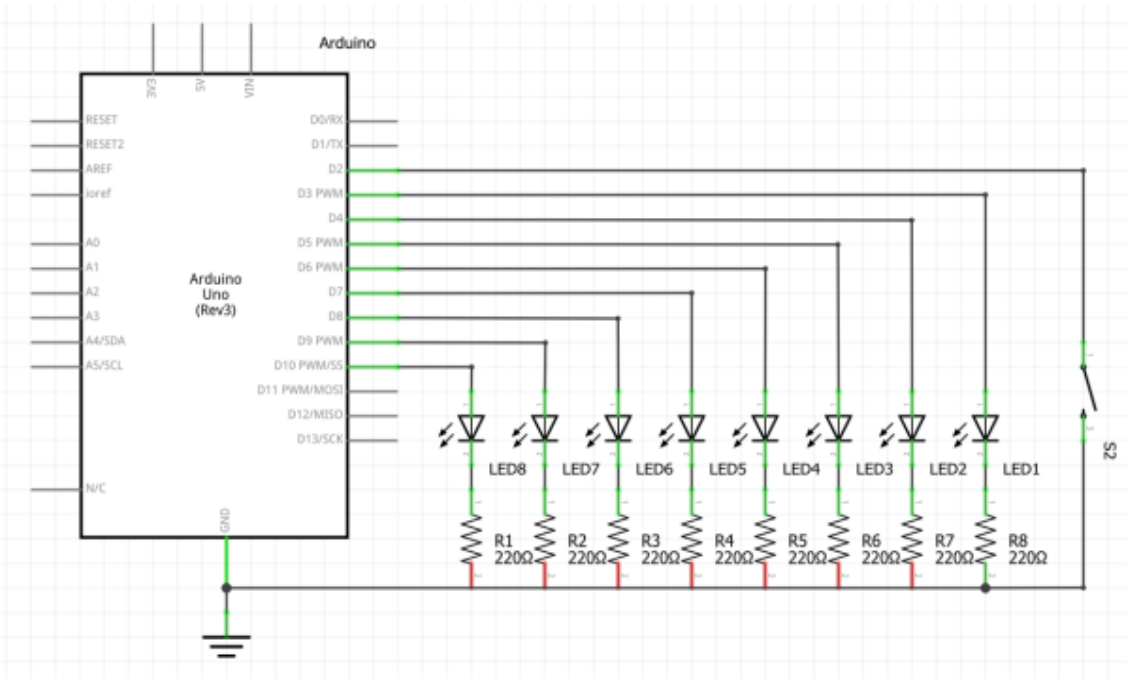
Cours : Architecture systèmes, systèmes embarqués et IoT

Correction TD1

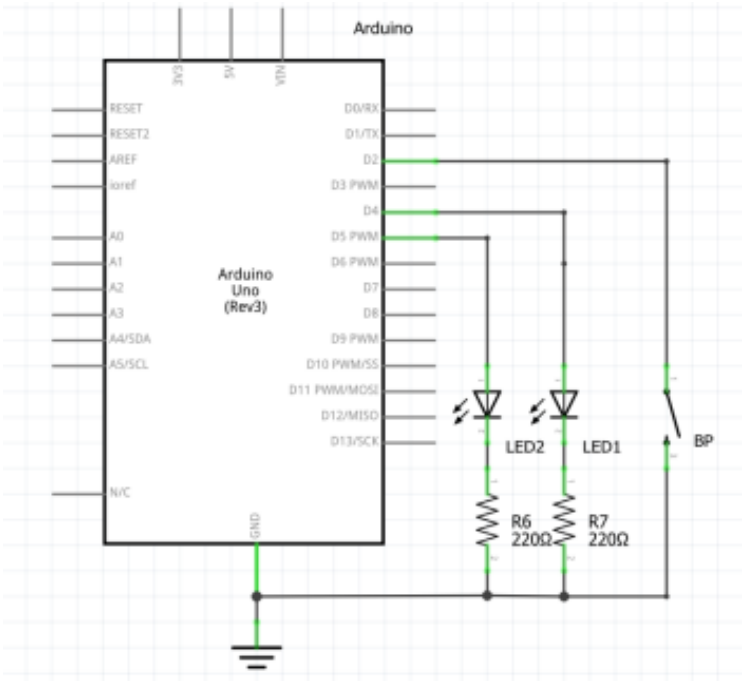
Exercice1 :




```
int Broche ;
const int Boutton = 2;
long Intervalle = 600;
int switchState;
void setup(){
  Serial.begin(9600);
  for(Broche = 3; Broche <= 10; Broche++)
  {
    pinMode(Broche,OUTPUT);
    digitalWrite(Broche, LOW);
  }
pinMode(Boutton ,INPUT);

}
void loop(){
switchState = digitalRead(2);
if (switchState == HIGH)
{
  for(Broche = 3; Broche <= 10; Broche++)
```

```
   {
      digitalWrite(Broche, HIGH);
      delay(Intervalle);
      Serial.print("LED allumé");
      digitalWrite(Broche, LOW);
      delay(Intervalle);
      Serial.print("LED éttinte");
   }
}
else {
 for(Broche = 10; Broche >= 3; Broche--)
   {
      digitalWrite(Broche, HIGH);
      delay(Intervalle);
      Serial.print("LED allumé");
      digitalWrite(Broche, LOW);
      delay(Intervalle);
      Serial.print("LED éttinte");

   }

  }
}
```

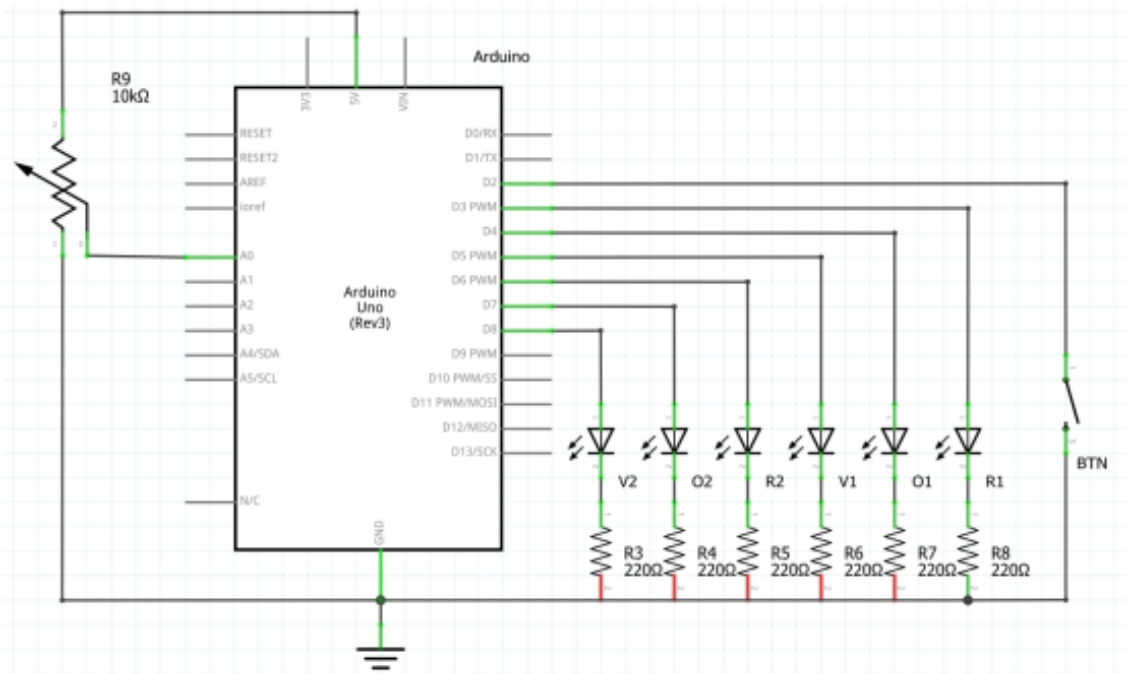
Exercice2




```
#define PinLED1 4
#define PinLED2 5
```

```
const int Boutton = 2;

int switchState;
void setup(){
  Serial.begin(9600);
