

NOM :

BTS Systèmes Numériques

TP 23 : Échantillonnage (suite) & CNA

1ère partie

(1 H 30)

Étude d'un montage l'échantillonneur-bloqueur

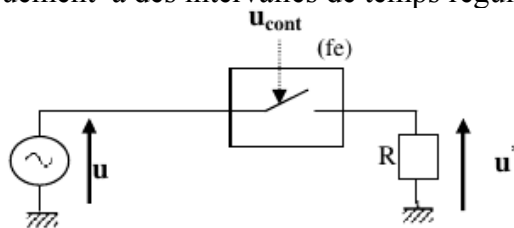
Matériel : circuit CMOS 4066
 $R = 1 \text{ k}\Omega$
 $C = 10 \text{ nF}$

L'étude porte sur un montage échantillonneur-bloqueur construit autour de l'interrupteur électronique CMOS 4066.

A - Étude temporelle

1 - Principe de l'échantillonnage

Pour échantillonner un signal analogique $u(t)$ et le transformer en une suite discrète d'échantillons $u^*(t)$, on prélève périodiquement à des intervalles de temps réguliers T_e la valeur de ce signal.



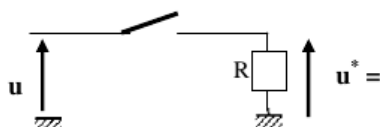
L'interrupteur analogique est commandé à l'ouverture ou à la fermeture par un signal de contrôle

$u_{\text{cont}}(t)$:

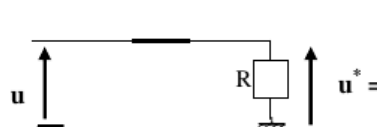
- si $u_{\text{cont}} = 0$, l'interrupteur est ouvert,
- si $u_{\text{cont}} = V_{DD} = 5 \text{ V}$, l'interrupteur est fermé.

➤ Préciser la valeur ou l'expression de u^* dans chacun des deux cas :

Si $u_{\text{cont}} = 0$:



Si $u_{\text{cont}} = 5 \text{ V}$:



NOM :

2 - Montage

Le brochage du quadruple interrupteur bidirectionnel CMOS 4066 est donné ci-contre.

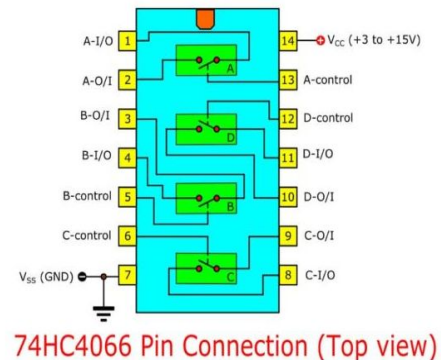
- Utiliser de l'alimentation stabilisée de 0-5V pour l'alimentation VSS (broche 7) -VDD (broche 14) du circuit.

Utiliser le premier interrupteur situé entre les broches 1, 2 et 13.

- Brancher la résistance $R = 1\text{ k}\Omega$ entre les broches 2 et 7.

Préparer sur le GBF les deux tensions u et u_{cont} :

- Voie 1 :
Le signal de contrôle $u_{\text{cont}}(t)$ suite d'impulsions, variant de 0 à 5 V, de fréquence $f_e = 10\text{ kHz}$ de rapport cyclique faible 3 % par exemple (signal de type pulse)
- Voie2 :
La tension sinusoïdale $u(t)$ positive, variant de 0 à 2 V, de fréquence $f = 1\text{ kHz}$



Bien vérifier la connection des masses entre elles et faire vérifier le montage avant d'appliquer la tension u sur la broche 1 du support, la tension u_{cont} sur la broche 15. Observer le signal d'entrée $u(t)$ et la tension échantillonnée $u^*(t)$ à l'oscilloscope.

3 - Mesures

- **Compter** le nombre d'échantillons N prélevés par période. Le comparer au rapport f_e/f .
- **Modifier** le rapport cyclique du signal de contrôle à 30 %. Sur une copie d'écran indiquer les phases de fermeture et d'ouverture de l'interrupteur.

4 - Blocage

Remettre le rapport cyclique à 1 % et remplacer la résistance R par un condensateur C de 10 nF.

