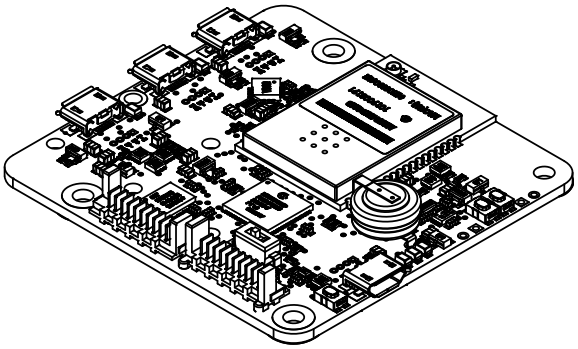
YoctoHub-Wireless, Mode d'emploi

Table des matières

1. Introduction

Le module YoctoHub-Wireless est un module de 60x58mm qui permet de contrôler d'autres modules Yoctopuce à travers une connection réseau sans fil. Vu de l'extérieur, ce module se comporte exactement comme un ordinateur classique faisant tourner un VirtualHub[1]: même interface, mêmes fonctionnalités.



Le YoctoHub-Wireless

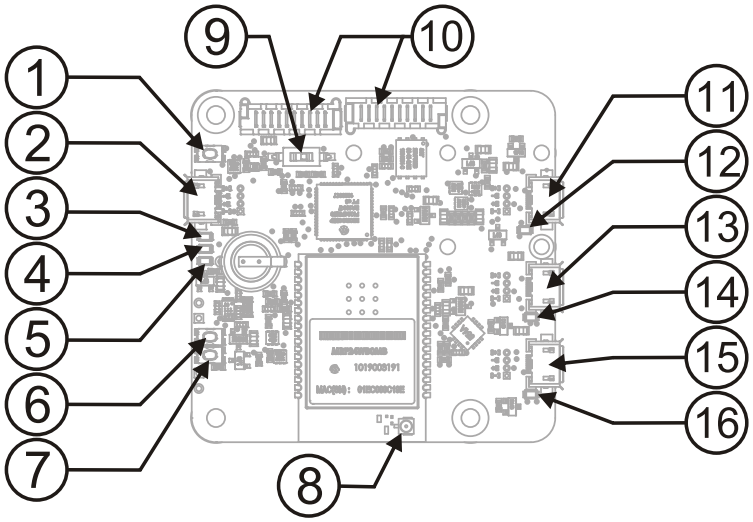
Le YoctoHub-Wireless a été conçu pour être déployé facilement et ne pas demander de maintenance particulière. Contrairement à un mini-PC, il n'utilise pas un système d'exploitation complexe. Quelques réglages simples permettent son utilisation dans toutes sortes d'environnements réseau. Ces réglages peuvent être effectués manuellement ou de manière automatisée, par USB. Il convient de ce fait beaucoup mieux à une industrialisation qu'un mini-PC. En revanche, il ne permet pas l'exécution de programmes supplémentaires écrits par l'utilisateur.

Le YoctoHub-Wireless n'est pas un hub USB standard avec accès réseau. Bien qu'utilisant du câblage USB, ses ports descendants utilisent un protocole propriétaire, plus simple qu'USB. Il n'est par conséquent pas possible de contrôler, ni même d'alimenter, des périphériques USB standards avec un YoctoHub-Wireless.

Yoctopuce vous remercie d'avoir fait l'acquisition de ce YoctoHub-Wireless et espère sincèrement qu'il vous donnera entière satisfaction. Les ingénieurs Yoctopuce se sont donnés beaucoup de mal pour que votre YoctoHub-Wireless soit facile à installer n'importe où et soit facile utiliser en toutes circonstances. Néanmoins, si ce module venait à vous décevoir, n'hésitez pas à contacter le support Yoctopuce[2].

[1] http://www.yoctopuce.com/FR/virtualhub.php
[2] support@yoctopuce.com

2. Présentation



1: Yocto-bouton
2: Port USB de contrôle + alimentation
3: Yocto-Led
4: Indicateur de sur-consommation
5: Indicateur de transfert réseau
6: Touche réveil
7: Touche mise en sommeil
8: Connecteur d'antenne
9: Neutralisatation de la mise en sommeil
10: Connexion dorsale
11: Port descendant 1
12: Indicateur port descendant 1
13: Port descendant 2
14: Indicateur port descendant 2
15: Port descendant 3
16: Indicateur port descendant 3

2.1. Les éléments du YoctoHub-Wireless

Le numéro de série

Chaque Yocto-module a un numéro de série unique attribué en usine, pour les modules YoctoHub-Wireless ce numéro commence par YHUBWLN1. Le module peut être piloté par logiciel en utilisant ce numéro de série. Ce numéro de série ne peut pas être changé.

Le nom logique

Le nom logique est similaire au numéro de série, c'est une chaîne de caractères sensée être unique qui permet référencer le module par logiciel. Cependant, contrairement au numéro de série, le nom logique peut être modifié à volonté. L'intérêt est de pouvoir fabriquer plusieurs exemplaires du même

projet sans avoir à modifier le logiciel de pilotage. Il suffit de programmer les mêmes noms logiques dans chaque exemplaire. Attention, le comportement d'un projet devient imprévisible s'il contient plusieurs modules avec le même nom logique et que le logiciel de pilotage essaye d'accéder à l'un de ces modules à l'aide de son nom logique. A leur sortie d'usine, les modules n'ont pas de nom logique assigné, c'est à vous de le définir.

Le Yocto-bouton

Le Yocto-bouton a deux fonctions. Premièrement, il permet d'activer la Yocto-balise (voir la Yocto-Led ci-dessous). Deuxièmement, si vous branchez un Yocto-module en maintenant ce bouton appuyé, il vous sera possible de reprogrammer son firmware avec une nouvelle version. Notez qu'il existe une méthode plus simple pour mettre à jour le firmware depuis l'interface utilisateur, mais cette méthode-là peut fonctionner même lorsque le firmware chargé sur le module est incomplet ou corrompu.

La Yocto-Led

En temps normal, la Yocto-Led sert à indiquer le bon fonctionnement du module: elle émet alors une faible lumière bleue qui varie lentement mimant ainsi une respiration. La Yocto-Led cesse de respirer lorsque le module ne communique plus, par exemple s'il est alimenté par un hub sans connexion avec un ordinateur allumé.

Lorsque vous appuyez sur le Yocto-bouton, la Led passe en mode Yocto-balise: elle se met alors à flasher plus vite et beaucoup plus fort, dans le but de permettre une localisation facile d'un module lorsqu'on en a plusieurs identiques. Il est en effet possible de déclencher la Yocto-balise par logiciel, tout comme il est possible de détecter par logiciel une Yocto-balise allumée.

La Yocto-Led a une troisième fonctionnalité moins plaisante: lorsque le logiciel interne qui contrôle le module rencontre une erreur fatale, elle se met à flasher SOS en morse[1]. Dans ce cas, débranchez puis re-branchez le module. Si le problème venait à se reproduire, vérifiez que le module contient bien la dernière version du firmware et, dans l'affirmative, contactez le support Yoctopuce[2].

Le connecteur de contrôle et d'alimentation (Power / Control port)

Ce connecteur permet d'alimenter le YoctoHub-Wireless et les modules qui lui sont connectés à l'aide d'un simple chargeur USB. Ce connecteur permet aussi de prendre le contrôle du YoctoHub-Wireless par USB, exactement comme on pourrait le faire avec un module Yoctopuce classique. C'est particulièrement utile lorsque que l'on désire configurer le YoctoHub-Wireless sans connaître son adresse IP.

Les ports descendants

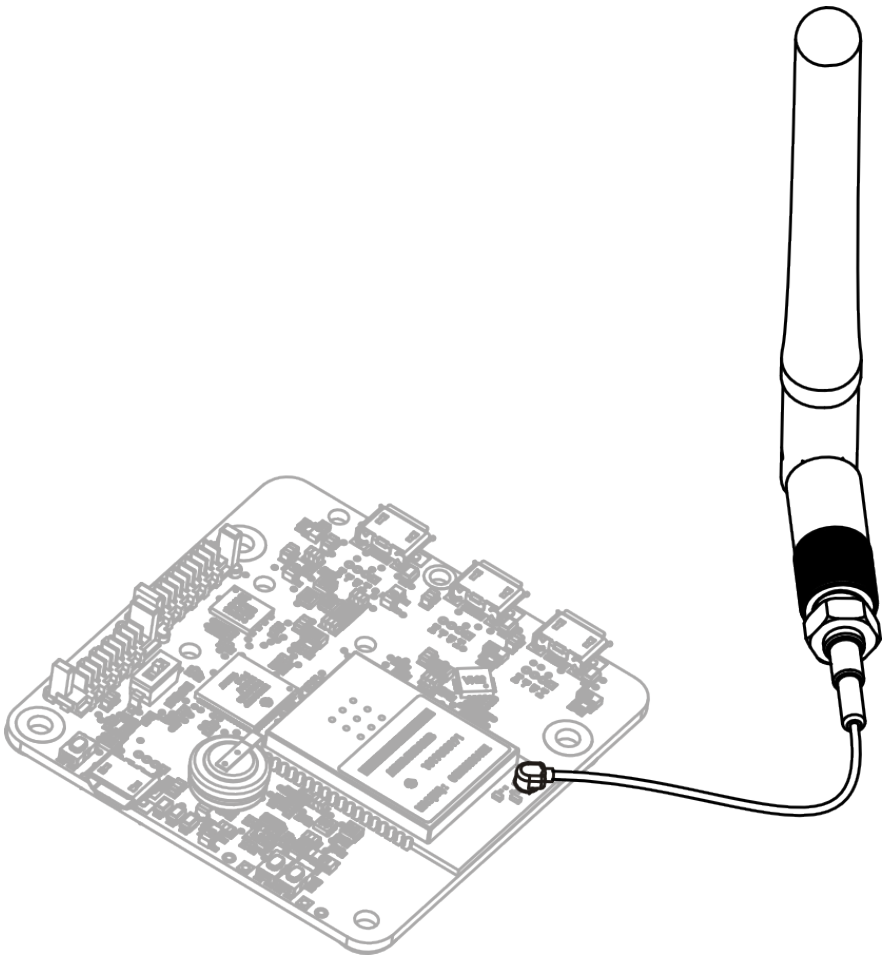
Vous pouvez connecter jusqu'à trois modules Yoctopuce sur ces ports. Ils seront alors accessibles comme s'ils étaient branchés à un ordinateur faisant tourner un VirtualHub. Attention, le protocole entre le YoctoHub-Wireless et le module Yoctopuce n'est pas de l'USB mais un protocole propriétaire plus léger. De ce fait le YoctoHub-Wireless ne peut pas gérer des périphériques autres que des modules Yoctopuce. Un hub USB standard ne fonctionnera pas non plus[3]. Si vous désirez brancher plus de trois modules Yoctopuce, utilisez le connecteur dorsal pour y connecter un ou plusieurs YoctoHub-Shield[4].

Attention, les connecteurs USB du YoctoHub-Wireless sont simplement soudés en surface et peuvent être arrachés si la prise USB venait à faire fortement levier. Si les pistes sont restées en place, le connecteur peut être ressoudé à l'aide d'un bon fer et de flux. Alternativement, vous pouvez souder un fil USB directement dans les trous espacés de 1.27mm prévus à cet effet, près du connecteur.

[1] court-court-court long-long-long court-court-court
[2] support@yoctopuce.com
[3] Le Micro-USB-Hub fabriqué par Yoctopuce est un hub USB standard et ne fonctionnera pas avec le YoctoHub-Wireless.
[4] www.yoctopuce.com/FR/products/yoctohub-shield

Le connecteur d'antenne

Le YoctoHub-Wireless dispose d'un connecteur d'antenne coaxial ultra miniature (UFL). Prenez soin du connecteur UFL, il est fragile et n'est pas conçu pour supporter beaucoup de cycles de connexion/déconnexion. Le YoctoHub-Wireless est livré en standard avec un petit câble UFL vers RP-SMA femelle (reverse polarity SMA: filetage extérieur et pin mâle au centre) et une antenne correspondante RP-SMA mâle (filetage intérieur et tube femelle au centre). Vous pouvez utiliser une autre antenne de votre choix, pour autant qu'elle soit conçue pour la gamme de fréquence 2.4 GHz et qu'elle ait le bon connecteur. Méfiez vous des différentes variantes de connecteurs SMA: il existe des antennes pour chacune des quatre combinaisons SMA/RP-SMA et mâle/femelle, mais seule une antenne RP-SMA mâle est utilisable avec le câble d'antenne fourni. Prenez garde aussi au fait que l'utilisation d'antennes à fort gain peut vous amener à émettre un signal supérieur à la norme autorisée dans votre pays.



Connexion de l'antenne.

-

Indicateur de sur-consommation

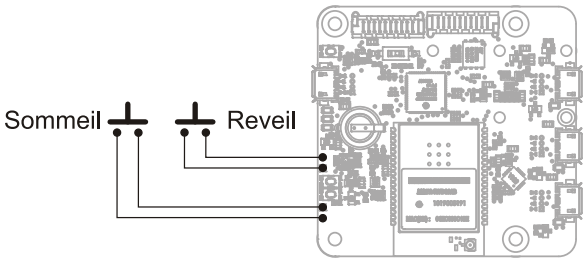
Le YoctoHub-Wireless analyse en permanence sa consommation. S'il détecte une consommation globale de plus de 2A suite à une surcharge sur un des ports descendants par exemple, il va automatiquement désactiver tous les ports descendants et allumer l'indicateur de sur-consommation. Pour isoler la source du problème, vous pouvez réactiver les ports un à un, en surveillant l'augmentation de la consommation. Alternativement, si connaissez la source du problème de sur-consommation et savez l'avoir résolu, vous pouvez redémarrer le YoctoHub-Wireless pour réactiver tous les ports.

Notez que l'indicateur de sur-consommation est une mesure de protection qui peut éviter la surchauffe, mais ce n'est pas une garantie de protection contre les court- circuits.

Mise en sommeil

En temps normal, le YoctoHub-Wireless consomme environ 0,5 Watt (110mA), auquel il faut ajouter la consommation des modules qui lui sont connectés. Mais il est capable de se mettre en sommeil pour réduire sa consommation d'énergie au strict minimum, et de se réveiller à une heure précise (ou lorsqu'un contact extérieur est fermé). Cette fonctionnalité est très utile pour construire des installations de mesure fonctionnant sur batterie. Lorsque que le YoctoHub-Wireless est en sommeil, la quasi totalité de l'électronique du module ainsi que les modules Yoctopuce connectés sont hors tension, ce qui réduit sa consommation totale à 75 µW (15 µA).

La mise en sommeil et le réveil peuvent être soit programmés sur base horaire, soit contrôlés par logiciel, soit contrôlés manuellement à l'aide de deux boutons poussoirs présents sur le circuit du YoctoHub-Wireless. Vous y trouverez aussi deux paires de contacts qui permettent de dériver ces deux boutons.



Dérivation des boutons de mise en sommeil et de réveil.

Le YoctoHub-Wireless dispose d'un interrupteur qui permet de désactiver au niveau hardware la fonctionnalité de mise en sommeil. Cette fonctionnalité est utile en particulier durant les phases de développement/déverminage de votre projet, ainsi que pour effectuer les mises à jour du firmware.

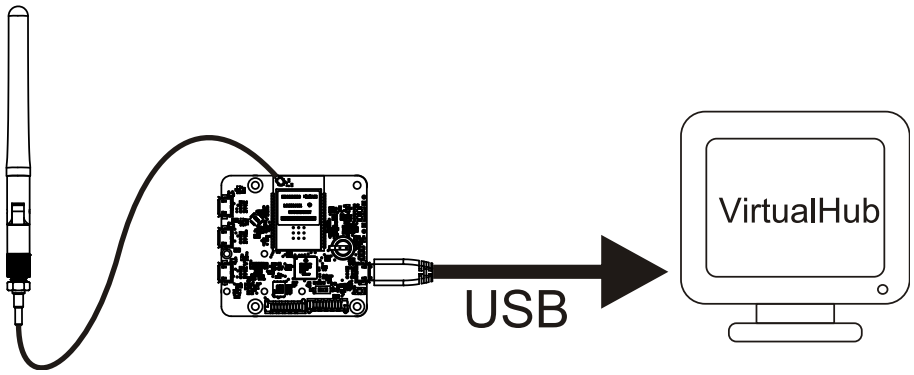
3. Premiers pas

Ce chapitre a pour but de vous aider à connecter et configurer votre YoctoHub-Wireless pour la première fois

3.1. Configuration manuelle

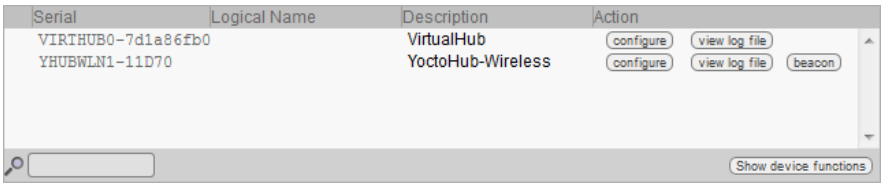
Si vous devez utiliser une configuration réseau particulière, vous pouvez configurer votre YoctoHub-Wireless via son port de contrôle USB, en utilisant le *VirtualHub*[1].

Lancez un *VirtualHub* sur votre ordinateur favori et raccordez votre ordinateur au port *power / control port* du YoctoHub-Wireless. Vous aurez besoin d'un câble USB A-MicroB.



Configuration: raccordez par USB votre YoctoHub-Wireless à un ordinateur

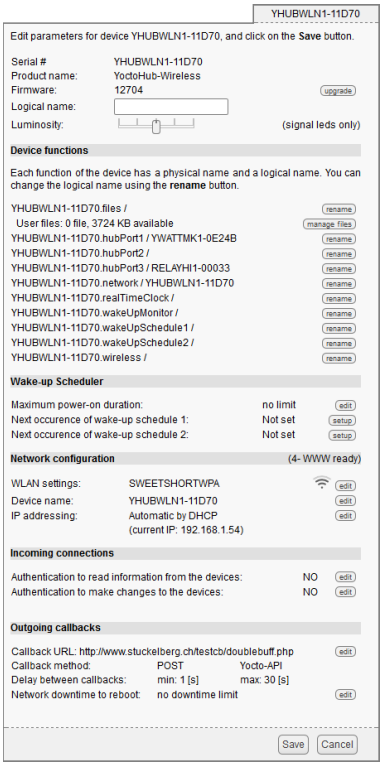
Lancez alors votre browser favori[2] sur l'url de votre *VirtualHub*. Il s'agit généralement *http://127.0.0.1:4444*. Vous obtiendrez la liste des modules Yoctopuce connectés par USB, dont votre YoctoHub-Wireless



Liste des modules Yoctopuce raccordés par USB à votre ordinateur, dont votre YoctoHub-Wireless

[1] *http://www.yoctopuce.com/FR/virtualhub.php*
[2] Sauf Opera

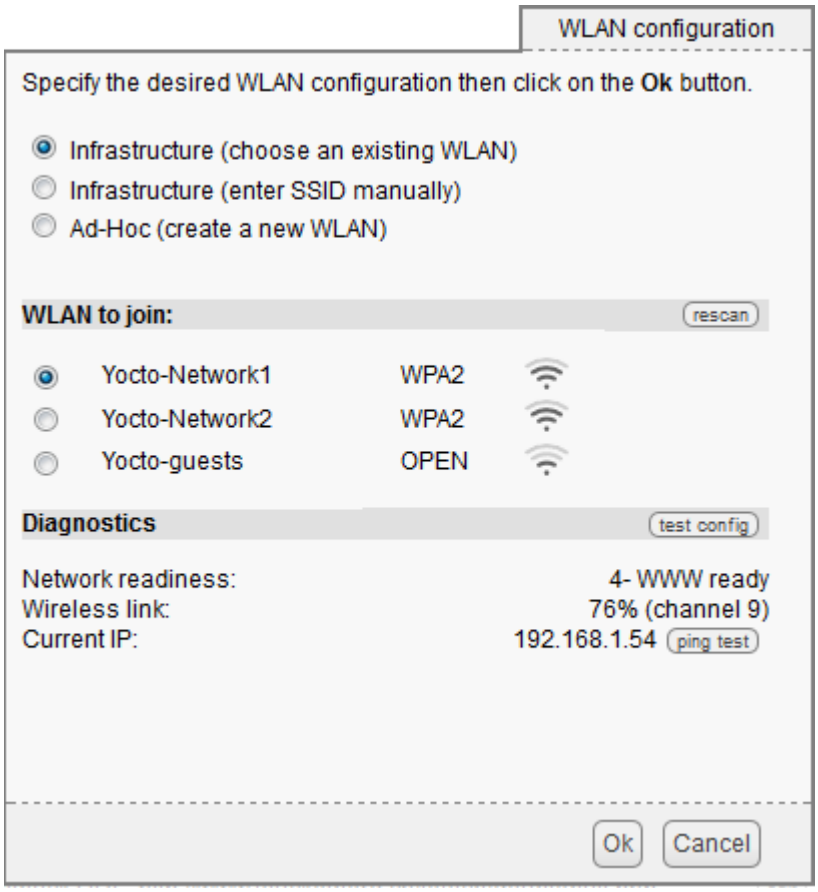
Cliquez sur le bouton **configure** correspondant à votre YoctoHub-Wireless, votre obtiendrez la fenêtre de configuration du module. Cette fenêtre comporte une section **Network configuration**.



Fenêtre de configuration du module YoctoHub-Wireless

Connexion au réseau sans fil

La première chose à faire consiste à configurer votre YoctoHub-Wireless pour qu'il se connecte à votre réseau WiFi. Pour cela cliquez sur le bouton **edit** correspondant à **WLAN settings** dans la section **Network configuration**, la fenêtre de configuration du réseau sans fil apparaît:



Fenêtre de configuration du réseau sans fil.

Vous pouvez alors choisir si vous souhaitez connecter votre YoctoHub-Wireless à un reseau existant, ou si vous préférez entrer manuellement le SSID du réseau que vous voulez utiliser.

Vous pouvez aussi configurez le YoctoHub-Wireless pour qu'il génère son propre réseau sans fil en mode *ad-hoc*. Vous pourrez alors vous connecter directement sur le YoctoHub-Wireless sans avoir a passer par un serveur d'infrastructure (point d'accès). Soyez toutefois conscients que ce mode de fonctionnement *ad-hoc* a des limitations importantes par rapport à un vrai réseau WiFi. En particulier, les appareils sous Android ne sont pas capable de s'y connecter.

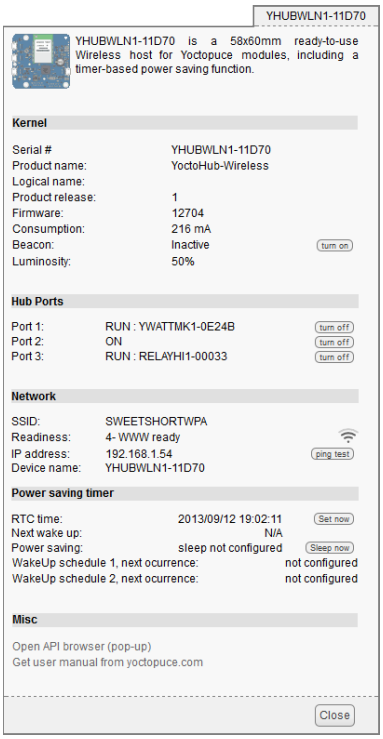
Après avoir entré les paramètres de réseau sans fil et éventuellement les avoir testés, vous pouvez cliquer sur le bouton **Ok** pour fermer cette fenêtre de configuration et retourner à la fenêtre de configuration générale.

Vous pouvez aussi, en cas de besoin, configurer quelle adresse IP doit être attribuée au YoctoHub-Wireless. Pour ce faire, cliquez sur le bouton **edit** en face de la ligne **ip adressing**.

Vous pouvez alors choisir si l'adresse IP de votre YoctoHub-Wireless doit être attribuée par DHCP ou si elle doit être fixe. L'option DHCP est recommandée dans la mesure où cette fonctionnalité est supportée par la plupart des boîtiers ADSL (c'est la configuration par défaut). Si vous ne savez pas ce qu'est un serveur DCHP mais avez l'habitude de brancher des appareils sur votre réseau et de les voir marcher sans problème, ne touchez à rien.

Vous pouvez aussi choisir le nom réseau de votre YoctoHub-Wireless. Vous pourrez ainsi accéder à votre YoctoHub-Wireless en utilisant ce nom plutôt que son adresse IP. Une fois la partie réseau configurée, cliquez sur le bouton **Save**. Cela qui aura pour effet de sauver vos modifications et de fermer la fenêtre de configuration. Ces modifications étant sauvées dans la mémoire persistante du YoctoHub-Wireless, il s'en rappellera même après avoir été privé de courant.

Cliquez sur le numéro de série correspondant à votre YoctoHub-Wireless. Cela ouvrira la fenêtre détails de votre module:



Les propriétés du YoctoHub-Wireless

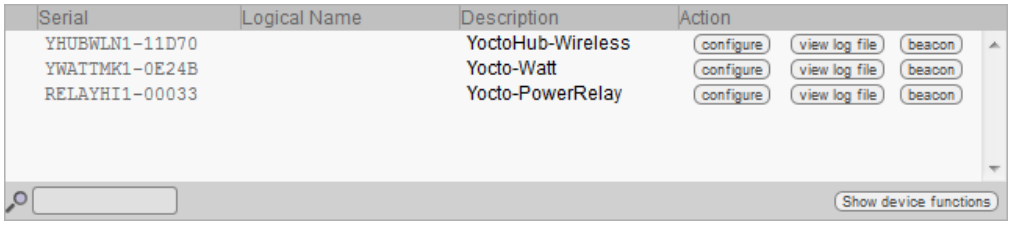
Cette fenêtre comporte une section qui relate l'état de la partie réseau du YoctoHub-Wireless. Vous y trouverez son adresse MAC, adresse IP courante et nom de réseau. Cette section donne aussi l'état de la connexion réseau. Ces états peuvent être:

- 0- search for link: Le module cherche une liaison réseau. Si cet état persiste, il est probable que le réseau wifi recherché n'est pas à proximité
- 1- network exists: le réseau wifi recherché a été détecté
- 2- network linked: le YoctoHub-Wireless a pu se connecter au réseau.
- 3- DHCP ready: le réseau local est opérationnel (adresse IP obtenue)
- 4- DNS ready: Un server DNS est joignable par le réseau
- 5- WWW ready: le module a pu vérifier la connectivité à Internet en se connectant à un serveur de temps (NTP).

