DD2 Monitoring

tTh

28 janvier 2019

1 Introduction

Dans le cadre du futur projet *phytotron*, nous aurons besoin d'un système de surveillance de divers paramêtres, du genre la température et l'humidité de l'air ambiant ou de l'enceinte thermostatée.

Et comme nous sommes au Tetalab, nous allons essayer d'en faire le plus possible nous même, tant pour l'aspect didactique que pour la possible ré-utilisation de la chose dans un contexte artistique comme nous arrivons parfois à le faire.

La première étape sera donc la mise au point d'un *cadriciel* permettant d'essayer diverses options en se basant sur une conception modulaire de l'ensemble.

Les valeurs à mesurer étant de diverses natures, nous aurons l'occasion de découvrir plein de capteurs différents : température, humidité, `pH`, transparence, lumière, vibrations...

2 Présentation générale

2.1 Capteurs

Actuellement nous disposons d'un capteur température et humidité, le `RDing TEMPERHUM1V1.2` qui semble un peu étrange à interpeller, et qui sera destiné à mesurer l'ambiance météo du Double Dragon [1], l'influence humaine n'étant pas à négliger en cas d'afflux du public..

Par la suite, nous pourrons récupérer (par liaison série, cf page 7) des données diverses en provenance de l'automate de contrôle de l'enceinte.

Le premier capteur de température sélectionné est le LM35 qui fournit en sortie une tension linéairement proportionnle à la température. Ils seront connectés sur l'automate qui s'en servira pour la régulation thermostatique.

Il nous reste à choisir d'autres capteurs pour d'autres métriques : humidité, lumière, vibrations...

Notre dd2monitor devrait donc aussi être capable d'envoyer des alertes en cas de souci, par exemple si des algues [2] tentent de s'échapper.

1. Pas de mauvais esprit, merci...
2. ou des pleurotes.

2.2 Stockage

Pour entreposer toutes ces valeurs numériques, il existe plusieurs choix, et nous allons en évaluer quelques uns :

`rrdb` (page 5), `influxdb` (page 6), `gnocchi` (page 7) `Sqlite` (page 7)

La représentation interne des valeurs reste à définir pour la plupart d'entre elles.

2.3 Affichage

Nous allons laisser un petit bac-à-sable pour Fred Fermion, qui nous tartine les oreilles depuis bien trop longtemps avec son machin.js.

Mais sachez déja que l'automate sera équipé d'un minitel, pourquoi ne pas en mettre un second sur le monitoring ? Après tout, un peu d'eyecandy ne peut pas faire de mal. C'est expliqué à la page 8.

3 Outils

3.1 Simulations

Bien entendu, mettre au point un système de monitoring pour un futur phytotron pose un problème : d'où viennent les premières mesures ?

C'est pour ça qu'il y a déja un générateur de *fake-values* qui ne demande qu'à grandir.

Nous avons également un générateur de nombres divers et incohérents qui envoie des quadruplets d'entiers 10 bits précédés d'un caractere de bonne efficacité.

3.2 Exemples

Promis, on va en mettre ! Dès que ça marche...

Un premier exemple avec rrdb en page 5. Le second cause des premiers essais du LM35, capteur de température analogique branché sur un Arduino et relié par un port série (page 8).

Pour continuer dans une démarche disruptive, des outils avancés de visualisation sont proposés dans la rubrique 10.2 qui parle de `vt100`.

3.3 Analyses

Bla bla bla... Corrélations, Gnuplot, toussa...

Peut-être demander à Schmod777 des références de documents bourbakistes ? J'ai entendu parler dans IRC de choses étranges, comme ça :

<schmod777> s/booz/booze pour la courbe qui majore les autres en moyenne serait d'une criante justesse scientifique ;)

Ce qui donne quand même à réfléchir. À se demander quel savoir allons-nous pouvoir déduire de ces[3] chiffres improbables. (Re-)Découvrir les lois de l'inertie thermique ? Générer des formes d'ondes spatialisables ? Déplacer des petites *bubulles* colorées ? Une histoire pour la section 8 ?

3.4 Archivage

La création de cet outil de surveillance est un long parcours pavé d'essais et d'erreurs. Tout cela peut générer beaucoup de données. À titre d'exemple, la capture de température des premier essais sort environ 800 Ko par jour. 99.99% de ces chiffres sont inutiles, mais il peut arriver qu'on désire conserver l'historique d'une expérience réussie[4] ou d'un *epicfail*[5].

Nous devons donc rencontrer quelqu'un qui maitrise cette partie de la mouvance détournementale de l'espionnage. Je pense que Yaya pourra nous éclairer de ses lumières, si on lui demande gentiment.

4 Configuration

Pour adapter cet outil de surveillance aux variations du monde réel, nous devons nous-même lui décrire ce monde. Une description qui se fera avec des lignes de la forme '`input_device s /dev/ttyACM0`' qui représentent des tuples *clef-type-valeur*[6] de choses diverses.

La syntaxe n'est pas encore fixée, mais un fichier exemple est disponible pour des explication plus récentes, donc plus en rapport avec la réalité du code. En voici un extrait :

3. non, c'est pas du bigdata.
4. En fait, c'est comme ça que la science existe
5. En fait, c'est comme ça que la science avance
6. Laissons les canards tranquilles.

```
# serial input from the control cpu
input_device            s       /dev/ttyACM0
input_speed             i       9600
# -------------------------------------------------
# some values for the eyecandy displays
eyecandy_banner         s       hacked by tTh
```

5 RRDB

Première tentative d'utilisation le lendemain du premier apéro 2019 du Tetalab [7].

