



## NOS PROJETS COMPLETS

# CHUCK NORRIS MACHINE

LES BLAGUES  
QUI TUENT !

### CE QU'IL VOUS FAUT

#### Raspbian

Où le trouver :  
[www.raspberrypi.org](http://www.raspberrypi.org)

- Un Raspberry Pi
- Une carte micro SD
- Un afficheur LCD de type Hitachi HD44780 ou similaire
- Une imprimante thermique de type CSN-A2-T ou similaire
- Une plaque de prototypage, un «cobbler» et des fils
- 2 boutons reprenant leurs positions initiales (comme sur une manette de jeu)
- 2 LED (optionnelles ou intégrées aux boutons)

Difficulté :

Voici un projet amusant mêlant connaissance du matériel et programmation. Dans cet article, nous allons voir comment utiliser un afficheur LCD sur un Raspberry Pi, y connecter une imprimante thermique et utiliser une interface homme-machine réduite à sa plus simple expression : deux boutons ! Le but est de réaliser une boîte à blagues «Chuck Norris». Appuyez sur un bouton pour afficher une blague et appuyez sur le deuxième pour l'imprimer !



**V**ous connaissez les blagues Chuck Norris ? Cet acteur d'action des années 70/80 et fervent républicain est connu sur Internet pour être surhumain, voire même immortel. Les petits traits d'humour qui entourent son existence fantasmée sont réunis sous le terme de «Chuck Norris Facts». La préférée de la rédaction ? «Chuck Norris a déjà compté jusqu'à l'infini. Deux fois.» Allez, encore une autre : «Certaines personnes portent un pyjama Superman. Superman porte un pyjama Chuck Norris.». Pour ce projet, il est question d'afficher, une blague Chuck Norris choisie aléatoirement parmi un gisement de Chuck Norris Fact. Selon la qualité de la blague,

l'utilisateur peut (ou non) en demander l'impression. L'interface Homme/machine se fait par l'intermédiaire d'un écran d'affichage LCD 16×2 et de deux switchs momentanés, 1 rouge et 1 vert. En matière d'ergonomie, on reste pas loin du principe disant que : «Une bonne interface ne devrait avoir qu'un seul bouton». Côté impression, le choix de l'imprimante thermique est un excellent compromis coût/rétention de l'information. Et puis, pouvoir repartir avec sa petite blague sur un ticket de caisse, ça claque !

## ➔ LE PORT GPIO

Pour ce projet, nous allons utiliser le port GPIO du Raspberry Pi. On peut voir ces broches comme des prises de courant pour lesquels deux modes sont possibles :

- INPUT (ou mode lecture) : le Raspberry lit l'état (0 ou 3,3V) des broches ;
- OUTPUT (ou mode écriture) : le Raspberry «allume» (de 0V à 3,3V) ou «éteint» (de 3,3V à 0V) les broches.

Notez aussi que lorsque l'on développe un programme, il y a deux façons de se référer aux GPIOs : la numérotation «board» qui correspond à l'emplacement physique et la numérotation BCM qui correspond à la numérotation logique des GPIOs telles qu'elles sont «vues» par le Raspberry. Nous utiliserons ici la seconde (voir notre tableau pour le mapping complet).



**Pour afficher ce que nous percevons comme un caractère, le contrôleur de l'écran doit appliquer des patterns prédefinis et stockés en mémoire. Selon votre modèle, l'emplacement des caractères dans la mémoire peut varier : référez-vous à la fiche technique de votre matériel !**



**STÉPHANE BENNEVAULT**

Stéphane Bennevault a 36 ans, vit dans l'Essonne et est ingénieur d'études et de fabrication au Centre d'Appui aux Systèmes d'Information du Ministère de la Défense. Créatif et motivé par les défis techniques, il a fait du Raspberry son «jouet d'éveil» favori. C'est notamment pour s'initier au langage Python qu'il a réalisé ce projet ludique destiné à diffuser des blagues «Chuck Norris».

Lien :  
<http://raspberryland.noip.me>

LEADER	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
x0000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x0000001 (2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x0000010 (3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x0000011 (4)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00000100 (5)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00000101 (6)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00000110 (7)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00000111 (8)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00001000 (1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00001001 (2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00001010 (3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00001011 (4)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00001100 (5)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00001101 (6)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00001110 (7)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x00001111 (8)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



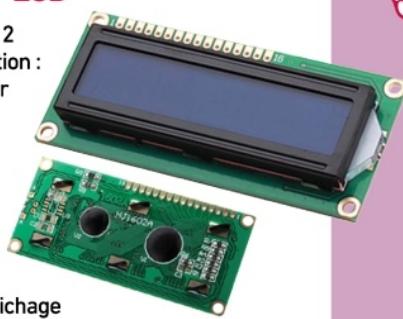
## Votre boîte à blagues

PAS À PAS



### L'afficheur LCD

L'afficheur propose 2 modes de communication : un 4 bits et un 8 bits. Pour simplifier les branchements, nous avons opté pour le premier mode. Chaque ligne de l'écran est composée de 16 matrices de 8x5 pixels. Alors concrètement, comment demander l'affichage d'un caractère sur l'écran depuis un script Python ? Et bien le mécanisme utilisé dans l'API HD44780 est très simple. Lorsque l'on demande l'affichage du caractère **a** par exemple, Python le convertit en binaire soit **0110 0001**. Ensuite, la fonction d'affichage demandera à l'écran d'afficher le pattern stocké à l'adresse **0110 0001** (0110 étant les bits de poids forts identifiant la colonne dans la map CGRAM et 0001 étant les bits de poids faibles identifiant la ligne dans la map CGRAM). Et quel caractère est enregistré à cette adresse ? Le caractère **a** ! Plus précisément, chaque caractère (qu'on identifie par une adresse sur 1 octet) pointe lui-même sur une séquence de 8 octets qui définissent par des 0 et des 1 si les 8 lignes de 5 pixels sont allumées (1) ou non (0). C'est ce que nous allons voir...



### L'imprimante

La page de test nous renseigne sur le jeu de caractères imprimables et les caractéristiques de l'imprimante. Côté transmission de données, il s'agit d'une interface TTL avec une vitesse de 9600 Bauds. Attention, malgré ce que dit la notice, elle n'a voulu fonctionner qu'avec une alimentation 5V-6A. L'imprimante peut envoyer et recevoir des données. Pour cela, elle dispose d'un pin TX et d'un pin RX. Nous n'enverrons pas de données vers le Raspberry depuis l'imprimante, le pin TX de l'imprimante ne sera donc pas connecté. Le pin RX de l'imprimante est connecté à la broche GPIO 14 du Raspberry. Les broches VH et GND sont reliées à l'adaptateur d'alimentation. Il est nécessaire de relier la masse de l'imprimante à celle du Raspberry pour garantir la synchronisation de la transmission de données entre le Raspberry et l'imprimante.



### Le programme fortune

Il est temps d'aborder le côté logiciel du projet. Avant d'entrer dans les détails du code, commençons par aborder la commande **fortune** sur laquelle s'appuie le script Python. La commande **fortune** affiche un message choisi au hasard parmi une source de citations. Il peut s'agir de proverbes, citations de célébrités, de blagues ayant trait à l'informatique ou la programmation. On peut évidemment créer sa propre source. Commençons par installer le programme :

**sudo apt-get install fortune**

Vous disposez maintenant de l'outil, allez testons-le :

**fortune**

L'outil utilise une source par défaut, mais pour notre projet, nous allons le faire utiliser deux fichiers : un fichier texte listant les citations séparées par une ligne avec le caractère % et un fichier .dat généré par le programme strfile pour l'accès aléatoire aux données. Le fichier contenant les blagues Chuck Norris est disponible en téléchargement à cette adresse : <https://goo.gl/waWdcg>. Notez qu'il ne contient pas de caractères accentués, car l'afficheur ne supporte pas nativement leur affichage.

```
Fichier Édition Onglets Aide
Dépaquetage de fortunes-min (1:1.99.1-7) ...
Traitement des actions différences (< triggers >) pour man-db (2.7.0.2-5) ...
Traitement des actions différences (< triggers >) pour man-db (2.7.0.2-5) ...
Paramétrage de fortunes-min (1:1.99.1-7) ...
Traitement des actions différences (< triggers >) pour libc-bin (2.19-18+deb8u7) ...
pi@raspberrypi: ~ fortune
$ fortune
Q: What's buried in Grant's tomb?
A: A corpse.
pi@raspberrypi: ~ fortune
You are destined to become the commandant of the fighting men of the
department of transportation.
pi@raspberrypi: ~ fortune
She was a stowaway in a cargo hold in high places; particularly lonely stewardesses.
pi@raspberrypi: ~ fortune
Your lucky color has faded.
pi@raspberrypi: ~ fortune
You will never come across a piece of your mind, which you can ill afford.
pi@raspberrypi: ~ fortune
You shall be rewarded for a dastardly deed.
pi@raspberrypi: ~ fortune
Your heart is pure, and your mind clear, and your soul devout.
```



## Intégration des blagues

Créer le fichier source :  
**nano chucknorris**

Si vous transférez le fichier source depuis une machine Windows, le caractère de fin de ligne (EOL) doit être modifié par la commande suivante : **sed -i.old 's/\r//' chucknorris**. Un simple copier-coller de l'ensemble du texte à travers l'utilitaire PuTTY réalisera. Une fois le gisement chargé et enregistré, lancez la commande :

**strfile chucknorris**

La commande strfile permet de créer un fichier de données contenant une structure permettant l'accès aléatoire aux chaînes de caractères enregistrées dans le fichier source. Copiez les deux fichiers à l'endroit adéquat :

**sudo cp chucknorris chucknorris.dat /usr/games**

On peut maintenant afficher une blague Chuck Norris :

