

## Module 3: contrôles des entrées

Applications mobiles et objets connectés 420-W48-SF

# Objectifs

- Entrées numériques
- Entrées analogiques
- Potentiomètre
- Port série

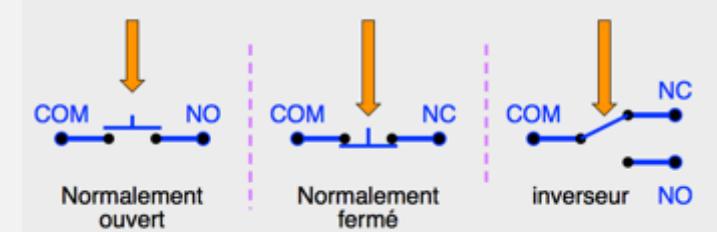
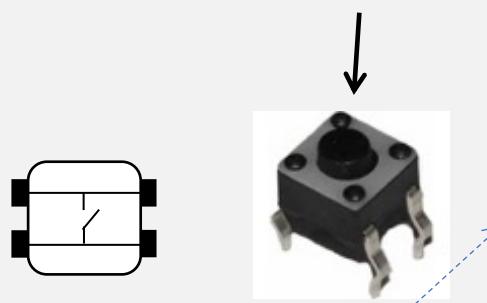
# Entrée numérique

- Interrupteur

- Composant coupant ou laissant passer le courant
  - Modèles répondant à des besoins différents
  - Grande variété sur le marché

- Bouton poussoir est du type Normalement ouvert

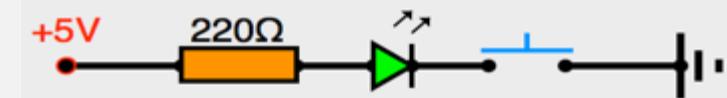
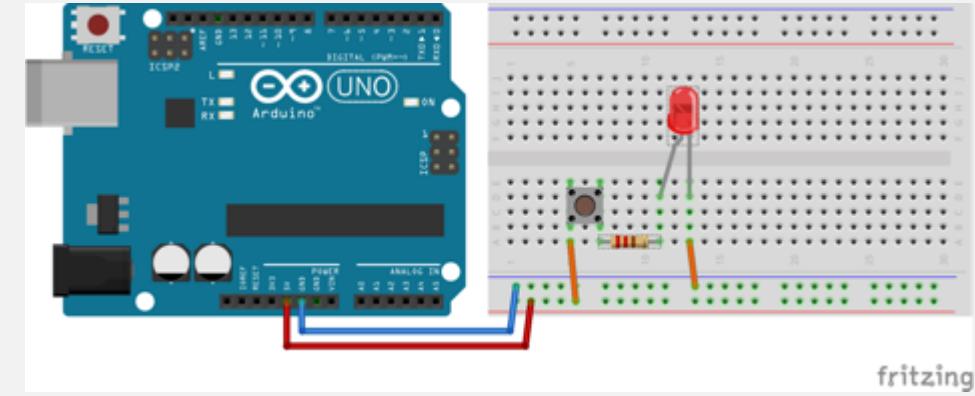
- La borne (pin) passe au niveau LOW lorsque reliée au ground
- La borne (pin) passe au niveau HIGH lorsque reliée au 5 V
- Tolérances aux écarts logiques en entrée pour Arduino



État	Logique	Électricité
LOW	0	0 V - 1,5 V
HIGH	1	3 V -- 5 V
Incertain	???	1,5 V - 3 V

