

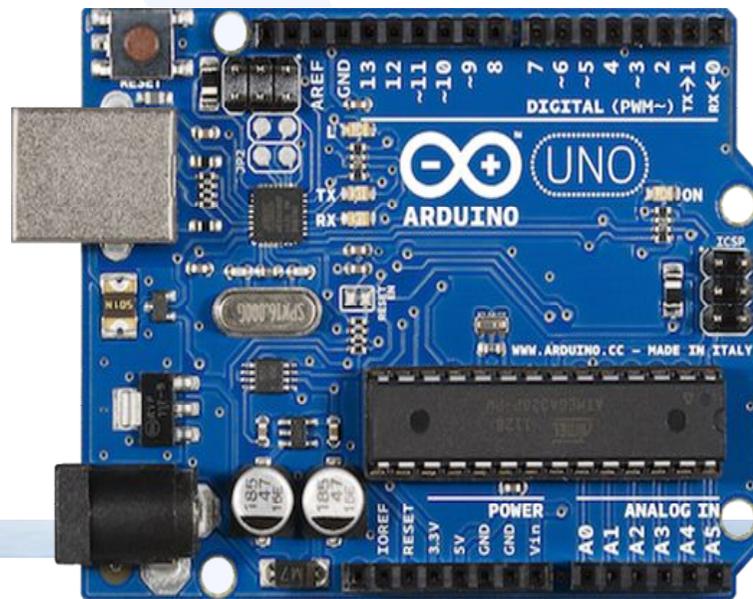


ATELIER MICROCONTROLEUR

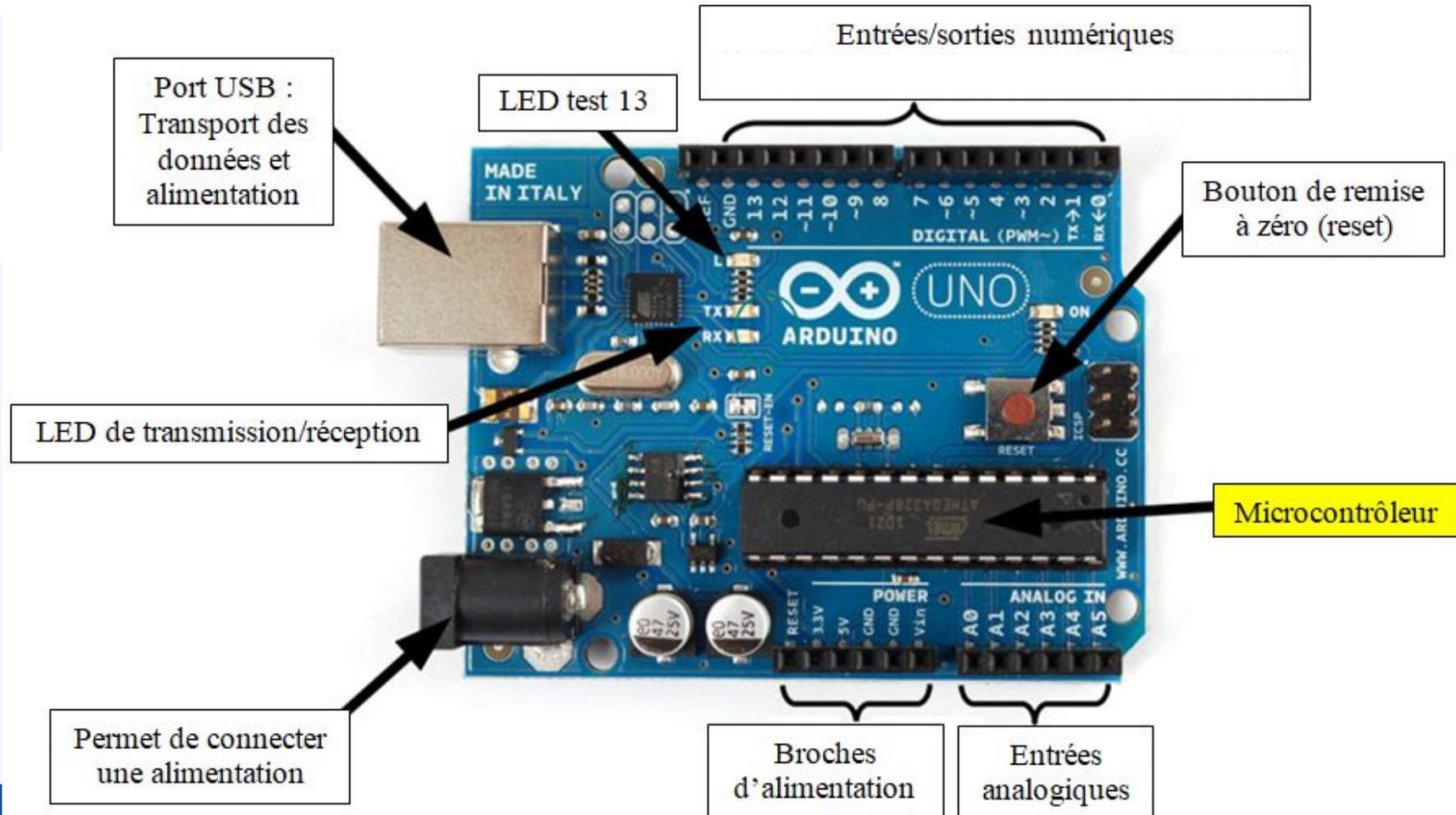
- Présentation du matériel (carte, plaque), du logiciel Arduino Software IDE et du langage (syntaxe)
- Activités de prise en main et de mise en œuvre d'un microcontrôleur

