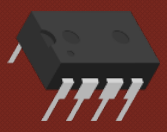


www.sylvainmahe.xyz

LE BLOG
de Sylvain Mahé
contact@sylvainmahe.xyz

Retour
Suite

Lire un récepteur R/C avec PwmRead.h

L'une des utilisations possibles de la classe **PwmRead.h** est la lecture des sorties PWM d'un récepteur de modélisme R/C standard afin de piloter un modèle R/C. Bien d'autres applications sont possibles, mais c'est celle-ci qui fera l'objet d'un exemple ici.

La **modulation de la largeur d'impulsion** (PWM) est un moyen pratique de communication entre deux (ou plusieurs) périphériques, ce principe ne demande qu'un seul fil de liaison.

> ***PwmRead.h*** *permet de mesurer la largeur d'impulsion PWM avec une précision d'**1 microseconde**.*

Exemple d'utilisation de PwmRead.h:

```
#include "../module/1284p/PwmRead.h"

int main()
{
	PwmRead channelThrottle = PwmRead (1, true);
	PwmRead channelPitch = PwmRead (2, true);
	PwmRead channelRoll = PwmRead (3, true);
	PwmRead channelYaw = PwmRead (4, true);
	PwmRead channelCut = PwmRead (5, true);

	PwmRead::start (100);

	while (true)
	{
		channelThrottle.read();
		channelPitch.read();
		channelRoll.read();
		channelYaw.read();
		channelCut.read();

		//si l'interrupteur de coupure des gaz est enclenché:
		if (channelCut.us > 1500)
		{
			//coupure des gaz
		}
		else
		{
			//sinon, utilisation des valeurs pour piloter le modèle R/C:
			//channelThrottle.us (gaz)
			//channelPitch.us (tangage)
			//channelRoll.us (roulis)
			//channelYaw.us (lacet)
		}
	}

	return 0;
}
```

Cet exemple montre un peu comment organiser un programme pour pouvoir lire les voies PWM d'un récepteur de modélisme standard afin de piloter un modèle radio-commandé.

Au début du code, 5 objets de type **PwmRead** sont déclarés:
- Le 1er paramètre indique le numéro du port de l'automate programmable en entrée qui va servir à lire le signal PWM, respectivement les ports numéro **1**, **2**, **3**, **4**, et **5** dans l'exemple.

> *Avec la classe PwmRead.h, **tous les ports de l'automate programmable** peuvent servir à la lecture de signaux PWM.*

- Le 2ème paramètre **true** permet de définir que la lecture du PWM débute par un **front montant**, et termine par un front descendant (le cas le plus courant), ou inversement si le paramètre était indiqué à **false**.

> *Ce paramètre n'a pas d'utilité dans le cadre de la lecture de sorties PWM type récepteurs de modélisme standards, ou l'on s'attends toujours à un front montant lorsque le signal débute. En revanche, cette fonctionnalité peut être utile voir indispensable pour d'autres applications.*

Ensuite, la lecture PWM est démarrée en appelant la fonction statique **start** prenant en paramètre **100**, un temps arbitraire (dépendant de la fréquence du PWM) indiqué en milliseconde, relatif à la sécurité de votre montage si besoin (voir plus loin).

À la suite du programme dans la boucle, les 5 objets de type **PwmRead** appellent leurs fonctions **read** respectives, ce qui permet de mettre à jour les variables **us** (largeur d'impulsion en microsecondes) relatives aux objets déclarés, et de s'en servir pour piloter le modèle R/C.

Une sécurité dans le cas d'un récepteur R/C partiellement défaillant:
La lecture PWM s'effectuant de manière **séquentielle dans l'ordre de déclaration des objets**, imaginez le cas extrême d'une défaillance de l'une des voies de votre récepteur R/C, cas ou la lecture de la voie deviendrait impossible. Il serait alors judicieux de considérer que si au bout d'un certain temps, le signal PWM ne peut être mesuré, la lecture passe à l'objet suivant (à la voie suivante) sans mettre à jour la variable **us** relative à l'objet considéré. C'est exactement ce que permet ce paramètre, dans l'exemple:

> ***100 millisecondes*** *sera le temps au bout duquel on considère que la lecture du PWM a échouée pour cause d'une défaillance en amont.*

Ce paramètre peut être indiqué à **0**, mais si vous débranchez un fil de liaison de votre récepteur R/C, par exemple la voie branchée sur l'objet **channelPitch**, dans ce cas toutes les autres voies ne seront plus mises à jour dans la boucle, alors que cela n'aurait pas été le cas si vous aviez indiqué un paramètre supérieur à 0.

Pour choisir ce temps, il vous faut considérer la fréquence de votre PWM. Si votre récepteur R/C produit un PWM à une fréquence de **50Hz**, une valeur de sécurité indiquée en paramètre d'au moins **40ms** (2 fois la période de 50Hz) semble cohérente.

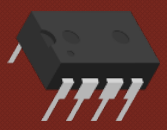
Ports des automates programmables concernés par la lecture du PWM:

Automate programmable MODULABLE M20:
- Port 1 (PD0)
- Port 2 (PD1)
- Port 3 (PD2)
- Port 4 (PD3)
- Port 5 (PD4)
- Port 6 (PD5)
- Port 7 (PD6)
- Port 8 (PD7)
- Port 9 (PB0)
- Port 10 (PB1)
- Port 11 (PB2)
- Port 12 (PB3)
- Port 13 (PB4)
- Port 14 (PB5)
- Port 15 (PC0)
- Port 16 (PC1)
- Port 17 (PC2)
- Port 18 (PC3)
- Port 19 (PC4)
- Port 20 (PC5)

Automate programmable MODULABLE M32:
- Port 1 (PB0)
- Port 2 (PB1)
- Port 3 (PB2)
- Port 4 (PB3)
- Port 5 (PB4)
- Port 6 (PB5)
- Port 7 (PB6)
- Port 8 (PB7)
- Port 9 (PD0)
- Port 10 (PD1)
- Port 11 (PD2)
- Port 12 (PD3)
- Port 13 (PD4)
- Port 14 (PD5)
- Port 15 (PD6)
- Port 16 (PD7)
- Port 17 (PC0)
- Port 18 (PC1)
- Port 19 (PC2)
- Port 20 (PC3)
- Port 21 (PC4)
- Port 22 (PC5)
- Port 23 (PC6)
- Port 24 (PC7)
- Port 25 (PA7)
- Port 26 (PA6)
- Port 27 (PA5)
- Port 28 (PA4)
- Port 29 (PA3)
- Port 30 (PA2)
- Port 31 (PA1)
- Port 32 (PA0)

Récapitulatif des fonctions et variables de cette classe:

```
unsigned long us = 0;
PwmRead (const unsigned char PIN, const bool RISING);
static void start (const unsigned int TIME_OUT);
void read();
```



www.sylvainmahe.xyz

LE BLOG
de Sylvain Mahé
contact@sylvainmahe.xyz

Retour
Suite