

# Systèmes robotisés communicants

J. Lanteri & J.M. Ribero



**1 – Introduction**

**2 – Projet**

**3- Servomoteur et capteur**

**4 - Environnement Arduino**

# Introduction



*But de ce module: faire fonctionner ce robot sur un parcours en évitant les obstacles*

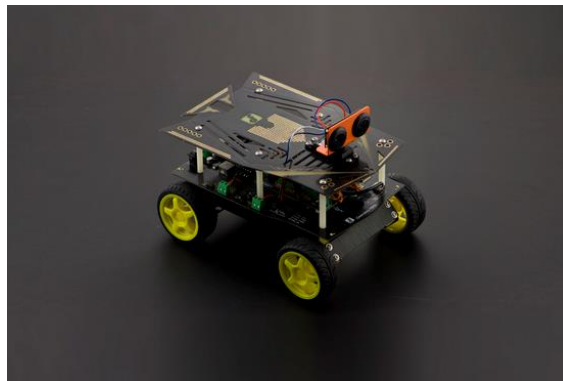
## PROJET

*L'objectif de ce projet est de programmer un robot Cherokee en langage Arduino.*

*Déroulement du projet :*

- Assemblage du robot,*
- Mise en place de la partie électronique (servomoteurs, capteurs, batterie...)*
- Utilisation d'une carte Arduino Uno pour la programmation*

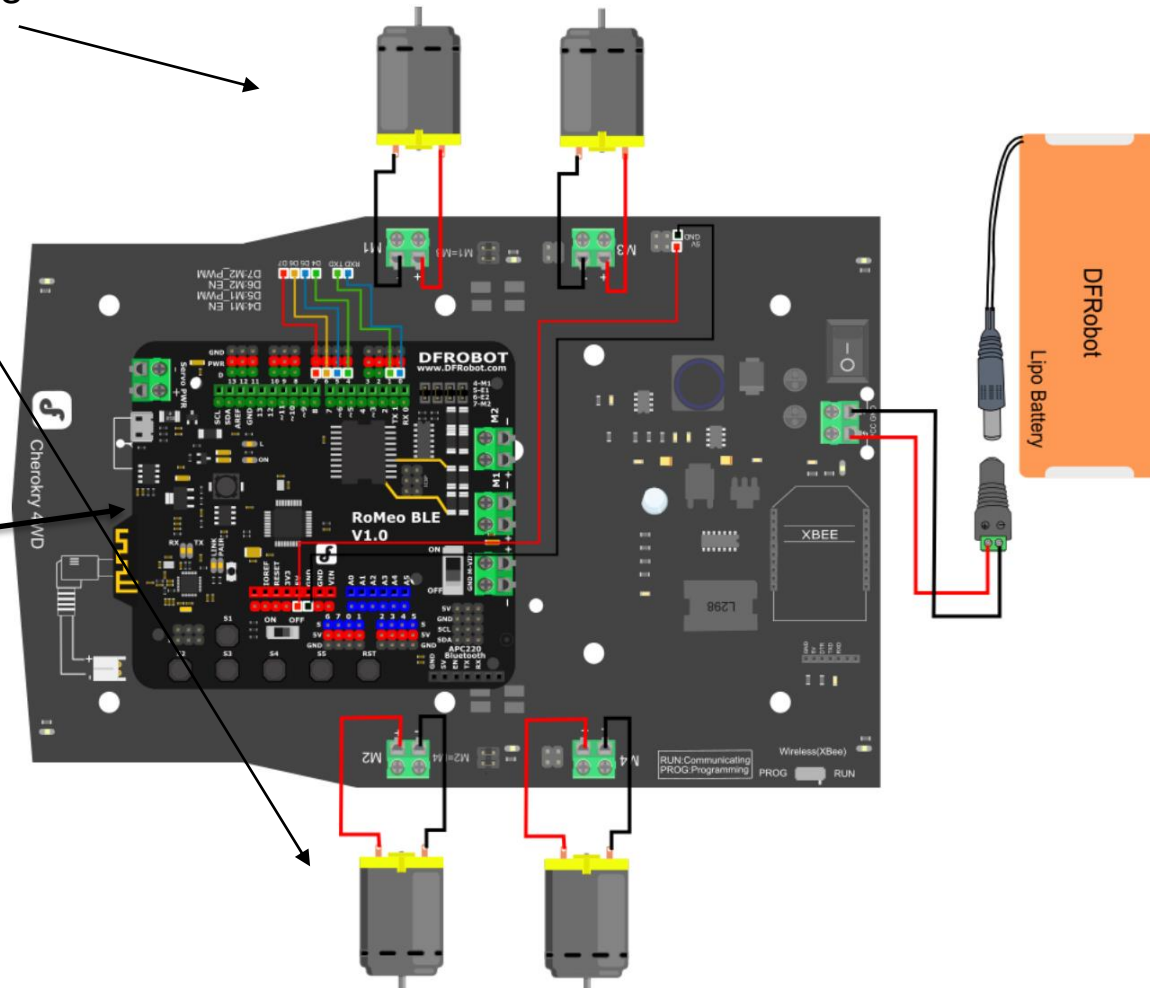
*Grace à son capteur ultrason, Il sera capable d'évoluer dans un environnement physique et prendre des décisions c'est à dire aller d'un point A à un point B en évitant les obstacles.*

