ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

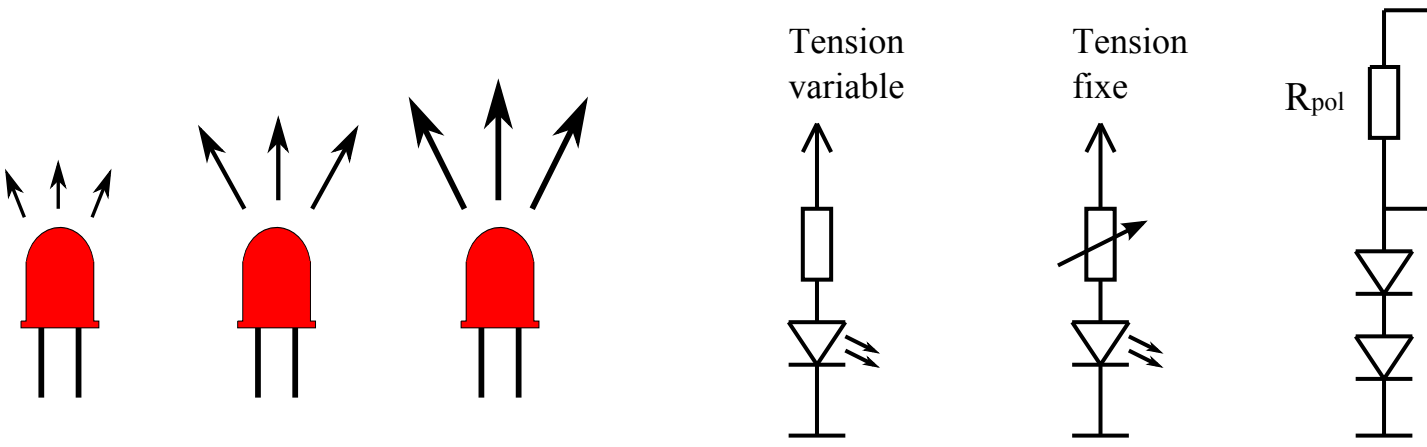
PWM : MODULATION DE LARGEUR D'

PIERRE-YVES ROCHAT, EPFL

RÉV 2015/07/19

VARIER L'INTENSITÉ D'UNE LED

Beaucoup d'enseignes à LED se contentent d'allumer et d'éteindre des LED ou des groupes de LED. C
plus intéressants et variés en ayant la possibilité de changer leurs intensités lumineuses.

