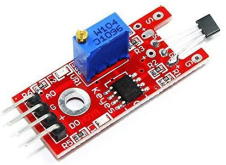
TP 2 : Les Capteur et les Actionneurs

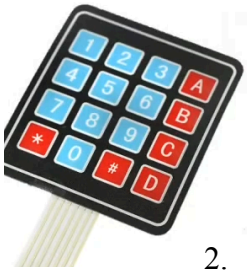
Dans le domaine de l'**IoT** et des systèmes embarqués, les **capteurs** et les **actionneurs** jouent un rôle fondamental dans la conception de **systèmes intelligents**. Ils permettent aux microcontrôleurs et aux systèmes informatiques d'interagir avec leur environnement en collectant des données et en effectuant des actions en conséquence.
Ce TP présente des applications pour différents capteurs et actionneurs. Nous utiliserons des simulateurs comme Tinkercad et Wokwi.

1. Capteurs :

1. **Capteurs numériques :**



   - **Capteur à effet Hall** : Détecte la présence d'un champ magnétique et peut fournir une sortie numérique ou analogique selon le modèle.



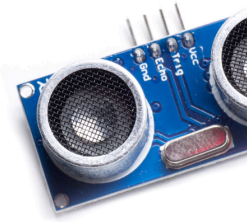
   - **Les claviers matriciels(TP2.2, TP2.3):** sont généralement des dispositifs numériques, chaque touche correspondant à une combinaison spécifique de lignes et de colonnes. Cependant, il existe des versions analogiques où chaque touche est associée à une résistance distincte, produisant une tension unique lorsqu'elle est pressée.

2. **Capteurs analogiques :**

   - **Potentiomètre** :
     - *Description* : Un diviseur de tension réglable qui fournit une sortie analogique en fonction de la position du curseur.

   - **Capteur de température TMP36** :
     - *Description* : Fournit une tension analogique proportionnelle à la température ambiante.

   - **Photoresistance (LDR)** :
     - *Description* : Sa résistance varie en fonction de l'intensité lumineuse, permettant de mesurer la luminosité ambiante.



   - Le **capteur ultrason** est un capteur de **distance**.Ce capteur comporte un émetteur et un récepteur d'onde, ce qui permet de calculer la distance d'un

objet en face de lui en comparant les temps d'émission et de réception de l'écho (en connaissant la vitesse de propagation du son : environ 340 m/s).

3. **Capteurs logiques :**

- **Bouton-poussoir (Pushbutton)** :Un simple interrupteur qui envoie un signal numérique haut ou bas lorsqu'il est pressé ou relâché.
- **Capteur de mouvement PIR (TP2.1)**:Détecte la présence de mouvement infrarouge dans son champ de vision et envoie un signal numérique lorsqu'un mouvement est détecté.

2. Actionneurs :

1. **Actionneurs numériques :**

- **Écran LCD (TP2.2, TP2.3)** :Affiche des informations textuelles ou graphiques en fonction des commandes reçues.
- **Servomoteur** :Moteur contrôlé par un signal PWM, permettant de positionner son axe à un angle précis.

2. **Actionneurs analogiques :**

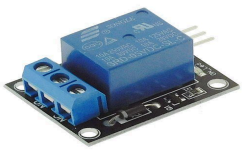
- **Moteur à courant continu (DC)** :Moteur qui tourne lorsqu'il est alimenté en courant continu, avec une vitesse proportionnelle à la tension appliquée.

- **Électrovannes proportionnelles** :Ces électrovannes permettent un contrôle précis du débit en ajustant l'ouverture de la vanne proportionnellement au signal de commande reçu.

3. **Actionneurs logiques :**

- **LED** :Diode électroluminescente qui s'allume lorsqu'un courant électrique la traverse.

- **Buzzer (TP2.1)** :Émet un son lorsqu'il est alimenté électriquement.



- **Relais** :Interrupteur électromécanique contrôlé par un signal électrique, permettant de contrôler des charges plus importantes.

Électrovannes tout ou rien (TOR): Dans un lave-linge, une électrovanne TOR contrôle l'entrée d'eau en s'ouvrant ou en se fermant complètement.

