

# Club robotique - Explication et fonctionnement des composants électroniques

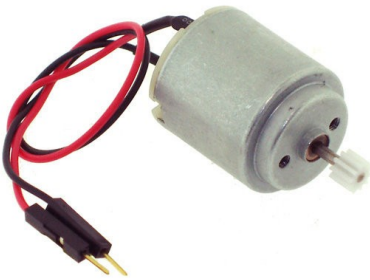
- [Qu'est ce qu'un servo-moteur ?](#)
- [Programmation d'un servomoteur 180°](#)
- [Programmation d'un servomoteur RC 360°](#)
- [Qu'est ce qu'un capteur à ultrasons ?](#)
- [Batterie et Shield 18650 & Co](#)
- [Capteur de température et d'humidité \(DHT22\)](#)
- [Qu'est ce qu'un ESP32](#)
- [Qu'est ce qu'un microcontrôleur ?](#)
- [Fonctionnement et programmation d'une LED RVB \(ESP32\)](#)



# Qu'est ce qu'un servo-moteur ?

## I Introduction

Lorsque vous créez un projet qui intègre des éléments robotique vous arrivez forcément à devoir faire un choix de moteur pour automatiser votre création. Plusieurs solutions existent, comme par exemple des moteurs à courant continu:



Ceux-ci ont l'avantage d'être très simple d'utilisation. Il suffit de connecter leurs deux fil d'alimentation à une batterie et ils se mettent à tourner.

Une autre solution serait d'utiliser des moteurs pas à pas :





Ces moteurs sont très pratique pour déplacer quelque-chose à un endroit précis, du fait que vous pouvez précisément le positionner à un angle donné. Par contre leur utilisation n'est pas aussi triviale que pour un moteur à courant continu.

Ce que je vous propose dans ce tutoriel c'est de découvrir les Servomoteurs. il en existe deux type :

- A contrôle d'angle
- A contrôle de vitesse de rotation

Les premiers ne tournent pas en continu, mais en général entre 0° et 180° et vous pouvez contrôler leur angle de rotation. Très pratique quand vous voulez faire un bras robotisé, piloter l'orientation des roues avant d'une voiture, ou bien contrôler un petit mécanisme. Les Servomoteurs suivants sont très pratiques lorsque vous souhaitez faire tourner un objet en contrôlant sa vitesse, par exemple des roues! :

[tiptopboards.com\\_169-thickbox\\_mini-servo-moteur-sg90-9g-mod\\_c3\\_a9lisme-arduino.jpg](http://tiptopboards.com_169-thickbox_mini-servo-moteur-sg90-9g-mod_c3_a9lisme-arduino.jpg)

Ce sont des Servomoteurs à contrôle d'angle sur 180° et le tutoriel va donc porter sur ce type de moteur.

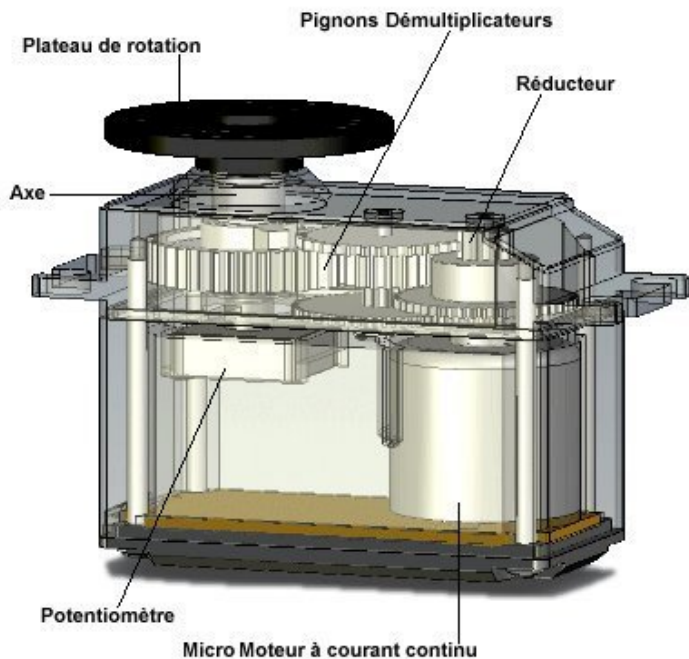
## II Fonctionnement

<https://fr.wikipedia.org/wiki/Servomoteur> --> fonctionnement

Les Servomoteurs intègrent au sein d'un même boîtier un moteur à courant continu, un potentiomètre, un réducteur et un circuit de contrôle. L'idée est que la valeur d'angle est mesurée grâce au potentiomètre et le circuit de contrôle fait tourner le moteur et corrige l'orientation. Voilà une image qui donne une idée du fonctionnement interne:

[upload.wikimedia.org\\_wikipedia\\_commons\\_d\\_d3\\_micro\\_servo.jpg](upload.wikimedia.org_wikipedia_commons_d_d3_micro_servo.jpg)

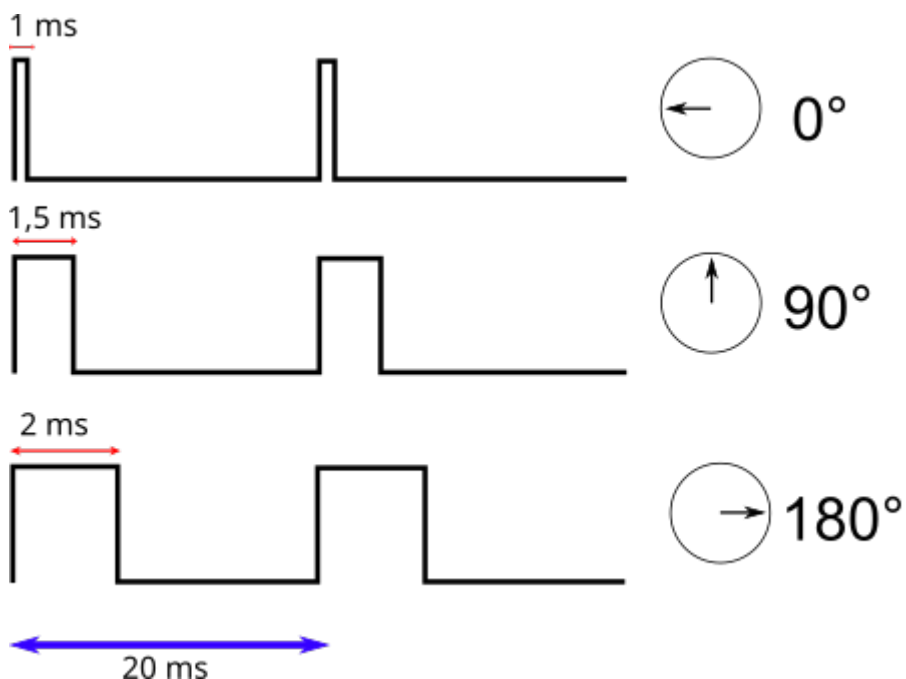




*Eric G*

Vous l'avez sûrement remarqué, le Servomoteur a trois fils. Le fil marron correspond à la masse, le fil rouge au 5 Volts et le fil orange à l'envoi de données. C'est par le fil orange que nous allons envoyer le signal pour la commande de l'angle voulu au Servomoteur.

