MARTIGNAS

Stage
Initiation à la programmation et à l'électronique avec ARDUINO

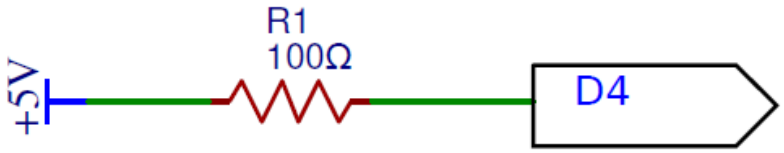
Jour 3

Programme du jour 3

- PROJET 5: Les entrées numériques
  - Resistance pull-up/pull-down
  - Le bar graph pour afficher des valeurs numériques
  - Déparasiter à l'aide d'un condensateur
- PROJET 6: Les entrées analogiques
  - La photorésistance
  - Conversion analogique numérique
  - Faire varier la luminosité d'une LED en fonction de la luminosité ambiante

Projet 5 : les entrées numériques

- Protéger les pin d'Arduino
  - Nous allons utiliser les pin sous forme d'entrée (INPUT); c'est-à-dire nous allons appliquer une tension sur la pin.
    Tout se passera bien, si au niveau du programme, la broche est configurée comme input. Mais si elle devait être configurée en output par erreur, il est presque certain que le microcontrôleur finira immédiatement au paradis des puces électroniques grillées.
  - Ainsi, pour éviter de condamner notre microcontrôleur à la chaise électrique, nous allons devoir le protéger. Cela peut se faire en connectant sur la broche du microcontrôleur utilisé comme input une résistance de 100 à 200Ω en série avec le signal à acquérir



Projet 5 : les entrées numériques

- Protéger les pin d'Arduino
  - ATTENTION!

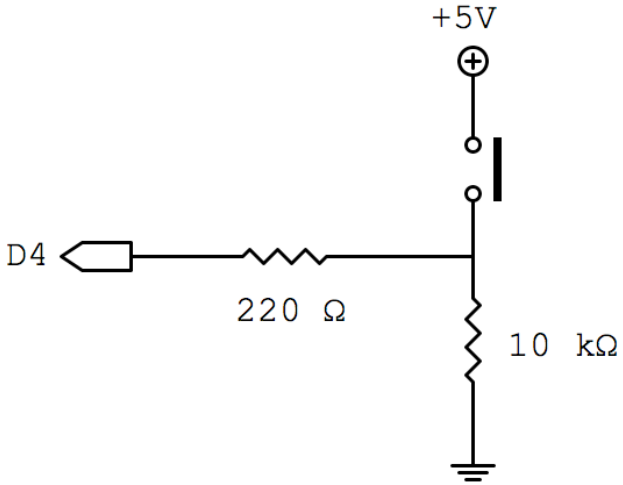
    Lorsque vous branchez l'Arduino à l'ordinateur, le dernier programme reçu est exécuté. Avant de commencer un nouveau montage, il est plus que conseillé d'envoyer un programme d'initialisation, du modèle de celui-ci:

```
//Programme vide d'initialisation
void setup() {
}

void loop() {
}
```

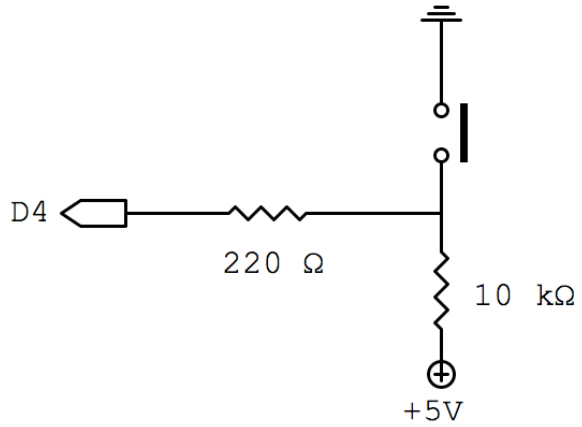
Projet 5 : les entrées numériques

- Résistance Pull-Down
  - Lorsque l'on presse le bouton poussoir, on envoie un courant électrique sur la broche 4, qui sera interprété comme un 1.
    Lorsqu'on relâche le bouton, il n'y a plus de courant qui arrive sur la broche 4, ce qui sera interprété comme un 0.
    On a donc un signal binaire: allumé/éteint (on/off)
  - Ce montage contient une résistance de 10 kΩ (soit 10000 Ω), qu'on appelle pull-down (littéralement tirer-en- bas). Cette résistance permet de tirer le potentiel vers le bas (pull-down). En français, on appelle aussi ceci un rappel au moins.
  - **Avec une résistance pull-down, par défaut, l'entrée sur la broche est égale à 0 (LOW)**



