www.didel.com/EduMod3.pdf

EduC – Programmer en C/Arduino - Module 3
Hasard, analogique en entrée et en sortie

Nombres au hasard

Arduino propose la fonction **`random (Min,Max)`**; qui rend un nombre entre Min et Max-1.
Vérifions en programmant un dé, qui à chaque PousG affiche un nombre entre 1 et 6;

```
//Edu31aLanceDe.ino  Lance un dé
#include "EduC.h"
void setup () { SetupEduC(); }

void WaitPousG () {
  while (PousG) {DelMs(20);}
  while (!PousG) {DelMs(20);}
}
byte val;
void loop() {
  WaitPousG();
  val= random (1,6+1);
  Digit(val);
}
```

En mettant des coches dans un tableau de 6 colonnes, on peut vérifier que la distribution est bonne. Mais à chaque reset, cela commence et continue de la même façon! Evidemment, c'est un programme, avec les mêmes conditions à chaque reset.
Pour modifier la séquence de hasard, il faut trouver un paramètre qui change d'une fois à l'autre. Utilisons la température, qui est bruitée et variable. La fonction **`randomSeed(valeur);`** agit quelque part dans la fonction random() pour tenir compte du paramètre donné.
Le nouveau programme économise aussi une variable.

```
//Edu31b.ino  Lance un dé
#include "EduC.h"
void setup () { SetupEduC(); randomSeed(GetPotG()); }

void WaitPousG () {
  while (PousG) {DelMs(20);}
  while (!PousG) {DelMs(20);}
}

void loop() {
  WaitPousG();
  Digit (random (1,6+1));
}
```

Quels sont vos réflexes?

Quand la led blanche s'allume, il faut presser sur PousG. Evidemment, on ne va pas utiliser un délai fixe, il faut un délai aléatoire pour ne pas anticiper l'allumage.

Le plus simple. On allume après un temps aléatoire et on mesure la durée jusqu'à ce que l'on presse sur le poussoir.

```
//Edu32a.ino  Réflexes
#include "EduC.h"
#include "Oled.h"
void setup () {
  SetupEduC();  SetupOled();
  LiCol(0,0); Sprite (smile); LiCol(0,100); Sprite (sad);
   LiCol(2,40); BigText("Reflexe");
}

int cnt;  // compte les centièmes de seconde
void loop() {
  DelMs(random(300,2000));
  BlancOn; cnt=0;
  while (!PousG) {DelMs(10); cnt++;}
  BlancOff;
  LiCol(5,20); BigDec9999(cnt);
}
```

On voit que ce programme a un problème. Si on tombe sur deux délais courts qui se suivent, on n'a pas le temps de relâcher le bouton. Si on le maintien pressé, le temps est toujours zéro.
Un remède et d'augmenter la valeur 300. Un autre, meilleur, est d'attendre que l'on relâche.
Si on veut faire un programme bien fait, il faut, avec la petite expérience acquise avec se programme "naif" réfléchir à tout ce qui doit se passer, et l'exprimer dans un langage qui est proche de notre façon de penser, un "pseudo-langage" qui tient un peu compte de comment on pourra programmer en C.
Imaginons le nouveau programme comme suit:

Grande boucle
- On attend 3 secondes
- on flash la led pour annoncer un nouveau test
- on attend au hasard 0.3 à 2 secondes et on allume la led
- on teste si le poussoir est activé.
  - Si non, continuer
  - Si oui, annuler ou pénaliser (break, goto)
- nouvelle boucle
  - on attend 10ms, on compte,
  - on teste si le poussoir est activé
    - si non, on recommence la boucle
    - si oui, on quitte la boucle
- On éteint la led
- On affiche la durée

```
//Edu33aReflexe2.ino  Réflexes
#include "EduC.h"
#include "Oled.h"
void setup () {
  SetupEduC();  SetupOled();
  LiCol(0,0); Sprite (smile); LiCol(0,100); Sprite (sad);
  LiCol(2,40); BigText("Reflexe");
}

int cnt;  // compte les centièmes de seconde
void loop() {
  DelMs(3000); VertOn; DelMs(100); VertOff;
  DelMs(random(300,2000));
  BlancOn; cnt=0;
  while (!PousG) {DelMs(10); cnt++;}
  BlancOff;
  LiCol(5,20); BigDec9999(cnt);
}
```

Ce programme ne teste pas si on a pressé avant que Blanc s'allume. Il y a 2 cas à prévoir: on presse et relâche avant le blanc. On presse avant et relâche après.

Tiens, on pourrait afficher le meilleur temps et avec le pire – un nouveau défi.