L'instruction par le fil orange s'envoie sous la forme d'un signal PWM (Pulse Width modulation=Modulation en largeur d'impulsion). Le principe est que l'envoi d'instruction se fait par un signal électrique qui passe de façon régulière et rapide (30-50Hz ou 300Hz) de 0 à 5 Volts. La valeur de l'angle voulu est définie par le rapport entre le temps où le signal est à 5 Volts et le temps où celui-ci est à 0 Volt. Par exemple pour un angle de 0°, on envoie 5V pendant 1ms puis 0V pendant 19ms : Le signal est à 5V pendant 5% du temps (1ms/20ms). Une image vous donnera une meilleure idée:





Source : <https://upload.wikimedia.org/wikipedia/commons/thumb/f/f6/TiemposServo.svg/220px-TiemposServo.svg.png>

# Servomoteur contrôlé en position "180°" ou en vitesse "360°"

[https://www.makerguides.com/how-to-control-a-360-degree-servo-motor-with-arduino/#What\\_makes\\_a\\_continuous\\_servo\\_motor\\_special](https://www.makerguides.com/how-to-control-a-360-degree-servo-motor-with-arduino/#What_makes_a_continuous_servo_motor_special)

[https://www.atelierdelarobotique.fr/servomoteur-standard-vs-servomoteur-a-rotation-continue-avec-arduino-programmation-et-applications#elementor-toc\\_\\_heading-anchor-2](https://www.atelierdelarobotique.fr/servomoteur-standard-vs-servomoteur-a-rotation-continue-avec-arduino-programmation-et-applications#elementor-toc__heading-anchor-2)



# Programmation d'un servomoteur 180°

## Programmation des Servomoteurs avec mBlock et Arduino IDE

### Objectif :

Comprendre le fonctionnement des servomoteurs et apprendre à les programmer avec **mBlock** et **Arduino IDE**.

---

## 1. Introduction aux Servomoteurs

### Qu'est-ce qu'un servomoteur ?

Un **servomoteur** est un moteur à rotation limitée, généralement 180°, qui est contrôlé par un signal **PWM (Pulse Width Modulation)**. Il est utilisé dans de nombreux projets tels que les **bras robotisés**, les **portes automatiques** et les **robots éducatifs**.

→ Pour en savoir plus sur les servomoteurs, cliquez [ici](#).

---

## 2. Configuration et Programmation avec mBlock



## Présentation

∞ régler la sortie du port PWM 5 sur 0

∞ set servo pin 9 angle as 90

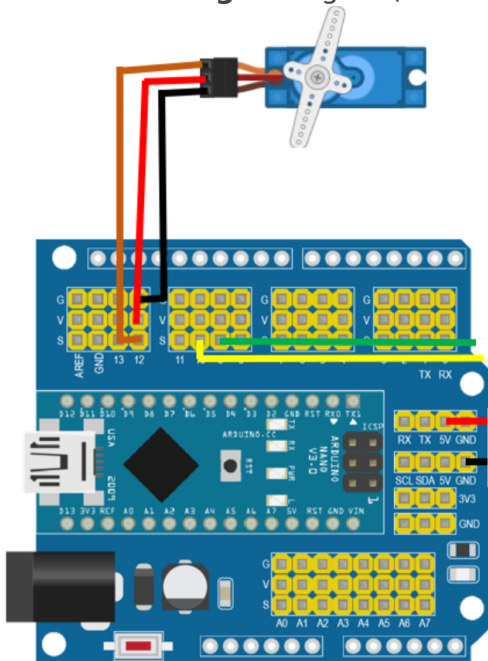
## servomoteurs

Dans mBlock, la gestion des servomoteurs se fait à l'aide d'un **bloc dédié** permettant de définir un angle précis entre 0° et 180°.

## Branchement d'un servomoteur à un Arduino Nano

Un servomoteur possède **trois câbles** :

- **Noir ou Marron** → GND (Masse)
- **Rouge** → 5V (Alimentation)
- **Jaune ou Orange** → Signal (Entre les broches **2 et 13** du Nano)



## Exercice 1 : Programmer un servomoteur en boucle

**Consigne** : Programmer un **servomoteur 180°** pour qu'il effectue en boucle un mouvement de **0°** à **180°** puis de **180°** à **0°**, avec une pause de **2 secondes** entre chaque mouvement.

## Correction :





## Exercice 2 :

### Programmation d'un Mouvement Automatisé avec mBlock

Créer un programme permettant :

- **Faire bouger le servomoteur progressivement** de 0° à 180° et inversement.
- **Répéter** le mouvement avec une **boucle continue**.

### Correction :





## 3. Configuration et Programmation avec Arduino IDE

### Exercice 1 : Programmer un servomoteur pour un mouvement en boucle

**Consigne :** Faire osciller un servomoteur de **0° à 180°** et inversement, avec une pause de **2 secondes** entre chaque mouvement.

#### Correction : Code Arduino IDE

```
#include <Servo.h>
```

```
Servo monServo; // Création de l'objet Servo
```



```

void setup() {
    monServo.attach(9); // Connexion du servomoteur sur la broche D9
}

void loop() {
    monServo.write(0);    // Position à 0°
    delay(2000);          // Pause de 2 secondes
    monServo.write(180);  // Position à 180°
    delay(2000);          // Pause de 2 secondes
}

```

## Exercice 2 :

### Programmation d'un Mouvement Automatisé avec Arduino IDE

Créer un programme permettant :

- **Faire bouger progressivement** le servomoteur de **0° à 180°** et inversement.
- D'utiliser une **boucle pour répéter** ce mouvement en continu.\*

### Correction :

```

#include <Servo.h>

Servo monServo;

void setup() {
    monServo.attach(9);
}

void loop() {
    for (int pos = 0; pos <= 180; pos++) { // De 0° à 180°
        monServo.write(pos);
        delay(15); // Ajuste la vitesse du mouvement
    }

    for (int pos = 180; pos >= 0; pos--) { // De 180° à 0°
        monServo.write(pos);
        delay(15);
    }
}

```



## 4. Discussion et Applications

### Applications pratiques

- **B**ras robotisé
- **P**oubelle connectée
- **L**ittle Bot

### Questions et suggestions d'amélioration

- Ajouter un **potentiomètre** pour contrôler l'angle du servomoteur.
- Utiliser un **bouton poussoir** pour déclencher le mouvement.
- Remplacer le **servomoteur 180°** par un **servomoteur à rotation continue**.



