Date : …/…./……	**Réalisation d'un radar de recul**	Classe :
Thème :**système communicant**		Nom :
		Prénom :

Mise en situation :

Dans de nombreux véhicules récents, on trouve des radars de recul situés dans les pare-chocs. Ils permettent d'aider les conducteurs lors d'une manœuvre et éviter ainsi de toucher un obstacle. Dans cette activité, il est question de mettre en œuvre un dispositif équivalent à l'aide d'une carte Arduino UNO et des capteurs utilisant une technologie à ultrasons.

Objectifs :

- Le but est de prendre en main la carte Arduino au travers d'exemples d'application, par la suite réaliser un radar de recul.

Matériels :

- -un ordinateur - une carte Arduino – kit dev pour Arduino - un capteur a ultrason SF05

Savoirs associés	**Compétences visées**	**Résultats attendus**	**Evaluation**
S2 – 2 Traitement de l'information	**C1-1** Appréhender la mise en œuvre d'un projet simulé ou réel d'installation d'un système	-Le rôle de tout ou partie des éléments répertoriés est énoncé - Les traces d'échange entre équipements sont exploitées	
S2 - 2 Les fonctions logiques de base en programmation	**C3-2** Réaliser l'intégration matérielle ou logicielle d'un équipement.		
S4 - 2 : Langage de programmation	C4-4 Installer, configurer les éléments du système et vérifier la conformité du fonctionnement.		

Trop d'erreurs | Réussite partielle | Non-évaluable

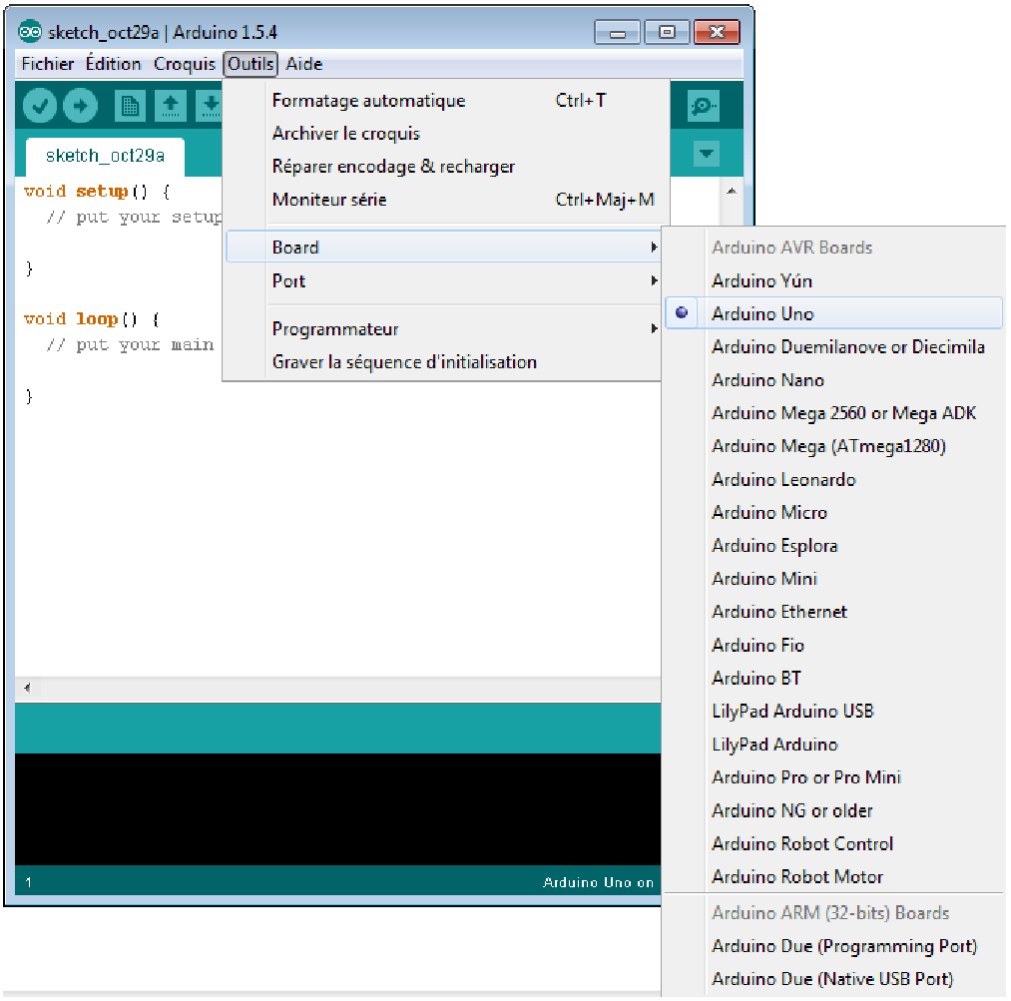
En cours d'acquisition | Réussite totale | Absent

