

# Recherche

Votre recherche de *domotique* a donné **42** résultats.

Domotique : Ouvrir et fermer ses volets aux horaires du soleil  
(<http://www.geek.org/domotique-ouvrir-et-fermer-ses-volets-aux-horaires-du-soleil-039.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: lundi 26 janvier 2015, 00:53 :: Domotique  
(<http://www.geek.org/category/technologie/domotique>)



Vous avez franchi la première étape qui est l'arduinoisation de votre télécommande pour ouvrir vos volets (/comment-domotiser-ses-volets-radio-pour-moins-de-50-960.html) ? Voici la seconde étape qui consiste à automatiser l'ouverture et la fermeture de vos volets aux horaires du lever et du coucher du soleil.

J'utilise pour cela un simple fichier **NodeJS** se basant sur les deux librairies suivantes :

- **Cron** pour l'automatisation de tâches

- **Suncalc** pour le calcul des horaires de lever et coucher du soleil.

Le programme réalisé est très simple, il programme un job Cron au prochain coucher ou lever du soleil. Le job se limite à réaliser un appel HTTP au service REST exposé par l'Arduino sur lequel est connecté la télécommande des volets. Dès lors que le job se termine, il programme une nouvelle tâche à l'inverse de la tâche qu'il vient d'effectuer.

```
/**
 * DomoGeek v0.1
 * https://github.com/Ltoinel/domogeek
 *
 * Copyright 2014 DomoGeek
 * Released under the Apache License 2.0 (Apache-2.0)
 *
 * @desc: Task that opens and closes my shutters
 */

var request = require('request');
var CronJob = require('cron').CronJob;
var SunCalc = require('suncalc');

//Local require
var config = require('../config');
var multipush = require('../../libs/multipush');

/**
 * Initializes a new Cron job.
 *
 * @param name : the job name
 * @param uri : the uri to request
 * @param timeFunction : the time function to call
 * @param date : the date
 */
function initJob(name, uri, timeFunction, date){

    var time = timeFunction(date);
    console.log("Job initializing : " + name + " at " + time);

    var job = new CronJob(time,
```

```
// OnTick
function(){

    console.log("Job starting : " + name);

    request.get(uri).on('error', function(err) {
        console.error("Shutter API error : " + err);
    });

    multipush.send(config.multipush,"Job
starting",name,config.shutters.channel);

    // Stop the current job
    this.stop();

    // OnCompleted
    },function(){

        console.log("Job stop : " + name);

        // Schedule the next job
        scheduleNextJob(name);

    }, false, config.timezone).start();
}

/**
 * Schedules the next jobs.
 *
 * @param previousJobName The previous job name
 */
function scheduleNextJob(previousJobName){

    // If the previous job was closing the shutters
    if (previousJobName == config.shutters.close.message) {

        // Schedules the task that will open the shutters tomorrow
```

```
        initJob(config.shutters.open.message, config.shutters.open.url,
getOpenTime, getTomorrowDate());

        // If the previous job was opening the shutters
        } else if (previousJobName == config.shutters.open.message){

            // Schedules the task that will close the shutters today
            initJob(config.shutters.close.message, config.shutters.close.url,
getCloseTime, new Date());
        }
    }

/**
 * Return the date of tomorrow.
 */
function getTomorrowDate(){
    var today = new Date();
    var tomorrow = new Date();
    tomorrow.setDate(today.getDate()+1);
    return tomorrow;
}

/**
 * Return the times regarding the configured latitude and longitude.
 *
 * @param date : the date to use for the times calculation.
 */
function getTimes(date){
    var times = SunCalc.getTimes(date, config.shutters.latitude,
config.shutters.longitude);
    return times;
}

/**
 * Return the next time in cron format to open the shutters.
 *
 * @param date : the date to use for the sunrise calculation.
 */
function getOpenTime(date){
```

```

    // Fix configured time
    if (config.shutters.open.time !== undefined){
        return config.shutters.open.time;
    }

    var times = getTimes(date);
    console.log("Get sunrise time for " + date + " => " + times.sunrise);

    return times.sunrise.getSeconds()+" "+times.sunrise.getMinutes()+"
"+times.sunrise.getHours()+" * * *";
}

/**
 * Return the next time in cron format to close the shutters.
 *
 * @param date : the date to use for the sunset calculation.
 */
function getCloseTime(date){

    // Fix configured time
    if (config.shutters.close.time !== undefined){
        return config.shutters.close.time;
    }

    var times = getTimes(date);
    console.log("Get sunset time for " + date + " => " + times.sunset);

    return times.sunset.getSeconds()+" "+times.sunset.getMinutes()+"
"+times.sunset.getHours()+" * * *";
}

// Starts the right job now
var now = new Date();
var times = getTimes(now);

// Sunrise will arrive today, we have to open the shutters today
if (times.sunrise > now){

```

```
        initJob(config.shutters.open.message, config.shutters.open.url, getOpenTime, new
Date());
    }
    // Sunrise was previous, sunset will arrive today, we have to close the shutters today
    else if (times.sunset > now ){
        initJob(config.shutters.close.message, config.shutters.close.url, getCloseTime,
new Date());

    // Sunrise and sunset are past, we have to open the shutters tomorrow
    } else if (times.sunset < now ){
        initJob(config.shutters.open.message, config.shutters.open.url, getOpenTime,
getTomorrowDate());
    }
}
```

Le fichier de configuration nécessaire au bon fonctionnement du programme :

```
// Shutters task service
config.shutters = {};
config.shutters.enabled = true;
config.shutters.latitude = 47.2;
config.shutters.longitude = -1.5;
config.shutters.open = {};
config.shutters.open.time = '0 0 9 * * *'; // If not set, use the sunset time
config.shutters.open.message = 'Ouverture automatique des volets';
config.shutters.open.url = 'http://xxx.xxx.xxx.xxx/up';
config.shutters.close = {};
//config.shutters.close.time = '0 0 22 * * *'; // If not set, use the sunrise time
config.shutters.close.message = 'Fermeture automatique des volets';
config.shutters.close.url = 'http://xxx.xxx.xxx.xxx/down';
config.shutters.channel = 'openkarotz';
```

