

Enseignes et afficheurs à LED

# PWM : Modulation de Largeur d'Impulsion

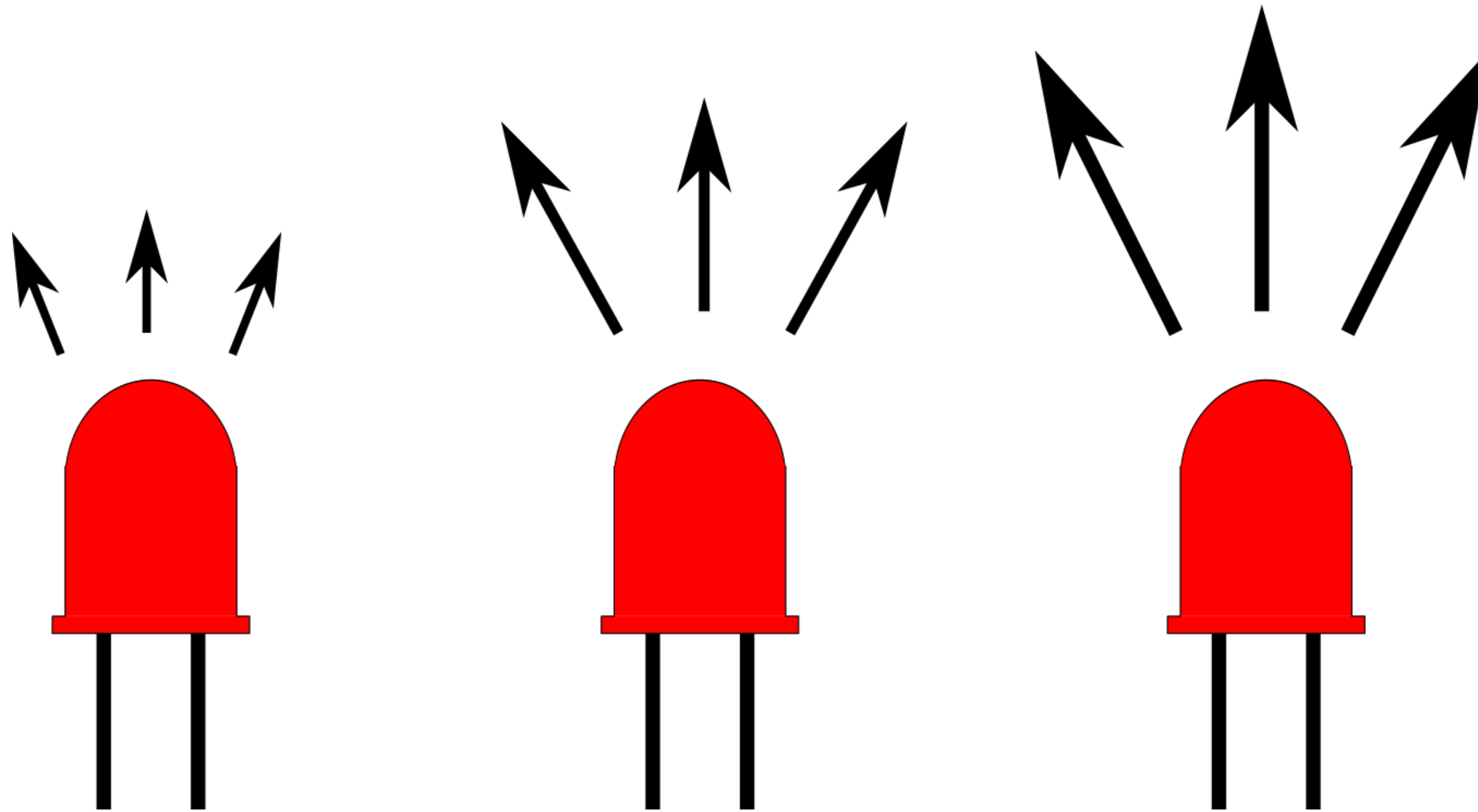
# PWM : Modulation de Largeur d'Impulsion

**Pierre-Yves Rochat**

# PWM : Modulation de Largeur d'Impulsion

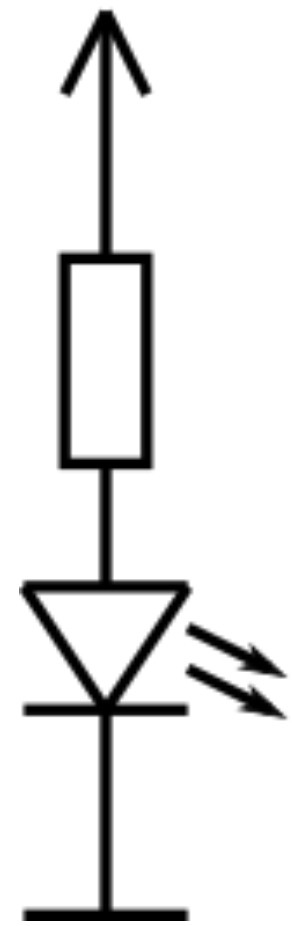
- Principe
- Fréquence
- Programmer un PWM
- Convertisseur numérique-analogique
- Réalisation par des circuits logiques

Comment faire varier l'intensité d'une LED ?

