Poêle à pellets domotique

Projet en cours d'élaboration

Sur cette page sont décrites toutes les étapes pour domotiser son poêle à pellets, de la mise en place d'un thermostat à base de Raspberry PI et d'une sonde extérieure, au pilotage à distance du poêle, à la mise en place d'un calendrier afin d'économiser du carburant et de gagner en confort 😎 !

Le matériel de base :

- Une Raspberry Pi de préférence avec un port Ethernet ou un dongle Wifi, on supposera que l'on part sur une Raspberry Pi B (ou dérivée) pour faciliter les choses par la suite, qu'on nommera plus simplement RPI ;
- Son alimentation 5V (2A de préférence, pour alimenter tous les systèmes électroniques que nous souhaiterons ajouter au fur et à mesure) ;
- Une carte SD de 8 Go au minimum, on est vite à court d'espace disque si l'on est dispendieux en tests et installations logicielles diverses, et que l'on ne veut pas passer une temps fou à optimiser tout ça 🙂 !
- Un lecteur de carte SD adapté au modèle de SD utilisé sur votre RPI.

Première étape - Installation de Raspbian et configuration de base

- Télécharge une image disque de Raspbian Jessie (le module de gestion de la sonde de température intégré par défaut dans les précédentes version est bugué, ça a été corrigé depuis, après 1 an d'attente !) : https://www.raspberrypi.org/downloads/raspbian/.
- Dé-zippe l'image.
- Utilise la commande dd pour tout OS basé sur le noyau Linux (alias Linux) ou Win32DiskImager sur Windows pour copier l'image sur la carte SD du RPI. Pour plus d'info, réfère toi à ceci ❗ un peu d'anglais : https://www.raspberrypi.org/documentation/installation/installing-images/README.md.
- Une fois la copie faite, démonte proprement la carte SD afin d'éviter toute mauvaise surprise 😉.
- Insère la carte SD dans le RPI, mets le sous tension, branche le câble Ethernet, et c'est parti 🙂 !
- Connectons nous à ce RPI ! Nous allons utiliser le protocole SSH, avec comme identifiants de base `pi`, et mot de passe `raspberry`. Selon qu'on se trouve sous Linux, Windows ou autre, tu peux utiliser le client SSH qui te plait : pour moi ce sera la commande `ssh` sous Linux ou le logiciel Putty pour Windows.
- Pour commencer entre la commande `sudo raspi-config` et sélectionne Expand Filesystem afin de bien utiliser tout l'espace disponible sur la carte SD. Profites-en pour personnaliser le

hostname du RPI, Vulcain, à tout hasard 😛, pour paramétrer la timezone, etc.
- Redémarre le RPI comme suggéré.

Deuxième étape - Branchement d'une sonde de température et configuration de Raspbian

Matériel nécessaire pour cette phase du projet :

- Une sonde de température DS18B20 (existe en modèle étanche ou non)
- Une résistance de 4,7 kΩ
- Du matériel pour relier tout ça au RPI, au choix, connecteurs divers, platine de prototypage électronique, matériel de soudure : à toi de voir en fonction de ce que tu veux faire 😉.

Explications matérielles à faire

Le déroulé de cette étape consiste à paramétrer Raspbian afin de pouvoir lire notre sonde de température qu'on utilisera pour faire un thermostat filaire de la longueur que l'on souhaite, sans toucher au poêle d'origine :

- Commence par installer l'éditeur de texte de votre choix, pour moi ce sera vim via cette commande : `sudo aptitude install vim`.
- Une fois cela fait, modifie le fichier `/boot/config.txt` avec `sudo vim /boot/config.txt` et rajoute cette ligne : `dtoverlay=w1-gpio,gpiopin=4`.
- Redémarre ton RPI avec la commande `sudo reboot`.
- Une fois reconnecté, vérifie que tout marche bien, tout d'abord en listant les périphériques One Wire à l'aide de la commande `ls /sys/bus/w1/devices/`. Tu dois obtenir ceci : `28-0000061521af w1_bus_master1`. `28-0000061521af` étant un identifiant unique, il sera forcément différent dans ton cas. Ensuite, en prenant mon exemple, tu peux afficher la température avec la commande `cat /sys/bus/w1/devices/28-0000061521af/w1_slave`. Si tout va bien, tu dois obtenir quelque chose comme ça :

```
80 01 4b 46 7f ff 10 10 c6 : crc=c6 YES
80 01 4b 46 7f ff 10 10 c6 t=24000
```

Sinon, reprends les étapes du dessus 😉 ou change ta sonde qui peut être défectueuse.

