

# La carte Arduino

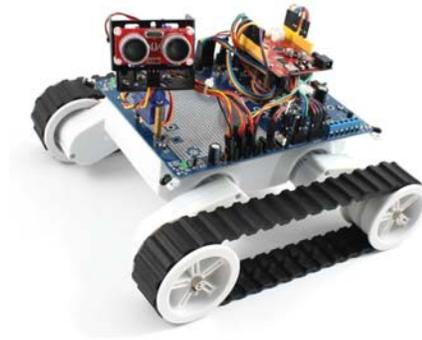
---

## ✳ Objectifs :

1. Passer d'un code python à Arduino
2. utiliser un détecteur ultra son
3. programmer des boucles et des structures oconditionnelles

## I. Présentation d'une carte Arduino

---



### 1 Ecriture d'un programme

```
1 int trig = 12;
2 int echo = 11;
3 int lecture;
4 float cm;
5 void setup()
6 {
7   pinMode(trig, OUTPUT);
8   digitalWrite(trig, LOW);
9   pinMode(echo, INPUT);
10  Serial.begin(9600);
11 }
12 void loop()
13 {
14   digitalWrite(trig, HIGH);
15   delayMicroseconds(10);
16   digitalWrite(trig, LOW);
17   lecture = pulseIn(echo, HIGH);
18   cm = lecture / 59;
19   Serial.print("Distance : ");
20   Serial.println(cm);
21   delay(1000);
22 }
```

## 2 Définition des données

Toutes les lignes de code Arduino doivent se terminer par ;. Les commentaires sont indiqués par / ou //. Pour utiliser des variables numériques, la syntaxe nécessite de préciser le type de variable utilisé :

- byte : entier sur 8 bits (0-255)
- int : entier sur 16 bits (-32768 à 32767)
- long : entier sur 32 bits ( $-2.10^6$  à  $2.10^6$ )
- float : flottant sur 32 bits
- char : chaîne de caractère

```
1 byte a=12
2 int b=143 ;
3 float x=12.5 ;
4 char chaine []="arduino"
```

En Python, la syntaxe  $a = 12$ ,  $b = 143$ , et  $x = 12.5$  permettent de définir automatiquement le type de ces grandeurs.

## 3 Entrées/sortie

### a) Définition

L'Arduino possède 14 entrées/sorties numériques que l'on peut activer par la fonction suivante

```
1 pinMode(pin, INPUT);
```

Cette fonction contient deux arguments, le numéro de la broche et le mode de fonctionnement : INPUT / OUTPUT

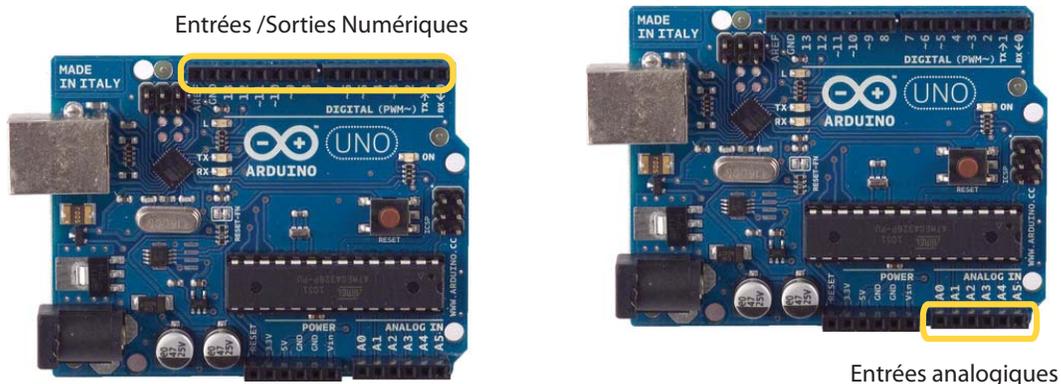


Figure 1 – Entrée analogiques et numériques

### b) Sortie numérique

La valeur numérique peut prendre deux valeurs ou état : HIGH (+5 V) ou LOW (0 V). Le changement d'état s'effectue grâce à la fonction suivante :

```
1 digitalWrite(pin, HIGH);
2 digitalWrite(pin, LOW);
```

### c) entrée numérique

De la même façon, on peut récupérer la valeur d'une broche par la fonction

```
1 X=digitalRead (pin)
```

La valeur de X vaut alors HIGH ou LOW.

