Jeu Simon: analyse

1 Algorithme du jeu :

Effectuer un tirage de la séquence des couleurs (20 éléments, 3 couleurs)

Répéter

> Afficher la séquence des couleurs et émission des notes associées
>
> Répéter
>
> > Entrée de chaque élément de la séquence par l'opérateur en temps limité
>
> jusqu'à (valeurs de la séquence exactes OU une erreur)
>
> Si pas d'erreur :
>
> > Augmentation de la longueur de la séquence.
> >
> > Si longueur de la séquence est de 20 éléments : Messages et arrêt
>
> sinon : Messages et arrêt

2 Constitution du programme :

Initialisations (setup) : Configuration des sorties utilisées pour l'affichage sur des LEDS, activation de l'UART de la sortie série, tirage des valeurs de la séquence et mémorisation dans un tableau.

Affichage et sons (jouesequence) : les valeurs du tableau sont lues une à une et le son associé est émis sur le haut parleur. (La séquence ne comporte qu'un seul élément au départ, elle augmentera d'un élément à chaque tour de boucle.)

Fonction de reproduction de la séquence par le joueur (jeujoueur) : le temps est limité. La limite s'aggrandit au fur et à mesure que la séquence s'allonge. En cas de dépassement de la limite de temps sortie de l'étape. Attente d'une touche, vérification de la validité de l'entrée. En cas d'erreur sortie de l'étape. Si la séquence est entrée sans erreur en entier, sortie de l'étape.

Deux fonctions servent à créer une séquence de quelques notes pour le cas où la partie est gagnée ou celui où elle est perdue.

Gestion du jeu (bilan) : Selon le résultat de la fonction « jeujoueur » soit

la partie est gagnée (on est arrivé à la séquence la plus longue sans erreur) : musique joyeuse.

La partie est perdue, on a eu une erreur ou un dépassement du temps limite : musique triste.

La partie est en cours (pas d'erreur jusqu'à présent) : la séquence s'allonge d'une unité.

3 Eléments du programme :

3.1 Variables globales : utilisables partout dans le programme

```
uint8_t colors[numOfColors];        // mémoire du code
int ColorsIndex;                    // pointeur fin de séquence
unsigned long time;                 // Mémoire de la date
unsigned long attente=3000;         // Durée maximimum de la réponse
```

Le code est contenu dans un tableau de « uint8_t » (unsigned int 8 bits) nommé « colors », de longueur « numOfColors ».

Cette valeur est une constante :

```
const int numOfColors = 20;          // longueur max de séquence
```

longueur de la séquence :

```
int ColorsIndex;                     // pointeur fin de séquence
```

A chaque tour de jeu, il faut noter la date de début du jeu du joueur :

```
unsigned long time;                  // Mémoire de la date
```

La durée maximum est donnée par la variable « attente ». Sa valeur de départ est 3000 (ms).

« Unsigned long » est une variable sur 4 octets (0 à 2^32)

```
unsigned long attente=3000;          // Durée maximimum de la réponse
```

3.2 Valeurs symboliques : un symbole remplace un N° de broche

Définies par des constantes « int » au lieu des « define » qui seraient peutêtre plus logiques mais avec lesquelles j'ai rencontré des erreurs (compilation OK, erreurs d'exécution …).

```
const int bpBleu = 4;
const int bpVert = 5;
const int bpRouge= 6;
const int touche= 2;
const int LedBleue=7;
const int LedVerte=8;
const int LedRouge=9;
const int Son=10;
```

3.3 Setup : void setup() (void = pas de valeur de retour ; () = pas de paramètre

```
pinMode(LedBleue, OUTPUT);           // Configuration de broches en sortie
pinMode(LedVerte, OUTPUT);
pinMode(LedRouge, OUTPUT);
Serial.begin(9600);                  // Initialisation de l'UART
```

Remarque : par défaut les broches sont configurées en entrées

Initialisation de la fonction aléatoire en prenant la tension sur une entrée analogique ouverte :

```
randomSeed(analogRead(0));           // initialisation du Random
```

Remplssage du tableau : boucle « for ...next », indice x, de 0 à numOfColors (constante)

La fonction random(1,4) fournit des entiers de 1 à 3 dans la variable « y », valeur qui est copiée dans la case « x » du tableau « colors »

```
for (int x = 0; x < numOfColors; x++) {
    uint8_t y=random(1,4);
    colors[x]=y;
    //Serial.print(y,DEC);           // Instruction utilisée pour la mise au point
```

Initialisation des variables ColorIndex et attente :

```
ColorsIndex = 1;
attente=3000;
```

3.4 Affichage de la séquence :

Fonction « JoueSequence » pas de paramètre, pas de valeur de retour

```
void joueSequence() {
```

Variable interne « led » qui contiendra le N° de led à allumer

```
uint8_t led;
```

Boucle d'indice x, variant de 0 à ColorsIndex :

```
for (int x = 0; x < ColorsIndex; x++) {
```

Récupère le N° de LED dans le tableau

```
led = colors[x];
```

Instructions utilisées pour la mise au point : // …..