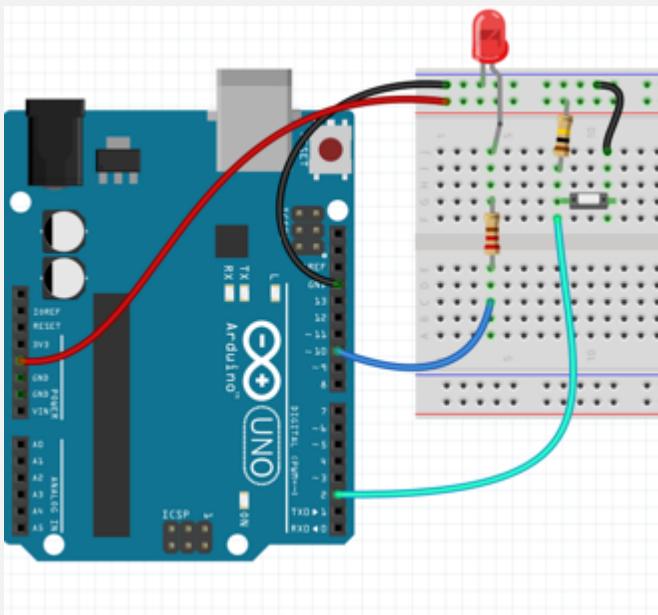
# Entrée numérique

- Contrôler l'éclairage de la DEL
  - Le bouton-poussoir « ferme » le circuit
- Bouton poussoir au laboratoire
  - 4 bornes sur 2 côtés
  - 2 bornes du même côté sont en contact si le bouton est enfoncé
  - Vérification possible avec le multimètre en position **Ω**



# Entrée numérique

- Programmation des bornes numériques en entrée
  - Déclaration du sens de l'usage : **pinMode (<NoBorne>, INPUT)**
  - Obtenir l'état de la borne : *variable = digitalRead (<NoBorne>)*
  - Retourne LOW ou HIGH



```
const int borneEntree = 2;
const int borneSortie = 10;
int valBouton;

void setup() {
    pinMode(borneEntree, INPUT);
    pinMode(borneSortie, OUTPUT);
}

void loop() {
    valBouton = digitalRead(borneEntree); // retourne LOW ou HIGH
    digitalWrite(borneSortie, !valBouton);
    delay(500);                      // s'assure de la stabilité du contact
}
```

# Entrée numérique

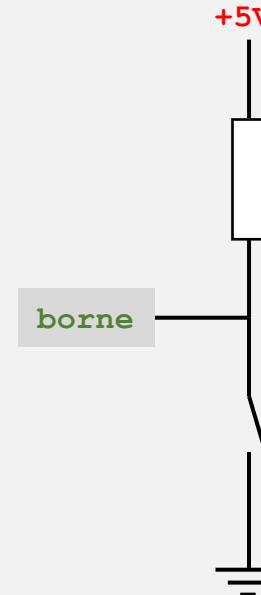
- Phénomène d'antenne parasite
  - Lorsque les bornes numériques sont utilisées en entrée, les fils agissent comme des antennes d'ondes électromagnétiques
  - Provoque de fausses interprétations:
    - Arduino peut lire des milliers de signaux par seconde, faisant croire à des mouvements rapides de l'interrupteur
    - L'oeil ne perçoit pas ce phénomène
  - Dépend de la qualité des matériaux de l'interrupteur
- Solution : usage d'une résistance Pull-up

```
void setup() {  
    pinMode(borneEntree, INPUT);
```

# Entrée numérique

- Résistance Pull-up interne
  - Paramètre INPUT\_PULLUP
  - Avantage : simplicité
  - Inconvénient : dépend de la valeur de la résistance interne, possibilité d'erreur de programmation (suppression du \_PULLUP)
  - Utilisé durant les essais
- Résistance Pull-up externe
  - Avantage: contrôle absolu par le programmeur
  - Inconvénient: circuit supplémentaire à mettre en place
  - Utilisé sur un circuit permanent
- (Le format Pull\_down existe aussi, mais non traité ici)

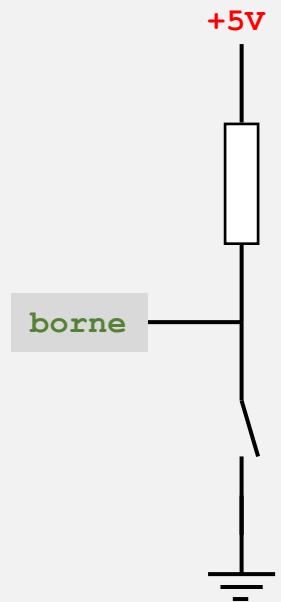
```
void setup() {  
    pinMode(borneEntree, INPUT_PULLUP);
```