Exercice 1: On souhaite faire clignoter une LED connectée à la broche numérique n°7 de l'Arduino.

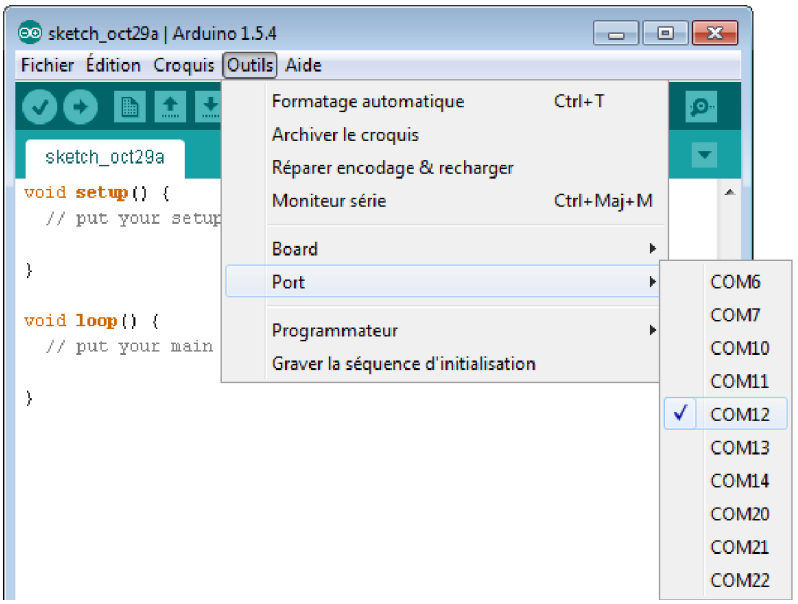
1. Réaliser le montage suivant :
2. Connecter l'Arduino UNO au PC à l'aide du câble USB fourni.

3. Lancer le l'EDI Arduino.

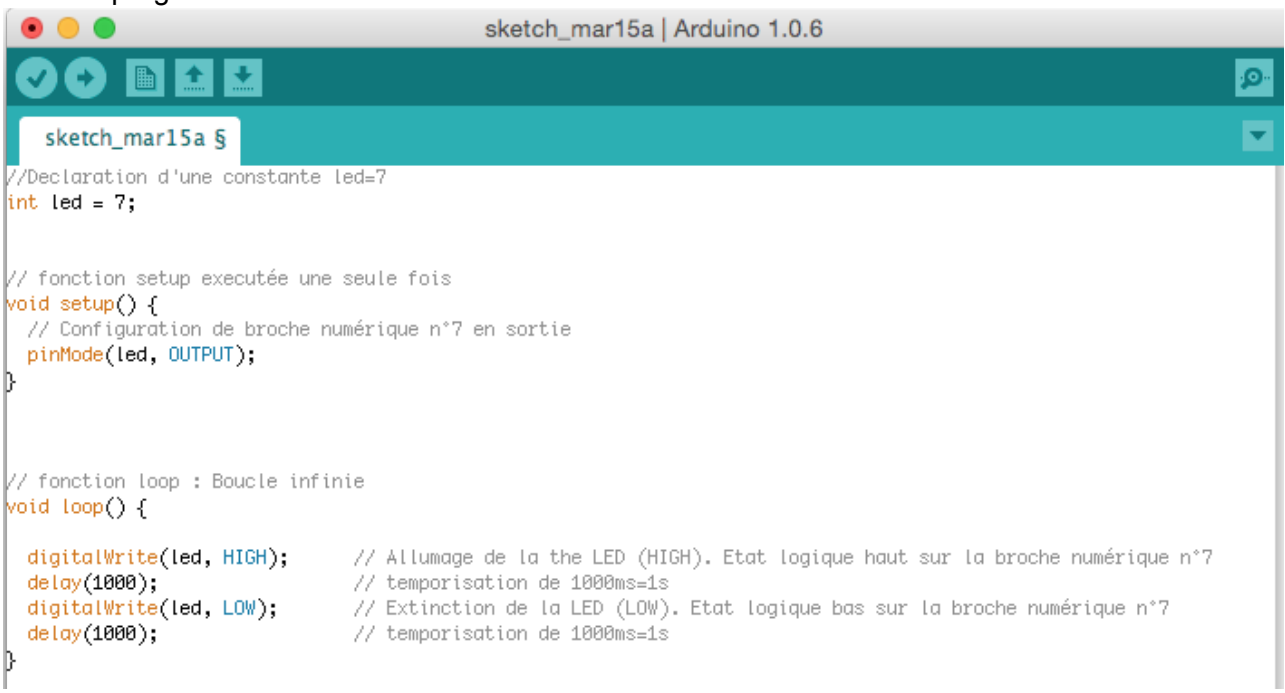
4. Sélectionner la carte Arduino Uno dans le logiciel