Projet 5 : les entrées numériques

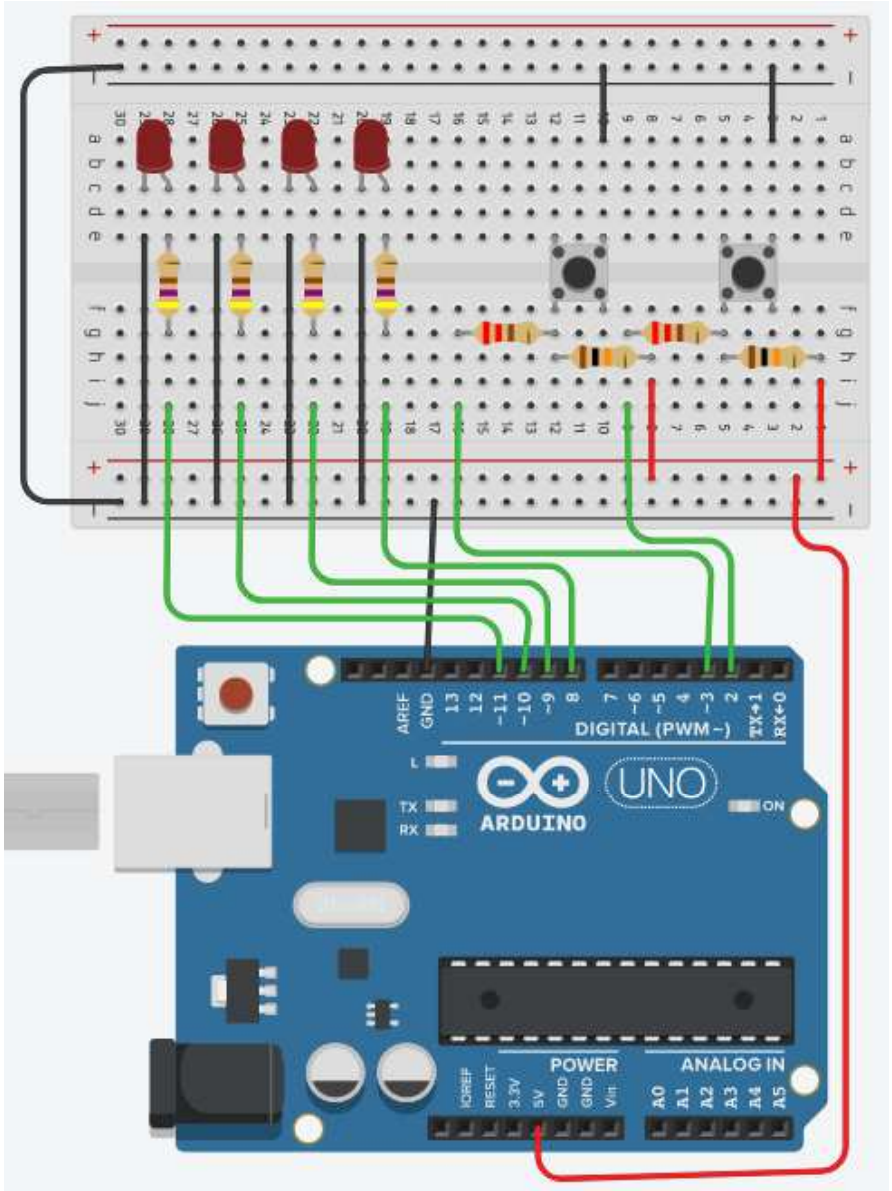
- Résistance Pull-Up
  - Si on le compare au schéma d'un circuit avec une résistance pull- down, on constate une inversion entre la terre et le +5V.
  - En effet, lorsque le circuit est ouvert, une tension de +5V est appliquée à l'entrée de la broche 4.
  - Lorsqu'on appuie sur le bouton poussoir, le courant électrique va passer par le chemin offrant le moins de résistance, soit directement par la masse (GND), sans passer par l'entrée de la broche 4.
  - Le fonctionnement est donc inversé par rapport à la résistance pull-down
  - **Avec une résistance pull-up, par défaut, l'entrée sur la broche est égale à 1 (HIGH)**



