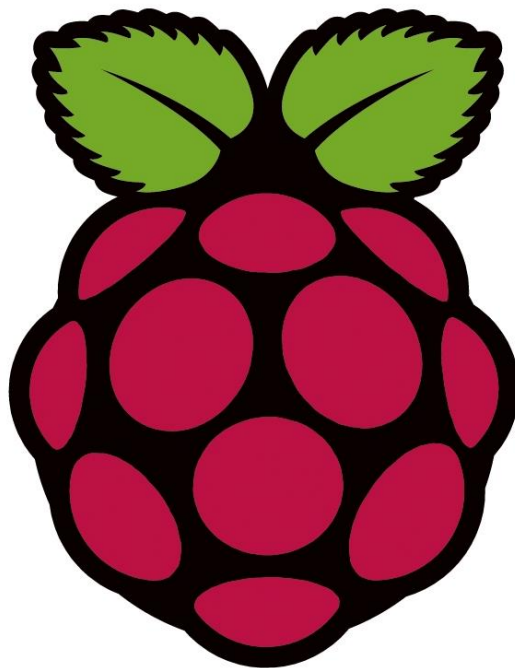


# Projet linux



**GROUPE:**

CATHERIN Félicien  
PACI Jean-Baptiste

## Table des matières

Introduction .....	3
Installation de Rasbian sur une Raspberry Pi.....	4
Prise en main à distance du bureau .....	5
Allumer une LED.....	7
Réception de mail .....	10
Automatisation du script.....	12
Test du montage .....	13



## Introduction

Nous devons pour ce rapport effectuer un projet Linux basé sur ce que nous avons vu en cours d'initiation Linux. Nous avons donc dû chercher une idée de projet intéressant (c'est toujours mieux pour avoir une bonne note) et ludique (c'est toujours mieux pour se motiver).

C'est alors que je me suis souvenu d'une phrase que me répète souvent mon père à propos de mes projets à l'école : « C'est bien sympa ce que tu fais, mais bon... ça fait pas le café ! »

C'est donc dans un esprit de contradiction que nous avons décidé de lui prouver le contraire en utilisant nos connaissances pour démarrer une cafetière à distance grâce à une Raspberry Pi, qui fonctionne sous des distributions Unix.

Ce rapport va donc expliquer toute la démarche que nous avons effectuée pour arriver à nos fins.



## Installation de Raspbian sur une Raspberry Pi

Pour installer Raspbian sur une Raspberry Pi, il faut tout d'abord savoir ce qu'est une Raspberry, et ce qu'est Raspbian.

Une Raspberry Pi est tout simplement un ordinateur, diminué à l'essentiel : une carte mère avec toutes les sorties nécessaires pour l'utiliser comme un ordinateur (ports USB/HDMI/ethernet...), cela permet d'en obtenir à un coût très bas (environ 35€) Raspbian quant à lui, est un des nombreux systèmes d'exploitation Linux que l'on peut installer sur une Raspberry.

Il existe des dizaines de tutoriels qui expliquent comment installer Raspbian sur le micro-ordinateur, faire notre propre tutoriel sur cela serait soit moins bien, soit du plagiat par rapport à ce que nous avons nous-mêmes appris sur internet.

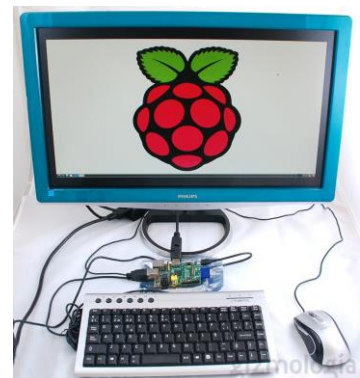
Voici donc un lien utile pour cette installation :

<http://www.tomshardware.fr/faq/id-2929165/configurer-raspberry-installer-raspbian.html>

Voici tout de même les grandes lignes de l'installation :

- Télécharger une image de Raspbian
- Télécharger Win32DiskImager pour pouvoir installer l'image sur une carte SD
- Mettre la carte SD dans la Raspberry et attendre la fin de l'installation
- Vous pouvez maintenant configurer et utiliser votre micro-ordinateur

Vous disposez désormais d'un ordinateur fonctionnel, mais le problème est qu'à ce point-là, nous avons besoin de brancher un écran, un clavier et une souris sur les ports de la Raspberry. Cela n'est pas forcément toujours pratique pour les utilisations souhaitées.



