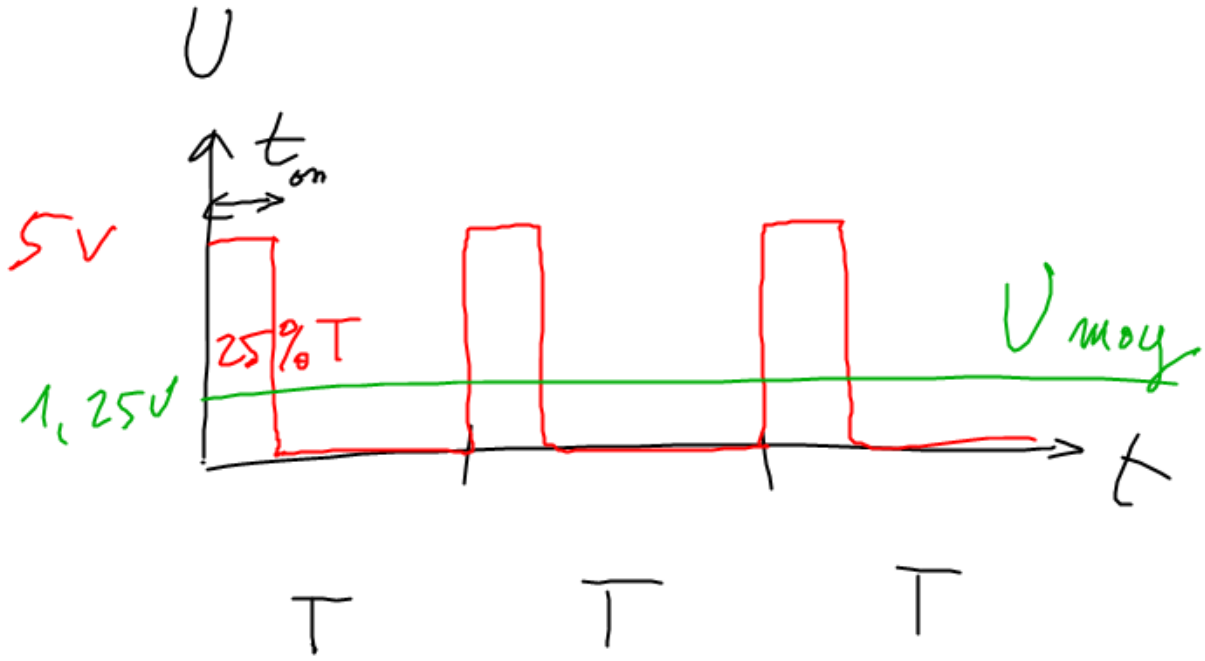


$U_{moy} = \alpha \cdot U_{max}$

$= 0{,}25 \cdot 5$

$= 1{,}25\,V$

α : rapport cyclique

$\alpha = \frac{t_{on}}{T} = 25\%$

- La tension moyenne de sortie et le rapport cyclique en fonction de la valeur de commande.

Complétez le tableau ci-dessous sachant que le rapport cyclique et la tension moyenne de sortie sont proportionnels à la valeur de commande.

Valeur de commande (de 0 à 255)	Rapport cyclique (t_{on} / T)	Tension moyenne de sortie
0	0 %	0 V
1	0,392 %	0,0196 V
64	25 %	1,25 V
128	50 %	2,5 V
192	75 %	3,76 V
219	86 %	4,3 V
255	100 %	5 V

Remarque : la valeur de commande doit être une variable de type entier notée ''int'' en programmation (integer).

4. Programmation d'un MLI sous IDE Arduino

Du coté programmation, le signal MLI (ou PWM) est généré par la fonction : `analogWrite(pin_MLI,valeur_MLI);`

A partir du montage du cours « Entrées numériques » et du code à compléter ci-dessous (voir fichier prog_4_LED_analogWrite), réalisez un programme permettant d'allumer progressivement la led par paliers de 5 toutes les 200 ms sur la commande du MLI après pression du bouton poussoir.
Evaluez le nombre de paliers puis le temps d'allumage de la led (et la vérifier)

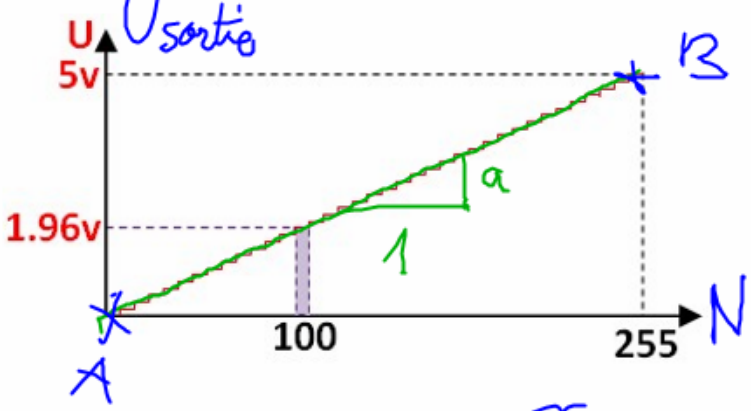
Sur la carte Arduino Uno, seuls les ports numériques précédés d'un tilde " ~ " sont capables de générer un signal MLI, cela concerne les broches 3, 5, 6, 9, 10 et 11.

