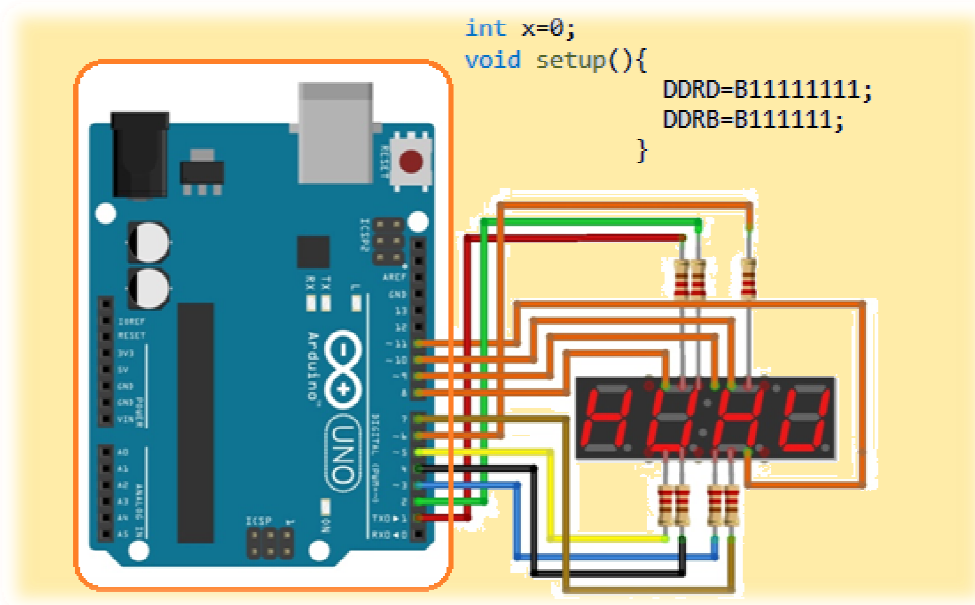


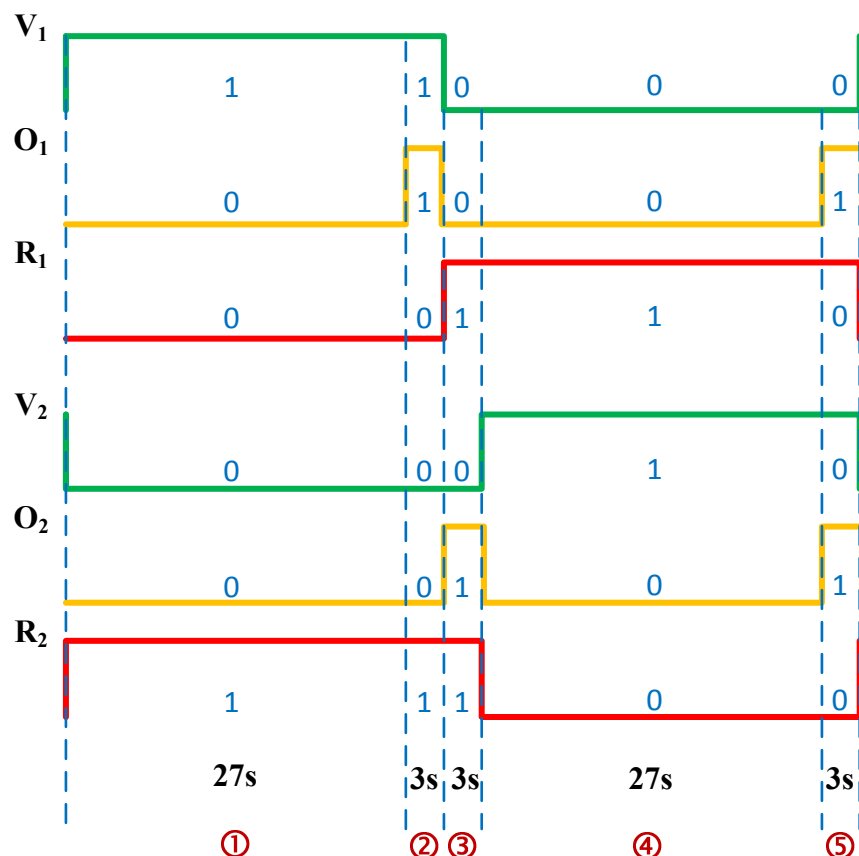
# Corrigé du TD4

---



**Exercice N°1**

Avant d'écrire le programme on subdivise les chronogrammes en durées pendant lesquelles toutes les lampes n'ont pas changé d'état.



