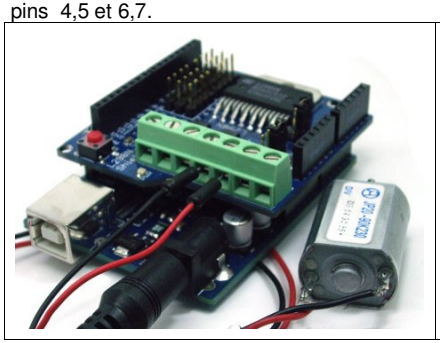
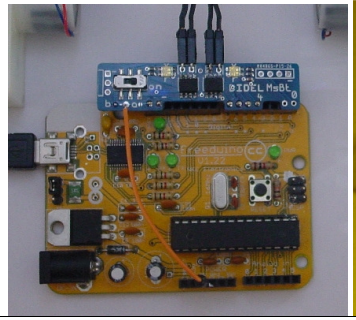
http://www.didel.com/ info@didel.com

www.didel.com/diduino/CommandeMoteurs.pdf

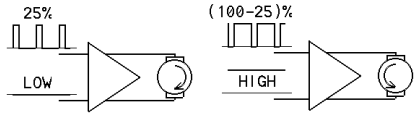
Commande de moteurs avec Arduino

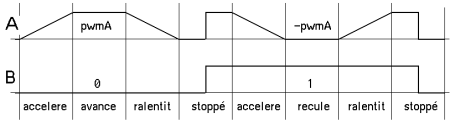
La plupart des shields Arduino pour commander des moteurs, ainsi que le minishield MsMot de Didel utilisent les pins 4 à 7 pour commander deux moteurs, avec la possibilité de commande en PWM (digitalWrite) seulement sur les pins 5 and 6. Las amplis Moteurs (drivers) sont câblés sur les pins 4,5 et 6,7.

D'autres shields utilisent les pins 5,6, 9,10. Sur le XBot, le câblage est libre, mais le câblage sur les pins 4,5,6,7 est facilité

Pour commander un moteur bidirectionnel, on applique le signal PWM d'un côté ou de l'autre de l'ampli. S'il n'y a que l'un des côtés qui peut être piloté en PWM, comme c'est le cas ici, il faut pour changer de sens activer la sortie fixe et complémenter la valeur du PWM sur l'autre sortie.





On voit bien sur la figure comment accélérer et ralentir le moteur, dans un sens et dans l'autre.

Si on a un robot avec deux moteurs, pour faciliter l'écriture du programme, on définit des fonctions Avance, Recule, TourneDroite (tourne sur soi) etc, qui ont un seul paramètre. La valeur PWM 8 bits est envoyée sur les pins 5 et 6, avec le bon l'état sur les pins 4 et 7, selon le sens de rotation.

```
void Avance (byte vv)  // 0-255
{
  analogWrite(AvG, vv);
  digitalWrite(RecG, LOW);
  analogWrite(AvD, vv);
  digitalWrite(RecD, LOW);
}
```

```
void Recule (byte vv)
{
  analogWrite(AvG, ~vv);
  digitalWrite(RecG, HIGH);
  analogWrite(AvD,  ~vv);
  digitalWrite(RecD, HIGH);
}
```

```
void TourneD (byte vv)
{
  analogWrite(AvG, vv);
  digitalWrite(RecG, LOW);
  analogWrite(AvD,  ~vv);
  digitalWrite(RecD, HIGH);
}
```

C'est un peu limitatif dans une application d'avoir des appels différents selon le comportement. Pour tourner avec un rayon de courbure quelconque, il faut deux paramètres.

Ecrivons une fonction unique Bouge à deux paramètres qui accèpte des vitesses signées de –255 à +255.
La fonction sépare avec des if les cas pwm positif (avance) et négatif (recule).Si c'est négatif, il faut d'une part inverser pour avoir une valeur positive, et prendre le complément à 255 parce que l'autre pôle du moteur est à un. Donc calculer 255 – (–gg) = 255+gg.
Le type déclaré doit être int, 16 bits signé.
A noter que dans les exemples précédents on aurait aussi pu, comme cela se fait d'habitude sans réfléchir, utiliser le type int.

```
// pwm entre –255 et +255 (non saturé)
void Bouge (int gg, int dd)  {
  if (gg==0) {
    analogWrite(bAvG, 0);
    digitalWrite(bRecG, LOW);
  } else if (gg > 0) {
    analogWrite(bAvG, gg);
    digitalWrite(bRecG, LOW);
  }  else {
    analogWrite(bAvG,255-(-gg));
    digitalWrite(bRecG, HIGH);
  }
  if (dd==0) {
    analogWrite(bAvD, 0);
    digitalWrite(bRecD, LOW);
  } else if (dd > 0) {
    analogWrite(bAvD, dd);
    digitalWrite(bRecD, LOW);
  } else {
    analogWrite(bAvD, 255-(-dd));
    digitalWrite(bRecD, HIGH);
  }
}
```

La vitesse nulle demande un décodages spécial: on était peut-être en vitesse négative avant.
C'est mentionné entre parenthèses que les paramètres utilisés par la fonction ne sont pas saturés. Si on dépasse 255, on se retrouve avec des valeurs qui ne correspondent probablement pas à ce que l'on veut.
Pour saturer, c'est-à-dire empêcher un dépassement, il faudrait rajouter 4 lignes type `if (dd>255) dd= 255;`
Vous voulez avancer en tournant un peu à droite? Il faut que le moteur gauche aille un peu plus vite

```
Bouge (210,200) ;
```

Vous voulez écrire un programme qui avance et recule comme dans la figure plus haut?
Il faut après avoir les déclarations, le set-up et la fonction, écrire
- une boucle for pour augmenter le pwm de 0 à 255
- une boucle for pour diminuer le pwm de 255 à -255
- une boucle for pour augmenter le pwm de –255 à 0

```
int v;
void loop()
{
  for (v=0;   v<255;  v++) { Bouge (v,v); delay (8); }
  for (v=255; v>-255; v--) { Bouge (v,v); delay (8); }
  for (v=-255; v<=0;  v++) { Bouge (v,v); delay (8); }
  delay (2000) ; // attente avant de recommencer
}
```

Note pour les spécialistes: Les driver moteur L293 et le Si9986 n'ont pas le même décodage. Avec le 9986, l'état HIGH sur les deux entrées est roue libre. Pour le même PWM, la vitesse est un peu différente en avant et en arrière.

jdn130328/161120