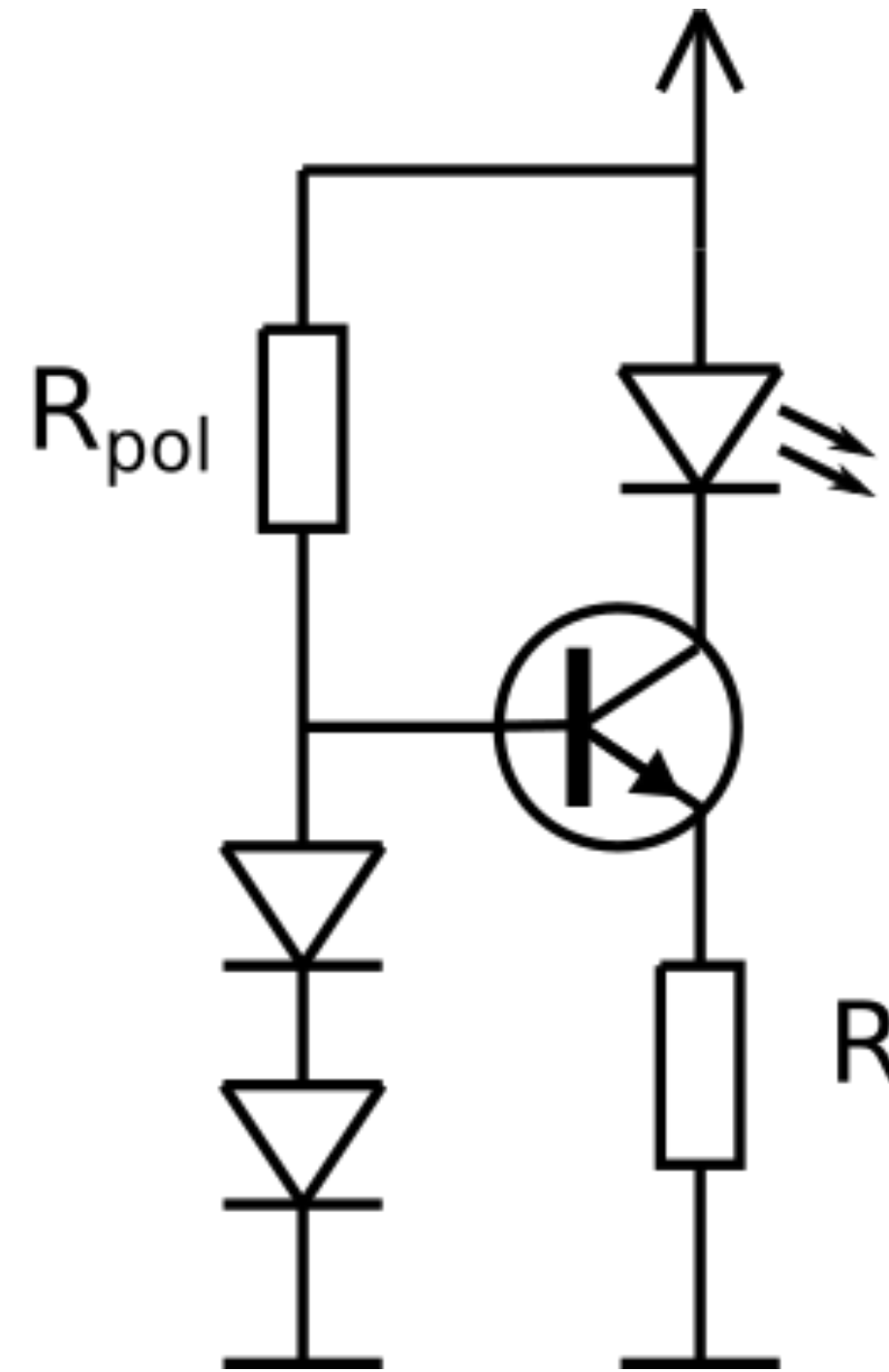
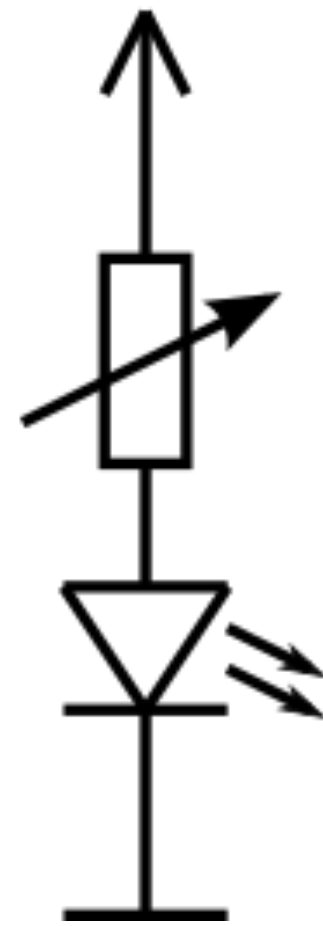


Comment faire varier l'intensité d'une LED ?

