

Programmation : Introduction, Simulation, Arduino IDE

Nom :

Prénom :

2025-2026

Programmation 1**Table des matières**

Introduction à la programmation.....	3
Introduction à la simulation TinkerCAD.....	14
Découverte d'un système électronique embarqué : Arduino.....	20
Découverte d'une carte type ESP32 (ou ESP8266).....	26
Découverte mBot.....	28
Simulation TinkerCAD.....	33
Analyser un code C++ Arduino.....	37
Programmation : Conditions et boucles.....	41

Dossiers ressources utiles :

nlardon.github.io : Programmation

Compte TinkerCAD : <https://www.tinkercad.com/joinclass/Q9YNQECWY>

Code classe : Q9YNQECWY

Pseudo :

Compétences :**E2 : Réalisation et maintenance de produits électroniques**

- C03 : Participer à un projet ;
- C07 : Réaliser des maquettes et prototypes ;
- C11 : Maintenir un système électronique ou réseau informatique.

E31 : mise en œuvre de réseaux informatiques

- C06 : Valider la conformité d'une installation ;
- C09 : Installer les éléments d'un système électronique ou informatique ;
- C10 : Exploiter un réseau informatique.

E32 : valorisation de la donnée et cybersécurité

- C01 : Communiquer en situation professionnelle (français/anglais) ;
- C04 : Analyser une structure matérielle et logicielle ;
- C08 : Coder.

Programmation : Introduction, Simulation, Arduino IDE
--

Introduction à la programmation

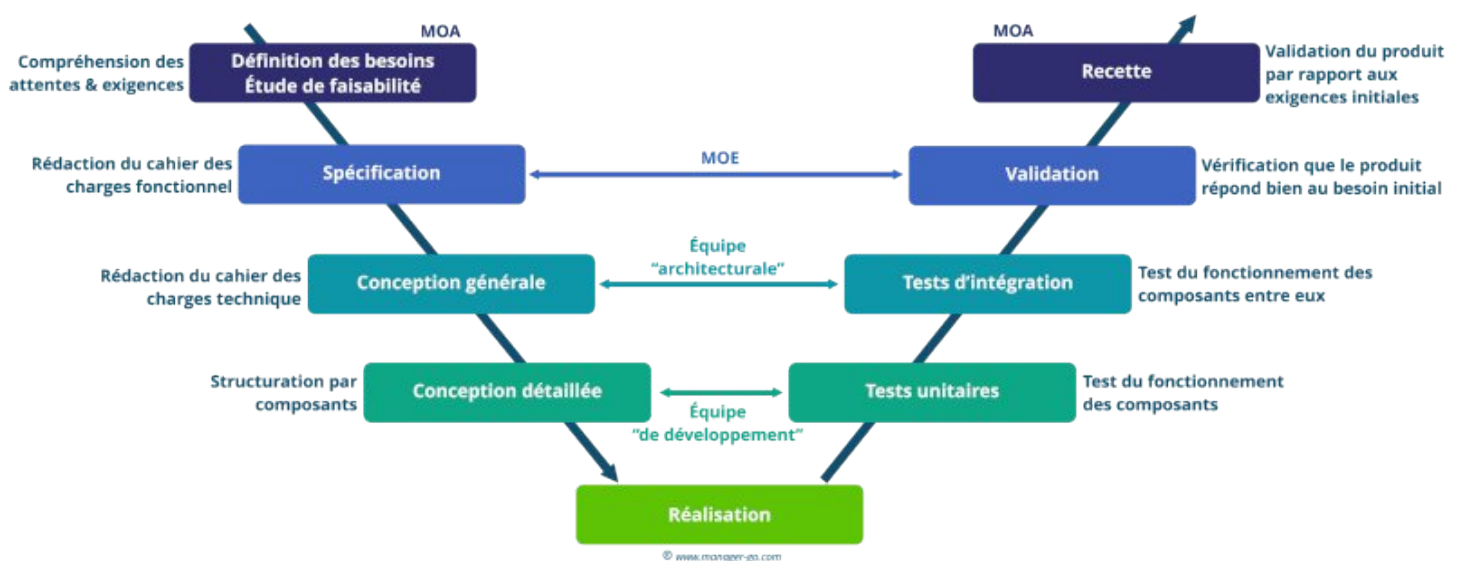
1. C'est quoi la programmation informatique ?

Dans la conception de systèmes informatiques, la programmation est la clef du développement informatique qui se réfère aux logiciels (software en anglais) et aux matériels (hardware en anglais).

La programmation implique le codage et l'écriture de langages informatiques. Le plus souvent, les développeurs sont des professionnels qui conçoivent ces programmes informatiques et ont des compétences solides pour la maîtrise des langages informatiques comme le PHP, Javascript, HTML, CSS, etc.

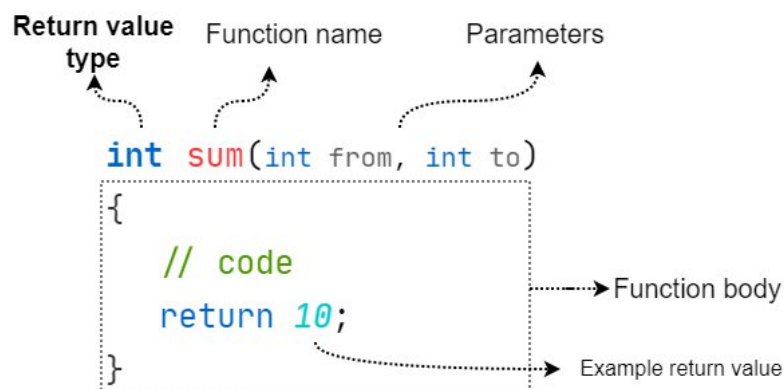
Lors de l'écriture d'un programme, vous devez utiliser un langage de programmation.

La programmation d'un système informatique comprend des étapes principales (exemple d'un cycle en V) :



Nous identifions généralement dans un code :

- les données que le programme va exécuter (ce sont les données d'entrée ou « parameters »)
- la méthode choisie (il s'agit d'un algorithme ou « fonction »)
- et le résultat (ce sont les données de sortie ou « return »).

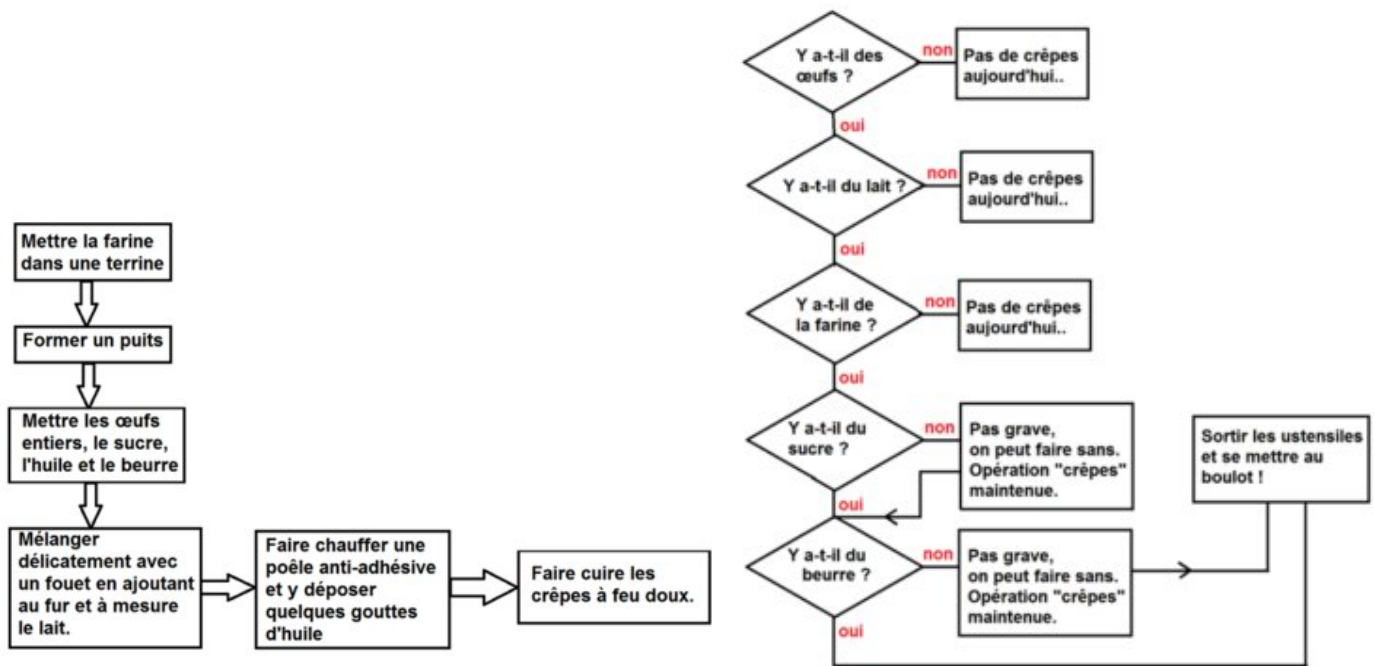


Programmation : Introduction, Simulation, Arduino IDE

2. Les algorithmes

Un algorithme est une suite finie et non ambiguë d'instructions et d'opérations permettant de résoudre une classe de problèmes.

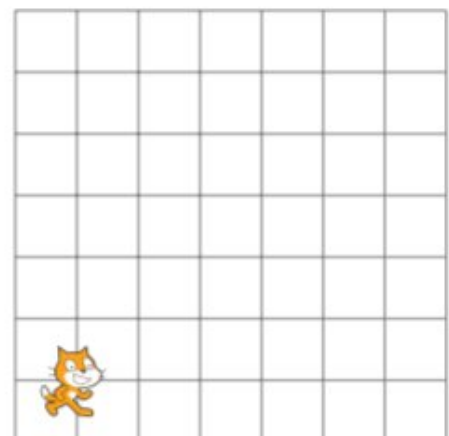
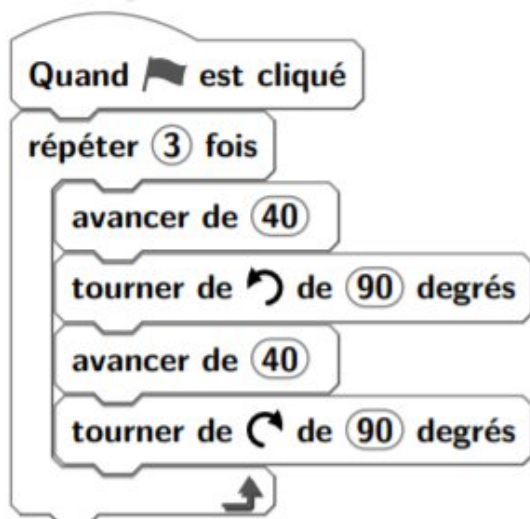
Exemple d'un algorithme graphique :



Exercice 1 : Les carreaux font 40 unités de large.

A l'aide du script ci-dessous à gauche, dessiner à droite le chemin du lutin-chat.

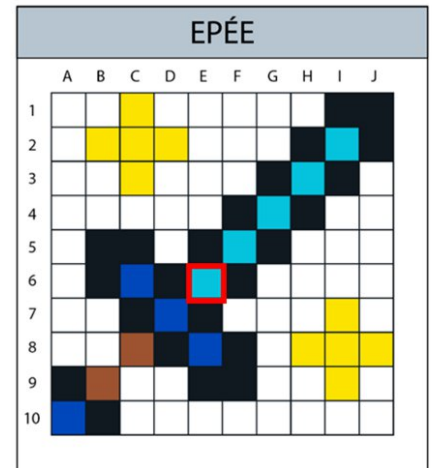
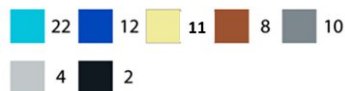
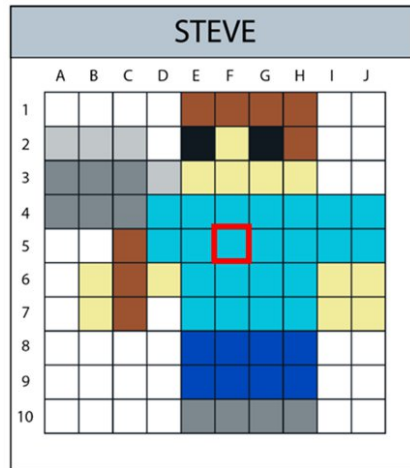
La position initiale du lutin-chat est à l'intersection des segments qu'il cache.



Programmation : Introduction, Simulation, Arduino IDE

Exercice 2 :

Donner les coordonnées de la case d'arrivée :



Programme 1 sur « STEVE » :

```

1 haut()
2 droite()
3 droite()
4 haut()
    
```

Programme 2 sur « STEVE » :

```

1 haut()
2 répète (3) {
3     droite()
4 }
5 haut()
    
```

Programme 3 sur « STEVE » : J4

```

1 répète (33) {
2     si couleur = 22 {
3         droite()
4     }
5 }
6 haut()
7 gauche()
    
```

Programme 4 sur « EPÉE »

```

1 fonction remue(){
2     gauche()
3     droite()
4     gauche()
5 }
6
7 fonction danse () {
8     remue()
9     gauche()
10 }
11
12 danse()
13 danse()
14 danse()
    
```

Exercice 3 :

Tester le site : <http://compute-it.toxicode.fr/>

Programmation : Introduction, Simulation, Arduino IDE

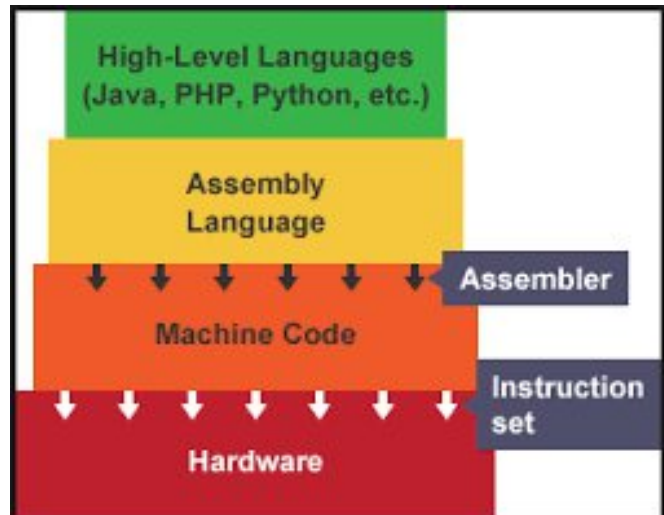
3. Initiation au langage machine et assembleur

Un système numérique ne compte pas sur une base 10 (de 0 à 9) mais sur une base 2 (0 ou 1).

Le langage machine est la suite de bits qui est interprétée par le processeur de l'ordinateur lors de l'exécution d'un programme. C'est le langage natif du processeur et le seul qui soit reconnu par celui-ci.

Un processeur ne peut pas comprendre un langage rentré par l'utilisateur, un programme python ou java par exemple.

Décimal	Binaire
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
Etc..	

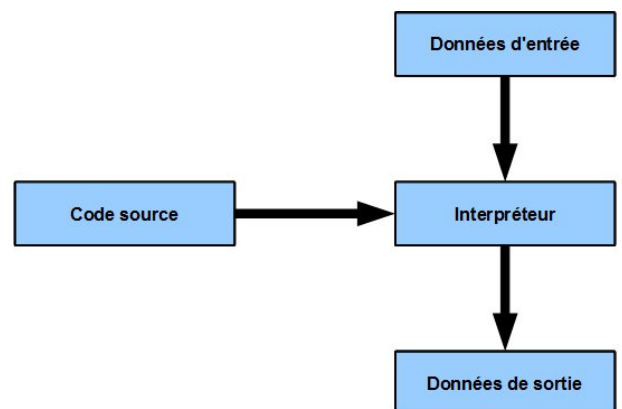


4. Langage interprété

Nous avons besoin

- D'un fichier contenant le code source,
- D'un interpréteur qui lit chaque ligne du code source et la transforme en code binaire lisible par l'OS qui lui se chargera de l'envoyer au processeur.
- Les lignes sont lues et traduites au fur et à mesure des besoins,
- Le code source doit être présent dans l'ordinateur,
- L'interpréteur doit lui aussi être présent dans l'ordinateur.