Projet 5 : les entrées numériques

- Données d'entrée :
  - 4 LED rouge + 2 interrupteur poussoir
  - Interrupteurs à connecter sur les pin 2 et 3 de la carte
  - LED à connecter sur les pin 8 à 11 de la carte
  - Résistances de limitation de courant de 330 Ω pour les LED
  - Résistances de 10 K Ω montées en pull-up
  - Résistances de 220 Ω en série avec les entrées

- Travail à effectuer:
  - Un interrupteur pour incrémenter l'autre pour décrémenter un compteur (valeurs de 0 à 4)
  - Allumer les LED en fonction de la valeur du compteur (Nbre à allumer)



- Fonction : int digitalRead()
  - **Description**
    - Lit l''état (= le niveau logique) d'une broche précise en entrée numérique, et renvoie la valeur HIGH (HAUT en anglais) ou LOW (BAS en anglais)**.**
  - **Syntaxe**
    - digitalRead(broche);
  - **Paramètres**
    - ***broche:*** le numéro de la broche numérique que vous voulez lire. (int)
  - **Valeur retournée**
    - Renvoie la valeur HIGH (HAUT en anglais) ou LOW (BAS en anglais) (int)

Projet 5 : les entrées numériques

- Rappel des fonctions de base :
  - **pinMode(*broche, mode*);** : Configure la broche spécifiée pour qu'elle se comporte soit en entrée, soit en sortie 0=entrée, 1=sortie.
  - **digitalWrite(*broche, valeur*);** : Met un niveau logique HIGH (1) ou LOW (0) sur une broche numérique définie en sortie.
  - **delay (ms);** : Réalise une pause dans l'exécution du programme pour la durée en millisecondes indiquée en paramètre.

Projet 5 : les entrées numériques

- Code du projet 6 : déclarations et setup()

```
// Les inputs sont des pull_up
// l'etat actif sera donc LOW

// Declarations des boutons
const int btp = 2;  // le bouton plus
const int btm = 3;  // le bouton moins

// Declaration des LED
const int led1 = 8;
const int led2 = 9;
const int led3 = 10;
const int led4 = 11;

// Declaration des variables
int bt;     // lecture des boutons
int memp;   // memoire de l'etat de btp
int memm;   // memoire de l'etat de btm
int cpt;    // compteur de 0 à 4 car 4 LED
```

```
void setup()
{
  // Declaration des entrées
  pinMode(btp, INPUT);
  pinMode(btm, INPUT);

  //Declaration des sorties
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);

  // Initialisation des variables
  memp = HIGH;  // etat bouton btp relaché
  memm = HIGH;  // etat bouton btm relaché
  cpt = 0;      // toutes les LED sont eteintes au depart
}
```

