

## Compte rendu séance 8 :

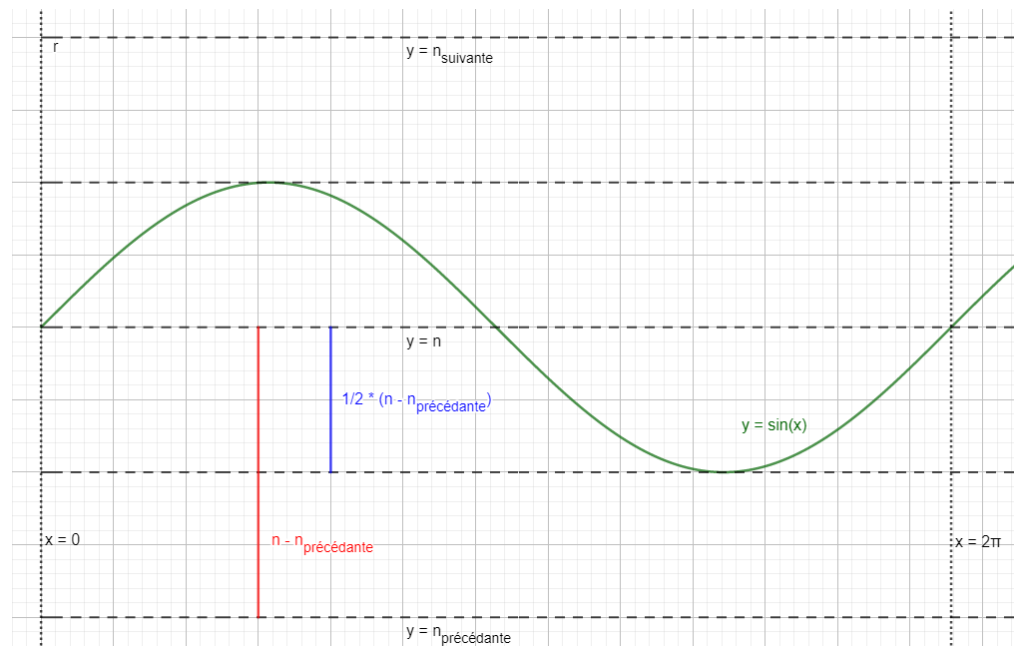
Lors de cette dernière des 8 séances permettant d'avancer le projet, j'ai continué de coder un troisième mode pour le *Variophuino* qui se prénommera le mode *périssando*. Il sera activé quand la variable globale *mode* vaudra 2.

Rappel:  $mode = 0 \rightarrow$  « normal »  
 $mode = 1 \rightarrow$  « arpège »

Le mode *périssando* désigne un effet glissando périodique, c'est-à-dire qu'une note jouée aura une fréquence qui va osciller périodiquement (monter puis descendre...) et de façon continue, du moins en théorie car seules les fréquences entières peuvent être jouées par un buzzer Arduino.

Voici ci-contre un schéma de la représentation théorique de ce que je veux faire.

Le nombre  $n$  représente la fréquence associée à une certaine note, le nom des deux autres est explicite.



Je veux que la courbe verte représente la fréquence de la note jouée en appuyant sur une touche (et en restant appuyé). Pour calculer les écarts, on se servira de la fonction *change\_for\_pitchbend()* déjà codée :

```
void change_for_pitchbend(int n, int s) {
  ...
  if (mode == 1) {special_r_gaps[n] = new_gap;}
  else {
    pitchbend_variations_set[n] = new_gap;
    if (mode == 2) {perissando_gaps_live[n] = new_gap / 2;}
  }
};
```







Ces « demi-écarts » seront stockés dans la liste d'entiers nommée

*perissando\_gaps\_live[]*. On négligera la différence entre l'écart entre une note et celle précédente ET l'écart entre cette même note et celle suivante.

```
void pianos_checking() {
...
while (tl_played || tr_played) {
...
if (correct_left_playing() || correct_right_playing()) {
    if (mode == 1) {arpege_mode();}
    else if (mode == 2) {perissando_mode();}
    else {normal_mode();}
}
...
}
left_tone.stop();
right_tone.stop();
};
```

Comme *pianos\_checking()* est la seule fonction appelée par *loop()*, il est donc facile de tester une valeur (celle d'un potentiomètre par exemple) en ajoutant « // » juste devant. Je précise qu'il faut penser à vérifier si des procédures très importantes sont mises en commentaire ou non avant de décoder ce que l'on pense être une erreur majeure empêchant le bon fonctionnement du *Variophuino*, on met parfois du temps à s'en rendre compte, ce qui empêche d'être productif, alors que c'est parfois juste un « // » en trop... ce qui est, ma foi, fort enquiquinant...