Exemples de langages interprétés : **Matlab, PHP, Python**



Programmation : Introduction, Simulation, Arduino IDE

5. Langage compilé

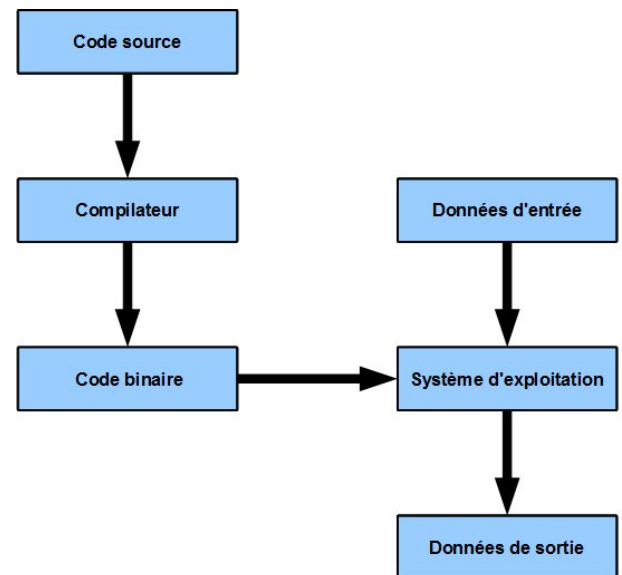


Sous Windows, cela donne un fichier avec l'extension « .EXE »

Pour exécuter le programme sur la machine, il n'y a besoin que de ce fichier exécutable (le code source n'est plus nécessaire).

A chaque modification du programme par le développeur, il faudra recompiler le programme.

Exemple de langage compilé : **C++**



6. Langage mixte

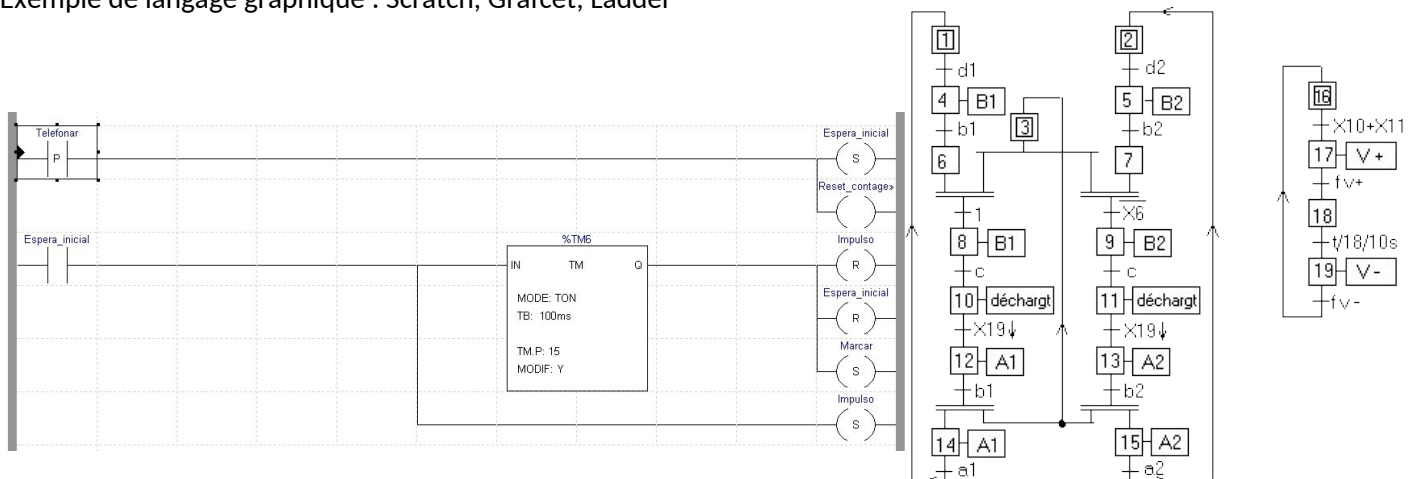


Exemple de langage mixte : **JAVA**

7. Langage graphique

Un langage de programmation graphique ou visuel est un langage de programmation dans lequel les programmes sont écrits par assemblage d'éléments graphiques. Sa syntaxe concrète est composée de symboles graphiques et de textes, qui sont disposés spatialement pour former des programmes.

Exemple de langage graphique : Scratch, Grafcet, Ladder



8. Développement en Assembleur

En langage assembleur, on peut rencontrer des calculs simples :

- addition, soustraction, multiplication
- des affectations (placer telle valeur de la mémoire vive dans tel registre et vice versa)
- et enfin des instructions de saut.

Syntaxe du langage :

Syntaxe	Signification	Syntaxe	Signification	Syntaxe	Signification
ADD	Addition	LDR	Affectation	HALT	Fin du programme
SUB	Soustraction	STR	Affectation		
MOV	Affectation	B	Aller à		

Exemple d'instructions

MOV R0, #17	Affecte la valeur 17 au registre R0
MOV R1, #3	Affecte la valeur 3 au registre R1
ADD R2, R0, #18	Ajoute 18 à la valeur du registre R0 et stocke le résultat dans R2
STR R0, 20	Place la valeur stockée dans le registre R0 à l'adresse mémoire 20
LDR R4, 20	Place la valeur stockée à l'adresse mémoire 20 dans le registre R4

Notez bien que si vous voulez désigner un nombre, le # est nécessaire, sinon vous ferez référence à une adresse de la mémoire.

Accès au simulateur <https://www.peterhigginson.co.uk/AQA>

Exercice 1 : Écrire en langage assembleur les instructions suivantes :

- Affecter la valeur 3 au registre R0
- Additionner 7 avec la valeur du registre R0 et stocker le résultat dans R1
- Placer la valeur stockée dans le registre R0 à l'adresse mémoire 30
- Soustraire 10 à R0 puis stocker le résultat dans le registre R2

Instructions :

Programmation : Introduction, Simulation, Arduino IDE

9. Développement en langage interprété Python

Ce langage de programmation présente de nombreuses caractéristiques intéressantes :

- Il est multiplateforme. C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.
- Il est gratuit. Vous pouvez l'installer sur autant d'ordinateurs que vous voulez.
- C'est un langage de haut niveau. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.
- C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++.
- Il est relativement simple à prendre en main.

Accès au simulateur : <https://pythontutor.com/> -> Python -> Edit code

Exemples de code en Python

```
a = 1
b = 2
c = -4

delta = b ** 2 - 4 * a * c
print('Le discriminant est :')
print(delta)
```

```
prenom = input('Ton prénom : ')
age = int(input('Ton age : '))
annee = 2023

b = annee - age

print(prenom)
print(' tu est né(e) en ')
print(b)
```



Exercice : Écrire un programme qui demande l'intensité et la tension puis retourne la puissance. ($P = U \times I$)

Programmation : Introduction, Simulation, Arduino IDE

10. Développement en langage compilé C++

C++ est un langage de programmation compilé. Ses bonnes performances, et sa compatibilité avec le C en font un des langages de programmation les plus utilisés dans les applications où la performance est critique.

Le plus grand avantage de ce langage est que les langages de programmation contemporains en sont quasiment tous plus ou moins dérivés. Ainsi, une fois que l'on connaît le langage C, les autres langages deviennent beaucoup plus simples.

Un inconvénient du langage C++ est la nécessité d'avoir un compilateur, qui va compiler le code pour une architecture spécifique. Par exemple d'un code source il faudra le compiler pour Windows, pour iOS, pour MacOS, pour Android, ...

Exemple 1 : un premier exemple

```
#include <iostream>

using namespace std;

int main()
{
    string nom ;

    cin >> nom;

    cout << "BONJOUR";

    cout << nom;

    return 0;
}
```

Exercice : Écrire un programme qui demande l'intensité et la tension puis retourne la puissance. ($P = U \times I$)

```
#include <iostream>

using namespace std;

int main ()
{

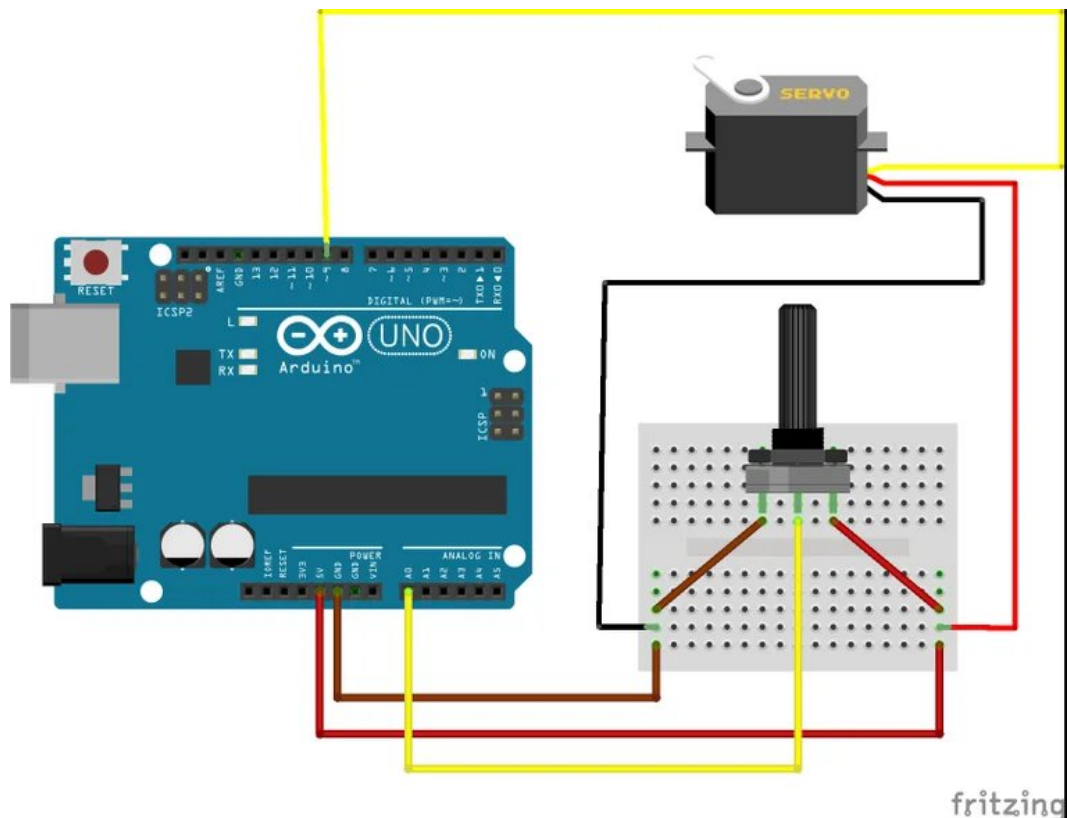
}

}
```

Programmation : Introduction, Simulation, Arduino IDE

Particularité du C++ utilisé sur l'IDE Arduino

Exemple :



```
#include <Servo.h>
```

```
Servo myservo; // create servo object to control a servo
```

```
int potpin = A0; // analog pin used to connect the potentiometer
```

```
int val; // variable to read the value from the analog pin
```

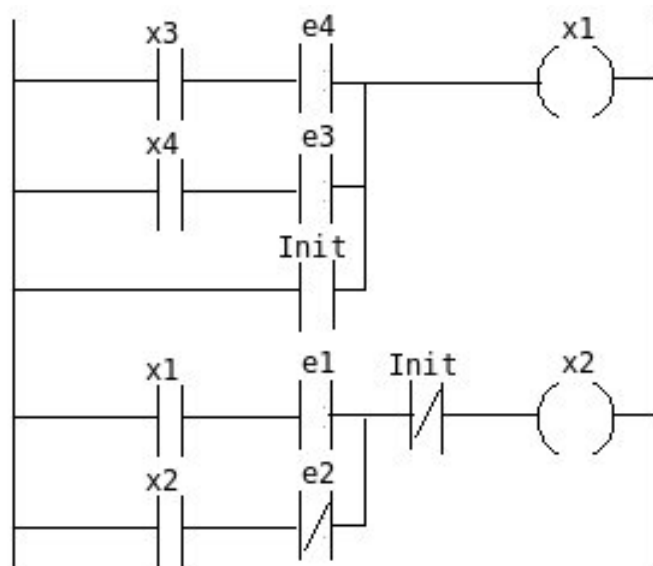
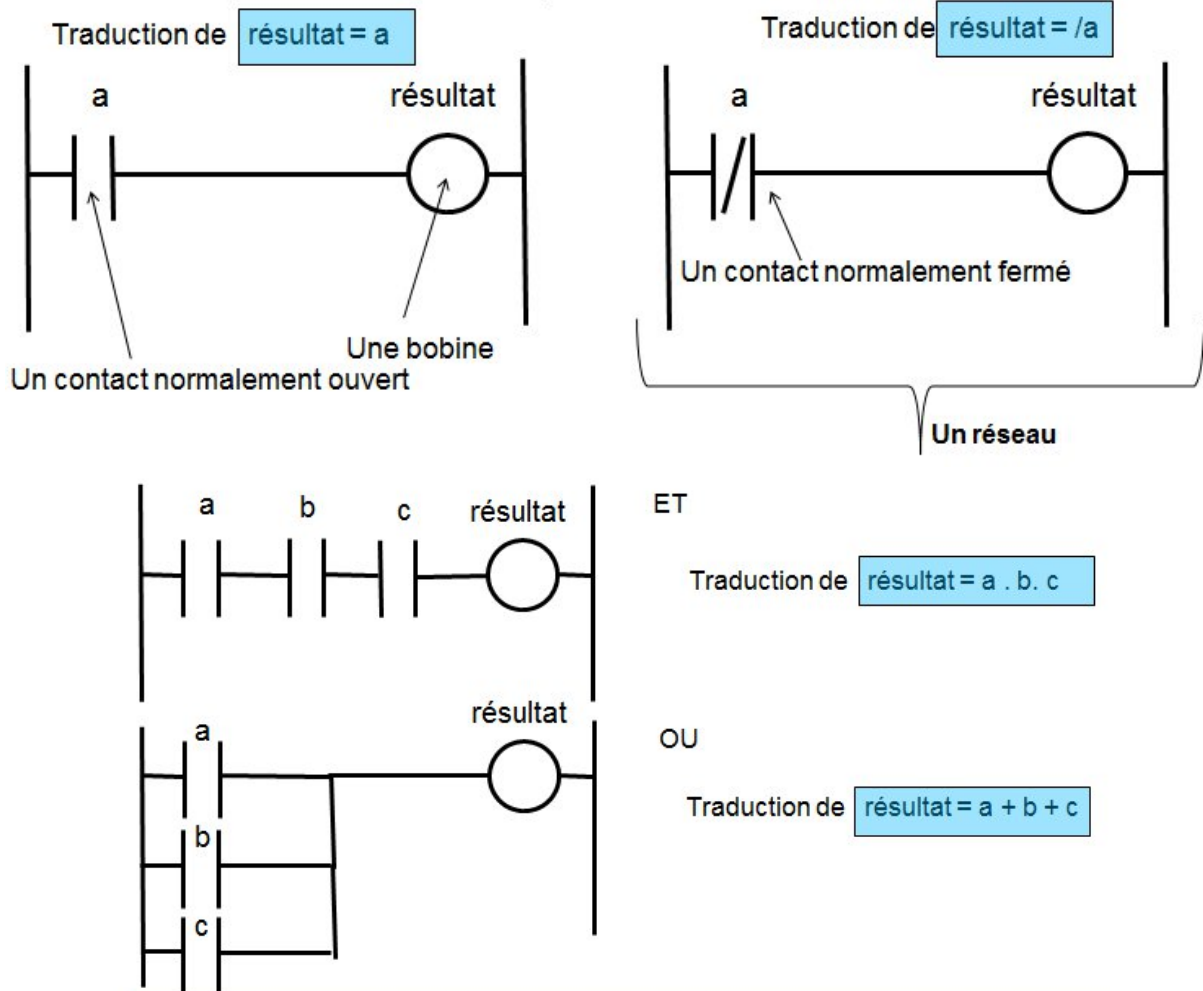
```
void setup() {
  myservo.attach(9); // attaches the servo on pin 9 to the servo object
}
```

```
void loop() {
  val = analogRead(potpin); // reads the value of the potentiometer (value between 0-1023)
  val = map(val, 0, 1023, 0, 180); // scale it for use with the servo (value between 0-180)
  myservo.write(val);        // sets the servo position according to the scaled value
  delay(15);                 // waits for the servo to get there
}
```

Programmation : Introduction, Simulation, Arduino IDE

11. Développement en LADDER

Le langage LADDER est constitué de "réseaux". Chaque réseau correspond à une équation logique. On manipule donc des signaux TOR (Tout ou rien). Cette équation permet de calculer un résultat dépendant de signaux d'entrée. Ce résultat est représenté par une bobine. Chaque signal d'entrée est représenté par un contact (interrupteur) normalement ouvert ou normalement fermé. Il ne peut y avoir qu'une seule bobine dans un même réseau.



