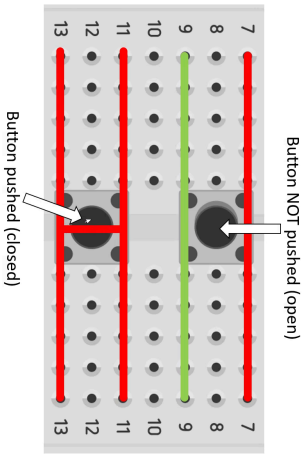
TP 1 : Arduino avec Tinkercad (suite)

TP1.3 : Les résistances de Pull-Up et Pull-Down

Objectif :

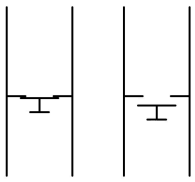
Ce TP a pour but d'illustrer l'utilisation des résistances pull-up et pull-down dans des circuits avec Arduino. Vous réaliserez trois configurations différentes pour comprendre leur rôle et testerez leur fonctionnement

Note: Un bouton-poussoir, symbolisé par deux bornes , ouvertes (séparées) ou fermées (en contact).



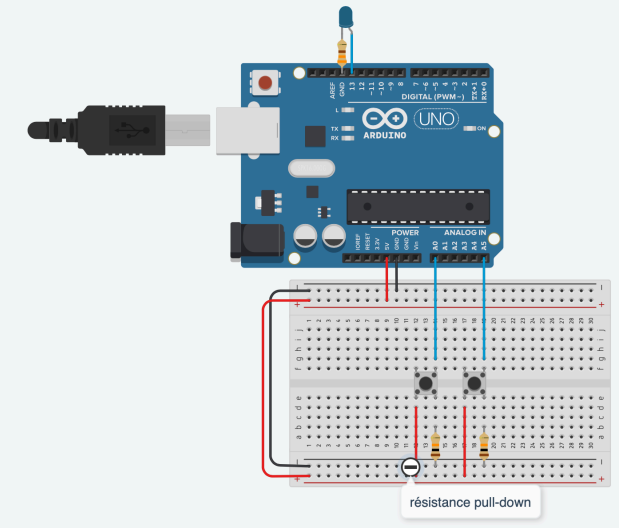
Matériel requis :

- Arduino Uno
- 2 Boutons poussoirs
- 1 LED
- 1 Résistance pour la LED (220Ω ou 330Ω)
- 2 Résistances **10kΩ** (Pull-up ou Pull-down)
- Fils de connexion
- Breadboard



Partie 1 : Résistance Pull-Down

Schéma de câblage :



- Référez-vous à l'image.
- Les résistances **pull-down** maintiennent les broches des boutons à l'état **LOW (0)** lorsqu'aucun bouton n'est pressé.

Câblage :

Bouton ON (A0) :

- Une patte connectée à **A0**.
- L'autre patte connectée à **GND**.
- Ajoutez une résistance **10kΩ** entre **A0 et GND**.

Bouton OFF (A5) :

- Une patte connectée à **A5**.
- L'autre patte connectée à **GND**.
- Ajoutez une résistance **10kΩ** entre **A5 et GND**.

LED :

- Anode (+) → Broche **13**.
- Cathode (-) → Résistance 220Ω → **GND**.

Code:

```
// Assignation des broches
const int LED = 13;       // Broche de la LED
const int boutonON = A0;  // Bouton ON connecté à A0
const int boutonOFF = A5; // Bouton OFF connecté à A5

void setup() {
  pinMode(LED, OUTPUT);    // LED en sortie
  pinMode(boutonON, INPUT); // Bouton ON en entrée
  pinMode(boutonOFF, INPUT); // Bouton OFF en entrée
}

void loop() {
  if (digitalRead(boutonON) == HIGH) { // Si bouton ON pressé
    digitalWrite(LED, HIGH);          // Allumer la LED
  }

  if (digitalRead(boutonOFF) == HIGH) { // Si bouton OFF pressé
    digitalWrite(LED, LOW);            // Éteindre la LED
  }
}
```

