arno0x0x

Sécurité informatique, Domotique, Raspberry Pi, des 0 et des 1

Commandez vos prises électriques à distance avec le Raspberry Pi

 2 avril 20158 octobre 2015 arno0x0x

Voici enfin venu le premier article traitant réellement de domotique. Il va être ici question de télécommander des prises électriques à commande radio depuis un RPi.

Nous verrons comment commander deux types de prises électriques:

- Modèle DI-O de marque Chacon
- Prise simple de marque Phenix

Ces deux modèles de prises ont en commun l'utilisation de la bande de radio-fréquence 433MHz, très courante en domotique, avec un codage simple des commandes sur cette bande de fréquence.

L'utilisation de prises électriques à commande radio va nous permettre de piloter facilement, c'est à dire sans avoir à tirer de câbles supplémentaires, tout équipement électrique de la maison branché sur une prise. La portée du dispositif présenté ici dépend de la puissance d'alimentation du module, des parois de votre maison, mais en principe on peut compter sur une portée minimale de 30m.

Je me sers de ces prises pour commander des lampes, et des radiateurs électriques (*n'ayant pas de fil pilote*). On prêtera bien attention à la puissance de coupure de ces prises, en particulier pour les radiateurs: les Chacon DI-O peuvent couper jusqu'à 2400W ce qui est largement suffisant pour mes radiateurs de 1500W.

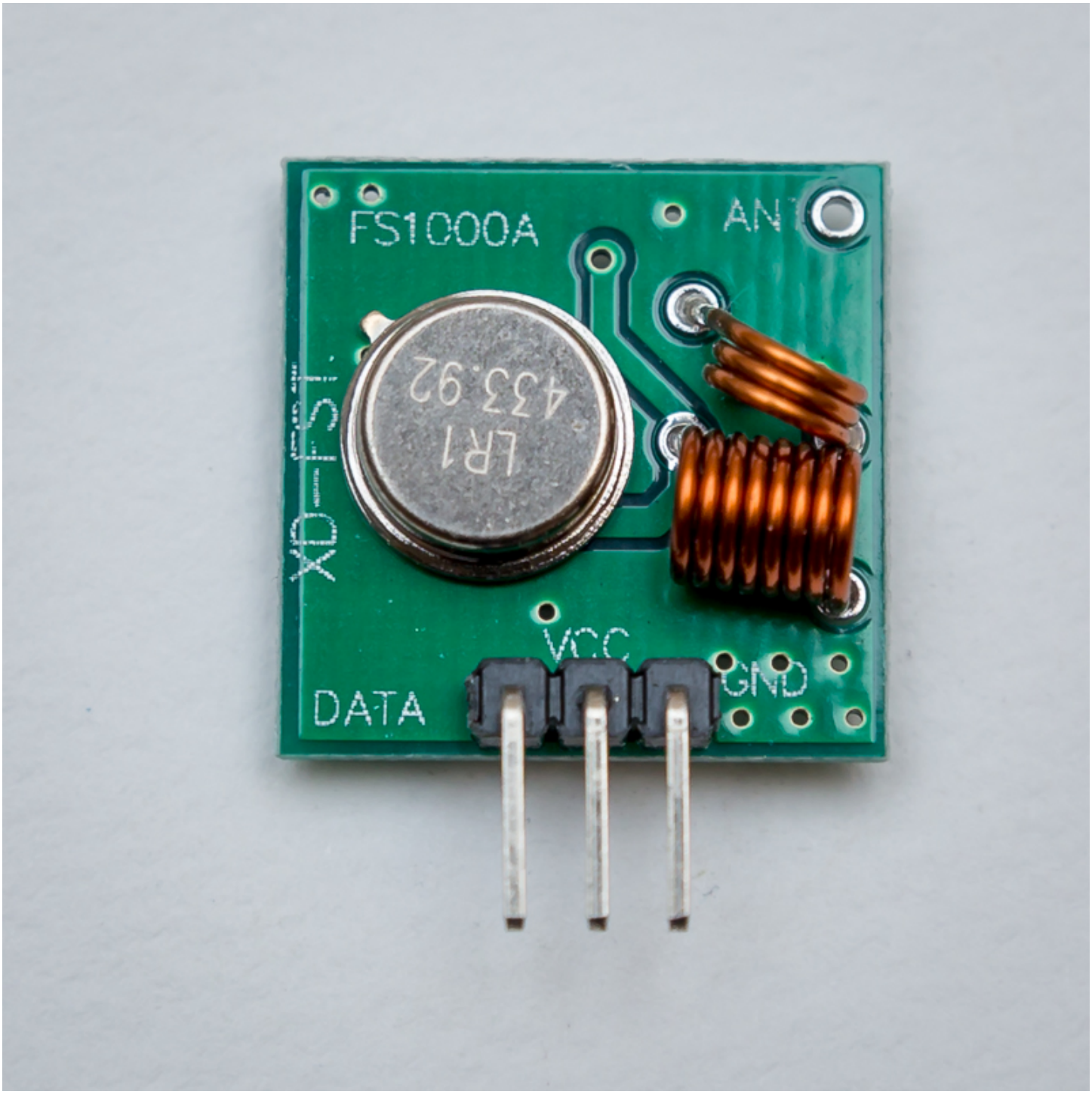
Le matériel

D'abord les prises électriques radio commandées:

(https://www.dropbox.com/s/lpf8myfdz9tjt3s/rf433-outlet-01.jpg?dl=0)

On trouve ce genre de prises assez facilement sur internet. Les Chacon DI-O sont plus chères, mais elles ont un pouvoir de coupure supérieur et on les trouvera sans doute plus facilement que le modèle de prise Phenix qui commence à dater un peu. Concernant les prises Phenix, bien que ce tutoriel ait été testé et validé uniquement avec le modèle présenté ici, il est fort probable que cela fonctionne aussi sur des modèles plus récents, ça serait à tester.