Programmation : Introduction, Simulation, Arduino IDE

12. Les variables

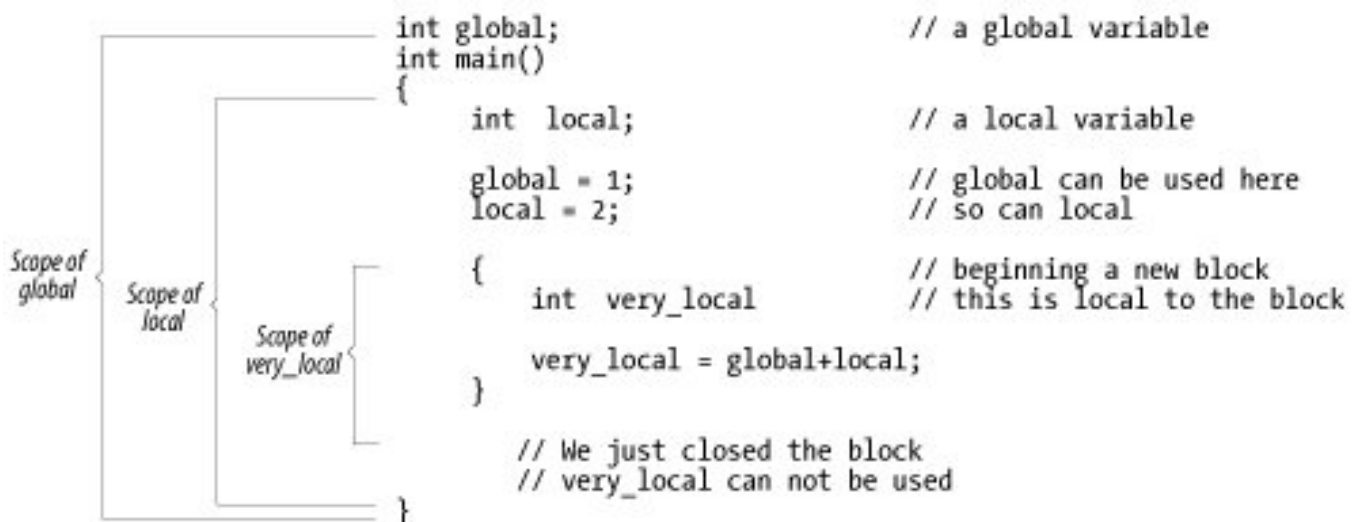
a) Les types entiers et flottants

Type	Minimum	Maximum
<code>_Bool</code>	0	1
<code>signed char</code>	-127	127
<code>unsigned char</code>	0	255
<code>short</code>	-32 767	32 767
<code>unsigned short</code>	0	65 535
<code>int</code>	-32 767	32 767
<code>unsigned int</code>	0	65 535
<code>long</code>	-2 147 483 647	2 147 483 647
<code>unsigned long</code>	0	4 294 967 295
<code>long long</code>	-9 223 372 036 854 775 807	9 223 372 036 854 775 807
<code>unsigned long long</code>	0	18 446 744 073 709 551 615
<code>float</code>	-1×10^{37}	1×10^{37}
<code>double</code>	-1×10^{37}	1×10^{37}
<code>long double</code>	-1×10^{37}	1×10^{37}

b) Le type char

A B C D E F G H I J K L M
 N O P Q R S T U V W X Y Z
 a b c d e f g h i j k l m
 n o p q r s t u v w x y z
 0 1 2 3 4 5 6 7 8 9
 ! " # % & ' () * + , - . / :
 ; < = > ? [\] ^ _ { | } ~

string est pour une chaîne de caractères



Introduction à la simulation TinkerCAD

<https://www.tinkercad.com/classrooms>

Code de la classe : Q9Y-NQE-CWY

Pseudo :

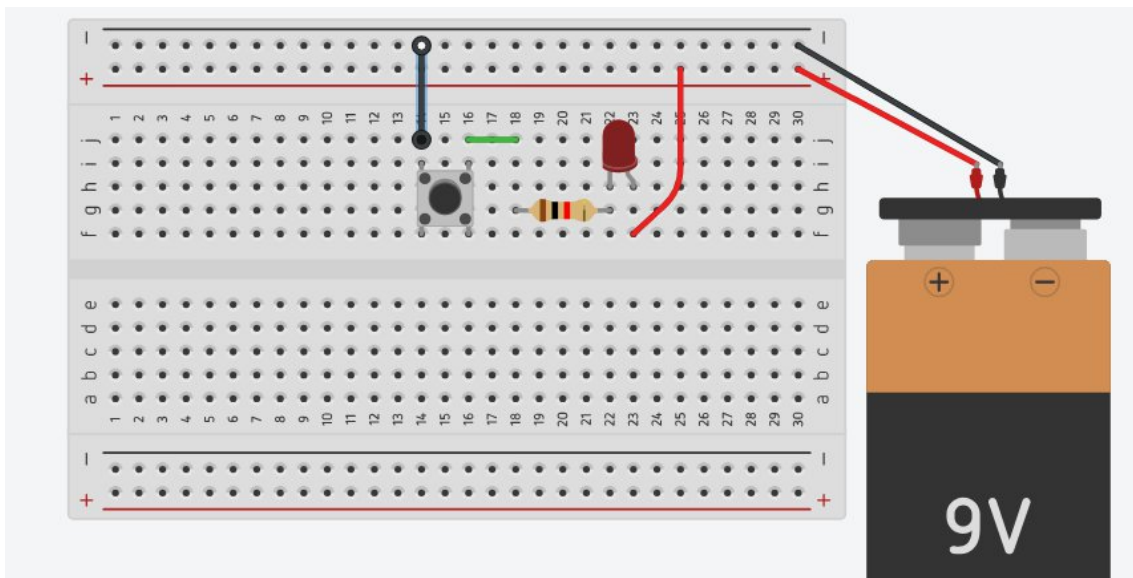
Document de l'activité : <https://nlardon.github.io> -> tinkercad



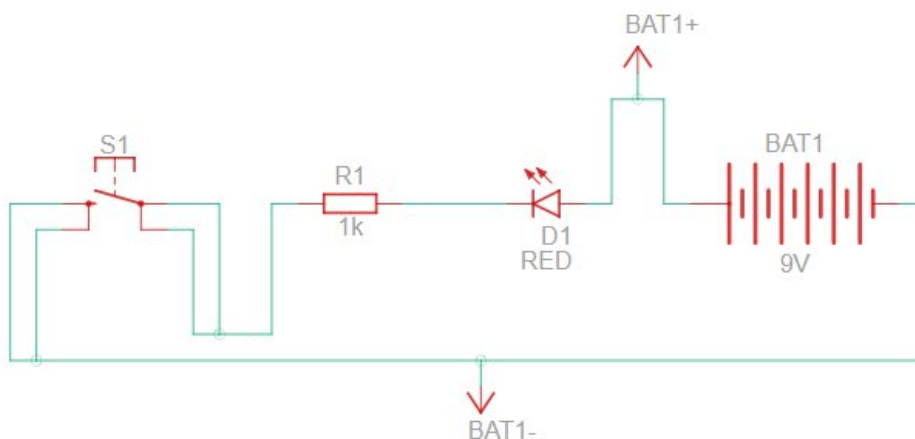
1. C'est quoi TinkerCAD ?

Le logiciel en ligne Circuit TINKERCAD permet de réaliser des circuits avec des composants (capteurs, servo-moteurs, voltmètres, ...) et d'en simuler le fonctionnement piloter par un programme en lignes de commande Arduino ou Scratch.

Vue de circuit :

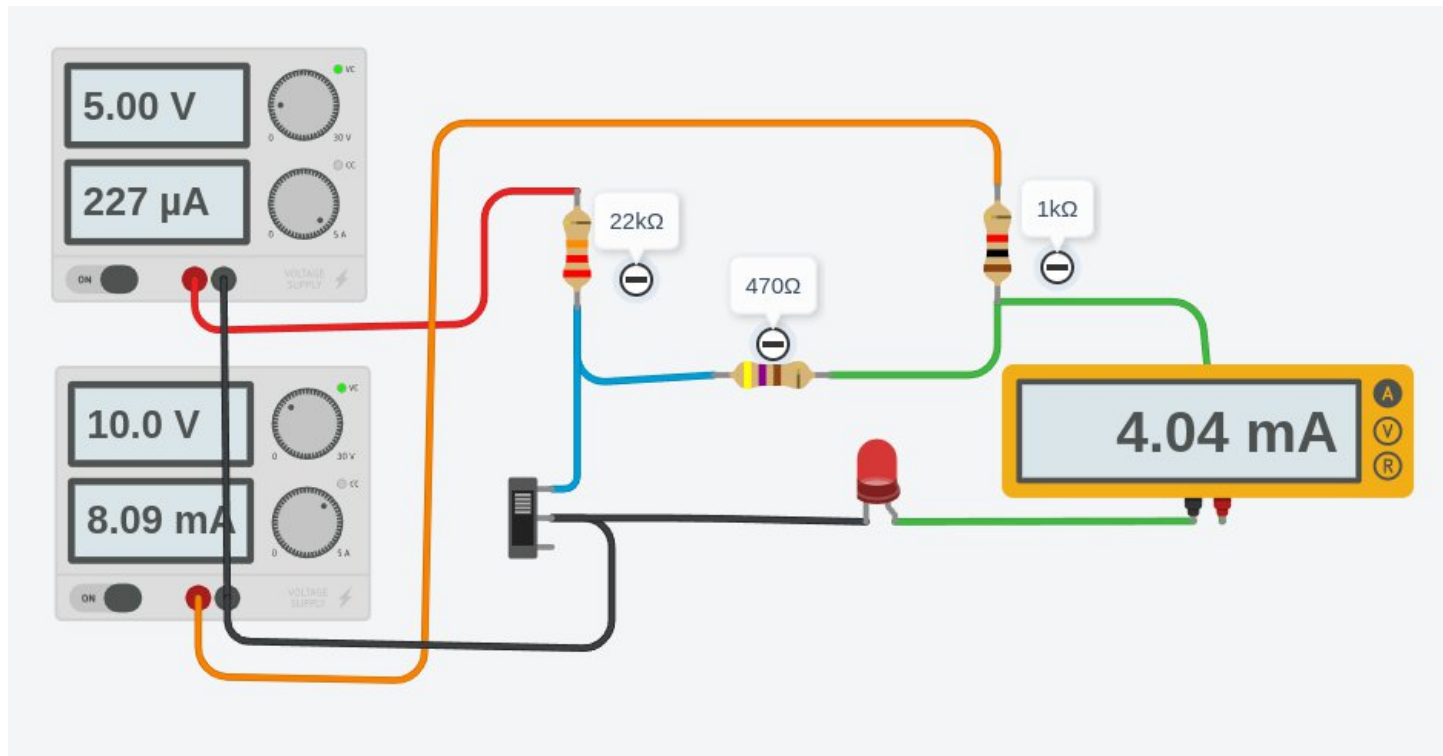


Vue schématique :

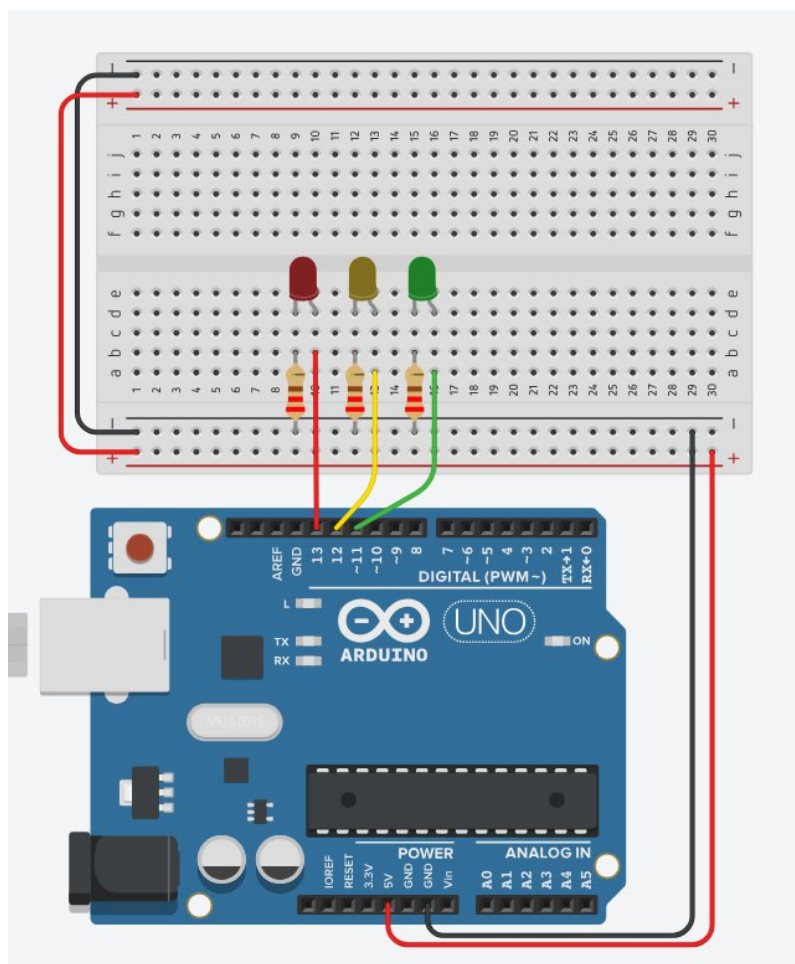


Programmation : Introduction, Simulation, Arduino IDE

2. Réaliser un circuit



3. Réaliser un circuit et son code



Code :