Partie 2 : Résistance Pull-Up Externe

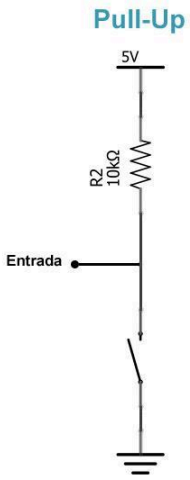
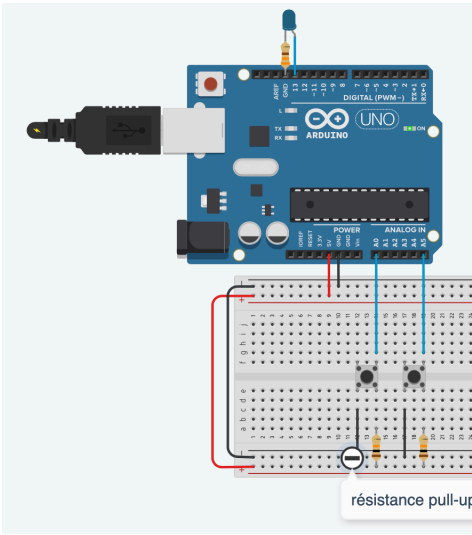


Schéma de câblage :

- Référez-vous à l'image du **deuxième circuit**.
- Les résistances **pull-up externes** maintiennent les broches des boutons à l'état **HIGH (1)** lorsqu'aucun bouton n'est pressé.

Câblage :

1. **Bouton ON (A0)** :
    - Une patte connectée à **A0**.
    - L'autre patte connectée à **GND**.
    - Ajoutez une résistance **10kΩ** entre **A0 et VCC (5V)**.
2. **Bouton OFF (A5)** :
    - Une patte connectée à **A5**.
    - L'autre patte connectée à **GND**.
    - Ajoutez une résistance **10kΩ** entre **A5 et VCC (5V)**.
3. **LED :**
    - Anode (+) → Broche **13**.
    - Cathode (-) → Résistance 220Ω → **GND**



Code :

```
// Assignation des broches
const int LED = 13;       // Broche de la LED
const int boutonON = A0;  // Bouton ON connecté à A0
const int boutonOFF = A5; // Bouton OFF connecté à A5

void setup() {
  pinMode(LED, OUTPUT);    // LED en sortie
  pinMode(boutonON, INPUT); // Bouton ON en entrée
  pinMode(boutonOFF, INPUT); // Bouton OFF en entrée
}

void loop() {
  if (digitalRead(boutonON) == LOW) { // Si bouton ON pressé
    digitalWrite(LED, HIGH);         // Allumer la LED
  }

  if (digitalRead(boutonOFF) == LOW) { // Si bouton OFF pressé
    digitalWrite(LED, LOW);           // Éteindre la LED
  }
}
```

Partie 3 : Résistance Pull-Up Interne

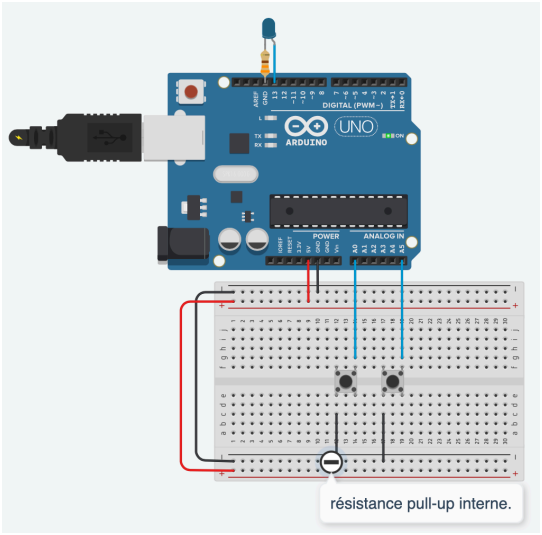
Schéma de câblage :

- Référez-vous à l'image du **troisième circuit** que vous avez partagé (avec "Résistance Pull-Up Interne").
- Aucune résistance externe n'est nécessaire. La pull-up est activée dans le code.

Entrada

Câblage :

1. **Bouton ON (A0)** :
   - Une patte connectée à **A0**.
   - L'autre patte connectée à **GND**.
2. **Bouton OFF (A5)** :
   - Une patte connectée à **A5**.
   - L'autre patte connectée à **GND**.
3. **LED :**
   - Anode (+) → Broche **13**.
   - Cathode (-) → Résistance 220Ω → **GND**.