Ensuite, nous allons avoir besoin d'un émetteur radio-fréquence 433MHz. Ce petit composant étant assez répandu, il coûte peu cher, on le trouve par exemple sur Amazon (*recherchez simplement « rf433 »*) sous la forme d'un couple émetteur/récepteur. Nous n'aurons ici besoin que de l'émetteur:



(https://www.dropbox.com/s/i2yf7iwo1mdxmjp/rf433-outlet-02.jpg?dl=0)

Le principe

La fréquence de 433MHz n'est que la *porteuse* du signal, l'information en elle même est transmise à seulement quelques kilohertz, ce que les ports GPIO d'un RPi peuvent parfaitement atteindre. Le principe est donc de connecter un des ports GPIO du RPi sur la broche « DATA » de l'émetteur RF433, et d'envoyer sur ce port les

bonnes trames de communication (*trame: séquence de 1 et de 0 matérialisés par des états haut ou bas d'un signal électrique, ou par des fronts montant ou descendant, et ce pendant une certaine durée*). Chaque marque ou modèle de prise utilise un protocole de communication propre, plus ou moins standard et plus ou moins bien documenté.

Les DI-O de Chacon, par exemple, utilisent le protocole *HomeEasy*, qui n'est apparemment pas bien documenté, mais sur lequel l'inénarrable Idleman (http://blog.idleman.fr)a néanmoins effectué de la rétro-ingénierie afin de comprendre le codage utilisé et le format des trames (*lien* (http://blog.idleman.fr/raspberry-pi-10-commander-le-raspberry-pi-par-radio/)). Un travail similaire a été fait par d'autres personnes pour les modèles de prise Phenix.

On remercie donc tout ce beau monde de nous fournir les programmes qui permettent de communiquer avec nos modèles de prises.

Les branchements

On va commencer par souder une antenne sur l'émetteur RF433. L'antenne peut être faite à partir de n'importe quel type de fil conducteur électrique, dans l'idéal suffisamment rigide pour tenir droit. La taille idéale de l'antenne est dictée par la fréquence utilisée (*ici 433.92MHz pour être précis*). On choisira de faire une antenne de la taille d'un 1/4 d'onde:

> *longueur onde entière = vitesse de la lumière / fréquence en MHz*
> *soit longueur d'onde = 300 000 / 433,92 = 691,37 mm*
> *1/4 d'onde = 691,37 mm / 4 = **~ 173**mm*

On fera donc une antenne d'environ 17cm (*si vous arrivez à faire 17,3cm, tant mieux*). On soude cette antenne sur l'émetteur RF433 sur son bornier identifié « ANT »:

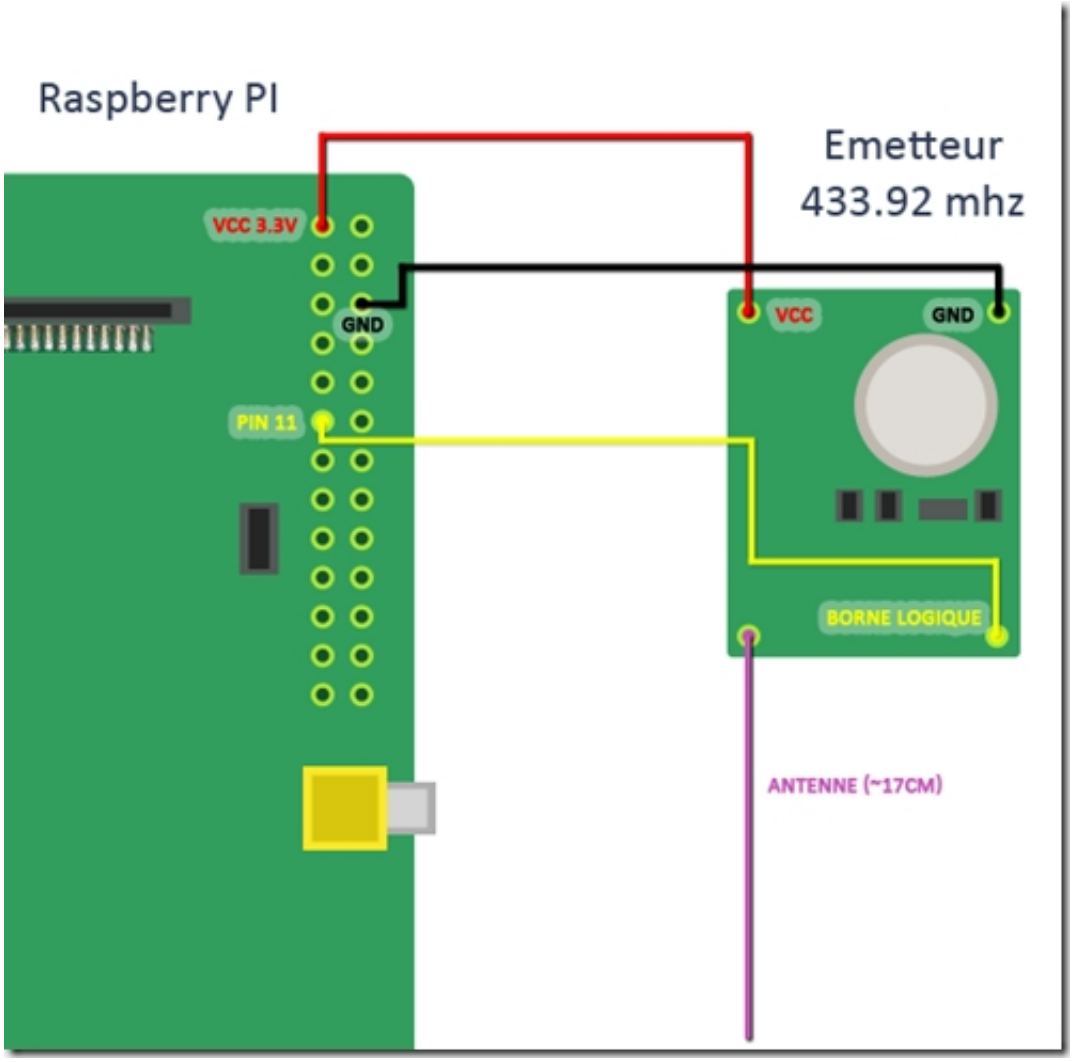
(https://www.dropbox.com/s/rfc3t96wgc63qhy/rf433-outlet-04.jpg?dl=0)