Tension  
variable

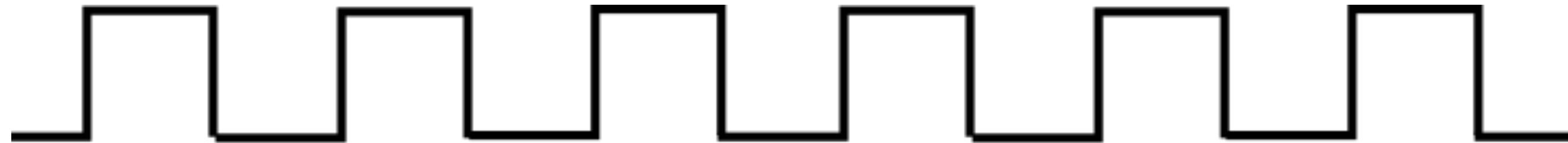


Tension  
fixe

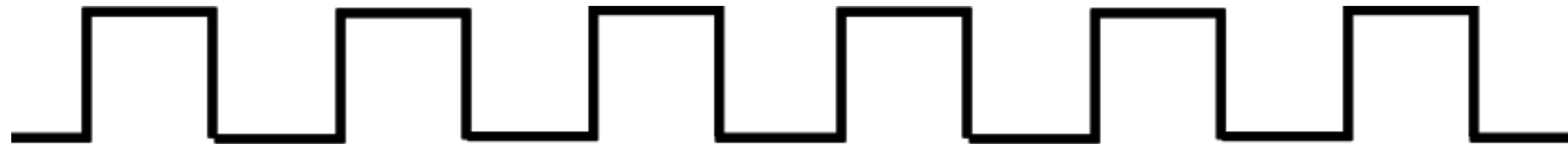


# Autre solution

Clignotement



Clignotement



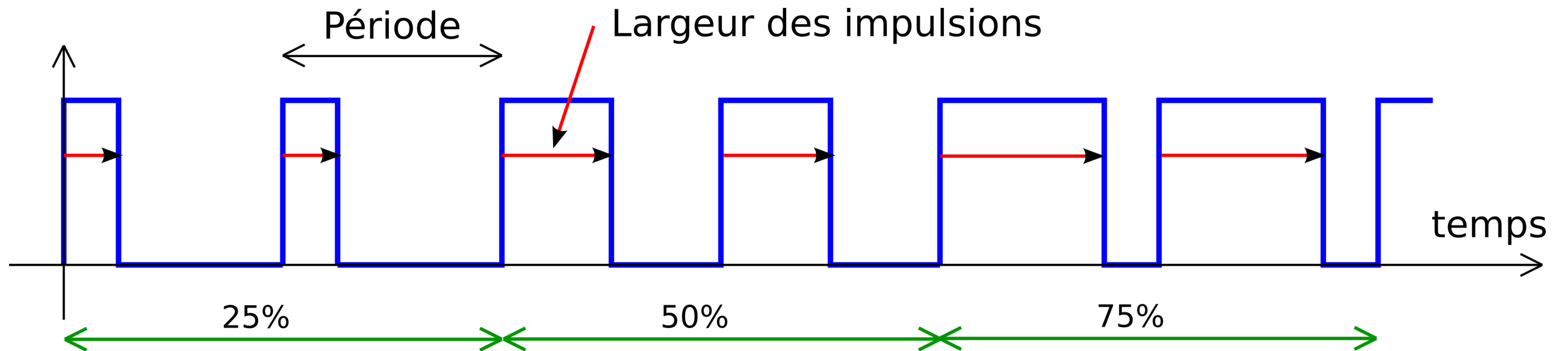
... plus rapide



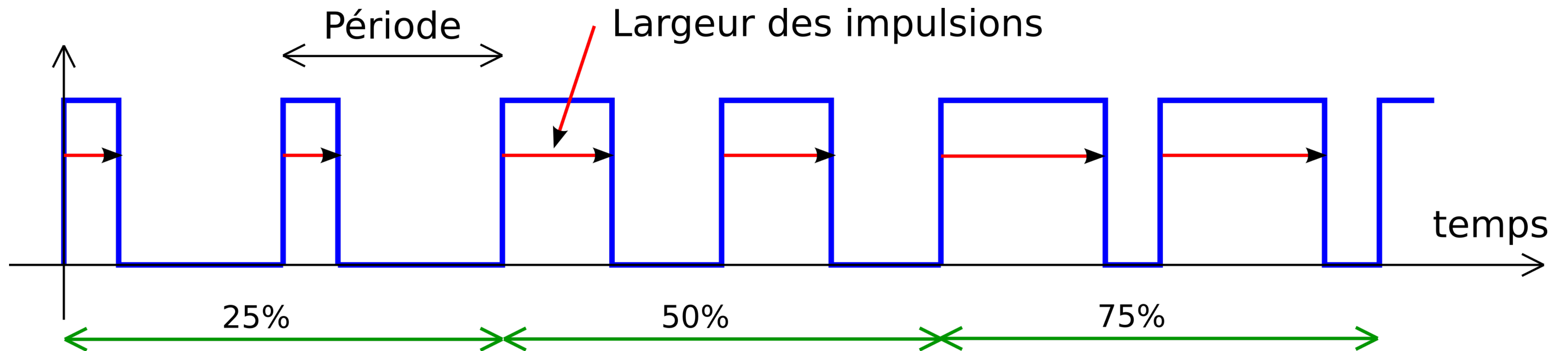
< 20ms



# PWM : principe



# PWM : principe



***Pulse Width Modulation*** = Modulation de Largeur d'Impulsion

- Selon les applications du PWM, les fréquences sont très différentes, de quelques Hz à des dizaines de MHz.

- Selon les applications du PWM, les fréquences sont très différentes, de quelques Hz à des dizaines de MHz.

Pour des applications visuelles :

- Selon les applications du PWM, les fréquences sont très différentes, de quelques Hz à des dizaines de MHz.

Pour des applications visuelles :

- L'œil a une fréquence limite de perception du clignotement

- Selon les applications du PWM, les fréquences sont très différentes, de quelques Hz à des dizaines de MHz.

Pour des applications visuelles :

- L'œil a une fréquence limite de perception du clignotement
- On ne voit pas clignoter un tube fluorescent, à 100 Hz (2 x 50 Hz)

- Selon les applications du PWM, les fréquences sont très différentes, de quelques Hz à des dizaines de MHz.

Pour des applications visuelles :

- L'œil a une fréquence limite de perception du clignotement
- On ne voit pas clignoter un tube fluorescent, à 100 Hz (2 x 50 Hz)
- Les cônes et les bâtonnets n'ont pas la même fréquence limite

# Programmation d'un signal PWM

Comment programmer des signaux PWM avec un microcontrôleur ?



Comment programmer des signaux PWM avec un microcontrôleur ?

- Allumer – attendre

Comment programmer des signaux PWM avec un microcontrôleur ?

- Allumer – attendre
- éteindre – attendre

Comment programmer des signaux PWM avec un microcontrôleur ?

- Allumer – attendre
- éteindre – attendre
- et répéter !

# Programmation par période

```
1 #define LedOn digitalWrite(P1_0, 1)
2 #define LedOff digitalWrite(P1_0, 0)
3 uint16_t pwmLed; // valeur du PWM, 0 à 100
4
5 void setup() { // Initialisations
6     pinMode(P1_0, OUTPUT); // LED en sortie
7     pwmLed = 25; // valeur du PWM.
8 }
9
10 void loop() { // Boucle infinie, durée 10ms => un cycle du PWM à 100 Hz
11     LedOn;
12     delayMicrosecond(100*pwmLed); // durée de l'impulsion
13     LedOff;
14     delayMicrosecond(100*(100-pwmLed); // solde de la période
15 }
```

Comment programmer plusieurs signaux PWM en même temps ?

Comment programmer plusieurs signaux PWM en même temps ?