Ce bout de code est OpenSource, vous pouvez l'enrichir et proposer des améliorations via GitHub (<https://github.com/ltoinel/domogeek/blob/master/apps/scheduler/tasks/shutters.js>).



aucun commentaire (<http://www.geek.org/domotique-ouvrir-et-fermer-ses-volets-aux-horaires-du-soleil-039.html#comments>)

## MQTT : Un standard pour la domotique ? (<http://www.geek.org/mqtt-un-standard-pour-la-domotique-038.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: lundi 26 janvier 2015, 00:40

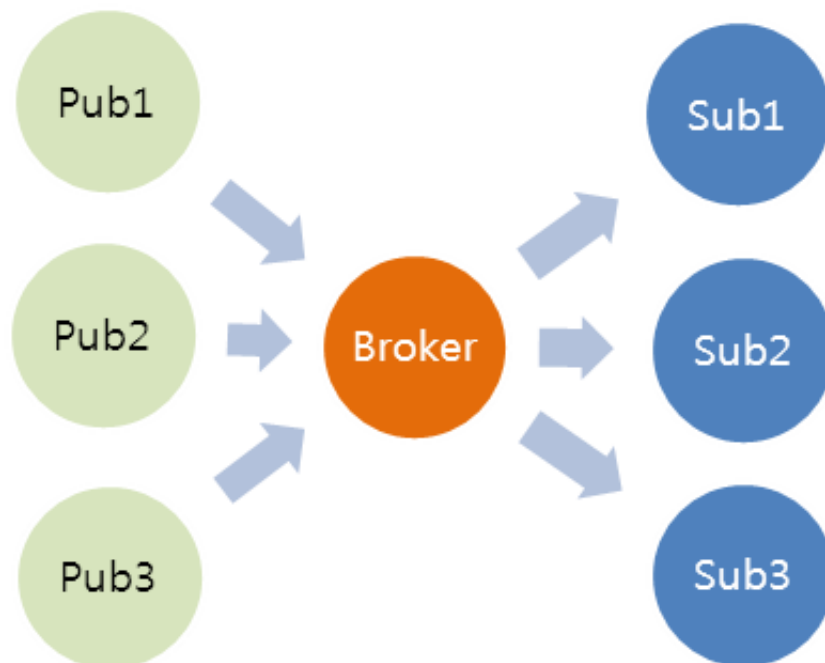


Connaissez-vous le **MQTT** ? Il s'agit d'un standard OASIS (<http://mqtt.org/>) permettant de simplifier la communication machine à machine (M2M), idéalement conçu pour l'Internet des objets.

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios. It is also ideal for mobile applications because of its small size, low power usage, minimised data packets, and efficient distribution of information to one or many receivers.

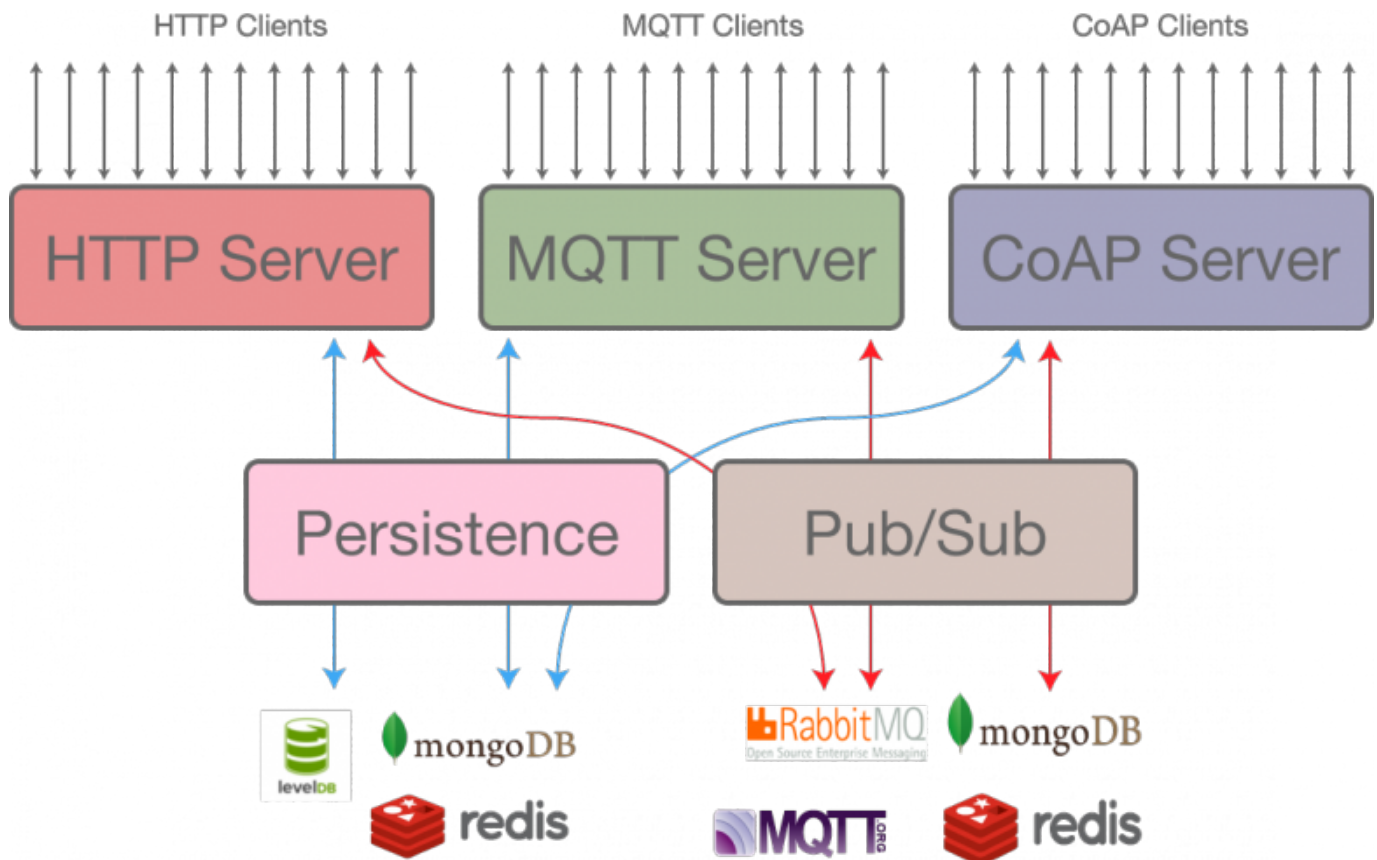
### C'est sympa, mais cela signifie quoi dans la pratique ?

