Servomoteurs et Interruptions

Le problème

Faire de la musique en évitant des obstacles

Commençons par les interruptions. Les interruptions sont typiquement utilisées pour faire du multi-tâche. Il ne permet pas de réduire le temps de calcul des tâches ou de faire plusieurs calculs en même temps, mais plutôt de les faire au bon moment.

Les interruptions sont des modules hardware (c'est à dire des circuits électriques dans le chip) qui sont capables d'interrompre le programme principal pour faire une autre fonction. A noter qu'il n'est pas conseillé de lancer une interruption dans une interruption (ce qui n'est pas possible avec la librairie robopoly).

Ce que le MCU fait d'habitude

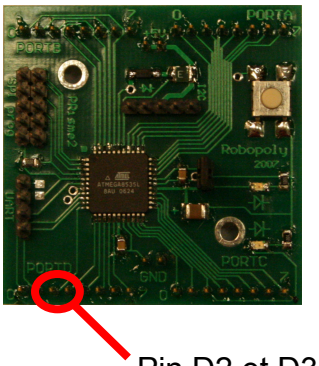
```
int main(void)
{
    init_programme();

    while(1)
    {
        if(detecte_obstacle()) // dure moins de 1ms
        {
            eviter_obstacle();
        }
        jouer_musique(); // dure plus de 10 s
    }
}
```

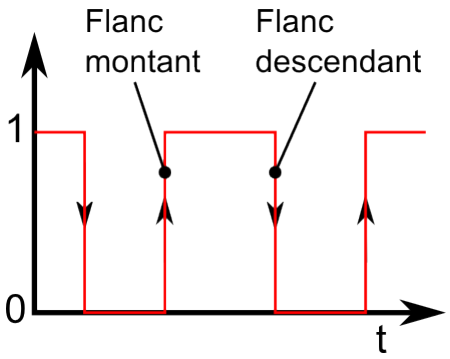
Si le robot doit faire deux tâches simultanément (ici jouer la musique de epic sax guy et s'arrêter devant l'obstacle) et que l'une des deux fonctions est longue à être executée (la musique de epic sax guy est typiquement de 10 secondes), alors le robot va manquer de réactivité.

En effet, le robot doit d'abord finir de jouer la musique pour ensuite regarder si il y a un obstacle devant lui. Le résultat est qu'il ne prend une décision que toutes les 10 secondes, ce qui est bien assez pour foncer dans l'obstacle.

Il est possible de raccourcir ce temps en fractionnant la fonction musique (jouer note à note au lieu de tout le morceau par exemple) mais il est possible d'utliliser une chose bien plus efficace : les interruptions externes.