# Programmation d'un servomoteur RC 360°

## Objectif :

Comprendre le fonctionnement des servomoteurs et apprendre à les programmer avec **mBlock** et **Arduino IDE**.

---

## 1. Introduction aux Servomoteurs RC 360°

Qu'est-ce qu'un servomoteur RC 360° ?

Un **servomoteur RC 360°** (ou « continu ») est un petit moteur qui **peut tourner en continu dans les deux sens** (comme une roue), contrairement aux servos classiques qui ne tournent que sur un angle limité (en général 180° maximum).

⚙ À quoi ça sert dans un robot comme le LittleBot ?

Dans le LittleBot, ces servos sont utilisés pour faire **tourner les roues** du robot. En envoyant des signaux différents, on peut :

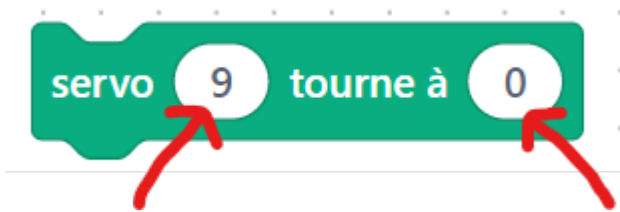
- **Faire avancer** le robot (les deux roues tournent en avant),
- **Faire reculer** le robot (les deux roues tournent en arrière),
- **Faire tourner** (une roue avance, l'autre recule ou reste fixe).

## 2. Configuration et Programmation avec mBlock

Présentation des blocs spécifiques à la gestion des servomoteurs

Pour un servomoteur RC 360° on va utiliser le bloc ci-dessous que l'on peut trouver dans l'extensions "Servo360" ou ici "[servo.next](#)" à télécharger et faire un glisser-déposer sur Mblock.



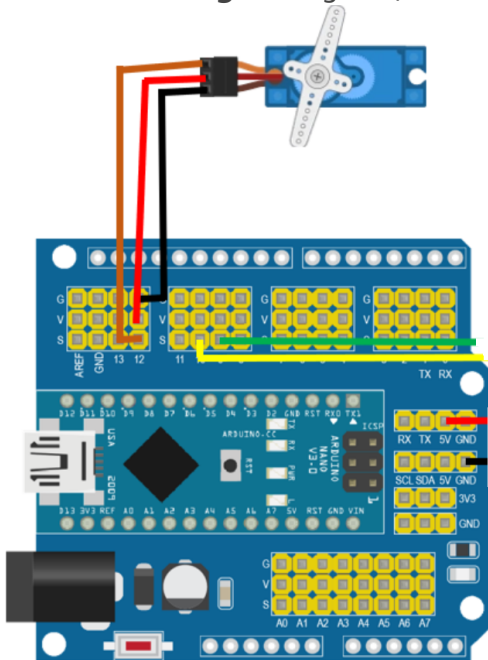


Numéro de broche                      vitesse du servomoteur (dans un sens entre [-100 et 0[ et dans l'autre entre ]0 et 100]). A 0 on est donc à l'arrêt

## Branchement d'un servomoteur à un Arduino Nano

Un servomoteur possède **trois câbles** :

- **Noir ou Marron** → GND (Masse)
- **Rouge** → 5V (Alimentation)
- **Jaune ou Orange** → Signal (Entre les broches **2 et 13** du Nano)



## Exercice sur Mblock

**Consigne :** Programmer un **servomoteur RC 360°** pour qu'il effectue en boucle un changement de **sens de rotation** à **vitesse maximale** et cela toute les **2 secondes**.

# 3. Configuration et Programmation avec Arduino IDE

## Exercice sur Arduino IDE



**Consigne :** Programmer un **servomoteur 360** pour qu'il effectue en boucle un changement de **sens de rotation** à **vitesse maximale** et cela toute les **2 secondes**.

## Correction :

```
#include <Servo.h>

Servo monServo; // Création de l'objet Servo

void setup() {
    monServo.attach(9); // Connexion du servomoteur sur la broche D9
}

void loop() {
    monServo.writeMicroseconds(2000); // Vitesse maximale dans un sens
    delay(2000); // Tourne pendant 2 secondes
    monServo.writeMicroseconds(1000); // Vitesse maximale dans l'autre sens
    delay(2000); // Tourne pendant 2 secondes
}
```

# 4. Discussion et Applications

## Applications pratiques

-  **Bras robotisé**
-  **Poubelle connectée**
-  **Little Bot**

## Questions et suggestions d'amélioration

- ☐ Ajouter un **potentiomètre** pour contrôler l'angle du servomoteur.
- ☐ Utiliser un **bouton poussoir** pour déclencher le mouvement.
- ☐ Remplacer le **servomoteur 180°** par un **servomoteur à rotation continue**.



# Qu'est ce qu'un capteur à ultrasons ?



HC-SR04 est un capteur à ultrasons qui est

principalement utilisé pour la mesure de distance. En émettant des ondes ultrasonores et en mesurant le temps pris par ces ondes pour rebondir après avoir frappé un objet, le HC-SR04 peut déterminer avec précision la distance à laquelle cet objet se trouve. De par son coût abordable et sa facilité d'intégration avec des plateformes telles qu'Arduino, le HC-SR04 est devenu un choix prisé parmi les amateurs de bricolage et les professionnels.