Et bien cela ressemble à un bus d'événement très simple à contacter sur lequel des producteurs et des consommateurs de message peuvent s'abonner.



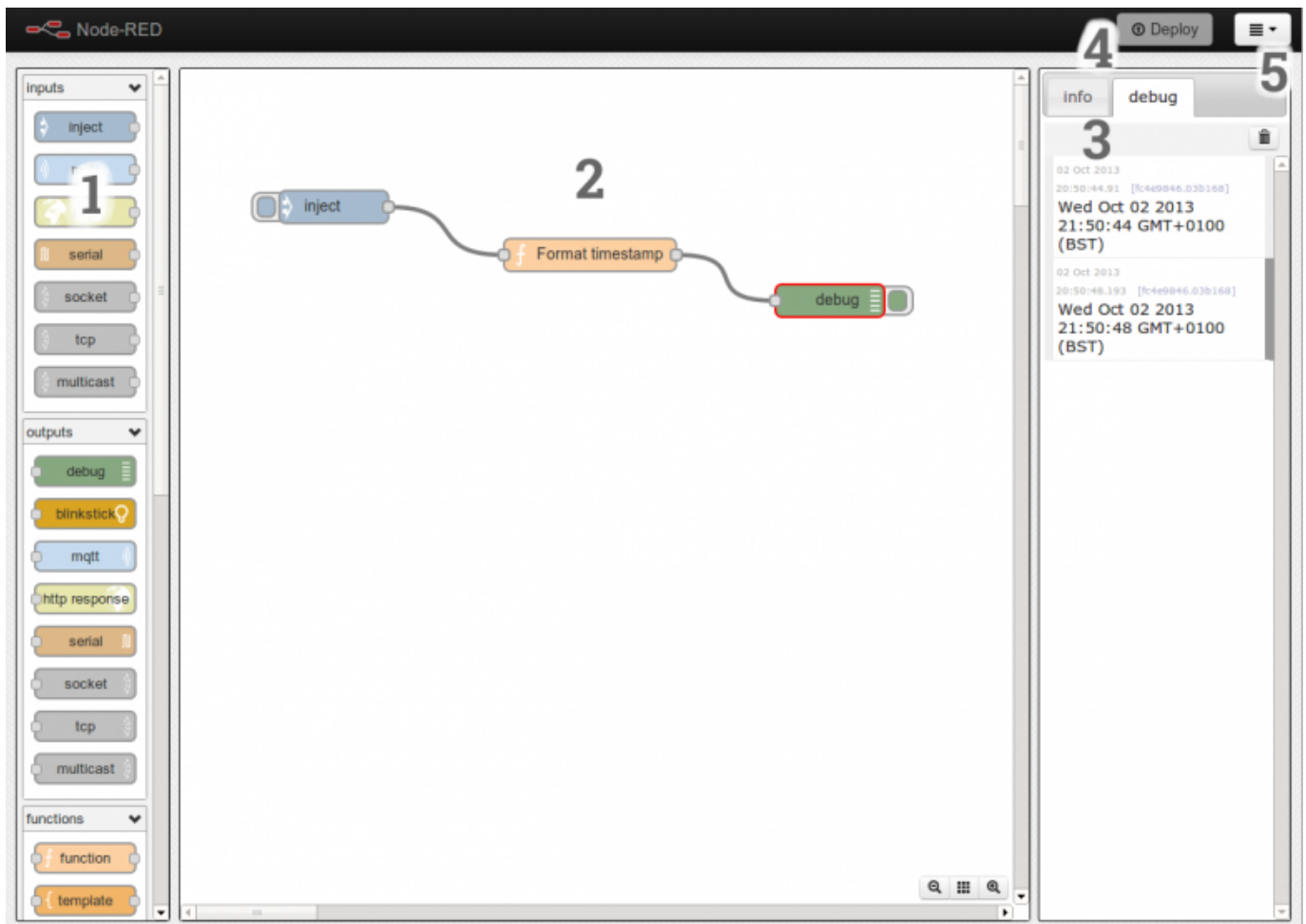
Il existe de nombreuses implémentations de broker MQTT sur Internet. L'un des plus simples à utiliser est très certainement Mosca (<https://github.com/mcollina/mosca>), une implémentation NodeJS de MQTT ultra-configurable et adaptable à tout besoin. Quelques lignes de code Javascript permettent d'instancier son propre serveur MQTT.

Pour des besoins plus lourds, notamment l'exposition REST/COAP des objets, il est possible d'utiliser Eclipse Ponte (<https://eclipse.org/ponte/>) une surcouche de Mosca actuellement en incubation à la fondation Eclipse.



Ensuite si vous souhaitez ajouter de l'intelligence à votre broker MQTT, la solution la plus simple est de connecter une instance Node-Red (<http://nodered.org/>) à votre broker MQTT.





Vous pourrez créer les scénarios de votre choix, orchestrer des actions dès lors qu'un certain événement arrive sur le broker ...

C'est autour de ces deux produits que je suis en train de revoir l'architecture de mon logiciel de domotique pour les geeks : Domogeeek (<https://github.com/ltoinel/domogeeek>). Une branche de développement a été créée spécifiquement pour la migration. Le gros avantage du MQTT est que les producteurs de messages et les consommateurs peuvent être répartis sur différentes machines.

Je vous tiendrai au courant de mes avancements, une v1.0 devrait bientôt voir le jour dès que tous les modules auront été migrés.

aucun commentaire (<http://www.geeek.org/mqtt-un-standard-pour-la-domotique-038.html#comments>)

Votre Karotz vous informe quand vous consommez trop d'énergie (<http://www.geeek.org/votre-karotz-vous-informe-quand-vous-consommez-trop-d-energie-034.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: dimanche 28 décembre 2014, 23:49 :: Domotique

(<http://www.geek.org/category/technologie/domotique>)



Mon projet Domogeeek (</domogeeek-domotique-nodejs-raspberry-pi-011.html>) (box domotique NodeJS sur Raspberry) avance petit à petit, mon dernier hack est de faire parler mon Karotz (</tag/Karotz>) quand ma maison consomme trop d'électricité.

J'ai pour cela installé un compteur d'électricité **Z-Wave HEM2 de chez Aeon** directement sur mon installation électrique. J'ai connecté la pince ampérométrique du module directement sur la phase arrivant sur mon panneau, ce qui me permet d'obtenir la consommation électrique exacte de la maison.



J'ai ensuite développé en quelques lignes de code un plugin Javascript (<https://github.com/ltoinel/domogeeek/blob/master/apps/zwavebus/listeners/power.js>) pour le module ZwaveBus (<https://github.com/ltoinel/domogeeek>) de mon projet Domogeeek (<https://github.com/ltoinel/domogeeek>).

Ce module de quelques lignes est capable de faire parler le **Karotz** dès qu'un rapport de consommation est transmis par le module de chez Aeon sur la base des paramètres d'émission de rapport définis.

```
/**
 * DomoGeeek v0.1
 * https://github.com/ltoinel/domogeeek
 *
 * Copyright 2014 DomoGeeek
 * Released under the Apache License 2.0 (Apache-2.0)
 *
 * @desc: Power module for the Aeon HEM2 power energy meter
 */

// Local require
var bus = require( '../bus' );
var config = require('../../config');
var openkarotz = require('../../libs/openkarotz');

// The command to listen
var COMMAND_CLASS_METER = 50;
```

```

// RGB to Hex
function rgbToHex(r, g, b) {
    return ((1 << 24) + (r << 16) + (g << 8) + b).toString(16).slice(1);
}

/**
 * We listen for a COMMAND_CLASS_METER event.
 * This event is sent by the Aeon HEM2 power energy meter.
 */
bus.on(COMMAND_CLASS_METER, function(nodeid, value){

    if (nodeid == 7 && value['label'] == "Power"){
        var power = value['value'];
        console.log("Saving the power value : " + power);

        // Max possible consume is 12000 Wh
        var n = power / 12000 * 100;
        var red = (255 * n) / 100;
        var green = (255 * (100 - n)) / 100;
        var blue = 0;

        var color = rgbToHex(red,green,blue);
        console.log("Color : " + color);

        // Change the OpenKarotz led
        openkarotz.led(config.openkarotz, color);

        // Make the OpenKarotz talking
        openkarotz.talk(config.openkarotz, "La maison consomme "+ power + "
Watt");

    }

});

```

En plus de faire parler le lapin, le module est capable de faire changer la couleur de **la LED du lapin**. Vert signifie une consommation faible et rouge signifie une consommation importante.

C'est ludique et cela permet de connaître la consommation des appareils que l'on allume. Il est aussi possible de donner en temps réel le coût de l'énergie par heure. Il suffit pour cela de diviser la consommation en Watt par 1000 et de la multiplier par 0,13 pour avoir un prix en euro.

$3000 \text{ Watt sur une heure} = 3 \text{ Kwh} = 3 \times 0,13 \text{ euros} = 0,39 \text{ €/h} = 0,39 \times 24 \times 365 = 3416 \text{ € / an}$

Dans l'exemple ci-dessous, mon ballon d'eau chaude vient tout juste de s'allumer, ce qui explique la consommation importante. Voici en vidéo, le rendu du plugin, il est aussi possible de faire tourner les oreilles du lapin lors de la réception d'un rapport.



3 commentaires (<http://www.geek.org/votre-karotz-vous-informe-quand-vous-consommez-trop-d-energie-034.html#comments>)

## Karotz : Sa mort est annoncée ! (<http://www.geek.org/karotz-mort-019.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: jeudi 30 octobre 2014, 23:32 :: Domotique (<http://www.geek.org/category/technologie/domotique>)



J'avais prédit la mort prochaine du Karotz il y a un an sur ce blog ([/karotz-un-firmware-libre-pour-les-geeks-850.html](#)), la date de sa mort vient d'être annoncée officiellement par Bruno Maisonnier, le PDG d'Aldebaran.

Le Karotz ([/tag/Karotz](#)) ne sera plus maintenu par la société Aldebaran et les serveurs qui permettent de tenir en vie le Karotz ([/tag/Karotz](#)) s'arrêteront à partir du **18 février 2015**. A l'heure d'aujourd'hui, seuls 10 % des Karotz ([/tag/Karotz](#)) commercialisés utilisent encore les serveurs du groupe Aldebaran.

Aldebaran refuse aujourd'hui de proposer l'OpenKarotz ([/openkarotz-liberez-votre-lapin-du-cloud-974.html](#)) comme solution de remplacement de **firmware** pour rendre le lapin autonome et réfléchit sur un modèle OpenSource ([/tag/OpenSource](#)), mais doute sur la durabilité d'un tel modèle sans aucun investissement du groupe. Traduction, il ne vous reste plus que **3 mois** pour mettre à jour le firmware du Karotz ([/tag/Karotz](#)) vers celui de l'OpenKarotz ([/openkarotz-liberez-votre-lapin-du-cloud-974.html](#)) sans aucune garantie de la part d'Aldebaran.



La mort du Karotz (/tag/Karotz) suite à **2 acquisitions consécutives** de la société Violet qui a créé le Nabaztag (/tag/Nabaztag) (grand-père du Karotz) et le Nabaztag Tag (papa du Karotz) montre que le modèle économique de ce type d'objet est loin d'être simple. Les coûts de maintenance des serveurs sur le cloud (/tag/cloud) pour faire fonctionner ces objets connectés sont importants. Pour assurer une rentabilité à des objets connectés, il faut savoir monétiser des services à ses utilisateurs où alors trouver de la valeur aux données produites par ces objets.

Les tentatives de commercialisation d'abonnements par Violet ont échoué, le modèle d'App Store (/tag/App%20Store) avec des applications payantes n'a lui aussi pas abouti. Le seul moyen de gagner de l'argent sur ce type d'objet est d'éviter de rendre l'objet adhérent à toute infrastructure sur le cloud tant qu'un modèle économique sûr et durable n'ait été validé et d'augmenter autant que possible le prix de sa commercialisation.

Le manque de services utiles proposés sur le lapin et le manque de qualité des applications proposées font que beaucoup d'acheteurs n'utilisent plus le lapin. Aujourd'hui, la vraie utilité d'un tel service est réelle dès lors que celui-ci est connecté à un système de domotique. Le lapin ainsi devient **l'âme de la maison** et communique sur les événements détectés par la domotique. C'est la raison pour laquelle beaucoup de box de domotique proposent des connecteurs vers le Karotz.

Nous sommes dans une période de transition, nous aurons d'ici une dizaine d'années de nombreuses offres de mini-robots connectés et capables de communiquer avec votre maison. Mais aujourd'hui, le marché n'est pas encore au rendez-vous. Le Karotz (/tag/Karotz) ne fait fantasmer que quelques geeks passionnés comme moi de nouvelles technologies et nouveaux usages.

**Adieu Karotz !**



3 commentaires (<http://www.geek.org/karotz-mort-019.html#comments>)

## DomoGeek : Domotique + NodeJS + Raspberry Pi

(<http://www.geek.org/domogeek-domotique-nodejs-raspberry-pi-011.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: mardi 7 octobre 2014, 23:29 :: Domotique (<http://www.geek.org/category/technologie/domotique>)



Pour les fans comme moi de bricolage en tout genre, je suis heureux de vous partager mes expérimentations geek (/tag/geek) autour de la domotique (/tag/domotique).

J'ai initialisé il y a quelques mois un projet sur GitHub (/tag/GitHub) que j'alimente quand j'ai un peu de temps de libre. Il s'agit d'un socle modulaire de domotique développé en **NodeJS + MongoDB** et fonctionnant sur un **Raspberry Pi** équipé d'un dongle **Z-wave**.



Le code source du projet est disponible sur mon espace GitHub (<https://github.com/ltoinel/domogeeek>). Le projet se découpe en **4 modules simples** :

- Un **module de communication à plusieurs par SMS** pour permettre de créer une communauté de voisins via SMS et communiquer très rapidement avec eux en cas d'urgence (vol, détériorations ...).
- Un **module de planification d'événement** permettant d'ouvrir les volets (/comment-domotiser-ses-volets-radio-pour-moins-de-50-960.html), faire parler le lapin Karotz (/openkarotz-liberez-votre-lapin-du-cloud-974.html) ....
- Un **module de communication** permettant de transmettre un message par Mail, SMS, voix sur le Karotz.
- Un **module de détection d'événement Z-wave** capable d'écouter des événements et de transmettre l'événement au module qui a en charge de le traiter : détection de présence, de fumée ...



Pour le moment, il s'agit d'un projet totalement expérimental. L'idée est de proposer des modules facilement ré-adaptables pour d'autres besoins.

Si vous souhaitez vous lancer vous aussi dans une box domotique custom, mon projet ZwaveBus peut être un bon départ pour communiquer avec vos capteurs Z-wave. Il se base lui-même sur la librairie OpenZwave et un wrapper NodeJS.



### Prochaines étapes du projet :

- Automatiser la captation de présence dans la maison au travers de l'API de la Freebox (/domotique-freebox-api-996.html) permettant d'identifier les terminaux mobiles connectés au sein de la maison.
- Développement du listener pour la détection de fumée.
- Industrialiser le lancement des daemon NodeJS et la récupération des dépendances.
- Développer une interface mobile pour commander les services.

Pour découvrir le projet plus en détail : <https://github.com/ltoinel/domogeek> (<https://github.com/ltoinel/domogeek>)



2 commentaires (<http://www.geek.org/domogeek-domotique-nodejs-raspberry-pi-011.html#comments>)

## Domotique : Freebox API et détection d'absence (<http://www.geek.org/domotique-freebox-api-996.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: dimanche 14 septembre 2014, 22:10 :: Domotique  
(<http://www.geek.org/category/technologie/domotique>)



Je travaille actuellement sur un projet personnel de box domotique (/demenagement-dans-ma-maison-de-geek-931.html) OpenSource (/tag/OpenSource) basée sur du NodeJS (/tag/NodeJS) / MongoDB (/tag/MongoDB) / OpenZwave (/tag/OpenZwave), le tout hébergé sur un Raspberry Pi (/tag/Raspberry%20Pi). Dans le cadre du développement de cette box domotique, je me suis demandé comment je pouvais activer **automatiquement** l'alarme et les automatisations quand je pars de ma maison. Je pense avoir trouvé la solution grâce à une API cachée de ma **Freebox V6** ....

Vous avez probablement déjà pu le constater en vous connectant sur l'interface Web d'administration de votre **Freebox V6** (<http://freebox>), votre **Freebox** sait tout de ce qu'il se passe sur votre réseau local.

Vous pouvez depuis cette interface y observer vos PC, mais aussi très probablement votre Smartphone (/tag/Smartphone) si la configuration Wifi (/tag/Wifi) est active. Basé sur ce constat, si chaque habitant possède un smartphone connecté au Wifi (/tag/Wifi), il est donc possible de savoir au travers de la Freebox (/tag/Freebox) si quelqu'un est à votre domicile ou non.

