

N° 07 - NOV 2012

Soutenez-nous en achetant un exemplaire imprimé de ce mag!



The

MagPi™

Un Magazine pour les utilisateurs de Raspberry Pi

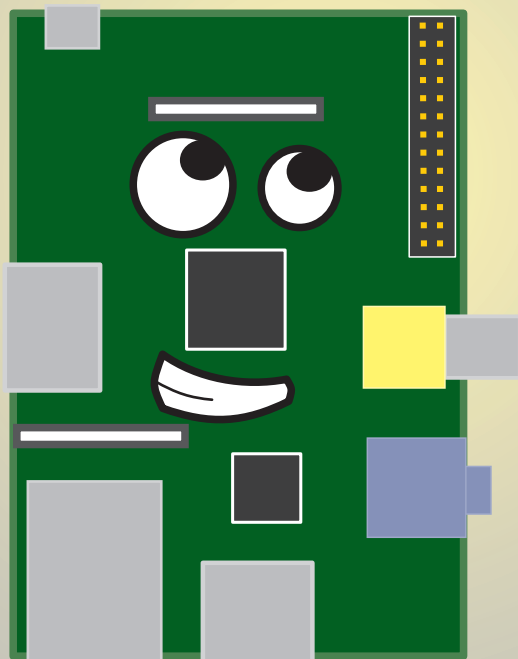
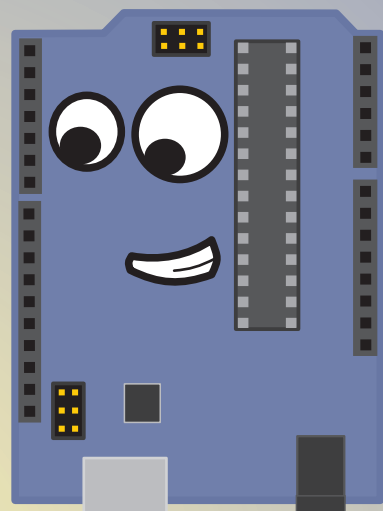
Arduino et RasPi sont connectés!

Dans ce numéro...

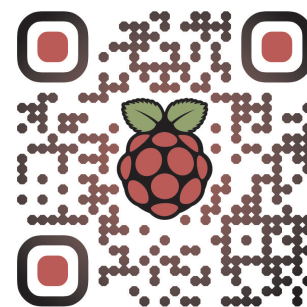
- Interruptions
- Pi Solaire
- Mode Turbo
- Pi Evolution
- C++

Plus...

- Une interview des développeurs de Raspbian
- Fabriquez votre jeu d'échelle sur circuit imprimé
- Les bases de GNU make



Gagnez un boîtier de Raspi ROSE !



Created At QRt.co

<http://www.themagpi.com>



The MagPi™

Raspberry Pi est une marque déposée par la Fondation Raspberry Pi. Ce magazine a été réalisé avec un ordinateur Raspberry Pi.



The MagPi™

Bienvenue dans le Numéro 7,

Le Raspberry Pi et l'Arduino sont parfaits dans les applications temps réel, pour lesquelles la puissance d'un microprocesseur (CPU) est requise. L'interconnexion de ces deux systèmes ouvre également la possibilité d'utiliser de nombreuses extensions de l'Arduino (les shields). Attendons-nous à voir apparaître des projets vraiment intéressants dans le futur.

Vous trouverez une interview du développeur principal de Raspbian (Debian pour le Raspberry Pi), des concours et une sélection d'articles sur la programmation pour vous faire les dents.

Si vous préférez recevoir une version papier de MagPi, visitez le site <http://www.modmypi.com> et passez votre commande dès maintenant!

Ash Stone, Rédacteur en chef du MagPi

Ash Stone

Rédac. en chef / Administrateur / (Header)

Jason 'Jaseman' Davies

Auteur / Site web / Design des pages

Meltwater

Auteur / Photographie / Design des pages

Chris 'tzj' Stagg

Auteur / Photographie / Design des pages

Ian McAlpine

Design des pages / Graphisme / Auteur

Joshua Marinacci

Design des pages / Graphisme

Lix

Design des pages / Graphisme

PaisleyBoy

Design des pages / Graphisme

Sam Marshall

Design des pages / Graphisme

Aaron Shaw

Design des pages / Graphisme

Nick

Design des pages / Graphisme

Matt '0the0judge0'

Administrateur / Site Web

Matthew Timmons-Brown

Auteur

Gordon Henderson

Auteur

Colin Deady

Auteur / Design des pages

Stewart C. Russell

Auteur

W.H.Bell

Auteur

Colin Norris

Éditeur / Graphisme / (C Cave Header)

Antiloquax

Auteur

Richard Ryniker

Auteur

Alex Kerr

Auteur



Sommaire

04 PI ET ARDUINO EN ACTION

Programmez l'Arduino avec un Raspberry Pi, par Stewart C. Russell

07 LE CONCOURS DU MOIS

Gagnez de bons éléments à ajouter à votre installation, de PC Supplies UK

08 PI SOLAIRE

Quand le soleil en mouvement peut garder votre Pi en marche, par Meltwater

10 LA CARTE A ÉCHELLE DE GORDON

Fer à souder à la main, par Gordon Henderson

12 GPIO ET INTERRUPTIONS

Un article sur la façon de gérer le GPIO depuis la ligne de commande, par Richard Ryniker

16 RASPBIAN, DES DÉBUTS JUSQU'A AUJOURD'HUI

Une interview de Mike Thompson, le "lead developer" de Raspbian, par Colin

18 CONFIGURATION TURBO POUR DES PERFORMANCES MAXIMALES

Un article sur la façon d'optimiser le Pi, par Matthew Timmons-Brown

21 LISTE DES ÉVÉNEMENTS DU MOIS

Raspberry Jams et autres événements communautaires

22 ÉVOLUTION-PI

Un article sur le développement du Raspberry Pi, par Jaseman

24 LES BASES DE GNU MAKE

Accélérer le développement de code avec GNU Make, par W. H. Bell

26 BIENVENUE AU C++ CACHÉ

Prenez en mains le C++, par Alex Kerr

28 LE PATCH EN SCRATCH

En avant vers la programmation défensive, par Antiloquax.

30 LE REPAIRE DU PYTHON

Utilisez les arguments de la ligne de commande, par Colin Deady

32 RÉACTIONS ET MENTIONS LÉGALES

Raspberry Pi & Arduino

Alors que de nombreuses cartes d'entrée/sortie sont en développement pour le Raspberry Pi, l'Arduino est lui, bien établi. Cet article montre comment dialoguer avec un Arduino, en utilisant Python et le protocole Firmata.

DIFFICULTÉ : MOYENNE

Cet exemple combine une sortie (le réglage de la luminosité d'une DEL avec un curseur) et une entrée (la lecture d'une température avec un LM35).

Matériel nécessaire

- Raspberry Pi
- Arduino
- Connexion Internet
- Petite carte prototype sans soudure (breadboard)
- Capteur de température LM35 (<http://www.ti.com/product/lm35>)
- DEL rouge de 5mm
- Résistance de 120Ω
- 4 fils de liaison mâle-mâle (utilisés ici : rouge, jaune, bleu et noir)
- 1 fil de liaison court (17,8 mm) (utilisé ici : noir)

Firmata (<http://firmata.org>) est un protocole série simple, qui vous permet de lire et d'écrire sur les ports d'entrée/sortie de l'Arduino depuis un ordinateur hôte. Il est le plus souvent utilisé avec le langage de programmation graphique "Processing" (<http://processing.org>), mais il peut être utilisé avec d'autres langages.

Installation de l'IDE Arduino et de Firmata

L'IDE Arduino est déjà présent sur les dépôts Raspbian, vous pouvez donc l'installer avec

```
$ sudo apt-get install arduino
```

Si c'est la première fois que vous le démarrez, l'IDE vous demande de créer un dossier pour ses programmes (appelés "sketches").

Ensuite vous devrez choisir le type de carte Arduino que vous utilisez dans le menu Tools/Board (j'utilise une Uno, mais j'ai aussi testé ceci avec une carte Duemilanove).

Vous aurez également à choisir le port série que vous utiliserez dans le menu Tools/Serial

Port - pour une Uno ce devrait être /dev/ttyACM0, les cartes plus anciennes utilisaient souvent /dev/ttyUSB0.



Pour installer le sketch Firmata sur votre Arduino, sélectionnez File / Examples / Firmata / StandardFirmata et cliquez sur le bouton Upload.



L'IDE compile votre sketch et le charge dans l'Arduino. Si tout ce que vous observez ce sont des DELs clignotantes et un message 'Done Uploading', Bingo ! Si par contre des messages écrits en rouge apparaissent, c'est qu'il y a un problème avec la connexion ou l'alimentation de l'Arduino.

Je vous recommanderai fortement de connecter votre Arduino soit via un hub alimenté, soit en lui connectant une alimentation extérieure, car le Raspberry Pi est un peu limité sur le courant disponible sur ses ports USB.

Si vous ne vous en sortez pas, visitez le site <http://www.ladyada.net/learn/arduino/>

Installation de pyFirmata

pyFirmata est le magicien qui permet à votre Arduino exécutant Firmata de dialoguer avec Python. Il faut peu de commandes pour l'installer :

```
$ sudo apt-get install python-serial me
rcurial
$ hg clone https://bitbucket.org/tino/p
yfirmata
$ cd pyfirmata
$ sudo python setup.py install
```

Si l'installation se passe bien, vous pouvez supprimer le répertoire pyfirmata.

```
$ cd .. ; sudo rm -r pyfirmata
```

Montage du circuit

Placez les composants comme indiqué sur le schéma, en vous aidant du tableau ci-

curseur de l'interface graphique, comprise entre 0 et 100, en nombre à virgule flottante, compris entre 0.0 et 1.0, puis l'écrit sur la broche D3.

cleanup() - tout ce que fait cette routine est d'éteindre la DEL, et essayer d'arrêter le programme proprement. Cependant elle ne parvient pas toujours à le faire; parfois il vous faudra en plus appuyer sur CTRL-C dans le terminal.

3. Activer l'interface graphique Tkinter. Tk (et sa version Python, Tkinter) est un GUI plutôt ancien, mais il est aussi relativement simple à utiliser.

Ainsi ici, je crée un curseur graphique de 400 pixels de large qui appelle la routine set_brightness() avec la valeur actuelle du curseur lorsqu'il change de valeur.

Poussé à fond à droite, le curseur appelle set_brightness(100), allumant la DEL au maximum.

Puisque la fenêtre est simple - juste un curseur et une étiquette - j'utilise la méthode basique pack() de Tk pour organiser les items dans la fenêtre.

Il dessine d'abord les items, puis les rassemble, façon Tetris, dans la fenêtre.

Une fois que c'est fait, il lance la première lecture de température (qui lance la suivante et ainsi de suite), et demeure finalement dans la boucle principale mainloop() de Tk jusqu'à la fin du programme, en répondant à vos entrées.

Autres pistes

Ceci est un exemple très simple du contrôle d'un Arduino en Python. Alors que Firmata peut piloter d'autres sorties plus complexes comme des servos, il prend le pas sur la totalité de la logique de la carte Arduino.

Un autre capteur qui impliquerait une mise en œuvre complexe ou un contrôle temps réel ne fonctionnerait certainement pas aussi bien.

Cela mis à part, vous avez connecté toute la puissance du Raspberry Pi à la robustesse de l'Arduino ; la seule limite, c'est votre imagination !

Auteur Stewart C. Russell

Stewart C. Russell vit à Toronto, où il crée des fermes éoliennes et des centrales électriques solaires. Quand il ne disparaît pas dans les zones venteuses ou ensoleillées de la planète, il est chez lui, pratiquant la radio-amateur (indicatif VA3PID), ou jouant du banjo, bricolant des ordinateurs, ou évitant de jardiner. Son site web est <http://scruss.com/blog>.

```
import pyfirmata
from Tkinter import *

# Création d'un nouvel objet carte
# spécifiant le port série ;
# à remplacer par /dev/ttyUSB0
# pour les anciens Arduinos
board = pyfirmata.Arduino('/dev/ttyACM0')

# Lance un nouveau thread itérateur pour
# que
# le buffer série ne déborde pas
iter8 = pyfirmata.util.Iterator(board)
iter8.start()

# Configuration des broches
# A0 Entrée (LM35)
pin0 = board.get_pin('a:0:i')
# D3 Sortie PWM (LED)
pin3 = board.get_pin('d:3:p')

# IMPORTANT ! ignorer les premières
# lectures jusqu'à ce que A0 retourne
# quelque chose de valide
while pin0.read() is None:
    pass

def get_temp():
    # Lecture du LM35 en °C
    label_text = "Temp: %6.1f C" % (
        pin0.read() * 5 * 100)
    label.config(text = label_text)
    # Redémarre après 1/2 seconde
    root.after(500, get_temp)
```

```
def set_brightness(x):
    # Configuration de la DEL
    # Le curseur renvoie 0 .. 100
    # pyfirmata s'attend à 0 .. 1.0
    pin3.write(float(x) / 100.0)

def cleanup():
    # Nettoyage à la sortie
    # et extinction de la DEL
    pin3.write(0)
    board.exit()

# Configuration de l'interface graphique
root = Tk()
# cleanup() sera appelée en quittant
root.wm_protocol("WM_DELETE_WINDOW",cleanup)

# dessin du curseur (luminosité de la DEL)
scale = Scale(root,
               command = set_brightness,
               orient = HORIZONTAL,
               length = 400,
               label = 'Brightness')
scale.pack(anchor = CENTER)

# place l'étiquette à côté du curseur
label = Label(root)
label.pack(anchor = 'nw')

# Démarre boucle lecture de température
root.after(500, get_temp)

# Lance boucle d'événements Tk
root.mainloop()
```

CONCOURS DE NOVEMBRE



Une nouvelle fois The MagPi et PC Supplies Limited sont fiers de vous donner une chance de gagner quelques accessoires R-Pi !

Ce mois-ci, il y aura CINQ prix !

Chaque gagnant recevra un boîtier couleur pour Raspberry, offert par PCSL. Il convient aussi bien au Modèle A qu'au Modèle B avec accès pour câble GPIO et guide de lumière pour les DELs.

