REPRÉSENTEZ GRAPHIQUEMENT VOS DONNÉES COLLECTÉES EN MQTT

Hackable n° 026 (/Hackable/HK-026) | septembre 2018 | Denis Bodor (/auteur/view/8925-bodor_denis)

Voyez ceci comme un article bonus car, naturellement, lorsqu'on collecte des données de capteurs ou sondes et qu'on utilise judicieusement MQTT pour cela, on s'ouvre à des options très intéressantes. MQTT étant un protocole reconnu et largement utilisé, il devient très facile d'utiliser des outils presque clés en main pour stocker et représenter les informations publiées sur les topics que vous aurez créés.

Tout va pour le mieux, vous avez structuré vos topics, l'ensemble du système fonctionne et communique joyeusement en MQTT, tout est bien chiffré et sécurisé et vos sondes publient leurs mesures régulièrement… Et maintenant quoi ?

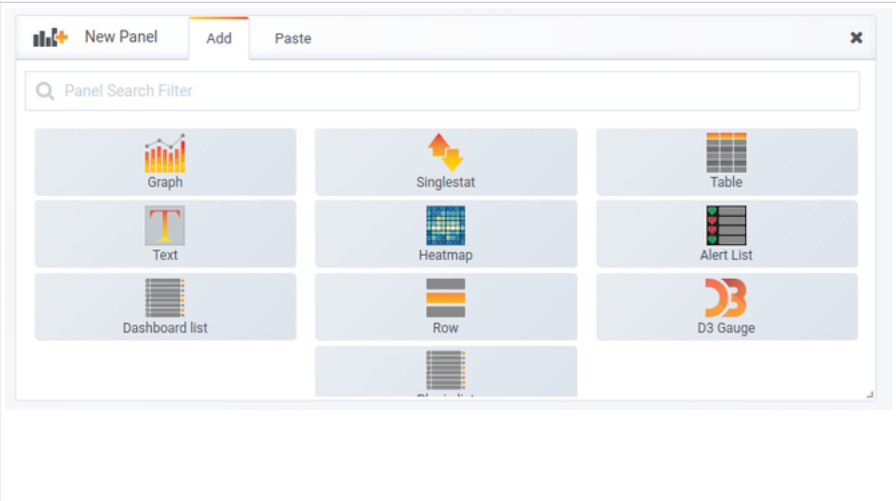
Les messages envoyés par vos capteurs, qu'ils véhiculent des informations sur la température, l'hygrométrie, la détection de mouvements, la luminosité, la pression, etc., sont émis dans le vide si rien n'est là pour les recevoir et les enregistrer. Certes, il peut être intéressant d'avoir une vision en temps réel à l'aide de **mosquitto_sub** ou même d'un bout de script en Python, mais si les informations sont là, autant les conserver. Et si elles sont conservées, autant en faire de jolis graphiques avec, en prime la possibilité de créer des alertes selon les conditions de votre choix.

Pour arriver à ce résultat, nous aurons besoin de trois éléments :

- Une base de données orientée « séries temporelles » (ou TSDB pour *Time Series DataBase*) spécialement conçue et optimisée pour stocker des données sur une base temporelle. Il en existe plusieurs installables sur Raspbian, mais nous choisirons ici la plus populaire : InfluxDB.
- Un outil pour remplir cette base avec les données provenant de nos capteurs ESP8266 communiquant via MQTT. Ceci pourrait tout aussi bien être un programme écrit pour l'occasion (en Python, en Perl ou même en Shell), mais les développeurs d'InfluxDB proposent un utilitaire complet et modulaire permettant de collecter diverses données, les traiter, les agréger et les enregistrer : Telegraf.
- Un outil de visualisation des données piochant les informations dans une base, comme InfluxDB, et permettant leur affichage sous la forme de graphiques divers directement dans un navigateur web : Grafana.

1. INSTALLATION DE GRAFANA

Grafana existe dans Raspbian 9 Stretch, le système Debian GNU/Linux le plus récent pour Raspberry Pi. Malheureusement, la version proposée est très en retard par rapport à celle actuellement disponible directement sur le site officiel du projet. Vous pourriez vous en contenter, mais personnellement, étant donné le nombre de bugs qui ont été corrigés entre l'une et l'autre version, la question ne se pose pas.



