` : (optionnel, valeur par défaut : `8`) configure le nombre de mesures successives avec une valeur d'alarme nécessaires pour déclencher l'alarme. Les valeurs possibles sont : `1`, `2`, `4` et `8`. Lorsque l'alarme est configurée, le capteur de luminosité réalise des mesures en continu avec une période de 800 ms. Après chaque mesure, le capteur compare la luminosité à ses seuils d'alarme pour détecter un dépassement de seuil. Ces détections de dépassement passent au travers d'un filtre

avant de déclencher l'alarme. Ce filtre compte le nombre de dépassements successifs. L'alarme est déclenchée si ce compte dépasse un seuil donné, configuré par le paramètre `filterNb`. Ce paramètre met donc en œuvre un filtre temporel et permet de supprimer un éventuel bruit. Ainsi, si nous utilisons une valeur de `8`, alors la luminosité doit avoir une valeur d'alarme pendant 8 × 0,8 = 6,4 secondes pour que l'alarme soit effectivement activée. Si les 6 dernières mesures avaient une valeur d'alarme, mais que la septième non, alors le compteur du filtre est remis à zéro, l'alarme ne sera pas déclenchée, et il faudra à nouveau huit valeurs d'alarmes successives pour déclencher l'alarme. Une valeur de `1` déclenche l'alarme dès la première mesure avec une valeur d'alarme, et désactive de fait le filtre.

Voici un exemple de fichier de configuration :

```
    "sensors": [{
        "name": "Lumi",
        "type": "OPT3001",
        "periodSec": 600,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": false,
            "filterNb":     8,
            "highSetLux":   500.0,
            "highClearLux": 450.0,
            "lowClearLux":  15.0,
            "lowSetLux":    10.0
        }
    }],
```

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

Ce capteur est en mesure de remplir automatiquement le paramètre `uniqueId`. La valeur automatiquement générée suit le format suivant : `CNSS-XXXXXXX-LIS3DH`, où `XXXXXXX` est le numéro de série de la carte capteur interne du nœud (visible sur l'étiquette collée à cette carte interne). Si le nœud n'est pas en mesure de générer automatiquement cette valeur, alors cela veut probablement dire que le fichier `proc/board_sensors` n'est pas présent dans la mémoire interne (carte SD) du nœud ; auquel cas rapprochez-vous d'une personne compétente pour créer ce fichier.

Ce capteur propose une alarme. Il 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
        }
    }],
```

12.6 Configuration de pluviomètres de type à augets

Pour ces capteurs il faut utiliser le type `RainGaugeContact`. Les pluviomètres à augets génèrent un « tic » à chaque basculement qui correspond à une quantité de pluie donnée. Le nœud se chargera donc de compter le nombre de tics entre deux demandes de lectures ainsi que le temps écoulé entre deux tics 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 `""`).

12.6.1 Les paramètres obligatoires

`tickInterrupt`: (**obligatoire**) indique le nom de l'interruption utilisée pour compter les tics. 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 tics. Une valeur de quelques centaines de millisecondes est en général appropriée.

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

12.6.2 Les paramètres optionnels

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

`thresholdSetMMPerMinute` ou `thresholdSetMMPerHour`: (optionnel) défini 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. À 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 une 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 une hystérésis.

12.7 Configuration de capteurs d'humidité du sol de type Watermark au travers du boîtier d'adaptation I2C

Ces capteurs sont de type `soilMoistureWatermarkI2C`. Ils sont supportés à partir de la version 0.8.0 du micrologiciel (firmware) du nœud. Voici un exemple de configuration :

```
    "sensors": [{
        "name": "SoilMoisture",
        "type": "soilMoistureWatermarkI2C",
        "periodSec": 3600,
        "depth1Cm": 30,
        "depth2Cm": 60,
        "depth3Cm": 80,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": true,
            "lowSetCb":    15,
            "lowClearCb":  20,
            "highSetCb":   80,
            "highClearCb": 75
        }
    }],
```

12.7.1 Les paramètres obligatoires

`depth1Cm`, `depth2Cm`, `depth3Cm` : (**obligatoire**) la profondeur de la mesure, en centimètres. La valeur est un entier positif compris entre 0 et 511 centimètres. `depth1Cm` correspond à la sonde 1, `depth2Cm` à la sonde 2 et `depth3Cm` à la sonde 3. Il n'est pas obligatoire de définir tous ces paramètres, mais il faut en définir au moins un. Une valeur de 0, ou l'absence d'un paramètre, fait que la sonde associée n'est pas mesurée et sa valeur n'est par conséquent ni sauvegardée ni envoyée.

12.7.2 Les paramètres optionnels

Ils servent à configurer l'alarme et se trouvent donc dans la section `alarm` du capteur. Vous pouvez définir un seuil haut sans seuil bas, un seuil bas sans seuil haut, ou les deux seuils. Il est également possible de ne définir ni seuil haut, ni seuil bas, auquel cas la fonctionnalité d'alarme est désactivée. L'alarme est commune à toutes les sondes et les seuils sont partagés. Il suffit que l'une des sondes franchisse l'un des seuils pour que l'alarme soit déclenchée.