En approfondissant un peu plus la documentation des API exposées par la Freebox (<http://dev.freebox.fr/sdk/os/lan/>), on remarquera qu'il est possible pour une application tierce de lister l'ensemble des terminaux connectés sur son réseau local via une **API REST** qui restitue les mêmes informations que l'interface Web d'administration de la Freebox. **Miracle !**

```
GET /api/v3/lan/browser/{interface}/
```

```
{
  "success": true,
  "result": [
    {
      "l2ident": {
        "id": "d0:23:db:36:15:aa",
        "type": "mac_address"
      },
      "active": true,
      "id": "ether-d0:23:db:36:15:aa",
      "last_time_reachable": 1360669498,
      "persistent": true,
      "names": [
        {
          "name": "iPhone-r0ro",
          "source": "dhcp"
        }
      ],
      "vendor_name": "Apple, Inc.",
      "l3connectivities": [
        {
          "addr": "192.168.69.20",
          "active": true,
          "af": "ipv4",
          "reachable": true,
          "last_activity": 1360669498,
          "last_time_reachable": 1360669498
        }
      ]
    }
  ]
}
```

```
    ],  
    "reachable": true,  
    "last_activity": 1360669498,  
    "primary_name_manual": true,  
    "primary_name": "iPhone r0ro"  
  }  
]  
}
```

Vous l'aurez compris, il ne reste plus qu'à appeler cette API (/tag/API) dès lors qu'aucun mouvement est détecté dans la maison. Si jamais plus aucun Smartphone (/tag/Smartphone) n'est détecté et qu'aucune présence n'est détectée par les détecteurs infrarouges, on peut considérer à 98% que la maison est vide et donc déclencher l'alarme et l'ensemble des automatisations planifiées.

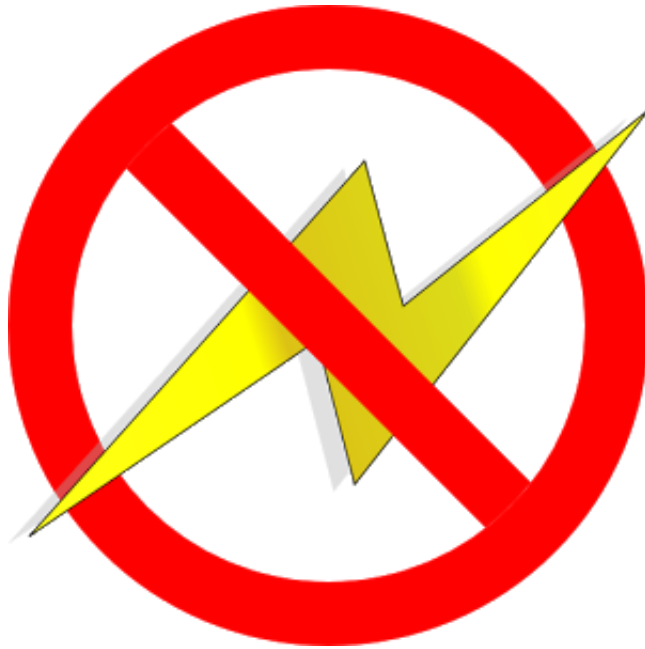
Grâce à cette API, ma maison va donc devenir intelligente. Je n'aurai plus besoin d'informer ma box domotique que je ne suis plus là. Merci Free !



11 commentaires (<http://www.geeek.org/domotique-freebox-api-996.html#comments>)

**Domotique : Recevez gratuitement un SMS en cas de coupure de courant !** (<http://www.geeek.org/domotique-recevez-gratuitement-un-sms-en-cas-de-coupure-de-courant-976.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: jeudi 28 août 2014, 22:51 :: Domotique  
(<http://www.geeek.org/category/technologie/domotique>)



Vous souhaitez être prévenu rapidement par SMS en cas de **coupure de courant** à votre domicile à votre lieu de travail ? Voici une solution très simple et peu onéreuse que vous pouvez mettre en place.

Cette solution consiste à utiliser un vieux téléphone Android (/tag/Android) équipé d'une SIM Free Mobile à 0€ ou 2€.

Une application Android "Power Outage Alarm" (<https://play.google.com/store/apps/details?id=uk.co.sjbcomputerconsultants.powercut.pro&hl=fr>) vous permet d'être alerté par SMS (/tag/SMS) à la moindre coupure de courant. Il suffit d'installer l'application et de configurer le numéro de téléphone contacter. Au moindre arrêt de l'alimentation du téléphone, un SMS sera transmis automatiquement pour vous prévenir de la coupure de courant, cela peut-être pratique si votre système d'alarme ne fonctionne pas sur un onduleur ou si vous avez des systèmes critiques à superviser (chambre frigorifique, baie de serveur ...).

Pour les geeks, il est possible de développer très simplement une application similaire. L'événement transmis par le système Android est **ACTION\_POWER\_CONNECTED** lorsque que le téléphone est alimenté et **ACTION\_POWER\_DISCONNECTED** lorsque le téléphone n'est plus alimenté.

Il vous suffira donc de remplir le *Manifest* avec les bons *Intent* à écouter:

```
<receiver android:name=".PowerConnectionReceiver">
  <intent-filter>
    <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
    <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
  </intent-filter>
</receiver>
```

Vous pourrez ainsi envoyer un SMS dès lors que l'un de ces deux événements se produisent :

```
public class PowerConnectionReceiver extends BroadcastReceiver {
    @Override public void onReceive(Context (http://www.google.com/search?
```

```
hl=en&q=allinurl%3Acontext+java.sun.com&btnI=I%27m%20Feeling%20Lucky) context, Intent intent) {  
    if(intent.getAction() == Intent.ACTION_POWER_CONNECTED) {  
        // Envoi de SMS pour informer du retour de courant.  
    } else if(intent.getAction() == Intent.ACTION_POWER_DISCONNECTED){  
        // Envoi de SMS pour informer de L'arrêt de courant.  
    }  
}
```

Par contre, je n'ai aucune idée du coût que cela peut représenter de laisser brancher un téléphone et la durée de vie de la batterie si celle-ci est mise en charge en permanence ...