5. Sélectionner le port de communication utilisé par votre PC pour dialoguer avec l'Arduino. (Utiliser le gestionnaire de périphériques pour connaître le port COM utilisé par l'Arduino.)



6. Saisir le programme suivant :

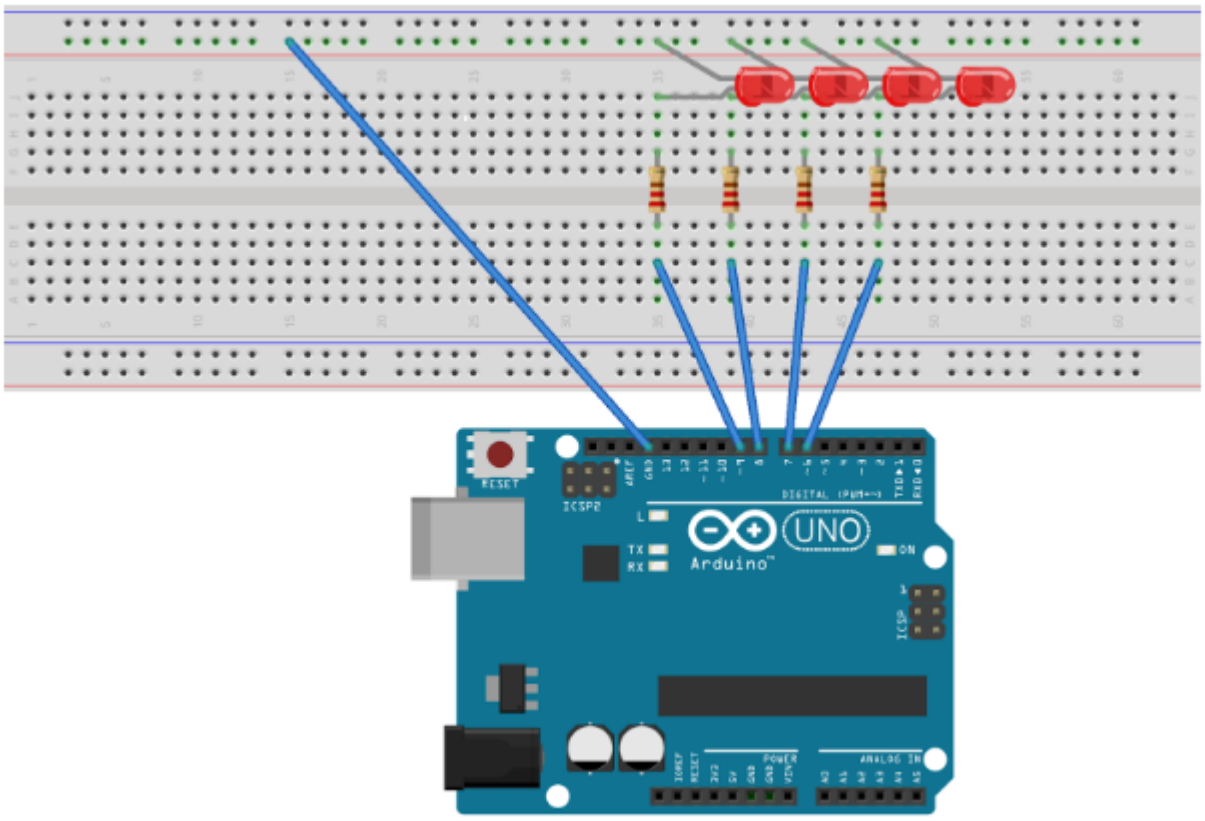



```
//Declaration d'une constante led=7
int led = 7;


// fonction setup executée une seule fois
void setup() {
  // Configuration de broche numérique n°7 en sortie
  pinMode(led, OUTPUT);
}



// fonction loop : Boucle infinie
void loop() {

  digitalWrite(led, HIGH);      // Allumage de la the LED (HIGH). Etat logique haut sur la broche numérique n°7
  delay(1000);                  // temporisation de 1000ms=1s
  digitalWrite(led, LOW);       // Extinction de la LED (LOW). Etat logique bas sur la broche numérique n°7
  delay(1000);                  // temporisation de 1000ms=1s
}
```