`lowSetCb` : (optionnel) défini le seuil bas de déclenchement de l'alarme, en centibars. S'il est défini, alors le paramètre `lowClearCb` doit également l'être et avec une valeur strictement supérieure à celle de `lowSetCb`. Ces deux paramètres permettent d'introduire une hystérésis. Il peut ici être important de rappeler que plus la valeur en centibars est faible plus le sol est humide.

`highSetCb` : (optionnel) défini le seuil haut de déclenchement de l'alarme, en centibars. S'il est défini, alors le paramètre `highClearCb` doit également l'être et avec une valeur strictement inférieure à celle de `highSetCb`. Ces deux paramètres permettent d'introduire une hystérésis. Il peut ici être important de rappeler que plus la valeur en centibars est grande plus le sol est sec.

12.8 Configuration de capteurs d'eau dits « piezos » de type Aqua TROLL-200, fabriqués par In-Situ

À l'heure actuelle, il est uniquement possible de relier ces capteurs au nœud ConnecSenS au moyen de leur interface SDI-12. Ils sont supportés à partir de la version 0.8.0 du micrologiciel du nœud et leur type est :`InSituAquaTROLL200SDI12`.

De plus, il faut que vos capteurs soient configurés pour sortir au moins la température de l'eau, sa pression et sa conductivité. La pression doit être exprimée en PSI et pas en millimètres de mercure. Les valeurs en surplus de celles listées plus haut sont ignorées par le nœud. L'ordre de sortie des données est sans importance. En cas de doute sur la configuration du capteur, vous pouvez toujours le connecter au nœud, faire la configuration du capteur comme nous allons le voir, et si des données nécessaires ne sont pas disponibles sur l'interface SDI-12, alors le nœud vous en avertira au travers de messages d'erreur dans ses logs.

Pour le moment, ces capteurs ne proposent pas d'alarme.

Voici un exemple de configuration :

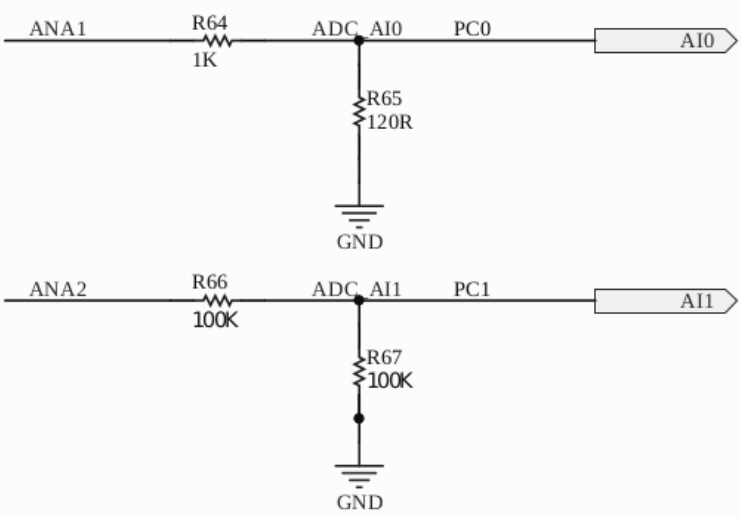
```
    "sensors": [{
        "name": "Piezo",
        "type": "InSituAquaTROLL200SDI12",
        "periodSec": 3600,
        "address": "0"
    }],
```

12.8.1 Les paramètres obligatoires

`address`: (**obligatoire**) l'adresse SDI-12 du capteur. Est une chaîne de caractères qui contient un seul caractère compris entre '0' et '9' ou 'a' et 'z' ou 'A' et 'Z'. Par défaut, en sortie d'usine, les capteurs ont l'adresse '0'.

12.9 Configuration de la mesure de la tension d'une batterie externe connectée à l'une des entrées analogiques du nœud.

Ce capteur est disponible à partir de la version 0.8.0 du firmware.



Le nœud ConnecSenS dispose de deux entrées analogique appelées `ANA1` et `ANA2`. Par défaut, la première est configurée pour utiliser le protocole de communication 4-20 mA et la seconde pour mesurer une tension comprise entre 0 et 6,5 Volts. Il est cependant possible de modifier ces entrées analogiques pour leur faire mesurer d'autres grandeurs analogiques ou pour modifier les plages de mesure. Ces modifications se font par le changement des résistances R64 et R65 pour l'entrée `ANA1` et des résistances R66 et R67 pour l'entrée `ANA2`. Le convertisseur analogique-numérique converti une tension comprise entre 0 et 3,3 V en une valeur numérique sur 12 bits. Les résistances servent donc soit à réduire la tension d'entrée pour qu'elle soit compatible avec la plage 0-3,3 V, soit à transformer un courant en une tension comprise dans cette même plage de tensions. Il faut noter que les entrées analogiques du microcontrôleur qui anime le nœud ne sont pas protégées contre des tensions supérieures à 5 V,

aussi vous devez vous assurer que la tension entre les résistances R64 et R65, ou R66 et R67, selon l'entrée analogique utilisée, ne dépasse jamais 5 V. Il est même fortement conseillé de ne jamais dépasser 3,3 V en ce point. Calculez donc avec soin vos valeurs de résistances. De manière générale, si des modifications matérielles sont nécessaires, adressez-vous à une personne qualifiée.