Après avoir vérifié que votre module a bien une adresse IP valide, vous pouvez fermer la fenêtre détails, arrêter votre VirtualHub et débrancher le câble USB de contrôle de votre ordinateur: il suffira que le module soit alimenté pour l'utiliser.

Vous pouvez désormais accéder à votre YoctoHub-Wireless en tapant directement son adresse IP dans la barre d'adresse de votre browser favori. Le module répond au port HTTP standard, mais aussi au port 4444 utilisé par le VirtualHub. Si l'adresse IP de votre module est *192.168.0.10*, vous pourrez donc le joindre avec l'URL *http://192.168.0.10*.

Vous obtiendrez alors directement l'interface du YoctoHub-Wireless. Cette interface est en tout point identique à celle du *VirtualHub*. Vous retrouvez le YoctoHub-Wireless sur la première ligne et les modules connectés au YoctoHub-Wireless sur les suivantes.



L'interface du YoctoHub-Wireless est identique à celle d'un VirtualHub.

Si vous avez attribué un nom à votre YoctoHub-Wireless, vous pouvez aussi utiliser ce nom sur le réseau local. Par exemple, si vous avez utilisé le nom réseau *yoctohub*, vous pouvez contacter le module avec l'URL *http://yoctohub* sous Windows et avec l'URL *http://yoctohub.local* sous Mac OS X et Linux. Notez que cette technique est limitée au sous-réseau du YoctoHub-Wireless. Si vous voulez contacter le module par nom depuis un autre réseau, vous devez utiliser une infrastructure DNS classique.

3.2. Configuration automatisée

Il est possible d'industrialiser la configuration réseau du YoctoHub-Wireless. Vous trouverez dans les chapitres suivants de cette documentation la description des fonctions de programmation permettant de relire l'adresse Ethernet d'un module (adresse MAC), et de configurer tous ses paramètres réseau.

Les fonctions de configuration réseau sont aussi accessibles par ligne de commande, en utilisant l'utilitaire `YNetwork` disponible dans la librairie de programmation en ligne de commande[3].

Après avoir effectué un changement de réglage par programmation, prenez garde à appeler la fonction `saveToFlash()` pour vous assurez que les réglages soient sauvés de manière permanente dans la mémoire flash du module.

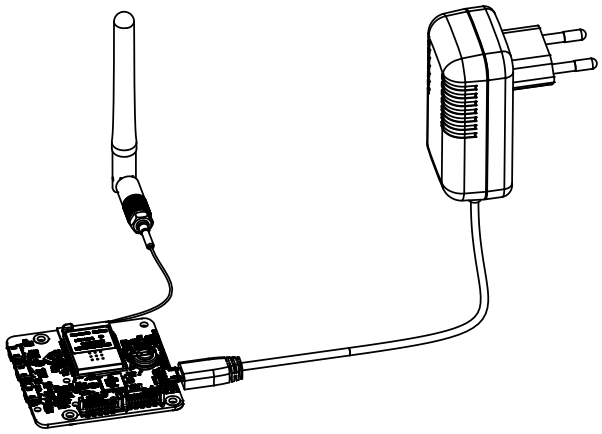
3.3. Connexions

Alimentation

Le YoctoHub-Wireless doit être alimenté par la prise USB de contrôle.

USB

Branchez simplement un chargeur USB dans le port *power / control port*, assurez-vous tout de même que le chargeur soit d'une puissance électrique suffisante: le YoctoHub-Wireless consomme environ 110mA, auxquels il faudra ajouter la consommation de chaque sous-module. Le YoctoHub-Wireless est conçu pour gérer 2A au maximum, c'est pourquoi un chargeur USB capable de délivrer au moins 2A est recommandé. Par ailleurs, vous devrez veiller à ce que la consommation totale de l'ensemble hub + sous-modules ne dépasse pas cette limite.

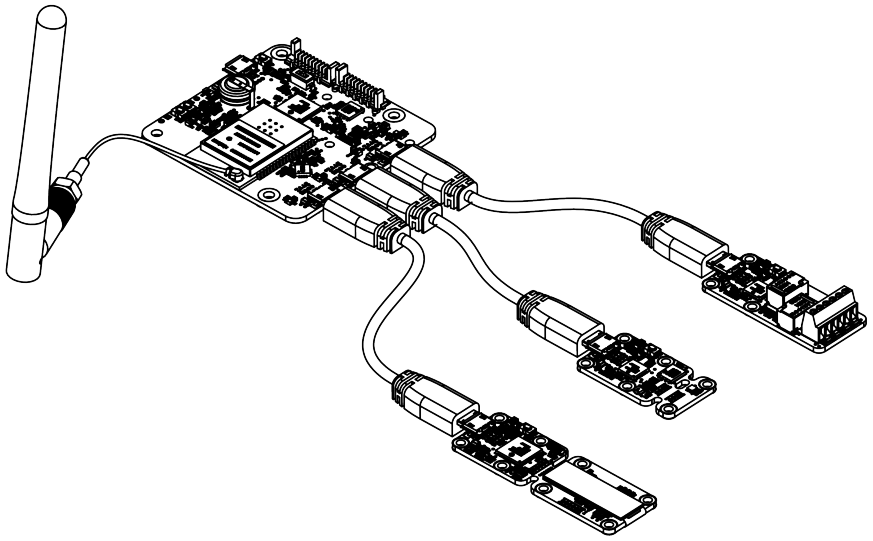


Le YoctoHub-Wireless peut être alimenté par un chargeur USB

Sous-modules

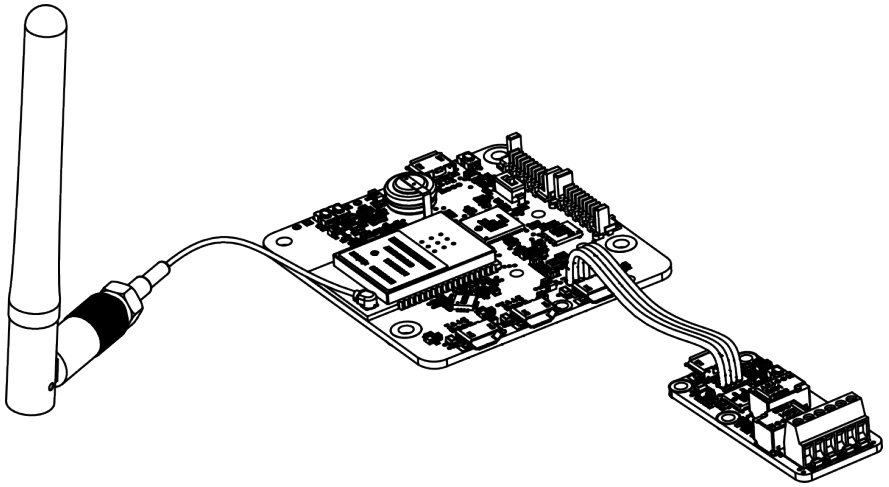
Le YoctoHub-Wireless est capable de piloter tous les modules Yoctopuce de la gamme *Yocto*. Ces modules peuvent êtres connectés directement aux ports descendants, ils seront détectés automatiquement. Vous aurez besoin pour cela de câbles USB MicroB-MicroB. Vous pouvez utiliser des câbles OTG ou non, cela n'a pas d'importance.

[3] http://www.yoctopuce.com/FR/libraries.php



Connexion des sous-modules à l'aide de câbles USB

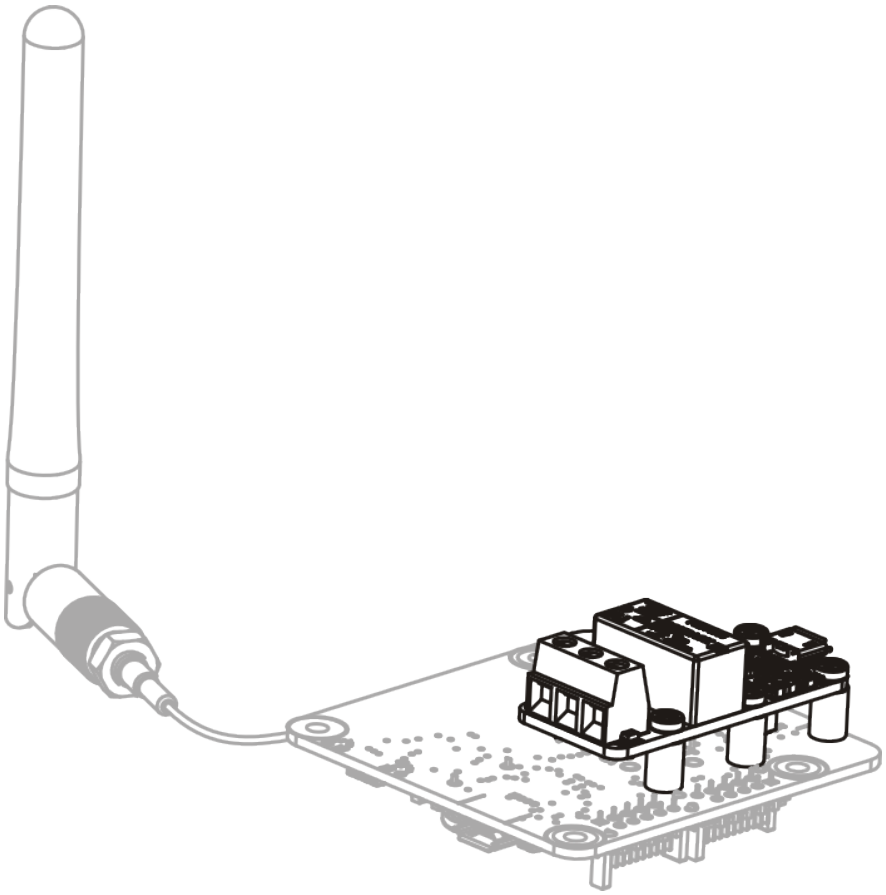
Alternativement, vous pouvez connecter vos modules de manière plus compacte à l'aide de câbles au pas 1.27mm: tous les modules Yoctopuce disposent en effet de contacts à cet effet. Vous pouvez soit souder des connecteurs 1.27mm sur les modules et utiliser des câbles avec connecteurs 1.27mm, soit souder directement du câble plat au pas 1.27mm. Si vous choisissez cette dernière option, il est recommandé d'utiliser du câble plat mono-brin, moins souple que le multi-brin mais beaucoup plus facile à souder. Soyez particulièrement attentif aux polarités: Le YoctoHub-Wireless, tout comme l'ensemble de modules de la gamme Yoctopuce, n'est pas protégé contre les inversions de polarité. Une telle inversion a toutes les chances de détruire vos équipements. Assurez-vous que la position du contact carré de part et d'autre du câble correspondent.



Connexion des sous-modules à l'aide de câble nappe

Le YoctoHub-Wireless est conçu pour que vous puissiez fixer un module simple largeur directement dessus. Vous aurez besoin de vis, d'entretoises[4] et d'un connecteur au pas 1.27mm[5]. Vous pouvez ainsi transformer un module Yoctopuce USB en en module réseau tout en gardant un format très compact.

[4] http://www.yoctopuce.com/FR/products/accessoires-et-connectique/fix-2-5mm
[5] http://www.yoctopuce.com/FR/products/accessoires-et-connectique/board2board-127



Fixation d'un module directement sur le hub

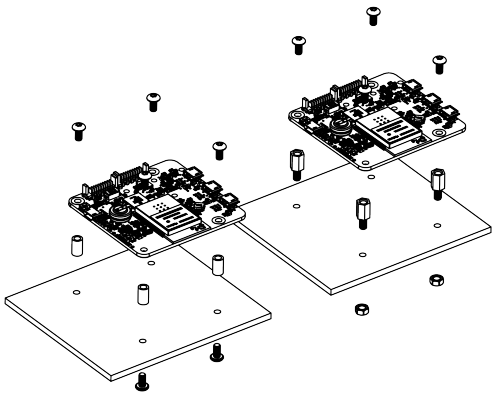
Attention, le YoctoHub-Wireless est conçu pour piloter des modules Yoctopuce uniquement. En effet le protocole utilisé entre le YoctoHub-Wireless et les sous-modules n'est pas de l'USB mais un protocole propriétaire, beaucoup plus léger. Si d'aventure vous branchez un périphérique autre qu'un module Yoctopuce sur un des ports descendants du YoctoHub-Wireless, le port en question sera automatiquement désactivé pour éviter d'endommager le périphérique.

4. Montage

Ce chapitre fournit des explications importantes pour utiliser votre module YoctoHub-Wireless en situation réelle. Prenez soin de le lire avant d'aller trop loin dans votre projet si vous voulez éviter les mauvaises surprises.

4.1. Fixation

Pendant la mise au point de votre projet, vous pouvez vous contenter de laisser le hub se promener au bout de son câble. Veillez simplement à ce qu'il ne soit pas en contact avec quoi que soit de conducteur (comme vos outils). Une fois votre projet pratiquement terminé, il faudra penser à faire en sorte que vos modules ne puissent pas se promener à l'intérieur.



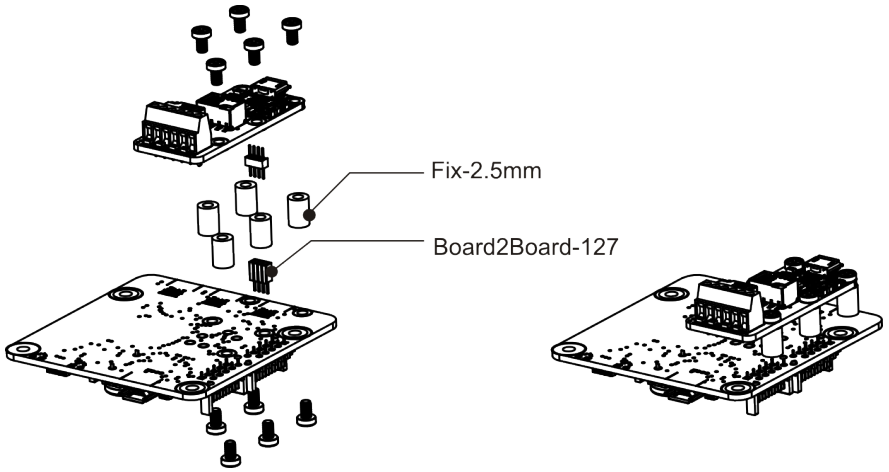
Exemples de montage sur un support.

Le module YoctoHub-Wireless dispose de trous de montage 3mm. Vous pouvez utiliser ces trous pour y passer des vis. Le diamètre de la tête de ces vis ne devra pas dépasser 8mm, sous peine d'endommager les circuits du module.

Veillez à que l'électronique module ne soit pas en contact avec le support. La méthode recommandée consiste à utiliser des entretoises. Vous pouvez monter le module dans le sens qui vous convient: mais vous devez conscient du fait que les composants électroniques du YoctoHub-Wireless, la partie réseau en particulier, dégagent de la chaleur. Vous devrez donc faire en sorte que la chaleur ne puisse pas s'accumuler.

4.2. Fixation d'un sous-module

Le YoctoHub-Wireless est conçu pour que vous puissiez visser un module simple largeur directement dessus. Par simple largeur, on entend les modules de 20 mm de large. Tous les modules simple largeur ont leurs 5 trous de fixation et le connecteur USB au même endroit. Le sous-module peut être fixé à l'aide de vis et d'entretoises. Il y a derrière les connecteurs USB du YoctoHub-Wireless et du sous-module un ensemble de 4 contacts qui permettent d'effectuer la connexion électrique entre le hub et le sous-module. Si vous ne vous sentez pas suffisamment à l'aise avec un fer à souder, vous pouvez aussi aussi utiliser un simple câble USB MicroB-MicroB, OTG ou non.



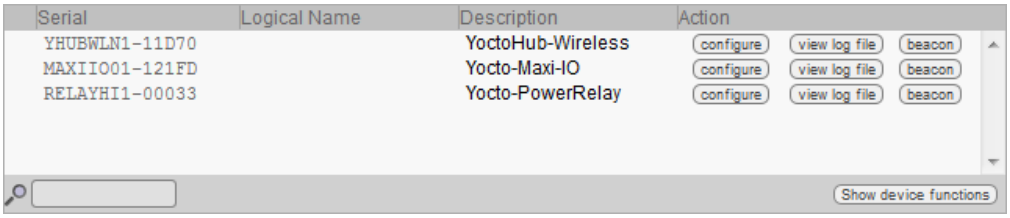
Fixation d'un module directement sur le hub

Prenez garde à bien monter le module sur la face prévue, comme illustré ci-dessus. Les 5 trous du module doivent correspondre aux 5 trous du YoctoHub-Wireless, et le contact carré sur sur le module doit être connecté au contact carré sur le port descendant du YoctoHub-Wireless. Si vous montez un module sur l'autre face ou d'une autre manière, la polarité du connecteur sera inversée et vous risquez fort d'endommager définitivement votre matériel.

Tous les accessoires nécessaires à la fixation d'un module sur votre YoctoHub-Wireless sont relativement courants. Vous pourrez les trouver sur le site de Yoctopuce tout comme sur la plupart des sites vendant du matériel électronique. Attention cependant, la tête des vis servant à fixer le sous-module devra avoir un diamètre maximum de 4.5 millimètres, sous peine d'endommager les composants électroniques.

5. Utilisation du YoctoHub-Wireless

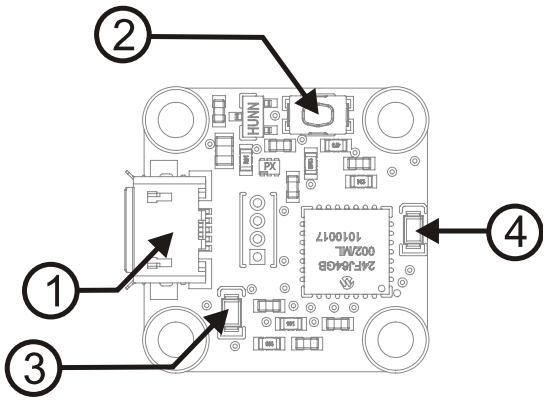
Outre fournir un accès réseau au module Yoctopuce, le YoctoHub-Wireless permet de tester et configurer vos modules Yoctopuce. Pour ce faire connectez-vous sur votre YoctoHub-Wireless à l'aide de votre navigateur internet favori[1]. Utilisez l'adresse IP du YoctoHub-Wireless ou encore son nom réseau, par exemple `http://192.168.0.10`. Une liste comprenant votre YoctoHub-Wireless ainsi que les modules qui lui sont connectés devrait apparaître.



Interface Web du YoctoHub-Wireless.

5.1. 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.

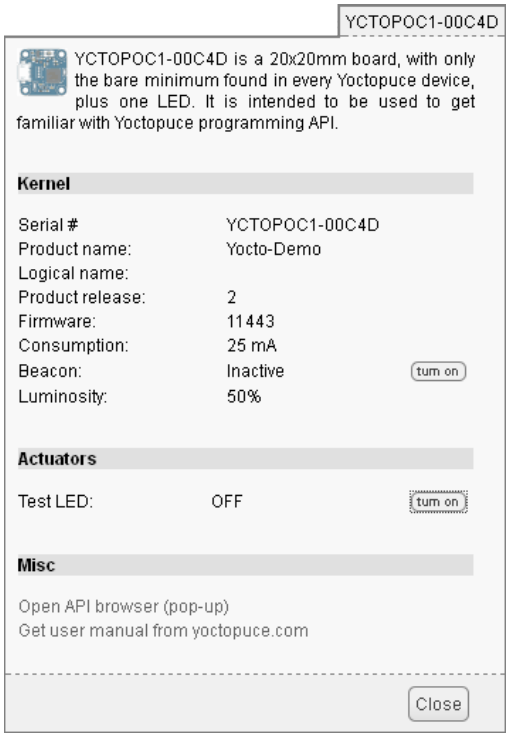


Yocto-bouton(1) et Led(2) de localisation d'un module Yocto-Demo. Ces deux éléments sont généralement placés au même endroit sur les modules.

[1] L'interface du YoctoHub-Wireless est régulièrement testée sur Internet Explorer, Firefox, Chrome et Safari. Elle ne fonctionne pas avec Opéra

5.2. Test des modules

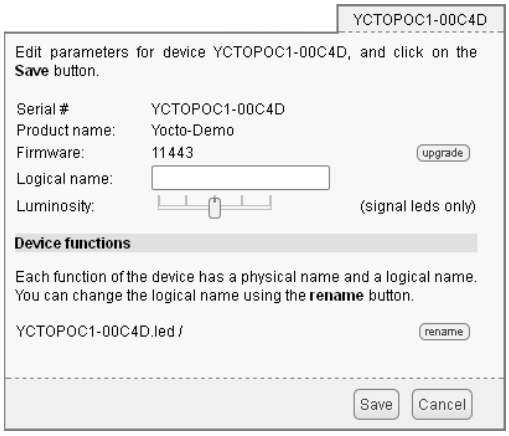
Pour tester un module, cliquez simplement sur le numéro de série d'un module dans l'interface, une fenêtre spécifique au module s'ouvrira. Cette fenêtre permet généralement d'activer les fonctions principales du module. Reportez-vous au manuel du module correspondant pour plus de détails.[2]



Fenêtre "détails" du module Yocto-Demo, obtenue via l'interface du YoctoHub-Wireless.

5.3. 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.



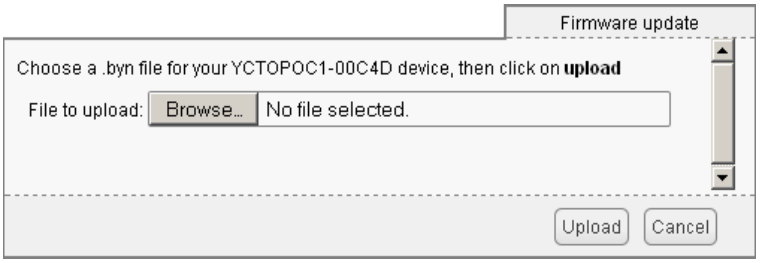
Fenêtre "configure" du module Yocto-Demo.

[2] Vous n'êtes pas obligé d'avoir un YoctoHub-Wireless plus récent qu'un module pour le tester/configurer: tous les éléments spécifiques aux interfaces des modules sont stockés dans la ROM des modules, et non pas dans le YoctoHub-Wireless.

5.4. Upgrades 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[3].

Pour mettre à jour le firmware d'un module, vous devez d'abord vous procurer le firmware, il peut être téléchargé depuis la page produit du module sur le site de Yoctopuce [4]. L'interface propose aussi un lien direct si elle détecte que le firmware n'est pas à jour[5]. Ces firmwares se présentent sous la forme de fichiers `.byn` de quelques dizaines de Kilo-octets, sauvez celui qui vous intéresse sur votre disque local.



Fenêtre de mise à jour du firmware.

Une fois votre fichier de firmware disponible localement, ouvrez la fenêtre **configuration** d'un module et cliquez sur le bouton **upgrade**. L'interface va vous demander de choisir le fichier de firmware que vous désirez utiliser. Entrez le nom du fichier et cliquez sur **Upload**. A partir de là, tout est automatique, le YoctoHub-Wireless va faire redémarrer le module en mode "mise à jour", mettre à jour le firmware, puis redémarrer 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.

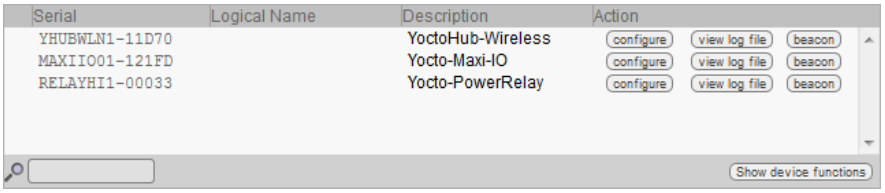
Le firmware du YoctoHub-Wireless peut être mis à jour de la même manière

En cas de perte de contrôle pendant une mise à jour du firmware (panne de courant ou débranchement involontaire), il est toujours possible de forcer manuellement un rechargement du firmware, même si le sous-module n'apparaît même plus dans la fenêtre du YoctoHub-Wireless. Pour ce faire, débranchez le sous-module et rebranchez-le en maintenant le Yocto-bouton pressé. Le module démarrera alors dans le mode de mise à jour du firmware, et vous pourrez recommencer la procédure de chargement.

[3] Ne faites jamais confiance à des gens qui vous disent que leur logiciel n'a pas de bug :-)
[4] www.yoctopuce.com
[5] A condition qu'elle ait réussi à accéder au site Web de Yoctopuce

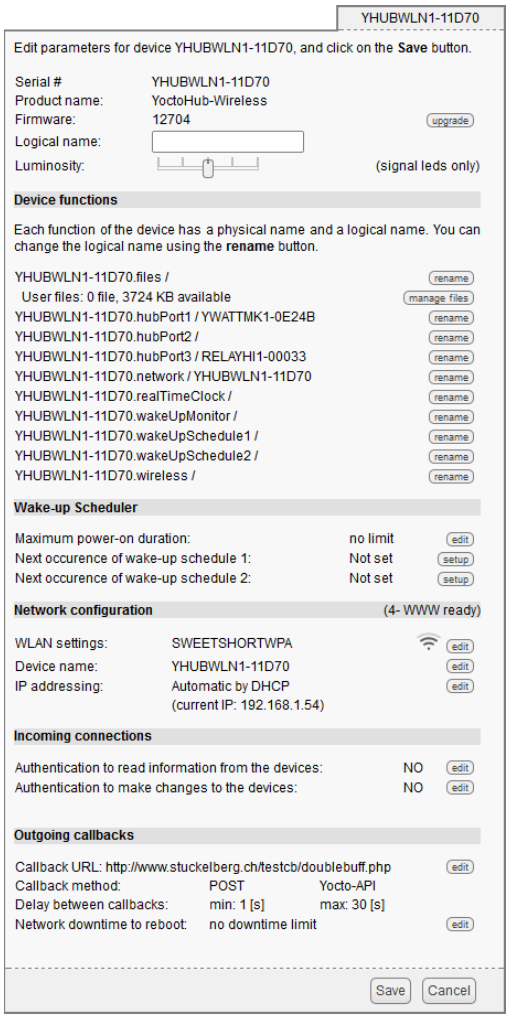
6. Contrôle d'accès

Le YoctoHub-Wireless vous permet d'instaurer un contrôle d'accès à vos modules Yoctopuce. Pour ce faire, cliquez simplement sur le bouton **Configure** de la ligne correspondant au YoctoHub-Wireless dans l'interface.