- **Quelle** modification observez-vous sur la tension de sortie $u_s(t)$?
- **Quel** rôle joue le condensateur ?

B - Spectre des signaux

1 - Spectre de la tension $u(t)$

Préciser les valeurs théoriques des deux harmoniques de la tension u :

$$\text{Composante continue : } \begin{cases} f_0 = \\ \hat{U}_0 = \end{cases} \quad \text{Fondamental : } \begin{cases} f_1 = \\ \hat{U}_1 = \end{cases}$$

NOM :

2 - Spectre de la tension de sortie $u^*(t)$ échantillonnée

Remplacer le condensateur C par la résistance R.

Utiliser la fonction FFT de l'oscilloscope numérique (menu Math) pour observer le spectre de la tension $u^*(t)$ obtenue en sortie lors d'un débit sur la résistance R.

Réglages préconisés : *plage 50 kHz , centre 25kHz*

affiner le tracé en observant le signal temporel sur plusieurs périodes

- **Quel** est l'effet de l'échantillonnage sur la forme du spectre ?
- **Commenter** les valeurs des fréquences des harmoniques observées.

3 - Spectre de la tension de sortie u_{eb} échantillonnée et bloquée

Remplacer la résistance R par le condensateur C.

- **Commenter** le spectre de u_{eb} .

2nde partie

(1 H 30)

Conversion Numérique / Analogique avec Arduino

Méthode 1 : CAN avec n sorties numériques et un montage électronique

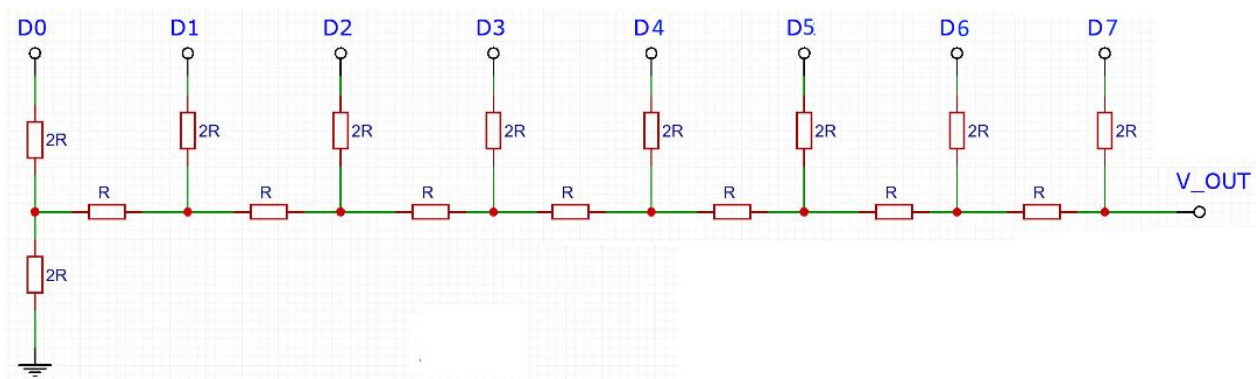
Objectifs :

Les sorties sur l'Arduino sont toutes numériques. Ces sortie peuvent générer seulement deux niveau de tension : un niveau « BAS » ou « 0 » de 0V ou un niveau « HAUT » ou « 1 » de 5V.

Comment dès lors générer avec une carte Arduino une tension de 2V par exemple ?

Comment à partir d'une carte Arduino créer une tension réglable entre 0 et 5V.

Plusieurs solutions sont possible, nous utiliserons le **circuit R-2R** suivant :



D0 à D7 représentent 8 sorties numériques de l'Arduino. Les tensions D0 à D7 valent donc 0V ou 5V. La tension V_{OUT} est la tension générée par ce circuit.

Cette tension, nous allons le voir est réglable et continue. Nous venons de constituer un CNA.

Ce circuit avec l'Arduino a été implémentée dans la simulation [Tinkercad CNA R-2R](#). Il vous sera certainement nécessaire de créer un compte Tinkercad pour pouvoir éditer le code de la simulation.

Le code pour démarrer est le suivant :

```
uint8_t n = B00000000;

void setup()
{
    DDRD = B11111111; // déclaration des broches du port D en sortie (OUTPUT)
}

void loop()
{
    PORTD = n;
}
```

NOM :

Quelques explications du code précédent :

Le Port D de l'Arduino Uno vise les broches numériques 0 à 7.