**fortune chucknorris**

Le système est maintenant prêt. Le script Python pourra lancer la commande système.

```
pi@raspberrypi: ~
cp: impossible de créer le fichier standard « /usr/games/chucknorris »: Permission non accordée
cp: impossible de créer le fichier standard « /usr/games/chucknorris.dat »: Permission non accordée
pi@raspberrypi: ~ sudo cp chucknorris chucknorris.dat /usr/games
pi@raspberrypi: ~ fortune chucknorris
Quand Chuck Norris fait du ski nautique, c'est lui qui tire le bateau.
pi@raspberrypi: ~ fortune chucknorris
Chuck Norris fait tourner le monocle avec un chapeau.
pi@raspberrypi: ~ fortune chucknorris
Quand Chuck Norris fait du ski nautique, c'est lui qui tire le bateau.
pi@raspberrypi: ~ fortune chucknorris
Chuck Norris peut tuer une taupe en l'enterrant vivante.
pi@raspberrypi: ~ fortune chucknorris
Chuck Norris peut se mettre un coup de pied dans la tronche avec une seule main.
pi@raspberrypi: ~ fortune chucknorris
Quand Chuck Norris a perdu sa première dent de lait, la petite souris a déposé l'émail.
pi@raspberrypi: ~ fortune chucknorris
Quand Chuck Norris était petit, le monstre sous son lit avait peur la nuit...
pi@raspberrypi: ~ fortune chucknorris
Quand Chuck Norris a perdu sa première dent de lait, la petite souris a déposé l'émail.
```

## ➔ ATTRIBUTION DES BROCHES DU PORT GPIO

ID	Pin LCD HD44780	Fonction	GPIO Raspberry (BCM)
1	VSS	Masse	GND
2	VDD	Alimentation (+5V)	+5V
3	V0	Contraste de l'affichage 0V = High contrast 5V = Low contrast	GND
4	RS	Register Select RS = 0 : commande RS = 1 : donnée	GPIO 7*
5	RW	Read/Write : permet de lire (High) ou écrire (Low)	GND
6	E	Enable (Clock) cadencement transmission, validation R/W	GPIO 8*
7	D0	Entrée numérique	Not Used
8	D1	Entrée numérique	Not Used
9	D2	Entrée numérique	Not Used
10	D3	Entrée numérique	Not Used
11	D4	Entrée numérique	GPIO 25*
12	D5	Entrée numérique	GPIO 24*
13	D6	Entrée numérique	GPIO 23*
14	D7	Entrée numérique	GPIO 18*
15	A	Anode pour alimenter le rétroéclairage (backlight)	+5V
16	K	Cathode pour alimenter le rétroéclairage (backlight)	GND



[www.raspberrypi-spy.co.uk](http://www.raspberrypi-spy.co.uk)

ID	Pin RED Switch	Fonction	GPIO Raspberry (BCM)
17	Pin +	RED Switch LED +	26*
18	Pin -	RED Switch LED -	GND
19	IN/OUT	Entrée/sortie de l'interrupteur	21*
20	IN/OUT	Entrée/sortie de l'interrupteur	GND

ID	Pin GREEN Switch	Fonction	GPIO Raspberry (BCM)
21	Pin +	GREEN Switch LED +	4
22	Pin -	GREEN Switch LED -	GND
23	IN/OUT	Entrée/sortie de l'interrupteur	12*
24	IN/OUT	Entrée/sortie de l'interrupteur	GND

ID	Pin Thermal Printer	Fonction	GPIO Raspberry (BCM)
25	VH	Alimentation + 5V	Power Adapter
26	GND	Alimentation GND	Power Adapter + GND du Raspberry
27	RX	Receive Data	14**
28	TX	Transmit Data	Not Used
29	DTR	GND	Not Used



# NOS PROJETS COMPLETS



## Réglage du port série

Pour pouvoir utiliser le port série, l'utilisateur (pi dans notre cas) doit disposer des permissions nécessaires.

Vérifiez que pi est membre du groupe dialout qui octroie les droits d'utiliser le port série :

**groups**

**pi adm dialout cdrom sudo audio video plugdev games users input netdev gpio i2c spi**

Si l'utilisateur pi n'est pas membre du groupe dialout, y remédier avec :

**sudo usermod -a -G dialout pi**

Le port série est maintenant ouvert pour l'utilisateur pi, mais encore inaccessible, car utilisé par le système. Pour rappel, le problème à résoudre est le suivant : le Raspberry envoie les données en parallèle (autant de fils que de bits de données) alors qu'on souhaite un envoi en série. Il faut donc transformer ces données parallèles pour les transmettre via une liaison série qui utilise un seul fil pour faire passer tous les bits de données, c'est le rôle du composant UART. Ce dernier est désigné par **/dev/ttyAMA0** sous Raspbian. Par défaut, le système Raspbian utilise ce port série, des modifications sont donc nécessaires pour demander au système de libérer le **/dev/ttyAMA0** et pouvoir ainsi y accéder.

```
pi@raspberrypi:~ $ groups
pi adm lp dialout cdrom sudo audio video plugdev games users input netdev gpio i
2c spi
```



## Libérer l'accès du port série

Le fichier **/proc/cmdline** contient la commande avec les arguments passés au noyau lors du démarrage. Il faudra l'édition :

**sudo nano /boot/cmdline.txt**

Vous devez supprimer toutes les références à **ttyAMA0**, ce qui doit vous donner la ligne suivante : **dwc\_otg.lpm\_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait**

Attention, une erreur de syntaxe dans la commande du fichier **/boot/cmdline.txt** peut empêcher le boot du système. Il faudra ensuite éditer le fichier suivant :

**sudo nano /etc/inittab**

et commenter la ligne :

**T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100**

Il ne reste plus qu'à redémarrer pour appliquer les modifications. Le port **/dev/ttyAMA0** peut être utilisé comme un port série classique sans trafic parasitant le matériel connecté sur le port série. Il faudra enfin désactiver le service Serial getty pour empêcher son lancement au prochain redémarrage :

**sudo systemctl stop serial-getty@ttyAMA0.**

**service**

**sudo systemctl disable serial-getty@ttyAMA0.**

**service**

[Vous trouverez plus de détails concernant l'utilisation du port série sur le site de l'auteur, NDLR]



## La librairie RPi.GPIO

Pour pouvoir configurer, lire et écrire sur les GPIOs du Raspberry depuis un script Python, il faut installer la librairie RPi.GPIO avec la commande :

**sudo apt-get install python-rpi.gpio**

La librairie installée, il suffira de l'importer dans le script Python avec un alias plus court :

**import RPi.GPIO as GPIO**

Avec cet import, l'ensemble des fonctions définies dans la librairie est accessible depuis le script. Le script Python HD44780.py (voir à la fin de l'article), permet de définir les fonctions nécessaires pour utiliser l'écran LCD : la fonction **send\_string** qui permet d'afficher un message sur la ligne 1 ou 2, l'ensemble des fonctions de «bas niveau» qui permettent d'initialiser les GPIOs, d'envoyer un octet de données sur les 4 lignes DATA de l'écran, de pulser la ligne **Enable** pour cadencer la transmission des données ou encore pour initialiser l'afficheur LCD.





## Les modules Python pour l'imprimante

Pour pouvoir utiliser le port série, installer le module Python `python-serial` avec la commande :

`sudo apt-get install python-serial`

Si vous souhaitez imprimer des images, le module `python-imaging-tk` est également nécessaire :

`sudo apt-get install python-imaging-tk`

Il existe aussi un module pour exploiter l'imprimante. Vous pouvez la télécharger en clonant le répertoire git avec la commande :

`git clone git://github.com/luopio/py-thermal-printer.git`

Il est nécessaire d'effectuer quelques modifications/contrôles :

`cd py-thermal-printer`

`nano printer.py`

```
pi@raspberrypi:~ $ git clone git://github.com/luopio/py-thermal-printer.git
Clonage dans 'py-thermal-printer'...
remote: Counting objects: 167, done.
remote: Total 167 (delta 0), reused 0 (delta 0), pack-reused 167
Récception d'objets: 100% (167/167), 32.92 KiB | 0 bytes/s, fait.
Résolution des deltas: 100% (99/99), fait.
Vérification de la connectivité... fait.
```



## Réglages de l'imprimante

Vous pouvez utiliser la recherche dans l'éditeur nano avec **Ctrl+w** pour accéder rapidement aux paramètres **SERIALPORT** et **BAUDRATE** pour vérifier que :

**SERIALPORT = '/dev/ttyAMA0'**

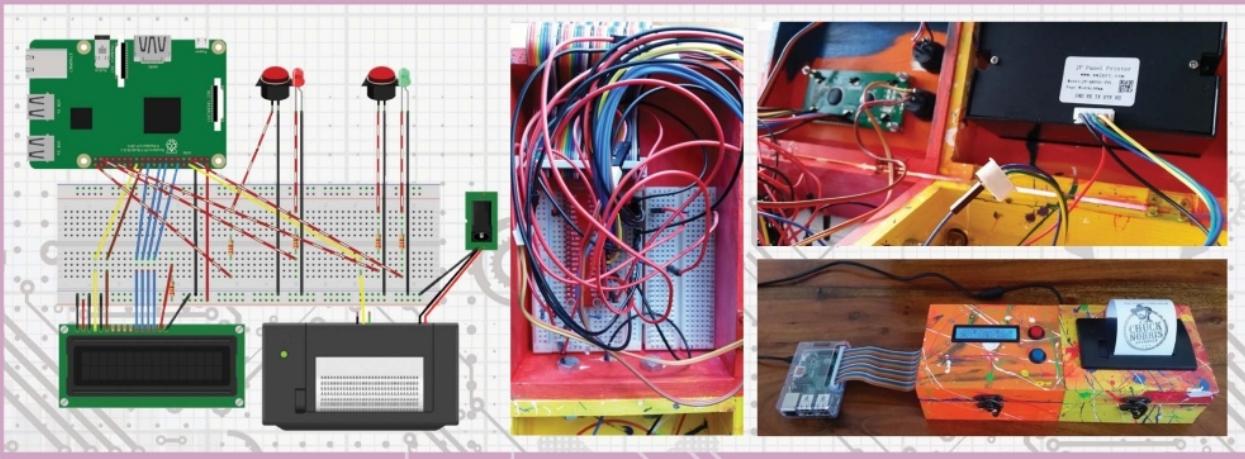
**BAUDRATE = 9600 #peut varier en fonction de l'imprimante**

Il est possible de jouer sur la qualité de l'impression à l'aide de 3 paramètres (**heatTime**, **heatInterval** et **heatingDots**). La librairie `printer.py` vous permettra de jouer avec les deux tailles de police, imprimer des images ou des codes barres. Vous pouvez tester l'envoi de données vers l'imprimante avec la commande :

`./printer.py`

Vous devriez obtenir l'impression d'un ticket démontrant les possibilités de l'imprimante. Comme l'indiquent les premières lignes, le code principal du script Python `chucknorris.py` comporte une série d'imports parmi lesquels évidemment `HD44780.py` et `printer.py`.