Potentiomètre

Un potentiomètre fournit une tension variable, dite analogique, que le microcontrôleur traduit en une valeur 8 bits (Arduino utilise la fonction analogRead(); et rend une valeur 10 bits).
Pour lire et afficher la valeur du potentiomètre gauche, il faut en général définir une variable pour transporter le résultat.

```
byte potG;
potg=GetPotG();
Hex8(potG);  // ou Dec8(potG);
```

Note: dans d'anciens programme, on voit AnaG au lieu de PotG. Il faut modifier pour la nouvelle notation.

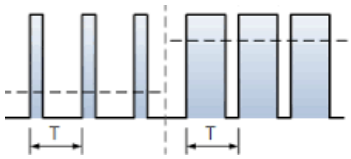
On a dit que `GetPotG()` est comme une variable. On peut en effet écrire `Hex8(GetPotG());` sans définir une variable intermédiaire.
Pour le test, on affiche sur le Oled.

```
//Edu34aPots.ino Lecture entrées analogiques
#include "EduC.h"
#include "EduC.h"
void setup(){SetupEduC(); SetupOled();}

void loop() {
   LiCol(0,0); Sprite (smile); Text("  1 Pots et temp");
   while(1) {
       LiCol(3,25); BigDec8 (GetAnaG());
       LiCol(3,65); BigDec8 (GetAnaD());
       DelMs (20);  // Inutile de lire trop souvent
      }
}
```

PWM (analogWrite) pour une intensité variable

La sortie d'un microcontrôleur est nécessairement du tout ou rien.
Pour donner moins d'énergie à une Led ou à un moteur, on pulse au moins 50 fois par seconde pour que l'œil ne voie pas le clignotement. La largeur de l'impulsion définit l'intensité.



C'est ce que l'on appelle le **PWM** (Pulse Width Wodulation) et pour les leds de la carte EduC, on a défini les fonctions **`LedG(v5); LedD(v5); Rouge(v5); Vert(v5); Bleu(v5);`**

Pourquoi v5? Pour de bonnes raisons, seulement 32 niveaux d'intensité ont été programmés, et ils correspondent à la sensibilité de l'œil, c'est-à-dire la valeur 16, la moitié de 32, donne une impression de demi intensité.

Un a vu avec le premier programme Edu01a. que l'oeil perçoit très bien des impulsions même très courtes. Avec du 50% de PWM sur une Led, on a déjà l'impression d'avoir le maximum. C'est facile à vérifier en changeant les paramètres de ce programme Edu01a.

Les valeurs permises sont donc de 0 à 31 et si on continue on recommence à zéro.

Si on a une valeur 8 bits pour l'intensité (0-255), il suffit de la diviser par 8 et on a une valeur de 0 à 31.

```
//Edu35aPWM.ino test le plus simple
#include "EduC.h"
void setup () { SetupEduC(); }

void loop() {
    LedG (GetPotG()/8);    // max 31
    LedD (GetPotD()/8);
}
```

En secouant le Edu-C au bout de son câble, on voit les traînees lumineuses du PWM et aussi celles de l'affichage, mais pour une autre raison: l'affichage est balayé par sa circuiterie interne.

Mélange de couleurs

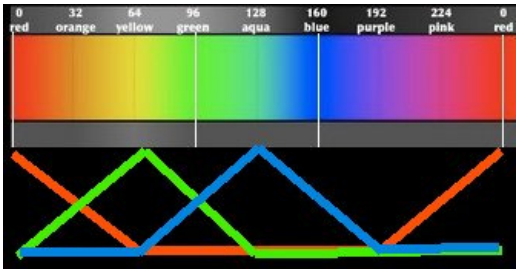
Varions l'intensité des Leds. Il faudrait un 3e potentiomètre pour pouvoir mélanger les couleurs RGB.

```
//Edu36aCoul.ino Mélange de couleurs
#include "EduC.h"
#include "Oled.h"
void setup () { SetupEduC(); SetupOled(); }

byte potG,potD;
void loop() {
    potG = GetPotG(); potD = GetPotD();
    LiCol(5,25); BigDec8 (potG);
    LiCol(5,65); BigDec8 (potD);
    LiCol(7,25); BigDec8 (potG/8);
    LiCol(7,65); BigDec8 (potD/8);
    LedG (potG/8); LedD (potD/8);
    Rouge (potG/8); Vert (potD/8);
    DelMs (20);
}
```

Note: L'affichage en "Big" prend beaucoup de temps, le DelMs(20); est inutile.

Defi – programmer au mieux les couleurs de l'arc-en-ciel



jdn 171210/180118