Comment allumer le Raspberry, et qu’il commande son extinction ?

# LTC2951 (ze best solution?)

<http://cds.linear.com/docs/en/datasheet/295112fb.pdf>

<https://www.raspberrypi.org/forums/viewtopic.php?f=107&t=113789>

<https://www.raspberrypi.org/forums/viewtopic.php?t=128019>

<https://www.raspberrypi.org/forums/viewtopic.php?t=117863>

# Simple (ze KISS solution?)

<https://raspberrypi.stackexchange.com/questions/4719/how-to-shut-down-rpi-when-running-headless>

<http://www.recantha.co.uk/blog/?p=13999>

<https://github.com/TonyLHansen/raspberry-pi-safe-off-switch>

# MOS-FET comme interrupteur

<http://www.locoduino.org/spip.php?article120>

Piloter un moteur

### La PWM : Qu’est-ce que c’est ?

* [La PWM : Qu’est-ce que c’est ? (1) - Application aux diodes électroluminescentes](http://www.locoduino.org/spip.php?article47)
* La PWM : Qu’est-ce que c’est ? (2) - Piloter un moteur
* [La PWM : Qu’est ce que c’est ? (3) - Changer la fréquence de la PWM](http://www.locoduino.org/spip.php?article202)
* [La PWM : Qu’est ce que c’est ? (4) - Monter en fréquence](http://www.locoduino.org/spip.php?article203)

Le 2 février 2016. Par : [Dominique](http://www.locoduino.org/spip.php?auteur5), [Guillaume](http://www.locoduino.org/spip.php?auteur2), [Jean-Luc](http://www.locoduino.org/spip.php?auteur1)

Dans le premier article concernant la [PWM](http://www.locoduino.org/spip.php?mot20), « [La PWM : Qu’est-ce que c’est ? (1)](http://www.locoduino.org/spip.php?article47) », nous avons abordé l’alimentation d’une [DEL](http://www.locoduino.org/spip.php?mot12) par une tension hachée. Nous allons maintenant aborder la commande d’un moteur à courant continu, c’est à dire le type de moteur qui équipe nos locomotives, par une [PWM](http://www.locoduino.org/spip.php?mot20).

La manière de s’y prendre diffère pour plusieurs raisons :

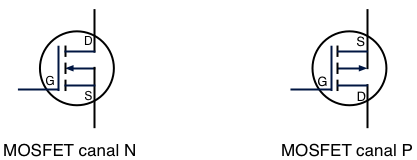
* les moteurs employés en modélisme ferroviaire ont une tension d’alimentation qui est plus élevée que la tension de l’Arduino, généralement 12V ;
* le courant nécessaire est également plus élevé, environ 10 fois, que ce que peut fournir une sortie de l’Arduino ;
* contrairement à une [DEL](http://www.locoduino.org/spip.php?mot12), un moteur est une charge dite *inductive*. C’est à dire que lorsque l’alimentation du moteur est coupée, le courant ne s’arrête pas instantanément de circuler. Il diminue progressivement.

La conséquence directe est qu’**une sortie PWM de l’Arduino ne peut pas commander directement un moteur**. Il est nécessaire d’amplifier le signal, à la fois en tension et en courant. L’amplificateur doit être adaptée au courant consommé par le moteur et doit fournir la tension nécessaire. Elle doit aussi être capable de suivre la fréquence de la PWM.

Deux possibilités s’offrent à nous : utiliser un transistor MOSFET de puissance ou bien un pont en H. Nous allons tout d’abord examiner la première solution.

### Amplification via un transistor MOSFET

Il existe 2 types de MOSFET, ceux à canal N et ceux à canal P. Les MOSFET sont des composants à 3 broches : le Drain (D), la Source (S) et la Grille (G). Voici le symbole des deux types de MOSFET.