# Pull-up externe : principe de fonctionnement

## Bouton en position **ouverte**

- L'électricité circule de la borne +5 V vers la borne d'entrée
- La borne détecte l'état HIGH

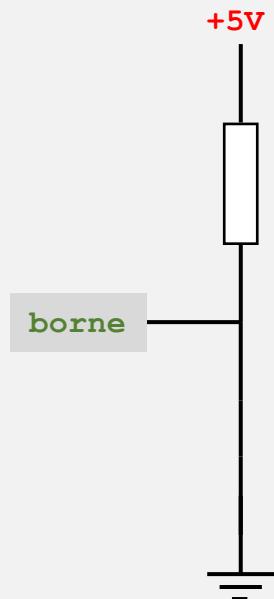
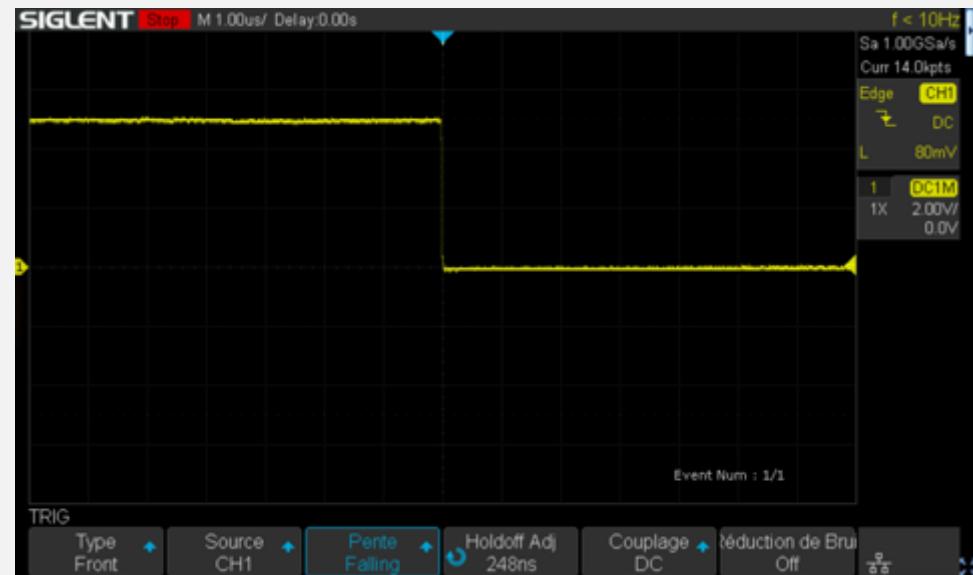


ATTENTION: la résistance Pull-up doit avoir une valeur élevée ( 5 KΩ ou plus)

