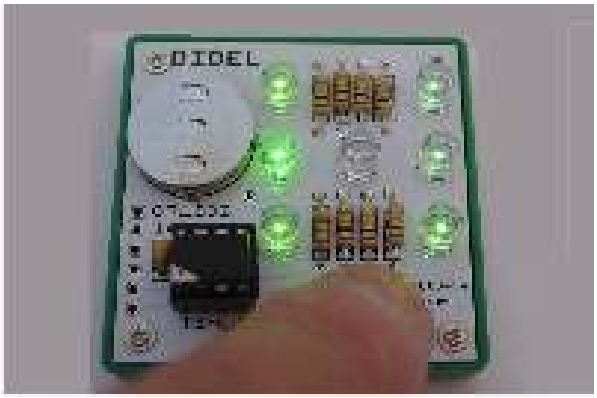
Diduino – reprogrammer le Dé de PYR (P.-Y. Rochat)

Le dé électronique est un kit intéressant à souder et jouer. Ce document montre qu'il peut contribuer à la culture numérique des élèves en expliquant la fonctionnalité interne, la programmation et les outils de développements.
Du point de vue électronique, ce dé n'a pas d'interrupteur: le processeur s'endort après une dizaine de secondes et la pile dure des années.
La pile a une capacité de 160 mAh. Allumé, le courant est de 1 à 3mA. Endormi, ~5 microAmpères. En une année, cela fait 0.005 x 24 x 365 = 44 mAh
Il est piloté par un AVR AtTiny 13 (ou 25) préprogrammé.
Ce document montre qu'on peut le reprogrammer facilement si on dispose d'une carte Arduino/Diduino et un shield AtTiny Programmer. Le but est alors de montrer aux élèves des programmes qu'ils comprennent facilement, qu'ils ont éventuellement déjà vus avec le Kidule dé, et de leur faire découvrir comment se fait la programmation.



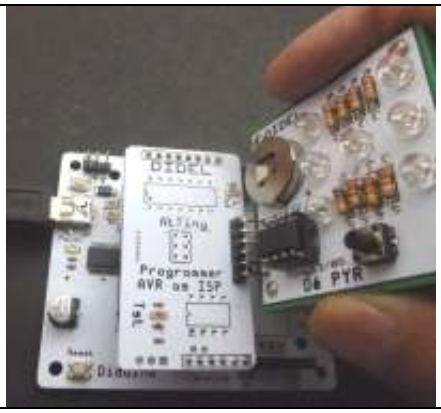
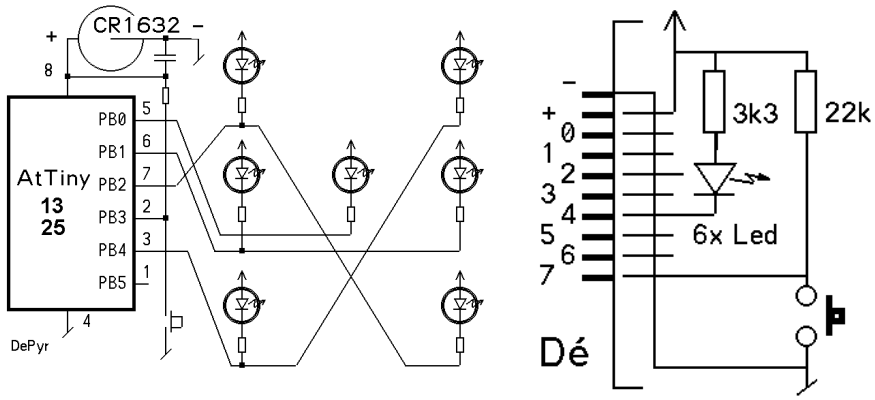


Schéma du dé
Le Kidule Dé est connecté sur un port 8 bits, ce qui permet de commander directement les 7 leds et le poussoir. Le processeur AtTiny13 a 8 broches, mais 3 broches sont réservées pour l'alimentation et la programmation. Il reste 5 pins, une pour le poussoir et 4 pour les leds. Comme toutes les combinaisons ne sont pas possibles sur un dé, il suffit de regrouper les leds opposées.



Les leds sont cablées sur des pins reliées à l'intérieur du processeur sur les bits du registre d'entrée/sortie appelé port B On voit sur le schéma que la correspondance est

bit 0 PB0 led du centre
bit 1 PB1 leds horizontales
bit 2 PB2 leds diagonales \
bit 3 PB3 poussoir (entrée)
bit 4 PB4 leds diagonales /
bit 5 PB5 réservé programmation

Pour les 6 combinaisons de leds, il faut activer les bits suivants:

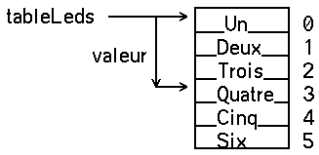
```
                        //(PB7)(PB6)(PB5) PB4  (PB3) PB2  PB1  PB0
    #define Zero   0x16  //  o    o    o    1    o    1    1    1   o sans effet
    #define Un     0x16  //  o    o    o    1    o    1    1    0   0=allumé
    #define Deux   0x15  //  o    o    o    1    o    1    0    1
    #define Trois  0x13  //  o    o    o    1    o    0    1    1
    #define Quatre 0x03  //  o    o    o    0    o    0    1    1
    #define Cinq   0x02  //  o    o    o    0    o    0    1    0
    #define Six    0x01  //  o    o    o    0    o    0    0    1
```

