SESSION 05 · DÉBUTANT 🟢 - Mesure de la température et de l'humidité

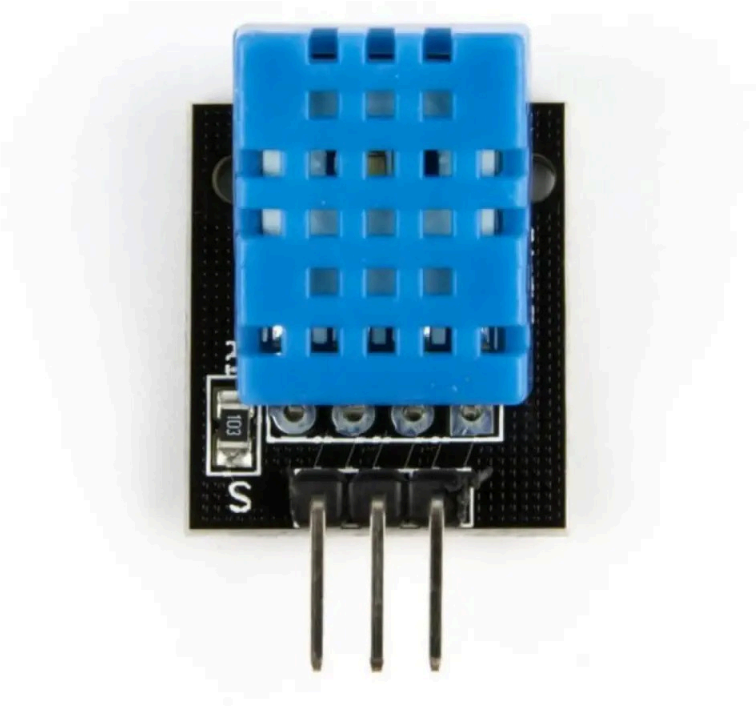
🎯 Objectifs

- Initiation à la transmission numérique
- Apprendre à utiliser un capteur de température et d'humidité
- Installation et utilisation d'une bibliothèque (library)

🧰 Matériel

Composant	Qté	Note
Arduino Uno + câble USB	1	
Breadboard	1	
Module capteur de température et d'humidité DHT11	1	
Câbles Dupont	3	

🧰 Le capteur DHT11



Module de capteur de température et d'humidité DHT11

- Mesure la température et l'humidité
- Tension de fonctionnement : 3,0 à 5,5 V
- Courant de fonctionnement : 0,3 mA
- Précision : ± 1 °C et ± 1 %.

Le Module de capteur de température et d'humidité DHT11 est l'un des capteurs les plus courants. La fonction principale de ce capteur est de mesurer la température et l'humidité de l'environnement.

Le capteur est également calibré en usine et donc facile à interfacer avec n'importe quel microcontrôleur comme Arduino. Le capteur peut mesurer la température de 0 à 50 °C et l'humidité de 20 à 90 % avec une précision de ±1 °C et ±1 %.

Le module DHT11 est constitué de :

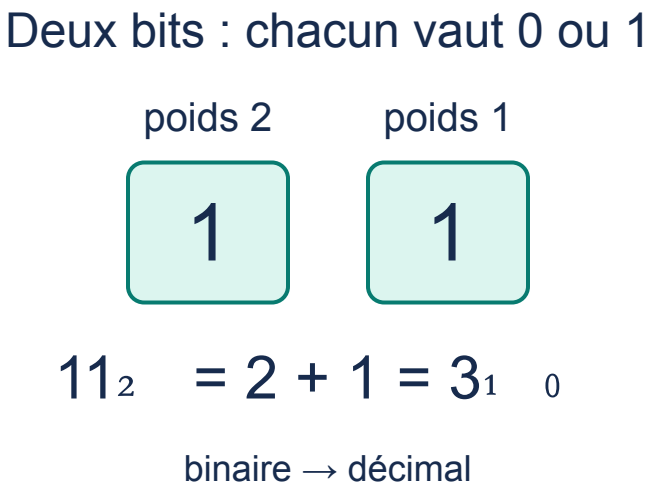
- Un capteur d’humidité résistif : sa résistance varie selon l’humidité de l’air.

