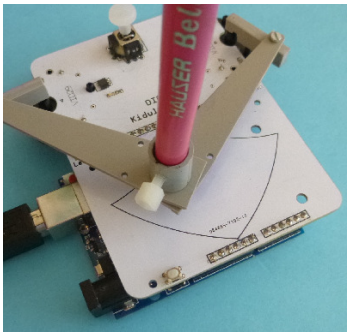
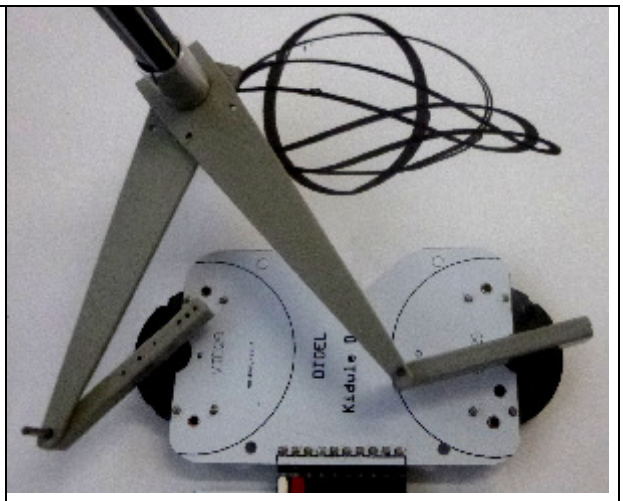
Delta à 2 dimension avec deux moteurs pas-à-pas

Le kidule Delta, inspiré de l'invention du prof Clavel, utilise deux moteurs pas à pas Vid26.05 (google "Vid26") qui sont utilisé en instrumentation (tableau de bord des voitures). Les bobines ont une résistance de ~1k Ohm, on peut donc les connecter directement par des sorties d'un microcontrôleur.

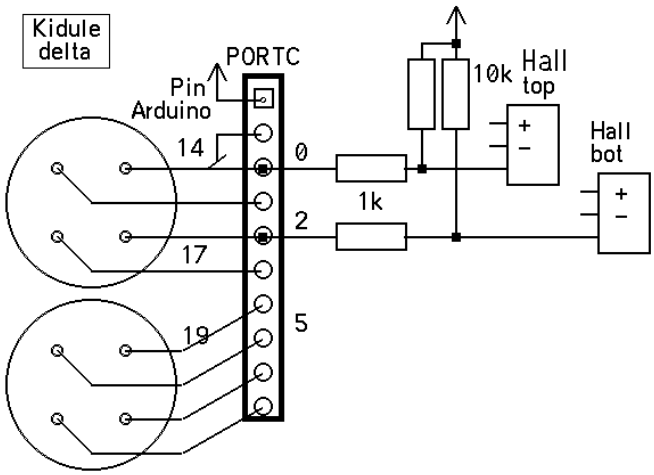


Existe en EduShield



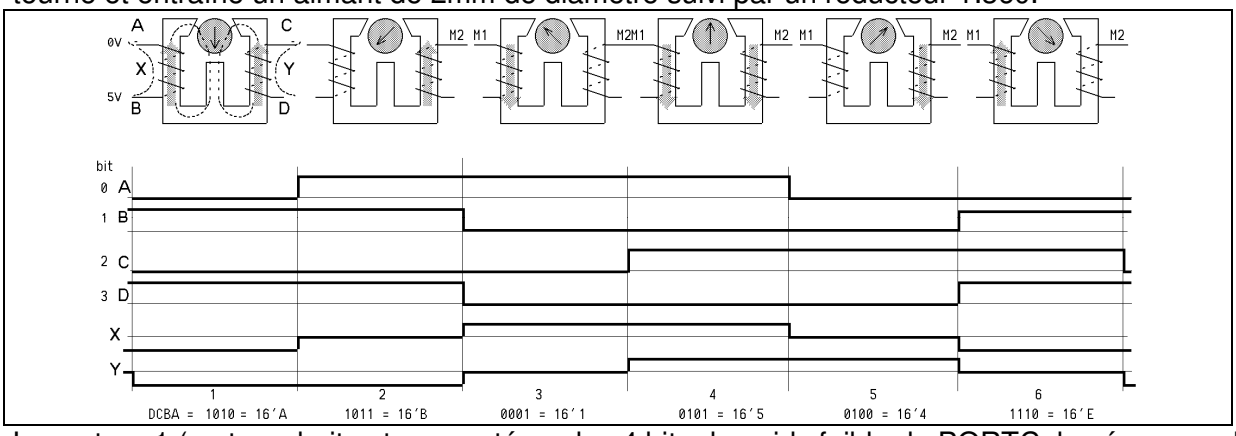
Des capteurs de hall ont été ajoutés avec résistance de protection. bit 7 pour les heures, 5 pour les minutes. Les 4 bits 4-7 (qui sont sur 2 ports différents) doivent être en entrée pour lire les Hall.