C'est donc pour cela que nous allons vous montrer comment prendre en main votre micro-ordinateur à distance.

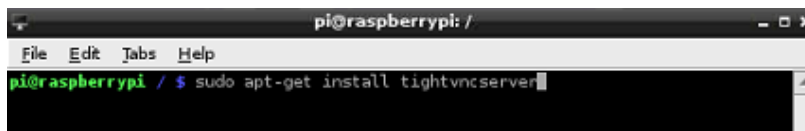
## Prise en main à distance du bureau

Nous nous sommes tournés vers tightVNC, qui permet de prendre en main un ordinateur à distance en installant un serveur sur le poste distant et un client sur l'ordinateur qui souhaite s'y connecter.

- Installez la version client pour votre ordinateur (Windows pour notre part) en téléchargeant sur <http://www.tightvnc.com>

- Ouvrez un terminal console sur la Raspberry et tapez :

«**sudo apt-get install tightvncserver** » Cela permet d'installer la version serveur de tightVNC sur la machine.



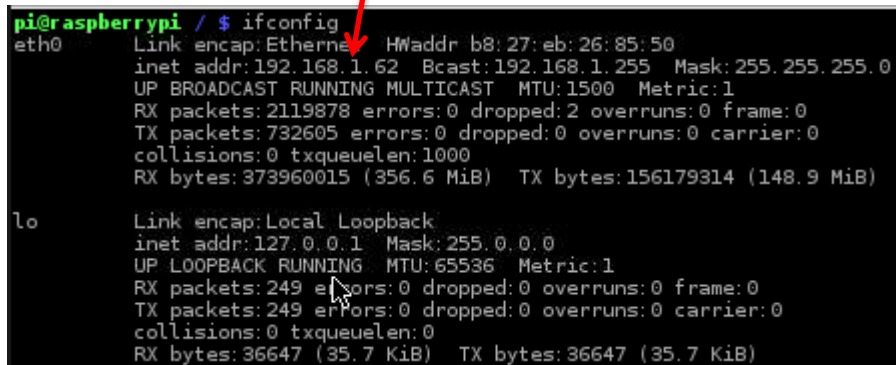
```
pi@raspberrypi: /  
File Edit Tabs Help  
pi@raspberrypi / $ sudo apt-get install tightvncserver
```

- C'est génial ça marche tout seul ! L'installation s'effectue et il nous reste plus qu'à lancer le programme en écrivant la ligne suivante



```
pi@raspberrypi / $ tightvncserver
```

On récupère ensuite l'adresse IP locale du Raspberry en tapant « **ifconfig** »



```
pi@raspberrypi / $ ifconfig  
eth0      Link encap:Ethernet HWaddr b8:27:eb:26:85:50  
          inet addr:192.168.1.62 Bcast:192.168.1.255 Mask:255.255.255.0  
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
          RX packets:2119878 errors:0 dropped:2 overruns:0 frame:0  
          TX packets:732605 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:373960015 (356.6 MiB) TX bytes:156179314 (148.9 MiB)  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1 Mask:255.0.0.0  
          UP LOOPBACK RUNNING MTU:65536 Metric:1  
          RX packets:249 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:249 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:36647 (35.7 KiB) TX bytes:36647 (35.7 KiB)
```

Puis on entre cette même adresse IP dans le client VNC sur l'ordinateur, avant de donner le mot de passe (« raspberry » par défaut).

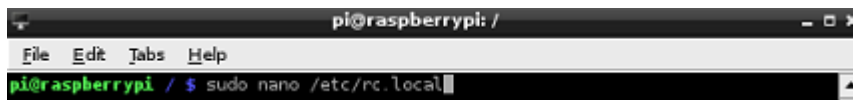
Ca y est, vous êtes connecté à distance, et pouvez contrôler l'ordinateur distant.