- Une thermistance NTC : sa résistance varie avec la température.
- Un circuit électronique intégré : il mesure ces variations, applique l'étalonnage et transmet la température et l'humidité sous forme numérique.
- Un boîtier ajouré : il permet à l'air d'atteindre les éléments sensibles.

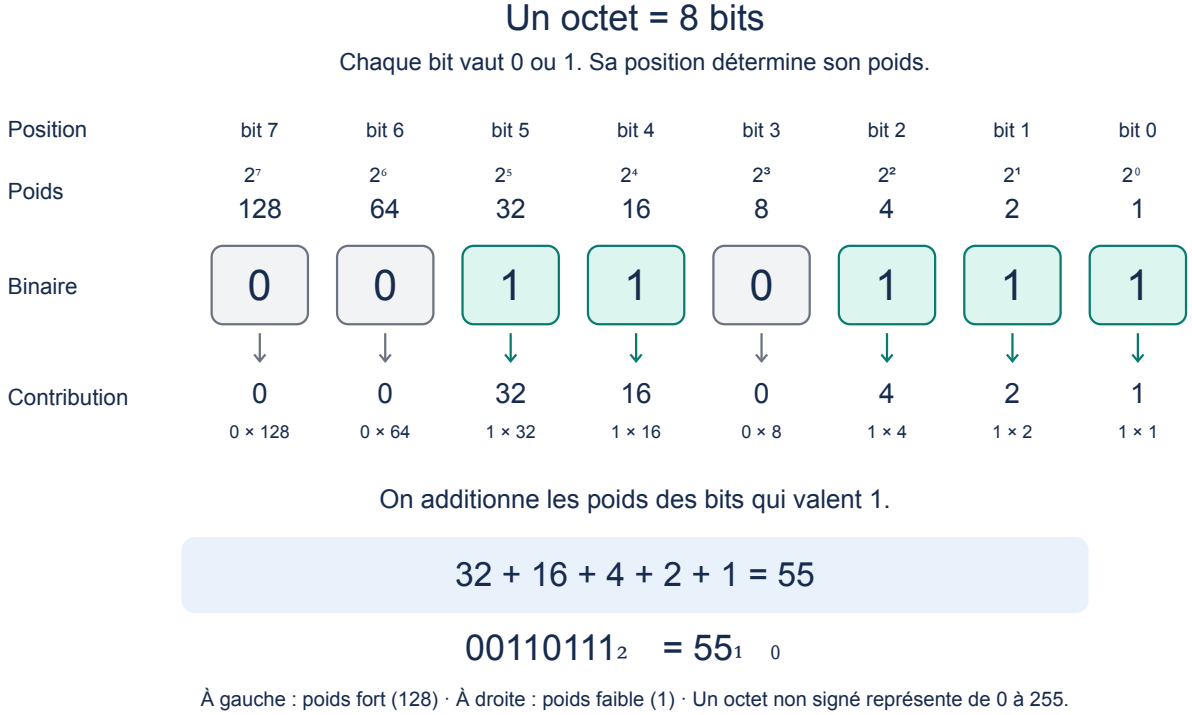
Sur les modules montés sur une petite carte, on trouve aussi généralement une résistance de rappel pour la ligne de données, parfois un voyant d'alimentation et un condensateur.

L'interface DATA du DHT11 utilise un seul fil pour transmettre la température et l'humidité sous forme numérique. Ce fil est bidirectionnel : l'Arduino lance la communication, puis le DHT11 répond. Les deux appareils doivent partager la même masse GND.