Pour avoir une chance de participer au concours de ce mois, visitez : <http://www.pcsllshop.com/info/magpi>
La date de clôture est le 20 novembre 2012.
Les gagnants seront avertis dans le magazine du mois prochain ainsi que par courriel.
Bonne chance !

Win 1 of 5 Cases



Made by:  from **PCSLSHOP.COM**

Pour découvrir la large gamme d'accessoires pour Raspberry Pi de PCSL visitez <http://www.pcsllshop.com>

Les gagnants du mois dernier!

Les 5 gagnants de l'édition limitée du support pour écran ACL de PCSL sont **Mike Bradbury (Manchester, UK)**, **David Corne (Birmingham, UK)**, **Brian Bowman (Chelmsford, UK)**, **Bart Sekulski (Bristol, UK)** and **William Green (Doncaster, UK)**.

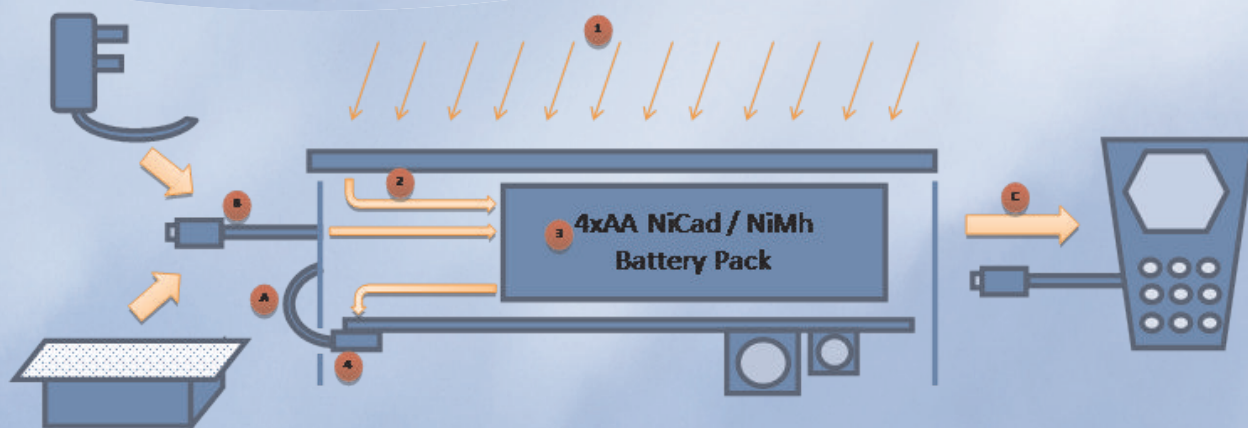
Félicitations. Nous vous enverrons prochainement par courriel la procédure à suivre pour récupérer vos cadeaux !



Un petit rayon de soleil...



En naviguant sur le web à la recherche de compléments intéressants pour le Raspberry Pi, je suis tombé sur cet objet chez CottonPickersPlace.



Fonctionnement du module :

Cet appareil à énergie solaire est essentiellement un chargeur solaire pour 4 piles de type AA situées à l'intérieur, susceptibles, à leur tour, de fournir l'alimentation du Raspberry Pi lui-même. Cet appareil sert aussi de support adapté pour le Raspberry Pi, et en prime, il peut même recharger la plupart des dispositifs alimentés par USB.

1. Le soleil éclaire le panneau solaire
2. La puissance disponible est utilisée pour recharger les piles (vous n'avez qu'à insérer vos propres 4xAA en NiCd/NiMh de la capacité requise).
3. Les piles emmagasinent l'énergie et fournissent du courant, même quand il y a peu ou pas du tout de soleil.
4. Le Raspberry Pi est alimenté !

A. La prise d'alimentation micro USB se connecte directement au Raspberry Pi (confortablement installé la tête en bas dans le boîtier).

- B. La prise USB de taille normale peut être utilisée pour alimenter directement le Raspberry Pi, et si les piles sont à plat, remettre un peu de charge supplémentaire (la recharge avec le soleil est nécessaire pour charger les piles à fond). Cette prise peut aussi être utilisée pour chaîner d'autres unités solaires, pour fournir plus d'énergie et allonger la durée d'utilisation de la batterie. (en la connectant à la prise USB latérale (C)).
- C. Une prise USB additionnelle sur le côté de l'appareil fournit un point de rechargement auxiliaire pour les téléphones mobiles et autres dispositifs USB. Les appareils Apple comme les iPods / iPhones / iPads sont supportés avec une version mise à jour si nécessaire.

Spécifications	
Panneau solaire	12 cellules Polycristallines 330 mA 2 Watts - 110x135mm
Boîtier	Imprimé en PLA (Acide PolyLactique) avec une imprimante 3D BRB-3000 - bio-dégradable. Temps d'impression typique : 2h15. Le Raspberry Pi est bien protégé à l'intérieur. Taille : 60x90x45mm

Valeurs typiques des sorties	
Orienté vers le soleil	env. 300mA+
Posé à plat	200-250mA
Ciel nuageux	30mA

	Temps de charge estimés		
	Soleil	à plat	nuageux
piles 3000mAh	10h	13h20m	100h
piles 2100mAh	7h	9h20m	70h

L'appareil ne surchargera pas les piles, et il n'y a pas de courant de décharge quand il ne charge pas les piles.

Temps de fonctionnement estimé du Raspberry Pi				
Modèle B	Soleil			
	A plat	Nuages	Nuit	
piles 3000mAh	23h	14h30m	7h30m	7h
piles 2100mAh	16h	10h15m	5h15m	4h50m
Modèle A				
piles 3000mAh	Charge	40h	11h	10h
piles 2100mAh	Charge	28h	7h45m	7h
Modèle B – consommation typique estimée 430mA selon ce qui est connecté dessus (en moyenne de 322mA à 480mA + pointes). Modèle A – Eben a annoncé qu'elle est aux environs de 200mA, donc 300mA devrait être une estimation raisonnable.				

Utilisation du chargeur :

Mes tests personnels, avec quelques vieilles piles 2500mAh, ont permis un usage nocturne d'environ 4 heures, ce qui semble raisonnable vu que les piles ont plusieurs années.

En utilisant le minimum de périphériques ainsi que le Raspberry Pi Modèle A dont la consommation est moindre (quand il sera disponible), on devrait fortement étendre le temps de fonctionnement. Le rendement du Raspberry Pi peut également être amélioré par le remplacement du régulateur linéaire 3,3V d'origine par un modèle à découpage - une mesure extrême, mais qui devrait diminuer de 25% la consommation (voir <http://goo.gl/dqUZL> pour les détails).

Bien que dans de nombreuses situations, on puisse utiliser le Raspberry Pi à distance, vous souhaiterez souvent le connecter à un écran. Généralement les écrans auront besoin d'autant de puissance que le Raspberry Pi, sinon plus. Ceci pourra s'améliorer quand la fondation pourra fournir des écrans ACL pilotés directement par le Raspberry Pi, ce qui consommera enfin moins d'énergie qu'un moniteur ACL classique ou une TV. Pour cet usage, un afficheur basé sur la super-efficace e-ink serait une excellente option, croisons les doigts pour que ça devienne une réalité.

Pour un fonctionnement à distance de 24 heures (ce que souhaitent certains utilisateurs de Raspberry Pi), il vaudrait mieux prévoir 2 unités solaires et avoir le plein soleil. De plus, en mettant au point une méthode pour couper à distance l'alimentation du Raspberry Pi, et la remettre en service quand c'est nécessaire, un pilotage à distance sans surveillance deviendrait viable. Un tel système pourrait même surveiller

l'état de charge des piles et piloter des opérations à des intervalles donnés. Nous serions heureux d'être informés d'un tel développement !

Conclusions:

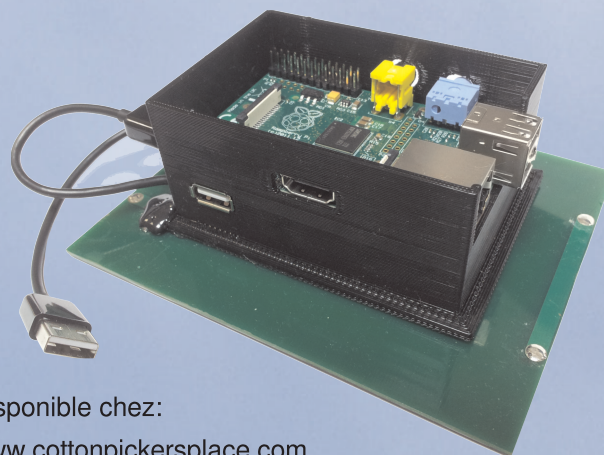
Cette unité compacte offre nombre de fonctions souples, et est une solution intéressante pour alimenter un Raspberry Pi à partir de piles. L'ajout du panneau solaire étend bien le temps de fonctionnement du système (particulièrement si vous vivez sous un climat ensoleillé), ou permet de charger le jour et l'utilisation pendant des heures la nuit. Alimenté par le secteur, il offre une excellente protection contre les coupures de courant, fonctionnant comme un onduleur. L'unité solaire offre aussi la possibilité de transporter l'ensemble sans éteindre l'unité centrale (idéal pour se déplacer vers le grand écran pour visionner un film, sans le souci d'avoir à trouver une prise de courant disponible, de booter...).

CottonPickersPlace travaille sur un modèle un peu plus grand de panneau solaire qui supporte également des piles plus grosses, qui devrait être capable de gérer le fonctionnement 24h/24 et 7j/7 et/ou alimenter des écrans et autres en même temps.

Par-dessus tout, cette unité solaire offre une grande flexibilité pour un prix abordable (25£ (31€) livrée en GB). Il est clair que du temps et des efforts sont nécessaires pour fabriquer chaque unité. CottonPicker a pris soin dans la conception de conserver un prix bas, sans sacrifier la qualité ou les fonctionnalités, avec un bon équilibre à mon avis.

Ce pourrait simplement être la pièce qui vous manque pour un projet complexe, ou juste un boîtier portable qui vous libère du secteur où que vous le souhaitiez !

Article de Meltwater



Disponible chez:

www.cottonpickersplace.com

Lien: goo.gl/w9Rs3

La carte à échelle Raspberry

La carte à échelle Raspberry est un ensemble de pièces créé dans le but d'introduire la soudure des composants et la programmation des GPIO sur le Raspberry Pi.

La carte à échelle est basée sur mon jeu d'échelle original, réalisé plus tôt cette année sur une platine Labdec, dont les détails peuvent être trouvés ici :

<https://projects.drogon.net/raspberry-pi/gpio-examples/ladder-game/>

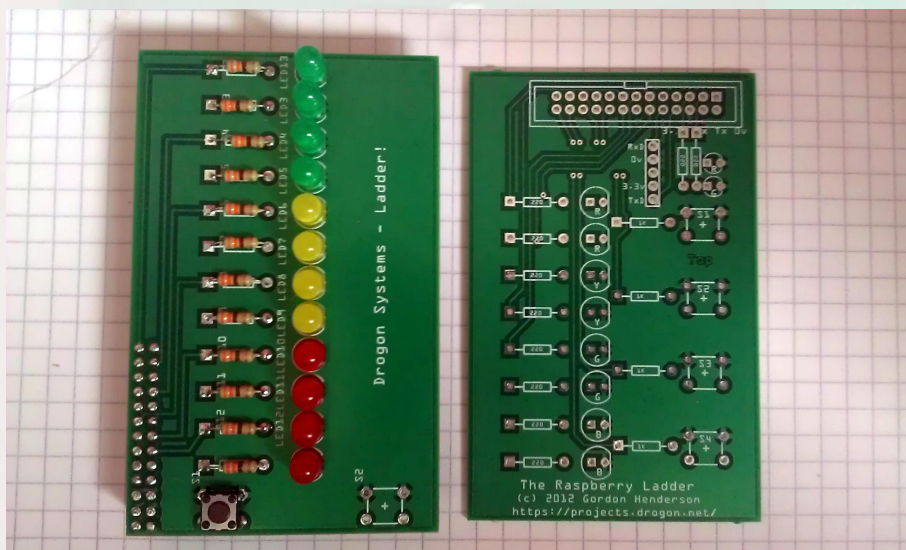
Cette version a été modifiée afin de la rendre plus utilisable pour d'autres projets et je l'espère, vous encourager à écrire vos propres jeux et simulations. Le logiciel que je vous fournis est une version de mon jeu d'échelle original, mon simulateur de feux de circulation Tuxx et un nouveau jeu de "Simon a dit".

L'ensemble inclut une plaque de circuit imprimé fabriqué en usine, 8 LEDs normalisées (2 de chacune des couleurs bleu, vert, jaune et rouge), 2 LEDs plus petites (verte et rouge), 4 interrupteurs à poussoir, 14 résistances pour les LEDs et interrupteurs et un bout de câble plat (déjà assemblé) avec un connecteur IDC pour brancher la carte à échelle sur votre Raspberry Pi.

Vous aurez besoin d'un minimum de matériel pour la soudure (fer à souder, soudure, pinces coupantes) et d'un peu de temps pour faire l'assemblage. De plus, un peu de nettoyant à circuit imprimé peut être utile pour nettoyer votre circuit quand il sera terminé, mais ce n'est pas obligatoire. Si vous êtes à l'aise avec la soudure, 10 à 15 minutes devraient suffire pour achever l'assemblage, sinon, vous pouvez procéder par étapes en vérifiant chacune de celles-ci, à mesure que vous progressez.

Vous pouvez acheter l'ensemble complet, incluant le circuit imprimé de Tandy pour environ £7 (11€), frais de port (GB) inclus.

<http://www.tandyonline.co.uk/electronicskits/raspberry-ladder-kit.html>



Prototype Raspberry Ladder Board and PCB

Soudure des composants :

La soudure n'est pas difficile, mais demande de la pratique. Avant de commencer, s.v.p. regardez cette bande dessinée (en anglais) :

<http://mightyohm.com/soldercomic>

ou en français

http://mightyohm.com/files/soldercomic/translations/Souder%20c%27est%20facile_FR.pdf

Une fois la lecture terminée, regardez le circuit imprimé et les composants - ne les retirez pas de leur emballage pour le moment - le kit Tandy est emballé dans des sachets dans l'ordre où vous les utiliserez, mais essayez d'abord d'identifier chaque pièce. Étudiez la photo d'un circuit complètement assemblé pour voir le résultat attendu.

