Comment alimenter un Arduino ou une Raspberry Pi

Par Kilian Sanfins

Introduction

Quelque soit le projet, il sera nécessaire de lui apporter de l'énergie d'une manière ou une autre. Se pose alors la question de l'alimentation électrique.

- Le projet a-t-il accès à une prise secteur?
- Doit-il évoluer de manière autonome?
- Pendant combien de temps?
- Quel est son rythme de fonctionnement? Peut-il se mettre en état de veille pour consommer moins?

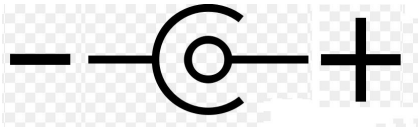
Il n'y a pas de solution unique, chaque projet a des contraints différentes et va influencer sa manière de s'alimenter. Cependant il est possible de dégager les principales manières de s'alimenter.
Nous allons d'abord voir où alimenter un Arduino/RaspberryPi, puis les différentes technologies pour alimenter avec ou sans batteries et enfin comment diminuer la consommation des circuits, en particulier pour des autonomies de très longues durées.

Alimentation des cartes

Arduino

L'Arduino est équipé d'un régulateur de tension qui s'assure que la tension délivré au microcontrôleur sera toujours 5V (c'est le composant juste au dessus de la prise jack, avec 3 pattes d'un côté et une plus grosse de l'autre côté). Ainsi l'Arduino peut être alimenté par 4 entrées:

- La prise Jack (1 sur l'image ci dessous), elle alimente le régulateur de tension interne à l'Arduino. La tension recommandée est de 7 à 12V et de 6 à 20V maximum. Le courant recommandé est compris entre 0.5 et 2A.
  Voici le symbole de la prise Jack:

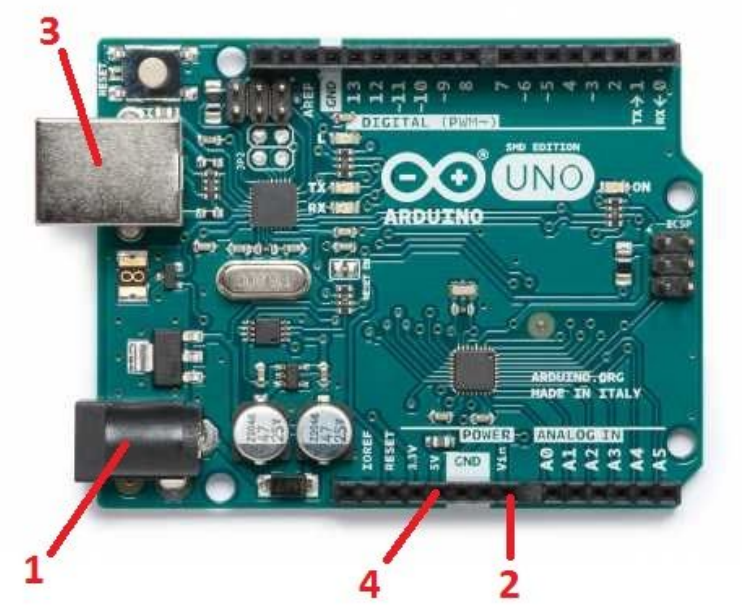


  Il renseigne que la borne positive se trouve au milieu tandis que la borne négative se trouve à l'extérieur. Si jamais le connecteur jack mâle n'indique pas lequel des 2 fils est le positif ou le négatif, il est possible d'utiliser le multimètre pour tester la continuité et savoir comment souder la prise.
  Attention, cette solution n'existe pas pour l'arduino nano, puisqu'il n'y a pas le connecteur! Pour cela il faut utiliser la solution qui suit.

- La pin Vin et GND (2), Vin est connecté directement à l'entrée du régulateur, GND est la masse commune. Les recommandations d'alimentations sont les mêmes que pour la prise Jack.