Le signal MLI de la carte Arduino Uno est caractérisé par :

- La fréquence des créneaux : de 500 Hz pour les broches 3, 5, 10 et 11 à 1000 Hz pour les broches 5 et 6
- L'amplitude des créneaux : 5V
- La profondeur du convertisseur numérique/analogique (CNA) : 8 bits

Le nombre de bits du convertisseur numérique/analogique (CNA) correspond au nombre de valeurs pseudo-analogiques possibles. Un CNA de 8 bits permet une commande sur 256 valeurs (2^8), dans notre cas de 0 à 255 en décimal.

Conséquence : cela scinde la tension moyenne de "5V" de sortie en 256 pallier donc 255 valeurs (2^8-1).



A partir de ces informations on peut calculer, en fonction des besoins :

- Le quantum, qui correspond à la plus petite variation de tension du CNA (pour un bit) :

$$q = \frac{Amplitude\ des\ créneaux}{(2^{nombre\ de\ bits\ du\ CNA}) - 1} =$$

$y = a \cdot x$

$U_{sortie} = q \cdot N$

$\frac{U_{sortie}}{q} = N$

$a = \frac{y_B - y_A}{x_B - x_A} = \frac{y_B}{1 x_B} = \frac{5}{255} = q$

A partir du montage du cours « Entrées numériques » et du code à compléter ci-dessous (voir fichier prog_4_LED_analogWrite), réalisez un programme permettant d'allumer progressivement la led par paliers de 5 toutes les 200 ms sur la commande du MLI après pression du bouton poussoir.
Evaluez le nombre de paliers puis le temps d'allumage de la led (et le vérifier).

prog_4_LED_analogWrite §

```
int broche_LED = 3;
int broche_BP = 2;
int etat_BP;

void setup() {
  pinMode(broche_LED, OUTPUT);
  pinMode(broche_BP, INPUT);
}

void loop() {

  etat_BP = digitalRead(broche_BP);   //enregistre l'état du bouton poussoir dans la variable etat_BP

    if(etat_BP == 1) {
    for(int i = 0; i <= 255; i =i+5 ){
      analogWrite(broche_LED, i);
      delay( 200 );
    }
  }
  else
  {
    digitalWrite(broche_LED, 0);
  }
}
```

$t = \frac{200 \cdot 255}{a}$ ex: $t = 2000$ $a \simeq 25$

$a = \frac{200 \cdot 255}{t}$

temps boucle for

$t = 200 \cdot \frac{255}{5} \simeq 10200\,ms$

```
if (etat_BP == 1) {
  for (int i = 0; i <= 255    ; i = i + 25     ) {
    analogWrite(broche_LED, i);
    delay(  200   );
  }
  for (int i = 255; i >= 0    ; i = i - 25     ) {
    analogWrite(broche_LED, i);
    delay(  200   );
  }
```

i = i + 10

delay (20)

else
{
}

https://www.toutes-les-couleurs.com/code-co

prog_5_RGB_analogWrite

```
int bouton = 2;
int etat_bouton;

int led_R = 11;
int valeur_R = 255; // de 0 à 255
int led_G = 10;
int valeur_G =126 ; // de 0 à 255
int led_B = 9;
int valeur_B = 67 ; // de 0 à 255


void setup(){
  pinMode(bouton,INPUT);
  pinMode(led_R,OUTPUT);
  pinMode(led_B,OUTPUT);
  pinMode(led_G,OUTPUT);
  //mise à zéro
  analogWrite (led_R,0);
  analogWrite (led_B,0);
  analogWrite (led_G,0);
}

void loop(){

  etat_bouton = digitalRead(bouton);
  if(etat_bouton == 1) {
    analogWrite(led_R,valeur_R);
```

Pantone 164C

Hexa : #FF7E43

R:255 V:126 B:67

programme ci-dessous (voir fichier prog_6_RGB_analogWrite_seq) :

prog_6_RGB_analogWrite_seq

```
int bouton = 2;
int etat_bouton;

int led_R = 11;
int Seq_R[]={ 255, 100,      };
int led_G = 10;
int Seq_G[]={ 126, 160,      };
int led_B = 9;
int Seq_B[]={ 67, 20,        };

void setup(){
  pinMode(bouton,INPUT);
  pinMode(led_R,OUTPUT);
  pinMode(led_G,OUTPUT);
  pinMode(led_B,OUTPUT);
  //mise à zéro
  analogWrite (led_R,0);
  analogWrite (led_G,0);
  analogWrite (led_B,0);
}

void loop(){

  etat_bouton = digitalRead(bouton);
  if(etat_bouton == 1) {
    for(int i = 0;i <=       ;i++){
      analogWrite(led_R,Seq_R[i]);
      analogWrite(led_G,Seq_B[i]);
      analogWrite(led_B,Seq_G[i]);
      delay(       );
    }
```

nombre de valeurs - 1

500