Contrôlez le circuit - vérifiez qu'il ne présente pas de détérioration et identifiez l'endroit où chacun des composants sera placé. Les symboles imprimés en blanc vous aideront. Les résistances sont représentées par un petit rectangle, les interrupteurs des rectangles plus gros (presque carrés), et les LEDs, des cercles avec un méplat. Il y a une petite rangée de 5 trous qui ne sont pas utilisés pour ce projet et 2 rangées plus longues utilisées pour le connecteur du câble plat.

Premièrement, nous devons identifier les résistances. Dans le kit, il y a 2 valeurs, 220Ω et 1000Ω. Les 220Ω sont identifiées par leur code de couleurs - rouge, rouge, brun et les 1000Ω sont brun, noir, rouge. Cependant, si vous avez un doute, il y a 10 résistances de 220Ω et 4 de 1000Ω - il suffit de les compter pour les identifier.

Commencez par les 4 résistances de 1000Ω. Pliez les pattes aux bouts des résistances et insérez-les dans les 4 endroits prévus sur le circuit. Les résistances peuvent être insérées dans n'importe quel sens, mais le résultat sera plus joli si elles sont toutes montées dans le même sens. Introduisez les pattes des résistances dans les trous et pliez les pattes vers l'extérieur comme indiqué à la page 4 de la bande dessinée citée plus haut.

Pour ma part, je monte les 4 à la fois et ensuite j'utilise de la gomme adhésive pour les tenir en place lorsque je retourne le circuit pour les souder, mais vous préférerez peut-être les souder une par une pour commencer.

Assemblage :

Vous aurez besoin de 2 mains, alors assurez-vous que le circuit est bien fixé. C'est aussi une bonne idée d'avoir un éclairage correct, de façon

à voir ce que vous faites ! Voir la vidéo pour d'autres idées, quoi qu'il en soit, en général mettez en contact le fer à souder avec la patte du composant et en même temps avec la pastille du circuit imprimé, attendez 1 à 2 secondes et appliquez la soudure sur la pastille ou l'extrémité de la panne du fer à souder - la soudure devrait couler immédiatement et s'étaler parfaitement. Retirez la soudure, et ensuite (très important !) laissez le fer encore une ou deux secondes.

La plupart des composants seront endommagés en cas de surchauffe, mais n'ayez pas peur de chauffer jusqu'à 10 secondes si nécessaire. Avec un peu de pratique, vous devriez être capable d'effectuer une soudure en environ 5 secondes. Appliquez le fer sur la pastille et le composant, pause, soudure jusqu'à écoulement, pause, retirez le fer. Si vous n'êtes pas sûr de la soudure, attendez un moment que le composant refroidisse et recommencez.

Assurez-vous de bien nettoyer votre fer à souder avant chaque soudure - utilisez une éponge humide ou le nouveau système à sec qui ressemble à des ressorts cuivrés.

Une fois que vous aurez soudé vos premiers composants (ou les 4 premiers !), vous pouvez couper les pattes. Encore une fois, ceci requiert l'usage des 2 mains, vous devez tenir l'extrémité de la patte lorsque vous la coupez - sinon, elle risque de s'envoler et de vous faire mal en vous touchant. (De plus, votre mère, compagne, etc... ne sera pas contente de devoir ramasser ces petits bouts de métal !) Tenez l'extrémité de la patte, coupez au ras de la soudure et jetez la patte à la poubelle.

Lorsque vous aurez soudé les 4 premières résistances, vous pouvez passer aux résistances de 220Ω. Commencez par les 2 situées en haut de la carte et ensuite passez aux 8 situées sur le côté.

Passez ensuite aux interrupteurs. Ceux-ci devraient tenir d'eux-mêmes sur le circuit pendant que vous les soudez, mais assurez-vous de les installer dans le bon sens - ils sont légèrement rectangulaires, donc s'ils ne rentrent pas facilement, tournez-les d'un quart de tour et recommencez.

Maintenant, les LEDs. Heureusement à ce stade vous devriez maîtriser la soudure. J'ai gardé les LEDs pour la fin pour 2 raisons - premièrement, il est préférable de souder les composants courts en premier, puis les plus grands, et aussi pour que vous acquériez de la pratique en soudant les résistances et interrupteurs qui sont moins sensibles à la chaleur que les LEDs. Une soudure en 10 secondes ne devrait pas les endommager, mais heureusement, vous devriez maintenant être un peu plus rapide et confiant.

Les LEDs ont un sens de montage, alors regardez-les bien. Elles ont un côté plat qui correspond au méplat du dessin blanc sur le circuit imprimé. Le côté plat est toujours branché au négatif du circuit et l'autre côté (qui a une patte plus longue) au positif.

Prenez votre temps lorsque vous soudez les LEDs - assurez-vous qu'elles sont toutes bien plaquées contre le circuit et bien alignées.

Pour finir, le connecteur GPIO. Insérez-le dans le circuit, maintenez-le bien, soudez une broche et ensuite vérifiez avant de souder les autres. De préférence, soudez toutes les broches d'une ligne avant de tourner le circuit et de souder l'autre ligne.

Voilà, c'est fini. Votre circuit terminé devrait ressembler à celui de la page précédente.

Il est maintenant temps de le brancher sur un Raspberry Pi et d'exécuter le programme de test. Note : lorsque vous alimentez votre Raspberry Pi, ou le redémarrez avec la carte à échelle branchée, les 2

petites LEDs peuvent briller faiblement. Ceci est normal car elles sont alimentées par les résistances de tirage du bus I2C présentes sur le Pi.

Test :

Le code de test utilise la commande gpio de wiringPi, vous devez donc installer wiringPi d'abord.

Pour wiringPi (si vous ne l'avez pas encore) :

```
$ cd
$ git clone
git://git.drogon.net/wiringPi
$ cd WiringPi
$ ./build
```

Pour le logiciel à échelle pour Raspberry :

```
$ cd
$ git clone
git://git.drogon.net/ladder
$ cd ladder
```

Et pour exécuter le code :

```
$ ./ladderTest.sh
```

En quelques étapes simples, vous pourrez vérifier que votre circuit fonctionne correctement.

Une version légèrement modifiée du logiciel de feux de circulation Tux est aussi disponible - exécutez-la avec :

```
$ ./tuxx.sh
```

Quand l'initialisation est terminée, appuyez sur le bouton du bas pour démarrer la séquence, vous aurez plus de code et de détails le mois prochain !

La documentation complète (en anglais) sur le branchement des LEDs est fournie dans le fichier README, et le programme ladderTest est un script bash que vous pouvez copier et modifier au besoin, vous pouvez également regarder quelques exemples de code GPIO inclus dans le paquetage wiringPi, mais le plus intéressant commencera le mois prochain lorsque nous écrivons d'autres programmes pour la carte à échelle.

Interruptions et autres activités avec les broches du GPIO

Comment partager les ressources du GPIO entre plusieurs applications, et utiliser les interruptions pour remplacer le gaspillage que sont les boucles de contrôle d'état.

Après quelques expériences de début dans lesquelles un Raspberry Pi pilote des DELs et lit l'état d'interrupteurs, quand l'euphorie du "Ça marche !" s'évanouit, les utilisateurs astucieux peuvent facilement comprendre qu'ils vont rencontrer des problèmes lorsqu'ils vont entreprendre d'étendre ces programmes simples à des environnements plus complexes.

Je vais aborder deux de ces questions ici : comment partager les ressources du GPIO entre plusieurs applications, et utiliser les interruptions pour remplacer le gaspillage que sont les boucles de contrôle d'état.

Il y a eu une effrayante quantité d'instructions "Exécutez ce programme en tant que root" publiées pour les utilisateurs de Raspberry Pi. Pour un utilisateur expérimenté, ça ressemble plutôt à : "Tiens, gamin. Voilà des lames de rasoir. Emmène-les et regarde ce que tu peux couper avec."

Le privilège root ne devrait être utilisé qu'en dernier recours. Son utilisation normale est la création du système et la configuration - l'établissement d'un environnement protégé où les erreurs dans un programme n'affectent pas les autres applications, et ne peuvent pas provoquer un plantage du système. Au pire, un programme d'utilisateur qui plante ne devrait compromettre que les ressources allouées à ce programme.

Linux possède un nombre important de pilotes de matériel, programmes intégrés au noyau qui assurent l'interface entre les ressources matérielles et les applications. Les systèmes de fichier sont un bon exemple. Ils fournissent des fonctions conviviales comme ouvrir un fichier, lire, écrire, alors qu'ils gèrent les accès matériels et maintiennent les structures de données nécessaires pour allouer et libérer l'espace disque, partager l'accès de manière appropriée entre plusieurs programmes, et gérer la récupération de données après des

événements comme les coupures de courant.

Le privilège root facilite les interactions avec les activités du système. Si vous êtes chanceux, le résultat est une panique immédiate et un plantage système. Dans des circonstances moins favorables, un logiciel malicieux pourrait être installé dans le système : ce logiciel peut alors communiquer via une connexion Internet avec des criminels à la recherche d'informations personnelles ou qui pourraient exploiter votre Raspberry Pi pour des activités néfastes.

Linux dispose de moyens génériques pour gérer les ressources du GPIO. Il crée une interface pratique à manipuler par les programmes utilisateurs, protège les ressources GPIO utilisées par les pilotes comme I2C et SPI, et fournit un accès spécifiques aux broches du GPIO, de façon telle qu'une application n'a pas à s'inquiéter de ce que d'autres programmes peuvent faire avec les autres broches du GPIO. Cet accès individuel aux broches du GPIO est important, parce que sans lui, chaque application gérant le GPIO devrait surveiller les accès concurrents des autres applications partageant des broches du GPIO (verrous, gestion des interruptions, et autres casse-têtes à mettre en place).

Le service GPIO sous Linux utilise des fichiers présents dans le répertoire `/sys/class/gpio/`. Oui, comme beaucoup d'autres fichiers de configuration système ou de contrôle, ces fichiers appartiennent au root. Je n'en tiendrai pas compte pour l'instant, pour rendre la description de cette interface plus facile. Je vous promets d'y revenir plus tard et vous présenterai un utilitaire pour encapsuler les opérations privilégiées de manière responsable.

Configuration des broches

La commande `echo` est couramment utilisée dans les procédures shell pour afficher des

messages sur la sortie standard, ou avec une redirection pour écrire dans un fichier. Par exemple :

```
echo Salut tout le monde.
```

écrit sur l'écran "Salut tout le monde". Avec une redirection de sortie :

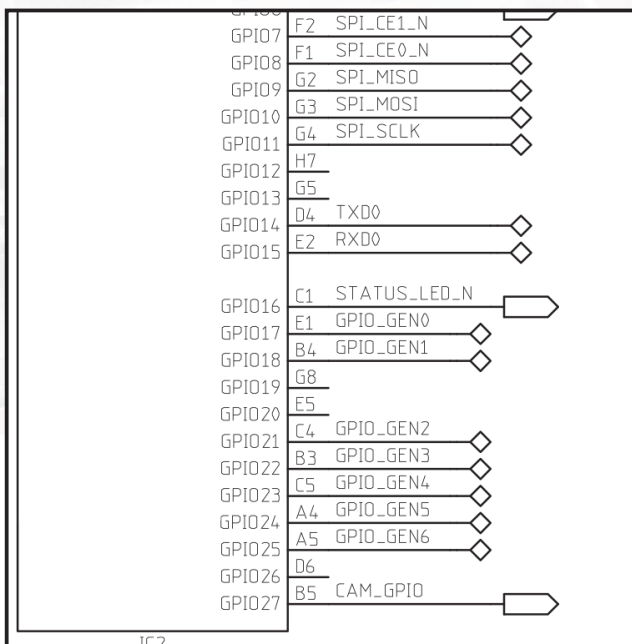
```
echo Salut tout le monde. >fichier_01
```

crée le fichier "fichier_01" qui contient le même message.

La commande `echo` sera utilisée pour certains exemples d'utilisation du GPIO. La broche 23 est utilisée parce qu'elle est pratique et facilement accessible sur la borne 16 du port à 26 broches du Raspberry Pi. Elle est marquée `GPIO_GEN4` sur le schéma du Raspberry Pi (<http://www.raspberrypi.org/wp-content/uploads/2012/04/Raspberry-Pi-Schematics-R1.0.pdf>).

Pour créer une interface utilisateur pour la broche 23, utilisez `sudo` ou, en tant que `root`, exécutez :

```
echo 23 >/sys/class/gpio/export
```



Ceci fait créer par le noyau un répertoire `/sys/class/gpio/gpio23` qui contient 4 fichiers dont nous allons discuter : `active_low`, `direction`, `edge` et `value`. Les valeurs initiales contenues dans ces fichiers (s'il n'y a pas de connexion externe sur cette broche) sont :

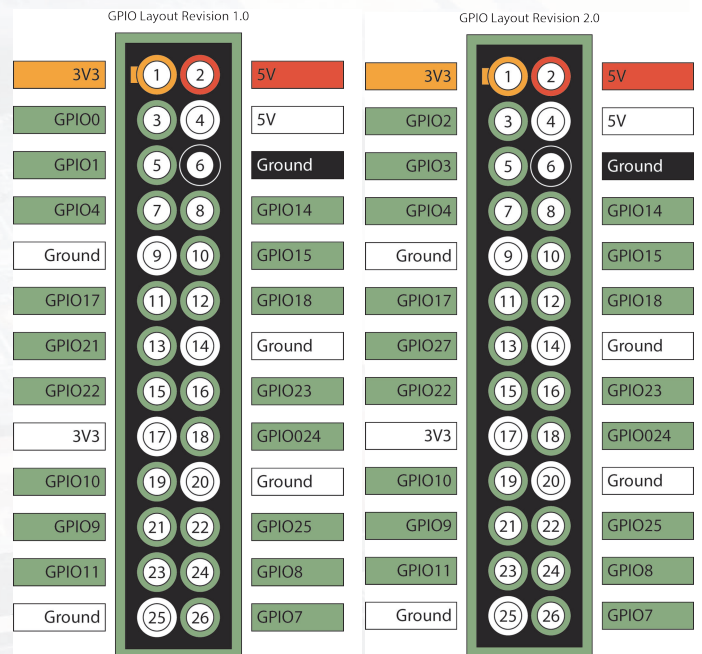
```
active_low 0
direction  in
edge       none
value     0
```

Pour en faire une broche de sortie :

```
echo out >/sys/class/gpio/gpio23/direction
```

Si vous voulez initialiser la valeur de sortie, avant que le pilote de sortie ne soit validé, une des lignes ci-dessous peut être utilisée pour définir la direction de la broche avec une valeur initiale :

```
echo low >/sys/class/gpio/gpio23/direction
echo high >/sys/class/gpio/gpio23/direction
```



SVP notez la modification des broches GPIO 3,5,7 & 13 sur la Revision 2.0

Pour mettre cette sortie on ou off :

```
echo 1 >/sys/class/gpio/gpio23/value
echo 0 >/sys/class/gpio/gpio23/value
```

Pour inverser la logique de la broche :

```
echo 1 >/sys/class/gpio/gpio23/active_low
```

Faites ceci avant de lire une entrée ou de définir une valeur de sortie. Si `active_low` est à 1 (ou n'importe quoi d'autre que 0) et que `value` est mis à 1, la broche est mise à l'état bas, etc.