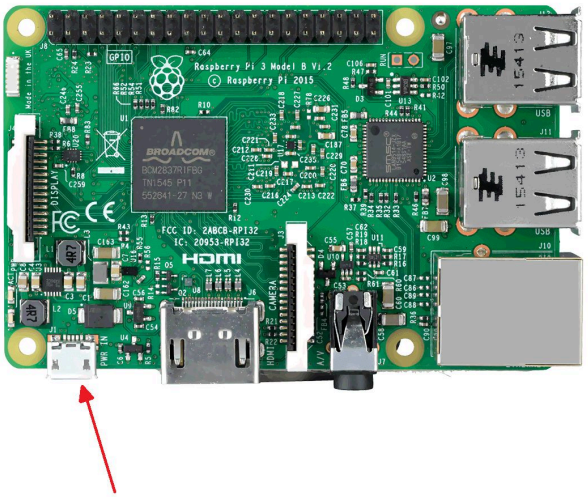
- Par USB (3), qui fournit une source de tension de 5V directement au microcontrôleur (en court-circuitant le régulateur de tension donc). Cependant il existe une protection qui limite le courant à 500 mA (pour éviter de griller le port USB du PC).

- Directement sur le pin 5V et GND (4), si on est sûr que la source de tension utilisée pour générer les 5V est précise (sinon le microcontrôleur sera grillé)



Raspberry Pi

Le Raspberry Pi est alimenté par un port micro USB 5V. Il n'y a pas de régulateur embarqué, donc une tension supérieur à celle ci, et la Raspberry Pi est endommagée! Il faut cependant que la source de tension puisse fournir suffisamment de courant, car la HDMI, la wifi, etc… consomment énormément d'énergie. Il est recommandé d'avoir une source d'alimentation qui puisse fournir au moins 2.5A pour éviter d'avoir des problèmes.
Le +5V des GPIO du Raspberry Pi est directement connecté à la sortie du port micro usb, donc il est également possible de l'alimenter par là!



Circuit pour l'alimentation

Pré-requis

Avant de commencer, je vais m'assurer qu'il n'y ait pas de méconception sur ce que représente la tension, le courant ou la puissance.
2 cas peuvent être mentionné, celui d'un tension continue ou variable (en particulier ici alternative):

- **Tension continue**

La puissance s'exprime comme le produit de la tension et du courant:

$$P = U \times I$$

- **Tension alternative**

La puissance instantanée vaut le produit de la tension et du courant à un instant t:

$$p_i = u \times i$$

Cela sous entend que pour un courant continu et positif et une tension alternative alors la puissance sera également variable (toujours positive, négative cela voudrait dire que l'on récupère de l'énergie). Cependant, combien absorbe-t-on en moyenne? En effet on mesure généralement la consommation d'un appareil électrique par sa puissance moyenne absorbée. Pour ce faire la puissance moyenne s'écrit:

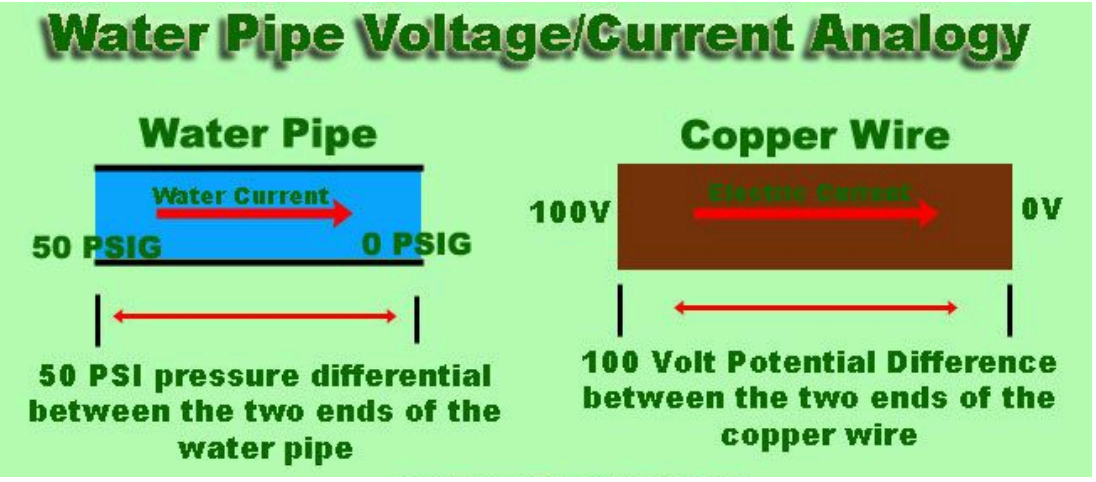
$$P = \int p_i = \int u \times i$$

Ce qui représente la moyenne de la puissance instantanée. De manière pratique, pour une tension sinusoïdale on a:

$$P = \frac{\sqrt{2}}{2} P_{instantanée\ max} = \frac{\sqrt{2}}{2} U_{max} I_{max}$$

