PWM, PFM et BCM

Pour modifier la vitesse d'un moteur (ou l'intensité d'une LED) on agit sur la tension moyenne en pulsant l'alimentation à une fréquence adéquate. Le moteur est un mauvais haut-parleur et on entend cette fréquence, parfois exploitée dans des jouets. Avec des moteurs industriels, on doit avoir des fréquences inaudibles. Par contre les robots commandés par microcontrôleurs peuvent avoir des fréquences très basses, aussi inaudibles ; les irrégularités de vitesse sont absorbées par le train d'engrenages.
Il faut encore comprendre que les moteurs à courant continu ont une tension de démarrage qui dépend de leur dimension et prix. Un Maxon de 30mm démarre avec 5% de sa tension nominale. Un moteur vibrant de 6mm récupéré dans un téléphone a besoin de 50%.

Pulser le courant se fait en PWM (Pulse Width Modulation) ou PFM (Pulse Frequency Modulation). La figure ci-dessous montre la différence.

Le PFM est peu connu, puisqu'il n'est pas supporté par les microcontrôleurs et n'est pas nécessaire pour les bons moteurs à haute fréquence. En robotique amateur par contre, le PWM n'est pas utilisable, puisque le moteur ne va pas démarrer à moins de 30% (par exemple) de PWM et va prendre de suite une vitesse importante, 30% de la vitesse max.
Avec le PFM, on choisit une durée d'impulsion suffisante pour que l'énergie apportée dépasse le frottement et fasse bouger le moteur. Par exemple, une impulsion de 2ms fait décoller un moteur pager de 6mm et il fait une fraction de tour. Avec 1/256 de PFM, le moteur va un peu bouger toutes les 500ms. Les saccades seront éventuellement visibles, mais la vitesse moyenne sera proportionnelle à la valeur PFM.

PWM

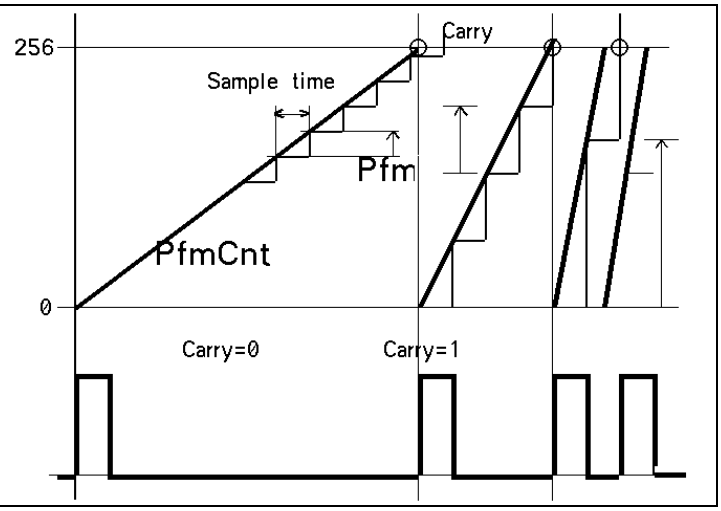
Le PWM pour Pinguino est expliqué en détail sous www.didel.com/kidules/KiPwm.pdf
Sur Arduino, il y a 6 sorties PWM bien connues. Le PWM pour des Leds est expliqué sous www.didel.com/diduino/CommandeLeds.pdf

PFM unidirectionnel

Une résolution de 8 bits n'a pas de sens pour des applications non professionnelles. Prenons l'exemple avec 4 bits, 16 vitesses.
Une variable pfmCnt est augmenté à chaque cycle de la valeur pfm. Si le résultat dépasse 16, on masque les bits de poids 4 à 7 et on active le bit moteur.

```
pfmCnt+=pfm ;
if (pfmCnt>15) {
   pfmCnt &= 0x0F ;
   MoteurOn ;
else MoteurOff;
```

La fréquence du Pfm est variable, maximale à 50% de PFM.



.

PFM bidirectionnel

Il y a plusieurs représentations possibles pour une vitesse signée. La plus logique est d'utiliser le type char (signé 8 bits) avec les vitesses positives 0 1 2 15 négatives –1=255 max –15.
Toutes les vitesses négatives ont le bit de signe (bit 7) à un, et la valeur négative doit être complémentée pour calculer l'apparation des impulsions, à envoyer sur l'autre pin du moteur.

```
if(pfm&0x80) {      // teste signe
  pfmCnt+=pfm ;
  if (pfmCnt>15) {MotAv; pfmCnt&=0x0F;}
     else MotStop;
```

```
else {
  pfmCnt+=pfm ;
  if (pfmCnt>15) {MotRec; pfmCnt&=0x0F;}
  else MotStop;
```

Plusieurs canaux

Il faut pour chaque canal redéfinir des variables et répéter les instructions ci-dessus.

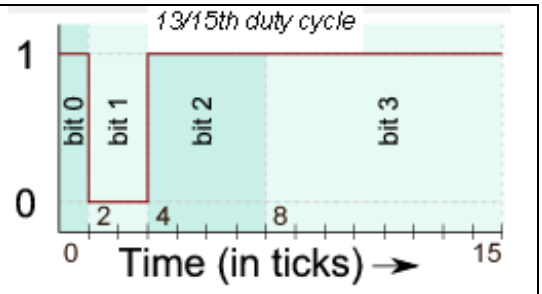
Interruptions

Les interruptions se gèrent comme pour le PWM. Voir www.didel.com/diduino/PfmParInter.pdf

Codage BCM

Le principe du codage BCM www.batsocks.co.uk/readme/art_bcm_3.htm est d'analyser la valeur binaire correspondant au % d'intensité . Si le bit 0 est à 1, on active la sortie une unité de temps, par exemple 1ms. Si c'est le bit 1, 2ms, etc. La séquence est irrégulière, mais la moyenne est correcte. La routine de codage BCM est présentée comme plus efficace que le PWM, mais l'algorithme de Ogden est bien plus efficace, en particulier pour gérer plusieurs canaux, voir www.didel.com/kidules/KiLs16.pdf .

```
u8 poids [4] = {1,2,4,8} ;
u8 ledi ; u8 ni ; u8 cni ;
void loop ()
{
      ledi = GetPortAC () ;
      for (ni=0;ni<4;ni++) {   // codage 4 bits
            if (ledi&1) Led=1;
            else Led=0;
            ledi = ledi >>1;
            cni = poids [ni] ;
            delay (cni) ;
      }
```



Ce document a été adapté et amélioré à partir de www.didel.com/pic/PwmPfm.pdf (assembleur CALM)

jdn 110725