

A ctuellement tous les servomoteurs utilisés en maquettisme fonctionnent sur le même principe. Bien que les fournisseurs ne respectent pas forcément ce qui semble devenir une norme de fait, globalement tous les moteurs s'approchent de ce qui devient un standard. Ces moteurs intégrant une électronique d'interfaçage et un capteur de position se raccordent avec seulement trois fils. Le fil noir ou marron va à la masse, le fil central rouge va au +Vcc et le fil blanc ou orange reçoit le signal de pilotage.

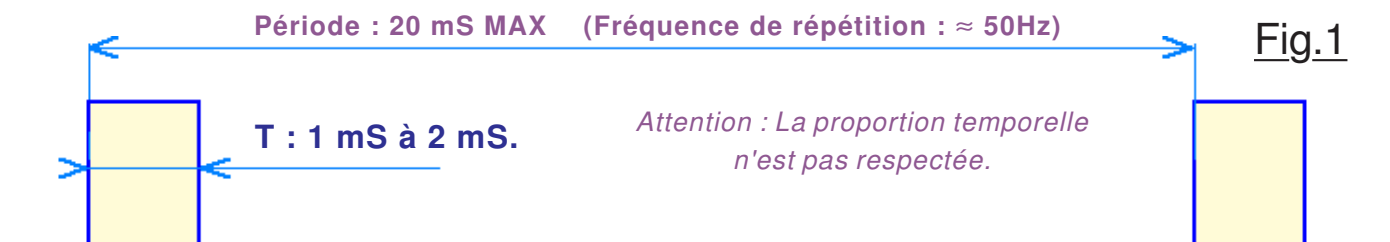


Fig.1

Comme montré sur la Fig.1 la fréquence de répétition standard choisie pour le pilotage est de 50 Hz, soit une période de 20 mS. C'est la durée **T** de l'impulsion à l'état "1" qui positionne le moteur en absolu par rapport à son amplitude possible de rotation. Mis à part les moteurs sans limitation de rotation angulaire, les plages les plus courantes avoisinent les 180°.

En standard :

- T = 1 mS : Le moteur se place en position minimale.
- T = 1,5 mS : Le moteur se place en position neutre. (*Position moyenne*)
- T = 2 mS : Le moteur se place en position maximale.

Caractéristiques techniques des deux moteurs utilisés :

Quand la consigne angulaire passe de 0° à 180° ou que la consigne en impulsion passe de 540µS à 2400µS, le moteur MC-1811 de la Fig.2 tourne dans le sens des aiguilles d'une montre. Par contre, le modèle RS2 MG/BB montré en Fig.3 tourne en sens inverse pour les mêmes variations de commande de pilotage.

ATTENTION : Un seul de ces moteurs branché sur le 5Vcc d'Arduino provoque de brusques baisses de tension lors des appels de courant. Ces perturbations sont parfaitement mesurables quand on branche un voltmètre quelconque sur le +5Vcc d'Arduino. Si la tension chute, c'est que le régulateur intégré à Arduino passe en limitation de courant. Outre que le microcontrôleur et les circuits intégrés périphériques ne fonctionnent pas dans des conditions fiables, le régulateur intégré à Arduino va chauffer, et ce d'autant plus qu'il n'est pas refroidi par radiateur. **Il est donc fortement recommandé d'alimenter les moteurs séparément** avec un bloc externe sans oublier naturellement de réunir les masses pour la référence des signaux de commande.

Fig.2



Fig.3



Moteur	Alimentation	Couple	Réaction	Paliers	Engrenages
MC-1811	4,8Vcc à 6Vcc	15 à 18cmN	0,19S (60°)	---	PTFE
RS2 MG/BB	4,8Vcc à 6Vcc	31 à 35cmN	0,1S (60°)	Roulements	Métalliques

PROGRAMMATION des servomoteurs.