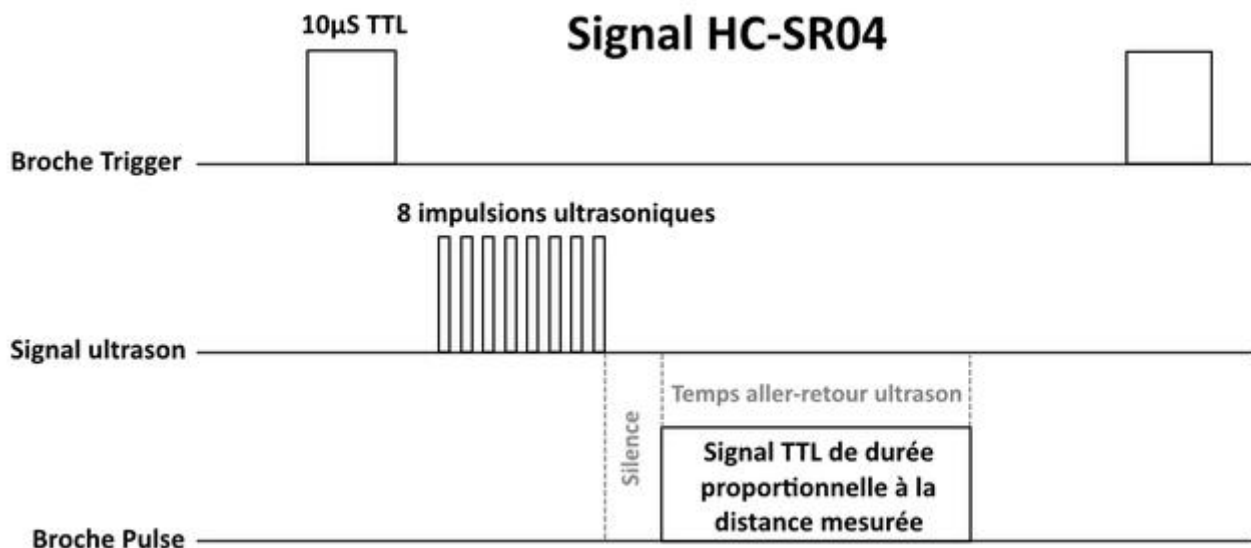
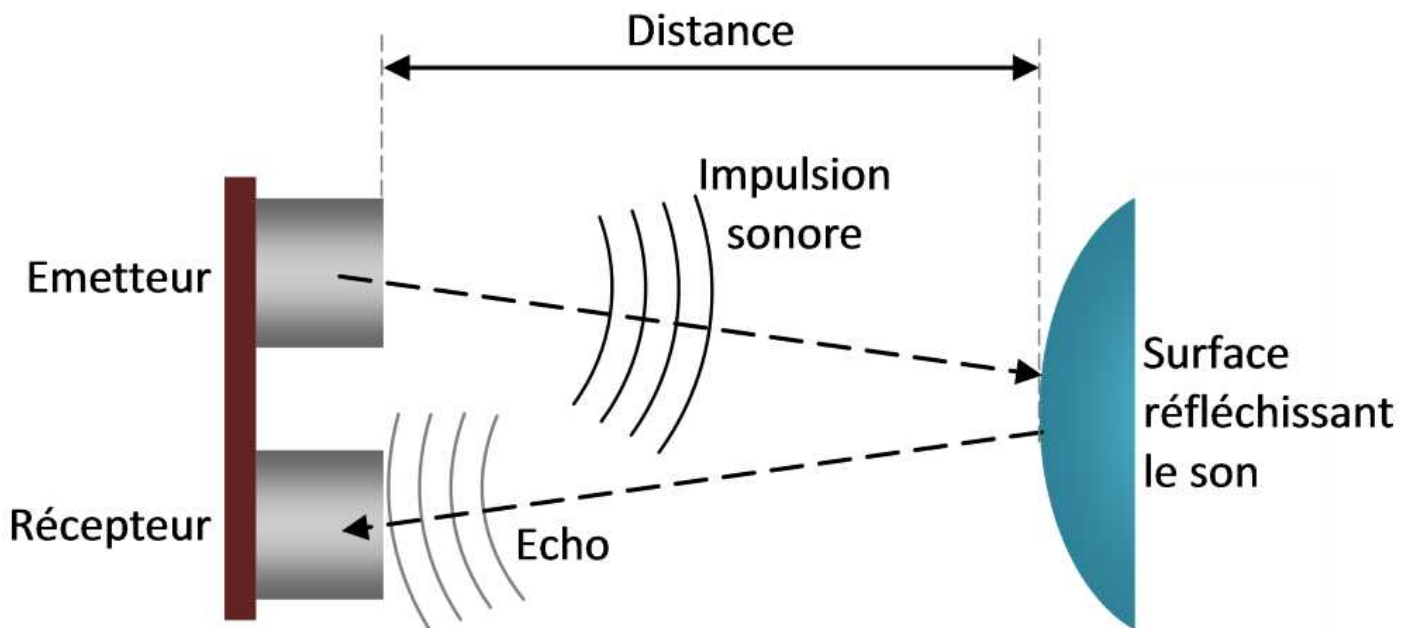
## Principe de fonctionnement

Le fonctionnement d'un capteur à ultrason comme le HC-SR04 est assez simple. Il comporte deux éléments principaux : un émetteur ultrasonore et un récepteur ultrasonore. Voici les étapes clés du fonctionnement du capteur :

1. Lorsque le capteur est alimenté, l'émetteur envoie une série de 8 impulsions ultrasoniques de  $10\mu s$  à une fréquence spécifique (généralement de 40 kHz).
2. Lorsque une impulsion sonore atteint un objet, elle rebondit et est renvoyée vers le récepteur ultrasonore comme un écho.

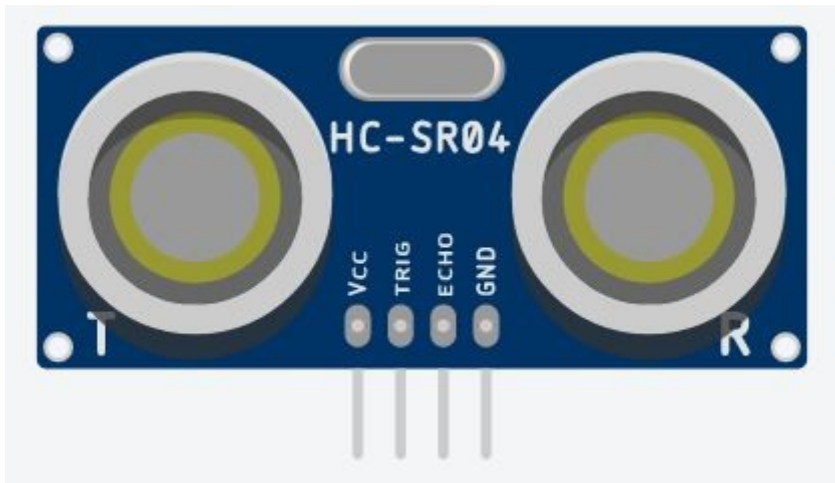


3. Le capteur mesure le temps entre le moment où l'impulsion a été émise et celui où l'écho a été reçu.
4. En utilisant la vitesse connue du son dans l'air (environ 343 m/s ou 34,3 cm/ $\mu$ s) et la durée de l'écho mesurée, le capteur calcule la distance jusqu'à l'objet en utilisant la formule :  $\text{distance} = (\text{durée de l'écho} / 2) * \text{vitesse du son}$ .
5. Le résultat est ensuite converti en une distance numérique et envoyé au Arduino via une sortie numérique