Le registre DDRD (1 octet) du port D détermine le sens d'utilisation des broches numériques en tant qu'ENTREE ou SORTIE ce qui permet d'éviter l'utilisation répétitive de la fonction [pinmode](#).

Le registre PORTD (1 octet) du port D fixe le niveau 0V (0 ou BAS) ou 5V (1 ou HAUT) des broches numérique en sortie ce qui permet d'éviter l'utilisation répétitive de la fonction [digitalWrite](#).

- Écrire PORTD = B00000000 ; met au niveau 0 (0V) les broches 0 à 7 de l'Arduino.
- Écrire PORTD = B11111111 ; met à mettre au niveau 1 (5V) les broches 0 à 7 de l'Arduino.
- Écrire PORTD = B00000001 ; met au niveau 1 la broche 0 et au niveau 0 les broches 1 à 7.
- etc ...

1. Au lancement de la simulation $V_{OUT}=0V$. Changer la variable n dans le code et **remplir** le tableau suivant :

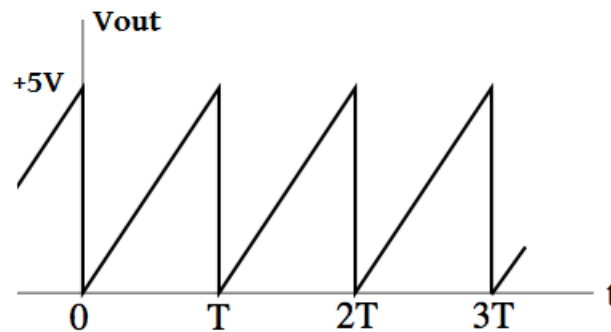
n (décimal)	n (binaire)	V_OUT
0		
1		
2		
4		
16		
17		
32		
64		
65		
128		
129		
255		

2. Pourquoi la variable n ne peut elle pas dépasser 255 ?
3. Déterminer la tension maximale en sortie du CNA, la tension pleine échelle notée V_{PE} .
4. La plus petite différence de tension, le plus petit « saut » que peut fournir le CNA est appelé quantum q. Évaluer q ?
5. En supposant le CNA linéaire (V_{out} proportionnel à n), quelle est la valeur théorique du quantum : q_{th} ? Comparer avec le résultat précédent.

NOM :

Les questions 6 à 8 sont optionnelles

Nous voulons maintenant générer une tension en dent de scie à partir du code précédent :



6. **Faire** évoluer le code. **Expliquer** votre raisonnement.
7. Visualiser la dent de scie à l'oscilloscope.
Évaluer la période T de la dent de scie (régler le temps par division). Expliquer la valeur pour cette période. Et estimer le temps d'une boucle loop.
Qu'indique le voltmètre ?
8. **Modifier** votre code pour augmenter cette période avec la fonction [`delay\(\)`](#) ou la fonction [`delayMicroseconds\(\)`](#)
9. **Proposer** une solution pour réaliser un signal en dent de scie évoluant de 1V à 4 V avec une fréquence de 5 Hz.

Vous pouvez vous inspirer du code suivant :

```
uint8_t n = B00000000;

void setup()
{
    DDRD = B11111111;
}

void loop()
{
    PORTD = n;
    n++;
    if (n==10) n = 0;
    delay(1);
}
```

NOM :