Sans transition, voici les couleurs qui serviront pour les deux leds RGB :

	code_fred_testons	aled.h	fonctions_basiques.h	general_play.h	melodies.h	notes.h	other_left_piano.h	principal_right_piano.h
(l'aspect de chaque couleur a été rajoutée à droite)	//----Couleurs----							
	int bleu_lavande[3] = {145, 145, 255};				// #9191FF			
	int rose_bonbon[3] = {255, 104, 169};				// #FF68A9			
	int golden[3] = {255, 215, 0};				// #FFD700			
	int tomato[3] = {255, 99, 71};				// #FF6347			
	int deep_sky_blue[3] = {0, 191, 255};				// #00BFFF			
	int dark_violet[3] = {148, 0, 211};				// #9400D3			
	//-----							

Les deux premières servent pour savoir si on est en accord *mineur* ou *majeur* grâce à *led\_m*, les trois suivantes pour savoir si on est en mode *normal*, *arpège* ou *périssando* (ou aussi *alarmodique*, mode qui ne sera pas prêt à temps pour la présentation finale du projet) grâce à *led\_b*, et la dernière pour tester une autre *couleur* tout simplement.

Pendant ce créneau de 3h, j'ai commencé à imaginer un texte qui s'afficherait sur l'écran de 64 leds en défilant. J'ai donc dû définir les

listes de 8 nombres (en hexadécimal), chacune étant associée à une lettre, en voici ci-dessous quelques-unes :

```
int A[8] = {0x00, 0x00, 0x18, 0x24, 0x3C, 0x24, 0x24, 0x00}; // largeur: 4
int C[8] = {0x00, 0x00, 0x1C, 0x20, 0x20, 0x20, 0x1C, 0x00}; // largeur: 4
int E[8] = {0x00, 0x00, 0x3C, 0x20, 0x38, 0x20, 0x3C, 0x00}; // largeur: 4
int I[8] = {0x00, 0x00, 0x10, 0x10, 0x10, 0x10, 0x10, 0x00}; // largeur: 1
int J[8] = {0x00, 0x00, 0x3C, 0x08, 0x08, 0x38, 0x10, 0x00}; // largeur: 4
```

Le message qui défilera (si jamais cela est prêt à temps) est :  
« PAR LENA COPPIN ET JULIEN NOEL »

Cela me fait d'ailleurs penser que je dois penser à espacer chaque mot de  $1+4+1=6$  leds, en effet une lettre fait au maximum 4 leds de largeur et chaque led sera espacée de 1 led.

J'en ai aussi profité pour créer les listes correspondant aux chiffres manquants qui sont 0, 1, 8 et 9, même s'ils ne serviront pas.

Je précise que le codage de la fonction *perissando\_mode()* est d'une difficulté inédite pour moi, je dirais que c'est à la fois plus difficile et pas comparable par rapport à *arpege\_mode()*.

Je précise aussi que le codage d'autres chansons pré-enregistrées ne se fera (lui aussi) sûrement pas à temps. C'est juste un bonus, l'important est que tout ce qu'il y a de disponible pour l'utilisateur du *Variophuino* marche correctement.

Une ultime précision:

Le module bluetooth est définitivement branché sur les ports RX1 et TX1 (soient les pins 19 et 18, et non 15 et 14 comme énoncé dans mon rapport précédent) de la carte Arduino Mega et non les ports RX0 et TX0. Ainsi il n'y aura pas de confusion entre une donnée lambda transmise et un téléversement du programme complet.

C'est tout pour moi, merci à mon binôme Léna pour son travail manuel impressionnant et son imagination débordante et merci à Pascal Masson et ses collègues d'avoir permis une telle expérience, ce fut un plaisir !

