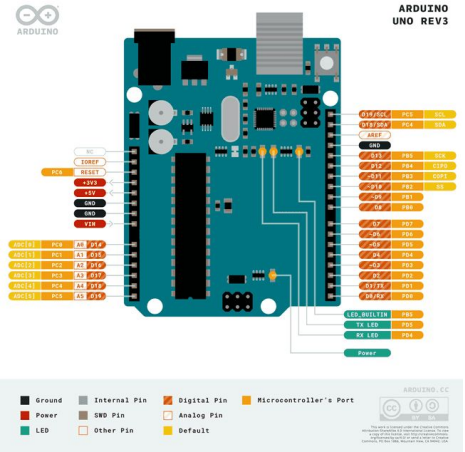
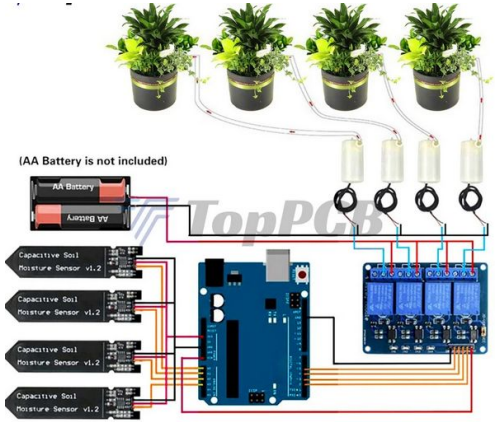
Electrolab #2

Cem, Joël & Malik

Programme du jour

- Retour sur la première session. Eventuelles questions.
- Explications des différents pins.
- C’est quoi des relais ?
- Découverte des capteurs et actuateurs pour le projet “Arrose moi ça ! ”