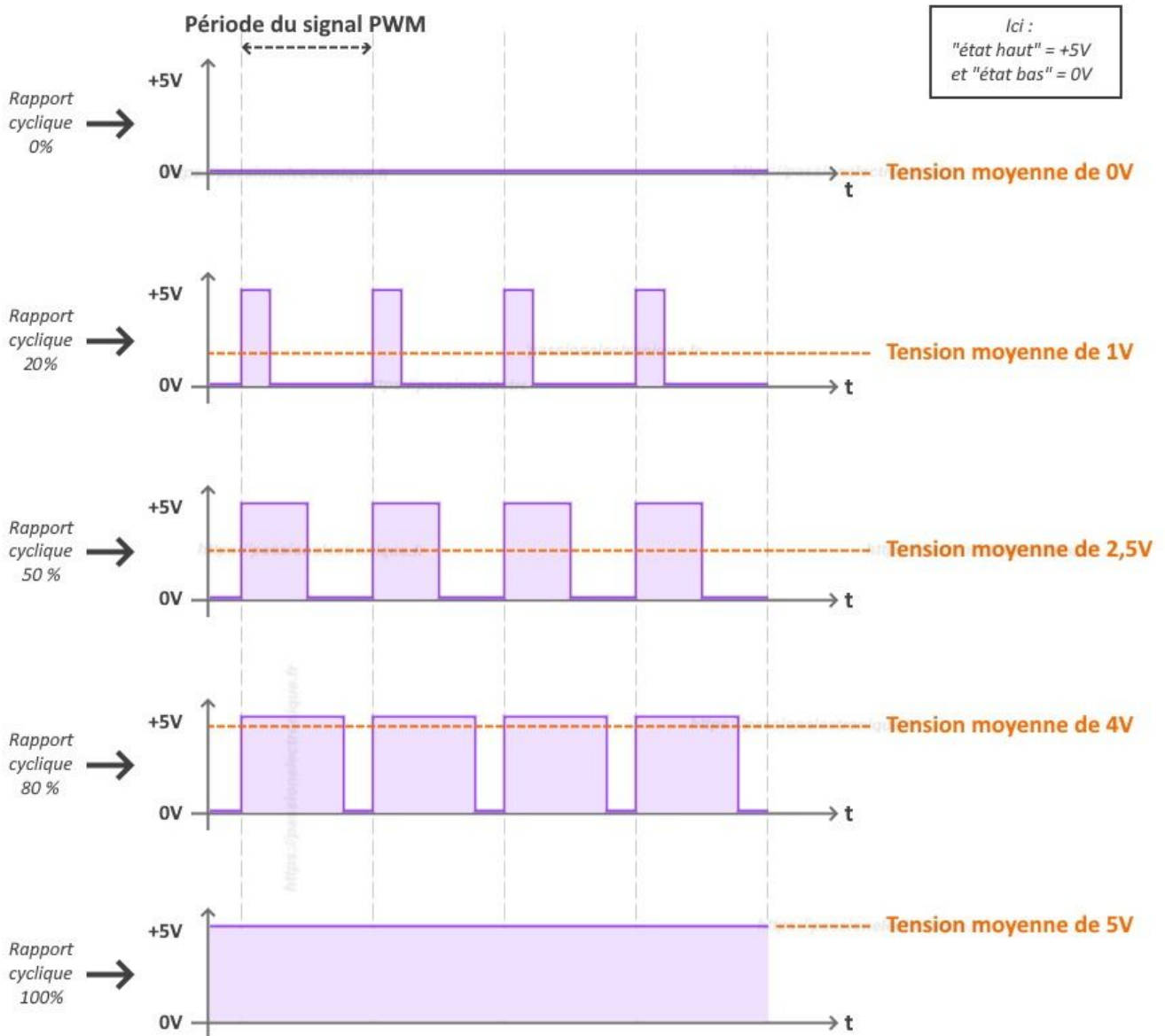
Méthode 2 : CAN via un signal PWM ou MLI

"PWM" signifie : **Pulse Width Modulation** et en français cela donne **Modulation à Largeur d'Impulsion** (MLI).

Un signal PWM est un signal numérique (0V-5V) de **fréquence** donnée, qui a un **rapport cyclique** qui peut être fixé et modifier par programmation.

Le rapport cyclique, $\alpha = t_{\text{haut}}/T$ est mesuré en pourcentage (%). Plus le pourcentage est élevé, plus le niveau logique 1 est présent dans la période et moins le niveau logique 0 l'est.

Exemple de signaux PWM (avec différents rapports cycliques)



PWM (Pulse Width Modulation) = MLI (Modulation de Largeur d'Impulsion)



PassionElectronique.fr

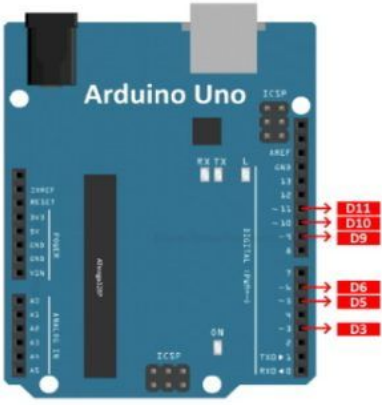
NOM :

Avec un arduino, on peut facilement générer un signal PWM en sortie sur une des pins D3, D5, D6, D9, D10 ou D11 repérées avec le symbole "~".