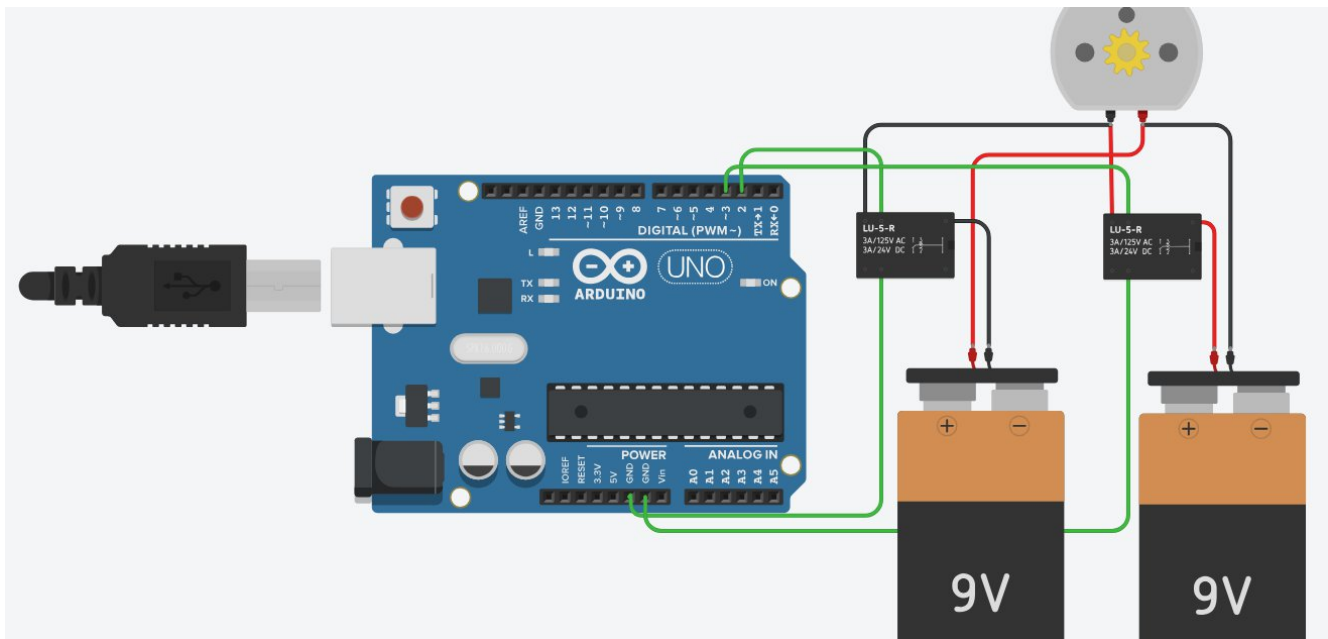
```
// C++ code
int animationSpeed = 0;

void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
  animationSpeed = 400;
  digitalWrite(13, HIGH);
  delay(animationSpeed);
  digitalWrite(13, LOW);
  delay(animationSpeed);
  digitalWrite(12, HIGH);
  delay(animationSpeed);
  digitalWrite(12, LOW);
  delay(animationSpeed);
  digitalWrite(11, HIGH);
  delay(animationSpeed);
  digitalWrite(11, LOW);
  delay(animationSpeed);
}
```


Programmation : Introduction, Simulation, Arduino IDE

4. Réaliser un circuit et son code



Code :

```

void setup()
{
  pinMode(3, OUTPUT);
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(3, HIGH);
  delay(1000); // wait for 1000 millisecond(s)
  digitalWrite(3, LOW);
  delay(1000); //wait just to see the result
  digitalWrite(2, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(2, LOW);
  delay(1000); //wait just to see the result
}

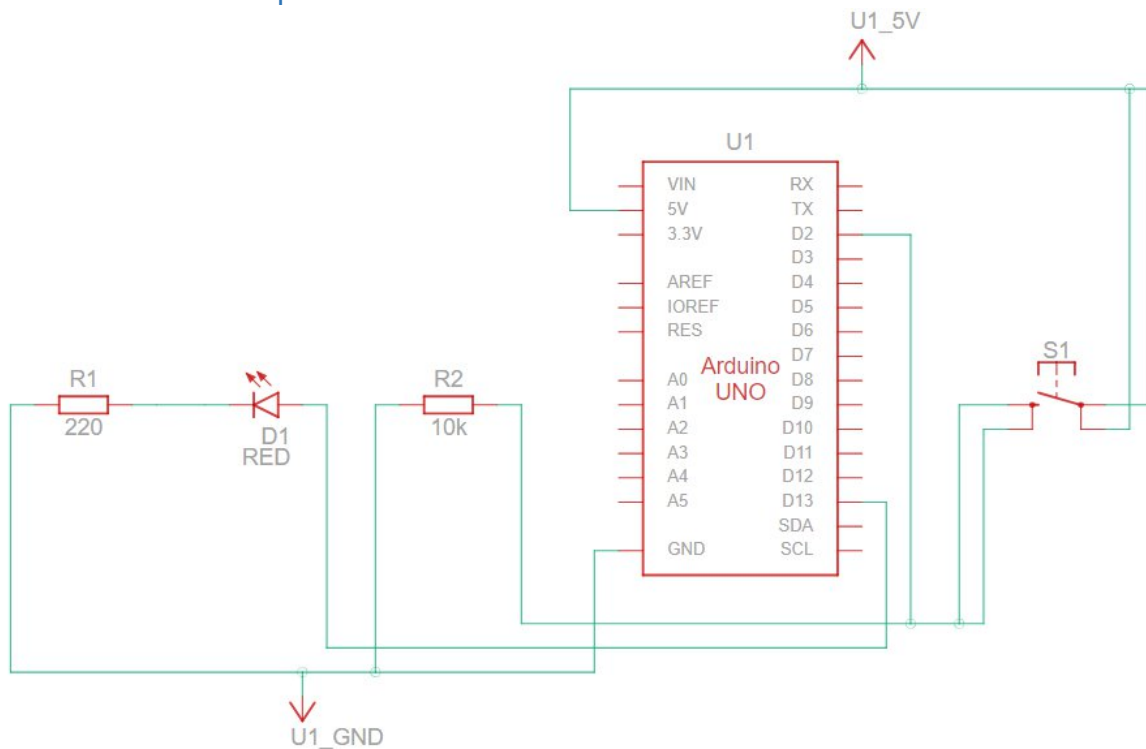
```

Liste de composants :

Nom	Quantité	Composant
M1	1	Moteur à courant continu
U2	1	Arduino Uno R3
K1 K2	2	Relais unipolaire bidirectionnel
BAT1 BAT2	2	Pile 9 V

Programmation : Introduction, Simulation, Arduino IDE

5. Réaliser un circuit à partir du schéma et son code



Code :

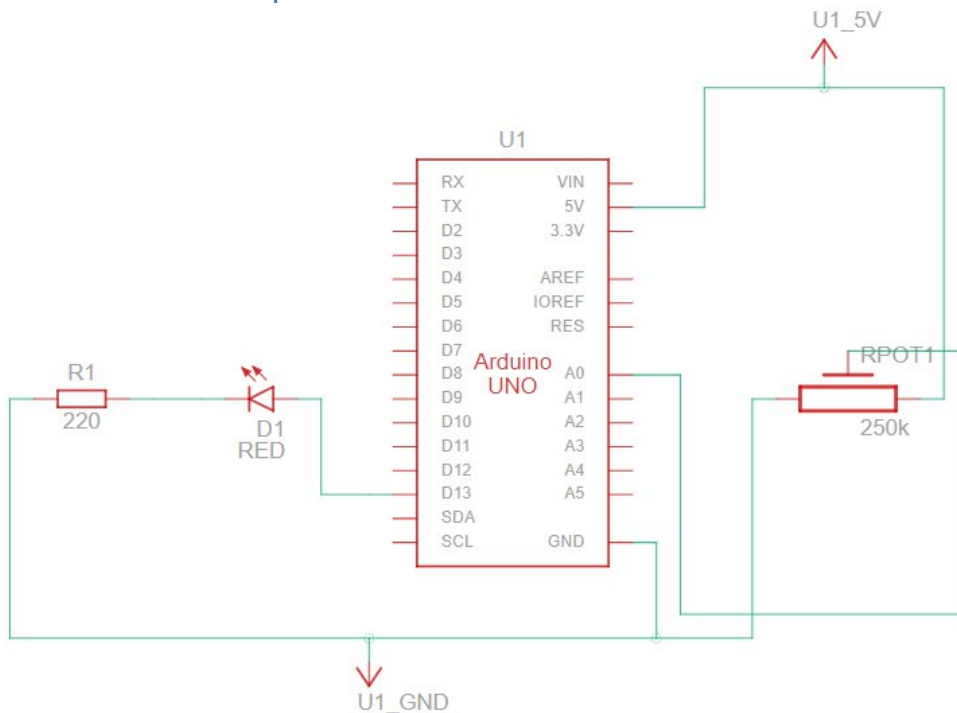
```
// C++ code
//
int buttonState = 0;

void setup()
{
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
  // read the state of the pushbutton
  buttonState = digitalRead(2);
  // check if pushbutton is pressed. if it is, the button state is HIGH
  if (buttonState == HIGH) {
    digitalWrite(LED_BUILTIN, HIGH);
  }
  else {
    digitalWrite(LED_BUILTIN, LOW);
  }
  delay(10);
}
```

Programmation : Introduction, Simulation, Arduino IDE

6. Réaliser un circuit à partir du schéma et son code



Code :

```
// C++ code
//
int sensorValue = 0;

void setup()
{
  pinMode(A0, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  // read the value from the sensor
  sensorValue = analogRead(A0);
  // turn the LED on
  digitalWrite(13, HIGH);
  // pause the program for <sensorValue> milliseconds
  delay(sensorValue); // Wait for sensorValue millisecond(s)
  // turn the LED off
  digitalWrite(13, LOW);
  delay(sensorValue);
}
```

Programmation : Introduction, Simulation, Arduino IDE

7. Réaliser un circuit à partir de son code et la liste des composants

Code :

```
// C++ code
//
int sensorValue = 0;

void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);
  pinMode(9, OUTPUT);
}

void loop()
{
  // read the value from the sensor
  sensorValue = analogRead(A0);
  // map the sensor reading to a range for the LED
  analogWrite(9, map(sensorValue, 0, 1023, 0, 255));
  delay(100); // Wait for 100 millisecond(s)
}
```

Liste de composants :

Nom	Quantité	Composant
U1	1	Arduino Uno R3
D1	1	Rouge LED
R1	1	220 Ω Résistance
R2	1	Photorésistance
R6	1	4.7 k Ω Résistance

Fonction du prototype :

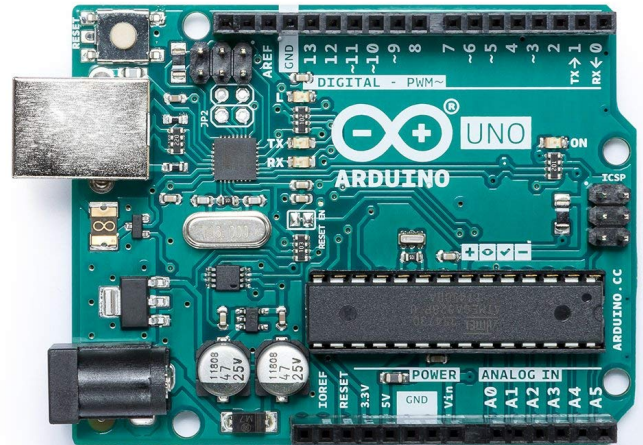
On mesure l'intensité lumineuse avec une photorésistance. Selon cette valeur on éclaire plus ou moins la LED.

Programmation : Introduction, Simulation, Arduino IDE**Découverte d'un système électronique embarqué : Arduino**

Tout comme le projet Raspberry Pi et le projet Arduino a révolutionné le monde de l'électronique embarquée et de l'informatique. La miniaturisation et le faible coût a ouvert une offre sur le marché pour la création de projets très divers.

Les champs d'applications sont : la domotique, la robotique, l'automatisme, la sécurité, les serveurs informatique, IoT (Internet of Thing) ...

Ces deux projets ont réussi à simplifier et mettre à la portée de tous l'électronique embarquée et l'informatique.



De nombreux systèmes complexes et coûteux il y a encore quelques années, sont aujourd'hui réalisés en interne dans les entreprises. Ces systèmes sont peu coûteux, flexibles, adaptables et recyclables pour d'autres projets.

13. Présentation de la carte Arduino

Pour répondre aux questions suivantes utilisées le dossier technique : nardon.github.io/arduino

Quelle carte avez-vous pour cette activité (référence exacte) ?

1. Quelle est la plage en tension d'alimentation (recommandée) de la carte Arduino ?

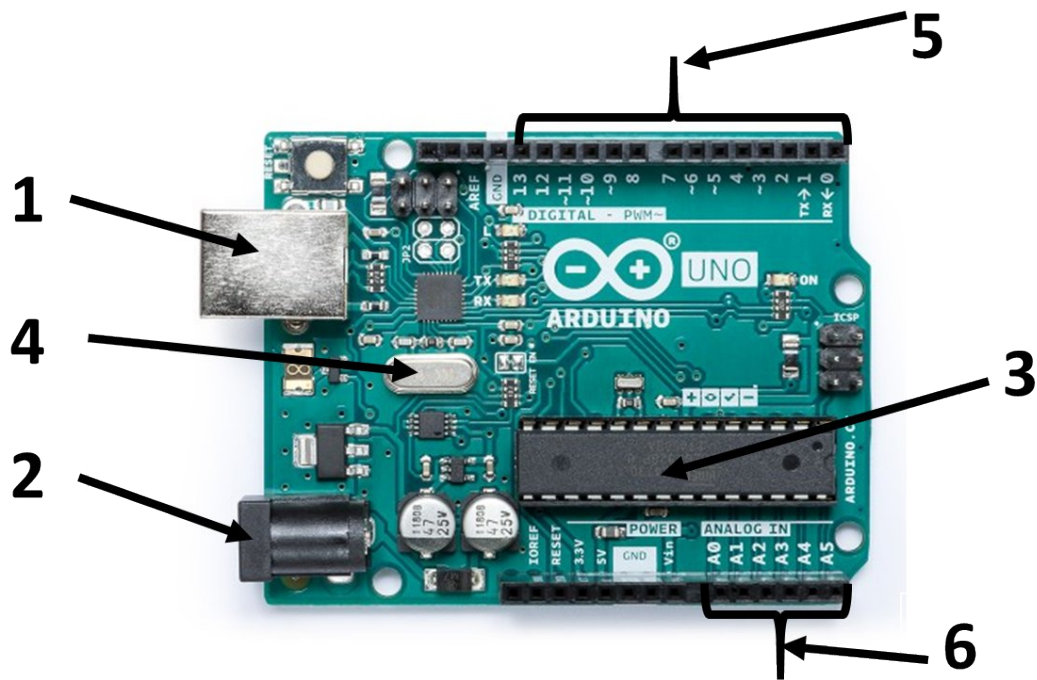
2. Quelle est l'intensité maximum que peut délivrer chaque sortie de la carte Arduino ?

3. Quel microcontrôleur utilise cette carte ?

4. Combien d'entrée/sortie (E/S) (I/O en anglais) sont disponible sur cette carte ?

Programmation : Introduction, Simulation, Arduino IDE

Identifier les composants sur la carte et leurs rôles.



1	
2	
3	
4	
5	
6	

Identifier en fluo ROSE sur la photo de la carte les points de potentiel à +5V.

Identifier en fluo BLEU sur la photo de la carte les points de potentiel à 0V.

Programmation : Introduction, Simulation, Arduino IDE

1) Mise en service de la carte Arduino

Programme simple : LED clignotant

Brancher le câble USB du PC à la carte.