Pour la suite des branchements, on peut utiliser une plaque de prototypage (*voir photo ci-dessus*) qui permet de faire des connexions sans soudure pour les composants électroniques ayant un pas d'écartement standard entre leurs broches (*2,54mm*).

On relie l'émetteur RF433 au RPi:

- la broche « GND » du module à la masse du RPi (*pin numéro 6*)

- la broche « VCC » du module à l'alimentation 3.3V du RPi (*pin numéro 1*)
- la broche « DATA » du module au GPIO sur le pin numéro 11



(https://www.dropbox.com/s/7in1xtdbmm241ob/rf433-outlet-05.jpg?dl=0)

– source http://blog.idleman.fr (http://blog.idleman.fr) –

La partie logicielle

Nous allons maintenant télécharger, compiler, et installer sur le RPi les deux programmes correspondant à chacun des modèles de prise, ainsi que les modules ou bibliothèques dont ils dépendent. Il nous faut:

- La bibliothèque *WiringPi*, qui est une dépendance pour nos deux programmes
- Un programme de commande pour les prises Chacon DI-O
- Un programme de commande pour les prises Phenix

Installation de WiringPi:

Cette bibliothèque propose une interface de communication pratique pour le développement d'applications utilisant les ports GPIO des RPi. L'installation est expliquée sur la page mentionnée ci-dessous, elle est très simple et je suggère d'utiliser le « Plan B » proposé par le sieur Gordon. Le script d'installation compile la dernière version de WiringPi et l'installe sur le système automatiquement.

Tout est là: http://wiringpi.com/download-and-install/ (http://wiringpi.com/download-and-install/)

Programme de commande des prises Chacon DI-O:

Idleman (http://blog.idleman.fr) a développé un programme en C++ (*sous licence CC-BY-SA*) qui permet d'envoyer des ordres aux prises Chacon DI-O en implémentant une partie du protocole *HomeEasy*. Le package qu'Idleman propose en téléchargement contient plusieurs choses qui ne nous intéressent pas dans le cadre de cet article (*mais allez jeter un œil à son blog (http://blog.idleman.fr/), ça vaut vraiment le coup*). J'ai légèrement modifié son code pour supprimer un appel système qui me générait une erreur. Je vous propose donc une petite archive zip contenant juste le fichier à compiler.

1. Télécharger le petit package que j'ai préparé ici (https://www.dropbox.com/s/cjk6iydb1ugizd4/chacon_send.zip?dl=0) puis copiez le sur le RPi
2. Dézipper l'archive et placez vous dans le répertoire résultant
3. Compiler le programme avec g++, le compilateur C++ fournit en standard sur le RPi. **NB**: l'installation de WiringPi ci-avant est indispensable.

Ce qui donne la séquence suivante, en ligne de commande:

```
pi@domopi ~ $ wget -q -O chacon_send.zip https://www.dropbox.com/s/cjk6iydb1ugizd4/chacon_send.zip
pi@domopi ~ $ unzip chacon_send.zip
Archive:  chacon_send.zip
   creating: chacon_send/
  inflating: chacon_send/chacon_send.cpp
  inflating: chacon_send/wiringPi.h
pi@domopi ~ $ cd chacon_send/
pi@domopi ~/chacon_send $ g++ -o chacon_send chacon_send.cpp -lwiringPi
pi@domopi ~/chacon_send $ ls -la
total 40
drwxr-xr-x  2 pi pi  4096 Apr  1 20:11 .
drwxr-xr-x 11 pi pi  4096 Apr  1 20:10 ..
-rwxr-xr-x  1 pi pi 13177 Apr  1 20:11 chacon_send
-rw-r--r--  1 pi pi  5686 Apr  1 19:52 chacon_send.cpp
-rw-r--r--  1 pi pi  6405 Mar  8 18:59 wiringPi.h
```

Le résultat de la compilation est un fichier exécutable « chacon_send ». On verra dans le paragraphe suivant comment l'utiliser.