# Pull-up externe : principe de fonctionnement

## Bouton en position fermée

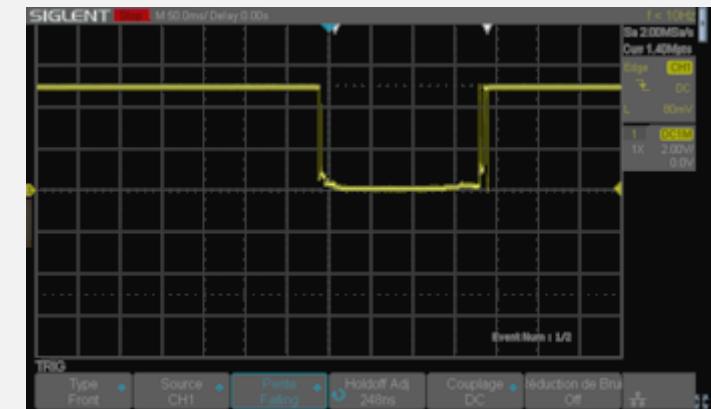
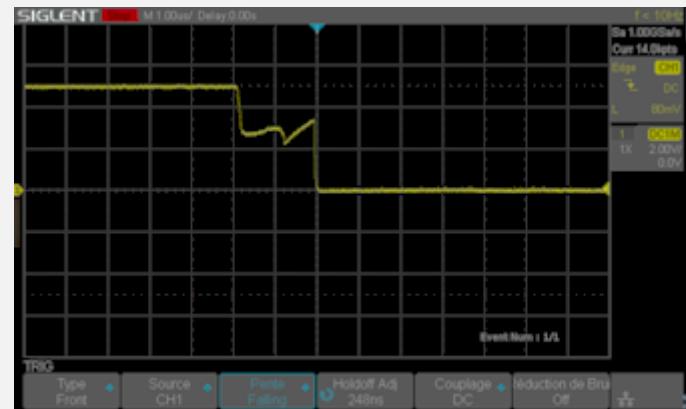
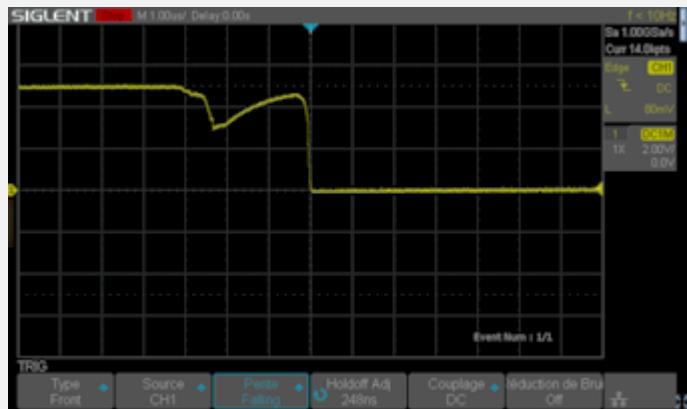
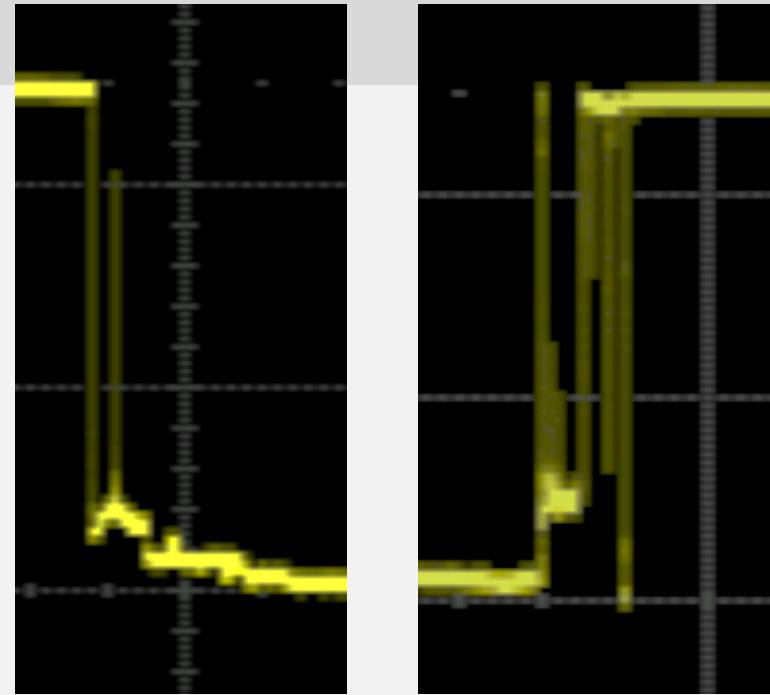
- Prend le chemin le plus facile
- La borne détecte l'état `LOW`
- **Attention sans résistance, il y aurait un court-circuit**



ATTENTION: la résistance Pull-up doit avoir une valeur élevée ( 5 KΩ ou plus)