Lancer le logiciel « Arduino », dans Fichier -> Exemples -> Basics

Ouvrir le programme « Blink »

Dans Outils -> Type de carte -> Choisir votre carte Arduino

Dans Outils -> Port -> Choisir le Port de la carte Arduino (sous linux /dev/ttyUSB0)

Blink signifie clignoter. Le programme écrit en langage C s'ouvre.

`void setup()` cette fonction va contenir les fonctions pour initialiser la carte au démarrage de la carte

`void loop()` cette boucle va contenir les fonctions qui seront lues en boucle, c'est le programme

Téléverser le programme.



Observer la LED « L » sur la carte.

```

Blink$
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}

```

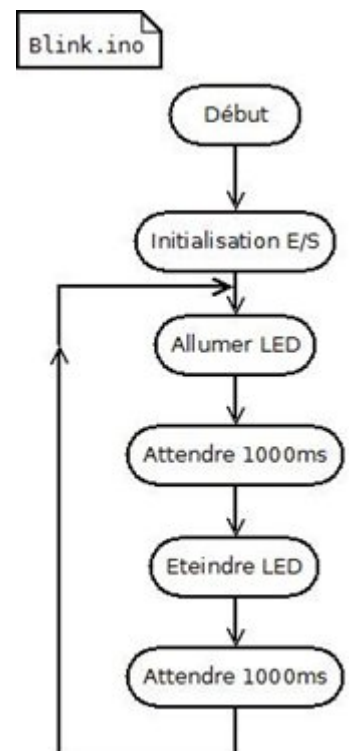
`LED_BUILTIN` = LED soudée sur la carte

12. Entourer en bleu : la partie du code qui permet l'initialisation des entrées/sorties E/S.

13. Entourer en noir : la partie du code qui allume la LED.

14. Entourer en rouge : la partie du code qui permet d'attendre 1 seconde.

15. Entourer en vert : la partie du code qui éteint la LED.



Programmation : Introduction, Simulation, Arduino IDE

STOP Appeler le professeur STOP

Détail de la commande : `digitalWrite("Numéro de la sortie", "Etat de la sortie")`

Modifier le code pour allumer la LED 3 secondes et éteindre la LED pendant 4 secondes. Ecrire ci-dessous seulement les fonctions modifiées.

STOP Appeler le professeur STOP

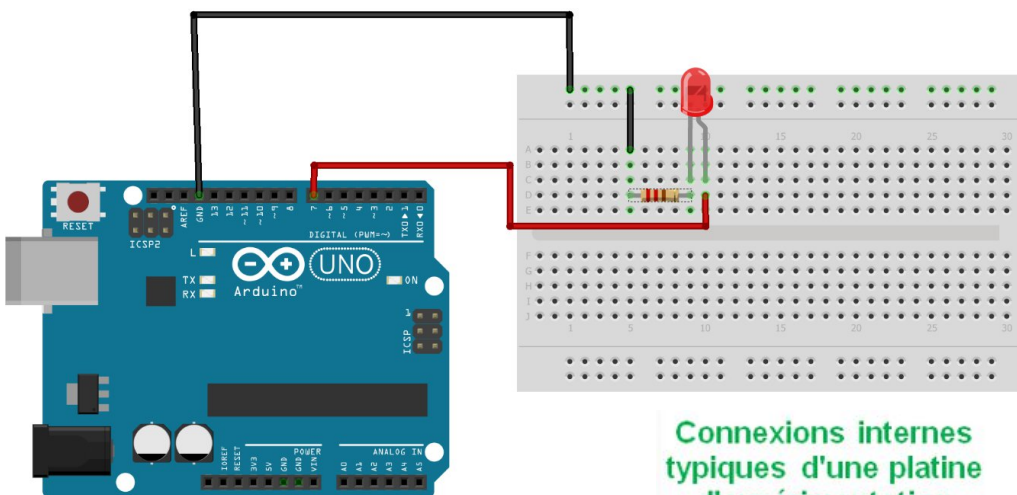
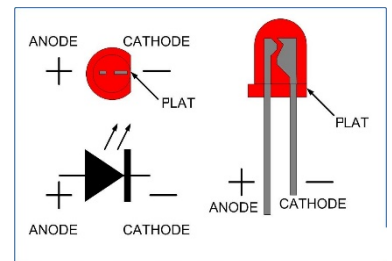
La LED soudée (LED BUILTIN) sur la carte est connectée à la sortie 13.

!!! Débranchez le câble USB de la carte !!!

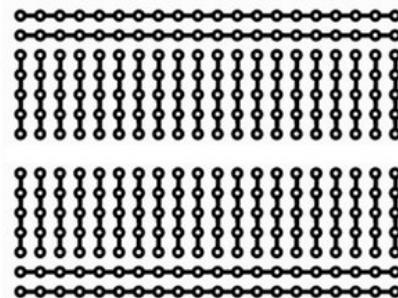
Modifier le code pour faire clignoter la sortie « 7 » de la carte.

Réaliser ce câblage sur une plaque d'essai à l'aide de fils mâle/mâle :

- Brancher la LED à la sortie 7 de la carte Arduino. (ATTENTION à choisir la bonne résistance Rouge/Rouge/Marron)
- Attention une LED est polarisée, elle a un sens



19



STOP Appeler le professeur STOP

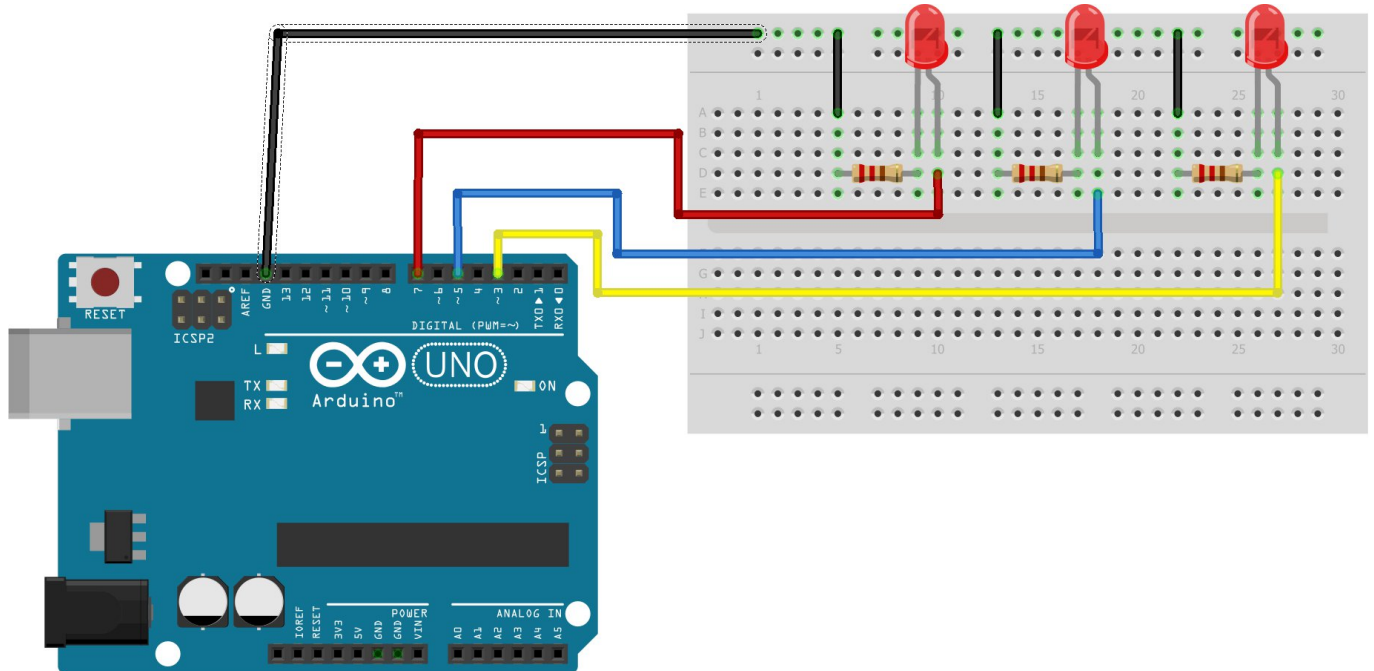
Programmation : Introduction, Simulation, Arduino IDE

Brancher et Téléverser le programme modifié.

!!! Débranchez le câble USB de la carte !!!

Réaliser ce câblage sur une plaque d'essai à l'aide de fils mâle/mâle :

(ATTENTION à choisir les bonnes résistances Rouge/Rouge/Marron)



fritzing

Modifier le programme pour que les LED s'allument les unes après les autres.

STOP Appeler le professeur STOP

Brancher et Téléverser le programme modifié.

Programmation : Introduction, Simulation, Arduino IDE

Programme : 1 bouton poussoir + 2 LED

Lancer le logiciel « Arduino », dans Fichier -> Ouvrir -> Ouvrir le programme « switch_2leds.ino » (présent dans ressources)

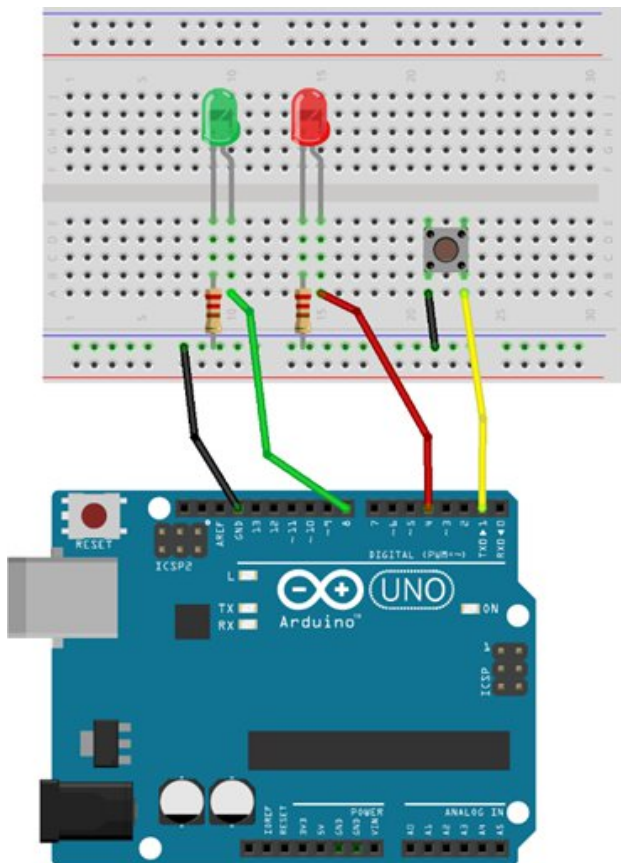
16. Sur quelle sortie la LED 1 est branchée ?

17. Sur quelle sortie la LED 2 est branchée ?

18. Sur quelle entrée le bouton poussoir est branché ?

Réaliser ce circuit :

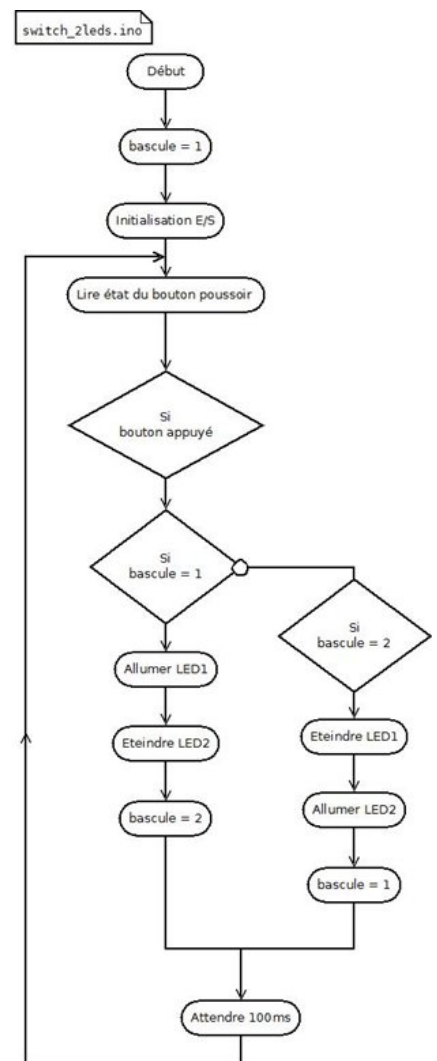
(ATTENTION à choisir les bonnes résistances Rouge/Rouge/Marron)



fritzing

STOP Appeler le professeur STOP

19. Brancher et Téléverser le programme.



Programmation : Introduction, Simulation, Arduino IDE

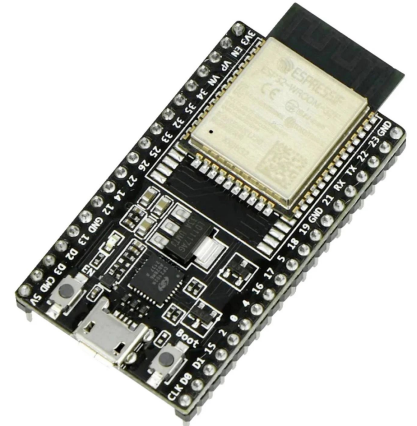
Découverte d'une carte type ESP32 (ou ESP8266)

L'ESP32 est une carte de développement électronique polyvalente et puissante, particulièrement adaptée aux projets de domotique, d'IoT (Internet des objets) et autres applications nécessitant de la connectivité sans fil et une bonne puissance de calcul.

L'ESP32 est un SOC développé par la société Espressif dédié à l'internet des objets (IoT) et plus particulièrement les communications sans fil Wifi et Bluetooth pour un coût réduit.