- **Comment se représenter intuitivement la tension et le courant?**

Imagines toi un tuyau d'eau:



- Le courant est la quantité d'eau qui parcourt le tuyau à tout instant. Pour un cable c'est la quantité d'électrons.

- La tension est la différence de pression entre le point A (début du tuyau) et le point B (fin du tuyau). L'énergie potentielle est plus grande au point A que au point B, l'énergie coule donc du point A vers le point B. Dans un câble électrique, la différence de potentiel est électrique et ce sont les électrons qui essayent de circuler du point A au point B.

- La tension est donc la différence de potentiel (électrique) entre 2 points. Si la différence de potentiel (ou la tension) est suffisamment élevée, les électrons se déplaceront et forment alors un courant. Mais ces électrons n'ont pas forcément besoin d'être dans un câble pour se déplacer, si 2 câbles sont dénudés ou non protégés et que la tension est suffisante, les électrons se déplaceront même dans l'air en formant un arc électrique pour passer d'un câble à un autre. Ce problème est d'autant plus important avec les microcontrôleurs, où une tension trop haute (5v des fois!) suffit de générer un arc électrique à travers un transistor et endommager à tout jamais le microcontrôleur.

- Même si le courant représente la quantité d'électrons qui traversent un cable, nous utilisons que très rarement rarement des sources de courant, c'est à dire que nous ne forçons pas un courant dans un cable, mais généralement nous utilisons des sources de tension, qui maintiennent une tension donnée au borne du générateur (chargeur de téléphone 5v, prise 220v, allume-cigare de voiture 12v, etc…).
  Que se passe-t-il pour le courant alors?
  La fameuse formule $U = RI$ nous donne la réponse. Si la tension est fixée par le générateur et que la résistance est donnée, alors le courant I est naturellement imposé.
  En pratique cela veut dire que le système va absorber de lui-même le courant nécessaire à son fonctionnement, il faudra seulement s'assurer de pouvoir le lui fournir, sans quoi il sera sous alimenté avec de nombreux malfonctionnement à la clé.

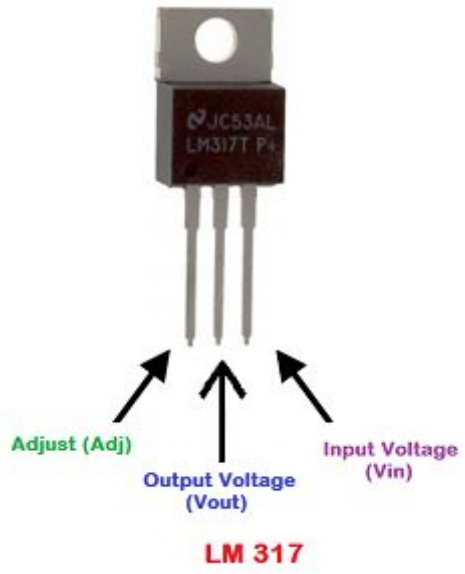
Si $R \simeq 0$, c'est à dire un court circuit, avec un cable par exemple, alors pour une tension U donnée, on a : $I = \frac{U}{R} = \frac{constante}{\simeq 0}$ qui devient très très grand. C'est la raison pour laquelle le disjoncteur saute sur le 220v!

Le transformateur

L'avantage d'avoir une tension d'entrée alternative est qu'il est possible d'utiliser directement un transformateur pour abaisser la tension. En sortie du transformateur la tension est toujours sinusoïdale mais elle est séparée du signal en entrée. Leurs masses ne sont pas connectées, ce qui évite que par hasard le 220v ne remonte dans le circuit à basse tension.
On utilise alors généralement un pont de diodes pour rectifier le signal (c'est à dire le rendre uniquement positif) et un condensateur pour lisser le signal en sortie et ainsi obtenir une tension continue.
Ce genre de circuit très basique est implémenté dans des chargeurs de téléphone premier prix, ils sont certes fonctionnels mais de nos jours pour quelques euros de plus, il est possible d'avoir un circuit plus robuste qu'un "simple" transformateur (le transformateur conserve cependant sa place dans nos circuit car c'est pour le moment le seul moyen pour garantir un séparation des masses).