# Entrée numérique

- Rebond: émission d'impulsions de très courte durée
  - Causées par le passage rapide de  $0\text{ V}$  à  $V_{\max}$
  - Provoque de fausses informations:
    - Arduino peut lire des milliers de signaux par seconde, faisant croire à des mouvements rapides de l'interrupteur
    - L'oeil ne perçoit pas ce phénomène



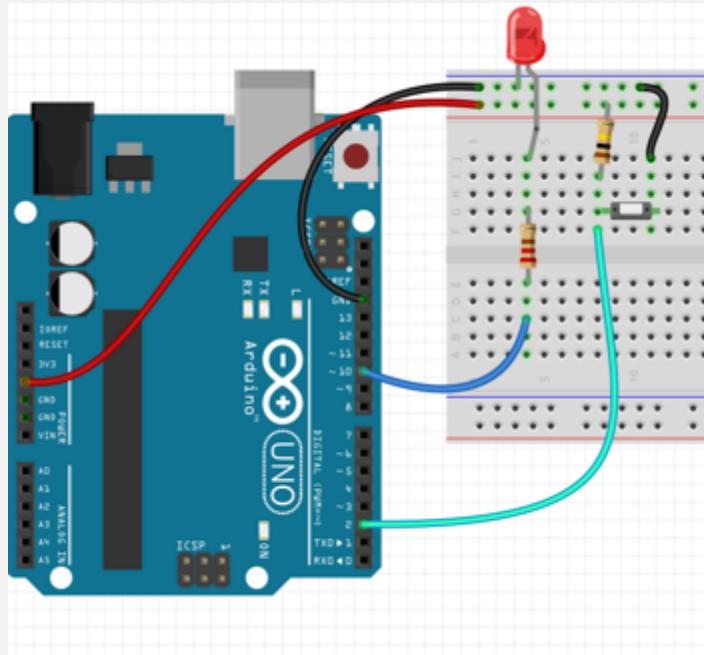
# Entrée numérique

- Solutions
  - Matériel :
    - Interrupteurs certifiés anti-rebond (\$\$\$)
    - Utilisation d'un condensateur
  - Logiciel :
    - Attendre quelques millisecondes avant d'analyser l'état du bouton
    - Considérer que le bouton est appuyé si le délai mesuré comme appuyé dépasse une valeur prédéterminée
    - Etc.



# Entrée numérique – Solution Logicielle 1

- Attendre un certain temps avant de décider si le bouton est actionné
- Inconvénient : ralentissement du traitement général
- Utilisé seulement si le délai d'attente n'est pas critique au traitement



```
const int borneEntree = 2;
const int borneSortie = 10;
int valBouton;

void setup() {
    pinMode(borneEntree, INPUT);
    pinMode(borneSortie, OUTPUT);
}

void loop() {
    valBouton = digitalRead(borneEntree); // retourne LOW ou HIGH
    digitalWrite(borneSortie, !valBouton);
    delay(500);                         // s'assure de la stabilité du contact
}
```

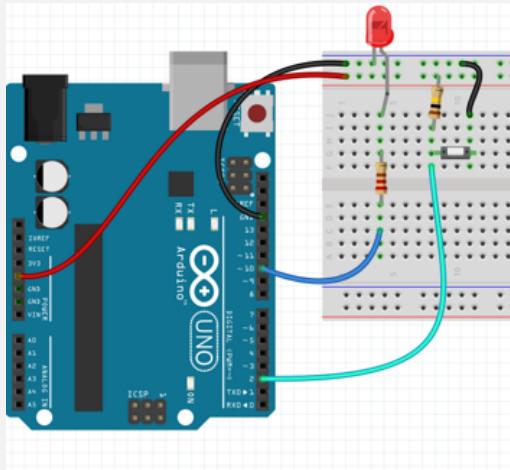
# Entrée numérique – Solution logicielle 2

```
const int borneEntree = 2;
const int borneSortie = 10;

int dernierEtatLed = LOW;

long dernierTemp = 0;
int dernierEtatBouton = HIGH;
int dernierEtatStableBouton = HIGH;
const int delaiMinPression = 25;

void setup(){
  pinMode(borneSortie, OUTPUT);
  pinMode(borneEntree, INPUT);
}
```

