ARDUINO

Présentation

Arduino est une plateforme matérielle et logicielle open-source destinée à la création de projets électroniques interactifs. Les cartes Arduino permettent de piloter des capteurs, des actionneurs (LED, moteurs, relais) et de communiquer avec d'autres appareils.

Elle se compose principalement de :

- cartes électroniques équipées d'un microcontrôleur
- broches d'entrées/sorties Numériques 0 à 13 et Analogique A0 à A5
- un port USB pour la programmation et l'alimentation

Les sorties Arduino délivrent généralement 5V.
Le courant circule du + (5V ou 3,3V) vers le – (GND), toujours dans le même sens : il part de la borne positive (+) de l'alimentation, traverse les composants, puis revient vers la borne négative (masse/GND).
Si la masse n'est pas reliée, le circuit ne fonctionne pas, car le courant ne peut pas « boucler ».

Langage de programation

La programmation Arduino s'effectue dans l'environnement de développement intégré (IDE), avec un langage basé sur C/C++

Définitions importantes

Tension U : différence de potentiel électrique, exprimée en volts **V**
Intensité I : courant électrique, exprimé en ampères **A**
Résistance R : opposition au passage du courant, exprimée en ohms **Ω**
Il est essentiel d'ajouter une résistance en série avec une LED pour limiter le courant et éviter de l'endommager.

Loi d'Ohm

La relation fondamentale entre ces grandeurs est donnée par la **loi d'Ohm**

$$U=R\times I$$

Pour calculer la **résistance nécessaire** : R=U/I
Pour calculer l'**intensité** : I=U/R

TUTORIEL BASES ARDUINO

Arduino IDE

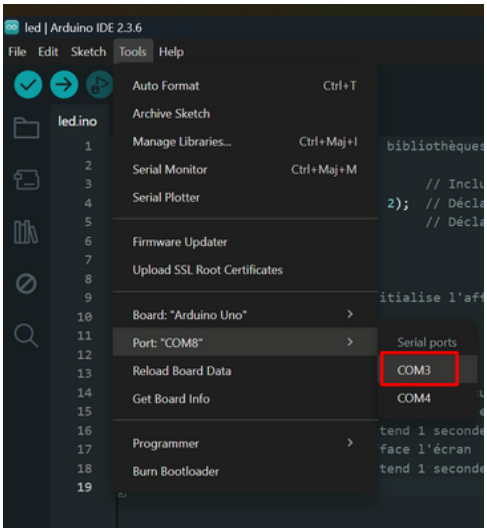
Integrated Development Environment est un logiciel open source qui sert à écrire, vérifier, compiler et transférer des programmes sur les cartes Arduino et d'autres microcontrôleurs compatibles.

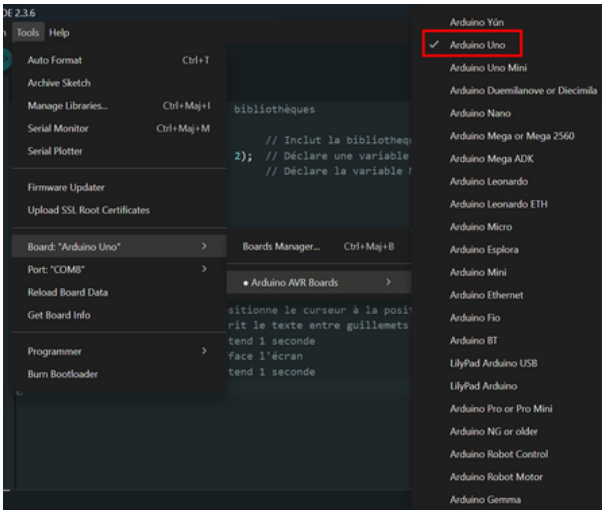
L'IDE permet :

