VirtualHub

Mode d'emploi

Table des matières

1. Introduction **1**

2. Installation **3**

2.1. Installation à partir d'un fichier .zip 3
2.2. Installation avec msi sous Windows 4
2.3. Installation avec apt_get sous Linux 4
2.4. Linux et USB 5
2.5. Limitation d'accès à USB 6
2.6. Paramètres de la ligne de commande 6

3. Configuration et test des modules **9**

3.1. Localisation des modules 10
3.2. Test des modules 10
3.3. Configuration des modules 11
3.4. Mise à jour des firmwares 11
3.5. Accès à l'enregistreur de données des capteurs 13

4. Utilisation de VirtualHub comme une passerelle **15**

4.1. Passerelle pour contourner la limitation d'accès à USB 15
4.2. Passerelle REST 16
4.3. Passerelle OpenMetrics (Prometheus) 16

5. Contrôle d'accès **19**

5.1. Accès "admin" 20
5.2. Accès "user" 20
5.3. Influence sur les API 20

6. Envoi de données vers l'extérieur **21**

6.1. Configuration 21
6.2. Callbacks HTTP vers des services tiers 22
6.3. Callbacks vers un broker MQTT 23
6.4. Callbacks de type Yocto-API 25
6.5. Callbacks HTTP définis par l'utilisateur 26
6.6. Noms associés aux valeur postées 27

6.7. Planification des callbacks 30
6.8. Tests 30
6.9. Connexions spontanées 31

7. Compléments optionnels **33**
7.1. L'utilitaire Windows VirtualHub Control 33
7.2. Installation de Yocto-Visualization (for web) 34

8. VirtualHub 2.0 ..... **35**
8.1. Connexions entrantes 35
8.2. Connexions sortantes 35
8.3. IPv6 35
Blueprint 37
8.4. Systèmes d'exploitation supportés 37

1. Introduction

VirtualHub est une application essentiellement destinée à gérer les modules USB conçus par Yoctopuce. C'est un genre de boîte à outils qui a pour but:

- d'offrir l'accès aux modules USB depuis des languages qui, tels que Javascript et PHP, ne permettent pas d'accéder aux couches matérielles d'un ordinateur.
- d'offrir l'accès aux modules USB à travers une connection réseau, et ce dans tous les langages disponibles.
- de configurer et tester les modules USB Yoctopuce.
- d'offrir la connectivité nécessaire pour permettre aux modules Yoctopuce d'interragir avec un service cloud-based.

VirtualHub n'est **pas** indispensable pour contrôler directement des modules Yoctopuce dans les langages qui permettent un accès natif au hardware (C++, C#, Python, Java, Android, Delphi, Visual Basic, LabView et l'API en ligne de commande). Dans ces langages, les modules USB Yoctopuce peuvent être contrôlés **directement** sans même l'aide d'un driver.

VirtualHub est disponible pour les systèmes d'exploitation Windows, macOS et Linux (Intel et ARM). Son fonctionnement est identique sur les trois systèmes.

2. Installation

VirtualHub ne nécessite pas véritablement d'installation. Vous pouvez le télécharger soit en version .zip pour Windows, Linux et macOS, soit en version .msi pour Windows, ou encore utiliser apt_get sous Linux.

2.1. Installation à partir d'un fichier .zip

Le fichier .zip contient quelques fichiers .txt et un simple fichier exécutable qui peut se trouver soit au même niveau que les fichiers .txt, soit dans un répertoire correspondant à son architecture. Copiez l'exécutable correspondant à votre architecture où bon vous semble et lancez-le depuis une ligne de commande. Aucun driver n'est nécessaire.

VirtualHub a besoin de sauvegarder quelques paramètres de configuration. Ils sont sauvés dans le fichier `.virtualhub.dat`, qui sera placé dans le répertoire `AppData\Roaming` de l'utilisateur sous Windows, et dans le *homedir* de l'utilisateur sous Linux et macOS. Ce comportement peut être modifié à l'aide de l'option `-c` sur la ligne de commande.

Sous Windows, si vous ne souhaitez pas devoir lancer explicitement VirtualHub à chaque fois que vous en avez besoin, vous pouvez l'installer comme un service. Pour ce faire, il vous suffit d'ouvrir une fenêtre de commande avec les droits administrateurs, et de lancer une fois VirtualHub en ligne de commande avec l'option `-i`. Il sera alors installé comme service avec démarrage automatique. Si vous changez d'avis ultérieurement, lancez-le à nouveau en ligne de commande avec l'option `-u` pour supprimer le service. Notez que lorsque que VirtualHub fonctionne comme un service, il est lancé par l'utilisateur spécial `SYSTEM` et le répertoire dans lequel il sauve sa configuration est `\windows\system32\config\systemprofile\AppData\Roaming`.

Sous Linux, pour permettre aux utilisateurs sans privilèges de communiquer avec les périphériques Yoctopuce, vous devez apporter quelques modifications aux règles udev. Vous pouvez le faire en exécutant la ligne de commande suivante :

```
sudo VirtualHub --install_udev_rule
```

Vous devez ensuite redémarrer l'ordinateur pour que la nouvelle règle soit prise en compte (pour plus de détails sur les modifications apportées aux règles udev, voir la section 2.4 ci-dessous).

En outre, si vous ne souhaitez pas lancer explicitement VirtualHub à chaque fois que vous en avez besoin, vous pouvez l'installer en tant que service. Pour ce faire, exécutez la commande suivante :

```
sudo VirtualHub -i
```

Il est alors installé en tant que service système et démarre automatiquement au démarrage. Si vous changez d'avis par la suite, exécutez à nouveau le programme en ligne de commande avec l'option `-u` pour supprimer le service.

2.2. Installation avec msi sous Windows

Si vous préférez installer VirtualHub à l'aide du fichier .msi, téléchargez-le et double-cliquez dessus pour démarrer l'installeur.

L'installeur de VirtualHub est très simple, il demande d'accepter les conditions d'utilisation de VirtualHub et quel répertoire utiliser. Par défaut, l'installeur va utiliser le répertoire `C:\Program Files (x86)\Yoctopuce\VirtualHub\`. Il effectue ensuite les tâches suivantes:

1. Extraire le VirtualHub dans le répertoire de destination
2. Extraire l'utilitaire VirtualHub Control dans le répertoire de destination
3. Créer une section Yoctopuce dans le Menu Démarrer
4. Créer une entrée "VirtualHub UI" et "VirtualHub Control" dans cette section
5. Ajouter le VirtualHub au PATH de la machine
6. Ajouter le VirtualHub à la liste des programmes installés sur la machine

A la fin, il est possible de lancer VirtualHub directement depuis le menu Démarrer.

Grâce à l'installeur, VirtualHub est reconnu comme un programme Windows à part entière. En particulier, vous pouvez vérifier directement depuis le panneau de configuration de Windows quelle est la version installée, gérer son installation par des outils d'entreprise centralisés et le désinstaller en un click depuis le panneau de configuration.

Comme la majorité des installeurs, dans la plupart des cas, vous pouvez simplement lancer l'installeur, cliquer "Next" à chaque panneau et finalement sur "Install". L'installeur est capable de se rendre compte si une ancienne version de VirtualHub est installée et simplement mettre à jour les exécutables sans perdre les fichiers de configuration.

L'installeur ajoutera également un utilitaire, spécifique à Windows, qui s'appelle **VirtualHub Control** et pour lequel vous trouverez des informations dans le chapitre "L'utilitaire Windows VirtualHub Control" à la fin de ce manuel.

2.3. Installation avec apt_get sous Linux

Le repository APT de Yoctopuce permet d'installer VirtualHub et d'autres utilitaires sur n'importe quelle distribution Linux qui utilise `apt-get`. Les distributions les plus connues sont Debian, Ubuntu et Raspbian, mais toutes les distributions qui sont basées sur ces dernières devraient aussi fonctionner.

Avant de pouvoir utiliser `apt-get`, le repository APT de Yoctopuce doit avoir été ajouté au système. Si ce n'est pas déjà le cas, il faut l'ajouter avant l'installation de VirtualHub.

Ajouter le repository APT de Yoctopuce

1. Installer la clef GPG du repository dans le répertoire `/usr/share/keyrings/`

```
wget -q -O - https://www.yoctopuce.com/apt/KEY.gpg | gpg --dearmor  | sudo tee -a /usr/
share/keyrings/yoctopuce.gpg > /dev/null
```

2. Ajouter le repository Yoctopuce à la liste des serveurs utilisés par `apt-get`

```
echo 'deb [signed-by=/usr/share/keyrings/yoctopuce.gpg] https://www.yoctopuce.com/ apt/
stable/' | sudo tee -a /etc/apt/sources.list.d/yoctopuce.list  > /dev/null
```

Installer VirtualHub

Il faut commencer par mettre à jour la liste des packages disponibles, avant de lancer l'installation proprement dite du VirtualHub:

```
sudo apt-get update
sudo apt-get install virtualhub
```

Finalement, pour permettre aux utilisateurs non privilégiés de communiquer avec les modules Yoctopuce, il faut faire quelques modifications aux règles *udev*. Ceci peut être fait automatiquement en lançant la ligne de commande suivante:

```
sudo VirtualHub --install_udev_rule
```

Il faut ensuite redémarrer la machine pour que la nouvelle règle soit prise en compte.

En outre, si vous ne souhaitez pas lancer explicitement VirtualHub à chaque fois que vous en avez besoin, vous pouvez l'installer en tant que service. Pour ce faire, exécutez la commande suivante :

```
sudo VirtualHub -i
```

Il est alors installé en tant que service système et démarre automatiquement au démarrage. Si vous changez d'avis par la suite, exécutez à nouveau le programme en ligne de commande avec l'option `-u` pour supprimer le service.

Si vous désirez comprendre le détail des changements effectués sur les règles *udev*, ou les effectuer manuellement vous-même, vous pouvez lire le paragraphe ci-dessous. Sinon, vous pouvez passer directement à la section suivante.

2.4. Linux et USB

Pour fonctionner correctement sous Linux, VirtualHub a besoin d'avoir accès en écriture à tous les périphériques USB Yoctopuce. Or, par défaut, sous Linux les droits d'accès des utilisateurs non-root à USB sont limités à la lecture. Afin d'éviter de devoir lancer les exécutables en tant que root, il faut créer une nouvelle règle *udev* pour autoriser un ou plusieurs utilisateurs à accéder en écriture aux périphériques Yoctopuce.

Pour ajouter une règle *udev* à votre installation, il faut ajouter un fichier avec un nom au format "`##-nomArbitraire.rules`" dans le répertoire "`/etc/udev/rules.d`". Lors du démarrage du système, *udev* va lire tous les fichiers avec l'extension "`.rules`" de ce répertoire en respectant l'ordre alphabétique (par exemple, le fichier "`51-custom.rules`" sera interprété APRES le fichier "`50-udev-default.rules`").

Le fichier "`50-udev-default`" contient les règles *udev* par défaut du système. Pour modifier le comportement par défaut du système, il faut donc créer un fichier qui commence par un nombre plus grand que 50, qui définira un comportement plus spécifique que le défaut du système. Notez que pour ajouter une règle vous aurez besoin d'avoir un accès root sur le système.