TP2.1: Détecteur de Mouvement avec Arduino

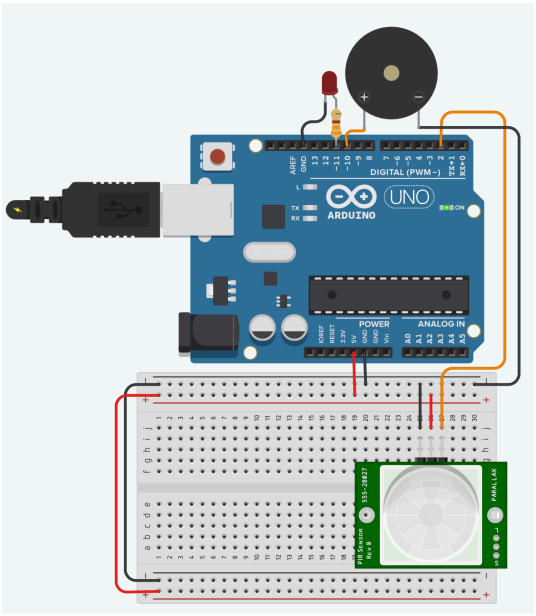
Objectif : Réaliser un détecteur de mouvement utilisant un capteur PIR (Passive Infrared Sensor), un buzzer et une LED avec une carte Arduino.

Capteur PIR (Passive Infrared Sensor):
Le capteur PIR est utilisé pour détecter les mouvements d'objets émettant une chaleur, comme le corps humain. Il fonctionne en mesurant les variations infrarouges dans son champ de vision. Lorsqu'un mouvement est détecté, il envoie un signal logique HAUT.

Matériel requis :

1. 1x Arduino Uno
2. 1x Capteur PIR (détecteur de mouvement)
3. 1x LED (de couleur au choix)
4. 1x Buzzer
5. 1x Résistance de 220Ω
6. 1x Mini breadboard
7. Câbles de connexion

1. Montage du circuit



- **Connexion du capteur PIR** :
  - VCC du PIR → 5V de l'Arduino
  - GND du PIR → GND de l'Arduino
  - Signal du PIR → Entrée numérique D2 de l'Arduino
- **Connexion de la LED** :
  - Anode (+) de la LED → Résistance de 220Ω → Broche numérique 11 de l'Arduino
  - Cathode (-) de la LED → GND de l'Arduino

- **Connexion du buzzer** :
  - Pôle positif (+) du buzzer → Broche numérique 10 de l'Arduino
  - Pôle négatif (-) du buzzer → GND de l'Arduino

2. Programmation de l'Arduino

Configuration du code dans Arduino IDE :

1. Déclarez les broches utilisées pour le capteur PIR, la LED et le buzzer.
2. Lisez la valeur du capteur PIR et affichez-la sur le moniteur série.
3. Si un mouvement est détecté (signal HAUT), activez la LED et le buzzer.
4. Si aucun mouvement n'est détecté, désactivez la LED et le buzzer.

Code :

```
int pirPin = 2;
int ledPin = 11;
int buzzerPin = 10;
int pirValue = 0;

void setup() {
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  pirValue = digitalRead(pirPin);
  Serial.println(pirValue);

  if (pirValue == HIGH) {
    digitalWrite(ledPin, HIGH);
    tone(buzzerPin, 1000, 200);
  } else {
    digitalWrite(ledPin, LOW);
    noTone(buzzerPin);
  }
  delay(500);
}
```

3. Test et validation

1. Cliquez sur **Démarrer la simulation**.
2. Ouvrez le **moniteur série** pour voir les valeurs envoyées par le capteur PIR.
3. Passez la souris sur le capteur PIR pour simuler un mouvement.

4. Vérifiez que lorsque le mouvement est détecté, la LED s'allume et le buzzer émet un son.
5. Si aucun mouvement n'est détecté, la LED et le buzzer restent éteints.

4. Conclusion

Ce TP vous permet de comprendre comment utiliser un **capteur PIR** avec **Arduino** et de tester votre montage sans matériel physique grâce à **Tinkercad**. Vous pouvez améliorer ce projet en ajoutant un affichage LCD ou une connexion sans fil pour envoyer des alertes.