Troisième étape - Installation et configuration du logiciel de gestion de sondes de température

Ce logiciel développé en C, qu'on va appeler sobrement DS18B20Manager nécessite une bibliothèque MQTT client et un serveur MQTT.

- Pour plus d'infos sur la bibliothèque client MQTT utilisée se référer à ce lien :

https://eclipse.org/paho/clients/c/.
- Pour télécharger les sources, les compiler et installer la bibliothèque :

```
sudo aptitude install libssl-dev
git clone http://git.eclipse.org/gitroot/paho/org.eclipse.paho.mqtt.c.git
cd org.eclipse.paho.mqtt.c
make
sudo make install
```

- Ensuite, installe le serveur/broker MQTT Mosquitto, avec la commande `sudo aptitude install mosquitto`. Tu peux installer ce serveur sur n'importe quel OS et n'importe quel RPI. Seule la configuration réseau changera par rapport à notre cas de figure.
- Compile DS18B20Manager avec la commande `g++ -lpthread -lpaho-mqtt3c main.c -o DS18B20Manager`. *à faire : mettre à disposition le code source*
- Pour tester le logiciel, fais ./DS18B20Manager et télécharge MyMqtt sur Android par exemple. Dans le menu Settings paramètre les options réseau, ensuite dans l'onglet Subscribe, abonne toi au topic `temperature_sensor/#`. Tu devrais recevoir la température à jour toutes les secondes.
- Ici, l'idée est que n'importe quel logiciel puisse s'interfacer avec n'importe quel autre, par exemple on peut tracer facilement les messages qui circulent via cette appli Android. En pratique, on va pouvoir facilement créer une interface pour paramétrer les sondes de température, facilement rajouter une application thermostat, une application Twitter Bot et une application de sauvegarde des températures, tout en pouvant choisir sur quelle machine tourne chaque programme. Souple comme utilisation non  ?
- La dernière phase de cette étape consiste à lancer DS18B20Manager au démarrage du RPI. On peut procéder en rajoutant `nohup /home/pi/DS18B20Manager/DS18B20Manager > /dev/null 2>&1 &` au fichier `/etc/rc.local` avant `exit 0`. Une autre solution, bien plus élégante consiste à écrire un script init.
- Entre les commandes suivantes :

```
sudo cp /home/pi/DS18B20Manager/DS18B20Managerd /etc/init.d/
sudo chmod 0755 /etc/init.d/DS18B20Managerd
sudo update-rc.d DS18B20Managerd defaults
```

à faire : fournir le script documenter DS18B20Manager

Quatrième étape - Interfaçage matériel, installation et configuration du logiciel de contrôle on/off du poêle

Nous allons avoir besoin d'un simple relais électromécanique pour faire le montage électronique.

Ajouter schéma

Pour la partie logicielle je me suis amusé à le faire en Bash, vue la fréquence à laquelle vont être effectuées les actions d'allumage et d'extinction, il n'y a pas besoin de passer par du C. De plus vues

les opérations faites, pas la peine de se casser la tête à re-développer tout un client MQTT quand quelques lignes suffisent !

- Nous allons utiliser un utilitaire bien pratique pour gérer les sorties du RPI : la commande gpio de WiringPi. Je passe les détailles, mais tu vas devoir l'installer ! Récupère le répertoire comme cela : `git clone git://git.drogon.net/wiringPi`.
- Ensuite rends toi dans le répertoire nouvellement créé et compile/installe le avec les commandes `cd wiringPi` et `./build`.
- Il faut que tu installes un client mqtt en ligne de commande : `sudo aptitude install mosquitto-clients`.
- Dernière ligne droite, comme pour l'étape précédente, copie `PelletStoveDriverd` dans `/etc/init.d` et lance la commande `sudo update-rc.d DS18B20Managerd defaults`.