Plusieurs intérêts :

- Prix faible ~5€ pour le module ~8€ une carte de développement
- Intègre du Wifi 802.11 b/g/n/e/i
- Intègre Bluetooth 4.2 - BLE Bluetooth Low Energy
- Intègre un microcontrôleur double cœur 32 bits performants et de nombreux périphériques (ADC 12bit, DAC, 3xUART, PWM, I2C, SPI, etc ...)
- S'alimente directement en USB
- Compatible avec divers langages (C, C++, Python, MicroPython) et environnements de développement (Arduino, ESP-IDF)



Utilisations typiques

- Domotique et objets connectés (capteurs, actionneurs)
- Robotique et électronique embarquée
- Applications nécessitant du WiFi ou Bluetooth
- Prototypage rapide grâce à la compatibilité Arduino

1. Quel modèle de carte avez vous.

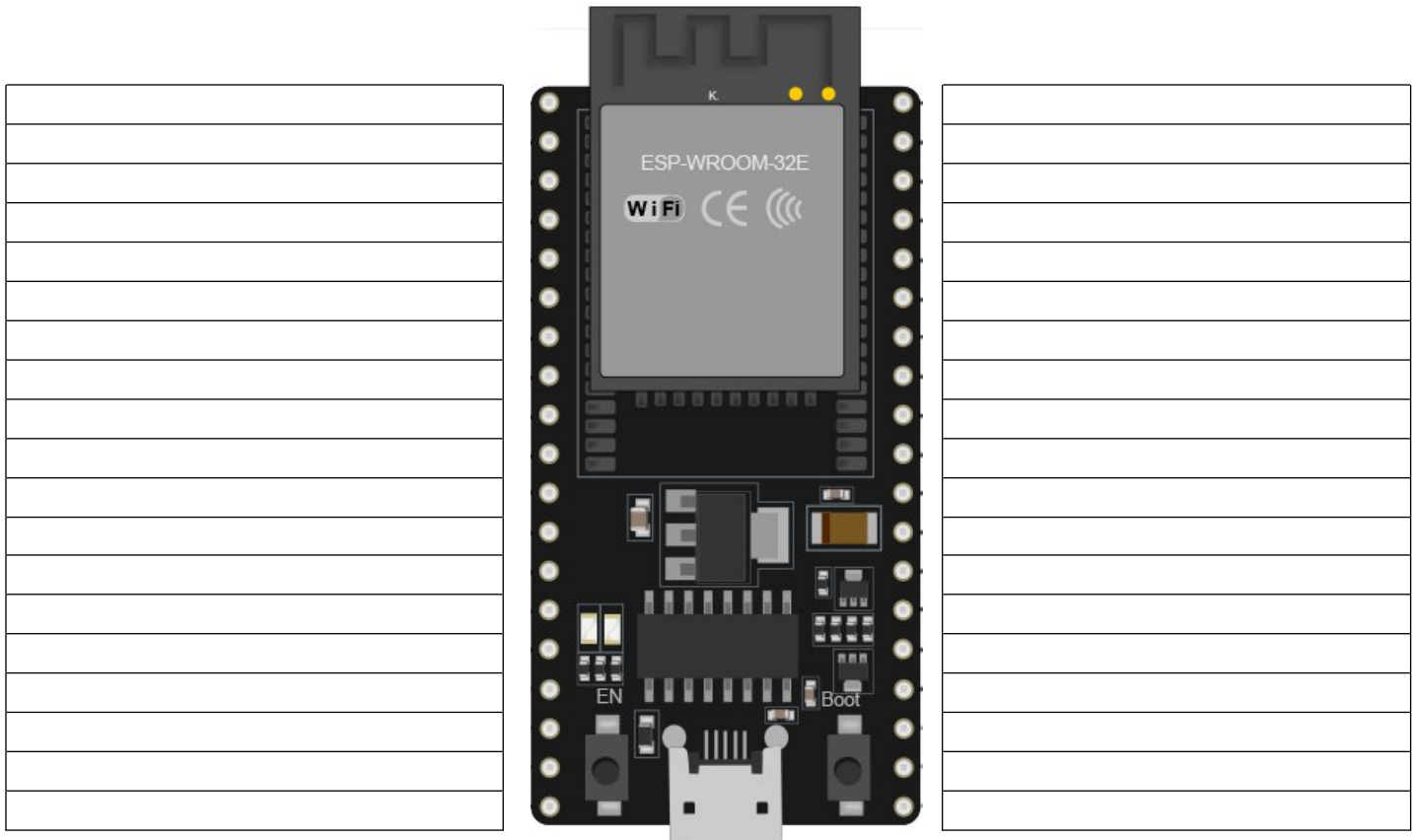
2. Quelle est la plage en tension d'alimentation (recommandée) de la carte.

ressource : <https://nlardon.github.io/esp>

3. Quelle est la tension de sortie ou d'entrée maximum pour cette carte.

4. Identifier les broches (pins) de la carte.

ressource : <https://nlardon.github.io/esp>



STOP Appeler le professeur STOP

Utiliser le langage C++ (Arduino IDE) et une carte ESP

Réaliser les étapes décrites ici : <https://nlardon.github.io/ide-arduino>

STOP Appeler le professeur STOP

Utiliser le langage Python (MicroPython/Thonny IDE) et une carte ESP

Réaliser les étapes décrites ici : <https://nlardon.github.io/micropython/>

STOP Appeler le professeur STOP

Programmation : Introduction, Simulation, Arduino IDE

Découverte mBot

1. Présentation de mBot :

Il s'agit d'un robot mBot du commerce.

Il est doté d'une carte électronique programmable. Celle-ci peut être programmée ou télé-opérée grâce à un ordinateur soit par fil (port USB), soit par Bluetooth.

On utilisera le logiciel « mBlock » pour créer nos programmes. Le langage graphique utilisé est Scratch.

Le robot **mBot** interagit avec son environnement en fonction du programme qu'on lui implante. Pour cela, il est capable de collecter des informations grâce à **ses capteurs** et de réaliser des actions grâce à **ses actionneurs**.



Actions et actionneurs :

le robot vendu de base, est capable de **se déplacer** : il est équipé de **deux moteurs** indépendants reliés chacun à une roue (qui devient donc **une roue motrice**).

il peut **émettre des sons** grâce à un **buzzer**.

il peut **émettre de la lumière** grâce à **2 DEL 3 couleurs** (RGB) dont la couleur est paramétrable.

d'autres actionneurs peuvent être branchés **en option** (afficheur 128 leds, motoréducteur, blocs 4 leds, afficheur 7 segments...).

Boutons et capteurs :

Pour interagir avec son environnement et y recueillir des informations, on retrouve sur le robot :

un **capteur de luminosité** qui le renseigne sur la luminosité ambiante.

un **module à ultrasons** qui lui permet de « voir » les obstacles à l'avant et d'en connaître la distance.

un **module de suivi de ligne au sol** à infrarouge.

un **bouton** paramétrable.

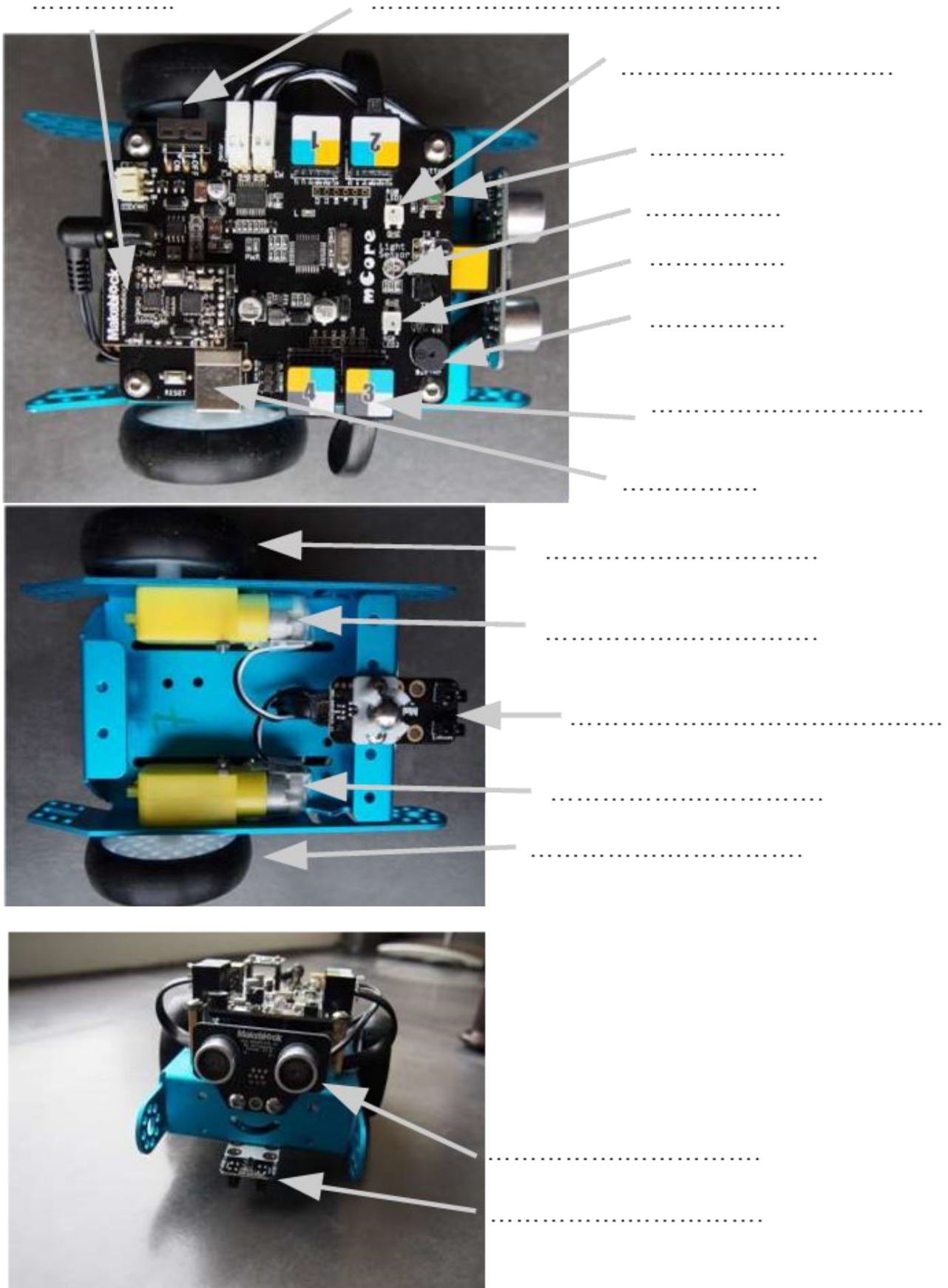
un **bouton de mise sous tension**.

d'autres capteurs peuvent être branchés **en option** (humidité, flamme, fumée, gyroscope...)



Programmation : Introduction, Simulation, Arduino IDE

LOCALISER LES CAPTEURS ET ACTIONNEURS



Programmation : Introduction, Simulation, Arduino IDE

1. Lancez le logiciel mBlock
2. Dans le menu « *appareils* », supprimer CyberPi et ajouter mBot
3. Assurez-vous que dans l'onglet « **Connecter** »

3/ Mbot en « mode connecté » :

Le **mode connecté correspond au mode où le câble USB reste branché à mBot ou connection Bluetooth constante**. Ce mode permet de télécommander mBot et/ou charger des programmes dans celui-ci.

En mode connecté, votre programme commencera par :

Il démarrera donc lorsque vous cliquerez sur le drapeau vert situé sur l'écran d'accueil.

Ce mode est « lent » car mBot envoie constamment les informations aux PC et attends les réponses du PC. Dans le mode mBot n'est pas autonome.

lorsque vous cliquez sur

Programme n°1 : faire clignoter (1s) une DEL RGB (droite ou gauche) en rouge 10 fois de suite

Méthode

Réaliser une boucle et mettre à l'intérieur

Allumer la del

Attendre 1s

Eteindre la del

Attendre 1s

Programmer la boucle pour qu'elle fasse le programme 10 fois



Programme n°2 : Identique au Programme 1 mais faire clignoter les 2 DEL RGB en alternance

Méthode

Réaliser une boucle et mettre à l'intérieur

Allumer la del1 éteindre la del 2

Attendre 1s

Eteindre la del1 allumer la del2

Attendre 1s

Programmer la boucle pour qu'elle fasse le programme 10 fois

Eteindre les 2 dels après avoir fini la boucle



Programme n°3 : Utiliser une condition et le détecteur de lumière pour faire fonctionner le P2

Méthode

Avant la boucle du P2,

Ajouter une condition Si ... Alors

Ici l'événement déclencheur est la baisse de la lumière mesurée sur la carte en dessous de 300 lux



Programmation : Introduction, Simulation, Arduino IDE

Programme n°4 : Utiliser le capteur de luminosité pour créer une alarme.

Méthode

Utiliser la condition si alors/sinon

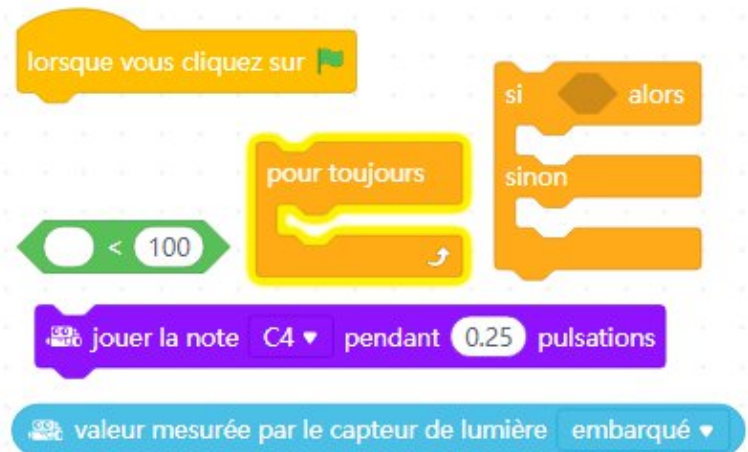
Une condition Si ... Alors

Ici l'événement déclencheur est la baisse de la lumière mesurée sur la carte en dessous de 300 lux

Jouer la note C4 sur le buzzer

Sinon

Jouer la note C2 sur le buzzer



Programme n°5 : Il ne s'exécute qu'une seule fois. Le robot avance durant 3 secondes à la vitesse 100. Puis il s'arrête

Méthode

Réaliser une boucle et mettre à l'intérieur

Avancer le robot à vitesse 100

Attendre 3s

Stopper le robot



Programme n°6 : Dissocier la vitesse des moteurs ; Le robot tourne en rond 3s (sens horaire) puis 3s (sens trigonométrique) 2 fois de suite

Méthode

Réaliser une boucle et mettre à l'intérieur

Faire tourner la roue gauche seule

Attendre 3s

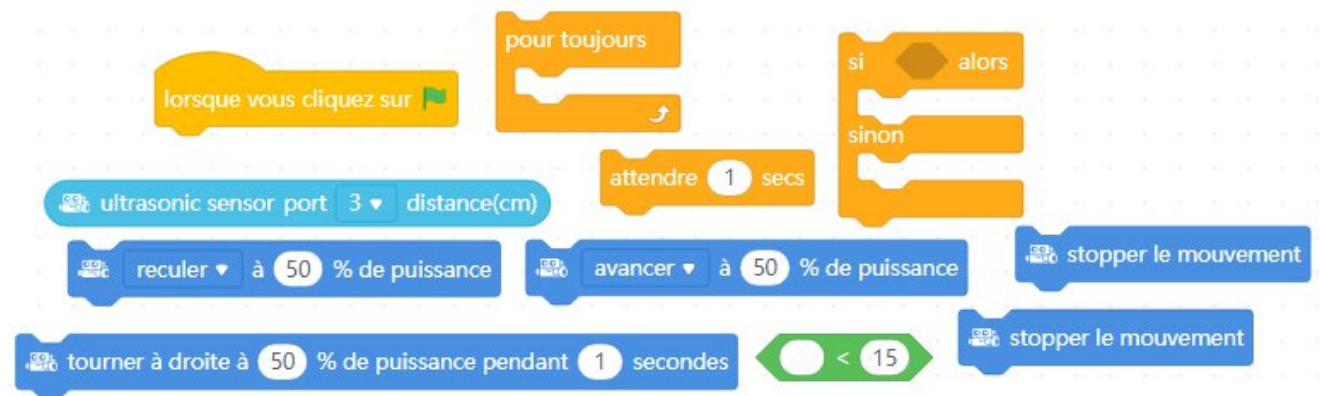
Faire tourner la roue droite seule

Attendre 3s

Stopper le robot



Programme n°7 : Le robot avance s'il voit un obstacle à 15cm devant lui, il recule, tourne à droite et reprend son avance



Programmation : Introduction, Simulation, Arduino IDE

Programme n°8 : Le robot suit une ligne

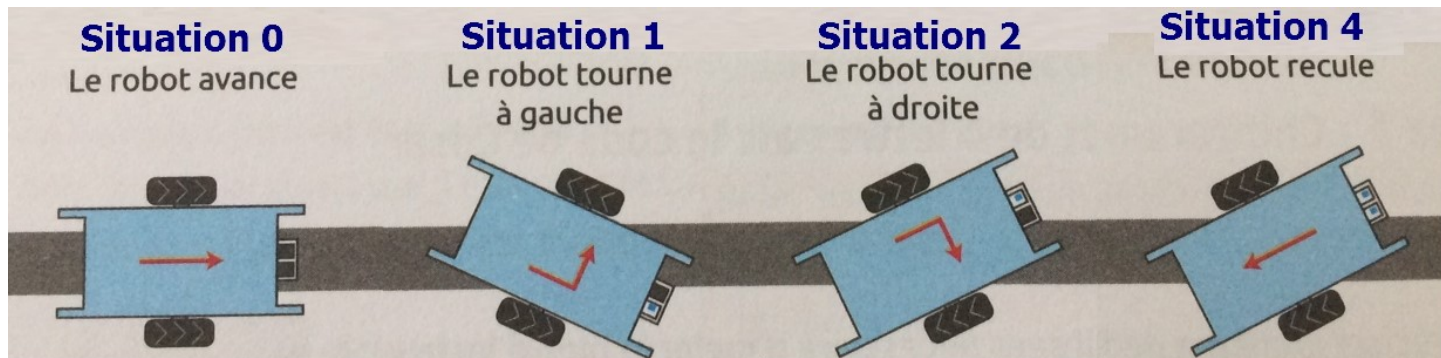
Nous allons maintenant programmer le robot en utilisant les capteurs "suiveurs de ligne".

