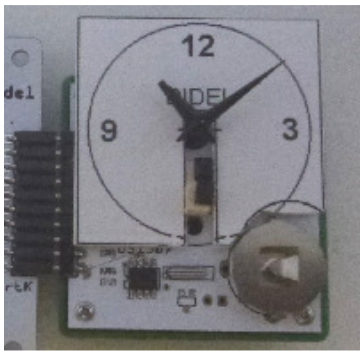
http://www.didel.com/ **info@didel.com**

www.didel.com/kidules/CKiClock.pdf

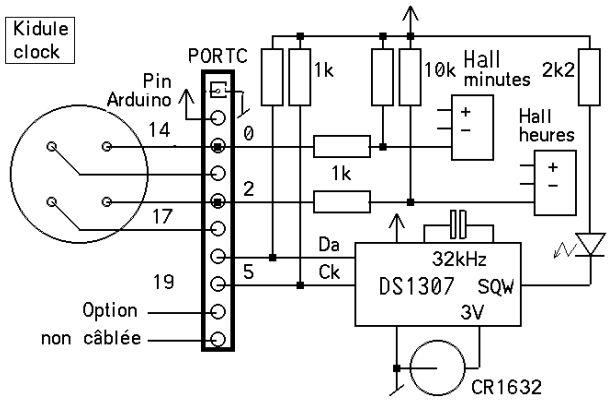
Kidule Clock



Vous devez avoir le Kidule-Clock ou un montage équivalent
Ce document suppose que les notions de bases de la programmation C/Arduino sont acquise. Les liens pour se former et progresser se trouvent dans www.didel.com/kidules/Liens.pdf
Le Kidule-Clock comporte un moteur pas-à-pas, deux capteurs de Hall et un circuit horloge I2C.
C'est un outil pédagogique très intéressant pour les écoles techniques pour étudier le moteur pas à pas, I2C, l'économie d'énergie.

1 Schéma

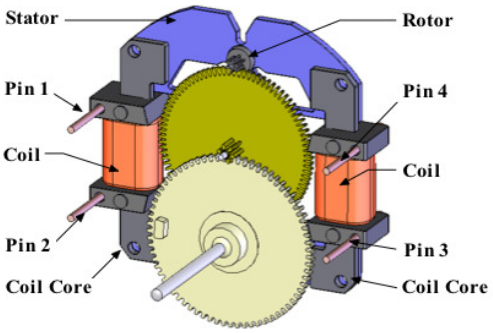
Le kidule Clock comporte, en plus du moteur horloger, deux capteurs pour repérer la position des aiguilles et un circuit DS1307. Ce schéma est expliqué au fur et à mesure de sa mise en œuvre. Les signaux sont câblés sur le portC du microcontrôleur AtMega328 de la carte Diduino (connecteur Kidule). Un adaptateur est disponible pour les cartes Arduino.

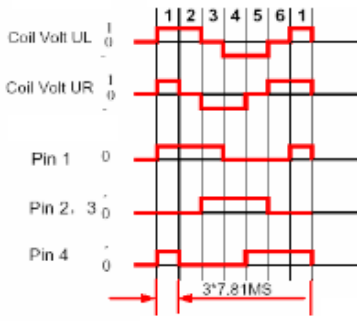


2 Moteur horloge

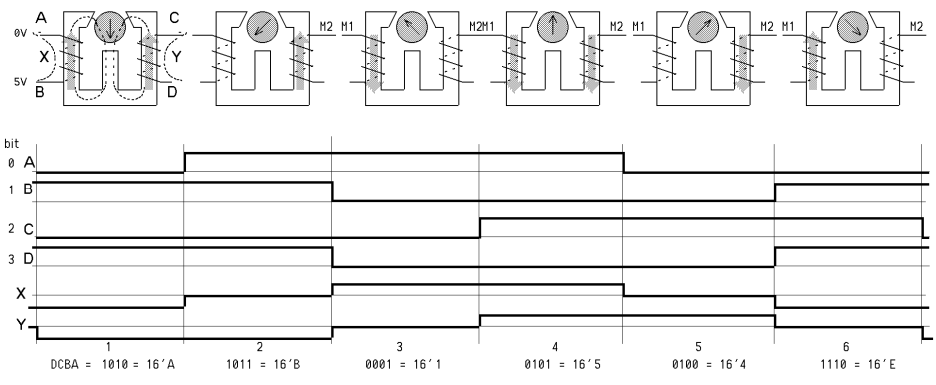
Le moteur est un Wellgain Vid69 de type Lavet.
http://www.vid.wellgain.com/UploadFile/down/2009111391212_VID69%20manual%20EN-080623.pdf

Le rotor est un aimant de de 2mm de diamètre. Pour le moteur horloge, une première réduction d'un facteur 12 commande l'aiguille des minutes. Deux pignons supplémentaires commandent l'aiguille des heures.