Dans le répertoire `udev_conf` de l'archive de VirtualHub[1] pour Linux, vous trouverez deux exemples de règles qui vous éviteront de devoir partir de rien.

Exemple 1: 51-yoctopuce.rules

Cette règle va autoriser tous les utilisateurs à accéder en lecture et en écriture aux périphériques Yoctopuce USB. Les droits d'accès pour tous les autres périphériques ne seront pas modifiés. Si ce scénario vous convient, il suffit de copier le fichier "`51-yoctopuce_all.rules`" dans le répertoire "`/etc/udev/rules.d`" et de redémarrer votre système.

[1] http://www.yoctopuce.com/EN/virtualhub.php

```
# udev rules to allow write access to all users
# for Yoctopuce USB devices
SUBSYSTEM=="usb", ATTR{idVendor}=="24e0", MODE="0666"
```

Exemple 2: 51-yoctopuce_group.rules

Cette règle va autoriser le groupe "yoctogroup" à accéder en lecture et écriture aux périphériques Yoctopuce USB. Les droits d'accès pour tous les autres périphériques ne seront pas modifiés. Si ce scénario vous convient, il suffit de copier le fichier "`51-yoctopuce_group.rules`" dans le répertoire "`/etc/udev/rules.d`" et de redémarrer votre système.

```
# udev rules to allow write access to all users of "yoctogroup"
# for Yoctopuce USB devices
SUBSYSTEM=="usb", ATTR{idVendor}=="24e0", MODE="0664", GROUP="yoctogroup"
```

2.5. Limitation d'accès à USB

Les modules USB Yoctopuce ont une limitation: sur une machine donnée, il ne peut y avoir qu'une seule application à la fois qui les contrôle nativement. Et il se trouve que VirtualHub compte pour une application native. En conséquence, si vous tentez de lancer une application qui contrôle nativement des modules Yoctopuce USB, veillez à ce que VirtualHub ne soit pas en train de tourner, que ce soit en ligne de commande ou en service.

Notez que du point de vue programmation, cette limitation peut facilement être contournée en faisant en sorte que vos applications utilisent un VirtualHub comme passerelle pour contrôler les modules au lieu de les contrôler directement. Pour ce faire, il ne faut pas passer "usb" comme paramètre lors de l'appel à `YAPI.RegisterHub`, mais l'adresse de la machine sur lequel tourne VirtualHub. L'utilisation de VirtualHub comme passerelle est décrite dans le chapitre "Utilisation de VirtualHub comme une passerelle".

```
YAPI.RegisterHub('localhost:4444',errmsg);
```

2.6. Paramètres de la ligne de commande

VirtualHub accepte divers paramètres sur la ligne de commande.

Option	Brève description
-c	Indique le chemin où le fichier de configuration est enregistré
-d	Tourne en arrière-plan
-F	Active le mode Files
-g	Redirige les logs dans un fichier
-h	Affiche une aide succinte en anglais
-i	Installe le service
--install_udev_rule	Donne accès aux modules USB à tous les utilisateurs (Linux)
-n	Utilise l'interface réseau
-o	Active l'interface OSControl
-p	Change le port
-T	Utilise un modèle pour créer le fichier de configuration
-u	Désinstalle le service
-v	Indique la version

-c : Fichier de configuration

Sous Windows, quand on lance manuellement VirtualHub dans une fenêtre de commande, il utilise le fichier de configuration à l'emplacement suivant:

```
C:\Users\username\AppData\Roaming\.virtualhub.dat
```

Par contre, lorsque VirtualHub est lancé par Windows comme un service, il utilise l'emplacement:

```
C:\Windows\System32\config\systemprofile\AppData\Roaming\.virtualhub.dat
```

Sous Linux et macOS, le fichier de configuration .virtualhub.dat se trouve directement dans le Home directory de l'utilisateur.

L'option -c permet de changer cet emplacement. Par exemple

```
>virtualhub -c C:\tmp\mysetting.bin
```

Par ailleurs, pour le retrouver plus facilement, l'emplacement du fichier de configuration est indiqué dans les logs.

-d : Démarrage en service/démon

Démarre VirtualHub en arrière plan. S'utilise uniquement quand VirtualHub tourne comme service.

-F : Activation de l'option Files

Active l'interface Files pour l'ajout de fichiers personnalisés dans VirtualHub. Le conteneur est un fichier au format .tar et sera créé s'il n'existe pas.

```
>virtualhub -F container.tar
```

-g : Enregistrement des informations de debug dans un fichier

Enregistre dans un fichier les informations de debug.

Exemple de ligne de commande:

```
>virtualhub -g tracefile.log
```

-h : Aide

Force VirtualHub à afficher une aide succinte

-i : Installation du service

VirtualHub peut fonctionner en service, cette option installe le service et le démarre. Ainsi VirtualHub sera disponible en permance, même si la machine redémarre.

--install_udev_rule : Installation d'une règle UDEV

Par défaut, Linux ne donne pas accès aux périphériques USB pour les utilisateurs standards. Pour autoriser tous les utilisateurs à communiquer avec les modules Yoctopuce, il faut ajouter une règle UDEV et redémarrer la machine. L'installation de cette règle peut être faite grâce à l'otpion --install_udev_rule. **Ne pas oublier de redémarrer la machine** pour que la règle soit prise en compte. De plus, comme il s'agit d'une modification du système, il faudra exécuter VirtualHub avec les droits superviseur à l'aide de la commande `sudo`.

Exemple de ligne de commade :

```
>sudo VirtualHub --install_udev_rule
```

Une fois la machine redémarrée, VirtualHub peut utiliser les modules branchés sur les ports USB sans sudo.

-n : Utilisation de l'interface réseau avec une adresse IP

Utilise l'interface réseau avec l'adresse IP spécifiée en paramètre.

-o : Activation de la fonction osControl

Ajoute la fonctionnalité *osControl* au VirtualHub, ce qui permet entre autres d'éteindre à distance la machine qui fait tourner VirtualHub en utilisant l'API Yoctopuce.

-p : Changement de port

Par défaut, VirtualHub utilise le port TCP 4444, cette option permet d'en utiliser un autre. Par exemple:

```
>virtualhub -p 8889
```

-T : Création d'un fichier de configuration à partir d'un template

Si aucun fichier de configuration n'est trouvé, crée un fichier de configuration sur la base du modèle de configuration. Cette option permet d'utiliser des réglages par défaut autres que ceux proposés d'usine (par exemple pour imposer un mot de passe dès l'installation), mais sans empêcher ultérieurement de modifier la configuration. Le fichier de configuration résultant est enregistré à l'emplacement par défaut (voir option -c).

Exemple de ligne de commade :

```
>virtualhub -T config_template
```

-u : Désinstallation du service

Désinstalle le service préalablement installé avec l'option -i

-v : Version

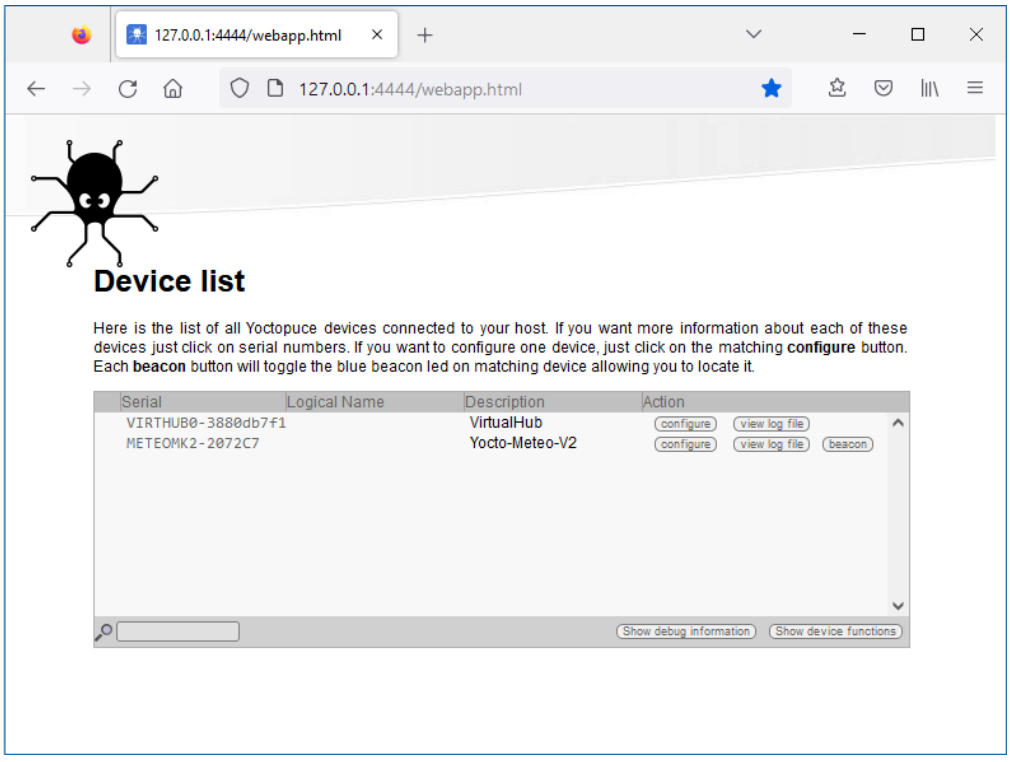
Permet d'afficher la version de VirtualHub:

```
>virtualhub -v

Version v1.0 (4237)
```

3. Configuration et test des modules

Une fois installé et configuré, VirtualHub permet de tester et configurer vos modules Yoctopuce. Pour ce faire, ouvrez votre navigateur internet favori[1]. Connectez-vous en *HTTP* au port 4444 de la machine sur laquelle tourne VirtualHub. S'il s'agit de la machine locale, utilisez l'adresse `http://127.0.0.1:4444`. La liste des modules connectés à la machine devrait apparaître.



VirtualHub: Interface Web

[1] L'interface du VirtualHub est régulièrement testée sur Firefox, Chrome, Opera et Brave. Elle fonctionne probablement avec Safari.

En bas de page se trouvent deux boutons. Le premier bouton, **Show debug information**, permet d'afficher et ensuite d'enregistrer toutes les informations nécessaires pour débugger un problème lié au VirtualHub, c'est-à-dire:

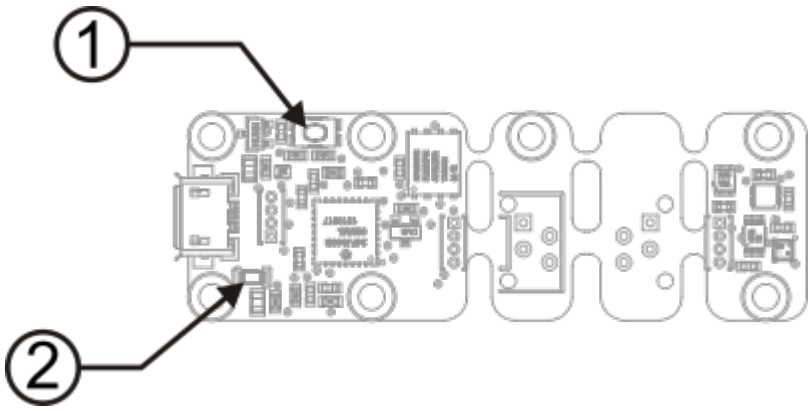
- La liste de tous les modules détectés.
- La valeur de tous les paramètres de tous les modules (sans les mots de passe).
- Les logs de tous les modules.
- La liste de tous les fichiers uploadés sur les modules, mais pas leur contenu.
- Le contenu des éventuels core dump de VirtualHub.