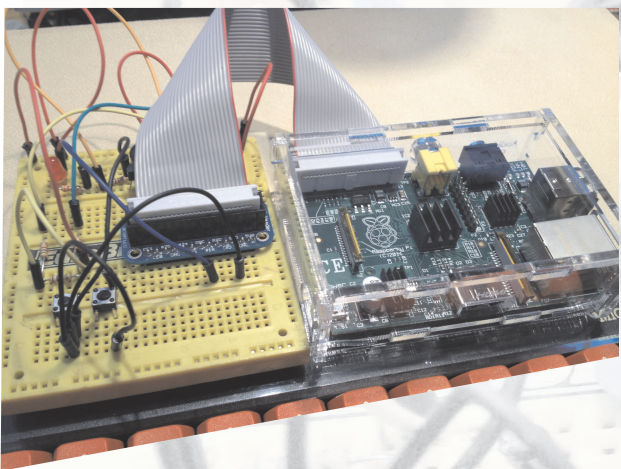
À quelle vitesse le système modifie-t-il la valeur des broches du GPIO ? Un programme simple en python <http://ryniker.ods.org/raspberrypi/MagPi/gpio23-max.py> génère des impulsions à 19 kHz. Si on l'écrit en C (voir <http://ryniker.ods.org/raspberrypi/MagPi/23->

maxa.c) la fréquence monte en gros à 120 kHz. La fréquence réelle varie car le Raspberry Pi fait d'autres choses qui suspendent momentanément la boucle - maintenir l'horloge à jour, gérer l'activité réseau, et autres processus système et utilisateur.

Le Programme

Comme promis précédemment, voici le programme

http://ryniker.ods.org/raspberrypi/MagPi/gpio_control.c qui exécute les opérations nécessitant le privilège root pour qu'un utilisateur normal puisse utiliser les broches du GPIO. Les commentaires au début du programme expliquent comment le compiler



et l'installer. Une fois installé (par root), du fait qu'il dispose du droit "setuid", le programme s'exécute avec le userid de root, il a donc le privilège requis pour lire ou écrire une broche du GPIO, et configurer les permissions requises pour les fichiers utilisés pour contrôler cette broche.

Les programmes qui s'exécutent avec le privilège root devraient n'être écrits que par des programmeurs vraiment paranoïaques. La plupart du code de gpio_control.c ne fait que vérifier si les arguments sont vraisemblables, et essaie d'être informatif si quelque chose d'inattendu arrive.

Pour utiliser gpio_control pour contrôler la broche 23 de sorte que toutes les manipulations de broches mentionnées plus tôt ne requièrent pas le privilège root, exécutez simplement :

```
gpio_control 23 export
```

gpio_control peut être facilement configuré, avant compilation, pour permettre l'accès du GPIO à tous les utilisateurs, ou aux seuls utilisateurs figurant dans un groupe donné. Chacune des 54 broches du GPIO peut être

configurée individuellement pour autoriser ou interdire l'accès.

Le Raspberry Pi utilise la broche 16 du GPIO pour contrôler la DEL verte "Status OK". Si quelqu'un tente d'accéder à la broche 16 du GPIO, l'opération échoue car le noyau utilise cette ressource :

```
ryniker@raspberrypi:~$ gpio_control 16
export
export failed: Device or ressource busy
```

Les autres programmes du noyau peuvent gérer les broches du GPIO, ce qui les rend indisponibles pour les autres utilisateurs. C'est bien. Un petit souci pourrait venir d'un utilisateur qui allumerait/éteindrait la DEL d'état, mais que se passe-t-il pour le pilote I2C du noyau ? Il peut facilement subir des plantages aléatoires si les broches qu'il utilise sont modifiées de façon incompréhensible pour lui.

Le noyau mémorise l'état des broches du GPIO. Par exemple, supposons qu'une broche est utilisée, configurée par l'utilisateur comme sortie, puis libérée. Les fichiers de l'espace utilisateur disparaissent, mais la broche reste une broche de sortie et conserve la dernière valeur reçue. Si cette broche est utilisée à nouveau, les fichiers de l'espace utilisateur sont recréés pour refléter l'état sauvegardé.

La commande echo est pratique pour être utilisée dans des scripts shell, éventuellement en ligne de commande, mais Python est plus adapté pour écrire de vrais programmes. La douzaine de lignes de gpio23-max.py vous en fournit un exemple simple.

Maintenant que j'ai exposé les éléments de base du contrôle du GPIO, cette facilité peut être utilisée pour remplacer la "boucle infinie", dans laquelle un programme lit sans cesse la valeur d'un signal d'entrée et exécute une opération lorsqu'il change, par un programme bien plus efficace qui ne s'exécute que quand la valeur du signal change. Avec une seule entrée et rien d'autre à faire jusqu'au changement de sa valeur, une boucle pourrait faire l'affaire. Quoiqu'il en soit, cette boucle consommerait 100% des ressources CPU, et ainsi concurrencerait agressivement tout autre programme qui aurait besoin d'une ressource du Raspberry Pi.

On peut introduire un délai dans la boucle, disons une commande "sleep 0.5" pour

attendre une demi-seconde avant chaque parcours de la boucle. Ceci permet à d'autres applications de tourner pendant le temps de sommeil, mais signifie qu'il existe un temps moyen d'un quart de seconde avant que tout changement sur l'entrée ne soit pris en compte. Délai plus court, réponse plus rapide, CPU plus chargé... un choix cornélien!

À mesure que le nombre d'entrées augmente, et que le nombre de réponses à ces entrées devient plus important et plus varié, il est souvent nécessaire de gérer les tâches avec des priorités différentes. Les interruptions sont le moyen de connecter rapidement une entrée comme "Il y a un gouffre juste devant la voiture" avec la réponse "Stop!".

Un autre programme en Python

http://ryniker.ods.org/raspberrypi/MagPi/interrupt_test23.py montre comment gérer les interruptions. Ce programme configure la broche 23 en entrée, configure le fichier edge en "both" (les deux), de sorte que les interruptions se produisent sur un front montant ("rising") ou descendant ("falling"), ouvre ensuite le fichier value de la broche. L'appel de `select.poll()` crée un objet de polling (balayage des entrées) "po", ensuite `po.register()` ajoute le fichier value de la broche du GPIO en tant que source qui peut répondre à une requête de `po.poll()`. Ce programme utilise cette seule source d'interruption, mais d'autres broches du GPIO, et de nombreuses autres sources, peuvent être enregistrées dans l'objet de polling. Par exemple, un pipe connecté à un autre processus pourrait être une source d'interruption, ou encore un socket qui reçoit des données sur le réseau depuis un système distant.

Le second paramètre de `po.register` spécifie laquelle des 3 conditions sera reconnue comme interruption. La valeur `select.POLLPRI` spécifie que seule "donnée prioritaire à lire" provoquera une interruption. Les autres conditions possibles - "données disponibles" et "prêt pour une sortie" - sont toujours vraies pour une broche de GPIO, donc une opération de polling lorsque l'une d'elles est activée sera réalisée immédiatement. Si d'autres sources d'interruptions sont enregistrées avec `po`, elles peuvent utiliser ces conditions.

Parfois, l'absence d'un signal attendu peut être importante. L'appel à `po.poll(60000)`

attendra une interruption, mais seulement pendant 60 secondes (60.000 millisecondes), avant de retourner une liste vide de signaux d'interruption, indiquant qu'il a atteint la limite fixée (time out).

Le noyau maintient le fichier value d'une broche du GPIO avec 2 valeurs : un caractère 0 ou un 1 qui représente la valeur actuelle sur la broche, et un caractère LF (Line feed, '\n', 0x0A). `f.seek(0)` remet la position courante au début du fichier, de façon à ce que la valeur du premier caractère puisse être lue à nouveau.

Extension du GPIO

Seules quelques broches du GPIO sont accessibles sur le Raspberry Pi, mais plusieurs personnes ont montré que des circuits intégrés peu chers comme le MCP23017 peuvent utiliser le bus I2C pour augmenter ce nombre. Un schéma comme <http://shop.ciseco.co.uk/k002-slice-of-pi-o/> peut être utilisé jusqu'à 8 fois pour ajouter 128 broches d'entrées/sorties numériques au Raspberry Pi. Utiliser la configuration drain-ouvert de la sortie interruption du MCP23017 pour relier les interruptions de plusieurs circuits à une seule broche du GPIO. Une résistance de tirage au +3V3 met l'entrée à 1, jusqu'à ce qu'un des circuits connectés la mette à 0. Quand une interruption survient, le gestionnaire d'interruption doit lire les valeurs de tous les dispositifs générant des interruptions pour savoir lequel a activé le signal d'interruption (il peut y en avoir plusieurs), lancer les programmes de traitement appropriés, puis supprimer toutes les demandes d'interruptions (donc l'entrée du GPIO repasse à l'état haut) pour autoriser le déclenchement d'une nouvelle interruption.

Résumé des URL

Schéma du Raspberry Pi :

<http://www.raspberrypi.org/wp-content/uploads/2012/04/Raspberry-Pi-Schematics-R1.0.pdf>

Les programmes :

<http://ryniker.ods.org/raspberrypi/MagPi/gpio23-max.py>

<http://ryniker.ods.org/raspberrypi/MagPi/23-maxa.c>

Extension des entrées/sorties :

<http://shop.ciseco.co.uk/k002-slice-of-pi-o/>

Auteur : Richard Ryniker



= Raspbian

Une interview avec Mike Thompson

La version Raspbian de la distribution Debian fournit le système d'exploitation sur lequel tournent la plupart des installations de Raspberry Pi. Dans Le MagPi de ce mois-ci, nous interviewons Mike Thompson, le fondateur de Raspbian.

Q : Pourquoi avez-vous décidé de créer Raspbian ?

J'ai lu en janvier que la Fondation avait prévu d'utiliser Fedora et je me demandais si je pouvais faire la même chose mais avec Debian, car c'est ma distribution Linux préférée sur processeurs ARM. Depuis longtemps, je suis intéressé par le fait d'apprendre à construire un système d'exploitation mais aussi de contribuer en apportant quelque chose en retour à la communauté Debian. Raspbian m'a donné cette opportunité.

J'ai réalisé que la création de Raspbian allait nécessiter une certaine quantité de ressources, de temps et d'efforts. J'ai commencé en posant des questions sur des forums pour comprendre ce qu'il y aurait à prendre en compte si cela devait être entrepris. J'ai fouillé pour voir comment je pouvais faire et à partir de là les choses ont avancé.

Raspbian est un effort commun de moi-même et Peter Green (Plugwash), développeur Debian.

Q : Pourquoi le Raspberry Pi alors qu'il existe beaucoup d'autres cartes à prix relativement bas ?

J'ai un attrait personnel pour les systèmes Linux à faible coût et j'étais très motivé par la Fondation Raspberry Pi selon laquelle il y a sur le marché le désir et le besoin d'avoir un système tel que le Raspberry Pi. En fin de compte, mon souhait est de voir ces systèmes à des prix se situant dans une fourchette de 5 à 10 dollars. Il peut se passer encore quelques années avant d'en arriver là, mais les choses pourraient devenir beaucoup plus intéressantes dans le monde si des ordinateurs à très bas prix mais relativement sophistiqués comme le Raspberry Pi devenaient accessibles à grande échelle.

Q : Comment avez-vous fait pour qu'une version de Raspbian fonctionnelle, devienne le SE officiel de la Fondation Raspberry Pi ?

Peter Green et moi étions déjà bien dans le cadre du projet quand, vers la mi-juin, nous avons eu des échos selon lesquels la Fondation était intéressée par Raspbian. Quand j'ai démarré, mes attentes étaient de créer une alternative à Fedora qui serait utilisée par peut-être 10 à 20% des utilisateurs de Raspberry Pi, et je ne

m'attendais pas à ce que Raspbian devienne la distribution Linux "officielle" sur Raspberry Pi. Après la diffusion des premières images de test de Raspbian, et la construction d'une partie significative du dépôt, des gens ont commencé à s'enthousiasmer et à dire qu'ils espéraient que la Fondation choisisse une distribution basée sur Debian. Je savais que la Fondation connaissait Raspbian grâce à des forums et qu'elle pouvait choisir de l'utiliser si elle la jugeait digne d'intérêt.

Q : En quoi la version Raspbian diffère-t-elle de celle de la Fondation ?

Raspbian est une recompilation des paquets de Debian Wheezy pour ARM avec support matériel des calculs sur les nombres à virgule flottante, avec des paramètres de compilation optimisés pour le processeur ARMv6 du Raspberry Pi. Nous utilisons tels quels les travaux effectués par la Fondation sur le noyau car aucun composant des interfaces binaires du noyau n'utilise de virgule flottante. Cela nous économise beaucoup d'efforts, ce qui nous permet de nous concentrer sur la recompilation des paquets.