Voilà ! Maintenant avec une simple commande `on` ou `off` envoyée sur le topic `pellet_stove_driver/ground_floor/=` de notre broker MQTT nous pourrons démarrer notre sacré poêle à granulés.

modifier le script pour faire une commande de retour + retain message

Cinquième étape - Installation et configuration du logiciel de thermostat

Maintenant nous allons installer les scripts de gestion du thermostat, toujours de mon cru. Pour apprendre à faire tout ça je te conseille d'y jeter un œil, de le modifier, par exemple pour sélectionner la bonne sonde de température. Ne modifie pas l'installation du Fablab en cours STP, ou alors documente ceci dans ce wiki et vérifie que tout fonctionne bien , que l'on ne meurt ni de chaud, ni de froid .

- Nous allons avoir besoin de la commande `bc`, qu'on installe comme suit : `sudo aptitude install bc`. Elle sera utilisée dans l'un des scripts, en particulier, pour comparer des nombres à virgule flottante.
- Comme dans la précédente étape, copie le script de démarrage `ThermostatCored` dans `/etc/init.d`.
- Le script `ThermostatController` va nous servir dans la prochaine étape, afin d'interface le calendrier à notre thermostat.

Comme ce script n'est pas encore documenté, je vais le faire ici, afin de pouvoir le hacker dans de bonnes conditions ! Alors :

- Comme tout thermostat, il est réglable : par exemple, envoie le message MQTT `19.5` sur le topic `/thermostat/ground_floor/target_temperature/=`. Si tout se passe bien, tu auras la confirmation de en t'abonnant au topic `/thermostat/ground_floor/target_temperature`.
- Ce thermostat dispose de 4 modes : Normal, Eco, Démarrage Forcé et Extinction Forcée. Tu

peux faire cela en envoyant, respectivement, les commandes, `normal`, `eco`, `always on` et `always off` sur le topic `/thermostat/ground_floor/mode/=`. De la même manière, pour vérifier que le nouveau mode a bien changé, abonne toi au topic `/thermostat/ground_floor/mode`.
- Pour connaître l'état du système, abonne toi au topic `/thermostat/ground_floor/state`. Tu sauras si le chauffage est démarré si tu reçois le message `running`, ou arrêté, si tu reçois le message `stopped`.
- Pour le reste, je te laisse découvrir ! Par exemple comment change-t-on la sonde de température utilisée par le thermostat ? Ou encore comment changer de système de chauffage ? Tout simplement une histoire de message et de topic 🙂 !

Sixième étape - Installation et configuration du logiciel de calendrier pour affiner les heures de chauffage

Grâce à cette sixième étape nous allons interfacer notre système de chauffage à un calendrier Google afin d'obtenir un système utilisable par tous les bénévoles du Fablab et ce, à distance : l'idéal aurait été d'utiliser un calendrier OwnCloud (méthode autonome et 100% libre 😃), mais n'ayant pas spécialement le temps de re-développer une solution je laisse à quelqu'un d'autre la charge de remplacer cette étape par une autre méthode, ou j'y reviendrais plus tard 😉.

En attendant, voici ce que je te propose, cher lecteur : nous allons utiliser le script GCalCron python disponible à cette adresse : https://github.com/fabriceb/gcalcron/blob/master/README.md.

- Comme indiqué, entre la commande `sudo pip install –upgrade google-api-python-client`.
- Récupère le script en question : `git clone https://github.com/fabriceb/gcalcron.git $HOME/gcalcron`.
- Installe cet outil pour python : `sudo pip install python-dateutil`.
- Installe la commande `at` : `sudo aptitude install at`. Tout le script est basé sur cette commande, je te conseille d'aller voir à quoi elle sert !
- Rends toi sur le site https://cloud.google.com/console#/project et suis les infos du README du script pour récupérer l'ID du calendrier que tu veux utiliser et pour récupérer le fichier JSON d'identification client, qu'il faut renommer en `client_secrets.json` et placer dans le dossier du script.
- Entre dans le répertoire `gcalcron` et lance le script qui s'y trouve comme suit : `python gcalcron.py python gcalcron.py –noauth_local_webserver`.
- Entre l'ID du calendrier que tu veux utliser, copie le lien généré, ouvre le dans ton navigateur Internet, copie le code généré demandé par le script. C'est terminé pour la partie authentification !
- Il ne nous reste plus qu'à synchroniser régulièrement (toutes les 10min) le calendrier en modifiant ce qu'on appelle la crontab, avec la commande `crontab -e` ajouter les lignes :

```
PATH=/opt/bin:/bin:/usr/bin:/sbin:/usr/sbin
*/10 * * * * python /your/home/directory/gcalcron/gcalcron.py
```

Pour utiliser le calendrier il te suffit de créer un événement et d'ajouter la commande que tu veux lancer dans la description, en suivant ce formalisme :

- `-30 /home/pi/ThermostatController/ThermostatController -b` pour lancer le chauffage 30 min avant l'événement en question.
- `end +5: /home/pi/ThermostatController/ThermostatController -e` pour stopper le chauffage 5 min après l'événement en question.

