

Contrôles des entrées 1/2

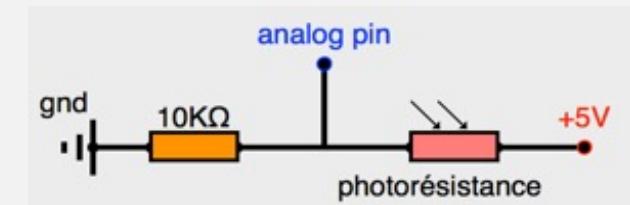
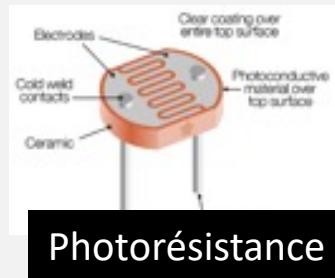
Applications mobiles et objets connectés 420-W48-SF

Objectifs

- Entrées analogiques
 - Convertisseur analogique-numérique
 - Application au potentiomètre
 - Application au détecteur à ultra-sons
- Port série
 - Affichage et outil de dépannage en sortie
 - Collecte de données en entrée

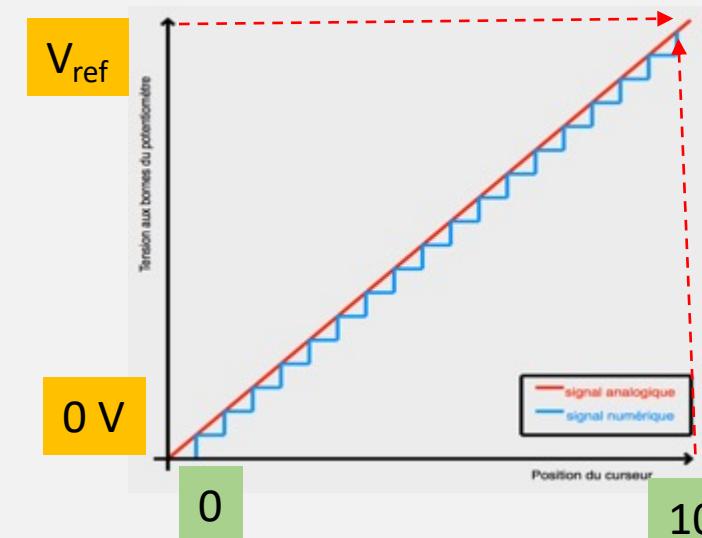
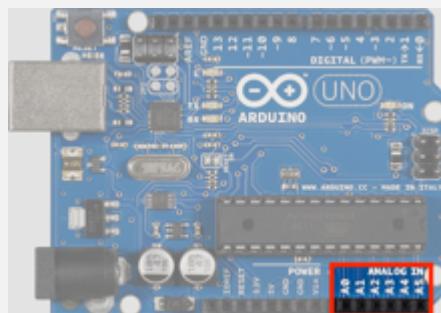
DéTECTEURS ANALOGIQUES

- Usage varié de détecteurs pour toutes les occasions
 - Potentiomètre analogique
 - Détecteur à ultra-son (HC-SR04) : mesurer des distances
 - Détecteur infra-rouge : détecter la présence d'un objet
 - Photorésistance : mesurer l'intensité de l'éclairage
 - Thermo-résistance: TMP36, LM35, ...



Entrées analogiques

- Bornes d'entrées mesurant des valeurs analogiques de tensions
 - Nombreux détecteurs analogiques sur le marché
- Arduino est équipé de 6 bornes analogiques, notées A0, A1, ..., A5
 - Toutes possèdent la même caractéristique électronique
 - Circuit ADC (Analogic Digital Converter) échantillonne la tension d'entrée qu'il convertit en signaux numériques sur 10 bits
donc: $2^{10} = 1024 \Rightarrow 0 \text{ à } 1023$