Le robot se déplace en suivant un marquage au sol (ligne noire). Pour assurer cette fonction, il dispose à l'avant d'un module suiveur de ligne, composé de deux capteurs optiques.

Tant que les deux capteurs détectent la ligne, le robot avance (situation 0).

Lorsqu'un des deux capteurs ne détecte plus la ligne, le robot doit tourner sur lui-même pour se remettre dans l'axe (situation 1 ou 2).

Si deux capteurs sont en dehors de la ligne, le robot recule (situation 4).



Le principe de fonctionnement du capteur est le suivant :









Lorsque les deux capteurs détectent une couleur claire la valeur état suiveur est à 3.

Lorsque le capteur de droite détecte une couleur foncée et le capteur de gauche détecte une couleur claire la valeur état suiveur est à 2.

Lorsque le capteur de droite détecte une couleur claire et le capteur de gauche détecte une couleur foncée la valeur état suiveur est à 1.

Lorsque les deux capteurs détectent une couleur foncée la valeur état suiveur est à 0.

Compléter le tableau selon les situations :

Capteur gauche	Capteur droit	Valeur renvoyée par le capteur	Position du robot	Instruction à envoyer
		3		
		2		
		1		
		0		

Programmation : Introduction, Simulation, Arduino IDE

Simulation TinkerCAD

1. Exemple 1 LED

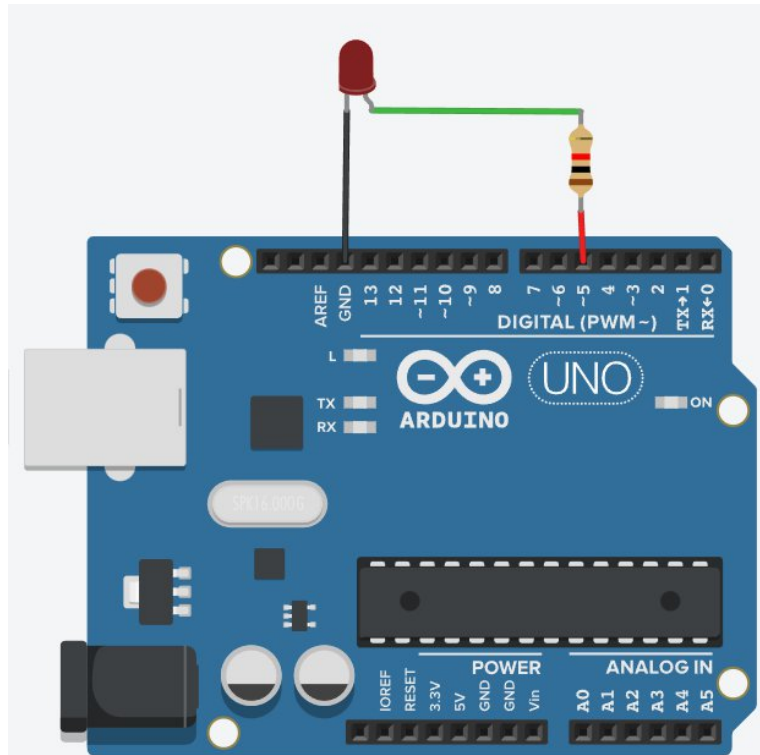
R1 = 220 Ω

```

void setup() {
    pinMode(5,OUTPUT);
}

void loop() {
    digitalWrite(5, HIGH);
    delay(500);
    digitalWrite(5, LOW);
    delay(500);
}

```



2. Exemple 1 bouton

R1 = 10 k Ω

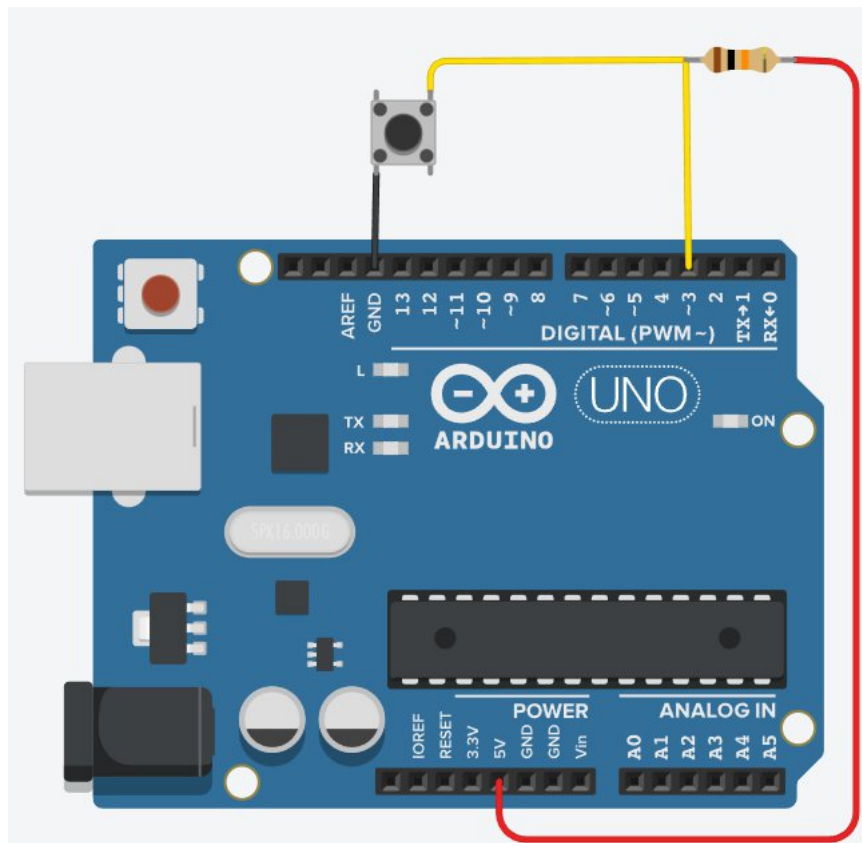
```

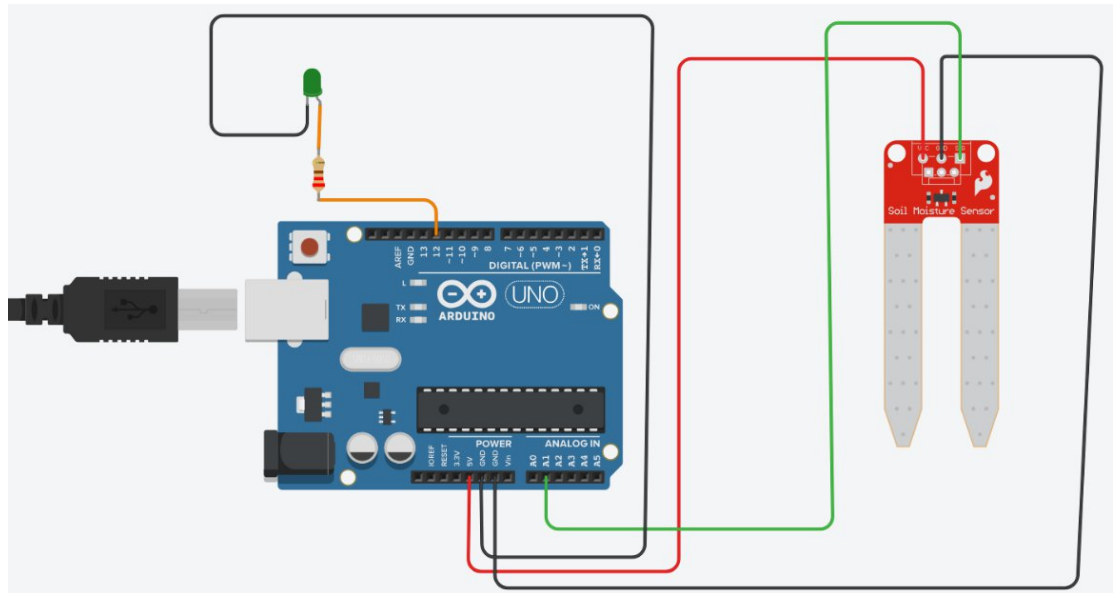
bool buttonState;

void setup() {
    pinMode(3,INPUT);
    pinMode(13,OUTPUT);
    //LED sur la carte
}

void loop() {
    buttonState = digitalRead(3);
    digitalWrite(13, buttonState);
    delay(100);
}

```



Programmation : Introduction, Simulation, Arduino IDE**3. Exemple de code : Mesurer d'une humidité****Le montage :**R1 = 220 Ω **Le code :**

```

int sensorPin = A1; // déclaration de la pin de mesure du capteur d'humidité de sol
int sensorValue = 0; // déclaration variable et initialisation de la valeur du capteur
int led = 12; // déclaration de la pin de la LED
int seuil_alerte = 300; // déclaration variable et initialisation du seuil que l'alerte est donnée

void setup() {
  Serial.begin( 9600 ); // initialiser le port série
  pinMode( led, OUTPUT ); // initialiser la broche/pin
}

void loop() {
  sensorValue = analogRead( sensorPin ); // lire la valeur
  delay(1000); // attendre 1 s

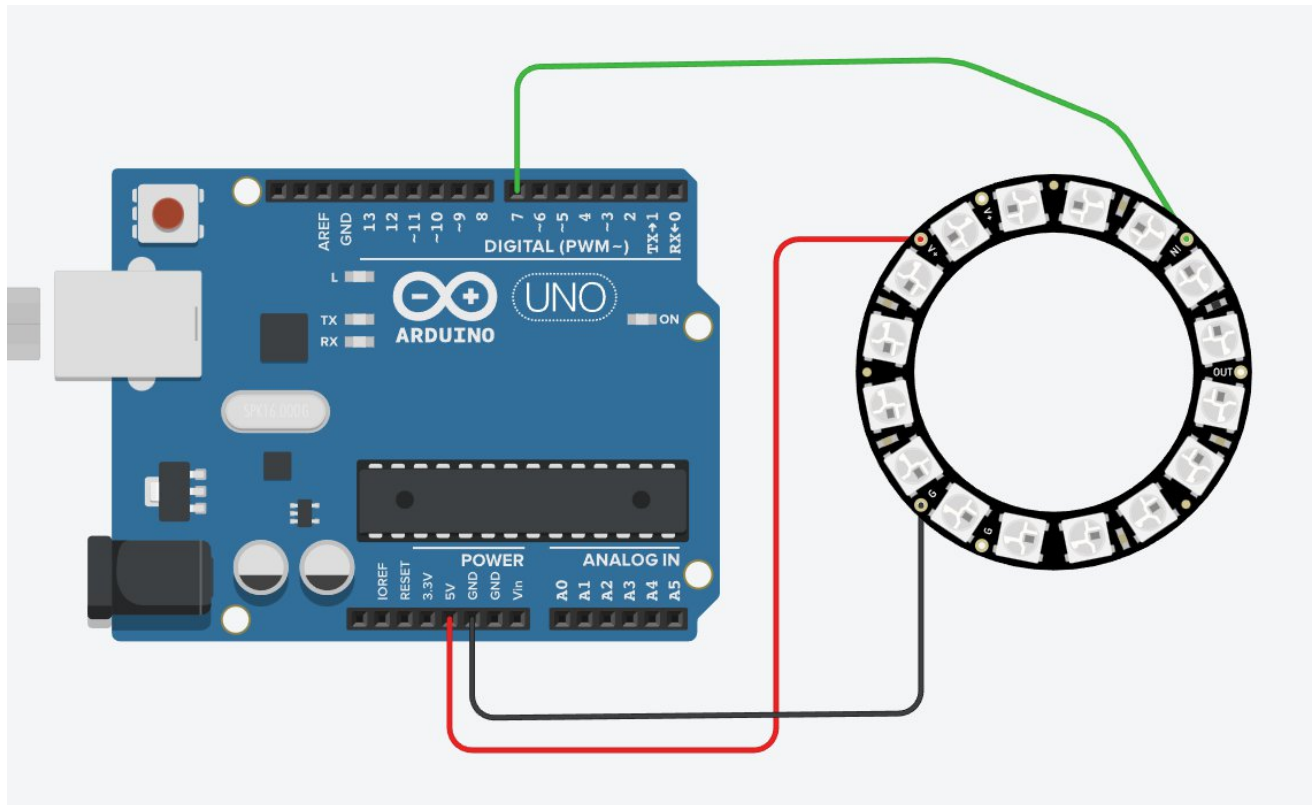
  if ( sensorValue < seuil_alerte ) {
    digitalWrite( led, HIGH ); // allumer de la LED
    delay( 1000 ); // attendre 1 s
    digitalWrite( led, LOW ); // éteindre de la LED
  }

  Serial.println( "L'humidité est de " + String( sensorValue ) ); // Afficher dans le moniteur
  série // la valeur
}

```

Programmation : Introduction, Simulation, Arduino IDE

4. Exemple contrôle LEDs



```
#include <Adafruit_NeoPixel.h>
#define PIN 7 // input pin Neopixel is attached to
#define NUMPIXELS 16 // number of neopixels in Ring
```

```
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
```

```
void setup() {
    pixels.begin(); // Initializes the NeoPixel library.
}
```

```
void loop() {
    for(int i=0;i<NUMPIXELS;i++){
        // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
        pixels.setPixelColor(i, pixels.Color(255, 0, 0));
        pixels.show(); // This sends the updated pixel color to the hardware.
        delay(100); // Delay for a period of time (in milliseconds).
    }
    for(int i=NUMPIXELS;i>=0;i--){
        // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
        pixels.setPixelColor(i, pixels.Color(0, 255, 0));
        pixels.show(); // This sends the updated pixel color to the hardware.
        delay(100); // Delay for a period of time (in milliseconds).
    }
}
```

Programmation : Introduction, Simulation, Arduino IDE

5. Exemple avec 1 LED et Bouton poussoir

Faire un circuit et un programme pour : Clignoter une LED pendant que l'on presse un bouton poussoir.

Consignes :

- Un bouton poussoir
- Une LED est câblée sur
- Si on presse le bouton (période 500 ms).

