SESSION 06 · DÉBUTANT 🟢 - Affichage de la température et de l'humidité sur afficheur LCD

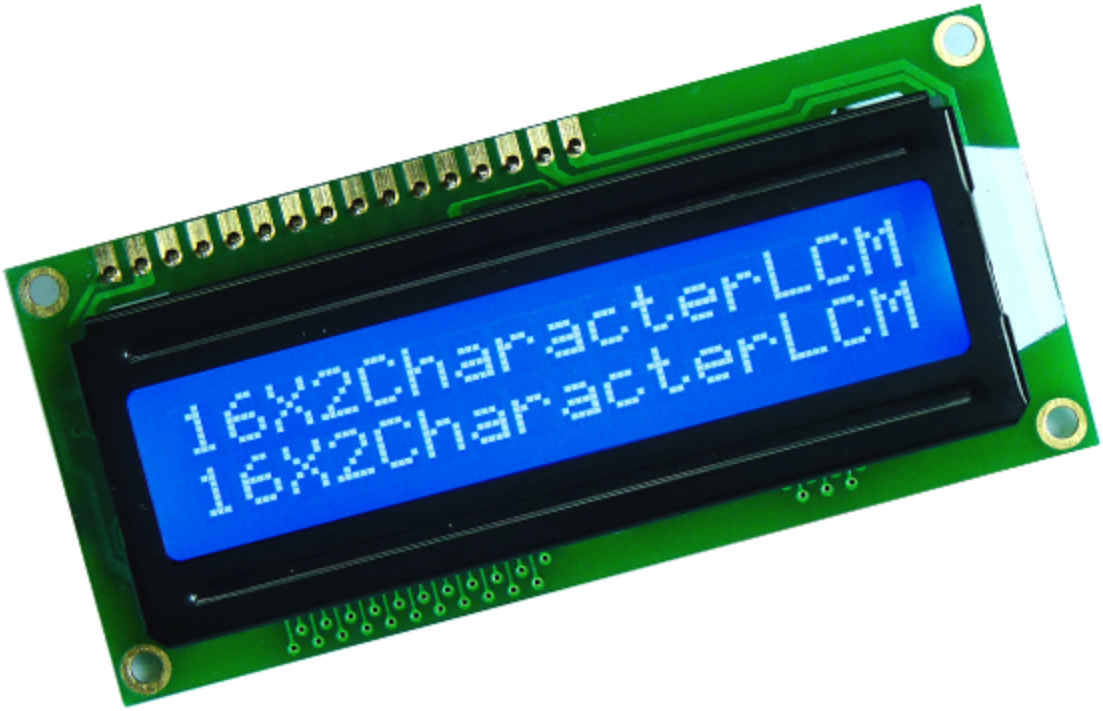
🎯 Objectifs

- Apprendre à utiliser un afficheur LCD
- Combiner la lecture de la température et humidité du capteur DHT11 et son affichage sur LCD
- Rendre le projet autonome (fonctionnement sur pile)

🧰 Matériel

Composant	Qté	Note
Arduino Uno + câble USB	1	
Breadboard	1	
Module capteur de température et d'humidité DHT11	1	
Ecran LCD1602	1	
Potentiomètre 10k	1	
Câble adaptateur alimentation pile 9V	1	
Pile 9V	1	
Câbles Dupont	19	