9 commentaires (<http://www.geek.org/domotique-recevez-gratuitement-un-sms-en-cas-de-coupure-de-courant-976.html#comments>)

## OpenKarotz : Libérez votre lapin du cloud !

(<http://www.geek.org/openkarotz-liberez-votre-lapin-du-cloud-974.html>)

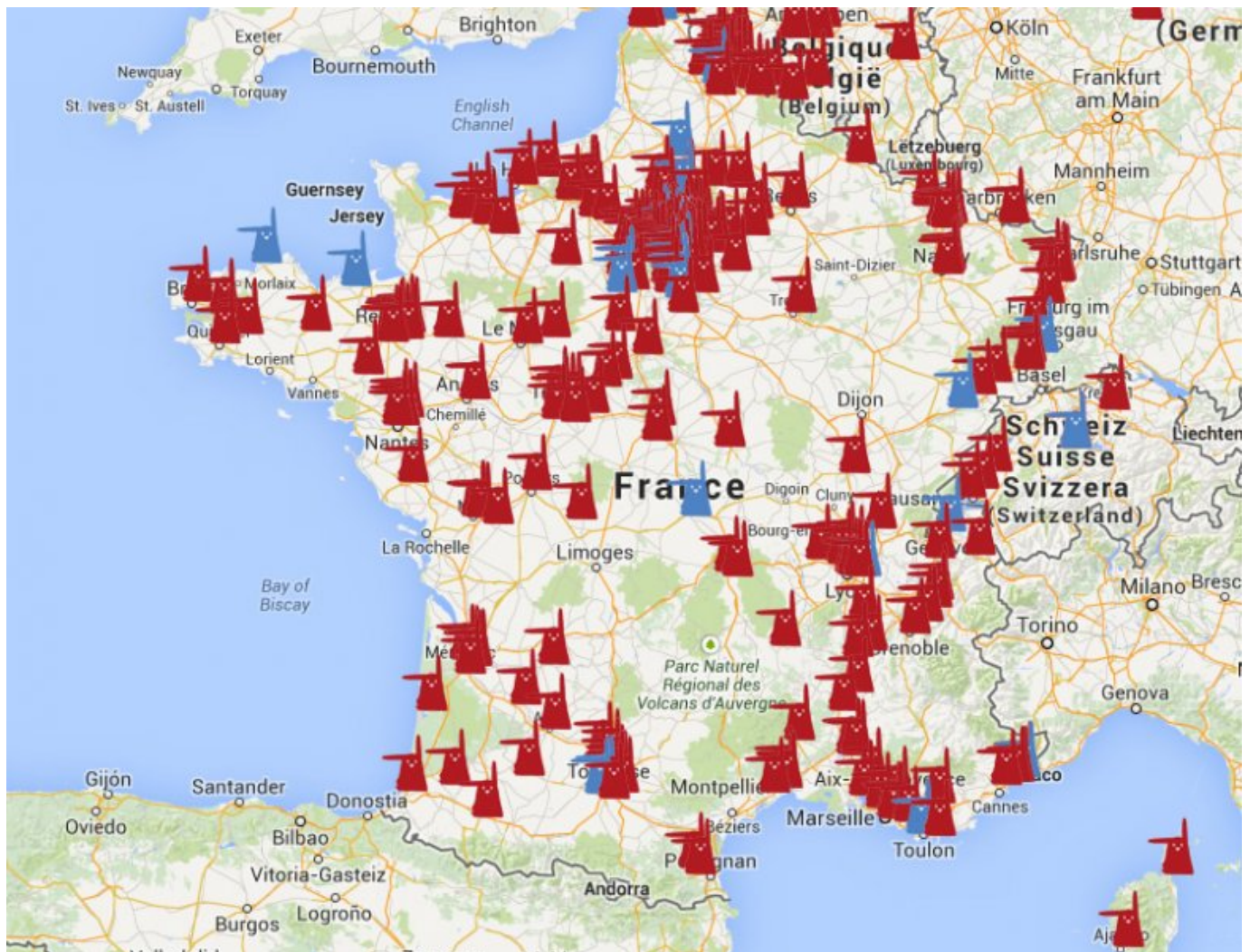
Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: mardi 12 août 2014, 01:15 :: Domotique (<http://www.geek.org/category/technologie/domotique>)



Et si vous libériez définitivement votre Karotz (/tag/Karotz) du Cloud ?

C'est désormais possible avec le firmware d'OpenKarotz (<http://www.openkarotz.org>) qui permet de faire fonctionner votre lapin en autonomie totale sans adhérence le cloud.

La dernière version du firmware a été déployée sur plus de **400 lapins** maintenant. Une petite communauté d'utilisateurs s'est formée autour de ce firmware ouvert permettant d'interagir avec son lapin directement sans passer par Internet via des serveurs sur le cloud.



(<http://www.openkarotz.org/openkarotz-location/>)

De nombreuses box de domotique comme **Eedomus** proposent même d'interagir directement avec les Karotz (/tag/Karotz) équipés d'**OpenKarotz**.


**OpenKarotz** fournit une interface **Web** et **mobile** permettant d'interagir avec le lapin directement sans connaissance technique. Pour les plus geeks d'entre vous, le firmware expose de nombreuses **API REST** utilisables directement ou via des librairies vous simplifiant la tâche (<https://github.com/guillaumewuip/openkarotz-nodejs>).



## OPEN KAROTZ


[Informations](#) [State](#) [Leds](#) [Ears](#) [RFID](#) [TTS](#) [Picture](#) [Sounds](#) [Apps](#) [Update](#)





About



Web Version : 210  
RootFs : 2.0.r8  
Patch Level : 219

You can find informations about the **API usage here**.

You can also use this icon to see API sample on each page .