N°	Etat des lampes						Durée en secondes
	V <sub>1</sub>	O <sub>1</sub>	R <sub>1</sub>	V <sub>2</sub>	O <sub>2</sub>	R <sub>2</sub>	
1	1	0	0	0	0	1	27
2	1	1	0	0	0	1	3
3	0	0	1	0	1	1	3
4	0	0	1	1	0	0	27
5	0	1	1	1	1	0	3

Pour éviter les instructions inutiles, on ne commande à chaque fois que les lampes qui ont changé d'état (cases du tableau remplies et marquées en rouge).

Programme :

```
//Commande de feux de carrefour
const byte V1=3;
const byte O1=4;
const byte R1=5;
const byte V2=6;
const byte O2=7;
const byte R2=8;

void setup() {
  for(int i=3;i<9;i++)
  {
    pinMode(i,OUTPUT);
  }
}

void loop() {
  //Commande état 1
  digitalWrite(V1,1);
  digitalWrite(O1,0);
  digitalWrite(R1,0);
  digitalWrite(V2,0);
  digitalWrite(O2,0);
  digitalWrite(R2,1);
  delay(27000);

  //Commande état 2
  digitalWrite(O1,1);
  delay(3000);

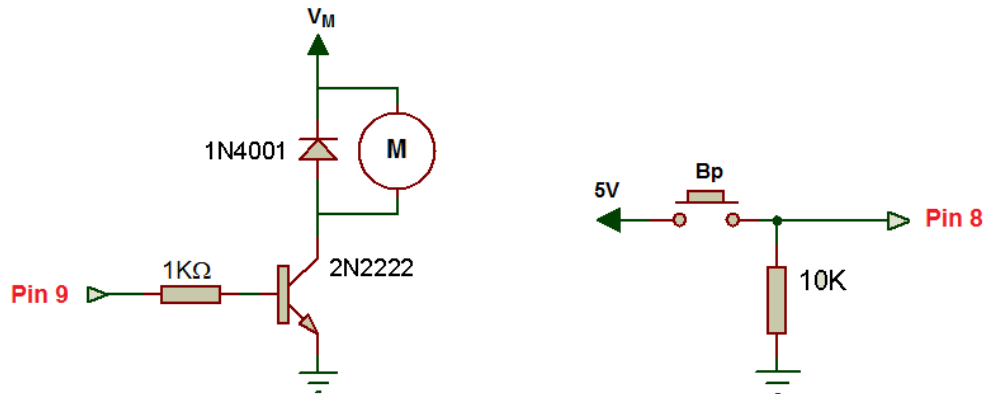
  //Commande état 3
  digitalWrite(V1,0);
  digitalWrite(O1,0);
  digitalWrite(R1,1);
  digitalWrite(O2,1);
  delay(3000);

  //Commande état 4
  digitalWrite(V2,1);
  digitalWrite(O2,0);
  digitalWrite(R2,0);
  delay(27000);

  //Commande état 5
  digitalWrite(O1,1);
  digitalWrite(O2,1);
  delay(3000);
}
```

**Exercice N°2**

Commande d'un moteur à courant continu (5V) dans un seul sens de rotation via une carte Arduino UNO.

**1. Commande à vitesse constante :****a. Erreurs que contient le programme donné.**

Ligne	Instruction initiale	Instruction modifiée
7	Manque le « ; » à la fin de la ligne	byte etat=0 ;
17	La variable « bouton » n'existe pas	etat=digitalRead(Bouton) ;
22	On utilise 0 et non pas 100	digitalWrite(Moteur,0) ;

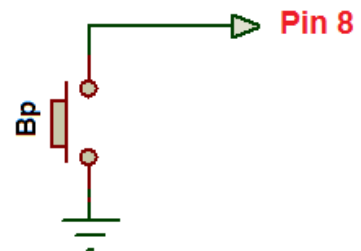
**b. Programme optimisé :**

```
void loop() {
  etat=digitalRead(Bouton) ;
  digitalWrite(Moteur,etat) ;
}
```

On peut aussi écrire le programme de la façon suivante :

```
void loop() {
  digitalWrite(Moteur, digitalRead(Bouton) ) ;
}
```