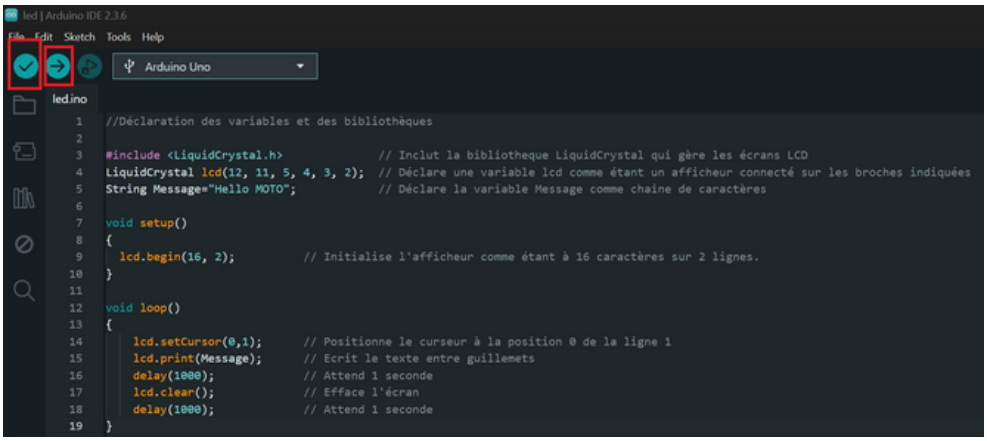
- d'écrire le code (appelé « sketch ») en langage Arduino (basé sur le C/C++),
- de vérifier et compiler ce code pour détecter les erreurs,
- de téléverser le programme directement sur la carte Arduino via un câble USB,
- d'échanger des données en temps réel avec la carte grâce au moniteur série, utile pour le débogage et l'affichage de résultats.

Il est très important de choisir dans les Tools :

- le modèle Arduino utilisé
- le port USB sur lequel l'Arduino est branché

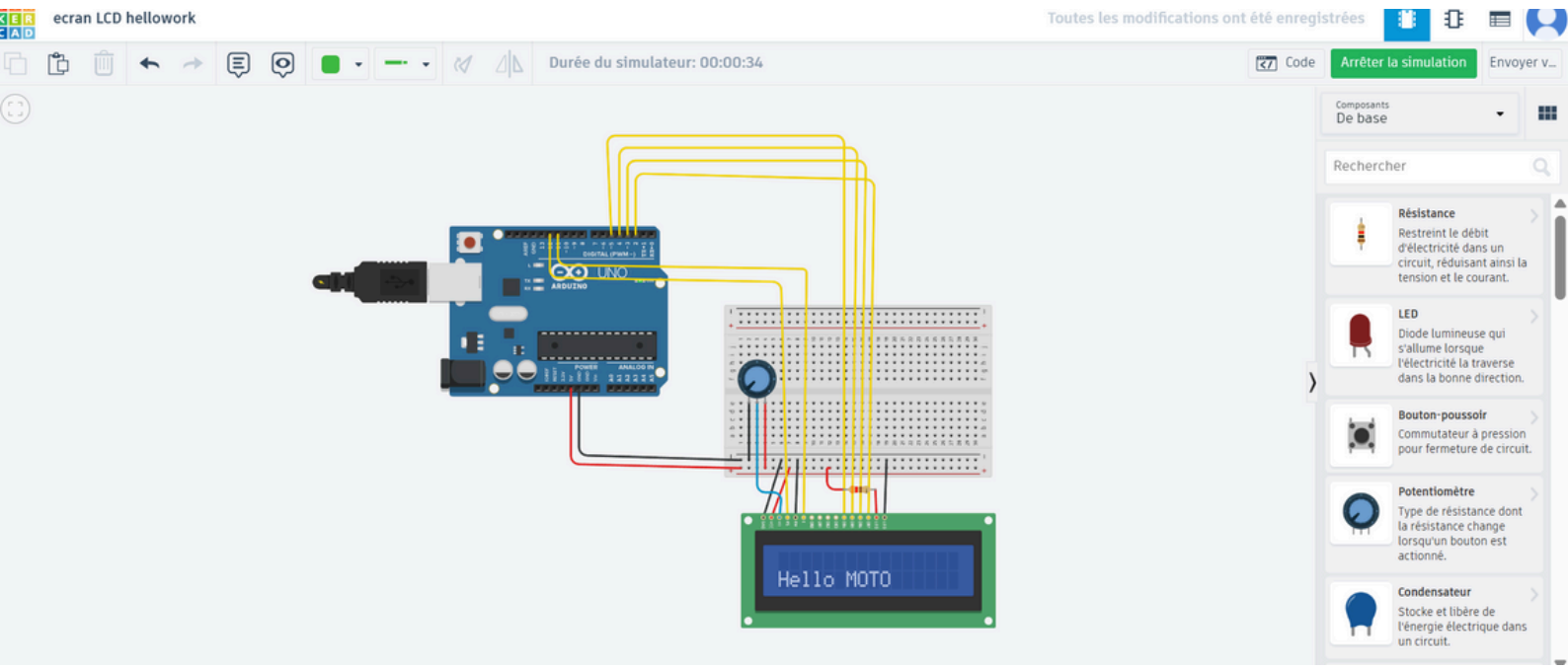






[Thinkercard]