Programme de commande des prises Phenix:

Pour les prises Phenix, il est possible d'utiliser le code du projet *RCSwitch-Pi (dépôt Github (https://github.com/r10r/rcswitch-pi))*, lui même basé sur le code du projet *RCSwitch* initialement développé pour les micro-contrôleurs Arduino. J'ai, à nouveau, légèrement modifié le code proposé afin de permettre de choisir, lors de l'appel du programme, quel port GPIO doit être utilisé. Je vous propose donc une petite archive zip contenant les modifications, le tout prêt à compiler.

1. Télécharger le petit package que j'ai préparé ici

(https://www.dropbox.com/s/236v1hyx4icz2qm/phenix_send.zip?dl=0) puis copiez le sur le RPi
2. Dézipper l'archive et placez vous dans le répertoire résultant
3. Lancer la commande « make all ». **NB**: l'installation de WiringPi ci-avant est indispensable

Ce qui donne la séquence suivante, en ligne de commande:

```
pi@domopi ~ $ wget -q -O phenix_send.zip https://www.dropbox.com/s/236v1hyx4icz2qm/phenix_send.zip
pi@domopi ~ $ unzip phenix_send.zip
Archive:  phenix_send.zip
   creating: phenix_send/
  inflating: phenix_send/Makefile
  inflating: phenix_send/phenix_send.cpp
  inflating: phenix_send/RCSwitch.cpp
  inflating: phenix_send/RCSwitch.h

pi@domopi ~ $ cd phenix_send/

pi@domopi ~/phenix_send $ make all
g++    -c -o RCSwitch.o RCSwitch.cpp
RCSwitch.cpp: In member function 'char* RCSwitch::getCodeWordB(int, int, boolean)':
RCSwitch.cpp:194:61: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:194:61: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:194:61: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:194:61: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:194:61: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp: In member function 'char* RCSwitch::getCodeWordA(char*, int, boolean)':
RCSwitch.cpp:229:74: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:229:74: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:229:74: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:229:74: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:229:74: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
RCSwitch.cpp:229:74: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
g++    -c -o phenix_send.o phenix_send.cpp
g++   RCSwitch.o phenix_send.o -o phenix_send -lwiringPi

pi@domopi ~/phenix_send $ ls -la
total 80
drwxr-xr-x  2 pi pi  4096 Apr  2 09:43 .
drwxr-xr-x 12 pi pi  4096 Apr  2 09:43 ..
-rw-r--r--  1 pi pi   138 Apr  2 09:09 Makefile
-rwxr-xr-x  1 pi pi 16982 Apr  2 09:43 phenix_send
-rw-r--r--  1 pi pi   880 Apr  2 09:12 phenix_send.cpp
-rw-r--r--  1 pi pi  1980 Apr  2 09:43 phenix_send.o
-rw-r--r--  1 pi pi 16706 Aug 12  2012 RCSwitch.cpp
-rw-r--r--  1 pi pi  3838 Aug 12  2012 RCSwitch.h
-rw-r--r--  1 pi pi 12400 Apr  2 09:43 RCSwitch.o
```

Le résultat de la compilation, hormis quelques warning sans conséquence, est un fichier exécutable « phenix_send ». On verra dans le paragraphe suivant comment l'utiliser.

Envoyer les ordres aux prises

A ce stade, l'émetteur RF433 est bien connecté au RPi, et on dispose de deux programmes permettant d'envoyer des ordres à chacun des deux modèles de prises.

Toute prise, quel que soit le modèle et quel que soit le protocole de communication, doit se voir attribuer un numéro qui permet de l'identifier. Lorsqu'une commande est émise sur la fréquence de communication, ce numéro d'identification est envoyé dans la trame, entre autre information, afin que la prise concernée et uniquement elle, réagisse à la commande. C'est en quelque sorte l' « adresse » de la prise.

Prises Chacon DI-O

Pour les prises Chacon DI-O, l'attribution du numéro d'identification de la prise se fait lors du branchement sur le secteur. Celle-ci passe temporairement dans un mode de programmation qui fait que la première trame de commande qu'elle reçoit lui fixe son numéro d'identification.