**Contact** : [openkarotz@filippi.org](mailto:openkarotz@filippi.org)  
**Web Site** : <http://www.openkarotz.org>   
**Release Notes** : <http://openkarotz.filippi.org/release-notes>   
**Support Forum (FR)** :   
**Support OpenKarotz** 


► Thanks to ...

L'installation du firmware se fait en quelques clics, il faut ensuite se rendre sur l'interface Web pour appliquer les dernières mises à jour.

Attention, une fois ce firmware installé, notre lapin sera n'aura plus accès aux applications proposées sur le site de Karotz (/tag/Karotz).

<http://www.openkarotz.org> (<http://www.openkarotz.org>)



 aucun commentaire (<http://www.geek.org/openkarotz-liberez-votre-lapin-du-cloud-974.html#comments>)

## Karotz : Manipulez votre lapin à distance avec NodeJS !

(<http://www.geek.org/karotz-manipulez-votre-lapin-a-distance-avec-nodejs-973.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: dimanche 10 août 2014, 00:56 :: Domotique  
(<http://www.geek.org/category/technologie/domotique>)



Le Karotz (/tag/Karotz) n'est pas encore mort. Même si le portail pour les développeurs a été retiré d'Internet, les API du Karotz (/tag/Karotz) sont toujours fonctionnelles sur le cloud.

Il vous est même possible de manipuler très facilement votre lapin à distance au travers d'un module NodeJS (/tag/NodeJS) disponible sur GitHub (/tag/GitHub) :

<https://github.com/guillaumewuip/Karotz-NodeJS-Plugin> (<https://github.com/guillaumewuip/Karotz-NodeJS-Plugin>)

```
var karotz = require('./karotz');

var installid = '12345', // Récupéré sur la page de paramétrage d'une app Karotz
    apikey     = '12345', // Récupéré sur la page de déclaration d'une Karotz
    secret     = '12345'; // Récupéré sur la page de déclaration d'une Karotz

karotz.authentication(apikey, installid, secret, true, function(app){

karotz.tts('speak', 'EN', "I want a carrot !", function(msg) {
    console.log(msg); //Output 'Speaking' or 'Error'
});
```



```
});
```

L'utilisation de ce plugin nécessite auparavant de créer une application sur la zone lab portail du Karotz (<http://www.karotz.com/>). Pour vous simplifier la tâche voici un exemple d'App Karotz que vous pouvez utiliser et que je viens de partager sur GitHub :

<https://github.com/ltoinel/domogeeek/tree/master/misc/karotz-app>  
(<https://github.com/ltoinel/domogeeek/tree/master/misc/karotz-app>)

Si jamais vous êtes passé au firmware OpenKarotz (<http://www.openkarotz.org/>), une librairie équivalente est aussi disponible sur GitHub (/tag/GitHub), elle est développée par le même auteur :

<https://github.com/guillaumewuip/openkarotz-nodejs> (<https://github.com/guillaumewuip/openkarotz-nodejs>)

Elle a l'énorme avantage de ne nécessiter aucune authentification quelconque pour pouvoir fonctionner, il suffit juste de connaître l'adresse IP de son lapin sur son réseau local.

```
var openkarotz = require('openkarotz');
```

```
var karotz = new openkarotz('192.168.0.1');
```

```
karotz.tts( "I want a carrot !", 'EN', true, function(msg) {  
    console.log(msg);  
});
```

Ces modules NodeJS permettent de jouer avec les LED, les oreilles, le lecteur RFID, la voix du lapin, la prise de photo et la lecture de fichiers multimédia ...

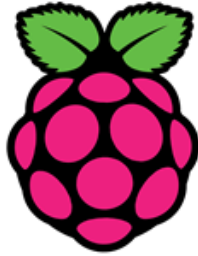
Ils fournissent tous les éléments vous amuser rapidement si vous avez des projets de domotique en tête.



■ aucun commentaire (<http://www.geeek.org/karotz-manipulez-votre-lapin-a-distance-avec-nodejs-973.html#comments>)

## OpenZwave + NodeJS + Raspberry Pi : Les ingrédients pour développer sa box domotique ? (<http://www.geeek.org/openzwave-nodejs-raspberry-pi-les-ingredients-pour-creer-sa-box-domotique-972.html>)

Par Ludovic (<https://plus.google.com/+LudovicToinel>) :: dimanche 27 juillet 2014, 22:16 :: Domotique (<http://www.geeek.org/category/technologie/domotique>)



## RaspberryPi

Vous êtes un geek bricoleur à la recherche d'une solution alternative aux solutions commerciales pour construire votre box domotique ?

Voici une alternative intéressante qui consiste à utiliser un Wrapper NodeJS (/tag/NodeJS) pour OpenZave (/tag/OpenZave), une librairie OpenSource (/tag/OpenSource) permettant de contrôler des périphériques Z-wave.

Le code source de la librairie est disponible sur GitHub (/tag/GitHub) :

<https://github.com/jperkin/node-ope...> (<https://github.com/jperkin/node-openzwave>)

Il existe aussi un "fork" intéressant utilisant la dernière version de la librairie OpenZave (/tag/OpenZave) :

<https://github.com/FrozenCow/node-o...> (<https://github.com/FrozenCow/node-openzwave>)

Pour développer votre propre box de domotique, il vous suffira ensuite de vous inspirer du programme d'exemple pour comprendre les événements mis à la disposition des développeurs. C'est simple, efficace ... Je n'ai pas encore testé toutes les fonctionnalités du Wrapper, mais le script d'exemple reconnaît correctement mon matériel.

Si vous prévoyez d'utiliser un Raspberry (/tag/Raspberry), la librairie fonctionne correctement sur un Raspberry Pi.

Cette librairie NodeJS offre une solution simple à prendre en main pour développer très rapidement des interactions avec votre matériel Z-wave.



3 commentaires (<http://www.geek.org/openzwave-nodejs-raspberry-pi-les-ingredients-pour-creeer-sa-box-domotique-972.html#comments>)

◀ billets précédents (<http://www.geek.org/page/2?q=domotique>)