À noter que celui-ci sera :

- Uniquement « diffusable » sur certaines broches de sorties spécifiques
- De fréquence fixe (modifiable dans le code, dans une certaine limite, en changeant la valeur de certains registres du microcontrôleur).
Par défaut la fréquence est de 490,2 Hz pour les pins 3, 9, 10, 11 et de 976,6 Hz pour les pins 5, 6. Ces fréquences sont obtenues à partir de celle du quartz de l'Arduino UNO à 16 MHz.
- D'amplitude 3,3 ou 5 volts (selon si le microcontrôleur fonctionne en 3V3 ou 5V)
- Et de rapport cyclique modifiable logiciellement, avec la fonction « analogWrite » ou via les registres du μC

Fréquences PWM de l'Arduino Uno (ATmega328P)



Sorties D5 et D6
→ dépend du timer0 (registres TCCR0A et TCCR0B)
→ sortie D5 câblée sur OC0B et D6 sur OC0A

		Prédiviseur du timer 0 (registre TCCR0B)				
		1	8	64	256	1024
Mode	Fast PWM	62500 Hz	7812,5 Hz	976,6 Hz	244,1 Hz	61 Hz
	Phase Correct PWM	31372,5 Hz	3921,6 Hz	490,2 Hz	122,5 Hz	30,6 Hz

(en gris les valeurs "par défaut")

Sorties D9 et D10
→ dépend du timer1 (registres TCCR1A et TCCR1B)
→ sortie D9 câblée sur OC1A et D10 sur OC1B

		Prédiviseur du timer 1 (registre TCCR1B)				
		1	8	64	256	1024
Mode	Fast PWM, 8-bit	62500 Hz	7812,5 Hz	976,6 Hz	244,1 Hz	61 Hz
	Fast PWM, 9-bit	31250 Hz	3906,3 Hz	488,3 Hz	122,1 Hz	30,5 Hz
	Fast PWM, 10-bit	15625 Hz	1953,1 Hz	244,1 Hz	61 Hz	15,3 Hz
	Phase Correct PWM, 8-bit	31372,5 Hz	3921,6 Hz	490,2 Hz	122,5 Hz	30,6 Hz
	Phase Correct PWM, 9-bit	15655,6 Hz	1956,9 Hz	244,6 Hz	61,2 Hz	15,3 Hz
	Phase Correct PWM, 10-bit	7820,1 Hz	977,5 Hz	122,2 Hz	30,5 Hz	7,6 Hz

(en gris les valeurs "par défaut")

Sorties D3 et D11
→ dépend du timer2 (registres TCCR2A et TCCR2B)
→ sortie D3 câblée sur OC2B et D11 sur OC2A

		Prédiviseur du timer 2 (registre TCCR2B)						
		1	8	32	64	128	256	1024
Mode	Fast PWM	62500 Hz	7812,5 Hz	1953,1 Hz	976,6 Hz	488,3 Hz	244,1 Hz	61 Hz
	Phase Correct PWM	31372,5 Hz	3921,6 Hz	980,4 Hz	490,2 Hz	245,1 Hz	122,5 Hz	30,6 Hz

(en gris les valeurs "par défaut")

Remarques :

- (1) le changement de mode PWM se fait via les bits WGM02 à WGM00 pour le timer 0, WGM13 à WGM10 pour le timer 1, et WGM22 à WGM20 pour le timer 2 (ces bits étant "à cheval" sur les registres TCCRnA et TCCRnB).
- (2) le changement de rapport de division de fréquence se fait via les bits CS02 à CS00 pour le timer 0, CS13 à CS10 pour le timer 1, et CS22 à CS20 pour le timer 2 (ces bits étant situés dans les registres TCCRnB).
- (3) le mode "Fast PWM" est un mode où les signaux PWM sont "découpés" en 2^n pas (par exemple 256 pas, en mode Fast PWM-8 bits). C'est un mode rapide, mais moins précis que le "Phase Correct PWM".
- (4) le mode "Phase Correct PWM" est un mode où les signaux PWM sont "découpés" en $(2^n-1)*2$ pas (par exemple 510 pas, en mode PWM-8 bits). C'est un mode plus précis, mais plus "lent" que le "FastPWM".
- (5) les modes "Fast PWM" et "Phase Correct" présentés ici, sont ceux qui considèrent le parcours de leur timer de "BOTTOM" à "MAX" (c'est-à-dire, le "full range" du timer auquel ils sont associés).
→ Sous-entendu qu'il existe d'autres modes de génération de signaux PWM, pouvant porter le même nom, mais avec des valeurs de registre et comportements différents (et donc, avec d'autres fréquences potentielles possibles)