Si vous devez contacter le support, il est important de télécharger ces informations et de les envoyer avec votre demande.

Le deuxième bouton, **Show device functions**, montre toutes les fonctions du VirtualHub et de chacun des modules connectés au VirtualHub.

3.1. Localisation des modules

L'interface principale vous montre une ligne par module connecté, si vous avez plusieurs modules du même modèle, vous pouvez localiser un module particulier en cliquant sur le bouton **beacon** correspondant: cela aura pour effet de faire clignoter la led bleue du module et d'afficher sur l'interface une pastille bleue au début de la ligne correspondante. Vous pouvez faire la même manipulation en appuyant sur le Yocto-bouton d'un module connecté.

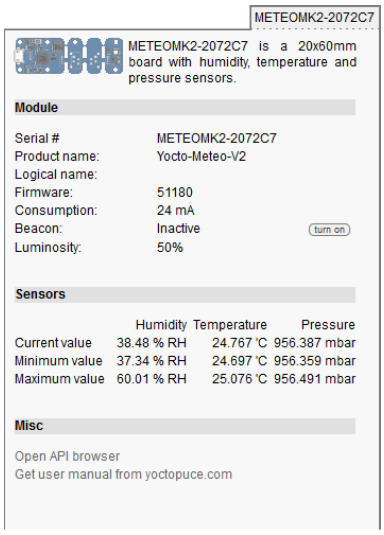


Yocto-bouton (1) et led de localisation (2) d'un module Yocto-Meteo-V2. Ces deux éléments sont toujours placés au même endroit, quelque soit le module.

3.2. Test des modules

Pour tester un module, cliquez simplement sur le numéro de série d'un module dans l'interface, une fenêtre spécifique au module s'ouvrira. Cette fenêtre permet généralement d'activer les fonctions principales du module. Reportez vous au manuel du module correspondant pour plus de détails.

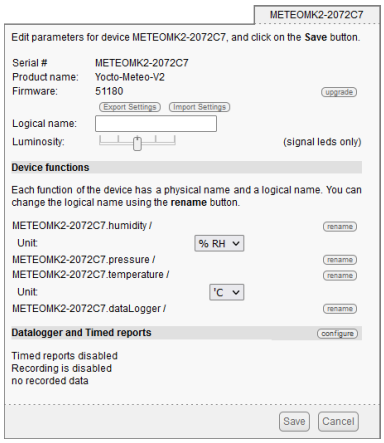
En général, vous n'êtes pas obligé d'avoir une version de VirtualHub plus récente que le module que vous voulez tester/configurer: la plupart des éléments spécifiques aux interfaces des modules sont stockés dans le firmware des modules eux-même. Il y a toutefois quelques exceptions, donc si vous rencontrez une erreur dans l'interface Web d'un module, vérifiez si une mise à jour de VirtualHub est disponible, et le cas échéant installez-là. Il peut ensuite être nécessaire de recharger la page dans le navigateur avec Shift-Reload, ou de vider le cache de votre navigateur, afin de forcer la mise à jour du code JavaScript.



Fenêtre "détails" du module Yocto-Meteo-V2

3.3. Configuration des modules

Vous pouvez configurer un module en cliquant sur le bouton **configure** correspondant dans l'interface principale, une fenêtre spécifique au module s'ouvre alors. Cette fenêtre permet au minimum de donner un nom logique au module ainsi que de mettre à jour son firmware. Reportez-vous au manuel du module correspondant pour plus de détails.



Fenêtre "configure" du module Yocto-Meteo-V2

3.4. Mise à jour des firmwares

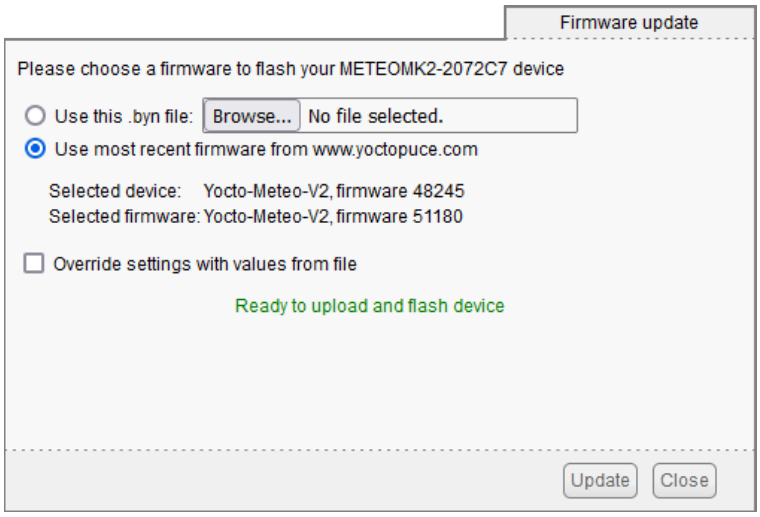
Les modules Yoctopuce sont en fait de véritables ordinateurs, ils contiennent même un petit serveur web. Et comme tous les ordinateurs, il est possible de mettre à jour leur logiciel de contrôle (firmware). Des nouveaux firmwares pour chaque module sont régulièrement publiés, ils permettent généralement d'ajouter de nouvelles fonctionnalités au module, et/ou de corriger d'éventuels bugs[2].

Méthode recommandée

Pour mettre à jour le firmware d'un module, il suffit de d'ouvrir, dans l'interface de VirtualHub, la fenêtre de configuration du module à mettre à jour, puis cliquer sur le bouton **upgrade**. Si vous cliquez simplement sur le bouton **Update**, VirtualHub utilisera la version le plus récente du firmware publiée sur le site Web de Yoctopuce et l'installera.

VirtualHub vous permet aussi de choisir un fichier .byn que vous avez préalablement téléchargé depuis le site web de Yoctopuce, par exemple pour réinstaller une version antérieure.

[2] Ne faites jamais confiance à des gens qui vous disent que leur logiciel n'a pas de bug :-)

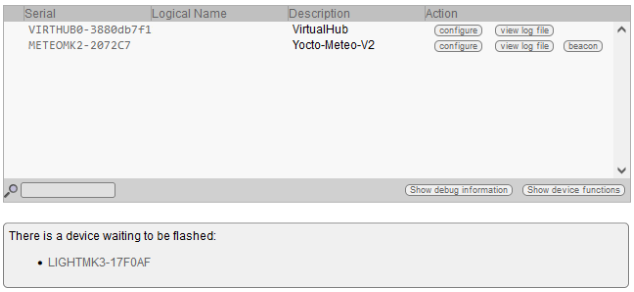


Fenêtre de mise à jour du firmware

Une fois que vous avez cliqué sur **Update**, tout est automatique, VirtualHub fait redémarrer le module en mode "mise à jour", met à jour le firmware, puis rédémarre le module en mode normal. Les réglages de configuration du module seront préservés. Ne débranchez pas le module pendant la procédure de mise à jour.

Méthode alternative

Si la mise à jour d'un module se passe mal, en particulier si le module a été débranché pendant le processus, il risque fort de ne plus fonctionner et de ne plus apparaître dans la listes des modules. Dans ce cas débranchez-le, attendez quelques secondes, et rebranchez-le en maintenant le Yocto-bouton appuyé. Cela a pour effet de faire démarrer le module en mode "mise à jour". Ce mode de fonctionnement est protégé contre les corruptions et devrait toujours être accessible. Une fois le module rebranché, provoquez un rafraîchissement de la liste des modules dans l'interface de VirtualHub et votre module devrait être listé dans le bas de l'interface. Cliquez dessus pour mettre à jour son firmware. Ce mode de mise à jour est une procédure de récupération, elle ne sauvegarde pas les réglages du module.



Les modules en mode "mise à jour" sont listés dans l'interface.

La mise à jour normale préserve la configuration des modules, en revanche la méthode alternative avec le Yocto-bouton remet le module dans sa configuration d'usine. Ce qui fait que vous pouvez aussi utiliser cette méthode pour réinitialiser complètement un module.

Par ligne de commande ou programmation

Tous les outils en lignes de commandes ont la possibilité de mettre à jour les modules Yoctopuce grâce à la commande `downloadAndUpdate`. Le mécanisme de sélection des modules fonctionne comme pour une commande traditionnelle. La [cible] est le nom du module qui va être mis à jour. Vous pouvez aussi utiliser les alias "any" ou "all", ou encore une liste de noms, séparés par des virgules, sans espace.

L'exemple suivant, qui utilise la librairie Yoctopuce en ligne de commande, télécharge automatiquement les derniers firmwares depuis le site web de Yoctopuce et met à jour tous les modules Yoctopuce connectés par USB:

```
C:\>YModule all downloadAndUpdate
ok: Yocto-PowerRelay RELAYHI1-266C8(rev=15430) is up to date.
ok: 0 / 0 hubs in 0.000000s.
ok: 0 / 0 shields in 0.000000s.
ok: 1 / 1 devices in 0.130000s 0.130000s per device.
ok: All devices are now up to date.
C:\>
```

Ce second exemple installe le firmware LIGHTMK3.51180.byn, stocké dans le répertoire local C:\tmp \yfirmware, sur le module dont le numéro de série est LIGHTMK3-23BBDF. Les fichiers de firmware peuvent être téléchargés manuellement depuis le site web de Yoctopuce[3].

```
C:\>ymodule LIGHTMK3-23BBDF updateFirmware C:\tmp\yfirmware\LIGHTMK3.51180.byn
OK: LIGHTMK3-23BBDF.module.updateFirmware = 36% Wait for device.
OK: LIGHTMK3-23BBDF.module.updateFirmware = 50% Flash zone.
OK: LIGHTMK3-23BBDF.module.updateFirmware = 57% Flash zone.
OK: LIGHTMK3-23BBDF.module.updateFirmware = 64% Flash zone.
OK: LIGHTMK3-23BBDF.module.updateFirmware = 72% Flash zone.
OK: LIGHTMK3-23BBDF.module.updateFirmware = 73% Device info retrieved.
OK: LIGHTMK3-23BBDF.module.updateFirmware = 90% Firmware updated.
OK: LIGHTMK3-23BBDF.module.updateFirmware = 100% success.
OK: LIGHTMK3-23BBDF.module.updateFirmware = 100% Firmware Updated Successfully in: 14.305s.
```

Par défaut, `downloadAndUpdate` met à jour les firmwares uniquement vers une version plus récente, grâce à un second argument optionnel, `onlyNew`, qui est toujours vrai s'il est omis. Si vous voulez installer un firmware plus ancien (downgrade), vous devez passer le firmware comme premier paramètre et `false` comme deuxième.