La séquence à 6 phases est donnée ci-dessous. Les champs X et Y créent le champ tournant. On remarque que l'on peut commander le moteur par 3 fils, mais dans le cas d'une commande directe, cela augmente le courant sur la sortie commune du microcontrôleur.



Note: Les programmes nommés CKiCkxx.ino se trouvent sous www.didel.com/kidules/CKiClock.zip (attention aux minuscules et majuscules si vous ne pouvez pas cliquer sur les liens: C–Ki–Ck-P1). Ce sont en général des "programmes amorces" qui doivent initier des variantes. Pour chaque variante, renommer le fichier avant de le modifier. Arduino permet de garder plusieurs programmes dans l'écran. Si vous n'êtes pas familier avec notre usage des librairies locales, consultez www.didel.com/C/FichiersImportes.pdf

3 Commande

Si le moteur est connecté sur les 4 bits de poids faible d'un port, 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 ....
```

Le programme qui fait tourner le moteur pas-à-pas à vitesse constante est simple:
CKiCkP1.ino

```
char i=0;
byte etat[6]={0x0A,0x0B,0x01,0x05,0x04,0x0E};
void loop()
{
   i++; if(i==6) i=0;  // compte 0 1 2 3 4 5 0 1 2 ....
   PORTC = etat[i];
   delayMicroseconds(1000); // min ~500 à 5V
 }
```

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]` --- a vous de tester, créer CKiCkP1back.ino

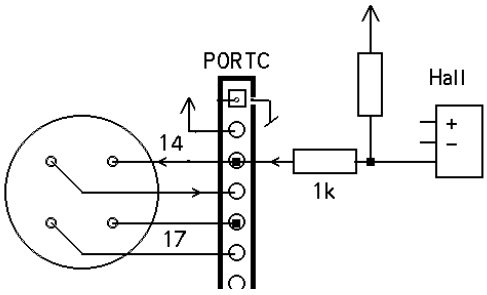
On peut accélérer le moteur, déterminer sa vitesse maximum de démarrage, sa vitesse maximum en modifiant `CKiCkP2.ino`. La Led au bas du Kidule (pin2) est utilisé pour ce test. La led vers le milieu du connecteur est parfois allumée, parfois éteinte selon l'état du circuit horloge.

Post et pré-modification

Le moteur pas-à-pas aide à comprendre une notion importante. Un pointeur (i dans les exemples ci-dessus) mémorise quelle est la position du moteurs. Mais est-ce la position avant l'instruction, ou après l'instruction ? Dans le premier cas, on doit calculer la position suivante avant d'assigner la position. Dans le second, on assigne et on décide quelle sera la position suivante.
Le moteur est dans une position donnée et on ne sait pas si l'ordre suivant le fera tourner dans un sens ou dans l'autre. Donc il faut calculer la position suivante et l'appliquer, et pas l'inverse.

4 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.



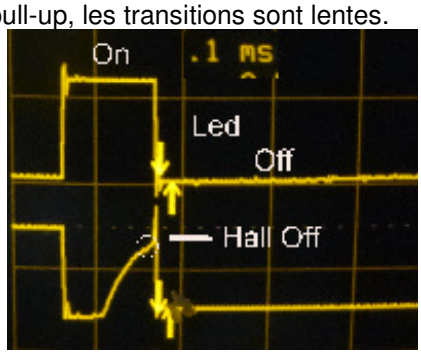
Le programme `CKiCkP3.ino` stoppe l'aiguille des minutes à chaque passage sur le capteur.

C'est important de bien comprendre ce programme.
Le définitions précisent le câblage et la direction des ports. On remarque que la Led, câblée sur le portD puisque le PORTC n'a que 6 bits, est définie "a la Arduino", ce qui n'intervient que dans les définitions pour garantir la portabilité. Dans le setup, on initialise la direction du PORTC, de la Led et on dit que le moteur est arrêté et la Led éteinte.
On pourrait remarquer que ce n'est pas nécessaire, après un cycles, tout sera configuré. Mais c'est très dangereux de façon générale dans une applications de ne pas donner l'état initial.
Le programme est simple, mais sa mise au point a nécessité l'usage d'un ocilloscope. Quand on change la direction pour lire le Hall, l'établissement des signaux prend 100 microsecondes. Le signal sur la Led synchronise l'oscilloscope.