Dans le cas qui nous intéresse ici, mesurer la tension d'une batterie externe, deux solutions s'offrent à vous. La plus simple est d'utiliser l'entrée `ANA2`, déjà configurée pour mesurer une tension, et d'y ajouter une résistance en série pour adapter la plage de mesure si nécessaire. Par défaut, R66 et R67 ont toutes deux une valeur de 100 kΩ et forment donc un pont diviseur de rapport 2. R66 est connectée à l'entrée analogique du nœud et à l'entrée du convertisseur analogique-numérique (CAN), tandis que R67 est connectée à l'entrée du CAN et à la masse. La résistance ajoutée viendrait donc se mettre en série de R66. Prenons un exemple pratique. Supposons que nous souhaitions mesurer la tension d'une batterie au plomb 12 V. Ce type de batterie, chargée au maximum, présente une tension de presque 14 V. Cette tension est bien supérieure aux 6,5 V acceptables par l'entrée `ANA2` par défaut. Il nous faut un facteur de division de la tension d'au moins : 14 / 3,3, soit ≥ 4,2. Nous viserons un rapport de 5 pour avoir une marge de sécurité. Le facteur de division de la tension batterie est donné par la formule `r = (R66 + R67 + Rs) / R67`, où `Rs` est la résistance ajoutée en série de R66. Nous en déduisons que `Rs = r x 67 − R66 − R67`, soit `Rs = 5 × 100 − 100 − 100 = 300` kΩ. C'est une valeur standard, mais si cela n'avait pas été le cas alors nous aurions choisi la valeur standard supérieure la plus proche. Nous choisissons la valeur supérieure pour augmenter la marge de précaution que nous avions prise, une valeur de résistance inférieure aurait réduit cette marge. Pour améliorer la précision de la mesure, il est conseillé de mesurer la valeur réelle des résistances R66, R67 et Rs avec un même Ohmmètre et de calculer le rapport réel de division de tension avec la formule donnée plus haut.

La seconde solution, pour mesurer une tension batterie hors de la plage de tension par défaut, est de remplacer les résistances du nœud.

Une fois ces problématiques matérielles résolues, de préférence par une personne avec des compétences en électronique, nous pouvons passer à la configuration logicielle. Voici un exemple de configuration :

```
13    "sensors": [{
14        "name": "Batt",
15        "type": "batteryADC",
16        "periodHr": 8,
17        "useADCLine": "ANA2",
18        "voltageDivisor": 5.867,
19        "batteryType": "Pb",
20        "batteryVoltageV": 12,
21        "batteryLowV": 11.9
22    }],
```

12.9.1 Les paramètres obligatoires

`useADCLine` : (**obligatoire**) indique l'entrée analogique utilisée pour mesurer la tension batterie. Les valeurs possibles sont : `ANA1` et `ANA2`.

`voltageDivisor` : (**obligatoire**) indique le facteur de division de la tension d'entrée (la variable r dans les équations données plus haut). C'est un nombre réel strictement positif qu'il est conseillé de calculer à partir de la valeur mesurée des résistances du pont diviseur pour une plus grande précision de mesure de la tension batterie. Par exemple, une valeur de 3,5 indique que la tension batterie est divisée par 3,5 avant d'être envoyée au convertisseur analogique-numérique. Le capteur saura ainsi qu'il doit multiplier la valeur retournée par le CAN par 3,5 pour retrouver la tension batterie réelle.

12.9.2 Les paramètres optionnels

Ils servent à détecter et à signaler que la tension batterie est basse, et qu'il est probablement temps de remplacer ou de recharger ladite batterie. Lorsqu'une tension batterie basse est détectée un drapeau de tension batterie basse est inclus dans les données envoyées par radio. Ce système n'est pas une alarme complète puisqu'il n'est pas possible de déclencher une mesure ou un envoi lors du franchissement du seuil. Il existe deux méthodes pour indiquer ce seuil bas.

La première méthode s'applique lorsque vous connaissez bien votre batterie et votre application et que vous êtes en mesure de déterminer directement le seuil bas. Un seul paramètre est alors nécessaire :

`batteryLowV` : (optionnel) indique la tension, en Volts, en dessous de laquelle, valeur comprise, la tension batterie est considérée comme basse.

La seconde méthode s'appuie sur au moins deux paramètres. Elle s'applique si votre connaissance de la batterie est moins pointue. En spécifiant la technologie utilisée par la batterie et sa tension nominale le capteur est en mesure de déterminer une tension basse standard. Il est également possible d'utiliser le paramètre `batteryLowV`, avec cette seconde méthode et en supplément des paramètres que nous allons évoquer, au cas où la tension basse standard ne convient pas à votre application.

`batteryType` : (optionnel) indique la type de la batterie ; la technologie des cellules qui composent la batterie. Si ce paramètre est défini alors le paramètre `batteryVoltageV` doit également l'être. Les valeurs possibles sont : `Pb`, `LiPo`, `LiFePO4`, `NiMH`, ou `NiCd`, pour les batteries au plomb, lithium-polymère, lithium-fer-phosphate, nickel-métal-hydrure et nickel-cadmium respectivement. La valeur (le nom de la technologie) n'est pas sensible à la casse. A titre d'information, la tension basse par cellule alors utilisée est de 2,0 V pour `Pb`, 3,2 V pour `LiPo`, 2,9 V pour `LiFePO4` et 1,18 V pour `NiMH` et `NiCd`.