*[Vous trouverez plus de détails concernant les réglages de l'imprimante sur le site de l'auteur, NDLR]*





## Chucknorris.py : le code complet

Le code complet de Stéphane Bennevault ainsi que les scripts printer.py et HD44780.py sont disponibles en téléchargement à cette adresse : <https://goo.gl/7GjBfZ>. Si le lien venait à mourir, contactez la rédaction : [raspberry@idpresse.com](mailto:raspberry@idpresse.com)

```
#!/usr/bin/python
# -*- coding:Utf-8 -*-

#####
#      VERSIONNING : chucknorris.py v1.4
#      Utilisation de textwrap et split() pour affichage LCD
par 16 char
#      Thread blink LED_RED_BUTTON
#      Log avec la fonction log()
#      Impression sur ticket
#      2016-03-29 : Ajout du deuxième switch
#      Mise en forme des logs
#####

from HD44780 import *
import os
import math
import textwrap
import threading
import datetime
import printer
import sys

LED_RED = 26
LED_GREEN = 4

# La fonction main doit être située avant if __name__ == '__main__':
def main():
    log("### START SCRIPT PYTHON ###\n")
    init_gpio()
    # On initialise les GPIO en sortie
    GPIO.setup(21, GPIO.IN)          #
    On met le GPIO 21 (PIN 40) en entrée sur lequel est connecté le
    switch
    GPIO.setup(LED_RED, GPIO.OUT, initial=GPIO.LOW)        # On
    met le GPIO 26 (PIN 37) en sortie pour commander la LED du RED
    SWITCH : état initial : False = 0

    GPIO.setup(12, GPIO.IN)          #
    On met le GPIO 12 (PIN 40) en entrée sur lequel est connecté le
    switch
    GPIO.setup(LED_GREEN, GPIO.OUT, initial=GPIO.LOW)       # On
    met le GPIO 4 (PIN 7) en sortie pour commander la LED du GREEN
    SWITCH : état initial : False = 0
    init_lcd()
    # On initialise l'écran HD44780
    load_custom_symbol()           # On
    charge des caractères personnalisées dans l'espace CGRAM
    balayage()
    # On effectue un balayage de l'écran avec des caractères

    while True:
        send_string("PUSH BUTTON FOR",LCD_LINE_1)
        send_string("C.NORRIS FACT ->",LCD_LINE_2)

        while GPIO.input(21) == True:          # Tant que état
        GPIO 21 sur laquelle est connectée le switch
            time.sleep(0.1)                  # reste à 1
            (cad pas à la masse par appui) : on ne fait rien et on attend
            0.1s
            #
            pour ne pas surcharger le CPU

        # Si sortie de while, c'est que le switch a été actionné
        #os.system('systemctl stop serial-getty@ttyAMA0.service')
        log("***** RED SWITCH ACTIVE : BLAGUE DEMANDEE*****")
        log("**** RED SWITCH ACTIVE : BLAGUE DEMANDEE****")
        cmd = os.popen('cat /usr/games/fortune /usr/games/
chucknorris',r)
        joke = cmd.read()                    # Récupération
        dans l'objet cmd du résultat de la commande
        lcd_display(joke)                 # Appel de la
        fonction d'affichage LCD de joke
```

```
ticket_print(joke)                                # Appel de la
fonction d'impression de joke
#Thread_Blink_Green_Button.join()
log("### FIN CYCLE CHUCK NORRIS FACT ###")
log("***** FIN CYCLE CHUCK NORRIS FACT *****\n\n")

#####
#      Fonction pour l'impression sur l'imprimante thermique
#      Timeout à 5 s pour lancer l'impression, Utilisation
d'un Thread pour BLINK LED_GREEN_SWITCH
#####

def ticket_print(texte):
    send_string("PRINT TICKET ?",LCD_LINE_1)
    send_string("YES:GRN - NO:RED",LCD_LINE_2)
    timeout=6
    while GPIO.input(12) == True and GPIO.input(21) == True and
    timeout >= 0:
        time.sleep(0.1)
        timeout -= 0.1
        send_string("PRINT TICKET ? " + str(math.
floor(timeout)),LCD_LINE_1)

    if timeout>0 and GPIO.input(12) == False:          # Si
    on sort du while et timeout non écoulé + GREEN SWITCH actionné
        Thread_Blink_Green_Button = ButtonBlink(LED_GREEN)
        Thread_Blink_Green_Button.start()
        log("### GREEN SWITCH ACTIVE : IMPRESSION DEMANDEE ###")
        send_string("TICKET PRINTING",LCD_LINE_1)
        send_string("PLEASE, WAIT...",LCD_LINE_2)
        p = printer.ThermalPrinter(serialport='/dev/ttyAMA0')
        p.justify("C")
        p.inverse(True)
        p.print_text(" --- CHUCK NORRIS FACT --- ")
        p.linefeed(1)
        p.justify("L")
        p.inverse(False)
        texte_wrapped = textwrap.fill(texte,40)          #
        Découpage par lignes de 16 char max sans couper de mot
        p.font_b(True)
        p.print_text(texte_wrapped)
        p.linefeed(1)
        print_time = datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
        p.justify("R")
        p.print_text(print_time)
        p.linefeed(1)
        p.print_text("SB INDUSTRIES")
        p.linefeed(1)
        p.print_text("stephane.bennevault@gmail.com")

    # Impression de l'image
    from PIL import Image
    i = Image.open('/home/pi/PYTHON/sticker.png')
    data = list(i.getdata())
    w, h = i.size
    p.print_bitmap(data, w, h, True)
    p.linefeed(3)

    time.sleep(5)
    send_string("TICKET PRINTED",LCD_LINE_1)
    send_string("YOU CAN CUT IT",LCD_LINE_2)
    time.sleep(2)
    Thread_Blink_Green_Button.stop()
    log("### FIN IMPRESSION ###")
    log("### FIN BLINK ###")

    elif GPIO.input(21) == False:                      # Si
    RED SWITCH activé
        log("### RED SWITCH ACTIVE : IMPRESSION REFUSEE ###")
        time.sleep(0.5)                               # Pour éviter que l'appui
        pour annuler le print ne lance la blague suivante
```



```

#####
# Fonction pour loguer l'exécution du script à l'écran et
# dans un fichier de log
# Permet de faire des statistiques sur l'utilisation (nb
blague affichées, nb blagues imprimées)
#####

def log(log_msg):
    file = open('/home/pi/chuckylog', 'a')           # Ouverture du
fichier en mode append, comme w, mais à la suite
    log_time = datetime.datetime.now().strftime('%Y-%m-%d
%H:%M:%S')
    log_msg = log_time + « : » + log_msg# Concatène le timestamp
et le message
    if «-noprint» in sys.argv:
        -noprint est fournie en paramètre
        pass
    passe à la suite du code
    else:
        affiche à l'écran le log_msg
        print(log_msg)
    file.write(log_msg + «\n»)                      #
Ecriture du message dans le fichier
    file.flush()                                     #
l'écriture dans le fichier
    file.close()                                     #
Fermature du
fichier

#####
# Classe héritant de la classe Thread pour exécuter un
traitement parallèle
# Au thread principal (BLINK LED_SWITCH)
#####

class ButtonBlink(threading.Thread):
    def __init__(self, switch):
        threading.Thread.__init__(self)
        self.gpio_id = switch
        # Enrichissement avec l'attribut gpio_id
        self.Terminated = False
        # Variable témoin pour fin de Thread

    def run(self):
        try:
            while not self.Terminated:
                GPIO.output(self.gpio_id, not GPIO.input(self.
gpio_id)) # Toggle sur l'état de la GPIO
                time.sleep(1)
                GPIO.output(self.gpio_id, not GPIO.input(self.
gpio_id)) # Toggle sur l'état de la GPIO
                time.sleep(1)
                GPIO.output(self.gpio_id, GPIO.LOW)
                # Evite de laisser la GPIO à HIGH après le thread

            except RuntimeError:
                # Exception ajoutée pour gérer la remontée d'erreur
                # (set GPIO.mode) lorsque Ctrl+C pendant l'exécution d'un thread
                log(«Script interrompu par l'utilisateur pendant
l'exécution d'un thread.»)

                #log(«*** FIN BLINK ***\n»)
        def stop(self):
            self.Terminated = True

    def lcd_display(joke):
        Thread_Blink_Red_Button = ButtonBlink(LED_RED)# Instanciation
de l'objet Thread en passant en paramètre l'id 26 pour la GPIO
        Thread_Blink_Red_Button.start()          #
Lancement du Thread avec la méthode start(), recommandé de ne pas
utiliser run()

        joke = joke.upper()
        # Passage en MAJUSCULES
        log(«*** BLAGUE EN MAJUSCULES : ***»)
        log(joke)
        joke_wrapped = textwrap.fill(joke,16)      # Découpage par
lignes de 16 char max sans couper de mot
        log(«*** BLAGUE EN MAJUSCULES WRAPPEE : ***»)
        log(«\n» + joke_wrapped)
        list_joke_wrapped_16 = joke_wrapped.split(«\n»)  #
Remplissage d'une liste (en supprimant le char séparateur)

        if len(list_joke_wrapped_16)%2 ==1:          # Si
le nombre d'éléments dans la liste est impair

