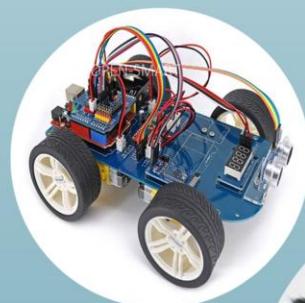


# Fiches d'accompagnement **ROBOTIQUE**

Tronc commun



Réalisé par :

-  EL HAYANI Chayma
-  ACHOUR Khaoula
-  AHMICHE Salma

Encadré par :

M. EL HAJJI Mohamed

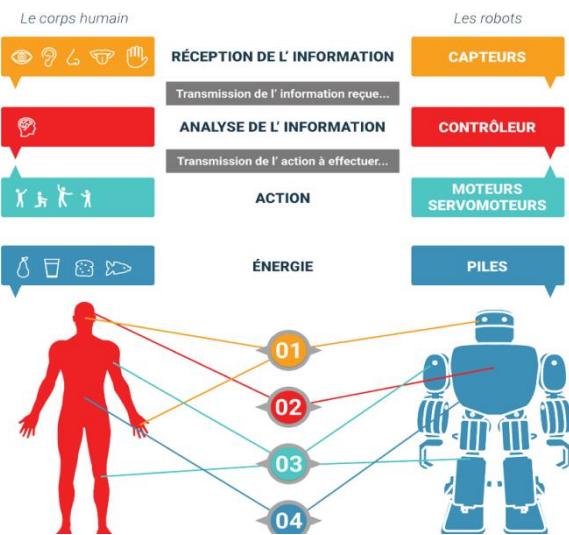
## COURS :

## INTRODUCTION A LA ROBOTIQUE :

Qu'est ce qu'un robot

Le terme robot fait partie de notre quotidien depuis les années 1960 où il apparaît dans l'électroménager. Pourtant, aujourd'hui encore définir un robot est moins facile qu'il n'y paraît, la question étant de déterminer les caractéristiques qui font basculer une machine dans le monde de la robotique. Il n'existe pas de définition unique néanmoins il est d'usage de dire qu'un robot est l'association indissociable de trois éléments :

- **Les capteurs** qui servent à appréhender l'environnement, éléments qui s'apparentent aux sens (la vue, le toucher,...)
- **L'unité de programmation** pour "raisonner", élément électronique qui fait office de cerveau.
- **Les actionneurs** pour interagir avec le monde réel, les "membres". Ces trois éléments donnent la capacité aux robots d'analyser leur environnement, de décider et d'agir sur le monde réel.
- **L'énergie** permet l'autonomie du robot.

Composition d'un robot :

Un robot est l'assemblage complexe de pièces mécaniques et de pièces électroniques pouvant être pilotées par une intelligence artificielle afin d'exécuter des actions autonomes.

*Les robots autonomes possèdent une source d'énergie embarquée (batterie, piles,...).*

**Les capteurs**

Il en existe une grande variété. Par exemple :

- **Capteurs à Infrarouge**
  - **Capteurs tactiles**
  - **Les sondeurs** (ou télémètres) à Ultra-son ou LASER. Ces derniers sont à la base des scanners laser permettant à l'unité centrale du robot de prendre "conscience" de son environnement en 3D.
  - **Les caméras** sont les yeux des robots. Il en faut au moins deux pour permettre la vision en trois dimensions. Le traitement automatique des images pour y détecter les formes, les objets, voire les visages, demande en général un traitement matériel car les microprocesseurs embarqués ne sont pas assez puissants pour le réaliser.
- Dans le cadre d'un robot roulant sur des roues, les **roues codeuses** permettent un déplacement précis en mesurant les angles de rotation. (Information proprioceptive).

**Les circuits électroniques**

Les microprocesseurs ou les microcontrôleurs sont des éléments primordiaux d'un robot, car ils permettent l'exécution de logiciels informatiques donnant son autonomie au robot. On trouve souvent dans un robot des modèles à très faible consommation, notamment pour des robots de petite taille, qui ne peuvent pas emporter avec eux une source d'énergie importante.

## Les actionneurs

Les actionneurs les plus usuels sont :

- des moteurs électriques rotatifs, qui sont fréquemment associés à des réducteurs mécaniques à engrenages et les servomoteurs.
  - des vérins hydrauliques, reliés par une tuyauterie à des pompes fournissant des pressions élevées.
- Généralement, un actionneur peut être considéré comme un constituant d'un système mécanique (exemple : bras, patte, roue motrice...) et correspond à un degré de liberté.
- Les interfaces haptiques permettent au robot de saisir des objets. Les moteurs permettent à des éléments mobiles de bouger suivant un ou plusieurs degrés de liberté; elles sont plutôt des constituants appartenant au domaine de la télémanipulation.

## Autonomie

On cherche à réaliser des systèmes capables de réagir seuls à l'environnement, c'est à-dire à un certain imprévu. C'est ce plus ou moins grand degré d'autonomie (permis par une intelligence artificielle) qui rapproche les robots des systèmes complètement autonomes envisagés par la science-fiction et la recherche de pointe.

Une certaine capacité d'adaptation à un environnement inconnu peut, dans les systèmes semi-autonomes

actuels, être assurée pourvu que l'inconnu reste relativement prévisible : l'exemple déjà opérationnel du robot aspirateur en est une bonne illustration

: le logiciel qui pilote cet appareil est en mesure de réagir aux obstacles qui peuvent se rencontrer dans une habitation, de les contourner, de les mémoriser. Il sauvegarde le plan de l'appartement et peut le modifier en cas de besoin. Il retourne en fin de programme se connecter à son chargeur. Il doit donc fournir une réponse correcte au plus grand nombre possible de stimulations, qui sont autant de données entrées, non par un opérateur, mais par l'environnement. L'autonomie suppose que le programme d'instructions prévoit la survenue de certains événements, puis la ou les réactions appropriées à ceux ci. Lorsque l'aspirateur évite un buffet parce qu'il *sait* que le buffet est là, il exécute un programme intégrant ce buffet, par exemple les coordonnées X-Y de son emplacement. Si ce buffet est déplacé ou supprimé, le robot est capable de modifier son plan en conséquence et de traiter une zone du sol qu'il ne prenait pas en compte jusqu'alors.

### Origine de la robotique

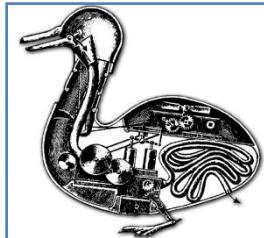
Le terme Robot a été créé en 1920 par Josef Capek, c'est la déclinaison de robota, mot tchèque signifiant "corvée". Sa première utilisation en a été faite par son frère Karel Capek dans la pièce de science fiction "Rossum's Universal Robots" jouée en 1922 à New York. Les robots de la pièce sont proches de ce qu'on appelle aujourd'hui des androïdes ou des clones, machines biologiques à l'apparence humaine.

Ce mot est vite adopté par le corps scientifique pour nommer un dispositif mécatronique alliant mécanique, électronique et informatique. Les robots ont pour fonction d'accomplir automatiquement des tâches dangereuses, pénibles, répétitives ou impossibles pour les humains.

La vulgarisation de ce terme entraîne aujourd'hui son utilisation pour évoquer la haute technicité d'un dispositif.

#### **Les premiers robots**

Les ancêtres des robots sont les automates. Un automate très évolué fut présenté par Jacques de Vaucanson en 1738 : il représentait un homme jouant d'un instrument de musique à vent. Jacques de Vaucanson créa également un automate représentant un canard mangeant et refoulant sa nourriture après ingestion de cette dernière. Néanmoins les automates sont bien plus anciens et l'on peut citer le lion automate conçu par Léonard de Vinci et présenté en 1515 à François 1<sup>e</sup>



Canard de Vaucanson



Le lion de Leonardo da Vinci



Joueur de flute Vaucanson

### La conception d'un robot

Les concepteurs de robots, quelqu'en soient les formes et les fonctions, sont confrontés à de nombreuses problématiques, la robotique est un concentré de technologies complexes et variées faisant appel à différents domaines tel que la programmation, la mécanique, la mécatronique, la cognitive, le design...

Les problématiques techniques rencontrées sont :

- **L'intelligence** du robot qui renvoie au domaine de la programmation et des logiciels.
- **L'architecture** qui peut se définir comme l'art de choisir les meilleures façons de structurer et combiner les composants du robot tel que les robots industriels de soudage qui doivent assembler des tôles très complexes et auront des bras avec 4 à 5 articulations.
- **Les actionneurs et préhenseurs** tels que les moteurs, les vérins, les pinces,... et tous les composants nécessaires à leur commande et au contrôle de leur position.
- **Les capteurs**, organes par lesquels le robot perçoit son environnement, qui doivent à la fois donner aux robots des informations de base sur son environnement (température, proximité d'un obstacle...) et lui permettre d'acquérir des informations plus fines (micro pour prendre en compte des instructions orales, ou caméra pour identifier un objet ou un obstacle).
- **La communication** du robot qui n'est pas limitée à l'échange d'informations entre le robot et l'usager, il est aussi en lien avec d'autres appareils (ordinateur, équipements domotiques...) situés dans son environnement de travail entraînant des questions de protocole de transmission et de reconnaissance mutuelle.
- **L'énergie**, le plus souvent l'électricité par alimentation directe, qui reste également un enjeu important pour optimiser la consommation et augmenter l'autonomie des robots.

## Usages des Robots

- **Robots domestiques**: tâches ménagères
- **Robots mobiles** : capables de se déplacer dans leur environnement, avec manipulateurs ou non.
- **Humanoïdes** : apparence d'un bipède
- **Robot industriel**
- **Robotique de service**
- **Robotique d'assistance à la personne**
- **Robotique à la santé**
- ...
- **robot collaboratif** : hommes et robots travaillent ensemble pour faciliter le travail de l'opérateur.  
Précision des mouvements, diminution de la pénibilité

## Métiers de la Robotique

### **Préambule**

Si dès à présent nous pouvons lister les métiers autour de la robotique, ce domaine n'en étant qu'à ses balbutiements les besoins et les métiers vont énormément évoluer dans les quelques années à venir.

*Le nombre de robots est exponentiel et il est encore difficile aujourd'hui d'imaginer l'explosion de la robotique dans les décennies à venir. Ce qui est sûre c'est que la « Robolution » est en marche et qu'il est désormais indispensable de s'y intéresser plutôt que d'en avoir peur.*

## Didacticiel installation de la carte ARDUINO :

### Installation de la carte Arduino Uno

La carte Arduino est une carte électronique open source et simple d'utilisation. Elle est développée par un italien nommé Massimo Banzi. Cette carte permet de rendre l'électronique accessible au plus grand nombre. Le modèle de base permet de contrôler 14 entrées/sorties digitales dont 6 sorties PWM(Modulation de Largeur d'Impulsion) pour faire varier la tension de sortie, et 6 entrées analogiques grâce à un microcontrôleur ATMEGA-328. Ressources en ligne.

Ressources	PDF	NOTES
Présentation de la carte ARDUINO	<a href="https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:presentation_arduino.pdf">https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:presentation_arduino.pdf</a>	
Installation d'Arduino sous windows	<a href="https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:installation_arduino_ardublock_windows.pdf">https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:installation_arduino_ardublock_windows.pdf</a>	Ardublock facultatif pour le moment
Installation d'Arduino sous Mac	<a href="https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:installer_arduino_sur_un_mac.pdf">https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:installer_arduino_sur_un_mac.pdf</a>	Ardublock facultatif pour le moment
Installation d'Arduino sous GNU/Linux	<a href="https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:installation_arduino_linux.pdf&amp;php?media=robotsarduino:installation_arduino_linux.pdf">https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:installation_arduino_linux.pdf&amp;php?media=robotsarduino:installation_arduino_linux.pdf</a>	Installation orientée Raspberry

Les types des robots :**Robot Mê khano centré :**

Robots industriels pour manipulation de pièces ou travail spécifique sur une chaîne de fabrication.



Robot Tripod

**Robot Anthropocentré - Humanoïde :**

Un **robot humanoïde** ou androïde est un robot dont l'apparence générale rappelle celle d'un corps humain. Généralement, les robots humanoïdes ont un torse avec une tête, deux bras et deux jambes mais certains modèles ne représentent qu'une partie du corps, par exemple à partir de la taille. Certains robots humanoïdes peuvent avoir un « visage », avec des « yeux » et une « bouche ».



Robot NAO



Robot Poppy



Robot Atlas de Boston Dynamics

Robot zoo ou bio centré :

Robots qui ont des formes des animaux.



Robot de Boston Dynamics



Robot Snake

Quelques robots faits avec une carte Arduino :



Sources pour plus d'informations :

<http://duino4projects.com/>

<https://seriousrobotics.files.wordpress.com>

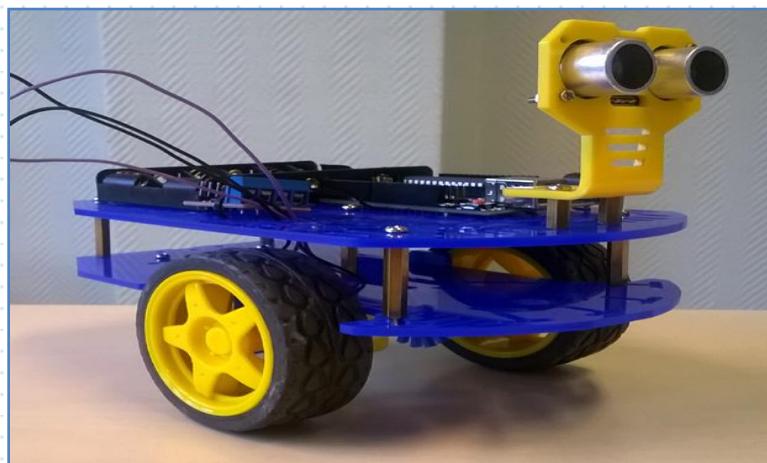
<http://cdn.instructables.com/>

## Fiches pour le montage:

### Montage du châssis ROSA :

#### Introduction :

Le Magician chassis est une plateforme robotique facile à assembler soi-même. Il propose un support 6 piles AA pour l'alimentation. Une fois le châssis monté, on peut l'équiper de capteurs ou d'actionneurs pour en faire un véritable robot intelligent et autonome.

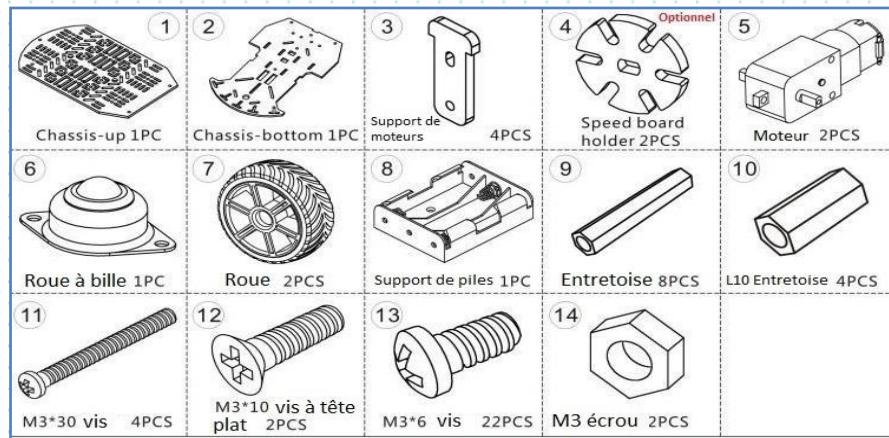


Robot ROSA déjà monté

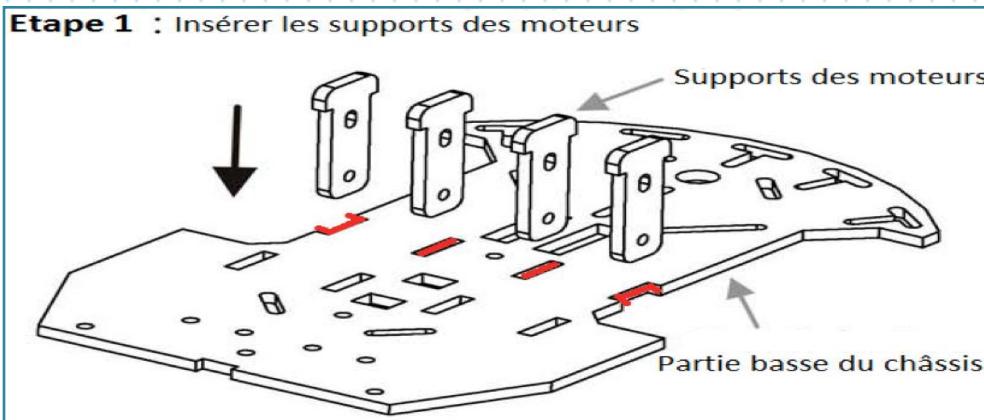
#### Contenu

- 2 plaques en acrylique de 3mm découpées au laser
- 2 moteurs DC 6V double axe
- 2 roues complètes
- 1 roue à bille
- 1 support de 6 piles AA
- 1 support ultrason
- 1 ensemble de quincaillerie, vis et écrous
- 1 ensemble de fil de câblage

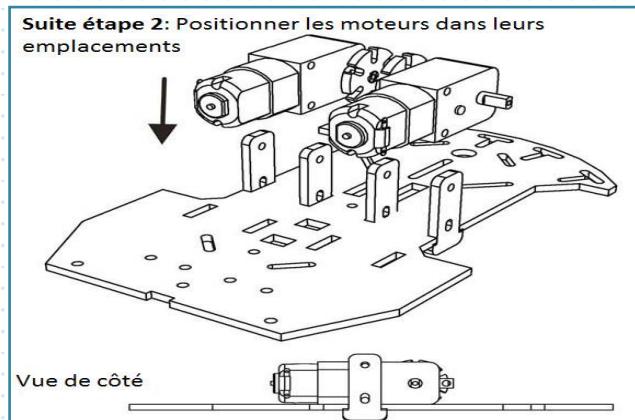
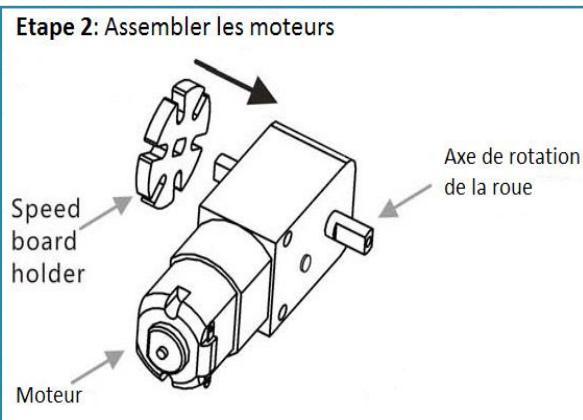


**Liste des composants pour la première partie du robot:****1ère étape :**

Positionnez les quatre supports des moteurs à leur emplacement respectif. Ils seront ensuite fixés sur le châssis une fois les moteurs positionnés et vissés.

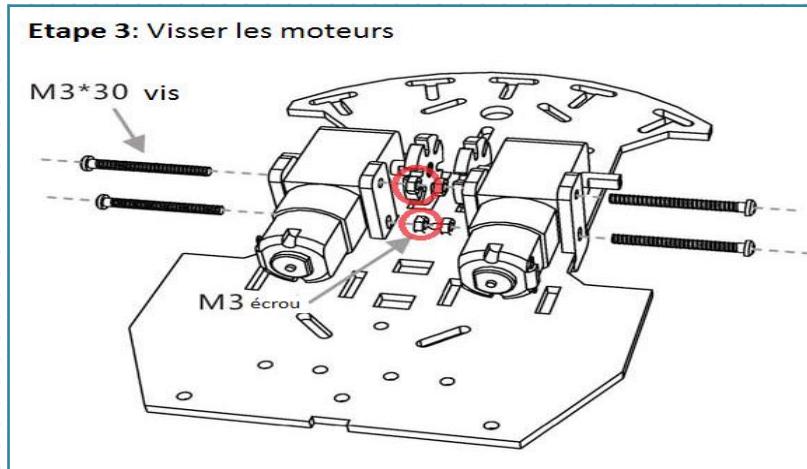
**2ème étape :**

Retournez le châssis. Assemblez le « speed board holder » sur l'axe de rotation de la roue de chaque moteur. Il ne faut pas les enfoncer entièrement pour pouvoir ensuite les positionner correctement et éviter qu'elles frottent contre le châssis. Cette pièce vous permettra à l'aide d'un capteur additionnel de mesurer la vitesse de rotation du moteur.

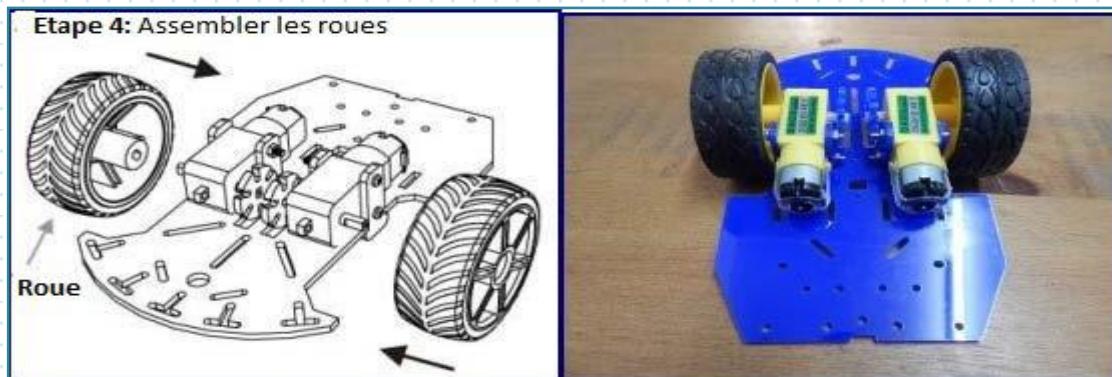


**3ème étape :**

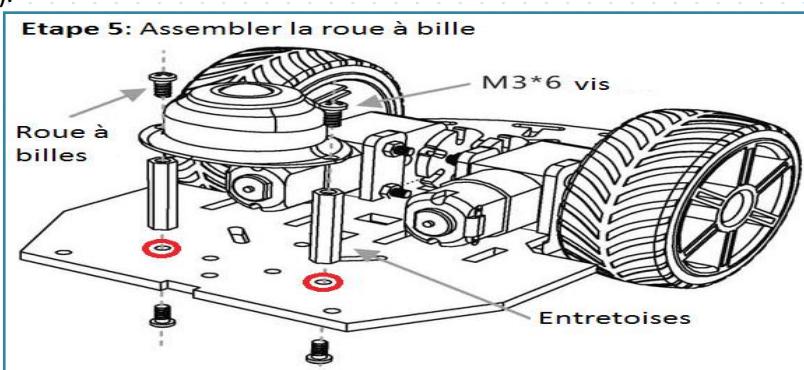
Une fois le moteur positionné sur le châssis entre les deux supports, on peut visser l'ensemble avec les vis M3\*30, deux vis par support. Attention, les vis peuvent être un peu difficiles à insérer. Les animateurs peuvent donc aider les enfants pour visser les moteurs. Les deux vis de bas sont optionnelles.

**4ème étape :**

Ajoutez les deux roues de chaque côté des moteurs, sur les axes de rotation. De même, évitez de les emboîter entièrement pour ne pas qu'elles frottent contre le châssis.

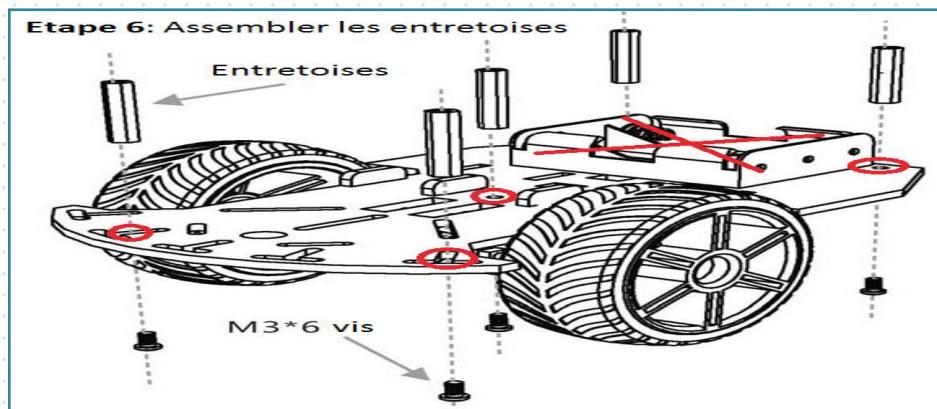
**5ème étape :**

On vient à présent fixer la roue à bille sur le châssis. Comme indiqué sur la notice du montage, on vient placer la roue sur deux entretoises (L25 spacers) et on visse l'ensemble avec quatre petites vis (M3\*6 screws).

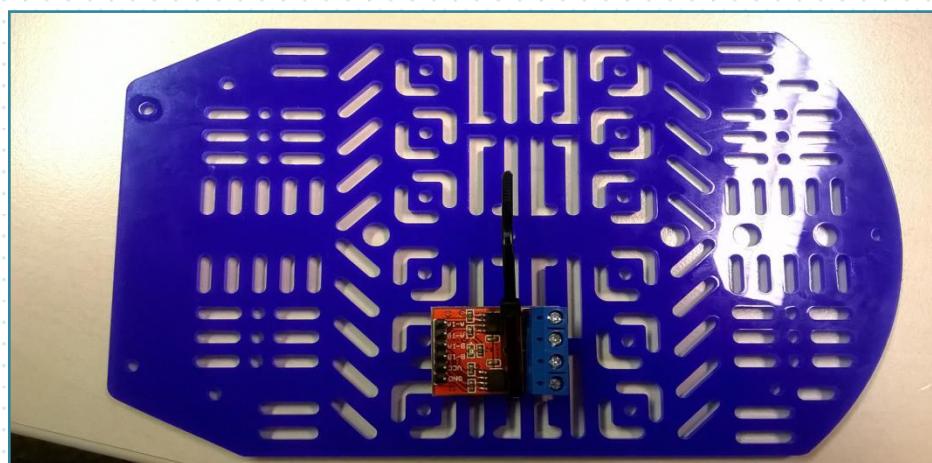


**6ème étape :**

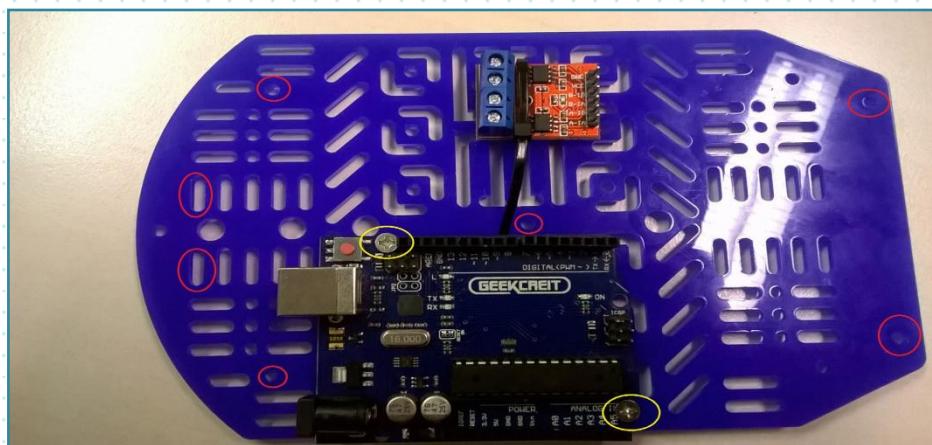
Pour pouvoir ensuite fixer la partie supérieure du châssis, on ajoute 5 entretoises (L25 spacers) sur notre montage. On vient visser ses entretoises avec les vis M3\*6 aux emplacements indiqués par des points rouges sur la photo ci-dessus. Pour les deux du haut, il faut bien les visser sur la « Base » du T pour pouvoir positionner correctement la plaque supérieure du châssis dans l'étape suivante.

**7ème étape :**

Fixer le contrôleur de moteur sur la partie haute du châssis en utilisant le support noir

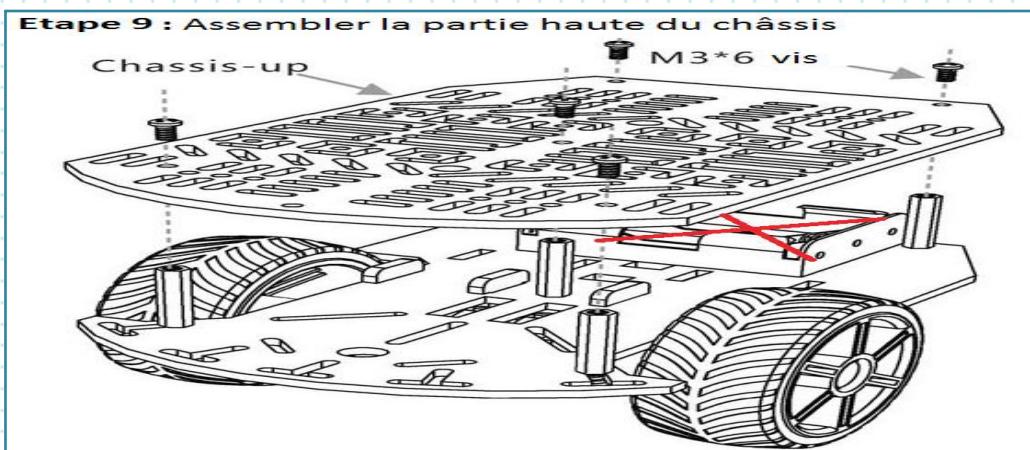
**8ème étape :**

Visser la carte Arduino (montrer en jaune sur la photo). Vous aurez besoin de deux vis à tête plate. Attention à ne pas cacher les trous pour les 7 vis (montrer en rouge sur la photo) dont on aura besoin plus tard.



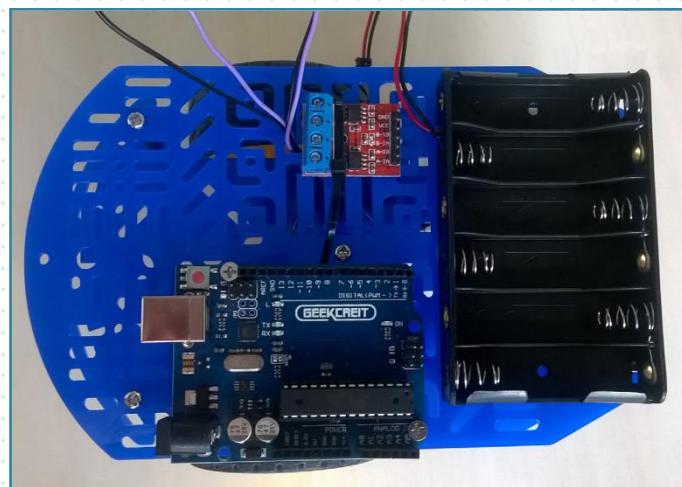
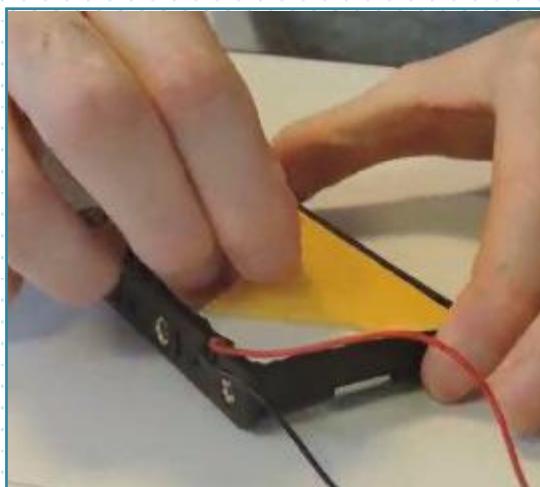
**9ème étape :**

On peut maintenant venir fixer la plaque supérieure du châssis sur les entretoises à l'aide de cinq vis M3\*6.

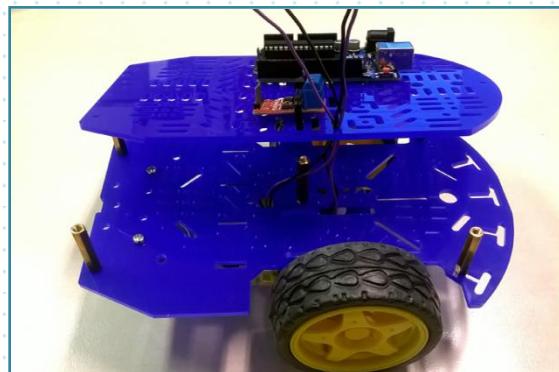
**10ème étape**

Enfin, vous pouvez coller ou visser le support de piles à la partie basse ou haute du châssis. Nous vous conseillons la partie haute car c'est plus pratique pour changer les piles.

Il peut être maintenu par deux vis à tête plate (M3\*10). Il faut forcer un peu pour faire rentrer les vis dans les trous déjà réalisés dans le support de piles. Sinon, l'utilisation du scotch double face est bien plus pratique pour retirer le support lors du changement des piles.

**11ème étape :**

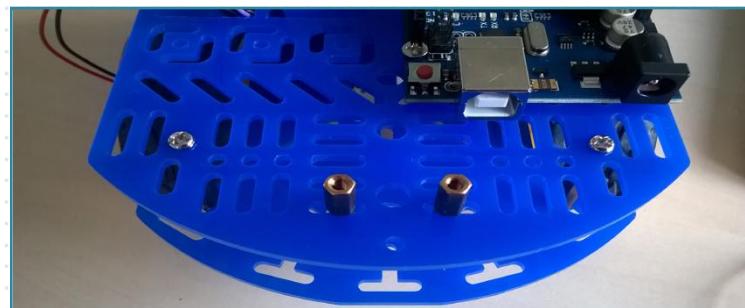
Passez les fils (2 violettes et 2 noirs) des moteurs via les deux parties du châssis. Il faut qu'ils soient près du contrôleur de moteurs.



**12ème étape :****Deuxième partie : montage de la tête du robot.**

Pour cet étape vous avez besoin de 4 vis M3\*6 et 2 entretoises.

Fixez d'abord les deux vis avec les deux entretoises sur l'avant du châssis. Conseil : ne vissez pas trop fort pour que vous puissiez bouger les entretoises afin de fixer le support de la tête.



Ajoutez ensuite, le support de la tête est fixez-le en utilisant les deux vis restantes.



Pour finir, vissez le capteur de distance, en utilisant deux vis M3\*30 et deux écrous.



Conseil : vous pouvez faire cet étape après la séance 3 « câblage » de ROSA.



**Et voilà, vous avez réussi !  
ROSA est prête !**

## La transformation des mouvements :

### Translation

Une translation est une transformation géométrique qui correspond à l'idée intuitive de « glissement » d'un objet, sans rotation, retournement ni déformation de cet objet.

### Rotation

La rotation (du latin rotare : «tourner») est le mouvement d'un corps autour d'un point ou d'un axe.

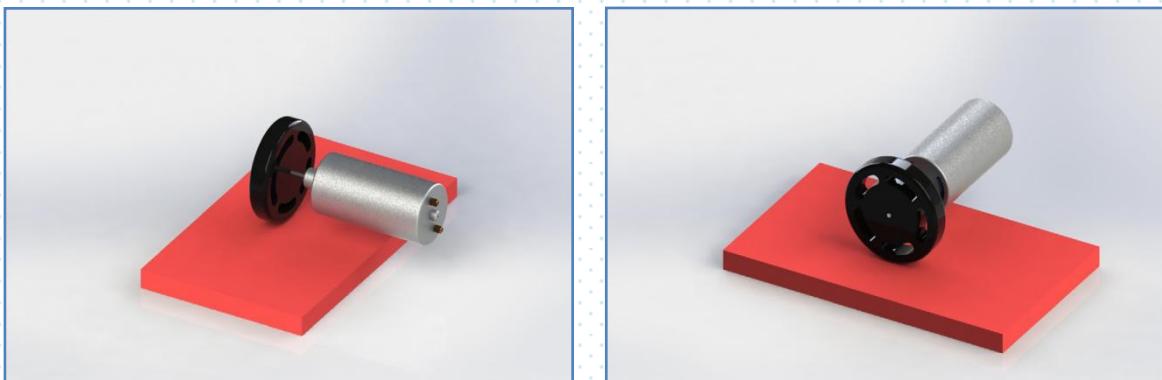
La rotation d'un moteur caractérise de mouvements circulaires.

Les transformations de mouvements qui nous intéressent en robotique, sont des systèmes qui utilisent la rotation d'un moteur électrique (l'axe du moteur tourne) pour réaliser un mouvement de translation (gauche à droite, haut en bas, avant en arrière).

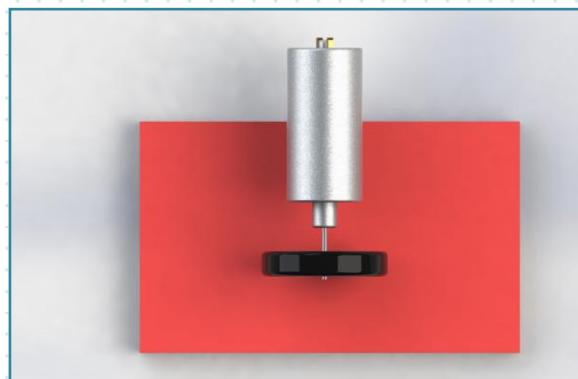
### La roue

Le système le plus connu est la roue. Si on colle une roue à l'axe du moteur, la roue tourne et fait avancer ou reculer un véhicule. Le moteur doit être collé au châssis du véhicule.

Voici une roue vue de côté. La roue déplace le châssis sur un sol fixe

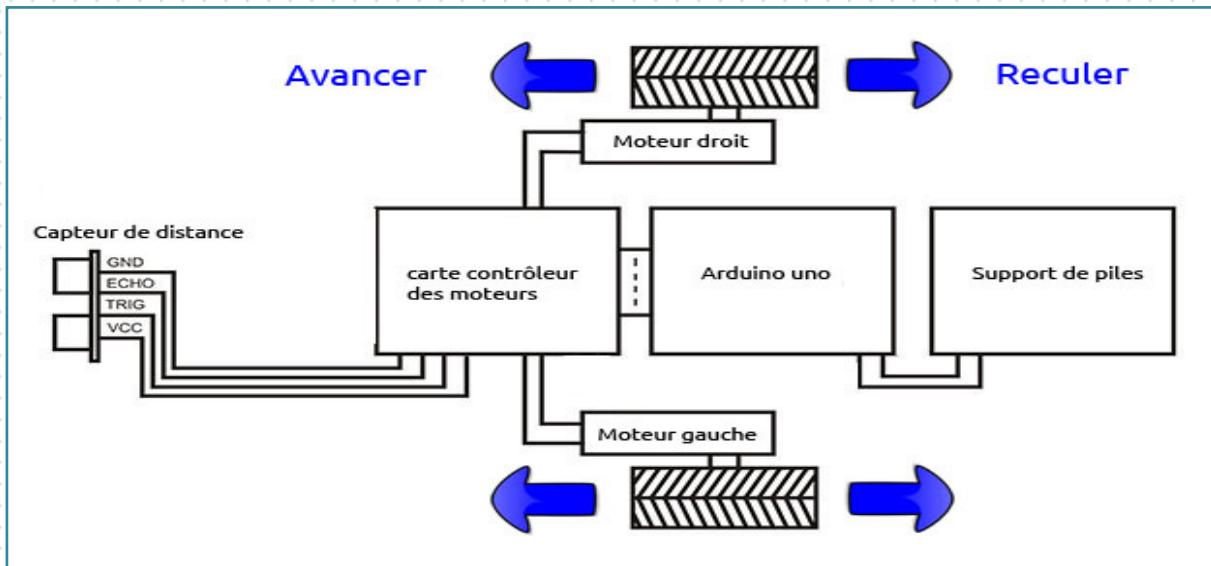


Vu de haut, la roue est collée à l'axe du moteur.



Ce système est utilisé dans les voitures et les trains. Les roues tournent et font avancer le véhicule sur la route.

## Les déplacements d'un robot



### Le robot peut:

- Avancer
- Reculer
- Tourner à droite
- Tourner à gauche
- Tourner sur place

Le tout plus ou moins rapidement, en changeant la vitesse des moteurs dans le code.

## Test des moteurs avec une pile

Pour tester le sens des moteurs, il suffit d'utiliser pile AA ou 9v. L'inversion du sens de rotation d'un moteur s'effectue en inversant la polarité de la pile.

### Lecture de la vidéo

[http://rosaliph.bzh/test\\_moteurs.html](http://rosaliph.bzh/test_moteurs.html)

### Télécharger la vidéo

[http://rosaliph.bzh/videos/test\\_moteurs.zip](http://rosaliph.bzh/videos/test_moteurs.zip)

## COURS :

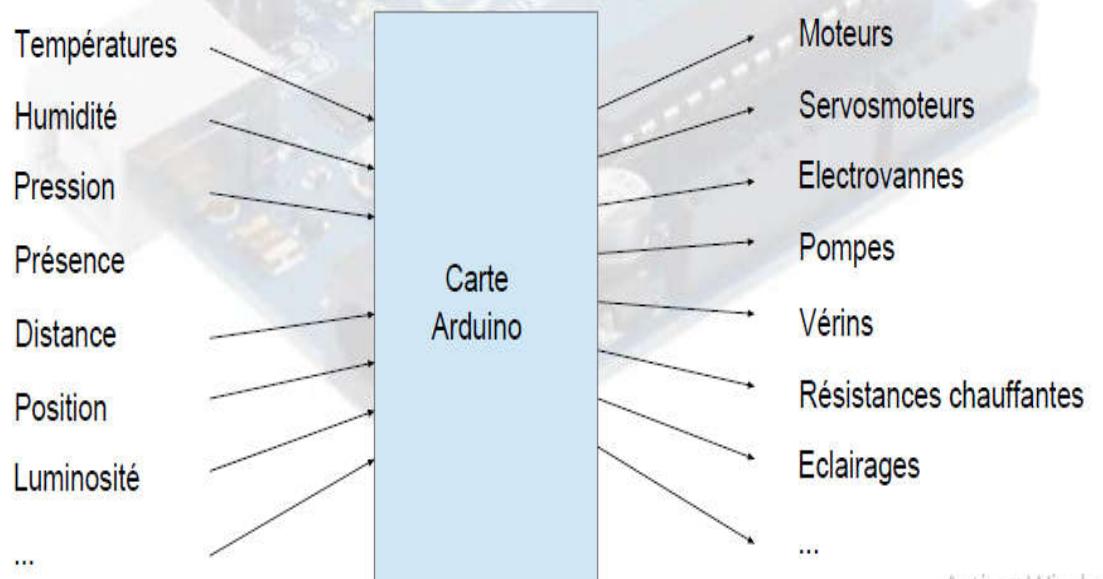
## Présentation Arduino

## Pour quoi faire ?

ARDUINO

Dans le parcours « robotique » nous utilisons la carte Arduino pour faire interagir le robot avec l'environnement.

Arduino peut interagir avec le monde réel:



Activer Windows  
Accédez aux paramètres pri



## Pour qui?



- Le projet « Arduino » a été initié par un groupe d'enseignants et d'étudiants d'une école de design italienne en 2004 – 2005.
- 
- Les utilisateurs d'Arduino sont :
  - des « bidouilleurs » dont beaucoup ont des connaissances très limitées en électronique ;
  - des artistes qui ont besoin d'animer leurs œuvres ou de créer des interactions avec elles ;
  - des étudiants et des élèves ;
  - et bien sûr, des animateurs, des enfants et des jeunes, dans les différents temps éducatifs et de loisirs

Activer Window  
Accédez aux param

## C'est quoi?



Une plate-forme de développement et de prototypage Open Source.

Le rôle de la carte Arduino est de stocker un programme et de le faire fonctionner.

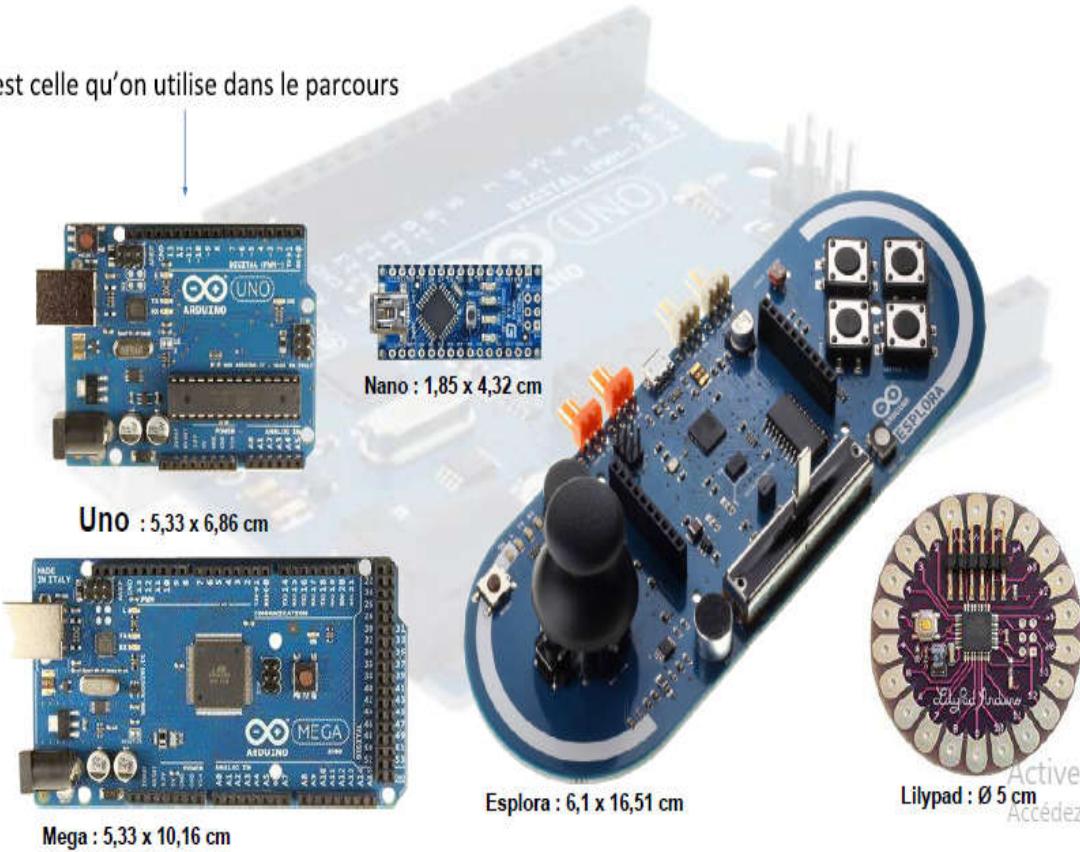
- **Shields** (cartes d'extension) qui permettent d'ajouter des fonctions qui s'enfichent sur la carte Arduino :
  - Commande pour faire tourner les moteurs, lecteur carte SD...
  - Ethernet, WIFI, GSM (téléphone portable), GPS...
  - Afficheurs LCD...
- **IDE** (Environnement de Développement Intégré) multi OS (système d'exploitation) :
  - édition du programme
  - compilation du programme
  - transfert du programme dans la carte via le port USB

Activer Windov  
Accédez aux param

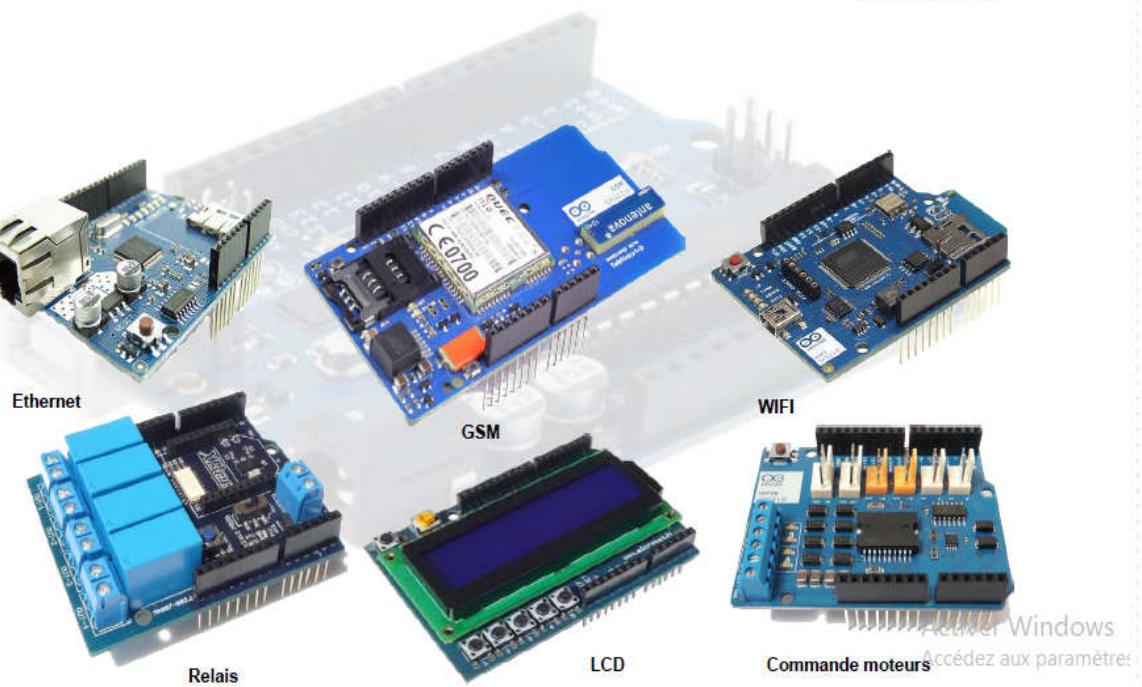
# Diverses Cartes Arduino



C'est celle qu'on utilise dans le parcours



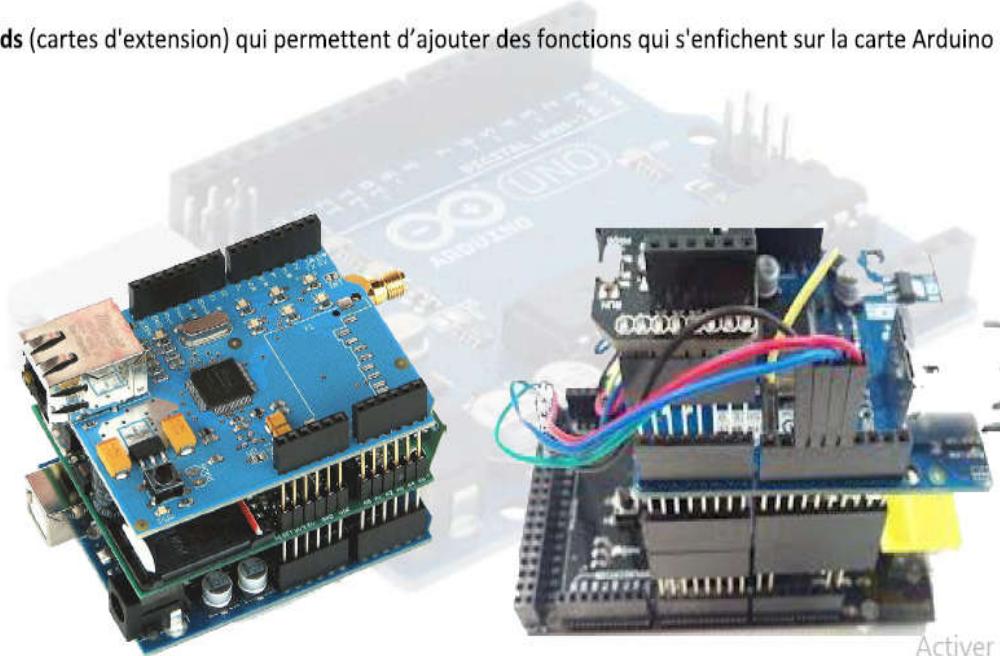
# Divers Shields Arduino



# Carte Arduino + Shields



**Shields** (cartes d'extension) qui permettent d'ajouter des fonctions qui s'enfichent sur la carte Arduino



## Activer Windows

## C'est quoi?



Un environnement de développement intégré fonctionnant sur divers systèmes d'exploitation (*Windows, Mac OS, Gnu/Linux*) qui permet d'éditer le programme sur un ordinateur et de le transférer via le port USB.

The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.0.5+dfsg2". The menu bar includes "Fichier", "Édition", "Croquis", "Outils", and "Aide". Below the menu is a toolbar with icons for file operations. The main window displays the "Blink" sketch. The code is as follows:

```
/*
 * Blank
 * Turns on an LED on for one second, then off for one second, repeat
 *
 * This example code is in the public domain.
 */

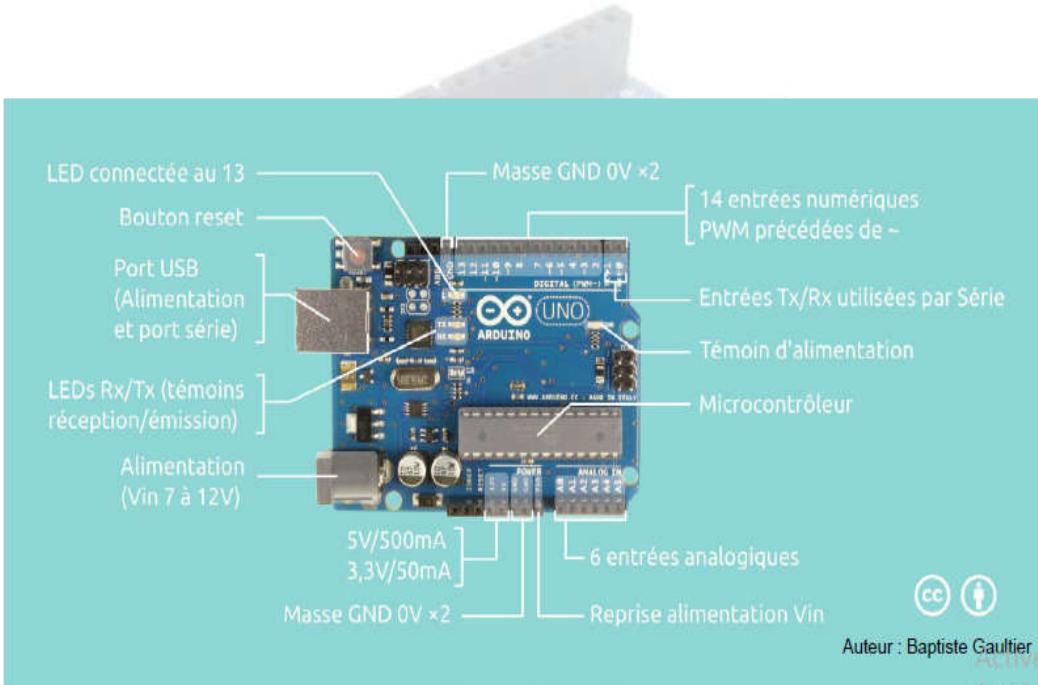
// Pin 13 has an LED connected in most Arduino boards,
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
    // initialize the digital pin as an output:
    pinMode(led, OUTPUT);
}

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

Activer Windows

# En quoi cela consiste?



## Quel langage de programmation ?



- 
- Langage proche du C
- 
- Programme structuré :
  - une section « setup » 1 seule exécution après RàZ (remise à zéro);
  - une section « loop » exécutée indéfiniment en boucle.

Setup

Loop

- De très nombreuses librairies logicielles disponibles.

Activer Windo  
Accédez aux paramètres

# Programmation avec Arduino

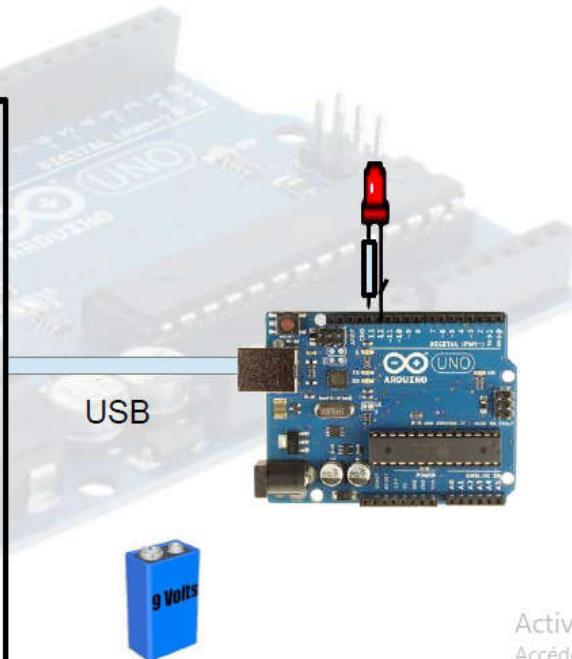


PC

```
Sketch: Blink | Arduino 1.0.1
File Edit Sketch Tools Help
Blink
int LED = 12;

void setup() {
  pinMode(LED, OUTPUT);
}
void loop() {
  digitalWrite(LED, LOW);
  delay(500);
  digitalWrite(LED, HIGH);
  delay(500);
}

Done Saving
1 Arduino Duemilanove (ATmega328) on CDM8
```

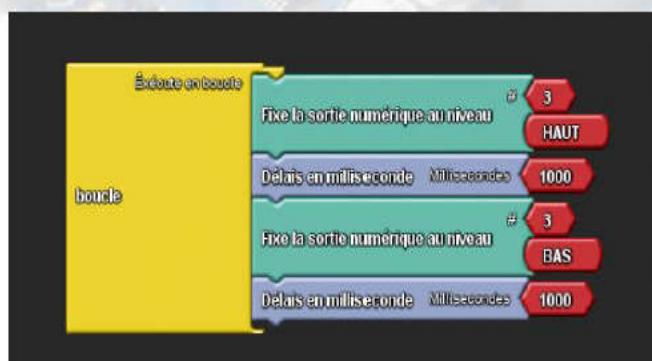


Activer Windc  
Accédez aux parar

## D'autres outils de programmation



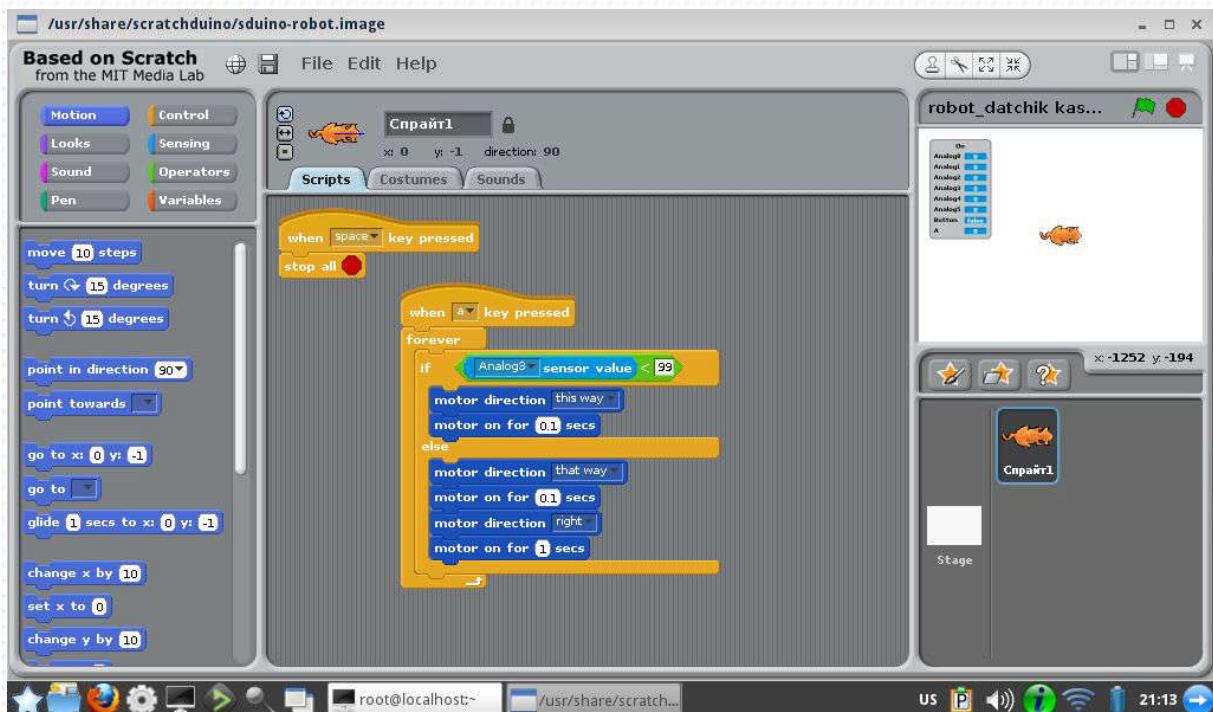
- Ardublock (programmation en mode graphique)
- C'est un outil qui se greffe au logiciel Arduino. Il suffit de créer des blocs et de les paramétriser. Ce logiciel est vraiment un outil de qualité pour démarrer facilement sur Arduino, sans connaissances en programmation.



Activer Windov  
Accédez aux param

## D'autres outils de programmation

- Scratch pour Arduino(programmation en mode graphique)
- Permet de piloter un Arduino à partir du code SCRATCH et de ce fait rend accessible à tout public la programmation d'un robot à partir d'un environnement aussi ludique, visuel et intuitif que celui de SCRATCH.



## Arduino : faut-il des connaissances en électronique ?

- Pas ou peu si on utilise des cartes et des modules tout faits.
- 
- La communauté francophone est très active sur le forum => entraide, tutoriels, exemples de réalisations...
- 
- Il faut des connaissances en électronique si on veut optimiser ou faire du sur-mesure.

## 5.4 – Programmation : OBJECTIF « Allumer la LED quand un obstacle se trouve devant le capteur »

*Déclaration et Initialisation setup()*

```

1 // définitions et déclarations
2 #include <NewPing.h>
3 #define trigPin 12
4 #define echoPin 11
5 NewPing distanceCM (trigPin,
6 echoPin);
7 int maximumDistance = 50;
8 int minimumDistance = 0;
9 boolean PasObstacle = false;
10 const int distanceObstacle = 15;
11 int attenteCapteur = 100;
12
13 // the setup function
14 Serial.begin(115200);
15 pinMode(13, OUTPUT);
16 digitalWrite(13, LOW);
17 delay(300);

```

**Déclarations nécessaires pour l'utilisation du capteur**

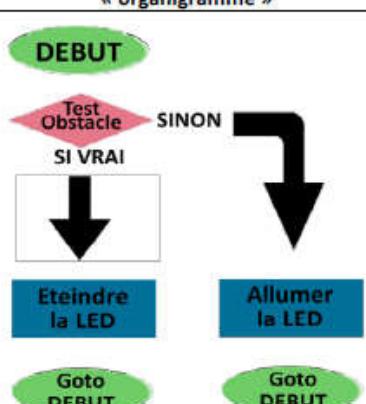
Déclaration de la library NewPing.h du capteur  
la broche (pin) Trigger est branchée sur la broche 12 de l'Arduino  
la broche (pin) Echo est branchée sur la broche 11 de l'Arduino  
on initialise la fonction distanceCM

distance maximale acceptée (en cm)  
distance minimale acceptée (en cm)  
valeur à false (ou 0) si pas d'obstacle détecté sinon true (ou 1)  
on définit la distance de détection d'un obstacle, ici 15 cm  
valeur en ms de l'attente de détection du capteur

**Initialisations nécessaires pour la LED**

On initialise la vitesse du moniteur série à 115 200 Bauds  
On initialise la Pin 13 (pour la LED) en Sortie  
On éteint la LED  
On attend 0.3 sec avant de démarrer le programme

*Programme principal loop()*

Représentation Graphique « organigramme »	Programme en C++	Description
 <pre>         DEBUT         ↓         Test Obstacle         SI VRAI         ↓         Eteindre la LED         Goto DEBUT         ↓         SINON         ↓         Allumer la LED         Goto DEBUT     </pre>	<pre>     {     int distance = mesureDistance();     delay(attenteCapteur);     if (PasObstacle == false)     {         digitalWrite(13, LOW);     }     else     {         digitalWrite(13, HIGH);         Serial.print(distanceCM.ping_cm());         delay(500);     } }     </pre>	<p>Début du programme  Distance de l'objet, on met sa valeur dans variable distance.  TEST : Présence d'un obstacle ?  début  Si pas d'obstacle on éteint la LED  fin  Sinon  début  On allume la LED  Affichage : distance de l'obstacle (1)  On attend 0.5 sec  fin  Fin du programme ↳ début</p>

*Fonction spécifique : Création de la fonction qui va renseigner la variable PasObstacle*

```

1 unsigned int mesureDistance()
2 {
3     int cm = distanceCM.ping_cm();
4     if (cm > distanceObstacle || cm
5         <= minimumDistance)
6     {
7         PasObstacle = vrai;
8     }
9     else
10    {
11        PasObstacle = true;
12    }
13    return cm;
14 }

```

Déclaration de la variable mesureDistance  
DEBUT  
On déclare et on renseigne la variable cm. (1)  
Si la distance à l'obstacle est sup. à la valeur renseignée dans distanceObstacle ou inf. à la valeur renseignée dans minimumDistance  
début  
SI VRAI on met vrai dans la variable PasObstacle  
fin  
SINON  
début  
On met true dans la variable PasObstacle  
fin  
La fonction retourne la distance de l'obstacle  
FIN

## CODE ARDUINO :

**Code (Arduino) « Pour allumer une LED quand un obstacle se trouve devant le capteur entre 10 et 20cm »**

```
/* Programme permettant d'allumer une LED (branchée sur la Pin 13 de la carte Arduino) lorsque le capteur de
distance détecte un objet à une distance comprise entre 10 et 20 cm. */

#include <NewPing.h>
// Serial.print(distanceCM.ping_cm()); // On affiche la distance de l'obstacle
#d Serial.println("cm");
#d delay(500); // pendant 500 ms
Ne }
int // the personal fonction
int unsigned int mesureDistance() // Déclaration de la variable mesureDistance
bo {
co int cm = distanceCM.ping_cm(); // déclaration de la variable locale (cm) à qui on assigne la distance
int if (cm > distanceObstacle || cm <= minimumDistance) // on définit la plage de détection du capteur
{
    PasObstacle = true; // on renvoie true si pas d'obstacle dans la plage définie dans le test
}
vo else
{
    PasObstacle = false; // sinon on renvoie false (si il y a un obstacle dans la plage définie)
Se
pir return cm; // on retourne la distance de l'obstacle à la fonction
dig
de-----}
}

// the loop function
void loop()
{
int distance = mesureDistance(); // on stock dans la variable distance la valeur du capteur en cm
delay(attenteCapteur); // attente en ms entre chaque mesure du capteur
if (PasObstacle == vrai) // S'il n'y a pas d'obstacle
{
    digitalWrite(13, LOW); // On éteind la LED
}
else // Sinon, un obstacle est détecté
{
    digitalWrite(13, HIGH); // On allume la LED
Serial.print("Distance: ");
}
```

## Code (Arduino) « du Robot ROSA »

```

/* Programmation : Si le robot rencontre un obstacle il s'arrête, recule puis tourne à droite et repart en avant */
#include <NewPing.h>

// définitions et déclarations des variables
// déclaration pour le capteur
#define trigPin 12
#define echoPin 11
NewPing distanceCM(trigPin, echoPin);
int maximumDistance = 50;
int minimumDistance = 10;
boolean PasObstacle = true;
const int distanceObstacle = 20;
int attenteCapteur = 100;
// déclaration pour les moteurs
#define vitesse_MG 150           // vitesse du moteur gauche
#define vitesse_MD 150           // vitesse du moteur droit
// Les moteurs ne tournent pas exactement à la même vitesse, on peut jouer sur les valeurs (Maximum 255 = 100)
int A0A = 5;                  // Branche A du moteur gauche connecté à la Pin 5 de l'Arduino
int A0B = 6;                  // Branche B du moteur gauche connecté à la Pin 6 de l'Arduino
int B0A = 9;                  // Branche A du moteur droit connecté à la Pin 9 de l'Arduino
int B0B = 10;                 // Branche B du moteur droit connecté à la Pin 10 de l'Arduino

// the setup function
void setup()
{
pinMode(A0A, OUTPUT);
pinMode(A0B, OUTPUT);
pinMode(B0A, OUTPUT);
pinMode(B0B, OUTPUT);
stopRobot();
delay(300);
}

// the loop function
void loop()
{
int distance = measureDistance();           // On stock dans la variable distance la valeur du capteur en cm
delay(attenteCapteur);                     // attendre un peu entre chaque mesure du capteur
if (PasObstacle == true)                   // Si il n'y a pas d'obstacle
{
    avanceRobot();                         // On lance la fonction avanceRobot()
}
else
{
    stopRobot();                          // On lance la fonction stopRobot()
    delay(300);                           // On attend 0.3 sec
    reculeRobot();                        // On lance la fonction reculeRobot()
    delay(300);                           // On attend 0.3 sec
    tourneDroite();                      // On lance la fonction tourneDroite()
    delay(300);                           // On attend 0.3 sec
}
}

/* Le code peut être modifié facilement avec les enfants en utilisant toutes les fonctions préédites : avanceRobot()
reculeRobot() tourneDroite() tourneGauche() reculerEnAvantGauche() reculerEnAvantDroite() stopRobot() */

```

```

// the personal function // Fonction de mouvement des moteurs
void avanceRobot() { // Fonction : Le robot avance
    analogWrite(AIA, vitesse_MG);
    analogWrite(AIB, LOW);
    analogWrite(BIA, vitesse_MD);
    analogWrite(BIB, LOW);
}

void reculeRobot() { // Fonction : le robot recule
    analogWrite(AIA, LOW);
    analogWrite(AIB, vitesse_MG);
    analogWrite(BIA, LOW);
    analogWrite(BIB, vitesse_MD);
}

void tourneDroite() { // Fonction : Le robot tourne à droite
    analogWrite(AIA, vitesse_MG);
    analogWrite(AIB, LOW);
    analogWrite(BIA, LOW);
    analogWrite(BIB, LOW);
}

void tourneGauche() { // Fonction : Le robot tourne à gauche
    analogWrite(AIA, LOW);
    analogWrite(AIB, LOW);
    analogWrite(BIA, vitesse_MD);
    analogWrite(BIB, LOW);
}

void robotSurPlaceGauche() { // Fonction : Le robot fait le tournis à gauche
    analogWrite(AIA, LOW);
    analogWrite(AIB, vitesse_MG);
    analogWrite(BIA, vitesse_MD);
    analogWrite(BIB, LOW);
}

void robotSurPlaceDroite() { // Fonction : Le robot fait le tournis à droite
    analogWrite(AIA, vitesse_MG);
    analogWrite(AIB, LOW);
    analogWrite(BIA, LOW);
    analogWrite(BIB, vitesse_MD);
}

void stopRobot() { // Fonction : Le robot s'arrête
    digitalWrite(AIA, LOW);
    digitalWrite(AIB, LOW);
    digitalWrite(BIA, LOW);
    digitalWrite(BIB, LOW);
}

// Fonction de capteur
unsigned int mesureDistance() { // Calculer la variable mesureDistance
    int cm = distanceCM.ping_cm(); // déclaration de la variable lente (cm) il faut lui assigne la distance
    if (cm > distanceObstacle || cm <= minimumDistance) // on définit la plage de détection du capteur
    {
        PasObstacle = true; // on renvoie true si pas d'obstacle dans la plage définie dans le test
    }
    else
    {
        PasObstacle = false; // sinon on renvoie false (si il y a un obstacle dans la plage définie)
    }
    return cm; // cm renvoie la distance de l'obstacle à la fonction
}

```

## COURS (La programmation expliquée) :

### LA PROGRAMMATION EXPLIQUEE AUX ENFANTS / ADOLESCENTS

#### 1. Est-ce qu'un ordinateur est intelligent ?

[Poser la question : Qui croit qu'il est plus intelligent qu'un ordinateur et faire un vote à main levée en demandant

à chacun d'expliquer sa position.

Puis demander à un des participants d'exécuter une série d'instructions :

- S'asseoir, se lever plusieurs fois d'affilée
- Avancer d'un pas, puis d'un autre pas, et ainsi de suite jusqu'à arriver devant le mur et continuer de demander au participant d'avancer d'un pas

Puis demander aux autres enfants, si exécuter ces tâches requièrent de l'intelligence.

#### Explication :

Un ordinateur calcule très vite, il peut répéter une action plusieurs millions de fois sans se lasser, mais il n'est pas intelligent.

La différence entre un humain et un ordinateur est que face à une situation nouvelle, l'être humain peut s'adapter : il essayera de trouver des similitudes entre son expérience et cette nouvelle situation, expérimentera, fera des suppositions, bref il peut improviser.

Alors qu'un ordinateur lui est incapable d'agir en dehors de ce pourquoi il a été programmé. C'est d'ailleurs un des enjeux du développement des intelligences artificielles.

#### 2. Vous avez dit programmation ?

Un ordinateur, on vient de le voir, ne fait qu'exécuter les instructions qu'on lui a données.

Derrière chaque programme de l'ordinateur, une personne lui a dit quoi faire et comment le faire.

Et cette personne qui lui a dit quoi faire, c'est un programmeur.

Que ça soit pour créer un jeu vidéo, ou un site internet ou même une application de téléphone : il y a toujours un ou plusieurs développeurs qui ont expliqué aux ordinateurs, téléphones, tablettes quoi faire.

#### 3. Qu'est-ce que le langage informatique ?

[Poser la question : qui sait quelle langue parle l'ordinateur ?]

Alors le souci quand on veut expliquer quoi faire à un ordinateur, c'est qu'il parle une langue qui s'appelle le binaire : des 0 et des 1 et rien d'autre !

Le binaire, l'ordinateur le comprend très bien mais pour les humains, c'est compliqué à parler.

[Voici un petit exemple :

J'ai un ami néerlandais qui parle néerlandais mais qui ne parle pas français et moi je parle français mais pas néerlandais. Comment pouvons-nous faire pour nous comprendre l'un l'autre ?

Peut-être que nous connaissons une langue commune. Effectivement, lui et moi parlons anglais : on va pouvoir se comprendre ! ]

Avec l'ordinateur c'est pareil, il comprend le binaire, moi pas, on va donc trouver un langage commun : un langage informatique que lui et moi pourrons comprendre.

Des langages informatiques, il y en a pleins mais ils ont tous la même fonction : donner des instructions.

#### 4. L'algorithme :

Mais même si on peut communiquer avec l'ordinateur, il a sa façon à lui de penser : le binaire. 0 et 1, on peut les assimiler à « Oui » et « Non ». Alors pour expliquer à quelqu'un qui ne comprend que les oui et non, comment faire pour jouer à *Super Mario* ? Il va falloir structurer ce qu'on lui demande, être clair, précis et méthodique. Et pour cela, on utilise les algorithmes.

Derrière ce nom un peu effrayant se cache en réalité quelque chose de très commun.

Un algorithme est une succession d'actions (difficile de faire plus bref comme définition).

Et le meilleur exemple d'algorithme : une recette de cuisine !

*Prendre des carottes.  
Les éplucher.  
Les râper.  
Les mettre dans un saladier.  
Ajouter de la vinaigrette.  
Servir.*

**Voici l'algorithme des carottes râpées.**

Alors vous vous doutez bien, que tous les algorithmes ne sont pas aussi simples. En réalité, il existe ce qu'on appelle des structures de contrôle qui permettent de les complexifier :

Voici les principales :

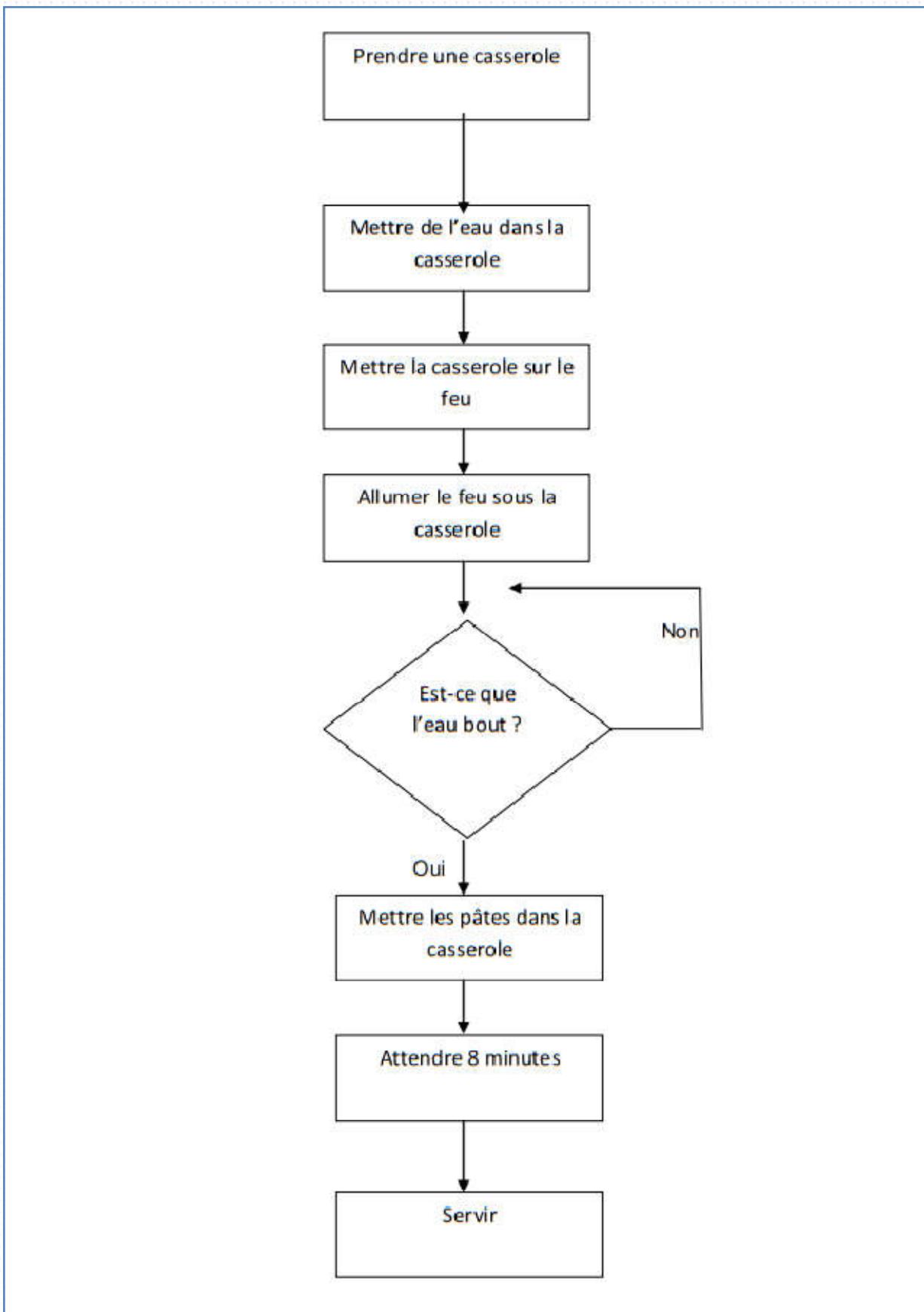
- Les conditions,
- Les boucles,
- Les variables.

**SEANCE 4**

**TRONC COMMUN**

**L'algorithme de la cuisson des pâtes**

Demandez à votre public d'écrire la recette pour faire des pates



## Les conditions :

Très souvent, nos actions dépendent de plusieurs paramètres :

*S'il fait beau → Je ne prends pas de pull.*

Les conditions permettent à un programme de faire une action en fonction d'une ou plusieurs informations. Les termes utilisés sont « Si » et « Sinon » (« If » et « Else » en anglais). Les conditions peuvent s'imbriquer les unes dans les autres.

*S'il fait beau et s'il fait chaud → Alors je sors en Tee-shirt.*

**Dans notre exemple :** Si l'eau bout, alors je mets les pâtes.

## Les boucles :

En programmation, un des maîtres-mots est **optimisation**. Alors souvent plutôt que de réécrire des lignes de codes, on demande à l'ordinateur de répéter une opération en utilisant une boucle.

Une boucle peut se répéter un certain nombre de fois, ou tant qu'une condition n'est pas vérifiée.

**Dans notre exemple :** tant que l'eau ne bout pas, il se repose en boucle la question (« Est-ce que l'eau bout ? »).

## Un autre exemple :

Imaginons que vous n'ayez une casserole ne pouvant faire des pâtes que pour 2 personnes et que vous êtes 4, il faudra donc répéter 2 fois, votre recette.

## Les variables :

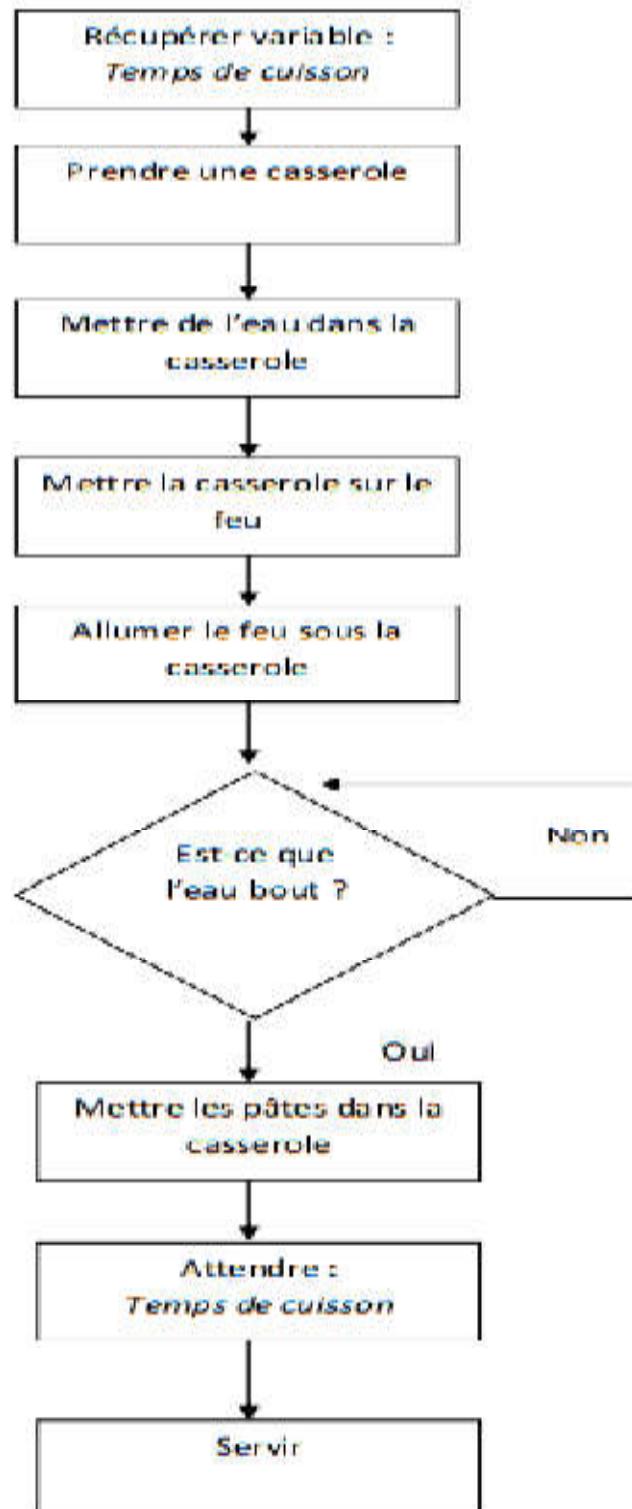
Les variables sont des espaces de mémoire dans l'ordinateur dans lesquels on peut lui demander de garder une information pour nous (soit sous la forme d'un nombre soit sous la forme de mots).

Dans un jeu vidéo, votre nombre de vie ou votre score sont des variables :

- Vous pouvez commencer avec votre variable « Vie » = 3,
- Quand vous touchez un ennemis, retire 1 à votre variable « Vie »
- Si vous tombé à variable « Vie » = 0, vous avez perdu

**Dans le cas de notre exemple :** la recette ne fonctionne que pour des pâtes ayant un temps de cuisson de 8 minutes. Pour du riz longue cuisson, le programme ne fonctionne pas.

Ce que nous allons donc faire, c'est qu'au début de notre recette, nous allons créer une variable que nous appellerons « temps de cuisson » qui va dépendre de ce que l'on veut faire cuire et après avoir versé le riz dans la casserole, on va « attendre : temps de cuisson »



## Algorithme VS organigramme de programmation

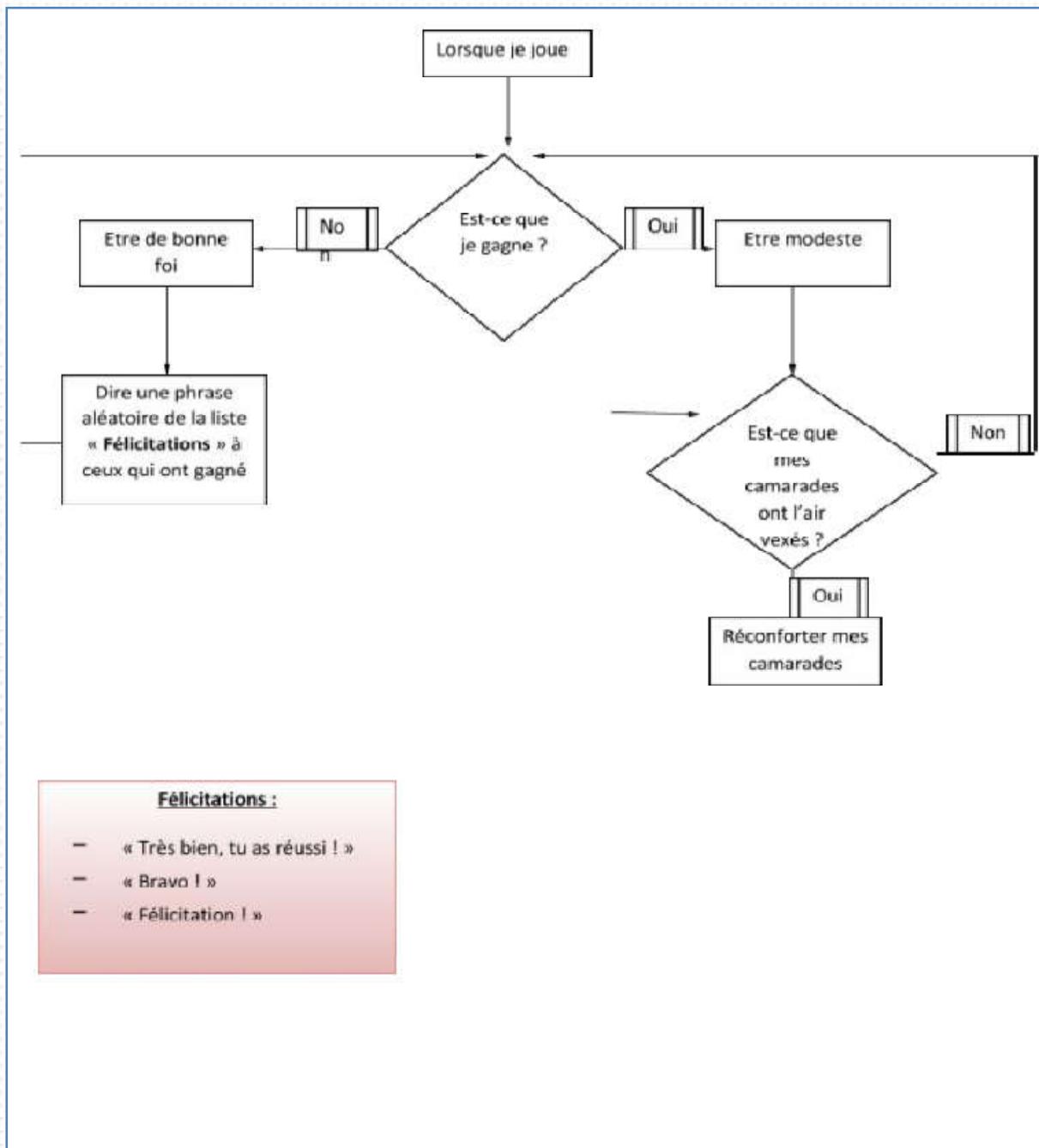
Un **algorithme** est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat (<https://fr.wikipedia.org/wiki/Algorithme>).

Un **organigramme de programmation** (parfois appelé algorigramme, logigramme ou plus rarement ordinogramme) est une représentation graphique normalisée de l'enchaînement des opérations et des décisions effectuées par un programme d'ordinateur.  
([https://fr.wikipedia.org/wiki/Organigramme\\_de\\_programmation](https://fr.wikipedia.org/wiki/Organigramme_de_programmation)).

Une **différence** significative entre algorithme et programme (représenté par un organigramme) est que l'exécution d'un algorithme doit toujours se terminer avec un résultat, alors que celle d'un programme peut conduire à une boucle infinie (ne jamais s'arrêter). On peut donc utiliser des organigrammes de programmation pour représenter un algorithme néanmoins un organigramme ne représente pas forcément un algorithme.

Pour aller plus loin: <https://openclassrooms.com/courses/introduction-aux-algorigrammes>

### Organigramme de programmation: bon(ne) perdant(e)



## Organigramme de programmation : La récréation



**COURS :****LA ROBOTIQUE :****Introduction**

Si on considère qu'un robot est un dispositif mécatronique (mécanique, électronique & informatique) pouvant réaliser des tâches programmées de manière autonome, alors il y a longtemps qu'ils sont parmi nous: lave-vaisselle, lave-linge, magnétoscope, sans même parler des robots tondeuses ou aspirateurs! Ce qui les distingue des simples appareils électromécaniques (comme une ancienne machine à laver), c'est leur gestion des tâches à l'aide d'un microcontrôleur. Il s'agit d'une sorte de petit ordinateur minimaliste et de faible puissance, mais aussi de faible consommation. C'est lui qui commande les tâches en fonction de son environnement. Dans le cas d'un lave-vaisselle moderne, il s'agit de commander de nombreuses actions (ouverture de vannes, pompes, chauffage de l'eau, ouverture du réservoir de poudre...) selon divers programmes, et en fonction de différents capteurs (porte fermée, arrivée d'eau ouverte, température de l'eau, débitmètre, choix du programme...) et de réagir en fonction de capteurs de pannes (pompe défectueuse, arrivée d'eau fermée, capteur antidébordement...). L'ensemble du dispositif informatique et électronique commandant le lave-vaisselle est appelé système embarqué.

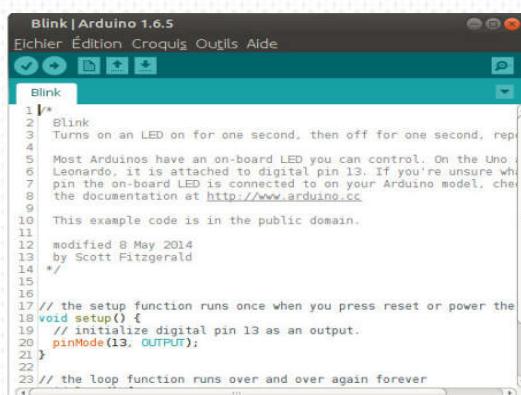
Arduino est une famille de cartes contenant un microcontrôleur et de nombreux connecteurs (entrées/sorties) qui peuvent recevoir des informations de capteurs et produire des signaux à même d'interagir avec des moteurs, des relais ou d'autres circuits électroniques.

Sa particularité est d'être entièrement Open Source, d'un faible prix et aisément programmable. Dès lors, Arduino est devenu une référence dans le monde des bricoleurs et bidouilleurs de toutes sortes. Il suffit de voir la diversité des réalisations présentées ici pour s'en rendre compte!

Arduino permet de faire une introduction à l'automation, à l'électronique, à la mécanique et à la programmation à moindre coût. Tout ceci nous amène à penser qu'il a parfaitement sa place sur Edurobot et à l'école!

**Présentation**

Le projet Arduino est intéressant à plus d'un titre; à commencer par son origine: l'Italie du Nord. Le magazine en ligne OWNI a publié un bon article sur l'origine et l'histoire d'Arduino; la lecture de cet article est vraiment recommandée, car elle permet d'aborder Arduino en comprenant la philosophie qui a mené au projet actuel.

**Installation de la bibliothèque NewPing**


```

Blink | Arduino 1.6.5
Fichier Édition Croquis Outils Aide
Blink
1 /*
2 * Blink
3 * Turns on an LED on for one second, then off for one second, repeating
4 * Most Arduinos have an on-board LED you can control. On the Uno and Leonardo, it is attached to digital pin 13. If you're unsure which pin the on-board LED is connected to on your Arduino model, check the documentation at http://www.arduino.cc
5
6 This example code is in the public domain.
7
8 modified 8 May 2014
9 by Scott Fitzgerald
10 */
11
12
13
14
15
16
17 // the setup function runs once when you press reset or power the board
18 void setup() {
19   // initialize digital pin 13 as an output.
20   pinMode(13, OUTPUT);
21 }
22
23 // the loop function runs over and over again forever

```

Bien qu'il soit possible de programmer la carte Arduino avec Ardublock, Scratch, ou encore Blockly nous allons utiliser l'environnement Arduino.

### Lancer l'IDE Arduino

### Késako une bibliothèque sous Arduino ?

Les bibliothèques (ou librairie) permettent d'appeler des fonctions toutes prêtées par rapport à un shield ou un capteur. La bibliothèque NewPing permet d'utiliser facilement le capteur de distance (le HC-SR04).

<b>Bibliothèque NewPing</b>	Télécharger le fichier : <a href="https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:newping.zip">https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:newping.zip</a>
<b>Installation d'une bibliothèque</b>	<a href="http://rosa.lph.bzh/installation_librairie.html">http://rosa.lph.bzh/installation_librairie.html</a> Télécharger le fichier : <a href="http://rosa.lph.bzh/videos/installation_librairie.zip">http://rosa.lph.bzh/videos/installation_librairie.zip</a>
<b>Code test des moteurs</b>	Télécharger le fichier : <a href="https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:test_moteurs.ino.zip">https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:test_moteurs.ino.zip</a>
<b>Code du robot</b>	Télécharger le fichier : <a href="https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:robot_ligue_l9110.ino.zip">https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:robot_ligue_l9110.ino.zip</a>

### Le code ROSA : Déclaration des constantes et variables

On définit les broches du capteur ultrason:

```
#include <NewPing.h> On importe la bibliothèque New Ping. Elle possède une fonction qui mesure la distance.
```

On définit les broches du capteur ultrason:

```
#define trigPin 12 la broche (pin) Trigger est branchée sur la broche 12 de l'Arduino  
#define echoPin 11 la broche (pin) Echo est branchée sur la broche 11 de l'Arduino
```

NewPing distanceCM (trigPin, echoPin); on initialise la fonction

```
int maximumDistance = 50; distance maximale acceptée (de 0-450 cm)
```

```
int minimumDistance = 0; distance minimale acceptée (en cm)
```

```
DistanceCM
```

```
boolean PasObstacle = false; valeur à false (ou 0) si pas d'obstacle détecté
```

```
sinon la valeur prend true (ou 1)
```

`const int distanceObstacle = 15;` on définit la distance de détection d'un obstacle, ici 15 cm  
`int attenteCapteur = 100;` valeur en ms

### Déclaration des variables pour la vitesse de chaque moteur

Les moteurs ne tournant pas exactement à la même vitesse, vous pouvez modifier les valeurs suivantes.

La valeur maximale acceptée est 255 (sans unité)

Analogie avec l'électricité : 255 correspond à 5 volts

`#define vitesse_MG 150` vitesse du moteur gauche

`#define vitesse_MD 150` vitesse du moteur droit

On déclare les broches 5, 6, 9, 10 en sortie pour piloter les 2 moteurs

Moteur A (moteur Gauche)

Moteur B (moteur Droit)

`int BIA = 9;` connecté à la broche 9 de l'Arduino

`int BIB = 10;` connecté à la broche 10 de l'Arduino

*\*Vous pouvez inverser les moteurs, en respectant bien les branchements sur la carte Arduino. Ex. le moteur gauche devient le moteur droit*

`int AIA = 5;` connecté à la broche 5 de l'Arduino

`int AIB = 6;` connecté à la broche 6 de l'Arduino

### La fonction SETUP

La fonction `setup()` est appelée au démarrage du programme. Cette fonction est utilisée pour initialiser les variables, le sens des broches, les librairies utilisées. La fonction `setup` n'est exécutée qu'une seule fois, après chaque mise sous tension ou reset (réinitialisation) de la carte Arduino.

```
void setup()
{
    Moteur gauche
    pinMode(AIA, OUTPUT);
    pinMode(AIB, OUTPUT);
    Moteur droit
    pinMode(BIA, OUTPUT);
    pinMode(BIB, OUTPUT);
    stopRobot(); le robot est à l'arrêt
    delay(300); pendant 300 ms
```

### La fonction LOOP

La boucle (`loop`) = le programme principal. Le code dans cette fonction est exécuté en boucle. C'est la partie qu'on utilisera avec les enfants.

```
void loop() On stocke dans la variable distance la valeur du
capteur en cm
{
    int distance = mesureDistance(); Attente en ms entre chaque mesure du capteur
```

```

delay(attenteCapteur);

if (PasObstacle == false)
{
    avanceRobot(); le robot avance.
}
else{Sinon, un obstacle est détecté,
stopRobot();on arrête le robot
delay(300);pendant 300 ms (cette valeur est modifiable)
reculeRobot();on recule le robot
delay(200);pendant 200 ms (on augmente la valeur
si on souhaite reculer plus longtemps)
tourneDroite();le robot tourne à droite
delay(200);pendant 200ms
}}Fin de la boucle principale

```

### **Les fonctions du robot préprogrammées**

Nous pouvons utiliser les fonctions suivantes avec les enfants et changer le code principal qui se trouve dans la fonction LOOP :

- avanceRobot()
- reculeRobot()
- tourneDroite()
- tourneGauche()
- robotSurPlaceGauche()
- robotSurPlaceDroite()
- stopRobot()

Pour appeler une fonction dans la boucle principale (loop) on utilise la syntaxe suivante :

avanceRobot();

On rajoute un point-virgule en fin de ligne.

### **5 Écriture du code pour les fonctions du robot**

```

void avanceRobot()
{
    Moteur A
    analogWrite(AIA, vitesse_MG);
    analogWrite(AIB, LOW);

    Moteur B
    analogWrite(BIA, vitesse_MD);
    analogWrite(BIB, LOW);
}

```

## SEANCE 5

## TRONC COMMUN

```
void reculeRobot()
{
Moteur A
analogWrite(AIA, LOW);
analogWrite(AIB, vitesse_MG);
Moteur B
analogWrite(BIA, LOW);
analogWrite(BIB, vitesse_MD);
}
void tourneDroite()
{
Moteur A
analogWrite(AIA, vitesse_MG);
analogWrite(AIB, LOW);
Moteur B
analogWrite(BIA, LOW);
analogWrite(BIB, LOW);
}
void tourneGauche()
{
Moteur A
analogWrite(AIA, LOW);
analogWrite(AIB, LOW);
Moteur B
analogWrite(BIA, vitesse_MD);
analogWrite(BIB, LOW);
}
void robotSurPlaceGauche()
{
Moteur A
analogWrite(AIA, LOW);
analogWrite(AIB, vitesse_MG);
Moteur B
analogWrite(BIA, vitesse_MD);
analogWrite(BIB, LOW);
}
void robotSurPlaceDroite()
{
Moteur A
analogWrite(AIA, vitesse_MG);
analogWrite(AIB, LOW);
Moteur B
analogWrite(BIA, LOW);
analogWrite(BIB, vitesse_MD);
}
```

```
void stopRobot()
{
digitalWrite(AIA, LOW);
digitalWrite(AIB, LOW);
digitalWrite(BIA, LOW);
digitalWrite(BIB, LOW);
}
```

Fonction qui retourne la valeur en cm du capteur ultrason

```
unsigned int mesureDistance()
{
déclaration de la variable locale où sera stockée la distance en cm
int cm = distanceCM.ping_cm();on lit la valeur du capteur ultrason
on définit la plage de détection du capteur
if (cm > distanceObstacle || cm <= minimumDistance)
{
PasObstacle = false;on renvoie false car il n'y a pas d'obstacle
}
else sinon
{
PasObstacle = true;on renvoie truesi un obstacle est détecté
}
return cm; on retourne la distance du capteur en cm
}
```

### Récapitulatif des instructions :

- **Déclaration d'une variable** : on vient avec cette ligne stocker la valeur à droite du signe égal (=) dans la variable à gauche du signe égal.

```
int maximumDistance = 50;
```

Dans notre cas, cela signifie que la variable appelée maximumDistance viendra prendre la valeur 50. Le mot clé int en début de phrase signifie que la variable sera un nombre entier.

- **Les blocs d'instructions** : setup regroupe toutes les instructions qui seront exécutées au démarrage du programme. La fonction setup n'est exécutée qu'une seule fois, après chaque mise sous tension ou reset (réinitialisation) de la carte Arduino. loop (boucle en anglais) contient les instructions que l'on souhaite voir exécutées en core et encore tant que l'Arduino est branché.

```
void setup() {}
void loop() {}
```

- **Les fonctions** : sont des instructions qui permettent d'exécuter une ou plusieurs actions. Les fonctions sont définies avec :

Un nom : ce qu'on devra taper pour appeler la fonction.

Une ou des entrées : ce sont des variables passées à la fonction appelées paramètres ou arguments. Ces arguments sont placés entre parenthèses.

Une sortie : le résultat de la fonction qui peut être stocké dans une variable.

Prenons l'exemple de la fonction suivante :

`analogWrite(A1B, LOW);`

Dans ce cas, le nom de la fonction est `analogWrite`. Nous passons deux paramètres à la fonction : A1B et LOW. La fonction `analogWrite` n'a pas de sortie.

Avec cette fonction, nous éteignons la broche située sur la broche passée avec le premier paramètre (qui peut être un nombre ou une variable).

Lorsque le second argument est placé à LOW, on vient d'arrêter le moteur.

Tandis qu'on mettra en marche le moteur si le second argument utilise un nombre entier supérieur à 0. Ce nombre est compris entre 0 et 255. Faisons l'analogie avec l'électricité :

0 correspond à 0 volt.

255 correspond à 5 volts.

Plus la tension est élevée plus le moteur tournera vite.

- **Autres fonctions**

`pinMode`configure la broche spécifiée dans le premier paramètre pour qu'elle se comporte soit en entrée (INPUT), soit en sortie (OUTPUT) passée avec le second paramètre :

`pinMode(A1A, OUTPUT);`

`delay`correspond au temps d'exécution d'une fonction. La durée est mesurée en millisecondes:

`avanceRobot()`

`delay(200);`Le robot avancera pendant 200ms

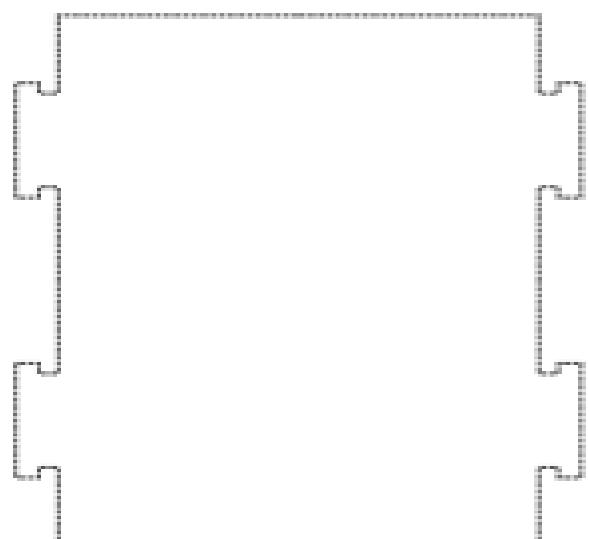
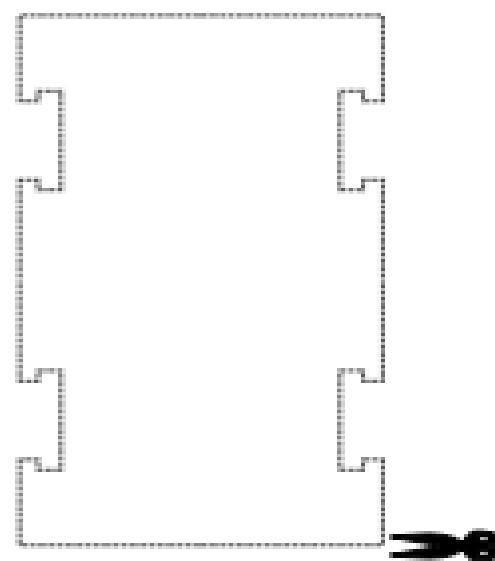
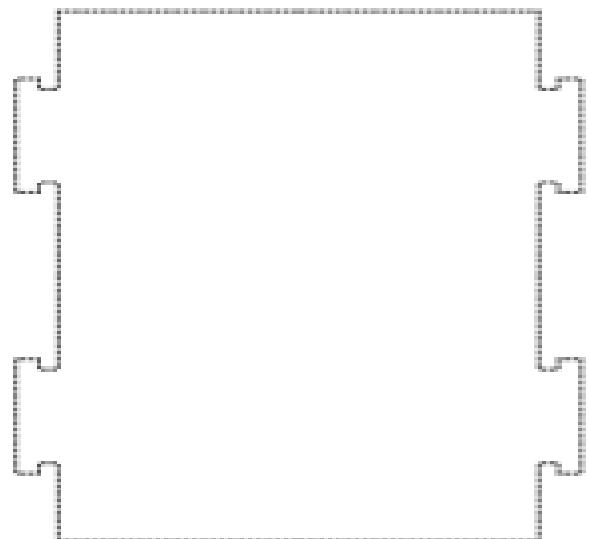
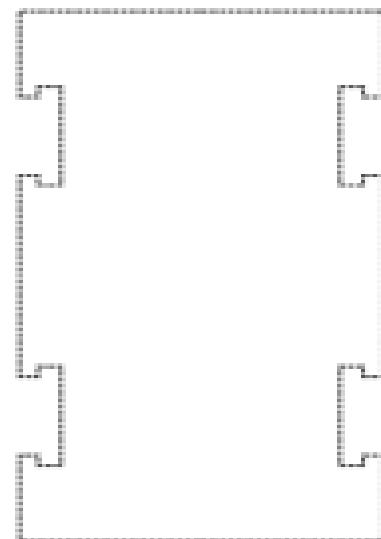
## Fiches à remplir :

<b>Fiche individuelle</b>	
<p><b>Mon projet prend comme base un robot (cocher les bonnes cases)</b></p>	Humanoïde <input type="checkbox"/>
	mékhano centré <input type="checkbox"/>
	zoo centré <input type="checkbox"/>
	autres, à préciser <input type="checkbox"/>
<p><b>Quel est le thème ou scénario?</b></p>	
<p><b>Le robot sera capable de (cocher les bonnes cases)</b></p>	Avancer <input type="checkbox"/>
	reculer <input type="checkbox"/>
	tourner à droite <input type="checkbox"/>
	tourner à gauche <input type="checkbox"/>
	détecter un obstacle <input type="checkbox"/>
	tourner sur place à droite <input type="checkbox"/>
	tourner sur place à gauche <input type="checkbox"/>

<b>Notre Projet robotique</b>	
<b>Nom de notre projet</b>	
<b>Nom de notre studio</b>	
<b>Nom des personnes composant notre studio</b>	
<b>Description de notre projet</b>	
<b>Notre projet prend comme base un robot (cocher les bonnes cases)</b>	humanoïde mékhano centré zoo centré autres, à préciser
<b>Il lui apporte les transformations suivantes (personnalisation)</b>	
<b>Quel est le thème ou scénario ?</b>	
<b>Le robot sera capable de (cocher les bonnes cases)</b>	avancer reculer tourner à droite tourner à gauche détecter un obstacle tourner sur place à droite tourner sur place à gauche
<b>Les grandes étapes (lister les différentes choses à faire pour la réalisation du projet)</b>	

### Fiche pour découper :

Exemple d'un élément structurel, vous pouvez faire d'autres selon votre choix :



- ✿ Pour la séance n°8, utilisez les fichiers de la séance n°5 : (Programmation de ROSA)

**Fiches à remplir :**

<b>Nom du robot :</b>	
<b>Présentation du robot et de l'environnement:</b>	
<b>Quels sont les commandes informatiques utilisées:</b>	
<b>Quel est le but de notre robot:</b>	
<b>Comment avons-nous créé ce robot (forme déjà existante, robot créé complètement, quelle construction du décor etc.)</b>	

## Documentation :

### Créer

#### Tutoriels

- Votre première présentation  
(<http://sozi.baierouge.fr/pages/tutorial-first-fr.html>)
- Utiliser les calques  
(<http://sozi.baierouge.fr/pages/tutorial-layers-fr.html>)
- Les effets de transition  
(<http://sozi.baierouge.fr/pages/tutorial-transitions-fr.html>)
- Créer un lien vers une vue ou une URL  
(<http://sozi.baierouge.fr/pages/tutorial-links-fr.html>)
- Insérer une présentation Sozi dans une page HTML  
(<http://sozi.baierouge.fr/pages/tutorial-embedding-fr.html>)
- Montrer et cacher des objets  
(<http://sozi.baierouge.fr/pages/tutorial-showing-hiding-fr.html>)
- Insérer de l'audio ou une vidéo  
(<http://sozi.baierouge.fr/pages/tutorial-media-fr.html>)
- Convertir les présentations Sozi en PDF ou en vidéo  
(<http://sozi.baierouge.fr/pages/tutorial-converting-fr.html>)
- Améliorer les performances  
(<http://sozi.baierouge.fr/pages/tutorial-performance-fr.html>)

#### Obtenir de l'aide et signaler un problème

- Foire Aux Questions et résolution des problèmes  
(<http://sozi.baierouge.fr/pages/faq-fr.html>)
- Rejoindre le groupe de discussion des utilisateurs de Sozi  
(<http://groups.google.com/group/sozi-users>)
- Signaler un problème et proposer de nouvelles fonctionnalités  
(<http://github.com/senshu/Sozi/issues>)

#### Partager vos présentations

Il n'y a actuellement aucune plate-forme de partage des présentations Sozi.  
Vous pouvez trouver des exemples de présentations et ajouter des liens vers vos propres présentations sur le site Sozi Community Wiki (<http://sozi.wikidot.com/>).