Convertisseur linéaire

Un transistor est amplificateur de courant et grâce à ce principe un convertisseur linéaire peut abaisser une tension. Un convertisseur linéaire connu est le LM317, qui peut prendre en entrée une tension maximum de 40V et d'abaisser la tension en sortie (et pas la relever!) de 1.25V à 37V (courant maximum en sortie 1.5A) en fonction de la résistance qu'on lui a adjoint. Le rendement de ces convertisseurs décroissent très rapidement avec la différence de tension entre la tension d'entrée et la tension de sortie (pour faire court, plus on lui demande d'abaisser la tension, moins son rendement est bon).



Cependant c'est la solution la plus économique et pour une faible réduction de tension cela est parfaitement adapté (on les retrouve sur les arduinos d'ailleurs!)
Ces convertisseurs chauffent également très rapidement avec l'augmentation de courant, donc il faut bien penser à le refroidir ou mettre un radiateur!

Circuit hacheur

Ce système consiste à avoir en entrée une tension continue et de la hacher (on génère un signal carré) et de la lisser (avec un condensateur) pour abaisser la tension. Son rendement est supérieur à 90% et on le retrouve de nos jours dans les ordinateurs, chargeurs de téléphone et pc portable etc…
La source de tension continue provient soit directement d'une batterie, soit d'un transformateur dont la sortie a été lissée.
Le HLK-PM01 (3.3V), HLK-PM03(5v) et le HLK-PM12 (12V) sont des hacheurs compact et peu cher qui prennent en entrée 220v et sortent une tension utilisable pour un microcontroleur.

Hacheur HLK-PM12 qui convertit le 220V en 12V

Batterie

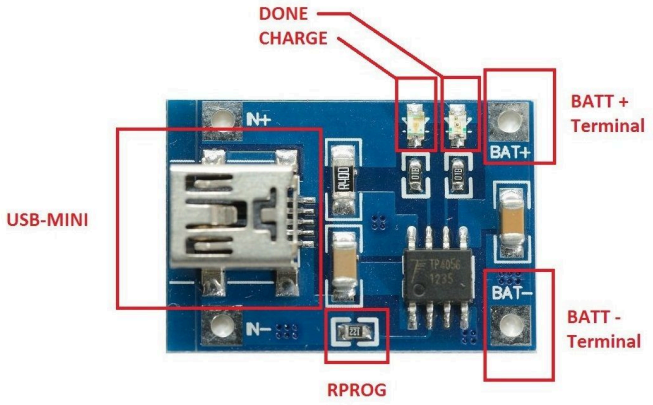
La capacité d'une batterie correspond à la quantité de courant qu'elle peut débiter par unité de temps. L'unité est généralement le mAh. Il y a également le taux de décharge, qui intervient lorsqu'on fait appel à de fort courant, mais ceci ne sera pas traité ici.

Si notre application consomme 1A ou 1000mAh à 5V et que la capacité de la batterie est de 3000mAh à 5V alors il sera possible de tenir 3h. Si maintenant elle consomme 1000mAh à 2.5V, et que la capacité de la batterie est la même alors il ne sera possible de ne tenir que 6h.

$$Autonomie = \frac{Tension\ de\ la\ batterie * Capacité\ de\ la\ batterie}{Tension\ d'alimentation\ du\ système * Courant\ consommé}$$

Il ne faut pas oublier que sur la fin de la capacité d'une batterie, elle ne fournit plus autant de tension, ce qui peut impacter les performances voir le fonctionnement du microcontrôleur.