```

```

list_joke_wrapped_16.append(« »)          #
Ajout d'une chaîne vide de 16 char pour LCD_LINE_2
log(«*** BLAGUE EN MAJUSCULES WRAPPEE ET COMPLETEE AVEC
UNE LIGNE D'ESPACES : ***»)
log(str(list_joke_wrapped_16))

for (i, subjoke) in enumerate(list_joke_wrapped_16):  #
enumerate retourne l'index et sa valeur de chacun des items
    if i%2 == 0:
        send_string(subjoke, LCD_LINE_1)
    else:
        send_string(subjoke, LCD_LINE_2)
        time.sleep(3)

Thread_Blink_Red_Button.stop()
#del Thread_Blink_Red_Button : inutile puisque l'objet sera
déinitialisé à la fin de la fonction

log(«*** FIN AFFICHAGE BLAGUE SUR LCD ***»)

#####
# Balayage de l'écran
#####

# Affichage de caractères par balayage
def balayage():
    send_1byte(0x00, LCD_CMD)          #On se place sur la ligne
1

    for i in range (16):
        send_1byte(0x00,LCD_CHR)       #On affiche la caractère
situé à l'adresse 0x00 dans la CGRAM
        time.sleep(0.05)

    send_1byte(0x00, LCD_CMD)          #On se place sur la ligne
2
    for i in range (LCD_WIDTH):
        send_1byte(0x04,LCD_CHR)
        time.sleep(0.05)

#####
# Custom Character Generation
#####

# Enregistre des caractères dans les 8 premiers octets de la
CGRAM
def load_custom_symbol():
    symbols = [
définition des 5 lignes de pixels pour le caractères
[0x4, 0x2, 0xe, 0x1, 0xf, 0x11, 0xf, 0x0],      # à
[0x4, 0x8, 0xe, 0x11, 0x1f, 0x10, 0xe, 0x0],      # é
[0x4, 0x2, 0xe, 0x11, 0x1f, 0x10, 0xe, 0x0],      # è
[0x1b, 0x2, 0xa, 0x4, 0x11, 0xe, 0x0], # smiley
[0x1b, 0x0, 0xa, 0x4, 0x11, 0xe, 0x0], # TODO
[0x1b, 0x0, 0xa, 0x4, 0x11, 0xe, 0x0], # TODO
[0x1b, 0x0, 0xa, 0x4, 0x11, 0xe, 0x0], # TODO
[0x1b, 0x0, 0xa, 0x4, 0x11, 0xe, 0x0], # smiley
]
#smiley = [0x1b,0x0,0xa,0x0,0x4,0x11,0xe,0x0]
# Pour écrire un char en CGRAM, il faut envoyé la commande
0x40 +
    for i in range (len(symbols)):
        cmd = LOADSYMBOL + (i<<3) # Permet de faire un
décalage de 3 bits vers la gauche pour avoir l'adresse du premier
emplacement dans la CGRAM
        send_1byte(cmd, LCD_CMD)
        for octet in (symbols[i]):
            send_1byte(octet)

#####
# Main
#####

if __name__ == '__main__':          # Si le script
est lancé depuis ce fichier, lancement de la fonction main()
try:
    # sinon, c'est
    que le code du fichier chucknorris.py a été importé dans un autre
    script
    main()
except KeyboardInterrupt:          # Si une
interruption au clavier est reçue
    pass
    # On
passe, ce qui permet de ne pas remonter l'erreur à l'écran
    finally:
        send_1byte(CLEARDISPLAY, LCD_CMD)      # Envoi la
commande CLEARDISPLAY (0x01)
        send_string(«Goodbye !», LCD_LINE_1)
        GPIO.cleanup()

```



# SERVOMOTEUR : CONTRÔLE MULTIPLE

Nous l'avons vu dans notre précédent numéro, le Raspberry Pi n'est pas très à l'aise lorsqu'il s'agit de contrôler un servo. Pour l'aider, les cartes de marque Adafruit sont parfaites et peu onéreuses. Nous avons opté pour un modèle 16 canaux à moins de 20 € mais libre à vous de choisir une autre solution ou même d'utiliser une carte Arduino...

## CE QU'IL VOUS FAUT

### Raspbian

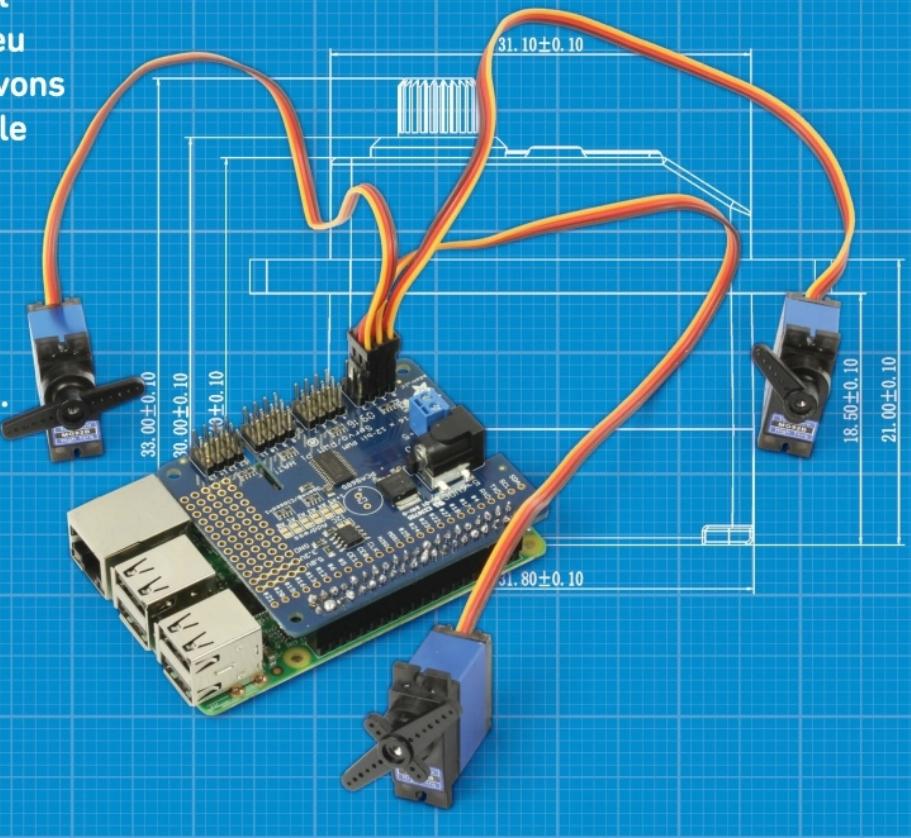
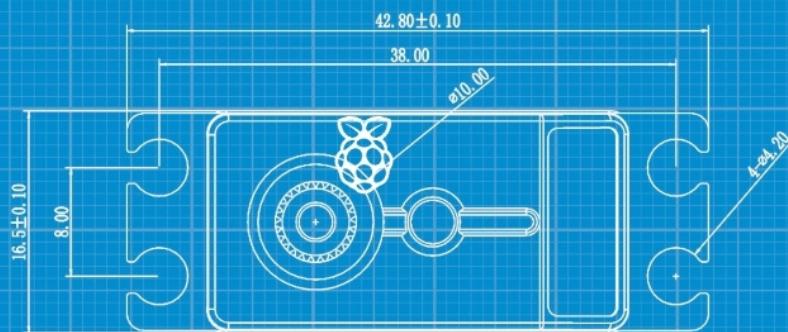
Où le trouver ? :  
[www.raspberrypi.org](http://www.raspberrypi.org)

- Un Raspberry Pi
- Un HAT Adafruit 16 canaux ou équivalent (<https://goo.gl/P2pZEH>)
- Un servomoteur ou plus

Difficulté :

### Si vous avez raté le début...

Si vous avez raté notre premier numéro, retrouvez notre premier article sur le fonctionnement du servomoteur à cette adresse : <https://goo.gl/8p1GHo>. Et si le lien est cassé, faites la demande ici : [raspberry@idpresse.com](mailto:raspberry@idpresse.com)





**P**our contrôler plusieurs servomoteurs, le Raspberry Pi a besoin d'un peu d'aide. En effet ce dernier a un peu de mal à envoyer les pulsations électriques très précises qui permettent de bouger les servos. Il n'est tout simplement pas fait pour cela. Pour ce faire, nous avons donc besoin d'un contrôleur PWM (Pulse Width Modulation ou modulation de largeur d'impulsions) qui va se charger de cette tâche (voir notre numéro précédent). Il est possible d'utiliser une carte Arduino mais pour éviter des branchements un peu compliqués, nous avons opté pour un HAT de marque Adafruit. Cette société, basée à New York est pionnière dans ce domaine et propose des tutoriels en complément de ses produits (nous nous en sommes d'ailleurs librement inspirés pour permettre aux non-anglophones de pourvoir se débrouiller). Il faudra souder les différentes parties avec un fer relativement fin (nous en avons trouvé un à moins de 18 €) puis connecter le HAT comme un Lego sur le port GPIO du Raspberry Pi. Nous verrons comment télécharger les programmes nécessaires, activer le mode de

communication I²P et vérifier que tout est opérationnel pour la suite de notre article.

### LA CARTE ADAFRUIT FONCTIONNE AVEC N'IMPORTE QUEL SERVO FONCTIONNANT AVEC UNE TENSION DE 5V ET ACCEPTANT UN SIGNAL LOGIQUE DE 3,3V

Voici un bras mécanique articulé grâce à des servomoteurs. C'est exactement le genre de projet que vous pourrez réaliser avec notre tutoriel...

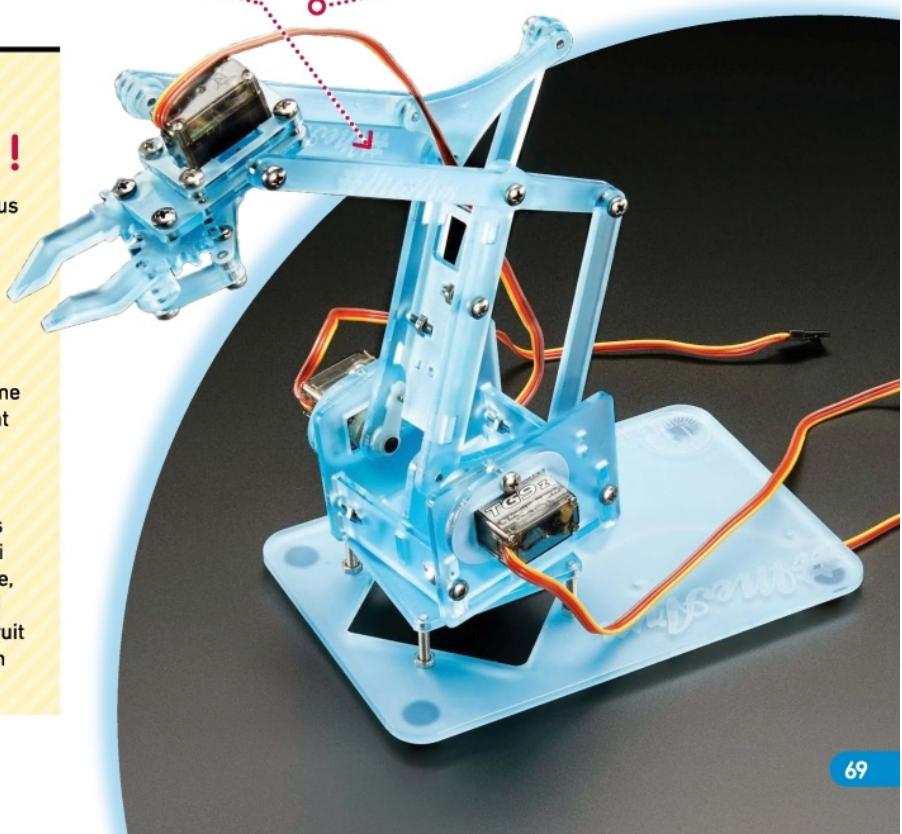
#### LEXIQUE

\***BUS I²C OU I2C :** C'est une méthode de communication inventée par Philips permettant de relier un processeur à des circuits domestiques : télécommandes, horloges, circuits de domotique, etc. Ici, ce bus servira à contrôler les servos.