`batteryVoltageV` : (optionnel) indique la tension nominale de la batterie, en Volts. Ce paramètre permet, en conjonction avec le paramètre `batteryType`, de déterminer le nombre de cellules qui composent la batterie, et ainsi de déterminer la tension basse. La valeur de ce paramètre n'a pas besoin d'être très précise puisqu'elle est uniquement utilisée pour déterminer le nombre de cellules.

Vous disposez d'une marge de plus ou moins la moitié de la tension nominale d'une cellule. Par exemple, les batteries au plomb 12 V ont en réalité une tension nominale plus proche de 12,5 – 13 V, mais indiquer une valeur de 12 V fait parfaitement l'affaire.

12.10 Configuration d'une station météorologique Gill MaxiMet

Seule la communication SDI-12 est supportée. Les stations GMX100, GMX101, GMX200, GMX240, GMX300, GMX301, GMX400, GMX500, GMX501, GMX531, GMX541, GMX550, GMX551 et GMX600 sont supportées, avec ou sans GPS. Le driver pour ces capteurs ne propose pas d'alarme pour le moment. Le type à indiquer dans la configuration pour ces capteurs est : `GillMaxiMetSDI12`.

Voici un exemple de configuration :

```
    "sensors": [{
        "name": "Meteo",
        "type": "GillMaxiMetSDI12",
        "periodHr": 1,
        "address": "0",
        "partNumber": "1957-0600-60-000",
        "measurements": "all"
    }],
```

12.10.1 Les paramètres obligatoires

`address`: (**obligatoire**) l'adresse SDI-12 du capteur. Est une chaîne de caractères qui contient un seul caractère compris entre '0' et '9'. Par défaut, en sortie d'usine, les capteurs ont l'adresse '0'.

`partNumber` : (**obligatoire**) est le numéro de produit. Son format est : 1957-0XXX-60-X00. À partir de cet identifiant, le nœud est en mesure de déterminer le modèle de la station météo, et si elle est équipée d'un GPS ou non, et donc de savoir quelles sont les mesures proposées par cette station. Vous trouverez cet identifiant sur l'étiquette collée à la base de la station (sur son pied), ou encore sur son carton d'emballage.

`measurements` : (**obligatoire**) indique les mesures que vous souhaitez récupérer de la station météo. Les stations météo sont en mesure de produire des mesures dérivées des mesures physiques, comme le point de rosée, la densité de l'air et ainsi de suite, mais nous ne récupérons pas ces données. Nous pouvons simplement récupérer les données qui correspondent directement à un capteur physique. La valeur `all` récupérera toutes les mesures physiques proposées par ce modèle de station météo. Alternativement, il vous est possible de spécifier la liste des mesures que vous souhaitez récupérer en indiquant comme valeur du paramètre une liste des noms de ces mesures. Les noms des mesures sont : `relativeWindDirection`, `relativeWindSpeed`, `correctedWindDirection`, `correctedWindSpeed`, `temperature`, `relativeHumidity`, `pressure`, `precipitation` et `solarRadiation`. Toutes ces mesures ne sont pas proposées par tous les modèles de stations météo. Si vous indiquez une mesure non disponible sur votre modèle de station météo, alors une erreur sera écrite dans le fichier de log pour vous signaler cette indisponibilité. Pour les mesures du vent, il n'est pas possible de récupérer les valeurs relatives et corrigées d'une même valeur, il vous faut choisir l'une ou l'autre. Ainsi, si vous indiquez `relativeWindDirection` et

`correctedWindDirection`, alors vous récupérerez uniquement la valeur corrigée (si elle est disponible sur votre station) ; cette dernière est prioritaire. D'ailleurs, la valeur `all` retourne les valeurs corrigées lorsqu'elles sont disponibles. Par exemple, pour récupérer uniquement la température et la pression, utilisez la configuration suivante : `"measurements": ["temperature", "pressure"]`.

12.10.2 Les paramètres optionnels

`stationResetsRainCountAtMidnight`: (optionnel, disponible à partir de la version logicielle `1.3.1`, valeur par défaut : `true`). La station météo propose deux modes de fonctionnement pour le décompte de la quantité de pluie tombée. Par défaut la station remet ce compte à zéro à minuit, heure de la station. Puisqu'il n'est pas possible de configurer l'heure de la station en SDI-12 il faut donc que le nœud s'adapte au rythme de cette dernière pour ne pas perdre de mesures à l'approche du minuit de la station. Cette adaptation se fait en modulant la période de mesure de la station par le nœud de telle manière à faire une mesure quelques secondes avant le minuit de la station. Cette modulation entraîne donc un rythme de mesure irrégulier au moins une fois par jour. Il est possible d'obtenir un rythme régulier en configurant la station météo pour qu'elle fasse un compte continu, sans remise à zéro journalière. Le nœud, en SDI-12, ne peut pas connaître la configuration de la station. L'utilisateur doit donc indiquer que ce mode de fonctionnement est utilisé au moyen du paramètre de configuration décrit ici. Utilisez la valeur `true` pour indiquer que la station procède à une remise à zéro journalière de la quantité de pluie tombée et la valeur `false` sinon.