- c. Lorsqu'on branche le bouton poussoir de la façon ci-contre, l'entrée de la carte arduino (pin 8) se trouve à zéro quelque soit l'état du bouton (ouvert ou fermé). Pour résoudre ce problème sans ajouter de composant matériel on doit ajouter par programmation une résistance dite de tirage en déclarant l'entrée Bouton comme suite :

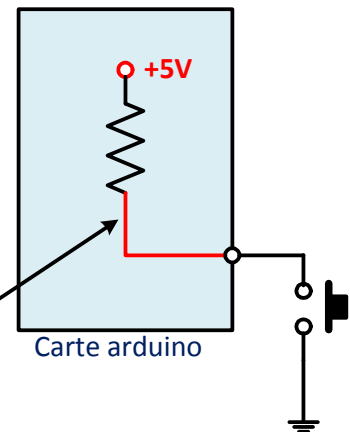


```
pinMode(Bouton, INPUT_PULLUP);
```

Une telle configuration permet de brancher automatiquement une résistance de tirage interne entre +5V et l'entrée voulue de la carte arduino.

On remarque que l'entrée reliée à +5V (égale à 1) lorsque le bouton est ouvert et à GND (égale à 0) lorsque le bouton est fermé.

Liaison interne établie lorsque l'entrée est configurée INPUT\_PULLUP



Il faut changer le programme initial de la façon suivante :

Ligne	Erreur	Correction
12	<code>pinMode(Bouton , INPUT) ;</code>	<code>pinMode(Bouton , INPUT_PULLUP) ;</code>
19	<code>if (etat==1)</code>	<code>if (etat==0) ou bien if ( !etat)</code>

Le programme optimisé devient :

```
void loop() {
    etat=digitalRead(Bouton) ;
    digitalWrite(Moteur, !etat) //on remplace etat par !etat
}
```

2. Pour faire varier la vitesse du moteur il faut utiliser une sortie PWM (marqué ~ sur la carte arduino). Ici on utilise le pin 9 qui est bien une sortie PWM.
  - a. Fixer la valeur au début du programme.

```
//Commande d'un moteur à courant continu dans un seul sens de //rotation
à vitesse fixée au début du programme.

const byte Moteur=9;
byte vitesse=150;          //on fixe ici la vitesse

void setup() {
    // configuration des entrées/sorties
    pinMode(Moteur,OUTPUT);
}

void loop() {
    analogWrite(Moteur, vitesse); //commande du moteur à la vitesse
                                //choisie
}
```

**b.** Lecture de la valeur de la vitesse sur le moniteur série.

```
//Commande d'un moteur à courant continu dans un seul sens de //rotation
à vitesse introduite sur le moniteur série.

const byte Moteur=9;
int vitesse;

void setup() {
  // configuration des entrées/sorties:
  pinMode(Moteur,OUTPUT);
  Serial.begin(9600);
  while (! Serial);
  Serial.println("Vitesse entre 0 et 255");
}

void loop() {
  if (Serial.available())
  {
    vitesse = Serial.parseInt();
    if (vitesse >= 0 && vitesse <= 255)
    {
      analogWrite(Moteur, vitesse);
    }
  }
}
```

**c.** Variation de la vitesse au moyen d'un potentiomètre

```
//Commande d'un moteur à courant continu dans un seul sens de //rotation
et à vitesse variable au moyen d'un potentiomètre

const byte Moteur=9;
int pot=A0;
int vitesse=0; // variable dans laquelle on lit la valeur de l'entrée
               // analogique A0

void setup() {
  pinMode(Moteur,OUTPUT);
}

void loop() {
  vitesse = analogRead(pot); //lecture de la valeur donnée par le
                             //potentiomètre
  vitesse = map(vitesse,0,1023,0,255); //convertir la valeur stockée
                                       //dans vitesse de 0-1023 à 0-255
  analogWrite(Moteur, vitesse); //commander le moteur
}
```

- d. Utilisation de deux boutons poussoirs, le premier permet d'augmenter la vitesse et le second permet de la diminuer.

```
//Commande d'un moteur à courant continu dans un seul sens de //rotation
et à vitesse variable au moyen de deux boutons

const byte Moteur=9;
const byte Bouton1=8;
const byte Bouton2=7;

int vitesse=0;
int vitessemin=50; //vitesse minimale, 50 par exemple
int vitessemax=255; //vitesse maximale
int incdec=20; //variable pour incrémenter ou décrémenter
//la vitesse (20 par exemple).

byte etat1=0;
byte etat2=0;

void setup() {
  pinMode(Moteur,OUTPUT);
  pinMode(Bouton1,INPUT);
  pinMode(Bouton2,INPUT);
}

void loop() {
  etat1 = digitalRead(Bouton1);
  etat2 = digitalRead(Bouton2);

  if (etat1==1)
  {
    vitesse = vitesse + incdec; //augmenter la vitesse
    if (vitesse > vitessemax)
    {
      Vitesse = vitessemax;
    }
  }

  if (etat2==1)
  {
    vitesse = vitesse - incdec; //diminuer la vitesse
    if (vitesse <= vitessemin)
    {
      Vitesse = vitessemin;
    }
  }

  analogWrite(Moteur, vitesse); //commander le moteur
}
```