\***HAT :** «Hardware Attached on Top» ou «matériel attaché sur le dessus» en français. Il s'agit de cartes électroniques additionnelles permettant d'ajouter des fonctionnalités à votre Raspberry Pi. C'est amusant puisque «hat» veut aussi dire «chapeau».

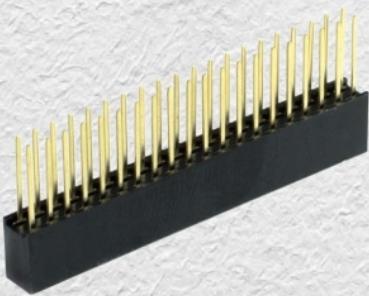
### ATTENTION AUX SERVOS À UTILISER !

Dans nos illustrations vous pouvez voir que nous avons connecté 3 servos MG92B de marque Tower Pro vendus aussi par Adafruit. Or, ces derniers ne fonctionnent pas avec les cartes Adafruit alors même que ces derniers figurent au catalogue. Nous avons prévenu la société - qui nous a d'ailleurs remercié - de leur avoir signalé le problème. En effet, les MG92B (comme apparemment toutes les séries MG) nécessitent un signal de 5V alors que la carte ne permet d'émettre que du 3,3V. Aucun problème par la suite avec nos SG92R... Attention, il ne faut pas confondre ce voltage qui permet de donner des ordres avec le voltage de fonctionnement (celui qui permet de faire mouvoir le servo). De même, vous devrez utiliser des modèles de servos qui autorisent un signal analogique. La carte Adafruit est compatible avec les servos standards ou en rotation continue.





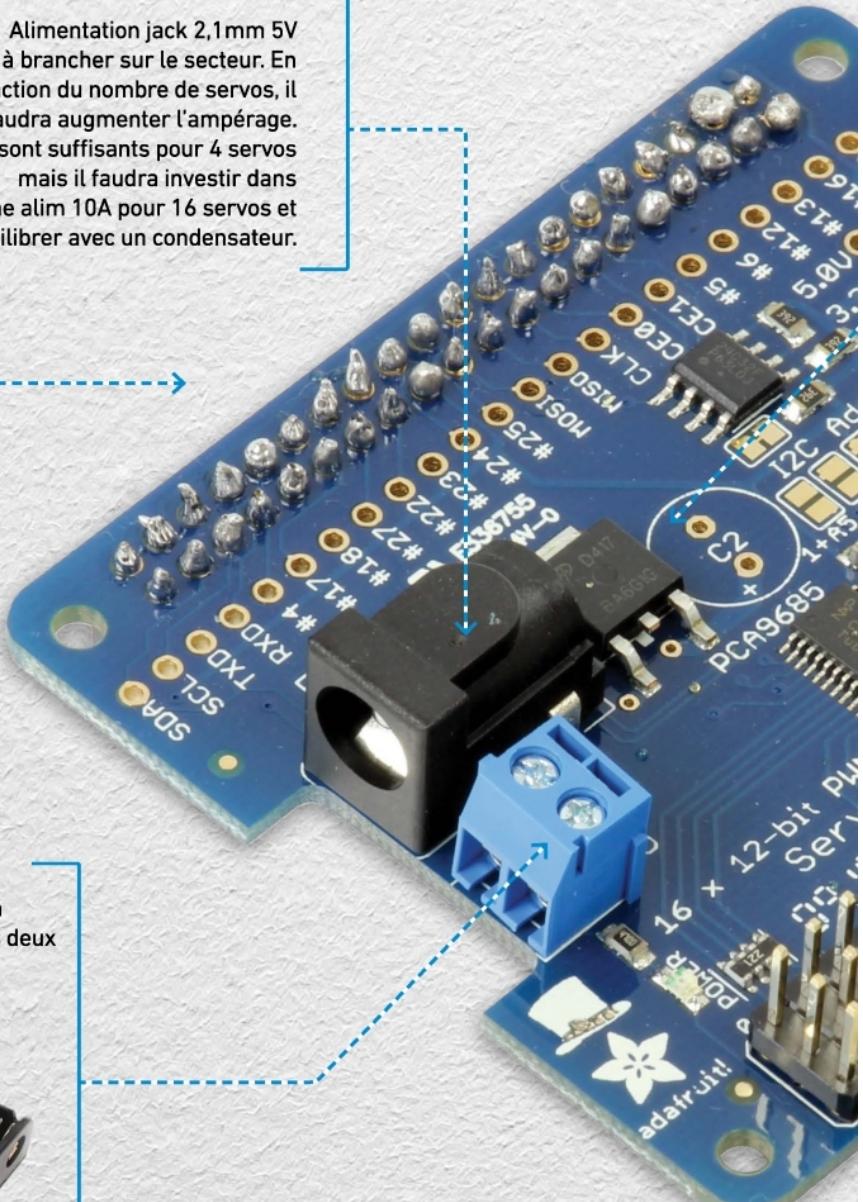
# Contrôleur PWM



C'est sur ce connecteur à souder (il fait dos à la carte) que vous devrez connecter le HAT sur le port GPIO de votre Raspberry Pi

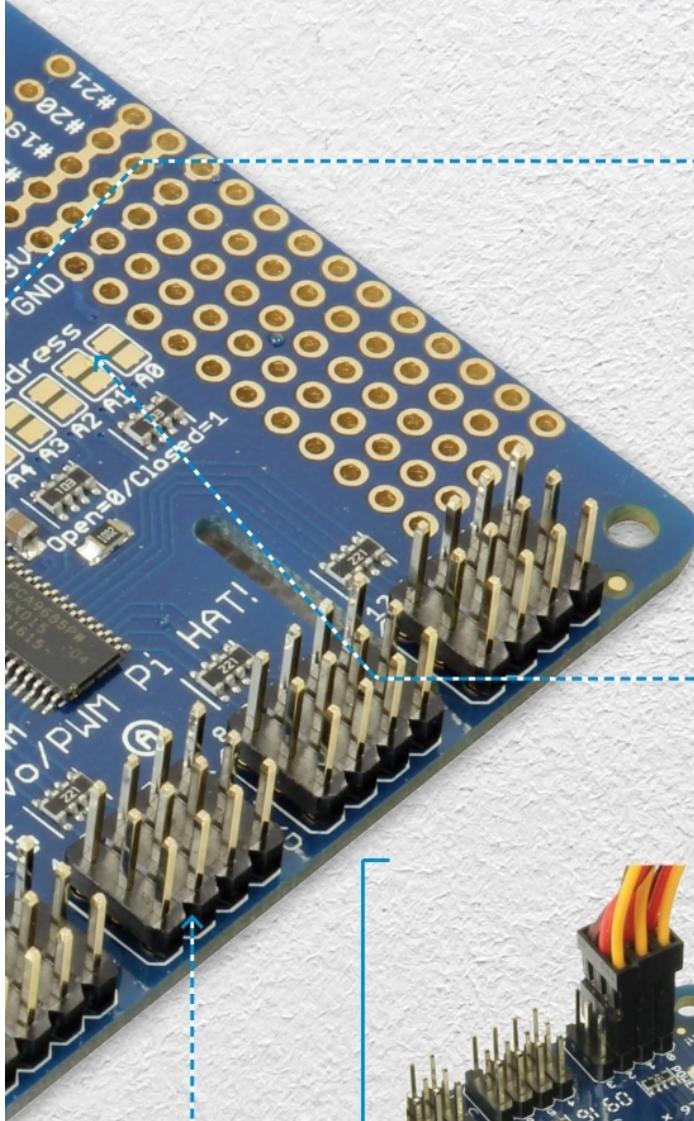
Alimentation jack 2,1mm 5V à brancher sur le secteur. En fonction du nombre de servos, il faudra augmenter l'ampérage. 2A sont suffisants pour 4 servos mais il faudra investir dans une alim 10A pour 16 servos et équilibrer avec un condensateur.

Cosse d'alimentation secondaire pour brancher un boîtier de piles AA par exemple. Ne pas utiliser les deux alimentations simultanément !





# Adafruit

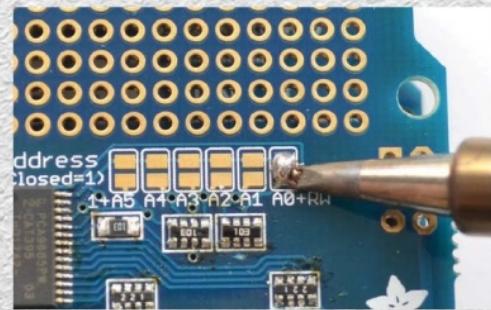
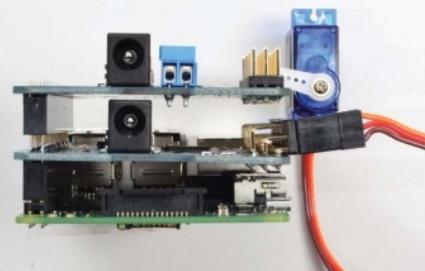


16 broches pour brancher les servos numérotés de 0 à 15. La masse est en bas, le + au milieu et le connecteur pour le signal PWM est sur la rangée du haut.

Cet emplacement est réservé à un condensateur, un composant électronique permettant de stocker une charge électrique. Cela a une utilité lorsque vous voulez faire fonctionner beaucoup de servos simultanément ou que votre alimentation n'est pas assez puissante. La formule simplifiée est  $n \times 100\mu F$  où  $n$  correspond au nombre de servos. Pour 8 servos, un condensateur de  $800\mu F$  ferait l'affaire.



