

www.sylvainmahe.xyz

LE BLOG

de Sylvain Mahé

contact@sylvainmahe.xyz



Article: Sylvain Mahé

contact@sylvainmahe.xyz

[Retour](#)

[Suite](#)

Lire un potentiomètre avec AnalogRead.h

La classe **AnalogRead.h** permet de lire la **tension analogique** (0V à +5V) des GPIO connectées au convertisseur analogique/numérique du microcontrôleur.

Cette tension analogique de 0V à +5V est convertie en binaire avec une précision sur 10 bits (de 0 à 1023 en base 10).

Note importante concernant l'impédance de votre montage:

*Le circuit de conversion analogique/numérique du microcontrôleur est optimisé pour fonctionner avec des impédances en entrée de **10kΩ**, et peut fonctionner correctement de **1kΩ** à **100kΩ**.*

Exemple d'utilisation de AnalogRead.h:

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

int main()
{
    AnalogRead myPotentiometer = AnalogRead (25);

    while (true)
    {
        myPotentiometer.read();

        //myPotentiometer.value est la valeur lue:
        if (myPotentiometer.value > 512)
        {
            //effectuer une action si la tension lue est supérieure à +2.5V
        }
    }

    return 0;
}
```

Dans cet exemple, un objet **myPotentiometer** de type **AnalogRead** est déclaré, en paramètre est indiqué d'utiliser le port numéro **25** de l'automate programmable en entrée, puis cet objet **myPotentiometer** appelle la fonction **read** ce qui permet de lire l'entrée analogique concernée, et donc de mettre à jour la variable **myPotentiometer.value**.

Puis aux lignes suivantes, si cette variable est supérieure à **512** (+2.5V), le déroulement du programme rentre dans la condition logique.

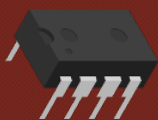
Ports des automates programmables concernés par l'analogique:

Automate programmable MODULABLE M20:

- Port 15 (PC0)
- Port 16 (PC1)
- Port 17 (PC2)
- Port 18 (PC3)
- Port 19 (PC4)
- Port 20 (PC5)

Automate programmable MODULABLE M32:

- Port 25 (PA7)
- Port 26 (PA6)
- Port 27 (PA5)
- Port 28 (PA4)
- Port 29 (PA3)
- Port 30 (PA2)
- Port 31 (PA1)
- Port 32 (PA0)



www.sylvainmahe.xyz

LE BLOG

de Sylvain Mahé

contact@sylvainmahe.xyz

[Retour](#)[Suite](#)

Récapitulatif des fonctions et variables de cette classe:

```
unsigned int value = 0;
AnalogRead (const unsigned char PIN);
void read();
```

Détection de niveau de batterie faible:

Avec une entrée analogique disponible et un montage en **pont diviseur de tension**, il vous est possible aisément de détecter si votre batterie électrique est déchargée.

Imaginons le cas suivant, vous disposez comme alimentation électrique d'une batterie NIMH d'une **tension maximale de +12V**. Une précaution d'usage s'impose alors:

*Ne mettez jamais sur un port d'entrée/sortie de l'automate programmable une tension supérieure à **+5V**, vous risqueriez d'endommager irrémédiablement le microcontrôleur !*

Le but est donc de diviser cette tension pour qu'elle ne soit jamais supérieure à +5V **accumulateur complètement chargé + marge de sécurité**.

Pour faire un pont diviseur de tension, vous devez connecter une résistance sur le **pôle positif** de la batterie (cathode), une autre sur le **pôle négatif** de la batterie (masse/anode), et relier les autres extrémités encore non connectées des deux résistances entre elles, puis relier le tout sur une broche connectée au convertisseur analogique/numérique du microcontrôleur qui va servir à mesurer et convertir cette tension.

Calcul du pont diviseur de tension:

*Tension maximale de la batterie = **12V**
Tension niveau de batterie faible = **7V**
Tension d'alimentation du microcontrôleur = **5V**
Résistance connectée à la cathode de la batterie = **10kΩ**
Résistance connectée à l'anode de la batterie = **5kΩ***

Valeurs en sortie du pont diviseur de tension:

*Tension maximale = $(5k\Omega / (10k\Omega + 5k\Omega)) * 12V = \mathbf{4V}$ (1V de sécurité)
Intensité maximale = $(12V / (10k\Omega + 5k\Omega)) = \mathbf{0.0008A}$ (0.8mA)
Tension niveau de batterie faible = $(5k\Omega / (10k\Omega + 5k\Omega)) * 7V = \mathbf{2.33V}$
Convertie en base 10 = $(1023 / (5V / ((5k\Omega / (10k\Omega + 5k\Omega)) * 7V))) = \mathbf{478}$*

Avec ce montage, la tension entre la masse et l'entrée analogique du microcontrôleur n'excédera pas **+4V**, l'intensité maximale sera de **0.8mA**, et le seuil de niveau de batterie faible à indiquer en programmation sera de **478** en base 10 (arrondi à l'entier supérieur), ce qui donne le programme suivant:

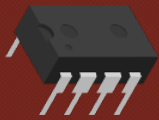
```
#include "../module/1284p/AnalogRead.h"

int main()
{
    AnalogRead voltage = AnalogRead (25);

    while (true)
    {
        voltage.read();

        //si la tension de la batterie est inférieure à +7V:
        if (voltage.value < 478)
        {
            //niveau de batterie faible
        }
    }

    return 0;
}
```



www.sylvainmahe.xyz

LE BLOG

de Sylvain Mahé

contact@sylvainmahe.xyz

[Retour](#)[Suite](#)

Si la tension de votre source d'alimentation fluctue de façon importante, il vous est possible de lisser la valeur à l'aide de la classe **Filter.h**, comme le montre l'exemple suivant:

```
#include "../module/1284p/AnalogRead.h"
#include "../module/1284p/Filter.h"

int main()
{
    AnalogRead voltage = AnalogRead (25);
    Filter filterVoltage = Filter (1000);

    while (true)
    {
        voltage.read();
        filterVoltage.set (voltage.value);

        //si la tension de la batterie est inférieure à +7V:
        if (filterVoltage.value < 478)
        {
            //niveau de batterie faible
        }
    }

    return 0;
}
```

Dans ce présent cas, le filtre va effectuer une moyenne sur **1000 échantillons**, la tension récupérée sera alors débarrassée des fluctuations venants parasiter la mesure.

design du blog: sylvain mahé