TP2.2: Affichage LCD et Clavier Matriciel avec Arduino

Objectif :

Apprendre à utiliser un écran LCD et un clavier matriciel avec un Arduino en utilisant la plateforme de simulation **Tinkercad**.

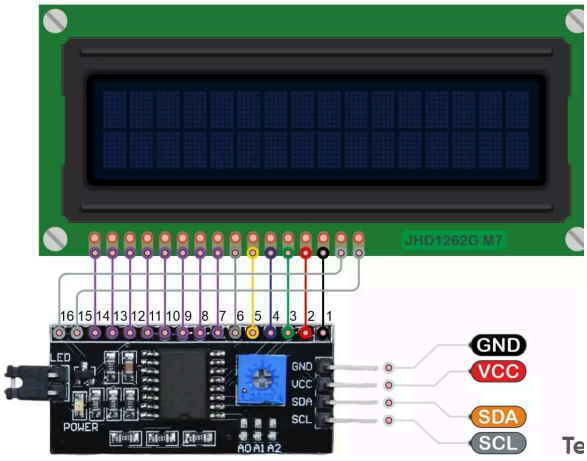
1. Concepts Clés

1.1. Écran LCD 16x2 avec I2C

Un **écran LCD (Liquid Crystal Display) 16x2** permet d'afficher des informations sur deux lignes de 16 caractères chacune. L'utilisation d'un **module I2C** permet de réduire le nombre de connexions en passant par un protocole de communication série simplifié.

Avantages de l'I2C :

- Moins de câbles (seulement 2 fils : SDA et SCL)
- Communication simplifiée avec l'Arduino
- Possibilité de connecter plusieurs périphériques sur le même bus I2C.



1.2. Clavier Matriciel 4x4

Un **clavier matriciel 4x4** est un dispositif d'entrée composé de 16 boutons organisés en **4 lignes et 4 colonnes**.

- Lorsqu'une touche est pressée, une connexion est établie entre une ligne et une colonne, ce qui permet de détecter quelle touche a été pressée.
- Un **clavier matriciel 4x4** nécessite **8 broches** sur l'Arduino, tandis qu'un **clavier 4x3** en utilise **7**.

2. Matériel requis dans Tinkercad

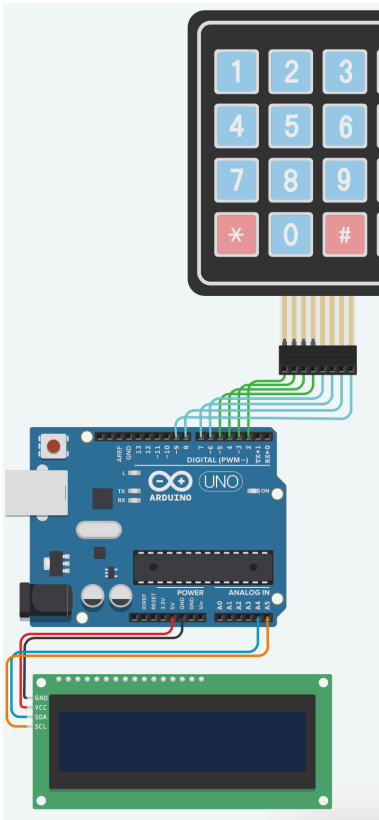
- 1x **Arduino Uno (virtuel)**
- 1x **Écran LCD 16x2 avec module I2C**
- 1x **Clavier matriciel 4x4**
- Câbles de connexion

3. Montage du circuit dans Tinkercad

- **Ouvrir Tinkercad** :

  Connectez-vous à Tinkercad.

  Créez un nouveau projet de circuit.



- **Ajout des composants** :

  Recherchez **Arduino Uno** et ajoutez-le à votre espace de travail.

  Recherchez **Keypad 4x4** et ajoutez-le.

  Ajoutez un **écran LCD 16x2 avec module I2C**.