7. Compiler le programme.
8. Téléverser le programme dans la carte Arduino.

9. Modifier le programme pour que la LED soit allumée 3 secondes puis éteinte 6 secondes en boucle, enregistrer le nouveau programme

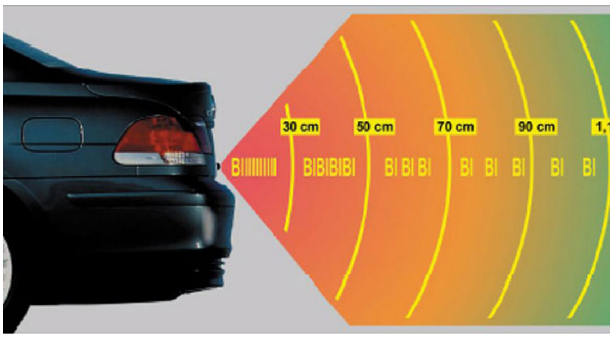
10. Réaliser le montage suivant. Modifier le programme précédent afin de réaliser un chenillard. Chaque LED s'allume 1 seconde et s'éteint les unes après les autres. Enregistrer votre programme.



Faire vérifier par le professeur

Partie II Synthèse : réalisation d'un radar de recul

Introduction :



Les radars de recul pour les véhicules automobiles sont des systèmes d'assistance qui avertissent de la présence d'un obstacle à l'arrière du véhicule, lors d'une opération de marche arrière. Les principaux constituants de ce type de système sont :

- Les capteurs à ultrasons, composés d'émetteurs et de récepteurs
- Le calculateur, qui estime la distance entre l'obstacle et l'arrière du véhicule, à partir des informations délivrées par les capteurs à ultrasons
- un système d'émission sonore qui avertit le conducteur. Habituellement, un bip sonore est émis lorsque la distance obstacle/véhicule devient inférieure à une distance limite, puis, plus l'obstacle se rapproche, plus la fréquence des bips est élevée, jusqu'à devenir une émission sonore continue lorsque le véhicule est très proche de l'obstacle.

Objectif :

