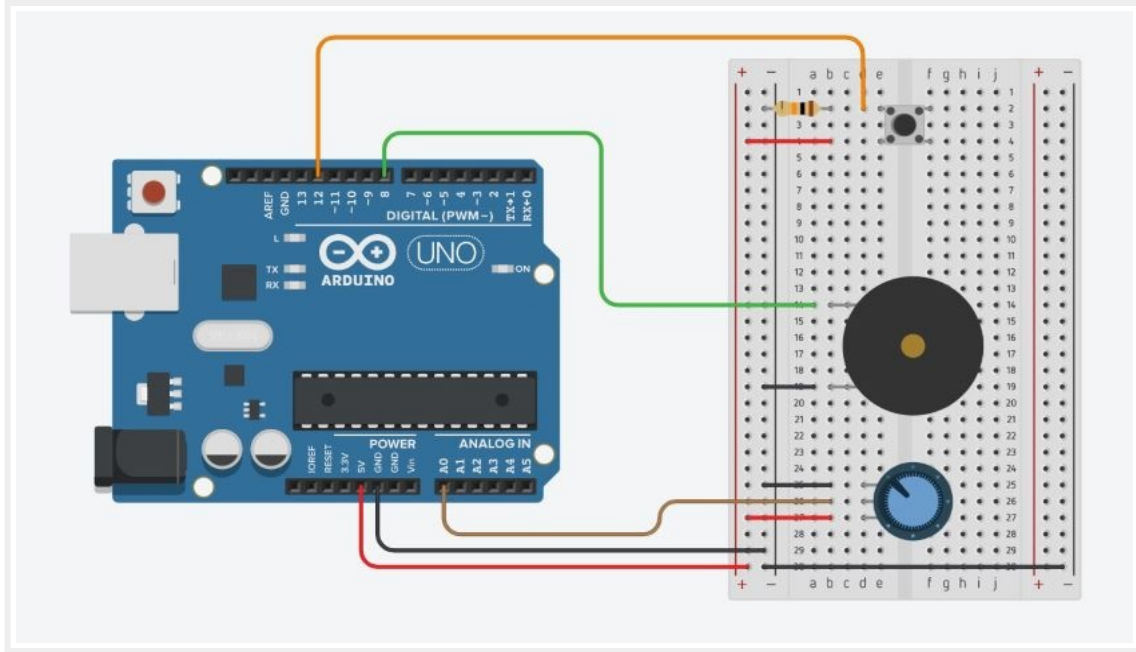


# Émission sonore

## (Régler la fréquence d'une onde sonore avec un potentiomètre)



### Liste des composants :

- . 1 bouton poussoir
- . 1 résistance de 10 k $\Omega$  (résistance du circuit du bouton poussoir)
- . 1 potentiomètre de 10 k $\Omega$
- . 1 buzzer
- . 1 plaque d'essai
- . Fils de connexion

### Objectif

Dans cette activité, l'appui sur le bouton-poussoir produit une onde sonore dont la fréquence est réglée à l'aide du potentiomètre. L'émission sonore est arrêtée en appuyant de nouveau sur le bouton poussoir.

Le potentiomètre est connecté sur la broche **A0** de l'Arduino. La tension de cette broche varie donc entre **0** et **5 V** (voir le principe de fonctionnement du potentiomètre de l'activité d'apprentissage des entrées analogiques) en fonction de la position du curseur du potentiomètre.

La lecture de la valeur de la broche **A0** convertie par le convertisseur analogique numérique de l'Arduino donne donc un nombre entier entre **0** et **1023** qui sera utilisé pour régler la fréquence du son émis, en le multipliant par un coefficient préalablement choisi par l'intermédiaire du moniteur série.

## . Envoi de données du moniteur série vers l'Arduino

Nous avons vu, avec le programme d'apprentissage des entrées analogiques, que l'on pouvait envoyer des données de l'Arduino vers le moniteur série avec la fonction "**Serial.print()**".

Il est également possible d'envoyer des données du moniteur série vers l'Arduino, en réponse à une demande d'information par le programme téléversé dans sa mémoire.

Pour cela, on va utiliser la fonction "**Serial.available()**" qui donne le nombre de caractères (octets) disponible pour lecture dans la file d'attente (buffer) du port série et la fonction "**Serial.read()**" qui lit les données contenues dans la mémoire tampon (buffer) du port série.

Les données seront envoyées sous la forme d'une chaîne de caractères par le moniteur série. L'Arduino va lire la chaîne, caractère par caractère, et la reconstituer.

Programme pour la réception d'une chaîne de caractères :



```
void setup() {
  Serial.begin(9600);
}

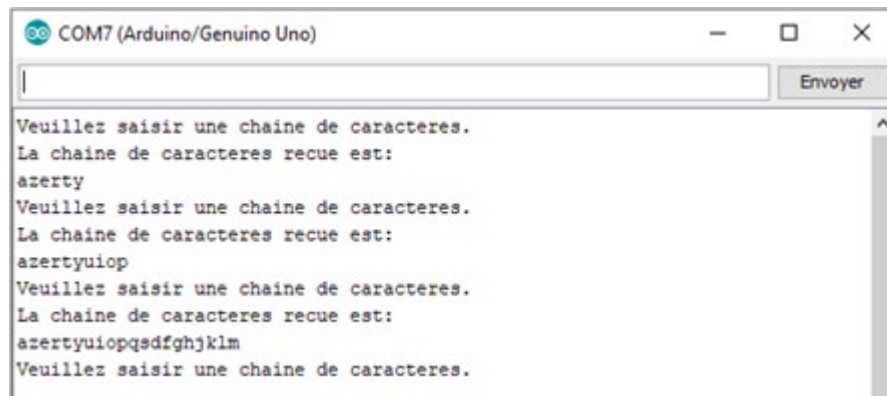
void loop() {
  int Val=0;
  char tampon[20]="";
  Serial.println("Veuillez saisir une chaîne de caractères.");
  while(Val==0)
  {
    delay(200);
    Val=Serial.available();
  }
  for (int i=0; i < Val; i++)
  {
    tampon[i]=Serial.read();
    delay(15);
  }
  Serial.println("La chaîne de caractères reçue est:");
  Serial.println(tampon);
  Serial.flush();
}
```