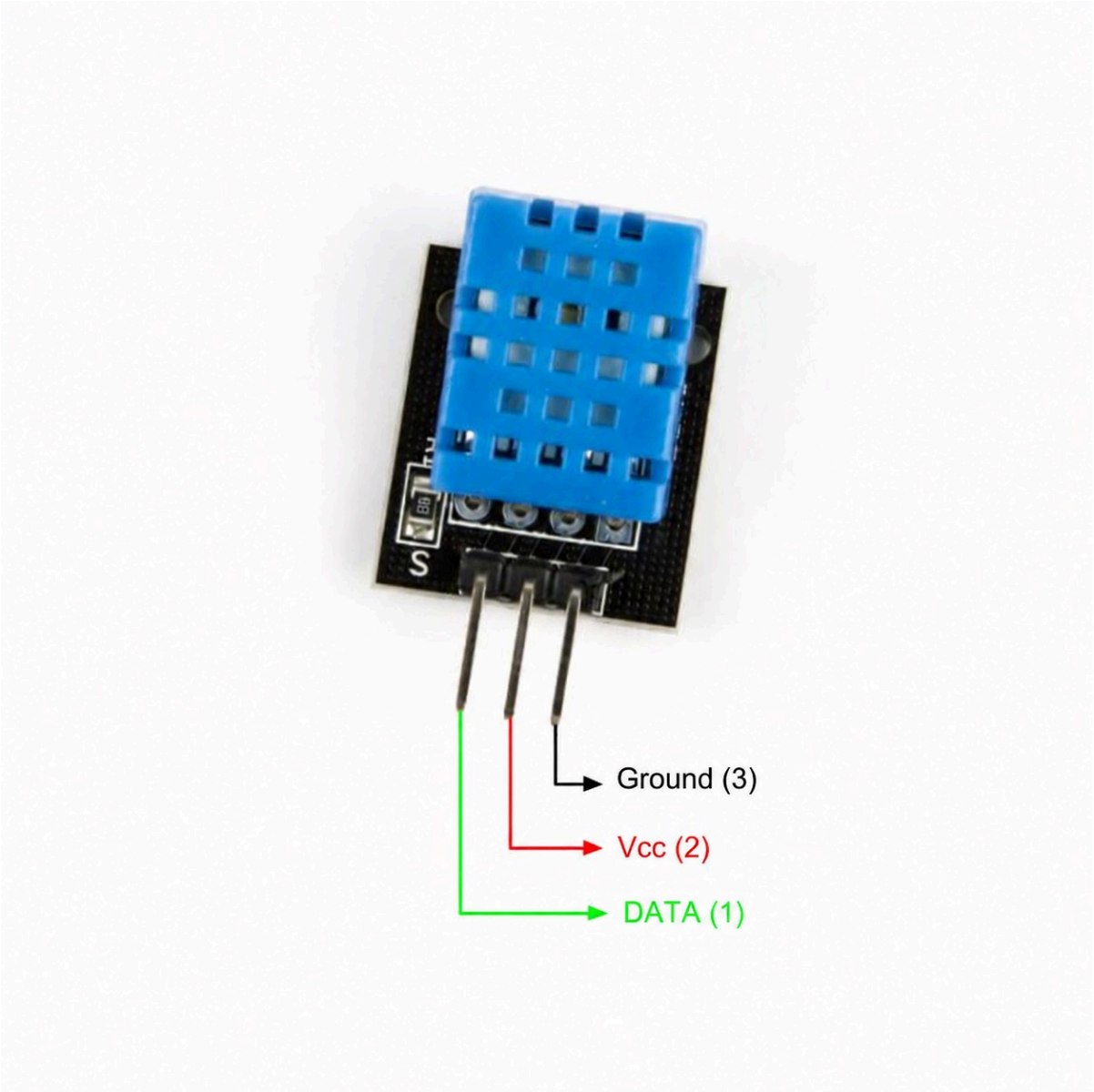
Rappel sur la notation binaire



> 📍 **Valeur d'un octet**
> Un octet est un groupe de 8 bits. Chaque bit peut prendre la valeur 0 ou 1.
> Un octet permet de représenter 256 valeurs différentes, soit les nombres entiers de 0 à 255 lorsqu'il est non signé.
> Exemple : 00110111 en binaire correspond à 55 en décimal.