  pinMode(PinLED1,OUTPUT);
  pinMode(PinLED2,OUTPUT);
  digitalWrite(PinLED1,LOW);
  digitalWrite(PinLED2,LOW);
  pinMode(Boutton ,INPUT_PULLUP);

}
void loop(){
switchState = digitalRead(2);
if (switchState == LOW)
{
      digitalWrite(PinLED1, HIGH);
      delay(1000) ;
      digitalWrite(PinLED2, HIGH);
      delay(1000);
      digitalWrite(PinLED1, LOW);
      delay(1000);
      digitalWrite(PinLED2, LOW) ;}
}
```

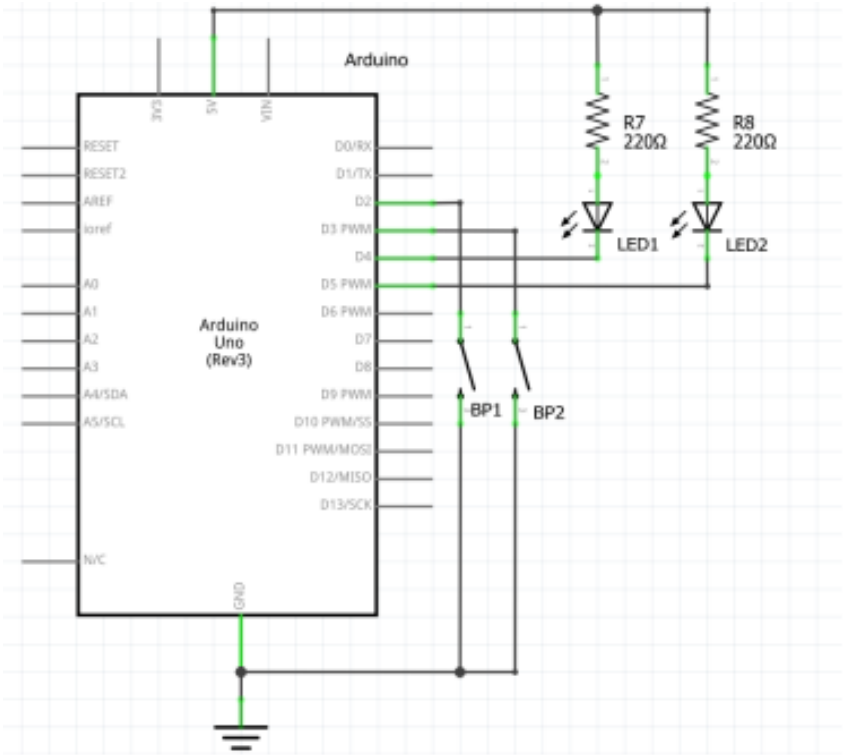
Exercice3



```
const int R1=3, O1= 4, V1 =5, R2=6, O2= 7, V2 =8;
int Broche ;
const int Bouton = 2;
long Pot, val;
void setup(){
  Serial.begin(9600);
  for(Broche = 3; Broche <= 8; Broche++)
  {
    pinMode(Broche,OUTPUT);
    digitalWrite(Broche, LOW);
  }
pinMode(Bouton ,INPUT_PULLUP);
}
void loop()
{
Pot = analogRead(A0);
Serial.println(Pot);
val = map(Pot, 0, 1023, 30000, 132000);
Serial.println(val);
delay(1000);
if (digitalRead(Bouton) == LOW) { //Bouton pressé
digitalWrite(V1, HIGH);
digitalWrite(R2,HIGH);
delay(val);
digitalWrite(V1, LOW);
digitalWrite(O1, HIGH);
delay(3000);
digitalWrite(O1, LOW);
digitalWrite(R1, HIGH);
delay(val-3000);
digitalWrite(R2,LOW);
digitalWrite(V2,HIGH);
delay(val);
digitalWrite(V2, LOW);