Cliquez sur le bouton "Configure" de la première ligne

Cela aura pour effet de faire apparaître la fenêtre de configuration du YoctoHub-Wireless.



La fenêtre de configuration du YoctoHub-Wireless

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

6.1. Accès "admin" protégé

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 ce YoctoHub-Wireless comme ils le souhaitent.

6.2. Accès "user" protégé

Le mot de passe *user* verrouille toute utilisation des modules. Lorsqu'il est configuré, toute utilisation sans mot de passe devient impossible. Les accès de type *user* ne permettent d'accéder aux modules qu'en lecture seule c'est-à-dire seulement pour consulter l'état des modules. Si vous instaurez 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 ce YoctoHub-Wireless.

Si vous configurez un accès *admin*, sans configurer d'accès *user*, les utilisateurs pourront continuer à consulter vos modules en lecture sans avoir à entrer de mot de passe.

Pour configurer l'accès au YoctoHub-Wireless, cliquez sur le bouton **edit** des lignes **Authentication to read the information from the devices** ou **Authentication to write information to the devices**

6.3. Influence sur les API

Attention, le contrôle d'accès agira aussi sur les API Yoctopuce qui tenteront de se connecter à ce YoctoHub-Wireless. 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 du YoctoHub-Wireless sous la forme *login:password@adresse:port*, par exemple:

```
yRegisterHub("admin:mypass@192.168.0.10:4444",errmsg);
```

6.4. Effacement des mots de passe

Si vous perdez le mot passe de votre YoctoHub-Wireless, vous pouvez reprendre le contrôle de votre module en réinitialisant tous ses réglages à la valeur par défaut. Pour ce faire, procurez-vous un câble USB pour le YoctoHub-Wireless, et branchez-le à un ordinateur avec le VirtualHub[1] installé en maintenant le Yocto-bouton pressé. Ceci va forcer le YoctoHub-Wireless à démarrer en mode de mise à jour du firmware. Il apparaît alors dans le VirtualHub en dessous de la liste des modules. Cliquez sur son numéro de série et choisissez un fichier de firmware à charger sur le module. Une fois le firmware rechargé avec cette méthode, le module sera réinitialisé avec les réglages d'usine, sans contrôle d'accès.

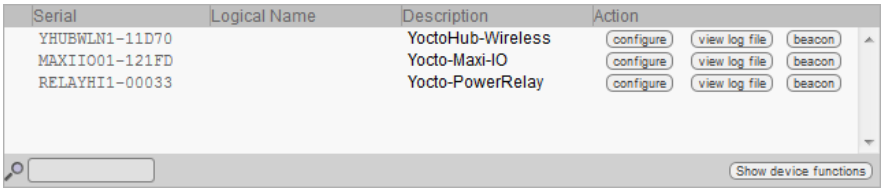
[1] http://www.yoctopuce.com/FR/virtualhub.php

7. Interactions avec l'extérieur

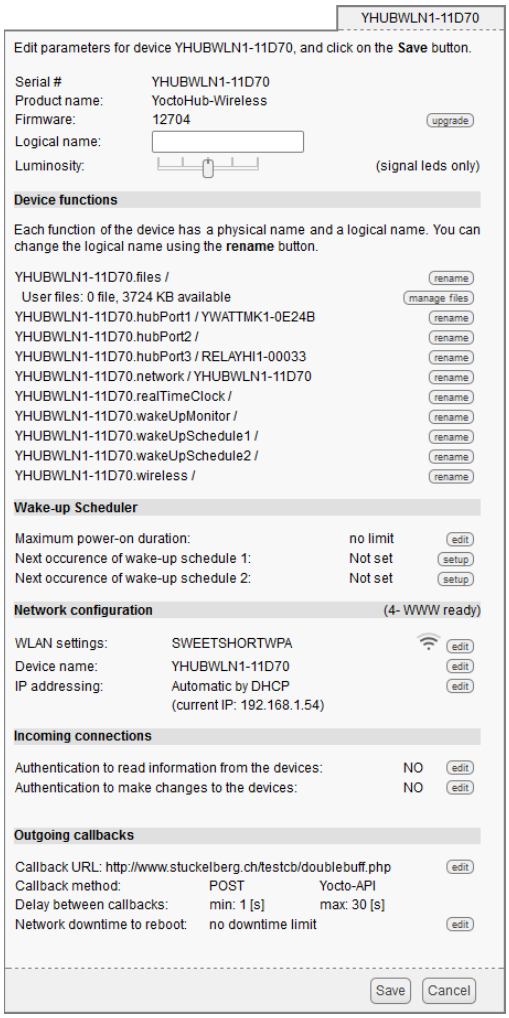
Le YoctoHub-Wireless est capable de poster sur le site web de votre choix l'état des modules qu'il voit. Les valeurs sont postées à intervalles réguliers et à chaque fois qu'une valeur change de manière significative. Cette fonctionnalité vous permettra d'interfacer vos modules Yoctopuce avec divers services web.

7.1. Configuration

Pour utiliser cette fonctionnalité, cliquez simplement sur le bouton **Configure** de la ligne correspondant au YoctoHub-Wireless dans l'interface, puis cliquez sur le bouton **edit** de la section **Outgoing callback**.



Cliquez sur le bouton "Configure" de la première ligne

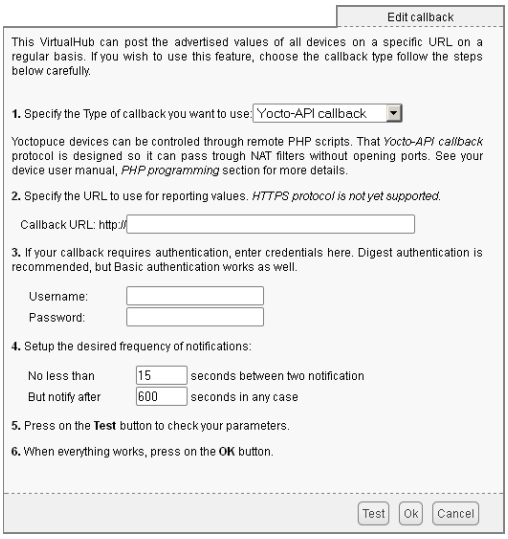


Puis éditez la section Outgoing callbacks.

La fenêtre de configuration des callbacks apparaît. Cette fenêtre va vous permettre de définir comment votre YoctoHub-Wireless va pouvoir interagir avec un serveur Web externe. Vous avez plusieurs type d'interactions a votre disposition.

7.2. User defined callback

Les "*User defined callback*" vous permettent de personnaliser la manière dont votre YoctoHub-Wireless va interagir avec un site Web externe. Vous avez besoin de définir l'URL du serveur Web sur lequel le YoctoHub-Wireless va poster l'état de ses devices. Notez que seul le protocole HTTP est supporté (pas de HTTPS).



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. Le YoctoHub-Wireless 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.

Le YoctoHub-Wireless poste avec la méthode POST les valeurs notifiées[1] des modules à intervalle régulier, et à chaque fois qu'une de ces valeurs change de manière significative. Vous pouvez changer les délais d'attente entre les posts.

Tests

Afin de vous permettre de déboguer le processus, le YoctoHub-Wireless vous permet de visualiser la réponse au callback envoyé par le serveur Web. Cliquez simplement sur le bouton **test** une fois que vous avez renseigné tous les champs. Si le résultat vous paraît satisfaisant, fermez la fenêtre de debug, et cliquez sur **Ok**.

Formats

Les valeurs sont postées sous une des formes suivantes:

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:

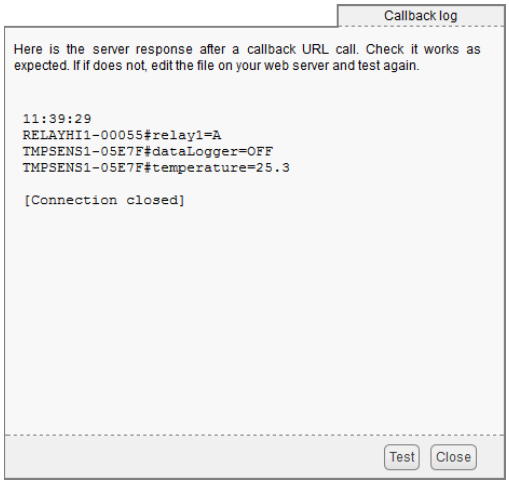
```
NUMERO_DE_SERIE#NOM_HARDWARE = VALEUR
```

Voici un script PHP qui vous permettra de visualiser le contenu des données postées par le callback, suivi du résultat dans la fenêtre de debug.

```
<?php
  Print(Date('H:i:s')."\r\n");
```

[1] Les valeurs notifiées sont celles que vous voyez quand vous cliquez sur *show functions* dans l'interface principale du YoctoHub-Wireless .

```
  foreach ($_POST as $key=>$value) {
      Print("$key=$value\r\n");
  }
?>
```



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

7.3. Yocto-API callback

L'API Yoctopuce PHP est capable de fonctionner en mode *callback*. Dans ce mode un script PHP peut prendre le contrôle de vos modules à travers un filtre NAT sans que vous ayez à ouvrir un port. Typiquement cela permet de contrôler depuis un site Web public des modules Yoctopuce installés derrière un router ADSL privé. Le YoctoHub-Wireless sert alors de relais. Vous avez simplement à définir l'URL du script PHP de contrôle et éventuellement les crédentiels nécessaires pour y accéder. Vous trouvez plus d'informations sur le fonctionnement du mode "callback" de l'API PHP dans le mode d'emploi de vos modules Yoctopuce.

7.4. Xively (anciennement Cosm)

Xively[2] est un service de cloud payant (mais avec un mode de développement limité gratuit) qui vous permettra tracer des graphes avec les données issues de vos capteurs Yoctopuce, et ce sans écrire la moindre ligne de code. Vous avez besoin d'un compte Xively et de définir un feed et une clef API sur le site de Xively. Entrez ces deux paramètres dans l'interface du YoctoHub-Wireless , c'est tout. Au besoin vous trouverez des explications plus détaillées sur le blog de Yoctopuce[3]. Yoctopuce n'est en aucune manière affilié à Xively.

7.5. Thinkspeak

ThingSpeak[4] est un autre service de cloud, entièrement gratuit lui, qui permet aussi de tracer des graphes avec des données issues de vos capteurs Yoctopuce. Ce service présente quelques limitations de fonctionnalité par rapport à Xively, mais a l'avantage de ne nécessiter aucune license payante. Vous trouverez sur le blog de Yoctopuce[5] comment configurer votre YoctoHub-Wireless pour poster des données directement sur ThingSpeak. Yoctopuce n'est en aucune manière affilié à ThingSpeak.

[2] www.xively.com
[3] http://www.yoctopuce.com/FR/article/enregistrer-des-mesures-sur-internet
[4] www.thingspeak.com
[5] http://www.yoctopuce.com/FR/article/alternatives-a-cosm-pour-enregistrer-des-mesures

8. Programmation

8.1. Accès aux modules connectés

Le YoctoHub-Wireless se comporte exactement comme un ordinateur faisant tourner un *VirtualHub*. La seule différence entre un programme utilisant l'API Yoctopuce utilisant des modules en USB natif et ce même programme utilisant des modules Yoctopuce connecté à un YoctoHub-Wireless se situe au niveau de l'appel a *registerHub*. Pour utiliser des modules USB connectés en natif, le paramètre de *RegisterHub* est *usb*, pour utiliser des modules connectés à un YoctoHub-Wireless, il suffit de remplacer ce paramètre par l'adresse IP du YoctoHub-Wireless. Par en exemple, en Delphi:

```
YRegisterHub("usb",errmsg);
```

devient

```
YRegisterHub("192.168.0.10",errmsg); // l'adresse IP du hub est 192.168.0.10
```

8.2. Contrôle du YoctoHub-Wireless

Du point de vue API de programmation, le YoctoHub-Wireless est un module comme les autres. Il est parfaitement contrôlable depuis l'API Yoctopuce. Pour ce faire, vous aurez besoin des classes suivantes.

Module

Cette classe, commune à tous les modules Yoctopuce permet de contrôler le module en temps que tel. Elle vous permettra de contrôler la Yocto-Led, de connaître la consommation sur USB du YoctoHub-Wireless, etc.

Wireless

Cette classe permet de contrôler la configuration du réseau sans fil du YoctoHub-Wireless, en particulier le SSID et la clé d'accès au réseau sans fil.

Network

Cette classe permet de contrôler la partie réseau du YoctoHub-Wireless, vous pourrez contrôler l'état du link, lire l'adresse MAC, changer l'adresse IP du YoctoHub-Wireless, connaître la consommation sur PoE, etc.

HubPort

Cette classe permet de contrôler la partie hub. Vous pourrez activer ou désactiver les ports du YoctoHub-Wireless, vous pourrez aussi savoir quel module est connecté à quel port.

Files

Cette classe permet d'accéder aux fichiers stockées dans la mémoire flash du YoctoHub-Wireless. Le YoctoHub-Wireless dispose en effet d'un petit système de fichiers qui vous permet de stocker par exemple une Web App contrôlant les modules connectés au YoctoHub-Wireless.

WakeMonitor

Cette classe permet de contrôler la mise en sommeil du YoctoHub-Wireless.

WakeSchedule

Cette classe permet d'agender un ou plusieurs réveils du YoctoHub-Wireless.

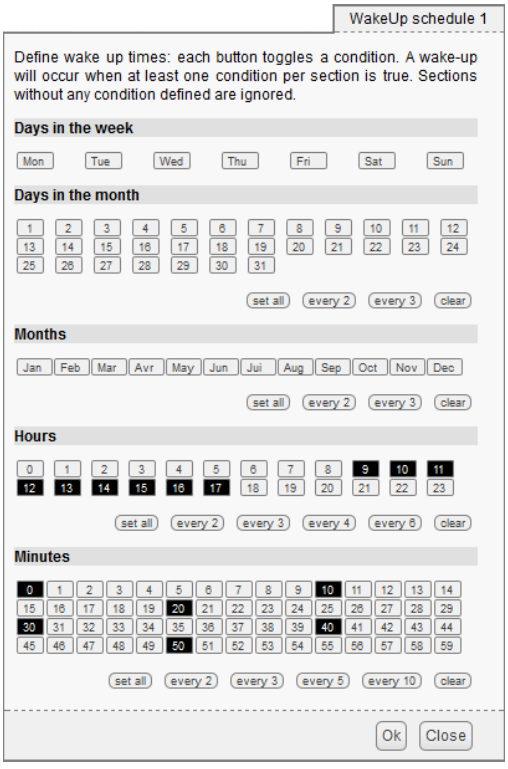
Vous trouverez quelques exemples de contrôle du YoctoHub-Wireless par programmation dans les librairies Yoctopuce, disponibles gratuitement sur le site de Yoctopuce.

9. Mise en sommeil

Le YoctoHub-Wireless dispose d'une horloge en temp réel (RTC) alimentée par un super condensateur, qui se recharge automatiquement lorsque le module est sous tension mais permet de maintenir l'heure sans aucune alimentation pendant plusieurs jours. Ce RTC est utilisé pour piloter un système de mise en sommeil afin d'économiser l'énergie. Le système de mise en sommeil peut être configuré manuellement à l'aide d'une interface, ou piloté par logiciel.

9.1. Configuration manuelle du système de réveil

Les conditions de réveil peuvent être configurées manuellement en vous connectant sur l'interface du YoctoHub-Wireless. Dans la section **Wake-up scheduler** de la fenêtre de configuration générale, cliquez sur le bouton setup correspondant à l'un des "wakeup-schedule". Une fenêtre qui permet d'agender des réveils plus ou moins réguliers s'ouvre. Il suffit de sélectionner les cases correspondant aux occurrences désirées. Les sections laissées vides seront ignorée.



Fenêtre de configuration des réveils, ici toutes les 10 minutes entre 9h et 17h

De même, vous pouvez configurer directement sur l'interface du YoctoHub-Wireless le temps d'éveil maximal désiré, après lequel le module retournera automatiquement en sommeil profond. Si vous utilisez votre YoctoHub-Wireless sur batteries, ceci vous assurera de ne pas vider les batteries même si aucun ordre de mise en sommeil explicite n'est reçu.

9.2. Paramétrage du système de réveil par logiciel

Au niveau de l'interface de programmation, le système de réveil est implémenté a l'aide de deux types de fonction : La fonction *wakeUpMonitor* et la fonction *WakeUpSchedule*.

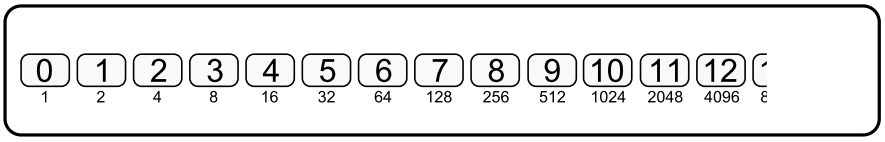
wakeUpMonitor

La fonction *wakeUpMonitor* gère l'éveil et la mise en sommeil proprement dits. Elle met à disposition toutes les fonctionnalités de contrôle immédiate: éveil immédiat, mise en sommeil immédiate, calcul de la date du prochain réveil etc... Le fonction *wakeUpMonitor* permet aussi de définir le temps maximum pendant lequel le YoctoHub-Wireless peut rester éveil avant de de se mettre automatiquement en sommeil.

wakeUpSchedule

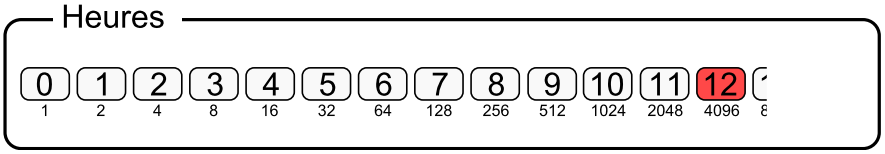
La fonction *wakeUpSchedule* permet de programmer une condition de réveil. Elle dispose de cinq variables qui permet de définir des correspondance sur les minutes, heure, jour de la semaine, jour dans le mois, et mois. Ces variables sont des variables entières dont chaque bit défini une correspondance. Schématiquement, chaque ensemble de minutes, jours, heures est représenté sous la forme d'un ensemble de case avec chacune un coefficient qui est une puissante de deux, exactement comme dans l'interface correspondante du YoctoHub-Wireless.

Par exemple le bit 0 des heures correspond à l'heure zéro, le bit 1 correspond à l'heure une, le bit 2 correspond à l'heure 2 etc.



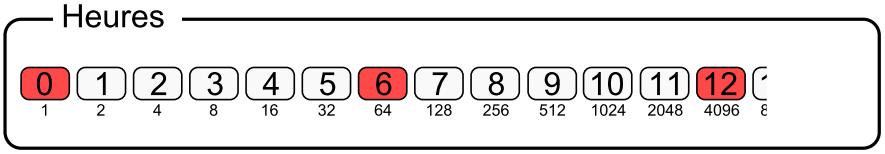
Chaque case se voit affecter une puissante de deux

Ainsi pour programmer le YoctoHub-Wireless pour qu'il se réveille tout les jours a midi, il mettre le bit 12 à 1, ce qui correspond à la valeur 2^12 = 4096.



Exemple de réveil à 12 H

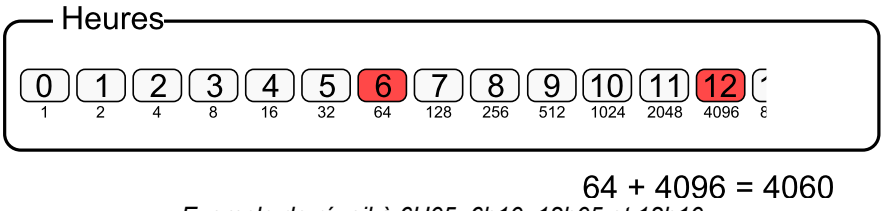
Pour que le module se reveille à 0 heure, 6 heures et 12 heures, il faut mettre les bit 0,6,et 12 à un, ce qui correspondant à la valeur 2^0+ 2^6 + 2^12 = 1 + 64 + 4096 = 4161



1 + 64 + 4096 = 4151

Exemple de réveil à 0, 6 et 12 H

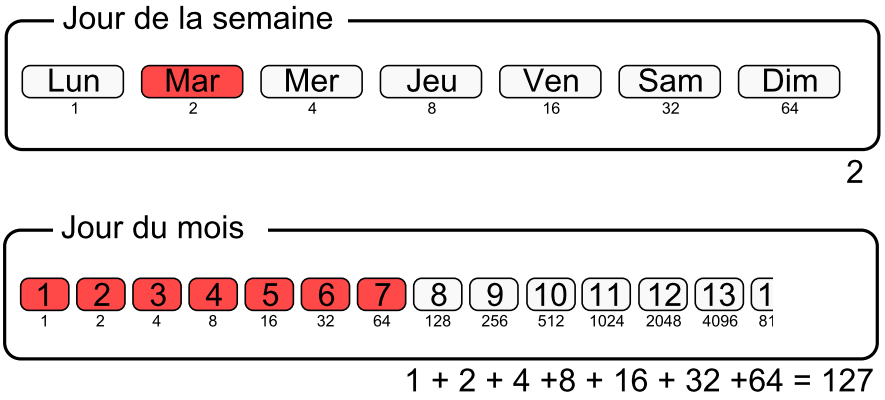
Les variables peuvent être combinées, pour qu'un réveil ait lieu tous les jours à 6H05, 6h10, 12h05 et 12h10 il suffit de mettre les heures à 2^6 + 2^12 = 4060 et les minutes à 2^5 + 2^10= 1056. Les variables laissée à zéro sont ignorées.



Exemple de réveil à 6H05, 6h10, 12h05 et 12h10

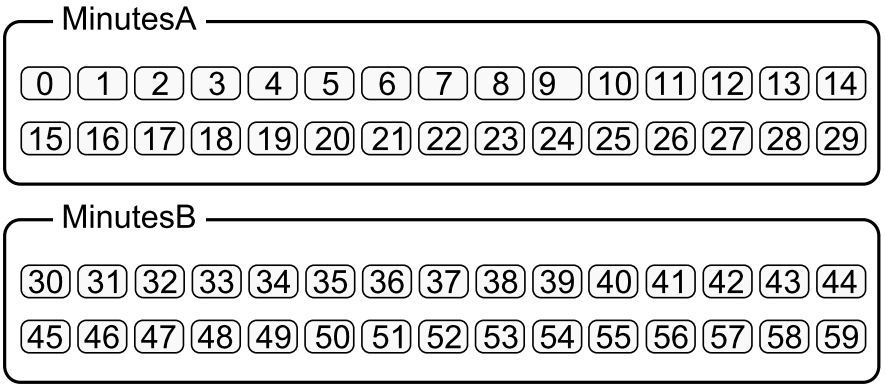
Notez que si vous désirez programmer un réveil à 6H05 et 12h10, mais pas 6h10 et 12h10, vous aurez besoin d'utiliser deux fonctions *wakeUpSchedule* différentes.

Ce paradigme permet de programmer des réveils assez complexe. Ainsi pour programmer un réveil tous les premiers mardis du mois, faut mettre à un le deuxième bit des jours de la semaine et les sept premiers bit des jours du mois.



Exemple de réveil tous les premiers mardi du mois

Certains languages de programmation, dont Javascript et Python, ne supportent pas les entiers 64 bits, ce qui pose un problème pour encoder les minutes. C'est pourquoi les minutes sont à la fois accessibles via un entier 64 bits *minutes* et deux entiers 32 bits, *minutesA* et *minutesB*, qui eux sont disponibles dans tous les languages actuels.



Les minutes sont aussi disponibles sous forme de deux entiers 32 bits.

La fonction *wakeUpSchedule* dispose d'une variable supplémentaire qui permet de définir le temps, en secondes, durant lequel le module restera éveillé après un réveil. Si cette variable est mise à zéro, le module restera éveillé.

Le YoctoHub-Wireless dispose de deux fonctions *wakeUpSchedule* ce qui permet de programmer jusqu'a deux types de réveils indépendants.

10. Personnalisation de l'interface Web

Votre YoctoHub-Wireless dispose d'un petit système de fichiers embarqué, qui permet de stocker des fichiers personnalisés utilisables par le YoctoHub-Wireless. Le système de fichiers se manipule grâce à la libraire *yocto_files*. Vous pourrez y stocker les fichiers de votre choix. Au besoin, vous pourrez y stocker une application Web permettant de gérer les modules connectés à votre YoctoHub-Wireless.

10.1. Utilisation

Utilisation interactive

L'interface Web du YoctoHub-Wireless fournit une interface sommaire pour manipuler le contenu du système de fichiers: cliquez simplement sur le bouton *configuration* correspondant à votre module dans l'interface du hub, puis sur le bouton *manage files*. Les fichiers présents sont listés, et vous pouvez les visualiser, les effacer ou en ajouter (téléchargement).

En raison de sa petite taille, le système de fichiers ne possède pas de notion explicite de répertoire. Vous pouvez toutefois utiliser la barre oblique "/" à l'intérieur des noms de fichiers pour les *classer* comme s'ils étaient dans des répertoires.

Utilisation programmée

Le système de fichiers s'utilise avec la librairie *yocto_files*. Les fonctions de bases sont disponibles:

- *upload* vous permet de créer un nouveau fichier sur le module, dont vous fournissez le contenu;
- *get_list* vous permet de connaître la liste de fichier présents sur le module, y compris la taille et le CRC32 du contenu;
- *download* vous permet de le récupérer dans une variable le contenu d'un fichier présent sur le module;
- *remove* permet d'effacer un fichier du module.
- *format* permet de réinitialiser le système de fichiers à un état vide, non fragmenté.