🧰 Le module afficheur LCD1602



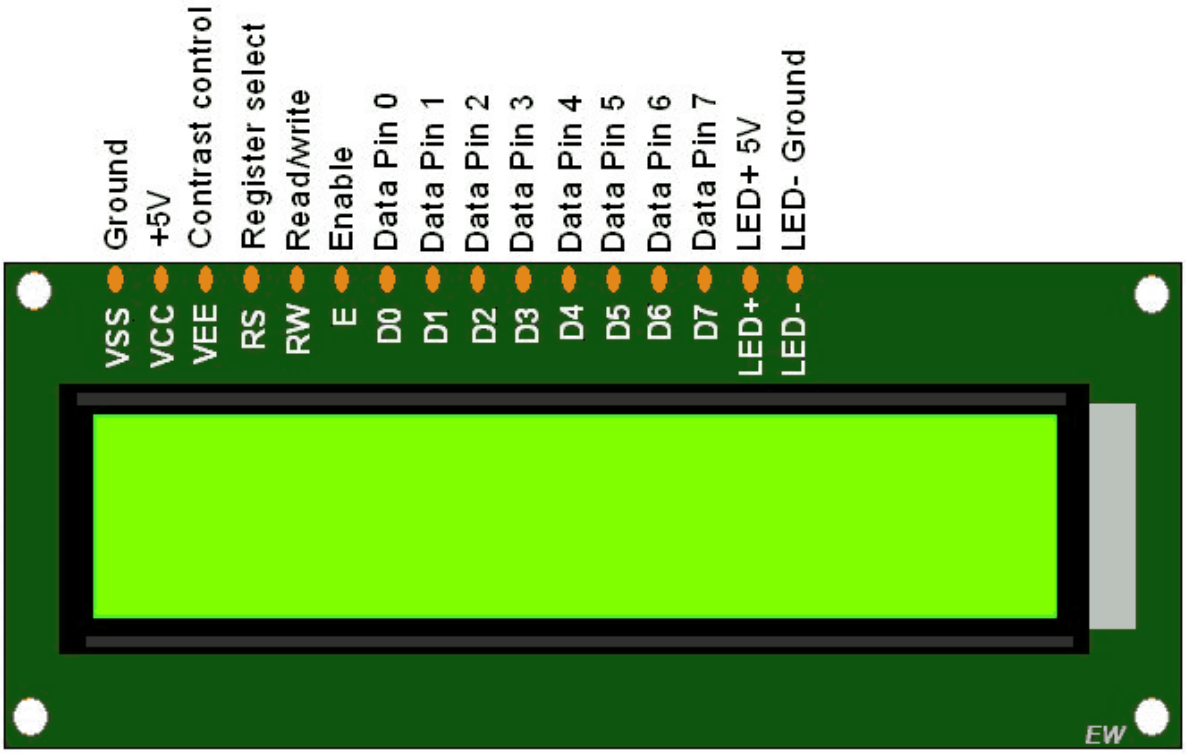
Les afficheurs LCD existent sous toutes tailles, de couleurs différentes et pouvant afficher plusieurs lignes et plusieurs caractères par lignes.

Le LCD 1602 utilise le contrôleur HD44780. 1602 signifie que l'afficheur va pouvoir afficher **16 caractères sur chacune des 2 lignes**.

Le contrôleur HD44780 qui possède un bus de données de 8 bits. On peut l'utiliser:

- **soit en mode 4 bits, et il utilisera 7 broches de l'Arduino**
- soit en mode 8 bits, et il utilisera 11 broches de l'Arduino

sans oublier les 3 pins pour le contrôle , l'alimentation et la luminosité.



❓ Comment ça marche ?

Encodage des caractères

L'**ASCII** (American Standard Code for Information Interchange - Code américan normalisé pour l'échange d'information) est une norme informatique d'encodage des caractères sur 8 bits (1 octet). C'est la plus répandue à ce jour. Plus récemment, de nouvelles normes étendant l'ASCII ont vu le jour (régionales (ISO/CEI 8859), nationales (GB 18030) ou internationales (Unicode)).

Les premières lettres de la table ASCII

À chaque caractère correspond un code numérique.

Lettre	Décimal	Binaire (8 bits)
A	65	0100 0001
B	66	0100 0010
C	67	0100 0011
D	68	0100 0100
E	69	0100 0101
F	70	0100 0110
G	71	0100 0111
H	72	0100 1000
I	73	0100 1001
J	74	0100 1010