Selon le N°, allumage de la LED et émission de la note :

```
if (led==1) {
        digitalWrite(LedBleue,HIGH);
        tone(Son,140);}
else if (led==2){
        digitalWrite(LedVerte,HIGH);
        tone(Son,120);}
else {  digitalWrite(LedRouge,HIGH);
        tone(Son,100);}
```

Noter le « == » dans le test

Noter que Son est un symbole valant 10 car le HP est connecté à la broche N°10

Durée de la note 0,5s :

```
delay(500);
noTone(Son);
```

Eteindre la LED allumée … on ne sait pas laquelle … on éteint tout pendant 0,1s

ceci est nécessaire pour séparer les éléments de même couleur.

```
digitalWrite(LedBleue,LOW);
digitalWrite(LedVerte,LOW);
digitalWrite(LedRouge,LOW);
Delay(100);
```

On note la date de fin d'affichage = début de réponse du joueur :

```
time=millis();
```

3.5 Réponse du joueur

Fonction « jeuJoueur » qui renvoie un paramètre de format « boolean »

```
boolean         jeuJoueur() {
```

variable interne « key » pour mémoriser le nombre correspondant aux BP actionnés :

```
uint8_t key;
```

variables internes de type booléen pour noter si un BP est actionné

```
boolean bleu,vert,rouge;
```

séquence d'entrée : indice x, de 0 à ColorsIndex (comme pour l'affichage)

```
for (int x=0;x<ColorsIndex;x++) {
```

Séquence « do » lire les BP « while » pas de BP activé ET limite de temps non dépassée.

```
do {
        bleu= (digitalRead(bpBleu)==HIGH);
        vert= (digitalRead(bpVert)==HIGH);
        rouge= (digitalRead(bpRouge)==HIGH);
}
while (!(bleu || vert || rouge) && ((millis() - time)< attente));
```

Remarque sur la syntaxe : la variable booléenne reçoit le résultat d'un test (==).

« ! » est une négation, « || » = OU, « && » = ET, millis() = temps (en ms) entre le début du programme et l'instant présent.

Attente du relâchement du BP : un OU câblé est réalisé entre les différents BP, vers l'entrée « touche » = tant qu'une touche est enfoncée, ne rien faire ({ })

```
while (digitalRead(touche)== HIGH) {};
delay (100);            // Anti-rebond
```

Si la limite de temps a été dépassée : valeur de retour faux et la fonction se termine ici :

```
if ((millis() - time)>=attente) {
        return false;
```

Selon la BP enfoncé, on met son N° dans la variable « key »

```
 if (bleu) key=1;
 else if (vert) key=2;
 else if (rouge) key=3;
```

Si le BP enfoncé ne correspond pas à la valeur du tableau : valeur de retour faux et la fonction se termine ici :

```
if (colors[x]!=key) {
        return false;
```

Si la fonction ne s'est pas terminée avant (pas de temps dépassé ou de touche fausse)alors la valeur de retour est « juste » et on en a fini avec cette fonction.

```
return true;
```

3.6 Gestion du jeu :

Fonction « bilan », pas de valeur de retour, paramètre booléen en entrée (true ou false)

Ce booléen est le résultat de la fonction « jeuJoueur() » ci-dessus.

Si le joueur n'a pas fait de faute , on augmente la longueur de séquence, on augmente aussi la durée maximale de réponse de 0,5s

```
 if (examen){
        ColorsIndex++;
        attente+=500;
```

Si la longueur de la séquence est au maximum (numOfColors), la partie est gagnée, envoi d'un message sur la liaison série, petite musique.

```
if (ColorsIndex==numOfColors+1) {
        Serial.println("BRAVO !");
        gagne();
```

Blocage du programme

```
        while (true) {};
```

Sinon la partie est perdue, envoi d'un message sur la liaison série, petite musique, blocage du programme

```
else {
        Serial.println("PERDU ...sniff ");
        perdu();
        while (true) {};
```

3.7 Programme (loop) :

```
void loop() {
boolean gagne;                          // Variable locale
        joueSequence();                 // Appel de la fonction
        gagne=jeuJoueur();
        bilan(gagne);
        delay(2000);                    // Pause de 2s entre deux phases de jeu
```