Chapitre VI : La librairie standard Arduino

I. Introduction

Cette librairie indispensable contient beaucoup de bonnes choses. En effet, le langage C nu ne permet pas d'être très productif. Nous en avons déjà utilisé un certain nombre, comme pinMode(), digitalWrite() ou analogWrite(). Mais il y a bien d'autres fonctions standard. Vous pouvez ainsi répondre à des besoins dans le domaine des mathématiques, de la génération de nombres aléatoires, la gestion d'interruptions...etc.

II. Nombres aléatoires (random)

Imaginez que vous vouliez simuler le lancer d'un dé, votre carte Arduino devant générer au hasard un nombre entre 1 et 6. La librairie standard Arduino possède une fonction destinée à cet usage : la fonction random(). Elle renvoie une valeur entière int et accepte en entrée un ou deux paramètres. Lorsque vous n'en fournissez qu'un, elle renvoie une valeur au hasard entre 0 et la valeur du paramètre moins 1.

Si vous fournissez deux paramètres, la fonction génère une valeur au hasard entre la valeur du premier paramètre (y compris) et le second paramètre moins 1. Autrement dit, random(1, 10) renvoie un nombre au hasard entre 1 et 9.

Le programme suivant affiche dans le Moniteur série sans relâche des nombres entre 1 et 6.

```
void setup()
{
Serial.begin(9600);
}
void loop()
{
int nombre = random(1, 7);
Serial.println(nombre);
delay(500);
}
```

Si vous transférez ce programme sur votre carte Arduino, vous devriez voir apparaître dans le Moniteur série les chiffres que présente la Figure 1.

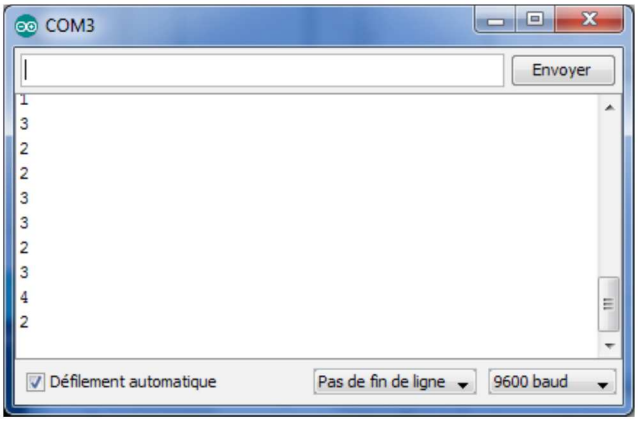

Fig. 1 Exemple de génération de nombres pseudo-aléatoires.

III. Fonctions mathématiques

Vous aurez en quelques rares occasions besoin de réaliser un peu de mathématiques sur la carte Arduino, au-delà de l'arithmétique de base. Vous disposez à cet effet d'une librairie de fonctions mathématiques complètes. Le tableau suivant présente une sélection des fonctions les plus utiles dans cette catégorie :

Fonction	*Description*	*Exemple*
`abs()`	Renvoie la valeur absolue de son argument.	`abs(12)` renvoie 12
		`abs(-12)` renvoie 12
`constrain()`	Interdit à un nombre de déborder d'une plage spécifiée. Le premier argument est le nombre à limiter, le second est le début de la plage et le troisième la fin de la plage.	`constrain(8, 1, 10)` renvoie 8
		`constrain(11, 1, 10)` renvoie 10
		`constrain(0, 1, 10)` renvoie 1
`map()`	Convertit un nombre d'une échelle vers une autre. Le premier argument est le nombre à convertir, le deuxième et le troisième constituent la plage source. Les deux derniers arguments constituent la plage cible. Cette fonction est très utile pour convertir des valeurs d'entrées analogiques.	`map(x, 0, 1023, 0, 5000)`
`max()`	Renvoie le plus grand de ses deux arguments.	`max(10, 11)` renvoie 11
`min()`	Renvoie le plus petit de ses deux arguments.	`min(10, 11)` renvoie 10

VI. Interruptions

Vous pouvez en effet gérer des interruptions sur deux des broches de la carte Arduino (D2 et D3). Ces deux broches fonctionnent en tant qu'entrées de telle manière que lorsque l'une des deux reçoit un signal dans un certain format, cela provoque la suspension de l'action que réalise le processeur Arduino à ce moment suivie du branchement sur une fonction qui a été associée au préalable à cette interruption.

Le programme suivant fait clignoter une LED, mais la fréquence du clignotement change lorsqu'une interruption survient. Pour simuler cette interruption, il suffit de connecter un fil entre la broche D2 et la masse GND, en activant la résistance de rappel interne pour que l'interruption reste bien à l'état haut.

```
int brocheINTER = 2;
int brocheLED = 13;
int periode = 500;
void setup()
{
pinMode(brocheLED, OUTPUT);
pinMode(brocheINTER, INPUT);
digitalWrite(brocheINTER, HIGH); // Résistance de rappel
attachInterrupt(0, accelerer, FALLING);
}
void loop()
{
digitalWrite(brocheLED, HIGH);
delay(periode);
digitalWrite(brocheLED, LOW);
delay(periode);
}
void accelerer()
{
periode = 100;
}
```

L'instruction capitale de ce programme se trouve dans la fonction setup() :

```
attachInterrupt(0, accelerer, FALLING);
```

Le premier argument de la fonction sert à définir laquelle des deux interruptions vous voulez exploiter : la

valeur 0 signifie la broche 2 et la valeur 1 la broche 3.

Le deuxième argument doit spécifier le nom de la fonction qui doit être appelée en cas d'interruption. Le dernier argument est une constante qui peut posséder l'une des trois valeurs CHANGE, RISING ou FALLING. La Figure III.4 présente la différence entre ces trois valeurs.

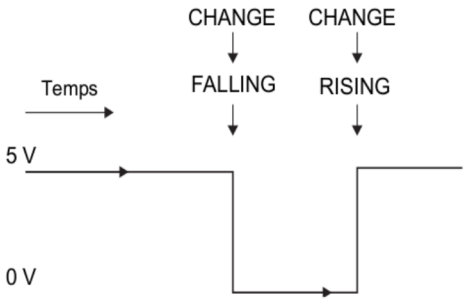

Figure III.4 Front montant et descendant d'un signal d'interruption.

Si vous choisissez le mode d'interruption CHANGE, aussi bien un front montant RISING (entre 0 et 1) qu'un front descendant FALLING (entre 1 et 0) provoquent le déclenchement de l'interruption.

Pour inhiber les interruptions, vous appelez la fonction noInterrupts() qui invalide toutes le interruptions des deux canaux. Pour relancer les interruptions, vous appelez la fonction interrupts().

IV. Conclusion

Ce chapitre vous a permis de découvrir quelques fonctions spécifiques offertes par la librairie de fonctions standard Arduino. Ces capacités vous feront gagner du temps de programmation.