Exemple : « A » → 65 → 0100 0001

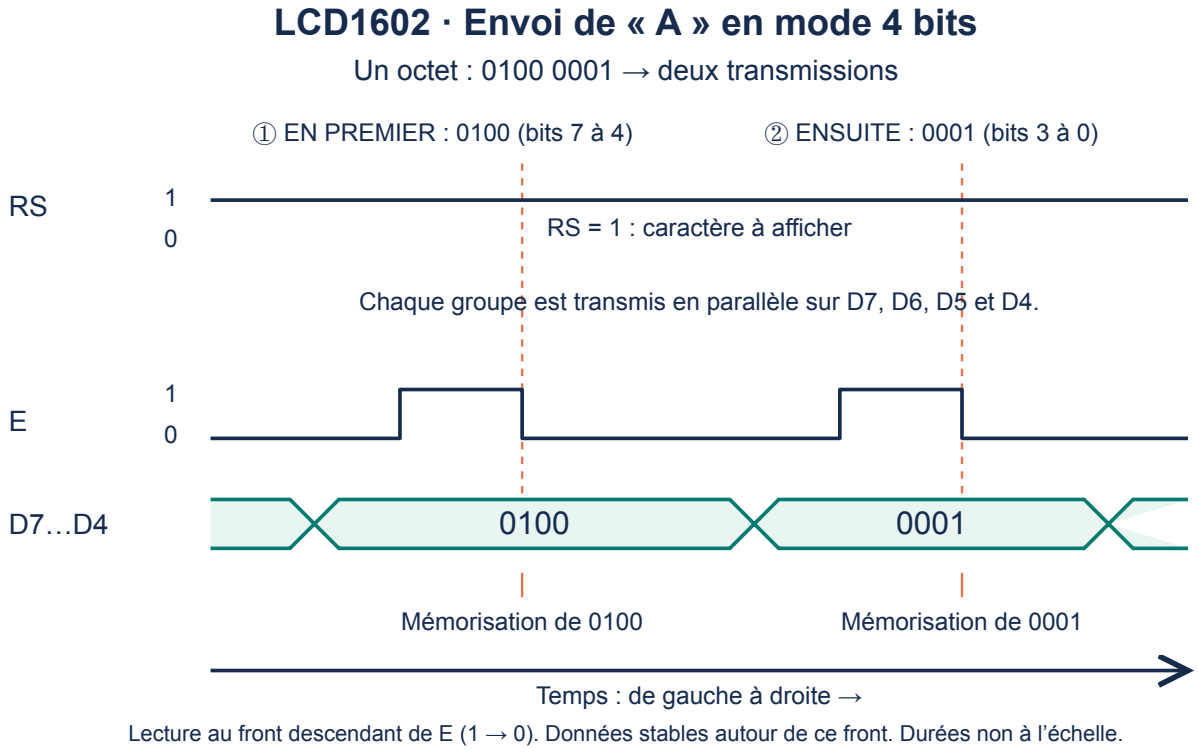
LCD en mode 4 bits : envoi de 0100, puis de 0001.

Les minuscules ont d'autres codes : « a » = 97.

Référence complète ASCII: https://www.ascii-code.com/fr

Envoi du caractère A par l'Arduino

L'Arduino transfère le code du caractère en 2 fois sur les 4 lignes de connexion D4 à D7 (mode 4 bits): poids fort d'abord, puis poids faible.



🧰 Le Potentiomètre

Un **potentiomètre** (souvent abrégé "pot") est une **résistance variable** à trois bornes. En tournant un bouton ou en déplaçant un curseur, on modifie la résistance électrique, et donc la tension de sortie du composant.

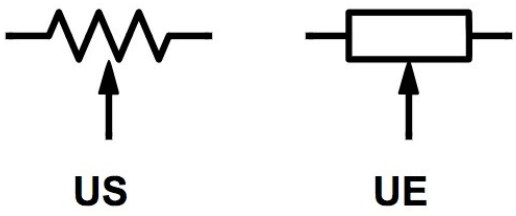


C'est le principe derrière le bouton de volume de votre chaîne hi-fi, le réglage de la luminosité d'une lampe, ou encore la manette de jeu vidéo.