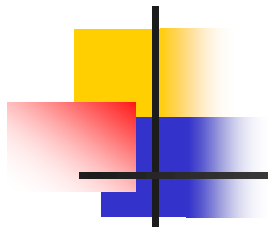


# Chassis Robot Cherokey

Moteurs pour les roues

Carte Arduino BLE



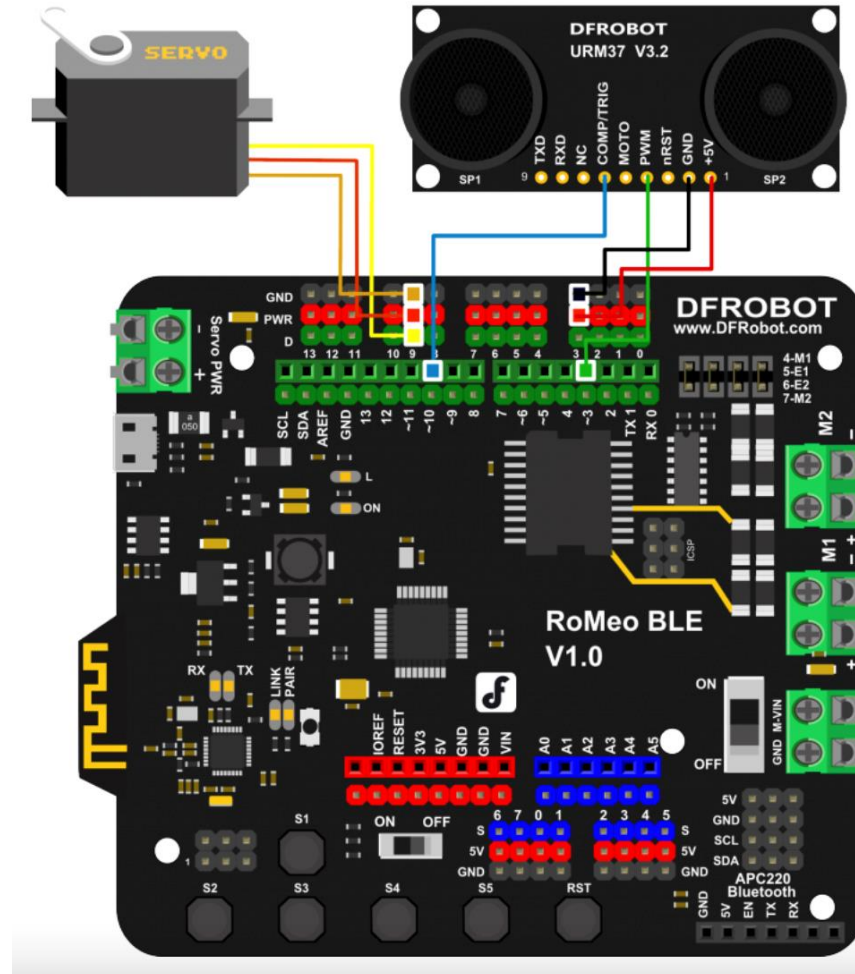


# ROBOT CHEROKEY

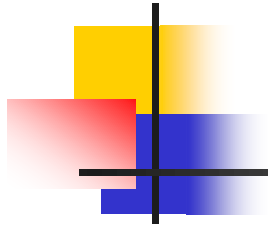
*Servomoteur*



*Capteur ultrason*



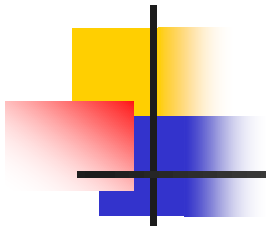
*Carte Arduino BLE*



# *SERVOMOTEUR et CAPTEUR*

---

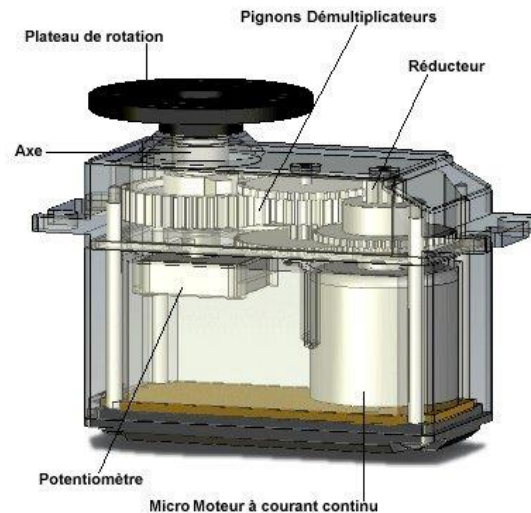
- Servomoteur
- PWM
- Capteur



# SERVOMOTEUR

*Un servomoteur est un moteur à courant continu avec quelques spécificités en plus.*

*Un servomoteur est capable d'atteindre des positions pré-déterminées, puis de les maintenir.*

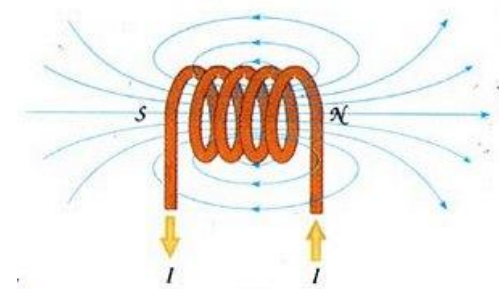
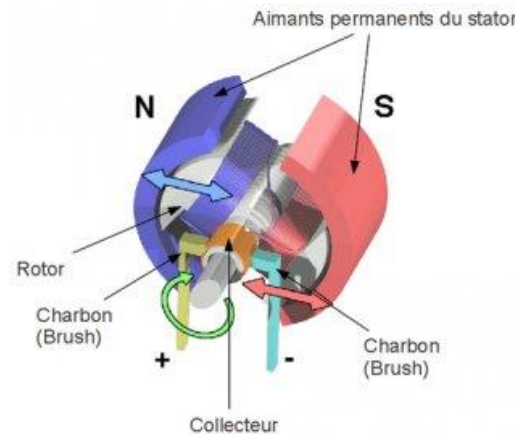


Eric G.



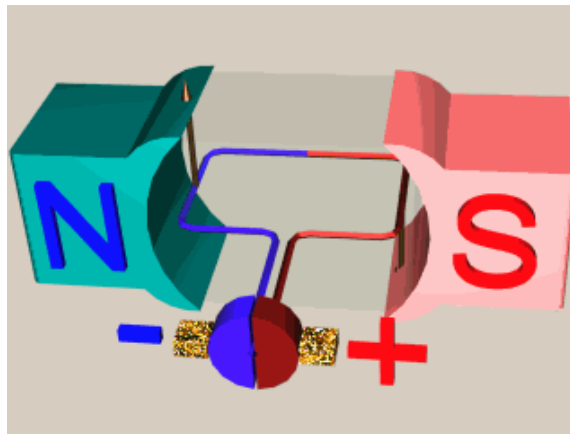
# DC Motor

- Un moteur à courant continu est une classe de machines électriques rotatives qui convertit l'énergie électrique à courant continu en énergie mécanique.



# DC Motor

- *Le moteur électrique à courant continu génère un couple directement à partir du courant continu fourni au moteur en utilisant une commutation interne, des aimants fixes (permanents ou électroaimants) et des électroaimants rotatifs.*



# *SERVOMOTEUR*

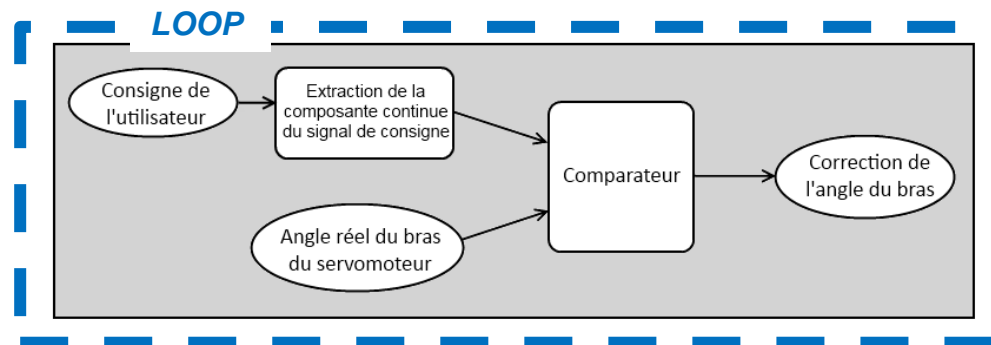
- Caractéristiques principales d'un servomoteur :
  - Vitesse de rotation, pour arriver à la position désirée.
  - Couple c'est-à-dire la force du servomoteur pour garder la position souhaitée
  - Angle maximal de rotation
- Les servomoteurs sont souvent utilisés dans le modélisme (orienter des roues, gouvernail,...)
- Les servomoteurs à rotation continue peuvent être utilisés comme des moteurs classiques.



# SERVOMOTEUR

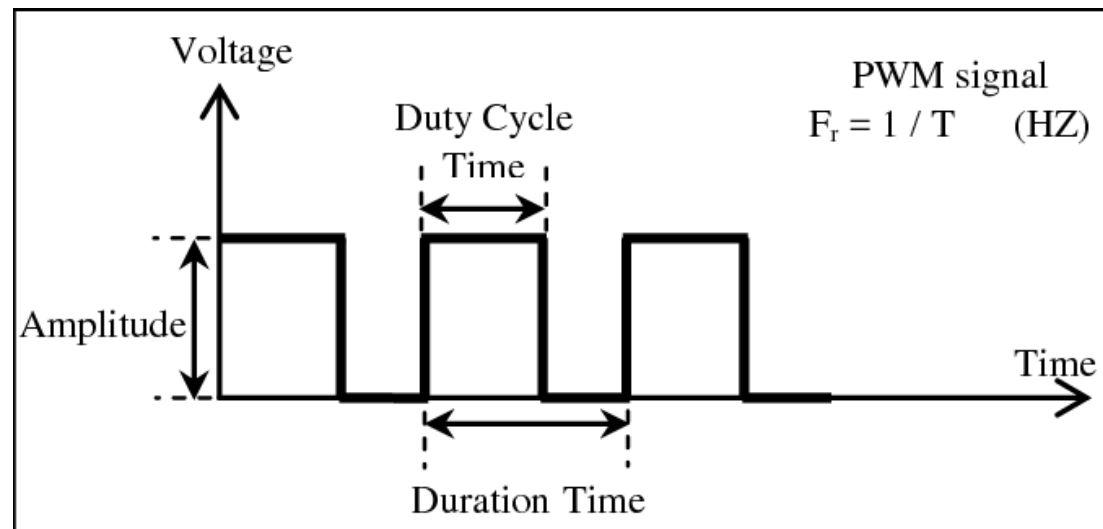
## ■ Fonctionnement

- La position d'un servomoteur est vérifiée en continu et corrigée en fonction de la mesure.
- Un servomoteur est commandé grâce à un microcontrôleur. Le microcontrôleur lui envoie une impulsion.
- La durée de cette impulsion (PWM) définit la position du servo.



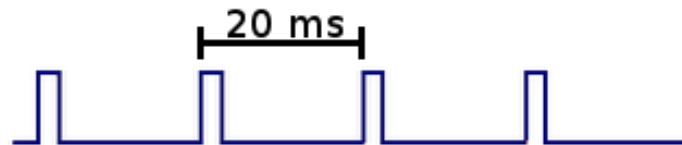
# Pulse Width modulation (PWM)

- *Un signal MLI (Modulation de Largeur d'Impulsions) ou PWM en anglais (Pulse Width Modulation) est un signal dont le rapport cyclique varie.*



# Pulse Width modulation (PWM)

Il a deux caractéristiques essentielles pour que le servomoteur fonctionne : la fréquence fixe et la durée de l'état ON



## FRÉQUENCE FIXE

Le signal généré doit avoir une fréquence de 50 Hz. En d'autres termes, le temps entre deux fronts montants est de 20 ms.

*Un microcontrôleur est uniquement numérique, la sortie peut passer de 0 à 5V .*

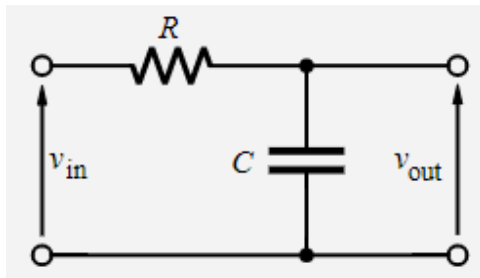
*Comment générer une sortie analogique ? Plus de deux valeurs de sortie ?*

# Pulse Width modulation (PWM)

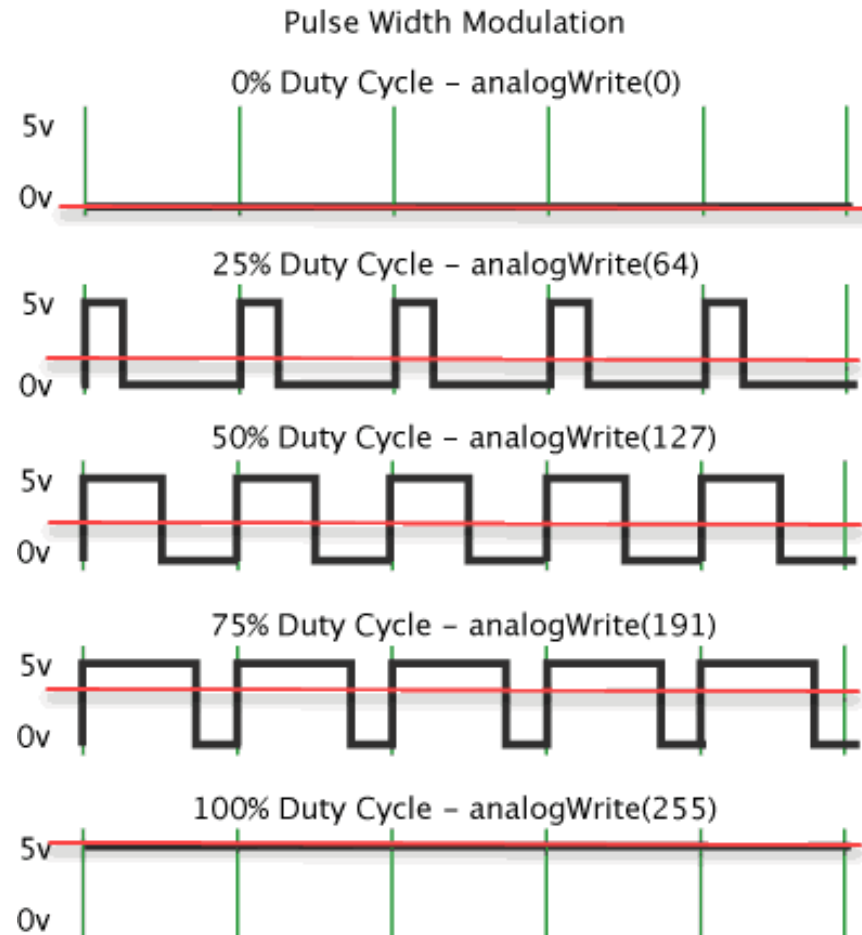
*Exemple :*

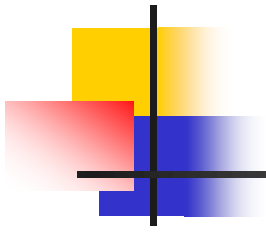
*What is the red line ?*

*How to generate it ?*



*Filtre Passe-Bas*

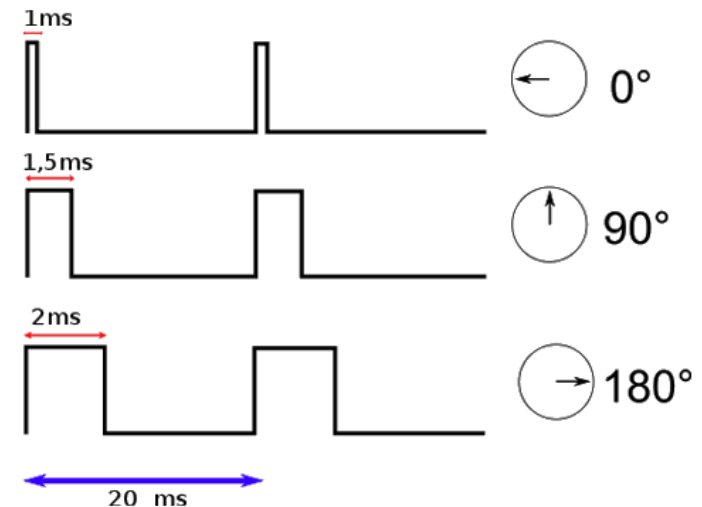




# Servo moteurs: Control signal(2/2)

## DUTY CYCLE

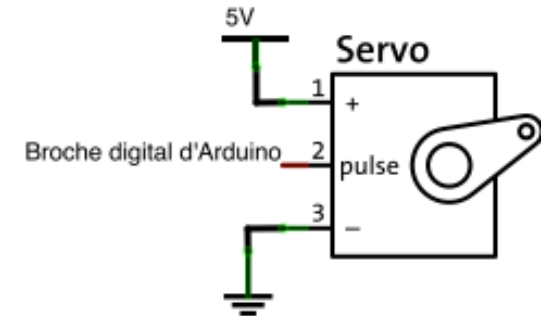
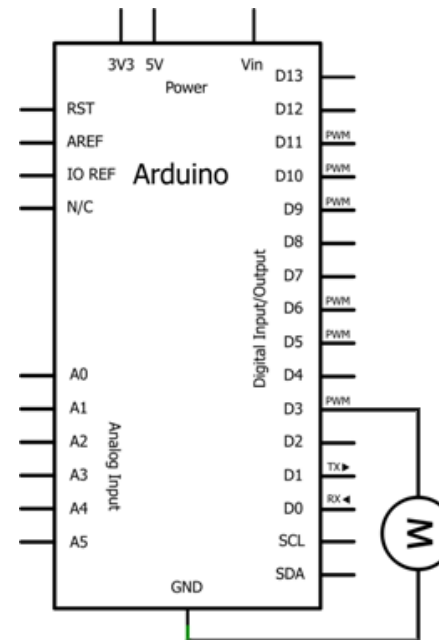
- Cette durée indique à l'actionneur l'angle précis qui est souhaité par l'utilisateur.
- Un signal avec une durée d'état ON de 1ms donnera un angle à  $0^\circ$ , le même signal avec une durée d'état ON de 2ms donnera un angle au maximum de ce que l'actionneur peut accepter.





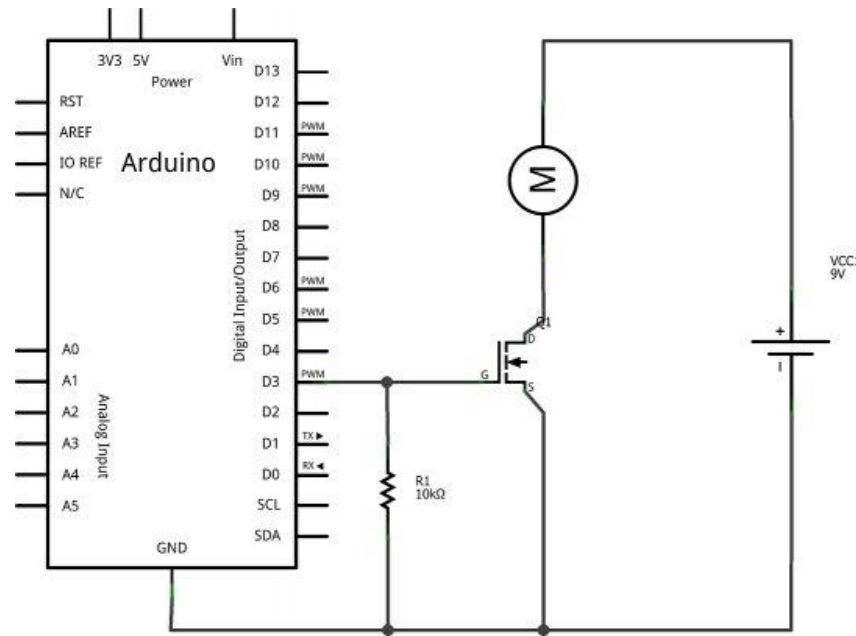
# Alimentation pour le moteur

- Est ce simple que de connecter le moteur aux sorties numériques de votre Arduino ?
- Courant requis trop élevé pour l'Arduino (max 40mA)
- Un circuit de contrôle est requis
- NE FAITES PAS CA!