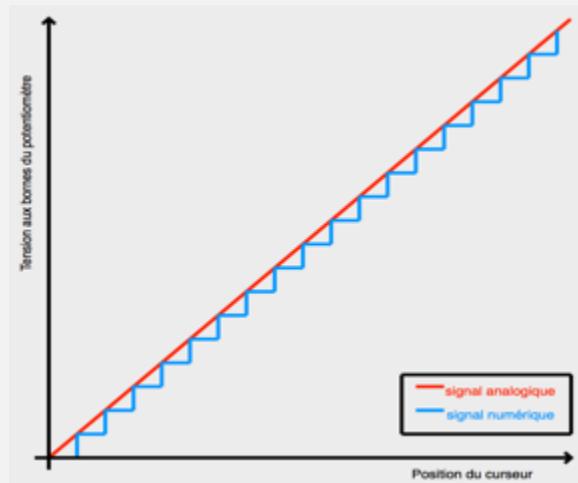
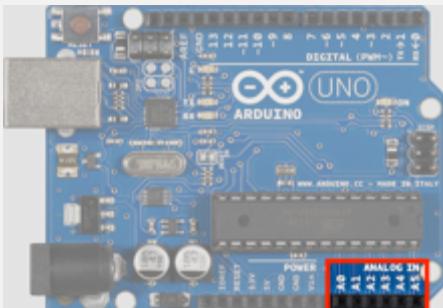


```
void loop(){
  int etatBouton = digitalRead(borneEntree);
  if (etatBouton != dernierEtatBouton) {
    dernierTemp = millis();
    dernierEtatBouton = etatBouton;
  }

  if(millis() - dernierTemp > delaiMinPression) {
    if (etatBouton == LOW && dernierEtatStableBouton == HIGH) {
      // bouton appuyé
      // Action à réaliser
    } else if (etatBouton == HIGH && dernierEtatStableBouton == LOW) {
      // bouton relaché
      // ... et donc comme click
      // Action à réaliser
      dernierEtatLed = !dernierEtatLed;
      digitalWrite(borneSortie, dernierEtatLed);
    }
    dernierEtatStableBouton = etatBouton;
  }
}
```

# Entrées analogiques

- Bornes d'entrées mesurant des valeurs analogiques de tension
  - Permettre au microcontrôleur de réagir au monde extérieur
  - DéTECTEURS produisent de l'électricité analogique
- Arduino est équipé de 6 bornes analogiques, notées A0, A1, ..., A5
  - Toutes possèdent la même caractéristique électronique
  - ADC (Analogic Digital Converter) de 10 bits :  $2^{10} = 1024 \Rightarrow 0 \text{ à } 1023$

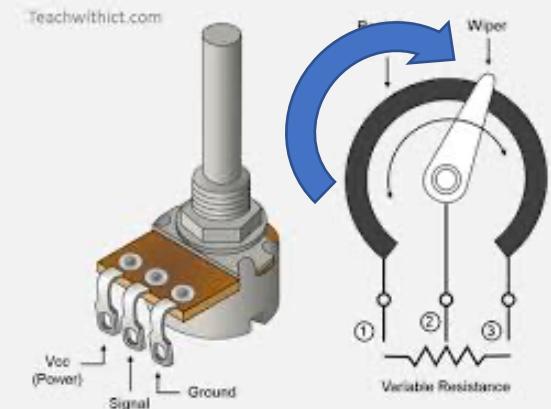


# Entrées analogiques

- Programmation des bornes analogiques
  - Obtenir l'état de la borne : `variable = analogRead (NoBorne)`
  - Tensions d'entrée traduites en valeur entre 0 et 1023
  - Si on utiliser la tension de 5V de référence de l'arduino Uno :
    - 0 correspond à 0V
    - 1023 correspond à 5V
  - `valeurVolt = valeurLue * 5 / 1023`
    - Si 434 lue, on a  $434 * 5 / 1023 = 2,12V$
  - `valeurLue = valeurVolt * 1023 / 5`
    - Si 3,3V, on a  $3,3 * 1023 / 5 = 675$