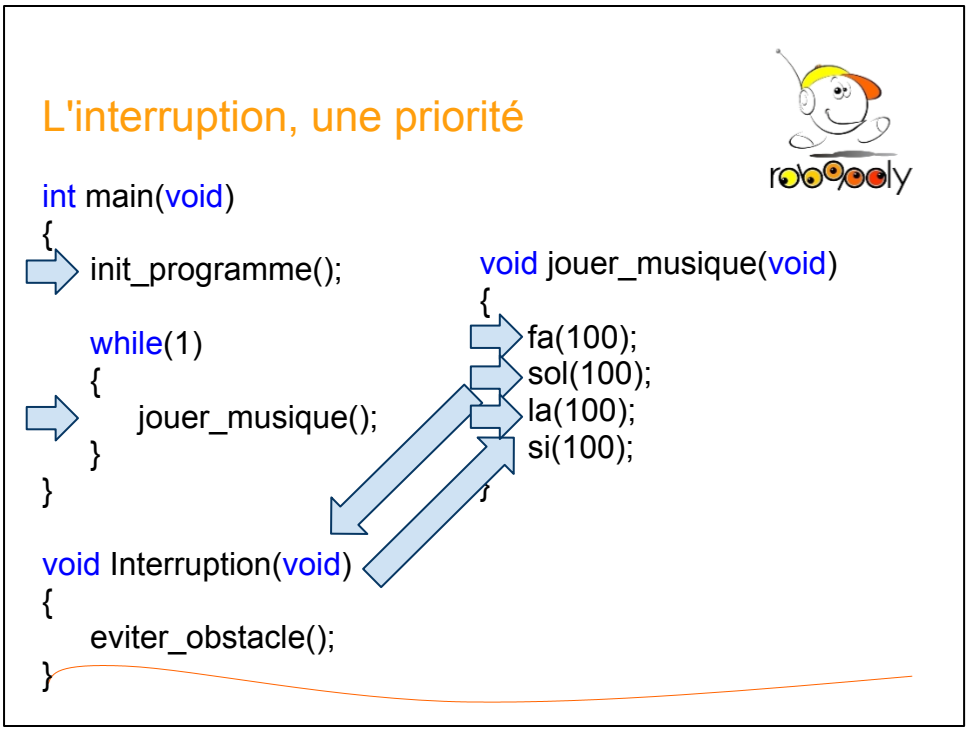
Les interruptions externes





Les interruptions externes sont des modules connectés sur les deux pins D2 et D3 et qui interrompent le programme dès qu'ils voyent un changement de flanc. Ces interruptions peuvent être configurés pour se déclencher sur le flanc montant (quand le pin passe de l'état logique 0 à 1 *), descendant (de 1 à 0) ou les deux.

*Pour être précis, ceci se produit quand la tension sur le pin passe d'en dessous de la moitié de la tension d'alimentation à au-dessus. Pour le flanc descendant, c'est l'inverse.



Reprenons notre exemple. Dans notre cas, seule la fonction jouer_musique() est laissée dans la boucle principale, la fonction d'évitement d'obstacle étant placée dans la fonction d'interruption. L'interruption est initialisée dans la fonction init_programme().

Le programme joue en boucle la musique, jusqu'à ce qu'un obstacle se présente devant le robot. A cet instant, le robot arrête le programme principal (jouer de la musique) et entre dans l'interruption et fait son évitement d'obstacle. Une fois cet évitement d'obstacle fait, le robot reprend le programme principal là ou il l'avait laissé.

Programmation interruptions (1)

#include "int_ext.h" au début du programme

Ne pas oublier : inclure int_ext.h et int_ext.c dans le dossier du projet !

L'emploie des interruptions nécessite l'ajout au projet des fichiers int_ext.h et int_ext.c. De plus, il faut inclure, au début du programme int_ext.h à l'aide de la ligne de code suivante : #include "int_ext.h".

Programmation interruptions (2)

void configure_INT0(état, fonction); (PIN2)

ou

void configure_INT1(état, fonction); (PIN3)

Les différents modes :

RISING_EDGE
FALLING_EDGE
ANY_CHANGE
OFF

Il existe une fonction pour chaque interruption externe (pin D2 ou D3). La variable état définit la manière d'enclenchement de l'interruption :

flanc montant (**RISNING_EDGE**),

flanc descendant (**FALLING_EDGE**),

les deux **(ANY_CHANGE**),

ou permet de désactiver l'interruption (**OFF**).

La fonction qui est appelée doit être déclarée dans le programme.