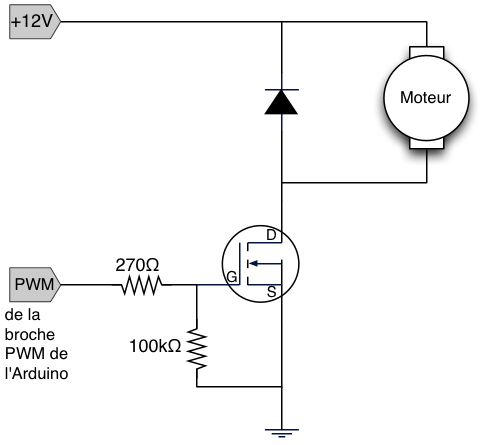
Dans l’emploi que nous allons en faire, on peut voir ces transistors comme des interrupteurs. D et S sont les deux bornes de l’interrupteur et la tension de G détermine la position de l’interrupteur. Contrairement à un interrupteur mécanique, les MOSFET on un sens. Pour un MOSFET canal N, la tension de S doit être plus faible que la tension de D. On connectera donc S côté masse et D côté +12V. Pour un MOSFET canal P, c’est l’inverse. S doit être connecté côté +12V et D côté masse. La tension qui est appliquée entre G et S détermine si l’interrupteur est fermé ou non. Un MOSFET possède une tension de seuil. Pour un MOSFET canal N si la tension entre G et S est plus grande que la tension de seuil, l’interrupteur est fermé (le MOSFET est passant) et le courant circule sinon l’interrupteur est ouvert (le MOSFET est bloqué). Pour un MOSFET canal P, si la tension entre S et G est plus petite que la tension de seuil, l’interrupteur est fermé et le courant circule.

Les transistors MOSFET sont extrêmement sensibles à l’électricité statique. Les manipuler sans précaution peut aboutir à les endommager voire à les détruire. Il est impératif de s’équiper d’un bracelet antistatique qui sera relié à la terre. Les stations de soudage possèdent une prise de terre destinée à recevoir le câble du bracelet antistatique. De manière générale, il est préférable de procéder de même pour les circuits intégrés. Si la plupart des composants à base de transistor MOSFET sont protégés contre l’électricité statique, le risque de casse existe.

Commençons par un MOSFET canal N. Un des modèles les plus répandus est l’IRF540N. Le courant maximum est de 33A et la tension maximum de 100V. très largement suffisant pour nos usages. Sa tension de seuil est comprise entre 2 et 4V. On peut donc en mettant 5V sur la grille fermer l’interrupteur de manière fiable.



On trouve de nombreux exemple de connexion pour l’Arduino, par exemple celui-ci :

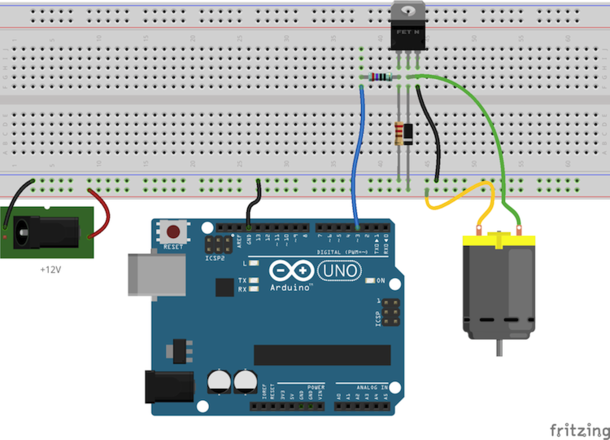


Vous pouvez remarquer la diode en parallèle avec le moteur, il s’agit d’une diode roue libre. Ce type de montage a déjà été présenté dans « [Les diodes classiques](http://www.locoduino.org/spip.php?article30) » pour une commande de relais. Comme cela a été dit plus haut le moteur est une charge inductive et lorsque le MOSFET cesse de conduire le courant ne s’arrête pas instantanément de circuler. Comme il doit aller quelque part, la diode lui offre un chemin. Le courant circule donc en boucle dans le moteur via la diode jusqu’à ce que la résistance du moteur le dissipe sous forme de chaleur. **En l’absence de diode roue libre ce courant provoquerait une augmentation de tension importante sur le Drain du MOSFET avec le risque de l’endommager**.

**La résistance de 100kΩ** a pour rôle de maintenir le MOSFET bloqué en tirant G à la masse tant que la broche de l’Arduino n’est pas programmée. En son absence G *flotterait* et selon la charge électrique présente le MOSFET pourrait devenir passant avec un démarrage intempestif du moteur.