Formations Nouveaux programmes de lycée
Physique-Chimie (J2)
Mai - Juin 2019

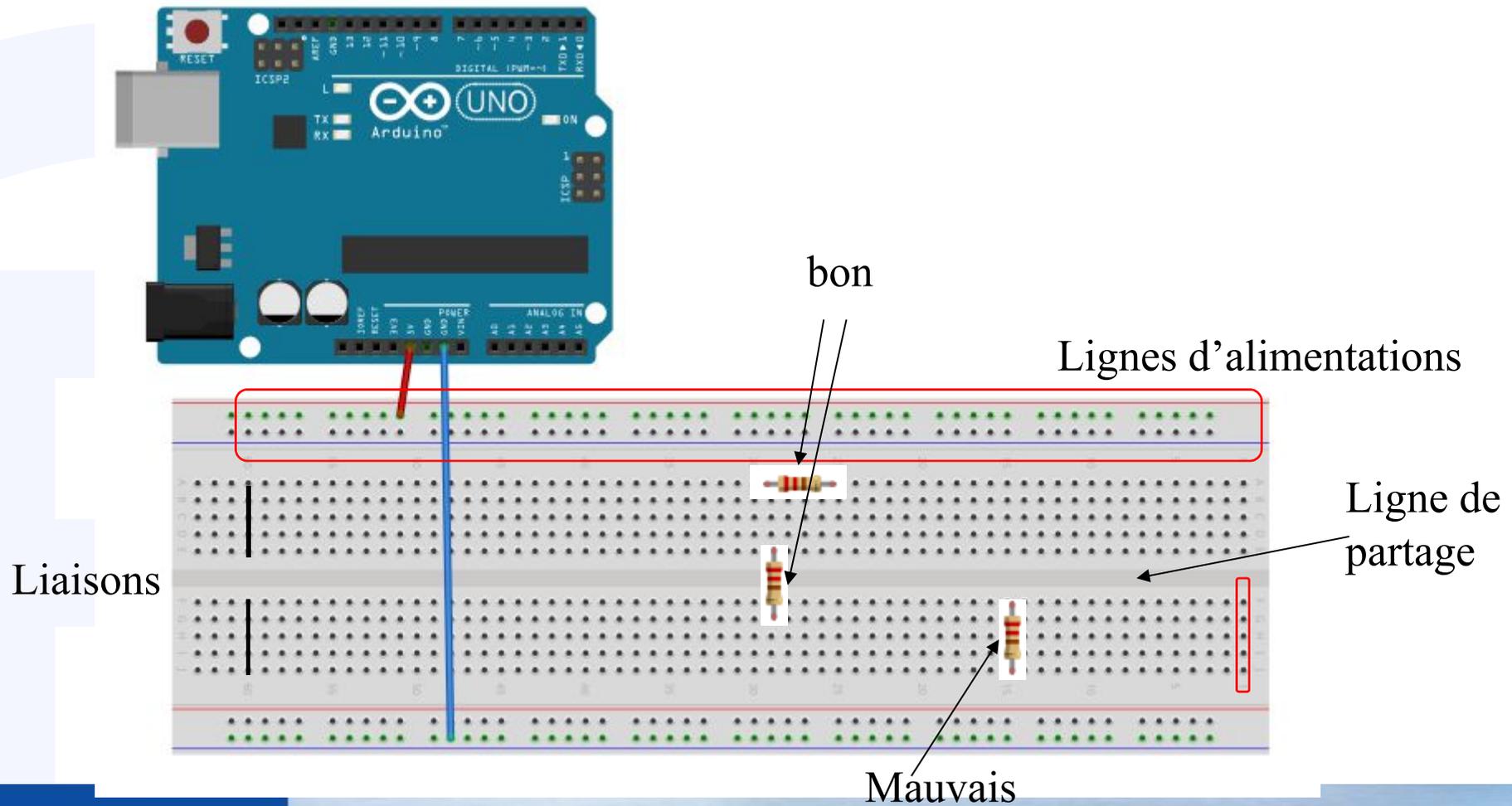
Présentation du microcontrôleur Arduino Uno