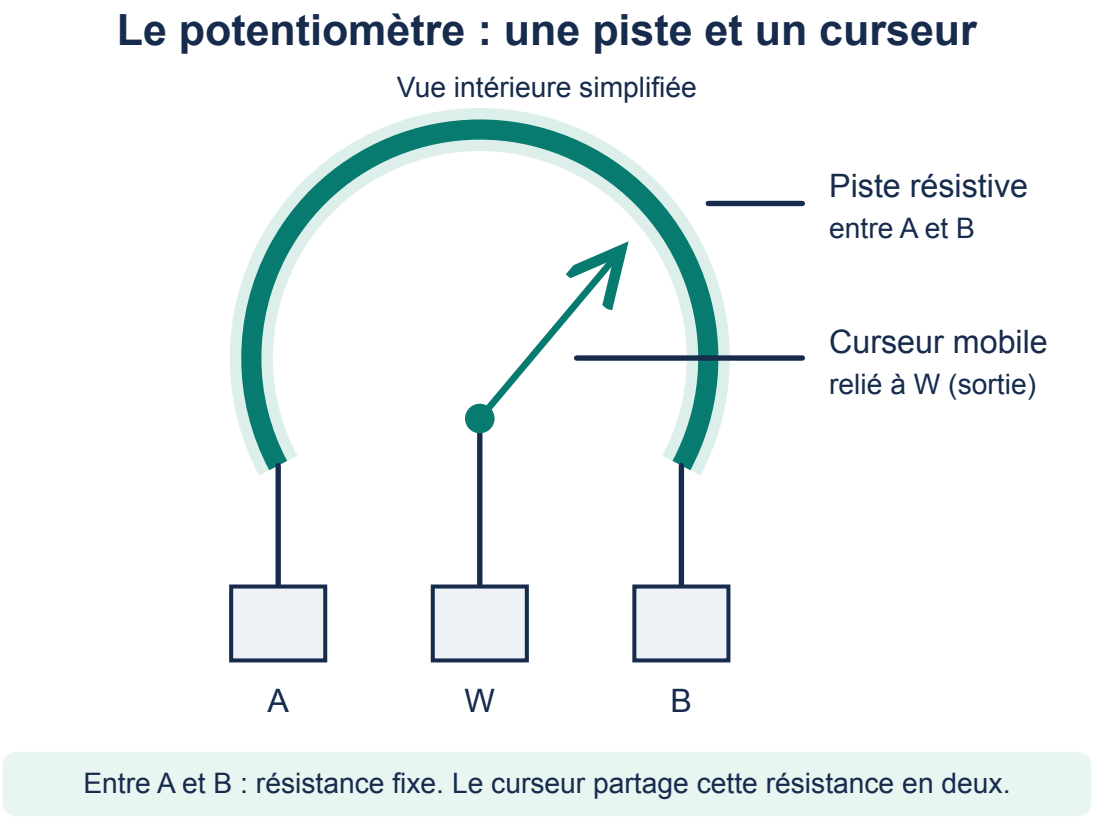
Différents types de potentiomètres



Symbole du potentiomètre



🔩 Anatomie d'un potentiomètre



Entre A et B : résistance fixe. Le curseur partage cette résistance en deux.

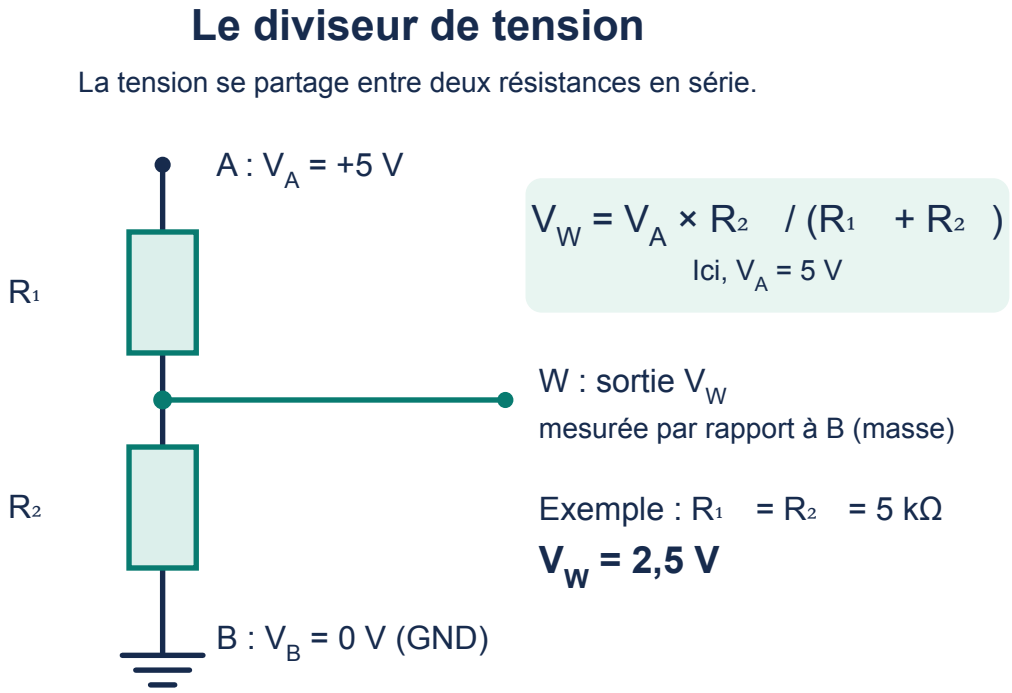
Un potentiomètre est composé de trois éléments principaux :

Élément	Rôle
Piste résistive	Arc de matériau résistif (carbone, cermet…) entre les bornes A et B
Curseur (wiper)	Contact mobile qui glisse sur la piste et prélève la tension intermédiaire
3 bornes	Borne A (extrémité), Borne B (extrémité), Borne W/2 (curseur = sortie)

> **Note**
> La résistance totale entre A et B est **fixe**. C'est la position du curseur W qui détermine la part de résistance de chaque côté.