- **Connexion du clavier matriciel** :

  Broches du clavier **L1, L2, L3, L4** → Broches numériques **2, 3, 4, 5** de l'Arduino

  Broches du clavier **C1, C2, C3, C4** → Broches numériques **6, 7, 8, 9** de l'Arduino

- **Connexion de l'écran LCD avec I2C** :

  VCC du LCD → 5V de l'Arduino

  GND du LCD → GND de l'Arduino

  SDA du LCD → Broche A4 de l'Arduino

SCL du LCD → Broche A5 de l'Arduino

4. Code :

```
#include <Wire.h> // For I2C communication
#include <Adafruit_LiquidCrystal.h> // For I2C LCD
#include <Keypad.h> // For the keypad

// Initialize the LCD with I2C address 0
Adafruit_LiquidCrystal lcd(0);

// Define the size of the keypad
const byte LIGNES = 4;
const byte COLONNES = 4;

// Define the keys on the keypad
char touches[LIGNES][COLONNES] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

// Define the pins used for the keypad
byte brochesLignes[LIGNES] = {2, 3, 4, 5};
byte brochesColonnes[COLONNES] = {6, 7, 8, 9};

// Initialize the keypad
Keypad clavier = Keypad(makeKeymap(touches), brochesLignes, brochesColonnes, LIGNES,
COLONNES);

void setup() {
  lcd.begin(16, 2); // Initialize the LCD with 16 columns and 2 rows
  lcd.setBacklight(HIGH); // Turn on the backlight
  lcd.setCursor(0, 0); // Position the cursor at the top-left corner
  lcd.print("Clavier pret"); // Display "Clavier pret" on the LCD
}

void loop() {
  char touche = clavier.getKey(); // Check if a key is pressed
  if (touche) { // If a key is pressed
    lcd.clear(); // Clear the LCD
    lcd.setCursor(0, 0); // Position the cursor at the top-left corner
    lcd.print("Touche : "); // Display "Touche : "
    lcd.setCursor(8, 0); // Move the cursor to the 9th position
    lcd.print(touche); // Display the pressed key
  }
  delay(200); // Short delay for debounce
}
```

5. Test et validation dans Tinkercad

1. Cliquez sur **Démarrer la simulation**.
2. Appuyez sur une touche du clavier matriciel.
3. Vérifiez que la touche pressée s'affiche sur l'écran LCD.
4. Répétez avec différentes touches pour vous assurer du bon fonctionnement.

Remarque:

- Les broches I2C sont fixés sur Arduino (SDA → A4, SCL → A5)
- Sur ESP32 on définit nous même les broches à utiliser : ex. `Wire.begin(21, 22);`

6. Conclusion

Ce TP vous permet de comprendre **comment interfacer un clavier matriciel avec un écran LCD via Arduino**. Grâce à Tinkercad, vous pouvez tester ce montage sans matériel physique. Vous pouvez améliorer ce projet en ajoutant un système de menu interactif ou en intégrant des actions spécifiques à chaque touche.

TP2.3: Réalisation d'une Calculatrice avec Arduino, Clavier Matriciel 4x4 et Écran LCD 16x2

Objectif : Concevoir une calculatrice capable d'effectuer des opérations arithmétiques de base (addition, soustraction, multiplication, division) en utilisant un clavier matriciel 4x4 pour l'entrée des données et un écran LCD 16x2 pour l'affichage des résultats.

Matériel requis :

- Arduino Uno
- Clavier matriciel 4x4
- Écran LCD 16x2 avec interface I2C
- Câbles de connexion
- Potentiomètre (optionnel, pour ajuster le contraste de l'écran LCD)

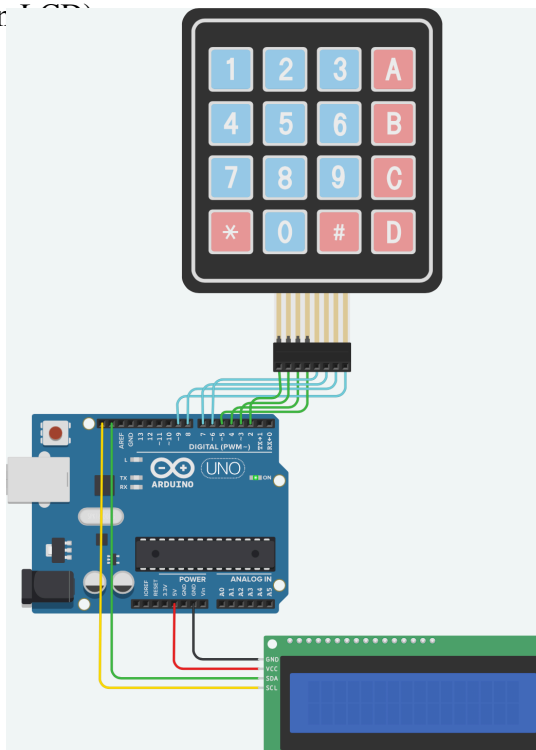


Schéma de câblage :