La carte Arduino Uno



Plaque d'essais sans soudure



L'interface de programmation Arduino

The image shows the Arduino IDE interface with several callouts explaining its components and code functions:

- Dans Outils, sélectionner le type de carte et le port**: Points to the 'Outils' menu in the top toolbar.
- Vérifier : permet de vérifier le programme**: Points to the 'Verify' button (checkmark icon) in the top toolbar.
- Téléverser : envoi du programme sur l'Arduino**: Points to the 'Upload' button (upload icon) in the top toolbar.
- Bouton d'affichage du moniteur série**: Points to the 'Serial Monitor' button (terminal icon) in the top toolbar.
- Fonction setup() : fonction d'initialisation ne s'exécute qu'une seule fois**: Points to the `void setup() {` line in the code editor.
- Fonction loop() : le contenu du programme s'exécute à l'infini**: Points to the `void loop() {` line in the code editor.

```
sketch_oct23a | Arduino 1.8.7
Fichier Édition Croquis Outils Aide
sketch_oct23a $
void setup() {
}
void loop() {
}
```

5 Arduino/Genuino Uno sur COM8

Les principales commandes du langage Arduino (C/C++)

Conditionnelles :

- ```
if(test vrai){
 action 1;
}
else
{
 action 2;
}
```

## Boucles

- ```
for (pour...)
```
- ```
while (tant que...)
```

## Comparaisons

- `==` (équivalent à)
- `!=` (différent de)
- `<`; `>`; `<=`; `>=`

## Variables :

- `char` (caractère)
- `int` (entier)
- `float` (décimal)
- `boolean` (true/false)

## Etat/Niveaux logiques des sorties numériques

- `LOW` (état bas, 0V)
- `HIGH` (état haut, 5V)

## Entrée/Sorties numériques :

- `INPUT` (entrée)
- `OUTPUT` (sortie)
- `pinMode(N°broche,INPUT/OUTPUT)` : fixe la broche en entrée ou en sortie
- `digitalWrite(broche,état)` : écrit un état sur une broche numérique
- `analogRead(broche)` : retourne un entier entre 0 et 1023 correspondant à 0 et 5V
- `analogWrite(broche,valeur)` : Mode PWL permet de simuler une sortie analogique sur une broche numérique ~ valeur est un nombre compris entre 0 et 255

## Affichage :

- `Serial.println(variable)` : affiche la valeur de la variable dans le moniteur série

## Gestion du temps :

- `delay(ms)` : attends la valeur indiquée en ms
- `delayMicrosecond(us)`

# Activités de prise en main et de mise en œuvre d'un microcontrôleur

## **Activité 1 : Allumer une LED**

Pour aller plus loin : la faire clignoter

Pour aller encore plus loin : moduler l'intensité lumineuse

## **Activité 2 : Produire un son**

Pour aller plus loin : jouer une mélodie (Au clair de la lune)