Un programme utilisant le système de fichier bien conçu devrait toujours commencer par s'assurer que les fichiers nécessaires à son fonctionnement sont présents sur le module, et si nécessaire les charger sur le module. Cela permet de gérer de manière transparente les mises à jour logicielles et le déploiement de l'application sur des nouveaux modules. Pour faciliter la détection des versions de fichiers présents sur le module, la méthode *get_list* retourne pour chaque fichier une signature sur 32 bit appelée CRC (Cyclic Redundancy Check), qui identifie de manière fiable le contenu du fichier. Ainsi, si le CRC du fichier correspond, il y a moins d'une chance sur 4 milliards que son contenu ne soit pas le bon. Vous pouvez même calculer dans votre programme par avance le CRC du contenu

que vous désirez, et ainsi le vérifier sans avoir à transférer le fichier. La fonction CRC utilisée par le système de fichiers Yoctopuce est la même que celle d'Ethernet, Gzip, PNG, etc. Sa valeur caractéristique pour la chaîne de neuf caractères "123456789" est 0xCBF43926.

Utilisation par HTTP

Les fichiers que vous avez chargés sur sur votre YoctoHub-Wireless sont accessibles par HTTP, à la racine du module (au même niveau que l'API REST). Cela permet de charger par exemple des pages d'interface HTML et Javascript personnalisées. Vous ne pouvez toutefois pas remplacer le contenu d'un fichier préchargé sur le module, mais seulement en ajouter des nouveaux.

Interfaces utilisateur et optimisation

Puisque que pouvez sauver des fichiers directement sur la mémoire flash du module et les accéder depuis l'extérieur, il est très facile de construire une application WEB pour contrôler les modules connectés au hub et de la stocker directement sur le hub. C'est un moyen très pratique pour construire des systèmes télécommandables depuis un smart-phone ou une tablette. Cependant le $PRODANME$ est plus limité qu'un serveur WEB normal: il n'accepte qu'un nombre limité de connexions en parallèle. La plupart des browsers WEB actuels ayant tendance à ouvrir un maximum de connexions en parallèle pour charger tous les éléments d'une page WEB, cela peut mener à des temps de chargement très long. Pour éviter cela, essayez de garder vos pages WEB aussi compactes que possible en incluant le code javascript et CSS directement dans la page. Si vous le pouvez, incluez aussi les images en base64.

10.2. Limitations

Le filesystem embarqué sur votre YoctoHub-Wireless a quelques limitations techniques:

- Son espace de stockage maximal est 3.5Mo, répartis en blocs permettant de stocker jusqu'à environ 800 fichiers
- L'effacement d'un fichier ne récupère pas nécessairement immédiatement la totalité de la place utilisée par le fichier. L'espace non libéré sera entièrement réutilisé si l'on recrée un fichier du même nom, mais pas forcément si l'on crée des fichiers utilisant chaque fois des noms différents. Pour cette raison, il n'est pas recommandé de générer automatiquement des fichiers avec des noms toujours différents.
- L'espace non libéré peut être entièrement récupéré avec la commande *format*, qui libère la totalité des fichiers.
- Chaque mise à jour du firmware provoque implicitement un formattage complet du filesystem.
- Comme toutes les mémoires flash, la mémoire utilisée pour stocker les fichiers a une durée de vie de 100'000 cycles d'effacement environ. C'est assez, mais ce n'est pas illimité. Prenez donc garde à ne pas écrire et effacer inutilement des fichiers en boucle très rapidement, sous peine de détruire votre module.

11. Référence de l'API de haut niveau

Ce chapitre résume les fonctions de l'API de haut niveau pour commander votre YoctoHub-Wireless. La syntaxe et les types précis peuvent varier d'un langage à l'autre mais, sauf avis contraire toutes sont disponibles dans chaque language. Pour une information plus précise sur les types des arguments et des valeurs de retour dans un langage donné, veuillez vous référer au fichier de définition pour ce langage (`yocto_api.*` ainsi que les autres fichiers `yocto_*` définissant les interfaces des fonctions).

Dans les langages qui supportent les exceptions, toutes ces fonctions vont par défaut générer des exceptions en cas d'erreur plutôt que de retourner la valeur d'erreur documentée pour chaque fonction, afin de faciliter le déboguage. Il est toutefois possible de désactiver l'utilisation d'exceptions à l'aide de la fonction `yDisableExceptions()`, si l'on préfère travailler avec des valeurs de retour d'erreur.

Ce chapitre ne reprend pas en détail les concepts de programmation des modules Yoctopuce. Vous trouverez des explications plus détaillées dans la documentation des modules que vous souhaitez raccorder à votre YoctoHub-Wireless.

11.1. Interface d'un port de Yocto-hub

Les objets YHubPort permettent de contrôler l'alimentation des ports d'un YoctoHub, ainsi que de détecter si un module y est raccordé et lequel. Un YHubPort reçoit toujours automatiquement comme nom logique le numéro de série unique du module Yoctopuce qui y est connecté.

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_hubport.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YHubPort = yoctolib.YHubPort;
php     require_once('yocto_hubport.php');
cpp     #include "yocto_hubport.h"
m       #import "yocto_hubport.h"
pas     uses yocto_hubport;
vb      yocto_hubport.vb
cs      yocto_hubport.cs
java    import com.yoctopuce.YoctoAPI.YHubPort;
py      from yocto_hubport import *
```

Fonction globales
yFindHubPort(**func**) Permet de retrouver un port de Yocto-hub d'après un identifiant donné.
yFirstHubPort() Commence l'énumération des port de Yocto-hub accessibles par la librairie.
Méthodes des objets YHubPort
hubport→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance du port de Yocto-hub au format TYPE(NAME)=SERIAL.FUNCTIONID.
hubport→get_advertisedValue() Retourne la valeur courante du port de Yocto-hub (pas plus de 6 caractères).
hubport→get_baudRate() Retourne la vitesse de transfert utilisée par le port de Yocto-hub, en kbps.
hubport→get_enabled() Retourne vrai si le port du Yocto-hub est alimenté, faux sinon.
hubport→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.
hubport→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.
hubport→get_friendlyName() Retourne un identifiant global du port de Yocto-hub au format NOM_MODULE.NOM_FONCTION.
hubport→get_functionDescriptor() Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
hubport→get_functionId() Retourne l'identifiant matériel du port de Yocto-hub, sans référence au module.
hubport→get_hardwareId() Retourne l'identifiant matériel unique du port de Yocto-hub au format SERIAL.FUNCTIONID.
hubport→get_logicalName()

Retourne le nom logique du port de Yocto-hub.

hubport→get_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

hubport→get_module_async(**callback**, **context**)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

hubport→get_portState()

Retourne l'état actuel du port de Yocto-hub.

hubport→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

hubport→isOnline()

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

hubport→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

hubport→load(**msValidity**)

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

hubport→load_async(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

hubport→nextHubPort()

Continue l'énumération des port de Yocto-hub commencée à l'aide de `yFirstHubPort()`.

hubport→registerValueCallback(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

hubport→set_enabled(**newval**)

Modifie le mode d'activation du port du Yocto-hub.

hubport→set_logicalName(**newval**)

Modifie le nom logique du port de Yocto-hub.

hubport→set_userData(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

hubport→wait_async(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YHubPort.FindHubPort() YHubPort yFindHubPort()

Permet de retrouver un port de Yocto-hub d'après un identifiant donné.

```
js      function yFindHubPort( func)
nodejs  function FindHubPort( func)
php     function yFindHubPort( $func)
cpp     YHubPort* yFindHubPort( const string& func)
m       +(YHubPort*) FindHubPort :(NSString*) func
pas     function yFindHubPort( func: string): TYHubPort
vb      function yFindHubPort( ByVal func As String) As YHubPort
cs      YHubPort FindHubPort( string func)
java    YHubPort FindHubPort( String func)
py      def FindHubPort( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le port de Yocto-hub soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YHubPort.isOnline()` pour tester si le port de Yocto-hub est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

func une chaîne de caractères qui référence le port de Yocto-hub sans ambiguïté

Retourne :

un objet de classe `YHubPort` qui permet ensuite de contrôler le port de Yocto-hub.

YHubPort.FirstHubPort() YHubPort yFirstHubPort()

Commence l'énumération des port de Yocto-hub accessibles par la librairie.

```
js      function yFirstHubPort( )
nodejs  function FirstHubPort( )
php     function yFirstHubPort( )
cpp     YHubPort* yFirstHubPort( )
m       +(YHubPort*) FirstHubPort
pas     function yFirstHubPort( ): TYHubPort
vb      function yFirstHubPort( ) As YHubPort
cs      YHubPort FirstHubPort( )
java    YHubPort FirstHubPort( )
py      def FirstHubPort( )
```

Utiliser la fonction `YHubPort.nextHubPort()` pour itérer sur les autres port de Yocto-hub.

Retourne :

un pointeur sur un objet `YHubPort`, correspondant au premier port de Yocto-hub accessible en ligne, ou `null` si il n'y a pas de port de Yocto-hub disponibles.

hubport→describe() YHubPort

Retourne un court texte décrivant de manière non-ambigüe l'instance du port de Yocto-hub au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant le port de Yocto-hub (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

hubport→get_advertisedValue() YHubPort
hubport→advertisedValue()

Retourne la valeur courante du port de Yocto-hub (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YHubPort target get_advertisedValue
```

Retourne :
une chaîne de caractères représentant la valeur courante du port de Yocto-hub (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

hubport→get_baudRate() YHubPort
hubport→baudRate()

Retourne la vitesse de transfert utilisée par le port de Yocto-hub, en kbps.

```
js      function get_baudRate( )
nodejs  function get_baudRate( )
php     function get_baudRate( )
cpp     int get_baudRate( )
m       -(int) baudRate
pas     function get_baudRate( ): LongInt
vb      function get_baudRate( ) As Integer
cs      int get_baudRate( )
java    int get_baudRate( )
py      def get_baudRate( )
cmd     YHubPort target get_baudRate
```

La valeur par défaut est 1000 kbps, une valeur inférieure révèle des problèmes de communication.

Retourne :
un entier représentant la vitesse de transfert utilisée par le port de Yocto-hub, en kbps

En cas d'erreur, déclenche une exception ou retourne `Y_BAUDRATE_INVALID`.

hubport→get_enabled() YHubPort
hubport→enabled()

Retourne vrai si le port du Yocto-hub est alimenté, faux sinon.

```
js      function get_enabled( )
nodejs  function get_enabled( )
php     function get_enabled( )
cpp     Y_ENABLED_enum get_enabled( )
m       -(Y_ENABLED_enum) enabled
pas     function get_enabled( ): Integer
vb      function get_enabled( ) As Integer
cs      int get_enabled( )
java    int get_enabled( )
py      def get_enabled( )
cmd     YHubPort target get_enabled
```

Retourne :

soit Y_ENABLED_FALSE, soit Y_ENABLED_TRUE, selon vrai si le port du Yocto-hub est alimenté, faux sinon

En cas d'erreur, déclenche une exception ou retourne Y_ENABLED_INVALID.

hubport→get_errorMessage() YHubPort
hubport→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

```
js      function get_errorMessage( )
nodejs  function get_errorMessage( )
php     function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     function get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      def get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du port de Yocto-hub.

hubport→get_errorType() YHubPort
hubport→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du port de Yocto-hub.

hubport→get_friendlyName() YHubPort
hubport→friendlyName()

Retourne un identifiant global du port de Yocto-hub au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et du port de Yocto-hub si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du port de Yocto-hub (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant le port de Yocto-hub en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

hubport→get_functionDescriptor() YHubPort
hubport→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
nodejs  function get_functionDescriptor( )
php     function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     function get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      def get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :

un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

hubport→get_functionId() YHubPort
hubport→functionId()

Retourne l'identifiant matériel du port de Yocto-hub, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

Retourne :

une chaîne de caractères identifiant le port de Yocto-hub (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

hubport→get_hardwareId() YHubPort
hubport→hardwareId()

Retourne l'identifiant matériel unique du port de Yocto-hub au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
nodejs  function get_hardwareId( )
php     function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
java    String get_hardwareId( )
py      def get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du port de Yocto-hub (par example RELAYLO1-123456.relay1).

Retourne :

une chaîne de caractères identifiant le port de Yocto-hub (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

hubport→get_logicalName() YHubPort
hubport→logicalName()

Retourne le nom logique du port de Yocto-hub.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YHubPort target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique du port de Yocto-hub.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

hubport→get_module() YHubPort
hubport→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
nodejs  function get_module( )
php     function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     function get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
java    YModule get_module( )
py      def get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

hubport→get_module_async() YHubPort
hubport→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module_async( callback, context)
nodejs  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

hubport→get_portState() YHubPort
hubport→portState()

Retourne l'état actuel du port de Yocto-hub.

```
js      function get_portState( )
nodejs  function get_portState( )
php     function get_portState( )
cpp     Y_PORTSTATE_enum get_portState( )
m       -(Y_PORTSTATE_enum) portState
pas     function get_portState( ): Integer
vb      function get_portState( ) As Integer
cs      int get_portState( )
java    int get_portState( )
py      def get_portState( )
cmd     YHubPort target get_portState
```

Retourne :

une valeur parmi `Y_PORTSTATE_OFF`, `Y_PORTSTATE_OVRLD`, `Y_PORTSTATE_ON`, `Y_PORTSTATE_RUN` et `Y_PORTSTATE_PROG` représentant l'état actuel du port de Yocto-hub

En cas d'erreur, déclenche une exception ou retourne `Y_PORTSTATE_INVALID`.

hubport→get_userData() YHubPort
hubport→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function **get_userData**( )
nodejs	function **get_userData**( )
php	function **get_userData**( )
cpp	void * **get_userData**( )
m	-(void*) userData
pas	function **get_userData**( ): Tobject
vb	function **get_userData**( ) As Object
cs	object **get_userData**( )
java	Object **get_userData**( )
py	def **get_userData**( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

hubport→isOnline() YHubPort

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache du port de Yocto-hub sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :

`true` si le port de Yocto-hub est joignable, `false` sinon

hubport→isOnline_async() YHubPort

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du port de Yocto-hub sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

hubport→load() YHubPort

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

hubport→load_async() YHubPort

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

hubport→nextHubPort() YHubPort

Continue l'énumération des port de Yocto-hub commencée à l'aide de `yFirstHubPort()`.

```
js      function nextHubPort( )
nodejs  function nextHubPort( )
php     function nextHubPort( )
cpp     YHubPort * nextHubPort( )
m       -(YHubPort*) nextHubPort
pas     function nextHubPort( ): TYHubPort
vb      function nextHubPort( ) As YHubPort
cs      YHubPort nextHubPort( )
java    YHubPort nextHubPort( )
py      def nextHubPort( )
```

Retourne :

un pointeur sur un objet `YHubPort` accessible en ligne, ou `null` lorsque l'énumération est terminée.

hubport→registerValueCallback() YHubPort

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function **registerValueCallback**(**callback**)
nodejs	function **registerValueCallback**(**callback**)
php	function **registerValueCallback**(**$callback**)
cpp	int **registerValueCallback**(YHubPortValueCallback **callback**)
m	-(int) **registerValueCallback** : (YHubPortValueCallback) **callback**
pas	function **registerValueCallback**(**callback**: TYHubPortValueCallback): LongInt
vb	function **registerValueCallback**( ) As Integer
cs	int **registerValueCallback**(ValueCallback **callback**)
java	int **registerValueCallback**(UpdateCallback **callback**)
py	def **registerValueCallback**(**callback**)

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

hubport→set_enabled() YHubPort
hubport→setEnabled()

Modifie le mode d'activation du port du Yocto-hub.

```
js      function set_enabled( newval)
nodejs  function set_enabled( newval)
php     function set_enabled( $newval)
cpp     int set_enabled( Y_ENABLED_enum newval)
m       -(int) setEnabled : (Y_ENABLED_enum) newval
pas     function set_enabled( newval: Integer): integer
vb      function set_enabled( ByVal newval As Integer) As Integer
cs      int set_enabled( int newval)
java    int set_enabled( int newval)
py      def set_enabled( newval)
cmd     YHubPort target set_enabled newval
```

Si le port est actif, il sera alimenté. Sinon, l'alimentation du module est coupée.

Paramètres :

newval soit Y_ENABLED_FALSE, soit Y_ENABLED_TRUE, selon le mode d'activation du port du Yocto-hub

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

hubport→set_logicalName() YHubPort
hubport→setLogicalName()

Modifie le nom logique du port de Yocto-hub.

```
js      function set_logicalName( newval)
nodejs  function set_logicalName( newval)
php     function set_logicalName( $newval)
cpp     int set_logicalName( const string& newval)
m       -(int) setLogicalName : (NSString*) newval
pas     function set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
java    int set_logicalName( String newval)
py      def set_logicalName( newval)
cmd     YHubPort target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique du port de Yocto-hub.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

hubport→set_userData() YHubPort
hubport→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

hubport→wait_async() YHubPort

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

11.2. Interface de la fonction Wireless

La fonction YWireless permet de configurer et de contrôler la configuration du réseau sans fil sur les modules Yoctopuce qui en sont dotés.

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_wireless.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YWireless = yoctolib.YWireless;
php     require_once('yocto_wireless.php');
cpp     #include "yocto_wireless.h"
m       #import "yocto_wireless.h"
pas     uses yocto_wireless;
vb      yocto_wireless.vb
cs      yocto_wireless.cs
java    import com.yoctopuce.YoctoAPI.YWireless;
py      from yocto_wireless import *
```

Fonction globales
yFindWireless(**func**) Permet de retrouver une interface réseau sans fil d'après un identifiant donné.
yFirstWireless() Commence l'énumération des interfaces réseau sans fil accessibles par la librairie.
Méthodes des objets `YWireless`
wireless→adhocNetwork(**ssid**, **securityKey**) Modifie la configuration de l'interface réseau sans fil pour créer un réseau sans fil sans point d'accès, en mode "ad-hoc".
wireless→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau sans fil au format `TYPE(NAME)=SERIAL.FUNCTIONID`.
wireless→get_advertisedValue() Retourne la valeur courante de l'interface réseau sans fil (pas plus de 6 caractères).
wireless→get_channel() Retourne le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé.
wireless→get_detectedWlans() Retourne une liste d'objets objet YFileRecord qui décrivent les réseaux sans fils détectés.
wireless→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.
wireless→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.
wireless→get_friendlyName() Retourne un identifiant global de l'interface réseau sans fil au format `NOM_MODULE.NOM_FONCTION`.
wireless→get_functionDescriptor() Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.
wireless→get_functionId() Retourne l'identifiant matériel de l'interface réseau sans fil, sans référence au module.
wireless→get_hardwareId()

Retourne l'identifiant matériel unique de l'interface réseau sans fil au format SERIAL.FUNCTIONID.

wireless→get_linkQuality()

Retourne la qualité de la connection, exprimée en pourcents.

wireless→get_logicalName()

Retourne le nom logique de l'interface réseau sans fil.

wireless→get_message()

Retourne le dernier message de diagnostique de l'interface au réseau sans fil.

wireless→get_module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

wireless→get_module_async(**callback**, **context**)

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

wireless→get_security()

Retourne l'algorithme de sécurité utilisé par le réseau sans-fil sélectionné.

wireless→get_ssid()

Retourne le nom (SSID) du réseau sans-fil sélectionné.

wireless→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.

wireless→isOnline()

Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.

wireless→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.

wireless→joinNetwork(**ssid**, **securityKey**)

Modifie la configuration de l'interface réseau sans fil pour se connecter à un point d'accès sans fil existant (mode "infrastructure").

wireless→load(**msValidity**)

Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.

wireless→load_async(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.

wireless→nextWireless()

Continue l'énumération des interfaces réseau sans fil commencée à l'aide de yFirstWireless().

wireless→registerValueCallback(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

wireless→set_logicalName(**newval**)

Modifie le nom logique de l'interface réseau sans fil.

wireless→set_userData(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get_userData.

wireless→wait_async(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YWireless.FindWireless() YWireless
yFindWireless()

Permet de retrouver une interface réseau sans fil d'après un identifiant donné.

```
js      function yFindWireless( func)
nodejs  function FindWireless( func)
php     function yFindWireless( $func)
cpp     YWireless* yFindWireless( string func)
m       +(YWireless*) FindWireless : (NSString*) func
pas     function yFindWireless( func: string): TYWireless
vb      function yFindWireless( ByVal func As String) As YWireless
cs      YWireless FindWireless( string func)
java    YWireless FindWireless( String func)
py      def FindWireless( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface réseau sans fil soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWireless.isOnline()` pour tester si l'interface réseau sans fil est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

func une chaîne de caractères qui référence l'interface réseau sans fil sans ambiguïté

Retourne :

un objet de classe `YWireless` qui permet ensuite de contrôler l'interface réseau sans fil.

YWireless.FirstWireless() YWireless
yFirstWireless()

Commence l'énumération des interfaces réseau sans fil accessibles par la librairie.

```
js      function yFirstWireless( )
nodejs  function FirstWireless( )
php     function yFirstWireless( )
cpp     YWireless* yFirstWireless( )
m       +(YWireless*) FirstWireless
pas     function yFirstWireless( ): TYWireless
vb      function yFirstWireless( ) As YWireless
cs      YWireless FirstWireless( )
java    YWireless FirstWireless( )
py      def FirstWireless( )
```

Utiliser la fonction `YWireless.nextWireless()` pour itérer sur les autres interfaces réseau sans fil.

Retourne :

un pointeur sur un objet `YWireless`, correspondant à la première interface réseau sans fil accessible en ligne, ou `null` si il n'y a pas de interfaces réseau sans fil disponibles.

wireless→adhocNetwork() YWireless

Modifie la configuration de l'interface réseau sans fil pour créer un réseau sans fil sans point d'accès, en mode "ad-hoc".

```
js      function adhocNetwork( ssid, securityKey)
nodejs  function adhocNetwork( ssid, securityKey)
php     function adhocNetwork( $ssid, $securityKey)
cpp     int adhocNetwork( string ssid, string securityKey)
m       -(int) adhocNetwork : (NSString*) ssid
                            : (NSString*) securityKey
pas     function adhocNetwork( ssid: string, securityKey: string): integer
vb      function adhocNetwork( ByVal ssid As String,
                               ByVal securityKey As String) As Integer
cs      int adhocNetwork( string ssid, string securityKey)
java    int adhocNetwork( String ssid, String securityKey)
py      def adhocNetwork( ssid, securityKey)
cmd     YWireless target adhocNetwork ssid securityKey
```

Si une clef d'accès est spécifiée, le réseau sera protégé par une sécurité WEP128 (l'utilisation de WPA n'est pas standardisée en mode ad-hoc). N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

ssid	nom du réseau sans fil à créer
securityKey	clé d'accès de réseau, sous forme de chaîne de caractères

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wireless→describe() YWireless

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau sans fil au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant l'interface réseau sans fil (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

wireless→get_advertisedValue() YWireless
wireless→advertisedValue()

Retourne la valeur courante de l'interface réseau sans fil (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YWireless target get_advertisedValue
```

Retourne :

une chaîne de caractères représentant la valeur courante de l'interface réseau sans fil (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

wireless→get_channel() YWireless
wireless→channel()

Retourne le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé.

```
js      function get_channel( )
nodejs  function get_channel( )
php     function get_channel( )
cpp     int get_channel( )
m       -(int) channel
pas     function get_channel( ): LongInt
vb      function get_channel( ) As Integer
cs      int get_channel( )
java    int get_channel( )
py      def get_channel( )
cmd     YWireless target get_channel
```

Retourne :

un entier représentant le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé

En cas d'erreur, déclenche une exception ou retourne `Y_CHANNEL_INVALID`.

wireless→get_detectedWlans() YWireless
wireless→detectedWlans()

Retourne une liste d'objets objet YFileRecord qui décrivent les réseaux sans fils détectés.

```
js      function get_detectedWlans( )
nodejs  function get_detectedWlans( )
php     function get_detectedWlans( )
cpp     vector<YWlanRecord> get_detectedWlans( )
m       -(NSMutableArray*) detectedWlans
pas     function get_detectedWlans( ): TYWlanRecordArray
vb      function get_detectedWlans( ) As List
cs      List<YWlanRecord> get_detectedWlans( )
java    ArrayList<YWlanRecord> get_detectedWlans( )
py      def get_detectedWlans( )
cmd     YWireless target get_detectedWlans
```

La liste n'est pas mise à jour quand le module est déjà connecté à un accès sans fil (mode "infrastructure"). Pour forcer la détection des réseaux sans fil, il faut appeler `addhocNetwork()` pour se déconnecter du réseau actuel. L'appelant est responsable de la désallocation de la liste retournée.

Retourne :

une liste d'objets `YWlanRecord`, contenant le SSID, le canal, la qualité du signal, et l'algorithme de sécurité utilisé par le réseau sans-fil

En cas d'erreur, déclenche une exception ou retourne une liste vide.

wireless→get_errorMessage() YWireless
wireless→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.

```
js      function get_errorMessage( )
nodejs  function get_errorMessage( )
php     function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     function get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      def get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau sans fil.

wireless→get_errorType() YWireless
wireless→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau sans fil.

wireless→get_friendlyName() YWireless
wireless→friendlyName()

Retourne un identifiant global de l'interface réseau sans fil au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et de l'interface réseau sans fil si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de l'interface réseau sans fil (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant l'interface réseau sans fil en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

wireless→get_functionDescriptor() YWireless
wireless→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
nodejs  function get_functionDescriptor( )
php     function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     function get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      def get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :

un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

wireless→get_functionId() YWireless
wireless→functionId()

Retourne l'identifiant matériel de l'interface réseau sans fil, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

Retourne :

une chaîne de caractères identifiant l'interface réseau sans fil (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

wireless→get_hardwareId() YWireless
wireless→hardwareId()

Retourne l'identifiant matériel unique de l'interface réseau sans fil au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
nodejs  function get_hardwareId( )
php     function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
java    String get_hardwareId( )
py      def get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface réseau sans fil (par example RELAYL01-123456.relay1).

Retourne :

une chaîne de caractères identifiant l'interface réseau sans fil (ex: RELAYL01-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

wireless→get_linkQuality() YWireless
wireless→linkQuality()

Retourne la qualité de la connection, exprimée en pourcents.

```
js      function get_linkQuality( )
nodejs  function get_linkQuality( )
php     function get_linkQuality( )
cpp     int get_linkQuality( )
m       -(int) linkQuality
pas     function get_linkQuality( ): LongInt
vb      function get_linkQuality( ) As Integer
cs      int get_linkQuality( )
java    int get_linkQuality( )
py      def get_linkQuality( )
cmd     YWireless target get_linkQuality
```

Retourne :
un entier représentant la qualité de la connection, exprimée en pourcents

En cas d'erreur, déclenche une exception ou retourne `Y_LINKQUALITY_INVALID`.

wireless→get_logicalName() YWireless
wireless→logicalName()

Retourne le nom logique de l'interface réseau sans fil.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YWireless target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique de l'interface réseau sans fil.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

wireless→get_message() YWireless
wireless→message()

Retourne le dernier message de diagnostique de l'interface au réseau sans fil.