Tinkercad sert à créer, simuler et tester des circuits électroniques Arduino virtuellement, permettant ainsi de programmer et de vérifier le fonctionnement de ses projets sans avoir besoin de matériel réel.
On peut ensuite programmer l'Arduino directement dans Tinkercad en écrivant du code, soit en mode texte (comme dans l'IDE Arduino), soit en utilisant la programmation par blocs, puis lancer la simulation pour tester le fonctionnement du circuit et du programme sans matériel réel

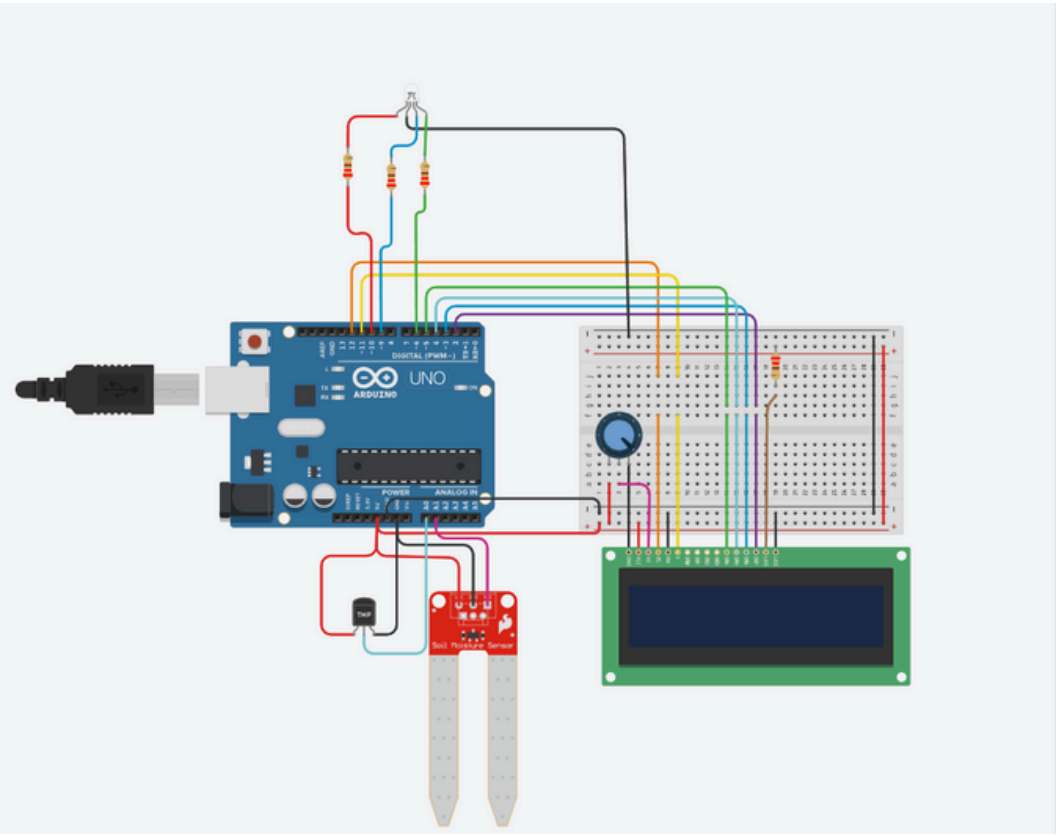


Projet : Station Météo

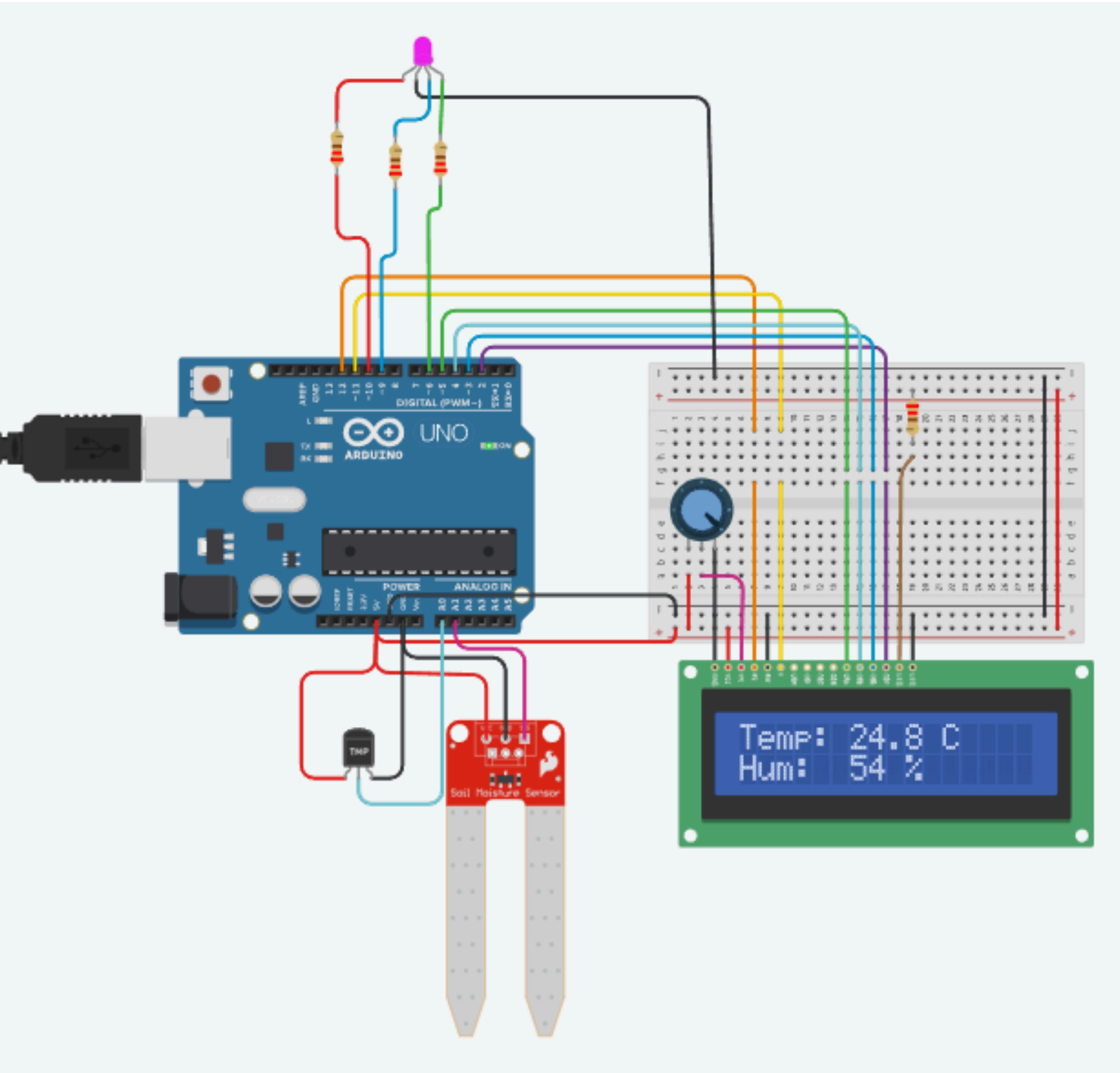
Une station météo est un ensemble de capteurs qui permet la mesure et l'enregistrement de la mesure physique et météorologique liée au changement du climat.
Le but de cet exercice est de créer une mini-centrale météo à l'aide du kit Arduino fourni qui affichera les différentes mesures sur une bande LED RGB et sur l'écran Oled ou sur l'écran LCD deux lignes.
Afficher les éléments suivant sur l'écran :

➔ La température de la pièce

➔ Le taux d'humidité de la pièce



```arduino
#include <LiquidCrystal.h>

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);