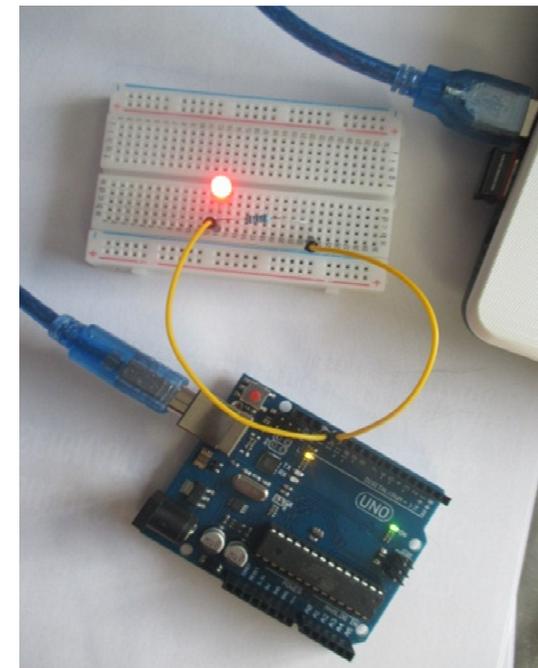
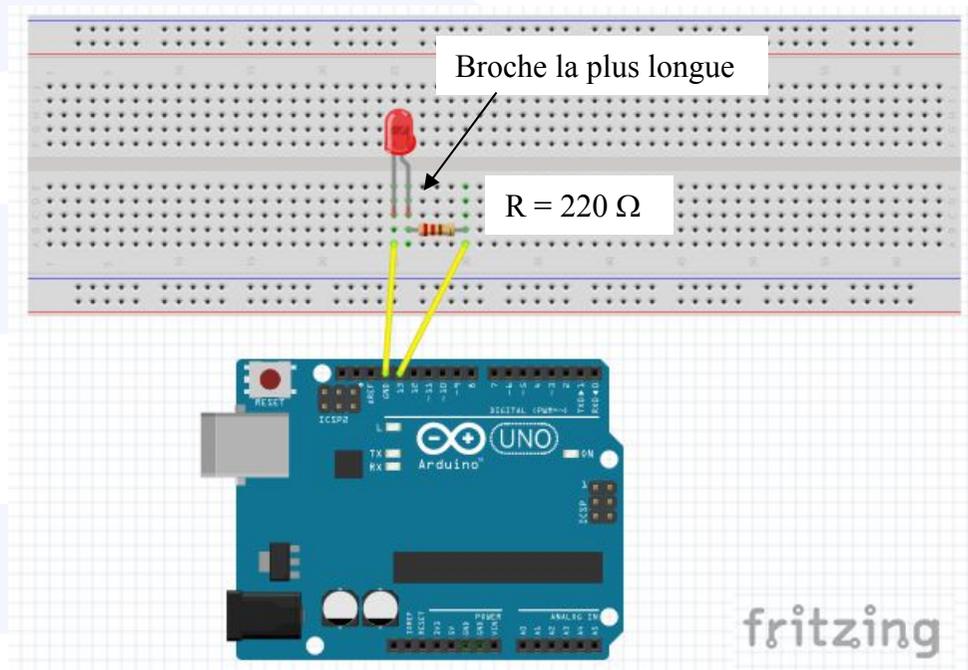
## **Activité 3 : Mesure d'une distance**

Pour aller plus loin : paramétrer la vitesse du son par une mesure de la température

## **Activité 4 : Eclairage automatique (J1)**

# 1- Allumer une diode

Brancher une LED et un conducteur ohmique ( $R = 220 \Omega$ ) entre la sortie numérique 13 et la masse :

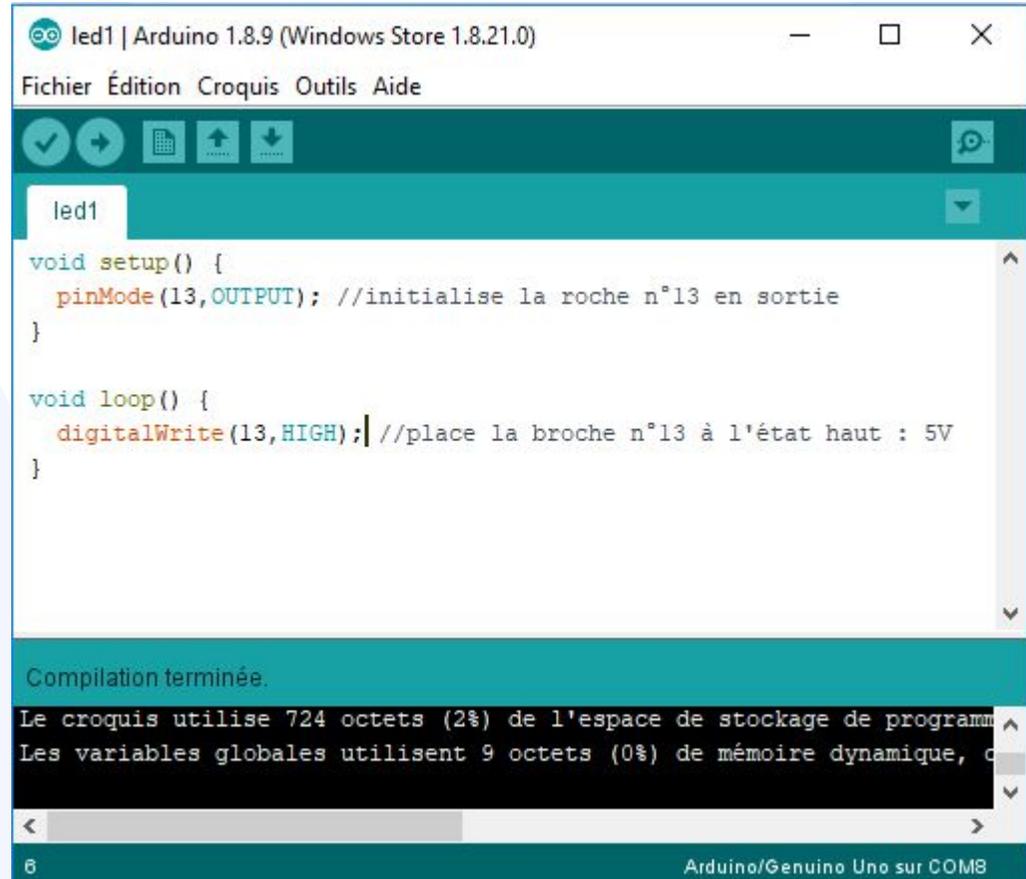


# 1- Allumer une diode

Dans le logiciel Arduino IDE, la programmation se fait de la façon suivante :