```
js      function get_message( )
nodejs  function get_message( )
php     function get_message( )
cpp     string get_message( )
m       -(NSString*) message
pas     function get_message( ): string
vb      function get_message( ) As String
cs      string get_message( )
java    String get_message( )
py      def get_message( )
cmd     YWireless target get_message
```

Retourne :
une chaîne de caractères représentant le dernier message de diagnostique de l'interface au réseau sans fil

En cas d'erreur, déclenche une exception ou retourne `Y_MESSAGE_INVALID`.

wireless→get_module() YWireless
wireless→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
nodejs  function get_module( )
php     function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     function get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
java    YModule get_module( )
py      def get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

wireless→get_module_async() YWireless
wireless→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module_async( callback, context)
nodejs  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wireless→get_security() YWireless
wireless→security()

Retourne l'algorithme de sécurité utilisé par le réseau sans-fil sélectionné.

```
js      function get_security( )
nodejs  function get_security( )
php     function get_security( )
cpp     Y_SECURITY_enum get_security( )
m       -(Y_SECURITY_enum) security
pas     function get_security( ): Integer
vb      function get_security( ) As Integer
cs      int get_security( )
java    int get_security( )
py      def get_security( )
cmd     YWireless target get_security
```

Retourne :

une valeur parmi `Y_SECURITY_UNKNOWN`, `Y_SECURITY_OPEN`, `Y_SECURITY_WEP`, `Y_SECURITY_WPA` et `Y_SECURITY_WPA2` représentant l'algorithme de sécurité utilisé par le réseau sans-fil sélectionné

En cas d'erreur, déclenche une exception ou retourne `Y_SECURITY_INVALID`.

wireless→get_ssid() YWireless
wireless→ssid()

Retourne le nom (SSID) du réseau sans-fil sélectionné.

```
js      function get_ssid( )
nodejs  function get_ssid( )
php     function get_ssid( )
cpp     string get_ssid( )
m       -(NSString*) ssid
pas     function get_ssid( ): string
vb      function get_ssid( ) As String
cs      string get_ssid( )
java    String get_ssid( )
py      def get_ssid( )
cmd     YWireless target get_ssid
```

Retourne :
une chaîne de caractères représentant le nom (SSID) du réseau sans-fil sélectionné

En cas d'erreur, déclenche une exception ou retourne `Y_SSID_INVALID`.

wireless→get_userData() YWireless
wireless→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
nodejs  function get_userData( )
php     function get_userData( )
cpp     void * get_userData( )
m       -(void*) userData
pas     function get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      def get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

wireless→isOnline() YWireless

Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache de l'interface réseau sans fil sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :

`true` si l'interface réseau sans fil est joignable, `false` sinon

wireless→isOnline_async() YWireless

Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'interface réseau sans fil sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wireless→joinNetwork() YWireless

Modifie la configuration de l'interface réseau sans fil pour se connecter à un point d'accès sans fil existant (mode "infrastructure").

```
js      function joinNetwork( ssid, securityKey)
nodejs  function joinNetwork( ssid, securityKey)
php     function joinNetwork( $ssid, $securityKey)
cpp     int joinNetwork( string ssid, string securityKey)
m       -(int) joinNetwork : (NSString*) ssid
                           : (NSString*) securityKey
pas     function joinNetwork( ssid: string, securityKey: string): integer
vb      function joinNetwork( ByVal ssid As String,
                              ByVal securityKey As String) As Integer
cs      int joinNetwork( string ssid, string securityKey)
java    int joinNetwork( String ssid, String securityKey)
py      def joinNetwork( ssid, securityKey)
cmd     YWireless target joinNetwork ssid securityKey
```

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

ssid	nom du réseau sans fil à utiliser
securityKey	clé d'accès au réseau, sous forme de chaîne de caractères

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wireless→load() YWireless

Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wireless→load_async() YWireless

Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wireless→nextWireless() YWireless

Continue l'énumération des interfaces réseau sans fil commencée à l'aide de `yFirstWireless()`.

```
js      function nextWireless( )
nodejs  function nextWireless( )
php     function nextWireless( )
cpp     YWireless * nextWireless( )
m       -(YWireless*) nextWireless
pas     function nextWireless( ): TYWireless
vb      function nextWireless( ) As YWireless
cs      YWireless nextWireless( )
java    YWireless nextWireless( )
py      def nextWireless( )
```

Retourne :

un pointeur sur un objet `YWireless` accessible en ligne, ou `null` lorsque l'énumération est terminée.

wireless→registerValueCallback() YWireless

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
nodejs  function registerValueCallback( callback)
php     function registerValueCallback( $callback)
cpp     int registerValueCallback( YWirelessValueCallback callback)
m       -(int) registerValueCallback : (YWirelessValueCallback) callback
pas     function registerValueCallback( callback: TYWirelessValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
py      def registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

wireless→set_logicalName() YWireless
wireless→setLogicalName()

Modifie le nom logique de l'interface réseau sans fil.

```
js      function set_logicalName( newval)
nodejs  function set_logicalName( newval)
php     function set_logicalName( $newval)
cpp     int set_logicalName( const string& newval)
m       -(int) setLogicalName : (NSString*) newval
pas     function set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
java    int set_logicalName( String newval)
py      def set_logicalName( newval)
cmd     YWireless target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique de l'interface réseau sans fil.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wireless→set_userData() YWireless
wireless→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

wireless→wait_async() YWireless

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

11.3. Interface de la fonction Network

Les objets YNetwork permettent de contrôler les paramètres TCP/IP des modules Yoctopuce dotés d'une interface réseau.

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_network.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YNetwork = yoctolib.YNetwork;
php     require_once('yocto_network.php');
cpp     #include "yocto_network.h"
m       #import "yocto_network.h"
pas     uses yocto_network;
vb      yocto_network.vb
cs      yocto_network.cs
java    import com.yoctopuce.YoctoAPI.YNetwork;
py      from yocto_network import *
```

Fonction globales
yFindNetwork(**func**) Permet de retrouver une interface réseau d'après un identifiant donné.
yFirstNetwork() Commence l'énumération des interfaces réseau accessibles par la librairie.
Méthodes des objets `YNetwork`
network→callbackLogin(**username**, **password**) Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter.
network→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau au format `TYPE(NAME)=SERIAL.FUNCTIONID`.
network→get_adminPassword() Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.
network→get_advertisedValue() Retourne la valeur courante de l'interface réseau (pas plus de 6 caractères).
network→get_callbackCredentials() Retourne une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.
network→get_callbackEncoding() Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback.
network→get_callbackMaxDelay() Retourne l'attente maximale entre deux notifications par callback, en secondes.
network→get_callbackMethod() Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback.
network→get_callbackMinDelay() Retourne l'attente minimale entre deux notifications par callback, en secondes.
network→get_callbackUrl() Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs.
network→get_discoverable()

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

network→get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

network→get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

network→get_friendlyName()

Retourne un identifiant global de l'interface réseau au format `NOM_MODULE.NOM_FONCTION`.

network→get_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

network→get_functionId()

Retourne l'identifiant matériel de l'interface réseau, sans référence au module.

network→get_hardwareId()

Retourne l'identifiant matériel unique de l'interface réseau au format `SERIAL.FUNCTIONID`.

network→get_ipAddress()

Retourne l'adresse IP utilisée par le module Yoctopuce.

network→get_logicalName()

Retourne le nom logique de l'interface réseau.

network→get_macAddress()

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

network→get_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

network→get_module_async(**callback**, **context**)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

network→get_poeCurrent()

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

network→get_primaryDNS()

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

network→get_readiness()

Retourne l'état de fonctionnement atteint par l'interface réseau.

network→get_router()

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

network→get_secondaryDNS()

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

network→get_subnetMask()

Retourne le masque de sous-réseau utilisé par le module.

network→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

network→get_userPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

network→get_wwwWatchdogDelay()

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

network→isOnline()

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

network→isOnline_async(callback, context)

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

network→load(msValidity)

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

network→load_async(msValidity, callback, context)

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

network→nextNetwork()

Continue l'énumération des interfaces réseau commencée à l'aide de `yFirstNetwork()`.

network→ping(host)

Ping str_host pour vérifier la connexion réseau.

network→registerValueCallback(callback)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

network→set_adminPassword(newval)

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

network→set_callbackCredentials(newval)

Modifie le laisser-passer pour se connecter à l'adresse de callback.

network→set_callbackEncoding(newval)

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

network→set_callbackMaxDelay(newval)

Modifie l'attente maximale entre deux notifications par callback, en secondes.

network→set_callbackMethod(newval)

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

network→set_callbackMinDelay(newval)

Modifie l'attente minimale entre deux notifications par callback, en secondes.

network→set_callbackUrl(newval)

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

network→set_discoverable(newval)

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

network→set_logicalName(newval)

Modifie le nom logique de l'interface réseau.

network→set_primaryDNS(newval)

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

network→set_secondaryDNS(newval)

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

network→set_userData(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

network→set_userPassword(newval)

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

network→set_wwwWatchdogDelay(newval)

Modifie la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

network→useDHCP(fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

network→useStaticIP(**ipAddress**, **subnetMaskLen**, **router**)

Modifie la configuration de l'interface réseau pour utiliser une adresse IP assignée manuellement (adresse IP statique).

network→wait_async(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YNetwork.FindNetwork() YNetwork yFindNetwork()

Permet de retrouver une interface réseau d'après un identifiant donné.

```
js      function yFindNetwork( func)
nodejs  function FindNetwork( func)
php     function yFindNetwork( $func)
cpp     YNetwork* yFindNetwork( const string& func)
m       +(YNetwork*) FindNetwork :(NSString*) func
pas     function yFindNetwork( func: string): TYNetwork
vb      function yFindNetwork( ByVal func As String) As YNetwork
cs      YNetwork FindNetwork( string func)
java    YNetwork FindNetwork( String func)
py      def FindNetwork( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface réseau soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YNetwork.isOnline()` pour tester si l'interface réseau est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

func une chaîne de caractères qui référence l'interface réseau sans ambiguïté

Retourne :

un objet de classe `YNetwork` qui permet ensuite de contrôler l'interface réseau.

YNetwork.FirstNetwork() YNetwork
yFirstNetwork()

Commence l'énumération des interfaces réseau accessibles par la librairie.

```
js      function yFirstNetwork( )
nodejs  function FirstNetwork( )
php     function yFirstNetwork( )
cpp     YNetwork* yFirstNetwork( )
m       +(YNetwork*) FirstNetwork
pas     function yFirstNetwork( ): TYNetwork
vb      function yFirstNetwork( ) As YNetwork
cs      YNetwork FirstNetwork( )
java    YNetwork FirstNetwork( )
py      def FirstNetwork( )
```

Utiliser la fonction `YNetwork.nextNetwork()` pour itérer sur les autres interfaces réseau.

Retourne :

un pointeur sur un objet `YNetwork`, correspondant à la première interface réseau accessible en ligne, ou `null` si il n'y a pas de interfaces réseau disponibles.

network→callbackLogin() YNetwork

Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter.

```
js      function callbackLogin( username, password)
nodejs  function callbackLogin( username, password)
php     function callbackLogin( $username, $password)
cpp     int callbackLogin( string username, string password)
m       -(int) callbackLogin : (NSString*) username : (NSString*) password
pas     function callbackLogin( username: string, password: string): integer
vb      function callbackLogin( ByVal username As String,
                                 ByVal password As String) As Integer
cs      int callbackLogin( string username, string password)
java    int callbackLogin( String username, String password)
py      def callbackLogin( username, password)
cmd     YNetwork target callbackLogin username password
```

Le mot de passe ne sera pas stocké dans le module, mais seulement une version hashée non réversible. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

username nom d'utilisateur pour s'identifier au callback

password mot de passe pour s'identifier au callback

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→describe() YNetwork

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant l'interface réseau (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

network→get_adminPassword() YNetwork
network→adminPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.

```
js      function get_adminPassword( )
nodejs  function get_adminPassword( )
php     function get_adminPassword( )
cpp     string get_adminPassword( )
m       -(NSString*) adminPassword
pas     function get_adminPassword( ): string
vb      function get_adminPassword( ) As String
cs      string get_adminPassword( )
java    String get_adminPassword( )
py      def get_adminPassword( )
cmd     YNetwork target get_adminPassword
```

Retourne :

une chaîne de caractères représentant une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_ADMINPASSWORD_INVALID`.

network→get_advertisedValue() YNetwork
network→advertisedValue()

Retourne la valeur courante de l'interface réseau (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YNetwork target get_advertisedValue
```

Retourne :
une chaîne de caractères représentant la valeur courante de l'interface réseau (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

network→get_callbackCredentials() YNetwork
network→callbackCredentials()

Retourne une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.

```
js      function get_callbackCredentials( )
nodejs  function get_callbackCredentials( )
php     function get_callbackCredentials( )
cpp     string get_callbackCredentials( )
m       -(NSString*) callbackCredentials
pas     function get_callbackCredentials( ): string
vb      function get_callbackCredentials( ) As String
cs      string get_callbackCredentials( )
java    String get_callbackCredentials( )
py      def get_callbackCredentials( )
cmd     YNetwork target get_callbackCredentials
```

Retourne :

une chaîne de caractères représentant une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKCREDENTIALS_INVALID`.

network→get_callbackEncoding() YNetwork
network→callbackEncoding()

Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback.

js	function **get_callbackEncoding**( )
nodejs	function **get_callbackEncoding**( )
php	function **get_callbackEncoding**( )
cpp	Y_CALLBACKENCODING_enum **get_callbackEncoding**( )
m	-(Y_CALLBACKENCODING_enum) callbackEncoding
pas	function **get_callbackEncoding**( ): Integer
vb	function **get_callbackEncoding**( ) As Integer
cs	int **get_callbackEncoding**( )
java	int **get_callbackEncoding**( )
py	def **get_callbackEncoding**( )
cmd	YNetwork **target get_callbackEncoding**

Retourne :

une valeur parmi `Y_CALLBACKENCODING_FORM`, `Y_CALLBACKENCODING_JSON`, `Y_CALLBACKENCODING_JSON_ARRAY`, `Y_CALLBACKENCODING_CSV` et `Y_CALLBACKENCODING_YOCTO_API` représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKENCODING_INVALID`.

network→get_callbackMaxDelay() YNetwork
network→callbackMaxDelay()

Retourne l'attente maximale entre deux notifications par callback, en secondes.

```
js      function get_callbackMaxDelay( )
nodejs  function get_callbackMaxDelay( )
php     function get_callbackMaxDelay( )
cpp     int get_callbackMaxDelay( )
m       -(int) callbackMaxDelay
pas     function get_callbackMaxDelay( ): LongInt
vb      function get_callbackMaxDelay( ) As Integer
cs      int get_callbackMaxDelay( )
java    int get_callbackMaxDelay( )
py      def get_callbackMaxDelay( )
cmd     YNetwork target get_callbackMaxDelay
```

Retourne :

un entier représentant l'attente maximale entre deux notifications par callback, en secondes

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKMAXDELAY_INVALID`.

network→get_callbackMethod() YNetwork
network→callbackMethod()

Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback.

```
js      function get_callbackMethod( )
nodejs  function get_callbackMethod( )
php     function get_callbackMethod( )
cpp     Y_CALLBACKMETHOD_enum get_callbackMethod( )
m       -(Y_CALLBACKMETHOD_enum) callbackMethod
pas     function get_callbackMethod( ): Integer
vb      function get_callbackMethod( ) As Integer
cs      int get_callbackMethod( )
java    int get_callbackMethod( )
py      def get_callbackMethod( )
cmd     YNetwork target get_callbackMethod
```

Retourne :

une valeur parmi `Y_CALLBACKMETHOD_POST`, `Y_CALLBACKMETHOD_GET` et `Y_CALLBACKMETHOD_PUT` représentant la méthode HTTP à utiliser pour signaler les changements d'état par callback

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKMETHOD_INVALID`.

network→get_callbackMinDelay() YNetwork
network→callbackMinDelay()

Retourne l'attente minimale entre deux notifications par callback, en secondes.

js	function **get_callbackMinDelay**( )
nodejs	function **get_callbackMinDelay**( )
php	function **get_callbackMinDelay**( )
cpp	int **get_callbackMinDelay**( )
m	-(int) callbackMinDelay
pas	function **get_callbackMinDelay**( ): LongInt
vb	function **get_callbackMinDelay**( ) As Integer
cs	int **get_callbackMinDelay**( )
java	int **get_callbackMinDelay**( )
py	def **get_callbackMinDelay**( )
cmd	YNetwork **target** **get_callbackMinDelay**

Retourne :
un entier représentant l'attente minimale entre deux notifications par callback, en secondes

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKMINDELAY_INVALID`.

network→get_callbackUrl() YNetwork
network→callbackUrl()

Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

```
js      function get_callbackUrl( )
nodejs  function get_callbackUrl( )
php     function get_callbackUrl( )
cpp     string get_callbackUrl( )
m       -(NSString*) callbackUrl
pas     function get_callbackUrl( ): string
vb      function get_callbackUrl( ) As String
cs      string get_callbackUrl( )
java    String get_callbackUrl( )
py      def get_callbackUrl( )
cmd     YNetwork target get_callbackUrl
```

Retourne :

une chaîne de caractères représentant l'adresse (URL) de callback à notifier lors de changement d'état significatifs

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKURL_INVALID`.

network→get_discoverable() YNetwork
network→discoverable()

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

```
js      function get_discoverable( )
nodejs  function get_discoverable( )
php     function get_discoverable( )
cpp     Y_DISCOVERABLE_enum get_discoverable( )
m       -(Y_DISCOVERABLE_enum) discoverable
pas     function get_discoverable( ): Integer
vb      function get_discoverable( ) As Integer
cs      int get_discoverable( )
java    int get_discoverable( )
py      def get_discoverable( )
cmd     YNetwork target get_discoverable
```

Retourne :

soit `Y_DISCOVERABLE_FALSE`, soit `Y_DISCOVERABLE_TRUE`, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour)

En cas d'erreur, déclenche une exception ou retourne `Y_DISCOVERABLE_INVALID`.

network→get_errorMessage() YNetwork
network→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

```
js      function get_errorMessage( )
nodejs  function get_errorMessage( )
php     function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     function get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      def get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau.

network→get_errorType() YNetwork
network→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau.

network→get_friendlyName() YNetwork
network→friendlyName()

Retourne un identifiant global de l'interface réseau au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et de l'interface réseau si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de l'interface réseau (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant l'interface réseau en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

network→get_functionDescriptor() YNetwork
network→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
nodejs  function get_functionDescriptor( )
php     function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     function get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      def get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :

un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

network→get_functionId() YNetwork
network→functionId()

Retourne l'identifiant matériel de l'interface réseau, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

Retourne :

une chaîne de caractères identifiant l'interface réseau (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

network→get_hardwareId() YNetwork
network→hardwareId()

Retourne l'identifiant matériel unique de l'interface réseau au format SERIAL.FUNCTIONID.

js	function **get_hardwareId**( )
nodejs	function **get_hardwareId**( )
php	function **get_hardwareId**( )
cpp	string **get_hardwareId**( )
m	-(NSString*) hardwareId
vb	function **get_hardwareId**( ) As String
cs	string **get_hardwareId**( )
java	String **get_hardwareId**( )
py	def **get_hardwareId**( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface réseau (par example RELAYLO1-123456.relay1).

Retourne :

une chaîne de caractères identifiant l'interface réseau (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

network→get_ipAddress() YNetwork
network→ipAddress()

Retourne l'adresse IP utilisée par le module Yoctopuce.

```
js      function get_ipAddress( )
nodejs  function get_ipAddress( )
php     function get_ipAddress( )
cpp     string get_ipAddress( )
m       -(NSString*) ipAddress
pas     function get_ipAddress( ): string
vb      function get_ipAddress( ) As String
cs      string get_ipAddress( )
java    String get_ipAddress( )
py      def get_ipAddress( )
cmd     YNetwork target get_ipAddress
```

Il peut s'agir d'une adresse configurée statiquement, ou d'une adresse reçue par un serveur DHCP.

Retourne :
une chaîne de caractères représentant l'adresse IP utilisée par le module Yoctopuce

En cas d'erreur, déclenche une exception ou retourne `Y_IPADDRESS_INVALID`.

network→get_logicalName() YNetwork
network→logicalName()

Retourne le nom logique de l'interface réseau.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YNetwork target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique de l'interface réseau.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

network→get_macAddress() YNetwork
network→macAddress()

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

```
js      function get_macAddress( )
nodejs  function get_macAddress( )
php     function get_macAddress( )
cpp     string get_macAddress( )
m       -(NSString*) macAddress
pas     function get_macAddress( ): string
vb      function get_macAddress( ) As String
cs      string get_macAddress( )
java    String get_macAddress( )
py      def get_macAddress( )
cmd     YNetwork target get_macAddress
```

L'adresse MAC est aussi présente sur un autocollant sur le module, représentée en chiffres et en code-barres.

Retourne :
une chaîne de caractères représentant l'adresse MAC de l'interface réseau, unique pour chaque module

En cas d'erreur, déclenche une exception ou retourne `Y_MACADDRESS_INVALID`.

network→get_module() YNetwork
network→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
nodejs  function get_module( )
php     function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     function get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
java    YModule get_module( )
py      def get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

network→get_module_async() YNetwork
network→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
  js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

network→get_poeCurrent() YNetwork
network→poeCurrent()

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

js	function **get_poeCurrent**( )
nodejs	function **get_poeCurrent**( )
php	function **get_poeCurrent**( )
cpp	int **get_poeCurrent**( )
m	-(int) poeCurrent
pas	function **get_poeCurrent**( ): LongInt
vb	function **get_poeCurrent**( ) As Integer
cs	int **get_poeCurrent**( )
java	int **get_poeCurrent**( )
py	def **get_poeCurrent**( )
cmd	YNetwork **target** **get_poeCurrent**

La consommation est mesurée après conversion en 5 Volt, et ne doit jamais dépasser 1800 mA.

Retourne :
un entier représentant le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères

En cas d'erreur, déclenche une exception ou retourne `Y_POECURRENT_INVALID`.

network→get_primaryDNS() YNetwork
network→primaryDNS()

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

```
js      function get_primaryDNS( )
nodejs  function get_primaryDNS( )
php     function get_primaryDNS( )
cpp     string get_primaryDNS( )
m       -(NSString*) primaryDNS
pas     function get_primaryDNS( ): string
vb      function get_primaryDNS( ) As String
cs      string get_primaryDNS( )
java    String get_primaryDNS( )
py      def get_primaryDNS( )
cmd     YNetwork target get_primaryDNS
```

Retourne :
une chaîne de caractères représentant l'adresse IP du serveur de noms primaire que le module doit utiliser

En cas d'erreur, déclenche une exception ou retourne `Y_PRIMARYDNS_INVALID`.

network→get_readiness() YNetwork
network→readiness()

Retourne l'état de fonctionnement atteint par l'interface réseau.

```
js      function get_readiness( )
nodejs  function get_readiness( )
php     function get_readiness( )
cpp     Y_READINESS_enum get_readiness( )
m       -(Y_READINESS_enum) readiness
pas     function get_readiness( ): Integer
vb      function get_readiness( ) As Integer
cs      int get_readiness( )
java    int get_readiness( )
py      def get_readiness( )
cmd     YNetwork target get_readiness
```

Le niveau zéro (DOWN_0) signifie qu'aucun support réseau matériel n'a été détecté. Soit il n'y a pas de signal sur le cable réseau, soit le point d'accès sans fil choisi n'est pas détecté. Le niveau 1 (LIVE_1) est atteint lorsque le réseau est détecté, mais n'est pas encore connecté. Pour un réseau sans fil, cela confirme la l'existence du SSID configuré. Le niveau 2 (LINK_2) est atteint lorsque le support matériel du réseau est fonctionnel. Pour une connection réseau filaire, le niveau 2 signifie que le cable est connecté aux deux bouts. Pour une connection à un point d'accès réseau sans fil, il démontre que les paramètres de sécurités configurés sont corrects. Pour une connection sans fil en mode ad-hoc, cela signifie qu'il y a au moins un partenaire sur le réseau ad-hoc. Le niveau 3 (DHCP_3) est atteint lorsque qu'une adresse IP a été obtenue par DHCP. Le niveau 4 (DNS_4) est atteint lorsqu'un serveur DNS est joignable par le réseau. Le niveau 5 (WWW_5) est atteint lorsque la connectivité globale à internet est avérée par l'obtention de l'heure courante sur une serveur NTP.