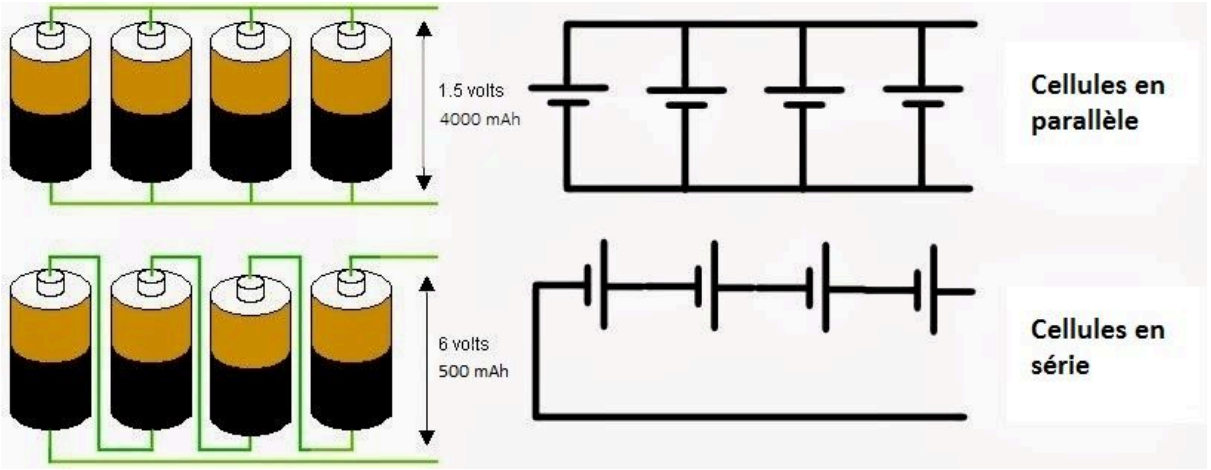
De plus en plus on trouve des batteries au lithium, à cause de leur meilleur capacité à stocker de l'énergie. Il existe des petits circuits qui assurent la charge et la décharge de la batterie, comme par exemple le TP4056:



Ce circuit est simple d'utilisation, un connecteur mini-usb (on en trouve avec micro usb) pour charger la batterie et 2 pins pour se connecter à l'alimentation et un indicateur led lorsque la batterie est en charge ou pleine, le tout pour un coût inférieur à 3$.
Cependant ce circuit ne prend en charge qu'une seule "cellule", qui fournit entre 4.2V et 3.7V.

Association de batterie

Une règle générale est qu'il faut toujours associer des batteries de même tension et capacité ou sinon elles se dégradent beaucoup plus vite.
Pour augmenter la tension, il suffit de connecter les batteries en série, tandis que pour augmenter la capacité total, il faudra les connecter en parallèle.
En associant les batteries en série et en parallèle, il est possible d'obtenir la tension et la capacité souhaitée.



Augmenter l’autonomie

Comme je l’ai mentionné précédemment, l’autonomie est dûe à 2 choses, une plus grande capacité de la batterie et une consommation réduite du système.
Augmenter la taille de la batterie est une solution mais on perd au fur et à mesure la portabilité initialement souhaitée. Si son mode de fonctionnement le permet, le prochain point abordé va couvrir différents moyens comment réduire la consommation.

Réduire la tension d’alimentation

Et oui alimenter son arduino avec une tension plus faible (mais toujours supérieur au minimum indiqué, sinon il s’éteindra) consomme moins d’énergie. Cependant alimenter en 3.3v l’arduino aura pour effet de réduire sa tension de sortie maximum pour les pins à 3.3V!
Pour certains capteurs cela ne pose pas de problème (et est même requis) pour d’autres ils ne pourront plus communiquer avec l’arduino.

Réduire la fréquence d’horloge du microcontrôleur

Réaliser moins de cycle par seconde réduit la consommation d’énergie mais réduit également les performances maximum. Cependant dans beaucoup de cas, cela n’aura que peu d’impact sur fonctionnement!
Pour cela il suffit d’écrire ces lignes dans le setup():

```
void setup()
{
  CLKPR = 0x80; // Active le changement de fréquence de l’horloge
  CLKPR = 0x08; // Définit la fréquence
}
```

Ces lignes agissent comme un diviseur de l’horloge interne à l’arduino, par puissance de 2^n avec n étant inférieur ou égale à 8, on note le diviseur par sa valeur hexadécimale dans la variable CLKPR, on obtient alors le tableau suivant:

CLKPR	Fréquence du microcontrôleur	Courant absorbé (mA)	Puissance consommée (mW)
0x00	16 Mhz	49	245
0x01	8 Mhz	43	215
0x02	4 Mhz	42	210
0x03	2 Mhz	37	185
0x04	1 Mhz	37	185
0x05	500 kHz	37	185
0x06	250 kHz	37	185
0x07	125 kHz	37	185
0x08	62.5 kHz	37	185

La fonction sleep()

L'arduino autorise de passer en mode sommeil et être réveillé par différentes conditions:

- Au travers de l'activation d'un pin interrupt
- Lors de la réception d'une donnée par le port série
- Par un timer interne, l'arduino se réveille à interval régulier, exécute une opération et se rendort
- Utilisation du watchdog timer pour réaliser la même fonction que précédemment, elle permet simplement de réduire encore plus la consommation.

A l'instar d'un humain, le microcontrôleur ne peut rien faire pendant son sommeil et prends un certain temps à se réveiller et à rétablir ses fonctions. De même, plus il tombe dans un sommeil profond, plus il perdra des fonctionnalités (pensez la différence entre sieste et gros sommeil). Cependant c'est un mode de fonctionnement très intéressant lorsque l'on sait que le système ne fera rien la majorité du temps ou en attente d'un signal puisqu'il permet d'économiser énormément d'énergie (pensez à une télécommande de télé qui ne fait rien jusqu'à ce que l'on appui sur un bouton, envoie le signal et se remet en veille… d'ailleurs même la télé se met en mode veille jusqu'à recevoir le signal de la télécommande pour s'allumer!)

Comme précisé précédemment le microcontrôleur autorise plusieurs modes de sommeil, désactivant plus ou moins les fonctions:

Table 9-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators		Wake-up Sources						
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X	
Power-down								X(3)	X				X	
Power-save					X		X(2)	X(3)	X	X			X	
Standby(1)						X		X(3)	X				X	

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

Plus on désactive des fonctions, moins on consomme d'énergie.

Pour activer le mode sommeil nous allons installer une bibliothèque, nommée LowPower, qui va gérer toute la partie difficile.
Il est possible de télécharger la bibliothèque ici.

Utilisation de la bibliothèque LowPower

La bibliothèque offre la possibilité de désactiver le module ADC (qui gère la conversion analogique/numérique), le timer0, timer1 et timer2, le module SPI (qui gère la liaison SPI), le module UART (qui gère la liaison Série) et le module TWI (pour twin wires, qui correspond à la liaison i2c). Plus on désactive de module, moins l'arduino consomme d'énergie. Il faut bien se rappeler que tous ces modules se réveilleront lorsque la condition de réveil sera valide, ils ne sont pas désactivés ad vitam eternam!
J'aurais tendance à recommander à tout désactiver par défaut, et si vraiment il te faut un module, il te semblera évident lequel laisser actif!
La bibliothèque propose aussi d'endormir l'arduino pendant différents délais:

- SLEEP_15MS -- 15ms de sommeil
- SLEEP_30MS -- 30ms de sommeil
- SLEEP_60MS -- 60ms de sommeil
- SLEEP_120MS -- 120ms de sommeil
- SLEEP_250MS -- 250ms de sommeil
- SLEEP_500MS -- 500ms de sommeil
- SLEEP_1S -- 1s de sommeil
- SLEEP_2S -- 2s de sommeil
- SLEEP_4S -- 4s de sommeil
- SLEEP_8S -- 8s de sommeil
- SLEEP_FOREVER -- sommeil infini jusqu'au reset du WatchDog Timer

Pourquoi autant de délai? Il est utile, surtout pour les objets connectés (pensez thermomètre, etc..) de faire des relevés réguliers, ou de vérifier régulièrement si une commande lui est arrivée. Les délais peuvent paraître faible, mais sur le le long terme cela fait une énorme différence!
Le dernier délai, permet d'endormir pour toujours l'arduino, et est utilisé pour réveiller l'arduino en cas d'activation du pin 2.