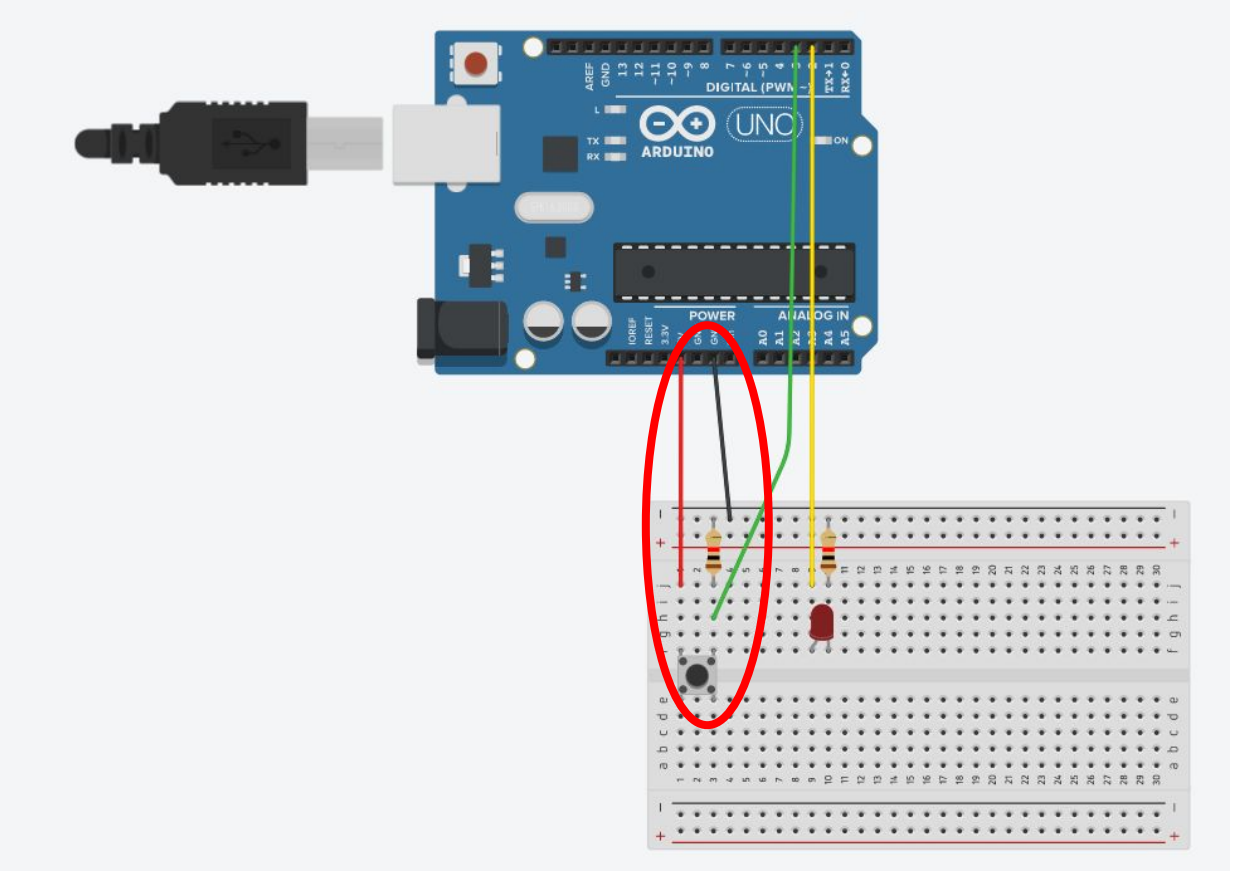


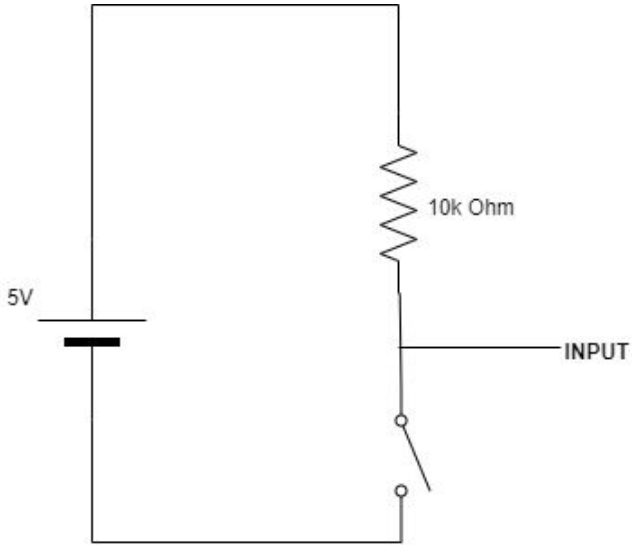
Retour sur la première session

20min.

Rappel

- Il doit y avoir un **chemin complet** à partir de la source d'énergie (alimentation) au point de moindre énergie (masse **- GND**) pour faire un circuit. S'il n'y a pas de chemin pour que l'énergie voyage, le circuit ne fonctionnera pas. **Circuit fermé**.
- Le courant électrique cherchera à emprunter le chemin ayant la plus petite résistance (flemmard). **En reliant l'alimentation directement à la masse, sans résistance, il y aura un court circuit.** Le courant n'est plus limité.
- Il faut qu'il y est une source d'énergie. Le pin mode **INPUT** "écoute" uniquement. Il ne produit pas d'électricité.





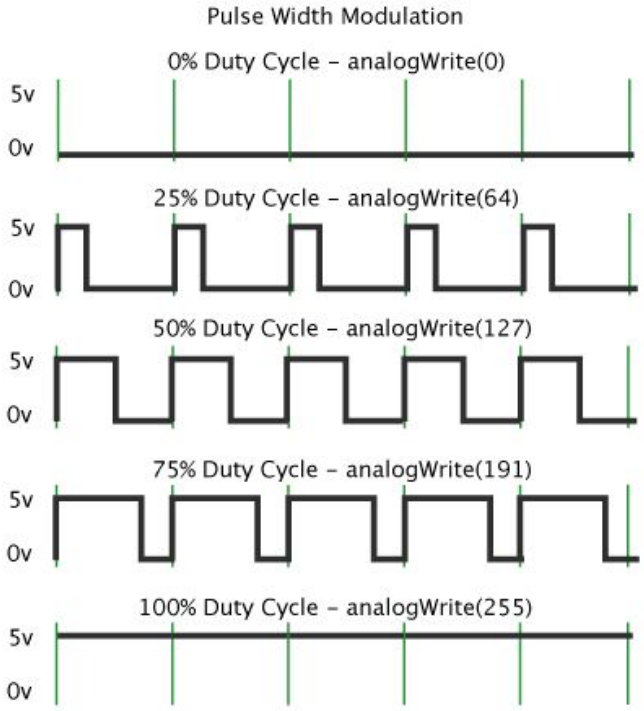
- Il faut que le courant vienne de quelque part.➜ **Circuit fermé**.
- Résistance ➜ **Court circuit**

```
// Déclaration et initialisation de la variable qui stocke l'état de l'interrupteur (bouton)
int switchState = 0;

// Déclaration et définition des broches utilisées
const int button = 3; // Broche où le bouton est connecté
const int LED = 2;    // Broche où la LED est connectée

// Variable pour stocker l'état actuel de la LED (1 = allumée, 0 = éteinte)
int ledState = 0;


void setup() {
  Serial.begin(9600);          // Démarre la communication série à 9600 bauds
  pinMode(button, INPUT);      // Configure la broche du bouton comme entrée
  pinMode(LED, OUTPUT);        // Configure la broche de la LED comme sortie
}

void loop() {

  // Lecture de l'état du bouton et stockage dans la variable switchState
  switchState = digitalRead(button);

  // Affichage de l'état du bouton sur le moniteur série
  Serial.println(switchState);

  // Si le bouton est pressé (switchState == HIGH)
  if (switchState == HIGH){
    // Si la LED est éteinte (ledState == 0)
    if (ledState == 0){
      digitalWrite(LED, HIGH); // Allume la LED
      ledState = 1;            // Met à jour l'état de la LED à "allumée"
      Serial.println("ON");
      delay(200);              // Retarde de 200 millisecondes
    }
    // Si la LED est déjà allumée
    else{
      digitalWrite(LED, LOW);  // Éteint la LED
      ledState = 0;            // Met à jour l'état de la LED à "éteinte"
      Serial.println("OFF");
      delay(200);              // Retarde de 200 millisecondes
    }
  }
}
```