**Exercice N°3**

Commande d'un afficheur 7 segments au moyen d'une carte arduino UNO.

1. D'après le schéma de branchement donné, on remarque que la broche commune de l'afficheur est reliée à la broche GND de la carte arduino. Donc l'afficheur est type à **cathode commune**.
2. Puisque l'afficheur est à cathode commune, l'allumage d'un segment se fait par le niveau logique 1. D'où les mots permettant d'afficher les chiffres de 0 à 9.

Segment	g	f	E	d	c	b	a
pin	8	7	6	5	4	3	2
0	0	1	1	1	1	1	1
1	0	0	0	0	1	1	0
2	1	0	1	1	0	1	1
3	1	0	0	1	1	1	1
4	1	1	0	0	1	1	0
5	1	1	0	1	1	0	1
6	1	1	1	1	1	0	1
7	0	0	0	0	1	1	1
8	1	1	1	1	1	1	1
9	1	1	0	1	1	1	1

3. Réalisation d'un compteur modulo 10

```
//Commande d'un afficheur 7 segments 1 digit
//Compteur modulo 10

int seg_a = 2;
int seg_b = 3;
int seg_c = 4;
int seg_d = 5;
int seg_e = 6;
int seg_f = 7;
int seg_g = 8;
int chiffre=0;

void setup()
{
  for (int i=2; i < 9; i++)
  {
    pinMode(i, OUTPUT);
  }
}
```



```
void loop()
{
  for (int i=0; i < 10; i++)
  {
    afficher(i);
    delay(1000);
  }
}

void afficher(int chiffre)
{
  switch (chiffre) {
    case 0:
      digitalWrite(seg_a,1);
      digitalWrite(seg_b,1);
      digitalWrite(seg_c,1);
      digitalWrite(seg_d,1);
      digitalWrite(seg_e,1);
      digitalWrite(seg_f,1);
      digitalWrite(seg_g,0);
      break;

    case 1:
      digitalWrite(seg_a,0);
      digitalWrite(seg_b,1);
      digitalWrite(seg_c,1);
      digitalWrite(seg_d,0);
      digitalWrite(seg_e,0);
      digitalWrite(seg_f,0);
      digitalWrite(seg_g,0);
      break;

    case 2:
      digitalWrite(seg_a,1);
      digitalWrite(seg_b,1);
      digitalWrite(seg_c,0);
      digitalWrite(seg_d,1);
      digitalWrite(seg_e,1);
      digitalWrite(seg_f,0);
      digitalWrite(seg_g,1);
      break;

    case 3:
      digitalWrite(seg_a,1);
      digitalWrite(seg_b,1);
      digitalWrite(seg_c,1);
      digitalWrite(seg_d,1);
      digitalWrite(seg_e,0);
      digitalWrite(seg_f,0);
      digitalWrite(seg_g,1);
      break;
  }
}
```

```
case 4:
    digitalWrite(seg_a,0);
    digitalWrite(seg_b,1);
    digitalWrite(seg_c,1);
    digitalWrite(seg_d,0);
    digitalWrite(seg_e,0);
    digitalWrite(seg_f,1);
    digitalWrite(seg_g,1);
    break;

case 5:
    digitalWrite(seg_a,1);
    digitalWrite(seg_b,0);
    digitalWrite(seg_c,1);
    digitalWrite(seg_d,1);
    digitalWrite(seg_e,0);
    digitalWrite(seg_f,1);
    digitalWrite(seg_g,1);
    break;

case 6:
    digitalWrite(seg_a,1);
    digitalWrite(seg_b,0);
    digitalWrite(seg_c,1);
    digitalWrite(seg_d,1);
    digitalWrite(seg_e,1);
    digitalWrite(seg_f,1);
    digitalWrite(seg_g,1);
    break;

case 7:
    digitalWrite(seg_a,1);
    digitalWrite(seg_b,1);
    digitalWrite(seg_c,1);
    digitalWrite(seg_d,0);
    digitalWrite(seg_e,0);
    digitalWrite(seg_f,0);
    digitalWrite(seg_g,0);
    break;

case 8:
    digitalWrite(seg_a,1);
    digitalWrite(seg_b,1);
    digitalWrite(seg_c,1);
    digitalWrite(seg_d,1);
    digitalWrite(seg_e,1);
    digitalWrite(seg_f,1);
    digitalWrite(seg_g,1);
    break;
```

```

    case 9:
        digitalWrite(seg_a,1);
        digitalWrite(seg_b,1);
        digitalWrite(seg_c,1);
        digitalWrite(seg_d,1);
        digitalWrite(seg_e,0);
        digitalWrite(seg_f,1);
        digitalWrite(seg_g,1);
        break;
    }
}