**La résistance de 270Ω** évite que trop de courant ne soit tiré de la broche de l’Arduino. En effet, les MOSFET ont sur G l’équivalent d’un condensateur de valeur assez importante. Sans cette résistance le courant serait ponctuellement important et dépasserait les capacités de l’Arduino. Avec 270Ω, le courant instantané ne dépasse pas 19mA.

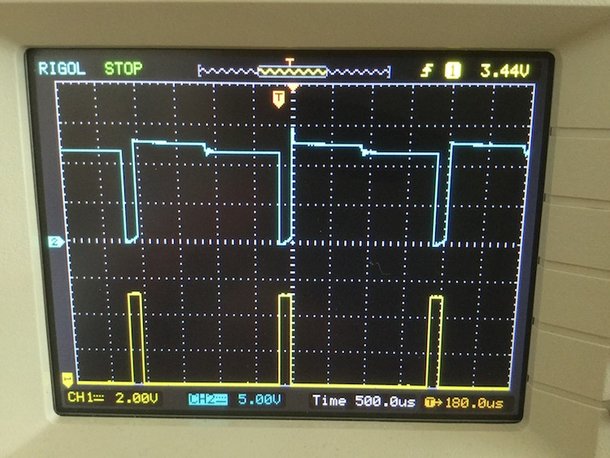
On choisit la broche 3 comme PWM. Le montage sur breadboard est le suivant :



Tout se passe très bien avec la fréquence par défaut de la PWM de l’Arduino, soit 490Hz sur la broche 3. Ici la PWM est réglée à 20, soit un rapport cyclique d’un peu moins de 8%. Le programme de test est réduit à sa plus simple expression.

1. void setup()
2. {
3. [analogWrite](http://www.locoduino.org/geshi/redirect.php?language=arduino&search=analogWrite)(3, 20);
4. }
6. void loop()
7. {
8. }

Le signal jaune est pris sur la broche 3 de l’Arduino et le signal bleu entre le MOSFET et le moteur, c’est à dire au niveau de drain. Quand le signal jaune est à l’état haut, le MOSFET est passant et le drain est tiré à 0. Le moteur est alimenté. Quand le signal jaune est à l’état bas, le drain flotte et le moteur n’est pas alimenté. Le MOSFET agit donc comme un interrupteur commandé par la PWM de l’Arduino.



Dans un prochain article, nous augmenterons la fréquence de la PWM afin de remédier à deux problèmes :

* **le bruit**. À 490Hz, le moteur émet un Si légèrement faux et assez agaçant ;
* **le problème de dissipation thermique** dans le moteur à cette fréquence. Ce problème est exposé sur mon blog : [Tension hachée et pertes par effet Joule](http://modelleisenbahn.triskell.org/spip.php?article44). Si vous exploitez en N ou en H0e, ce problème vous concerne et n’est pas à prendre à la légère.

# Bistable simple 2 transistors

<https://www.astuces-pratiques.fr/electronique/bistable-a-deux-transistors-schema>

Un montage bistable permet de conserver un état après un appui sur l'un ou l'autre des boutons poussoirs sur le montage. Le schéma présenté ici comprend deux boutons poussoirs et deux transistors astucieusement montés. Un appui sur un bouton déclenche l'état haut (qui restera après avoir relâché le bouton) et un appui sur l'autre bouton déclenche l'état bas. Les deux états sont "stables" (subsistent sans aucun appui), d'où le nom du montage : le bistable (= deux états stables possibles).

Abordons maintenant le schéma du montage bistable.

**Montage bistable : le schéma**

Voici le schéma du bistable à deux transistors, basé sur un principe proche du thyristor :

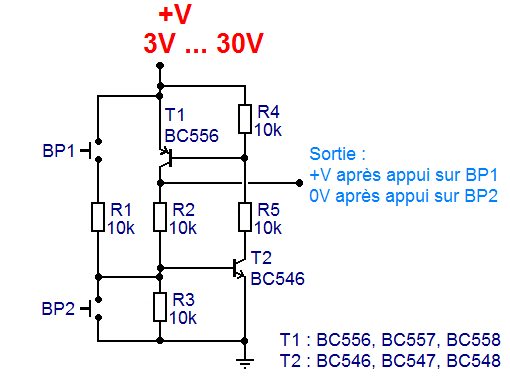
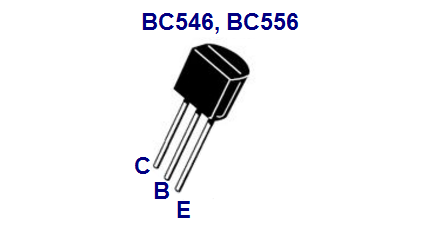