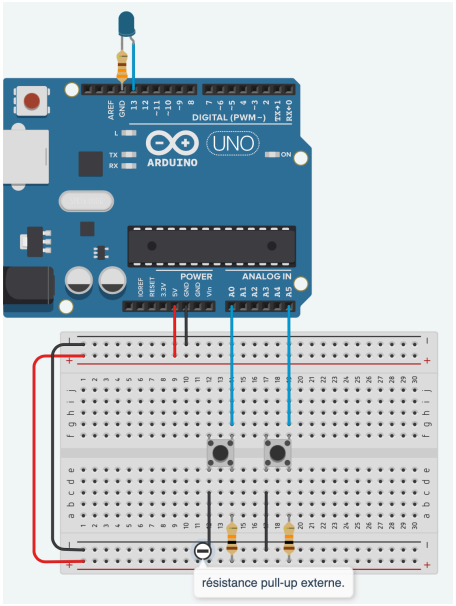


Code :

```
// Assignation des broches
const int LED = 13;       // Broche de la LED
const int boutonON = A0;  // Bouton ON connecté à A0
const int boutonOFF = A5; // Bouton OFF connecté à A5

void setup() {
  pinMode(LED, OUTPUT);           // LED en sortie
  pinMode(boutonON, INPUT_PULLUP); // Bouton ON en entrée avec pull-up interne
  pinMode(boutonOFF, INPUT_PULLUP); // Bouton OFF en entrée avec pull-up interne
}

void loop() {
  if (digitalRead(boutonON) == LOW) { // Si bouton ON pressé
    digitalWrite(LED, HIGH);         // Allumer la LED
  }

  if (digitalRead(boutonOFF) == LOW) { // Si bouton OFF pressé
    digitalWrite(LED, LOW);           // Éteindre la LED
  }
}
```

Partie 4 : Gestion Avancée d'une LED avec des Boutons Poussoirs
Matériel Nécessaire :

- **1 Arduino Uno**
- **1 LED**
- **2 résistances de 220Ω** (protection LED)
- **1 résistances de 1kΩ** (protection LED)
- **2 boutons poussoirs**
- **Câbles de connexion**



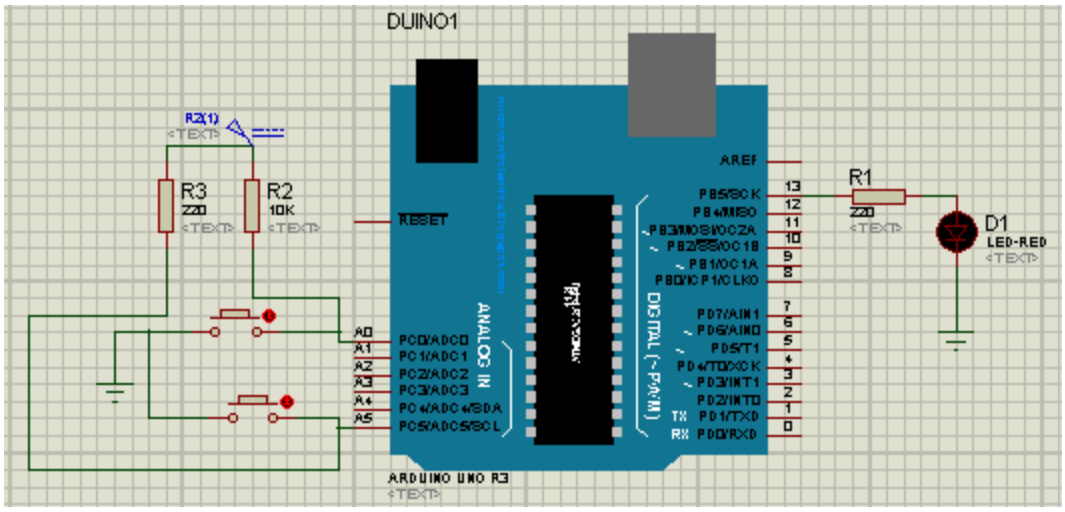
Étape 1 : Montage du Circuit dans Tinkercad

1.1. Connexion de la LED :

1. Placer une **LED** sur la **breadboard**.
2. Relier la **cathode (patte courte)** de la LED à la **masse (GND)** d'Arduino.
3. Relier **l'anode (patte longue)** de la LED à une **résistance de 220Ω**.
4. Relier l'autre extrémité de la **résistance** à la **broche 13** d'Arduino.