On utilise le programme « chacon_send » que nous avons préalablement compilé. La commande se présente comme suit:

```
chacon_send <wiringPI pin> <controler_code> <outlet_code> <on|off>
```

- *wiringPi pin* : Il s'agit du numéro wiringPi sur lequel on a branché la broche « DATA » de l'émetteur RF433. On a vu avant qu'il s'agissait du pin physique 11, qui correspond au wiringPi pin **0**, c'est donc ce numéro qu'on spécifiera
- *controler_code* : numéro identification de l'émetteur, d'une longueur de 26 bits. C'est arbitraire dans notre cas, on mettra n'importe quel nombre qui sera ramené à 26 bits
- *outlet_code* : numéro d'identification de la prise. Si une prise vient juste d'être mise sur le secteur, elle prendra ce code comme identifiant et le gardera en mémoire
- *on|off* : commande à envoyer, ON ou OFF pour allumer ou éteindre la prise

Exemple: Branchez une prise Chacon DI-O sur secteur et de suite après, entrez les lignes suivantes le RPi.

```
pi@domopi ~/chacon_send $ sudo ./chacon_send 0 12325261 1 on
pi@domopi ~/chacon_send $ sudo ./chacon_send 0 12325261 1 off
```

Si tout se passe bien, la prise Chacon DI-O devrait s'allumer et s'éteindre🙂

Prises Phenix

Pour les prises Phenix, l'attribution du numéro d'identification se fait par le biais de petits « *switch dip*« , situés dans la prise:

(https://www.dropbox.com/s/g2fz6r4oq6zao66/rf433-outlet-03.jpg?dl=0)

- Les *switchs dip* marqués 1 à 5 servent à identifier l'émetteur auquel la prise répondra
- Les *switchs dip* marqués A à D servent à identifier la prise elle-même. Le switch A étant le bit de poids le plus faible et le switch D le bit de poids le plus fort. Sur la photo: A=0, B=1, C=0, D=0, ce qui donne, en binaire et en remettant les bits dans le sens naturel de lecture : 0010, soit **2** en décimal

On utilise le programme « phenix_send » que nous avons préalablement compilé. La commande se présente comme suit:

```
1 phenix_send <wiringPi pin> <controler_code> <outlet_code> <1|0>
```

- *wiringPi pin* : Il s'agit du numéro wiringPi sur lequel on a branché la broche « DATA » de l'émetteur RF433. On a vu avant qu'il s'agissait du pin physique 11, qui correspond au wiringPi pin **0**
- *controler_code* : numéro identification de l'émetteur tel que définit par les dip switch correspondant. On l'écrit directement sous représentation binaire correspondante
- *outlet_code* : numéro d'identification de la prise tel que définit par les dip switch correspondant. On le spécifie en format décimal (*selon l'explication du calcul ci-avant*)
- *0|1* : commande à envoyer, 0 pour éteindre la prise ou 1 pour allumer

Exemple: Branchez une prise Phenix sur secteur, configurez le dip switch comme sur la photo, puis entrez les lignes suivantes le RPi.

```
1 pi@domopi ~/phenix_send $ sudo ./phenix_send 0 10110 2 1
2 pi@domopi ~/phenix_send $ sudo ./phenix_send 0 10110 2 0
```

Si tout se passe bien, la prise Phenix devrait s'allumer et s'éteindre🙂

Commande par le réseau

On pourrait bien sûr en rester là et créer localement des scripts qui déclenchent les prises de manière autonome et programmée. Mais dans le cadre de mon installation (*voir cet article (https://arno0x0x.wordpress.com/2015/03/17/environnement-de-base/)*), j'ai besoin de pouvoir actionner les prises depuis un autre RPi.

J'avais détaillé dans cet article comment faire communiquer les deux RPi (https://arno0x0x.wordpress.com/2015/03/22/communication-entre-rpi/): le RPi frontal et le DomoPi. Nous avons donc là de nouveaux services domotiques tout désignés à rajouter au script permettant d'y accéder à distance.