Schéma du bistable à deux transistors (NPN et PNP)

La tension d'alimentation n'a pas grande importance. T1 et T2 peuvent être des transistors assez quelquonques, comme les classiques BC547 et BC557, ou encore 2N3904 et 2N3906, etc.



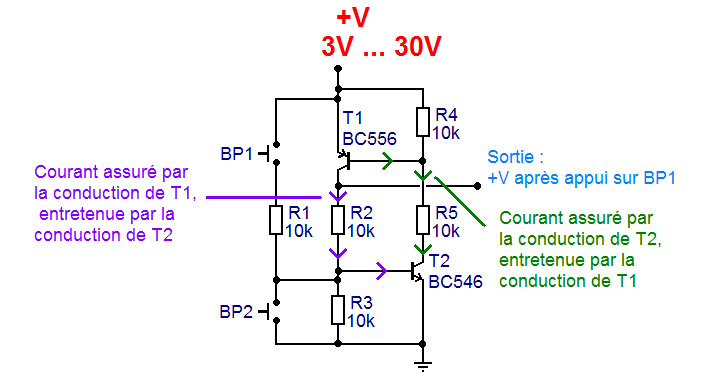
Transistors BC546, BC547 et BC556, BC557

**Bistable : principe de fonctionnement**

Voyons le fonctionnement de ce montage bistable. Pour comprendre le schéma du bistable, imaginons que les deux transistors (T1 et T2) soient bloqués. T1 et T2 se comportent comme des interrupteurs ouverts. Aucun courant ne circule dans aucune branche du circuit. Cet état subsiste puisque rien ne vient déclencher la conduction de l'un ou l'autre des transistors. La sortie est donc à l'état bas (0V). Si on souhaite le niveau logique (binaire) complémentaire, on prendra le collecteur de T2 qui lui, affiche une tension haute (tension d'alimentation) par rapport à la masse.

Le rôle de R3 et R4, entre base et émetteur, peut être négligé. Ces résistances servent juste à assurer un blocage fiable des transistors.

Imaginons un appui sur BP1, même bref. Lorsque BP1 est appuyé, le transistor T2 est passant : sa base reçoit un courant qui traverse R1. R3 évite un déclenchement intempestif du transistor T2. Comme T2 est passant (saturé vu l'ordre de grandeur des courants de base et de collecteur), du courant circule dans R4 et R5, ainsi que dans la base de T1. T2 forme un quasi court-circuit entre son émetteur et son collecteur. T1 est ainsi passant (lui aussi saturé), ce qui fait qu'il fournit un courant de base pour T2 par l'intermédiaire de R2. Lorsqu'on relâche le bouton poussoir BP1, T1 continue d'être passant grâce à T2 qui continue d'être passant grâce à T1. La sortie est donc au niveau de l'alimentation. Si on souhaite le niveau logique inverse, le collecteur de T2 offre un niveau logique bas (0V). Les deux transistors restent ainsi passants aussi longtemps qu'on ne coupe pas l'alimentation du circuit.



Conduction entretenue réciproquement des deux transistors : état stable du bistable

Imaginons maintenant un nouvel appui sur BP2. BP2 est en parallèle avec la base du transistor T2. Si on appuie sur BP2, on court-circuite la base de T2 et on force ainsi le blocage de T2. Le blocage de T2 coupe le courant dans R5 et ainsi le courant de base de T1. Lorsqu'on relâche l'appui sur BP2, comme T1 est bloqué, plus rien ne permet d'alimenter en courant la base de T2. Le bistable est revenu à l'état décrit au début (T1 et T2 = interrupteurs ouverts). La boucle est bouclée. Et ainsi de suite...