# Potentiomètre analogique

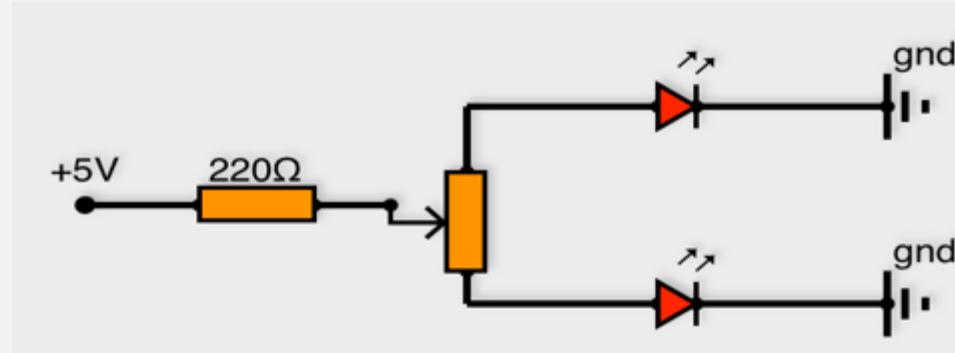
- La MLI (PWM) fait varier l'intensité de signaux numériques
- Le potentiomètre analogique fait varier l'intensité de signaux analogiques de 0 V à  $V_{max}$
- Construction du modèle standard
  - 3 bornes :
    - Borne centrale peut se déplacer entre les 2 extrémités
    - Variation linéaire ou logarithmique de la résistance selon le modèle
- Fonctionnement
  - Borne centrale en position 1:  $R_{efficace} = 0 \ \Omega$
  - Borne centrale au centre  $R_{efficace} = R_{max} / 2 \ \Omega^{(1)}$
  - Borne centrale en position 3:  $R_{efficace} = R_{max}$



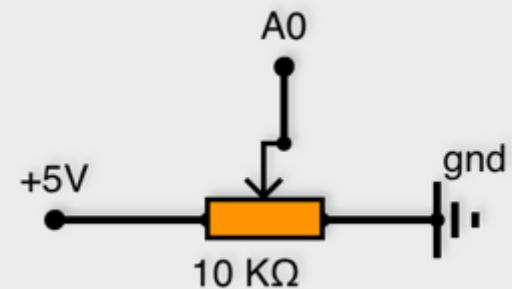
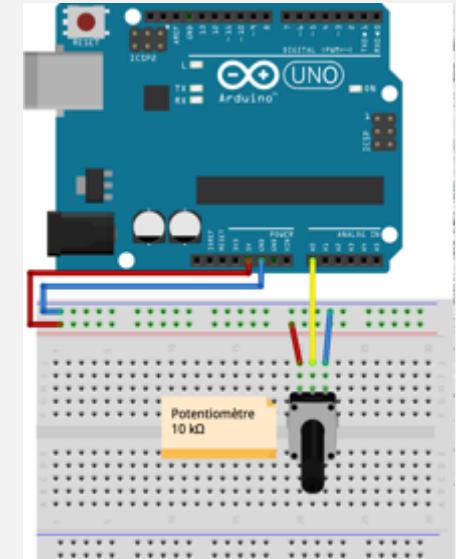
(1): modèle à résistance linéaire

# Potentiomètre analogique

- Fonction: contrôler des circuits de façon progressive
  - Exemples: moteur, chauffage