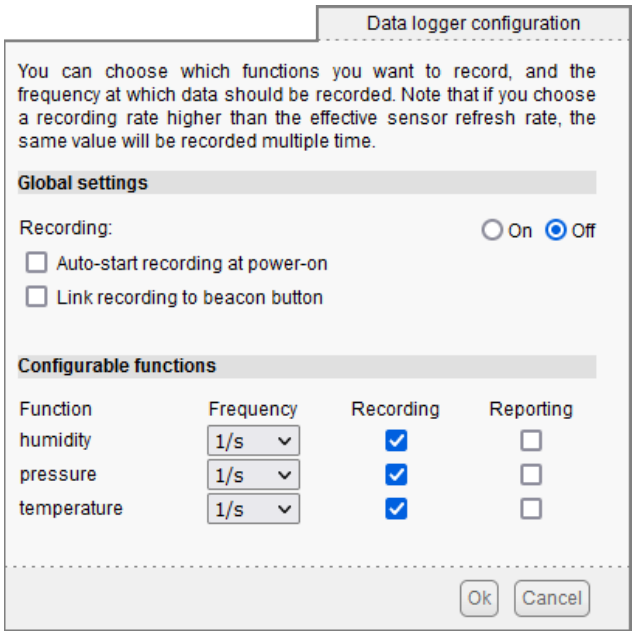
Il est également possible de mettre à jour le firmware de vos modules en utilisant la librairie de programmation Yoctopuce, en particulier à l'aide des méthodes `YModule.checkFirmware` et `YModule.updateFirmware`. Vous trouverez plus d'information à ce sujet dans les chapitres de programmation avancée figurant dans la documentation de chaque module.

3.5. Accès à l'enregistreur de données des capteurs

Pour tous les capteurs Yoctopuce qui incluent un enregistreur de données, la fenêtre de configuration inclut une section spéciale permettant de configurer l'enregistrement et de charger les données brutes contenues dans l'enregistreur, module par module.

Cliquez sur le bouton **configure** de la section **Datalogger and Timed reports**

[3] www.yoctopuce.com/FR/firmwares.php



Fenêtre de configuration du datalogger

Il est aussi possible d'installer comme plug-in de VirtualHub l'outil *Yocto-Visualization (for Web)*, qui offre des possibilité bien plus étendues pour visualiser les données sous forme de graphiques, et charger un fichier CSV correspondant à tous les capteurs connectés. Pour plus de détails, référez-vous au chapitre qui est consacré à l'installation de *Yocto-Visualization (for Web)* à la fin de ce manuel.

4. Utilisation de VirtualHub comme une passerelle

La fonction la moins spectaculaire, mais néanmoins la plus utile de VirtualHub, consiste à offrir une passerelle réseau pour contrôler les modules. Cela permet d'une part d'offrir un accès aux languages comme JavaScript, qui par nature interdisent d'accéder aux ressources physiques d'une machine. D'autre part cela permet d'offrir un accès aux modules à travers le réseau dans tous les langages: les libraries Yoctopuces sont en effet capables de se connecter à un VirtualHub à travers le réseau.

Pour utiliser VirtualHub comme passerelle, il vous suffit de le lancer en ligne de commande ou en service sur la machine à laquelle sont connectés les modules que vous voulez contrôler. Les applications qui veulent se connecter à un VirtualHub doivent initialiser l'API en appelant la fonction `YAPI.RegisterHub` avec d'adresse IP de la machine faisant tourner VirtualHub, le port par défaut est 4444. Par exemple

```
YAPI.RegisterHub('192.168.1.6:4444',errmsg);
```

Si l'application et VirtualHub tourne sur la même machine, utilisez l'adresse 127.0.0.1. Consultez la documentation de l'API de programmation [1] pour plus de détails.

4.1. Passerelle pour contourner la limitation d'accès à USB

Comme évoqué dans la section "Limitation d'accès à USB", une seule application à la fois peut avoir accès nativement aux modules Yoctopuce. Cette limitation est liée au fait que deux processus différents ne peuvent pas parler en même temps à un périphérique USB. En général, ce type de problème est réglé par un driver qui se charge de faire la police pour éviter que plusieurs processus ne se battent pour le même périphérique. Mais comme vous l'avez probablement remarqué, les produits Yoctopuce n'utilisent pas de drivers. Par conséquent, le premier processus qui arrive à accéder au mode natif le garde pour lui jusqu'à ce que `UnregisterHub` ou `FreeApi` soit appelé.

Si votre application essaie de communiquer en mode natif avec les modules Yoctopuce, mais qu'une autre application vous empêche d'y accéder, vous revecrez le message d'erreur suivant:

```
Another process is already using yAPI
```

La solution est d'utiliser VirtualHub localement sur votre machine et de vous en servir comme passerelle pour vos applications. Ainsi, si toutes vos application utilisent VirtualHub, vous n'aurez plus de conflit et vous pourrez accéder en tout temps à tous vos modules.

[1] http://www.yoctopuce.com/FR/libraries

Pour passer du mode natif au mode réseau sur votre machine locale, il vous suffit de changer le paramètre de l'appel à `YAPI.RegisterHub` et d'indiquer `127.0.0.1` à la place de `usb`:

```
YAPI.RegisterHub("usb",errmsg);  // mode natif USB

YAPI.RegisterHub("127.0.0.1",errmsg);  // utilisation en mode réseau local
```

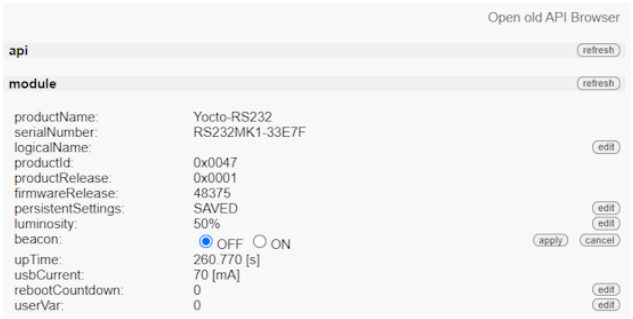
4.2. Passerelle REST

Vous pouvez également utiliser VirtualHub comme passerelle REST. Cela consiste à envoyer au module des requêtes HTTP à travers VirtualHub.

Pour expérimenter cette fonctionnalité, utilisez le lien **Open API Browser** disponible en bas de la fenêtre d'interface de votre module, à l'aide d'un navigateur Web.

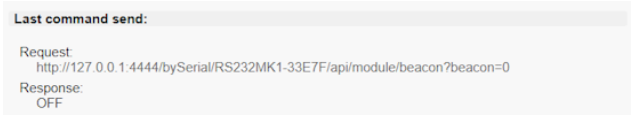
Lien pour ouvrir l'interface REST

Dans la fenêtre qui s'ouvre, vous pouvez alors modifier chaque attribut du module, à l'aide du bouton **edit**, puis en appliquant votre changement avec le bouton **apply**:



Modification d'un attribut manuellement

Après avoir effectué un changement, si vous descendez tout en bas de la page, vous verrez la requête HTTP qui a été effectuée pour appliquer le changement demandé:



Requête HTTP correspondante

Vous pourrez ainsi facilement découvrir comment accéder aux fonctions essentielles des modules par des requêtes HTTP: c'est là tout l'intérêt d'une interface REST. Si la sémantique d'un attribut en particulier vous échappe, vous trouverez des explications dans le manuel du module.

4.3. Passerelle OpenMetrics (Prometheus)

Il est aussi possible d'utiliser VirtualHub comme source de données pour un serveur Prometheus, afin de centraliser les mesures des capteurs Yoctopuce dans la même base de donnée que les informations sur l'état des infrastructures informatiques.

Les sources de données Prometheus sont appelées des exportateurs: l'idée est que chaque système ou service important peut mettre à disposition de Prometheus ses données vitales, permettant à l'administrateur système de les collecter et de les surveiller. Chaque exportateur est une URL accessible par HTTP, qui fournit ses données selon le standard OpenMetrics: une mesure par ligne, avec des conventions de nommage et de classification qui permettent à l'administrateur système de s'y retrouver parmi les centaines de mesures qu'il aura à disposition lorsqu'il configure son tableau de bord.

VirtualHub a la capacité d'être un exportateur OpenMetrics par l'intermédiaire de l'interface REST. Pour obtenir des données dans le format OpenMetrics, il suffit de charger l'URL `/api/services.om` de votre VirtualHub, soit `http://127.0.0.1:4444/api/services.om`. Par exemple, si vous accéder à cette URL alors que quelques capteurs sont connectés en local, vous obtiendrez quelque chose du genre (la présentation a été ici modifiée pour faciliter la lecture, mais en réalité chaque mesure tient sur une seule ligne):

```
yocto_temperature_advertisedValue{
    productName="Yocto-Thermocouple",
    serialNumber="THRMCPL1-16397A",
    deviceName="insideProbes",
    functionId="temperature1"} 21.78
yocto_temperature_advertisedValue{
    productName="Yocto-PT100",
    serialNumber="PT100MK1-BA496",
    functionId="temperature"} 28.57
yocto_temperature_advertisedValue{
    productName="Yocto-RangeFinder",
    serialNumber="YRNGFND1-1D1567",
    deviceName="rf",
    functionId="temperature1",
    functionName="rfTemp"} 25.13
yocto_lightSensor_advertisedValue{
    productName="Yocto-RangeFinder",
    serialNumber="YRNGFND1-1D1567",
    deviceName="rf",
    functionId="lightSensor1"} 56
yocto_rangeFinder_advertisedValue{
    productName="Yocto-RangeFinder",
    serialNumber="YRNGFND1-1D1567",
    deviceName="rf",
    functionId="rangeFinder1"} 1456
# EOF
```

Pour indiquer à Prometheus de collecter ces données directement depuis votre VirtualHub, il suffit donc d'ajouter à votre fichier `prometheus.yml` une section comme celle-ci:

```
  - job_name: "yoctohub_sensors"
    scrape_interval: 60s
    metrics_path: "/api/services.om"
    static_configs:
      - targets: ['127.0.0.1:4444']
```

Comme l'exportateur OpenMetrics est intégré au coeur de l'interface REST de VirtualHub, vous pouvez aussi l'utiliser pour obtenir des informations plus détaillées par capteur, en utilisant une URL qui pointe vers un capteur spécifique. Par exemple, si vous connectez un Yocto-Thermocouple au VirtualHub et que vous lui donnez le nom logique `tcProbes`, l'URL `/byName/tcProbes/api.om` donnera une réponse du genre:

```
yocto_module_luminosity{...,functionId="module",functionName="tcProbes"} 50
yocto_module_beacon{...,functionId="module",functionName="tcProbes"} 0
yocto_module_usbCurrent_mA{...,functionId="module",functionName="tcProbes"} 23
yocto_module_rebootCountdown{...,functionId="module",functionName="tcProbes"} 0
yocto_module_userVar{...,functionId="module",functionName="tcProbes"} 0
yocto_temperature_currentValue_degC{[...],functionName="heatSink"} 21.99
yocto_temperature_lowestValue_degC{[...],functionName="heatSink"} 20.51
yocto_temperature_highestValue_degC{[...],functionName="heatSink"} 22.25
yocto_temperature_currentRawValue_degC{[...],functionName="heatSink"} 21.988
yocto_temperature_signalValue_mV{[...],functionName="heatSink"} -0.162
yocto_temperature_signalValue_mV{[...],functionId="temperature2"} 999.999
yocto_dataLogger_currentRunIndex{[...],functionId="dataLogger"} 0
```

```
yocto_dataLogger_autoStart{[...],functionId="dataLogger"} 0
yocto_dataLogger_beaconDriven{[...],functionId="dataLogger"} 0
yocto_dataLogger_usage{[...],functionId="dataLogger"} 0
# EOF
```