Retourne :

une valeur parmi `Y_READINESS_DOWN`, `Y_READINESS_EXISTS`, `Y_READINESS_LINKED`, `Y_READINESS_LAN_OK` et `Y_READINESS_WWW_OK` représentant l'état de fonctionnement atteint par l'interface réseau

En cas d'erreur, déclenche une exception ou retourne `Y_READINESS_INVALID`.

network→get_router() YNetwork
network→router()

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

```
js      function get_router( )
nodejs  function get_router( )
php     function get_router( )
cpp     string get_router( )
m       -(NSString*) router
pas     function get_router( ): string
vb      function get_router( ) As String
cs      string get_router( )
java    String get_router( )
py      def get_router( )
cmd     YNetwork target get_router
```

Retourne :

une chaîne de caractères représentant l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*)

En cas d'erreur, déclenche une exception ou retourne `Y_ROUTER_INVALID`.

network→get_secondaryDNS() YNetwork
network→secondaryDNS()

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

```
js      function get_secondaryDNS( )
nodejs  function get_secondaryDNS( )
php     function get_secondaryDNS( )
cpp     string get_secondaryDNS( )
m       -(NSString*) secondaryDNS
pas     function get_secondaryDNS( ): string
vb      function get_secondaryDNS( ) As String
cs      string get_secondaryDNS( )
java    String get_secondaryDNS( )
py      def get_secondaryDNS( )
cmd     YNetwork target get_secondaryDNS
```

Retourne :

une chaîne de caractères représentant l'adresse IP du serveur de noms secondaire que le module doit utiliser

En cas d'erreur, déclenche une exception ou retourne `Y_SECONDARYDNS_INVALID`.

network→get_subnetMask() YNetwork
network→subnetMask()

Retourne le masque de sous-réseau utilisé par le module.

```
js      function get_subnetMask( )
nodejs  function get_subnetMask( )
php     function get_subnetMask( )
cpp     string get_subnetMask( )
m       -(NSString*) subnetMask
pas     function get_subnetMask( ): string
vb      function get_subnetMask( ) As String
cs      string get_subnetMask( )
java    String get_subnetMask( )
py      def get_subnetMask( )
cmd     YNetwork target get_subnetMask
```

Retourne :
une chaîne de caractères représentant le masque de sous-réseau utilisé par le module

En cas d'erreur, déclenche une exception ou retourne `Y_SUBNETMASK_INVALID`.

network→get_userData() YNetwork
network→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function **get_userData**( )
nodejs	function **get_userData**( )
php	function **get_userData**( )
cpp	void * **get_userData**( )
m	-(void*) userData
pas	function **get_userData**( ): Tobject
vb	function **get_userData**( ) As Object
cs	object **get_userData**( )
java	Object **get_userData**( )
py	def **get_userData**( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

network→get_userPassword() YNetwork
network→userPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

```
js      function get_userPassword( )
nodejs  function get_userPassword( )
php     function get_userPassword( )
cpp     string get_userPassword( )
m       -(NSString*) userPassword
pas     function get_userPassword( ): string
vb      function get_userPassword( ) As String
cs      string get_userPassword( )
java    String get_userPassword( )
py      def get_userPassword( )
cmd     YNetwork target get_userPassword
```

Retourne :

une chaîne de caractères représentant une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_USERPASSWORD_INVALID`.

network→get_wwwWatchdogDelay() YNetwork
network→wwwWatchdogDelay()

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

```
js      function get_wwwWatchdogDelay( )
nodejs  function get_wwwWatchdogDelay( )
php     function get_wwwWatchdogDelay( )
cpp     int get_wwwWatchdogDelay( )
m       -(int) wwwWatchdogDelay
pas     function get_wwwWatchdogDelay( ): LongInt
vb      function get_wwwWatchdogDelay( ) As Integer
cs      int get_wwwWatchdogDelay( )
java    int get_wwwWatchdogDelay( )
py      def get_wwwWatchdogDelay( )
cmd     YNetwork target get_wwwWatchdogDelay
```

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW.

Retourne :

un entier représentant la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet

En cas d'erreur, déclenche une exception ou retourne `Y_WWWWATCHDOGDELAY_INVALID`.

network→isOnline() YNetwork

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache de l'interface réseau sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :

`true` si l'interface réseau est joignable, `false` sinon

network→isOnline_async() YNetwork

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'interface réseau sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

network→load() YNetwork

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→load_async() YNetwork

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

network→nextNetwork() YNetwork

Continue l'énumération des interfaces réseau commencée à l'aide de `yFirstNetwork()`.

```
js      function nextNetwork( )
nodejs  function nextNetwork( )
php     function nextNetwork( )
cpp     YNetwork * nextNetwork( )
m       -(YNetwork*) nextNetwork
pas     function nextNetwork( ): TYNetwork
vb      function nextNetwork( ) As YNetwork
cs      YNetwork nextNetwork( )
java    YNetwork nextNetwork( )
py      def nextNetwork( )
```

Retourne :

un pointeur sur un objet `YNetwork` accessible en ligne, ou `null` lorsque l'énumération est terminée.

network→ping() YNetwork

Ping str_host pour vérifier la connexion réseau.

```
js      function ping( host)
nodejs  function ping( host)
php     function ping( $host)
cpp     string ping( string host)
m       -(NSString*) ping : (NSString*) host
pas     function ping( host: string): string
vb      function ping( ) As String
cs      string ping( string host)
java    String ping( String host)
py      def ping( host)
cmd     YNetwork target ping host
```

Envoie quatre requêtes ICMP ECHO_RESPONSER à la cible str_host depuis le module. Cette méthode retourne une chaîne de caractères avec le résultat des 4 requêtes ICMP ECHO_RESPONSE.

Paramètres :

host le nom d'hôte ou l'adresse IP de la cible

Retourne :

une chaîne de caractères contenant le résultat du ping.

network→registerValueCallback() YNetwork

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
nodejs  function registerValueCallback( callback)
php     function registerValueCallback( $callback)
cpp     int registerValueCallback( YNetworkValueCallback callback)
m       -(int) registerValueCallback : (YNetworkValueCallback) callback
pas     function registerValueCallback( callback: TYNetworkValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
py      def registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

network→set_adminPassword() YNetwork
network→setAdminPassword()

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

```
js      function set_adminPassword( newval)
nodejs  function set_adminPassword( newval)
php     function set_adminPassword( $newval)
cpp     int set_adminPassword( const string& newval)
m       -(int) setAdminPassword : (NSString*) newval
pas     function set_adminPassword( newval: string): integer
vb      function set_adminPassword( ByVal newval As String) As Integer
cs      int set_adminPassword( string newval)
java    int set_adminPassword( String newval)
py      def set_adminPassword( newval)
cmd     YNetwork target set_adminPassword newval
```

Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackCredentials() YNetwork
network→setCallbackCredentials()

Modifie le laisser-passer pour se connecter à l'adresse de callback.

```
js      function set_callbackCredentials( newval)
nodejs  function set_callbackCredentials( newval)
php     function set_callbackCredentials( $newval)
cpp     int set_callbackCredentials( const string& newval)
m       -(int) setCallbackCredentials : (NSString*) newval
pas     function set_callbackCredentials( newval: string): integer
vb      function set_callbackCredentials( ByVal newval As String) As Integer
cs      int set_callbackCredentials( string newval)
java    int set_callbackCredentials( String newval)
py      def set_callbackCredentials( newval)
cmd     YNetwork target set_callbackCredentials newval
```

Le laisser-passer doit être fourni tel que retourné par la fonction `get_callbackCredentials`, sous la forme `username:hash`. La valeur du hash dépend de la méthode d'autorisation implémentée par le callback. Pour une autorisation de type Basic, le hash est le MD5 de la chaîne `username:password`. Pour une autorisation de type Digest, le hash est le MD5 de la chaîne `username:realm:password`. Pour une utilisation simplifiée, utilisez la fonction `callbackLogin`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le laisser-passer pour se connecter à l'adresse de callback

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackEncoding() YNetwork
network→setCallbackEncoding()

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

```
js      function set_callbackEncoding( newval)
nodejs  function set_callbackEncoding( newval)
php     function set_callbackEncoding( $newval)
cpp     int set_callbackEncoding( Y_CALLBACKENCODING_enum newval)
m       -(int) setCallbackEncoding : (Y_CALLBACKENCODING_enum) newval
pas     function set_callbackEncoding( newval: Integer): integer
vb      function set_callbackEncoding( ByVal newval As Integer) As Integer
cs      int set_callbackEncoding( int newval)
java    int set_callbackEncoding( int newval)
py      def set_callbackEncoding( newval)
cmd     YNetwork target set_callbackEncoding newval
```

Paramètres :

newval une valeur parmi `Y_CALLBACKENCODING_FORM`, `Y_CALLBACKENCODING_JSON`, `Y_CALLBACKENCODING_JSON_ARRAY`, `Y_CALLBACKENCODING_CSV` et `Y_CALLBACKENCODING_YOCTO_API` représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackMaxDelay() YNetwork
network→setCallbackMaxDelay()

Modifie l'attente maximale entre deux notifications par callback, en secondes.

```
js      function set_callbackMaxDelay( newval)
nodejs  function set_callbackMaxDelay( newval)
php     function set_callbackMaxDelay( $newval)
cpp     int set_callbackMaxDelay( int newval)
m       -(int) setCallbackMaxDelay : (int) newval
pas     function set_callbackMaxDelay( newval: LongInt): integer
vb      function set_callbackMaxDelay( ByVal newval As Integer) As Integer
cs      int set_callbackMaxDelay( int newval)
java    int set_callbackMaxDelay( int newval)
py      def set_callbackMaxDelay( newval)
cmd     YNetwork target set_callbackMaxDelay newval
```

Paramètres :

newval un entier représentant l'attente maximale entre deux notifications par callback, en secondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackMethod() YNetwork
network→setCallbackMethod()

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

```
js      function set_callbackMethod( newval)
nodejs  function set_callbackMethod( newval)
php     function set_callbackMethod( $newval)
cpp     int set_callbackMethod( Y_CALLBACKMETHOD_enum newval)
m       -(int) setCallbackMethod : (Y_CALLBACKMETHOD_enum) newval
pas     function set_callbackMethod( newval: Integer): integer
vb      function set_callbackMethod( ByVal newval As Integer) As Integer
cs      int set_callbackMethod( int newval)
java    int set_callbackMethod( int newval)
py      def set_callbackMethod( newval)
cmd     YNetwork target set_callbackMethod newval
```

Paramètres :

newval une valeur parmi `Y_CALLBACKMETHOD_POST`, `Y_CALLBACKMETHOD_GET` et `Y_CALLBACKMETHOD_PUT` représentant la méthode HTTP à utiliser pour signaler les changements d'état par callback

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackMinDelay() YNetwork
network→setCallbackMinDelay()

Modifie l'attente minimale entre deux notifications par callback, en secondes.

```
js      function set_callbackMinDelay( newval)
nodejs  function set_callbackMinDelay( newval)
php     function set_callbackMinDelay( $newval)
cpp     int set_callbackMinDelay( int newval)
m       -(int) setCallbackMinDelay : (int) newval
pas     function set_callbackMinDelay( newval: LongInt): integer
vb      function set_callbackMinDelay( ByVal newval As Integer) As Integer
cs      int set_callbackMinDelay( int newval)
java    int set_callbackMinDelay( int newval)
py      def set_callbackMinDelay( newval)
cmd     YNetwork target set_callbackMinDelay newval
```

Paramètres :

newval un entier représentant l'attente minimale entre deux notifications par callback, en secondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackUrl() YNetwork
network→setCallbackUrl()

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

```
js      function set_callbackUrl( newval)
nodejs  function set_callbackUrl( newval)
php     function set_callbackUrl( $newval)
cpp     int set_callbackUrl( const string& newval)
m       -(int) setCallbackUrl : (NSString*) newval
pas     function set_callbackUrl( newval: string): integer
vb      function set_callbackUrl( ByVal newval As String) As Integer
cs      int set_callbackUrl( string newval)
java    int set_callbackUrl( String newval)
py      def set_callbackUrl( newval)
cmd     YNetwork target set_callbackUrl newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant l'adresse (URL) de callback à notifier lors de changement d'état significatifs

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_discoverable() YNetwork
network→setDiscoverable()

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

```
js      function set_discoverable( newval)
nodejs  function set_discoverable( newval)
php     function set_discoverable( $newval)
cpp     int set_discoverable( Y_DISCOVERABLE_enum newval)
m       -(int) setDiscoverable : (Y_DISCOVERABLE_enum) newval
pas     function set_discoverable( newval: Integer): integer
vb      function set_discoverable( ByVal newval As Integer) As Integer
cs      int set_discoverable( int newval)
java    int set_discoverable( int newval)
py      def set_discoverable( newval)
cmd     YNetwork target set_discoverable newval
```

Paramètres :

newval soit `Y_DISCOVERABLE_FALSE`, soit `Y_DISCOVERABLE_TRUE`, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour)

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_logicalName() YNetwork
network→setLogicalName()

Modifie le nom logique de l'interface réseau.

```
js      function set_logicalName( newval)
nodejs  function set_logicalName( newval)
php     function set_logicalName( $newval)
cpp     int set_logicalName( const string& newval)
m       -(int) setLogicalName : (NSString*) newval
pas     function set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
java    int set_logicalName( String newval)
py      def set_logicalName( newval)
cmd     YNetwork target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique de l'interface réseau.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_primaryDNS() YNetwork
network→setPrimaryDNS()

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

```
js      function set_primaryDNS( newval)
nodejs  function set_primaryDNS( newval)
php     function set_primaryDNS( $newval)
cpp     int set_primaryDNS( const string& newval)
m       -(int) setPrimaryDNS : (NSString*) newval
pas     function set_primaryDNS( newval: string): integer
vb      function set_primaryDNS( ByVal newval As String) As Integer
cs      int set_primaryDNS( string newval)
java    int set_primaryDNS( String newval)
py      def set_primaryDNS( newval)
cmd     YNetwork target set_primaryDNS newval
```

En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

newval une chaîne de caractères représentant l'adresse IP du serveur de noms primaire que le module doit utiliser

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_secondaryDNS() YNetwork
network→setSecondaryDNS()

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

```
js      function set_secondaryDNS( newval)
nodejs  function set_secondaryDNS( newval)
php     function set_secondaryDNS( $newval)
cpp     int set_secondaryDNS( const string& newval)
m       -(int) setSecondaryDNS : (NSString*) newval
pas     function set_secondaryDNS( newval: string): integer
vb      function set_secondaryDNS( ByVal newval As String) As Integer
cs      int set_secondaryDNS( string newval)
java    int set_secondaryDNS( String newval)
py      def set_secondaryDNS( newval)
cmd     YNetwork target set_secondaryDNS newval
```

En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

newval une chaîne de caractères représentant l'adresse IP du serveur de nom secondaire que le module doit utiliser

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_userData() YNetwork
network→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

network→set_userPassword() YNetwork
network→setUserPassword()

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

```
js      function set_userPassword( newval)
nodejs  function set_userPassword( newval)
php     function set_userPassword( $newval)
cpp     int set_userPassword( const string& newval)
m       -(int) setUserPassword : (NSString*) newval
pas     function set_userPassword( newval: string): integer
vb      function set_userPassword( ByVal newval As String) As Integer
cs      int set_userPassword( string newval)
java    int set_userPassword( String newval)
py      def set_userPassword( newval)
cmd     YNetwork target set_userPassword newval
```

Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_wwwWatchdogDelay() YNetwork
network→setWwwWatchdogDelay()

Modifie la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

```
js      function set_wwwWatchdogDelay( newval)
nodejs  function set_wwwWatchdogDelay( newval)
php     function set_wwwWatchdogDelay( $newval)
cpp     int set_wwwWatchdogDelay( int newval)
m       -(int) setWwwWatchdogDelay : (int) newval
pas     function set_wwwWatchdogDelay( newval: LongInt): integer
vb      function set_wwwWatchdogDelay( ByVal newval As Integer) As Integer
cs      int set_wwwWatchdogDelay( int newval)
java    int set_wwwWatchdogDelay( int newval)
py      def set_wwwWatchdogDelay( newval)
cmd     YNetwork target set_wwwWatchdogDelay newval
```

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW. La plus petite durée non-nulle utilisable est 90 secondes.

Paramètres :

newval un entier représentant la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→useDHCP() YNetwork

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

```
js      function useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
nodejs  function useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
php     function useDHCP( $fallbackIpAddr, $fallbackSubnetMaskLen, $fallbackRouter)
cpp     int useDHCP( string fallbackIpAddr,
                     int fallbackSubnetMaskLen,
                     string fallbackRouter)
m       -(int) useDHCP : (NSString*) fallbackIpAddr
                       : (int) fallbackSubnetMaskLen
                       : (NSString*) fallbackRouter
pas     function useDHCP( fallbackIpAddr: string,
                          fallbackSubnetMaskLen: LongInt,
                          fallbackRouter: string): integer
vb      function useDHCP( ByVal fallbackIpAddr As String,
                          ByVal fallbackSubnetMaskLen As Integer,
                          ByVal fallbackRouter As String) As Integer
cs      int useDHCP( string fallbackIpAddr,
                     int fallbackSubnetMaskLen,
                     string fallbackRouter)
java    int useDHCP( String fallbackIpAddr,
                     int fallbackSubnetMaskLen,
                     String fallbackRouter)
py      def useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
cmd     YNetwork target useDHCP fallbackIpAddr fallbackSubnetMaskLen fallbackRouter
```

En attendant qu'une adresse soit reçue (et indéfiniment si aucun serveur DHCP ne répond), le module utilisera les paramètres IP spécifiés à cette fonction. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

fallbackIpAddr	adresse IP à utiliser si aucun serveur DHCP ne répond
fallbackSubnetMaskLen	longueur du masque de sous-réseau à utiliser si aucun serveur DHCP ne répond. Par exemple, la valeur 24 représente 255.255.255.0.
fallbackRouter	adresse de la passerelle à utiliser si aucun serveur DHCP ne répond

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→useStaticIP() YNetwork

Modifie la configuration de l'interface réseau pour utiliser une adresse IP assignée manuellement (adresse IP statique).

```
js      function useStaticIP( ipAddress, subnetMaskLen, router)
nodejs  function useStaticIP( ipAddress, subnetMaskLen, router)
php     function useStaticIP( $ipAddress, $subnetMaskLen, $router)
cpp     int useStaticIP( string ipAddress,
                         int subnetMaskLen,
                         string router)
m       -(int) useStaticIP : (NSString*) ipAddress
                           : (int) subnetMaskLen
                           : (NSString*) router
pas     function useStaticIP( ipAddress: string,
                              subnetMaskLen: LongInt,
                              router: string): integer
vb      function useStaticIP( ByVal ipAddress As String,
                              ByVal subnetMaskLen As Integer,
                              ByVal router As String) As Integer
cs      int useStaticIP( string ipAddress,
                         int subnetMaskLen,
                         string router)
java    int useStaticIP( String ipAddress,
                         int subnetMaskLen,
                         String router)
py      def useStaticIP( ipAddress, subnetMaskLen, router)
cmd     YNetwork target useStaticIP ipAddress subnetMaskLen router
```

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

ipAddress	adresse IP à utiliser par le module
subnetMaskLen	longueur du masque de sous-réseau à utiliser. Par exemple, la valeur 24 représente 255.255.255.0.
router	adresse IP de la passerelle à utiliser ("default gateway")

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→wait_async() YNetwork

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

11.4. Interface de la fonction Files

L'interface de stockage de fichiers permet de stocker des fichiers sur certains modules, par exemple pour personnaliser un service web (dans le cas d'un module connecté au réseau) ou pour ajouter un police de caractères (dans le cas d'un module d'affichage).

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_files.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YFiles = yoctolib.YFiles;
php     require_once('yocto_files.php');
cpp     #include "yocto_files.h"
m       #import "yocto_files.h"
pas     uses yocto_files;
vb      yocto_files.vb
cs      yocto_files.cs
java    import com.yoctopuce.YoctoAPI.YFiles;
py      from yocto_files import *
```

Fonction globales
yFindFiles(**func**) Permet de retrouver un système de fichier d'après un identifiant donné.
yFirstFiles() Commence l'énumération des système de fichier accessibles par la librairie.
Méthodes des objets `YFiles`
files→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance du système de fichier au format `TYPE(NAME)=SERIAL.FUNCTIONID`.
files→download(**pathname**) Télécharge le fichier choisi du filesystème et retourne son contenu.
files→download_async(**pathname**, **callback**, **context**) Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.
files→format_fs() Rétabli le système de fichier dans on état original, défragmenté.
files→get_advertisedValue() Retourne la valeur courante du système de fichier (pas plus de 6 caractères).
files→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.
files→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.
files→get_filesCount() Retourne le nombre de fichiers présents dans le système de fichier.
files→get_freeSpace() Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets.
files→get_friendlyName() Retourne un identifiant global du système de fichier au format `NOM_MODULE.NOM_FONCTION`.
files→get_functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

files→get_functionId()

Retourne l'identifiant matériel du système de fichier, sans référence au module.

files→get_hardwareId()

Retourne l'identifiant matériel unique du système de fichier au format SERIAL.FUNCTIONID.

files→get_list(**pattern**)

Retourne une liste d'objets objet YFileRecord qui décrivent les fichiers présents dans le système de fichier.

files→get_logicalName()

Retourne le nom logique du système de fichier.

files→get_module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

files→get_module_async(**callback**, **context**)

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

files→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.

files→isOnline()

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

files→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

files→load(**msValidity**)

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

files→load_async(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

files→nextFiles()

Continue l'énumération des système de fichier commencée à l'aide de yFirstFiles().

files→registerValueCallback(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

files→remove(**pathname**)

Efface un fichier, spécifié par son path complet, du système de fichier.

files→set_logicalName(**newval**)

Modifie le nom logique du système de fichier.

files→set_userData(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get_userData.

files→upload(**pathname**, **content**)

Télécharge un contenu vers le système de fichier, au chemin d'accès spécifié.

files→wait_async(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YFiles.FindFiles() YFiles
yFindFiles()

Permet de retrouver un système de fichier d'après un identifiant donné.

```
js      function yFindFiles( func)
nodejs  function FindFiles( func)
php     function yFindFiles( $func)
cpp     YFiles* yFindFiles( string func)
m       +(YFiles*) FindFiles : (NSString*) func
pas     function yFindFiles( func: string): TYFiles
vb      function yFindFiles( ByVal func As String) As YFiles
cs      YFiles FindFiles( string func)
java    YFiles FindFiles( String func)
py      def FindFiles( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le système de fichier soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YFiles.isOnline()` pour tester si le système de fichier est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

func une chaîne de caractères qui référence le système de fichier sans ambiguïté

Retourne :

un objet de classe `YFiles` qui permet ensuite de contrôler le système de fichier.

YFiles.FirstFiles() YFiles
yFirstFiles()

Commence l'énumération des système de fichier accessibles par la librairie.

```
js      function yFirstFiles( )
nodejs  function FirstFiles( )
php     function yFirstFiles( )
cpp     YFiles* yFirstFiles( )
m       +(YFiles*) FirstFiles
pas     function yFirstFiles( ): TYFiles
vb      function yFirstFiles( ) As YFiles
cs      YFiles FirstFiles( )
java    YFiles FirstFiles( )
py      def FirstFiles( )
```

Utiliser la fonction `YFiles.nextFiles()` pour itérer sur les autres système de fichier.

Retourne :

un pointeur sur un objet `YFiles`, correspondant au premier système de fichier accessible en ligne, ou `null` si il n'y a pas de système de fichier disponibles.

files→describe() YFiles

Retourne un court texte décrivant de manière non-ambigüe l'instance du système de fichier au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant le système de fichier (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

files→download() YFiles

Télécharge le fichier choisi du filesystème et retourne son contenu.

```
js      function download( pathname)
nodejs  function download( pathname)
php     function download( $pathname)
cpp     string download( string pathname)
m       -(NSData*) download : (NSString*) pathname
pas     function download( pathname: string): TByteArray
vb      function download( ) As Byte
py      def download( pathname)
cmd     YFiles target download pathname
```

Paramètres :

pathname nom complet du fichier à charger, y compris le chemin d'accès.

Retourne :

le contenu du fichier chargé sous forme d'objet binaire

En cas d'erreur, déclenche une exception ou retourne un contenu vide.

files→download_async() YFiles

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.

```
js      function download_async( pathname, callback, context)
nodejs  function download_async( pathname, callback, context)
```

Paramètres :

pathname nom complet du fichier à charger, y compris le chemin d'accès.

callback fonction fournie par l'utilisateur, qui sera appelée lorsque la suite du chargement aura été effectué. La fonction callback doit prendre trois arguments: - la variable de contexte à disposition de l'utilisateur - l'objet YFiles dont la méthode download_async a été appelée - le contenu du fichier chargé sous forme d'objet binaire

context variable de contexte à disposition de l'utilisateur

Retourne :

rien.

files→format_fs() YFiles

Rétabli le système de fichier dans on état original, défragmenté.

```
js      function format_fs( )
nodejs  function format_fs( )
php     function format_fs( )
cpp     int format_fs( )
m       -(int) format_fs
pas     function format_fs( ): LongInt
vb      function format_fs( ) As Integer
cs      int format_fs( )
java    int format_fs( )
py      def format_fs( )
cmd     YFiles target format_fs
```

entièrement vide. Tous les fichiers précédemment chargés sont irrémédiablement effacés.

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→get_advertisedValue() YFiles
files→advertisedValue()

Retourne la valeur courante du système de fichier (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YFiles target get_advertisedValue
```

Retourne :
une chaîne de caractères représentant la valeur courante du système de fichier (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

files→get_errorMessage() YFiles
files→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

```
js       function get_errorMessage( )
nodejs   function get_errorMessage( )
php      function get_errorMessage( )
cpp      string get_errorMessage( )
m        -(NSString*) errorMessage
pas      function get_errorMessage( ): string
vb       function get_errorMessage( ) As String
cs       string get_errorMessage( )
java     String get_errorMessage( )
py       def get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du système de fichier.

files→get_errorType() YFiles
files→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du système de fichier.

files→get_filesCount() YFiles
files→filesCount()

Retourne le nombre de fichiers présents dans le système de fichier.

```
js      function get_filesCount( )
nodejs  function get_filesCount( )
php     function get_filesCount( )
cpp     int get_filesCount( )
m       -(int) filesCount
pas     function get_filesCount( ): LongInt
vb      function get_filesCount( ) As Integer
cs      int get_filesCount( )
java    int get_filesCount( )
py      def get_filesCount( )
cmd     YFiles target get_filesCount
```

Retourne :
un entier représentant le nombre de fichiers présents dans le système de fichier

En cas d'erreur, déclenche une exception ou retourne `Y_FILESCOUNT_INVALID`.

files→get_freeSpace() YFiles
files→freeSpace()

Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets.

```
js      function get_freeSpace( )
nodejs  function get_freeSpace( )
php     function get_freeSpace( )
cpp     int get_freeSpace( )
m       -(int) freeSpace
pas     function get_freeSpace( ): LongInt
vb      function get_freeSpace( ) As Integer
cs      int get_freeSpace( )
java    int get_freeSpace( )
py      def get_freeSpace( )
cmd     YFiles target get_freeSpace
```

Retourne :

un entier représentant l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets

En cas d'erreur, déclenche une exception ou retourne `Y_FREESPACE_INVALID`.

files→get_friendlyName() YFiles
files→friendlyName()

Retourne un identifiant global du système de fichier au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et du système de fichier si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du système de fichier (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant le système de fichier en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

files→get_functionDescriptor() YFiles
files→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
nodejs  function get_functionDescriptor( )
php     function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     function get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      def get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :

un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

files→get_functionId() YFiles
files→functionId()

Retourne l'identifiant matériel du système de fichier, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

Retourne :

une chaîne de caractères identifiant le système de fichier (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

files→get_hardwareId() YFiles
files→hardwareId()

Retourne l'identifiant matériel unique du système de fichier au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
nodejs  function get_hardwareId( )
php     function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
java    String get_hardwareId( )
py      def get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du système de fichier (par example RELAYLO1-123456.relay1).

