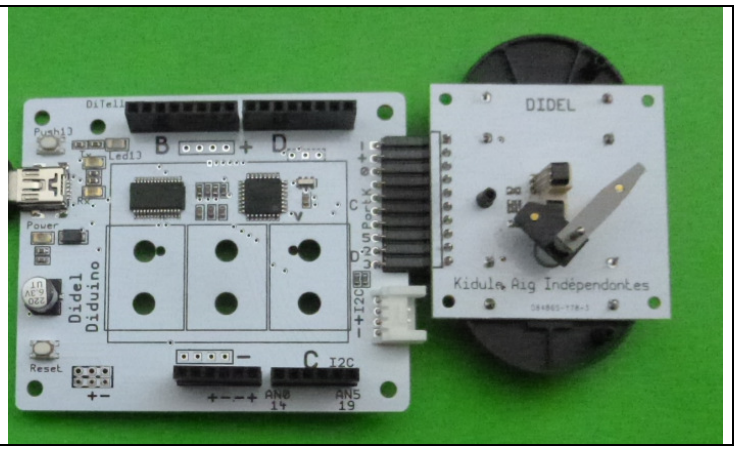
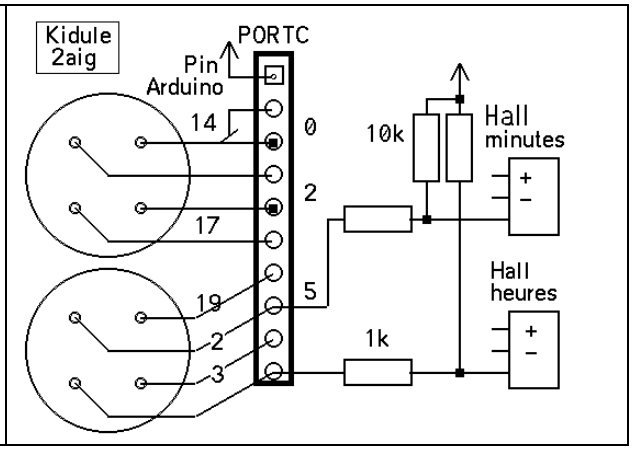
Deux moteurs pas-à-pas sur le même axe

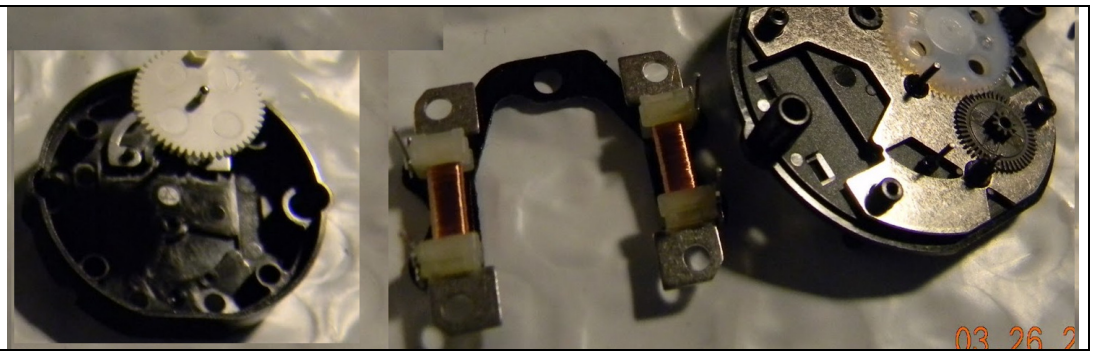
Le moteur double Vid28.05 (Google "Vid28") est utilisé en instrumentation (tableau de bord des voitures). Ce n'est pas un moteur à 4 phases, comme les moteurs documentés sous Arduino, mais un moteur Lavet à 6 phases. Les bobines ont une résistance de 280 Ohm, on peut donc les connecter directement par des sorties d'un microcontrôleur. Les axes sont entraînés via un réducteur 1:180.
Le moteur du haut commande la grande aiguille des minutes.