PassionElectronique.fr

Transformation PWM => signal analogique :

La valeur moyenne du signal PWM est proportionnelle à son rapport cyclique : $U = \alpha \cdot 5V$.

De ce fait, si on modifie le rapport cyclique de la PWM de façon maîtrisée, on va pouvoir créer un signal analogique de la forme qu'on le souhaite, compris entre 0 et 5V, en extrayant la valeur moyenne du signal.

Pour extraire cette valeur, il suffit d'employer un filtre passe-bas de coupure inférieure à la fréquence PWM, par exemple avec un circuit RC.

Parfois c'est directement un système physique qui le rôle d'un filtre passe bas (notre œil par ex) ...

NOM :

Manipulations :

Génération d'un signal PWM avec arduino :

- Déclarer « en sortie » une des broches pouvant émettre un signal PWM, ce qui se fait avec la commande **pinMode(numero_de_la_broche_souhaitée, OUTPUT);**
- Et lui envoyer la valeur de rapport cyclique souhaité, avec la commande **analogWrite(numero_de_la_broche_souhaitée, valeur_de_rapport_cyclique_souhaité)**

À noter que le rapport cyclique, au niveau du code arduino, est exprimé au travers d'un nombre compris entre 0 et 255.

exemple 1 : "générer un signal PWM"

```
#define brochePwm 3 // On choisit d'émettre sur la broche D3
#define RapportCycliquePWM 30 // avec un rapport cyclique égal à 30%

void setup() {

    // Conversion du rapport cyclique en % en une valeur comprise entre 0 et 255
    int rapportCycliqueEntre0et255 = map(RapportCycliquePWM, 0, 100, 0, 255);

    // Génération du signal PWM
    pinMode(brochePwm, OUTPUT); // Définition de la broche D5 en tant que « SORTIE »
    analogWrite(brochePwm, rapportCycliqueEntre0et255); // Génération du signal PWM
}

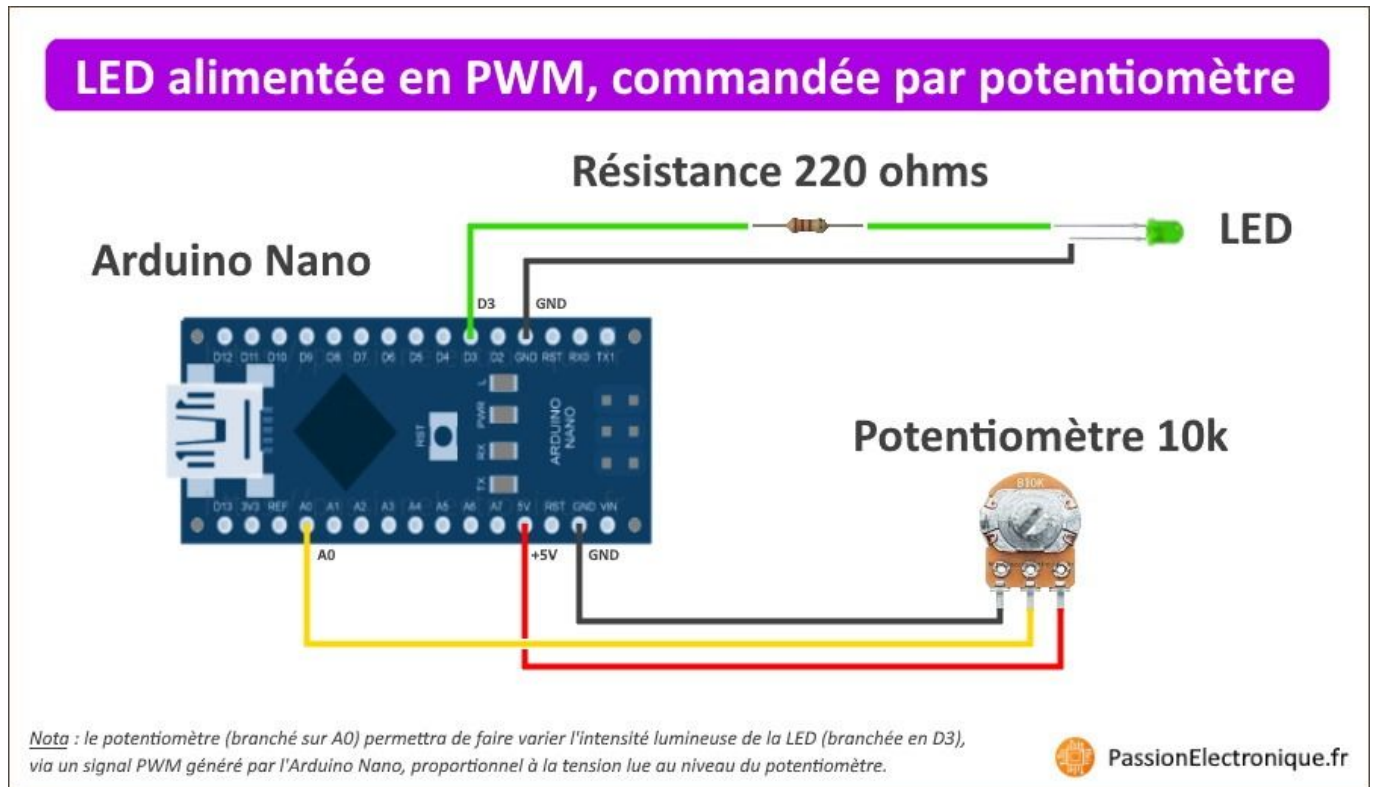
void loop() {

    // Pas de code ici, car tout se passe dans la partie setup !
}
```