Mais encore :

- -30: /home/pi/ThermostatController/ThermostatController –normal
- -30: /home/pi/ThermostatController/ThermostatController –always-on
- end +5: /home/pi/ThermostatController/ThermostatController –eco
- end +5: /home/pi/ThermostatController/ThermostatController –always-off

Septième étape - Ajout d'un tableau de bord pour suivre les paramètres vitaux du Fablab

Solution n°1 : freeboard.io

Dans cette n-ième étape du projet, nous allons configurer un tableau de bord, comme celui là : https://freeboard.io/board/rmr6-5. Pour cela nous allons avoir besoin d'un script qui vient récupérer les données des différentes applications que nous avons mis en place précédemment, du service https://dweet.io et du service https://freeboard.io. Quelques explications :

- https://dweet.io est aux objets connectés ce qu'est Twitter aux humains ! Avec un peu de JSON et des requêtes HTTPS nous allons rendre publiques ces données via ce service, dont le format sera le suivant :

```
{
    "state": "1",
    "mode": "Normal",
    "target_temperature": "25",
    "ground_floor_temperature": "23.9"
}
```

- Pour plus d'info sur ce service, je t'invite à regarder comment fonctionne tout ça sur https://dweet.io/play/ . Utilise les données précédentes pour faire tes propres tests à la main si ça t'intéresse !
- Ensuite crée un compte sur https://freeboard.io, crée un Freeboard, ajoute lui une source de données https://dweet.io (Add Datasource/Dweet.io).
- Maintenant tu peux ajouter des panneaux et des widgets. Je te laisse découvrir la suite !

Revenons en à nos moutons :

- Comme pour les cas précédents, copie le `FreeBoardControllerd` dans `/etc/init.d` et exécute la commande `sudo update.rc FreeBoardControllerd defaults`.

Et voilà, tout est opé tous nos petits logiciels s'exécutent chacun de leur côté et communiquent entre eux, le petit dernier va juste espionner les sondes de température et le thermostat, pour retransmettre toutes les données utiles à https://dweet.io. L'objet (“thing”) en question est `fablab_chantier_libre/thermostat/ground_floor` : le tableau de bord freeboard que nous avons préparé va exploiter ses données.

Solution n°2 : utiliser freeboard sur son propre serveur

Cette solution rend le projet plus autonome : nous allons nous passer des deux services webs présentés précédemment.

- Pour cela nous allons récupérer les sources de Freeboard, et générer le projet comme suit :

```
git clone https://github.com/Freeboard/freeboard.git
cd freeboard
sudo aptitude install npm
npm install grunt
sudo npm install -g grunt-cli
grunt
```

- Ensuite, installe un plugin client MQTT pour Freeboard. Pour cela, procède ainsi :

```
cd plugins/thirdparty/
wget
https://raw.githubusercontent.com/alsm/freeboard-mqtt/master/paho.mqtt.plugin.js
```

- Ce plugin nécessite une bibliothèque js, à récupérer et à placer de de la façon suivant :

```
cd ~/freeboard/lib/js/thirdparty/
wget
http://git.eclipse.org/c/paho/org.eclipse.paho.mqtt.javascript.git/plain/src/mqttws31.js
```

- Modifie `paho.mqtt.plugin.js` de la manière suivante :

```
(function()
{
    // ### Datasource Definition
    //
    // -------------------
    freeboard.loadDatasourcePlugin({
        "type_name"   : "paho_mqtt",
        "display_name": "Paho MQTT",
        "description" : "Receive data from an MQTT server.",
        "external_scripts" : [
            "/lib/js/thirdparty/mqttws31.js"
        ],
```

Modifie `/lib/js/thirdparty/mqttws31.js` en fonction de la racine du site en question, on verra

juste après comment procéder.