Projet 5 : les entrées numériques

- Code du projet 6 : loop()

```
38  void loop()
39  {
40    // acquisition du bouton plus
41    bt = digitalRead(btp);
42
43    // On incremente le compteur si appui sur btp detecté
44    if(bt != memp) {
45      if(bt == LOW) cpt++;
46      memp = bt;
47
48      // On teste la limite sup pour ne pas faire deborder cpt
49      if(cpt > 4) cpt = 4;
50    }
51
52    // acquisition du bouton moins
53    bt = digitalRead(btm);
54
55    // On decremente le compteur si appui sur btm detecté
56    if(bt != memm) {
57      if(bt == LOW) cpt--;
58      memm = bt;
59
60      // On teste la limite inf pour ne pas faire deborder cpt
61      if(cpt < 0) cpt = 0;
62    }
```

```
64    // Allumage des LED en fonction de la valeur de cpt
65    if(cpt > 0)
66      digitalWrite(led1, HIGH);   // LED1
67    else
68      digitalWrite(led1, LOW);
69
70    if(cpt > 1)
71      digitalWrite(led2, HIGH);   // LED2
72    else
73      digitalWrite(led2, LOW);
74
75    if(cpt > 2)
76      digitalWrite(led3, HIGH);   // LED3
77    else
78      digitalWrite(led3, LOW);
79
80    if(cpt > 3)
81      digitalWrite(led4, HIGH);   // LED4
82    else
83      digitalWrite(led4, LOW);
84
85    delay(100);   // 10 boucles par seconde environ
86  }
```

Projet 5 : les entrées numériques

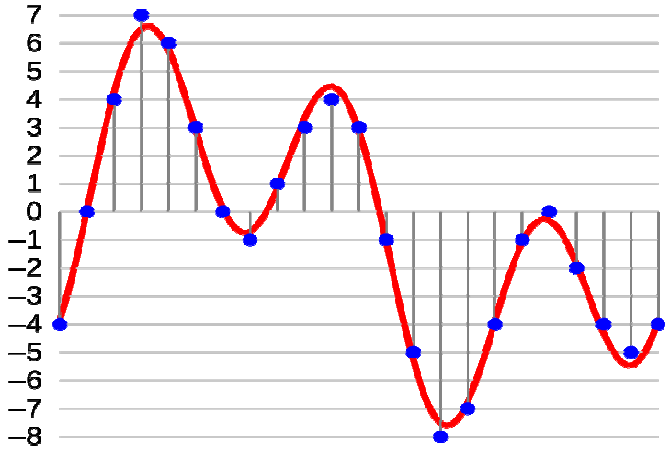
- Déparasiter à l'aide de condensateurs
  - Vous l'avez sans doute constaté: malgré des pull-up, les boutons-poussoirs sont peu précis. En effet, parfois, deux LEDs s'allument ou s'éteignes pour une pression sur le bouton. Cela est dû au fait que le bouton- poussoir n'est mécaniquement pas parfait. Lorsqu'on appuie dessus, le signal n'est pas forcément propre. Pendant quelques millisecondes, le signal va passer de 0V à 5V plusieurs fois avant de se stabiliser.
  - Il y a moyen de déparasiter le bouton-poussoir et d'absorber ces rebonds en montant en parallèle du bouton- poussoir un condensateur de faible capacité (10nF).

Condensateur céramique

Condensateur électrolytique

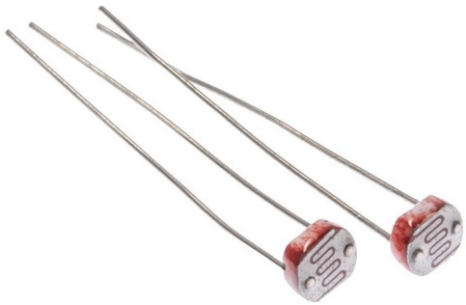
Projet 6 : les entrées analogiques

- Entrées analogiques
  - L'Arduino Uno possède 6 entrées analogiques, numérotées A0 à A5. En réalité, le microcontrôleur n'est pas capable de comprendre un signal analogique. Il faut donc d'abord le convertir en signal numérique par un circuit spécial appelé convertisseur analogique/numérique. Ce convertisseur va échantillonner le signal reçu sous la forme d'une variation de tension et le transformer en valeurs comprises entre 0 et 1023.
  - Attention à ne pas faire entrer une tension supérieure à 5V, ce qui détruirait l'Arduino