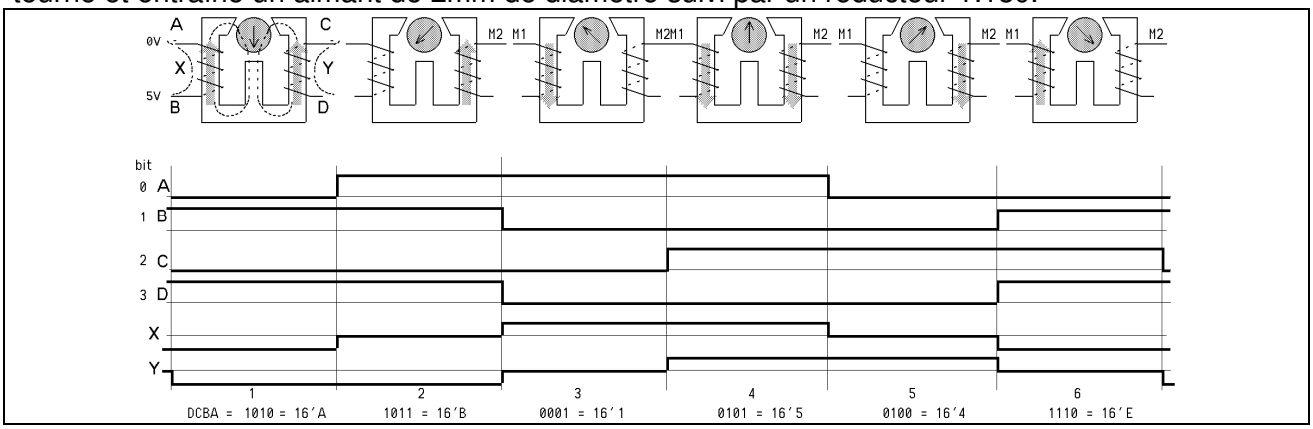
Des capteurs de hall ont été ajoutés avec résistance de protection. connecteur broche 9 pour les heures, 7 pour les minutes.
Cet interface montre la nécessité d'être attentif aux noms des signaux avant de commencer à programmer:
Numéros des broches du connecteur 1 à 10
Bits du port Kidules 0 à 7
Pins Arduino 14-19 et 2-3
Ports AVR 328 PORTC bits 0-5 PORTD bits 2-3



La construction interne des moteurs est simple avec très peu de pièces.
Ce moteur est utilisé sur les tableaux de bord des voitures.



Séquence à 3 phases, 2 bobines séparées
La séquence des phases est donnée ci-dessous. Le champ magnétique créé par des deux bobines tourne et entraîne un aimant de 2mm de diamètre suivi par un réducteur 1:180.



Pour le moteur connecté sur les 4 bits de poids faible du PORTC, la séquence de pas reprend directement les valeurs binaires du graphique ci-dessus. Elle se code dans un tableau de 6 valeurs.

```
byte step[6]={0x0A,0x0B,0x01,0x05,0x04,0x0E};
```

que l'on parcourt avec un compteur par 6, avec un `if` pour recommencer

```
i++; if(i==6) i=0; // compte 0 1 2 3 4 5 0 1 2 ....
```

Pour changer de sens, il faut compter dans l'autre sens

```
i--; if(i==0) i=6; // décompte 6 5 4 3 2 1 0 6
```

Il faut corriger l'index en demandant `etat[i-1]`

Le Ki2Aig incorpore un double moteur pas-à-pas qui commande deux aiguilles coaxiales.
Le moteur du haut est câblé sur les bits 0-3 et commande l'aiguille des heures.

Les programmes se trouvent dans le dossier par défaut www.didel.com/kidules/CKiPas2Aig.zip

Le programme `CKiPa1.ino` fait tourner l'aiguille des heures. Le moteur est câblé sur les 4 bits de poids faible du PORTC.
Il suffit de l'initialiser en sortie et d'écrire directement sur ce port.
Entre chaque pas, il faut naturellement une attente qui fixe la vitesse de rotation. La fréquence maximale des pas est supérieure à 1 kHz, les délais sont en microsecondes.

```
// KiPas1.ino
#define Perio 2000  // periode min ~1500 à 5V
// le moteur est sur les bits 0-3 du PORTC
byte i=0;
byte etat[6]={0x0A,0x0B,0x01,0x05,0x04,0x0E};

void setup() {
  DDRC = 0b11111111 ; //direction
  PORTC = 0 ;  //moteur non excité
}

void loop()  {
  i++; if(i==6) i=0;  // compte 0 1 2 3 4 5 0 1 2 ....
  PORTC = etat[i];
  delayMicroseconds(Perio); // min ~1500 à 5V
}
```