SOMMAIRE

PAR LE MÊME AUTEUR

LAUCHPAD MSP432 : EN ROUTE POUR LE MULTITÂCHE ! (/HACKABLE/HK-008/LAUCHPA MSP432-EN-ROUTE-POUR-LE-MULTITACHE)

Hackable n° 008 (/Hackable/HK-008) | septembre 2015 | Deni (/auteur/view/8925-bodor_denis)
Électronique (/content/search/?filter[]=attr_category_lk:"électronique"&activeFacets[attr_categ
Tests et prise en main (/content/search/?filter[]=attr_category_lk:"tests et prise en main"&activeFacets[attr_category_lk:Domaines]=tests et prise main)

LE MONDE DES MICROCONTRÔLEURS AVR 8 BI (/GNU-LINUX-MAGAZINE/GLMFHS-043/LE-MO DES-MICROCONTROLEURS-AVR-8-BITS)

GNU/Linux Magazine HS n° 043 (/GNU-Linux-Magazine/GLMFHS-043) | août 2009 | De Bodor (/auteur/view/8925-bodor_denis)

CRÉEZ UNE SONDE DE TEMPÉRATURE QUI VOU ALERTE PAR SMS (/HACKABLE/HK-023/CREEZ UNE-SONDE-DE-TEMPERATURE-QUI-VOUS-ALE PAR-SMS)

Hackable n° 023 (/Hackable/HK-023) | mars 2018 | Denis Bod (/auteur/view/8925-bodor_denis)

QUICKTIP : UN MODULE QUI GÈRE /PROC (/GN LINUX-MAGAZINE/GLMF-127/QUICKTIP-UN-MODULE-QUI-GERE-PROC)

GNU/Linux Magazine n° 127 (/GNU-Linux-Magazine/GLMF-12 mai 2010 | Denis Bodor (/auteur/view/8925-bodor_denis)

MANIFESTATION : RETOUR SUR LA MAKERFAIR PARIS 2014 (/HACKABLE/HK-002/MANIFESTA RETOUR-SUR-LA-MAKERFAIRE-PARIS-2014)

Hackable n° 002 (/Hackable/HK-002) | septembre 2014 | Deni (/auteur/view/8925-bodor_denis)
Électronique (/content/search/?filter[]=attr_category_lk:"électronique"&activeFacets[attr_categ
Embarqué (/content/search/?filter[]=attr_category_lk:"embarqué"&activeFacets[attr_categor

Grafana est livré de base avec un certain nombre de plugins permettant différentes présentations des données. Les plus classiques cependant sont « Graph » pour tracer des courbes et « Singlestat » pour des valeurs uniques éventuellement sous forme de jauges.

Pour installer une version actuelle de Grafana, le plus simple est encore de pointer votre navigateur sur **https://grafana.com/grafana/download (https://grafana.com/grafana/download)**, cliquer sur ***ARM*** et repérer la bonne version téléchargeable pour votre Pi ***Ubuntu & Debian(ARMv7)*** (il n'existe pas de version officielle téléchargeable pour ARMv6 et donc Raspberry Pi 1). Il suffit alors de copier-coller les deux commandes proposées :

```
$ wget https://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana_5.2.1_armhf.deb
$ sudo dpkg -i grafana_5.2.1_armhf.deb
```

Note : Des paquets non officiels existent pour Raspberry Pi 1 et, si vraiment vous souhaitez utiliser cette plateforme malgré la lenteur, vous trouverez votre bonheur sur **https://dl.bintray.com/fg2it/deb-rpi-1b/main/g/**. Remarquez que la dernière version disponible ici est la 5.1.4, c'est toujours mieux que la 2.6.0 proposée par Raspbian, mais en dessous de la 5.2.1 officielle.

Une fois le paquet installé, vous pourrez supprimer le fichier **grafana_5.2.1_armhf.deb** qui n'aura plus aucune utilité. Notez le message qui s'affichera au terme de l'installation du paquet :

```
### NOT starting on installation, please execute the
following statements to configure grafana to start
automatically using systemd
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable grafana-server
### You can start grafana-server by executing
sudo /bin/systemctl start grafana-server
```