```
// CKiCkP3.ino Test capteur de Hall
#define Led 2  // pin PORTD
#define LedOn digitalWrite (Led,LOW)
#define LedOff digitalWrite (Led,HIGH)
#define bHallMin 2  // bit portC
#define bHallHeure 0  // portC
#define ModePas  DDRC = 0b00111111
#define ModeHall DDRC = 0b00110000
#define MinuteTop !(PINC & 1<<bHallMin)
#define HeureTop  !(PINC & 1<<bHallHeure)

byte etat[6] =
{0x0A,0x0B,0x01,0x05,0x04,0x0E};
void setup() {
  ModePas ; //direction
  pinMode (Led,OUTPUT);
  PORTC = 0 ;
  LedOff;
}
```

Avec une pull-up, les transitions sont lentes.



On a décidé de lire le capteur tous les 6 pas. Quelles sont les justifications?

```
#define Periode 2000
void loop () {
  for (int i=0; i<6 ; i++ )  {
      PORTC = etat[i];
      delayMicroseconds(Periode);
  }
  ModeHall;
  LedOn;
  delayMicroseconds (500); // temps commut
    if (MinuteTop) {
       delay (200);  // stoppe moteur
    }
   LedOff;
   ModePas;
}
```

Le programme `CKiCkP4.ino` teste les deux capteurs de Hall et visualise l'angle pendant lequel ils sont actifs.
Exercice: ecrire le programme `CKiCkP5.ino` qui positionne les aiguilles à midi, puis à 10h10.

5 Circuit horloge DS1307

Le circuit DS1307 a une interface de commande I2C. Une pile maintien l'heure et le quartz 32 kHz donne une précision de quelques secondes par jour.
La documentation du DS1307 se trouve sous http://datasheets.maximintegrated.com/en/ds/DS1307.pdf

On va tester ce circuit en utilisant la librairie Arduino <wire.h>, qui cache plusieurs caractéristiques du circuit par rapport à la documentation. Il faut comprendre que, avec cette librairie, le circuit a une adresse globale 7 bits, 104 en décimal, et des adresses locales 0à7 comme documentées page suivante. Les fonctions wire sont les suivantes
Wire.begin(); Appelé dans le set-up pour charger la librairie (maître par défaut)
Wire.beginTransmission(Adr1307); Envoie l'adresse
Wire.write(value); Ecrit une adresse ou un byte – peut se répéter
Wire.endTransmission(); Termine l'envoi
Wire.requestFrom(address, quantity); demande des bytes et les mets dans le tampon
while (Wire.available() { . . si on attend plusieurs mots, vider le tampon
byte x = **Wire.read();** lit dans le tampon

Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

Les nombres sont représentés en BCD; on trouve des librairies pour exécuter les conversion Ascii-BCD. Elles supposent le dialogues avec un clavier-écran.

Notre problème est de synchroniser la position des aiguilles avec les heures et minutes du circuit. La mise à l'heure pourra se faire depuis le clavier, ou par les poussoirs usuels qui font avancer l'aiguille et gardent le 1307 en synchronisme.

Un problème annexe est de repositionner les aiguilles après une remise en service (reprendre l'heure dans le 1307, se souvenir de l'heure de la coupure ou mettre les aiguilles en position connue, positionner selon l'heure.

Le but ici n'est pas de résoudre ces jolis projets d'étudiant, mais de vérifier les ordres à utiliser.

Sortie SQW

La sortie SQW pilote une Led que l'on peut allumer, éteindre, clignoter- Le registre OUT à l'adresse locale 7 contient 4 bits de mode. Le programme `KiCkW1.ino` initialise uniquement la sortie à 1 HZ. Le programme `KiCkW2.ino` permet d'observer les signaux Ck et Da à l'oscilloscope. Le programme `KiCkW3.ino` montre comment afficher les secondes sur le terminal.

Pour clignoter à 1 Hz, il faut envoyer l'adresse du circuit, puis écrire la valeur 0x10 ou 1<<SQWE si on a défini SQWE à 4.