Enregistrer le programme et cliquer sur  pour le vérifier et sur  pour le téléverser dans la carte

*NB : en cas de message d'erreur, s'assurer que la carte est reconnue sur le bon port à l'aide du menu Outils / Ports*



The screenshot shows the Arduino IDE window titled "led1 | Arduino 1.8.9 (Windows Store 1.8.21.0)". The menu bar includes "Fichier", "Édition", "Croquis", "Outils", and "Aide". The toolbar contains icons for checkmark, upload, and download. The code editor shows the following code:

```
void setup() {
 pinMode(13,OUTPUT); //initialise la roche n°13 en sortie
}

void loop() {
 digitalWrite(13,HIGH); //place la broche n°13 à l'état haut : 5V
}
```

Below the code editor, the status bar indicates "Compilation terminée." and provides memory usage information: "Le croquis utilise 724 octets (2%) de l'espace de stockage de programme" and "Les variables globales utilisent 9 octets (0%) de mémoire dynamique, c". The bottom status bar shows "6" and "Arduino/Genuino Uno sur COM8".

# 1- Allumer une diode Pour aller plus loin

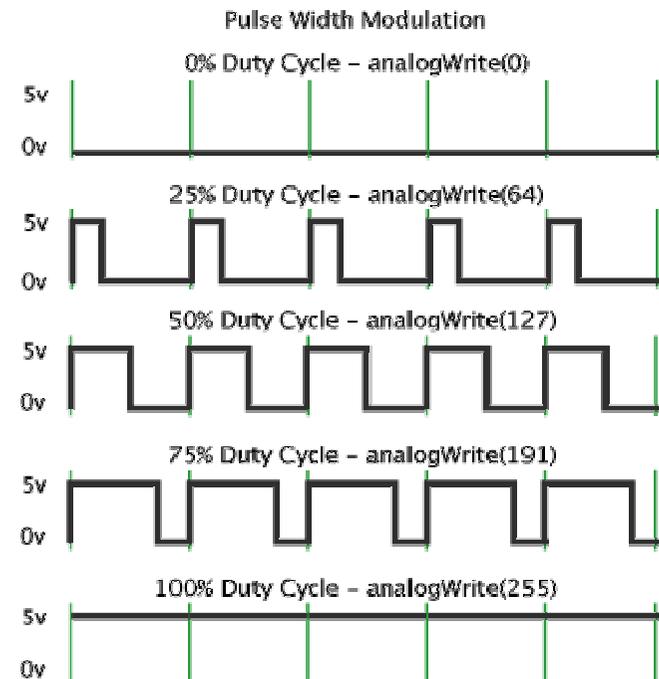
## Faire clignoter la diode :

Modifier votre programme pour faire clignoter la led (1s allumée et 0,5s éteinte)

L'instruction **delay(n)** réalise une pose de n millisecondes dans l'exécution du programme

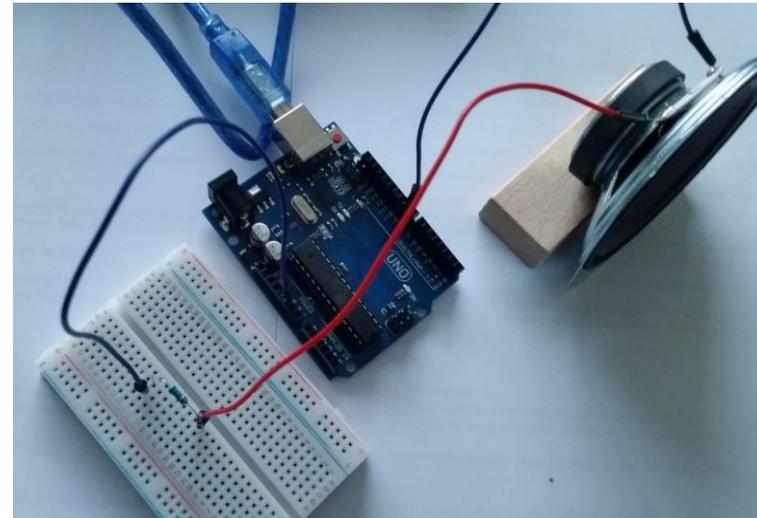
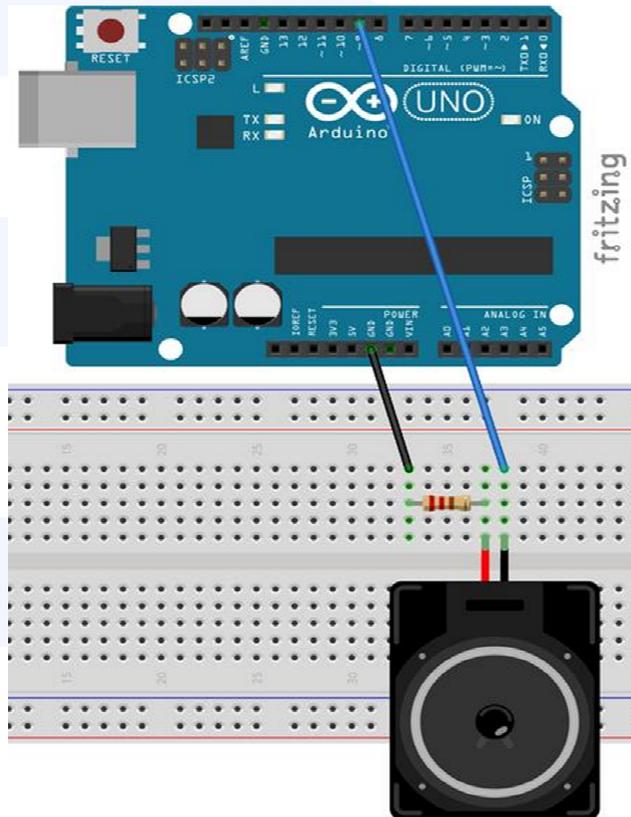
## Faire varier l'intensité lumineuse :