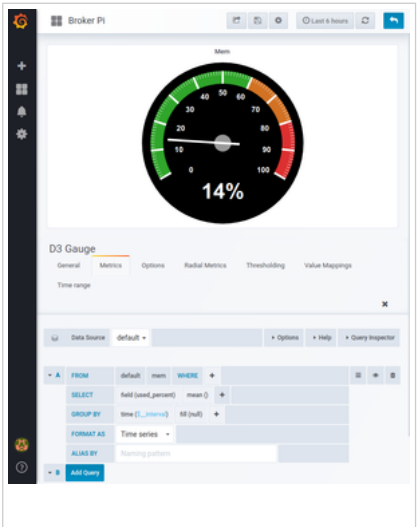
En effet, le service Grafana n'est ni démarré automatiquement, ni configuré pour être actif au démarrage du système. Il est donc nécessaire d'exécuter ces trois commandes permettant respectivement de recharger *systemd* pour qu'il prenne en compte le nouveau service, active le démarrage automatique de Grafana au boot de la Pi et démarre immédiatement Grafana. Ainsi, vous n'aurez plus rien à faire lors du prochain démarrage.

Dès cet instant, vous devez être en mesure de pointer votre navigateur sur l'adresse IP de la Pi en spécifiant le port 3000 avec quelque chose comme **http://192.168.0.42:3000**. Vous pouvez également utiliser mDNS en spécifiant le nom d'hôte de la Pi et en ajoutant « .local » ainsi : **http://raspberrypi.local:3000**. Notez que, sous Windows, mDNS n'est pas utilisable par défaut et vous devrez installer, par exemple, les « *Services d'impression Bonjour pour Windows* » d'Apple gratuitement téléchargeables (**https://support.apple.com/kb/DL999 (https://support.apple.com/kb/DL999)**) pour ajouter cette fonctionnalité.

À la première connexion à l'interface web, vous serez invité à entrer un nom d'utilisateur (« admin ») et un mot de passe (« admin ») puis à changer ce dernier pour des raisons évidentes de sécurité. Notez que toute l'interface de Grafana est en anglais et qu'il n'existe à l'heure actuelle pas de mécanisme simple et intégré de traduction permettant à des contributeurs d'apporter leur aide. En attendant une solution technique introduite dans une prochaine version du projet, il vous est toujours possible d'utiliser la fonction de traduction intégrée à votre navigateur qui, pour ce type d'interface, fonctionne relativement bien.

2. INSTALLATION D'INFLUXDB

Pour que Grafana puisse faire son travail de représentation des mesures, il lui faut une source de donnée. Plusieurs sont supportées par l'outil : Graphite, MySQL, PostgreSQL, OpenTSDB, ElasticSearch ou encore MS/SQL Serveur, mais surtout InfluxDB. Cette dernière est sans doute la plus populaire et j'avoue ne pas avoir tenté d'en utiliser d'autres tant cela semble évident. Notez au passage que des choses comme MySQL/MariaDB ou PostgreSQL sont des systèmes de gestion de bases de données (SGBD) qu'on peut qualifier de « classiques », car non orientés « séries temporelles ». À mon sens, il sera intéressant d'en faire usage uniquement si vous les mettez déjà en œuvre pour d'autres applications (généralement liées au Web). Dans le cas contraire, InfluxDB ou Graphite sont bien plus intéressants.



Chaque panel est configuré pour afficher les données contenues dans une base, qu'on extraira à l'aide d'une requête composée dans l'interface. Il n'est pas nécessaire de connaître excessivement le langage utilisé pour des requêtes simples puisque les différents choix possibles sont proposés automatiquement.

Tout comme pour Grafana, une version d'InfluxDB est disponible dans Raspbian 9.4 mais, là encore, il s'agit d'une version relativement ancienne. Mieux vaudra donc s'orienter vers une autre source et en particulier le site d'InfluxData, développeur d'InfluxDB. Ici, un dépôt peut être directement utilisé, mais devra être configuré au préalable. Pour ce faire, créez simplement un fichier portant une extension **.list** dans le répertoire **/etc/apt/sources.list.d** (**influx.list** par exemple) contenant une unique ligne :

```
deb https://repos.influxdata.com/debian stretch stable
```

Notez la présence du nom de code de la distribution Raspbian utilisé dans la ligne. Si vous n'êtes pas sûr de la version que vous utilisez, vous pouvez faire usage de la commande **lsb_release -a** :

```
No LSB modules are available.
Distributor ID: Raspbian
Description: Raspbian GNU/Linux 9.4 (stretch)
Release: 9.4
Codename: stretch
```

Avant de mettre à jour la liste des paquets de votre distribution, il sera nécessaire d'ajouter une clé permettant de valider la signature sur ces derniers et ainsi éviter d'avoir un désagréable message d'avertissement : **curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -**. Cette commande devrait simplement vous retourner **OK** et vous pourrez alors vous plier d'un **sudo apt-get update**. En cas de problème, il est possible que votre installation n'inclue pas le nécessaire pour gérer les dépôts HTTPS et vous devrez vous assurer que le paquet **apt-transport-https** soit effectivement installé.