C ompte tenu de la simplicité des signaux à générer, la première approche logique consiste à utiliser les instructions classiques d'Entrée/Sortie d'Arduino. Comme la période doit avoisiner les 20mS, l'utilisation de la PWM d'origine n'est pas possible. En effet, la fréquence de répétition de la PWM d'Arduino fait environ 1kHz, donc une période vingt fois trop faible. Mais rien n'interdit de générer directement de la PWM sur une sortie binaire en usant des fonctions de temporisation de base. Le petit programme **Commande_Servo_par_PWM.ino** génère des impulsions calibrées correspondant au standard. Bien que le fonctionnement soit correct, le moteur est affecté d'un bruit de "grenailage" prouvant qu'il corrige sans arrêt sa position. Ce phénomène n'est pas compréhensible en ce qui me concerne, car les signaux observés à l'oscilloscope sont correct et les informations envoyées sur la ligne série sont conformes. Les

transmissions série sont passées en remarque dès que les valeurs ont été vérifiées, car elles allongent la période de répétition du signal de commande. La librairie **Servo.h** fournie en standard dans l'environnement IDE corrige ce phénomène, donc inutile pour le moment de rechercher l'origine de ce problème.

Exemple de programme avec génération "binaire" du signal de PWM :

```
// Test de pilotage d'un Servomoteur par gestion binaire de la PWM.
// Un potentiomètre sur une entrées analogique génère le signal de commande.

const byte Entree_mesuree = 5; // Entrée analogique 5 utilisée pour commander.
const byte Sortie_PWM = 3; // Broche PWM 3 utilisée pour générer la commande.
float CNA; // Mesure analogique retournée par le CNA.
int Duree_Pulse; // Durée de l'impulsion de commande.

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

void loop() {
  CNA = analogRead(Entree_mesuree); // Valeur directe si RS2 MG/BB.
  CNA = 1024-CNA; // Inverser le sens si le moteur utilisé est le MC-1811.
  Duree_Pulse = ((1800 * CNA)/1023)+500;
  digitalWrite(Sortie_PWM, HIGH); delayMicroseconds(Duree_Pulse);
  digitalWrite(Sortie_PWM, LOW);
  delay(15); // Délai constant pour commencer le 20mS.
  delayMicroseconds(5000 - Duree_Pulse); // Ajustement pour une fréquence constante.
}
```

PROGRAMME :
Commande_Servo_par_PWM.ino

Le programme **Commande_Servo_PWM_librairie.ino** donné ci-dessous génère des commandes qui débordent volontairement du standard. Il permet de trouver les limites extrêmes du moteur utilisé.

Exemple de génération du signal de PWM avec la librairie **Servo.h** :

```
// Test de pilotage d'un Servomoteur par gestion librairie de la PWM.
// Un potentiomètre sur une entrées analogique génère le signal de commande.

#include <Servo.h> // Inclusion de la librairie Servo.
Servo Servomoteur; // déclare une variable de type Servo appelée Servomoteur.
const byte Entree_mesuree = 5; // Entrée analogique 5 utilisée pour commander.
float CNA; // Mesure analogique retournée par le CNA.
int Duree_Pulse; // Durée de l'impulsion de commande.

void setup() { Servomoteur.attach(3, 100, 3000); // Associe le servomoteur à la broche 3.
               Serial.begin(19200); }
               //      /      \
               impls_min, impls_max

void loop() {
  CNA = analogRead(Entree_mesuree); // Valeur directe si RS2 MG/BB.
  CNA = 1023-CNA; // Inverser le sens si le moteur utilisé est le MC-1811.
  Duree_Pulse = ((2900 * CNA)/1023)+100;
               (2900 = impus_max - impls_min)
  Servomoteur.writeMicroseconds(Duree_Pulse);
  // Positionne le moteur.
  Serial.print("Duree_Pulse = ");
  Serial.println(Duree_Pulse);
}
```

Limites mesurées sur les modèles utilisés

Servomoteur	T mini	T MAXI
MC-1811	660 à 700	2000
RS2 MG/BB	570	2000

Exemple de pilotage ANGULAIRE avec la librairie **Servo.h** :

C'est probablement la façon la plus "naturelle" d'envoyer des commandes à un servomoteur. La commande **NomMoteur.write(Angle)** positionne directement le moteur à la valeur Angle indiquée en degrés et comprise entre 0 et 180. La fonction se charge de générer l'impulsion qui placera le moteur à une position proportionnelle en fonction de l'amplitude de rotation possible sur le modèle utilisé. Le petit programme **Commande_Servo_par_Angles.ino** donne un exemple d'utilisation, c'est toujours un potentiomètre branché sur une entrée analogique qui permet d'envoyer les consignes.