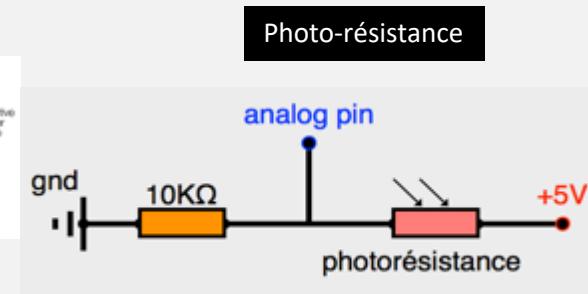
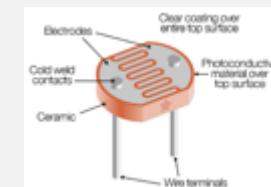
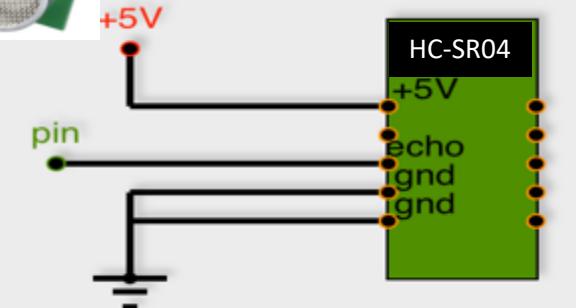


- Mesure
  - Par multimètre
    - $R_{max}$  entre les bornes 1 et 3 détermine la caractéristique
    - $R_{efficace}$  entre les bornes 1 et 2 ou entre 2 et 3



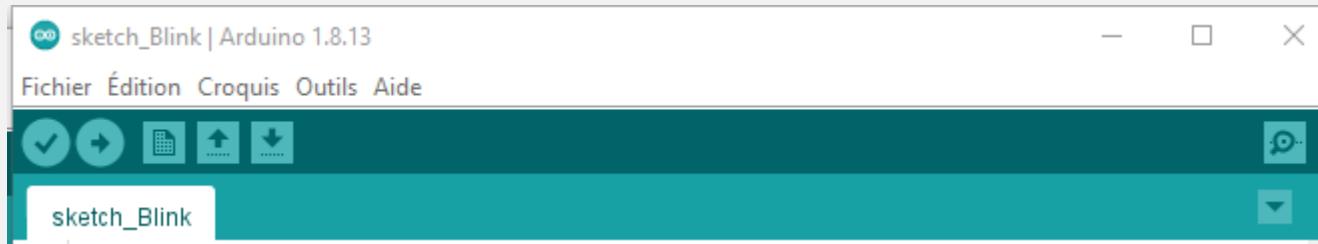
# DéTECTEURS ANALOGIQUES

- Usage varié de détecteurs pour toutes les occasions
  - Détecteur à ultra-son (HC-SR-04) : l'écho peut déterminer la distance aux obstacles
  - Détecteur infra-rouge : détecter la présence d'un objet
  - Photo-résistance : mesurer l'intensité de l'éclairage
  - Température : TMP36, LM35, ...



# Console – Configuration

- Interface de dialogue affichant des valeurs en entrée et en sortie
  - Utilisé pour tracer et dépanner



- Activé par l'instruction `Serial.begin(9600);`  
(Autres vitesses en bauds : 9600, 14400, 19200, 57600, 115200)

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

- Accessible sur les bornes RX et TX (UART : 3,3V ou 5V). Attention aux tensions de sorties qui ne sont pas que le RS232 (12V)

# Console – Sortie

- Écriture sur le port série

- Serial.print(<valeur>)
- Serial.println(<valeur>)

```
void setup() {  
    Serial.begin(9600);  
    Serial.println("démarrage de la console");  
}  
  
void loop() {  
    Serial.print("etat de la DEL : ");  
    Serial.println(valeurDel);  
}
```

# Console – Entrée

- Lecture sur le port série
  - `Serial.available()` -> int : nombre de bytes dans le tampon (max : 64 octets)
  - `Serial.read()` -> int : valeur du caractère lu sur 8 bits
  - `Serial.readString()` -> String : chaîne lue
  - [...]
- Par défaut, il y a un délai d'abandon de 1000ms (timeout) : on peut le changer avec `Serial.setTimeout(<délai>)`

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

void loop() {
  String texte = Serial.readString();
  Serial.println(texte);
}
```

# Références

- <https://www.arduino.cc/reference/en/>
- <https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino> : images prises dans ce cours