Des questions sur la première session?

Les différents pins

15min. + 10min. de pratique



Pins **PWM ~:** permet de faire la **marche-arrêt** pour simuler des tensions entre le Vcc complet de la carte (**5 V**) et la hors tension (**GND**).

En changeant la partie du temps que le signal passe par rapport au temps que le signal passe hors tension.

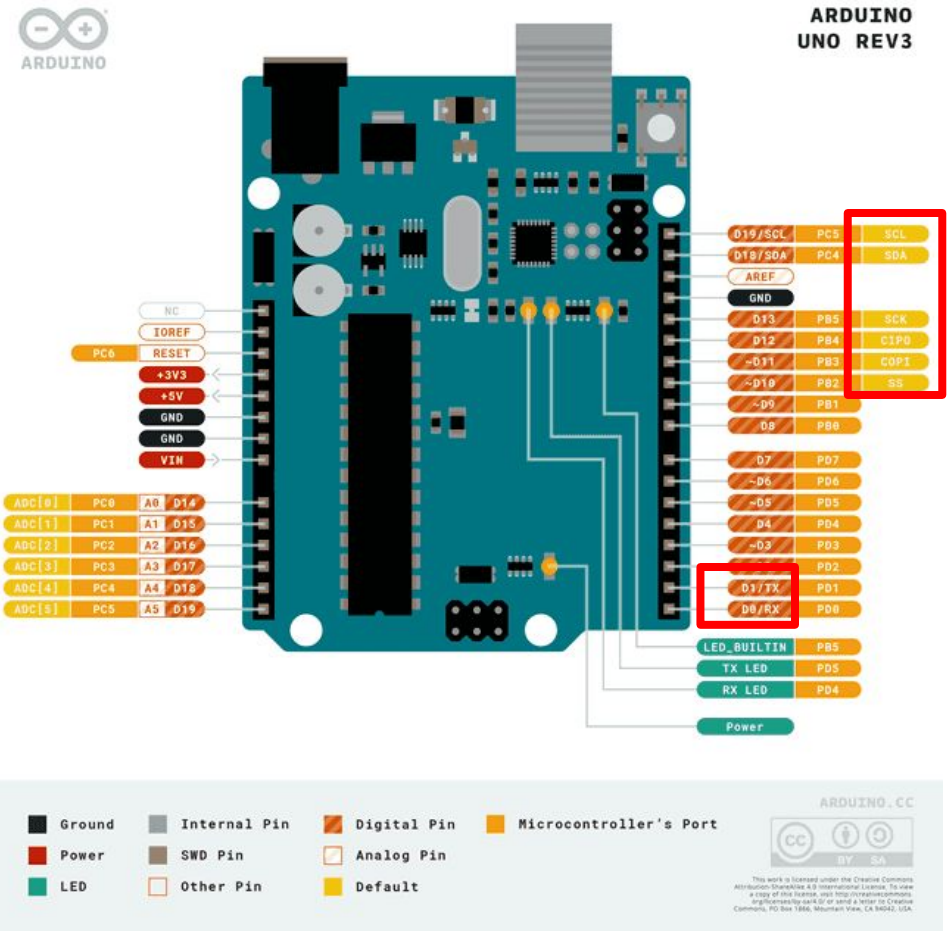
Permet d'obtenir des valeurs analogiques variables, on peut moduler la largeur d'impulsion.

Si vous répétez cette configuration de marche-arrêt assez rapidement avec une LED par exemple, **le résultat est comme si le signal est une tension constante entre 0 et Vcc contrôlant la luminosité de la LED**.