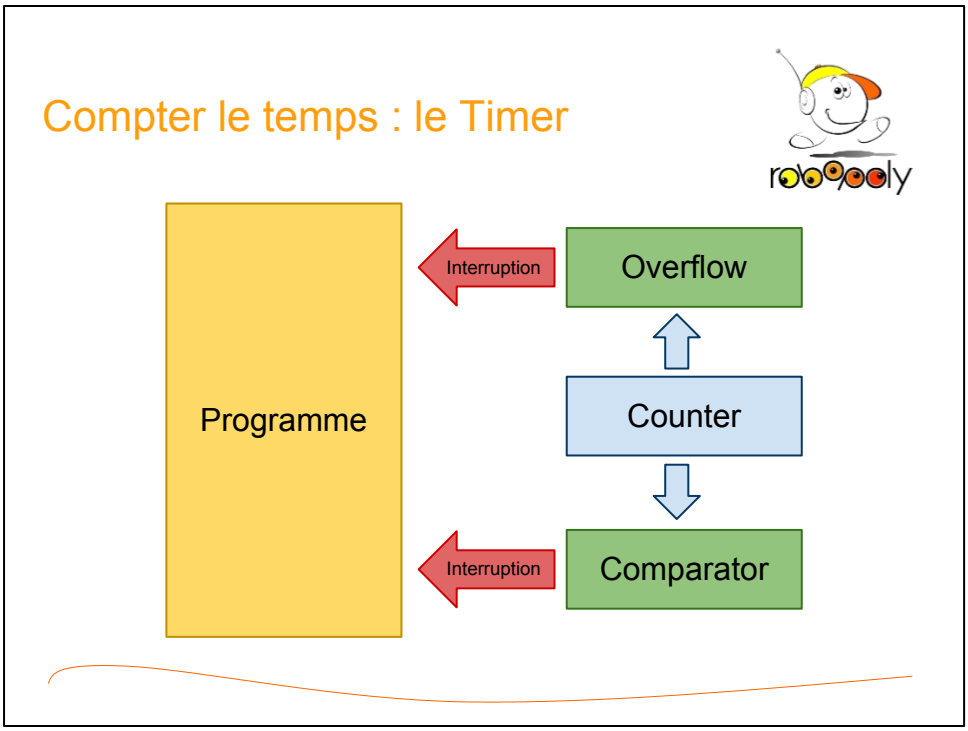
Exemple


```
#include "robopoly.h"
#include "int_ext.h"

int main(void)
{

	configure_INT0(FALLING_EDGE, Interruption);

	while(1)
	{
		jouer_musique();
	}
}

void Interruption(void)
{
	eviter_obstacle();
}
```

Exemple de code

Une autre interruption : Les Timers

Il existe aussi d'autres types d'interruptions en deorhs des interruptions externes. Il y a par exemple les timers, c'est-à-dire des modules qui comptent le temps.



Les timers sont composés d'un compteur qui s'incrémente à interval de temps régulier. Ce compteur est surveillé par des modules qui déclenchent des interruptions selon la valeur de ce premier. Il y a par exemple le comparateur qui déclenche une interruption lorsque que le compteur atteint une valeur prédefinie, ou l'overflow qui fait une interruption quand le le compteur dépasse sa valeur maximale (255 ou 65'535 selon les cas).

Il existe beacoups de configurations différentes pour ces timers et il est nécessaire de se référer à la datasheet pour les utiliser tel quel. La librairie robopoly contient des fonctions toute faites pour les utiliser plus facilement.

Programmation Timer (1)

addNewCallback (fonction, durée, nombre_execution) ;

L'utlilisation des timers est faite par cette fonction-ci. Elle permet de configurer l'appel d'une fonction à intervalles réguliers. De plus, un nombre limite de fois ou celle-ci est appelée peut être configuré. Jusqu'à 8 fonctions peuvent être mises dans la mémoire de l'agenda.
Les arguments sont les suivants :

fonction : fonction à appeler, ne doit retourner aucune valeur et ne prendre aucun argument
durée : temps entre chaque appel (en multiple de 2 ms). Le temps = durée*2 ms
nombre_execution : nombre total d'execution (entre 0 et 255). 0 signifie que la fonction est apppelée sans nombre limite de fois

Le temps absolu maximum est de 8589934s à savoir environ 100 jours et l'intervalle maximum du temps entre chaque appel est de 131s.

Lorsqu'une fonction a été appelée le nombre maximum de fois défini, elle est automatiquement retirée de la liste de l'agenda et on peut placer une autre fonction à sa place en utilisant la fonction addNewCallback à nouveau.