Comment utiliser la bibliothèque alors? Regardez le code qui suit:

```
/*
 * Ce code a pour but de mettre en sommeil l'arduino
 * pendant une durée définie.
 * Il a été tiré du github de la bibliothèque LowPower
 */

 // On inclut la bibliothèque LowPower
#include "LowPower.h"

void setup()
{
  // Pas de setup nécessaire
}

void loop()
{
  // Il faut choisir la bonne puce, mais l'arduino nano et duemilanove
  // utilisent le ATmega328P, ATmega168
  LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,
                SPI_OFF, USART0_OFF, TWI_OFF);

  // L'arduino Mega utilise le ATmega2560
  //LowPower.idle(SLEEP_8S, ADC_OFF, TIMER5_OFF, TIMER4_OFF, TIMER3_OFF,
  //      TIMER2_OFF, TIMER1_OFF, TIMER0_OFF, SPI_OFF, USART3_OFF,
  //      USART2_OFF, USART1_OFF, USART0_OFF, TWI_OFF);

  // Ces puces sont pour d'autres arduinos
  // ATmega32U4
  //LowPower.idle(SLEEP_8S, ADC_OFF, TIMER4_OFF, TIMER3_OFF, TIMER1_OFF,
  //      TIMER0_OFF, SPI_OFF, USART1_OFF, TWI_OFF, USB_OFF);

  // ATmega256RFR2
  //LowPower.idle(SLEEP_8S, ADC_OFF, TIMER5_OFF, TIMER4_OFF, TIMER3_OFF,
  //      TIMER2_OFF, TIMER1_OFF, TIMER0_OFF, SPI_OFF,
  //      USART1_OFF, USART0_OFF, TWI_OFF);

  // Faire quelque chose ici
  // Exemple: Lire un capteur, récupérer des informations,
  // transmettre des données, etc...
}
```

Comme vous pouvez le voir dans le code, il suffit d'une seule ligne pour endormir l'arduino pendant 8s en désactivant tous les modules. Après 8 secondes, l'arduino continue d'exécuter le code qui suit.

Une autre méthode est d'endormir l'arduino et de le réveiller avec la pin 2, suivez le code suivant:
(pour mieux comprendre le fonctionnement de ce programme et en particulier des interruptions, veuillez vous référer au tutoriel sur les pins!)

```
/*
   Ce code a pour but de mettre en sommeil l'arduino
   et de le réveiller grâce à la pin 2.
   Il a été tiré du github de la bibliothèque LowPower
*/

// On inclut la bibliothèque LowPower
#include "LowPower.h"

// On assigne à la pin 2 un nom
const int wakeUpPin = 2;

/*
   Cette fonction sera appelée lorsque
   l'interruption sur la pin 2 sera activé
*/
void wakeUp()
{
  // Si on veut faire quelque chose
  // juste au moment du réveil
}

void setup()
{
  // Configure la pin 2 comme une entrée
  pinMode(wakeUpPin, INPUT);
}

void loop()
{
  // Initialise l'interruption sur la pin 2
  // pour quelle soit appelée lorsque la pin
  // est à l'état bas
  attachInterrupt(0, wakeUp, LOW);

  // Met en sommeil l'arduino tant que la pin 2
  // n'est pas à l'état bas
  LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,
                     SPI_OFF, USART0_OFF, TWI_OFF);

  // Désactive l'interruption dès que l'Arduino se réveille
  detachInterrupt(0);

  // Faire quelque chose ici
  // Exemple: Lire un capteur, récupérer des informations,
  // transmettre des données, etc...
}
```

Sources pour approfondir

- locoduino.com, site en français qui traite de l'alimentation de l'Arduino
  https://www.locoduino.org/spip.php?article16

- Comment amener son arduino en mode sleep, en anglais
  http://donalmorrissey.blogspot.com/2010/04/putting-arduino-diecimila-to-sleep-part.html

- Sparkfun, un site de vente qui produit également des tutos d'excellente qualité, en anglais
  https://learn.sparkfun.com/tutorials/reducing-arduino-power-consumption/all

- Lien Github vers la bibliothèque LowPower
  https://github.com/rocketscream/Low-Power

- Raspberrypi.org est le site officiel et la documentation officielle pour le Raspberry Pi, en anglais
  https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md