Est-ce qu'on en a besoin pour notre projet ?

NON

Pins de communications



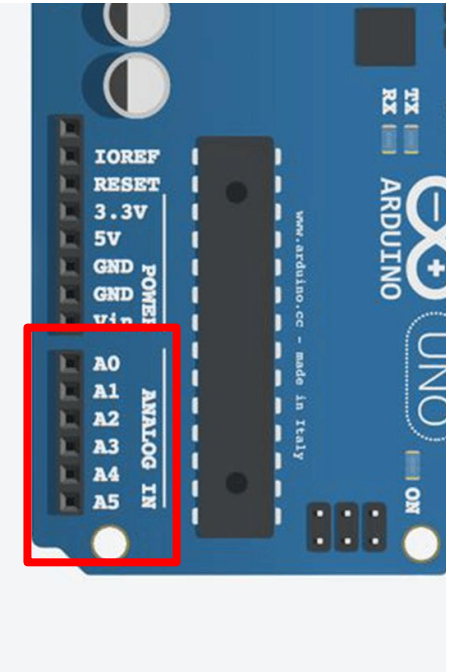
Pins de communications

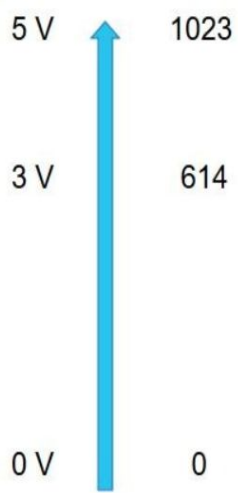
- **Communication Série - Pins 0 (RX) et 1 (TX):** Ces pins sont utilisés pour la communication série via l'UART (Universal Asynchronous Receiver/Transmitter). Le pin RX est utilisé pour recevoir des données, et le pin TX est utilisé pour transmettre des données. Cette communication est essentielle pour envoyer des données à un ordinateur ou à d'autres périphériques série comme des modules GPS, des modules Bluetooth, etc.
- **Communication I2C - SCL & SCA:** I^2C, ou Inter-Integrated Circuit, est un bus série qui utilise deux lignes pour la communication. SDA (Serial Data Line) est la ligne de données, et SCL (Serial Clock Line) est la ligne d'horloge. Cette interface est utilisée pour connecter divers capteurs et périphériques à faible vitesse de communication comme des écrans LCD, des capteurs de température, etc..
- **Communication SPI - Pins 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK):** Le SPI (Serial Peripheral Interface) est un protocole de communication rapide qui utilise quatre fils—MOSI, MISO, SCK et SS—pour connecter un microcontrôleur maître à un ou plusieurs périphériques esclaves. Il est particulièrement apprécié pour sa vitesse élevée et sa capacité à gérer des communications avec des dispositifs nécessitant un transfert de données intensif, comme des écrans ou des cartes mémoire.

Est-ce qu'on en a besoin pour notre projet ?

NON

Pins analogiques





- **A0** à **A5** : Ces pins servent à lire une tensions analogiques. Ils peuvent convertir une entrée analogique (de 0V à 5V) en une valeur entre 0 et 1023

- On est passé du **binaire** (pins numérique) à des valeurs **numérique** (pins analogiques)

- Convertit des Volts en valeur numérique (capteurs)

pinMode() →
ex. pinMode(A0, INPUT)

analogRead()

- Pour rappel, ***Capteurs*** *: convertissent des formes d'énergie en énergie électrique*

Est-ce qu'on en a besoin pour notre projet ?

OUI ! héhé

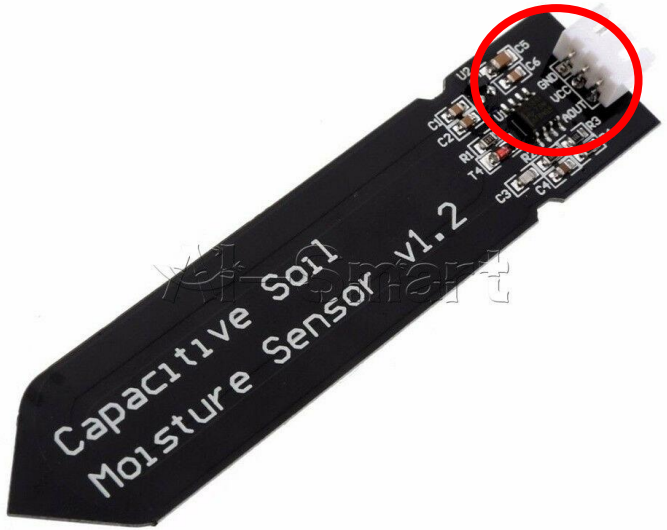
A v(n)ous ! #1 - Calibration des capteurs

- **Objectif**: interpréter les données du capteur d'humidités. **ET** déjà penser à une solution potentiel pour notre projet.