1. **Connexion du clavier matriciel 4x4 :**
   - Connectez les broches des lignes (R1 à R4) aux broches numériques 2, 3, 4 et 5 de l'Arduino.
   - Connectez les broches des colonnes (C1 à C4) aux broches numériques 6, 7, 8 et 9 de l'Arduino.
2. **Connexion de l'écran LCD 16x2 avec interface I2C :**
   - Connectez la broche VCC de l'écran LCD au 5V de l'Arduino.
   - Connectez la broche GND de l'écran LCD au GND de l'Arduino.

- Connectez la broche SDA de l'écran LCD à la broche A4 de l'Arduino.
- Connectez la broche SCL de l'écran LCD à la broche A5 de l'Arduino.

Programme Arduino : Le code suivant permet de capturer les entrées du clavier, de réaliser les calculs et d'afficher les résultats sur l'écran LCD.

- **Bibliothèques incluses :** Les bibliothèques Wire.h, Adafruit_LiquidCrystal.h et Keypad.h sont incluses pour gérer respectivement la communication I2C avec l'écran LCD et l'interaction avec le clavier matriciel.
- **Initialisation de l'écran LCD :** L'objet lcd est créé pour contrôler l'écran LCD via l'interface I2C.
- **Configuration du clavier :** Les dimensions du clavier sont définies, ainsi que les caractères associés à chaque

```ino
#include <Wire.h>
#include <Adafruit_LiquidCrystal.h>
#include <Keypad.h>

// Initialisation de l'écran LCD avec l'adresse I2C 0x27
Adafruit_LiquidCrystal lcd(0);

// Définition des dimensions du clavier
const byte LIGNES = 4;
const byte COLONNES = 4;

// Définition des touches du clavier matriciel
char touches[LIGNES][COLONNES] = {
  {'1', '2', '3', 'A'}, // 'A' pour addition
  {'4', '5', '6', 'B'}, // 'B' pour soustraction
  {'7', '8', '9', 'C'}, // 'C' pour multiplication
  {'*', '0', '#', 'D'}  // 'D' pour division, '*' pour effacer, '#' pour égal
};

// Définition des broches utilisées pour le clavier
byte brochesLignes[LIGNES] = {2, 3, 4, 5};
byte brochesColonnes[COLONNES] = {6, 7, 8, 9};

// Initialisation du clavier
Keypad clavier = Keypad(makeKeymap(touches), brochesLignes, brochesColonnes, LIGNES, COLONNES);

String nombre1 = "", nombre2 = "";
char operateur;
bool saisieDeuxiemeNombre = false;

void setup() {
  lcd.begin(16, 2); // Initialisation de l'écran LCD avec 16 colonnes et 2 lignes
  lcd.setBacklight(HIGH); // Activation du rétroéclairage
  lcd.setCursor(0, 0);
  lcd.print("Calculatrice Prete");
  delay(2000);
  lcd.clear();
}
```

Loop :

```
40  void loop() {
41    char touche = clavier.getKey();
42
43    if (touche) {
44      if (touche >= '0' && touche <= '9') {
45        if (!saisieDeuxiemeNombre) {
46          nombre1 += touche;
47          lcd.setCursor(0, 0);
48          lcd.print(nombre1);
49        } else {
50          nombre2 += touche;
51          lcd.setCursor(2, 0);
52          lcd.print(nombre2);
53        }
54      } else if (touche == 'A' || touche == 'B' || touche == 'C' || touche == 'D') {
55        if (nombre1 != "") {
56          saisieDeuxiemeNombre = true;
57          operateur = touche;
58          lcd.setCursor(1, 0);
59          lcd.print(getSymboleOperateur(operateur));
60        }
61      } else if (touche == '#') { // Touche '=' pour calculer
62        if (nombre1 != "" && nombre2 != "") {
63          float num1 = nombre1.toFloat();
64          float num2 = nombre2.toFloat();
65          float resultat = calculer(num1, num2, operateur);
66          lcd.setCursor(3, 0);
67          lcd.print(" = ");
68          lcd.setCursor(6, 0);
69          lcd.print(resultat);
70          delay(5000);
71          reinitialiserCalculatrice();
72        }
73      } else if (touche == '*') { // Touche pour effacer
74        reinitialiserCalculatrice();
75      }
76    }
77  }
```