12.11 Configuration d'un capteur de radon Algade

Cette configuration est applicable au capteur de radon personnalisé fourni par Algade, son nom est ÆR-TT. Il s'agit d'une modification de leur modèle standard ÆR+, auquel ils ont ajouté une interface de communication série TTL 3V pour que le nœud ConnecSenS puisse l'interroger. Ce modèle standard est incorporé dans un boîtier de protection avec une batterie de forte capacité pour assurer un fonctionnement autonome d'au moins trois mois.

Pour communiquer avec ce capteur il vous faut un nœud câblé pour que son interface série d'extension sorte sur l'un des connecteurs M12 selon un brochage adapté au capteur. Rapprochez-vous des personnes compétentes en la matière si vous ne disposez pas d'un tel nœud.

Le mode de fonctionnement « sensorFlow » est possible avec ce capteur, à partir de la version `2.0.0` du firmware, c'est-à-dire que le capteur donne le rythme de mesure et non pas le nœud. Ce mode de fonctionnement est d'ailleurs celui recommandé. Si vous adoptez le mode périodique classique, avec une période numérique fixée par vos soins, il faut alors que cette valeur soit inférieure ou égale à la période d'intégration programmée dans le capteur par vos soins. Vous ne raterez alors aucune valeur, mais risquez de vous trouver avec des mesures en doubles.

Le type de ce capteur est : `AlgadeAERTTSerial`.

Voici un exemple de fichier de configuration, en mode périodique « sensorFlow », avec une alarme définie :

```
    "sensors": [{
        "name": "Radon",
        "type": "AlgadeAERTTSerial",
        "period": "sensorFlow",
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": false,
            "radonLowSetBqm3":    30,
            "radonLowClearBqm3":  60,
            "radonHighSetBqm3":   200,
            "radonHighClearBqm3": 100
        }
    }],
```

Voici un exemple de fichier de configuration, en mode périodique classique, avec une alarme définie :

```
    "sensors": [{
        "name": "Radon",
        "type": "AlgadeAERTTSerial",
        "periodHr": 1,
        "alarm": {
            "sendOnAlarmSet":     true,
            "sendOnAlarmCleared": false,
            "radonLowSetBqm3":    30,
            "radonLowClearBqm3":  60,
            "radonHighSetBqm3":   200,
            "radonHighClearBqm3": 100
        }
    }],
```

12.11.1 Paramètres obligatoires

La configuration du capteur nécessite uniquement les paramètres de base de tout capteur. Il n'y a pas de paramètre spécifique de configuration.

12.11.2 Configuration d'une alarme

Ce capteur propose une alarme sur le niveau de radioactivité de l'air. Il est possible de configurer un seuil haut et/ou un seuil bas de déclenchement de l'alarme, avec la possibilité d'introduire une hystérésis pour chacun de ces seuils.

Les paramètres de configuration spécifiques de l'alarme sont :

`radonHighSetBqm3` : configure le seuil haut de déclenchement de l'alarme, en Becquerels par mètre cube d'air. Est un entier positif. L'absence de ce paramètre, ou une valeur 0, désactive le seuil haut d'alarme. L'alarme se déclenche dès que le niveau de radioactivité est supérieur ou égal à ce seuil.

`radonHighClearBqm3` : configure le seuil haut d'arrêt de l'alarme, en Becquerels par mètre cube d'air. Est un entier positif. Ce paramètre permet d'introduire une hystérésis sur le seuil haut d'alarme. Si vous n'utilisez pas ce paramètre, alors sa valeur est identique à celle de `radonHighSetBqm3` ; il n'y a alors pas d'hystérésis. L'alarme s'arrête dès que le niveau de radioactivité est inférieur ou égal à ce seuil.

`radonLowSetBqm3` : configure le seuil bas de déclenchement de l'alarme, en Becquerels par mètre cube d'air. Est un entier positif. L'absence de ce paramètre, ou une valeur 0, désactive le seuil bas d'alarme. L'alarme se déclenche dès que le niveau de radioactivité est inférieur ou égal à ce seuil.

`radonLowClearBqm3` : configure le seuil bas d'arrêt de l'alarme, en Becquerels par mètre cube d'air. Est un entier positif. Ce paramètre permet d'introduire une hystérésis sur le seuil bas d'alarme. Si vous n'utilisez pas ce paramètre, alors sa valeur est identique à celle de `radonLowSetBqm3` ; il n'y a alors pas d'hystérésis. L'alarme s'arrête dès que le niveau de radioactivité est supérieur ou égal à ce seuil.

12.12 Configuration d'une sonde de mesure d'humidité et de température du sol Truebner SMT100.

Seule la communication SDI-12 est supportée. Le type de ce capteur est : `TruebnerSMT100SDI12`.

Voici un exemple de fichier de configuration :

```
    "sensors": [{
        "name": "HumiSol",
        "type": "TruebnerSMT100SDI12",
        "periodHr": 1,
        "address": "0",
        "depthCm": 30,
        "measurements": [ "water", "temperature", "voltage", "permittivity", "raw" ]
    }],
```

12.12.1 Les paramètres obligatoires

`address`: (**obligatoire**) l'adresse SDI-12 du capteur. Est une chaîne de caractères qui contient un seul caractère compris entre '0' et '9'. Par défaut, en sortie d'usine, les capteurs ont l'adresse '0'.