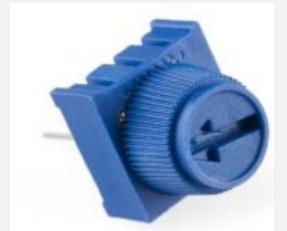


Entrées analogiques

- Tension de référence, notée V_{ref} de 5 V, par défaut
 - Tensions d'entrée traduites en valeur entre 0 et 1023
 - Valeur 0 correspond à 0 Volt
 - Valeur 1023 correspond à 5 Volts
- Conversion
 - $valeurVolt = valeurLue * 5 / 1023$
 - Si 434 lue, on a $434 * 5 / 1023 = 2,12 \text{ V}$
 - Inversement, $valeurLue = valeurVolt * 1023 / 5$
 - Si 3,3V mesuré, on a $3,3 * 1023 / 5 = 675$
- Programmation des bornes analogiques
 - Lecture de la valeur d'une borne : *variable* = **analogRead** (*NoBorne*)

Potentiomètre analogique

- Le potentiomètre analogique fait varier l'intensité de signaux analogiques de 0 Volt à V_{max}
- Construction du modèle standard
 - 3 bornes :
 - Borne centrale peut se déplacer entre les 2 extrémités
 - 2 modèles courants: variation linéaire ou variation logarithmique (audio)
- Caractéristique: R_{max}
 - Mesure par Ohmmètre
 - R_{max} Sondes entre les bornes extrêmes 1 et 3
 - $R_{efficace}$ Sondes entre les bornes 1 et 2 ou entre 2 et 3

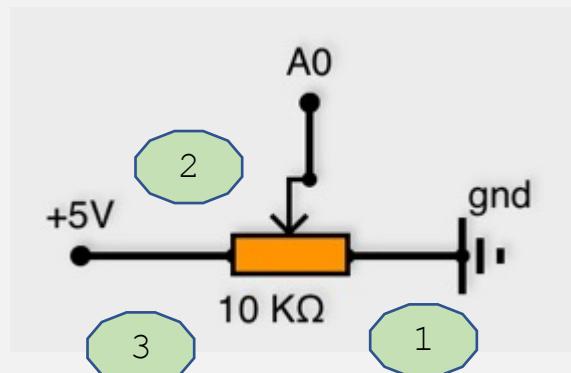
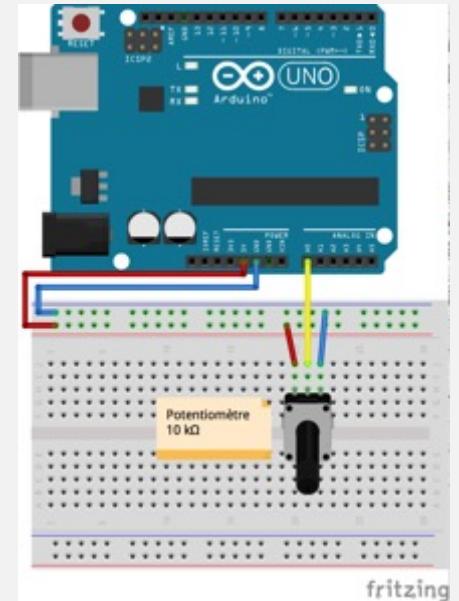
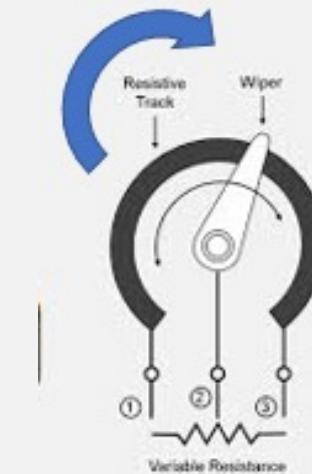


Potentiomètre analogique

- Fonction: contrôler des circuits de façon progressive
 - Exemples: éclairage, moteur, haut-parleurs