L'installation d'InfluxDB pourra alors se faire avec la commande **sudo apt-get install influxdb**. Comme avec Grafana, le service InfluxDB n'est pas automatiquement démarré après l'installation du paquet, mais il est cependant activé au démarrage du système comme le montrera la commande **sudo systemctl is-enabled influxdb**. Vous devrez donc démarrer le service avec la commande **sudo systemctl start influxdb**, si vous ne voulez pas redémarrer votre Pi.

Avant d'attaquer la représentation des données reçues en MQTT, il est important de se familiariser avec l'environnement et l'interface. Ça tombe bien, Telegraf collecte déjà automatiquement des données sur l'état de la Raspberry Pi sur laquelle il fonctionne…

3. INSTALLATION DE TELEGRAF

En configurant le dépôt pour obtenir une version récente d'InfluxDB, nous obtenons également la possibilité d'installer le troisième élément de notre ensemble. Pour installer Telegraf, le collecteur de données destinées à être placées dans une base InfluxDB, il nous suffit d'invoquer la commande :

```
$ sudo apt-get install telegraf
```

Dès l'installation, Telegraf se met en marche. « Pour collecter quoi ? » me demanderez-vous. Tout simplement ce pour quoi il est conçu à la base : les informations sur l'état de santé du système (processeur, disques, mémoire, processus, etc.). En effet, Telegraf est initialement prévu et configuré pour permettre la surveillance de systèmes et de serveurs en particulier. Il est cependant extrêmement modulaire et capable de relever tous types d'informations (des *metrics*) et ce par bien des moyens. Il suffit de jeter un œil à la page GitHub du projet pour se rendre compte que ce n'est pas moins de quelques 130 sources (ou *Input Plugins*) qui peuvent être utilisées. Et bien sûr, MQTT est de la partie avec un plugin **mqtt_consumer**.

4. TESTER LE TOUT

Une fois les trois éléments installés, tout est déjà en route et nous pouvons donc vérifier immédiatement le bon fonctionnement de l'ensemble. Pour cela, rendez-vous sur l'interface de Grafana à l'aide de votre navigateur. Sur la droite, vous trouverez une série d'icônes donnant accès à différentes sections du site. Dans la configuration (roue dentée), vous trouverez un onglet ***Data Source*** vous permettant de définir une ou plusieurs sources de données où Grafana ira piocher ses informations pour créer des graphiques.

En cliquant sur ***Add data source***, vous serez invité à renseigner plusieurs éléments :

- ***Name*** : un nom arbitraire pour votre source ;
- ***Type*** : le type de source, ici influxDB ;
- ***URL*** : est l'adresse pour accéder à influxDB, ici **http://localhost:8086** ;
- ***Database*** : le nom de la base influxDB à utiliser, ici « telegraf » puisque celle-ci a automatiquement été créée.

Cliquez ensuite sur ***Save & Test*** et, après un bref instant, la page sera rechargée avec en haut à droite un petit message vert indiquant ***Datasource added***, preuve que tout a bien fonctionné. Remarquez que nous n'avons pas touché aux autres paramètres concernant principalement l'authentification, qui n'est pas configurée par défaut avec InfluxDB. Ce n'est pas un problème puisque justement, par défaut également, InfluxDB n'est pas en attente de connexions extérieures, mais uniquement initiées depuis la machine elle-même, l'hôte local (localhost ou 127.0.0.1).

Une fois cette configuration effectuée, Grafana a accès aux données stockées par Telegraf et vous pouvez créer votre premier *dashboard* (tableau de bord). Un dashboard est un tableau sur lequel vous pouvez placer des éléments graphiques appelés *panels* (vous comprenez pourquoi j'utilise ici les désignations anglaises, utilisées par Grafana, et non l'équivalent français, « tableau » et « panneau » pouvant dire plus ou moins la même chose).