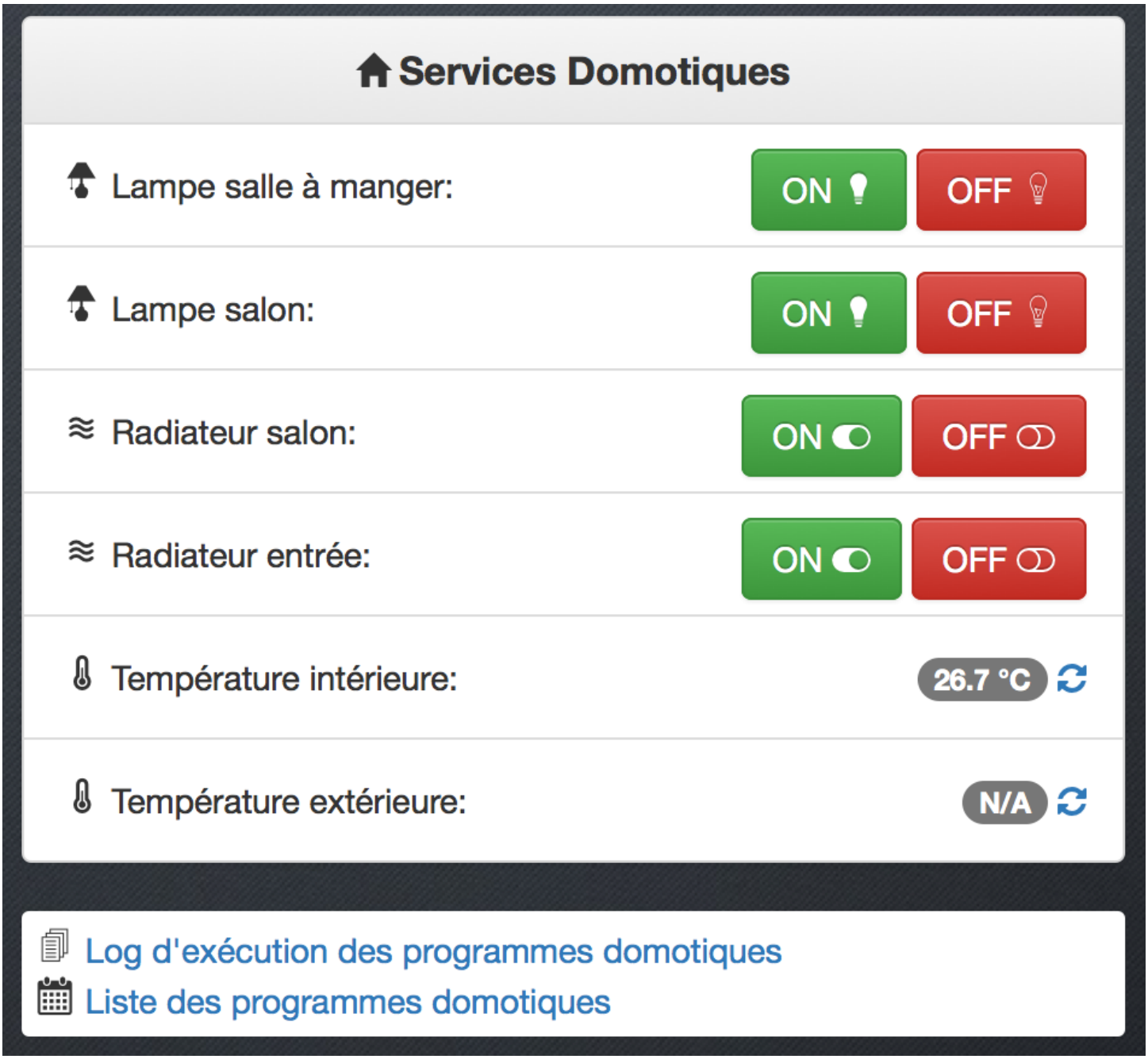
Voici donc une version plus complète, intégrant mes commandes de lampes (*branchées sur des prises Phenix*) et de radiateurs (*branchés sur des prises Chacon DI-O*):

+ expand source

Il y aurait bien sûr moyen d'optimiser ça en ayant qu'une seule commande associée à une variable passée en paramètre indiquant quelle prise on souhaite allumer ou éteindre, mais vu que je n'ai que 4 prises à commander, je vous laisse cette idée comme exercice😉

Tout ceci n'est qu'un début. Il est possible de faire d'autres choses amusantes sur cette base. A titre d'exemple, j'ai défini plusieurs programmes d'allumage/extinction des prises en fonction des jours/heures/minutes, chaque programme étant « Activable / Désactivable » via le même mécanisme de commande à distance.

In fine, dans un prochain article, une fois que j'aurai couvert un exemple de lecture de température, je montrerai un exemple d'interface web basique que j'ai créé permettant de commander tout ça, d'activer/désactiver les programmes automatiques, de vérifier les journaux (*log*) d'exécution de ces programmes :



(https://www.dropbox.com/s/i9xds61556hcwbr/rf433_outlet-06.png?dl=0)

A bientôt !

Vous aimez cet article ? Faites le savoir avec quelques bitcoins !

Domotique, Raspberry Pi domotique, Raspberry Pi

27 thoughts on “Commandez vos prises électriques à distance avec le Raspberry Pi”

1. **TITI**
   6 novembre 2016 / 15 h 39 min
   Bonjour,

   j’ai le même problème que ‘CAPITAINABLOC’ au même endroit j’ai ce meessage d’erreur :

   pi@raspberrypi:~/chacon_send $ g++ -o chacon_send chacon_send.cpp -lwiringPi
   chacon_send.cpp: In function ‘int main(int, char**)’:
   chacon_send.cpp:178:14: error: ‘setuid’ was not declared in this scope
   if (setuid(0))
   ^

   Avez-vous trouvé une solution ?

Et merci à l'auteur pour ses tutoriels !

- Répondre

- **TITI**
  9 novembre 2016 / 15 h 51 min
  Excusez-moi, j'ai vu la réponse plus bas un peu tard:
  il faut ajouter la ligne
  #include unistd.h
  Une fois fait, cela fonctionne, mais la prise, pourtant du même modèle ne bronche pas d'un poil.
  et je ne reçois aucun message d'erreur lors de la requète.

  Lors du 'ls -la' j'obtient un résultat légèrement différent :
  drwxr-xr-x 2 pi pi 4096 Apr 9 20:11 .
  drwxr-xr-x 26 pi pi 4096 Apr 9 20:10 ..
  -rwxr-xr-x 1 pi pi 13177 Apr 9 20:11 chacon_send
  -rw-r–r– 1 pi pi 5686 Apr 6 19:52 chacon_send.cpp
  -rw-r–r– 1 pi pi 6405 Mar 2 18:59 .DS store

  Quelqu'un pourrait m'aider?

  ps:Ce qui est étrange c'est qu'une fois l'émeteur branché la télécommande de la prise ne fonctionne plus qu'à 10 cm de portée?
  Une fois l'émeteur débranché ça re-fonctionne à distance…

  - Répondre

  - **TITI**
    9 novembre 2016 / 20 h 26 min
    j'oubliais 'd'apprendre' la commande à la prise… Maintenant ça fonctionne…!
    Merci pour vos super tutos.