Fonctions :

```
79  float calculer(float num1, float num2, char op) {
80    switch (op) {
81      case 'A': return num1 + num2;
82      case 'B': return num1 - num2;
83      case 'C': return num1 * num2;
84      case 'D': return num1 / num2;
85      default: return 0;
86    }
87  }
88
89  char getSymboleOperateur(char op) {
90    switch (op) {
91      case 'A': return '+';
92      case 'B': return '-';
93      case 'C': return '*';
94      case 'D': return '/';
95      default: return ' ';
96    }
97  }
98
99  void reinitialiserCalculatrice() {
100   nombre1 = "";
101   nombre2 = "";
102   saisieDeuxiemeNombre = false;
103   lcd.clear();
104 }
105
```

TP2.4. : Control de puissance par modulation de largeur d'impulsion (PWM)

Introduction

Le but de cette expérience est d'apprendre à utiliser un **potentiomètre** comme une entrée analogique afin de contrôler la luminosité d'une **LED** via la technique **PWM (Pulse Width Modulation)**. Ce type de modulation permet de simuler une tension analogique sur une sortie numérique d'Arduino.

1. Concepts

1.1. Le Potentiomètre

Un **potentiomètre** est un **résistor variable** qui permet de modifier une tension en sortie en fonction de la rotation de son axe. Il possède **trois bornes** :

- Une borne connectée au **5V** (tension d'alimentation)
- Une borne connectée au **GND** (masse)
- Une borne centrale (**curseur**) qui fournit une tension variable entre 0V et 5V.

Cette tension est **lue** par Arduino via une **entrée analogique** (broche **A0** dans notre cas).
L'Arduino ne peut pas générer directement un signal analogique en sortie.
Pour **simuler** une tension continue entre 0V et 5V, il utilise une technique appelée **PWM (Modulation de Largeur d'Impulsion)**.

Le **PWM** fonctionne en envoyant des **impulsions numériques** (ON/OFF) très rapides avec un rapport cyclique variable :

- **0%** du temps à l'état HAUT (**0V**) → LED éteinte
- **50%** du temps à l'état HAUT (**2.5V**) → LED à mi-luminosité
- **100%** du temps à l'état HAUT (**5V**) → LED allumée à fond

Sur Arduino, les broches **9, 10, 11, 3, 5 et 6** permettent d'utiliser le **PWM** grâce à la fonction analogWrite(). La valeur d'entrée est comprise entre **0 et 255**.

2. Expérience sur Tinkercad

Nous allons maintenant appliquer ces notions dans **Tinkercad**.

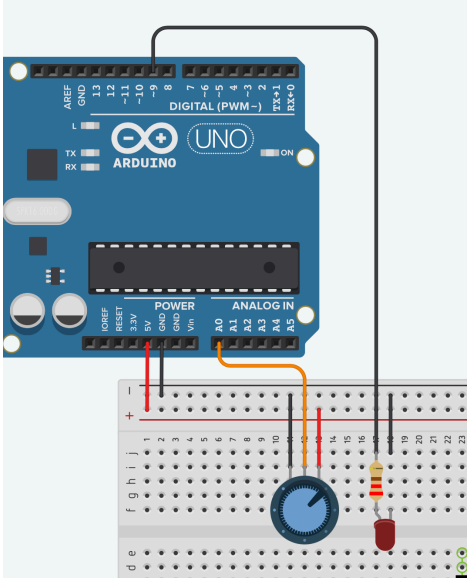
Objectif : Contrôler une LED avec une entrée analogique.
Principe : Régler l'intensité lumineuse de la LED avec le potentiomètre.
Matériel :

- 1 Arduino Uno
- 1 LED
- 1 résistance de 220Ω
- 1 potentiomètre
- Câbles de connexion

2.2. Montage du circuit

1. **Ouvrir Tinkercad** et créer un **nouveau circuit**.
2. Ajouter une **carte Arduino Uno**.
3. Placer une **LED** sur la planche d'essai (breadboard).
4. Connecter la **cathode (patte courte) de la LED** à la **masse (GND)** d'Arduino.