⚡ Comment ça fonctionne ? Le diviseur de tension

Le potentiomètre agit comme un **diviseur de tension** : il partage la tension d'entrée en deux selon la position du curseur.



> 📍 **Calcul de la tension de sortie**
> Le courant prélevé doit être négligeable: c'est le même courant qui parcourt le potentiomètre du point A vers le point B.

Application de la loi d'Ohm:

$$V_A - V_B = V_A = (R_1 + R_2) \times I$$

$$V_W - V_B = V_W = R_2 \times I$$

En divisant la 2ième équation par la 1ère:

$$\frac{V_W}{V_A} = \frac{R_2}{R_1 + R_2}$$

et finalement:

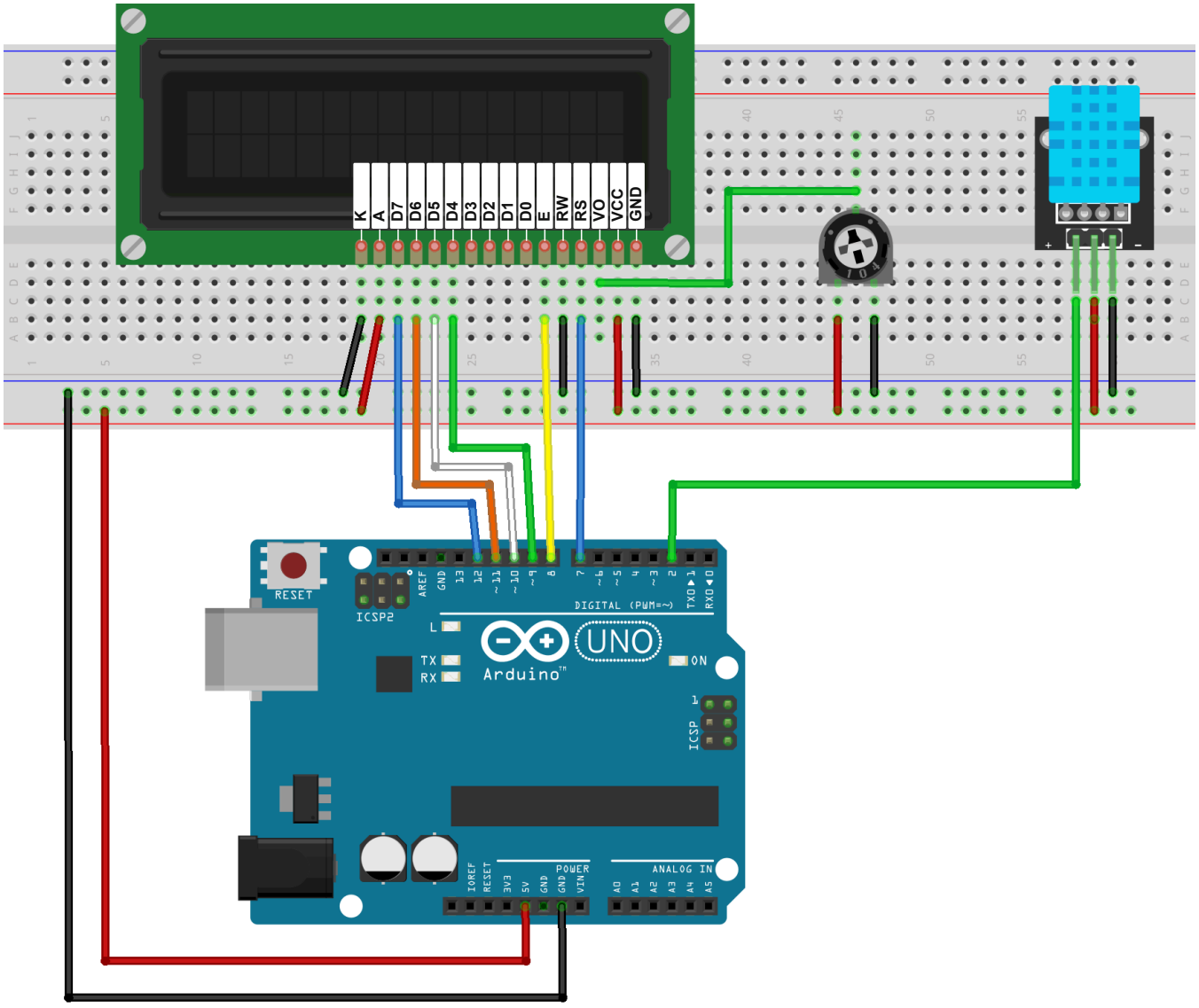
$$V_W = V_A \times \frac{R_2}{R_1 + R_2}$$