- Difficile si la boucle principale dure une période complète du PWM

Comment programmer plusieurs signaux PWM en même temps ?

- Difficile si la boucle principale dure une période complète du PWM
- Plus facile si la boucle principale dure le temps de la plus courte impulsion possible du PWM

# Programmer plusieurs PW

```
1 uint8_t pwmLed; // valeur du PWM, 0 à 255 (8 bits)
2 uint8_t cptPwm; // compteur du PWM
3
4 void setup() { // Initialisations
5     pinMode(P1_0, OUTPUT); // LED en sortie
6     pwmLed = 64; // valeur du PWM. Elle est ici fixe, mais pourrait changer
7                 // à tout moment en complétant le programme.
8     cptPwm = 0; // compteur du PWM
9 }
10
11 void loop() { // Boucle infinie, durée 39us (256 * 39us = ~10ms)
12     if ((cptPwm==0) && (pwmLed>0)) LedOn; // si la valeur est positive
13     if (cptPwm==pwmLed) LedOff;
14
15     cptPwm++; // passe automatiquement de 255 à 0 (overflow)
16     delayMicroseconds(39);
17 }
```

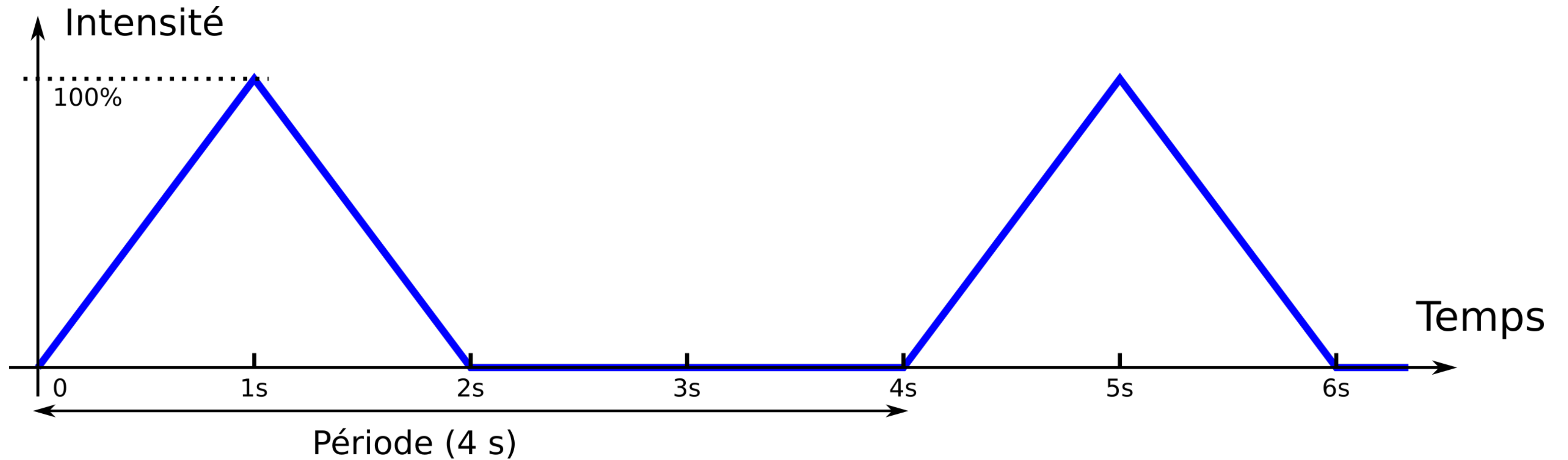


- Comment utiliser ce PWM ?

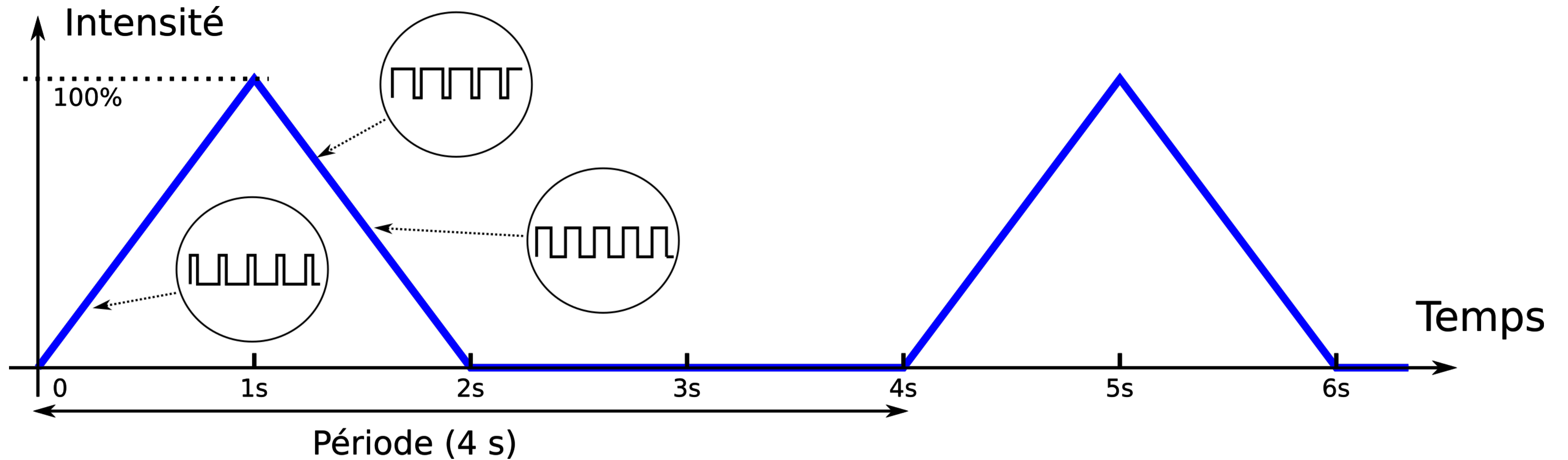
- Comment utiliser ce PWM ?
- Sur des enseignes et afficheurs, on peut créer des séquences.