1.2. Connexion des Boutons Poussoirs :

1. Ajouter **deux boutons poussoirs** sur la breadboard.
2. **Bouton ON :**
    - Une borne connectée à **5V**.
    - Une autre borne connectée à **A0** (entrée Arduino) d'un côté et de l'autre côté le GND.
    - Ajoutez une **résistance de 10 kΩ** entre **A0** et **Vcc** afin de stabiliser le signal et éviter les fluctuations indésirables.
3. **Bouton OFF :**
    - Une borne connectée à **5V**.
    - Une autre borne connectée à **A5** (entrée Arduino).
    - Ajouter une **résistance de 10kΩ** entre **A0** et **Vcc** afin de stabiliser le signal et éviter les fluctuations indésirables.



Restreindre l'allumage à une fois toutes les 20 secondes

- La LED ne peut être allumée **qu'une fois toutes les 20 secondes** après le dernier allumage.
- L'utilisateur ne peut pas spammer le bouton ON pour allumer la LED plusieurs fois d'affilée.
- La LED reste allumée **5 secondes**, sauf si le bouton OFF est appuyé avant.

Code Arduino

```
int LED = 13;
int boutonON = A0;
int boutonOFF = A5;
long lastOnTime = -20001; // Initialisation permettant un premier allumage immédiat
bool ledState = false; // État de la LED

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(boutonON, INPUT_PULLUP);  // Utilisation des résistances pull-up internes
  pinMode(boutonOFF, INPUT_PULLUP); // Les boutons sont actifs LOW (0 quand pressé)
}

void loop() {
  // Vérifier si l'on peut allumer la LED (toutes les 20s)
  if (digitalRead(boutonON) == LOW && millis() - lastOnTime > 20000) {
    digitalWrite(LED, HIGH);
    ledState = true;
    lastOnTime = millis(); // Enregistrer le moment de l'allumage
  }

  // Vérifier si la LED doit s'éteindre après 5 secondes ou si bouton OFF pressé
  if (ledState && (digitalRead(boutonOFF) == LOW || millis() - lastOnTime > 5000)) {
    digitalWrite(LED, LOW);
    ledState = false;
  }
}
```

Note :

- digitalRead(boutonON): lit l'état logique de la broche boutonON (0 ou 1).
- == 0 : signifie que la broche est à l'état **LOW (0V)**.
- Cela indique généralement que le **bouton est appuyé** si le circuit utilise une **pull-up interne ou externe**.

Cas d'usage :

Si une résistance **de pull-up** (interne ou externe) est utilisée, alors :

- **Bouton relâché** → la broche est **tirée vers le haut** (HIGH, soit 1).
- **Bouton appuyé** → la broche est connectée à la masse (LOW, soit 0).

Remarque :

Certaines choses peuvent se passer pendant que la fonction **delay()** est en contrôle de la puce **Atmega**. La fonction de retard ne désactive pas **les interruptions**, **la communication série** qui reçue par la broche **RX** est enregistrée, **les valeurs PWM (analogWrite),** les **états des broches sont maintenues et les interruptions fonctionnent comme prévu.**

- ### La fonction micros()

Retourne le nombre de microsecondes depuis la carte Arduino a commencé l'exécution du programme en cours. La fonction a une résolution de 4 microsecondes (Arduino UNO) et sa valeur de retour est remise à zéro après 70 minutes de fonctionnement.

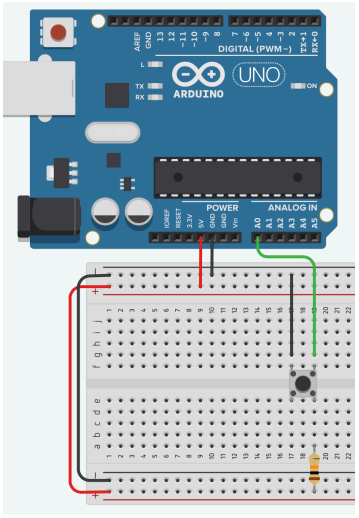
- ### La fonction delayMicroseconds()

Suspend le programme pour un laps de temps (en microsecondes) spécifié comme paramètre.