# Description du capteur HC-SR04



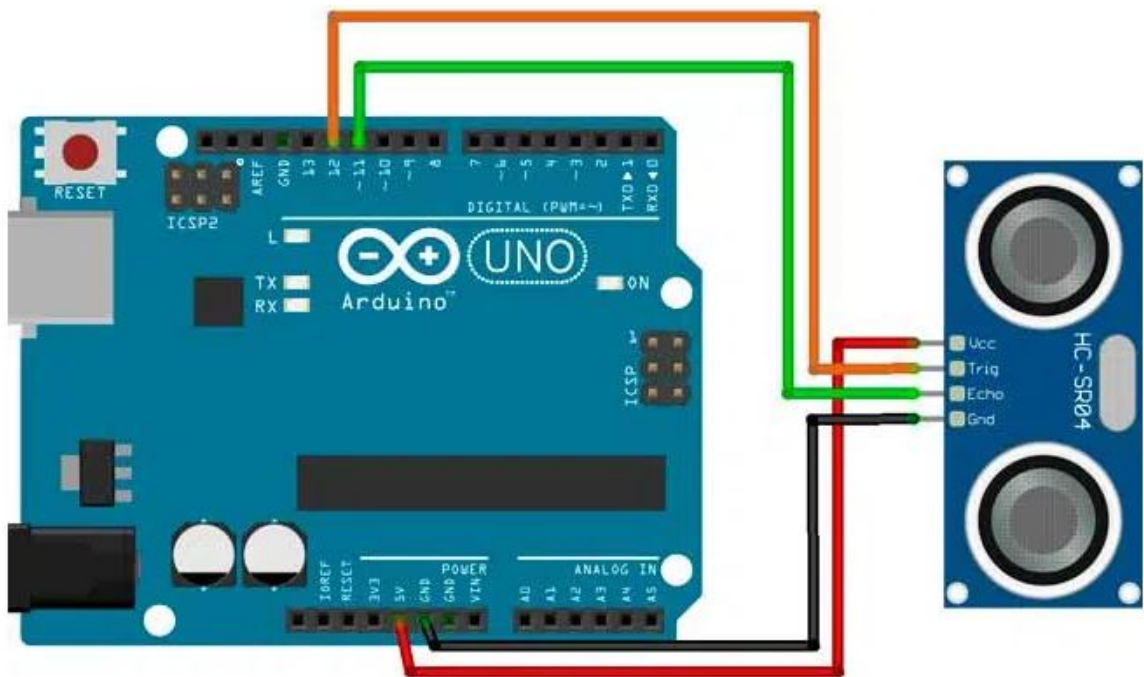
Broche	Description
VCC	Il s'agit de la broche d'alimentation. Elle nécessite généralement une entrée de 5V Courant Continu, la rendant directement compatible avec des cartes comme Arduino.
Trig (Déclenchement)	Cette broche est utilisée pour initier le capteur à émettre une onde ultrasonore. En envoyant une impulsion haute d'au moins 10µs à cette broche, le HC-SR04 émettra une série de 8 impulsions d'ultrasons à 40 kHz.
Echo	Une fois l'onde ultrasonore émise et qu'elle rebondit après avoir frappé un objet, la broche Echo fournit une impulsion de sortie. La largeur de cette impulsion est proportionnelle à la distance de l'objet par rapport au capteur. En mesurant la durée de cette impulsion, Arduino peut déterminer la distance jusqu'à l'objet.
GND (Masse)	Cette broche est connectée à la masse du circuit.

## Le câblage :



Pour connecter le capteur HC-SR04 à une carte Arduino, suivez ces étapes :

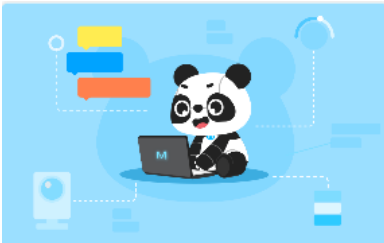
1. Connectez la broche VCC du HC-SR04 à la broche 5V sur la carte arduino uno. Cela assure que le capteur reçoive la puissance nécessaire pour son fonctionnement.
2. Reliez la broche GND (Masse) du HC-SR04 à l'une des broches de masse (GND) d'Arduino. Cela établit une masse électrique commune entre le capteur et arduino.
3. Connectez la broche Trig du HC-SR04 à une broche numérique d'Arduino, par exemple, la broche D12. Cette broche est responsable de l'envoi d'un signal pour déclencher le capteur afin qu'il émette les ondes ultrasonores.
4. Reliez la broche Echo du HC-SR04 à une autre numérique sur l'Arduino, comme la broche D11. Cette broche détecte l'onde ultrasonore écho après réflexion sur un objet.





## Programmer un HC-SR04

**!!Attention pour Mblock ne pas oublier d'installer l'extension en tapant "ultrasons" dans la barre de recherche.**





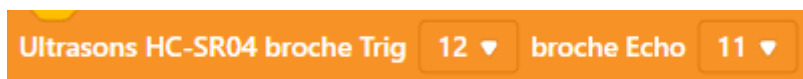
## ultrasons HC-SR04

Développeurs: bessa...  

+ Ajouter

Description des nouveaux blocs à utiliser :

- Ce bloc permet de d'indiquer à l'Arduino quelles broches l'on va utiliser pour le 'Trig' et pour l' 'Echo'



- Ce bloc permet de calculer à quel distance en cm l'on est d'un potentiel obstacle.



- Ce bloc contient la valeur calculer dans le bloc précédent



Attention à bien indiquer à coté de 'Trig' et de 'Echo' les broches sur lesquels vous êtes branchés.  
Si votre code ne marche pas vérifier bien cela d'abord.

- Ce bloc permet d'écrire dans le port série



C'est quoi un port série ?



Le port série, c'est un peu comme une voie de communication entre ton ordinateur et le robot.

Imagine que l'ordinateur et le robot discutent.

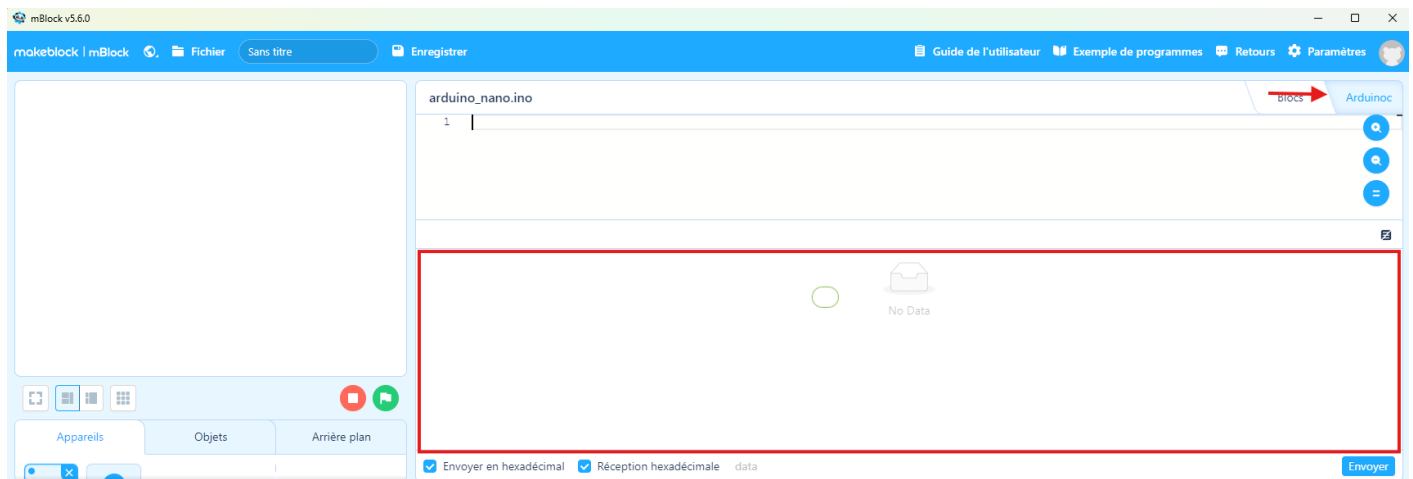
Le port série, c'est le fil (ou le tunnel) par lequel les messages passent, un caractère à la fois, dans un seul sens à la fois.