Alex Bradbury, le Lead Developer de la Fondation, a travaillé sur l'image Raspbian de la Fondation. Comme Raspbian est essentiellement un clone de Debian, il a pris les mêmes scripts que ceux qu'il avait utilisés pour l'image basée sur Debian, fait quelques modifications mineures et les a utilisés pour construire leur propre image basée sur Raspbian. Je pense qu'il a apprécié de voir que nous avions suivi étroitement Debian, ce qui a rendu assez trivial le processus de création d'une version optimisée d'une Debian basée sur nos paquets Raspbian, pour le Raspberry Pi.

Q : Est-ce que les cartes Freescale Quick Start Board i.MX53 que vous avez achetées en mars s'avèrent être une plateforme de construction convenable ?

Nous utilisons toujours 8 cartes Freescale i.MX53 pour construire les paquets Raspbian. Ce sont des systèmes assez rapides, avec un processeur ARMv7 à 1 GHz et 1 Go de RAM. Lors de leur construction, certains paquets nécessitent de grosses quantités de RAM pour construire d'énormes structures de liaison en mémoire et nous tournons avec 1,5 à 2 Go de swap car nous dépassons la mémoire disponible. Un PC moderne typique avec 4 Go de mémoire

pourrait construire un paquet important en une heure ou deux, mais sur ces systèmes ARM, cela peut monter jusqu'à 20 ou 30 heures. Avoir 8 systèmes disponibles pour paralléliser les constructions est devenu nécessaire en mai et juin quand nous avons construit la masse de 18000 paquets sources qui constituent un peu moins des 38000 paquets binaires de Raspbian. Si nous n'avions qu'un seul système, nous serions encore en train de construire des paquets aujourd'hui. Nous utilisons des versions modifiées des propres outils Debian de construction afin de distribuer la construction des paquets sur l'ensemble des 8 systèmes.

Je suis arrivé sur le projet avec une expérience très limitée en ce qui concerne la construction de systèmes d'exploitation et j'ai dû apprendre tout ce dont j'avais besoin. Heureusement pour moi, Peter Green m'a rejoint et son expérience sur Debian et sur les outils de construction Debian a été essentielle pour rendre Raspbian possible. J'ai été développeur de logiciels tout au long de ma carrière mais je n'aurais jamais tenté de construction à cette échelle. Je comprends maintenant parfaitement pourquoi les entreprises ont des ingénieurs dédiés qui se focalisent uniquement sur la construction de projets logiciels importants !

Q : A quel point êtes-vous dépendant du travail effectué en amont par la communauté Linux ?

Extrêmement dépendant. Raspbian n'aurait pas été possible si le groupe au sein de Debian ayant créé armhf n'avait pas fait leur travail en amont 18 mois auparavant, bien qu'il l'ait fait pour ARMv7 et non ARMv6. Peter Green est catégorique, et je pense à juste titre, que Raspbian est aussi proche que possible d'une distribution Debian officielle, sans pour autant être une version officielle. Aussi longtemps que nous maintiendrons cet engagement avec Raspbian, cela restera une base ferme pour la Fondation Raspberry Pi et la communauté.

En aval, garder Raspbian aussi proche de Debian diminue le risque de ne voir plus que deux gars travailler dessus. Peter Green s'est assuré que tout ce que nous avons fait est complètement ouvert. Si nous devons fermer boutique demain, notre travail est là, dupliqué sur 30 ou 40 miroirs à travers le monde. Toute personne maîtrisant la construction de Debian peut facilement tout récupérer et en assurer la maintenance. Raspbian représente donc un faible risque pour la Fondation.

Q : Peut-on s'attendre à ce qu'il y ait d'autres gains de performances dans Raspbian ?

Je pense que nous atteignons le maximum en ce qui concerne le côté logiciel des choses. Remplacer le CPU par un ARMv7 ou ajouter plus de mémoire [Note de l'éditeur : les livraisons de Pi avec 512 Mo de RAM viennent juste de commencer !] serait l'idéal car certaines personnes se heurtent aux limites, par exemple, en surfant sur Internet depuis un environnement de bureau

graphique.

Je pense qu'en général l'efficacité logicielle est partie à la dérive, en particulier avec les applications graphiques. J'apprécie toujours une utilisation parcimonieuse et efficace de la mémoire pour les calculs. Malheureusement, la réalité est que la plupart des applications graphiques ont besoin de tonnes de RAM et de CPU puissants. Nous devons toujours encourager les gens à programmer de manière efficace avec des ressources limitées. Si des systèmes comme le Raspberry Pi étaient apparus huit ans plus tôt, nous aurions pu voir une petite branche de productivité logicielle nécessitant globalement moins de ressources sur toutes les plates-formes informatiques.

Comparés à Turbo Pascal qui est sorti sur CP/M au début des années 80, et plus tard Turbo C, qui offraient tous deux des environnements de développement intégrés complets, compacts et rapides, les environnements de développement modernes, basés sur des interfaces graphiques, consomment des ressources énormes et ne fonctionnent pas correctement, voire pas du tout, sur le Raspberry Pi. Il est dommage qu'il n'y ait pas aujourd'hui de réel équivalent de Turbo Pascal ou Turbo C sur Raspberry Pi car ces systèmes ont disparu quand les environnements graphiques ont fait leur apparition. Je crois qu'il existe une très grande opportunité de faire revenir ce genre d'outils pour un environnement aux ressources relativement faibles tel que le Raspberry Pi.

Q : Quel travail reste-t-il à faire sur Raspbian ?

Nous sommes désormais largement en mode maintenance. Quand Debian diffuse des mises à jour de paquets, nous les récupérons, les construisons, et les renvoyons sur les dépôts. Personnellement, avec Raspbian, j'ai atteint mon objectif consistant à créer une version de Debian pour ARMv6 avec support matériel des calculs en virgule flottante.

Je suis heureux que Raspbian ait permis tant de choses dans la communauté Raspberry Pi. Cela est également important car j'ai été capable de donner en retour à la communauté Linux dans son ensemble, et j'espère que cela conduira des milliers d'autres utilisateurs vers Debian dans le futur.

Mike Thompson est un ingénieur informatique qui vit dans la région de la baie de San Francisco. Il a un bagage varié dans la conception de systèmes embarqués, le développement d'applications mobiles/portables, le développement d'applications pour PC et la conception de systèmes à grande échelle sur Internet. C'est un multi-entrepreneur qui a co-fondé deux précédentes entreprises et qui est le fondateur et le lead developer de Raspbian, le principal système d'exploitation sur Raspberry Pi.

TURBOPI

Envie de donner du punch à votre Pi ? Mettez le turbo !

Le Raspberry Pi a une fréquence d'horloge de 700 MHz. Cela signifie qu'il effectue 700 000 000 cycles par seconde. La vitesse d'horloge du processeur est une indication de la rapidité avec laquelle il peut exécuter les opérations. Elle se mesure en mégahertz (MHz) ou en gigahertz (GHz) avec 1000 MHz égal 1GHz. Donc plus le nombre de MHz est élevé, plus le processeur va travailler vite.

Qu'est-ce-que le surcadencage et le survolage ?

Alors que la vitesse originale du processeur Raspberry Pi est de 700 MHz, il y a un moyen d'obtenir de meilleures performances... Le surcadencage... Et grâce à la dernière image de Raspbian, c'est plus facile que jamais !

Le surcadencage consiste à faire fonctionner un composant d'ordinateur plus vite que la vitesse pour laquelle il a été conçu, sachant que cela implique un compromis avec une instabilité accrue, et une espérance de vie du processeur raccourcie. Pour le Raspberry Pi, ces effets induits sont si minimes que vous seriez stupide de NE PAS mettre en œuvre le surcadencage !

Le surcadencage nécessite une puissance plus élevée. Si vous voulez surcadencer votre Raspberry Pi à 900MHz ou plus, vous devrez apporter plus de puissance par survolage. De combien vous pourrez surcadencer dépend de plusieurs facteurs ; votre Raspberry Pi, la qualité de votre alimentation et peut-être aussi celle de votre carte SD. À cause des tolérances de fabrication, 700 MHz est la fréquence garantie par le fabricant. Mais chaque Raspberry Pi est différent et donc chacun d'eux a des limites différentes.

Il y a plusieurs paramètres pour le surcadencage et le survolage. Ceux-ci sont détaillés sur http://elinux.org/RPi_config.txt#Overclocking_options mais la

dernière image de Raspbian contient une option de configuration facile. Plus important, ceci vous permet de surcadencer et survolter votre Raspberry Pi tout en conservant le bénéfice de la garantie.

Considérations sur l'alimentation

Quand vous surcadencez, il est important d'utiliser une bonne alimentation. J'utilise un chargeur de Kindle (e-liseuse d'Amazon) qui est de bonne qualité et fournit un courant de 800mA. Les chargeurs d'iPhone d'origine Apple sont aussi un bon choix, ils fournissent un courant de 1A. Faites attention aux chargeurs iPhone d'autres fabricants. La Fondation Raspberry Pi a remarqué que certains d'entre eux ne répondent pas aux spécifications.

ATTENTION : Faites une sauvegarde

Avant de commencer, il convient de noter qu'il existe une possibilité que le surcadencage corrompe les données sur votre carte SD. Vérifiez que vous avez une sauvegarde de tout document important, mais aussi de votre fichier /boot/config.txt. La meilleure façon de faire ceci est de copier les

```
Raspi-config

info          Information about this tool
expand_rootfs Expand root partition to fill SD card
overscan     Change overscan
configure_keyboard Set keyboard layout
change_pass  Change password for 'pi' user
change_locale Set locale
change_timezone Set timezone
memory_split Change memory split
overclock    Configure overclocking
ssh          Enable or disable ssh server
boot_behaviour Start desktop on boot?
update      Try to upgrade raspi-config

<Select>          <Finish>
```

fichiers sur une clé USB ou de les transférer sur un service de stockage en ligne comme Dropbox.

Autrement, vous pouvez aussi faire une

sauvegarde complète de votre carte SD en utilisant le même programme Win32DiskImager que celui qui vous a servi à créer votre carte SD Raspbian. Cette fois, au lieu d'écrire une image sur la carte SD, vous allez créer une image de la carte en lisant les données de la carte SD. Insérez votre carte dans un lecteur de carte SD sur votre machine Windows et démarrez Win32DiskImager. Donnez un nom à votre image et cliquez sur Read. Quand la sauvegarde est terminée, la taille du fichier enregistré devrait être très proche de la taille de votre carte SD.

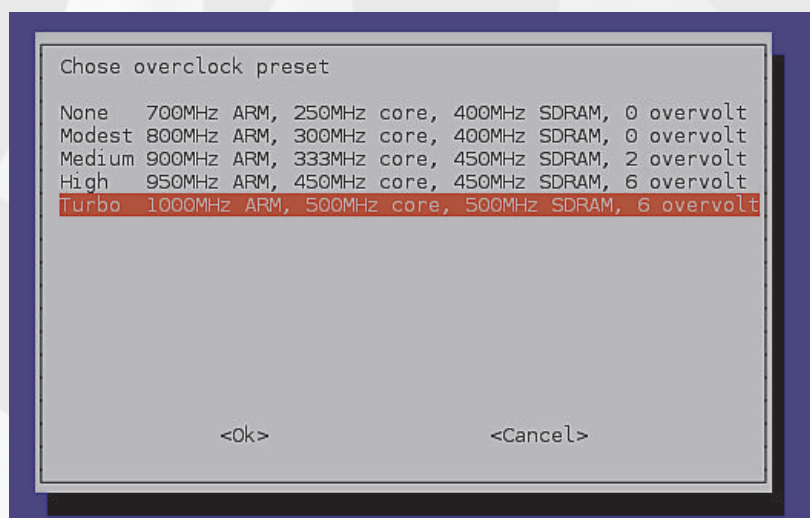
Surcadencage

Pour modifier les paramètres de surcadencage, démarrez votre Raspberry Pi. Dans un terminal, saisissez la commande suivante :

```
$ sudo raspi-config
```

Une fenêtre bleue apparaît, comme celle de la page précédente. C'est le même menu de configuration qui apparaît quand vous démarrez le Raspberry Pi pour la première fois. La première chose à faire est de mettre à jour l'utilitaire raspi-config lui-même. Avec les flèches de direction, descendez et choisissez l'option "update". Attendez que raspi-config vérifie et installe (si besoin) sa dernière version. Quand c'est fait, vous pouvez passer au surcadencage !

Maintenant descendez et choisissez l'option



"overclock". Vous pouvez choisir la vitesse à laquelle vous souhaitez surcadencer. Il y a cinq options pré-réglées parmi lesquelles vous choisirez ; Aucun, Faible, Moyen, Élevé ou Turbo.

Avec les flèches de direction, choisissez votre pré-réglage de surcadencage préféré. Il vaut mieux commencer par Turbo pour voir

s'il est stable. Si ce n'est pas le cas, essayez Élevé, suivi de Moyen puis Faible. Après avoir fait votre choix, quittez l'utilitaire raspi-config et redémarrez. Vous devez redémarrez pour que les modifications soient appliquées. Les modifications sont écrites dans le fichier /boot/config.txt.

Oups ! Ça a flingué mon Pi ?

Que faire si votre Raspberry Pi n'amorce plus ? Cela signifie que les réglages de surcadencage dépassent les limites de fonctionnement de votre Raspberry Pi, mais ne vous en faites pas, c'est simple à corriger. D'abord débranchez l'alimentation de votre Raspberry Pi, attendez quelques secondes, puis remettez sous tension. Aussitôt, appuyez et maintenez enfoncée la touche <SHIFT> de votre clavier. Regardez le texte qui apparaît sur votre écran. Vous verrez ce qui suit :

```
[ ok ] Checking if shift key is held down: Yes.  
Not switching scaling governor.
```

Cela signifie que les réglages de surcadencage sont ignorés et que vous allez redémarrer comme d'habitude. Si vous n'avez pas de chance, il se peut que le système de fichier de votre carte SD soit corrompu, et vous devrez restaurer votre image de sauvegarde. Si le démarrage se passe bien, ouvrez un terminal et entrez la commande :

```
$ sudo raspi-config
```

Cette fois, quand vous choisirez "overclock", essayez une option plus lente.

Si vous vous connectez à votre Raspberry Pi à distance via SSH ou une autre application d'accès à distance, et ne pouvez appuyer sur la touche <SHIFT> du clavier pendant le démarrage, vous devrez modifier les réglages de surcadencage manuellement, en utilisant un autre ordinateur muni d'un lecteur de carte SD. Vous éditez le fichier config.txt. Avec Linux et le Mac d'Apple, vous pouvez utiliser l'éditeur de fichier par défaut. Avec Windows, ce serait

maladroit de modifier le fichier config.txt avec le Bloc-notes (Notepad). Il vaut mieux utiliser le programme TextPad (shareware) que vous pouvez télécharger sur <http://www.textpad.com>. Les lignes que vous devrez éditer sont arm_freq, core_freq, sdram_freq et over_voltage. Utilisez les valeurs figurant sur l'image de gauche

comme références pour les valeurs à utiliser.

Vous avez choisi un réglage de surcadencage et votre Raspberry Pi semble redémarrer sans problème. Comment être certain que son fonctionnement sera stable ?

Tests de fiabilité

Un certain nombre de tests peuvent vous aider à vérifier si votre Raspberry Pi sera stable. Ce n'est pas marrant d'avoir un système rapide mais instable. Si vous avez installé Quake 3, c'est un test parfait pour le réglage de surcadencage du CPU et du GPU. Un autre test facile que j'utilise est tout simplement la mise à jour de l'image de Raspbian. Pour la réaliser, ouvrez un terminal et entrez les commandes suivantes :

```
$ sudo apt-get update
$ sudo apt-get upgrade -y
```

Ceci est un bon test des réglages de surcadencage du CPU et de la SDRAM. Selon le nombre de mises à jour, le test peut durer jusqu'à 30 minutes, mais vous avez maintenant le tout dernier système.

Une fois que c'est fini, redémarrez votre Raspberry Pi. Cette fois faites attention aux messages qui apparaissent pendant la séquence de démarrage. Surveillez le message à propos de "mmc" ou tout message signalant des erreurs du système de fichier (file system). Pendant que vous y êtes, surveillez l'existence de messages à propos de [warn] et [fail]. Si vous voyez un de ces messages, il y a une faiblesse potentielle, et il vaut mieux essayer un réglage de surcadencage moins élevé.

Si vous disposez de plusieurs cartes SD il est souhaitable de tester chacune d'entre-elles. J'ai testé trois Raspberry Pi avec 9 cartes SD différentes s'étalant en vitesse de la classe 2 à la classe 10. Chaque Raspberry était un modèle de Révision 1, avec les fusibles polymères des ports USB remplacés par un bout de fil. Ils étaient alimentés par le même hub USB avec son alimentation 2A. L'un des Raspberry Pi est allé jusqu'au réglage "Moyen" de surcadencage ; les deux autres sont allés jusqu'au réglage "Élevé". Aucun des Raspberry Pi n'a fonctionné de façon stable en mode "Turbo".

Il est à noter que les deux Raspberry Pi qui ont fonctionné en mode "Élevé", l'ont fait uniquement avec 7 des cartes SD. Ça ne fonctionnait pas avec les 2 autres cartes ; une Transcend 4Go de classe 6, et une Lexar 16 Go de classe 6. Cependant, vos résultats

peuvent être différents.

Surveillance

Lorsque vous surcadencez, il est utile de connaître la fréquence et la température du CPU. C'est très facile sous LXDE. Faites un clic droit sur la Barre des Tâches en bas de l'écran et choisissez Ajouter/enlever des éléments au tableau de bord. La boîte de dialogue Préférences du tableau de bord s'ouvre, cliquez en haut à droite sur le bouton "+ Ajouter". La fenêtre "Ajouter un greffon au tableau de bord" s'ouvre. Cliquez sur CPUFreq Frontend puis sur le bouton "+ Ajouter", recommencez l'opération avec Moniteur de température. Si vous le jugez utile, ajoutez les applets Contrôle de volume et Moniteur de l'état réseau.

Un autre test simple consiste à lancer Midori et à visiter <http://www.raspberrypi.org>. Pendant qu'il se lance, maintenez la souris sur l'applet CPUFreq Frontend. Vous verrez la fréquence passer de 700 MHz à la fréquence définie dans le réglage actuel de surcadencage.



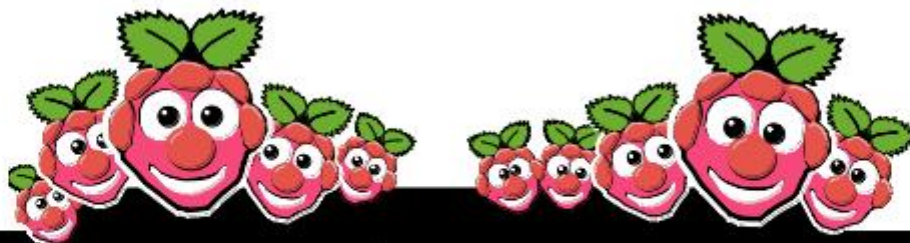
Pour visualiser une vidéo sur le surcadencage ainsi que d'autres trucs pour le Raspberry Pi, visitez ma chaîne YouTube : <http://www.youtube.com/user/TheRaspberrypiGuy>.

Article de Matthew Timmons-Brown & Ian McAlpine

LE SAVIEZ VOUS ?

En commandant une version papier de The MagPi sur <http://www.modmypi.com>, j'ai remarqué qu'ils vendaient un kit de refroidissement. Le kit de radiateurs du Raspberry Pi ("The Raspberry Pi Heat Sink Kit") comprend 3 petits radiateurs ainsi que du ruban thermique. Il peut améliorer la fiabilité de votre carte. Il y a un radiateur pour le SoC, un pour le GPU et un pour le régulateur de l'alimentation.

Lors de mes tests, la température du CPU grimpeait jusque 61°C sans radiateur, mais seulement à 52°C avec.



Le Guide des Événements du MagPi

Vous voulez savoir tout ce qui concerne le Raspberry Pi dans votre région ? Alors cette nouvelle section du MagPi est pour vous ! Nous avons pour objectif de lister tous les événements Raspberry Jam dans votre région en vous fournissant un calendrier RPi pour le mois à venir.

Organisez-vous un événement Raspberry Pi ? Voulez-vous le promouvoir ?
Contactez-nous par courriel à : editor@themagpi.com

Bloominglabs Raspberry Pi Meetup

Quand : Premier mardi de chaque mois, 19h00
Où : [Bloomington, Indiana, USA](#)

Les réunions ont lieu le premier mardi de chaque mois de 19h00 jusqu'à 21h00, tout le monde est bienvenu. Plus de détails disponibles sur <http://bloominglabs.org>

Durham Raspberry Jam

Quand: Mercredi 14 novembre, 17h30
Où: [Durham Johnston School, Durham, UK](#)

La réunion se déroulera de 17h30 jusqu'à 19h00 et le nombre de places sera limité.
Les billets et davantage d'informations sont disponibles sur <http://durhamjam-eorg.eventbrite.com>

Sheffield Raspberry Jam

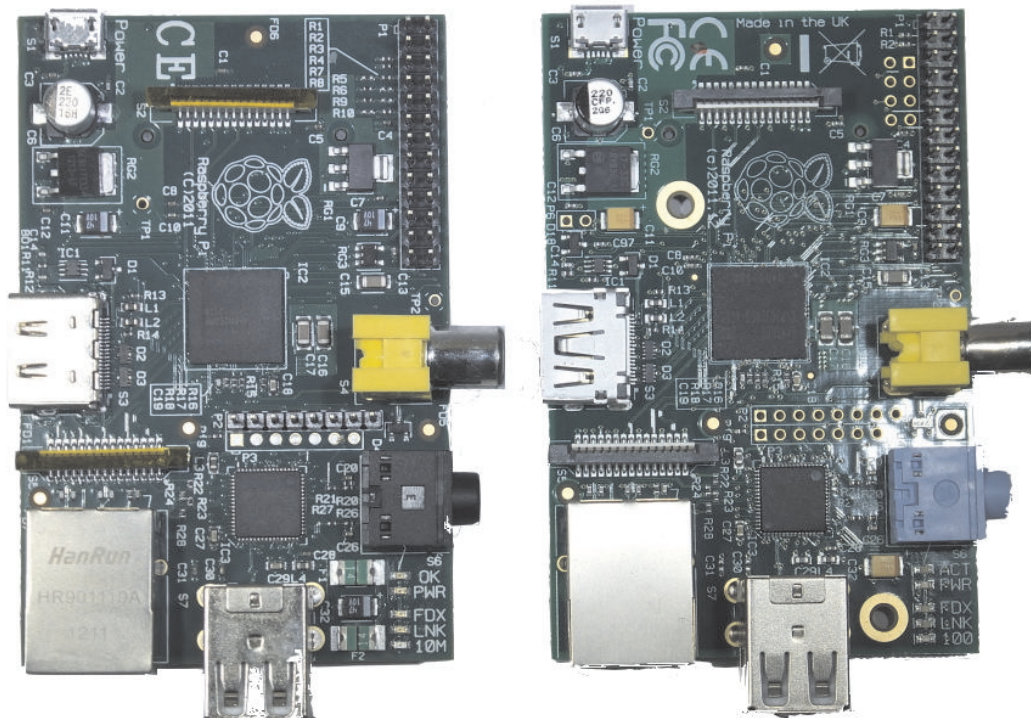
Quand: Dernier mercredi du mois, 18h30
Où: [1st Floor, The Workstation, Grinders Hill / Brown St., Sheffield, S1 2BX, UK](#)

Les réunions sont hébergées par GISThub. Les portes ouvrent à 18h20 et la réunion débute à 18h30 pour finir à 20h30. Plus d'informations disponibles sur <http://sheffieldraspi1210.eventbrite.com>

Annoncez votre **Raspberry Jam** ici

PI-EVOLUTION

Le rythme de développement autour du Raspberry Pi est difficile à suivre...



Si vous êtes l'un des premiers à avoir adopté le Raspberry Pi, vous avez sans doute démarré avec "Debian Squeeze" - un système d'exploitation Linux. Vous avez passé du temps à apprendre comment télécharger l'image, et l'écrire sur carte SD. Ensuite, il vous a probablement été présenté LXDE - l'interface graphique utilisateur x-window légère. Cela avait l'air semblable à Windows mais il y avait juste des choses légèrement différentes. Vous avez travaillé ensuite sur l'utilisation d'un gestionnaire de paquets - la commande "apt-get", "aptitude" ou peut-être "synaptic", afin de pouvoir télécharger et installer de nouvelles applications en paquets.

Vous veniez juste d'installer un ensemble décent d'applications utiles lorsque vous découvrez qu'une nouvelle version de Squeeze est disponible, celle-ci étant meilleure que la précédente, et donc vous recommencez tout. Ensuite, pas beaucoup plus tard, sort encore une nouvelle mise à jour du système d'exploitation - Debian Wheezy "Beta". Mais vous avez entendu dire qu'il y a quelques problèmes avec cette version, et vous ne savez pas si vous devez rester sur Squeeze ou passer à Wheezy. Tout était si confus.

Pendant ce temps, beaucoup d'autres personnes étaient toujours en train d'attendre patiemment que leur commande de Raspberry Pi soit traitée. Enfin, le petit colis est arrivé et vous avez été si content. Alors, vous avez appris qu'une nouvelle version améliorée du Pi était maintenant disponible, avec des trous de fixation et divers autres peaufinages, et vous vous êtes demandé si le vôtre devait maintenant être considéré comme dépassé. Alors pour rendre les choses encore

pires, il fut annoncé que le Raspberry Pi serait désormais livré avec le double de mémoire, et cela pour le même prix !

Enfin, une version améliorée de Wheezy arrive, mais maintenant c'est devenu quelque chose appelée "Raspbian Wheezy", qui est beaucoup plus rapide, et une grande quantité de bogues a été corrigée. Oh, attendez une seconde, en voici une nouvelle version, qui est encore meilleure, et j'apprends qu'une autre version est déjà prévue pour être diffusée d'ici quelques semaines.

Maintenant, si vous êtes du genre à détester le changement, arrivé à ce point, vous êtes sans doute en train de vous cogner la tête contre le mur, à vous demander si après tout l'informatique est faite pour vous.

En informatique, tout se passe souvent très vite, et traiter un flux constant de mises à jour est une chose à laquelle vous devez vous habituer. Si vous ne pouvez pas vous y adapter, vous serez vite dépassé.

Le rythme du développement va probablement commencer à ralentir un peu, une fois que les bases seront mises en place. Mais nous n'en sommes pas encore là. Vous pouvez vous attendre à une version du Pi pour l'éducation, dont les cartes seront intégrées à des boîtiers, et sans doute à une certaine forme de manuel d'instructions, ainsi qu'à une image grandement améliorée, et l'inclusion d'un grand nombre d'accessoires. Malgré ça, nous ne devons pas nous attendre à ce que les choses stagnent à ce stade

du développement. Il est prévu l'arrivée d'un certain nombre de composants et accessoires matériels, parmi lesquels des caméras, écrans et autres périphériques électroniques qui pourront se greffer sur votre Pi. Le plus probable à ce stade, c'est que votre Pi 256 Mio, actuellement sans boîtier, et auquel il manque les trous de fixation, va devenir pratiquement sans valeur.

Ce dont nous devons nous rappeler, c'est - L'objectif du Pi - d'apprendre les principes de la programmation, de l'interfaçage, ou simplement de l'informatique en général, et bien que beaucoup de choses changent, les principes fondamentaux restent les mêmes. Le Raspberry Pi est un périphérique pour l'apprentissage, et l'expérimentation. Ce n'est pas un accessoire de mode ou un gadget de luxe. Le Pi est plus qu'un simple objet jetable pas cher. Il y a des gens qui vendent leur iPhone 4 uniquement parce que l'iPhone 5 est maintenant disponible. Il y aura des possesseurs de Pi qui vont tomber dans le même piège, mais ceux qui obtiendront le meilleur du Pi sont les personnes qui arrêtent de s'inquiéter de ces choses et qui passeront plus de temps à bricoler avec, quelle que soit la version qu'ils pourraient avoir, et apprendront par l'expérience.

En ayant ça en tête, je me suis rendu compte que ma propre carte SD est dépassée, et je pense que je ferais mieux de jeter un œil sur "Raspbian Wheezy Édition du 18/09/2012" :

Dans le numéro 5, j'ai écrit un article comparant Debian Squeeze et Wheezy, et l'un des inconvénients majeurs que j'ai découverts, c'est que Wheezy, bien qu'étant vraiment plus rapide que Squeeze, offrait des performances très faibles pour la lecture multimédia. Je me suis dit que je pourrais d'abord voir si quelque chose a été amélioré avec cette version actualisée.

Tout d'abord, j'ai téléchargé l'image zippée de 439 Mo à partir de <http://raspberrypi.org/downloads>. Cela a pris environ 45 minutes en utilisant le lien de téléchargement direct à partir de mon ordinateur Windows. Ensuite, encore 45 minutes pour écrire l'image sur la carte SD avec Win32DiskImager. Je possède une carte MicroSD de classe 4 qui est plutôt lente, mais ça fait l'affaire.

Ensuite, j'ai inséré la carte MicroSD (placée dans un adaptateur de carte SD normal) dans mon Pi que j'ai démarré jusqu'à arriver sur l'écran de Raspi Config.

Curieusement, pour une fois, je n'ai pas eu à modifier les valeurs d'overscan de mon écran (j'avais toujours besoin de le faire sur les versions précédentes de Debian).

J'ai choisi d'étendre la partition racine de manière à pouvoir utiliser la totalité des 16 Go de la carte. J'ai aussi défini le fuseau horaire sur Londres et configuré le surcadencage en mode Turbo 1000 MHz, puis redémarré après avoir quitté le menu.

Après m'être logué, j'ai lancé LXDE en tapant "startx". Le bureau semblait presque identique à la version précédente, bien que l'image paraisse plus nette. J'ai ouvert une fenêtre LXTerminal et tapé "sudo amixer cset numid=3 1" pour rediriger l'audio vers la sortie

analogique.

J'ai ensuite utilisé "sudo apt-get update" et "sudo apt-get install xine-ui" pour récupérer le lecteur multimédia xine. Xine est un lecteur multimédia que j'avais regardé il y a quelques temps (pendant la réalisation du MagPi numéro 3). Il semblait prometteur, mais il était vraiment trop lent pour être utilisable, aussi je pensais que, peut-être, avec l'OS plus récent et le surcadencage, les choses pourraient aller mieux cette fois-ci.

Heureusement, mon nouveau Pi commandé chez RS Components est enfin arrivé (après six mois d'attente), et il pouvait gérer le "mode Turbo". Malheureusement, il est arrivé une semaine avant que la version 512 Mio ne sorte, et il lui manque également les trous de fixation.

J'ai eu une ancienne révision de la carte que Antiloquax a bien voulu m'envoyer, mais c'en est une qui ne gère pas le surcadencage - elle n'arrive pas à démarrer en mode Turbo, et était instable avec tous les niveaux plus bas de fréquences.

Xine a été capable d'ouvrir et jouer la plupart des formats que j'ai testés, bien que quelque chose de bizarre se passait quand je sélectionnais les fichiers à ouvrir. C'est seulement après avoir cliqué sur le bouton "play next chapter >|" qu'il est possible de sélectionner le média correct à lire, sinon un message indiquant "There is no MRL" reste affiché. Xine a réussi à lire des avi, mp3, mp4, mov tout comme des wma, wmv et... mpeg - OUI MPEG, bien qu'il ait commencé à sauter des images lors d'une tentative de jouer une vidéo avec un zoom supérieur à 100%. Je n'ai pas acheté le codec MPEG. Xine était plutôt bogué et instable, cependant le lecteur en ligne de commande "omxplayer" fonctionnait mieux, mais supportait beaucoup moins de formats. Encore une amélioration massive par rapport aux versions antérieures de Raspbian Wheezy, et je pense que celle-ci est digne de mettre Squeeze au placard une bonne fois pour toutes.

J'ai fait une pause avec les lecteurs multimédia et j'ai installé Scribus - le logiciel de PAO que nous utilisons pour faire le MagPi. J'ai chargé l'un des documents du numéro 6 sur Python, et remarqué que cela était particulièrement plus rapide grâce au surcadencage. C'était plus spécifiquement le cas lors de la bascule entre les calques, des agrandissements, et des clics-droits pour afficher la boîte de dialogue des propriétés.

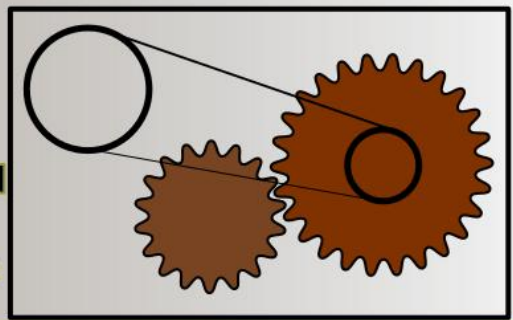
Pour finir, j'ai installé quelques jeux d'arcade : "beneath-a-steel-sky", "geki2", "geki3", "pangzero" et "xsoldier", tous ont très bien fonctionné.

Les distributions à jour ne sont pas forcément meilleures globalement - ça vaut la peine quelquefois de s'abstenir de faire la mise à jour jusqu'à ce que les bogues soient éliminés, mais s'enfuir la tête dans le sable et ignorer la marche du progrès est également une mauvaise idée. Gardez toujours un œil sur ce qui se passe au sein de la communauté pour éviter de passer à côté de mises à jour sympas et de gains de performances.

Article de JaseMan

MAKE

Make accélère le développement



Les bases de GNU Make

`make` est un élément central de la construction de paquets composés de code compilé ou de documentation sur un système LINUX. Quand `make` s'exécute, il lit un ou plusieurs fichiers texte contenant des cibles et des dépendances. Les cibles ne sont exécutées que si un fichier n'est pas présent ou s'il est plus récent que le fichier de sortie. La syntaxe autorise également des dépendances multiples, de telle sorte que la construction de quelques fichiers se fasse obligatoirement avant d'autres. Commençons avec un premier exemple :

```
# Crée newFile.txt s'il n'est pas présent
newFile.txt:
    touch newFile.txt
```

À l'instar des langages de script, les commentaires débutent par un caractère dièse (`#`). Il n'y a pas d'espace ni de tabulation devant la cible `newFile.txt` et elle est suivie par un caractère deux-points. Des dépendances pour la cible peuvent être ajoutées après le deux-points. Les actions sont placées sur des lignes après le nom de la cible et doivent être préfixées par un caractère de tabulation. Si des espaces blancs sont utilisés, `make` signalera une erreur.

Par défaut, `make` recherche un fichier appelé `Makefile`. Donc, utilisons `nano` (décrit dans l'article sur le C du numéro 3) pour créer un fichier nommé `Makefile` et contenant l'exemple ci-dessus. Ensuite, tapez `make`

La première fois que la commande `make` est invoquée, elle exécutera la cible, qui utilise la commande `touch` pour créer un fichier vide avec le nom spécifié. La seconde fois que `make` est appelé, il se rend compte que le fichier existe déjà, et aucune action ne sera nécessaire.

Un petit programme en C peut être utilisé pour démontrer les dépendances. Dans un nouveau répertoire, créez trois fichiers : `main.c`

```
#include "printString.h"
int main() {
    printString();
    return 0;
}
```

`printString.c`

```
#include <stdio.h>
void printString() {
    printf("Construit avec make !\n");
}
```

et `printString.h`

```
void printString();
```


Créons maintenant un nouveau fichier appelé `Makefile` contenant,

```
printString: main.o printString.o
    gcc -o printString main.o printString.o

main.o:
    gcc -c main.c

printString.o:
    gcc -c printString.c

clean:
    rm -f *.o
```

Cette fois, taper `make` va provoquer la compilation de chaque fichier `.c` pour produire des fichiers `.o`. Ces derniers seront ensuite liés ensemble pour former un exécutable appelé `printString`. La cible `printString` est la première cible du fichier et par conséquent celle par défaut. Quand `make` est lancé, il vérifie les dépendances de `printString`, c'est-à-dire que les fichiers `main.o` et `printString.o` existent et ne sont pas plus récents que la cible `printString`. Si l'un ou l'autre des fichiers n'existe pas, alors la cible pour le construire est exécutée. Toute cible autre que celle par défaut peut être lancée en tapant son nom après la commande `make` par exemple `make clean`

Écrire des fichiers `make` dans lesquels chacun des noms de fichier doit être spécifié peut rapidement devenir chronophage. Au lieu de spécifier des cibles de manière explicite, il est possible d'utiliser des variables automatiques,

```
printIt: main.o printString.o
    gcc -o $@ $^

%.o: %.c
    gcc -c $< -o $@

clean:
    rm -f *.o
```

Ce `Makefile` a exactement le même comportement que le précédent. La variable automatique `$@` correspond au nom de la cible, `$^` sont les noms de toutes les dépendances, et `$<` est le nom du premier prérequis. Pour chaque fichier `.o` nécessaire à la cible par défaut, `make` essaiera le joker `%.c`. Si le fichier `.c` est manquant, `make` signalera une erreur.

Des jokers peuvent aussi être utilisés pour définir une liste d'objets à partir des fichiers `.c` présents dans le répertoire courant,

```
OBJECTS = $(patsubst %.c,%.o, $(wildcard *.c))

printIt: $(OBJECTS)
    gcc -o $@ $^

%.o: %.c
    gcc -c $< -o $@
```

où `OBJECTS` est une variable. Dans ce cas, tous les fichiers `.c` du répertoire en cours d'utilisation seront utilisés pour construire un exécutable appelé `printIt`. L'instruction `wildcard` énumère tous les fichiers qui correspondent au motif `*.c`. Ensuite, `patsubst` enlève le `.c` final et le remplace par `.o`. La liste résultante est affectée à la variable `OBJECTS`. Essayez d'utiliser la commande `touch` pour actualiser l'horodatage de chaque fichier et relancez `make` pour voir ce qui se passe.

Les fichiers `Make` peuvent avoir plusieurs niveaux de dépendances. Pour la distribution de logiciels sur plusieurs plateformes, les `Makefiles` sont souvent générés à partir de modèles grâce à l'outil `autoconf`

Article de W. H. Bell



Bienvenue au C++ caché !

Une introduction au langage de programmation C++ - l'un des plus populaires utilisés aujourd'hui.

C++ est un langage de programmation, tel Python ou Java. Il est un peu plus évolué et très populaire, beaucoup de jeux vidéos récents et autres programmes sont écrits en C++. Il est rapide et facilement portable, ce qui signifie que le même code peut être réutilisé entre des machines Linux, Windows et Mac OS. Si vous êtes prêts à relever le défi, lisez la suite !

Essayez en tapant ce qui suit :

```
#include <iostream>
using namespace std;

int main()
{
    //Affiche un message.
    cout << "Bonjour, bienvenue sur C++" << endl;
    return 0;
}
```

Enregistrer le fichier sous le nom "hello.cpp" - .cpp indique qu'il s'agit de code source C++.

Après compilation et exécution, vous devriez voir afficher le message entre guillemets. Essayez de le modifier, recompilez et lancez-le, et voyez ce qui se passe. Cela peut paraître intimidant, mais regardez la suite, vous y trouverez une explication des termes utilisés.



WHAT' S GOING ON?

Bien, maintenant que nous avons écrit notre premier programme, comment savons-nous ce qu'il va faire, et qu'est-ce que tout ça signifie ? Je vais vous expliquer les aspects importants du code :

```
#include <iostream>
```

C'est la directive d'inclusion. On l'utilise pour importer des bibliothèques qui contiennent les informations nécessaires à la compréhension des instructions que vous donnez. La bibliothèque iostream contient les informations pour les entrées/sorties (input et output).

```
int main()
```

Il s'agit de la fonction principale. Tous les programmes ont besoin d'une telle fonction. Ce qui est à l'intérieur est exécuté. Toutes les fonctions commencent par "{" et se finissent par "}".

```
//Affiche un message.
```

En C++, tout ce qui commence avec "//" est un commentaire, et comme les commentaires dans d'autres programmes, ils sont ignorés par le compilateur.

```
cout et endl
```

Ce sont nos instructions. L'instruction cout dit au programme d'afficher tout ce qu'il y a entre les "<<" et le ";". endl veut simplement dire fin de ligne ("end line").

```
;
```

Quelques langages les utilisent. Pensez à la façon dont nous écrivons dans les langues tel le français. Nous terminons nos phrases avec un point final. De manière similaire, nous terminons l'action en cours avec un point-virgule. Ceci nous permet d'utiliser des caractères d'espacement, comme des espaces ou des retours à la ligne, où nous le souhaitons.

```
return 0;
```

Cela permet au programme de savoir que la fonction principale est terminée, qu'elle est finie. Passé ce point, l'exécution s'arrête.

Avec ces informations, voyons maintenant quelques autres exemples. Essayez ça :

```
#include <iostream>
using namespace std;

int main()
{
    //Création de 2 variables
    int a, b;
    a = 1;
    b = 2;

    //Affiche la somme de ces variables
    cout << "a + b = " << a + b << endl;
    return 0;
}
```

Ici, nous avons défini deux variables, a et b. Ce sont des int, c'est-à-dire des nombres entiers. Nous avons créé les deux, puis affiché leur somme.

Tout cela est bien beau, mais l'affichage sera toujours 3 à moins que nous ne changions le code, qui n'est pas très utile. Au lieu de ça, nous pouvons modifier le programme pour que l'utilisateur saisisse les variables, puis que le programme fasse la somme. Essayez ceci :

```
#include <iostream>
using namespace std;

int main()
{
    //Création de 2 variables
    int a, b;

    //Saisie et enregistrement de l'entrée de l'utilisateur
    cout << "Entrez le premier nombre : ";
    cin >> a;

    cout << "Entrez le second nombre : ";
    cin >> b;

    //Affiche la somme des variables
    cout << a << " + " << b << " = " << a + b << endl;
    return 0;
}
```

Cela vous permettra de lire les entrées de l'utilisateur et d'additionner les deux.