La carte ne possède pas de sortie analogique mais les sorties logiques n°3, 5, 6, 9, 10, et 11 possèdent un mode PWM(~) *Pulse Width Modulation* permettant de simuler une sortie analogique à l'aide de l'instruction **analogWrite(broche,n)** avec n compris entre 0 et 255



## 2- Produire un son

Réaliser le montage suivant : le HP, protégé par une résistance  $R = 220 \Omega$ , est alimenté par la sortie numérique 9 :



## 2- Produire un son

Compléter le code pour obtenir la note La<sub>3</sub> (440 Hz)

```
ledclignotante | Arduino 1.8.9 (Windows Store 1.8.21.0)
Fichier Édition Croquis Outils Aide

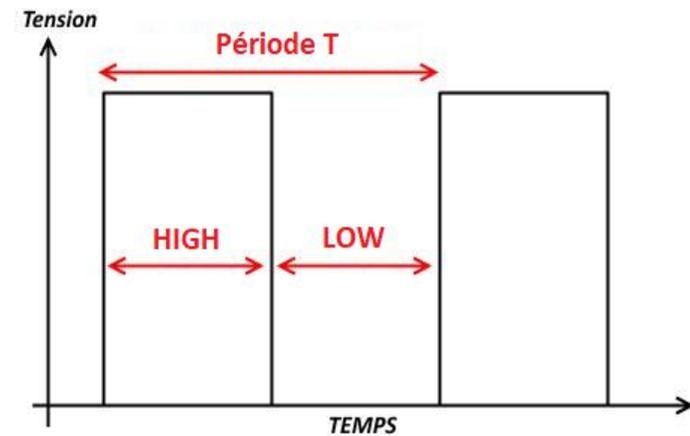
ledclignotante $

int hp=9;
void setup() {
 pinMode(hp,OUTPUT);
}

void loop() {
 digitalWrite(hp,HIGH);
 delayMicroseconds(.....);
 digitalWrite(hp,LOW);
 delayMicroseconds(.....);
}

Téléversement terminé
Le croquis utilise 766 octets (2%) de l'espace de stockage de programme
Les variables globales utilisent 9 octets (0%) de mémoire dynamique, d

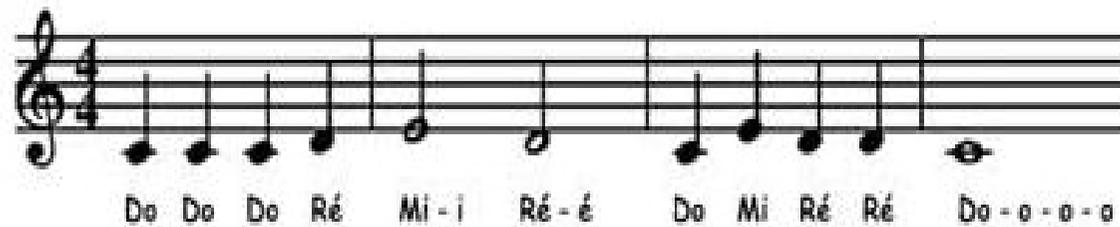
6 Arduino/Genuino Uno sur COM8
```