Pour créer votre dashboard, cliquez simplement dans les outils à gauche, sur l'icône ***+*** et choisissez ***Dashboard***. Vous serez alors dirigé vers une page vous permettant d'ajouter un panel. Cliquez sur le premier, ***Graph***, et celui-ci est alors ajouté automatiquement. Son emplacement et sa taille peuvent être modifiés à l'aide de la souris et, en cliquant sur son titre en haut, un menu vous permettra l'édition via une entrée ***Edit***. Le graphique occupera alors toute la largeur de l'écran et une série d'onglets situés en dessous vous permettront la configuration de l'élément graphique.

Vous arriverez automatiquement sur l'onglet ***Metrics*** où vous pourrez choisir la donnée à utiliser pour tracer le graphique. Là, choisissez tout d'abord votre source de données précédemment configurée puis intéressez-vous à la partie inférieure. Il s'agit de la requête qui piochera les données à afficher. La syntaxe est similaire à celle du langage SQL et est composée comme une phrase : « *DEPUIS tel_endroit AVEC cette_condition CHOISIR quelque_chose* ».

Ici, à des fins de démonstration, à la ligne ***FROM***, cliquez sur ***Select measurement*** et choisissez ***Mem***. Cliquez ensuite sur le ***+*** de la même ligne pour ajouter une condition, choisissez ***Host*** et, sur la droite, le seul nom d'hôte proposé. À la ligne ***SELECT***, cliquez sur ***Value*** et choisissez, par exemple, ***Used percent***. Immédiatement, le graphique doit présenter des données visuellement.

Ce que nous venons de faire est de demander à Grafana de regarder dans la base des informations enregistrées par Telegraf, afin de prendre les valeurs concernant la mémoire consommée par le système (en pourcentage), pour l'hôte de votre choix (la Pi elle-même puisque c'est tout ce que nous stockons). Ces valeurs, régulièrement enregistrées par Telegraf, sont alors reportées sur le graphique dans l'axe des ordonnées, avec le temps en abscisse.

Une fois satisfait de votre configuration et après avoir fait un petit tour sur l'onglet ***General*** pour donner un titre (***Title***) judicieux à votre graphique, cliquez simplement sur la petite croix à droite, et le dashboard reviendra à son état initial. Profitez-en pour cliquer sur l'icône de sauvegarde tout en haut à droite et donner un nom à votre dashboard.

Je vous recommande de jouer un peu avec Grafana et les données enregistrées par Telegraf à ce stade, pour vous accoutumer à l'interface qui parfois peut être assez déroutante et surtout dense. Ajoutez d'autres panels, d'autres dashboards, essayez différents panels et faites un tour sur le site officiel pour y trouver des plugins supplémentaires. Personnellement, concernant la relève de températures, et l'affichage en temps réel, j'aime tout particulièrement *D3 Gauge* de Brian Gann. Pour installer un plugin, visitez sa page et vous trouverez un onglet ***Installation*** précisant la ligne de commandes à utiliser avec **grafana-cli**.

Dans le cas de *D3 Gauge*, ceci sera :

```
$ sudo grafana-cli plugins install briangann-gauge-panel
installing briangann-gauge-panel @ 0.0.6
from url: https://grafana.com/api/plugins/briangann-gauge-panel/versions/0.0.6/download
into: /var/lib/grafana/plugins
Installed briangann-gauge-panel successfully
Restart grafana after installing plugins . <service grafana-server restart>

$ sudo systemctl restart grafana-server
```

Un autre panel intéressant offrant un affichage sous forme de jauge est « Singlestat » qui montre, par défaut, une simple valeur. Mais en cochant ***Gauge***/***Show*** dans son onglet ***Options*** et en réglant des ***Thresholds*** (seuils), il est possible d'obtenir quelque chose de très agréable à l'œil, et ce, sans installer un plugin supplémentaire.