Dans ce programme, il est demandé de saisir une chaîne de caractères. Tant que le nombre de caractères disponibles sur le port série est nul (**Val==0**), la mémoire tampon du port série est vérifiée (**Val=Serial.available()**), toutes les 200 ms, jusqu'à ce que tous les caractères de la chaîne soient comptabilisés.

Ceux-ci sont alors lus un par un à travers une boucle "**for**" (**tampon[i]=Serial.read()**) et sont stockés dans un tableau de 20 caractères au maximum appelé "**tampon**" qui est ensuite affiché dans le moniteur série. Puis la

mémoire du port série est vidée et il est de nouveau demander de saisir une chaîne de caractères.

Voici le résultat dans le moniteur série :



Remarque :

Il est souvent nécessaire de convertir des chaînes de caractères en une variable représentant un nombre entier ou décimal pour effectuer des calculs.

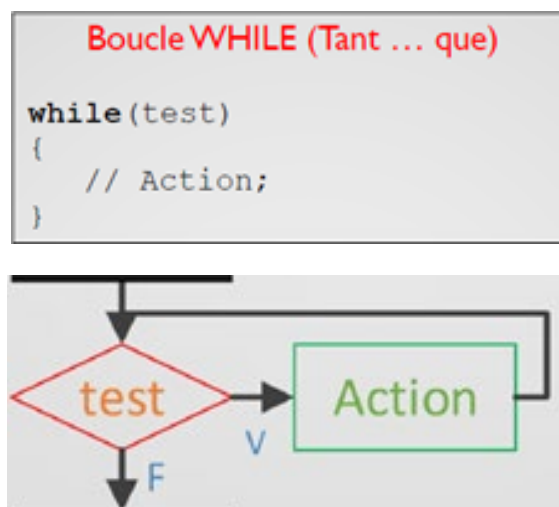
Il existe, pour cela, trois fonctions “**atoi()**“, “**atol()**“ et “**atof()**“, permettant respectivement de transformer une chaîne de caractères en nombre entier court, en nombre entier long et en nombre décimal.

Si notre chaîne de caractères “**tampon**“ représente un nombre, pour la convertir par exemple en nombre entier, on utilisera l’instruction :

**Int nombre = atoi(tampon)**

Ces fonctions retournent une valeur nulle en cas d’erreur (comme par exemple, si on essaye de convertir des lettres en nombre).

. Rappel sur la programmation d’une boucle **while**:



## . Le programme

Voici le programme de l'activité :

```
Emission_sonore
// Déclaration des constantes et variables

const int PinPOT = 0;
const int PinTone = 8;
const int PinButton = 12;

int State = 0;
int ValButton = 0;
int OldValButton = 0;
int ValPot = 0;
int Coeff=0;

// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode (PinButton, INPUT);
  while(Coeff<1 || Coeff>4)
  {
    int Val=0;
    char tampon[10]="";
    Serial.println("Veuillez choisir la gamme de frequence (valeur entre 1 et 4):");
    Serial.println("1 : 0 - 1023 Hz :");
    Serial.println("2 : 0 - 2046 Hz :");
    Serial.println("3 : 0 - 3069 Hz :");
    Serial.println("4 : 0 - 4092 Hz :");
    Serial.println(" ");
    while(!Val)
    {
      delay(200);
      Val=Serial.available();
    }
    for (int i=0; i < Val; i++)
    {
      tampon[i]=Serial.read();
      delay(15);
    }
    Coeff = atoi(tampon);
  }
  Serial.print("Gamme de frequence choisie : "); Serial.println(Coeff);
}
```

```
// Fonction principale en boucle

void loop() {
    ValButton = digitalRead(PinButton);
    delay(10);
    if (ValButton == HIGH and OldValButton == LOW) {
        State = 1 - State;
    }
    OldValButton = ValButton;
    if (State == 1) {
        ValPot = analogRead(PinPOT);
        tone(PinTone, ValPot*Coeff);
    }
    else {
        noTone(PinTone);
    }
}
```

---

### Déroulement du programme :

– Déclaration des constantes et variables :

- . **const int PinButton = 12** (constante nombre entier correspondant au n° de la broche sur laquelle le bouton poussoir est connecté)
- . **const int PinTone = 8** (constante nombre entier correspondant au n° de la broche sur laquelle le buzzer est connecté)
- . **const int PinPot = 0** (constante nombre entier correspondant au n° de la broche sur laquelle le potentiomètre est connecté)
- . **int State = 0** (variable nombre entier correspondant à l'action à effectuer)
- . **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 ValPot = 0** (variable nombre entier pour stocker la valeur de la broche du potentiomètre)
- . **int Coeff = 0** (variable nombre entier pour stocker la valeur du coefficient multiplicateur)

– Initialisation des entrées et sorties :

- . La liaison série est initialisée à 9600 bauds,
- . La broche du bouton poussoir est initialisée comme une entrée digitale,
- . Saisie du coefficient multiplicateur (nombre entier entre 1 et 4) dans le moniteur série,
- . Affichage dans le moniteur série du coefficient choisi.

– Fonction principale en boucle :