- Comment utiliser ce PWM ?
- Sur des enseignes et afficheurs, on peut créer des séquences.
- Exemple : LED imitant le repos.

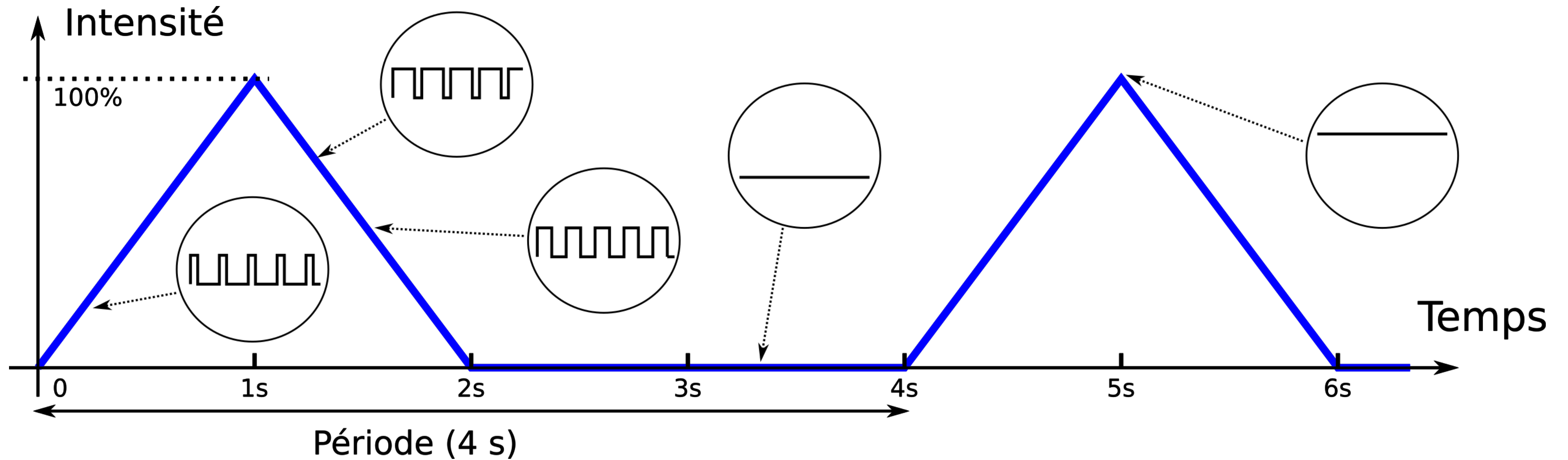
# Séquences en PWM



# Séquences en PWM



# Séquences en PWM



# Séquences en PWM

```
1 uint16_t pwmLed; // valeur du PWM, 0 à 255 (8 bits, 16 bits pour les calculs)
2 uint16_t cpt10ms = 0; // compteur des cycles, de 0 à 400 (par 10ms, total 4s)
3 void loop() { // Boucle infinie, durée 39us (256 * 39us = ~10ms)
4     if (cptPwm==0) {
5         cpt10ms++;
6         if (cpt10ms<100) { //première seconde
7             pwmLed = cpt10ms * 256 / 100; // droite montante
8         } else if (cpt10ms<200) { // deuxième seconde
9             pwmLed = 256 - ((cpt10ms-100) * 256 / 100); // droite descendante
10        } else {
11            pwmLed = 0;
12            if ( cpt10ms==400) cpt10ms = 0; // fin des 4 secondes
13        }
14    }
15    if ((cptPwm==0) && (pwmLed>0)) LedOn; // LED allumée si la valeur est positive
16    if (cptPwm==pwmLed) LedOff;
17    cptPwm++; // passe automatiquement de 255 à 0 (overflow)
18    delayMicroseconds(39);
19 }
```

# Convertisseur numérique-analogique en PWM

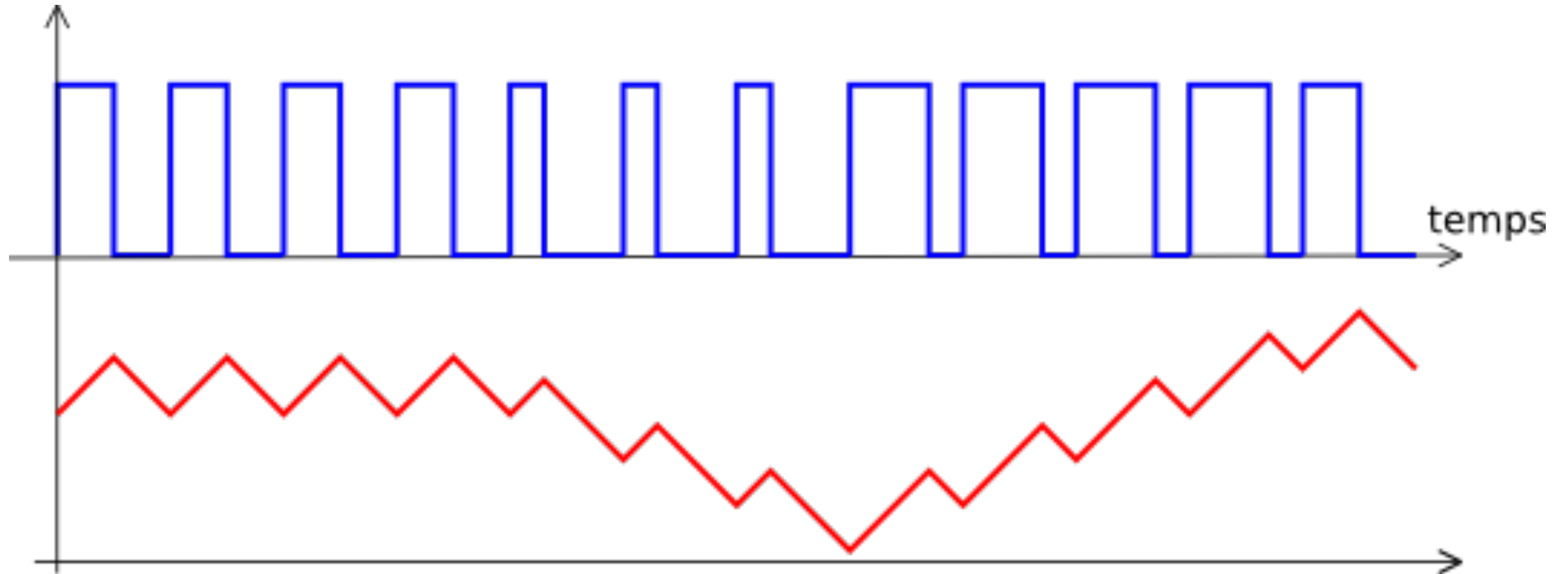
- Transmettre une information variable vers l'extérieur



# Convertisseur numérique-analogique en PWM

- Transmettre une information variable vers l'extérieur
- Conversion Numérique-Analogique  
DAC Digital to Analog Converter

# Convertisseur numérique-analogique en PWM



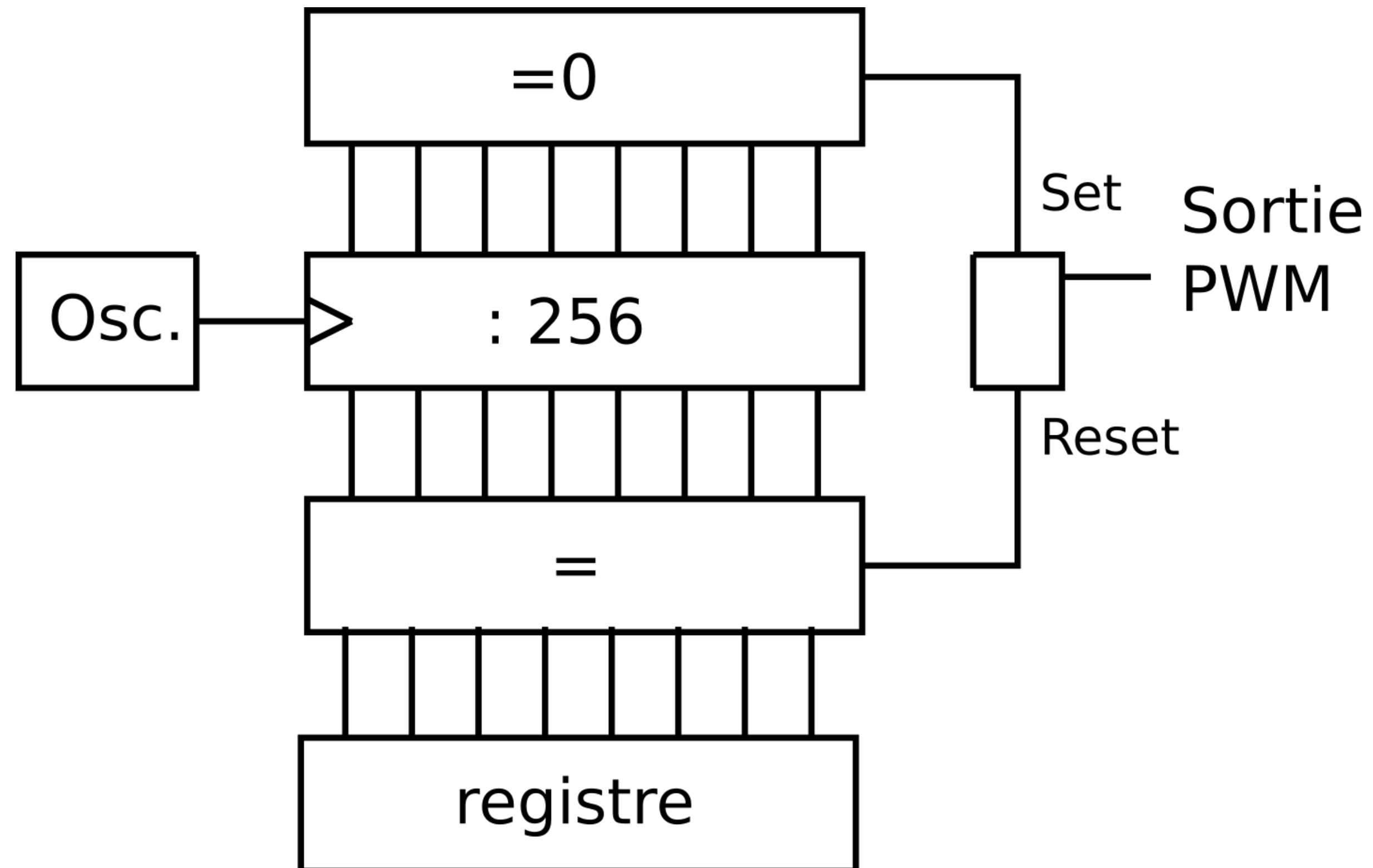
# PWM réalisé avec des circuits logiques

- Comment soulager le microcontrôleur de la génération du PWM ?

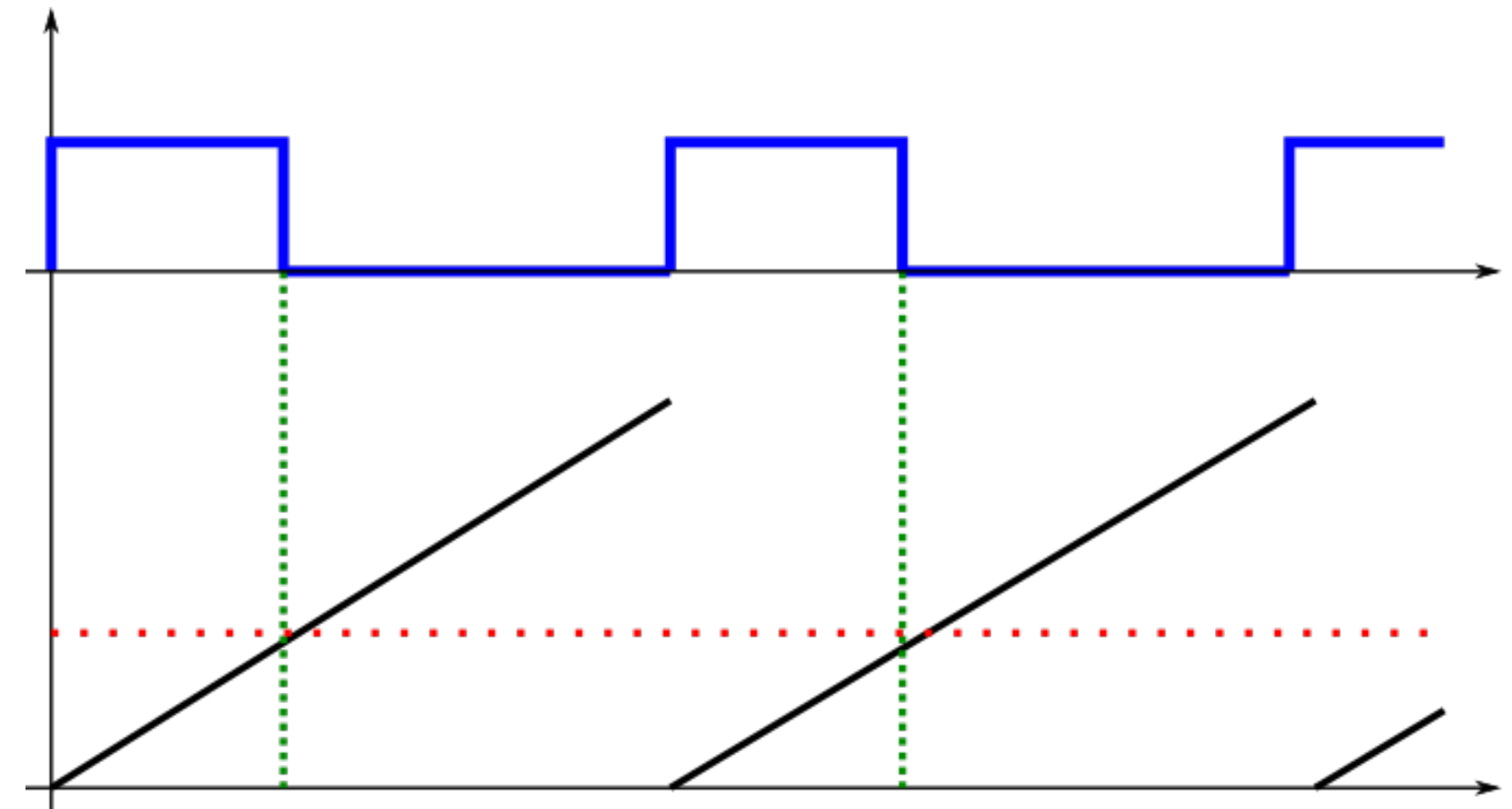
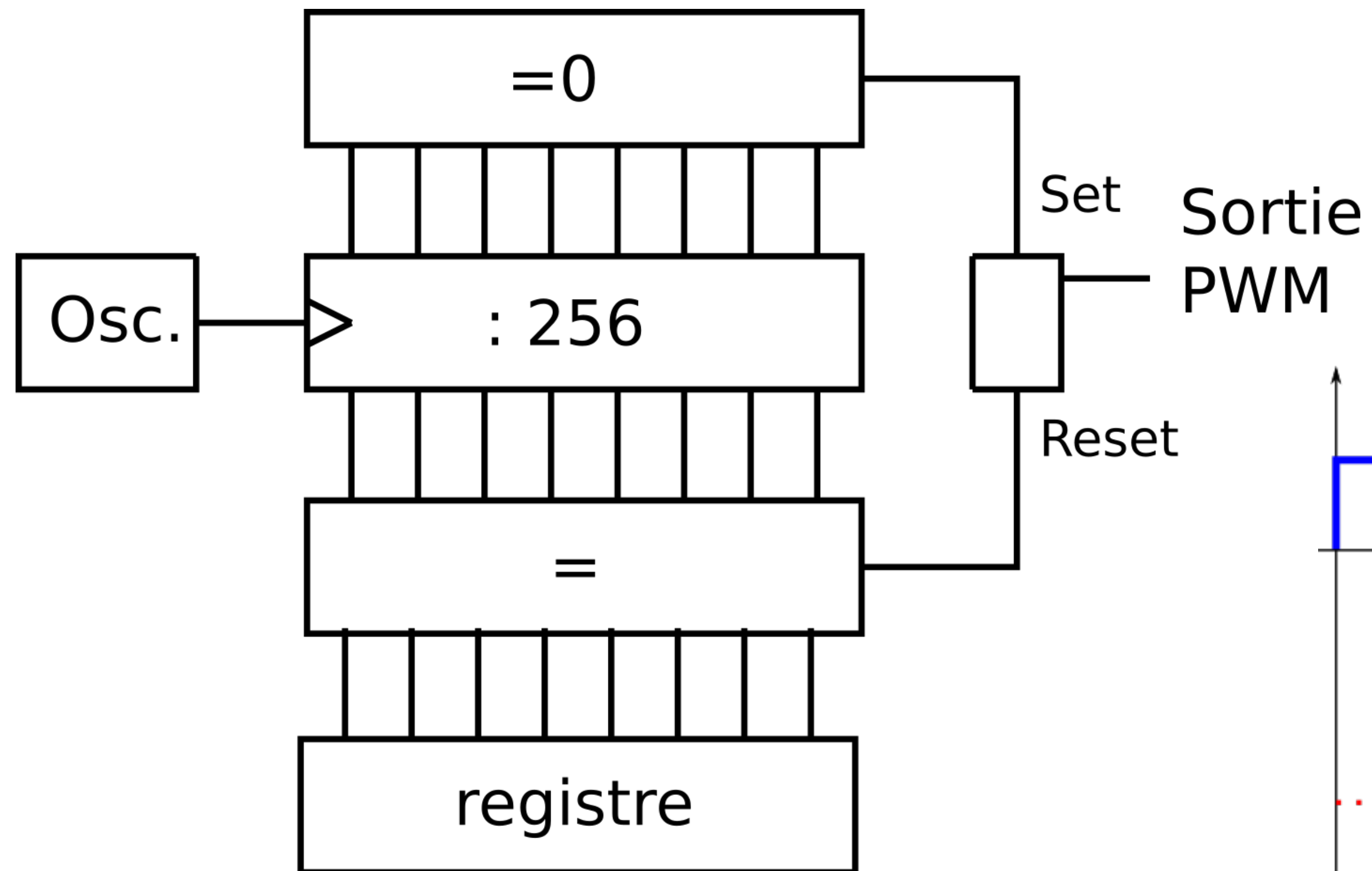
# PWM réalisé avec des circuits logiques

- Comment soulager le microcontrôleur de la génération du PWM ?
- En utilisant des circuits logiques spécialisés !

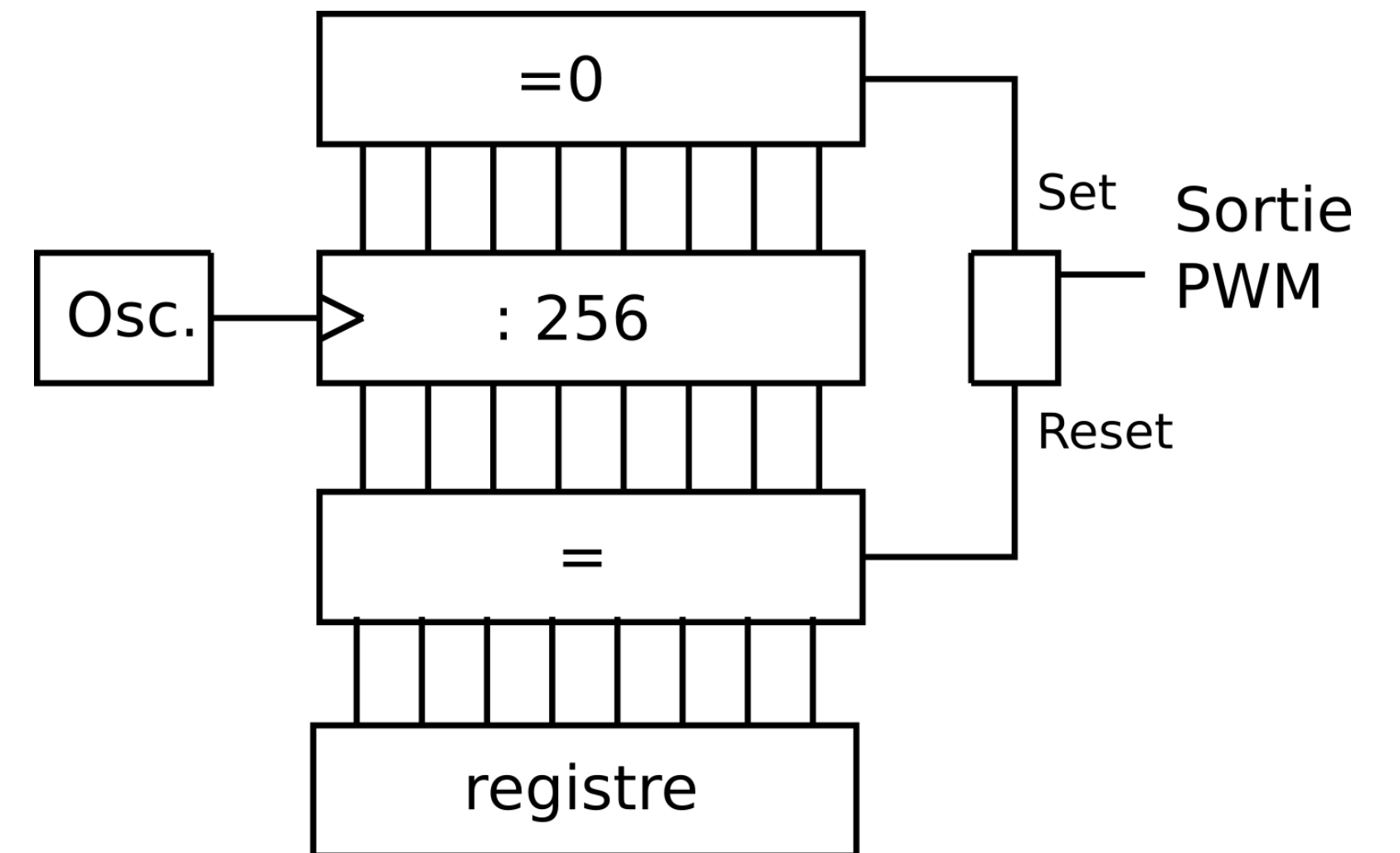
# PWM réalisé avec des circuits logiques



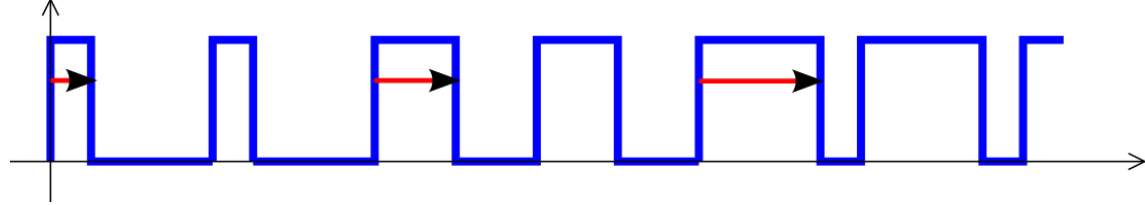
# PWM réalisé avec des circuits logiques



# C'est le Timer d'un microcontrôleur



# PWM : Modulation de Largeur d'Impulsion

- Principe : 
- Fréquence :  $> 100$  Hz pour l'oeil
- Programmer un PWM (occupe le proc.)
- Convertisseur DAC (+ filtre)
- Réalisation par des circuits logiques, inclus dans les microcontrôleurs