Matériel:

- Arduino Uno
- Capteur capacitive ➜ **GND, VCC, AOUT**
- Verre d'eau

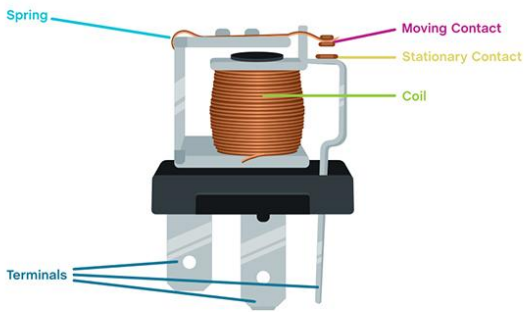
pinMode()
digitalRead()

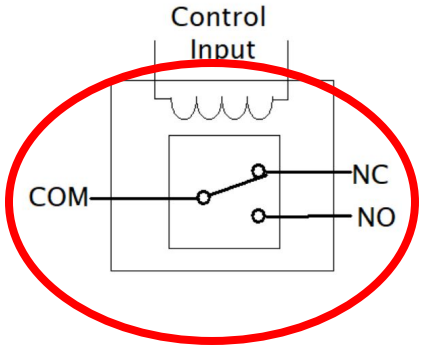


C’est quoi les relais ?