digitalWrite(O2, HIGH);
delay(3000);
digitalWrite(O2, LOW);
digitalWrite(R2, HIGH);
delay(val-3000);
digitalWrite(R1,LOW);
digitalWrite(V1, HIGH);
}
else { //Bouton relâché
digitalWrite(O1,HIGH);
digitalWrite(O2,HIGH);
delay(1500);
```

```
digitalWrite(O1,LOW);
digitalWrite(O2,LOW);
delay(1500);
}
}
```

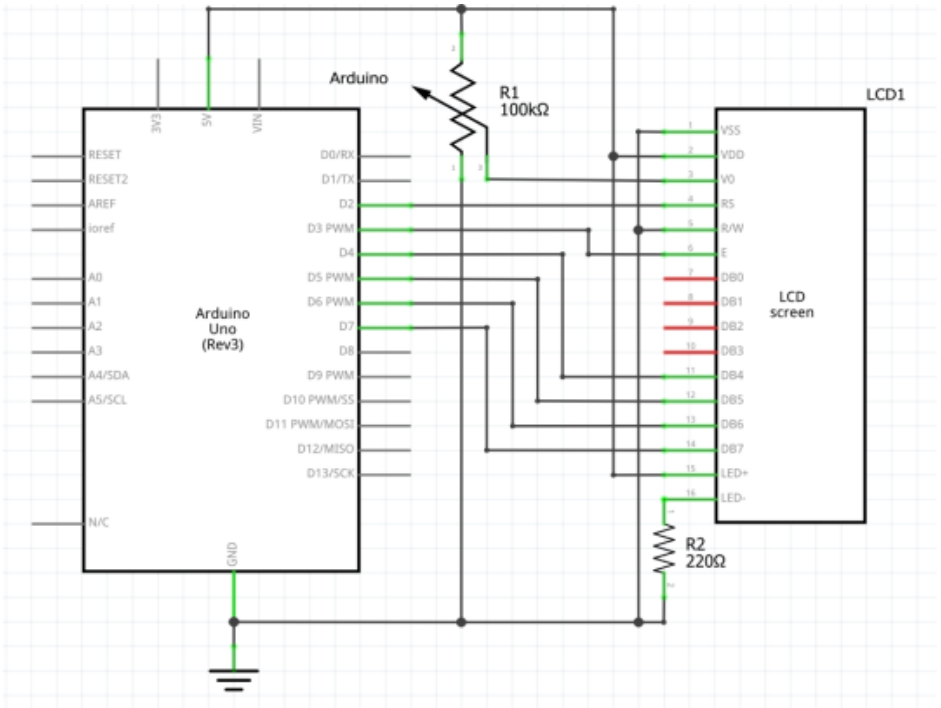
Exercice 4




```
// les broches des boutons
const int btn_entree = 2;
const int btn_sortie = 3;
// les leds de signalements
const int led_rouge = 4;
const int led_verte = 5;
// les mémoires d'état des boutons
int mem_entree = HIGH;
int mem_sortie = HIGH;
int nbr_voit = 0;
int etat = HIGH; // variable stockant l'état courant d'un bouton
void setup(){
  Serial.begin(9600);
    pinMode(btn_entree, INPUT_PULLUP);
    pinMode(btn_sortie, INPUT_PULLUP);
    pinMode(led_rouge, OUTPUT);
    pinMode(led_verte, OUTPUT);
    digitalWrite(led_rouge, LOW);
```

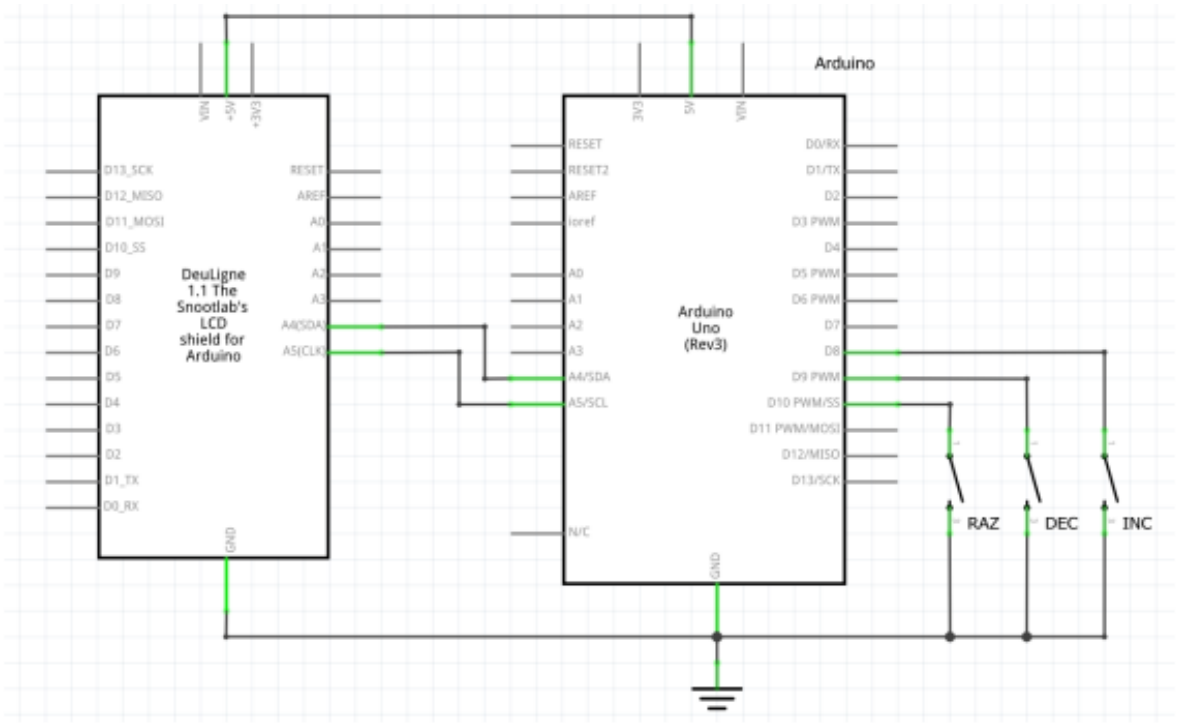
```
    digitalWrite(led_verte, HIGH); // vert par défaut
    Serial.println(nbr_voit);
    }

      void loop()
{
    // on test maintenant si les boutons ont subi un appui
    etat = digitalRead(btn_entree);