- **TONY**
  28 novembre 2016 / 9 h 29 min
  Bonjour,

  Merci pour ce super tuto!!!
  Est-il possible avec les prises DIO d'avoir le retour de marche ?
  Hier, j'ai dû envoyer trois fois le code ON pour que la prise marche.
  En ayant, le retour d'information je pourrai connaître l'état de la prise à distance.

  Merci par avance.

  - Répondre

2. **GUILLAUME**
   6 novembre 2016 / 1 h 28 min
   Bonjour,

   Elle semble sympa ta petite interface pour contrôler tes équipements domotique … Simple et efficace !
   Tu pourrais en partager le code ? J'ai un backend fonctionnel sur raspberry (chauffage, lumière, météo & co), mais je n'ai aucune créativité pour le design d'interfaces !

Merci,
Guillaume.

- Répondre

3. **CAMPTO**
   25 avril 2016 / 18 h 56 min
   Bonjour

   J'ai suivi votre tuto et vous en remercie.
   Tout fonctionne lorsque je suis en ssh par contre lorsque je lance via une page Php lorsque je lance un elec, il n'y a que le off qui fonctionne (que je mette on ou off d'ailleurs)

   Avez-vous déjà rencontré le problème?
   Merci par avance pour votre aide.

   - Répondre
   - **CAMPTO**
     25 avril 2016 / 20 h 13 min
     Lorsque je lance exec()

     - Répondre
     - **CAMPTO**
       25 avril 2016 / 22 h 50 min
       C'est bon j'ai réussi en passant pas un bash avec argument

4. **RETLAW**
   31 mars 2016 / 23 h 22 min
   Bonjour,

   Un grand merci pour ce tuto qui me permet désormais de commander mes prises Chacon depuis mon raspberry! celà va m'être d'une grosse utilité pour l'intégration de ces dernières à ma domotique!

   Une petite question coté interface web.

   J'ai une carte IPX800 à la maison gérant mes radiateurs, ballon eau chaude, portail, volets roulants, porte garage, box internet, lumière extérieur, alarme, capteurs divers (ouverture, facteur, température, etc etc.)

   je pilote cette dernière via impérihome grâce à des requètes http.

   Est-il possible de gérer les script/commandes ON et OFF de chaque prise via de telles requètes (http)? Je peux passer par une interface web embarquée au sein même du raspberry comme l'exemple proposé, mais il ne me faudrait que 6 prises au total (pas de température puisque déjà gérée par mon IPX)

   Si cela s'avère possible, comment dois-je m'y prendre (je ne m'y connais pas beaucoup en programmation mais je peux mettre les mains dans le cambuis sans problème.

   Merci d'avance pour votre aide!

   Retlaw

   - Répondre
   - **CAMPTO**

25 avril 2016 / 20 h 24 min
Bonjour

Vous pouvez en installant Apache et php5 sur le Pi
Ensuite via php et la commande exec (il faudra modifier le fichier /etc/sudoers pour autoriser le compte www-data à exécuter la commande exec)
Dans la page php, la commande sera :
Exec('Sudo -u www-data repertoirechaconsend/chacon_send 0 12325261 1 on',$taberreur,$coderetour);
pour activer la prise off pour la désactiver

$taberreur est le tableau contenant les erreurs en type string
$coderetour le code de retour si différent de 0 alors erreur

Cordialement

- Répondre
  - **RETLAW**
    27 avril 2016 / 15 h 10 min
    Merci pour votre réponse!

    Je pense avoir compris le principe. Je teste ça d'ici un mois une fois l'emménagement terminé et vous tiens au courant!

5. **CAMILLE DURET**
   28 mars 2016 / 11 h 03 min
   Bonjour CAPITAINABLOC,

   As tu résolu ton problème ?
   J'ai le meme problème que toi quand j'essaye de compiler :

   pi@raspberrypi:~/chacon_send $ g++ -o chacon_send chacon_send.cpp -lwiringPi
   chacon_send.cpp: In function 'int main(int, char**)':
   chacon_send.cpp:178:14: error: 'setuid' was not declared in this scope
   if (setuid(0))
   ^
   Merci beaucoup

   - Répondre
     ARNO0X0X
     28 mars 2016 / 12 h 52 min
     Comme l'a souligné CAPTAINABLOC, c'est peut-être un souci de compilation lié à la dernière version de Raspbian (v8 basée sur la Debian Jessie). Je n'ai eu aucun problème à compiler le script sur mes différents RPi (A ou B et même RPi2A), mais n'ayant pas de RPi tournant sous Raspbian 8, je ne peux pas tester ou vérifier d'où vient le problème…

     - Répondre
       - **CAPITAINABLOC**
         28 mars 2016 / 13 h 06 min
         bonjour,
         comme dit précedemment, il faut ajouter la ligne
         #include unistd.h

la mise en page wordpress enlève les signes « > » et « <", mais il faut utiliser la même mise en page que pour les autre #include du fichier cpp.

- **CAMILLE DURET**
  28 mars 2016 / 14 h 07 min
  Au top , ça fonctionne très bien !

  Merci beaucoup !!!