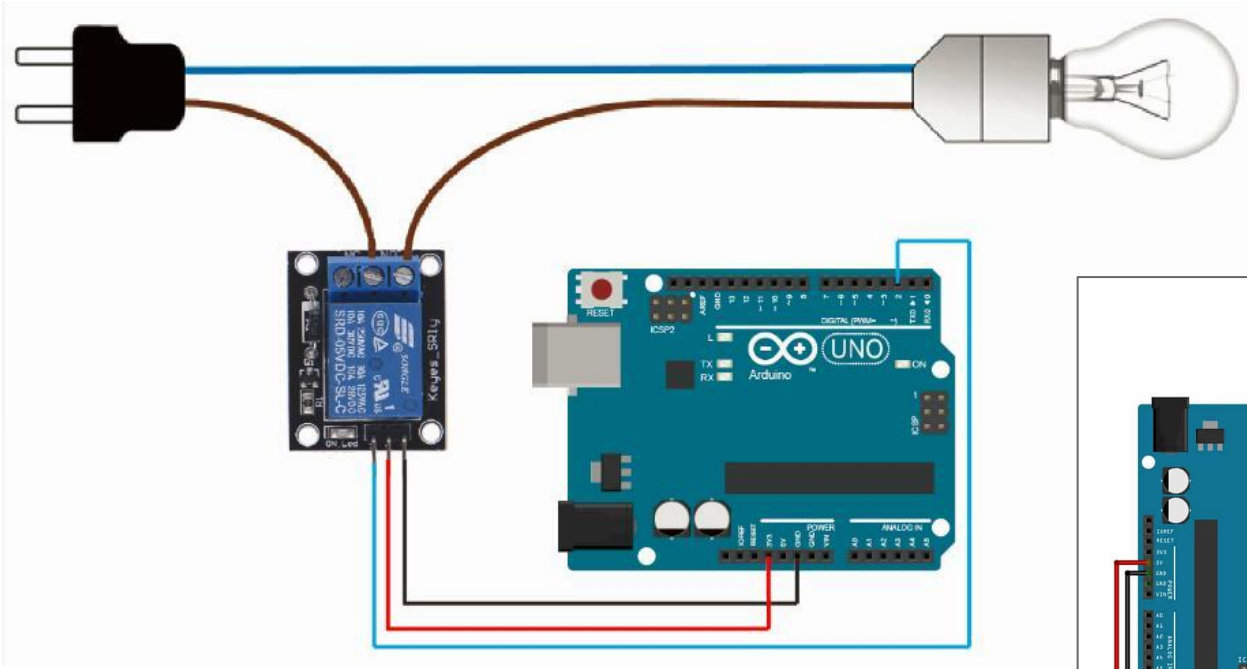
5min. + 20min. de pratique

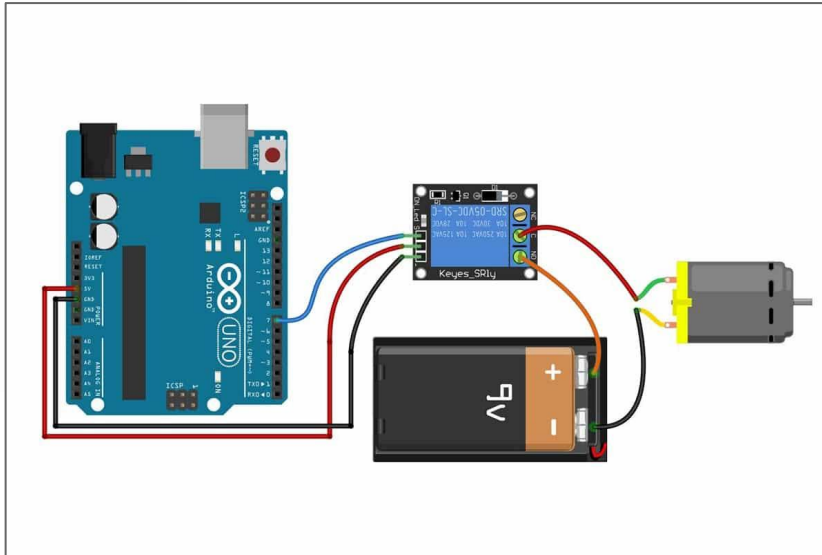
- Les relais permettent aux microcontrôleurs de **faible puissance** de gérer des circuits qui utilisent une **puissance beaucoup plus élevée** que ce que la carte peut gérer directement.
- Dans notre projet, l'Arduino n'a pas assez de puissance pour faire fonctionner 4 pompes à eau (même une).
- Les relais sont composés d'un électro-aimant qui déplace une minuscule planche métallique, appelée terminal **COM**, entre deux positions différentes **NC (Normally Closed)** terminal et **NO (Normally Opened)** terminal.