Le problème c'est qu'une fois que notre « Rasp » redémarre, on en perd le contrôle et il faut lancer le programme avec la ligne de code « tightvncserveur ».

Nous, on ne veut plus à avoir à connecter notre « Rasp » à un écran, une souris et un clavier.

Pour s'en débarrasser une bonne fois pour toute, on va changer le script de démarrage du mini-pc et démarrer ce contrôle à distance dans le démarrage de notre « Rasp ».

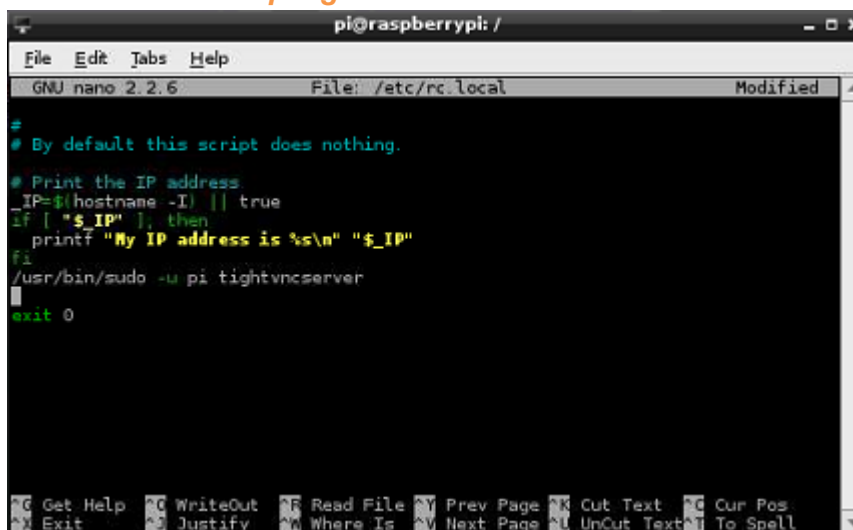
*On tape la commande « **sudo nano /etc/rc.local** »*



```
pi@raspberrypi: /  
File Edit Tabs Help  
pi@raspberrypi / $ sudo nano /etc/rc.local
```

Ajoutez juste avant exit 0

*« **/usr/bin/sudo -u pi tightvncserver** »*



```
pi@raspberrypi: /  
File Edit Tabs Help  
GNU nano 2.2.6 File: /etc/rc.local Modified  
#  
# By default this script does nothing.  
# Print the IP address  
_IP=$(hostname -I) || true  
if [ "$_IP" ]; then  
    printf "My IP address is %s\n" "$_IP"  
fi  
/usr/bin/sudo -u pi tightvncserver  
exit 0  
Get Help WriteOut Read File Prev Page Cut Text Cur Pos  
Exit Justify Where Is Next Page UnCut Text To Spell
```

Ce script s'exécute à chaque démarrage de la Raspberry, et la ligne ajoutée permet donc de lancer le serveur VNC à chaque démarrage.

CTRL+ O pour sauvegarder, CTRL+ X pour quitter.

Nous pouvons désormais passer à la partie pratique du projet :

## Allumer une LED

La Raspberry comme nous l'avons vu précédemment, possède de nombreux ports. Nous allons désormais nous intéresser aux GPIO (General Purpose Input/Output) qui sont des Pins avec lesquelles on peut interagir avec l'extérieur (comme une carte Arduino).