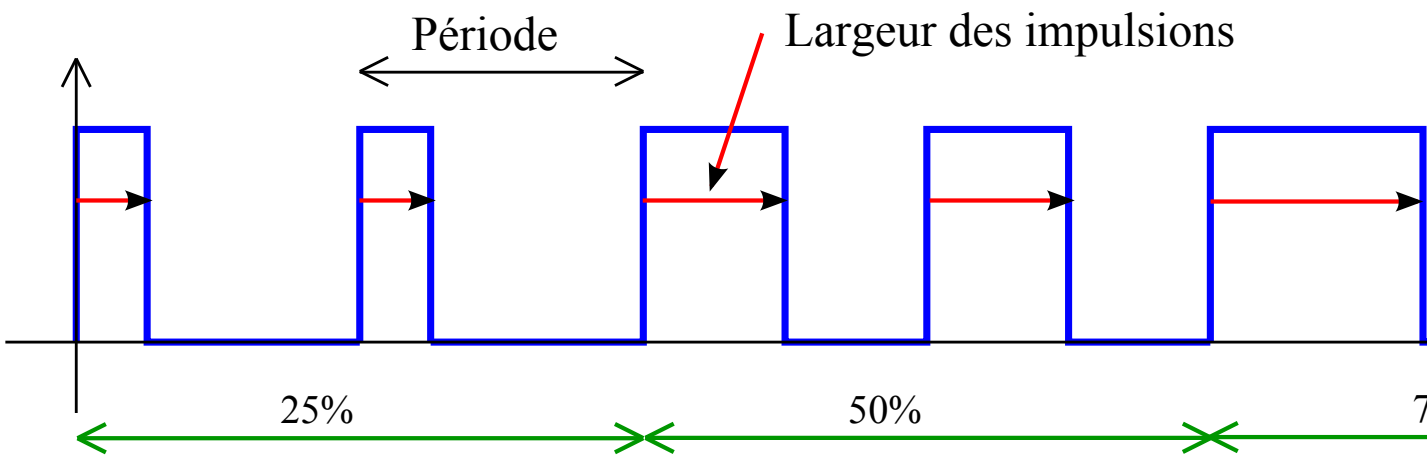


Comment faire varier l'intensité d'une LED ? On sait que c'est le courant qui la traverse qui produit d
généralement pas facile. On peut faire varier la tension qui alimente la LED, ou encore la résistance
ble de piloter le courant par un transistor, avec un montage adéquat.

Mais il existe une solution très différente et généralement beaucoup plus simple à implémenter.
LED. Que se passe-t-il lorsqu'on augmente la fréquence du clignotement ? À partir d'une certaine fr
voir qu'un scintillement. En augmentant encore la fréquence, il voit simplement la LED allumée, mais
était allumée en permanence.

Le PWM

À l'idée de faire clignoter rapidement la LED, ajoutons l'idée de faire varier le temps pendant leq
Modulation de Largeur d'Impulsion (MLI) ou Pulse Width Modulation (PWM) en anglais. Regardons l



{ width=15cm }

Le signal a une période constante (donc une fréquence constante). La durée de la partie active du
tracé, 25% de la puissance disponible peut être envoyée à la charge, vu que le signal est à *1* durant
du temps. De même, la puissance passe à 50% au milieu du tracé et à 75% dans la dernière partie.
puissance (0%) en laissant la sortie à *0*, ou toute la puissance (100%) en laissant la sortie à *1*.

Le rapport entre la durée de la partie active du signal et la durée de la période s'appelle le **rapport**

Fréquence du PWM

L'usage du PWM est très répandu, avec des applications dans beaucoup de domaines. La commande
PWM. Dans ce cas, c'est la nature inductive du moteur qui effectue l'intégration du signal, ainsi que

Dans le cas de la commande de diodes lumineuses, le temps d'allumage et d'extinction est très cou
plusieurs dizaines de mégahertz. C'est l'œil humain qui ne voit pas le clignotement. L'œil a une fre
vers 75 Hz. C'est bien lui qui effectue l'intégration du signal pour en percevoir une valeur moyenne.

Dans divers domaines de l'électronique, les fréquences des signaux PWM peuvent aller couramment
plus la fréquence est élevée, plus les pertes électriques à l'instant des changements de valeurs sont
dissipation importante d'énergie dans les éléments de commutation.

L'œil a une fréquence limite de perception du clignotement. Par exemple, on sait qu'un tube fluore
est commandé par du courant alternatif à 50 Hz, et que chaque période a une alternance positive

sensibles de notre œil (les cônes pour la vision en couleur et les batonnets pour la vision périphériqu
limite de perception du clignotement.

Pour les enseignes et afficheurs à LED, on vise généralement des fréquences de l'ordre de 100 à 2
difficile d'envoyer à ces fréquences toutes les informations à l'ensemble des LED, qui peuvent être
de LED capables d'afficher de la vidéo nécessitent des circuits électroniques complexes, qui seront

Programmation d'un PWM

Voici un programme très simple qui génère un signal PWM sur une sortie d'un microcontrôleur :

```
#define LedOn digitalWrite(P1_0, 1)
#define LedOff digitalWrite(P1_0, 0)

uint16_t pwmLed; // valeur du PWM, 0 à 100

void setup() { // Initialisations
  pinMode(P1_0, OUTPUT); // LED en sortie
  pwmLed = 25; // valeur du PWM.
}

void loop() { // Boucle infinie, durée 10ms => un cycle du PWM à 100 Hz
  LedOn;
  delayMicrosecond(100*pwmLed); // durée de l'impulsion
  LedOff;
  delayMicrosecond(100*(100-pwmLed)); // solde de la période
}
```

Le programme a été écrit de telle manière que les valeurs du PWM doivent être choisies entre 0 et
des *pour cents*. Les informaticiens choisissent plus souvent des valeurs dans une plage binaire, com
l'exemple suivant).