TP1.4. Communication série

Exercice1

Objectif :
Créer un circuit dans Tinkercad reliant une carte Arduino à un moniteur série (Virtual Terminal). Le moniteur série doit afficher l'état d'un bouton poussoir connecté à la broche A0 de l'Arduino en temps réel.



Composants matériels

- 1 carte Arduino Uno
- 1 bouton poussoir
- 1 résistance de 10 kΩ
- 1 Virtual Terminal (dans Tinkercad)
- Fils de connexion

Schéma de câblage

- **Bouton poussoir :**
  - Connectez une borne du bouton à l'entrée analogique A0 de l'Arduino.
  - Connectez l'autre borne du bouton à la masse (GND).
  - Ajoutez une résistance de **10 kΩ** entre la broche A0 et le **+5V** pour assurer un "pull-up".
- **Virtual Terminal :**
  - Connectez le **TX (broche 1)** de l'Arduino au **RX** du Virtual Terminal.
  - Connectez le **RX (broche 0)** de l'Arduino au **TX** du Virtual Terminal.

Code Arduino

```
int bouton = A0; // Déclare la broche du bouton
int valeur;

void setup() {
  pinMode(bouton, INPUT);     // Configure le bouton comme une entrée
  Serial.begin(9600);         // Initialise la communication série à 9600 bauds
}

void loop() {
  valeur = digitalRead(bouton); // Lire l'état du bouton
  if (!valeur)                  // Vérifie si le bouton est appuyé
    Serial.println("ON");       // Affiche "ON" dans le terminal
  else
    Serial.println("OFF");      // Affiche "OFF" dans le terminal
}
```

TP1.5. Exercice2: la communication série avec UART

Objectif :

- Envoyer des messages depuis le Moniteur Série de Tinkercad à l'Arduino.
- Afficher une réponse de l'Arduino dans le Moniteur Série.

Matériel nécessaire :

Dans Tinkercad Circuits :

- Une carte Arduino Uno
- Pas de composants supplémentaires requis (la communication série utilise les broches RX/TX internes).

Code Arduino :

```
void setup()
{
  Serial.begin(9600); // Initialiser la communication série à 9600 bauds
  Serial.println("Bienvenue dans le test UART !"); // Message d'accueil
}

void loop()
{
  // Vérifier si des données sont disponibles sur le port série
  if (Serial.available() > 0)
  {
    char receivedChar = Serial.read(); // Lire le caractère reçu
    Serial.print("Vous avez envoye : ");
    Serial.println(receivedChar); // Répondre avec le caractère reçu
  }
}
```

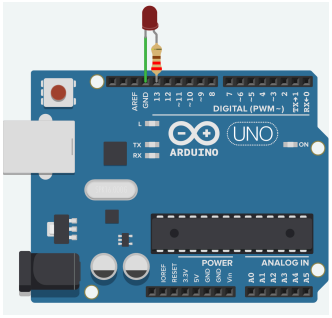
TP1.5. Exercice3: Contrôler une LED avec des commandes série

Objectif :

- Configurer une communication série bidirectionnelle entre l'ordinateur et l'Arduino.
- Envoyer des commandes spécifiques via le Moniteur Série pour contrôler une LED connectée à l'Arduino.
- Recevoir des réponses dynamiques de l'Arduino en fonction des commandes envoyées.

Matériel nécessaire :

- Une carte Arduino Uno
- Une LED
- Une résistance de 220Ω
- Fils de connexion



Étapes :

1. Schéma de câblage :

- Connectez la cathode (patte courte) de la LED à GND (terre).
- Connectez l'anode (patte longue) de la LED à une résistance de 220Ω.
- Connectez l'autre extrémité de la résistance à la broche **D13** de l'Arduino.