Projet 6 : les entrées analogiques

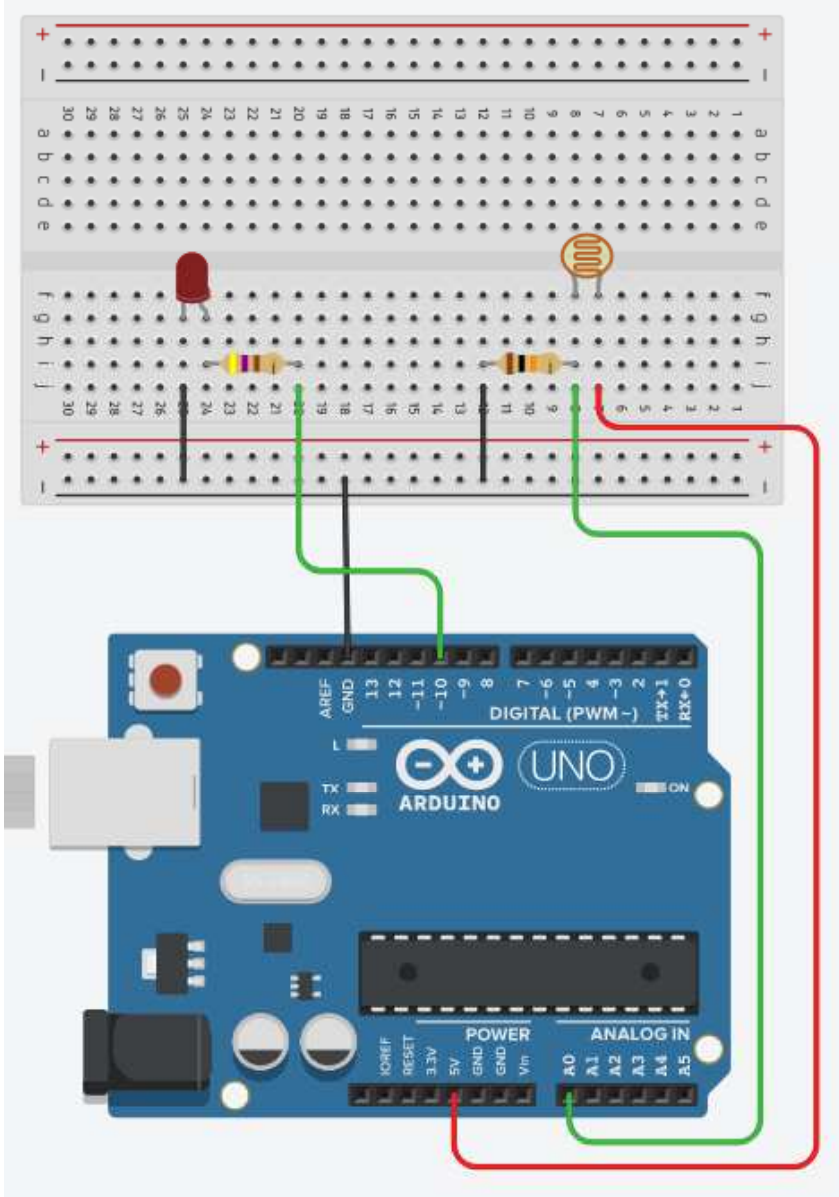
- La photorésistance
  - Une photorésistance est un composant électronique dont la résistance varie en fonction de l'intensité lumineuse. Plus la luminosité est élevée, plus la résistance est basse.
    On peut donc l'utiliser comme capteur lumineux pour:
    - Mesure de la lumière ambiante pour une station météo.
    - Détecteur de lumière dans une pièce.
    - Détecteur de passage …
  - Les photorésistances sont peu chère mais aussi peu précises on les utilisera donc pour mesurer une variation de lumière mais pas pour valeur une valeur de luminosité précise.



Projet 6 : les entrées analogiques

- Données d'entrée :
  - 1 LED rouge + 1 photorésistance
  - LED à connecter sur les pin 10 de la carte (sortie PWM)
  - Résistances de limitation de courant de 330 Ω pour les LED
  - Résistances de 10 K Ω montée en diviseur de tension avec la photodiode
  - Tension divisée connectée sur A0
  - Temporisation : 100 ms

- Travail à effectuer:
  - Conversion A/N de la tension sur A0
  - Faire varier la luminosité de la LED en proportion avec la mesure en utilisant le PWM
  - Utiliser le moniteur série pour observer les valeurs des variables