Si on appuie sur les deux boutons poussoirs à la fois, c'est BP2 qui est "prioritaire" puisqu'il force le court-circuit entre la base et l'émetteur de T2, obligeant ainsi son blocage.

**Le bistable : une idée, un concept**

Le bistable présenté ici du côté de l'électronique est un circuit qui maintient deux états stables possibles qui subsistent après l'action manuelle (ou électronique). Le changement ne peut être réalisé que par une action extérieure (ici, un appui sur bouton poussoir).

Le bistable est la base des mémoires vives en électronique puisque la "donnée" binaire subsiste et peut être consultée (lecture de l'état de sortie) autant de fois qu'on le souhaite.

Imaginons qu'on puisse stocker une pièce de monnaie dans une boite au choix (la boite A ou la boite B).



Bistable : 2 boites et une pièce de monnaie

Une fois placée dans une boite, la pièce de monnaie reste dans sa boite tant que personne n'y touche. L'état du système "boite A, boite B et pièce de monnaie" reste stable dans l'état : "la pièce est dans la boite A". De même si on déplace la pièce dans la boite B, le système restera stable dans l'état : "la pièce est dans la boite B". **Cela illustre le concept du bistable**. A la différence du montage électronique, le bistable à deux boites n'a pas besoin d'alimentation électrique pour conserver sa donnée. C'est une forme de mémoire binaire.

**Autres montages bistables**

Il existe de nombreux autres montages bistables à transistors. Certains utilisent deux transistors identiques (des NPN par exemples). On trouve encore des montages bistables basés sur des portes logiques, des bascules, des ampli op, des relais etc.

Le montage présenté ici repose sur un transistor NPN et un transistor PNP dont les conductions s'entretiennent une fois qu'elles ont été amorcées. Ce fonctionnement est assez proche du thyristor dont le schéma est proche de deux transistors (un NPN et un PNP) imbriqués l'un à l'autre en terme de jonctions PN.

# Alim du pi dans la voiture – extinction

<http://www.magdiblog.fr/boa-pi-carjukebox/9-alimentation-du-pi-dans-la-voiture-33-extinction/>

<http://aws-cf.caradisiac.com/prod/photos/2/8/1/377281/4990718/big-49907186f8.jpg?v=6>

<http://aws-cf.caradisiac.com/prod/photos/2/8/1/377281/4990717/big-4990717cdc.jpg?v=6>

Donc, premier schéma, simplifié, pour bien comprendre le principe :