On ne connectera pas les bras pour les premiers exercices dont le but est de comprendre la programmation, et pas de dessiner!



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 1:360.



Le moteur 1 (moteur droit est 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 etat[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 ....
```

Le 2e moteur est à cheval sur 2 ports du processeur. La fonction WriteKi (); permettra de voir les 2 moteurs comme câblés sur les 8 bits de poids faible et fort de la variable 8 bits données à writeKi(). Pour faire tourner le moteur, il suffit de parcourir sa table en incrémentant le pointeur circulairement avec un temps d'attente.

```
      i++; if(i==6) i=0;
      position = etat[i];
```

Une boucle for est possible, si on s'occupe d'un seul moteur.

```
      for (int i=0; i<6 ; i++ )
      position = etat[i];
```

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

`i--; if(i==0) i=6 ; ou for (int i=5; i>0 ; i-- )` décompte 6 5 4 3 2 1 6 5

Il faut corriger l'index en accédant la table `etat[i-1];`

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

Le programme `CkiDel1.ino` fait tourner le moteur 1
Entre chaque pas, il faut naturellement une attente qui fixe la vitesse de rotation. La fréquence maximale des pas est supérieure à 1 kHz, on va faire des délais en microsecondes.

```
//CkiDel1.ino sens horaire moteur 1 (top)
byte etat[6]={0x0A,0x0B,0x01,0x05,0x04,0x0E};
void setup() {
  DDRC = 0b11111111 ; //direction
  PORTC = 0 ;  //moteur non excité
}
void loop() {
  for (int i=0; i<6 ; i++ )  {
    PORTC = etat[i];
    delayMicroseconds(2000); // max vit 1000 à 3.5V
  }
}
```

Le programme `CKiDel2.ino` fait tourner les deux moteurs à la même vitesse.
Mais la solution d'avoir une table pour les deux moteurs n'est pas compatible avec la flexibilité que l'on voudrait avoir pour changer le sens et la vitesse des deux moteurs. Utilisons une seule table comme dans `CkiDel1.ino`. et préparons la variable `pos` 8 bits pour WriteKi()
Sens horaire moteur Top

```
  i++; if(i==6) i=0;
  pos = etat[i];
```

Même sens moteur Bot

```
  j--; if(j==0) j=6
  pos |= etat[j-1]<<4;
```

On envoie

```
   WriteKi (pos);
```

On attend selon la vitesse souhaitée.
Programme à faire en exercice. (solution CKiDel3.ino)

```
//CKiDel2.ino  Avec deux moteurs
// Les deux moteurs avancent ensemble dans le même sens
byte etat[6]={0x71,0x69,0xE8,0x8E,0x96,0x17};
// les deux moteurs tournent en sens contraire
//byte etat[6]={0x77,0x66,0xEE,0x88,0x99,0x11};

void setup()
{
  DDRC   = 0b00111111;  // bits 0-5
  DDRD  |= 0b00001100 ;  // bits 6.7
  PORTC  = 0b00000000;  // init à 0
  PORTD &= 0b11110011;  // init à 0
}

void WriteKi (byte kk)  {
  PORTC  = kk & 0x3F ;       // copie les 6 bits
  PORTD &= (kk>>4) & ~0x0C; // copie les zéros
  PORTD |= (kk>>4) &  0x0C; //copie les uns
}

void loop() {
  for (byte i=0; i<6; i++) {
    WriteKi (etat[i]) ;
    delayMicroseconds(3000); // max vit 1000 à 3.5V
  }
}
```

Si on veut que les deux moteurs tournent à des vitesses différentes, il faut que se baser sur un délai qui est le PGCD des deux délais. Par exemple, les moteurs ont des délais de 2500 et 6000. La boucle durera 500; toutes les 5 fois le moteur1 fera un pas, toutes les 12 fois le moteur 2 avancera.

```
//CKiDel4.ino  Deux moteurs vitesse diff.
void setup()
{
  DDRC   = 0b00111111;  // bits 0-5
  DDRD  |= 0b00001100 ;  // bits 6.7
  PORTC  = 0b00000000;  // init à 0
  PORTD &= 0b11110011;  // init à 0
}

void WriteKi (byte kk)  {
  PORTC  = kk & 0x3F ;       // copie les 6 bits
  PORTD &= (kk>>4) & ~0x0C; // copie les zéros
  PORTD |= (kk>>4) &  0x0C; //copie les uns
}
```

```
byte
etat[6]={0x0A,0x0B,0x01,0x05,0x04,0x0E};
byte pos;
byte i,j;
//Les rapports de vitesse sont 5/12
#define Div1 5
#define Div2 12
#define Perio 500
byte pdiv1, pdiv2;
void loop() {
  delayMicroseconds (Perio);
  if (pdiv1++ > Div1) { pdiv1=0;
    if(i++>5) i=0;
    pos = etat[i] ;
  }
  if (pdiv2++>Div2) { pdiv2=0;
```