Projet 6 : les entrées analogiques

- Fonction : analogRead()
  - **Description**
    - Lit la valeur de la tension présente sur la broche spécifiée. La carte Arduino comporte 6 voies connectées à un convertisseur analogique-numérique 10 bits. Cela signifie qu'il est possible de transformer la tension d'entrée entre 0 et 5V en une valeur numérique entière comprise entre 0 et 1023.
    - Une conversion analogique-numérique dure environ 100 µs pour convertir l'entrée analogique, et donc la fréquence maximale de conversion est environ de 10 000 fois par seconde.
  - **Syntaxe**
    - analogRead(broche_analogique);
  - **Paramètres**
    - ***broche_analogique :*** le numéro de la broche analogique sur laquelle il faut convertir la tension analogique appliquée (0 à 5 sur la plupart des cartes Arduino)
  - **Valeur retournée**
    - valeur int (0 to 1023) correspondant au résultat de la mesure effectuée

Projet 6 : les entrées analogiques

- Fonction : Serial.begin()
  - **Description**
    - Fixe le débit de communication en nombre de caractères par seconde (l'unité est le baud) pour la communication série.
    - Pour communiquer avec l'ordinateur, utiliser l'un de ces débits : 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.
  - **Syntaxe**
    - Serial.begin(debit);
  - **Paramètres**
    - ***int debit:*** debit de communication en caractères par seconde (ou baud). Pour communiquer avec l'ordinateur, utiliser l'un de ces débits : 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 11520
  - **Valeur retournée**
    - Aucune

Projet 6 : les entrées analogiques

- Fonction : Serial.print()
  - **Description**
    - Affiche les données sous le port série sous forme lisible pour les humains (texte ASCII). Cette instruction peut prendre plusieurs formes.
      - Les nombres entiers sont affichés en utilisant les caractères ASCII pour chaque chiffre.
      - Les nombres à virgules (float) sont affichés de la même façon sous forme de caractères ASCII pour chaque chiffre, par défaut avec 2 décimales derrière la virgule.
      - Les valeurs de type byte sont affichés sous la forme d'un caractère ASCII.
      - Les caractères et les chaînes sont affichés tels que. .
  - **Syntaxe**
    - Serial.print(val, format);
  - **Paramètres**
    - ***val:*** la valeur à afficher. N'importe quel type de données.
    - ***format :*** spécifie la base utilisée(pour les nombres entiers) ou le nombre de décimales (pour les nombres de type float)
  - **Valeur retournée**
    - Aucune

- Fonction : Serial.println()
  - **Description**
    - Affiche les données sur le port série suivi d'un caractère de "retour de chariot" (ASCII 13, or '\r') et un caractère de "nouvelle ligne" (ASCII 10, or '\n').
      Cette instruction a par ailleurs la même forme que l'instruction Serial.print().
  - **Syntaxe**
    - Serial.println(val, format);
  - **Paramètres**
    - ***val:*** la valeur à afficher. N'importe quel type de données.
    - ***format :*** spécifie la base utilisée(pour les nombres entiers) ou le nombre de décimales (pour les nombres de type float)
  - **Valeur retournée**
    - Aucune

Projet 6 : les entrées analogiques

- Code du projet

```
// Declaration des pin
const int led = 10;     // LED sur une sortie PWM
const int light = 0;    // Photorésistance sur A0

// Déclaration des variables
int light_val;      // Valeur numérique de lum ambiante
int led_val;        // Valeur de lum de la LED

void setup()
{
  pinMode(led, OUTPUT);

  Serial.begin(9600);   // Initialisation de la liaison de debug
}

void loop()
{
  // Acquisition de la lum ambiante
  light_val = analogRead(light);

  //Affichage de la valeur de "light_val" dans la console
  Serial.print("light_val = ");
  Serial.print(light_val);

  // light_val peut varier de 0 à 1023
  // led_val va au maximum à 255 il faut donc diviser par 4
  led_val = light_val / 4;

  Serial.print(" Led_val = ");
  Serial.println(led_val);


  // Affichage de la LED
  analogWrite(led, led_val);

  delay(100); // Wait for 100 millisecond(s)
}
```

MARTIGNAS

Merci pour votre attention

Avez-vous des questions ?