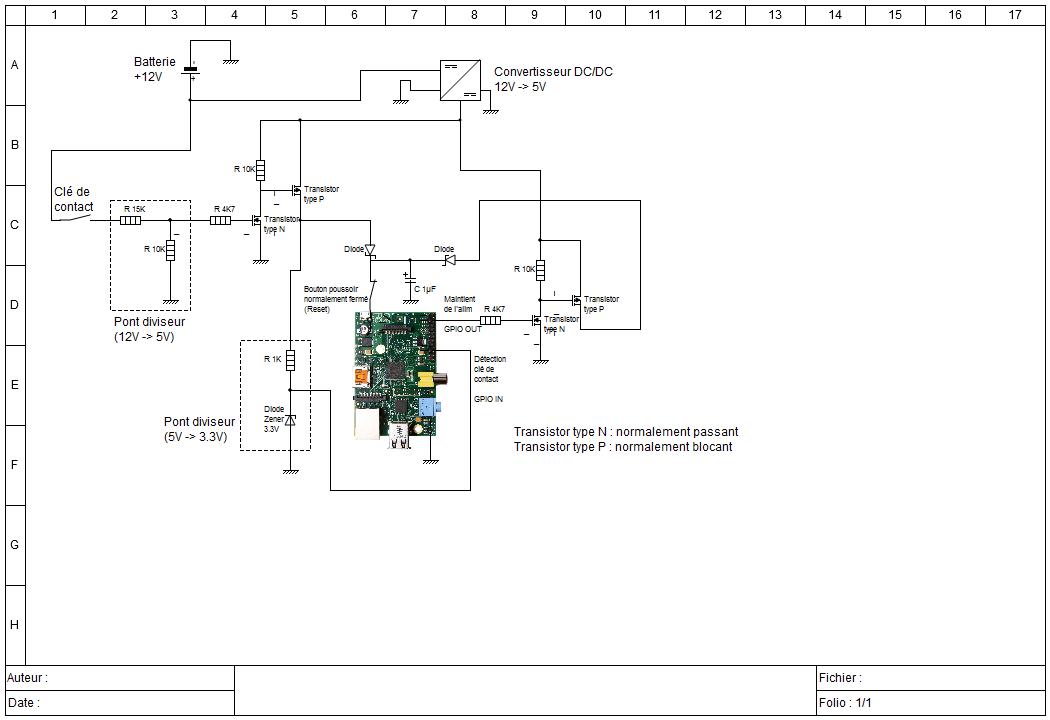
…/…

Le GPIO en mode IN détecte si la clé de contact et bien enclenchée. Lorsque ce n’est plus le cas, son entrée tombe à 0. On envoie alors la commande HALT (après une petite tempo) pour éteindre le PI

Le GPIO en mode OUT a été mis à 1 au démarrage du PI. Il restera dans cet état jusqu’à son extinction. Une fois le PI éteint, le GPIO tombe alors à 0, coupant ainsi l’alim du PI

Mais vous vous en doutez, ce n’est pas si simple. Au moins deux choses ne vont pas : on ne peut pas envoyer du 5V sur une entrée GPIO et on ne sait pas commander directement un relais avec une sortie 3.3V. Il faut donc un peu ruser ….

J’ai donc remplacé les deux relais par 2 couples de transistors qui ont la même fonction. Avantage, ça consomme moins et ça coûte moins cher !  
J’ai ajouté des diviseurs de tension pour adapter le voltage au PI.  
Voici donc le schéma détaillé, le vrai cette fois-ci !