5. AJOUTER NOS MESURES VIA MQTT

À ce stade, nous avons une installation de Grafana, InfluxDB et Telegraf déjà fonctionnelle et il est temps de nous pencher sur la représentation des données transmises par nos ESP8266 en MQTT. Pour ce faire, celles-ci doivent simplement être stockées dans la base InfluxDB par Telegraf, rien de plus. En jetant un coup d'œil à la configuration

de ce dernier, via le fichier **/etc/telegraf/telegraf.conf**, on se rend compte que même si le fichier fait presque 4000 lignes, il est très majoritairement composé de commentaires. En éliminant tout ceci, on arrive en réalité à quelque chose de très succinct :

```
[global_tags]
[agent]
interval = "10s"
round_interval = true
metric_batch_size = 1000
metric_buffer_limit = 10000
collection_jitter = "0s"
flush_interval = "10s"
flush_jitter = "0s"
precision = ""
debug = false
quiet = false
logfile = ""
hostname = ""
omit_hostname = false

[[outputs.influxdb]]
[[inputs.cpu]]
percpu = true
totalcpu = true
collect_cpu_time = false
report_active = false
[[inputs.disk]]
ignore_fs = ["tmpfs", "devtmpfs", "devfs"]
[[inputs.diskio]]
[[inputs.kernel]]
[[inputs.mem]]
[[inputs.processes]]
[[inputs.swap]]
[[inputs.system]]
```

Les différentes sections du fichier de configuration sont spécifiées par un mot-clé entre crochets, permettant de regrouper les options hiérarchiquement. Cependant, les mentions entre doubles-crochets (**[[** et **]]**) ne sont pas des sections, mais une liste de plugins à utiliser, suivis éventuellement d'options qui les concernent. C'est le cas, par exemple de **[[inputs.disk]]** récoltant des mesures (ou *metrics*) sur l'utilisation des supports de stockage où on choisit d'ignorer différents types de systèmes de fichiers dits « virtuels » avec un système GNU/Linux.

Une fois les données des ESP8266 collectées par Telegraf, on arrive rapidement à se composer un tableau de bord complet avec une facilité déconcertante. Ici, ce ne sont pas moins de 8 capteurs qui sont réunis en un seul écran, avec visualisation en temps réel sous forme de jauges et sur quelques jours sous forme de graphiques.

Je n'apprécie pas particulièrement la syntaxe du fichier en raison de la confusion possible entre les sections et la mention des plugins. En effet, on pourrait supposer qu'il ne s'agit que de sections vides n'ayant donc aucun impact sur la collecte de données. Pourtant, en ajoutant, par exemple, une simple ligne **[[inputs.net]]** puis en redémarrant Télégraf, vous vous mettrez immédiatement à collecter des mesures sur la ou les interfaces réseau.

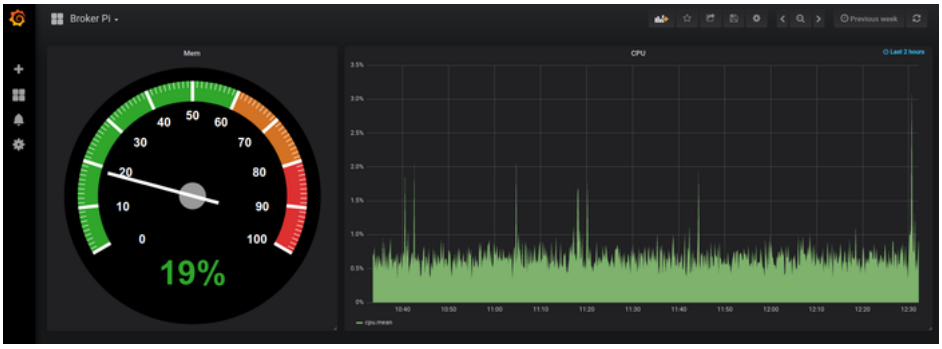
Pour ajouter nos mesures, il nous suffit donc d'ajouter un plugin : **mqtt_consumer**. Celui-ci agit comme un client MQTT en s'abonnant aux topics de votre choix et en stockant les valeurs dans la base Telegraf d'influxDB. Nous n'avons que quelques lignes à ajouter dans la configuration :

```
[[inputs.mqtt_consumer]]
# Broker
servers = ["ssl://raspibase.local:8883"]
# niveau de service
qos = 0
# topics auxquels s'abonner
topics = [
"maison/+/temp",
]
# Authentification
username = "sondes"
```

```
password = "mot2passe"
# TLS
tls_ca = "/etc/mosquitto/ca.crt"
insecure_skip_verify = false
# Format de données reçues
data_format = "value"
data_type = "float"
```

La plupart des lignes de configuration parlent d'elles-mêmes puisque nous connaissons déjà la plupart de ces concepts. Notez cependant différentes choses importantes :