## 2- Produire un son Pour aller plus loin

Alternative : la fonction **tone(broche,frequence,duree en ms)**

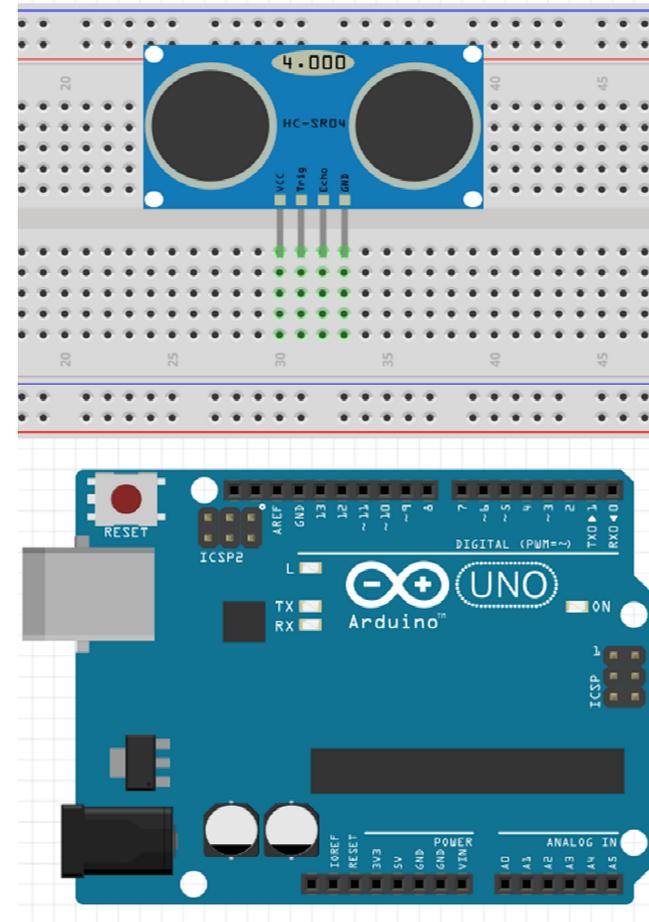
Réaliser le programme permettant de jouer la mélodie suivante :



## 3- Mesure de distance

Nous allons utiliser le module ultrasons HC-SR04 qui possède 4 bornes :

- Vcc : alimentation 5V
- Gnd : Masse 0V
- Trig : Emission signal
- Echo : Réception signal



## 3- Mesure de distance

Réaliser le câblage du montage correspondant au programme suivant :

```
MesureDistance §

int trig = 8; int echo = 9;
long temps;
long distance;
long vitesse;

void setup() {
 pinMode(trig, OUTPUT);
 pinMode(echo, INPUT);
 Serial.begin(9600);
 vitesse=340;
}
|
void loop() {
 digitalWrite(trig, HIGH); delayMicroseconds(10);
 digitalWrite(trig, LOW);
 temps = pulseIn(echo, HIGH);
 distance = temps*vitesse/(2*10000);
 Serial.print("Distance en cm : ");
 Serial.println(distance);
 delay(1000);
}
```

## 3- Mesure de distance Pour aller plus loin

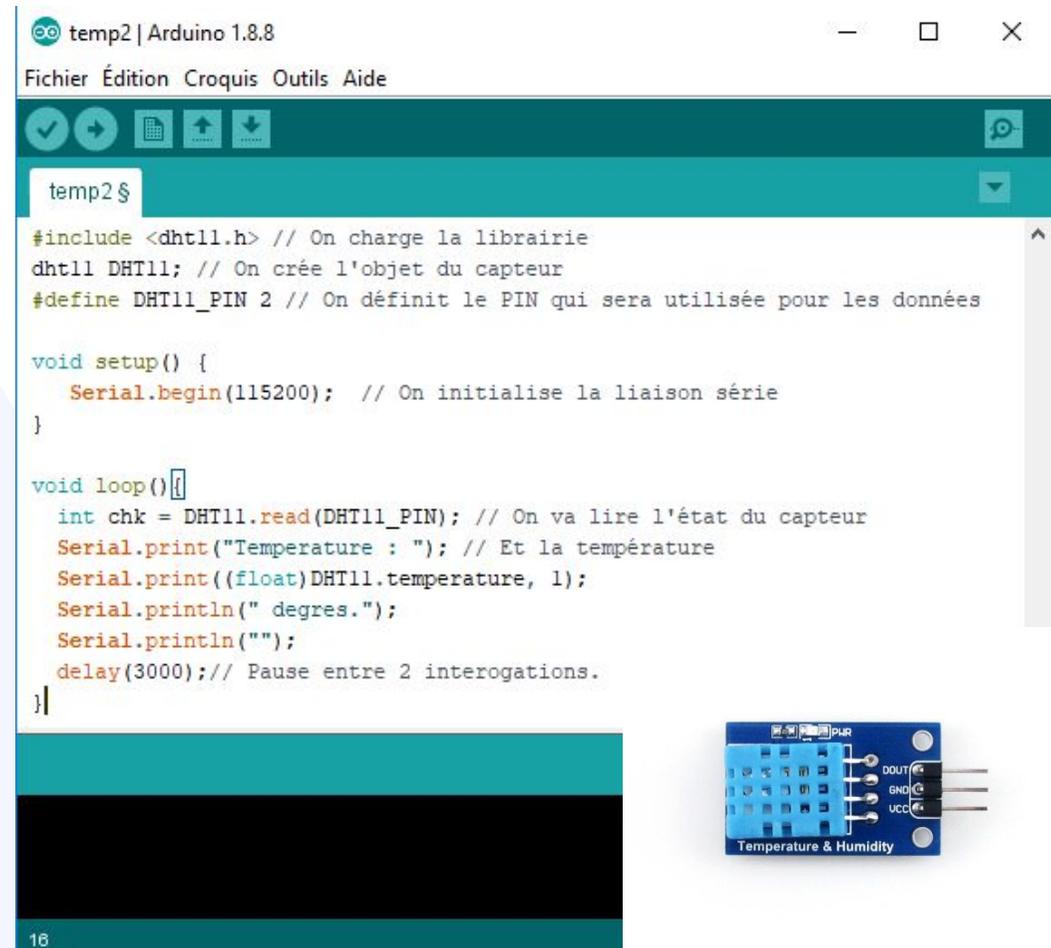
Dans le programme précédent la vitesse du son est rentrée « en dur » dans le script.