Comme il est possible d'empiler plusieurs cartes pour contrôler plus de 16 servos, vous devrez alors «expliquer» l'ordre des cartes au Raspberry Pi en soudant certains cavaliers entre eux. Si vous n'avez qu'une carte, cette étape n'est pas nécessaire.





# Installation et paramétrage du HAT

PAS À PAS



## 1 Installer la communication I2C

Après avoir soigneusement soudé les différentes parties de votre carte, branchez-la à la fois sur le port GPIO de votre Raspberry Pi et sur le secteur. Lancez le Raspberry Pi sous Raspbian et installez ces programmes :

**sudo apt-get install python-smbus  
sudo apt-get install i2c-tools**

Le deuxième permet de détecter les périphériques compatibles comme notre HAT.



```
pi@raspberrypi: ~
pi@raspberrypi: ~ sudo apt-get install python-smbus
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Fait
python-smbus est déjà la plus récente version disponible.

Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires

libboost-filesystem1.55.0 libboost-program-options1.55.0
libboost-repex1.55.0 libcoward1 libjs-prettify liblomm3.7 libqscintilla2-11
libqscintilla2-11 libqscintilla2-11 libqscintilla2-11 libqscintilla2-11 libtktk5 rtkit9.1 ruby1.9.1
ruby1.9.1-dev ruby1.9.1-examples ruby1.9.1-full ruby1.9.3 supercollider
superCollider-common supercollider-ide supercollider-language

Veuillez utiliser « apt-get autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 56 non mis à jour.

pi@raspberrypi: ~ sudo apt-get update
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Fait
I2C-tools est déjà la plus récente version disponible.

Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires

libboost-filesystem1.55.0 libboost-program-options1.55.0
libboost-repex1.55.0 libcoward1 libjs-prettify liblomm3.7 libqscintilla2-11
libqscintilla2-11 libqscintilla2-11 libqscintilla2-11 libqscintilla2-11 libtktk5 rtkit9.1 ruby1.9.1
ruby1.9.1-dev ruby1.9.1-examples ruby1.9.1-full ruby1.9.3 supercollider
superCollider-common supercollider-ide supercollider-language
```



## 2 I2C au démarrage

Nous allons maintenant faire en sorte de charger automatiquement au démarrage le support du protocole I2P dans le noyau Linux et dans le SoC. Il est possible de le faire depuis Raspi-config mais il est préférable de le faire "à la main". Tapez :

**sudo nano /etc/modules**

Ajoutez ces lignes à la fin de ce fichier :

**i2c-bcm2708**

**i2c-dev**

Faites **Ctrl + X** pour sortir, puis **O** (ou **Y**) pour valider et **Entrée**.

```
pi@raspberrypi: ~
pi@raspberrypi: ~ Fichier : /etc/modules
GNU nano 2.2.6 Fichier : /etc/modules

/etc/modules: kernel modules to load at boot time.

# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

#snd-bcm2835
i2c-bcm2708
i2c-dev
```



## 3 I2C dans le SoC

Ce n'est pas le cas de toutes les distributions, mais il faudra peut-être changer le fichier **raspi-blacklist.conf**. Faites :

**sudo nano /etc/modprobe.d/raspi-blacklist.conf**

Commentez les lignes suivantes comme ceci si elles apparaissent :

**#blacklist spi-bcm2708**

**#blacklist i2c-bcm2708**

Il faudra ensuite ajouter quelques lignes dans le fichier config.

Faites :

**sudo nano /boot/config.txt**

Ajoutez :

**dtparam=i2c1=on**

**dtparam=i2c\_arm=on**

Redémarrez enfin avec :

**sudo reboot**

```
pi@raspberrypi: ~
pi@raspberrypi: ~ sudo nano /boot/config.txt
GNU nano 2.2.6 Fichier : /boot/config.txt

# Uncomment some or all of these to enable the optional hardware interfaces
dtparam=i2c_arm=on
dtparam=i2s=on
dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on
program_usb_boot_mode=1

dtparam=i2c1=on
dtparam=i2c_arm=on
```





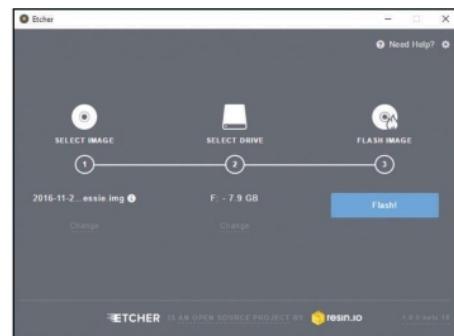
# LE COIN DES ASTUCES

Comme nous vous l'avions promis la dernière fois, voici nos réponses aux problèmes que vous avez pu rencontrer lors de vos pérégrinations avec la framboise. N'hésitez pas à poser vos questions, demander de l'aide ou même proposer vos propres astuces à [raspberry@idpresse.com...](mailto:raspberry@idpresse.com)

## #01 JE NE SUIS PAS SOUS WINDOWS ET JE SUIS EMBÊTÉ POUR PLACER LES IMAGES DES DIFFÉRENTS OS

Dans le numéro 1 de l'Officiel PC nous vous avions expliqué comment mettre une image de Raspbian sur une carte SD avec Windows (Win32 Disk Imager) et sous Linux en lignes de commande. Mais si vous voulez une solution plus simple ou si vous avez un Mac, le logiciel Etcher est fait pour vous. Disponible sur tous les principaux OS, il suffit de choisir l'image, la lettre de lecteur et c'est terminé ! Etcher permet aussi de bien vérifier l'intégrité des données avant d'utiliser votre carte SD.

Lien : <https://etcher.io>



## #02 J'AI UN RASPBERRY PI DONNÉ PAR UN AMI, MAIS JE NE SAIS PAS EXACTEMENT DE QUEL MODÈLE IL S'AGIT MÊME EN REGARDANT LES IMAGES SUR GOOGLE...

Pour connaître le type de votre Raspberry Pi, ouvrez un terminal et tapez :

**cat /proc/cpuinfo**

Vous verrez différentes informations concernant les différents coeurs de votre processeur, mais allez tout en bas à la ligne Revision.

Pour notre machine, nous voyons : **a02082**.

```
processor      : 1
model name   : ARMv7 Processor rev 4 (v7l)
BogoMIPS     : 38.40
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer: 0x41
CPU architecture: 7
CPU variant   : 0x0
CPU part      : 0xd03
CPU revision  : 4

processor      : 2
model name   : ARMv7 Processor rev 4 (v7l)
BogoMIPS     : 38.40
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer: 0x41
CPU architecture: 7
CPU variant   : 0x0
CPU part      : 0xd03
CPU revision  : 4

processor      : 3
model name   : ARMv7 Processor rev 4 (v7l)
BogoMIPS     : 38.40
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer: 0x41
CPU architecture: 7
CPU variant   : 0x0
CPU part      : 0xd03
CPU revision  : 4

Hardware      : BCM2709
Revision     : a02082
Serial       : 000000004bc013a6
pi@raspberrypi:~ $
```

Il s'agit en fait d'un code que vous pourrez "déchiffrer" en suivant notre lien un peu plus bas (regardez le tableau). Dans notre cas, il s'agit d'un Raspberry Pi 3 Model B (1Gb) commercialisé à partir de début 2016 avec une PCB v1.2 fabriquée par Sony. Nous savions presque tout ça déjà, mais pour les Raspberry Pi plus vieux ou ceux qui vous ont été donnés, vous saurez exactement quel type de Raspberry vous avez entre les mains. Peut-être aurez-vous la chance de tomber sur une carte rare que vous pourrez vendre sur eBay. Certains collectionneurs sont friands de ce genre de raretés.

Lien : [http://elinux.org/RPi\\_HardwareHistory](http://elinux.org/RPi_HardwareHistory)



### #03 J'AI BIEN AIMÉ VOTRE ARTICLE SUR LE SENSEHAT DANS VOTRE PREMIER NUMÉRO, MAIS N'EXISTE-T-IL PAS UN ÉMULATEUR POUR ESSAYER DES SCRIPTS SANS DÉPENSER 40 € ?

On peut dire que votre question arrive à point nommé puisque depuis peu, Raspbian

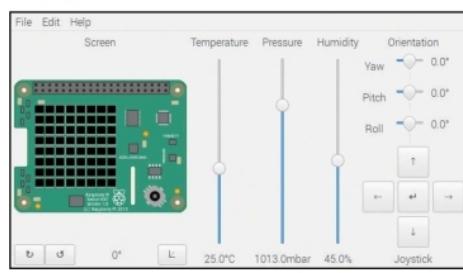
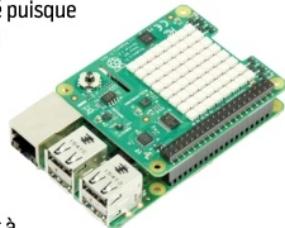
Jessie embarque un émulateur graphique de Sense HAT dans la partie **Programmation** du menu principal

(la framboise en haut à gauche). Pour nos nouveaux

lecteurs, rappelons d'abord que le Sense HAT est un petit module qui va se connecter au port GPIO de votre Raspberry Pi. En plus d'être équipé d'un afficheur 8x8 pixels, ce dernier embarque un joystick et pas moins de 6 capteurs permettant d'interagir avec le monde extérieur (humidité, température, pression, gyromètre, accéléromètre et magnétomètre). Sur cet émulateur, vous disposez alors de curseurs simulant les capteurs et de boutons pour contrôler le joystick.

Vous n'avez pas non plus de Raspberry pi pour l'essayer? Pas de problème, puisqu'il est possible de tester la bête depuis votre navigateur ici :

<https://trinket.io/sense-hat>



### #04 J'AI REPÉRÉ UN OUTIL GÉNIAL POUR FAIRE DES PROJETS DE CONSTRUCTION EN LEGO. IL FAUDRAIT EN PARLER À VOS LECTEURS !

Bien sûr! Après avoir mis à jour votre version de Raspbian et les paquets, faites :

**sudo apt-get install leocad**

Dans la rubrique Éducation, vous trouverez **LeoCAD** mais vous voudriez peut-être ajouter des pièces de Lego pour en avoir plus que celles proposées. Allez sur [www.ldraw.org](http://www.ldraw.org) et dans **Download**, cliquez sur **Parts download** puis choisissez **complete.zip** (34 Mo).