Programmation Timer (2)

La fonction addNewCallback retourne le numéro de la fonction.

stopCallback(num_callback) ;

Cette fonction force l'arrêt d'une fonction de l'agenda

La fonction addNewCallback renvoie le numéro de la fonction callback, il peut être utiliser pour arrêter l'appel de la fonction à l'aide de la fonction stopCallback. Cette fonction ne prend qu'un argument : le numéro de la fonction callback.

Exemple d'utilisation :

```
int main (void)
{       ...
        num_callback = addNewCallback(fonction,500,0);
        ...
        stopCallback(num_callback);
        ...
}
```

Exemple

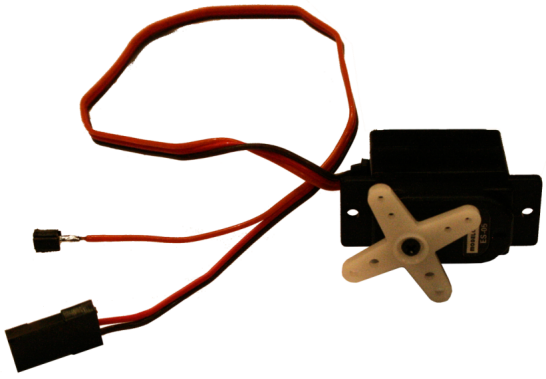

```
int main(void)
{

	addNewCallback(tourner_robot, 500, 0);

	while(1)
	{
		jouer_musique();
	}
}


void tourner_robot(void)

{
	setupMotorPWM(-50,50);
}
```

Exemple de code

Compte à rebours

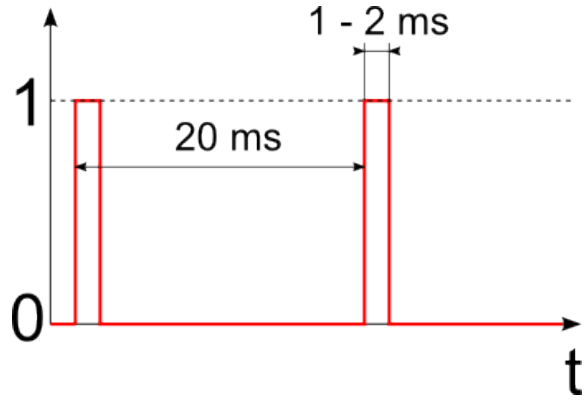
On peut utiliser les timers pour faire un compte à rebours.

Les servomoteurs



Le servomoteur est un moteur qui se contrôle en position. On lui donne un angle auquel aller et il se débrouille tout seul pour y aller, le rendant très simple à commander.