```

#### 4. Programme optimisé en utilisant les 8 pins du port D de la carte arduino

On doit modifier le branchement de l'afficheur de la façon suivante :

Segment	g	f	E	d	c	b	a	NC	Pin 0 du port D non connecté on le met toujours à 0
Pins du port D	7	6	5	4	3	2	1	0	

```

//Comande d'un afficheur 7 segments 1 digit en utilisant le port D
//Compteur modulo 10

//Tableau contenant les mots de commande en binaire correspondants //aux
10 chiffres.
int seg[]={B01111110,B00001100,B10110110,B10011110,B11001100,
           B11011010,B11111010,B00001110,B11111110,B11011110};

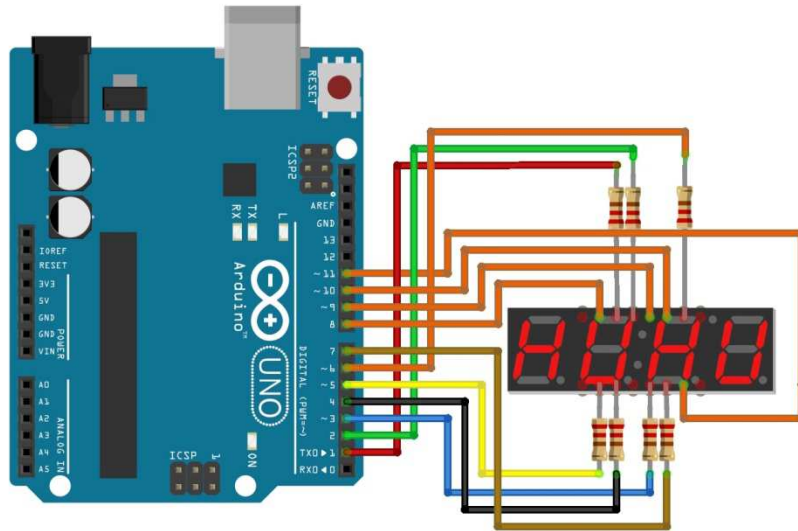
void setup(){
    DDRD=B11111111; //configure les 8 pins du port D
    }               //en sortie

void loop()
{
    for(int i=0;i<10;i++)
    {
        PORTD = seg[i]; //Envoyer au PORTD la commande du chiffre i
        delay(1000);
    }
}

```

**Exercice N°4**

Commande d'un afficheur 4 digits.

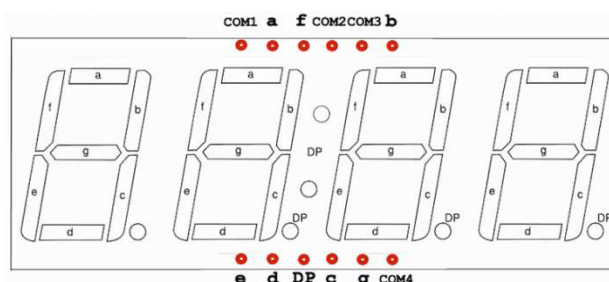


1. Ce circuit comporte principalement une carte arduino UNO, un afficheur 7segments 4 digits et 7 résistances de 220Ω pour protéger l'afficheur.
2. Puisque l'afficheur est à anode commune, l'allumage d'un segment se fait par le niveau logique 0. D'où les mots permettant d'afficher les chiffres de 0 à 9.