`depthCm`: (**obligatoire**) la profondeur, en centimètres, à laquelle est enterré le capteur.

12.12.2 Les paramètres optionnels

`measurements`: spécifie la liste des mesures du capteur à lire. Si ce paramètre est absent, ou que sa valeur est la chaîne de caractères `all`, alors toutes les mesures sont lues. En utilisant un tableau comme valeur de ce paramètre il est possible de spécifier la liste des mesures à lire. L'extrait de configuration donné plus haut donne un exemple de configuration de la liste des mesures à conserver au moyen d'un tableau. Cet exemple liste toutes les dénominations des mesures, et revient dans les faits à ne pas spécifier le paramètre `measurements` ou à lui donner la valeur `all`. Les noms des mesures sont :

- `water` pour la mesure calibrée du volume d'eau dans le sol en pourcents.
- `temperature` pour la mesure de la température du sol.
- `voltage` pour la mesure de la tension d'alimentation de la sonde.
- `permittivity` pour la mesure calibrée de la permittivité diélectrique relative du sol.

- `raw` pour la mesure brute non calibrée en relation avec l'humidité du sol.

12.13 Configuration de la lecture d'une entrée numérique du connecteur d'extension du nœud.

Ce capteur mesure le niveau logique d'une entrée numérique du connecteur d'extension interne du nœud. Le type du capteur est : `DigitalInput`.

Ce type de capteur est disponible à partir de la version de firmware `1.2.0`.

Voici un exemple de fichier de configuration :

```
    "sensors": [{
        "name": "DigI",
        "type": "DigitalInput",
        "periodHr": 8,
        "useEXTPin": "EXT_GPIO5",
        "inputId": 1
    }],
```

12.13.1 Les paramètres obligatoires

`useEXTPin` : (**obligatoire**) indique le nom de la broche du connecteur d'extension du nœud à utiliser comme entrée numérique. Le plus simple est d'utiliser l'un des noms sérigraphiés sur le circuit imprimé autour du connecteur d'extension. Toutes les broches ne sont pas utilisables pour cette fonction. Les noms ne sont pas sensibles à la case. En voici la liste exhaustive : `EXT_GPIO1`, `EXT_GPIO2`, `EXT_GPIO3`, `EXT_GPIO4`, `EXT_GPIO5`, `OPTO_1`, `OPTO_2`, `SPI_CS`, `SPI_MISO`, `SPI_MOSI`, `SPI_CLK`, `I2C_SDA`, `I2C_SCL`, `USART_TX` et `USART_RX`. Il est également possible d'utiliser les alias suivants : `GPIO1` pour `EXT_GPIO1`, `GPIO2` pour `EXT_GPIO2`, `GPIO3` pour `EXT_GPIO3`, `GPIO4` pour `EXT_GPIO4` , `GPIO5` pour `EXT_GPIO5`, `OPTO1` et `OPTO1_INT` pour `OPTO_1`, `OPTO2` et `OPTO2_INT` pour `OPTO_2`. Vous observez que les lignes d'interruptions ne sont pas utilisables comme des entrées numériques, à l'exception des entrées opto-isolées `OPTO_1` et `OPTO_2`.

12.13.2 Les paramètres optionnels

`inputId` : (optionnel, valeur par défaut : 0) spécifie un identifiant pour l'entrée numérique, au cas où cette information serait importante pour vous. L'identifiant est un nombre entier positif compris entre 0 et 127 inclus. Cet identifiant accompagne la valeur mesurée et se retrouve donc sur le serveur de données aux côtés de l'état de l'entrée. Il faut noter qu'il n'est pas vérifié qu'un identifiant est déjà utilisé dans le cas où plusieurs entrées numériques sont configurées ; c'est à vous de vous en assurer. Il est donc possible d'attribuer un même identifiant à plusieurs entrées numériques si cette option a un sens pour votre application.

12.14 Configuration d'une interface de mesure de capteur de température de type RTD au moyen du circuit intégré MAX31865

Ce driver permet de mesurer la température au moyen d'une sonde de type RTD (Resistance Temperature Detector). Il peut également calculer la variation de température entre deux mesures

consécutives. Le nœud ConnecSenS n'est pas capable de mesurer directement une résistance avec précision (en tenant compte de la résistance des fils de liaison), il faut donc passer par une interface spécialisée. Ce pilote s'applique uniquement aux interfaces qui emploient le circuit intégré référencé MAX31865. Cette interface est reliée au nœud ConnecSenS par l'interface SPI présente sur le connecteur d'extension du nœud. L'alimentation de l'interface et du capteur peut se faire par le nœud s'ils acceptent une plage d'alimentation comprise entre 3,3 V et 5,5 V. Il est alors conseillé d'utiliser l'alimentation réglable proposée par le nœud, pensez alors à utiliser le paramètre `use5VWhenActive` avec la valeur `true`.

Ce capteur est en mesure de produire deux alarmes : température haute et/ou température basse.

Il est disponible à partir de la version `1.4.0` du firmware du nœud ConnecSenS.

À l'heure actuelle seule les sondes au platine, de type PT, sont supportées. Il n'y a en revanche pas de restriction sur la valeur de la sonde à 0 °C (types PT100, PT200, PT500, PT1000, …).

Le nom de type de ce capteur est `MAX31865`. Voici un exemple de fichier de configuration :

```
    "sensors": [{
        "name": "PT100-1",
        "type": "MAX31865",
        "periodHr": 1,
        "RTDType": "PT100",
        "computeDeltaT": true,
        "nbWires": 4,
        "RrefOmhs": 412.0,
        "filterFreqHz": 50,
        "use5VWhenActive": true,
        "alarm": {
            "temperatureHighSetDegC":   300.0,
            "temperatureHighClearDegC": 250.0,
            "temperatureLowClearDegC" : -10.0,
            "temperatureLowSetDegC"   : -50.0
        }
    }],
```

12.14.1 Les paramètres obligatoires

`RTDType` : (**obligatoire**) indique le nom du type de sonde. Cette valeur est une chaîne de caractères insensible à la casse. Seules les sondes au platine sont actuellement supportées, le nom commence donc forcément par le préfixe « PT ». Le nombre accolé au préfixe indique la résistance de la sonde à 0 °C. Si vous connaissez précisément cette résistance, alors il est possible de l'indiquer. Par exemple si cette résistance à 0 °C est de 99,867Ω, alors vous pouvez utiliser la valeur `PT99.867`. Les sondes sont cependant généralement de précision et il est donc inutile d'aller chercher plus loin que la valeur indiquée par leur nom générique.

`nbWires` : (**obligatoire**) indique le nombre de fils utilisés pour la mesure. La valeur est un entier positif. Les seules valeurs possibles sont 2, 3 et 4. Il est déconseillé d'utiliser une mesure à deux fils si vous comptez avec une précision de mesure acceptable. Le choix entre les valeurs 3 et 4 est plutôt dicté par le nombre de fils proposé par votre sonde. Si vous avez le choix, il est alors conseillé d'utiliser le câblage qui met en œuvre le plus grand nombre de fils possible.

`RrefOmhs` : (**obligatoire**) indique la valeur de la résistance de référence utilisée par le MAX31865 pour mesurer la résistance de la sonde de température. Cette valeur est un nombre réel strictement positif. Le circuit intégré calcule le ratio entre les résistances, de référence et de sonde. Il est donc nécessaire de connaître le plus précisément possible cette valeur. Elle est susceptible de varier d'une carte d'adaptation à l'autre, mais également en fonction de la sonde. Le constructeur conseille par exemple d'utiliser une valeur de 400Ω pour des sondes PT100 et de 4 kΩ pour les sondes PT1000. Mais la platine d'Adafruit utilise des valeurs de 430Ω et 4300Ω respectivement. La carte UCA utilise une valeur de 412Ω ou de 402Ω selon les cas pour les sondes PT100. Donc renseignez-vous, ou mesurez la valeur de cette résistance avec un multimètre, de précision de préférence.

12.14.2 Les paramètres optionnels

`computeDeltaT` : (optionnel, valeur par défaut : `false`) indique si la différence de température entre la mesure courante et la précédente doit être calculée et envoyée. À défaut, seule la température mesurée sera enregistrée et transmise.

`filterFreqHz` : (optionnel, valeur par défaut : 50) indique la fréquence de filtre à utiliser pour éliminer le bruit ambiant. La valeur est une nombre entier positif, dont les seules valeurs possibles sont 50 et 60. L'objectif est d'éliminer le bruit produit par les alimentations secteur. Nous ne devrions pas être confrontés à ce problème dans nos applications. Mais si l'équipement est installé dans un pays non européen, il est alors tout de même préférable d'adapter la valeur à la fréquence secteur du pays. Utilisez par exemple la valeur 60 si le capteur est aux États-Unis d'Amérique.

`SSExtPin` : (optionnel, valeur par défaut : `SPI_CS`) indique la broche du connecteur d'extension du nœud à utiliser comme chip select (slave select) du circuit intégré MAX31865. `SSExtPin` correspond à : **S**lave **S**elect **Ext**ension **Pin**. La valeur de ce paramètre est le nom de la broche ; elle n'est pas sensible à la case. Toute broche du connecteur d'extension capable de fonctionner comme une sortie numérique est utilisable. La liste exhaustive des valeurs possibles est la suivante (les alias utilisables sont indiqués entre parenthèses) : `SPI_CS`, `EXT_GPIO1` (`GPIO1`), `EXT_GPIO2` (`GPIO2`), `EXT_GPIO3` (`GPIO3`), `EXT_GPIO4` (`GPIO4`), `EXT_GPIO5` (`GPIO5`), `I2C_SDA`, `I2C_SCL`, `USART_TX` et `USART_RX`.

12.14.3 Configuration d'une alarme

Comme indiqué plus haut, ce capteur est en mesure de gérer deux alarmes ; l'une pour le dépassement d'un seuil haut de température et l'autre pour un seuil bas. Il est possible d'utiliser l'un ou l'autre ou les deux simultanément. Les deux seuils utilisent un système d'hystérésis, pour éviter des basculements trop fréquents et injustifiés entre les états d'alarme et de non-alarme. La configuration se fait au moyen de la section optionnelle `alarm`, comme dans le cas de l'exemple de configuration donné plus haut.

La configuration du seuil haut se fait au moyen des paramètres `temperatureHighSetDegC` et `temperatureHighClearDegC`, pour le seuil de déclenchement et le seuil de relâche respectivement. Les valeurs sont des nombres réels. Le seuil de déclenchement doit être strictement supérieur au seuil de relâche.

La configuration du seuil bas se fait au moyen des paramètres `temperatureLowSetDegC` et `temperatureLowClearDegC`, pour le seuil de déclenchement et le seuil de relâche respectivement. Les valeurs sont des nombres réels. Le seuil de déclenchement doit être strictement inférieur au seuil de relâche.

13 Historique des révisions

Rév.	Date	Modifications
< 6	Juin 2018	L'historique n'est pas recensé…
6	Février 2019	• Ajout du paramètre `batteryLowV`. • Ajout du capteur `soilMoistureWatermarkI2C`. • Ajout du capteur `InSituAquaTROLL200SDI12`. • Rallongement de la longueur maximale du paramètre `name` de la configuration du nœud. • Rallongement de la longueur maximale du paramètre `name` de la configuration d'un capteur. • Ajout du paramètre de configuration `uniqueId` pour les capteurs. • Ajout du paramètre de configuration `uniqueId` pour le nœud. • Ajout des alias `OPTO_1` , `OPTO_2` , `SDI_INT` , `INT_1` et `INT_2` comme noms d'interruptions. • Ajout du paramètre de configuration `sendConfigPeriodDay`. • Ajout du capteur `batteryADC`. • Réorganisation de la configuration du temps. Introduction du paramètre de configuration `syncMethod` et de l'objet JSON `GPS`. • Introduction de la section de configuration `debug`. • Ajout du paramètre de configuration `firmwareVersion` pour les capteurs. • Ajout de la section `buzzer` et du paramètre `useBuzzer` dans la section `debug`. • Détection de la connexion USB en mode défaut ; mode déclenché à cause d'un fichier de configuration erroné. Il n'est plus nécessaire d'appuyer sur le bouton reset. • Ajout du support d'une périodicité de mesure des capteurs différente en cas d'alarme. • Renommage du paramètre de débogage `leds` en `useInternalLEDs`. • Renommage du paramètre général `watchdog` en `useWatchdog`. • Ajout du capteur `GillMaxiMetSDI12`. • Ajout du capteur `AlgadeAERTTSerial`. • Ajout du capteur `TruebnerSMT100SDI12`. • Description de la méthode améliorée pour la mise à l'heure manuelle

Rév.	Date	Modifications
		du nœud. • Ajout du paramètre réseau `publicNetwork`. • Ajout du paramètre GPS `timeoutMn`. • Ajout du paramètre réseau `periodSpreadXx`.
7	Avril 2019	• Ajout du paramètre général `outputCSVData`.
8	Juin 2019	• Ajout du capteur `DigitalInput`. • Ajout du paramètre général `experimentName`. • Ajout du paramètre de configuration GPS `blockingTimeoutSec`. • Change la valeur par défaut du paramètre GPS `timeoutSec` de 2 minutes à 5 minutes.
9	Juillet 2019	• Ajout du paramètre général `addGeoPosToAllReadings`. • Ajout du paramètre de configuration LoRaWAN `sendTimeoutSec`.
10	Août 2019	• Ces changements sont en relations avec le travail sur le changement de numéro majeur de version de la version 1 vers la version 2. • Ajout de la section `logs` et de ses paramètres `defaultLevel` et `serial`. • Suppression des paramètres `verbose` et `serial` de la section `debug`. • Suppression du paramètre `sendTimeoutSec` de configuration réseau de type `LoRaWAN`. • Ajout des paramètres `joinNbTrials` et `sendAckNbTrials` de configuration réseau `LoRaWAN`. • Ajout des paramètres généraux `joinOnSendFailsCount`, `joinTimeoutSec` et `sendTimeoutSec` de configuration réseau. • Suppression du paramètre de configuration général `useWatchdog`. • Ajout de la possibilité d'utiliser une période non numérique pour la lecture de certains capteurs, mais au rythme où ils produisent des données. Ce mode de fonctionnement est activé en utilisant le paramètre `period` avec la valeur `sensorFlow` en lieu et place du paramètre de période habituel. Tous les capteurs ne supportent pas ce mode de lecture. • Mise à jour de la documentation du capteur de radon Algade ÆR-TT pour indiquer qu'il supporte le mode de lecteur « sensorFlow ». • Ajout du paramètre de configuration `stationResetsRainCountAtMidnight` pour le driver SDI-12 de la station météo Gill MaxiMet.
11	Octobre 2019	• Ajout du paramètre de configuration général des capteurs `useInt3V3WhenActive`. • Ajout du capteur `MAX31865`.

Rév.	Date	Modifications
		• Changement de la version à partir de laquelle la section `logs` a été introduite. Car le système de log de la branche 2.x.x a été intégré dans la branche 1.x.x à partir de la version `1.4.0`. • Ajout du paramètre réseau `periodWhenAlarm` pour spécifier une période d'envoi différente si un capteur est en alarme.
12	Mars 2020	• Ajout du paramètre `SSExtPin` pour le capteur `MAX31865`. Ce paramètre n'est pas nouveau. Il s'agit ici de la correction d'un oubli.