varie entre 0 V et +5 V

🔌 Montage final avec l'afficheur

Schéma de câblage

> ⚠️ **Attention au sens de la plaque d'essai (ligne rouge en bas) !**

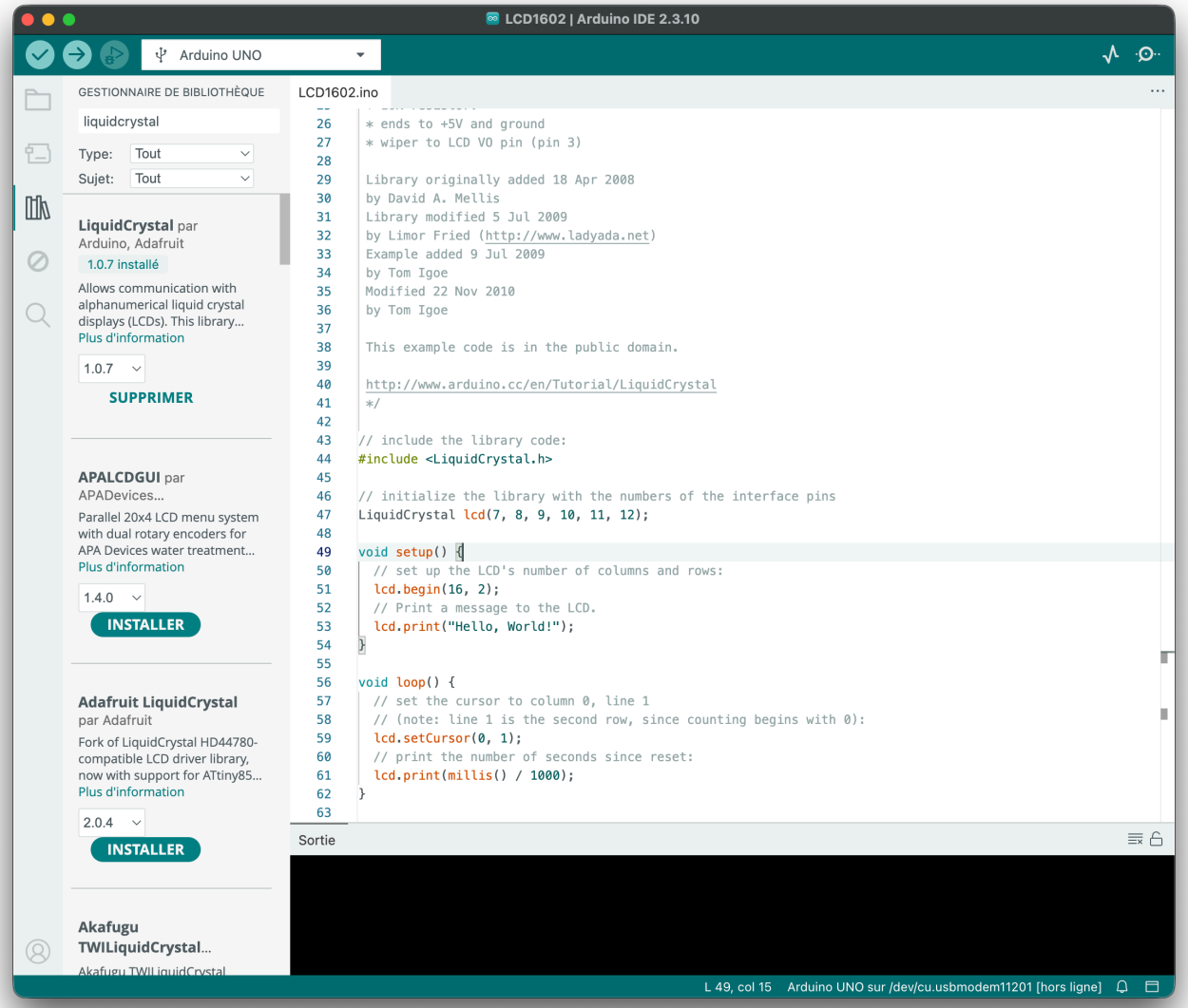


Câblage du LCD1602

Pins LCD	Noms	Roles	Branchements	Couleur fil
1	VSS	Gnd - Masse	GND	Noir
2	VCC	+5 volts	+5V	Rouge
3	VEE/ VO	Réglage du contraste	Pot. 10K	Vert
4	RS	Sélection du registre (commande ou données)	D7	Bleu
5	R/W	Lecture ou Ecritre (du registre)	GND	Noir
6	E	Entrée de validation	D8	Jaune
7	DB0	Bit de données 0 (data)	Non connecté	
8	DB1	Bit de données 1 (data)	Non connecté	
9	DB2	Bit de données 2 (data)	Non connecté	
10	DB3	Bit de données 3 (data)	Non connecté	
11	DB4	Bit de données 4 (data)	D9	Vert
12	DB5	Bit de données 5 (data)	D10	Blanc
13	DB6	Bit de données 6 (data)	D11	Orange
14	DB7	Bit de données 7 (data)	D12	Bleu
15	A	Anode du rétro-éclairage (+5v)	+5V	Rouge
16	K	Cathode du rétro-éclairage (Gnd - Masse)	GND	Noir

📚 Installation de la bibliothèque (library)

Pour ce projet, il vous faut installer la **LiquidCrystal** de **Adafruit**.



🩻 Le code pour tester le câblage du LCD1602

```
// Inclure la librairie LiquidCrystal
#include <LiquidCrystal.h>

// Définir les broches LCD
#define RSPin 7        // Sélection du registre (commande ou données)
#define EPin 8         // Entrée de validation
#define DB4Pin 9       // Bit de données 4 (data)
#define DB5Pin 10      // Bit de données 5 (data)
#define DB6Pin 11      // Bit de données 6 (data)
#define DB7Pin 12      // Bit de données 7 (data)


// Initialiser (créer un objet lcd) avec les pins connectées