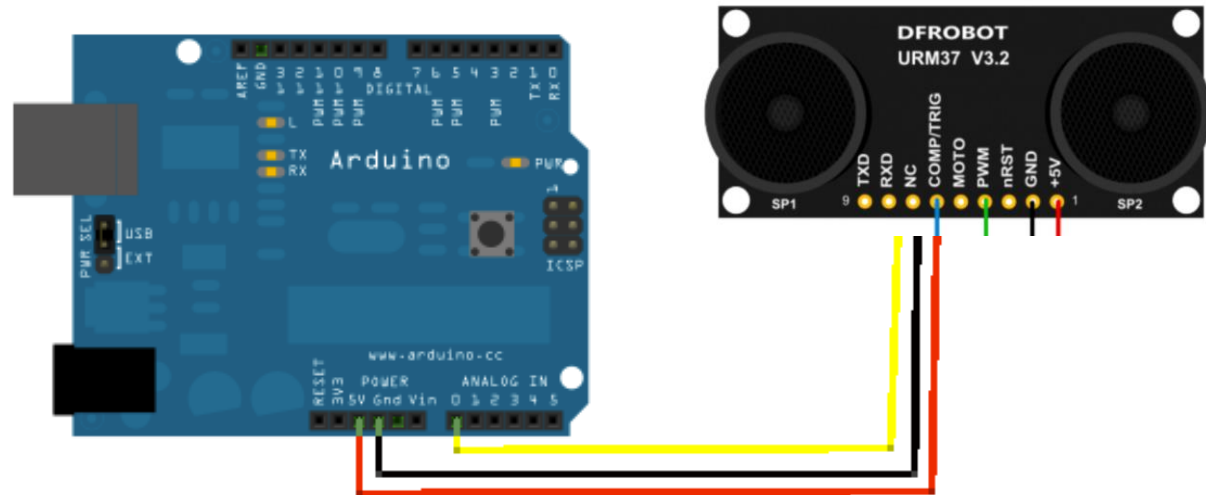


# Power supply for a motor

- Utilisation d'un MOSFET en commutation



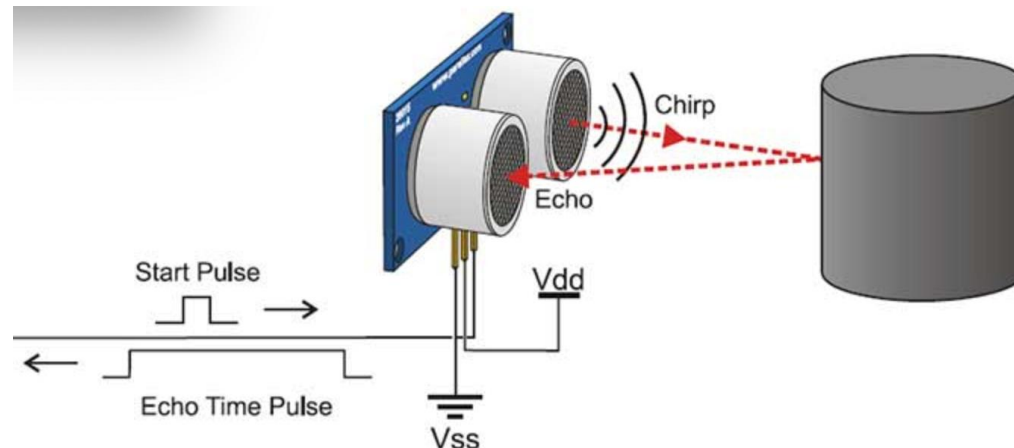
# Capteur de distance



# Capteur Ultrason

## Fonctionnement

Les capteurs à ultrasons émettent des impulsions ultrasonores qui se propagent dans un faisceau en forme de cône en utilisant un dispositif vibrant appelé transducteur, générant l'onde ultrasonore. La fréquence de vibration du transducteur détermine la plage d'un capteur à ultrasons. Les ondes sonores se transmettent sur des distances de plus en plus courtes au fur et à mesure que la fréquence augmente. Par conséquent, les capteurs à ultrasons à courte portée fonctionnent mieux à des fréquences élevées, tandis que les capteurs à ultrasons à longue portée fonctionnent mieux à des fréquences plus basses.



# Capteur URM37

*L'URM37 V5.0 est un puissant module de capteur à ultrasons.*

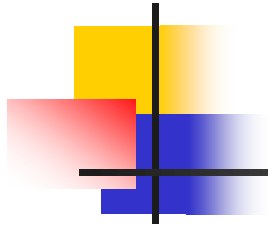
*Il possède une interface riche et offre diverses sorties : sortie analogique, commutateur, série (niveau TTL et RS232 en option), PWM et ainsi de suite.*

*Le module peut être utilisé pour mesurer l'angle de rotation du servo. Connecté à un servo externe, il se transforme en un scanner spatial à ultrasons.*

## Specification



- Operating Voltage: 3.3V ~ 5.5V
- Operating Current: 20mA
- Working temperature: -10°C ~ 70°C
- Detecting range: 2cm-800cm(ultimate range 1000cm)
- Resolution: 1cm
- Accuracy: 1%
- Measuring Period: <100ms (Max)



Quelques cartes de développement

# Carte Arduino UNO



LEDs  
(pgm blink)

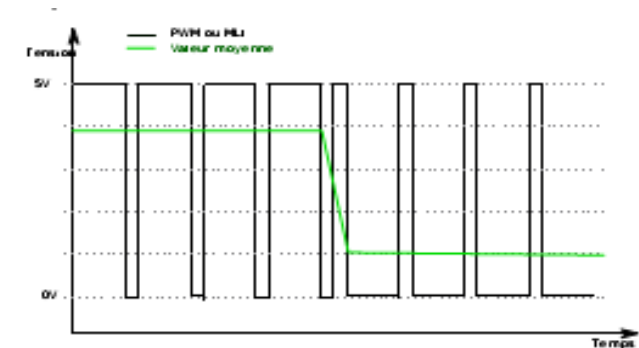
## Alimentation :

- Par le port USB de l'ordinateur (5 V)
- Batterie

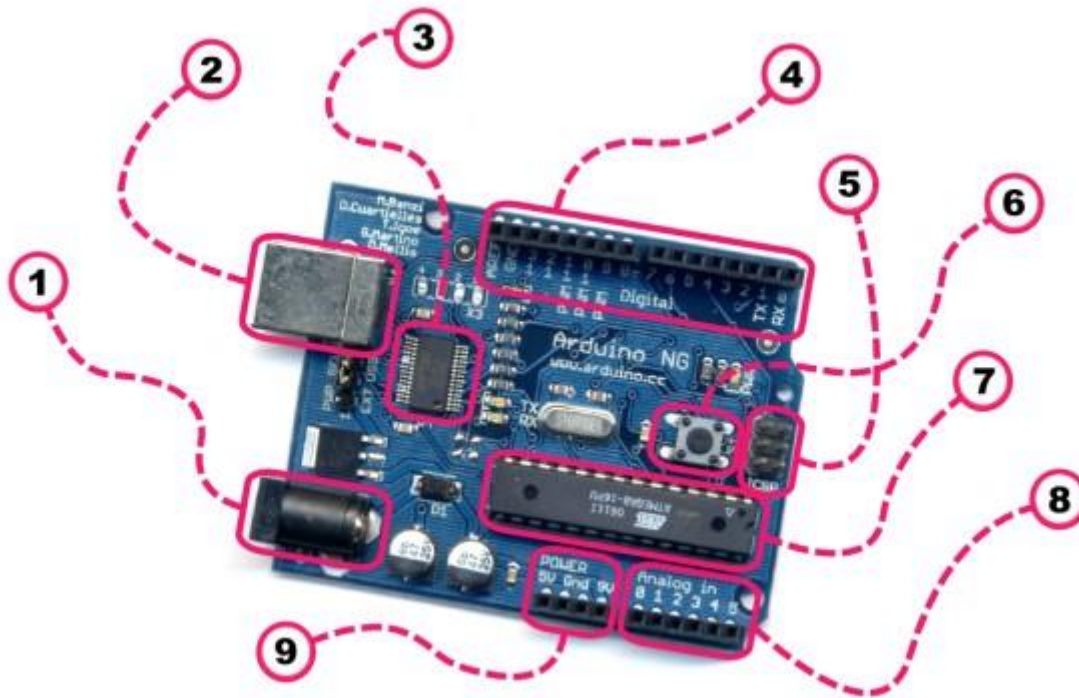
## PWM :

le signal de sortie est modulé sous forme d'un signal carré dont la largeur des créneaux varie pour faire varier la tension moyenne

- Microcontrôleur ATMEGA
- Ports :
  - 6 entrées Analogiques : Ai
  - 14 entrées/sorties Numérique (Digital) : Di
    - 3 à 6 peuvent produire un signal PWM (Pulse Width Modulation)
    - Courant de sortie : 40 mA
  - GND, Vcc



# Carte Arduino UNO



- (1) Alimentation extérieure,
- (2) programmation de la carte ou transmission des données au PC
- (3) puce spécialisée pour la gestion de l'USB (FT232RL),
- (4) 14 ports digitaux d'entrée-sortie,
- (5) port de programmation ICSP, afin de programmer le microcontrôleur en place.
- (6) bouton de reset
- (7) microcontrôleur, [Atmel ATmega8+](#) (8 KO à 16Ko de mémoire flash, 1 KO de SRAM, 16 MHz)..
- (8) 6 ports analogiques d'entrée pour relier des capteurs,
- (9) masse, Vcc= 5v et 3,3 V disponible pour alimenter les capteurs et circuits externes.



# Environnement de programmation



Barre de Menu  
Barre de Boutons  
Onglets des fichiers ouverts

Fenêtre d'édition  
des programmes

Zone de messages des actions en cours

Console d'affichage  
des messages de compilation

## IDE arduino - Configuration préalable:

### Outil : type de carte

- Carte déjà installée => sélectionner la carte
- Carte non installée : gestionnaire de carte + installation de la carte
- La carte reconnue apparait comme un port COM


### Choix du programmeur dans outil

### Bibliothèque :

Croquis : inclure une bibliothèque si la la bibliothèque n'est pas présente => gérer les bibliothèques puis importer la bibliothèque.