2. Code Arduino:

```
const int ledPin = 13; // Broche où est connectée la LED
String inputString = ""; // Stocker la commande reçue
bool ledState = false;  // État de la LED

void setup() {
  pinMode(ledPin, OUTPUT);     // Configurer la broche LED en sortie
  Serial.begin(9600);          // Initialiser la communication série
  Serial.println("Bienvenue au TP serie avance !");
  Serial.println("Envoyez 'ON.' pour allumer la LED, 'OFF.' pour l'éteindre, ou 'STATUS.' pour connaître son
état.");
}

void loop() {
  // Vérifier si des données sont disponibles
  if (Serial.available() > 0) {
    char receivedChar = Serial.read(); // Lire un caractère
    if (receivedChar == '.') {       // Fin de la commande (Enter pressé)
      Serial.println(inputString);
      processCommand();              // Traiter la commande complète
      inputString = "";              // Réinitialiser la commande
    } else {
      inputString += receivedChar;    // Ajouter le caractère à la commande

    }

  }

}

// Fonction pour traiter les commandes
void processCommand() {
  if (inputString == "ON") {
    digitalWrite(ledPin, HIGH); // Allumer la LED
```

```
    ledState = true;
    Serial.println("LED allumee !");
  } else if (inputString == "OFF") {
    digitalWrite(ledPin, LOW);  // Éteindre la LED
    ledState = false;
    Serial.println("LED eteinte !");
  } else if (inputString == "STATUS") {
    if (ledState) {
      Serial.println("La LED est allumee.");
    } else {
      Serial.println("La LED est eteinte.");
    }
  } else {
    Serial.println("Commande inconnue. Essayez 'ON.', 'OFF.', ou 'STATUS.'.");
  }
}
```

Note : Dans Tinkercad, l'utilisation de la condition `if (receivedChar == '\n')` peut poser problème, car dans le Moniteur Série (et dans certains autres environnements similaires), la touche "Entrée" (ou "Return") n'envoie pas toujours le caractère `'\n'` (nouvelle ligne). Si vous travaillez avec une carte Arduino, vous devez utiliser cette condition:

```
if (receivedChar == '\n') {      // Fin de la commande (Enter pressé)
```

3. Simulation dans Tinkercad

1. **Configurer le circuit :**
   - Chargez le code dans l'Arduino.
   - Connectez la LED et la résistance comme indiqué dans le schéma.
2. **Démarrer la simulation :**
   - Cliquez sur **"Démarrer la simulation"**.
   - Ouvrez le **Moniteur Série**.
3. **Tester les commandes :**
   - Tapez **"ON."** dans le Moniteur Série et appuyez sur Entrée : La LED s'allume, et le Moniteur Série affiche :

LED allumée !

   - Tapez **"OFF."** et appuyez sur Entrée : La LED s'éteint, et le Moniteur Série affiche :

LED éteinte !

   - Tapez **"STATUS."** et appuyez sur Entrée : Le Moniteur Série affiche l'état actuel de la LED, par exemple :

La LED est éteinte.

   - Tapez une commande inconnue, comme **"TEST."**, et le Moniteur Série affiche :

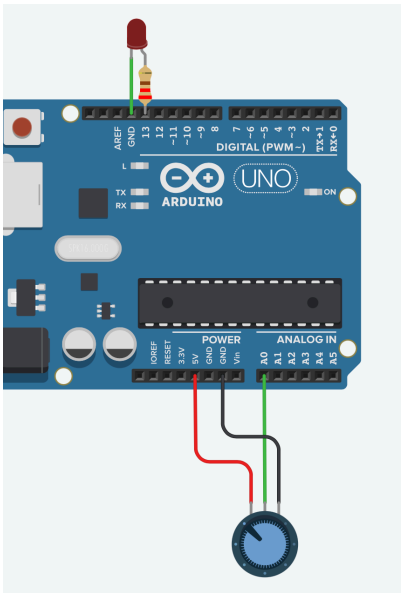
Commande inconnue. Essayez 'ON', 'OFF', ou 'STATUS'.

TP1.5. Conversion Analogique/Numérique

TP1.5. Exercice1: Variation de la tension avec un potentiomètre.

Objectif :

Pour créer une variation de la tension, nous allons utiliser un **potentiomètre** raccordé au pin A0. L'objectif de l'exercice est d'écrire un programme qui permettra de récupérer la valeur analogique de A0 et de l'afficher en temps réel dans le **Moniteur Série**.



Matériel requis :

- Une carte Arduino Uno
- Un potentiomètre
- fils de connexion

Schéma de câblage :

- **Borne gauche** du potentiomètre : connectée au **+5V** de l'Arduino.
- **Borne droite** du potentiomètre : connectée au **GND** de l'Arduino.
- **Borne centrale (curseur)** du potentiomètre : connectée à la broche **A0** de l'Arduino.