Retourne :

une chaîne de caractères identifiant le système de fichier (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

files→get_list() YFiles
files→list()

Retourne une liste d'objets objet YFileRecord qui décrivent les fichiers présents dans le système de fichier.

```
js      function get_list( pattern)
nodejs  function get_list( pattern)
php     function get_list( $pattern)
cpp     vector<YFileRecord> get_list( string pattern)
m       -(NSMutableArray*) list : (NSString*) pattern
pas     function get_list( pattern: string): TYFileRecordArray
vb      function get_list( ) As List
cs      List<YFileRecord> get_list( string pattern)
java    ArrayList<YFileRecord> get_list( String pattern)
py      def get_list( pattern)
cmd     YFiles target get_list pattern
```

Paramètres :

pattern un filtre optionel sur les noms de fichiers retournés, pouvant contenir des astérisques et des points d'interrogations comme jokers. Si le pattern fourni est vide, tous les fichiers sont retournés.

Retourne :

une liste d'objets `YFileRecord`, contenant le nom complet (y compris le chemin d'accès), la taille en octets et le CRC 32-bit du contenu du fichier.

En cas d'erreur, déclenche une exception ou retourne une liste vide.

files→get_logicalName() YFiles
files→logicalName()

Retourne le nom logique du système de fichier.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YFiles target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique du système de fichier.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

files→get_module() YFiles
files→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
nodejs  function get_module( )
php     function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     function get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
java    YModule get_module( )
py      def get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

files→get_module_async() YFiles
files→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
  js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

files→get_userData() YFiles
files→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
nodejs  function get_userData( )
php     function get_userData( )
cpp     void * get_userData( )
m       -(void*) userData
pas     function get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      def get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

files→isOnline() YFiles

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache du système de fichier sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :

`true` si le système de fichier est joignable, `false` sinon

files→isOnline_async() YFiles

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du système de fichier sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

files→load() YFiles

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→load_async() YFiles

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

files→nextFiles() YFiles

Continue l'énumération des système de fichier commencée à l'aide de `yFirstFiles()`.

```
js      function nextFiles( )
nodejs  function nextFiles( )
php     function nextFiles( )
cpp     YFiles * nextFiles( )
m       -(YFiles*) nextFiles
pas     function nextFiles( ): TYFiles
vb      function nextFiles( ) As YFiles
cs      YFiles nextFiles( )
java    YFiles nextFiles( )
py      def nextFiles( )
```

Retourne :

un pointeur sur un objet `YFiles` accessible en ligne, ou `null` lorsque l'énumération est terminée.

files→registerValueCallback() YFiles

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
nodejs  function registerValueCallback( callback)
php     function registerValueCallback( $callback)
cpp     int registerValueCallback( YFilesValueCallback callback)
m       -(int) registerValueCallback : (YFilesValueCallback) callback
pas     function registerValueCallback( callback: TYFilesValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
py      def registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

files→remove() YFiles

Efface un fichier, spécifié par son path complet, du système de fichier.

```
js      function remove( pathname)
nodejs  function remove( pathname)
php     function remove( $pathname)
cpp     int remove( string pathname)
m       -(int) remove : (NSString*) pathname
pas     function remove( pathname: string): LongInt
vb      function remove( ) As Integer
cs      int remove( string pathname)
java    int remove( String pathname)
py      def remove( pathname)
cmd     YFiles target remove pathname
```

A cause de la fragmentation, l'effacement d'un fichier ne libère pas toujours la totalité de l'espace qu'il occuppe. Par contre, la ré-écriture d'un fichier du même nom récupérera dans tout les cas l'espace qui n'aurait éventuellement pas été libéré. Pour s'assurer de libérer la totalité de l'espace du système de fichier, utilisez la fonction `format_fs`.

Paramètres :

pathname nom complet du fichier, y compris le chemin d'accès.

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→set_logicalName() YFiles
files→setLogicalName()

Modifie le nom logique du système de fichier.

```
js      function set_logicalName( newval)
nodejs  function set_logicalName( newval)
php     function set_logicalName( $newval)
cpp     int set_logicalName( const string& newval)
m       -(int) setLogicalName : (NSString*) newval
pas     function set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
java    int set_logicalName( String newval)
py      def set_logicalName( newval)
cmd     YFiles target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique du système de fichier.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→set_userData() YFiles
files→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

files→upload() YFiles

Télécharge un contenu vers le système de fichier, au chemin d'accès spécifié.

```
js      function upload( pathname, content)
nodejs  function upload( pathname, content)
php     function upload( $pathname, $content)
cpp     int upload( string pathname, string content)
m       -(int) upload : (NSString*) pathname
                      : (NSData*) content
pas     function upload( pathname: string, content: TByteArray): LongInt
vb      procedure upload( )
cs      int upload( string pathname)
java    int upload( String pathname)
py      def upload( pathname, content)
cmd     YFiles target upload pathname content
```

Si un fichier existe déjà pour le même chemin d'accès, son contenu est remplacé.

Paramètres :

pathname nom complet du fichier, y compris le chemin d'accès.

content contenu du fichier à télécharger

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→wait_async() YFiles

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

11.5. Interface de la fonction WakeUpMonitor

La fonction WakeUpMonitor prend en charge le contrôle global de toutes les sources de réveil possibles ainsi que les mises en sommeil automatiques.

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_wakeupmonitor.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YWakeUpMonitor = yoctolib.YWakeUpMonitor;
php     require_once('yocto_wakeupmonitor.php');
cpp     #include "yocto_wakeupmonitor.h"
m       #import "yocto_wakeupmonitor.h"
pas     uses yocto_wakeupmonitor;
vb      yocto_wakeupmonitor.vb
cs      yocto_wakeupmonitor.cs
java    import com.yoctopuce.YoctoAPI.YWakeUpMonitor;
py      from yocto_wakeupmonitor import *
```

Fonction globales
yFindWakeUpMonitor(**func**) Permet de retrouver un moniteur d'après un identifiant donné.
yFirstWakeUpMonitor() Commence l'énumération des Moniteurs accessibles par la librairie.
Méthodes des objets `YWakeUpMonitor`
wakeupmonitor→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance du moniteur au format `TYPE(NAME)=SERIAL.FUNCTIONID`.
wakeupmonitor→get_advertisedValue() Retourne la valeur courante du moniteur (pas plus de 6 caractères).
wakeupmonitor→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.
wakeupmonitor→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.
wakeupmonitor→get_friendlyName() Retourne un identifiant global du moniteur au format `NOM_MODULE.NOM_FONCTION`.
wakeupmonitor→get_functionDescriptor() Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.
wakeupmonitor→get_functionId() Retourne l'identifiant matériel du moniteur, sans référence au module.
wakeupmonitor→get_hardwareId() Retourne l'identifiant matériel unique du moniteur au format `SERIAL.FUNCTIONID`.
wakeupmonitor→get_logicalName() Retourne le nom logique du moniteur.
wakeupmonitor→get_module() Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.
wakeupmonitor→get_module_async(**callback**, **context**) Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

wakeupmonitor→get_nextWakeUp()

Retourne la prochaine date/heure de réveil agendée (format UNIX)

wakeupmonitor→get_powerDuration()

Retourne le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

wakeupmonitor→get_sleepCountdown()

Retourne le temps avant le prochain sommeil.

wakeupmonitor→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

wakeupmonitor→get_wakeUpReason()

Renvoie la raison du dernier réveil.

wakeupmonitor→get_wakeUpState()

Revoie l'état actuel du moniteur

wakeupmonitor→isOnline()

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

wakeupmonitor→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

wakeupmonitor→load(**msValidity**)

Met en cache les valeurs courantes du moniteur, avec une durée de validité spécifiée.

wakeupmonitor→load_async(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes du moniteur, avec une durée de validité spécifiée.

wakeupmonitor→nextWakeUpMonitor()

Continue l'énumération des Moniteurs commencée à l'aide de `yFirstWakeUpMonitor()`.

wakeupmonitor→registerValueCallback(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

wakeupmonitor→resetSleepCountDown()

Réinitialise le compteur de mise en sommeil.

wakeupmonitor→set_logicalName(**newval**)

Modifie le nom logique du moniteur.

wakeupmonitor→set_nextWakeUp(**newval**)

Modifie les jours de la semaine où un réveil doit avoir lieu.

wakeupmonitor→set_powerDuration(**newval**)

Modifie le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

wakeupmonitor→set_sleepCountdown(**newval**)

Modifie le temps avant le prochain sommeil .

wakeupmonitor→set_userData(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

wakeupmonitor→sleep(**secBeforeSleep**)

Déclenche une mise en sommeil jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

wakeupmonitor→sleepFor(**secUntilWakeUp**, **secBeforeSleep**)

Déclenche une mise en sommeil pour un temps donné ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

wakeupmonitor→sleepUntil(**wakeUpTime**, **secBeforeSleep**)

Déclenche une mise en sommeil jusqu'à une date donnée ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

wakeupmonitor→wait_async(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

wakeupmonitor→wakeUp()

Force un réveil.

YWakeUpMonitor.FindWakeUpMonitor() YWakeUpMonitor yFindWakeUpMonitor()

Permet de retrouver un moniteur d'après un identifiant donné.

```
js      function yFindWakeUpMonitor( func)
nodejs  function FindWakeUpMonitor( func)
php     function yFindWakeUpMonitor( $func)
cpp     YWakeUpMonitor* yFindWakeUpMonitor( const string& func)
m       +(YWakeUpMonitor*) FindWakeUpMonitor :(NSString*) func
pas     function yFindWakeUpMonitor( func: string): TYWakeUpMonitor
vb      function yFindWakeUpMonitor( ByVal func As String) As YWakeUpMonitor
cs      YWakeUpMonitor FindWakeUpMonitor( string func)
java    YWakeUpMonitor FindWakeUpMonitor( String func)
py      def FindWakeUpMonitor( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le moniteur soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWakeUpMonitor.isOnline()` pour tester si le moniteur est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

func une chaîne de caractères qui référence le moniteur sans ambiguïté

Retourne :

un objet de classe `YWakeUpMonitor` qui permet ensuite de contrôler le moniteur.

YWakeUpMonitor.FirstWakeUpMonitor() YWakeUpMonitor yFirstWakeUpMonitor()

Commence l'énumération des Moniteurs accessibles par la librairie.

js	function **yFirstWakeUpMonitor**( )
nodejs	function **FirstWakeUpMonitor**( )
php	function **yFirstWakeUpMonitor**( )
cpp	YWakeUpMonitor* **yFirstWakeUpMonitor**( )
m	+(YWakeUpMonitor*) **FirstWakeUpMonitor**
pas	function **yFirstWakeUpMonitor**( ): TYWakeUpMonitor
vb	function **yFirstWakeUpMonitor**( ) As YWakeUpMonitor
cs	YWakeUpMonitor **FirstWakeUpMonitor**( )
java	YWakeUpMonitor **FirstWakeUpMonitor**( )
py	def **FirstWakeUpMonitor**( )

Utiliser la fonction `YWakeUpMonitor.nextWakeUpMonitor()` pour itérer sur les autres Moniteurs.

Retourne :

un pointeur sur un objet `YWakeUpMonitor`, correspondant au premier moniteur accessible en ligne, ou `null` si il n'y a pas de Moniteurs disponibles.

wakeupmonitor→describe() YWakeUpMonitor

Retourne un court texte décrivant de manière non-ambigüe l'instance du moniteur au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant le moniteur (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

wakeupmonitor→get_advertisedValue() YWakeUpMonitor
wakeupmonitor→advertisedValue()

Retourne la valeur courante du moniteur (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YWakeUpMonitor target get_advertisedValue
```

Retourne :

une chaîne de caractères représentant la valeur courante du moniteur (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

wakeupmonitor→get_errorMessage() YWakeUpMonitor
wakeupmonitor→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.

```
js      function get_errorMessage( )
nodejs  function get_errorMessage( )
php     function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     function get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      def get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du moniteur.

wakeupmonitor→get_errorType() YWakeUpMonitor
wakeupmonitor→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du moniteur.

wakeupmonitor→get_friendlyName() YWakeUpMonitor
wakeupmonitor→friendlyName()

Retourne un identifiant global du moniteur au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et du moniteur si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du moniteur (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant le moniteur en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

wakeupmonitor→get_functionDescriptor() YWakeUpMonitor
wakeupmonitor→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function **get_functionDescriptor**( )
nodejs	function **get_functionDescriptor**( )
php	function **get_functionDescriptor**( )
cpp	YFUN_DESCR **get_functionDescriptor**( )
m	-(YFUN_DESCR) functionDescriptor
pas	function **get_functionDescriptor**( ): YFUN_DESCR
vb	function **get_functionDescriptor**( ) As YFUN_DESCR
cs	YFUN_DESCR **get_functionDescriptor**( )
java	String **get_functionDescriptor**( )
py	def **get_functionDescriptor**( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :

un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

wakeupmonitor→get_functionId() YWakeUpMonitor
wakeupmonitor→functionId()

Retourne l'identifiant matériel du moniteur, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

Retourne :

une chaîne de caractères identifiant le moniteur (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

wakeupmonitor→get_hardwareId() YWakeUpMonitor
wakeupmonitor→hardwareId()

Retourne l'identifiant matériel unique du moniteur au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
nodejs  function get_hardwareId( )
php     function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
java    String get_hardwareId( )
py      def get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du moniteur (par example RELAYLO1-123456.relay1).

Retourne :

une chaîne de caractères identifiant le moniteur (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

wakeupmonitor→get_logicalName() YWakeUpMonitor
wakeupmonitor→logicalName()

Retourne le nom logique du moniteur.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YWakeUpMonitor target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique du moniteur.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

wakeupmonitor→get_module() YWakeUpMonitor
wakeupmonitor→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function **get_module**( )
nodejs	function **get_module**( )
php	function **get_module**( )
cpp	YModule * **get_module**( )
m	-(YModule*) module
pas	function **get_module**( ): TYModule
vb	function **get_module**( ) As YModule
cs	YModule **get_module**( )
java	YModule **get_module**( )
py	def **get_module**( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

wakeupmonitor→get_module_async() YWakeUpMonitor
wakeupmonitor→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
  js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupmonitor→get_nextWakeUp() YWakeUpMonitor
wakeupmonitor→nextWakeUp()

Retourne la prochaine date/heure de réveil agendée (format UNIX)

```
js      function get_nextWakeUp( )
nodejs  function get_nextWakeUp( )
php     function get_nextWakeUp( )
cpp     s64 get_nextWakeUp( )
m       -(s64) nextWakeUp
pas     function get_nextWakeUp( ): int64
vb      function get_nextWakeUp( ) As Long
cs      long get_nextWakeUp( )
java    long get_nextWakeUp( )
py      def get_nextWakeUp( )
```

Retourne :

un entier représentant la prochaine date/heure de réveil agendée (format UNIX)

En cas d'erreur, déclenche une exception ou retourne `Y_NEXTWAKEUP_INVALID`.

wakeupmonitor→get_powerDuration() YWakeUpMonitor
wakeupmonitor→powerDuration()

Retourne le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

```
js      function get_powerDuration( )
nodejs  function get_powerDuration( )
php     function get_powerDuration( )
cpp     int get_powerDuration( )
m       -(int) powerDuration
pas     function get_powerDuration( ): LongInt
vb      function get_powerDuration( ) As Integer
cs      int get_powerDuration( )
java    int get_powerDuration( )
py      def get_powerDuration( )
cmd     YWakeUpMonitor target get_powerDuration
```

Retourne :

un entier représentant le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement

En cas d'erreur, déclenche une exception ou retourne `Y_POWERDURATION_INVALID`.

wakeupmonitor→get_sleepCountdown() YWakeUpMonitor
wakeupmonitor→sleepCountdown()

Retourne le temps avant le prochain sommeil.

```
js      function get_sleepCountdown( )
nodejs  function get_sleepCountdown( )
php     function get_sleepCountdown( )
cpp     int get_sleepCountdown( )
m       -(int) sleepCountdown
pas     function get_sleepCountdown( ): LongInt
vb      function get_sleepCountdown( ) As Integer
cs      int get_sleepCountdown( )
java    int get_sleepCountdown( )
py      def get_sleepCountdown( )
cmd     YWakeUpMonitor target get_sleepCountdown
```

Retourne :
un entier représentant le temps avant le prochain sommeil

En cas d'erreur, déclenche une exception ou retourne `Y_SLEEPCOUNTDOWN_INVALID`.

wakeupmonitor→get_userData() YWakeUpMonitor
wakeupmonitor→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
nodejs  function get_userData( )
php     function get_userData( )
cpp     void * get_userData( )
m       -(void*) userData
pas     function get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      def get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

wakeupmonitor→get_wakeUpReason() YWakeUpMonitor
wakeupmonitor→wakeUpReason()

Renvoie la raison du dernier réveil.

```
js      function get_wakeUpReason( )
nodejs  function get_wakeUpReason( )
php     function get_wakeUpReason( )
cpp     Y_WAKEUPREASON_enum get_wakeUpReason( )
m       -(Y_WAKEUPREASON_enum) wakeUpReason
pas     function get_wakeUpReason( ): Integer
vb      function get_wakeUpReason( ) As Integer
cs      int get_wakeUpReason( )
java    int get_wakeUpReason( )
py      def get_wakeUpReason( )
cmd     YWakeUpMonitor target get_wakeUpReason
```

Retourne :

une valeur parmi Y_WAKEUPREASON_USBPOWER, Y_WAKEUPREASON_EXTPOWER, Y_WAKEUPREASON_ENDOFSLEEP, Y_WAKEUPREASON_EXTSIG1, Y_WAKEUPREASON_SCHEDULE1 et Y_WAKEUPREASON_SCHEDULE2

En cas d'erreur, déclenche une exception ou retourne Y_WAKEUPREASON_INVALID.

wakeupmonitor→get_wakeUpState() YWakeUpMonitor
wakeupmonitor→wakeUpState()

Revoie l'état actuel du moniteur

```
js      function get_wakeUpState( )
nodejs  function get_wakeUpState( )
php     function get_wakeUpState( )
cpp     Y_WAKEUPSTATE_enum get_wakeUpState( )
m       -(Y_WAKEUPSTATE_enum) wakeUpState
pas     function get_wakeUpState( ): Integer
vb      function get_wakeUpState( ) As Integer
cs      int get_wakeUpState( )
java    int get_wakeUpState( )
py      def get_wakeUpState( )
```

Retourne :

soit `Y_WAKEUPSTATE_SLEEPING`, soit `Y_WAKEUPSTATE_AWAKE`

En cas d'erreur, déclenche une exception ou retourne `Y_WAKEUPSTATE_INVALID`.

wakeupmonitor→isOnline() YWakeUpMonitor

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache du moniteur sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :

`true` si le moniteur est joignable, `false` sinon

wakeupmonitor→isOnline_async() YWakeUpMonitor

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du moniteur sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupmonitor→load() YWakeUpMonitor

Met en cache les valeurs courantes du moniteur, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→load_async() YWakeUpMonitor

Met en cache les valeurs courantes du moniteur, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupmonitor→nextWakeUpMonitor() YWakeUpMonitor

Continue l'énumération des Moniteurs commencée à l'aide de `yFirstWakeUpMonitor()`.

```
js      function nextWakeUpMonitor( )
nodejs  function nextWakeUpMonitor( )
php     function nextWakeUpMonitor( )
cpp     YWakeUpMonitor * nextWakeUpMonitor( )
m       -(YWakeUpMonitor*) nextWakeUpMonitor
pas     function nextWakeUpMonitor( ): TYWakeUpMonitor
vb      function nextWakeUpMonitor( ) As YWakeUpMonitor
cs      YWakeUpMonitor nextWakeUpMonitor( )
java    YWakeUpMonitor nextWakeUpMonitor( )
py      def nextWakeUpMonitor( )
```

Retourne :

un pointeur sur un objet `YWakeUpMonitor` accessible en ligne, ou `null` lorsque l'énumération est terminée.

wakeupmonitor→registerValueCallback() YWakeUpMonitor

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
nodejs  function registerValueCallback( callback)
php     function registerValueCallback( $callback)
cpp     int registerValueCallback( YWakeUpMonitorValueCallback callback)
m       -(int) registerValueCallback : (YWakeUpMonitorValueCallback) callback
pas     function registerValueCallback( callback: TYWakeUpMonitorValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
py      def registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

wakeupmonitor→resetSleepCountDown() YWakeUpMonitor

Réinitialise le compteur de mise en sommeil.

```
js      function resetSleepCountDown( )
nodejs  function resetSleepCountDown( )
php     function resetSleepCountDown( )
cpp     int resetSleepCountDown( )
m       -(int) resetSleepCountDown
pas     function resetSleepCountDown( ): LongInt
vb      function resetSleepCountDown( ) As Integer
cs      int resetSleepCountDown( )
java    int resetSleepCountDown( )
py      def resetSleepCountDown( )
cmd     YWakeUpMonitor target resetSleepCountDown
```

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_logicalName() YWakeUpMonitor
wakeupmonitor→setLogicalName()

Modifie le nom logique du moniteur.

```
js      function set_logicalName( newval)
nodejs  function set_logicalName( newval)
php     function set_logicalName( $newval)
cpp     int set_logicalName( const string& newval)
m       -(int) setLogicalName : (NSString*) newval
pas     function set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
java    int set_logicalName( String newval)
py      def set_logicalName( newval)
cmd     YWakeUpMonitor target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique du moniteur.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_nextWakeUp() YWakeUpMonitor
wakeupmonitor→setNextWakeUp()

Modifie les jours de la semaine où un réveil doit avoir lieu.

```
js      function set_nextWakeUp( newval)
nodejs  function set_nextWakeUp( newval)
php     function set_nextWakeUp( $newval)
cpp     int set_nextWakeUp( s64 newval)
m       -(int) setNextWakeUp : (s64) newval
pas     function set_nextWakeUp( newval: int64): integer
vb      function set_nextWakeUp( ByVal newval As Long) As Integer
cs      int set_nextWakeUp( long newval)
java    int set_nextWakeUp( long newval)
py      def set_nextWakeUp( newval)
cmd     YWakeUpMonitor target set_nextWakeUp newval
```

Paramètres :

newval un entier représentant les jours de la semaine où un réveil doit avoir lieu

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_powerDuration() YWakeUpMonitor
wakeupmonitor→setPowerDuration()

Modifie le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

```
js      function set_powerDuration( newval)
nodejs  function set_powerDuration( newval)
php     function set_powerDuration( $newval)
cpp     int set_powerDuration( int newval)
m       -(int) setPowerDuration : (int) newval
pas     function set_powerDuration( newval: LongInt): integer
vb      function set_powerDuration( ByVal newval As Integer) As Integer
cs      int set_powerDuration( int newval)
java    int set_powerDuration( int newval)
py      def set_powerDuration( newval)
cmd     YWakeUpMonitor target set_powerDuration newval
```

Paramètres :

newval un entier représentant le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_sleepCountdown() YWakeUpMonitor
wakeupmonitor→setSleepCountdown()

Modifie le temps avant le prochain sommeil .

```
js      function set_sleepCountdown( newval)
nodejs  function set_sleepCountdown( newval)
php     function set_sleepCountdown( $newval)
cpp     int set_sleepCountdown( int newval)
m       -(int) setSleepCountdown : (int) newval
pas     function set_sleepCountdown( newval: LongInt): integer
vb      function set_sleepCountdown( ByVal newval As Integer) As Integer
cs      int set_sleepCountdown( int newval)
java    int set_sleepCountdown( int newval)
py      def set_sleepCountdown( newval)
cmd     YWakeUpMonitor target set_sleepCountdown newval
```

Paramètres :

newval un entier représentant le temps avant le prochain sommeil

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_userData() YWakeUpMonitor
wakeupmonitor→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

wakeupmonitor→sleep() YWakeUpMonitor

Déclenche une mise en sommeil jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

```
js      function sleep( secBeforeSleep)
nodejs  function sleep( secBeforeSleep)
php     function sleep( $secBeforeSleep)
cpp     int sleep( int secBeforeSleep)
m       -(int) sleep : (int) secBeforeSleep
pas     function sleep( secBeforeSleep: LongInt): LongInt
vb      function sleep( ) As Integer
cs      int sleep( int secBeforeSleep)
java    int sleep( int secBeforeSleep)
py      def sleep( secBeforeSleep)
cmd     YWakeUpMonitor target sleep secBeforeSleep
```

Paramètres :

secBeforeSleep nombre de seconde avant la mise en sommeil

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→sleepFor() YWakeUpMonitor

Déclenche une mise en sommeil pour un temps donné ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

```
js      function sleepFor( secUntilWakeUp, secBeforeSleep)
nodejs  function sleepFor( secUntilWakeUp, secBeforeSleep)
php     function sleepFor( $secUntilWakeUp, $secBeforeSleep)
cpp     int sleepFor( int secUntilWakeUp, int secBeforeSleep)
m       -(int) sleepFor : (int) secUntilWakeUp : (int) secBeforeSleep
pas     function sleepFor( secUntilWakeUp: LongInt,
                           secBeforeSleep: LongInt): LongInt
vb      function sleepFor( ) As Integer
cs      int sleepFor( int secUntilWakeUp, int secBeforeSleep)
java    int sleepFor( int secUntilWakeUp, int secBeforeSleep)
py      def sleepFor( secUntilWakeUp, secBeforeSleep)
cmd     YWakeUpMonitor target sleepFor secUntilWakeUp secBeforeSleep
```

Le compte à rebours avant la mise en sommeil peut être annulé grâce à resetSleepCountDown.

Paramètres :

secUntilWakeUp nombre de secondes avant le prochain réveil

secBeforeSleep nombre de secondes avant la mise en sommeil

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→sleepUntil() YWakeUpMonitor

Déclenche une mise en sommeil jusqu'à une date donnée ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

```
js      function sleepUntil( wakeUpTime, secBeforeSleep)
nodejs  function sleepUntil( wakeUpTime, secBeforeSleep)
php     function sleepUntil( $wakeUpTime, $secBeforeSleep)
cpp     int sleepUntil( int wakeUpTime, int secBeforeSleep)
m       -(int) sleepUntil : (int) wakeUpTime : (int) secBeforeSleep
pas     function sleepUntil( wakeUpTime: LongInt,
                             secBeforeSleep: LongInt): LongInt
vb      function sleepUntil( ) As Integer
cs      int sleepUntil( int wakeUpTime, int secBeforeSleep)
java    int sleepUntil( int wakeUpTime, int secBeforeSleep)
py      def sleepUntil( wakeUpTime, secBeforeSleep)
cmd     YWakeUpMonitor target sleepUntil wakeUpTime secBeforeSleep
```

Le compte à rebours avant la mise en sommeil peut être annulé grâce à resetSleepCountDown.