Je suis parti sur quelques scripts shell, pour créer, mettre à jour et analyser les enregistrements d'une valeur de type GAUGE.

5.1 Create

```
#!/bin/bash

source ./commun.sh
starttime=$(date +'%s')
echo creating $RRDB at ${starttime}s since epoch
rrdtool create $RRDB                        \
        --start $starttime                  \
        --step 60                           \
        DS:value:GAUGE:150:0:10             \
        RRA:AVERAGE:0.5:1:60
```

5.2 Update

Une fois la base créée, il faut bien la remplir. Dans cet exemple, nous allons utiliser le *load* de notre Linux.

```
#!/bin/bash

source ./commun.sh
ctime=$(date +'%s')
value=$(cut -d ' ' -f 1 /proc/loadavg)

# inject value in the rrdb file
rrdtool update $RRDB ${ctime}:${value}
```

7. Jean-Yves, je vous demande de vous calmer !

5.3 Analyze

Nous allons essayer d'exploiter les données dûrement acquises pendant les heures qui précèdent...

```
#!/bin/bash

source ./commun.sh
tmpf="somevalues.dat"
rrdtool fetch $RRDB LAST                                    |
tr -d ':'                                                   |
awk    '
       (!/nan/ && NF==2) { print $1, $2 }
       '                                                    \
                        > ${tmpf}

# as an example, we are gnuploting our datas
gnuplot << __EOC__
set term png size 800,600
set output "graphe.png"
set grid
plot "${tmpf}" with lines
__EOC__

rm ${tmpf}
```

Il semble bien que l'utilisation de `fetch` ne soit pas vraiment prévue pour ça, donc j'en arrive à la conclusion que quelque chose m'échappe.

On va laisser ça en suspens pour le moment.

6 InfluxDB

La communication avec la bédédé se fait *over HTTP*, un peu comme tous ces trucs de d'jeunz d'aujourd'hui...

Au programme : écriture d'un injecteur en Perl, en suivant plus ou moins l'exemple de rrdb..

Ceci dit, en Debian stable, on n'a que la version 1.0, qui ne correspond plus trop à l'actualité. Et la *current* est trop fatiguante à compiler pour ce soir, ni même pour ce week-end.

7 Gnocchi

`pip install gnocchi[postgresql,ceph,keystone]`, finalement, ça ne me donne pas trop envie. C'est du genre *usinagaz*.

8 Sqlite

À regarder de près

Est-il possible de traiter des *time series* en SQL ? Peut-on utiliser Sqlite depuis un programme en Perl ? Faut-il commencer à trouver un *usecase* crédible ? NodeJs peut-il lire du Sqlite ?

9 Serial coms

Il va y avoir deux liaisons série entre l'automate et le monitoring. La première, que nous allons juste entrevoir, passera par le port USB de la carte Arduino. La seconde passera par un port série auxiliaire [8] de celle-ci.

9.1 Un gros souci

```
tcgetattr(uart0, &options);
options.c_cflag = baudbits | CS8 | CLOCAL | CREAD;
options.c_iflag = IGNPAR;
options.c_oflag = 0;
options.c_lflag = 0;
tcflush(uart0, TCIFLUSH);
tcsetattr(uart0, TCSANOW, &options);
```

Et en fait, le `read` sur le *fd* du serial device n'est pas bloquant, `perror` annonce *success*, mais rien ne marche. L'année 2019 va commencer sur du vaudou programming [9].

29 décembre 2018 : le petit grain de magie est très simple à mettre en oeuvre, mais très difficle à spotter dans le gazillion d'options. Il semblerait que mettre `options.c_cc[VMIN]` à 1 permet d'avancer vers l'étape suivante.

Laquelle étape est une tentative d'utilisation de `select(2)`, dans l'objectif de pouvoir gérer nous-même le *timeout*, laquelle tentative n'est pas du tout concluante.

8. Il y en a 4 sur le mega 2560
9. aka shotgun debugging.

9.2 Protocole

L'automate va avoir plusieurs types de données à envoyer. Nous allons donc transférer ces valeurs sous forme de ligne de texte commençant par un caractère clef (par exemple T pour les températures) et se terminant par un *newline*.

9.3 Un peu de code

10 Détournements

Dans le contexte myryssien, il est évident que l'aspect artistique doit être dès le départ pris en compte. Les possibilités ne seront limitées que par votre manque d'imagination.

Si vous voulez un petit exemple, imaginez des courbes de température qui pilotent un `uGen` de Chuck ou des algues dont la lumière envoie du `cv/gate` en temps réel.

10.1 Variante sonore

Un peu de `awk` pipé dans du `sox` ?

10.2 Eye candy

Pour maximiser l'impact visuel, il sera convenant de sortir du cadre pseudo-moderne des omniprésents écrans de ces smartphones [10] qui nous lavent le cerveau. Sortons donc du contexte pixeliste et revenons aux fondamentaux : le caractère, avec toute la simplicité de son concept sémantique.

11 Conclusion

En fait, tout reste à faire. Mais ça peut être un beau projet aux implications et usages multiples.

Un couteau suisse de la capture du monde réel, un point pivot de nos diverses interprétations de l'univers [11] et des interactions étranges entre des paramètres sans relation clairement definie.

10. Quand les téléphones étaient attachés par un fil, les humains étaient libres.
11. non, la terre n'est pas plate

Table des matières

Index