Faites ensuite :

**cd Downloads**

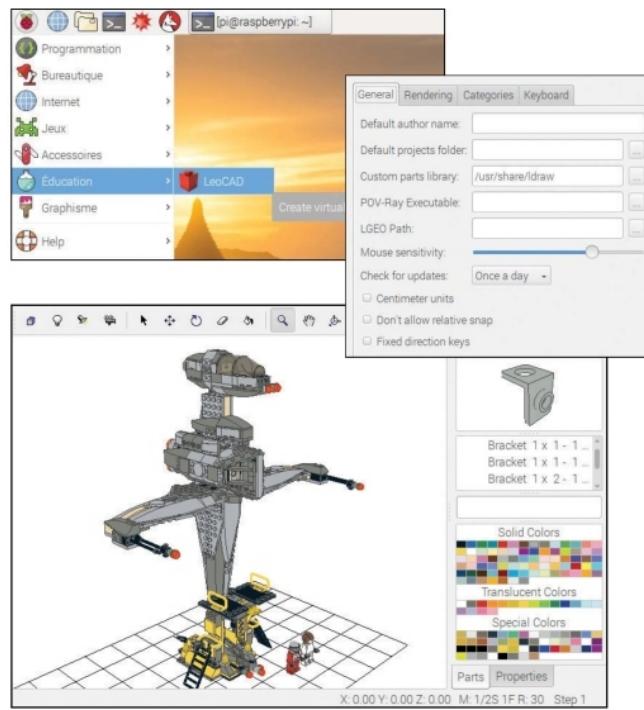
**sudo cp complete.zip /usr/share**

**cd /usr/share**

**sudo unzip complete.zip**

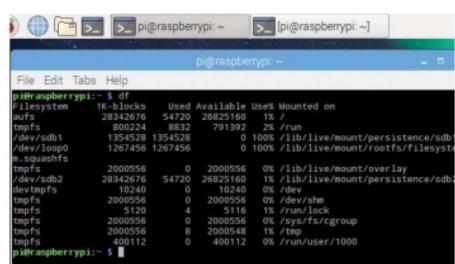
Cela aura pour effet de transférer le fichier et le décompresser à l'endroit adéquat. Ouvrez LeoCAD et dans **View**, allez dans

**Preferences** puis indiquez le chemin vers vos nouveaux Lego (**/usr/share/ldraw**) dans **Custom parts library**. Redémarrez LeoCAD et amusez-vous! Sur le site [www.ldraw.org](http://www.ldraw.org) vous trouverez des modèles à importer comme ce superbe B-Wing. Pour disposer d'une interface en français ou profiter de la dernière version, vous pouvez faire un tour sur le site Framboise314 : <https://goo.gl/mT3H00>



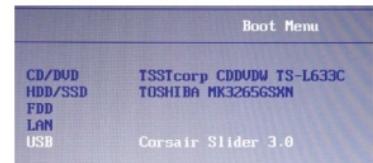
## #05 JE SUIS CURIEUX D'ESSAYER RASPBIAN, MAIS JE N'AI PAS DE RASPBERRY PI

Si le couple Raspbian/Raspberry Pi fonctionne si bien c'est bien sûr à cause de la puissance de la framboise, mais aussi parce que la distribution Linux Raspbian est bien conçue et ne contient rien de superflu. Pourquoi ne pas l'utiliser sur un ordinateur de bureau ou portable ? Tout simplement parce que le Raspberry Pi est une machine à base de processeur ARM et que les PC sont des machines x86 : deux architectures différentes donc. Or la Fondation Raspberry a eu la bonne idée de transposer Raspbian sur architecture x86 pour en profiter sous Mac ou PC. Certes il existe des distributions pour de vieux PC à configuration modeste (Handy Linux, Toutou Linux, etc.),



mais si vous êtes un habitué de Raspbian, pourquoi ne pas tenter l'expérience ? Il suffit de graver l'image sur un disque ou de la placer avec un logiciel comme Etcher (voir notre astuce n°1) puis de booter dessus au démarrage. Nous avons donc réussi à faire fonctionner Raspbian sur un PC équipé d'un processeur AMD vieux de 9 ans avec 512 Mo de RAM. Attention l'interface est en anglais et le système bugue parfois. Même s'il est possible de faire fonctionner le système en mode "persistant" pour garder les modifications et vos fichiers sur la clé USB, on vous dira comme installer l'OS sur un disque dur sur le site Framboise314 : <https://goo.gl/IqN79u>.

Lien : <https://goo.gl/MOXsey>



## #06 UTILISATEUR DE WINDOWS DEPUIS LA VERSION 95, J'AI APPRÉCIÉ VOTRE DERNIÈRE LISTE DE COMMANDES POUR LINUX CAR, À 62 ANS, JE SUIS DÉBUTANT AVEC CET OS. AURIEZ-VOUS D'AUTRES ASTUCES SOUS LE COUDE POUR AIDER UN VÉTÉRAN QUI GALÈRE COMME BEN HUR EN MER ÉGÉE ?

La métaphore est amusante. Notez que c'est en Mer Égée que Ben Hur a finalement收回ré la liberté ! Vous galéiez en fait depuis 1995 sans le savoir ! Pour notre "prince de Judée" (et les autres), voici la suite de nos "commandes à garder en tête" ...

**mkdir vidéo :**

Va créer un dossier **vidéo** dans le répertoire en cours.

**mv vidéo vidéos :**

Vous allez renommer le fichier ou le dossier **vidéo** en **vidéos**.

**mv vidéo ~/Desktop :**

Déplace le fichier vidéo du dossier en cours vers le dossier **~/Desktop** sans le renommer.

**cp essai1 essai2 :**

Pour faire une copie exacte du fichier **essai1** et l'appeler **essai2**. Notez que quand vous utilisez **mv** le fichier/dossier source disparaît tandis que si vous utilisez **cp** le fichier/dossier se copie sans être supprimé.

**sudo apt-get update :**

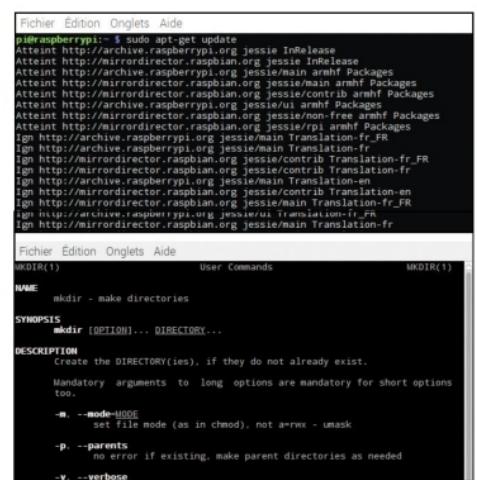
C'est la commande qu'il faut taper avant une installation. Elle permet de mettre à jour la liste des paquets disponibles.

**sudo apt-get upgrade :**

Va mettre à jour les paquets déjà installés pour profiter des patchs de sécurité et de diverses améliorations.

**man mkdir :**

Comme nous l'avons vu la dernière fois, **man** permet d'afficher des explications détaillées de la commande **mkdir**. Cela fonctionne aussi avec **mv**, **cp**, **apt-get**, etc. Faites **q** pour quitter.

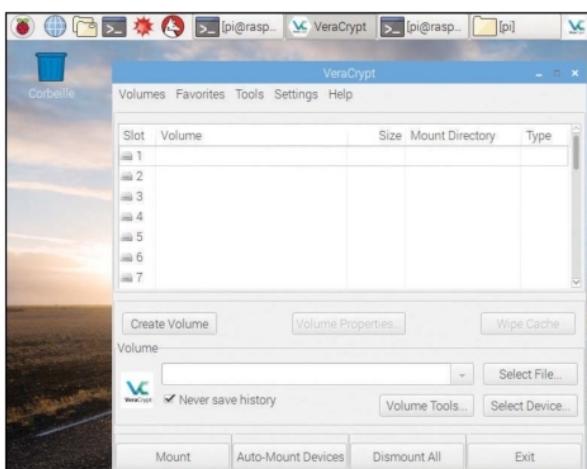
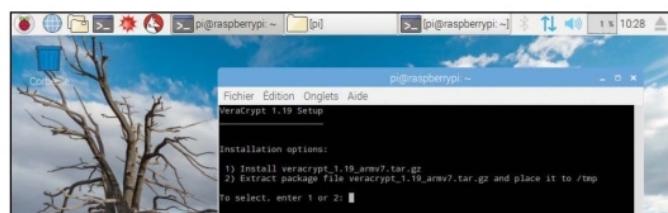


#07

## J'AI DÉCOUVERT LE RASPBERRY PI AVEC VOTRE MAGAZINE PIRATE INFORMATIQUE ET J'AI APPRIS QUE LE LOGICIEL DE CHIFFREMENT VERACRYPT EST AUSSI DISPONIBLE SUR LA FRAMBOISE. COMMENT L'UTILISER ?

C'est effectivement une bonne surprise d'autant que Mounir Idrissi, le créateur de VeraCrypt, est un garçon très sympa que nous avions interviewé dans notre autre revue justement. Pour info, VeraCrypt est une alternative sérieuse à TrueCrypt, un logiciel de chiffrement laissé à l'abandon par ses créateurs dans des circonstances étranges impliquant apparemment le gouvernement américain.