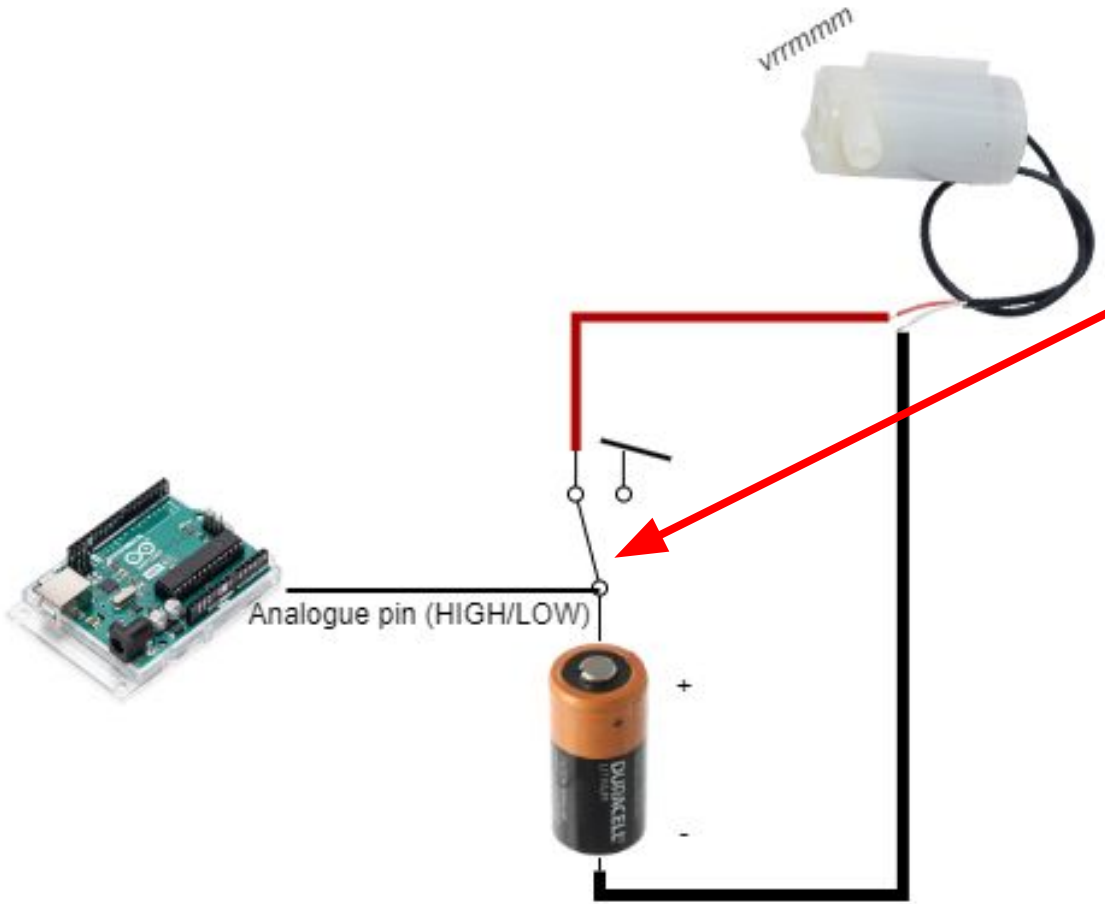




Exemple d'usage de relais





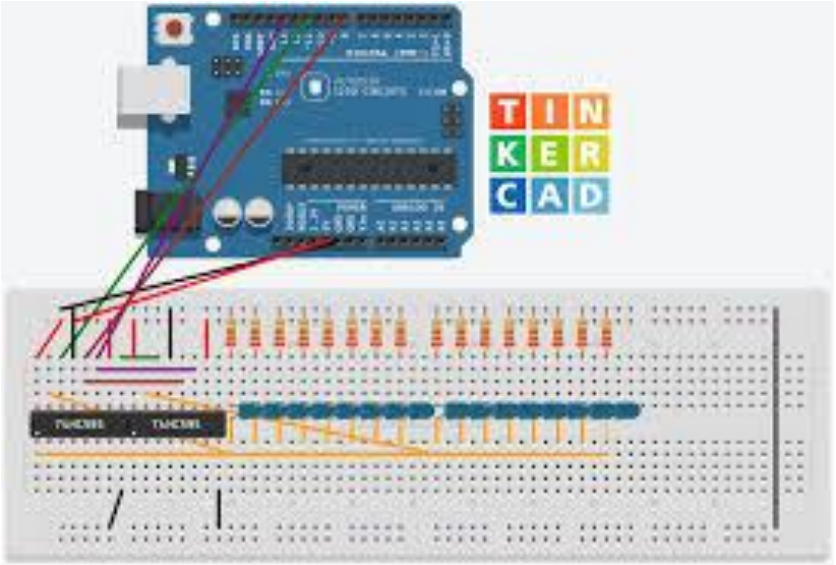


Chez nous c’est inversé !

(Why? En gros, l’Arduino n’a pas assez de puissance pour faire tourner à fond la bobine. Du coup, le module relais pompe du jus sur l’alimentation des pompes à eau. Et ceci, apparement, inverse le relais.)