- Fonctionnement

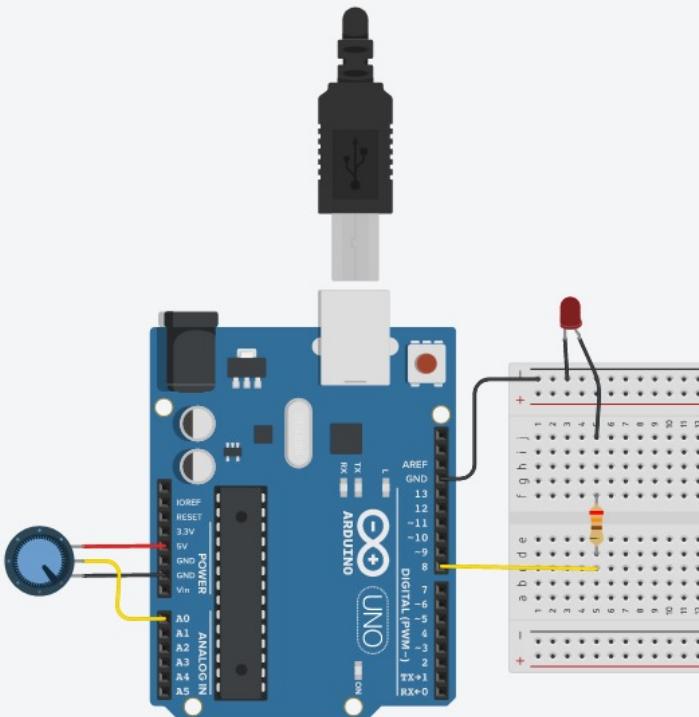
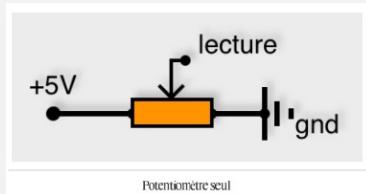
- Pivot central en position 1: $R_{\text{efficace}} = 0 \Omega$
- Pivot central au milieu $R_{\text{efficace}} = R_{\text{max}} / 2 \Omega^{(1)}$
- Pivot central en position 3: $R_{\text{efficace}} = R_{\text{max}}$



(1): modèle à résistance linéaire

Application avec le potentiomètre 1/2

- Afficher la tension d'entrée d'un potentiomètre



```
int bornePotentiometre = A0;

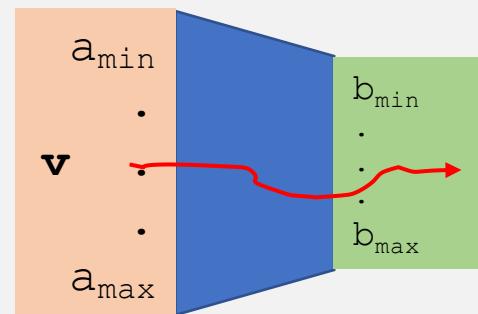
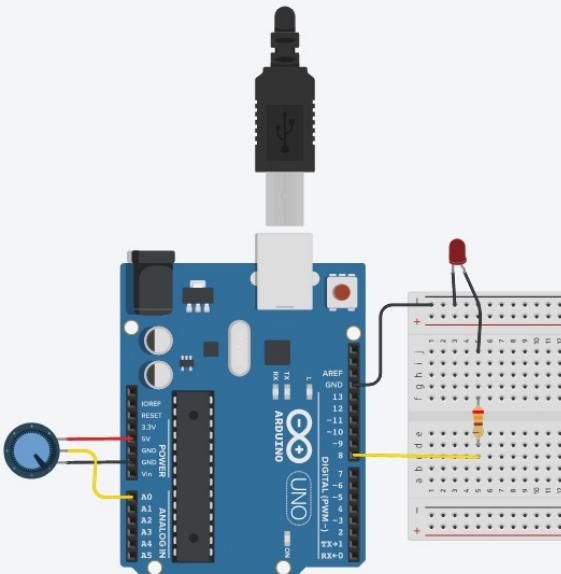
void setup() {
    Serial.begin(9600);
    Serial.println("démarrage de la console");
}

void loop() {
    static int valeurLue;
    valeurLue = analogRead(bornePotentiometre);
    Serial.print("Valeur Lue =");
    Serial.print(valeurLue);
    Serial.print(" Ve = ");
    Serial.print(valeurLue*5.0/1024.0);
    Serial.println("V");
}
```

Application avec le potentiomètre 2/2

- Subordonner une sortie numérique par un signal analogique
 - Fonction `map`: établit une relation linéaire entre une intervalle en entrée et une intervalle en sortie pour une valeur donnée

```
resultat = map(v, a_min, a_max, b_min, b_max)
```



```
const int borneSortie = 8;
int bornePotentiometre = A0;

void setup() {
    pinMode(borneRouge, OUTPUT);
}

void loop() {
    static int valeurLue;
    valeurLue = analogRead(bornePotentiometre);
    int valeurBorneSortie = map(valeurLue, 0, 1023, 0, 255);
    analogWrite(borneSortie, valeurBorneSortie);
}
```