- Puis rajoute le plugin `paho.mqtt.plugin.js` dans le freeboard en modifiant la page web `index.html`, de la façon suivante :

```
        head.js("js/freeboard_plugins.min.js",
    "plugins/thirdparty/paho.mqtt.plugin.js",
              // *** Load more plugins here ***
```

- Maintenant nous allons installer un serveur web, j'ai choisi nginx car il est très performant sur RPI, libre à toi de tenter d'autres expériences, avec apache, par exemple. On installe donc nginx :

```
sudo aptitude install nginx
```

Ensuite, nous allons paramétrer le serveur nginx en modifiant `/etc/nginx/site-enabled/default` de cette façon :

```
server {
      root /var/www/freeboard;
      index index.html index.htm;
      server_name Vulcain;
      location / {
              try_files $uri $uri/ /index.html;
      }
}
```

- Il ne nous reste plus qu'à copier le répertoire `~/freeboard` dans `/var/www` et à redémarrer le RPI :

```
sudo cp -r ~/freeboard /var/www
sudo reboot
```

- Il va aussi falloir mettre à jour le broker MQTT afin qu'il supporte les websockets :

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
sudo aptitude update
sudo aptitude upgrade mosquitto
```

Modifier le fichier de configuration de Mosquitto pour qu'il prenne en charge les websockets :

```
listener 1883
listener 9001 127.0.0.1
protocol websockets
```

Redémarrez le serveur ou le RPI :

```
sudo service mosquitto stop
```

```
sudo service mosquitto start
```

- Maintenant nous pouvons créer notre dashboard en utilisant directement le serveur MQTT ! Pratique, et beaucoup plus élégant que la précédente solution. Crée alors le freeboard comme tu le souhaites en utilisant le plugin MQTT pour les Datasources.

Ensuite récupère la configuration json via SAVE FREEBOARD.

Un fois cela fait, copie `dashboard.json` à la racine du serveur web : tu pourras charger la configuration automatiquement, par exemple via l'URL http://192.168.1.2/index.html#source=dashboard.json.

Il y a quand même un gros défaut à cette solution : un seul client à la fois peut être connecté dans la mesure où le plugin Datasource MQTT dans l'état ne permet pas de générer des identifiants de clients mqtt différents. C'est donc un point à travailler pour qui s'y intéresserait.

Huitième étape - Installation et configuration du Twitter Bot pour contrôler à distance le poêle

Neuvième étape - Installation et configuration du logiciel d'enregistrement des mesures de température

Nous allons ici utiliser le projet ThingSpeak, projet OpenSource développé par l'équipe de MatLab. Suivre ce lien pour plus d'informations sur l'utilisation de ThingSpeak : https://github.com/iobridge/ThingSpeak. Tu peux aussi utiliser le service en ligne https://thingspeak.com, dans ce cas tu n'auras pas besoin d'installer tout ce qui est relatif au projet TeamSpeak.

- Tout d'abord, récupérer le projet :

```
git clone git@github.com:iobridge/thingspeak.git
```

- Installer Ruby 2, Ruby Gems, rails 4 et un SGBD :

```
sudo aptitude install ruby2.1 rubygems rails mysql-server mysql-client
libmysqlclient-dev build-essential libxml2-dev libxslt-dev
curl -sSL https://get.rvm.io | bash -s stable
source /home/pi/.rvm/scripts/rvm
rvm install 2.1
source /home/pi/.rvm/scripts/rvm
```

- Paramétre tout ce qui est relatif au serveur mysql :

sudo gem install mysql

```
cd ~/thingspeak
cp config/database.yml.example config/database.yml
mysql -u root -p
```

```
CREATE USER 'thing'@'localhost' IDENTIFIED BY 'speak';
GRANT ALL PRIVILEGES ON * . * TO 'thing'@'localhost';
FLUSH PRIVILEGES;
exit
```

Modifie config/database.yml en fonction de l'utilisateur rajouté et de son mot de passe. Puis exécute ces commandes :

```
bundle install
sudo rake db:create
rake db:schema:load
```

- Lance le serveur :