LiquidCrystal lcd(RSPin, EPin, DB4Pin, DB5Pin, DB6Pin, DB7Pin);

void setup() {
  // Définir le nombre de colonnes et de lignes
  lcd.begin(16, 2);
  // Mettre le curseur en colonne 0 et ligne 0 (le comptage commence à 0)
  lcd.setCursor(0, 0);
  // Afficher un message
  lcd.print("Hello, World!");
}

void loop() {
  // Mettre le curseur en colonne 0 et ligne 1 (le comptage commence à 0)
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

🌡️ Le code complet: affichage du capteur sur LCD

```
// ------------------------ AFFICHEUR LCD ------------------------
//
// Inclure la librairie LiquidCrystal
#include <LiquidCrystal.h>

// Définir les broches LCD
#define RSPin 7        // Sélection du registre (commande ou données)
#define EPin 8         // Entrée de validation
#define DB4Pin 9       // Bit de données 4 (data)
#define DB5Pin 10      // Bit de données 5 (data)
#define DB6Pin 11      // Bit de données 6 (data)
#define DB7Pin 12      // Bit de données 7 (data)

// Initialiser (créer un objet lcd) avec les pins connectées
LiquidCrystal lcd(RSPin, EPin, DB4Pin, DB5Pin, DB6Pin, DB7Pin);

// ------------------------ CAPTEUR T/H ------------------------
//
// inclure la librairie DHT
#include <DHT.h>

#define DHTPin 2
#define DHTType DHT11

// Initialiser objet dht
DHT dht(DHTPin, DHTType);
// Variables globales
int temp;
int hum;


void setup() {
  // Démarrage du LCD, définir le nombre de colonnes et de lignes
  lcd.begin(16, 2);
  // Démarrage du senseur T/H
  dht.begin();
}

void loop() {
```

```
  // Effacement du lcd
  lcd.clear();