Si `valeurLue` = 341 (1/3 de la course du pivot, la fonction `map` retourne 85

Console – Sortie - Rappels

- Interface de dialogue affichant des valeurs en entrée et en sortie
 - Utilisé pour afficher des valeur OU pour dépanner
 - Activé par l'instruction `Serial.begin(9600);`
 - Utilisé par les instructions
 - `Serial.print(<<valeur>>);`
 - `Serial.println(<<valeur>>);`

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

Console – Sortie - Rappels

- Écriture sur le port série

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

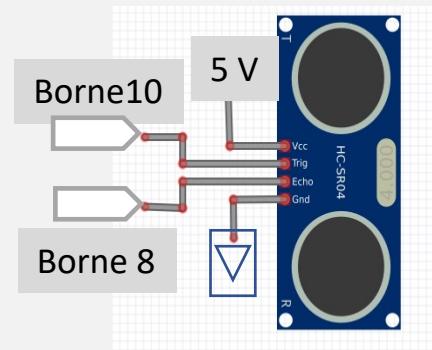
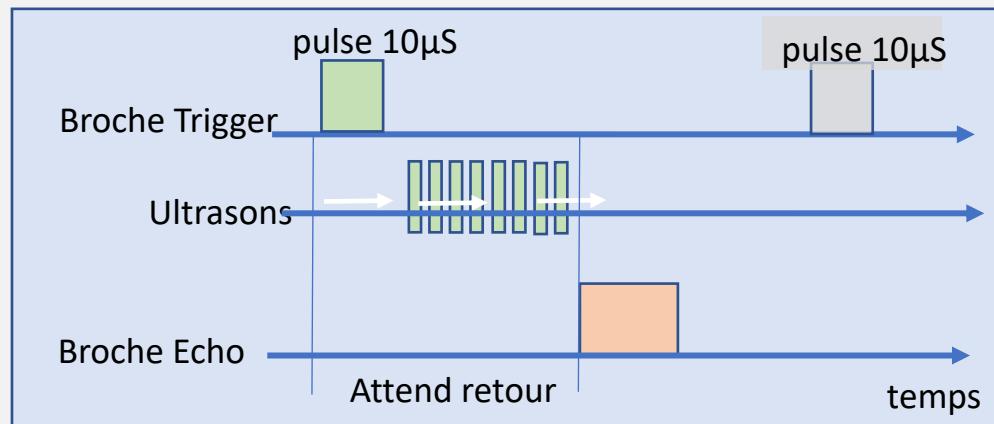
Syntaxe de <valeur>

- type [, format]
- type peut être n'importe quel type accepté par le langage
- format: option d'affichage pratique offrant la conversion comme BIN, HEX, DEC

