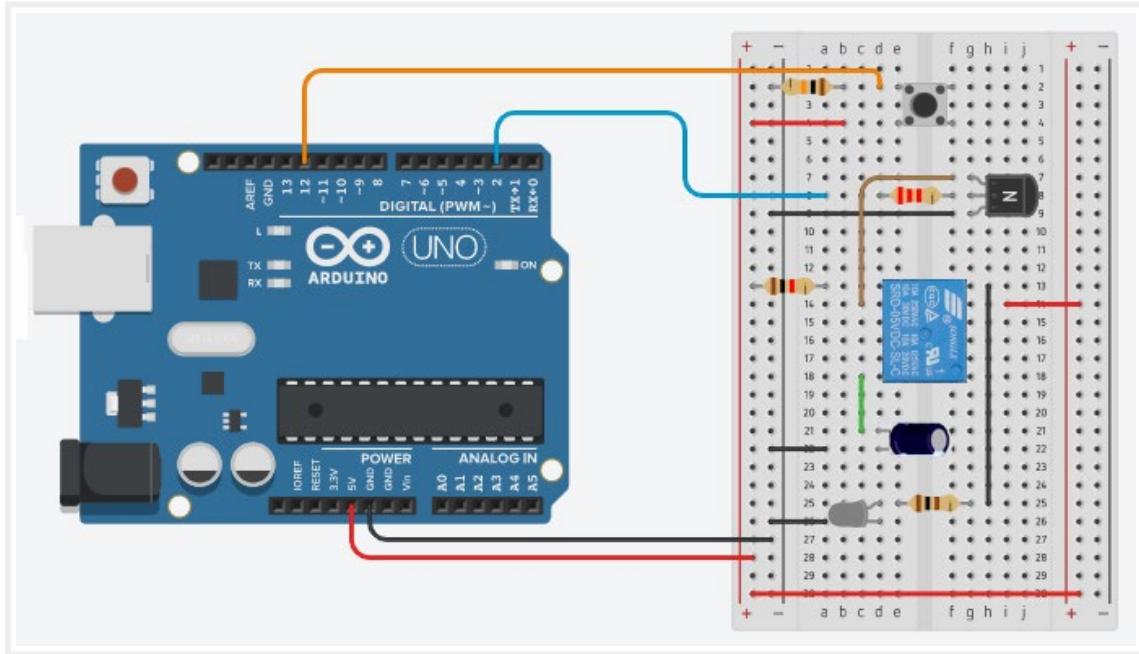


Dipôles RC – Phare

(Réaliser un flash périodique)



Liste des composants

- . 1 condensateur de 470 μF (C chimique : **attention à la polarité**)
- . 1 résistance de 10 $\text{k}\Omega$ (résistance du bouton poussoir)
- . 1 bouton poussoir
- . 1 transistor bipolaire NPN (BC547B)
- . 1 résistance de 2,2 $\text{k}\Omega$ (résistance du transistor)
- . 1 relais SRS-06VDC-SL
- . 1 résistance de 1 $\text{k}\Omega$ (résistance de charge du condensateur)
- . 1 résistance de 100 Ω (résistance de décharge du condensateur)
- . 1 DEL blanche

Objectif

Dans cette activité, nous allons modifier le programme de l'activité "[simulation d'un flash photographique](#)" pour que le flash s'effectue périodiquement après avoir appuyé sur le bouton poussoir.

Pour cela, il suffit de surveiller, en boucle, la tension U_c aux bornes du condensateur de façon à alterner entre la charge et la décharge :

- Si U_c est supérieure à une valeur **Uc max** à définir, le condensateur est déchargé, ce qui produit un flash.
- Si U_c est inférieure à une valeur **Uc min** à définir, le condensateur est chargé jusqu'à **Uc max**, puis le condensateur est déchargé jusqu'à U_c min et ainsi de suite...

Le flash périodique est arrêté en appuyant de nouveau sur le bouton poussoir.

. Le programme

Voici le code de l'activité, dont il sera possible de modifier les constantes **UcMax** et **UcMin** pour voir l'influence sur la période des flashes :

```
Dipoles_RC_Phare
// Déclaration des constantes et variables

const int PinUC = 0;
const int PinButton = 12;
const int PinRelay = 2;
const int UcMax = 1000;
const int UcMin = 800;

int ValPinUC = 0;
int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;

// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton, INPUT);
  pinMode(PinRelay, OUTPUT);
  digitalWrite(PinRelay, 0);
  while (ValPinUC < UcMax) {
    ValPinUC = analogRead(PinUC);
  }
  Serial.println("Appuyez sur le bouton poussoir pour commencer le flash.");
}

// Fonction principale en boucle

void loop() {

  ValButton = digitalRead(PinButton);
  delay(10);
```

```

if ((ValButton == HIGH) && (OldValButton == LOW))
{
    State=1-State;
}
OldValButton = ValButton;

if (State==1)
{
    if (OldState == 0)
    {
        Serial.println("Flash periodique en cours.");
        OldState=1;
    }
    ValPinUC = analogRead(PinUC);
    if (ValPinUC > UcMax)
    {
        digitalWrite(PinRelay, 1);
        while (ValPinUC>UcMin) {
            ValPinUC = analogRead(PinUC);
        }
    }
    else {
        digitalWrite(PinRelay, 0);
        while (ValPinUC<UcMax) {
            ValPinUC = analogRead(PinUC);
        }
    }
}
else{
    if (OldState == 1){
        Serial.println("Fin du flash.");
        OldState = 0;
        digitalWrite(PinRelay, 1);
        ValPinUC = analogRead(PinUC);
        while (ValPinUC>UcMin) {
            ValPinUC = analogRead(PinUC);
        }
    }
}
}
}

```

Déroulement du programme :

– 1. Déclaration des constantes et variables :

- . **const int PinUc = 0** (broche du condensateur : A0)
- . **const int PinButton = 12** (broche du bouton poussoir)
- . **const int PinRelay = 2** (broche du relais)
- . **const int UcMax = 1000** (valeur maximale de la broche du condensateur)
- . **const int UcMin = 800** (valeur minimale de la broche du condensateur)
- . **int ValPinUc = 0** (variable nombre entier pour stocker la valeur de la broche du condensateur)

int ValButton = 0 (variable nombre entier pour stocker la valeur de la broche du bouton poussoir)

. int OldValButton = 0 (variable nombre entier pour stocker la valeur précédente de la broche du bouton poussoir)

. int State = 0 (variable nombre entier correspondant à l'action à effectuer)

. int OldState = 0 (variable nombre entier correspondant à l'action effectuée précédemment)

– 2. Initialisation des entrées et sorties :

. Initialisation de la broche du relais en sortie,

. Initialisation de la broche du bouton poussoir en entrée,

. Charge du condensateur jusqu'à UcMax.

– 3. Fonction principale en boucle :