6. **JEREMY**
   28 février 2016 / 15 h 53 min
   Bonjour,

   J'ai exactement le même problème que capitainabloc. Je me demande si ce n'est pas une incomptibilité avec Raspbian Jessie car ça fonctionne très bien sur mon autre raspberry (qui a une ancienne version de raspbian).

   Quelqu'un sait-il comment on pourrait résoudre ce problème ?

   Merci.

   - Répondre
7. **FRED**
   27 février 2016 / 13 h 26 min
   Bonjour,
   merci pour ce tuto, vraiment impeccable.
   Les commande « chacon_send 0 12325261 1 on » ou « off » marchent très bien, sauf avec une crontab ou seul la commande « off » fonctionne, je n'arrive pas envoyer le « on ».
   Avez vous une idée ?
   Merci.
   fred.

   - Répondre

   ARNO0X0X
   27 février 2016 / 17 h 53 min
   Bonjour, merci pour votre commentaire.

   C'est en effet très étrange. Pouvez-vous montrer le crontab complet en exemple (je pense par exemple à des commandes/accès concurrents à l'émetteur, mais c'est une piste incertaine) ?
   Merci,
   Arno

   - Répondre

   - **FRED**
     28 février 2016 / 12 h 19 min
     Merci pour votre réponse.
     Je viens de voir que le souci ne viens pas du tout de la commande, mais de la façon dont j'insère la ligne dans la crontab.
     Je me suis fais une interface web (php) pour contrôler la crontab. de cette manière le « on » ne fonctionne pas. Mais si j'ouvre et enregistre avec crontab -e, le problème disparait.

Mise en route cafetière du matin
0 6 * * 1-5 /usr/local/bin/chacon_send 0 12325261 1 on
31 11 * * * /usr/local/bin/chacon_send 0 12325261 1 on
33 11 * * * /usr/local/bin/chacon_send 0 12325261 1 off
Arrêt de la cafetière du matin
0 20 * * 1-5 /usr/local/bin/chacon_send 0 12325261 1 off
0 20 * * 7 /usr/local/bin/chacon_send 0 12325261 1 off

ici le on de 11h31 ne fonctionne plus puisque j'ai rentré 11h33 par l'interface web. Mais 11h33 fonctionne bien.

Merci.
fred

8. **CAPITAINABLOC**
   13 octobre 2015 / 9 h 08 min
   Bon, j'ai trouvé!!

   il faut éditer le fichier ~/chacon_send/chacon_send.cpp

   et y ajouter au début:
   #include

   et ca fonctionne!

   merci!

   - Répondre
     - **CAPITAINABLOC**
       13 octobre 2015 / 9 h 10 min
       il a pas pris la commande c'est donc #include unistd.h, avec autour de unistd.h

       - Répondre
         ARNO0X0X
         13 octobre 2015 / 9 h 12 min
         ok, merci d'avoir mis à jour, ça pourra servir à d'autres personnes.

         C'était bien le header unistd.h dont je t'avais parlé. Ce que je ne comprends pas c'est pourquoi ça compile sans problème sur plusieurs de mes RPi sans avoir à rajouter le #include en question.

         - **BELZE88**
           8 novembre 2016 / 23 h 12 min
           Merci, j'ai eu le même problème et ça m'a bien débloqué ! (NB : penser au chevrons autour de unistd.h comme pour les autres include)

9. **CAPITAINABLOCCLEM**
   13 octobre 2015 / 8 h 12 min
   Bonjour,
   merci pour ce tuto.
   J'essaie de le mettre en oeuvre, mais je bloque dès le début sur une erreur lors de la commande:
   g++ -o chacon_send chacon_send.cpp -lwiringPi

j’obtiens l’erreur suivante:
chacon_send.cpp: In function ‘int main(int, char**)’:
chacon_send.cpp:178:14: error: ‘setuid’ was not declared in this scope
if (setuid(0))
^

Aurais-tu une piste pour résoudre ce problème?

Clem

- Répondre

ARNO0X0X
13 octobre 2015 / 8 h 33 min
Salut,

Est-ce sur un Raspberry Pi avec l’install raspbian par défaut ?

Il se pourrait qu’il te manque le header « unistd.h » qui est livré avec la package libc6-dev (sudo apt-get update && sudo apt-get install libc6-dev).

C’est une piste en tous cas🙂
Arno

- Répondre

- **CAPITAINABLOC**
  13 octobre 2015 / 8 h 39 min
  Bonjour, merci de ta réponse.
  je suis sur raspberry2, avec raspbian par défaut.
  ce package est à jour.
  Je suis vraiment newbie là dedans, et j’ai suivi ton tuto à la lettre.

  voilà où ca bloque:

  root@raspberrypi:/home/pi/chacon_send# g++ -o chacon_send chacon_send.cpp -lwiringPi
  chacon_send.cpp: In function ‘int main(int, char**)’:
  chacon_send.cpp:178:14: error: ‘setuid’ was not declared in this scope
  if (setuid(0))
  ^
  root@raspberrypi:/home/pi/chacon_send#

  ce qui est bizarre, c’est que quand tu dézippe, tu indiques avoir cette info:
  inflating: chacon_send/wiringPi.h

  mais je ne l’ai pas, et mon dossier ne contient que chacon_send.cpp et .DS_Store.

  merci pour ton aide

10. Ping : Envoyez des SMS avec votre Raspberry Pi – arno0x0x