La vitesse du son peut être calculée à l'aide de la formule  $v = 20,05 \sqrt{T}$

Le programme ci-contre exploite un capteur de température dht11 en affichant la température (en ° C) dans le moniteur série.

*NB : la bibliothèque dht11 (fournie dans le cours Moodle) est à installer dans le répertoire Arduino/libraries*

Modifier le programme précédent pour que la mesure de la distance soit paramétrée par la mesure de la température modifiant la valeur de v.



```
temp2 | Arduino 1.8.8
Fichier Édition Croquis Outils Aide

temp2 $
#include <dht11.h> // On charge la librairie
dht11 DHT11; // On crée l'objet du capteur
#define DHT11_PIN 2 // On définit le PIN qui sera utilisée pour les données

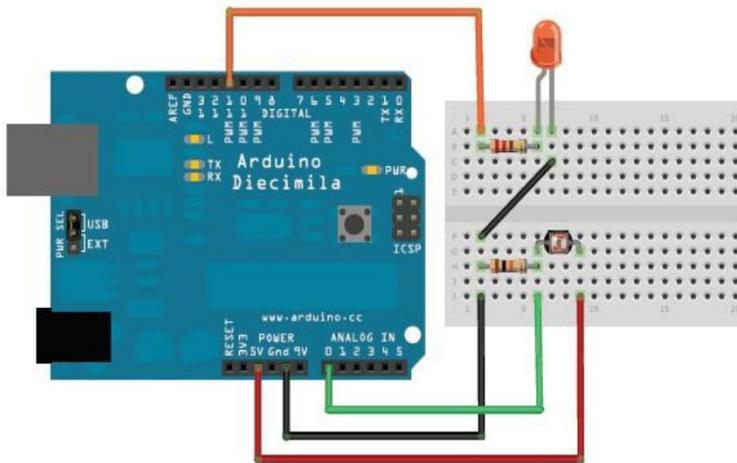
void setup() {
 Serial.begin(115200); // On initialise la liaison série
}

void loop() {
 int chk = DHT11.read(DHT11_PIN); // On va lire l'état du capteur
 Serial.print("Temperature : "); // Et la température
 Serial.print((float)DHT11.temperature, 1);
 Serial.println(" degres.");
 Serial.println("");
 delay(3000); // Pause entre 2 interogations.
}
```



## 4- Eclairage automatique (J1)

Câbler le montage de l'éclairage automatique étudié en J1 et vérifier son fonctionnement



```
4-InterrupteurCrepusculaire

int Valeur_A0;
float Tension_A0;
boolean Etat_lampe;

void setup() {
 pinMode(11,OUTPUT);
 // Serial.begin(9600);
}

void loop() {
 Valeur_A0=analogRead(A0);
 Tension_A0=(float)Valeur_A0*5/1024;
 if ((Tension_A0>4.0)&&(Etat_lampe==true)) {
 digitalWrite(11,LOW);
 Etat_lampe=false;
 }
 if ((Tension_A0<3.3)&&(Etat_lampe==false)) {
 digitalWrite(11,HIGH);
 Etat_lampe=true;
 }
 delay(250);
}
```