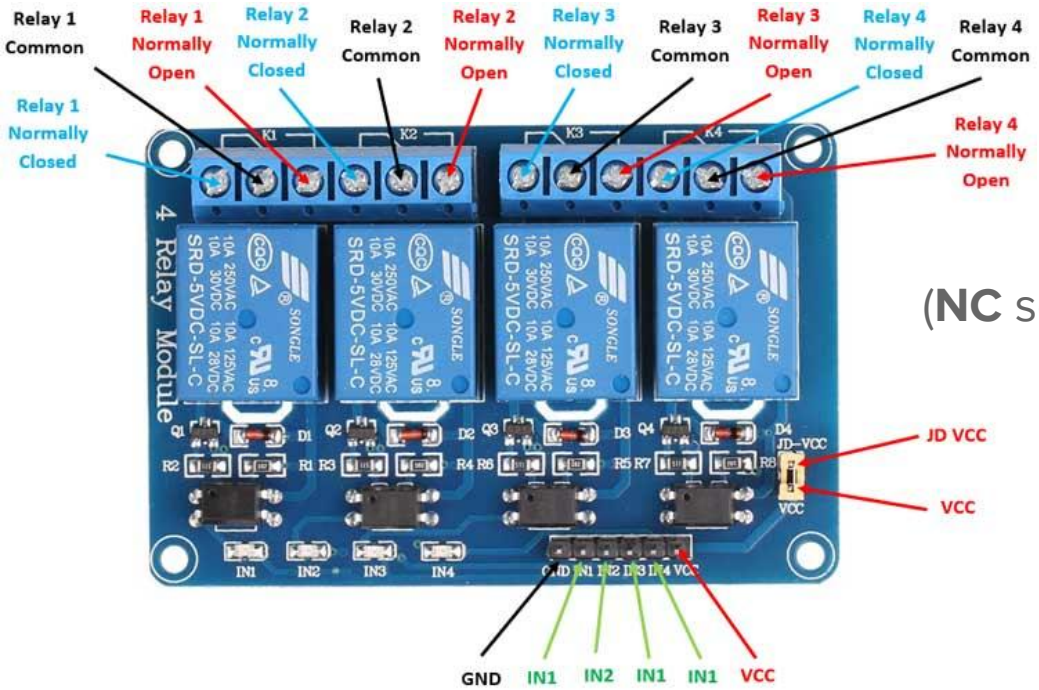
TINKERCAD

AUTODESK®
TINKERCAD®

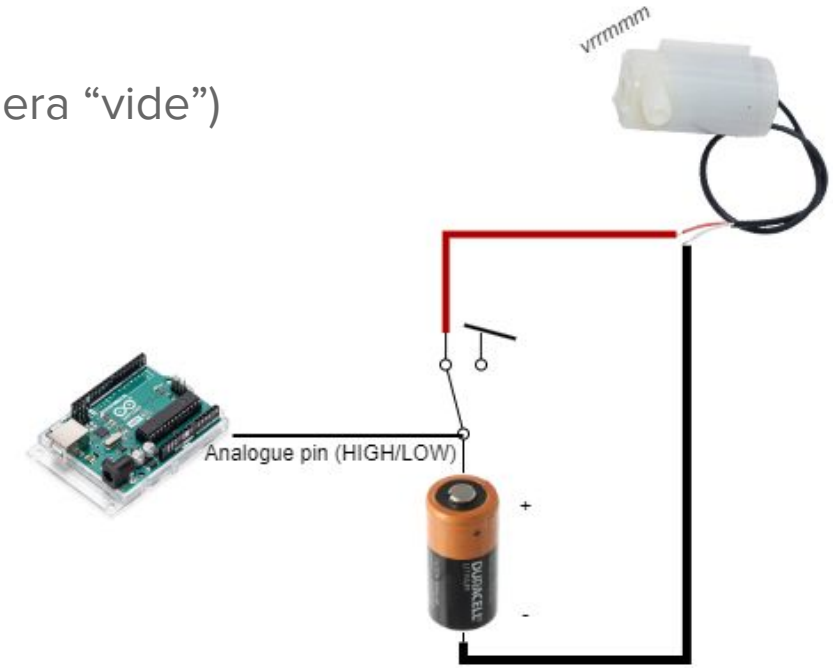


A vous ! #2 - Les pompes (en 3 groupes)

Objectif: faire tourner la pompe toute les 5 secondes pendant 10 secondes.



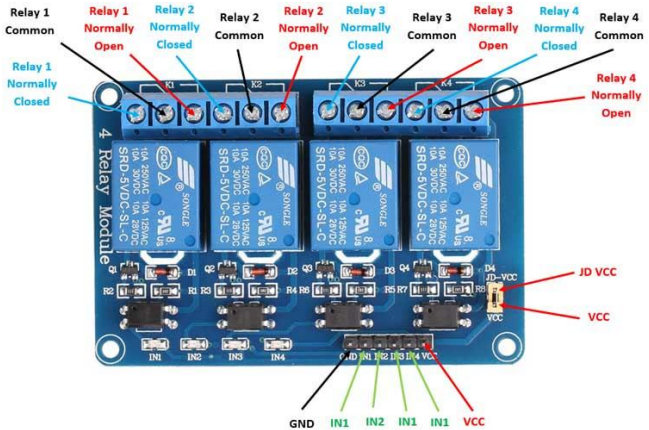
(**NC** sera “vide”)

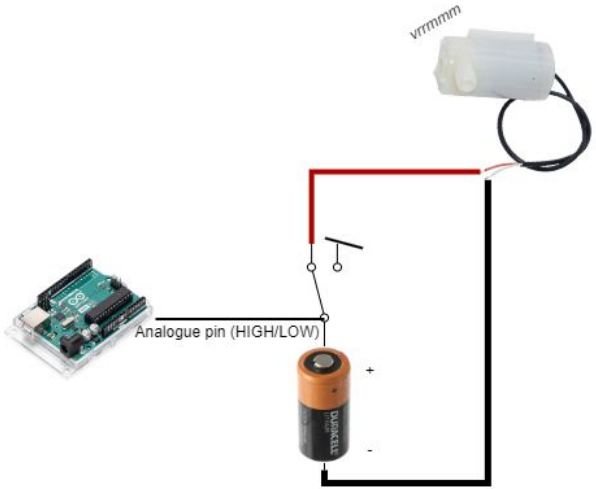


- **pinMode()**
- **digitalWrite()**
- Attention ! C’est inversé

A vous #3 (si le temps le permet)

Objectif: Faire tourner la pompe à eau quand le capteur d'humidité est hors de l'eau.





- **pinMode()**
- **digitalWrite()**
- **analogRead()**
- Attention ! C'est inversé

Semaine prochaine

- Brainstorming sur un système d’arrosage efficace et fiable.
- Prototype **complet** du système d’arrosage automatique.