### d) Valeur analogique

Sur une carte Arduino Uno, on retrouve 6 CAN. Ils se trouvent tous du même côté de la carte, là où est écrit "Analog IN" :

La fonction qui permet de lire la valeur lue sur une entrée analogique de l'Arduino. Elle prend un argument (le numéro de la broche d'entrée) et retourne la valeur lue :

```
1 analogRead (pin)
```

La valeur retournée (un int) sera le résultat de la conversion analogique->numérique. Il s'agit donc d'une grandeur comprise en 0 et 1023.

#### Exemple d'un capteur de température LM35

```
1 float temp;
2 int tempPin = 0;
3 char texte[]="la temperature est de " ;
4 void setup()
5 {
6   Serial.begin(9600);
7 }
8 void loop()
9 {
10  temp = analogRead(tempPin);
11  temp = temp * 0.488;
12  Serial.print(texte);
13  Serial.println(temp);
14  delay(2000);
15 }
```

## II. Les différentes structures

---

### 1 void setup() et loop()

La structure void setup() permet de définir les variables et les entrées sorties avant l'exécution du programme principal.

La structure void loop() est le corps même de la programmation. Les instructions situées dans ce programme seront effectuées en boucle un nombre infini de fois.

### 2 Les structures boucles for et while

Tout comme Python, les boucles FOR et WHILE sont disponibles. La syntaxe est légèrement plus lourde puisqu'il faut définir la variable dans la boucle :

```
1 for (int i=0 ; i<10 ; i++)
2 {
3   OPERATION ;
4 }
```

```
1 int i=0
2 while (i<10)
3 {
4 OPERATION ;
5 i++
6 }
```

Pour la boucle for, la structure peut se résumer par :

```
1 for (initialisation ; condition ; incrément)
2 { instructions }
```

L'indentation propre à Python est remplacé ici par des accolades. Les opérations systématiques de la boucle for ou while sont alors spécifiées entre parenthèses.

### 3 Les structures conditionnelles

```
1 if (CONDITION 1)
2 {
3 OPERATION 1 ;
4 }
5 else (CONDITION 2)
6 {
7 OPERATION 2 ;
8 }
9 else
10 {
11 OPERATION 3 ;
12 }
```

## III. Capteur ultra-son

---

### 1 Principe

Le capteur utilisé en TIPE est le HC-SR05 possédant 5 broches. Une d'alimentation Vcc, une de masse Gnd, une pour lancer les pulses de l'onde ultra-sonore (trig) et une délivrant un signal logique dont la largeur correspond au temps mis par l'onde pour faire un aller et retour.



Figure 2 – Capteur HC-SR05

L'onde acoustique se déplace à  $v = 340 \text{ m.s}^{-1}$ . Le temps fourni par la fonction `pulseIn()` est exprimé en microseconde. La conversion temps-distance est donc donné par

$$d = \frac{v \times \Delta t}{2}$$

Avec deux chiffres significatifs :

$$d[\text{cm}] = \frac{340 \Delta t[\mu\text{s}] \times 10^6}{2} \times 10^{-2} \approx \frac{\Delta t[\mu\text{s}]}{59}$$