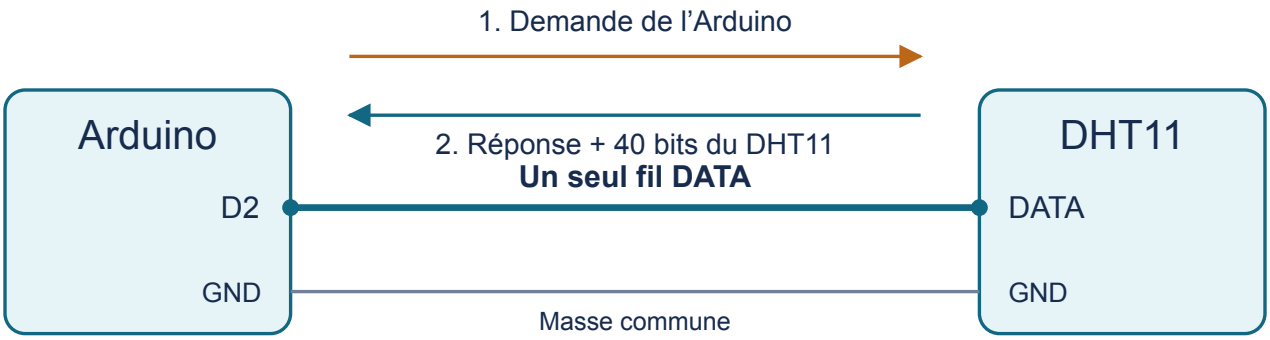
Echange de données entre Arduino et DHT11



La ligne **DATA** est maintenue au niveau haut par une résistance de rappel, généralement déjà présente sur le module.
La communication se déroule ainsi :

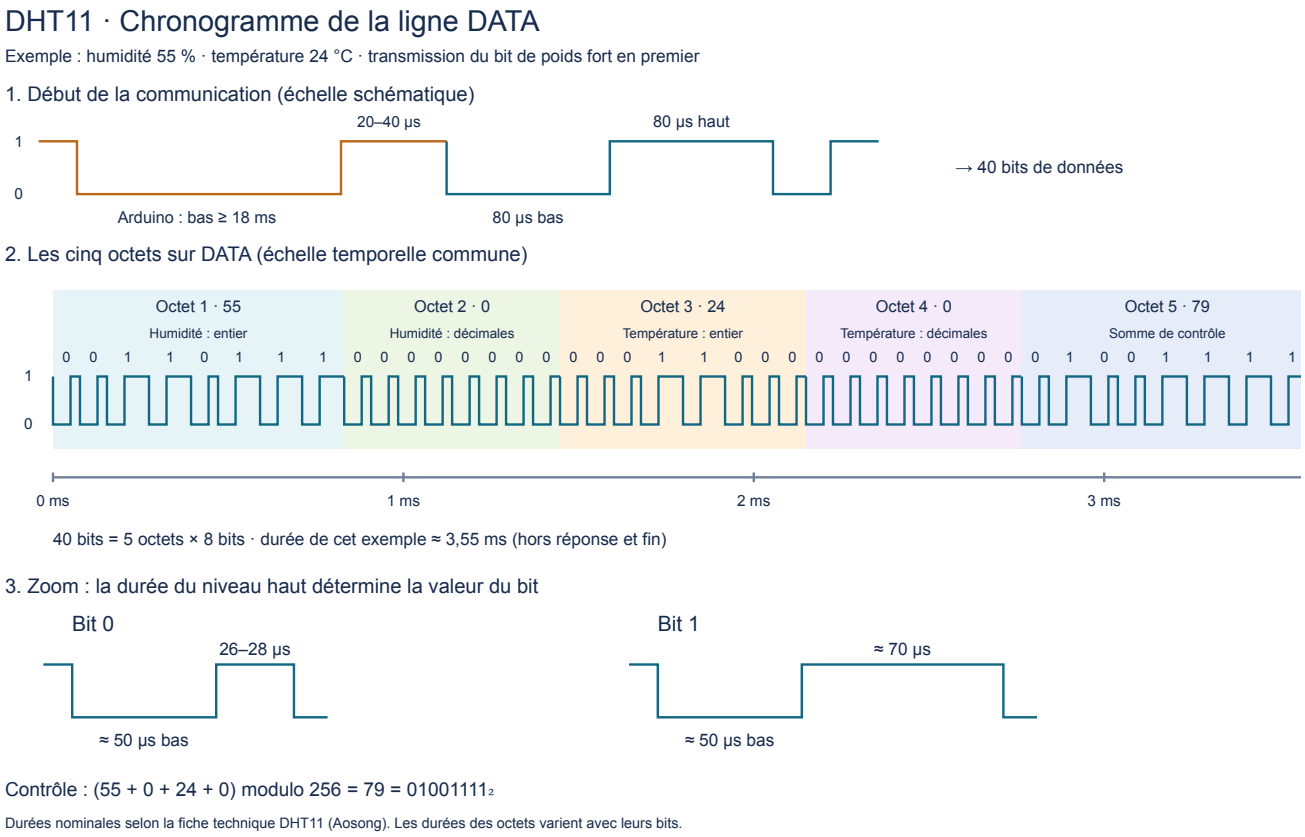
1. L'Arduino demande une mesure en maintenant DATA au niveau bas pendant au moins 18 ms, puis libère la ligne.
2. Le DHT11 répond par une impulsion basse d'environ 80 µs, suivie d'une impulsion haute d'environ 80 µs.
3. Le DHT11 transmet 40 bits, regroupés en cinq octets :