  // Lecture de la température et de l'humidité
  temp = int(dht.readTemperature());
  hum = int(dht.readHumidity());

  // Mettre le curseur en colonne 0 et ligne 0 (le comptage commence à 0)
  lcd.setCursor(0, 0);
  // Afficher la température
  lcd.print("Temp. : " + String(temp) + "C");

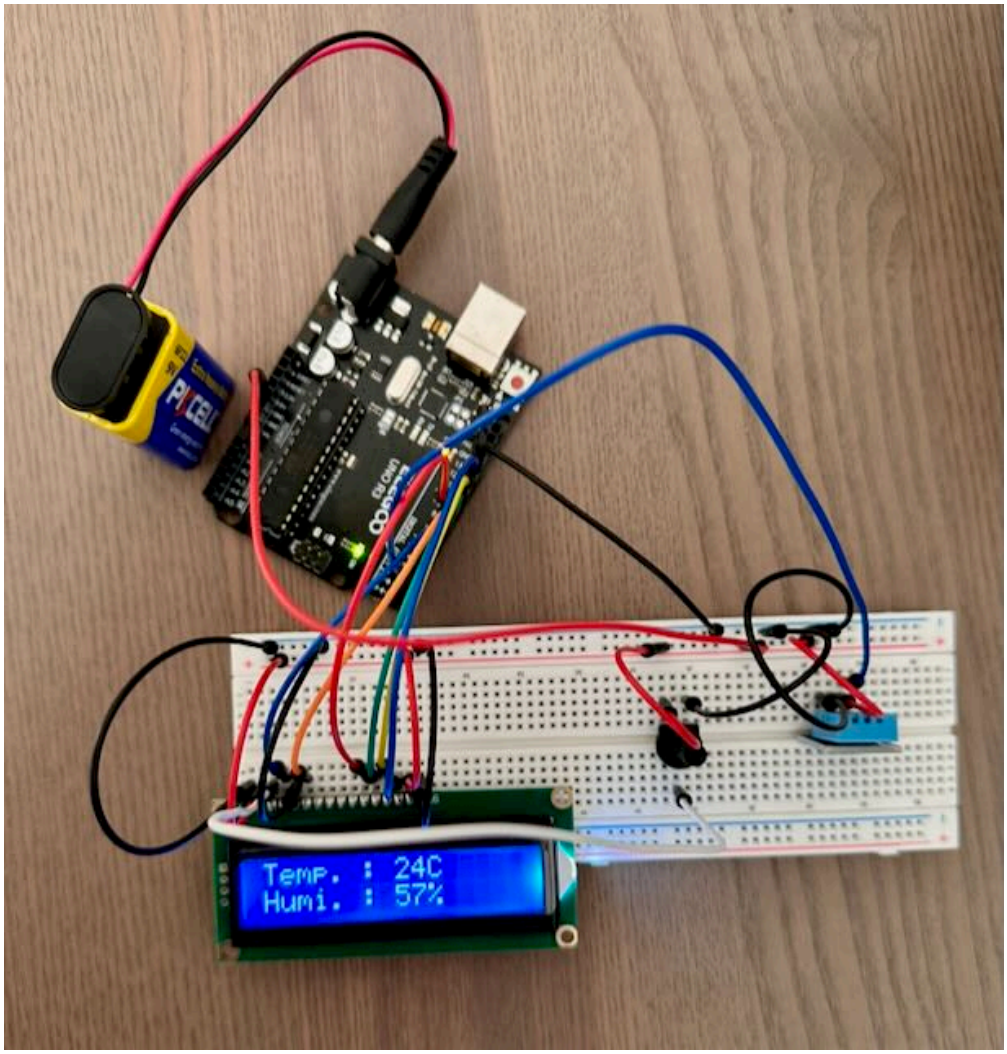
  // Mettre le curseur en colonne 0 et ligne 1
  lcd.setCursor(0, 1);
  // Afficher le pourcentage d'humidité
  lcd.print("Humi. : " + String(hum) + "%");

  // Attendre 2 s
  delay(2000);

}
```

🔌 Montage 2 - Le système devient autonome

- ⚠️ **Débranchez le câble USB**
- Connectez le câble adapteur et la pile 9V sur le connecteur d'alimentation externe


Temp. : 24C
Humi. : 57%