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

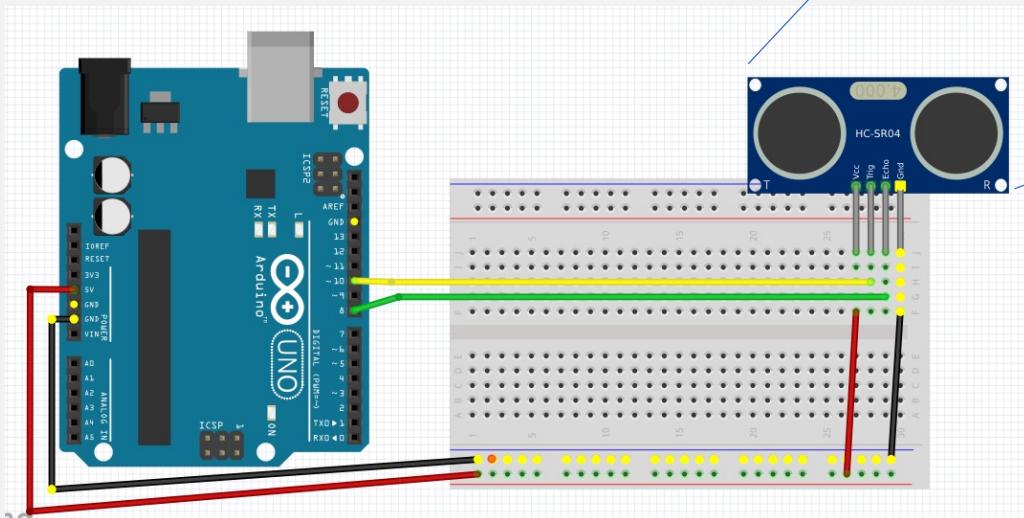
Application avec le sonar

- Principe: déterminer une distance à partir d'un écho sonar
 1. Borne « trigger » mise à l'état HIGH, Arduino débute un compte à rebours
 2. Émission d'un train d'ondes ultrasoniques (40 kHz)
 3. Une onde se déplace et rebondit sur un obstacle
 4. Borne « echo » positionne son état HIGH lorsque l'onde est reçue; Arduino calcule le temps écoulé
- Le temps de parcours = Vitesse_{son} / 2



Application avec le détecteur à Ultra-sons

- Branchement



- Programmation

```
const int borneTrigger = 10;
const int borneEcho = 8;
long duree, cm, pouces;
int distance;
void setup() {
    Serial.begin(9600);
    Serial.println("démarrage de la console");
    //completez
}
void loop() {
    pinMode(borneEcho, INPUT);
    digitalWrite(borneTrigger, LOW);
    delayMicroseconds(5);
    digitalWrite(borneTrigger, HIGH);
    delayMicroseconds(10);
    digitalWrite(borneTrigger, LOW);
    duree = pulseIn(borneEcho, HIGH);
    cm = (duree/2) /29.1;
    Serial.print(cm);
    Serial.println(" cm de distance");
}
```

Console – Entrée

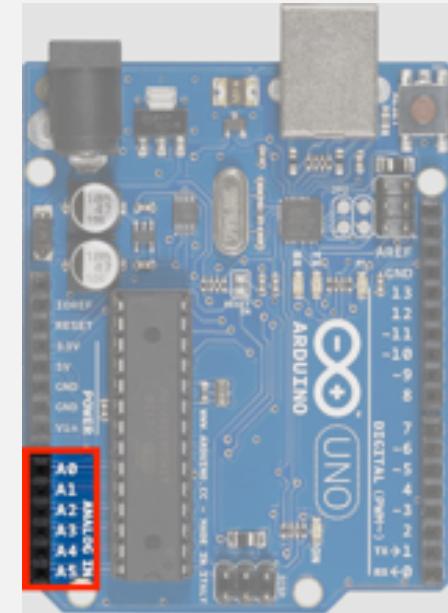
- Mode semi-interactif: fournir des données au microcontrôleur
- Lecture sur le port série
 - `Serial.available()` -> int : retourne le nombre d'octets dans le tampon d'entrée série (max : 64 octets)
 - `Serial.read()` -> int : lit la prochaine valeur dans le tampon
 - `Serial.readString()` -> String : chaîne lue
 - [...]

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

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

Résumé

- Bornes analogiques: A0 à A5
 - Lecture de données *variable* = **analogRead** (*NoBorne*)
 - étalonnée sur 1024 bits
 - Équivalent en Volts: *variable* **5.0/1024.0*
- Affichage sur console
 - Texte ou valeurs servant au dépannage
 - Serial.print() et Serial.println()
- Données saisies sur la console
 - Serial.available(), Serial.read<*formesVariées*>
- Sorties numériques par la fonction **map**
 - Intervalles [0 ,1023] → [0, 255] dans le cas d'Arduino UNO



Vidéos

- Principle Convertisseur ADC

<https://www.youtube.com/embed/aeNKKqq9s7o?start=0&end=258>

- Branchement

<https://www.youtube.com/embed/aeNKKqq9s7o?start=362&end=604>

- Programme

<https://www.youtube.com/embed/aeNKKqq9s7o?start=694&end=1045>

Références

- <https://www.arduino.cc/reference/en/>
- Principle Convertisseur ADC
<https://www.youtube.com/embed/aeNKKqq9s7o?start=0&end=258>
- Branchement
<https://www.youtube.com/embed/aeNKKqq9s7o?start=362&end=604>
- Programme
<https://www.youtube.com/embed/aeNKKqq9s7o?start=694&end=1045>
- images prises dans ce cours
 - <https://www.carnetdumaker.net/articles/mesurer-une-distance-avec-un-capteur-ultrason-hc-sr04-et-une-carte-arduino-genuino/>