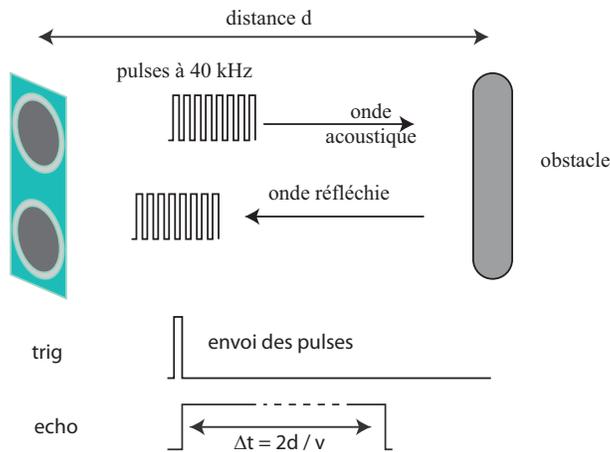


Figure 3 – Principe de mesure

## 2 Détection simple

Pour un simple affichage dans la console fourni par sketch, on pourra utiliser le code présent en première page. Le schéma de câblage ci-dessous est avec un LED que l'on oubliera dans un premier temps.

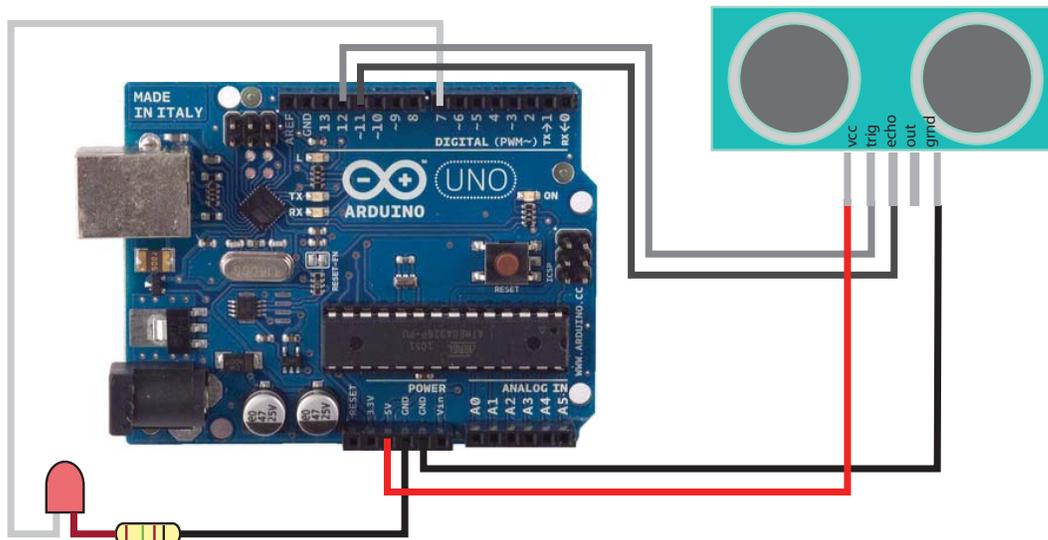


Figure 4 – Câblage principal

### 3 Détection avec affichage conditionnel



#### Exercice 1 :

Modifier le programme fourni pour effectuer un affichage supplémentaire "Attention" si la distance est inférieure à 20 cm.

### 4 Détection avec allumage conditionnel d'une LED

Une sortie logique haute (HIGH) permet de délivrer une tension de 5V suffisante pour allumer une LED. On utilise une résistance pour limiter l'intensité. Il faut donc rajouter un entier affecté de la broche correspondant à la LED

```
1 int LED = 7;
2 pinMode(LEDg, OUTPUT);
3 digitalWrite(trig, HIGH) ou digitalWrite(trig, LOW);
```



#### Exercice 2 :

Modifier le programme précédent pour permettre l'allumage de la LED si la distance est inférieure à 20 cm.

La fonction delay() permet de définir une pause dans le programme définie en milliseconde. Elle pourra être utilisée pour faire clignoter la LED.



#### Exercice 3 :

Modifier le programme précédent pour permettre l'allumage de la LED avec un clignotement de fréquence de plus en plus élevé si la distance se rapproche :

- clignotement de 1 fois par seconde si  $d < 40$  cm ;
- clignotement de 2 fois par seconde si  $d < 20$  cm ;
- éclairage continu si  $d < 10$  cm.