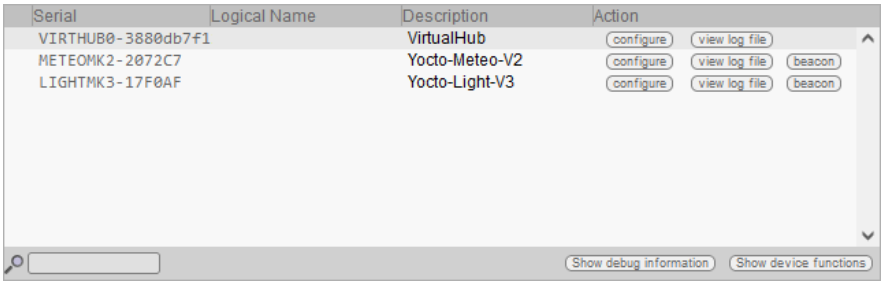
Ainsi, tous les attributs numériques du Yocto-Thermocouple sont mis à disposition. Vous pouvez donc connaître les valeurs min/max rencontrées, la tension mesurée aux bornes du thermocouple, etc. Notez aussi que dans ce cas, le symbole exporté inclut l'unité, comme recommandé par OpenMetrics. Lorsque le module détecte qu'une entrée n'est pas connectée (comme la fonction `temperature2` ci-dessus), les métriques qui ne peuvent être calculées sont automatiquement supprimées pour qu'elles soient manquantes, plutôt que de garder la dernière valeur mesurée.

Pour obtenir ces données supplémentaire par capteur, il suffit donc d'ajouter au fichier `prometheus.yml` une section supplémentaire, référençant le capteur soit par son numéro de série (*bySerial*) soit par son nom logique (*byName*):

```
- job_name: "thermocouple_probes"
  scrape_interval: 60s
  metrics_path: "/byName/tcProbes/api.om"
  static_configs:
    - targets: ['127.0.0.1:4444']
```

5. Contrôle d'accès

VirtualHub vous permet d'instaurer un contrôle d'accès à vos modules Yoctopuce. Pour ce faire cliquez simplement sur le bouton **Configure** de la ligne du VirtualHub dans l'interface.



*Cliquez sur le bouton **configure** de la première ligne*

Cela aura pour effet de faire apparaître la fenêtre de configuration de VirtualHub.

La fenêtre de configuration de VirtualHub

Ce contrôle d'accès est contrôlé depuis la section **Incoming connections**. Il peut se faire à deux niveaux distincts.

5.1. Accès "admin"

Le mot de passe *admin* verrouille les accès en écriture sur les modules. Lorsqu'il est configuré, seuls les accès de type *admin* permettent d'accéder aux modules en lecture et en écriture. Les utilisateurs utilisant le login *admin* pourront éditer la configuration des modules vus par VirtualHub comme ils le souhaitent.

5.2. Accès "user"

Le mot de passe *user* verrouille toute utilisation des modules. Lorsqu'il est configuré, toute utilisation sans mot de passe devient impossible.

Si vous configurez uniquement un mot de passe *user* sans configurer de mot de passe *admin*, tous les utilisateurs devront donner un mot de passe pour accéder aux modules, mais une fois autorisés, ils pourront aussi éditer la configuration des modules.

Si vous configurez simultanément un contrôle d'accès de type *user* et de type *admin*, les utilisateurs utilisant le login *user* ne pourront pas modifier la configuration des modules vus par VirtualHub. Les accès de type *user* ne permetteront d'accéder aux modules qu'en lecture seule, c'est-à-dire seulement pour consulter l'état des modules. Seuls les utilisateurs qui utilisent le login *admin* pourront changer la configuration des modules.

Si vous configurez uniquement un accès *admin* sans configurer d'accès *user*, tous les utilisateurs pourront continuer à consulter vos modules en lecture sans avoir à entrer de mot de passe, et seuls ceux qui connaîtront le mote de passe *admin* pourront changer la configuration des modules.

5.3. Influence sur les API

Attention, le contrôle d'accès agira aussi sur les API Yoctopuce qui tenteront de se connecter à VirtualHub. Dans les API Yoctopuce, la gestion des droits d'accès est réalisée au niveau de l'appel à la fonction `RegisterHub()` : vous devrez donner l'adresse de VirtualHub sous la forme *login:password@adresse:port*, par exemple:

```
YAPI.RegisterHub("admin:mypass@192.168.1.2:4444",errmsg);
```

Si vous perdez le mot passe de VirtualHub, vous pouvez le remettre à zéro en effaçant son fichier de configuration (`.virtualhub.dat`)

6. Envoi de données vers l'extérieur

VirtualHub est capable de se connecter à des services externes pour communiquer l'état des modules qui lui sont raccordés.

VirtualHub sait comment poster ses données au format accepté par quelques services Cloud tiers, tels que

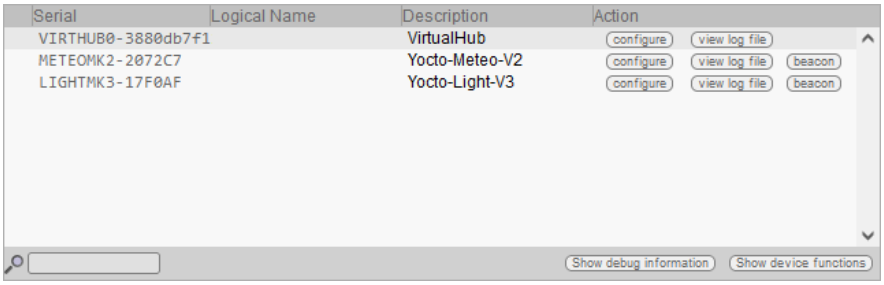
- Emoncms
- InfluxDB (versions 1.0 et 2.0)
- PRTG
- Valarm.net

VirtualHub peut aussi se connecter à des services externes à l'aide de protocoles avancés qui permettent une interaction plus poussée avec les modules Yoctopuce, mais qui vous demanderont un peu plus de connaissances pour pouvoir en tirer parti:

- MQTT
- Yocto-API

6.1. Configuration

Pour utiliser cette fonctionnalité, cliquez simplement sur le bouton **configure** de la ligne correspondant à VirtualHub dans l'interface, puis cliquez sur le bouton **edit** de la section **Outgoing callbacks**.



*Cliquez sur le bouton **configure** correspondant*

Puis éditez la section Outgoing callbacks

La fenêtre de configuration des callbacks apparaît. Cette fenêtre vous permet de définir comment VirtualHub peut interagir avec un serveur web externe. Vous avez plusieurs type d'interactions à votre disposition.

6.2. Callbacks HTTP vers des services tiers

VirtualHub est capable de poster sur des serveurs externes les valeurs des capteurs Yoctopuce à intervalles régulier et/ou à chaque fois qu'une valeur change de manière significative. Cette fonctionnalité vous permettra de stocker vos mesures et de tracer des graphiques sans écrire la moindre ligne de code.

Yoctopuce n'est en aucune manière affilié à ces services tiers et ne peut donc ni garantir leur pérennité, ni proposer des améliorations à ces services.

Emoncms

Emoncms est un service de Cloud open-source qui permet de poster les données des capteurs Yoctopuce et ensuite de les visualiser. Il est aussi possible d'installer son propre serveur en local.

Les paramètres à fournir sont la clé d'API Emoncms, le numéro de nœud que vous désirez utiliser, ainsi que l'adresse du serveur Emoncms si vous utilisez un serveur local.

Il est possible de personaliser les noms associés aux mesures postées sur Emoncms. Pour plus de détails, voir le paragraphe intitulé "Noms associés aux valeur postées" ci-dessous.

InfluxDB 1.0 and 2.0

InfluxDB est une base de données open-source dédiée spécifiquement à stocker des séries temporelles de mesures et d'événements. Notez que seules les installations locales sont supportées. En effet, le service *InfluxDB Cloud* n'est pas supporté car il nécessite une connexion SSL.

Les paramètres pour la version 1.0 d'InfluxDB sont l'adresse du serveur et le nom de la base de données.

La version 2.0 d'InfluxDB utilise une API différente et le YoctoHub a besoin de trois paramètres (`organization`, `bucket` et `token`) ainsi que l'adresse du serveur.

Il est possible de personaliser les noms associés aux mesures postées sur InfluxDB. Pour plus de détail, voir le paragraphe intitulé "Noms associés aux valeur postées" ci-dessous.

PRTG

PRTG est une solution commerciale, destinée à la supervision des systèmes et des applications. Il est possible d'enregistrer les mesures et obtenir des graphiques de vos capteurs avec ce service.

Les paramètres à fournir sont l'adresse du serveur PRTG et le `token` qui permet d'identifier VirtualHub.

Il est possible de personaliser les noms associés aux mesures postées sur PRTG. Pour plus de détail, voir le paragraphe intitulé "Noms associés aux valeur postées" ci-dessous.

Valarm.net

Valarm est un service de Cloud professionnel qui permet d'enregistrer les données des capteurs Yoctopuce mais permet aussi des fonctions plus élaborées comme la possibilité de géolocaliser les mesures ou de configurer les modules Yoctopuce à distance.

Le seul paramètre à fournir est un `Routing code` qui permet d'identifier VirtualHub.

6.3. Callbacks vers un broker MQTT

MQTT est un protocole de l'Internet des Objets permettant à des capteurs et des actuateurs de communiquer entre eux, via un serveur central appelé *broker MQTT*. MQTT est particulièrement utilisé en domotique, où il permet de fédérer de nombreuses technologies pour les rendre accessible à un système de contrôle central comme *Home Assistant*.

Les paramètres de base à fournir pour la configuration du callback MQTT sont l'adresse du broker MQTT, le client ID, le *root_topic* ainsi que les paramètres d'authentification. Notez que l'encapsulation du protocole MQTT dans une connexion SSL n'est pas supportée, ce qui exclut son utilisation avec les services comme AWS IoT Core.

Lorsqu'un callback MQTT est actif, VirtualHub est capable de publier des messages avec l'état des capteurs et actuateurs, et recevoir des messages de commande et de configuration, ce qui permet au système de contrôle central d'interagir pleinement avec les modules.

Le reste de cette section décrit en détail les messages MQTT supportés. Elle n'intéressera que les développeurs qui désirent développer leur propre intégration avec des modules Yoctopuce via MQTT. Si vous comptez simplement utiliser *Home Assistant*, vous pouvez sauter cette section et grâce au mécanisme *MQTT Discovery*, vos modules devraient automatiquement apparaître dans *Home Assistant*.

Racine commune des messages

Le *topic* de tous les messages commence par une partie commune, qui identifie le module Yoctopuce et la fonction particulière de ce module concernée par le message. Elle a la structure suivante:

root_topic*/*deviceName*/*functionName

Le *root_topic* peut être configuré librement, par exemple à la valeur `yoctopuce`. Si vous connectez plusieurs hubs au même *broker* MQTT, vous pouvez soit utiliser le même *root_topic* pour tous, soit un topic différent par hub. L'utilisation d'un *root_topic* distinct est recommandée si le hub est destiné à recevoir beaucoup de commandes par MQTT.

Le *deviceName* correspond au nom logique que vous avez donné au module Yoctopuce concerné. Si aucun nom logique n'a été configuré, le numéro de série du module est utilisé à la place du nom logique (par exemple `METEOMK2-012345`).

Le *functionName* correspond au nom logique que vous avez donné à la fonction concernée. Si aucun nom logique n'a été configuré, l'identifiant de la fonction est utilisé (par exemple `genericSensor1`).

Le fait d'utiliser les noms logique plutôt que les noms matériels dans la racine du *topic* a l'avantage de permettre d'identifier les modules et les fonctions par leur rôle et de ne pas devoir indiquer explicitement l'identifiant matériel de chaque module au client MQTT qui devra interagir avec ces modules. Le désavantage est que si vous décidez de changer le nom logique de vos modules ou de vos fonctions sans y penser, les *topics* MQTT utilisés changeront en conséquence.

Topic /api: état complet de la fonction

Sous `root_topic/deviceName/functionName/api`, chaque fonction publie une structure JSON décrivant l'état complet de la fonction, tous attributs compris. Dans cet encodage JSON,

- les booléens sont représentés par 0 et 1
- les types énumérés sont représentés par des constantes numériques
- les nombres réels tels que les mesures sont représentés sous forme d'entiers, après multiplication par 65536. Il faut donc les diviser par 65536.0 pour obtenir la valeur réelle.

Ce message est publié lorsque l'une des conditions suivante se produit:

- à l'établissement de la connexion du hub avec le broker MQTT
- toutes les cinq minutes
- après un changement de configuration du module
- en réponse à la réception d'une commande par cette fonction
- pour la fonction *module*, en cas d'activation ou de désactivation de la balise (*beacon*)

Topic de base: état instantané

Sous `root_topic/deviceName/functionName`, chaque fonction publie un résumé textuel de son état. Il ne s'agit pas de JSON mais d'une simple chaîne de caractères, correspondant à la valeur de l'attribut *advertisedValue* de la fonction. Par exemple, pour un capteur, il correspond à la valeur instantanée du capteur, alors que pour un relais il correspond à la lettre A ou B en fonction de l'état de commutation.

Ce message est publié lorsque l'une des conditions suivante se produit:

- à l'établissement de la connexion du hub avec le broker MQTT
- toutes les cinq minutes
- à chaque changement de l'attribut *advertisedValue*

Pour éviter de surcharger le broker MQTT avec les changements de valeurs instantanée des capteurs, il est possible de désactiver globalement l'envoi des messages de valeurs instantanée pour les capteurs uniquement, dans la configuration MQTT.

Topics /avg, /min, /max: valeurs moyenne et extrêmes

Sous `root_topic/deviceName/functionName/avg`, les fonctions de type capteur (sous-classes de *Sensor*) publient périodiquement la valeur moyenne observée durant l'intervalle de temps précédent, directement sous forme de nombre réel.

Sous `root_topic/deviceName/functionName/min`, la valeur minimale observée durant l'intervalle de temps précédent.

Sous `root_topic/deviceName/functionName/max`, la valeur maximale observée durant l'intervalle de temps précédent.

Ces messages sont l'équivalent direct des *timed reports* documentés dans le manuel de ces module. L'intervalle de temps doit avoir été configuré préalablement dans l'attribut *reportFrequency*. Dans le cas contraire, ces messages ne sont pas envoyés.

Topics /set/*attributeName*: envoi de commande et configuration

Sous `root_topic/deviceName/functionName/set/`*attributeName*, il est possible d'envoyer des message pour modifier les attributs des fonctions, dans le but de modifier leur état ou leur configuration. La valeur du message correspond à la nouvelle valeur désirée, telle quelle. Le

format est identique à celui utilisé par la asserelle REST de VirtualHub (voir la section "Passerelle REST" de ce manuel).

Par exemple, on pourrait commuter un relais en envoyant un message au *topic*
`yoctopuce/RelaiPompe/relay1/set/state`
avec la valeur 1.

On pourrait aussi déclencher une impulsion de 1500ms sur le même relais en envoyant un message au *topic*
`yoctopuce/RelaiPompe/relay1/set/pulseTimer`
avec la valeur 1500.

La réception de commande et de changements de configuration par MQTT doit avoir été activée explicitement dans la configuration MQTT sur le hub Yoctopuce. Par sécurité, le comportement de base du mode MQTT reste le mode en lecture seule.

Topic /rdy: état de connectivité

Sous `root_topic/deviceName/module/rdy`, la fonction module publie une indication binaire de l'état de disponibilité du module. La valeur est à 1 lorsque le module est en ligne, et à 0 lorsqu'il est hors ligne.

Ce message est publié par le hub pour son propre module lorsque l'une des conditions suivante se produit:

- à l'établissement de la connexion du hub avec le broker MQTT
- à la déconnexion du hub du broker MQTT

Ce message est publié pour les modules autres que le hub lorsque l'une des conditions suivante se produit:

- à l'établissement de la connexion du hub avec le broker MQTT
- au branchement d'un module sur le hub
- au débranchement d'un module du hub

Pour déterminer si un module est réellement atteignable, il faut donc vérifier son propre topic `/rdy`, pour savoir si il a été déconnecté, et le topic `/rdy` du hub, pour savoir si la connexion MQTT est active.

MQTT discovery

De plus, des messages particuliers sont publiés sous le topic `homeassistant/` juste après l'établissement de la connexion du hub avec le broker MQTT, et répétés toutes les 5 minutes, pour permettre la détection automatique des fonctionalités offertes, grace au mécanisme *MQTT discovery* supporté par Home Assistant et openHab.

6.4. Callbacks de type Yocto-API

Les callbacks de type Yocto-API utilisent un protocole spécifique défini par Yoctopuce, qui permet une interaction très poussée avec les modules Yoctopuce. A l'aide de certains langages commes PHP, TypeScript, JavaScript ou Java, ils permettent au programmeur du service Web d'utiliser directement les fonctions de la librairie de programmation Yoctopuce pour interagir avec les modules qui se connectent par callback HTTP. Cela permet en particulier de contrôler depuis un site web public des modules Yoctopuce installés derrière un router ADSL privé. Il est par exemple possible de commuter la sortie d'un relais en fonction de la valeur d'un capteur, tout en gardant le contrôle complet du système sur un serveur Web.

Yoctopuce met à disposition une application gratuite qui exploite au maximum les possibilités du Callback Yocto-API sur un serveur PHP: VirtualHub for Web. Cette application web permet

d'interagir à distance avec les modules qui se connectent périodiquement via un Callback Yocto-API. De plus amples informations sur VirtualHub for Web sont disponibles sur le blog de Yoctopuce [1].

En mode Callback Yocto-API ou Yocto-API-JZON, il est possible de choisir entre les protocoles "HTTP" et "WebSocket".

Callbacks Yocto-API en mode WebSocket

Lors d'une requête HTTP usuelle, le flux d'information est extrêmement simple: le client envoie une requête et écoute la réponse du serveur. Il ne s'agit pas à proprement parler d'une conversation, mais simplement d'une réponse à une question. Le client HTTP ne peut pas répondre à ce que le serveur lui a dit sans recommencer une nouvelle communication séparée.

Il y a néanmoins dans le standard HTTP 1.1 une porte ouverte vers une amélioration: le client peut demander d'upgrader le protocole de communication. Une méthode d'upgrade qui s'est standardisée s'appelle les WebSockets et est définie dans le RFC 6455. Cette upgrade transforme le simple canal question/réponse en un lien bidirectionnel permettant d'échanger des messages quelconques dans les deux directions.

Cette transformation de la connection HTTP exige que les deux parties en soient capables. VirtualHub et les librairies de programmation Yoctopuce le sont. Mais pour pouvoir transformer un callback HTTP en callback WebSocket, vous aurez aussi besoin d'un serveur Web basé sur une technologie qui permet l'upgrade de connection. C'est le cas par exemple de Java et Node.JS. Par contre, les implémentations de PHP sur Apache n'en sont à ce jour pas capables.

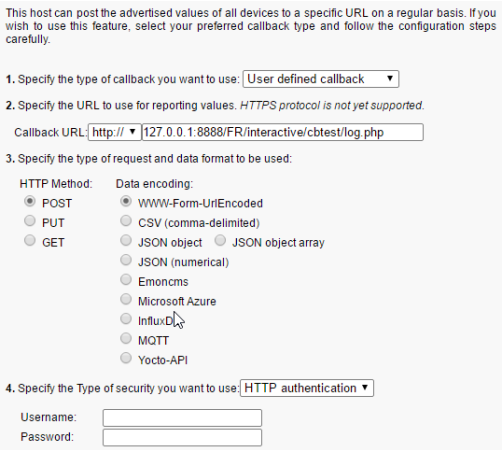
L'utilisation de WebSockets permet d'accéder à plusieurs fonctionnalités avancées des librairies Yoctopuce qui ne sont pas disponibles via un callback HTTP:

- un callback WebSocket peut énumérer et récupérer des données stockées dans l'enregistreur de données intégré dans chaque senseur Yoctopuce.
- un callback WebSocket peut utiliser les fonctions de communication bidirectionnelles des modules séries, par exemple pour exécuter des requêtes MODBUS avec un Yocto-RS485.
- un callback WebSocket peut profiter des notifications instantanées de changement de valeur par callback, et des notifications périodiques de valeurs moyennées des capteurs.
- un callback WebSocket peut garder une connection persistante entre le hub et le serveur, par exemple pour implémenter une interaction avec un utilisateur.
- un callback WebSocket peut même être utilisé pour effectuer une mise à jours de firmware.

6.5. Callbacks HTTP définis par l'utilisateur

Si aucune des autres options proposées pour la configuration de callback HTTP ne convient à vos besoins, vous pouvez essayer de spécifier vous-même la manière dont les données doivent être transmises. Les "*User defined callback*" vous permettent de personnaliser la manière dont VirtualHub envoie les informations au serveur. Notez que seul le protocole HTTP est supporté (pas de HTTPS).

[1] https://www.yoctopuce.com/FR/article/nouveau-un-virtualhub-qui-fonctionne-a-travers-le-web



La fenêtre de configurations des callbacks

Si vous désirez protéger votre script de callback, vous pouvez configurer un contrôle d'accès HTTP standard sur le serveur Web. VirtualHub sait comment gérer les méthodes standard d'identification de HTTP: indiquez simplement le nom d'utilisateur et le mot de passe nécessaires pour accéder à la page. Il est possible d'utiliser la méthode "Basic" aussi bien que la méthode "Digest", mais il est recommandé d'utiliser la méthode "Digest", car elle est basée sur un protocole de question-réponse qui évite la transmission du mot de passe sur le réseau et évite aussi les copies d'autorisation.

A titre d'exemple, voici un script PHP qui vous permettra de visualiser dans la fenêtre de debug le contenu des données postées par un callback HTTP défini par l'utilisateur en mode POST et WWW-Form-UrlEncoded.

```
<?php
  Print(Date('H:i:s')."\r\n");
  foreach ($_POST as $key => $value) {
      Print("$key=$value\r\n");
  }
?>
```

Il est possible de personaliser les noms associés aux mesures postées par un callback HTTP défini par l'utilisateur. Pour plus de détail, voir le paragraphe intitulé "Noms associés aux valeur postées" ci-dessous.

6.6. Noms associés aux valeur postées

A l'exception des callbacks de type Yocto-API qui donnent accès à la totalité des informations sur les modules Yoctopuce, les callbacks HTTP sont conçus pour ne transmettres que les informations les plus importantes au serveur, en associant chaque valeur avec un nom qui permette facilement de le rattacher à son origine.

Comportement de base

Le comportement standard est de transmettre la valeur de l'attribut `advertisedValue` pour chaque fonction présente sur les modules Yoctopuce. Le nom associé automatiquement à chaque valeur suit la logique suivante:

1. Si un nom logique a été défini pour une fonction:

```
NOM_LOGIQUE_DE_LA_FONCTION = VALEUR
```

2. Si un nom logique a été défini pour le module, mais pas pour la fonction:

```
NOM_DU_MODULE.NOM_HARDWARE = VALUE
```

3. Si aucun nom logique n'a été attribué:

```
NUMERO_DE_SERIE.NOM_HARDWARE = VALEUR
```

La manière la plus simple pour personnaliser les noms associés aux valeurs consiste donc à configurer le nom désiré comme nom logique de la fonction, ou sinon comme nom logique du module lui-même.

Voici un exemple des données postées par un callback HTTP défini par l'utilisateur en mode POST et au format *JSON (numerical)* pour un système comportant un Yocto-Watt où chaque fonction a reçu un nom logique explicite (par exemple `VoltageDC`) et un Yocto-Meteo-V2 où c'est au module lui-même qu'on a donné le nom logique `Ambiant`:

```
{"timestamp":1678276738,
"CurrentAC":0
,"CurrentDC":0
,"VoltageAC":0
,"VoltageDC":0
,"Power":0
,"Ambiant.temperature":22.17
,"Ambiant.pressure":949.36
,"Ambiant.humidity":30
}
```

Personnalisation avancée par un fichier

Si l'on désire une personnalisation plus poussée du format des données transmises, pour sélectionner spécifiquement quelle attribut de quel module doit être envoyé sous quel nom, ou pour y rajouter des informations contextuelles, c'est aussi possible, mais c'est un peu plus compliqué. Il faut à ce moment créer un fichier modèle définissant le format exact des données à envoyer. Le contenu et le nom de ce fichier est donc spécifique à chaque type de callback HTTP, et chaque cas sera expliqué individuellement ci-dessous.

Le point commun de tous les fichiers de modèle est que leur contenu sera envoyé tel quel au serveur, à l'exception des expressions englobées entre *accents graves* (le caractère `, code ASCII 96, appelé *backquote* ou *backtick* en anglais) qui seront évaluées par la passerelle REST de VirtualHub (voir la section "Passerelle REST" de ce manuel). Par exemple, si le fichier de modèle comporte le texte:

```
{ "origin": "`/api/module/productName`" }
```

alors le contenu effectivement posté sera

```
{ "origin": "VirtualHub" }
```

Fichier de personnalisation pour Emoncms

Le format de base utilisé par VirtualHub pour Emoncms a la forme ci-dessous. Notez que les données sont transmise dans l'URL, donc en une seule ligne, mais elles ont été mises ici sur plusieurs ligne pour faciliter la lecture.

```
time=1678277614
&json={
  "CurrentAC":0,
  "CurrentDC":0,
  "VoltageAC":0,
  "VoltageDC":0,
  "Power":0,
  "Ambiant.temperature":22.28,
  "Ambiant.pressure":949.54,
  "Ambiant.humidity":29.7
}
```

Pour personnaliser le format des données envoyées à Emoncms, il faut créer sur VirtualHub un fichier modèle de format portant le nom `EMONCMS_cb.fmt`.

Ce fichier ne doit comporter qu'une seule ligne, sans aucun retour de chariot, et commencer par une chaîne du type `&json=`. Par exemple, pour ne poster que l'humidité absolue et relative, vous pourriez utiliser (sans retour de chariot!):

```
&json={
  "absoluteHumidity"=`/byName/Ambiant/api/humidity/absHum`,
  "relativeHumidity"=`/byName/Ambiant/api/humidity/relHum`
}
```

Fichier de personnalisation pour InfluxDB

Le format de base utilisé par VirtualHub pour InfluxDB a la forme ci-dessous. Il associe toutes les valeurs à une base de mesures *yoctopuce* et ajoute un tag *name* avec le nom sur le réseau de VirtualHub et un tag *ip* avec son adresse IP. Ensuite, chaque valeur est postée dans un champ dont le nom suit la convention de base décrite précédemment. Les données sont transmises par un POST de type CSV, en une seule ligne, mais elles ont été mises ici sur plusieurs lignes pour faciliter la lecture.

```
yoctopuce,name=VIRTHUB0-12345678,ip=192.168.1.10
  CurrentAC=0,CurrentDC=0,VoltageAC=0,VoltageDC=0,Power=0,
  Ambiant_temperature=22.5,Ambiant_pressure=948.63,Ambiant_humidity=29.4
  1678281649
```

Pour personnaliser le format des données envoyées à InfluxDB, il faut créer sur VirtualHub un fichier modèle de format portant le nom `INFLUXDB_cb.fmt` (pour la version 1.0), ou `INFLUXDB_V2_cb.fmt` (pour la version 2.0).

Ce fichier peut comporter plusieurs lignes si vous le désirez, ce qui vous permettre d'utiliser des tags différents pour différents mesures, ou même de ventiler des mesures sur plusieurs bases de données.

Par exemple, pour poster l'humidité absolue et relative simultanément, mais avec tes tags différents, vous pourriez utiliser le fichier de format suivant:

```
humidity,type=relative,location=Library relHum=`/byName/Ambiant/api/humidity/relHum`
humidity,type=absolute,location=Library absHum=`/byName/Ambiant/api/humidity/absHum`
```

Attention: le serveur InfluxDB n'accepte que les retours de chariot au format UNIX (caractère `\n`, aussi appelé LF). Si vous éditez le fichier sur une machine Windows, prenez soin d'utiliser un éditeur de texte capable de ne pas ajouter le retour de chariot Windows (`\r\n`, aussi appelé CR LF).

Fichier de personnalisation pour PRTG

Le format de base utilisé par VirtualHub pour PRTG a la forme ci-dessous. Il poste chaque valeur dans un canal dont le nom suit la convention de base décrite plus procédemment. Les données sont transmises par un POST de type CSV, en une seule ligne, mais elles ont été mises ici sur plusieurs lignes pour faciliter la lecture.

```
{"prtg":{"result":[
{"channel":"CurrentAC","value":"0","float":"1","DecimalMode":"All"}
,{"channel":"CurrentDC","value":"0","float":"1","DecimalMode":"All"}
,{"channel":"VoltageAC","value":"0","float":"1","DecimalMode":"All"}
,{"channel":"VoltageDC","value":"0","float":"1","DecimalMode":"All"}
,{"channel":"Power","value":"0","float":"1","DecimalMode":"All"}
,{"channel":"Ambiant.temperature","value":"22.48","float":"1","DecimalMode":"All"}
,{"channel":"Ambiant.pressure","value":"948.68","float":"1","DecimalMode":"All"}
,{"channel":"Ambiant.humidity","value":"29.7","float":"1","DecimalMode":"All"}
]}}
```

Pour personnaliser le format des données envoyées à PRTG, il faut créer sur VirtualHub un fichier modèle de format portant le nom `PRTG_cb.fmt`.

Ce fichier doit comporter au minimum la même première et dernière ligne que l'exemple ci-dessus. La description des canaux pourra par contre être entièrement personnalisée.

Par exemple, pour poster l'humidité absolue et relative simultanément, dans deux canaux séparés, vous pourriez utiliser le fichier de format suivant:

```
{"prtg":{"result":[
{"channel":"relHum","value":"`/byName/Ambiant/api/humidity/relHum`",
  "float":"1","DecimalMode":"All"},
{"channel":"absHum","value":"`/byName/Ambiant/api/humidity/absHum`",
  "float":"1","DecimalMode":"All"}
]}}
```

6.7. Planification des callbacks

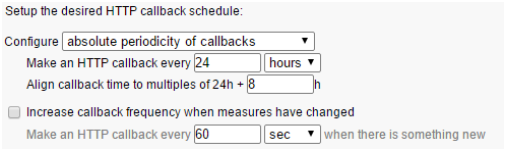
La section de plannification des callbacks est présente pour tous les types de callbacks. C'est la dernière section contenant des champs à remplir.

Un premier callback est toujours effectué quelques secondes après le lancement de VirtualHub. Pour les callbacks suivants, c'est le réglage de planification qui détermine la fréquence des callbacks.

Il est possible de choisir entre deux méthodes de planification: soit en configurant l'intervalle de temps entre deux callbacks consécutifs, soit en définissant une périodicité absolue, pour obtenir des callbacks à heure fixe. L'intervalle entre les callbacks peut être spécifié en secondes, en minutes ou en heures.

L'option `interval between subsequent callbacks` permet de spécifier le délais entre chaque callback. C'est-à-dire que si l'on configure un intervalle de 5 minutes, VirtualHub va attendre 5 minutes avant de déclencher le callback suivant. Si le premier callback est déclenché à 12h03, le suivant sera exécuté à 12h08, etc.

L'option `absolute periodicity of callbacks` permet de configurer des callbacks à heure fixe. C'est-à-dire que le callback est déclenché tous les multiples du délais configuré. Par exemple un délais de 5 minutes va déclencher un callback à 8h00, 8h05, 8h10, etc. Notez que dans ce mode il est aussi possible de spécifier un décalage par rapport au délais configuré. Par exemple avec un délais de 24h, il est possible d'utiliser un décalage de 8h pour déclencher le callback tous les jours à 8h du matin.



Planification des callbacks

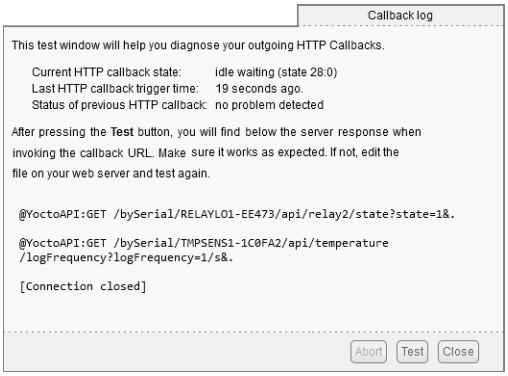
Vous pouvez choisir explicitement si vous désirez que la fréquence des callbacks varie lorsqu'aucune nouvelle mesure n'est détectée. Cela permet de choisir la fréquence minimale de transmission pour réduire la quantité de données transmises sur le réseau si rien ne se passe.