5. Connecter **l'anode (patte longue) de la LED** à une **résistance de 220Ω**.
6. Relier l'autre extrémité de la résistance à la **broche 9** d'Arduino.
7. Ajouter un **potentiomètre** sur la breadboard.
8. Connecter **la première borne** du potentiomètre au **5V** d'Arduino.
9. Connecter **la troisième borne** du potentiomètre au **GND** d'Arduino.
10. Connecter la **borne centrale (curseur)** du potentiomètre à l'entrée **A0** d'Ardui



3. Programmation Arduino

Maintenant que le circuit est monté, nous allons écrire le programme pour contrôler la LED en fonction de la position du potentiomètre.

📌 **Explication du code :**

1. On définit potPin comme **entrée analogique** (A0).
2. On définit ledPin comme **sortie numérique** (9).
3. Dans la boucle principale :
   - On lit la **valeur analogique** du potentiomètre (0 à 1023).
   - On convertit cette valeur en une plage adaptée au **PWM** (0 à 255).
   - On envoie cette valeur sur la LED avec analogWrite().

💻 **Code Arduino :**

```
int potPin = A0;  // Broche du potentiomètre
int ledPin = 9;   // Broche de la LED (PWM)

void setup() {
  pinMode(ledPin, OUTPUT);  // Définition de la LED en sortie
}

void loop() {
  int valeur = analogRead(potPin);       // Lire la valeur du potentiomètre (0-1023)
  int luminosite = map(valeur, 0, 1023, 0, 255); // Convertir l'échelle en PWM (0-255)
  analogWrite(ledPin, luminosite);       // Modifier la luminosité de la LED
}
```

---

4. Test et Validation

- **Téléverser** le code sur **Tinkercad**.
- **Tourner le potentiomètre** et observer la variation de la luminosité de la LED.
- **Explication attendue :** Plus le potentiomètre est tourné vers la droite, plus la LED **s'éclaire**. En tournant vers la gauche, elle **s'assombrit**.

---

5. Conclusion
Nous avons appris à : Lire une **entrée analogique** avec analogRead().
Utiliser la fonction map() pour **convertir une valeur** de 0-1023 à 0-255.
Contrôler une **LED avec le PWM** en utilisant analogWrite().
Ce projet est une base importante pour d'autres applications comme le **contrôle de moteur, de LEDs RGB, et bien plus !**

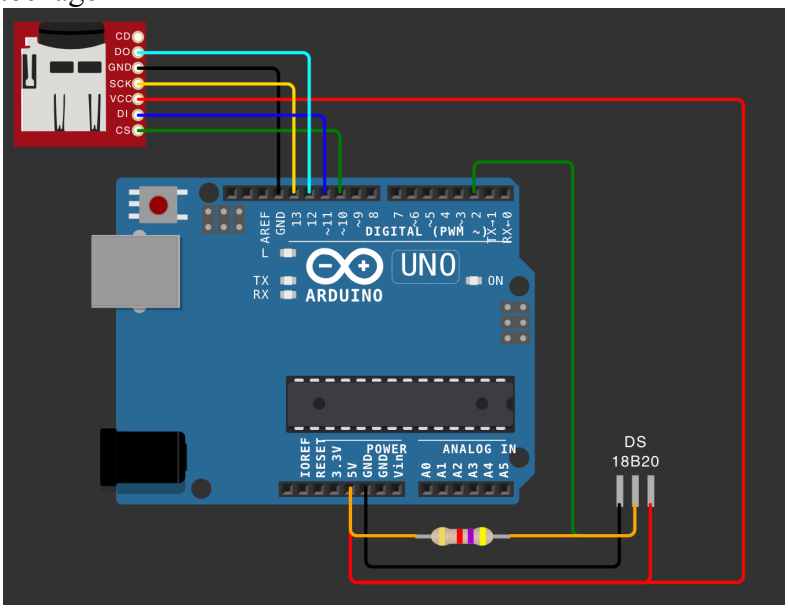
TP2.5 : Acquisition de Température avec DS18B20 et Stockage sur Carte SD