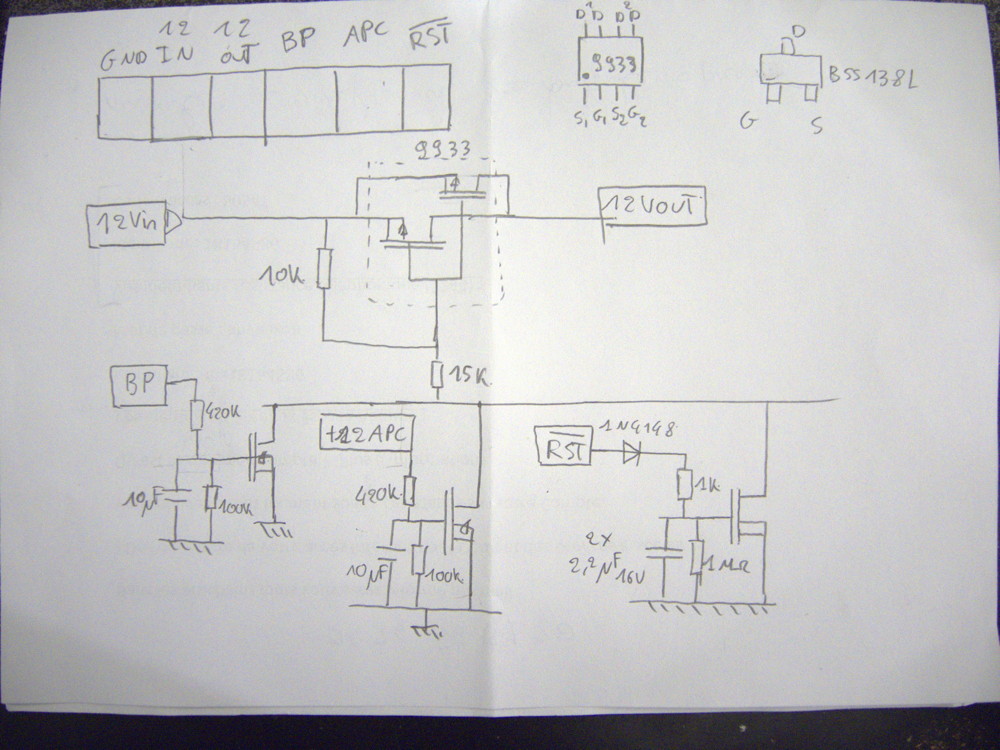


# pilotage alimentation à l'extinction

<https://www.raspberrypi.org/forums/viewtopic.php?f=65&t=141302>

mon RPI sera branché sur une batterie de voiture, l'alimentation se fera par le +12V, mais la commande d'alimentation par le +12V "après contact" (APC)  
lorsque que je coupe l'APC, l'alimentation +12V du RPI se coupe, logique.  
J'ai donc prévu un auto maintien, que le RPI vient couper lui même lorsqu'il s’éteint. Je m'attendais à ce qu'il tombe son reset (RUN) à 0, mais il apparait que non, le signal RUN reste à 1 (3V3)  
existe-il un moyen de commuter ce reset à l'extinction, ou bien de commuter un autre signal ?  
  
note : j'ai vu passer des choses pour activer le RPI avec un bouton, mais ce n'est pas ce que je veux. Je veux que le RPI coupe son alimentation lorsqu'il s’éteint, l'alimentation reviendra avec l'APC.  
je précise que je parle Python, mais très peu Linux.  
  
détails de mon projet : je compte monitorer une voiture de collection avec un ordinateur de bord discret (conso, vitesse, t°, etc) en installant un réseau CAN et des cartes d'IO CAN. J'ai donc une alimentation avec commande pilotée par plusieurs entrées, dont le RPI, l'APC et un bouton pour pouvoir démarrer sans mettre l'APC.

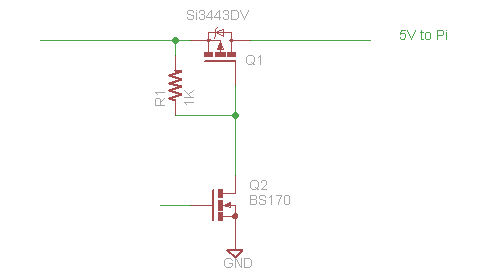
…/…

merci pour ta réponse, c'est exactement ce que je cherchais.  
Paradoxalement, cette manip à planté mon RPI, j'ai cru que j'allais pouvoir formater ma carte mémoire qui contient des infos non encore sauvegardées :-/ (je sais c'est pas bien)  
impossible de redémarrer avec le fichier dt-blob.bin généré, mais par contre la pate que je voulais lever passait bien à 1 dés la mise sous tension !  
Mais entre lignes se trouvait une solution alternative que je choisis pour le moment, et au pire si ça me pose problème j'y reviendrai plus tard : utiliser le TXD, qui monte à 1 dés que le RPI démarre, et tombe à 0 quand il fait un shutdown.  
  
Voici le schéma de mon gestionnaire d'alim HARD, je n'ai pas de scanner, j'ai fait un photo à l'ancienne ;-)  
  
  
LE 9933 est un double MOS dans un boitier SO8. j'ai mis les 2 MOS en //. Il est donné pour 20V de VGS et 3,5A environ par MOS. j'ose espérer que sans plage de dissipation, il tienne suffisamment pour mon usage (il doit alimenter aussi les modules d'IO au bout du bus CAN)  
le BSS138 est un petit MOS en SOT23 qui tient 50V, je ne tire pas de courant dessus.  
l'entrée /RST a une diode pour éviter le retours sur la commutation de la pate de commande, en l’occurrence TX.  
le 2x 2,2µF x 1MΩ font une constante de temps de 4,4 secondes, ce qui me fait un retard à l'extinction de 5 secondes environ.  
j'ai oublié de dessiner la diode de protection SMT 20V en entrée pour limiter les surtensions, fréquentes sur les (vieilles ?) voitures

# Infos à extraire !

<https://www.raspberrypi.org/forums/viewtopic.php?f=37&t=102015>

**Power-Switch**  
We need a power switch to disconnect the Pi from the power. Real men use relays, but were going to use an electronic switch using a P-channel MOSFET, driven by an N-channel MOSFET.



*Power Switch*

# +++ Overlay for on/off push-button controller.

<https://www.raspberrypi.org/forums/viewtopic.php?f=107&t=113789>

<https://www.raspberrypi.org/forums/viewtopic.php?f=41&t=114975>

Pas simple, mais excellent !

<https://www.raspberrypi.org/forums/viewtopic.php?f=65&t=141302>