Broche afficheur	g	f	e	D	c	b	a	DP	COM4	COM3	COM2	COM1
Pin arduino	7	6	5	4	3	2	1	NC	11	10	9	8
0	1	0	0	0	0	0	0	-	0	0	0	1
1	1	1	1	1	0	0	1	-	0	0	0	1
2	0	1	0	0	1	0	0	-	0	0	0	1
3	0	1	1	0	0	0	0	-	0	0	0	1
4	0	0	1	1	0	0	1	-	0	0	0	1
5	0	0	1	0	0	1	0	-	0	0	0	1
6	0	0	0	0	0	1	0	-	0	0	0	1
7	1	1	1	1	0	0	0	-	0	0	0	1
8	0	0	0	0	0	0	0	-	0	0	0	1
9	0	0	1	0	0	0	0	-	0	0	0	1

On a validé ici le digit 1 (COM1=1) seulement à titre d'exemple

NC : Non Connecté.



### 3. Exemple de programme de commander l'afficheur 4 digits :

```
int seg[]={B10000000,B11110010,B01001000,B01100000,B00110010,
           B00100100,B00000100,B11110000,B00000000,B00100000};

int d1=8,d2=9,d3=10,d4=11;    //4 variables pour la validations des
                                //4 digits
int x=1436;                    //entier quelconque

void setup() {
    DDRD=B11111111; //configurer les pins du PORTD en sortie
    DDRB=B111111;   //configurer les pins du PORTB en sortie
}

void loop() {
    digitalWrite(d1,1); //activer le premier digit
    digitalWrite(d2,0); //désactiver le deuxième digit
    digitalWrite(d3,0); //désactiver le troisième digit
    digitalWrite(d4,0); //désactiver le quatrième digit
    PORTD=seg[x/1000];   //afficher la valeur 1436/1000=1
    delay(2);           //temporisation très courte
    PORTD=B11111111;    //éteindre tous les segments

    digitalWrite(d1,0);
    digitalWrite(d2,1); //activer le deuxième digit
    digitalWrite(d3,0);
    digitalWrite(d4,0);
    PORTD=seg[x%1000/100]; //afficher la valeur 436/100=4
    delay(2);
    PORTD=B11111111;

    digitalWrite(d1,0);
    digitalWrite(d2,0);
    digitalWrite(d3,1); //activer le troisième digit
    digitalWrite(d4,0);
    PORTD=seg[x%1000%100/10]; //afficher la valeur 36/10=3
    delay(2);
    PORTD=B11111111;

    digitalWrite(d1,0);
    digitalWrite(d2,0);
    digitalWrite(d3,0);
    digitalWrite(d4,1); //activer le quatrième digit
    PORTD=seg[x%1000%100%10]; // afficher la valeur 6
    delay(2);
    PORTD=B11111111;
}
```

Ce programme permet d'afficher le nombre 1436 sur un afficheur 7 segments 4 digits. Cet afficheur possède 7 entrées, relatives aux 7 segments, communes aux 4 digits. Il faut donc réaliser un affichage multiplexé c'est-à-dire commander les différents digits l'un après l'autre en ajoutant à chaque fois une très courte temporisation (2ms).

Dans ce programme on effectue des divisions successives par 1000 puis par 100 puis par 10 et ce pour extraire les différents chiffres à afficher.

Le symbole / désigne division entière et le symbole % désigne le reste de la division.

#### 4. Programme réalisant un compteur modulo 10000.

On écrit une fonction nommée « **seconde()** » qui permet de répéter l'affichage d'un chiffre un certain nombre de fois de façon à avoir une durée presque égale à une seconde puis on appelle cette fonction dans une boucle de 0 à 9999 dans le programme principal (fonction loop()).

```
int seg[]={B10000000,B11110010,B01001000,B01100000,B00110010,
           B00100100,B00000100,B11110000,B00000000,B00100000};

int d1=8,d2=9,d3=10,d4=11;
int x=0;
void setup(){
    DDRD=B11111111;
    DDRB=B111111;
}

void seconde()
{
    for(int i=0;i<50;i++)
    {
        PORTB=B001110;
        PORTD=seg[x/1000];
        delay(2);
        PORTD=B00000000;

        PORTB=B001101;
        PORTD=seg[x%1000/100];
        delay(2);
        PORTD=B00000000;

        PORTB=B001011;
        PORTD=seg[x%1000%100/10];
        delay(2);
        PORTD=B00000000;

        PORTB=B000111;
        PORTD=seg[x%1000%100%10];
        delay(2);
        PORTD=B00000000;
    }
}

void loop()
{
    for(x=0;x<10000;x++)
    {
        seconde();
    }
}
```