Pour agir sur le moteurs des minutes, il faut remarquer qu'il a 2 bits sur le portC et 2 bits sur le PortD. Il faut initialiser ces 2 ports, en ne modifiant pour le portD que les bits concernées. On utilise la même table et on envoie pour ce premier test 2 bits sur C et 2 bits sur D.

```
//CKiPa1m.ino  Avance l'aiguille des minutes
byte avance[6]=
{0x0A,0x0B,0x01,0x05,0x04,0x0E};
void setup()  {
  DDRC  = 0b11111111 ;
  PORTC = 0 ;
  DDRD |= 0b00001100 ;
  PORTD = 0 ;
}
```

```
void loop() {
  for (int i=0; i<6 ; i++ )  {
    PORTC = (avance[i] << 4) & 0x30 ;
 // 00ba0000
    PORTD = avance[i] & 0x0C ;
 // 0000dc00
    delayMicroseconds(5000);
  }
}
```

On voit que avec cette table, l'aiguille des minutes tourne en sens inverse. On peut changer la table, ou permuter deux bits sur une des deux bobines, ce qui aurait été facile à faire sur le circuit imprimé, mais est plus délicat par programmation (<4 instructions supplémentaires!).
La solution est comme on a vu, de parcourir la table à l'envers.

Exercice: faire tourner les 2 moteurs en même temps dans le même sens.

Exercice: Aller-retour en modifiant la vitesse.

2 moteurs 2 vitesses

Le processeur soit s'occuper de 2 moteurs "en même temps". C'est du multitâche. Le processeur va s'occuper alternativement d'un moteur puis de l'autre, en commutant assez rapidement. Cela peut se faire par interruption (une horloge interrompt et appelle les fonctions à exécuter) ou en séquence programmée, sans avoir le droit de s'attarder, ce que l'on va faire ici.
Toutes les 50 microsecondes, on fait avancer un compteur pour chaque moteur. A une valeur limite, on réinitialise et fait un pas. La vitesse dépend des valeurs limites Per1 Per2, qui sont des multiples de 50 microsecondes.

```
// KiPas5.ino 2 vitesses
```

```
// les moteurs sont sur les bits 0-3 et 4-7 du PORTC
byte hav[6]={0x0A,0x0B,0x01,0x05,0x04,0x0E};
byte hrec[6]={0x0E,0x04,0x05,0x01,0x0B,0x0A};
byte minav[6]={0xA0,0x0B0,0x010,0x050,0x040,0x0E0};
byte minrec[6]={0x0E0,0x040,0x050,0x010,0x0B0,0x0A0};
void setup() {
  DDRC = 0b11111111 ; //direction
  PORTC = 0 ;  //moteur non excité
}
//Dans une boucle de 50us, on prédivise pour chaque moteur
#define Per1 20 // periode min 20x50 = 1ms
#define Per2 100  // le 2e moteur est 5 fois plus lent
byte p1,p2, i,j;
void loop()  {
  delayMicroseconds(50);
  if (p1++>Per1) { p1=0; i++; if(i==6) i=0; }
  if (p2++>Per2) { p2=0; j++; if(j==6) j=0; }
  PORTC = hav[i] | minav[j];
}
```

Lecture des Hall

On doit brièvement couper l'excitation moteur pour lire les capteurs de Hall, qui donnent un état 0 lorsque le champ magnétique est présent. C'est inutile de lire trop souvent, tous les 6 pas correspond à un angle de 2 degrés, selon la proximité de l'aimant, le Hall est actif pour 3 à 8 degrés de déplacement de l'aiguille.

Pour le programme de test, on a défini les deux actions qui définissent la direction des bits sur lesquels les Hall sont lus. Un délai de 500 us est nécessaire pour que le signal du Hall se stabilise.
(voir www.didel.com/kidules/CKiClock.pdf)

```
//CKiPa5.ino  Test des 2 capteurs deHall
. . . .  définitions, set-up
void loop () {
  for (int i=0; i<6 ; i++ )  {
    KiWrite ( etat[i] | (etat[i]<<4) );
    delayMicroseconds(Periode);
  }
  ModeHall;
  delayMicroseconds (500); // temps commut
  if (MinuteTop) { LedOn;
     delay (20);  // stoppe moteur
  }
  if (HeureTop) { LedOn;
     delay (20);  // stoppe moteur
  }
  ModePas;
  LedOff;
}
```

jdn 140106/160620/180602