Cependant, nous n'avons pas avec Raspbian un accès direct à ces pins, il faut installer une librairie (wiringPi) que l'on peut télécharger sur ce site :

<https://git.drogon.net/?p=wiringPi;a=summary>

On sélectionne la version la plus récente :

shortlog							
2013-08-03	Gordon Henderson	Added some tweaks to gpio to set alt modes on pins...	master	commit	commitdiff	tree	snapshot
2013-07-28	Gordon Henderson	Bumped version		commit	commitdiff	tree	snapshot
2013-07-28	Gordon Henderson	It helps if you add the files into GIT...		commit	commitdiff	tree	snapshot
2013-07-28	Gordon Henderson	Minor changes to the files and removed a bit of debug.		commit	commitdiff	tree	snapshot
2013-07-28	Gordon Henderson	tidied and tested DRC Serial (renamed it drcSerial...		commit	commitdiff	tree	snapshot
2013-07-24	Gordon Henderson	Added in the PiGlow devLib extension driver.		commit	commitdiff	tree	snapshot
2013-07-23	Gordon Henderson	Added in PiGlow devLib and a couple of examples for...		commit	commitdiff	tree	snapshot
2013-07-23	Gordon Henderson	Added in the SN3218 LED controller IC - as used in...		commit	commitdiff	tree	snapshot
2013-07-23	Gordon Henderson	Reverted gpio readall to older version - new version...		commit	commitdiff	tree	snapshot
2013-07-16	Gordon Henderson	Added in a max5322 SPI D to A chip		commit	commitdiff	tree	snapshot
2013-07-14	Gordon Henderson	gpio Makefile changed to add PREFIX & DESTDIR		commit	commitdiff	tree	snapshot
2013-06-30	Gordon Henderson	Typo in mcp3002.c		commit	commitdiff	tree	snapshot
2013-06-27	Gordon Henderson	Readall command in gpio changed.		commit	commitdiff	tree	snapshot
2013-06-27	Gordon Henderson	Properly added the max31855 files now		commit	commitdiff	tree	snapshot

On clique sur le lien de la flèche, on enregistre et on le dézippe avec la console :  
(On se place avant dans le dossier de téléchargement)

« **tar xfz wiringPi-98bcb20.tar.gz** »

On entre dans le dossier dézippé :

« **cd wiringPi-98bcb20** »

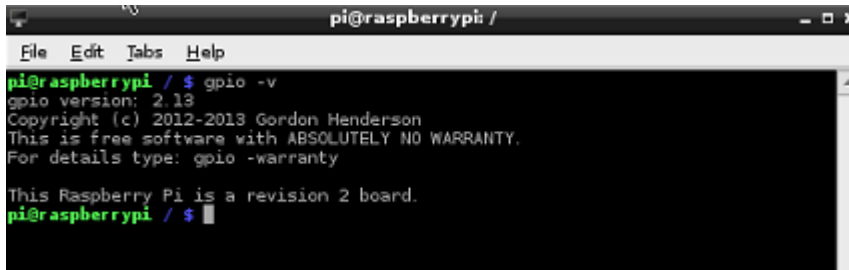
On exécute le tout pour installer les nouvelles fonctions:

« **./build** »

Pour tester le bon fonctionnement :

Afficher la version

« *gpio -v* »

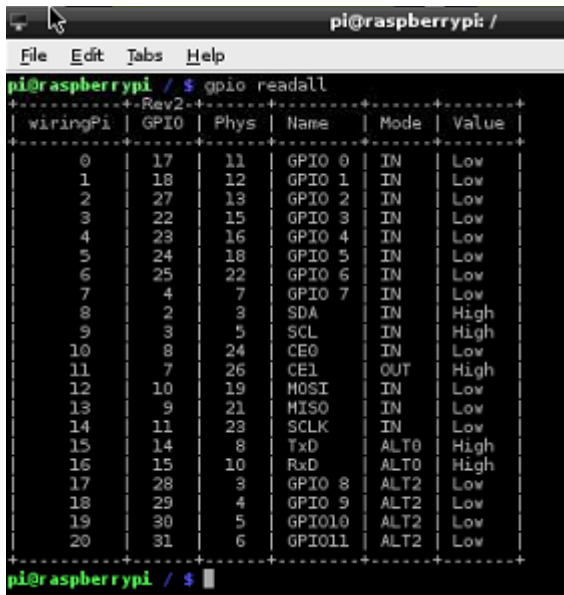


```
pi@raspberrypi / $ gpio -v
gpio version: 2.13
Copyright (c) 2012-2013 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

This Raspberry Pi is a revision 2 board.
pi@raspberrypi / $
```