La fréquence a été choisie ici à 100 Hz. En effet les deux délais de la boucle principale totalisent 100

Après les initialisations de la sortie et de la variable qui contient en permanence la valeur du PWM,
et la partie inactive du signal. Les attentes sont obtenues par des délais exprimés en us.

Ce programme donne l'impression que la diode lumineuse est à demi intensité, malgré une comman
de l'œil n'est pas linéaire, mais logarithmique. On remarque aussi que le PWM est visible en déplaça

Génération de plusieurs signaux PWM

Le principe du programme que nous venons de voir ne convient pas à la programmation de PWM s
programmer un PWM qui se prête à gérer plusieurs sorties :

```
uint8_t pwmLed; // valeur du PWM, 0 à 255 (8 bits)
uint8_t cptPwm; // compteur du PWM

void setup() { // Initialisations
  pinMode(P1_0, OUTPUT); // LED en sortie
  pwmLed = 64; // valeur du PWM. Elle est ici fixe, mais pourrait changer
               // à tout moment en complétant le programme.
  cptPwm = 0; // compteur du PWM
}

void loop() { // Boucle infinie, durée 39us (256 * 39us = ~10ms)
  if ((cptPwm==0) && (pwmLed>0)) LedOn; // pour une valeur positive
  if (cptPwm==pwmLed) LedOff;

  cptPwm++; // passe automatiquement de 255 à 0 (overflow)
  delayMicroseconds(39);
}
```

Dans ce cas, la boucle principale dure seulement le temps qui correspond à une fraction de la pér
L'usage de cette valeur, associée à un compteur de type uint8_t (8 bits non signés), simplifie le pro
zéro, qui s'effectue au moment du dépassement de capacité (overflow).

Voici comment modifier ce programme pour qu'il commande 8 LED, avec 8 valeurs différentes de PW