Charger le code et **visualiser** à l'oscilloscope le signal PWM. (copie écran)

NOM :

exemple 2 : "variation de la luminosité d'une LED avec un potentiomètre"



```
#define pinPotentiometre A0 // Le potentiomètre sera branché sur l'entrée A0
#define pinLED 3 // La LED sera branchée sur la sortie D3 de l'Arduino

int TensionAnalogique; // Variable qui contiendra la valeur de la
tension mesurée sur l'entrée analogique (valeur comprise entre 0 et 1023, car lecture
sur 10 bits)
int RapportCycliquePwm; // Variable qui contiendra la valeur du rapport
cyclique du signal PWM à générer

// =====
// Initialisation programme
// =====
void setup()
{
    // Définition de la broche où sera branché la LED en sortie
    pinMode(pinLED, OUTPUT);

    // Nota : pas besoin de déclarer l'entrée analogique en entrée, car c'est sous
    entendu, par défaut
}

// =====
// Boucle principale
// =====
void loop()
{

```

NOM :

```
//*****  
// Lecture de la tension présente sur l'entrée analogique, où est branché le  
potentiomètre (son "point milieu", plus exactement)  
// *****  
// Pour rappel : la valeur retournée sera comprise entre 0 et 1023, car il s'agit  
là d'une lecture sur 10 bits (0 correspondant à 0V, et 1023 à +5V)  
  
TensionAnalogique = analogRead(pinPotentiometre);  
  
// *****  
// Conversion tension -> rapport cyclique  
// *****  
// Comme la valeur mesurée sur l'entrée analogique sera exprimée sous la forme d'un  
nombre compris entre 0 et 1023,  
// et que la valeur du rapport cyclique à renseigner dans le code arduino devra  
être comprise entre 0 et 255,  
// alors il faut convertir les mesures 0-1023 en valeur 0-255  
  
RapportCycliquePwm = map(TensionAnalogique, 0, 1023, 0, 255);  
  
// *****  
// Génération du signal PWM  
// *****  
  
analogWrite(pinLED, RapportCycliquePwm);  
}
```

Charger le code et tester.

Exemple 3 : Produire une couleur parmi 16 millions (256*256*256)

inspirez vous du code précédent avec cette fois ci la led RGB (trois led reunies). Il vous faudra 3 potentiomètres.