Sa commande ; Un PWM spécial

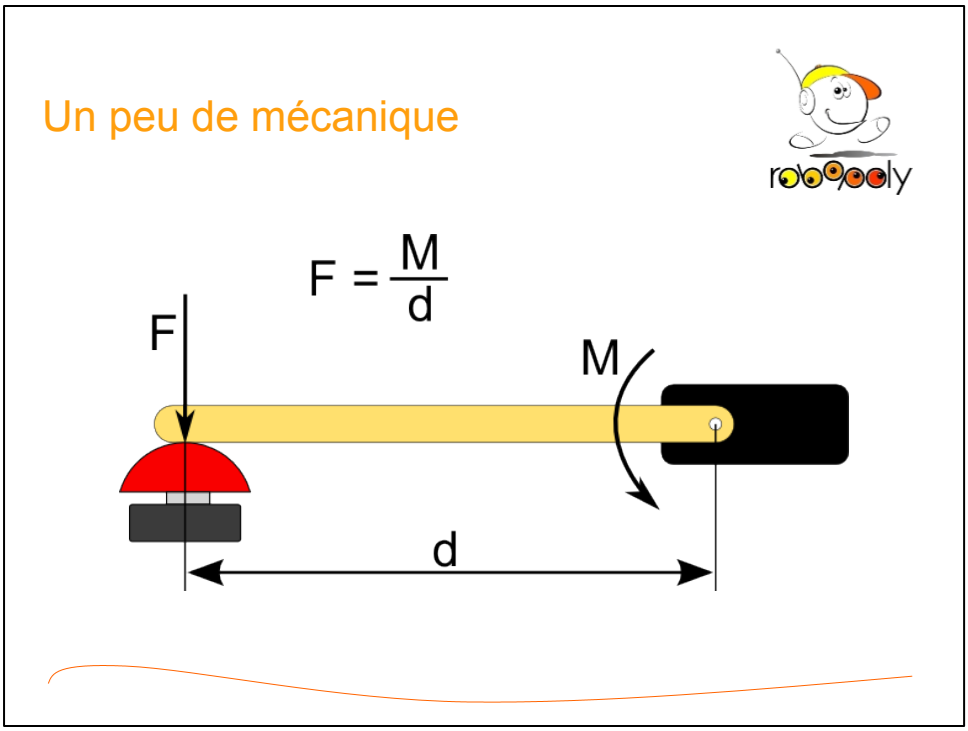


Pour commander le servomoteur, il faut lui envoyer une impulsion de 1 à 2 ms toutes les 20 ms environ. C'est la largeur (durée) de cette impulsion qui détermine l'angle. Une impulsion de 1 ms le fait aller à un angle de 0° * et une impulsion de 2 ms le fait aller à 180°.

Le moteur à une vitesse maximale, donc il ne faut pas changer sa position trop rapidement.

Il est a noter que la grande majorité les servomoteurs se commandent de cette façon, il est donc tout à fait possible de mettre un autre servomoteur sur le PRisme, sa commande sera toujours possible avec la librairie robopoly.

*Cet angle dépend de comment le bras est placé.



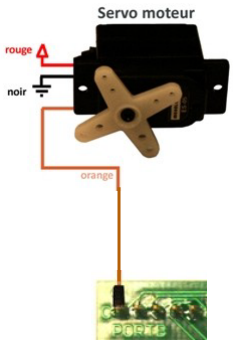
Le couple du servomoteur est constant. Il ne faut donc pas faire le bras trop long sous peine ne n'avoir plus assez de force pour effectuer la manipulation.

Le couple maximal est donné sur la boîte des servomoteurs.

Branchements

A brancher sur les ports :
B0 à 4 et C3 à 7.