Octet	Contenu
1	Partie entière de l’humidité relative (%)
2	Partie décimale de l’humidité
3	Partie entière de la température (°C)
4	Partie décimale de la température
5	Somme de contrôle pour vérifier les données



DATA au repos : niveau haut grâce à la résistance de rappel du module.

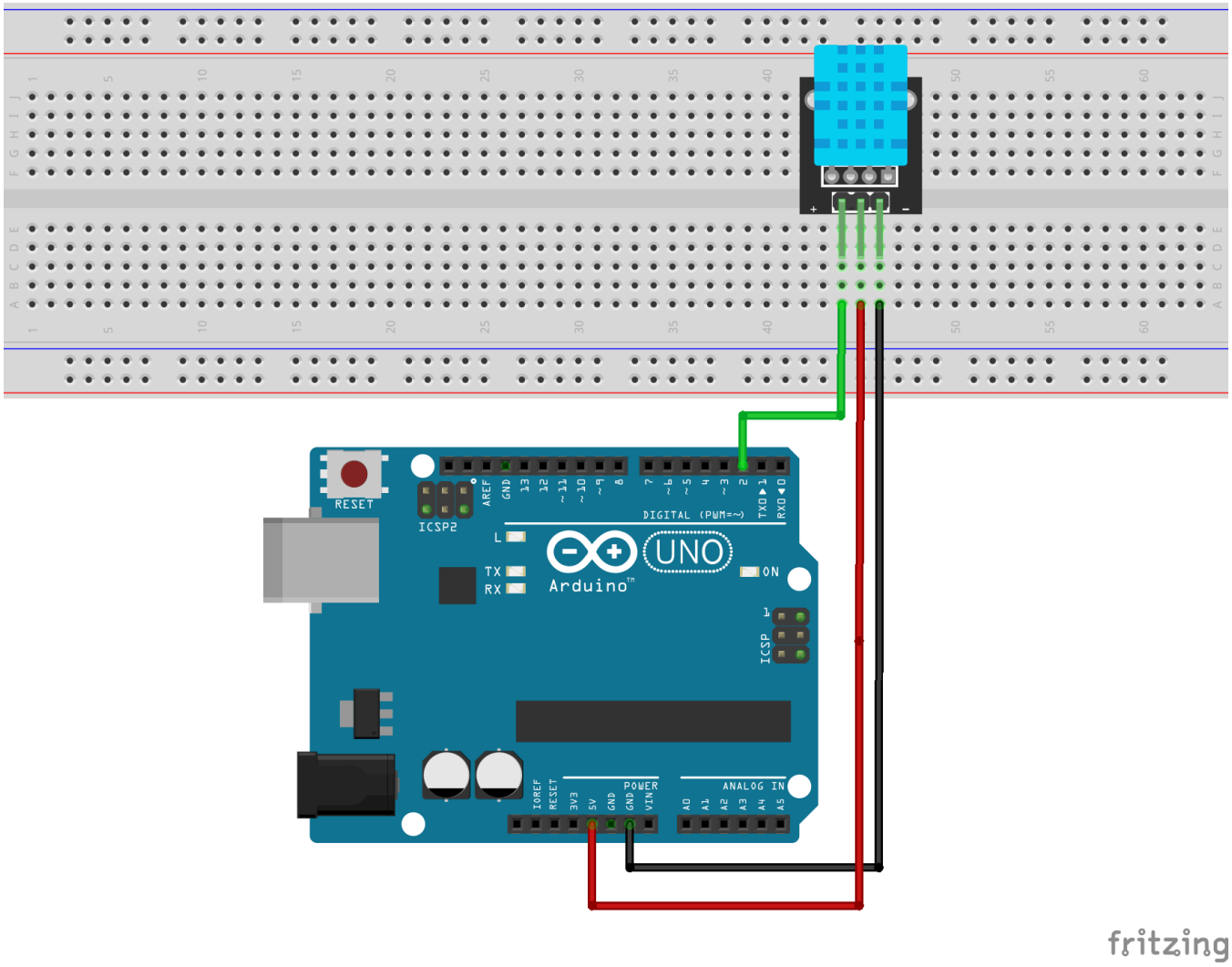
Chronogramme de la communication entre Arduino et le capteur



Chaque bit commence par environ 50 µs au niveau bas. La durée du niveau haut indique ensuite sa valeur :

- 26 à 28 µs : bit 0 ;
- environ 70 µs : bit 1.
  Il s'agit d'un protocole propriétaire à un fil. **Une bibliothèque Arduino DHT prend en charge ces temporisations et le décodage.**

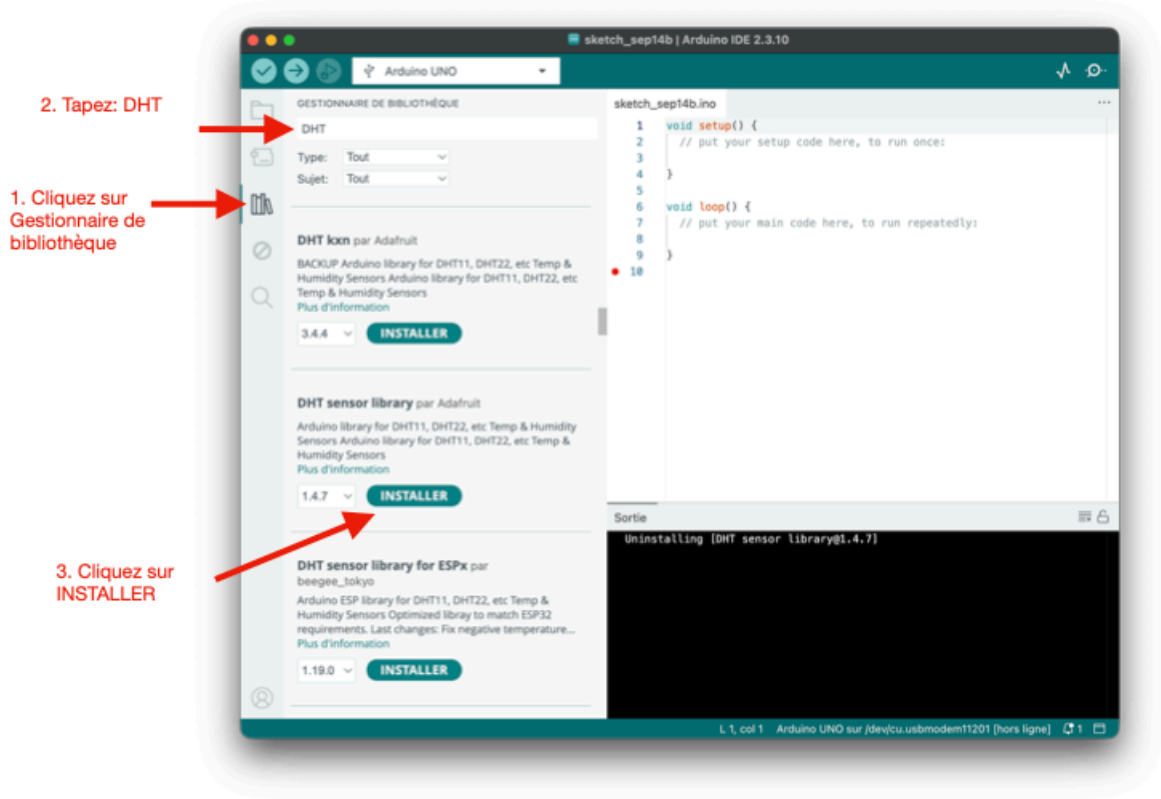
🔌 Schéma de câblage



> ⚠️ **Bien respecter les polarités**

📚 Installation de la bibliothèque (library)

Pour ce projet, il vous faut installer la **DHT sensor library** de **Adafruit**. Il faut installer les dépendances également.



🌡️ Le code complet

```
#include <DHT.h>
#define DHTPin 2
#define DHTType DHT11
DHT dht(DHTPin, DHTType);
float temp;
float hum;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  // put your main code here, to run repeatedly:

  temp = dht.readTemperature();
  hum = dht.readHumidity();

  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.print(" °C, ");
  Serial.print("Humidity: ");
  Serial.print(hum);
  Serial.println("%\t");

  delay(2000);
}
```

🌡️ Le code, pas à pas

```
#include <DHT.h>
```

- Le code a besoin de la bibliothèque DHT. La directive **include** demande au pré-

processeur d'ajouter le code de déclaration des fonctions de la bibliothèque.

```
#define DHTPin 2
```

- Déclaration de la pin utilisée pour le capteur

```
#define DHTType DHT11
DHT dht(DHTPin, DHTType);
```

- Déclaration du type de capteur DHT11 (la library peut gérer plusieurs types de capteurs)
- Déclaration de la variable objet dht de type DHT (classe). La fonction et le type sont définis par la library DHT écrite en C++.

```
float temp;
float hum;
```

- Déclaration des 2 variables pour la lecture de température et d'humidité
- type float est utilisé pour stocker les nombres à virgule (Plage de valeurs : De $-3{,}4028235 \times 10^{38}$ à $+3{,}4028235 \times 10^{38}$)

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  dht.begin();
}
```

- Fonction setup() exécutée une seule fois
- Configuration du moniteur série avec une vitesse de 9600 bauds
- Initialisation de la variable dht par la fonction membre (ou méthode) begin()

```
void loop() {
  // put your main code here, to run repeatedly:

  temp = dht.readTemperature();
  hum = dht.readHumidity();
```

- Lecture des valeurs de température et d'humidité par les méthodes readTemperature() et readHumidity()

🌡️ Le code complet

```
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.print(" °C, ");
  Serial.print("Humidity: ");
  Serial.print(hum);
  Serial.println("%\t");

  delay(2000);
}
```

- Affichage de la température et de l'humidité sur le moniteur série
- Serial.print() affiche un texte sans retour à la ligne
- Serial.println() affiche un texte avec retour à la ligne

🌡️ Le code complet

```
#include <DHT.h>
#define DHTPin 2
#define DHTType DHT11
DHT dht(DHTPin, DHTType);
float temp;
float hum;

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

void loop() {
  temp = dht.readTemperature();
  hum = dht.readHumidity();

  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.print(" °C, ");
  Serial.print("Humidity: ");
  Serial.print(hum);
  Serial.println("%\t");

  delay(2000);
}
```