    if((etat != mem_entree) && (etat == LOW) && (nbr_voit <10))
      {  nbr_voit += 1;
      Serial.println(nbr_voit);
      }
    mem_entree = etat; // on enregistre l'état du bouton pour le tour suivant

    // et maintenant pareil pour le bouton qui décrémente
    etat = digitalRead(btn_sortie);
    if((etat != mem_sortie) && (etat == LOW)&& (nbr_voit >0))
       { nbr_voit -= 1;
       Serial.println(nbr_voit);}
        mem_sortie = etat; // on enregistre l'état du bouton pour le tour suivant

    if(nbr_voit < 10) // s'il y a des places
       { digitalWrite(led_rouge, LOW);
        digitalWrite(led_verte, HIGH);}
    else
       { digitalWrite(led_verte, LOW);
        digitalWrite(led_rouge, HIGH);}

}
```

Exercice5




```
#include <LiquidCrystal.h>
// Create an LCD object. Parameters: (RS, E, D4, D5, D6, D7):
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);
String myString;

void setup(){
    Serial.begin(9600);
     lcd.begin(16, 2);
}

void loop(){
    if(Serial.available()){
        myString = Serial.readString() ;
        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print(myString);
        Serial.println(myString);
    }
}
```

Exercice6




```
#include <Wire.h>