# Enevironnement de Programmation - Sketch Arduino (.ino)

1. Déclaration des variables (optionnelle)
2. initialisation et configuration des entrées/sorties : la fonction **setup ()**
3. partie principale qui s'exécute en boucle : fonction **loop ()**



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

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);            // wait for a second
}
```

# Le langage – basé sur le C/C++

- Commentaires :
  - dans une ligne de code, tout ce qui se trouve après « **//** » **sera un commentaire.**
  - encadrer des commentaires sur plusieurs lignes entre « **/\*** » **et** « **\*/** ».
- **toute ligne de code se termine par un point-virgule « ; »**
- le contenu d'une **fonction est délimité par des accolades « { » et « } »**
- les **paramètres d'une fonction sont contenus pas des parenthèses « ( » et « ) ».**

**Variables** (espace réservé dans la mémoire de la carte) :

- boolean : valeurs vrai/faux (True/false)
- Octet : bytes
- nombres entiers (int)
- nombres à virgule flottante (float, exemple : 3.14)
- Char, texte (String)
- Tableau

# Le langage – basé sur le C/C++

## Fonctions :

- Une fonction: bloc d'instructions qui peut être appelé dans un programme.
- Qq Fonctions prédéfinies : `analogRead()`, `digitalWrite()`, `delay()`.
- Bibliothèque => ensemble de fonctions pour manipuler : capteurs, actionneurs, communication, ...

- Exemple :

```
void clignote()  
{  
  digitalWrite (brocheLED, HIGH) ;  
  delay (1000) ;  
  digitalWrite (brocheLED, LOW) ;  
  delay (1000) ;  
}
```

Appel de la fonction : `clignote()`

### ***paramètres dans une fonction :***

```
void clignote(int broche, int vitesse)  
{  
  digitalWrite (broche, HIGH) ;  
  delay (1000/vitesse) ;  
  digitalWrite (broche, LOW) ;  
  delay (1000/vitesse) ;  
}
```

*Appel de la fonction : clignote(5, 1000)*

# Les structures de contrôle

## ***if...else :***

```
//si la valeur du capteur dépasse le seuil  
if(valeurCapteur>seuil) {  
  clignote(); //appel de la fonction clignote  
} else {....}
```

## ***switch/case : fait un choix entre plusieurs codes parmi une liste de possibilités***

```
// faire un choix parmi plusieurs messages reçus  
switch (message) {  
  case 0: //si le message = 0 , activée la sortie  
    digitalWrite(3,HIGH);  
    digitalWrite(4,LOW);  
    digitalWrite(5,LOW);  
    break;
```

```
  case 1: //si le message = "1"  
    //activée la sortie 4  
    digitalWrite(3,HIGH);  
    digitalWrite(4,LOW);  
    digitalWrite(5,LOW);  
    break; ....
```

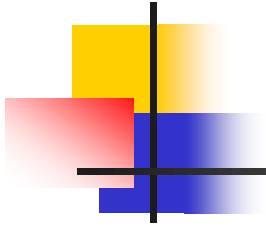
# Les structures de contrôle

***Boucle for : exécute un code un certain nombre de fois.***

```
//pour i de 0 à 255, par pas de 1  
for (int i=0; i <= 255; i++){  
  analogWrite(PWMPin, i);  
  delay(10);  
}
```

***while : exécute un code tant que les conditions sont vérifiées.***

```
//tant que la valeur du capteur est supérieure à 250  
while(valeurCapteur>250){ //allumer la sortie 5  
  digitalWrite(5, HIGH);  
  //envoi le message 1 au port serie  
  Serial.println(1);}  
Serial.println(0);  
digitalWrite(5, LOW);
```



# analogWrite()- Servo Moteur

- analogWrite() est utilisé pour définir le rapport cyclique d'une impulsion PWM
  - Après un appel à analogWrite(), la broche générera une onde carrée constante du rapport cyclique spécifié jusqu'au prochain appel à analogWrite(), digitalWrite() ou digitalWrite() sur la même broche
- Syntaxe
  - *pin*: the pin to write to *analogWrite(pin, value)*
  - valeur : le rapport cyclique entre 0 (toujours OFF) et 255 (toujours ON)

# Environnement de programmation

Vérifier Téléverser Moniteur Série



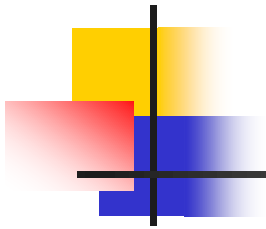
Statut

Console des messages

Librairies  
Choix carte  
Sélection PORT  
Choix programmeur

Sélectionner :  
- pas de fin de ligne  
- 9600 baud





# Projet

*Utilisation d'une carte arduino BLE :*

*Bluetooth low energy, identique à la carte Arduino*

*Uno + commande par bluetooth*



**Assemblage du robot en suivant le tutorial :**

**[https://wiki.dfrobot.com/Basic Kit for Cherokee 4WD SKU ROB0117](https://wiki.dfrobot.com/Basic_Kit_for_Cherrykey_4WD_SKU_ROB0117)**

**Cablage des connections entre :**

**- les moteurs, le servomoteur, le capteur et la carte arduino**

**Test des programmes de fonctionnement :**

**- des moteurs,**

**- du servomoteur et du capteur**