En langage simple :

- Quand tu programmes le robot avec mBlock, tu envoies le programme par le port série.
- Tu peux aussi recevoir des messages venant du robot, comme :
  - la distance détectée par un capteur,
  - des valeurs de température,
  - ou tout ce que tu lui demandes de t'envoyer.

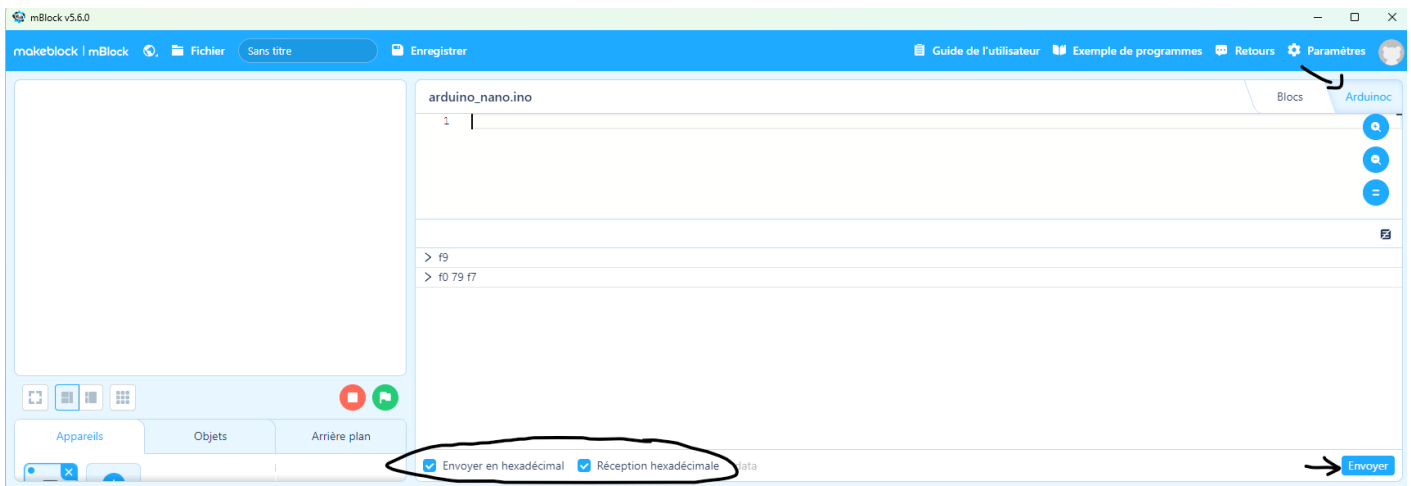
Où se trouve le port série ?

Allez dans 'Arduinoc', et il se trouve dans l'encadrée rouge



Avant de téléverser votre code dans l'Arduino nano, accéder au port série et décocher ses cases en allant dans 'Arduinoc' puis cliquer sur envoyer :





Chaque exercice est à faire valider par un animateur

## Exercice : Afficher les valeurs la distance de l'ultrason dans le port série

Ce code permet de mesurer une distance et de l'afficher sur le moniteur série du logiciel Arduino IDE.

```
// définition des numéros de broches

const int trigPin = 12;

const int echoPin = 11; // définition des variables

long duration;

int distance;

void setup()
```



```
{

pinMode(trigPin, OUTPUT); // Définit le trigPin comme sortie

pinMode(echoPin, INPUT); // Définit le echoPin comme entrée

Serial.begin(9600); // Commence la communication série

}

void loop()

{

// Efface le trigPin

digitalWrite(trigPin, LOW);

delayMicroseconds(2); // Met le trigPin à l'état HIGH pendant 10 microsecondes

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW); // Lit le echoPin, renvoie le temps de trajet de l'onde sonore en
microsecondes

duration = pulseIn(echoPin, HIGH); // Calcul de la distance

distance = duration * 0.034 / 2; // La vitesse du son est d'environ 0.034 cm par microseconde

Serial.print("Distance: "); // Affiche la distance sur le moniteur série

Serial.println(distance);

delay(2000);

}
```



Source :

<https://www.moussasoft.com/hc-sr04-capteur-ultrason-avec-arduino>

<https://www.carnetdumaker.net/articles/mesurer-une-distance-avec-un-capteur-ultrason-hc-sr04-et-une-carte-arduino-genuino/>



# Batterie et Shield 18650 & Co

## Carte de charge et décharge

Les shields de charge et décharge des batteries 18650, Lithion-Ion, 3.7V permettent :