Paramètres :

wakeUpTime	date/heure du réveil (format UNIX)
secBeforeSleep	nombre de secondes avant la mise en sommeil

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→wait_async() YWakeUpMonitor

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

wakeupmonitor→wakeUp() YWakeUpMonitor

Force un réveil.

```
js      function wakeUp( )
nodejs  function wakeUp( )
php     function wakeUp( )
cpp     int wakeUp( )
m       -(int) wakeUp
pas     function wakeUp( ): LongInt
vb      function wakeUp( ) As Integer
cs      int wakeUp( )
java    int wakeUp( )
py      def wakeUp( )
cmd     YWakeUpMonitor target wakeUp
```

11.6. Interface de la fonction WakeUpSchedule

La fonction WakeUpSchedule implémente une condition de réveil. Le réveil est spécifiée par un ensemble de mois et/ou jours et/ou heures et/ou minutes où il doit se produire.

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_wakeupschedule.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YWakeUpSchedule = yoctolib.YWakeUpSchedule;
php     require_once('yocto_wakeupschedule.php');
cpp     #include "yocto_wakeupschedule.h"
m       #import "yocto_wakeupschedule.h"
pas     uses yocto_wakeupschedule;
vb      yocto_wakeupschedule.vb
cs      yocto_wakeupschedule.cs
java    import com.yoctopuce.YoctoAPI.YWakeUpSchedule;
py      from yocto_wakeupschedule import *
```

Fonction globales
yFindWakeUpSchedule(**func**) Permet de retrouver un réveil agendé d'après un identifiant donné.
yFirstWakeUpSchedule() Commence l'énumération des réveils agendés accessibles par la librairie.
Méthodes des objets YWakeUpSchedule
wakeupschedule→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance du réveil agendé au format TYPE(NAME)=SERIAL.FUNCTIONID.
wakeupschedule→get_advertisedValue() Retourne la valeur courante du réveil agendé (pas plus de 6 caractères).
wakeupschedule→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.
wakeupschedule→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.
wakeupschedule→get_friendlyName() Retourne un identifiant global du réveil agendé au format NOM_MODULE.NOM_FONCTION.
wakeupschedule→get_functionDescriptor() Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
wakeupschedule→get_functionId() Retourne l'identifiant matériel du réveil agendé, sans référence au module.
wakeupschedule→get_hardwareId() Retourne l'identifiant matériel unique du réveil agendé au format SERIAL.FUNCTIONID.
wakeupschedule→get_hours() Retourne les heures où le réveil est actif..
wakeupschedule→get_logicalName() Retourne le nom logique du réveil agendé.
wakeupschedule→get_minutes() Retourne toutes les minutes de chaque heure où le réveil est actif.
wakeupschedule→get_minutesA()

Retourne les minutes de l'interval 00-29 de chaque heures où le réveil est actif.

wakeupschedule→get_minutesB()
Retourne les minutes de l'interval 30-59 de chaque heure où le réveil est actif.

wakeupschedule→get_module()
Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

wakeupschedule→get_module_async(**callback**, **context**)
Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

wakeupschedule→get_monthDays()
Retourne les jours du mois où le réveil est actif..

wakeupschedule→get_months()
Retourne les mois où le réveil est actif..

wakeupschedule→get_nextOccurence()
Retourne la date/heure de la prochaine occurence de réveil

wakeupschedule→get_userData()
Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

wakeupschedule→get_weekDays()
Retourne les jours de la semaine où le réveil est actif..

wakeupschedule→isOnline()
Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

wakeupschedule→isOnline_async(**callback**, **context**)
Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

wakeupschedule→load(**msValidity**)
Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

wakeupschedule→load_async(**msValidity**, **callback**, **context**)
Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

wakeupschedule→nextWakeUpSchedule()
Continue l'énumération des réveils agendés commencée à l'aide de `yFirstWakeUpSchedule()`.

wakeupschedule→registerValueCallback(**callback**)
Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

wakeupschedule→set_hours(**newval**, **newval**)
Modifie les heures où un réveil doit avoir lieu

wakeupschedule→set_logicalName(**newval**)
Modifie le nom logique du réveil agendé.

wakeupschedule→set_minutes(**bitmap**)
Modifie toutes les minutes où un réveil doit avoir lieu

wakeupschedule→set_minutesA(**newval**, **newval**)
Modifie les minutes de l'interval 00-29 où un réveil doit avoir lieu

wakeupschedule→set_minutesB(**newval**)
Modifie les minutes de l'interval 30-59 où un réveil doit avoir lieu.

wakeupschedule→set_monthDays(**newval**, **newval**)
Modifie les jours du mois où un réveil doit avoir lieu

wakeupschedule→set_months(**newval**, **newval**)
Modifie les mois où un réveil doit avoir lieu

wakeupschedule→set_userData(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

wakeupschedule→set_weekDays(newval, newval)

Modifie les jours de la semaine où un réveil doit avoir lieu

wakeupschedule→wait_async(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YWakeUpSchedule.FindWakeUpSchedule() YWakeUpSchedule yFindWakeUpSchedule()

Permet de retrouver un réveil agendé d'après un identifiant donné.

```
js      function yFindWakeUpSchedule( func)
nodejs  function FindWakeUpSchedule( func)
php     function yFindWakeUpSchedule( $func)
cpp     YWakeUpSchedule* yFindWakeUpSchedule( const string& func)
m       +(YWakeUpSchedule*) FindWakeUpSchedule :(NSString*) func
pas     function yFindWakeUpSchedule( func: string): TYWakeUpSchedule
vb      function yFindWakeUpSchedule( ByVal func As String) As YWakeUpSchedule
cs      YWakeUpSchedule FindWakeUpSchedule( string func)
java    YWakeUpSchedule FindWakeUpSchedule( String func)
py      def FindWakeUpSchedule( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le réveil agendé soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWakeUpSchedule.isOnline()` pour tester si le réveil agendé est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

func une chaîne de caractères qui référence le réveil agendé sans ambiguïté

Retourne :

un objet de classe `YWakeUpSchedule` qui permet ensuite de contrôler le réveil agendé.

YWakeUpSchedule.FirstWakeUpSchedule() YWakeUpSchedule yFirstWakeUpSchedule()

Commence l'énumération des réveils agendés accessibles par la librairie.

```
js      function yFirstWakeUpSchedule( )
nodejs  function FirstWakeUpSchedule( )
php     function yFirstWakeUpSchedule( )
cpp     YWakeUpSchedule* yFirstWakeUpSchedule( )
m       +(YWakeUpSchedule*) FirstWakeUpSchedule
pas     function yFirstWakeUpSchedule( ): TYWakeUpSchedule
vb      function yFirstWakeUpSchedule( ) As YWakeUpSchedule
cs      YWakeUpSchedule FirstWakeUpSchedule( )
java    YWakeUpSchedule FirstWakeUpSchedule( )
py      def FirstWakeUpSchedule( )
```

Utiliser la fonction `YWakeUpSchedule.nextWakeUpSchedule()` pour itérer sur les autres réveils agendés.

Retourne :

un pointeur sur un objet `YWakeUpSchedule`, correspondant au premier réveil agendé accessible en ligne, ou `null` si il n'y a pas de réveils agendés disponibles.

wakeupschedule→describe() YWakeUpSchedule

Retourne un court texte décrivant de manière non-ambigüe l'instance du réveil agendé au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant le réveil agendé (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

wakeupschedule→get_advertisedValue() YWakeUpSchedule
wakeupschedule→advertisedValue()

Retourne la valeur courante du réveil agendé (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YWakeUpSchedule target get_advertisedValue
```

Retourne :

une chaîne de caractères représentant la valeur courante du réveil agendé (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

wakeupschedule→get_errorMessage() YWakeUpSchedule
wakeupschedule→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

js	function **get_errorMessage**( )
nodejs	function **get_errorMessage**( )
php	function **get_errorMessage**( )
cpp	string **get_errorMessage**( )
m	-(NSString*) errorMessage
pas	function **get_errorMessage**( ): string
vb	function **get_errorMessage**( ) As String
cs	string **get_errorMessage**( )
java	String **get_errorMessage**( )
py	def **get_errorMessage**( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du réveil agendé.

wakeupschedule→get_errorType() YWakeUpSchedule
wakeupschedule→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du réveil agendé.

wakeupschedule→get_friendlyName() YWakeUpSchedule
wakeupschedule→friendlyName()

Retourne un identifiant global du réveil agendé au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et du réveil agendé si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du réveil agendé (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant le réveil agendé en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

wakeupschedule→get_functionDescriptor() YWakeUpSchedule
wakeupschedule→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
nodejs  function get_functionDescriptor( )
php     function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     function get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      def get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :

un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

wakeupschedule→get_functionId() YWakeUpSchedule
wakeupschedule→functionId()

Retourne l'identifiant matériel du réveil agendé, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

Retourne :

une chaîne de caractères identifiant le réveil agendé (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

wakeupschedule→get_hardwareId() YWakeUpSchedule
wakeupschedule→hardwareId()

Retourne l'identifiant matériel unique du réveil agendé au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
nodejs  function get_hardwareId( )
php     function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
java    String get_hardwareId( )
py      def get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du réveil agendé (par example RELAYLO1-123456.relay1).

Retourne :

une chaîne de caractères identifiant le réveil agendé (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

wakeupschedule→get_hours() YWakeUpSchedule
wakeupschedule→hours()

Retourne les heures où le réveil est actif..

```
js      function get_hours( )
nodejs  function get_hours( )
php     function get_hours( )
cpp     int get_hours( )
m       -(int) hours
pas     function get_hours( ): LongInt
vb      function get_hours( ) As Integer
cs      int get_hours( )
java    int get_hours( )
py      def get_hours( )
cmd     YWakeUpSchedule target get_hours
```

Retourne :
un entier représentant les heures où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_HOURS_INVALID`.

wakeupschedule→get_logicalName() YWakeUpSchedule
wakeupschedule→logicalName()

Retourne le nom logique du réveil agendé.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YWakeUpSchedule target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique du réveil agendé.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

wakeupschedule→get_minutes() YWakeUpSchedule
wakeupschedule→minutes()

Retourne toutes les minutes de chaque heure où le réveil est actif.

```
js      function get_minutes( )
nodejs  function get_minutes( )
php     function get_minutes( )
cpp     s64 get_minutes( )
m       -(s64) minutes
pas     function get_minutes( ): int64
vb      function get_minutes( ) As Long
cs      long get_minutes( )
java    long get_minutes( )
py      def get_minutes( )
cmd     YWakeUpSchedule target get_minutes
```

wakeupschedule→get_minutesA() YWakeUpSchedule
wakeupschedule→minutesA()

Retourne les minutes de l'interval 00-29 de chaque heures où le réveil est actif.

```
js      function get_minutesA( )
nodejs  function get_minutesA( )
php     function get_minutesA( )
cpp     int get_minutesA( )
m       -(int) minutesA
pas     function get_minutesA( ): LongInt
vb      function get_minutesA( ) As Integer
cs      int get_minutesA( )
java    int get_minutesA( )
py      def get_minutesA( )
cmd     YWakeUpSchedule target get_minutesA
```

Retourne :

un entier représentant les minutes de l'interval 00-29 de chaque heures où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_MINUTESA_INVALID`.

wakeupschedule→get_minutesB() YWakeUpSchedule
wakeupschedule→minutesB()

Retourne les minutes de l'interval 30-59 de chaque heure où le réveil est actif.

```
js      function get_minutesB( )
nodejs  function get_minutesB( )
php     function get_minutesB( )
cpp     int get_minutesB( )
m       -(int) minutesB
pas     function get_minutesB( ): LongInt
vb      function get_minutesB( ) As Integer
cs      int get_minutesB( )
java    int get_minutesB( )
py      def get_minutesB( )
cmd     YWakeUpSchedule target get_minutesB
```

Retourne :

un entier représentant les minutes de l'interval 30-59 de chaque heure où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_MINUTESB_INVALID`.

wakeupschedule→get_module() YWakeUpSchedule
wakeupschedule→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
nodejs  function get_module( )
php     function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     function get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
java    YModule get_module( )
py      def get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :

une instance de YModule

wakeupschedule→get_module_async() YWakeUpSchedule
wakeupschedule→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module_async( callback, context)
nodejs  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupschedule→get_monthDays() YWakeUpSchedule
wakeupschedule→monthDays()

Retourne les jours du mois où le réveil est actif..

```
js      function get_monthDays( )
nodejs  function get_monthDays( )
php     function get_monthDays( )
cpp     int get_monthDays( )
m       -(int) monthDays
pas     function get_monthDays( ): LongInt
vb      function get_monthDays( ) As Integer
cs      int get_monthDays( )
java    int get_monthDays( )
py      def get_monthDays( )
cmd     YWakeUpSchedule target get_monthDays
```

Retourne :
un entier représentant les jours du mois où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_MONTHDAYS_INVALID`.

wakeupschedule→get_months() YWakeUpSchedule
wakeupschedule→months()

Retourne les mois où le réveil est actif..

```
js      function get_months( )
nodejs  function get_months( )
php     function get_months( )
cpp     int get_months( )
m       -(int) months
pas     function get_months( ): LongInt
vb      function get_months( ) As Integer
cs      int get_months( )
java    int get_months( )
py      def get_months( )
cmd     YWakeUpSchedule target get_months
```

Retourne :
un entier représentant les mois où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_MONTHS_INVALID`.

wakeupschedule→get_nextOccurence() YWakeUpSchedule
wakeupschedule→nextOccurence()

Retourne la date/heure de la prochaine occurence de réveil

```
js      function get_nextOccurence( )
nodejs  function get_nextOccurence( )
php     function get_nextOccurence( )
cpp     s64 get_nextOccurence( )
m       -(s64) nextOccurence
pas     function get_nextOccurence( ): int64
vb      function get_nextOccurence( ) As Long
cs      long get_nextOccurence( )
java    long get_nextOccurence( )
py      def get_nextOccurence( )
```

Retourne :

un entier représentant la date/heure de la prochaine occurence de réveil

En cas d'erreur, déclenche une exception ou retourne `Y_NEXTOCCURENCE_INVALID`.

wakeupschedule→get_userData() YWakeUpSchedule
wakeupschedule→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function **get_userData**( )
nodejs	function **get_userData**( )
php	function **get_userData**( )
cpp	void * **get_userData**( )
m	-(void*) userData
pas	function **get_userData**( ): Tobject
vb	function **get_userData**( ) As Object
cs	object **get_userData**( )
java	Object **get_userData**( )
py	def **get_userData**( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

wakeupschedule→get_weekDays() YWakeUpSchedule
wakeupschedule→weekDays()

Retourne les jours de la semaine où le réveil est actif..

```
js      function get_weekDays( )
nodejs  function get_weekDays( )
php     function get_weekDays( )
cpp     int get_weekDays( )
m       -(int) weekDays
pas     function get_weekDays( ): LongInt
vb      function get_weekDays( ) As Integer
cs      int get_weekDays( )
java    int get_weekDays( )
py      def get_weekDays( )
cmd     YWakeUpSchedule target get_weekDays
```

Retourne :
un entier représentant les jours de la semaine où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_WEEKDAYS_INVALID`.

wakeupschedule→isOnline() YWakeUpSchedule

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache du réveil agendé sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :

`true` si le réveil agendé est joignable, `false` sinon

wakeupschedule→isOnline_async() YWakeUpSchedule

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du réveil agendé sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupschedule→load() YWakeUpSchedule

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→load_async() YWakeUpSchedule

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupschedule→nextWakeUpSchedule() YWakeUpSchedule

Continue l'énumération des réveils agendés commencée à l'aide de `yFirstWakeUpSchedule()`.

js	function **nextWakeUpSchedule**( )
nodejs	function **nextWakeUpSchedule**( )
php	function **nextWakeUpSchedule**( )
cpp	YWakeUpSchedule * **nextWakeUpSchedule**( )
m	-(YWakeUpSchedule*) **nextWakeUpSchedule**
pas	function **nextWakeUpSchedule**( ): TYWakeUpSchedule
vb	function **nextWakeUpSchedule**( ) As YWakeUpSchedule
cs	YWakeUpSchedule **nextWakeUpSchedule**( )
java	YWakeUpSchedule **nextWakeUpSchedule**( )
py	def **nextWakeUpSchedule**( )

Retourne :

un pointeur sur un objet `YWakeUpSchedule` accessible en ligne, ou `null` lorsque l'énumération est terminée.

wakeupschedule→registerValueCallback() YWakeUpSchedule

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
nodejs  function registerValueCallback( callback)
php     function registerValueCallback( $callback)
cpp     int registerValueCallback( YWakeUpScheduleValueCallback callback)
m       -(int) registerValueCallback : (YWakeUpScheduleValueCallback) callback
pas     function registerValueCallback( callback: TYWakeUpScheduleValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
py      def registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

wakeupschedule→set_hours() YWakeUpSchedule
wakeupschedule→setHours()

Modifie les heures où un réveil doit avoir lieu

```
js      function set_hours( newval)
nodejs  function set_hours( newval)
php     function set_hours( $newval)
cpp     int set_hours( int newval)
m       -(int) setHours : (int) newval
pas     function set_hours( newval: LongInt): integer
vb      function set_hours( ByVal newval As Integer) As Integer
cs      int set_hours( int newval)
java    int set_hours( int newval)
py      def set_hours( newval)
cmd     YWakeUpSchedule target set_hours newval
```

Paramètres :

newval un entier représentant les heures où un réveil doit avoir lieu

newval un entier

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_logicalName() YWakeUpSchedule
wakeupschedule→setLogicalName()

Modifie le nom logique du réveil agendé.

js	function **set_logicalName**(**newval**)
nodejs	function **set_logicalName**(**newval**)
php	function **set_logicalName**(**$newval**)
cpp	int **set_logicalName**(const string& **newval**)
m	-(int) setLogicalName : (NSString*) **newval**
pas	function **set_logicalName**(**newval**: string): integer
vb	function **set_logicalName**(ByVal **newval** As String) As Integer
cs	int **set_logicalName**(string **newval**)
java	int **set_logicalName**(String **newval**)
py	def **set_logicalName**(**newval**)
cmd	YWakeUpSchedule **target set_logicalName newval**

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique du réveil agendé.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_minutes() YWakeUpSchedule
wakeupschedule→setMinutes()

Modifie toutes les minutes où un réveil doit avoir lieu

```
js      function set_minutes( bitmap)
nodejs  function set_minutes( bitmap)
php     function set_minutes( $bitmap)
cpp     int set_minutes( s64 bitmap)
m       -(int) setMinutes : (s64) bitmap
pas     function set_minutes( bitmap: int64): LongInt
vb      function set_minutes( ) As Integer
cs      int set_minutes( long bitmap)
java    int set_minutes( long bitmap)
py      def set_minutes( bitmap)
cmd     YWakeUpSchedule target set_minutes bitmap
```

Paramètres :

bitmap Minutes 00-59 de chaque heure où le réveil est actif.

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_minutesA() YWakeUpSchedule
wakeupschedule→setMinutesA()

Modifie les minutes de l'interval 00-29 où un réveil doit avoir lieu

```
js      function set_minutesA( newval)
nodejs  function set_minutesA( newval)
php     function set_minutesA( $newval)
cpp     int set_minutesA( int newval)
m       -(int) setMinutesA : (int) newval
pas     function set_minutesA( newval: LongInt): integer
vb      function set_minutesA( ByVal newval As Integer) As Integer
cs      int set_minutesA( int newval)
java    int set_minutesA( int newval)
py      def set_minutesA( newval)
cmd     YWakeUpSchedule target set_minutesA newval
```

Paramètres :

newval un entier représentant les minutes de l'interval 00-29 où un réveil doit avoir lieu

newval un entier

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_minutesB() YWakeUpSchedule
wakeupschedule→setMinutesB()

Modifie les minutes de l'interval 30-59 où un réveil doit avoir lieu.

```
js      function set_minutesB( newval)
nodejs  function set_minutesB( newval)
php     function set_minutesB( $newval)
cpp     int set_minutesB( int newval)
m       -(int) setMinutesB : (int) newval
pas     function set_minutesB( newval: LongInt): integer
vb      function set_minutesB( ByVal newval As Integer) As Integer
cs      int set_minutesB( int newval)
java    int set_minutesB( int newval)
py      def set_minutesB( newval)
cmd     YWakeUpSchedule target set_minutesB newval
```

Paramètres :

newval un entier représentant les minutes de l'interval 30-59 où un réveil doit avoir lieu

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_monthDays() YWakeUpSchedule
wakeupschedule→setMonthDays()

Modifie les jours du mois où un réveil doit avoir lieu

```
js      function set_monthDays( newval)
nodejs  function set_monthDays( newval)
php     function set_monthDays( $newval)
cpp     int set_monthDays( int newval)
m       -(int) setMonthDays : (int) newval
pas     function set_monthDays( newval: LongInt): integer
vb      function set_monthDays( ByVal newval As Integer) As Integer
cs      int set_monthDays( int newval)
java    int set_monthDays( int newval)
py      def set_monthDays( newval)
cmd     YWakeUpSchedule target set_monthDays newval
```

Paramètres :

newval un entier représentant les jours du mois où un réveil doit avoir lieu

newval un entier

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_months() YWakeUpSchedule
wakeupschedule→setMonths()

Modifie les mois où un réveil doit avoir lieu

```
js      function set_months( newval)
nodejs  function set_months( newval)
php     function set_months( $newval)
cpp     int set_months( int newval)
m       -(int) setMonths : (int) newval
pas     function set_months( newval: LongInt): integer
vb      function set_months( ByVal newval As Integer) As Integer
cs      int set_months( int newval)
java    int set_months( int newval)
py      def set_months( newval)
cmd     YWakeUpSchedule target set_months newval
```

Paramètres :

newval un entier représentant les mois où un réveil doit avoir lieu

newval un entier

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_userData() YWakeUpSchedule
wakeupschedule→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

wakeupschedule→set_weekDays() YWakeUpSchedule
wakeupschedule→setWeekDays()

Modifie les jours de la semaine où un réveil doit avoir lieu

```
js      function set_weekDays( newval)
nodejs  function set_weekDays( newval)
php     function set_weekDays( $newval)
cpp     int set_weekDays( int newval)
m       -(int) setWeekDays : (int) newval
pas     function set_weekDays( newval: LongInt): integer
vb      function set_weekDays( ByVal newval As Integer) As Integer
cs      int set_weekDays( int newval)
java    int set_weekDays( int newval)
py      def set_weekDays( newval)
cmd     YWakeUpSchedule target set_weekDays newval
```

Paramètres :

newval un entier représentant les jours de la semaine où un réveil doit avoir lieu

newval un entier

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→wait_async() YWakeUpSchedule

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

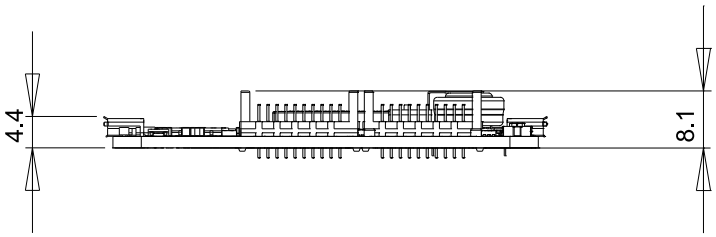
Retourne :

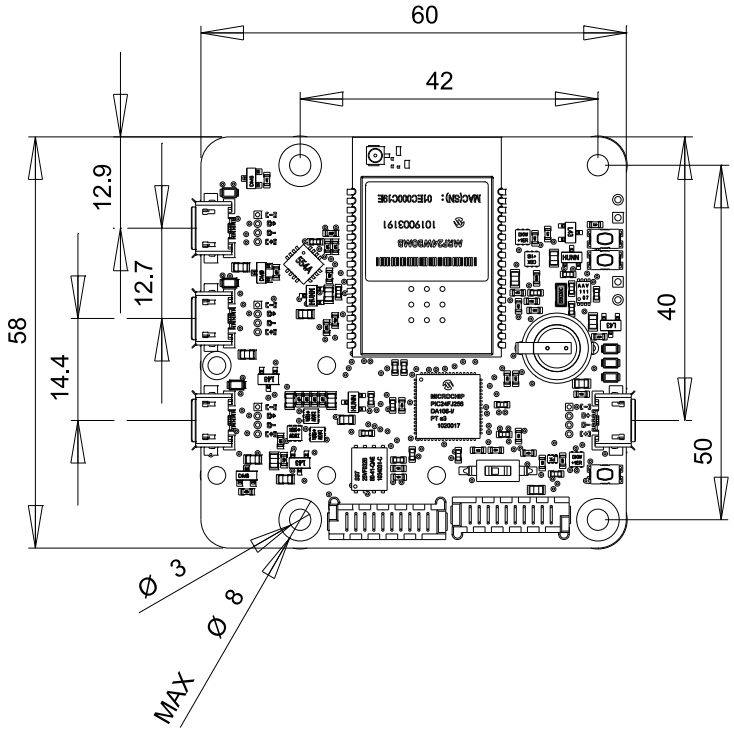
rien du tout.

12. Caractéristiques

Vous trouverez résumées ci dessous les principales caractéristiques techniques de votre module YoctoHub-Wireless

Epaisseur	8.1 mm
Largeur	58 mm
Longueur	60 mm
Poids	30.1 g
Connecteur USB	micro-B
Canaux	3 ports
Courant Max (continu)	2 A
Consommation USB	110 mA
Sytème d'exploitation supportés	Windows, Linux (Intel + ARM), Mac OS X, Android
Connection réseau	802.11b
Drivers	Fonctionne sans driver
API / SDK / Librairie (USB+TCP)	C++, Objective-C, C#, VB .NET, Delphi, Python, Java/Android
API / SDK / Librairie (seul.TCP)	Javascript, Node.js, PHP, Java
RoHS	oui
USB Vendor ID	0x24E0
USB Device ID	0x0023
Boîter recommandé	YoctoBox-HubWlan-Transp





All dimensions are in mm
Toutes les dimensions sont en mm

YoctoHub-Wireless

A4

Scale
1:1
Echelle

Index

I

J

L

M

N