On souhaite ensuite connaître l'état des Pins (entrées/sorties, Low/High) :

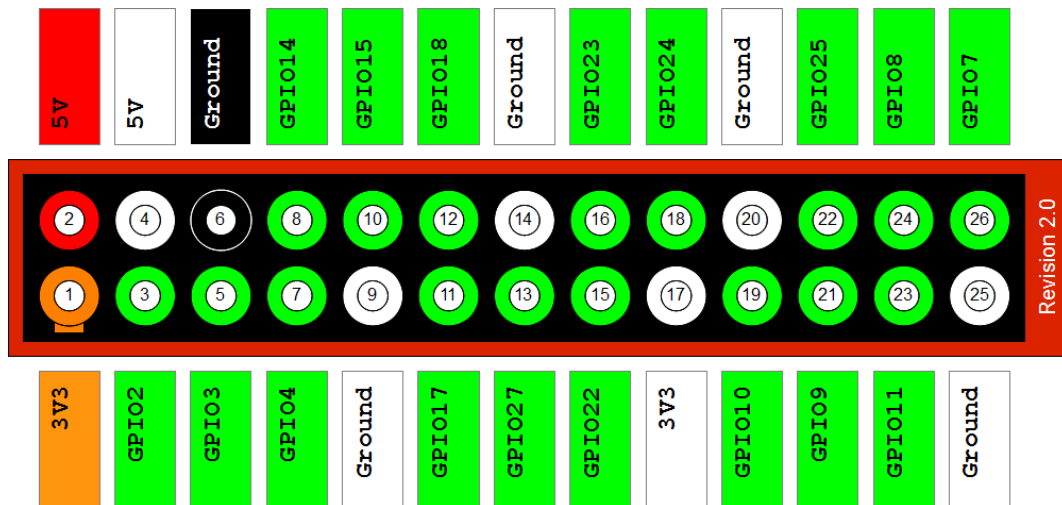
« *gpio readall* »



```
pi@raspberrypi / $ gpio readall
+-----+-----+-----+-----+-----+
| wiringPi | GPIO | Phys | Name | Mode | Value |
+-----+-----+-----+-----+-----+
| 0 | 17 | 11 | GPIO 0 | IN | Low |
| 1 | 18 | 12 | GPIO 1 | IN | Low |
| 2 | 27 | 13 | GPIO 2 | IN | Low |
| 3 | 22 | 15 | GPIO 3 | IN | Low |
| 4 | 23 | 16 | GPIO 4 | IN | Low |
| 5 | 24 | 18 | GPIO 5 | IN | Low |
| 6 | 25 | 22 | GPIO 6 | IN | Low |
| 7 | 4 | 7 | GPIO 7 | IN | Low |
| 8 | 2 | 3 | SDA | IN | High |
| 9 | 3 | 5 | SCL | IN | High |
| 10 | 8 | 24 | CE0 | IN | Low |
| 11 | 7 | 26 | CE1 | OUT | High |
| 12 | 10 | 19 | MOSI | IN | Low |
| 13 | 9 | 21 | MISO | IN | Low |
| 14 | 11 | 23 | SCLK | IN | Low |
| 15 | 14 | 8 | Tx0 | ALT0 | High |
| 16 | 15 | 10 | Rx0 | ALT0 | High |
| 17 | 28 | 3 | GPIO 8 | ALT2 | Low |
| 18 | 29 | 4 | GPIO 9 | ALT2 | Low |
| 19 | 30 | 5 | GPIO10 | ALT2 | Low |
| 20 | 31 | 6 | GPIO11 | ALT2 | Low |
+-----+-----+-----+-----+-----+
pi@raspberrypi / $
```



Voici le tableau des correspondances entre les numéros de pin et sa position sur la carte :



Nous souhaitons donc allumer une diode, que nous branchons entre la pin 26 (GPIO7) et une des masses.

Avec la console, on regarde le tableau de correspondance (gpio readall), la pin 26 correspond au WiringPi 11.