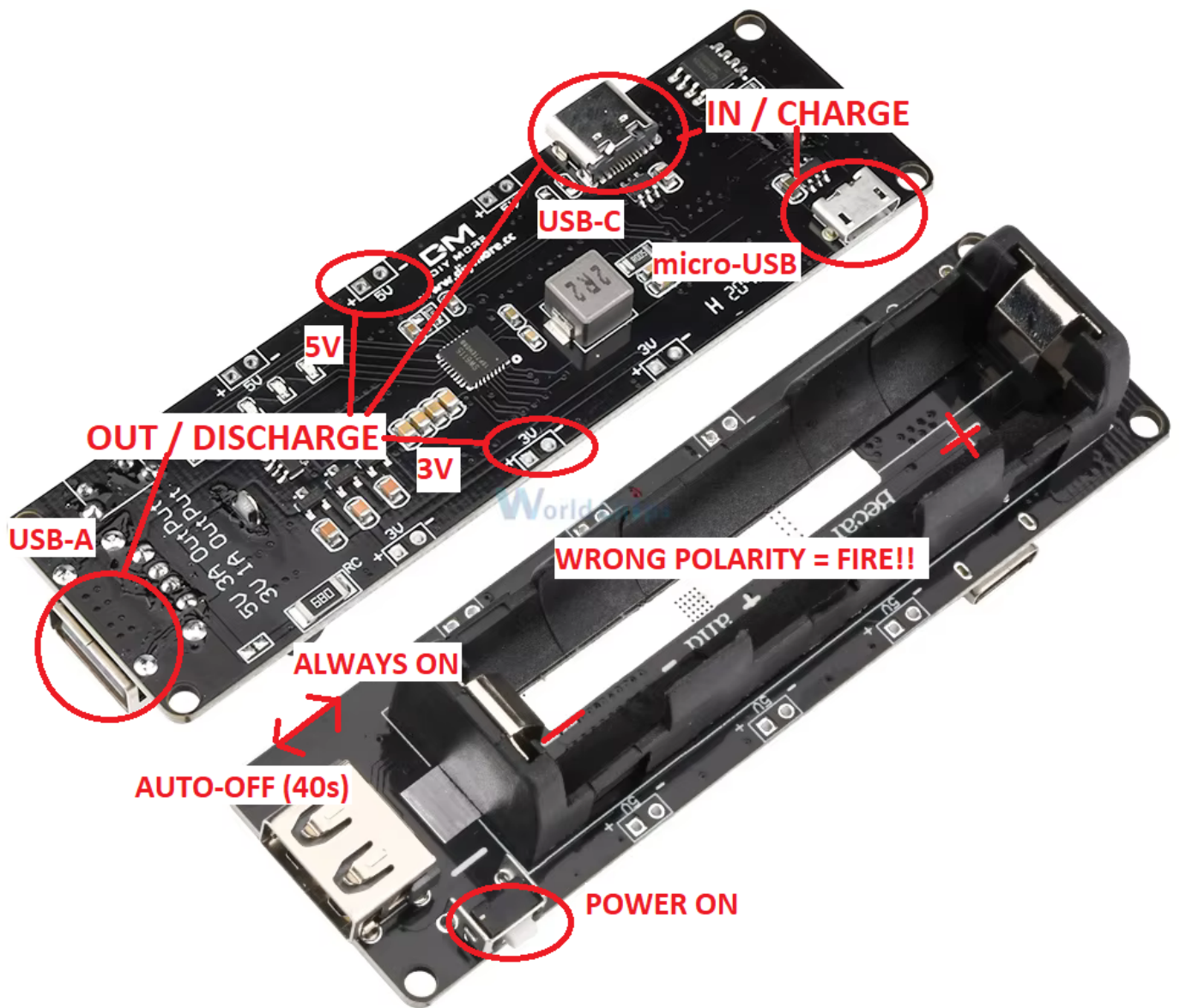
- INput : charge de la batterie via ports
  - USB-C ou micro-USB
  - 1.8A 5V (max 6.5V)
- OUTput : décharge de la batterie (alimentation d'une carte électronique) via ports
  - USB-C, USB-A
  - pins 5V (3A) / 3.3V (1A) dispos sur les bords de la carte
- Bouton noir SW1 :
  - "Vers l'intérieur de la carte " : Alimentation permanente **ALWAYS ON**
  - "Vers l'extérieur de la carte" : Mise en veille automatique au bout de 40s **AUTO-OFF**.  
Consommation 3uA après la mise en veille.
- Bouton blanc Power ON/OFF :
  - Appuyer pour allumer **ON**
  - Appuyer **2 fois** pour éteindre **OFF**

On a une version simple accu "V3" et une double accu "V8" :

1. simple accu "V3" :

- <https://fr.aliexpress.com/item/32853943437.html>

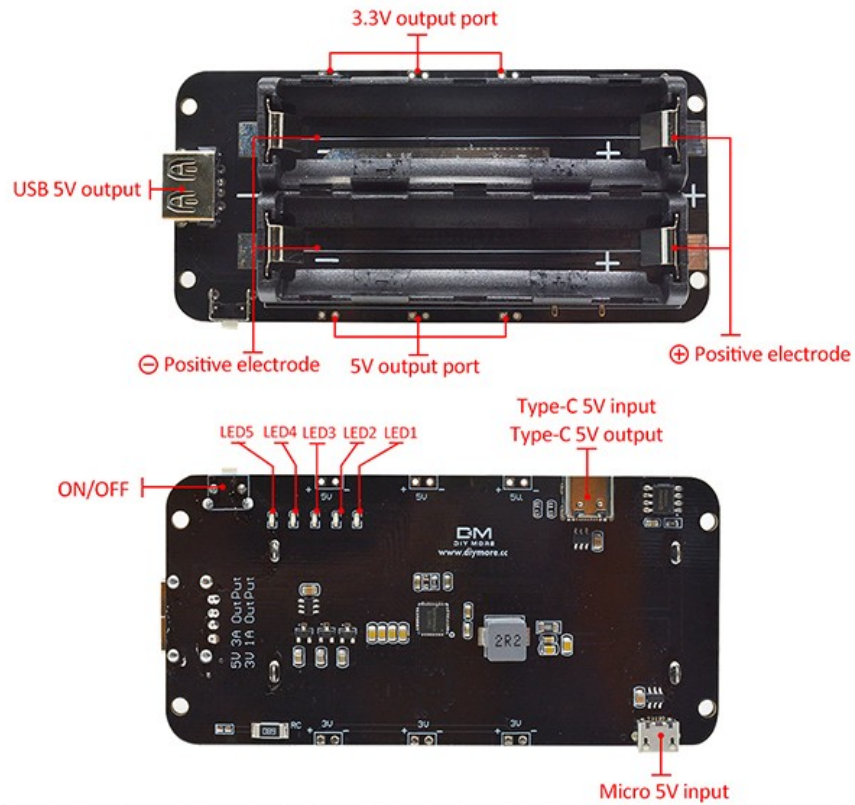




2. double accu "V8" :

- <https://fr.aliexpress.com/item/1005001285390889.html>
- <https://www.amazon.fr/dp/B0822Q4VS4>





**Note:** The installation of the battery must be determined positive and negative, the board has been clearly marked positive and negative! Installation errors will burn out the module!



# Capteur de température et d'humidité (DHT22)

## Objectif :

Comprendre le fonctionnement du capteur de température et d'humidité et apprendre à le programmer avec **mBlock** et **Arduino IDE**.

## 1. Introduction du capteur DHT22

Qu'est-ce qu'un capteur de température et humidité et comment il fonctionne?

Le capteur DHT22 est un dispositif utilisé pour mesurer la température et l'humidité de l'air. Il fonctionne grâce à deux composants principaux : un thermistor pour la mesure de la température et un capteur capacitif pour l'humidité.

1. **Température** : Le thermistor, qui change de résistance en fonction de la température, permet de mesurer celle-ci avec une précision d'environ  $\pm 0.5^{\circ}\text{C}$  dans une plage de  $-40^{\circ}\text{C}$  à  $80^{\circ}\text{C}$ .
2. **Humidité** : Le capteur capacitif d'humidité détecte les variations dans la capacité électrique causées par l'humidité dans l'air, offrant une précision d'environ  $\pm 2$  à 5% pour une humidité relative comprise entre 0% et 100%.