```
    if(j--==0) j=6;
    pos |= etat[j-1]<<4 ;
  }
  WriteKi (pos);
}
```

On a agi sur la vitesse. Il faut aussi savoir agir sur l'angle, c'est-à-dire sur le nombre de pas.

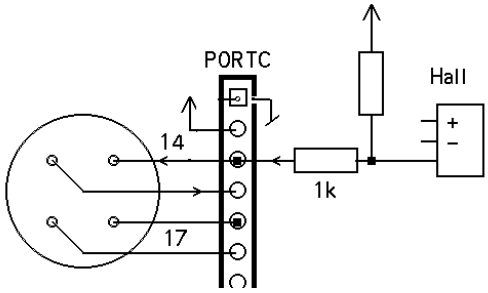
```
// CKiDel5.ino avance-recule de 100 pas
byte etat[6]=
{0x0A,0x0B,0x01,0x05,0x04,0x0E};
void setup() {
   DDRC = 0b11111111 ; //direction
   PORTC = 0 ;  //moteur non excité
}
```

```
void loop()  {
 // on fait 100 pas dans un sens
 for (byte j=0; j<100;j++) {
   if(i++>5) i=0;
   PORTC = etat[i] ;
   delayMicroseconds(2000);
 }
 // on fait 100 pas en sens inverse
 for (byte j=0; j<100;j++) {
   if(i--==0) i=6;
   PORTC = etat[i-1] ;
   delayMicroseconds(2000);
   }
 }
}
```

Exercices: Aller-retour en modifiant la vitesse.
Avance irrégulière (utiliser random (min,max) pour la vitesse et le nombre de pas)

Capteur de position des aiguilles

Deux capteurs de Hall permettent de repérer la position des aiguilles à 6h30, avec une précision de quelques minutes, que l'on peut calibrer pour être plus précis. Ces capteurs sont multiplexés avec les moteurs. Une résistance de protection limite le courant vers la sortie du Hall quand on commande les moteurs. Pour lire les Hall, il faut mettre les 4 lignes en entrée Le bobines ne sont plus excitées et on peut lire l'état du Hall minutes aussi bien sur les bits 0 et 1 du PortC (pins Arduino 14 et 15). Le Hall des heures est sur les pins 16 et 17.



Le programme `CKiDel6.ino` stoppe l'aiguille des minutes à chaque passage sur le capteur.

C'est important de bien comprendre ce programme.
Les définitions précisent le câblage et la direction des ports.

On a décidé de lire le capteur tous les 6 pas.

```
//CKiDel6.ino La led13 s'allume si l'un des capteur est actif
#include "PortK.h"

#define Led 13  // pin PORTB
#define LedOn digitalWrite (Led,HIGH)
#define LedOff digitalWrite (Led,LOW)

#define bHallTop 0  // bit portC
#define bHallBot 2  // portC
#define TopHome  !(PINC & 1<<bHallTop)
#define BotHome  !(PINC & 1<<bHallBot)
#define ModePas  DDRC = 0b00111111
#define ModeHall DDRC = 0b00110000
```

```
byte etat[6]={0x01,0x09,0x08,0x0E,0x06,0x07};
void setup() {
  ModePas ; //direction
  pinMode (Led,OUTPUT);
  LedOff;
}
#define Periode 2000
void loop () {
  for (int i=0; i<6 ; i++ )  {
      PORTC = etat[i];
      delayMicroseconds(Periode);
  }
  delayMicroseconds (500); // temps
commut
  if (TopHome) {
     LedOn;
     delay (20);  // stoppe moteur
  }
  if (BotHome) {
     LedOn;
     delay (50);  // stoppe moteur
  }
  LedOff;
  ModePas;
}
```

Modifions ce programme pour positionner les aiguilles. On travaille avec la même vitesse pour les 2 moteurs, et quatre bits dans le registre `step` décident quels moteurs tournent et dans quel sens.