THE SCRATCH PATCH

Bases défensives



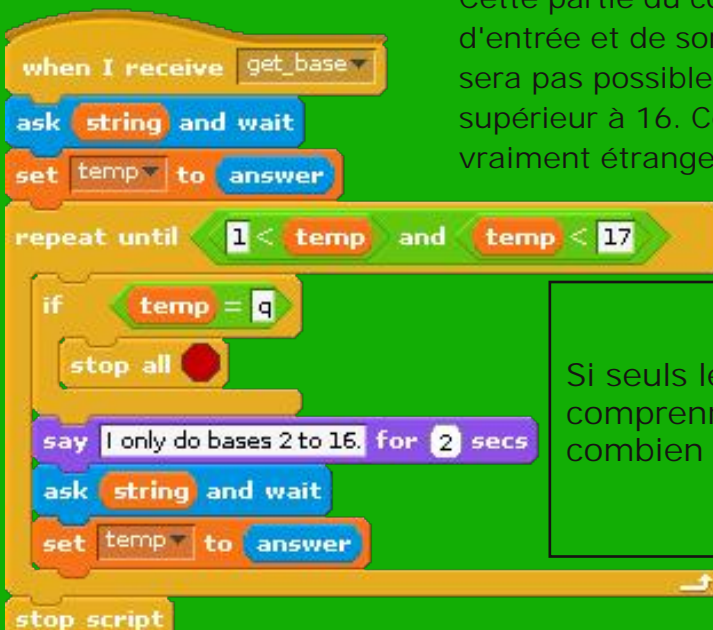
```
when I receive to_base_out
set result to 
repeat until input = 0
  set remainder to input mod base_out
  set result to join letter remainder + 1 of DIGITS result
  set input to input - remainder / base_out
stop script
```

Ce mois-ci, nous allons voir un peu de "programmation défensive" ; en d'autres termes, nous allons essayer d'écrire un programme capable de gérer des erreurs.

Cela convertit un nombre du décimal vers la base de sortie, celle-ci étant obtenue à partir des chiffres de la chaîne "DIGITS" (voir page suivante).

Désolé pour les scripts en désordre ce mois-ci - l'espace est restreint !

Le script "get_base"



```
when I receive get_base
ask string and wait
set temp to answer
repeat until 1 < temp and temp < 17
  if temp = q
    stop all
  say I only do bases 2 to 16. for 2 secs
  ask string and wait
  set temp to answer
stop script
```

Cette partie du code vous permet de fixer les bases d'entrée et de sortie que le programme va utiliser. Il ne sera pas possible d'entrer un nombre inférieur à 2 ou supérieur à 16. Cela permettra d'éviter des résultats vraiment étranges (et incorrects).

Blague hexa !

Si seuls les gens qui sont un peu FADA comprennent l'hexadécimal, combien de personnes le comprennent ?

(réponse page suivante !)

Début du programme principal ici.



Après que le script "get_base" ait obtenu vos bases d'entrée et de sortie, le script "get_number" s'assure que vous avez entré un nombre valide dans la base d'entrée choisie.

Le script "get-number" génère aussi la version décimale (base 10) de votre nombre, prêt à appeler "to_base_out".

```

when clicked
  set DIGITS to 0123456789ABCDEF
  say Bases for 1 secs
  forever
    set string to Base of input:
    broadcast get_base and wait
    set base_in to temp
    set string to Base of output:
    broadcast get_base and wait
    set base_out to temp
    set checknum to False
    repeat until checknum = True
      broadcast get_number and wait
    set input to round input
    broadcast to_base_out and wait
    ask result and wait
  
```

```

when I receive get_number
  ask Number to convert: and wait
  set temp to answer
  set counter to length of temp
  set input to 0
  set counter2 to 1
  set exp to 0
  
```



Essaie-le!

```

repeat until counter < 1
  set checknum to False
  repeat until counter2 = base_in + 1
    if letter counter of temp = letter counter2 of DIGITS
      set temp2 to 10 ^ of log of base_in * exp
      set input to input + temp2 * counter2 - 1
      set counter2 to 1
      change exp by 1
      change counter by -1
      set checknum to True
    else
      change counter2 by 1
  if checknum = False
    say Unrecognised digit! Capitals for Hex, please. for 2 secs
  stop script
stop script
  
```

Ce passage signifie que si vous êtes en base 4 (par exemple), les seuls chiffres autorisés seront 0, 1, 2 et 3.

Comme Scratch n'a pas de bloc "x à la puissance y" j'utilise des logarithmes :
 $10^{(\log x) \cdot y} = x^y$

Coincé ?
Récupérez tous les scripts sur :
<http://tinyurl.com/Scratchbases>

Réponse de la blague :
64218 personnes comprennent l'hexa.

Le mois dernier Le Repaire du Python a montré comment extraire des paramètres de configuration à partir d'un fichier texte externe. Une autre possibilité est présentée ce mois-ci.

L'utilisation des arguments de la ligne de commande apporte un contrôle précis sur un programme lors de son exécution.

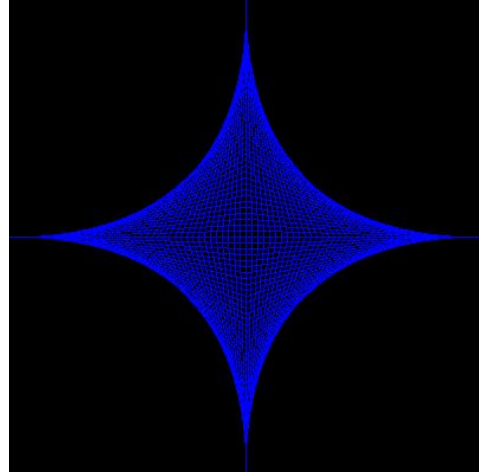
Ce moteur de rendu de lignes Pygame utilise argparse de Python : <http://docs.python.org/dev/library/argparse.html>
Lancez cette commande dans LXTerminal :

```
python lines.py -h
```

Cela affichera tous les arguments disponibles. Par exemple :

```
python lines.py -s 3 -t 4
```

va générer une surface plus large que celle par défaut (-s 2) et avec des lignes légèrement plus denses (-t 5). Expérimentez avec les options disponibles.



```
# générateur de lignes avec des arguments en ligne de commande
# Par Colin Deady - 03 octobre 2012

import os, pygame, argparse, sys
from pygame.locals import *

# initialise pygame (pour faire le rendu de l'image)
pygame.init()

# Définit deux fonctions qui seront utilisées :

# 1) fnAppend2Log va écrire une ligne dans un fichier journal
def fnAppend2Log( line2write ):
    logfile = open('lines.log', 'a')
    logfile.write(line2write + '\n')
    logfile.close()

# 2) fnPlotLines va afficher un quart de la surface.
# Utilise les coordonnées précédentes comme nouvelles coordonnées de début
def fnPlotLines(quarter, sX, sY, eX, eY, incSX, incSY, incEX, incEY ):
    fnAppend2Log(quarter + ' coordonnées du quart :')

# calcule et boucle sur les coordonnées de lignes
for i in range(0,iterations, args.step):
    nSX = sX + (incSX * i) # début X
    nSY = sY + (incSY * i) # début Y
    nEX = eX + (incEX * i) # fin X
    nEY = eY + (incEY * i) # fin Y

# dessine une ligne entre la paire de coordonnées.
pygame.draw.line(screen, (lineColour), (nSX, nSY), (nEX, nEY), 1)
```

PYTHON VERSION: 2.7.3rc2
PYGAME VERSION: 1.9.2a0
O.S.: Debian 7

TESTED!

```

# construit une chaîne pour le titre de la fenêtre et le journal
coordText = '('+str(nSX)+'/'+str(nSY)+'-
              ('+str(nEX)+'/'+str(nEY)+' )'
# rendu de l'image ligne par ligne (dure plus longtemps) ?
if args.renderlines == 'y':
    pygame.display.update();
    pygame.display.set_caption(coordText)

# envoie les coordonnées dans le journal
fnAppend2Log(coordText)

# retourne les coordonnées finales calculées
return (nSX, nSY, nEX, nEY);

# définit les arguments de la ligne de commande :
parser = argparse.ArgumentParser(description='Render shape')
parser.add_argument('-s', action='store', dest='scale', type=int,
                    default=2, help='Render size, default=2, 200x200px')
parser.add_argument('-t', action='store', dest='step', type=int,
                    default=5,
                    help='Diminuez l'écart pour des lignes plus denses')
parser.add_argument('-r', action='store', dest='renderlines',
                    choices=('o', 'n'), default='o',
                    help='Afficher par ligne (O) ou l'objet terminé (n)')
args = parser.parse_args()

# Définit les variables qui seront nécessaires
sz = 100*args.scale # taille en pixels horiz x vert d'un quart de l'image
iterations = sz +5 # nombre de lignes à calculer par quartier
lineColour = 0,0,255 # la couleur de la ligne à dessiner (bleu)

# ouvre un écran pygame sur lequel sera fait le rendu de nos objets
# la taille de l'image double de l'objet à générer car nous avons 4 quarts
screen = pygame.display.set_mode([sz*2,sz*2],0,32)

# Dessin des lignes, quartier, renvoyant les paires de coordonnées
# Les coordonnées de début sont celles de fin du dernier quart généré
sx, sy, ex, ey = fnPlotLines('Top left', sz, 0, sz, sz, 0, 1, -1, 0 )
sx, sy, ex, ey = fnPlotLines('Bottom left', ex, ey, sx, sy, 1, 0, 0, 1 )
sx, sy, ex, ey = fnPlotLines('Bottom right', ex, ey, sx, sy, 0, -1, 1, 0 )
sx, sy, ex, ey = fnPlotLines('Top right', ex, ey, sx, sy, -1, 0, 0, -1 )

# si le rendu de chaque ligne est supprimé, affichage de l'image finale
if args.renderlines == 'n':
    pygame.display.update();

# enregistre l'image générée dans un fichier
pygame.image.save(screen, 'lineimage.png')

# affiche le résultat pendant 10 secondes
pygame.time.wait(10000)

```

Essayez d'ajouter un argument supplémentaire pour qu'il soit possible de désactiver l'écriture dans le fichier journal. Cela va améliorer la durée du rendu. Indice : tout comme pour les arguments, vous devrez ajouter deux instructions If car l'écriture dans le journal est faite à deux endroits différents dans le code.

Quelques autres idées d'arguments de ligne de commande :

- Laisser l'utilisateur choisir le nom du fichier pour l'image de sortie
- Spécifier les couleurs de lignes et d'arrière-plan à partir d'une liste prédéfinie (noir, blanc, rouge, vert, bleu)
- Entrer en mode démo qui boucle sur le code, produisant des surfaces avec des couleurs aléatoires

L'heure des questions et des réactions

Q: En ce qui concerne la série Skutter, vous mentionnez un bras robot qui peut être monté. Pourriez-vous indiquer quel bras robot et où l'obtenir ?

Richard

A: Skutter est une série sur le long terme, dont la première partie a été publiée dans le numéro 1 du MagPi. Ce dernier contient des informations de base : "Le kit du bras robotique appelé OWI Edge est actuellement disponible chez Maplin electronics et il utilise une interface USB pour le contrôler." <http://www.maplin.co.uk/robotic-arm-kit-with-usb-pc-interface-266257>

Q: Serait-il possible de visualiser le mag sur un iPad dans un proche avenir ou aurais-je raté un épisode ?

John

A: Le site Issuu a commencé à prendre en charge HTML5, si bien que cela devrait maintenant fonctionner sur iPad et iPhone etc. Vous pouvez également télécharger le PDF et le visualiser dans un iBook. Nous travaillons actuellement avec un développeur sur une application Newsstand.

Merci d'avoir rendu une version imprimée disponible sur <http://www.modmypi.com>.

J'imprime toujours chaque numéro car je préfère lire sur papier plutôt qu'en ligne. Imaginez ma surprise quand j'ai découvert qu'il ne coûte que £2,49. Cela me coûte presque aussi cher d'imprimer 32 pages en couleurs moi-même !

Ian

Q: En tant que nouvel utilisateur de Raspberry Pi, je suis très

intéressé par votre excellent magazine. Je n'ai pas eu de difficultés à lire vos numéros 1 à 6 sur mon ordinateur sous Linux Mint 13. Imaginez ma déception lorsque j'ai trouvé que, sur les six numéros, je ne pouvais ouvrir que le numéro 3 sur mon Raspberry Pi. N'étant pas habitué au programme MuPDF utilisé sous Raspbian, j'ai pensé qu'il serait mieux de vérifier en l'installant sur ma machine Linux Mint. Sur celle-ci, chaque numéro du MagPi s'ouvre sans problème avec MuPDF. Ce n'est peut-être pas votre problème technique, mais je me suis dit que vous voudriez savoir qu'un nombre important et croissant d'utilisateurs de Raspberry Pi ne peut pas lire votre magazine.

Lloyd

A: Ce problème s'étend à beaucoup d'autres fichiers PDF. MuPDF semble bien fonctionner pour certains utilisateurs mais pas pour d'autres. Personnellement, j'ai supprimé MuPDF et j'utilise xPDF à la place, ce dernier fonctionnant avec tout. Vous pouvez effectuer cela avec les commandes suivantes :

```
$ sudo apt-get remove MuPDF
$ sudo apt-get install xPDF
```

The **MagPi**TM
editor@themagpi.com

The MagPi est une marque déposée de The MagPi Ltd. Raspberry Pi est une marque déposée de la fondation Raspberry Pi. Le magazine MagPi est réalisé collaborativement par un groupe indépendant de propriétaires de Raspberry Pi, et n'est en aucun cas affilié à la fondation Raspberry Pi. Il est interdit de reproduire ce magazine dans un but commercial sans l'autorisation de The MagPi Ltd. L'impression dans un objectif non commercial est autorisée conformément à la licence Creative Commons ci-dessous. Le MagPi n'est ni propriétaire ni responsable des contenus ou opinions exprimés dans les articles de cette édition. Tous les articles ont été vérifiés et testés avant la date de sortie mais des erreurs peuvent subsister. Le lecteur est responsable de toutes les conséquences, tant logicielles que matérielles, pouvant survenir suite à la mise en pratique de conseils ou de code imprimé. Le MagPi ne prétend pas détenir de droits d'auteur et tout le contenu des articles est publié sous la responsabilité de l'auteur de l'article.

Ce travail est placé sous les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions 3.0. Une copie de cette licence est visible sur :

<http://creativecommons.org/licenses/by-nc-sa/3.0/>



Ou envoyez un courrier à Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.