```
#define Ad1307 104
#define AdControl 7
```

```
dans le setup:  Wire.begin ();
  Wire.beginTransmission(Ad1307);
  Wire.write (AdControl);
  Wire.write (0x10);  //
  Wire.endTransmission();
```

Mise à l'heure

Pour assigner les heures et minutes à 23h12, il faut aussi initialiser les secondes pour agir sur le bit CH qui doit valoir 0. La représentation est en 24 heures si le 5 du registre heures est à zéro.

```
#define Ad1307 104
#define AdSeconds 0
```

```
  byte heures   = 2*16 + 3 ;
  byte minutes  = 1*16 + 2 ;
  byte secondes = 0x00 ;
 Wire.beginTransmission(Ad1307);
  Wire.write (AdSeconds);
  Wire.write (secondes);
  Wire.write (minutes);
  Wire.write (heures);
  Wire.endTransmission();
```

Lecture de l'heure

```
Comme pour lire le terminal, les caractères
passent par un tampon
#define Ad1307 104
#define AdMinutes 1
```

L'affichage par Serial.print (xx,HEX) qui supprime les zéros non significatifs convient bien pour l'affichage de l'heure.

```
  Wire.beginTransmission(Ad1307);
  Wire.write(byte(0));
  Wire.endTransmission();
  Wire.requestFrom (Ad1307, 6);
   byte minutes = Wire.read();
   byte heures  = Wire.read ();
   Serial.print("Il est  ");
   Serial.print(heures,HEX);
   Serial.print(" h ");
   Serial.print(minutes,HEX);
   Serial.print(" mn ");
```

Le DS1307 contient une mémoire non volatile aux adresses 8 à 0x3F qui peut permettre des fonctionnalités intéressantes.

6 Bus I2C

Le Kidule Clock permet de bien comprendre le bus I2C en gérant directement les signaux Ck et Da. Le document www.didel.com/C/I2C.pdf explique comment programmer en C les transferts I2C. L'intérêt est que ces routines utilisent 400 bytes au lieu de plus de 2000, pour la même vitesse de transfert.

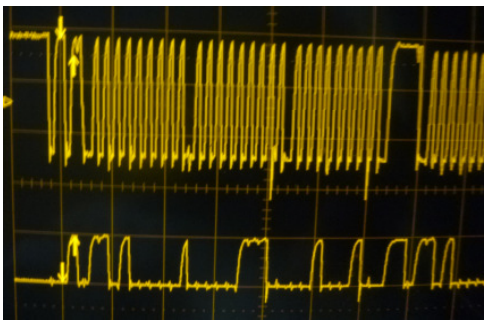
```
#define Ad1307Wr 0xD0
#define Ad1307Rd 0xD1
#define AdSeconds 0
```

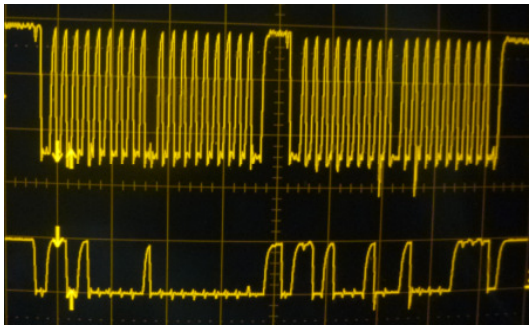
Mise à l'heure

```
  byte heures   = 2*16 + 3 ;
  byte minutes  = 1*16 + 2 ;
  byte secondes = 0 ;
  WrAd (Ad1307Wr);
  Wr8 (AdSeconds); Ack();
  Wr8 (secondes); Ack();
  Wr8 (minutes); Ack();
  Wr8 (heures); Na();
  Stop
```

Lecture de l'heure

```
  WrAd (Ad1307Wr);
  Wr8 (AdSeconds);
   Stop
  WrAd (1307Rd);
  byte secondes = ReadI2C(); GiveAck();
  byte minutes = ReadI2C();  GiveAck();
  byte heures  = ReadI2C();  GiveNack();
   Stop;
  Serial.print("Il est  ");
  Serial.print(heures,HEX);
  Serial.print(" h ");
  Serial.print(minutes,HEX);
  Serial.print(" mn ");
```





jdn 131227/180602