Pour afficher une position du dé, appelée valeur, on peut faire une suite de comparaisons:

```
// Procédure pour afficher la valeur;
void Affiche (byte valeur) {
  if (valeur==0) PORTB=Un;
  if (valeur==1) PORTB=Deux;
  if (valeur==2) PORTB=Trois;
  if (valeur==3) PORTB=Quatre;
  if (valeur==4) PORTB=Cinq;
  if (valeur==5) PORTB=Six;
}
```

Passer par une table est plus élégant et ne justifie pas une fonction. L'index, le pointeur dans table est numérotéde zéro à 5. Il en sera de même pour le compteur que l'on utilisera pour pointer dans la table.
Il faut définir comme variables:

```
byte valeur ;
byte tableLeds [] = {Un,Deux,Trois,Quatre,Cinq,Six}
```



La conversion se fait avec l'instruction:

```
PORTB = tableLeds [valeur] ;
```

Le dé qui roule, c'est un compteur, qui ici compte de 0 à 5.

```
valeur ++;                  // compte
if (valeur==6) {valeur=0}   //  == est une comparaison, = est une assignation
```

L'idée simple pour programmer un dé, c'est de le faire rouler quand on presse sur le poussoir. S'il tourne rapidement, on ne peut pas prévoir sur quelle valeur il va se bloquer, le hasard sera très bon.

```
while (PoussoirOff) {
   delay (10);
   valeur ++;
   if (valeur==6) {valeur=0}
   PORTB = tableLeds [valeur] ;
}
```

La condition `(PoussoirOn)` est une valeur booleenne (0 ou 1) qui résulte de la lecture de la pin PB3, en tenant compte du fait que lorsque l'on presse, la tension est de 0V, le processeur lit un zéro logique. Le C utilise un ! pour inverser une valeur booléenne, il faut définir

```
#define bPous  3    // no du bit sur lequel le poussoir est câblé
#define PoussoirOn (!(PINB&(1<<bPous))
```

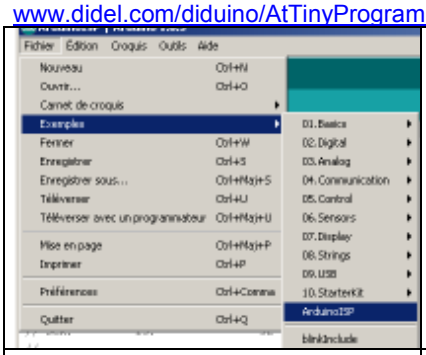
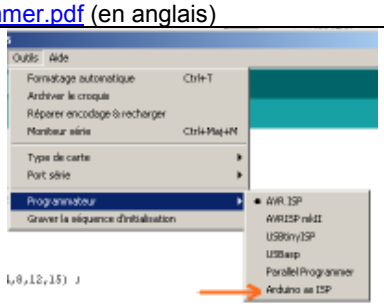
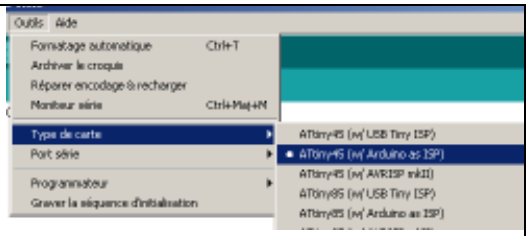
Le programme complet de ce dé est donc

```
// DeSimple.ino Le dé roule quand on presse
// Câblage et constantes
                    //(PB7)(PB6)(PB5) PB4 (PB3) PB2  PB1  PB0
#define Un     0x16  //  o    o    o    1    o    1    1    0   0=allumé
#define Deux   0x15  //  o    o    o    1    o    1    0    1
#define Trois  0x13  //  o    o    o    1    o    0    1    1
#define Quatre 0x03  //  o    o    o    0    o    0    1    1
#define Cinq   0x02  //  o    o    o    0    o    0    1    0
#define Six    0x01  //  o    o    o    0    o    0    0    1
 #define bPous  3    // no du bit sur lequel le poussoir est câblé
 #define PoussoirOff (PINB&(1<<bPous) // etat 1 si Off
//variables
byte valeur;
byte tableLeds [] = {Un,Deux,Trois,Quatre,Cinq,Six};
//programme (C compatible Arduino)
int main () {
  DDRB |= 0b00010111;  PB4,2,1,0 out, PB3 in
  while (1){                //faire toujours
    while (PoussoirOff) {   //faire si..
       delay (10);
       valeur ++;
       if (valeur==6) {valeur=0}
       PORTB = tableLeds [valeur] ;
     }
  }
}
```

Si on a relâché le poussoir, le programme ne fait rien d'autre que de vérifier que le poussoir n'est pas pressé. L'affichage reste dans le même état.
Le programme de PYR est plus complexe parce qu'il mémorise des états, s'endort si pas de touche pressée pendant 10 secondes, et se réveille quant on represse une touche. Charger le sketch De_2015
www.didel.com/diduino/De_2015.zip

jdn/150210/150515

Rappel: configurer la carte Diduino/Arduino en programmateur
La procédure détaillée est décrite sous
www.didel.com/diduino/ProgrammerUnAtTiny.pdf
www.didel.com/diduino/AtTinyProgrammer.pdf (en anglais)

		
Charger ArduinoISP	Selectionner le mode de programmation	Sélectionner le processeur cible

Annexe sans intérêt pédagogique

Adaptateur pour tester sur Diduino, sur portC kidules
Il faut dans le programme changer PORTB contre PORTC et DDRB contre DDRB

Adaptateur pour tester sur Diduino, sur portB
Le câblage des leds est le même, décalé. Les alimentations sont différentes.
Sur carte Arduino Uno, le + sur ce connecteur n'est pas utilisable. Il faut tirer un fil vers le + situé de l'autre côté de la carte.