On rentre la série de codes suivante :

```
pi@raspberrypi: /
File Edit Tabs Help
pi@raspberrypi / $ gpio mode 11 OUT
pi@raspberrypi / $ gpio write 11 1
pi@raspberrypi / $
```

« mode » permet de définir si la pin est une entrée (in) ou une sortie (out)

« write » permet de définir pour une sortie si elle est à 0V(Low) ou à 3.3V (High)

On vérifie avec readall :

11 7 26 CE1 OUT High C'est parfait, de plus, la LED s'est allumée :



Pour l'éteindre, on devine qu'on doit rentrer :

« *gpio write 11 0* ».

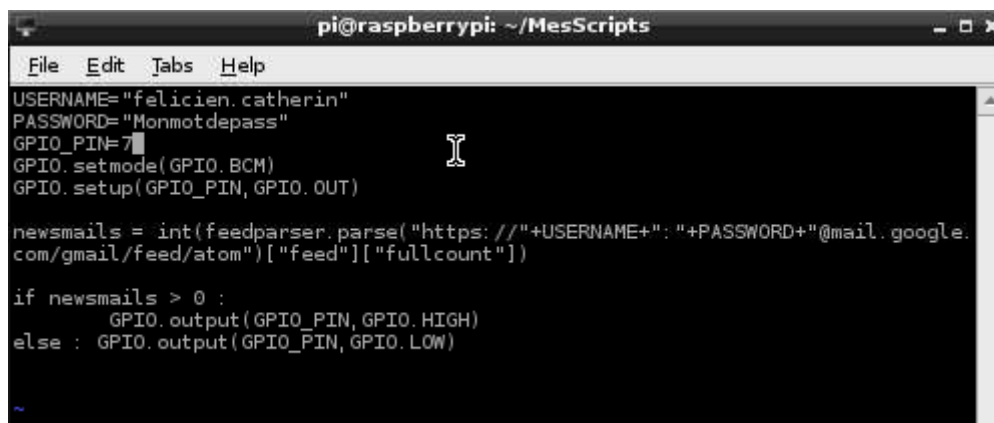
Nous pouvons maintenant passer à l'étape suivante :

## Réception de mail

Nous souhaitons maintenant écrire un script qui permet d'allumer la diode lorsque nous recevons un mail (sur Gmail). Pour cela nous avons dû faire des recherches sur la librairie feedparser.

Feedparser permet de récupérer des informations sur internet, et dans notre cas sur la boîte mail Gmail.

Le script que nous avons rédigé permet d'allumer un LED lorsque qu'il y a des mails non lus :



```
pi@raspberrypi: ~/MesScripts
File Edit Tabs Help
USERNAME="felicien.catherin"
PASSWORD="Monmotdepass"
GPIO_PIN=7
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_PIN, GPIO.OUT)

newsmails = int(feedparser.parse("https://"+USERNAME+": "+PASSWORD+"@mail.google.com/gmail/feed/atom")["feed"]["fullcount"])

if newsmails > 0 :
    GPIO.output(GPIO_PIN, GPIO.HIGH)
else : GPIO.output(GPIO_PIN, GPIO.LOW)
```

Cependant, il est rare que le nombre de mails non lus soit nul, il y a généralement un offset assez important, et donc dans ce cas-là la LED s'allumera en permanence.

Pour cela, nous avons essayé différentes méthodes pour pallier à ce problème :

- Soit écrire en dur l'offset dans le script (si on considère qu'on ouvrira ensuite tous les nouveaux mails)

- Soit en utilisant un offset variable que l'on note dans un fichier grâce à la librairie shelve :

<https://docs.python.org/2/library/shelve.html>

Ces deux solutions fonctionnent mais sont insuffisantes pour la fonction que nous voulons obtenir : nous voulons que la LED s'allume uniquement lorsqu'on lui envoie un mail particulier du type « allume la cafetière », il faut donc récupérer les objets des messages pour savoir si on envoie du courant dans la Pin :