Aide :

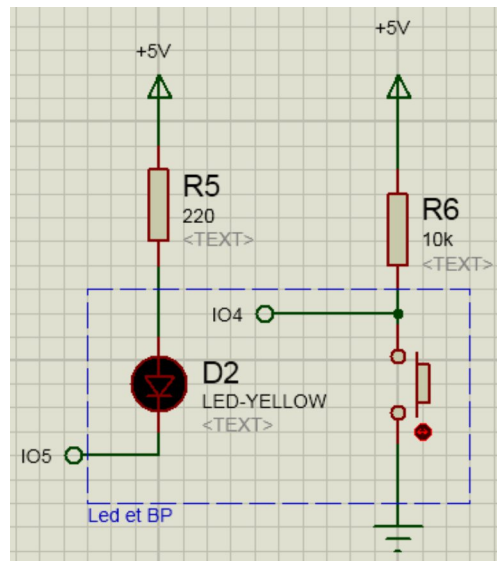
Lire l'état d'un bouton :

buttonState =

Cette fonction retourne soit

Condition en C++ :

```
if (buttonState == HIGH) {
    instruction ...
}
else {
    instruction ...
}
```



est câblé sur la pin 4.
la pin 5.
poussoir, la LED doit clignoter à 2 Hz

`digitalRead(buttonPin);`
HIGH soit LOW

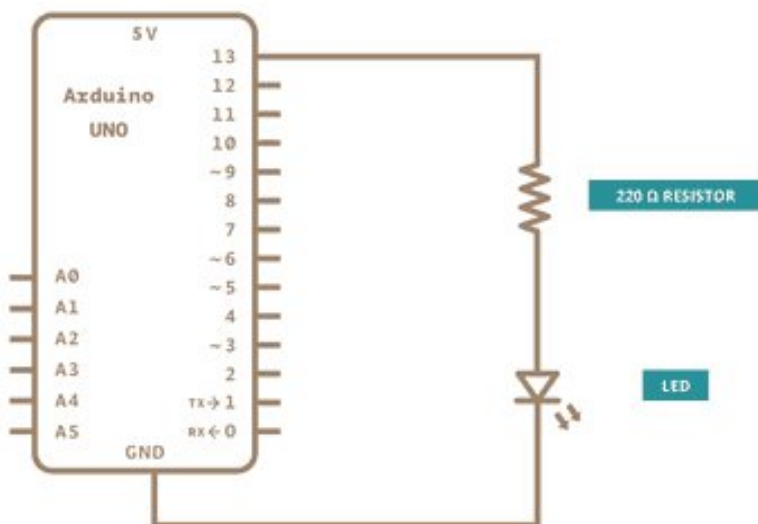
Programmation : Introduction, Simulation, Arduino IDE

Analyser un code C++ Arduino

sources : <https://docs.arduino.cc/built-in-examples/basics/>

1. Programme 1 : Blink

```
void setup() {  
  
    pinMode(LED_BUILTIN, OUTPUT);  
  
}  
  
void loop() {  
  
    digitalWrite(LED_BUILTIN, HIGH);  
  
    delay(1000);  
  
    digitalWrite(LED_BUILTIN, LOW);  
  
    delay(1000);  
  
}
```



Programmation : Introduction, Simulation, Arduino IDE

2. Programme 2 : DigitalReadSerial

```
int pushButton = 2;
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    pinMode(pushButton, INPUT);
```

```
}
```

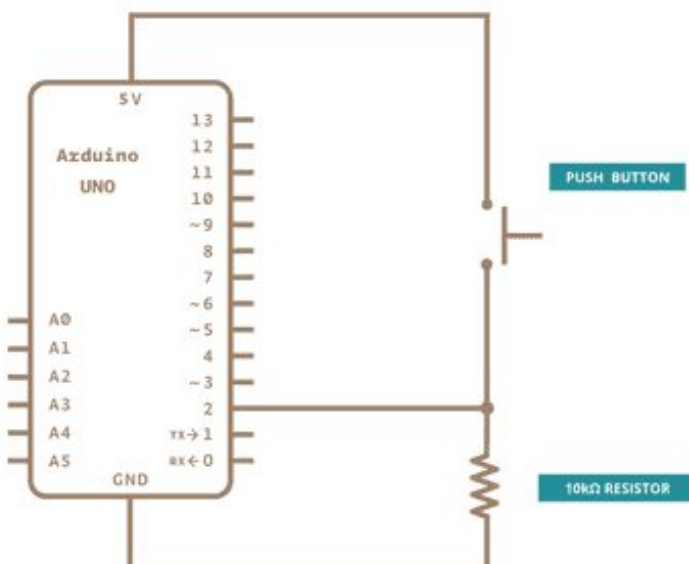
```
void loop() {
```

```
    int buttonState = digitalRead(pushButton);
```

```
    Serial.println(buttonState);
```

```
    delay(1);
```

```
}
```



Programmation : Introduction, Simulation, Arduino IDE

3. Programme 3 : Fade

```
int led = 9;
```

```
int brightness = 0;
```

```
int fadeAmount = 5;
```

```
void setup() {
```

```
    pinMode(led, OUTPUT);
```

```
}
```

```
void loop() {
```

```
    analogWrite(led, brightness);
```

```
    brightness = brightness + fadeAmount;
```

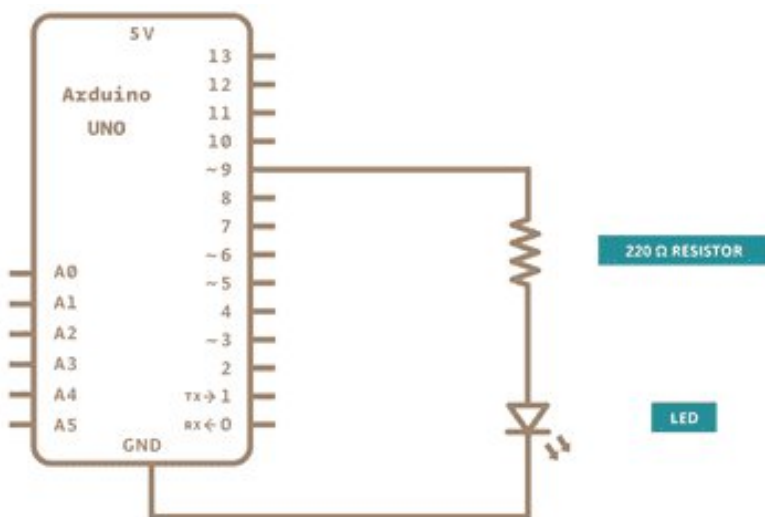
```
    if (brightness <= 0 || brightness >= 255) {
```

```
        fadeAmount = -fadeAmount;
```

```
    }
```

```
    delay(30);
```

```
}
```



Programmation : Introduction, Simulation, Arduino IDE

4. Programme 4 : AnalogInput

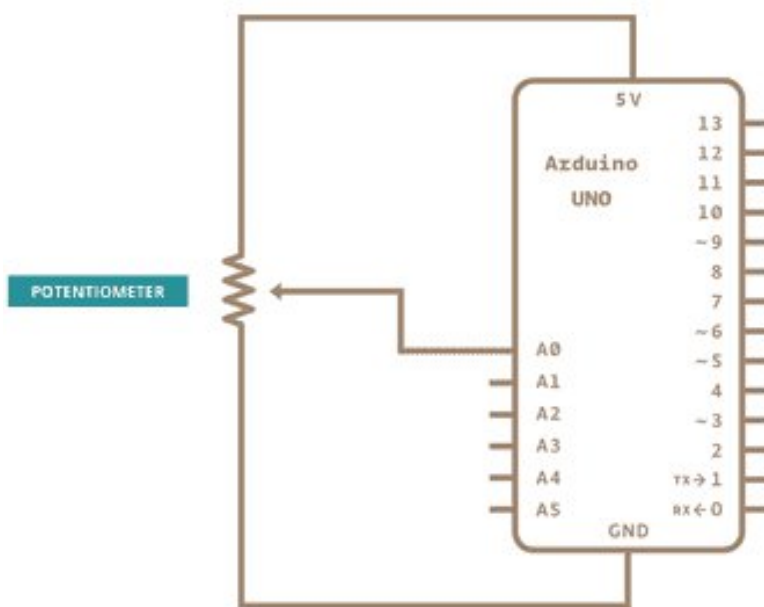
```
int sensorPin = A0;
```

```
int ledPin = 13;
```

```
int sensorValue = 0;
```

```
void setup() {  
    pinMode(ledPin, OUTPUT);  
}
```

```
void loop() {  
  
    sensorValue = analogRead(sensorPin);  
  
    digitalWrite(ledPin, HIGH);  
  
    delay(sensorValue);  
  
    digitalWrite(ledPin, LOW);  
  
    delay(sensorValue);  
}
```



Programmation : Introduction, Simulation, Arduino IDE

Programmation : Conditions et boucles

1. Les fonctions de base du langage C/C++

Les variables	Les fonctions de base sur Arduino IDE
bool : une valeur : vrai (true) ou faux (false). char : un caractère int : un nombre entier unsigned int : un nombre entier positif ou nul double : un nombre à virgule string : une chaîne de caractères	<pre>void setup() { } void loop() { }</pre>

Les opérateurs de comparaison	Les opérateurs logiques																						
Le tableau suivant liste les opérateurs utilisés en C/C++ pour définir des conditions :	Le tableau suivant liste les opérateurs logiques utilisés en C/C++ pour définir des conditions :																						
<table> <tr> <th colspan="2">Opérateurs</th></tr> <tr> <td>==</td><td>Egalité</td></tr> <tr> <td>!=</td><td>Différence</td></tr> <tr> <td><</td><td>Inférieur</td></tr> <tr> <td>></td><td>Supérieur</td></tr> <tr> <td><=</td><td>Inférieur ou égal</td></tr> <tr> <td>>=</td><td>Supérieur ou égal</td></tr> </table>	Opérateurs		==	Egalité	!=	Différence	<	Inférieur	>	Supérieur	<=	Inférieur ou égal	>=	Supérieur ou égal	<table> <tr> <th colspan="2">Opérateurs</th></tr> <tr> <td>&&</td><td>Et logique</td></tr> <tr> <td> </td><td>Ou logique</td></tr> <tr> <td>!</td><td>Non</td></tr> </table>	Opérateurs		&&	Et logique		Ou logique	!	Non
Opérateurs																							
==	Egalité																						
!=	Différence																						
<	Inférieur																						
>	Supérieur																						
<=	Inférieur ou égal																						
>=	Supérieur ou égal																						
Opérateurs																							
&&	Et logique																						
	Ou logique																						
!	Non																						

Exemples :

a = 2 ; b = 5 ; c = 4 ;	d = 2 ; e = 1 ; f = 8 ;		
Opération	Vrai ou Faux (True or False)	Opération	Vrai ou Faux (True or False)
a == b a != b a < b a > b a <= b a > a a >= a		a == d && a == c a == d a == c a < f && c > a a < d c > e	

Programmation : Introduction, Simulation, Arduino IDE**2. Conditions : if() ... else****Si Alors : if()**

La condition **Si Alors** permet de réaliser une ou plusieurs actions si seulement la condition énoncée est vraie, si ce n'est pas le cas le programme saute la condition et poursuit son exécution.

```
int a = 0; //Déclaration d'un entier "a", initialisé à 0
if(a<10)
{
    a=a+1;
}
```

Si Alors Sinon : if() else

```
int a = 0;
if(a<10)
{
    a=a+1; // Si "a" est inférieur à 10 alors on rajoute 1 à "a"
}
else
{
    a=a-1;
}
```

Exemple de condition avec des sous-fonctions ou programmes

<pre>void loop(){ if condition { program1(); } if condition { program2(); } if condition { program3(); } }</pre>	<pre>void program1(){ le code du programme1; } void program2(){ le code du programme2; } void program3(){ le code du programme3; }</pre>
--	--

Programmation : Introduction, Simulation, Arduino IDE

3. Pour : for()

La boucle **Pour** permet de répéter une suite d'instructions selon une évolution définie au début. La structure de cette boucle est donnée ci-dessous.

Paramétrage de la boucle for:

- Initialisation de la variable de comptage : La variable de comptage ici « i » est la variable qui s'incrémentera ou se décrémentera à **chaque passage dans la boucle** suivant le pas de comptage choisi, celle-ci est ici initialisée à 0.
- Condition de comptage : Nous définissons ici la condition d'exécution de la boucle, dans notre cas nous tournerons dans la boucle tant que la variable de comptage « i » est inférieure à 10.
- Pas de comptage : Ici nous définissons l'évolution de la variable de comptage « i » lors de chaque passage dans la boucle. Dans notre cas nous incrémentons de 1 à chaque passage dans la boucle, i++ revient à coder i=i+1.

Exemple utilisant cette fonction :

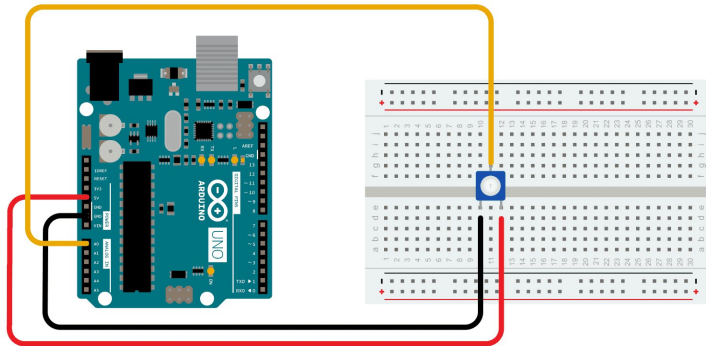
```
for(int i=0 ; i < 10 ; i++) //Boucle en initialisant i à 0 tant que i<10 et incrémente de 1 à chaque passage
{
    //Suite d'instructions à réaliser dans la boucle
}
```

Comparaison programme sans et avec boucle

Sans boucle	Avec boucle
<pre>void setup() { pinMode(2,OUTPUT); pinMode(3,OUTPUT); pinMode(4,OUTPUT); pinMode(5,OUTPUT); pinMode(6,OUTPUT); pinMode(7,OUTPUT); pinMode(8,OUTPUT); pinMode(9,OUTPUT); pinMode(10,OUTPUT); pinMode(11,OUTPUT); pinMode(12,OUTPUT); pinMode(13,OUTPUT); } void loop() { digitalWrite(2,HIGH); digitalWrite(3,LOW); digitalWrite(4,LOW); ...</pre>	<pre>void setup() { for (int i = 2 ; i <= 13 ; i++) { pinMode(i,OUTPUT); } } void loop() { for (int i = 2 ; i <= 13 ; i++) { digitalWrite(i,HIGH); } delay(1000); for (int i = 2 ; i <= 13 ; i++) { digitalWrite(i,LOW); } }</pre>

Programmation : Introduction, Simulation, Arduino IDE

Condition



```
const int analogPin = A0; // pin that the sensor is attached to
const int ledPin = 13;    // pin that the LED is attached to
const int threshold = 400; // an arbitrary threshold level
```

```
void setup() {
```

```
  pinMode(ledPin, OUTPUT);
```

```
}
```

```
void loop() {
```

```
  // read the value of the potentiometer:
  int analogValue = analogRead(analogPin);
```

```
  // if the analog value is high enough, turn on the LED:
  if (analogValue > threshold) {
```

```
    digitalWrite(ledPin, HIGH);
```

```
  }
```

```
  else {
```

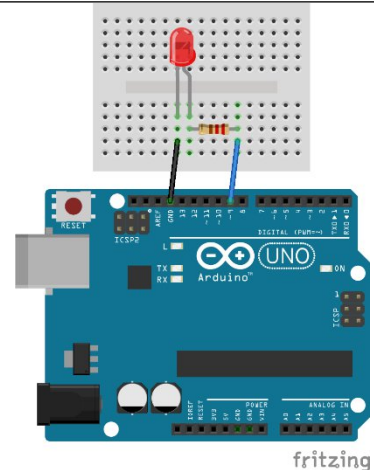
```
    digitalWrite(ledPin, LOW);
```

```
  }
```

```
  delay(1); // delay in between reads for stability
```

```
}
```

Boucle



```
const int ledPin = 9;    // pin that the LED is attached to
```

```
void setup() {
```

```
  pinMode(ledPin, OUTPUT);
```

```
}
```

```
void loop() {
```

```
  for (int i=0; i <= 255; i++) {
```

```
    analogWrite(ledPin, i);
    delay(10);
```

```
  }
```

```
  for (int i=255; i >= 0; i--) {
```

```
    analogWrite(ledPin, i);
    delay(10);
```

```
  }
```

```
}
```

Programmation : Introduction, Simulation, Arduino IDE