- L'adresse du broker débute ici par **ssl:** afin de préciser une utilisation via SSL/TLS. Sans cela, il faudrait utiliser **tcp:**. Remarquez également que Telegraf n'est pas en mesure d'utiliser mDNS (du moins dans cette version en paquet) et que l'utilisation d'une adresse IP est exclue en raison de l'utilisation simple de certificats TLS. Pour contourner le problème, il faudra ajouter une ligne dans le fichier **/etc./hosts** de la Pi en précisant l'adresse IP de celle-ci et son nom d'hôte suffixé de **.local** (par exemple ici **192.168.74.1 raspibase.local**).
- La ligne **qos** permet de préciser le niveau de qualité de service. Il en existe trois en MQTT, numérotés de 0 à 2. Avec une QOS à 0, les messages sont simplement envoyés sans autre forme de procès. En QOS 1, l'abonné va accuser réception du message et le broker s'assurera que le message a été livré au moins une fois. Enfin, en QOS 2, un échange plus complexe entre le broker et le client permet de s'assurer que le message est délivré une seule et unique fois, ni plus, ni moins. La bibliothèque Arduino *PubSubClient* que nous utilisons ne sait malheureusement gérer que le QOS de niveau 0.
- Telegraf est en mesure d'utiliser bon nombre de formats de données et ceci impactera le croquis que vous utiliserez sur vos sondes ou capteurs. Dans cet exemple de configuration, le croquis utilisé publie une simple valeur en virgule flottante (une température). Le format utilisé est donc une valeur de type **float**.
- Le ou les topics auxquels s'abonne Telegraf sont listés dans la portée de **topics={}**, un par ligne, séparés par une virgule. Comme vous pouvez le voir ici, il n'est pas nécessaire de lister l'ensemble des topics explicitement, mais simplement d'utiliser un caractère joker comme nous l'avons vu dans l'article d'introduction à MQTT. Les topics seront individuellement stockés dans la base avec la valeur contenue dans le message délivré.



Au-delà de l'utilisation via une connexion avec un navigateur web, on imagine sans peine une application plus visuelle. En effet, rien ne vous empêche d'utiliser un écran HDMI avec votre Pi et d'afficher en permanence les informations qui vous paraissent importantes, sous la forme de panels lisibles de loin.

Une fois cet ajout fait, et bien entendu adapté à vos topics et besoins, il ne vous restera plus qu'à redémarrer Telegraf et attendre que quelques données soient collectées. Dans Grafana, vous pourrez alors ajouter un nouveau panel en récupérant les « Metrics » dans la table « mqtt_consumer » de la base « telegraf » et en ajoutant une clause WHERE afin de sélectionner uniquement le topic qui vous intéresse. Vous pourrez alors avoir autant de jolies jauges et graphiques que vous avez de capteurs et créer un magnifique dashboard.

6. EXPLORER, EXPLORER ET EXPLORER

Nous avons ici jeté les bases de ce qui pouvait être fait avec et grâce à MQTT. Ce protocole fait office de base commune et de liant pour bon nombre d'applications, ce qui ouvre des perspectives très intéressantes et constitue un gain de temps très appréciable. Car c'est en réalité bien de ça qu'il s'agit, en effet, tout ceci pourrait être réalisé avec une approche plus « artisanale » : échange HTTP ou autre entre les ESP8266 et la Pi, collecte des données avec un script Python, enregistrement dans une base générique ou un fichier texte, génération de graphiques avec un outil quelconque ou « maison » et composition des pages HTML à la main.

C'est là beaucoup de travail, sans doute pour un résultat qui ne sera graphiquement, fonctionnellement et ergonomiquement pas à la hauteur de ce que proposent MQTT, InlufxDB, Telegraf et Grafana. Je vous invite cependant à ne surtout pas vous en tenir à ce que permettent de rapidement faire ces composants logiciels, mais d'explorer plus avant afin de découvrir les autres fonctionnalités disponibles, qu'il s'agisse du système de notification de Grafana ou des fonctionnalités de consolidation de données de Telegraf et InfluxDB. Comme toujours, le premier résultat n'est en réalité qu'un point de départ, des fondations, et c'est à vous d'en faire une monstrueuse cathédrale selon vos préférences et besoins...


LES
ÉDITIONS
DIAMOND
STIMULANT DE MATIÈRE GRISE