```
byte etat[6]={0x0A,0x0B,0x01,0x05,0x04,0x0E};
int posTop, posBot; // nombre de pas depuis le début
byte step;
#define bTopCW  0  // moteur top clock wise
#define bTopCCW 1  // moteur top counter clock wise
#define bBotCW  2  // moteur top clock wise
#define bBotCCW 3  // moteur top counter clock wise
volatile byte i;  volatile byte j;
void DoStep () {  // avance selon mode
```

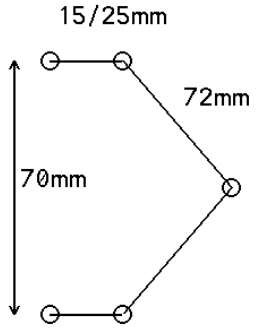
```
if (step & 1<<bTopCW) {
  posTop++;
  i++; if(i==6) i=0;
}
if (step & 1<<bTopCCW) {
  posTop--;
  i--; if(i==0) i=6;
}
```

```
  if (step & 1<<bBotCCW) {
    posBot++;
    j++; if(j==6) j=0;
  }
  if (step & 1<<bBotCW) {
    posBot--;
    j--; if(j==0) j=6;
  }
  KiWrite ( etat[i] | (etat[j]<<4) );
}
```

Le programme de test se trouve sous CKiDel7.ino. Le programme CKiDel8.ino positionne les aiguilles.

Générer des figures

Pour dessiner un segment de droite ou un cercle, il faut appliquer des transformation géométriques que la libraire math d'Arduino permet.
Les dimensions des leviers sont données c-contre.
Une première fonction positionne les bras sur les capteurs.
La fonction SetAngle (top,bot); positionne les moteurs à l'angle voulu.
Une difficulté, en plus du calcul des angles, est de définir la finesse des pas.



Demo

Cette démo nécessiterait beaucoup d'explications. L'idée était de pouvoir choisir des mouvement différents sans ajouter du matériel. Le poussoir de la carte Diduino a d'abord été utilisé pour stopper et redémarrer le mouvement quand on installe, règle la plume, change de papier.
Le perfectionnement a été de tenir compte du temps de pression. Toutes les 0.6 secondes environ, on passe au mode suivant, sans feedback immédiat, mais quand on relâche, le clignotement montre le nombre de secondes, donc le mode 1,2,3 (saturé à 3) Une impulsion brève redonne le mode 0 par défaut, sans clignotement.
Quand on presse la première fois, ou agit sur le premier moteur, quand on presse la 2e fois pour relancer le dessin, on agit sur le 2e moteur.

```
// CkiDelDemo.ino Sélection selon durée pressions
//
// poussoir pour arrêt /départ et modif
#define bPous 5 // bit 5 PORTB  arduino 13
#define PousOn (PINB&(1<<bPous))
#define LedOn PORTB |= 1<<bPous
#define LedOff PORTB &= ~(1<<bPous)
#define ModePous DDRB &= ~(1<<bPous)
#define ModeLed DDRB |= (1<<bPous)

void setup()  {
  DDRB   &= ~(1<<bPous) ;
  DDRC   = 0b00111111;  // bits 0-5
  DDRD  |= 0b00001100 ;  // bits 6.7
  PORTC  = 0b00000000;  // init à 0
  PORTD &= 0b11110011;  // init à 0
}
void WriteKi (byte kk) {
  PORTC  = kk & 0x3F ;    // cipie les 6 bits
  PORTD &= !0x0C ;       // prépare un troutrou
  PORTD |= (kk>>4) & 0x0C; //copie les 2 bits
}

byte etat[6]={0x0A,0x0B,0x01,0x05,0x04,0x0E};
#define Perio 500
char div1=12,div2=12;
byte pdiv1, pdiv2;
byte pos;
byte i,j;
int c1,c2;
char taPerPas [] = {7, -4, 13, 16};
void Bloque () {  // on bloque entre 2 poussoirs
  if (PousOn) {    // on bloque jusqu'à une 2e
action
    while (PousOn)  {delay (20); c1++;}
    c1 >>= 5;  if (c1 >3) c1=3;
    div1 = taPerPas [c1];
    Clignote (c1);
    while (!PousOn) {delay (20); }
    while (PousOn)  {delay (20); c2++;}
    c2 >>= 5; if (c2 >3) c2=3;
    div2 = taPerPas [c2];
    Clignote (c2);
  }
}
```

```
void Clignote (byte ncli) { //  clignote
  ModeLed;
  for (int k=0; k < ncli; k++)  {
    delay(300);
    LedOn ;
    delay(200);
    LedOff ;
  }
  ModePous;
}

void loop() {
  delayMicroseconds (Perio);
  Bloque();
  if (div1>0) {
    if (pdiv1++ > div1) {  // pas si div1 periodes
      pdiv1=0;
      if(i++>5) i=0;
      pos = etat[i] ;
    }
  }
  else {  // neg
    if (pdiv1++ > -div1) {
      pdiv1=0;
      if(i--==0) i=6;
      pos = etat[i-1] ;
    }
  }

  if (div2<0) {  //neg
    if (pdiv2++ > -div2) {
      pdiv2=0;
      if(j++>5) j=0;
      pos |= etat[j]<<4 ;
    }
  }
  else {
    if (pdiv2++ > div2) {
      pdiv2=0;
      if(j--==0) j=6;
      pos |= etat[j-1]<<4 ;
    }
  }
  WriteKi (pos);
}
```

jdn 140104/140207