```
rails server
```

Dixième étape - Création d'un tableau de contrôle physique

Maintenant que notre système de base est opérationnel, nous souhaiterions avoir un tableau de contrôle physique pour piloter notre chauffage simplement, ça n'est pas obligatoire, ma ça peut être sympa , par exemple pour les personnes qui ne sont pas adeptes de technologie :

- De quoi régler le thermostat et visualiser la température cible ;
- De quoi visualiser et naviguer entre les 4 modes de fonctionnement, à savoir le mode hors-gel, le mode normal, le mode on et le mode off ;
- De quoi visualiser la température extérieure et la température de la pièce ;
- De quoi visualiser l'état du chauffage (on ou off).

Pour tout cela, nous utiliserons le matériel suivant :

- Un ou plusieurs (soyons fous !) codeurs rotatifs incrémentaux : ce composant est la sorte de molette utilisée de façon commune pour régler le volume des autoradios ;
- Des matrices 8×8 de LEDs pour visualiser la température cible du thermostat, sous forme de jauge, pour visualiser les différents modes et l'état du chauffage ;
- Des interrupteurs capacitifs afin de pouvoir sélectionner les différents modes ;
- Deux écrans OLED miniatures pour afficher la température de la pièce et la température extérieure ;
- Notre Raspberry Pi préféré et tout ce qui va avec ;
- Un beau boîtier en bois.

De la même manière que pour toutes les étapes précédentes, j'ai découpé le projet en petites étapes, avec un petit programme à la clé à chaque fois, en C et/ou en Bash, en fonction de mon humeur (et des contraintes techniques, parce qu'il faut quand même que le système soit un minimum réactif !).

Le codeur rotatif incrémental - Késako ?!

Pour faire simple, c'est un capteur qui réagit lorsqu'on fait tourner son axe ; on peut alors en déduire plusieurs informations : le sens de rotation, le nombre de crans passés (“detent” en anglais), la vitesse de rotation, le sens de rotation, etc. Ce type de capteur a tout un tas d'applications diverses ; ici nous allons simplement utiliser le retour haptique de notre capteur (tous n'en ont pas) qui nous permet de “sentir” chaque cran, ainsi que le fait de savoir si la rotation en cours est dans le sens de rotation des aiguilles d'une montre ou dans le sens inverse. De ce fait nous pourrons incrémenter ou décrémenter la température de réglage du thermostat.

Pour les curieux et ceux qui voudraient modifier le système, voici la documentation du capteur : http://www.mouser.com/ds/2/15/EC12E-587908.pdf.

Le capteur dispose de 3 sorties, A, B et C, nous n'aurons besoin de lire que les deux premières sur le RPI. Passons à l'action :

- Fais le montage suivant (ou assimilé tu peux modifier les GPIO cibles comme cela t'arrange, en dehors de la masse et de l'alimentation) :

dessiner le montage

- Modifie/Compile/Installe le logiciel à partir du code source C suivant, logiciel qui décode le capteur et envoie un message MQTT `anticlockwise` ou `clockwise` à chaque cran passé, en fonction du sens de rotation.

mettre à disposition le code source

- C'est gagné  ! Nous pouvons régler le thermostat avec une commande physique simple et éprouvée depuis des années.

Les écrans OLED pour l'affichage de la température intérieure et extérieure

Maintenant nous allons utiliser deux mini écrans OLED 0.96“ de 128 x 64 pixels sur lesquels apparaîtront les températures intérieures et extérieures.

- Pour cela, branche un premier écran en suivant ce schéma :

faire schéma

- Dans notre cas, nous allons utiliser le bus de communication I²C. Pour cela, il va falloir charger les modules correspondants au démarrage. Pour cela utilise la commande `sudo raspi-config` et dans le menu `Advanced Options` sélectionner `I2C Enable/Disable automatic loading` et répondre `yes` aux différentes questions.
- Ensuite, nous allons utiliser ce driver pour piloter notre écran : `https://github.com/hallard/ArduiPi_OLED.git`. Installe d'abord ces bibliothèques : `sudo aptitude install build-essential git-core libi2c-dev i2c-tools lm-sensors`. Télécharge le et compile le avec les commandes `git clone https://github.com/hallard/ArduiPi_OLED.git` et `sudo make`.

- Compile les exemples avec `make` et teste le bon fonctionnement de l'écran avec la commande `sudo ./oled_demo –oled 3`.

Problème de remplissage de la sdcard

Il arrive que atd plante et remplisse la sdcard via syslog, il suffit d'utiliser la commande atq et d'effacer les pending jobs (atrm). Il faut aussi recommencer l'installation du script gcalcron, qui mériterait d'être redéveloppé.

— *Jonathan Alibert 2015/12/02 14:48*