La PWM : qu’est-ce ? (1)

Application aux diodes électroluminescentes

Le 14 mars 2015. Par : Dominique, Guillaume, Jean-Luc. URL : http://www.locoduino.org/spip.php?article47

Jusqu’à présent, dans les exemples qui ont illustré les articles sur LOCODUINO, nous avons vu l’émission de signaux numériques par le biais des broches numériques et de `digitalWrite(...)` et l’allumage et l’extinction de DEL. Ces signaux numériques ont soit une valeur égale à LOW, c’est à dire 0V, soit une valeur égale à HIGH, ce qui correspond à la tension d’alimentation de l’Arduino, VDD, soit 5V pour les Arduino à base de micro-contrôleurs AVR ou bien 3,3V pour les Arduino à base de micro-contrôleurs ARM, voir à ce sujet « Qu’est ce qu’une carte Arduino ? ».

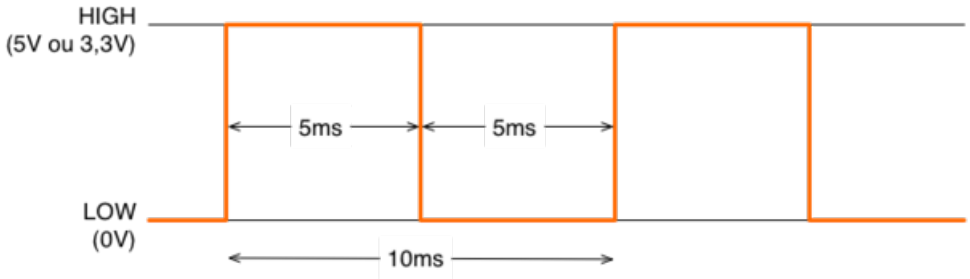
Mais les sorties numériques ne nous permettent pas, par exemple, de régler la luminosité de la DEL, ou de faire varier la vitesse de rotation d’un moteur [1]. Pour pouvoir faire cela, il serait nécessaire de pouvoir disposer de sorties permettant des tensions intermédiaires entre LOW et HIGH.

Mais ceci n’existe pas sur les Arduino à base d’AVR [2]. L’alternative est d’utiliser une PWM, pour Pulse Width Modulation, ou MLI en français pour Modulation de largeur d’impulsion [3].

Qu’est ce que la PWM ?

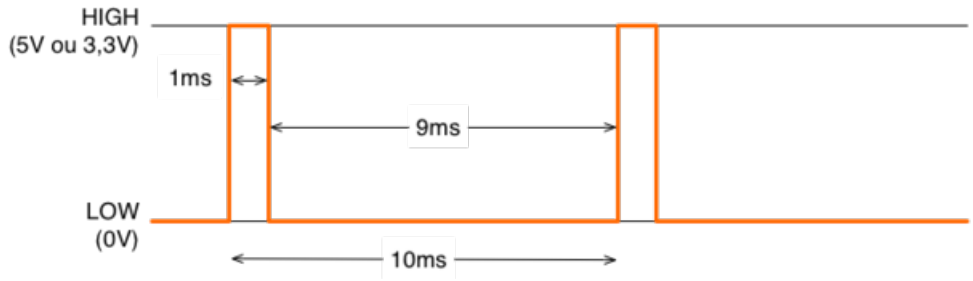
On reste en numérique, les signaux ont toujours une valeur LOW ou HIGH et le principe est de construire un signal qui est alternativement LOW et HIGH et de répéter très vite cette alternance. La DEL est donc alternativement allumée et éteinte mais le cycle est tellement rapide que la persistance rétinienne nous donne l’illusion d’une DEL allumée en permanence. Nous avons déjà rencontré ce phénomène dans « La programmation, qu’est ce que c’est » où, en tentant de faire clignoter une DEL, on obtenait un allumage permanent mais d’une luminosité moindre.

Prenons par exemple une période de 10ms, soit une fréquence de 100Hz. Si la DEL est allumée pendant 5ms et éteinte pendant 5ms, comme sur la figure ci-dessous, l’impression sera une luminosité de 50% de la luminosité maximum.



PMW à 50%
La fréquence est de 100Hz, le rapport cyclique de 50%

Si la DEL est allumée pendant 1ms et éteinte pendant 9ms, l’impression sera une luminosité de 10% comme sur la figure ci-dessous.



PWM à 10%
La fréquence est de 100Hz et le rapport cyclique de 10%.

Le pourcentage de temps passé à l'état HIGH sur la période du signal est appelé le *rapport cyclique*. Il varie donc de 0%, le signal est tout le temps LOW, à 100%, le signal est tout le temps HIGH.

Les capacités PWM des Arduino

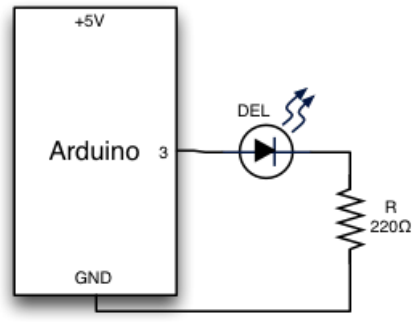
Les micro-contrôleurs proposent donc des dispositifs permettant de disposer de PWM autonomes où le signal voulu est engendré indépendamment de l'exécution du programme. Toutes les broches ne sont pas utilisables. Sur l'Arduino Uno, les broches concernée portent le symbole '~'. Sur le Uno et les Arduino à base d'ATMega 328, 6 broches sont disponibles, sur le Mega, 15 broches sont disponibles. Le tableau ci-dessous résume la disponibilité des PWM sur les cartes Arduino.

Modèle d'Arduino	**Broches PWM**
Uno, Pro Mini, Nano	3, 5, 6, 9, 10 et 11
Mega	2 à 13, 44 à 46
Leonardo	3, 5, 6, 9, 10, 11 et 13
Due	2 à 13
Zero	2 à 13

La fréquence de la PWM est prédéterminée sur l'Arduino. Il est possible de la changer comme nous le verrons dans un futur article mais ce n'est pas une possibilité accessible très simplement. La fréquence n'est pas la même selon les broches. Sur le Uno, la fréquence est de **490Hz** sur les broches **3, 9, 10 et 11** et de **980Hz** sur les broches **5 et 6**.

Mise en œuvre

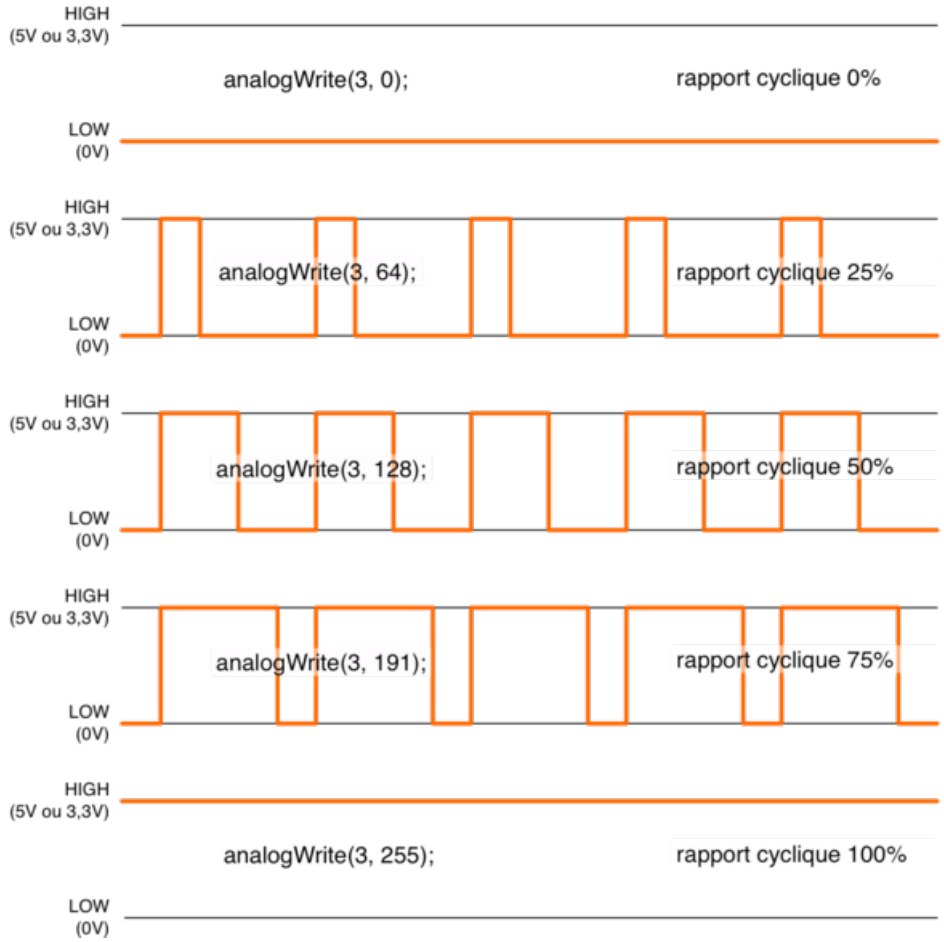
Nous allons mettre en œuvre la PWM avec une simple DEL. Le câblage reste identique à ce qui a été présenté dans « Fonctionnement et pilotage d'une DEL » dans la figure de gauche excepté que nous allons employer la broche 3 au lieu de la 2 car cette dernière n'a pas la fonction PWM.



En ce qui concerne la programmation, la fonction permettant de fixer le rapport cyclique de la PWM est la fonction `analogWrite(...)`. Nom bien mal choisi puisque, comme on l'a vu, tout est numérique. Le premier argument de `analogWrite(...)` est la broche concernée et le second argument le rapport cyclique. Le rapport cyclique **n'est pas** donné de **0 à 100** mais de **0 à 255**. Il faut donc faire une règle de 3 pour calculer la valeur à appliquer pour le rapport cyclique voulu. Par exemple, la valeur à appliquer pour un rapport cyclique de 75% sera égal à 0,75 x 255 = 191.

```
analogWrite(3, 191); // éclaire notre DEL à 75%
```

La figure ci-dessous montre différents réglages de PWM.



Voici pour commencer un programme très simple qui augmente progressivement, de 1 en 1, la luminosité de la DEL de 0, extinction totale, à 255, luminosité maximum, puis recommence à 0. On va fixer le temps de ce cycle à 1000ms. Le délai entre chaque augmentation est donc de 1000 / 255 = 3,922 ms. Arrondissons à 4ms. En utilisant un `byte` pour stocker la valeur, nous pouvons profiter du débordement qui se produit lorsque, étant à 255, l'ajout de 1 fait repasser la valeur 0. Voir à ce propos « Types, constantes et variables ». Vous noterez également qu'il n'est pas nécessaire de programmer la broche en sortie pour son utilisation en tant que sortie PWM. Cette programmation s'effectue automatiquement lors du premier `analogWrite()`. Malgré tout il faut que la fonction `setup()` soit présente même si elle est vide.

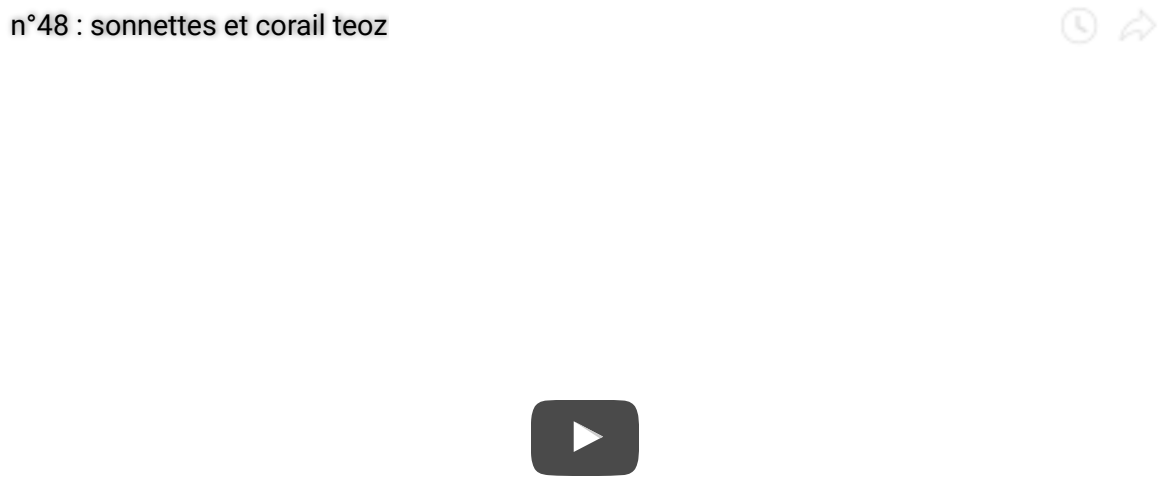
```
const byte pinDEL = 3;
byte luminosite = 0;

void setup()
{
}

void loop()
{
  analogWrite(pinDEL, luminosite);
  luminosite++;
  delay(4);
```

```
}
```

Télécharger

Un feu clignotant de passage à niveau

Nous allons maintenant prendre comme objectif de reproduire un feu clignotant comme montré dans cette vidéo.



Comme on peut le voir, le feu ne passe pas instantanément de 0 à 100% ni de 100 à 0% Il y a une phase d’augmentation et une phase de diminution. Après examen, on constate que le temps pendant que le feu est à 0% et à 100% est d’environ 200ms. Le temps de passage de 0 à 100% et de 100 à 0% est d’environ 250ms. La période de clignotement est de 900ms. Le programme doit donc :

1. mettre la PWM à 0
2. attendre 200ms
3. augmenter la PWM de 1 toutes les millisecondes jusqu’à atteindre 250
4. attendre 200ms
5. diminuer la PWM de 1 toutes les millisecondes jusqu’à atteindre 0
6. recommencer à l’étape 2

Fort de ce que nous savons déjà faire en programmation, nous allons déclarer une structure pour gérer notre feu et gérer le temps via la fonction `millis()`. Vous pouvez lire à ce propos « Les structures » et les articles précédents sur le programmation ainsi que « Comment gérer le temps dans un programme ? ».

Notre feu clignotant a 4 états : `ETEINT`, `EN_AUGMENTATION`, `ALLUME` et `EN_DIMINUTION` qui sont déclarés via un `enum`, voir « 3 façons de déclarer des constantes », et mémorisés dans un membre de notre structure. Nous avons également besoin de la date à laquelle le feu clignotant a changé d’état.

La structure contient donc 3 membres : la broche sur laquelle la DEL du feu est connectée, l’état du

feu et la date de dernier changement d'état :

```
struct FeuClignotant {
  const byte pin;
  byte etat;
  unsigned long dateDernierChangement;
};
```

Nous créons maintenant une variable de type `struct FeuClignotant` et l'initialisons :

```
struct FeuClignotant feuPN = {3, ETEINT, 0 };
```

Au lieu d'augmenter puis de diminuer la PWM de 1 toutes les millisecondes, nous allons calculer sa valeur : si le feu est dans l'état `EN_AUGMENTATION`, c'est le temps écoulé, en millisecondes depuis le passage du feu à l'état `EN_AUGMENTATION`. De même si le feu est `EN_DIMINUTION`, c'est 250 moins le temps écoulé, en millisecondes depuis le passage du feu à l'état `EN_DIMINUTION`. La fonction `gereFeuClignotant(...)` est donc programmée en utilisant un `switch ... case` pour faire évoluer l'état du feu en fonction de l'écoulement du temps, voir à ce propos l'article « Instructions conditionnelles : le switch ... case ».

```
void gereFeuClignotant(struct FeuClignotant& feu)
{
  unsigned long date = millis();
  int valeurPWM;
  switch (feu.etat)
  {
    case ETEINT:
      analogWrite(feu.pin, 0);
      // Attente de 200ms
      if (date - feu.dateDernierChangement >= 200) {
        // les 200ms sont passée, on change l'état du feu
        feu.dateDernierChangement = date;
        feu.etat = EN_AUGMENTATION;
      }
      break;
    case EN_AUGMENTATION:
      // La valeur de la PWM est donnée par la difference entre
      // la date et la dateDernierChangement
      valeurPWM = date - feu.dateDernierChangement;
      if (valeurPWM <= 250) analogWrite(feu.pin, valeurPWM);
      else {
        // Les 250 ms sont passées, on change l'état du feu
        feu.dateDernierChangement = date;
        feu.etat = ALLUME;
      }
      break;
    case ALLUME:
      analogWrite(feu.pin, 250);
      // Attente de 200ms
      if (date - feu.dateDernierChangement >= 200) {
        // les 200ms sont passée, on change l'état du feu
        feu.dateDernierChangement = date;
        feu.etat = EN_DIMINUTION;
      }
```

```
      break;
    case EN_DIMINUTION:
      // La valeur de la PWM est donnée par 250 moins la difference entre
      // la date et la dateDernierChangement. Attention aux parenthèses
      valeurPWM = 250 - (date - feu.dateDernierChangement);
      if (valeurPWM >= 0) analogWrite(feu.pin, valeurPWM);
      else {
        // Les 250 ms sont passées, on change l'état du feu
        feu.dateDernierChangement = date;
        feu.etat = ETEINT;
      }
      break;
  }
}
```

Enfin, dans loop, la fonction `gereFeuClignotant(...)` est appelée de manière répétitive.

```
void loop()
{
    gereFeuClignotant(feuPN);
}
```

Vous remarquerez que cette fonction gère le temps de manière non bloquante. Il est donc possible de gérer simultanément plusieurs feux de passage à niveau voire de gérer simultanément d’autres fonctions.

À vous de jouer !

Sketch du feu clignotant de PN

[1] Nous verrons cela plus tard

[2] En revanche, cette fonction est disponible sur l’Arduino Due sur les broches DAC0 et DAC1. Le micro-contrôleur du Due possède des convertisseurs numérique analogique.

[3] Peut-être n’avez vous jamais essayé d’alimenter une locomotive avec une tension continue, une vraie avec une alimentation de laboratoire car nos chers transformateurs ne délivre pas une tension continue. Le résultat est médiocre. La machine hoquette, sa vitesse est instable.