Le DHT22 communique les données via une seule broche numérique (data). Après une courte initialisation, il envoie les informations sous forme de bits, codées dans un protocole spécifique. L'appareil peut être interrogé à des intervalles d'environ 2 secondes, car il a une période de rafraîchissement des données assez lente.

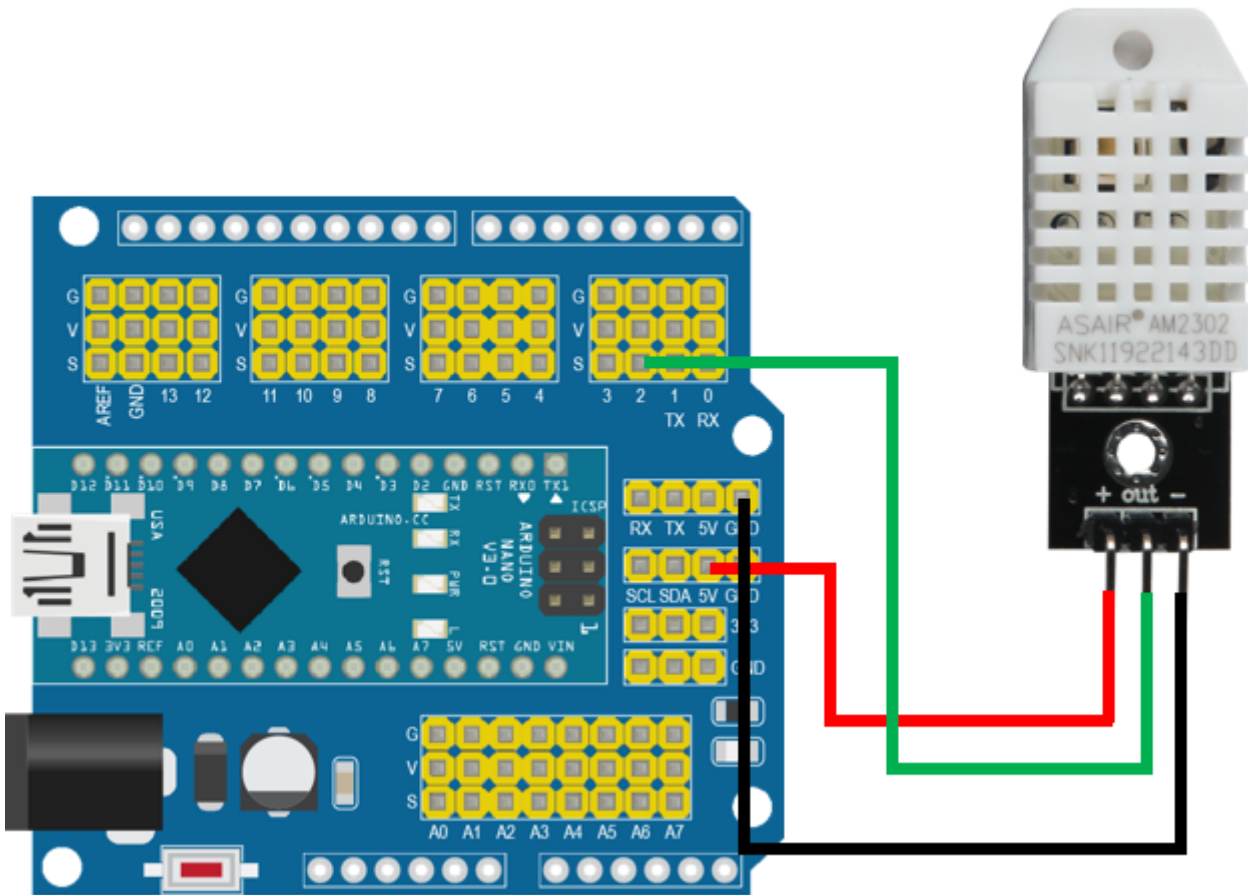
Ce capteur est populaire pour sa simplicité d'utilisation, notamment avec des microcontrôleurs comme Arduino ou Raspberry Pi, et est très utilisé dans des projets de domotique ou de surveillance de l'environnement.

## 2. Câblage du DHT22 avec l'Arduino nano

- VCC(+) - > pin 5V.
- DAT(out) -> pin 2 S.



- GND(-) -> pin GND



### 3. Configuration et Programmation avec mBlock

Commencer par installer l'extension pour le capteur en cliquant sur ce bouton [+](#) extension, puis taper dans la barre de recherche "DHT22" et l'ajouter.



Faire un temps de brainstorming sur la logique que devra avoir le code final. Puis laisser le groupe faire le code en les aidants si besoin.





Avant de téléverser le code n'oubliez pas de décocher ces 2 cases ci-dessous qui se situe dans le moniteur série.

☐ Envoyer en hexadécimal    ☐ Réception hexadécimale

## 4. Configuration et Programmation avec Arduino IDE

----- à remplir -----



# Qu'est ce qu'un ESP32



Qu'est ce qu'un  
microcontrôleur ?



# Fonctionnement et programmation d'une LED RVB (ESP32)

## 📋 Objectif

Faire changer la couleur d'une LED RGB (HW-479) contrôlée par une carte ESP32-WROOM-32E.  
Les élèves apprendront :

- Ce qu'est une LED RGB
- Comment connecter des composants électroniques
- Comment programmer avec Arduino IDE pour faire varier les couleurs de la LED RGB

## 📋 Matériel

- 1 carte ESP32 DevKit (ESP32-WROOM-32E)
- 1 module LED RGB HW-479 (à cathode commune)
- 4 fils Dupont
- 1 câble USB

## 1. Définition simple

Une **diode électroluminescente**, appelée **LED** (Light Emitting Diode en anglais), est un **petit composant électronique** qui émet de la lumière quand on fait passer un courant électrique.

## 2. Comment fonctionne une LED ?



- Une LED est une **petite lampe** qui s'allume quand on fait passer de l'électricité dans un seul sens.
- À l'intérieur, il y a un **matériau spécial** qui transforme l'électricité en lumière.
- Quand les électrons (petites particules d'électricité) se déplacent dans ce matériau, ils **libèrent de l'énergie** sous forme de lumière.
- La couleur de la lumière dépend du type de matériau utilisé.

## 3. Avantages des LED

- Consommation d'énergie **très faible** comparée aux ampoules classiques.
- **Durée de vie très longue** (des milliers d'heures).
- Allumage instantané sans délai.
- Possibilité d'avoir différentes couleurs sans filtres (LED rouges, vertes, bleues, blanches, etc.).

## 4. Utilisations courantes

- Indicateurs lumineux sur les appareils électroniques (ex : témoin d'allumage).
- Éclairage décoratif et domestique.
- Écrans et panneaux lumineux.
- Dans la robotique et l'électronique pour signaler des états ou réaliser des animations.

## 4. Programmation