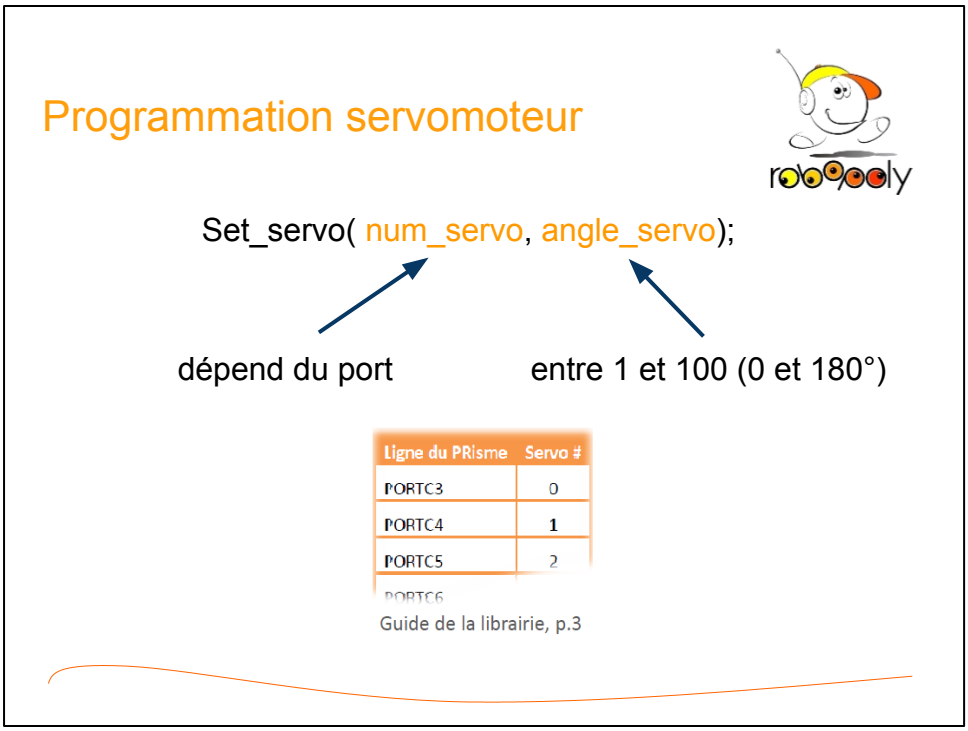
Alimentation



On peut brancher jusqu'à 10 servomoteurs sur le PRisme.

Contrôle

Le branchement du servomoteur est simple. Il suffit de couper le fil de contrôle (en orange) et de le connecter au PRisme. Cependant, il faut le mettre sur un pin spécifique B0 à4 et C3 à 7. Cela permet de brancher jusqu'à 10 servomoteurs sur le PRisme.

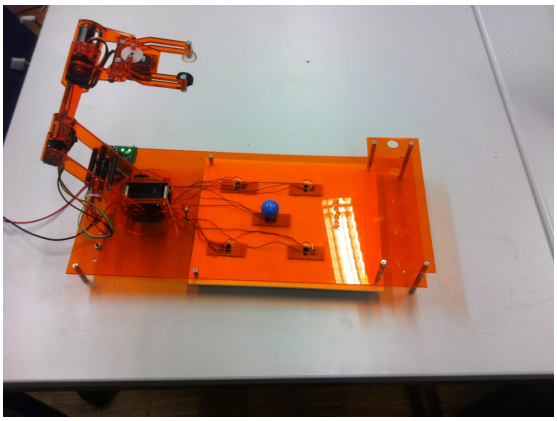


La commande du servo se fait à l'aide de la fonction Set_servo qui prend comme deux arguments :

num_servo : un numéro qui dépend du pin sur lequel il est branché. Le tableau est disponible dans le guide de la librairie en p.3

angle_servo : une valeur comprise entre 1 et 100, qui correspont à un angle entre 0 et 180°

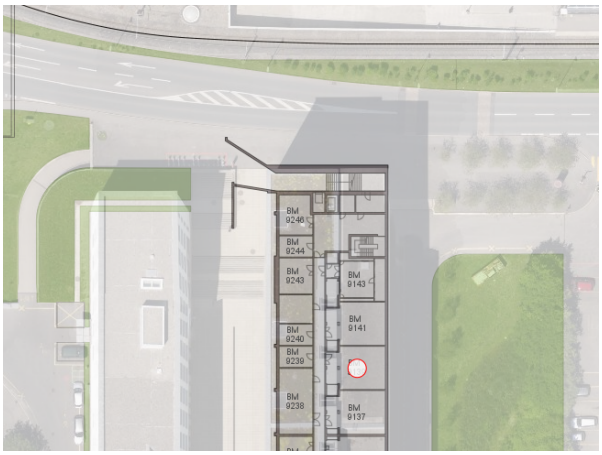
Le bras robotique



Une partie de la documentation et une vidéo se trouve sur [le wiki de Robopoly]

Déménagement

Salle : BM9139



La nouvelle salle de Robopoly se situe en BM9139.

Robopoly social

http://robopoly.epfl.ch

@RobopolyEPFL

Robopoly

robopoly@epfl.ch

RobopolyEPFL