Ce script récupère donc les 20 messages les plus récents et allume la LED si un des objets est « J'ai soif »

```
import feedparser, RPi.GPIO as GPIO

USERNAME="felicien.catherin"
PASSWORD="monmegamotdepasse"
GPIO.setwarnings(False)
GPIO_PIN=7
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_PIN, GPIO.OUT)

d = feedparser.parse("https://"+USERNAME+": "+PASSWORD+"@mail.google.com/gmail/feed/atom")

for i in range(0,20):
    ch= d['entries'][i]['title'].encode('utf8')
    print ch
    if ch == "J'ai soif":
        GPIO.output(GPIO_PIN, GPIO.HIGH)
```

`d['entries'][i]['title']` récupère pour chaque entrée (un mail), le titre (objet)

On comprend désormais que pour allumer la cafetière (ou tout autre objet), il suffit de remplacer la LED par un circuit qui va fermer un interrupteur lorsque la pin est à HIGH.

Nous avons donc utilisé un relai pour choisir si une multiprise est alimentée ou non :



On branche alors sur la multiprise ce que l'on veut (lampe, cafetière etc..) et on peut désormais les commander à distance.

Pour le cas de la cafetière, il ne faut pas oublier de l'éteindre, on peut insérer une pause dans le script au moment où celui-ci allume la cafetière d'une vingtaine de minutes avant de remettre la PIN à Low

Il faut importer la librairie time

Puis utiliser la fonction

*time.sleep(1200) #pour attendre 20 minutes*

avant d'éteindre le tout

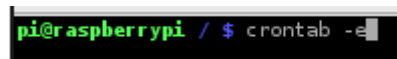
*GPIO.output(GPIO\_PIN,GPIO.LOW)*

## Automatisation du script

Au point où nous en sommes, il faut lancer le script pour qu'il vérifie les mails, autant directement brancher la cafetière nous direz-vous. C'est pourquoi il faut que le script s'exécute régulièrement pour savoir en permanence si un mail a été reçu.

Pour se faire, nous avons dû étudier le fonctionnement du programme crontab qui permet d'automatiser le lancement de tâches.

Il faut donc éditer ce programme et ajouter une ligne pour lancer le programme à intervalle régulier :



La commande permet d'ouvrir le fichier afin de l'éditer

La syntaxe est ensuite très bien expliquée sur le lien suivant :

<http://fr.wikipedia.org/wiki/Crontab>

Minute heure date(jour du mois) mois jour(de la semaine) commande

Pour notre part, pour lancer le programme toutes les 2 minutes on écrit :

*\*/2 \* \* \* \* sudo python /path/nomduscript.py #sudo nécessaire pour l'utilisation de GPIO*

(Les étoiles signifient « pour chaque »)

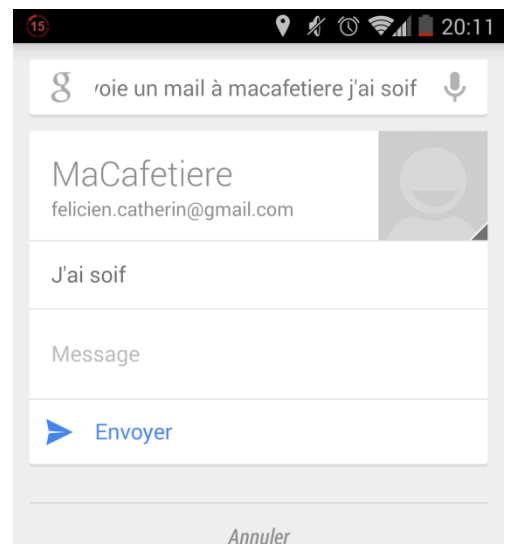
Après enregistrement, le script s'effectue toutes les deux minutes comme désiré.

## Test du montage

Nous avons branché une lampe sur la multiprise pour pouvoir voir plus facilement que cela fonctionne sur une photo :



Nous envoyons un mail grâce à la commande vocale d'un téléphone (cela montre bien le côté domotique du projet où il suffit de parler à son téléphone pour allumer sa cafetière à partir de n'importe où) :





Après quelques longues secondes de patience où on se demande pourquoi le script se répète uniquement toutes les 2 minutes et non pas toutes les minutes, la « magie » opère, et la lumière est ! :