Attention, si vous configurez de nombreux hubs pour effectuer le callback à la même heure, vous allez générer un pic de charge sur votre serveur web. Il est donc souhaitable d'utiliser le paramètre décalage pour équilibrer la charge.

6.8. Tests

Afin de vous permettre de débugger le processus, VirtualHub vous permet de visualiser la réponse au callback envoyé par le serveur web. Dans la fenêtre de configuration des callbacks, cliquez sur le bouton **test** une fois que vous avez renseigné tous les champs pour ouvrir la fenêtre de tests.

La fenêtre vous montre l'état actuel du système de callback, pour vous permettre de voir par exemple si un callback est actuellement en cours sur le serveur web. Dans tous les cas, tant que cette fenêtre est ouverte, aucun callback HTTP ne sera déclenché automatiquement. C'est en pressant le bouton **Test** que vous pourrez déclancher manuellement un callback. Une fois déclanché, la fenêtre vous montre les informations retournées par le service web, comme dans l'exemple ci-dessous:



Le résultat du test de callback avec un Yocto-PowerRelay et un Yocto-Temperature

Si le résultat vous paraît satisfaisant, fermez la fenêtre de debug, et cliquez sur **Ok**.

6.9. Connexions spontanées

En plus de connexions liées aux callbacks définis décrites dans les sections précédentes, VirtualHub va occasionnellement tenter d'établir des connexions vers l'extérieur. Ces connexions sont les suivantes:

- Installation de Yocto-Visualization (for web): Lorsque l'utilisateur déclenche l'installation de Yocto-Visualization (for web) depuis l'interface de VirtualHub, le hub va automatiquement télécharger le fichier d'installation le plus récent depuis www.yoctopuce.com
- Test de version: à chaque fois que la fenêtre de propriété ou de configuration d'un module est ouverte, VirtualHub effectue une requête sur www.yoctopuce.com pour vérifier si le firmware du module est à jour. Cette requête ne contient que le numéro de série du module et sert à indiquer à l'utilisateur si un nouveau firmware est disponible.
- Téléchargement de firmware: A chaque fois que la fenêtre de mise à jour de firmware est ouverte, VirtualHub effectue une requête sur www.yoctopuce.com pour y récupérer le firmware le plus plus récent. Cette connexion permet d'éviter à l'utilisateur d'avoir à télécharger manuellement le dernier firmware.

Notez que ces connexions sont en fait établies par le navigateur web qui affiche l'interface utilisateur de VirtualHub. De plus, ces connexions sont purement optionnelles, si elles ne peuvent pas être établies, l'application continuera à fonctionner normalement.

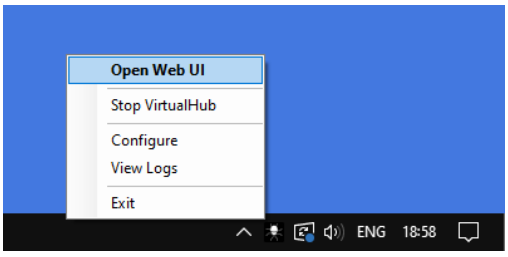
7. Compléments optionnels

7.1. L'utilitaire Windows VirtualHub Control

Cet utilitaire est automatiquement installé sous Windows lorsque vous utilisez la méthode d'installation par le fichier .msi. C'est une application qui reste dans la barre des tâches et qui permet de contrôler VirtualHub. Il permet d'effectuer les opérations suivantes:

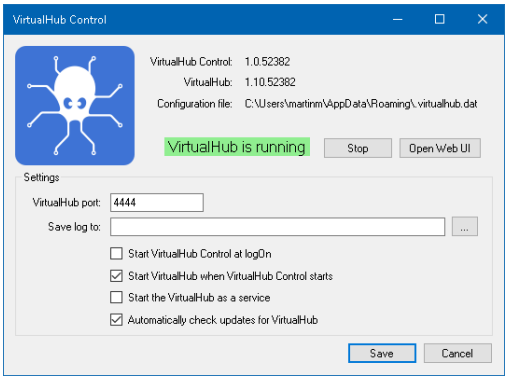
- Accéder à l'interface web de VirtualHub
- Démarrer/Stopper VirtualHub
- Obtenir les logs de VirtualHub
- Configurer VirtualHub

Un simple clic sur l'icône fait apparaître un menu avec les opérations disponibles, alors qu'un double-clic affiche l'interface web de VirtualHub dans le browser web par défaut.



Les opérations directement accessibles depuis la barre des tâches

Le menu **Configure** fait apparaître la fenêtre de configuration de VirtualHub. Dans cette fenêtre, vous trouverez quelques informations sur VirtualHub comme le numéro de version et l'emplacement des fichiers de configuration ou de logs.

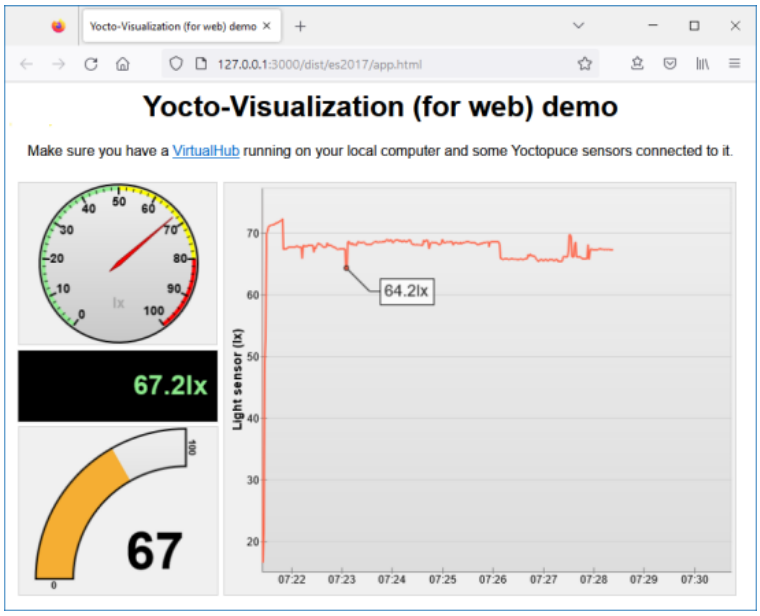


La fenêtre de configuration de VirtualHub

Cette fenêtre permet aussi de configurer les options de démarrage de VirtualHub. Il est possible de choisir le port utilisé par VirtualHub, ainsi que l'emplacement des fichiers de logs. Il est aussi possible de configurer VirtualHub pour qu'il fonctionne comme un service Windows. En fait, cette fenêtre permet de spécifier certains des paramètres qui auparavant étaient passés sur la ligne de commande.

7.2. Installation de Yocto-Visualization (for web)

Yocto-Visualization (for web) est une petite application Web qui permet de visualiser facilement les valeurs mesurées par des capteurs Yoctopuce.

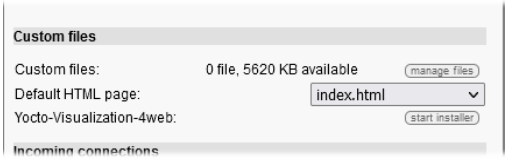


L'interface de Yocto-Visualization (for web)

Il est possible d'installer Yocto-Visualization (for web) comme plug-in de VirtualHub, directement via l'interface Web de configuration de VirtualHub. Pour cela, il faut qu'il ait été lancé avec l'option `-F`, c'est-à-dire que l'interface Files soit active et permette donc l'ajout de fichiers personnalisés dans VirtualHub. N'oubliez pas que cette option prend en argument le nom du fichier "container" au format tar dans lequel seront stockés les fichiers personalisés.

Pour installer Yocto-Visualization (for web) depuis l'interface Web de VirtualHub

1. Dans l'interface de VirtualHub, ouvrez la fenêtre de configuration de VirtualHub en cliquant sur **configure**.
2. Dans la section **Custom files**, en face de `Yocto-Visualization-4web`, cliquez sur **start installer**.
3. Il suffit ensuite de suivre les étapes en cliquant sur **Next** de manière répétée puis **OK**. Assurez-vous simplement que dans le champ **Hub address** le port ne soit pas présent deux fois.



*La section **Custom files** de la fenêtre de configuration de VirtualHub*

Pour utiliser Yocto-Visualization (for web), référez-vous aux articles du blog du site de Yoctopuce.

8. VirtualHub 2.0

Au moment de la rédaction (mars 2023) de ce mode d'emploi, il s'agit d'une version BETA. *VirtualHub 2.0* intègre le support HTTPS.

Malgré le changement de version majeure, le nouveau *VirtualHub 2.0* reste compatible avec tous les composants Yoctopuce. En fait, de l'extérieur la version 2.0 est identique à la version 1.0, mais en interne une large partie du code a été réécrite.

8.1. Connexions entrantes

La principale nouveauté est l'ajout du support SSL/TLS qui permet à VirtualHub d'accepter des connexions entrantes HTTPS. Dorénavant, VirtualHub accepte les connexions HTTP classiques sur le port 4444 mais aussi des connexions HTTPS sur le port 4443. Vous pouvez accéder au VirtualHub de manière sécurisée et chiffrée avec votre browser à l'aide de l'URL https://myhostname:4443.

Bien sûr, ces ports sont configurables à l'aide d'options à passer sur la ligne de commande:

- l'option `-p` permet de changer le port HTTP
- l'option `-P` permet de changer le port HTTPS
- l'option `--no_https` désactive complètement le support SSL/TLS
- l'option `--only_https` force l'utilisation d'HTTPS sur toutes les connexions entrantes

8.2. Connexions sortantes

VirtualHub 2.0 inclut aussi le support SSL/TLS pour les connexions sortantes. Il est donc possible de configurer un Outgoing HTTP callback sur un serveur qui requière une connexion HTTPS, comme par exemple InfluxDB OSS v2.0.

Pour utiliser cette option, lors de la configuration d'un callback sortant, ouvrez le menu déroulant permettant de sélectionner le type de connection et choisissez l'option **https://**.

8.3. IPv6

Par défaut, VirtualHub 2.0 fonctionne uniquement en IPv4 et donc ne répondra pas aux requêtes venant des interfaces IPv6. Si l'on veut utiliser IPv6, il faut lancer VirtualHub avec l'une des deux options suivantes: `--enable_ipv6` ou `--only_ipv6`. L'option `--enable_ipv6` active la pile réseau IPv6 mais garde la pile IPv4 activée. C'est-à-dire que VirtualHub utilisera les interfaces

réseau IPv6 et IPv4. L'option `--only_ipv6` quant à elle utilisera uniquement les interfaces réseau IPv6.

8.4. Systèmes d'exploitation supportés

A l'heure où cette documentation est rédigée, la version 2.0 de VirtualHub est une version beta. Seuls les systèmes d'exploitation et architectures suivants sont supportés:

- Windows Intel 32 et 64 bits
- Linux Intel 32 et 64 bits
- Linux ARM 64 bits
- macOS