Basé sur la dernière version «sûre» de TrueCrypt (comprenez sans «backdoor»), VeraCrypt a su dépasser le maître en éliminant certains bugs et en accélérant le processus de génération des clés de chiffrement. Bref, VeraCrypt propose de créer des volumes chiffrés,(sorte de coffre-fort numérique), sur l'espace de stockage de votre Raspberry Pi. Attention, le logiciel n'est disponible que pour les ARMv7 (Raspberry Pi 2), mais il fonctionne apparemment aussi avec le Raspberry Pi 3 (?). Pour l'utiliser il suffit de télécharger l'archive en suivant notre lien et de la décompresser dans le répertoire de votre choix avec :



**sudo tar xaf veracrypt-1.19-raspbian-setup.tar.bz2**

Installons ensuite quelques librairies :

**sudo apt-get install makeself libfuse-dev**

**sudo apt-get install libwxgtk3.0-dev**

Puis lançons le script d'installation pour la version «graphique» :

**bash veracrypt-1.19-setup-gui-armv7**

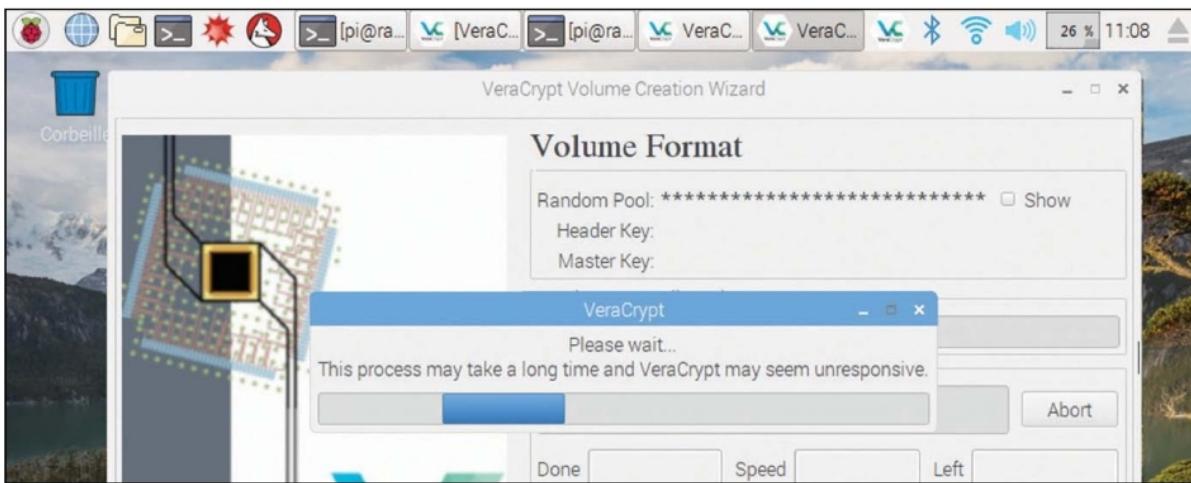
Allons dans le répertoire adéquat et lançons la bête.

**cd /usr/bin/**

**veracrypt**

Nous reviendrons sur VeraCrypt dans un prochain numéro si vous le désirez...

Lien : <https://goo.gl/QGjw34>



# PROJETS



## Cartographie aérienne

Un Raspberry Pi, un Xtrinsic Sensor Board (module de mesure de l'altitude, pression atmosphérique et champs magnétiques), la ProtoCam+ (module de prise de vue), un picavet (fixation suspendue par des câbles) et... un cerf-volant. Voilà ce qu'il a fallu à Richard Hayler pour faire du « kite-mapping », ou cartographie en cerf-volant. Utiliser un Raspberry Pi a notamment permis de programmer le déclenchement de la caméra uniquement lorsque celle-ci était à l'horizontale. Attention à l'atterrissement !

Lien : <https://goo.gl/EEvMyS>



## Photographie

James Mitchell a une obsession : photographier la lune, mais aucun appareil ne lui donne satisfaction, jusqu'au jour où il découvre la Picamera. Possédant un objectif Canon 78-300 mm, James fait imprimer une monture Canon EF pour Raspberry Pi ([www.thingiverse.com/thing:909176](http://www.thingiverse.com/thing:909176)) et démonte la lentille de la Picamera avant d'assembler le tout. Le capteur de cette dernière reçoit donc la lumière qui passe par l'objectif, et la traite avec sa définition maximale, 5 Mégapixels. Pour plus d'étanchéité à la lumière parasite, James recommande d'envelopper la monture de scotch noir. Un projet parfait pour tout travail de photo qui nécessite un zoom important (photographier le ciel, les animaux sauvages...).

Lien : <https://goo.gl/hC9Ohv>





## GIFs animés

Avec la taille miniature du Raspberry Pi Zero et l'imprimerie 3D, de plus en plus abordable, vous obtenez des projets comme la PIX-E, un appareil photo qui prend des GIFs. Nick Brewer l'a confectionnée simplement à partir du Pi Zero et du module Pi Camera. Le reste se passe dans la partie logicielle. En plus d'avoir eu cette idée originale, Nick explique parfaitement toutes les étapes nécessaires à la création de la PIX-E, et précise qu'elle est modifiable à souhait (longueur des GIFs, batterie, upload direct sur les réseaux sociaux...).

Lien : <https://goo.gl/5Cs4NU>



## Contrôle vocal

Jasper, c'est Siri, Google Now ou Cortana en mieux, et en open source. Concrètement, Jasper est un système de contrôle vocal fonctionnant sur Raspberry Pi, entièrement personnalisable puisque son code est en libre accès. Au-delà des fonctions basiques (donner l'heure, lancer une musique, fournir les prévisions météo...), tout est en théorie possible. Couplé à un micro USB (celui de votre choix), Jasper est un dispositif « always on », qui vous écoute donc en permanence. On doit cet assistant à deux étudiants de Princeton, Charles Marsh et Shubhro Saha.

Lien : <https://jasperproject.github.io>



# PROJETS

## Surveillance aérienne



J'ai un peu de temps aujourd'hui, je vais me fabriquer une mini-tour de contrôle aérien. D'accord, ce n'est pas exactement ça, mais l'idée de PiAware est bien de vous permettre de suivre les avions en temps réel grâce à votre framboise préférée. La boutique de FlightAware possède tout le matériel nécessaire (antenne, récepteur RTL-SDR, filtre ADS-B...), ainsi que la distribution PiAware donc. Vous devez créer un compte sur [www.flightradar24.com](http://www.flightradar24.com) afin de visualiser les avions. Pour savoir comment confectionner votre récepteur ADS-B, rendez-vous sur l'article de François Mocq (lien ci-dessous).

Lien : <https://goo.gl/sZwnBf>

## Anonymat

Un petit boîtier relié à une box Internet, devenant un point d'accès sans fil qui fait passer la connexion par le réseau TOR, vous rendant anonyme. On dirait un gadget de Mission Impossible, mais non : vous pouvez tout à fait créer le vôtre ! Pour faire simple, l'Onion Pi est un Raspberry Pi sous Raspbian couplé à une antenne Wi-Fi. Notez que le lien ci-dessous pointe vers un kit qui n'est plus d'actualité. Remettez-le au goût du jour en vous servant de la liste des composants, et à vous le boîtier spécial anonymat à emporter partout ! Une aide à la configuration de l'engin est disponible en anglais : <https://goo.gl/QDlyHn>.

Lien : <https://goo.gl/TTpQcf>



## Miroir intelligent

Miroir, mon beau miroir... Dis-moi quelle température il fait aujourd'hui. Et aussi les nouvelles du jour. L'équipe de Hacker House a revisité le conte de Blanche-Neige pour créer un miroir intelligent propulsé par un Raspberry Pi. Derrière la surface réfléchissante (une feuille d'acrylique) se trouve un écran d'ordinateur encastré dans l'armature en bois. La feuille d'acrylique permettant de voir ce qu'il y a derrière elle, vous voyez ce qu'affiche l'écran en même temps que votre reflet. Si vous voulez allier travail du bois et codage, ce projet est fait pour vous !

Lien : <https://goo.gl/VstvKj>



## Headlines

- Far From Over: Hermine to Batter Northeast Coast With High Winds
- US, Russia talk on Syria to go into Monday; no deal yet. Fox News
- In the Indian city where Mother Teresa founded her order, anti-abortionists protest
- World's Largest Ape Listed As Critically Endangered, Pandas Still Vulnerable
- Obama says security ties with Turkey undiminished since conflict

## GPS de navigation

Attention, projet potentiellement onéreux. Pas en lui-même, mais parce qu'il vous servira uniquement sur... un bateau. Oui, on parle bien d'un GPS maritime fabriqué avec un Raspberry Pi, un module GPS et un écran (entre autres). Jens Christoffersen, navigateur, n'avait pas envie



de dépasser des centaines, voire des milliers de dollars pour s'offrir un tel appareil (les entrées de gamme démarrent à 200 € en moyenne). Au final, son bricolage lui a coûté moins de 100 dollars. Reste à trouver comment protéger efficacement la machine du sel marin.

Lien : <https://goo.gl/he9tEG>

LES GUIDES DE RÉFÉRENCE RASPBERRY PI

L'officiel PC

# RASPBERRY PI

## Idées & Projets Clés en Main

ABONNEMENT  
1 AN POUR 25 €



SOIT 4 GUIDES  
100 % RASPBERRY !  
PRATIQUE &  
ÉCONOMIQUE !

LES GUIDES de L'UTILISATEUR pour  
TOUT SAVOIR et TOUT FAIRE avec votre RASPBERRY PI

- > Projets et tutos exclusifs
- > Codes inclus
- > Dossiers pratiques complets pour débutants et experts
- > Sélection et test de matériels
- > L'actu et les nouveautés !



À DÉCOUPER (OU À PHOTOCOPIER), À COMPLÉTER ET À RENVOYER SOUS ENVELOPPE AFFRANCHIE À :  
ID PRESSE - 27, BD CHARLES MORETTI - 13014 MARSEILLE

- Abonnement à L'Officiel PC - Raspberry Pi pour 4 numéros, je joins mon règlement de 25,00 €  
 Abonnement à L'Officiel PC - Raspberry Pi pour 8 numéros, je joins mon règlement de 50,00 €

OUI, JE M'ABONNE :

Nom .....

Prénom .....

Adresse .....

Code Postal .....

Ville .....

E-Mail .....

Signature obligatoire :

- Je joins mon règlement par chèque à l'ordre de ID PRESSE  
(France uniquement)

Offre valable en France métropolitaine uniquement.

POUR NOUS CONTACTER :  
raspberry@idpresse.com  
ou 04 91 48 59 87

Offre valable jusqu'au 31 décembre 2017. Les délais d'acheminement de La Poste varient selon les régions et pays. Conformément à la loi Informatique et Libertés du 6/1/1978, vous disposez d'un droit d'accès et de rectification quant aux informations vous concernant, que vous pouvez exercer librement auprès de ID Presse - 27 Bd Charles Moretti - 13014 Marseille

RÉDUCTION  
DE  
**-20%**

LES AVANTAGES :

- > - 20 % sur le prix en kiosques
- > Ne manquez aucun numéro
- > Vos magazines livrés chez vous gratuitement