1. Objectifs du TP
À la fin de ce TP, l'étudiant doit être capable de :
- Utiliser le protocole 1-Wire avec un capteur DS18B20
- Lire des données de température avec Arduino
- Utiliser le protocole SPI
- Enregistrer des données sur une carte SD
- Comprendre l'intégration de plusieurs protocoles dans un système IoT

2. Architecture du système
DS18B20 ----(1-Wire)----> Arduino ----(SPI)----> Carte SD
Température Traitement Stockage

3. Matériel nécessaire
- Arduino UNO
- Capteur DS18B20
- Résistance 4.7 kΩ
- Module carte SD
- Carte microSD
- Breadboard
- Fils de connexion

4. Schéma de câblage



Connexion DS18B20 (1-Wire)
DS18B20 -> Arduino
VCC -> 5V
GND -> GND
DQ -> D2
Ajouter une résistance 4.7 kΩ entre DQ et 5V.

Connexion module carte SD (SPI)
Carte SD -> Arduino
VCC -> 5V
GND -> GND
CS -> D10

MOSI(DI) -> D11
MISO(DO) -> D12
SCK -> D13

5. Bibliothèques Arduino nécessaires

Installer dans l'IDE Arduino :
- OneWire
- DallasTemperature
- SD

6. Code Arduino

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SPI.h>
#include <SD.h>

// Définition des broches
const int ONE_WIRE_PIN = 2;
const int SD_CHIP_SELECT = 10;

OneWire oneWire(ONE_WIRE_PIN);
DallasTemperature sensors(&oneWire);

File dataFile;

// Fonction lecture température
float readTemperature() {
  sensors.requestTemperatures();
  return sensors.getTempCByIndex(0);
}

// Fonction écriture dans la carte SD
void saveTemperature(float temp) {

  dataFile = SD.open("temp.txt", FILE_WRITE);

  if (dataFile) {
    dataFile.print("Temperature: ");
    dataFile.println(temp);
    dataFile.close();

    Serial.println("Donnee enregistree dans la carte SD");
  }
  else {
    Serial.println("Erreur ouverture fichier");
  }
}

// Fonction lecture du fichier SD
void readSD() {

  dataFile = SD.open("temp.txt");

  if (dataFile) {
```

```
    Serial.println("Contenu du fichier temp.txt :");

    while (dataFile.available()) {
      Serial.write(dataFile.read());
    }

    dataFile.close();
  }
  else {
    Serial.println("Erreur lecture fichier");
  }
}

void setup() {

  Serial.begin(9600);

  sensors.begin();

  if (!SD.begin(SD_CHIP_SELECT)) {
    Serial.println("Erreur carte SD !");
    while (1);
  }

  Serial.println("Carte SD initialisee");
}

void loop() {

  float temperature = readTemperature();

  Serial.print("Temperature mesuree : ");
  Serial.println(temperature);

  saveTemperature(temperature);

  Serial.println("----------------------");

  readSD();

  Serial.println("----------------------");

  delay(5000);
}
```

7. Résultat attendu

Dans le moniteur série :
Carte SD initialisée
Temperature: 24.50
Donnée enregistrée

Dans la carte SD (fichier temp.txt) :
Temperature: 24.50

Temperature: 24.70
Temperature: 24.90

8. Questions pour les étudiants

1. Pourquoi ajoute-t-on une résistance 4.7 kΩ sur la ligne 1-Wire ?
2. Pourquoi le protocole SPI utilise plusieurs lignes de communication ?
3. Quelle est la différence entre 1-Wire et SPI ?
4. Pourquoi la carte SD utilise le protocole SPI ?
5. Comment peut-on améliorer ce système pour une application IoT ?

9. Conclusion

Ce TP montre comment combiner plusieurs protocoles de communication dans un système embarqué : 1-Wire pour les capteurs et SPI pour le stockage des données. Cette architecture est souvent utilisée dans les systèmes IoT de monitoring environnemental.