const int pinTMP36 = A0;
const int pinHumidite = A1;

// LED RGB
const int pinRouge = 10;
const int pinBleu = 9;
const int pinVert = 6;

void setup() {
  lcd_1.begin(16, 2);
  pinMode(pinRouge, OUTPUT);
  pinMode(pinVert, OUTPUT);
  pinMode(pinBleu, OUTPUT);
}

void setColor(int r, int g, int b) {
  analogWrite(pinRouge, 255 - r);
  analogWrite(pinVert, 255 - g);
  analogWrite(pinBleu, 255 - b);
}

void loop() {
  int valTMP = analogRead(pinTMP36);
  float tension = valTMP * (5.0 / 1023.0);
  float temperature = (tension - 0.5) * 100.0;

  int humiditeBrute = analogRead(pinHumidite);
  int humiditePourcent = map(humiditeBrute, 1023, 0, 100, 0);

  // LED RGB selon température
  if (temperature < 18) {
    setColor(0, 0, 255);
  } else if (temperature < 25) {
    setColor(0, 255, 0);
  } else {
    setColor(255, 0, 0);
  }

  // Affichage propre
  lcd_1.setCursor(0, 0);
  lcd_1.print("Temp: ");
  lcd_1.print(temperature, 1);
  lcd_1.print(" C   "); // efface les anciens caractères

  lcd_1.setCursor(0, 1);
  lcd_1.print("Hum:  ");
  lcd_1.print(humiditePourcent);
  lcd_1.print(" %   "); // idem ici

  delay(1000);
}
```


UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
TMP
Soil Moisture Sensor
Temp: 24.8 C
Hum: 54 %

1. Inclusion des bibliothèques et déclarations

```
#include <LiquidCrystal.h>
LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);
```

On importe la bibliothèque pour contrôler l'écran LCD et on définit les broches de connexion.
Permettre l'affichage des données sur un écran 16x2 caractères. Explication : Les nombres (12, 11, 5, 4, 3, 2) correspondent aux broches Arduino connectées aux broches RS, Enable, D4, D5, D6, D7 de l'écran LCD.

2. Déclaration des broches des capteurs

```
const int pinTMP36 = A0;
const int pinHumidite = A1;
```

On définit les broches analogiques pour lire les capteurs.
Le TMP36 (capteur de température) est sur A0, le capteur d'humidité sur A1
Explication : Ces broches analogiques peuvent lire des valeurs entre 0 et 1023, correspondant à des tensions de 0 à 5V.

3. LED RGB

```
const int pinRouge = 10;
const int pinBleu = 9;
const int pinVert = 6;
```

Définition des broches PWM pour contrôler l'intensité de chaque couleur.
Créer un indicateur visuel coloré selon la température.
Ces broches permettent de moduler l'intensité (0-255) de chaque couleur primaire.

4. Fonction setup()

```
void setup() {
  lcd_1.begin(16, 2);
  pinMode(pinRouge, OUTPUT);
  pinMode(pinVert, OUTPUT);
  pinMode(pinBleu, OUTPUT);
}
```

Configuration initiale qui s'exécute une seule fois au démarrage.
Initialiser l'écran LCD (16 colonnes, 2 lignes) et configurer les broches LED en sortie.

5. Fonction setColor()

```
void setColor(int r, int g, int b) {
  analogWrite(pinRouge, 255 - r);
  analogWrite(pinVert, 255 - g);
  analogWrite(pinBleu, 255 - b);
}
```

Fonction personnalisée pour définir la couleur de la LED RGB.
Simplifier le contrôle des couleurs en utilisant des valeurs 0-255. Explication : Le "255 -" inverse la logique car cette LED RGB fonctionne en logique inverse (0 = allumé, 255 = éteint).

6. Lecture et calcul de la température

```
int valTMP = analogRead(pinTMP36);
float tension = valTMP * (5.0 / 1023.0);
float temperature = (tension - 0.5) * 100.0;
```

Conversion de la valeur analogique en température réelle. analogRead() lit une valeur 0-1023
Conversion en tension : valeur × (5V / 1023)
Conversion TMP36 : (tension - 0.5V) × 100 = température en °C Explication : Le TMP36 donne 0.5V à 0°C et augmente de 10mV par degré.

7. Lecture de l'humidité

```
int humiditeBrute = analogRead(pinHumidite);
int humiditePourcent = map(humiditeBrute, 1023, 0, 100, 0);
```

Conversion de la valeur analogique en pourcentage d'humidité.
La fonction map() convertit la plage 1023-0 vers 100-0%.
1023 (sec) devient 100%, 0 (humide) devient 0%.

8. Indicateur coloré selon température

```
if (temperature < 18) {
    setColor(0, 0, 255);     // Bleu = froid
} else if (temperature < 25) {
    setColor(0, 255, 0);     // Vert = tempéré
} else {
    setColor(255, 0, 0);     // Rouge = chaud
}
```

Affichage visuel par couleurs selon des seuils de température.
Indication intuitive : bleu (froid), vert (confortable), rouge (chaud).

9. Affichage sur LCD

```
lcd_1.setCursor(0, 0);
lcd_1.print("Temp: ");
lcd_1.print(temperature, 1);
lcd_1.print(" C   ");
```

Positionnement du curseur et affichage formaté.
setCursor(0, 0) : première ligne, première colonne
temperature, 1 : affiche avec 1 décimale
" C " : ajoute l'unité et efface les anciens caractères