 #include <LiquidCrystal_I2C.h>
//----- Adressage matériel -----

#define valeurMin 0
#define valeurMax 10

int Pin_plus=8;
int Pin_moins=9;
int Pin_Zero=10;
int j;

LiquidCrystal_I2C lcd(0x27,16,2)
void setup() {
  lcd.init(); // initialisation de l'afficheur
  Serial.begin(9600);
  pinMode(Pin_plus,INPUT_PULLUP);
  pinMode(Pin_moins,INPUT_PULLUP);
  j=0;
  Serial.println(j);
  lcd.backlight(); // Envoi du message
  lcd.setCursor(0,0);
  lcd.print(j);
```

```
  delay(100);
}


void loop() {
  if (digitalRead(Pin_plus)==HIGH)
  {
    if (j<valeurMax)
    j++;
    Serial.println(j);
    lcd.backlight(); // Envoi du message
    lcd.setCursor(0,0);
    lcd.print(j);
    delay(100);
   while(digitalRead(Pin_plus)==HIGH);
  }

  if (digitalRead(Pin_moins)==HIGH)
  {
    if (j>valeurMin)
    j--;
    Serial.println(j);
    lcd.backlight(); // Envoi du message
    lcd.setCursor(0,1);
    lcd.print(j);
    delay(100);
   while(digitalRead(Pin_moins)==HIGH);
}
if (digitalRead(Pin_Zero)==HIGH)
  {
     j = 0;
    Serial.println(j);
    lcd.backlight(); // Envoi du message
    lcd.setCursor(0,1);
    lcd.print(j);
    delay(100);
   while(digitalRead(Pin_Zero)== HIGH);}
}
```