Code Arduino:

```
int valeur; // Variable pour stocker la valeur analogique

void setup()
{
  Serial.begin(9600); // Initialiser la communication série à 9600 bauds
  pinMode(A0, INPUT); // Configurer la broche A0 comme entrée
}

void loop()

{
  valeur = analogRead(A0);            // Lire la valeur analogique de la broche A0
  Serial.print("Valeur analogique : "); // Message explicatif
  Serial.println(valeur);             // Afficher la valeur dans le Moniteur Série

}
```

3. Simulation dans Tinkercad :

1. **Démarrer la simulation :**
   - Cliquez sur **"Démarrer la simulation"**.

2. **Ouvrir le Moniteur Série :**

- Cliquez sur **"Moniteur Série"**.
- Vous verrez les valeurs analogiques lues depuis la broche A0 s'afficher.

3. **Tester la variation :**

- Faites glisser le curseur du potentiomètre dans l'interface Tinkercad.
- Observez les valeurs analogiques (entre **0** et **1023**) changer en temps réel dans le Moniteur Série.

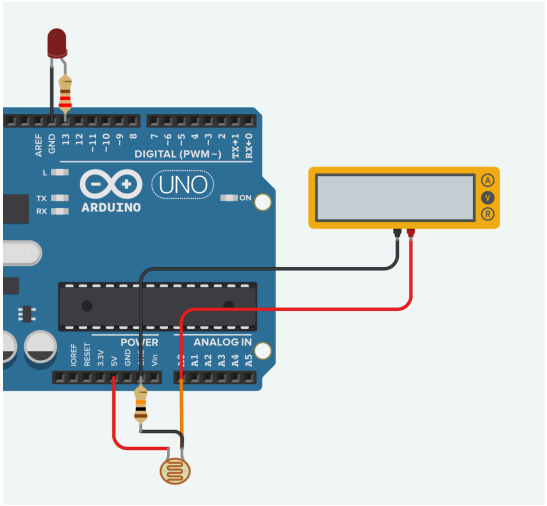
TP1.5. Exercice2: Contrôle de LED avec une photorésistance

Objectif :
Utiliser une **photorésistance** connectée à une **entrée analogique (A0)** pour **allumer ou éteindre une LED** en **fonction de la luminosité ambiante**.

- Si la valeur analogique dépasse un seuil défini, la LED s'éteint ; sinon, elle s'allume.

Matériel requis :

- Une carte Arduino Uno
- Une photorésistance (LDR)
- Une résistance de **10 kΩ** (pull-down)
- Une LED
- Une résistance de **220 Ω** pour la LED
- Fils de connexion



Étapes de réalisation :

1. Schéma de câblage :

Connectez la photorésistance (LDR) :

- Une borne de la photorésistance au **+5V**.
- L'autre borne connectée à **A0** et à une résistance de **10 kΩ**.
- L'autre extrémité de la résistance connectée au **GND**.

Connectez la LED :

- La cathode (patte courte) de la LED au **GND**.
- L'anode (patte longue) connectée à une résistance de **220 Ω**.
- L'autre extrémité de la résistance connectée à la broche **D13** de l'Arduino.

Voltmetre :

- Connectez le voltmètre en parallèle avec la photorésistance (une borne au **point de connexion entre la LDR et A0**, l'autre au **GND**).

2. Code Arduino :

```
// put function declarations here:
int LED = 13;       // Broche de la LED
int valeur;         // Variable pour stocker la valeur lue sur A0
const int seuil = 500; // Seuil de luminosité

void setup() {
  pinMode(A0, INPUT);   // Configurer la photorésistance en entrée
  pinMode(LED, OUTPUT); // Configurer la LED en sortie
}

void loop() {
  valeur = analogRead(A0); // Lire la valeur analogique de la photorésistance

  // Vérifier si la luminosité dépasse le seuil
  if (valeur >= seuil) {
    digitalWrite(LED, LOW); // Éteindre la LED
  } else {
    digitalWrite(LED, HIGH); // Allumer la LED
  }

  delay(10); // Petite pause pour stabiliser la lecture
}
```