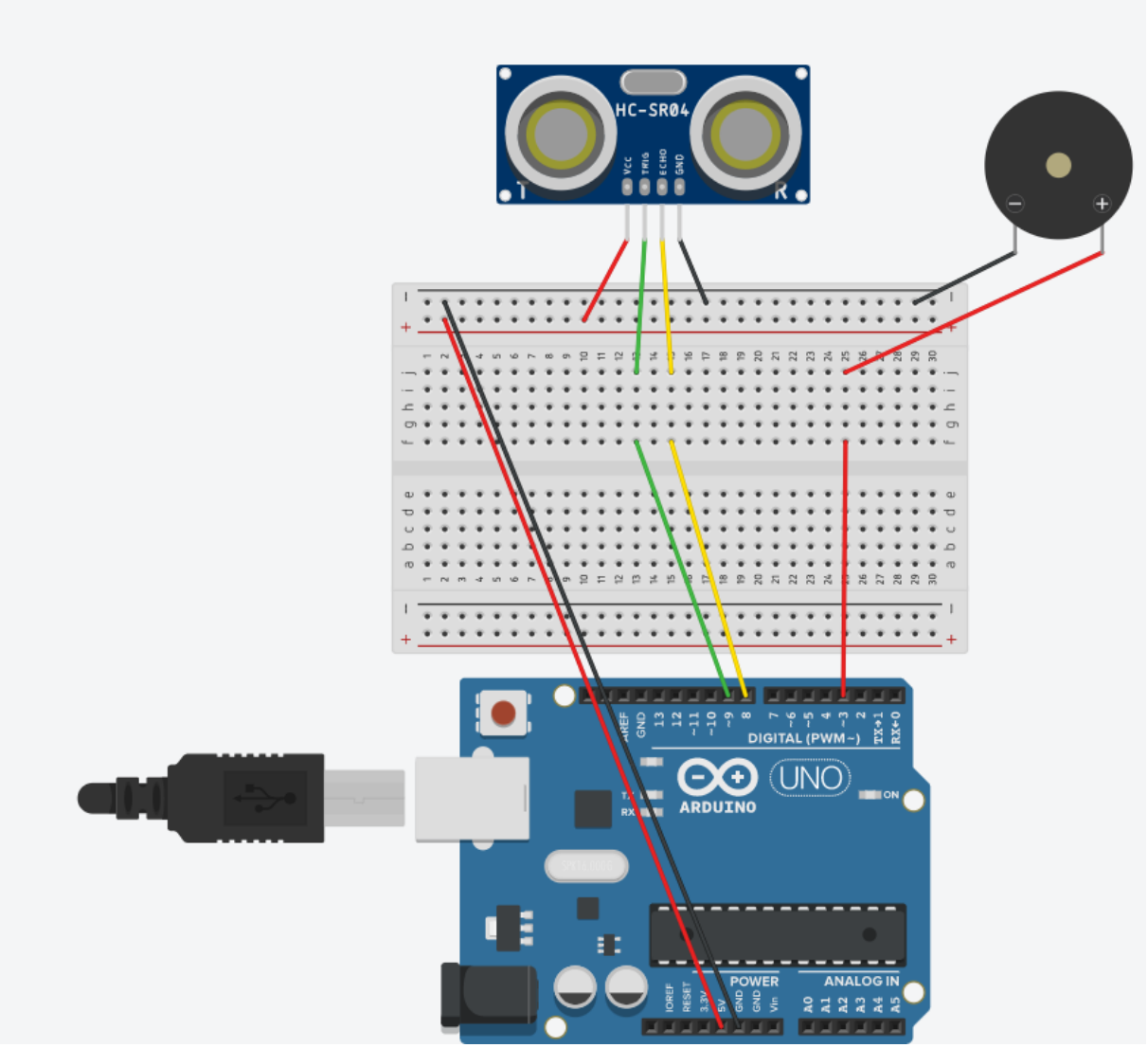
L'objectif est de reconstituer un radar de recul automobile. Le matériel mis à disposition est le suivant :

- Une carte Arduino-Uno et son câble USB
- Un capteur à ultrason de référence SRF04
- Un buzzer, (afficheur LCD)
- Une plaque d'essai (ou breadboard)
- Des câbles

Le système conçu devra émettre un son, fonction de la distance entre un obstacle et le capteur à ultrasons. Ce son sera constitué d'une succession de bips d'une durée de **30 ms**, puis d'un silence d'une durée **T**. Cette durée T diminue avec la distance. On note **d** la distance entre le capteur et l'objet et on impose :

- **T=0.8 s si 40 cm < d ≤ 50 cm**
- **T=0.6 s si 30 cm < d ≤ 40 cm**
- **T=0.4 s si 20 cm < d ≤ 30 cm**
- **T=0.2 s si 10 cm < d ≤ 20 cm**
- **T=0 ms si d ≤ 10 cm**

