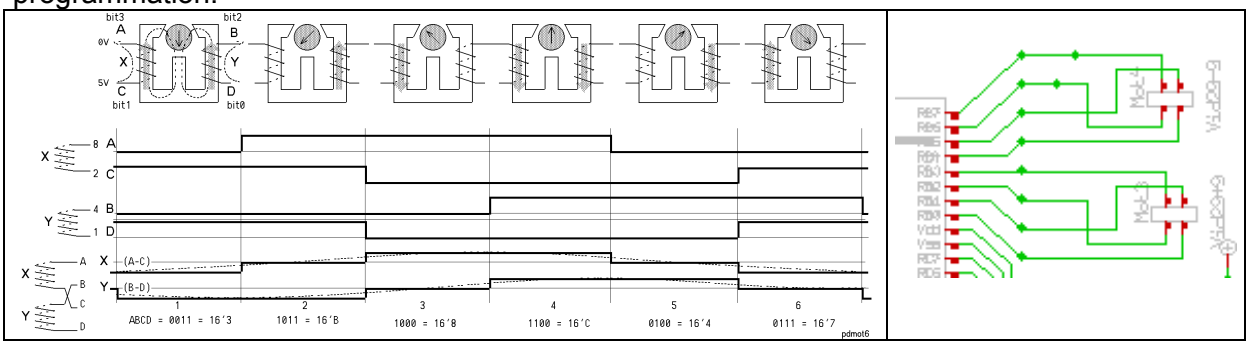
Programmation du Wellbot Wbot2 en C
Moteurs pas-à-pas

Avancer-reculer d'un pas

Le moteur Vid29 (google « Vid29 »), copie du moteur Switec, a deux bobines, mais trois phases, avec une séquence des phases donnée ci-dessous. Pour la coder il faut savoir comment les 4 fils sont câblés sur le processeur. Les 2 fils de la même bobine ne sont pas câblés sur des sorties consécutives pour éviter que RB6 RB7, utilisés par le programmateur, ne court-circuitent (résistance de 290 Ohms) ces deux lignes et empêchent la programmation.



Les deux moteurs sont câblés de façon similaire. Ils vont donc tourner en sens inverse avec la même table. Il faut de plus tenir compte du fait que les deux moteurs ne sont pas câblés sur les mêmes sorties. En C deux tables ont été définies.

Table des pas en C

```
int AvDroite[6]={0x70,0x60,0xe0,0x80,0x90,0x10};
int AvGauche[6]={0x01,0x09,0x08,0x0e,0x06,0x07};
```

Pour faire tourner un moteur, il suffit de parcourir sa table en incrémentant le pointeur circulairement.

```
i++; if(i==6) i=0;
PORTB = AvDroite[i];
```

Pour tourner dans l'autre sens, on décrémente circulairement le pointeur.

```
i--; if(i==-1) i=5;
```

Le programme `Wb2Pas1.pde` pour faire tourner les deux moteurs dans le même sens se trouve sous www.didel.com/Wbot2/Soft.zip

Puisque les moteurs vont à la même vitesse, un seul pointeur et une seule table d'avance serait possible.

```
//\title; Wb2Pas1.pde
int i=0;
int Avance[6]={0x71,0x69,0xE8,0x8E,0x96,0x17};
void setup()
{
  TRISB = 0b00000000 ;
  PORTB = 0 ;
}
void loop()
{
   i++; if(i==6) i=0; // compte 0 1 2 3 4 5 0 1 2...
   PORTB=Avance[i];
   delayMicroseconds(1000); // max vit 1000 à 3.5V
}
```

Exercices :

Modifier la période (le moteur peur aller plus vite si on le lance).
Varier la vitesse (comme dans le programme sirène).
Compter les pas et avancer d'un certain nombre de pas.
Avancer en zig-zag (un moteur après l'autre)
Commande individuelle des moteurs.
Il n'y a pas de corrigé pour les exercices. Le zip contient nos programmes de test, non retravaillés. En cas de gros problème, envoyez votre programme à info@didel.com

Distance et angle
Pour chaque moteur, un compteur compte et décompte les pas. Le type `Int` est 16 bits signés, donc on a +/- 32'000 pas, et chaque pas est de 0.4 mm (à vérifier).
La distance est la somme des pas, l'angle la différence. Une division entière par les bons paramètre donne la distance en mm et l'angle en degrés.
Exercice
Déterminer ces deux facteurs.

Sens programmable pour chaque moteur
Une variable définit le sens pour chaque moteur. Appelons-la Vitesse, mais pour le moment seul le signe compte.

```
if (SpeedDroite >= 0)
  { i++; if(i==6) i=0; }
else
  { i--; if(i==-1) i=5; }
```

Les deux moteurs ont la même vitesse, définire par delayMicroseconds() ;

Exercice
Faire des allers et retours et observer la déviation lente de la trajectoire due au patinage des roues. Tester sur différents sols.

Vitesse différente pour chaque moteur
La procédure delay() est monotâche ; elle bloque le processeur pendant le délai. Les deux moteurs ont besoin de délais différents. Pour bien faire, il faut une gestion par interruptions, ce que la librairie fournira (celles de l'Arduino ne conviennent pas, elles sont faites pour des moteurs 4 phases).
Une solution à notre portée est de prendre une décision d'évolution très fréquente (50 microsecondes par exemple) et utiliser deux compteurs de période pour décider chaque fois si le délai de chaque moteur est passé, et faire un pas si oui.
La période est calculée à partir de la vitesse par division et la vitesse peut être exprimée en mm/s. c'est le grand avantage du C sur l'assembleur, on mutiplie et divise facilement, sans se préoccuper de la place mémoire et du temps de calcul (mais attention aux 50 microsecondes !).

Donc pour chaque moteur, toutes les 50 microsecondes, on décompte un compteur, et lorsqu'il arrive à zéro, on fait un pas et réinitialise le décompteur Avec 50 microsecondes de cycle et 1200 microsecondes de pas minimum, le compteur est réinitialisé à la valeur 24 à la vitesse maximale. A quel moment faut-il convertir la vitesse en durée) ? Chaque 50 microsecondes ? Le plus facile, mais on ne va pas changer la vitesse si souvent. Toutes les 20 ms ? En même temps que l'on prend des décisions sur les capteurs. Logique, mais c'est une 3^{e} tâche qui va prendre plus que 50 microsecondes et va fausser l'espace entre les pas.

Programmons la solution basée sur des périoodes. Définissons les variables PeriodeDroite PeriodeGauche et les décompteurs associés PD et PG.
PeriodeDroite et PeriodeGauche ont des valeurs signées, PD et PG sont initialisés avec leur valeur absolue (voir le programme complet).

```
void  loop ()  {
      delayMicrosecondes (50) ;
      PD--; if (PD==0)  {
            PD = PeriodeDroite ;
            if (PeriodeDroite >= 0)   { id++; if(id==6) id=0; }
            else   { id=id-1;  if(id==-1) id=5; }
            }
      PG--; if (PG==0)  {
            PG = PeriodeGauche ;
            if (PeriodeGauche >= 0)   { id++; if(id==6) id=0; }
            else   { id=id-1;  if(id==-1) id=5; }
            }
      PORTB = AvDroite[id] | AvGauche[ig] ;
}
```

Il n'a a plus qu'à déclarer le début correctement pour tester le programme `Wd2Pas2.pde`

(pas de suite tant qu'il n'y a pas un lecteur qui me dit avoir fait les exercices) jdn 110306