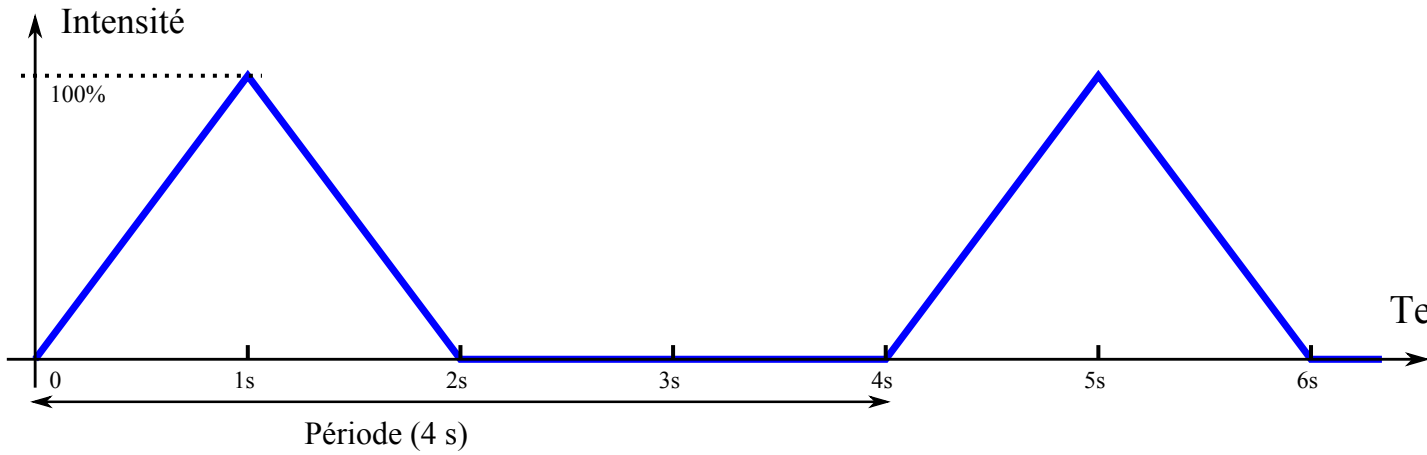
```
uint8_t pwmLed[8]; // valeurs des PWM, 0 à 255 (8 bits), pour 8 LED
uint8_t cptPwm; // compteur du PWM

void setup() ...

void LedOn (uint8_t n) ... procédure qui allume une des 8 LED
void LedOff (uint8_t n) ... procédure qui éteint une des 8 LED

void loop() { // Boucle infinie
  for (uint8_t i=0; i<8; i++) {
    if ((cptPwm==0) && (pwmLed[i]>0)) LedOn(i); // allume la LED concernée
    if (cptPwm==pwmLed[i]) LedOff(i); // éteint la LED correspondante
  }

  cptPwm++; // passe automatiquement de 255 à 0 (overflow)
  delayMicrosecond(39); // on pourrait diminuer cette valeur
  // pour tenir compte du temps d'exécution de la boucle for.
}
```

Programmation d'une séquence en PWM

Dans des applications comme la gestion des moteurs d'un robot mobile, les valeurs du PWM sont c
valeurs des capteurs (capteurs de distance, caméras, etc.)

Dans le cas des enseignes et afficheurs à LED, les valeurs sont calculées en fonction du temps qui p
ment connaître le temps qui s'écoule ? Une des manières consiste simplement à compter les cycles d

Prenons un exemple. Pour donner l'impression qu'un appareil est en mode "repos", une LED va av
suivante :



La séquence est cyclique et dure 4 s. Il imite la respiration d'une personne qui dort. Durant la p
jusqu'au maximum. Durant la deuxième seconde, elle diminue jusqu'à zéro, valeur qui reste jusqu'à l

La courbe se divise en 3 parties. Il est facile d'exprimer l'intensité en fonction du temps dans chacun
le programme :

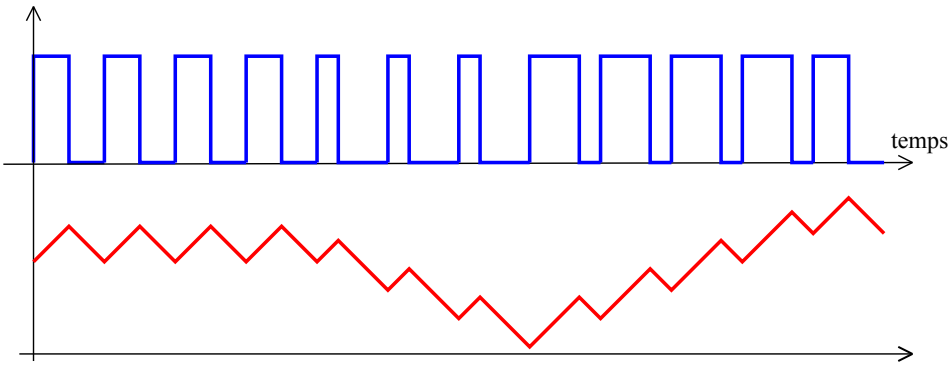
```
uint16_t pwmLed; // valeur du PWM, 0 à 255, 16 bits pour les calculs
uint16_t cpt10ms = 0; // compteur des cycles de 0 à 400, par 10ms, total 4s
...
void loop() { // Boucle infinie, durée 39us (256 * 39us = ~10ms)
  if (cptPwm==0) {
    cpt10ms++;
    if (cpt10ms<100) { //première seconde
      pwmLed = cpt10ms * 256 / 100; // droite montante
    } else if (cpt10ms<200) { // deuxième seconde
      pwmLed = 256 - ((cpt10ms-100) * 256 / 100); // droite descendante
    } else {
      pwmLed = 0;
      if ( cpt10ms==400) cpt10ms = 0; // fin des 4 s
    }
  }

```

```
18    if ((cptPwm==0) && (pwmLed>0)) LedOn; // seulement si valeur non nulle
19    if (cptPwm==pwmLed) LedOff;
20
21    cptPwm++; // passe automatiquement de 255 à 0 (overflow)
22    delayMicroseconds(39);
  }
```

Il est important de contrôler que les variables utilisées ne produisent pas de dépassement de capaci
pwmLed a été choisie ici sur 16 bits. Pour la première droite, *cpt10ms* peut aller jusqu'à *99*. Multiplié p
65'535 (qui est le nombre maximum que peut représenter un nombre entier de 16 bits). Le contrôle

Convertisseur DAC simple

Bien que le signal PWM soit un signal binaire (il est soit à *0* soit à *1* à un instant donné), il est port
effet de faire l'intégrale du signal pour s'en rendre compte. Il faut pondérer le *1* logique comme la v
-1, comme le montre la figure suivante :

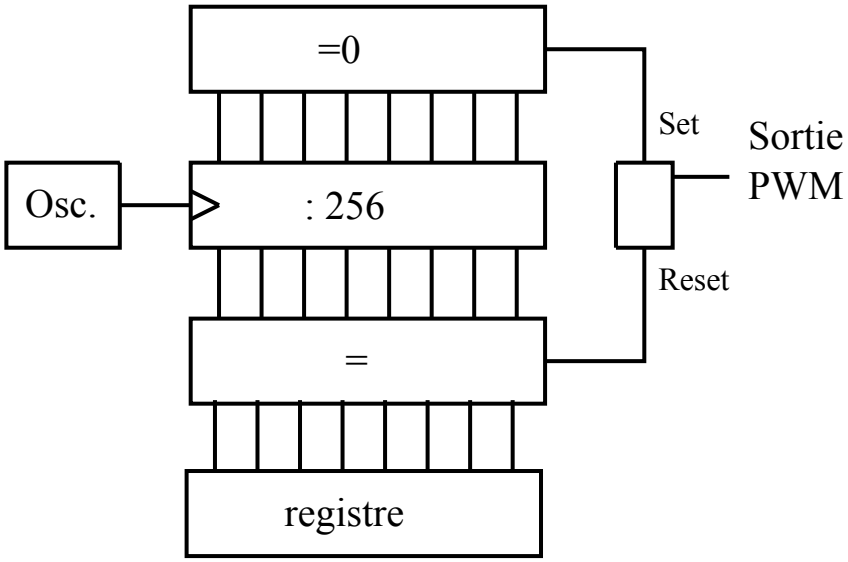


{ width=15cm }

Il est donc possible de réaliser un convertisseur numérique-analogique (Digital to Analog Converter,
suivi par un filtre passe-bas adapté à la fréquence du PWM. Ce filtre peut être un simple filtre R-C.

Circuits spécialisés

Comment générer un signal en PWM avec des circuits logiques ? Il faut utiliser un compteur, pilote
détecteur de 0 permet de signaler le début du cycle. Un comparateur, connecté à un registre qui co
tecter la fin de l'impulsion. Une bascule *Set-Reset* génère le signal.



{ width=10cm }

Les microcontrôleurs contiennent des circuits logiques supplémentaires ressemblant beaucoup à ce
plétés de circuits logiques dédiés, ils peuvent générer du PWM sans autre programmation que l'in
leur sont associés. Nous en parlerons plus tard dans ce cours.