Montage :



Démarche :

- La description du capteur à ultrason est donnée en annexe : la première chose à faire est de bien comprendre son fonctionnement.
- Réfléchir à un programme qui permette d'afficher, sur le moniteur série, la distance en cm entre l'objet et le capteur (on testera avec une surface "suffisamment" plane (une feuille). Pour mesurer la durée de l'impulsion fournie sur la broche Echo du capteur, vous utiliserez la fonction PulseIn (voir la description de cette fonction en annexe).
- Faire un autre programme, distinct du précédent, pour faire fonctionner le buzzer fourni. Vous utiliserez les fonctions tone et noTone (voir la description de ces fonctions en annexe).
- Enfin, réunir les deux codes précédents pour la réalisation finale.

ANNEXE 1 : Description de la fonction tone

Génère une onde carrée (onde symétrique avec "duty cycle" (niveau haut/période) à 50%) à la fréquence spécifiée (en Hertz (Hz), ou nombre de périodes par seconde) sur une broche. La durée peut être précisée, sinon l'impulsion continue jusqu'à l'appel de l'instruction noTone(). La broche peut être connectée à un buzzer piézoélectrique ou autre haut-parleur pour jouer des notes (les sons audibles s'étendent de 20Hz à 20 000Hz). Une seule note peut être produite à la fois. Si une note est déjà jouée sur une autre broche, l'appel de la fonction tone() n'aura aucun effet (tant qu'une instruction noTone() n'aura pas eu lieu). Si la note est jouée sur la même broche, l'appel de la fonction tone() modifiera la fréquence jouée sur cette broche.

Syntaxe :

```
tone(broche, frequence)
tone(broche, frequence, durée)
```

Paramètres :

- **broche**: la broche sur laquelle la note est générée.
- **frequence**: la fréquence de la note produite, en hertz (Hz)
- **durée**: la durée de la note en millisecondes (optionnel)

Valeur renvoyée

Aucune

ANNEXE 2 : Description de la fonction noTone

Stoppe la génération d'impulsion produite par l'instruction tone(). N'a aucun effet si aucune impulsion n'est générée.

Syntaxe :

```
noTone(broche)
```

Paramètres :

- **broche**: la broche sur laquelle il faut stopper la note.

Valeur renvoyée

aucune

ANNEXE 3 : Description de la fonction PulseIn

Lit la durée d'une impulsion (soit niveau HAUT, soit niveau BAS) appliquée sur une broche (configurée en ENTREE). Par exemple, si le paramètre valeur est HAUT, l'instruction pulseIn() attend que la broche passe à HAUT, commence alors le chronométrage, attend que la broche repasse au niveau BAS et stoppe alors le chronométrage. L'instruction renvoie la durée de l'impulsion en microsecondes (1 millions de microsecondes par secondes). L'instruction s'arrête et renvoie 0 si aucune impulsion n'est survenue dans un temps spécifié.

Le chronométrage de cette instruction a été déterminé de façon empirique et peut probablement donner des erreurs sur les impulsions de longue durée. Travailler avec des impulsions d'une durée de 10 microsecondes à 3 minutes.

Syntaxe :

```
pulseIn(broche, valeur)
pulseIn(broche, valeur, delai_sortie)
```

Paramètres :

- **broche**: le numéro de la broche sur laquelle vous voulez lire la durée de l'impulsion. (type int)
- **valeur**: le type d'impulsion à "lire" : soit HIGH (niveau HAUT) ou LOW (niveau BAS). (type int)
- **delai_sortie** (optionnel): le nombre de microsecondes à attendre pour début de l'impulsion. La valeur par défaut est 1 seconde. (type unsigned long)

Valeur renvoyée

la durée de l'impulsion (en microseconds) ou 0 si aucune impulsion n'a démarrer avant le délai de sortie. (type unsigned long)

Exemple :

```
int broche = 7; // variable de broche
unsigned long duree;  // variable utilisée pour stocker la durée

void setup()
{
  pinMode(broche, INPUT); // met la broche en entrée
}

void loop()
{
  duree = pulseIn(broche, HIGH); // met la durée de l'impulsion de niveau HAUT dans la variable duree
}
```

Validation par enseignante: