

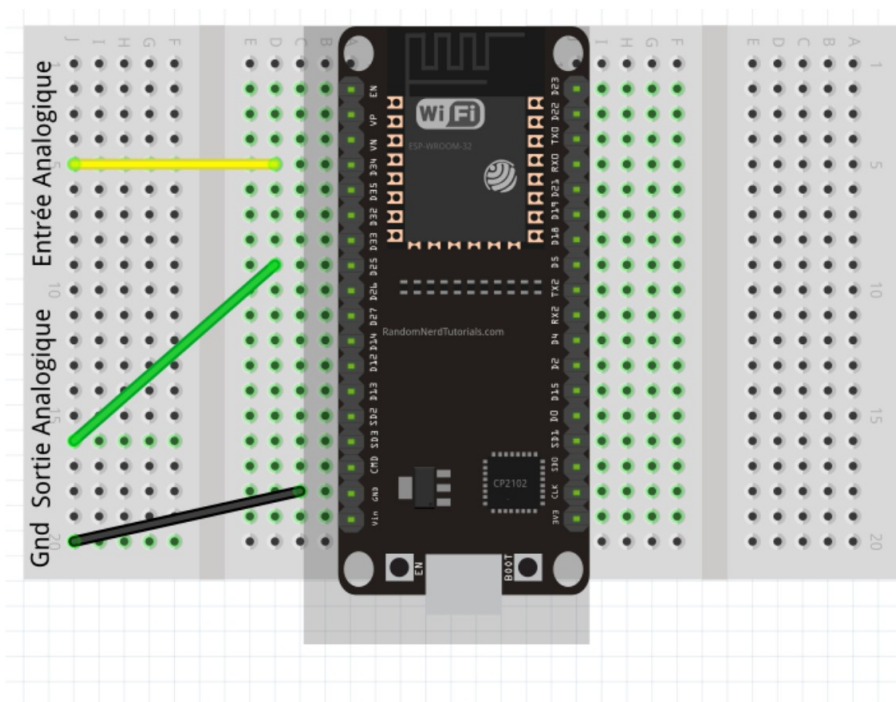
Filtrage Numérique

1 Introduction

Les applications embarquées dans les microcontrôleurs traitent des signaux provenant de différents capteurs (des capteurs de lumière, de températures, etc). Pour extraire des informations de ces signaux et déclencher des actions, ces signaux peuvent nécessiter un filtrage numérique. Par exemple, pour un capteur qui délivre un signal en principe lentement variable mais présentant un bruit important, il est préférable d'échantillonner le bruit puis de procéder à un filtrage passe-bas numérique. Un autre exemple courant est le filtre intégrateur ou le filtre dérivateur.

2 Premier montage

Le montage consiste à connecter une entrée analogique sur un GBF et une sortie analogique sur un oscilloscope.



Le premier programme permettra d'estimer la période minimale d'échantillonnage T_e .

$$y_n = x_n$$

```
#include <driver/dac.h>
#include <Arduino.h>

#define InAna 34

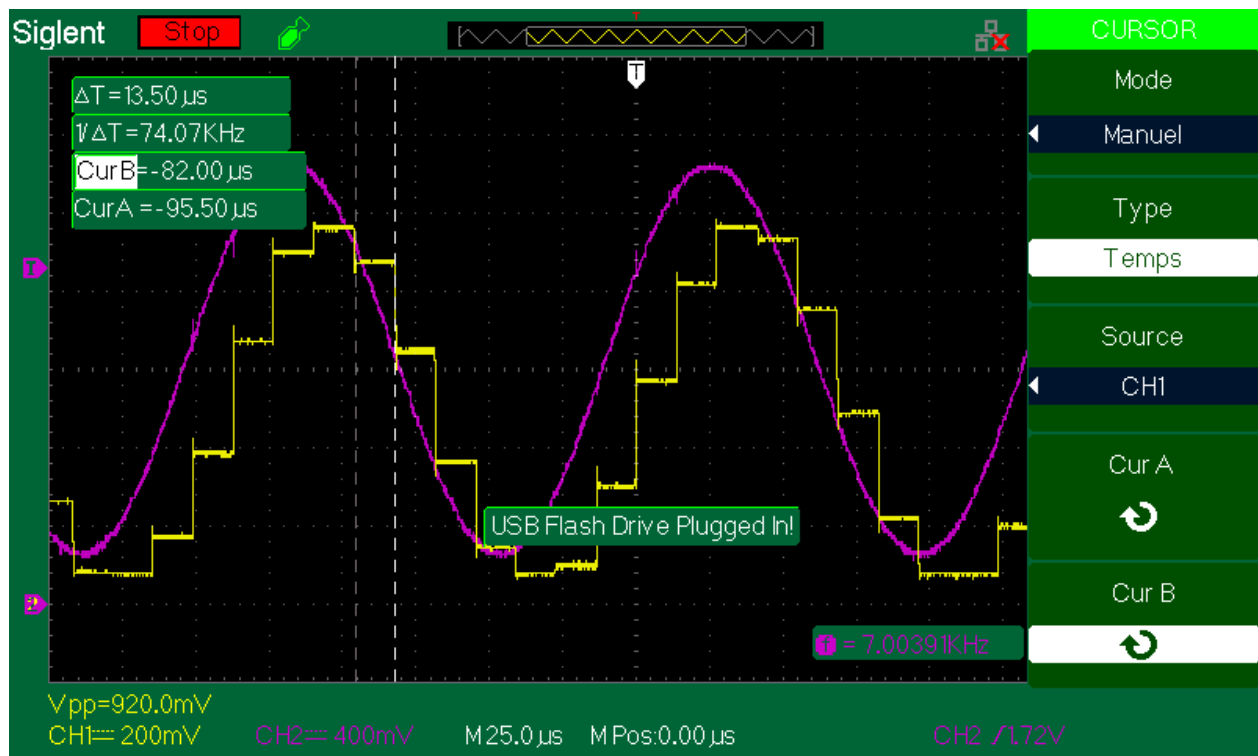
int x;
int y;

void setup() {
  dac_output_enable(DAC_CHANNEL_1);
}
```

```

}
void loop() {
  x = analogRead(InAna)/16;
  dac_output_voltage(DAC_CHANNEL_1,x );
}

```



On observe le signal d'entrée analogique (en **violet**), c'est un signal sinusoïdale fréquence 7kHz. Le signal de sortie du convertisseur (en **jaune**).

La période d'échantillonnage est mesurée à **13,5 us**

3 Filtre passe-bas du premier ordre

3-1 Filtre passe bas du premier ordre $\tau = 2 T_e$

$$y(n) = 0,333 x(n) + 0,666 y(n-1)$$

Le programme correspondant pour esp32

```
#include <driver/dac.h>
#include <Arduino.h>

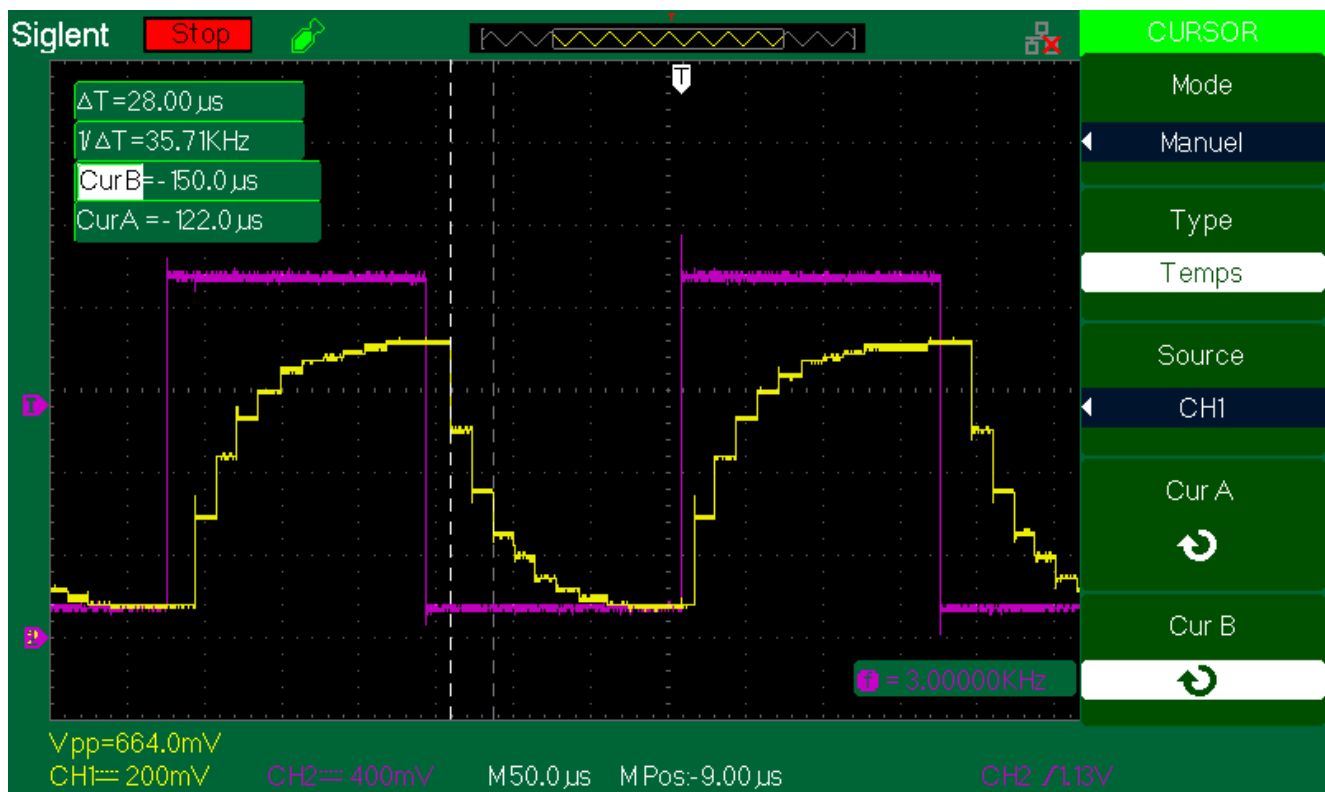
#define InAna 34

short x[2];
short y[2];
unsigned char n;

void setup() {
    dac_output_enable(DAC_CHANNEL_1);
    n = 1;
    y[n-1] = 0;
}

void loop() {
    x[n] = analogRead(InAna)/16;
    y[n] = (x[n] + 2 * y[n-1])/3; // équation de récurrence passe bas 1er ordre
    dac_output_voltage(DAC_CHANNEL_1, y[n]);
    y[n-1] = y[n]; // yn-1 <- yn
}
```

Réponse indicielle : $\tau = 2 T_e = 28\mu s$



3-2 Filtre passe bas du premier ordre $\tau = 10 T_e$

$$y(n) = (x(n) + 10 \times y(n-1)) / 11$$

Le programme correspondant pour esp32

```
#include <driver/dac.h>
#include <Arduino.h>

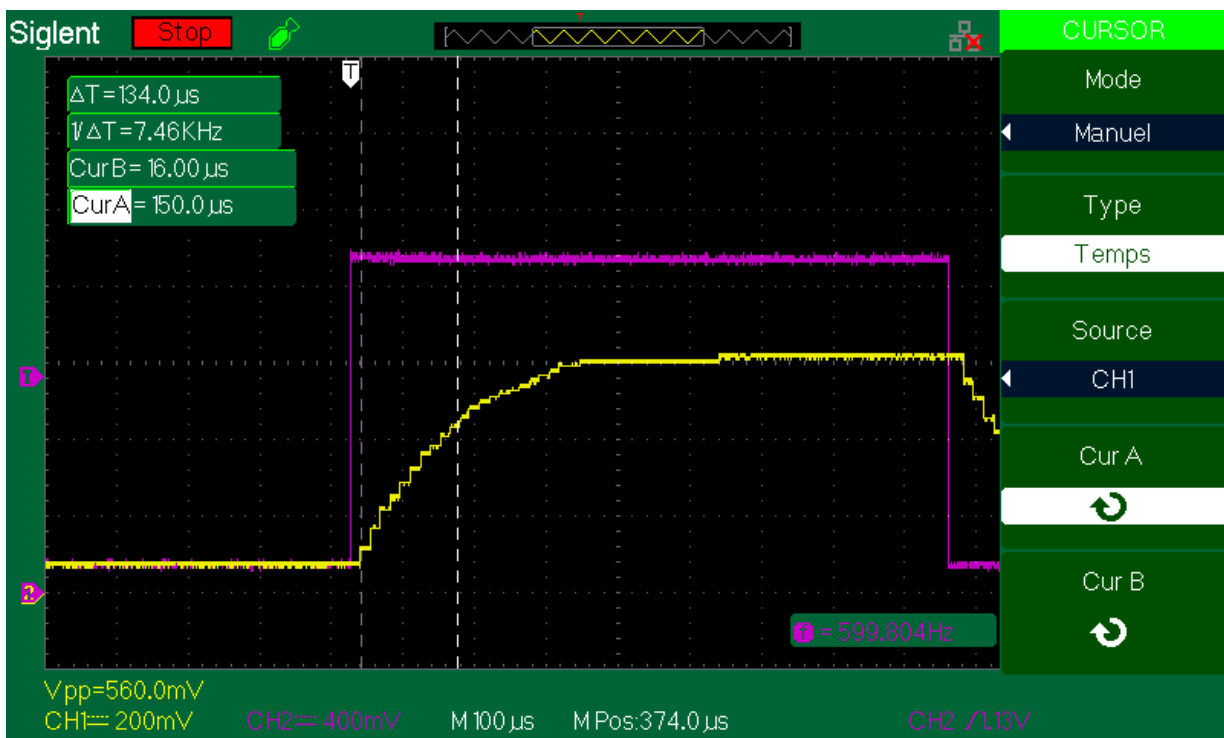
#define InAna 34

short x[2];
short y[2];
unsigned char n;

void setup() {
  dac_output_enable(DAC_CHANNEL_1);
  n = 1;
  y[n-1] = 0;
}

void loop() {
  x[n] = analogRead(InAna)/16;
  y[n] = (x[n] + 10 * y[n-1])/11; // équation de récurrence passe bas 1er ordre
  dac_output_voltage(DAC_CHANNEL_1, y[n]);
  y[n-1] = y[n];                // yn-1 <- yn
}
```

Réponse indicielle : $\tau = 10 T_e = 134 \mu s$



4 Filtre passe-bas du 2ième ordre

On se propose de synthétiser la transmittance

$$H(p) = \frac{Y(p)}{X(p)} = \frac{\omega_0^2}{\omega_0^2 + 2m\omega_0 p + p^2}$$

avec

- $\omega_0 = 1000\pi \text{ rad/s}$ $T_0 = 2 \text{ ms}$ soit $f_0 = 500 \text{ Hz}$
- $m = 0,1$
- $T_e = 0,2 \text{ ms}$ ou $200 \mu\text{s}$ soit $f_e = 5000 \text{ Hz}$

Filtre de pulsation propre $T_0 = 2\pi/\omega_0 = 10.T_e$

4-1 Par la transformation rectangulaire :

Aussi appelé transformation par équivalence de la dérivation

$$p \Leftrightarrow \frac{1-Z^{-1}}{T_e}$$

Équation de récurrence :

$$y_n = 0,25965 x_n + 1,39805 y_{n-1} - 0,65770 y_{n-2}$$

```
#include <driver/dac.h>
#include <Arduino.h>

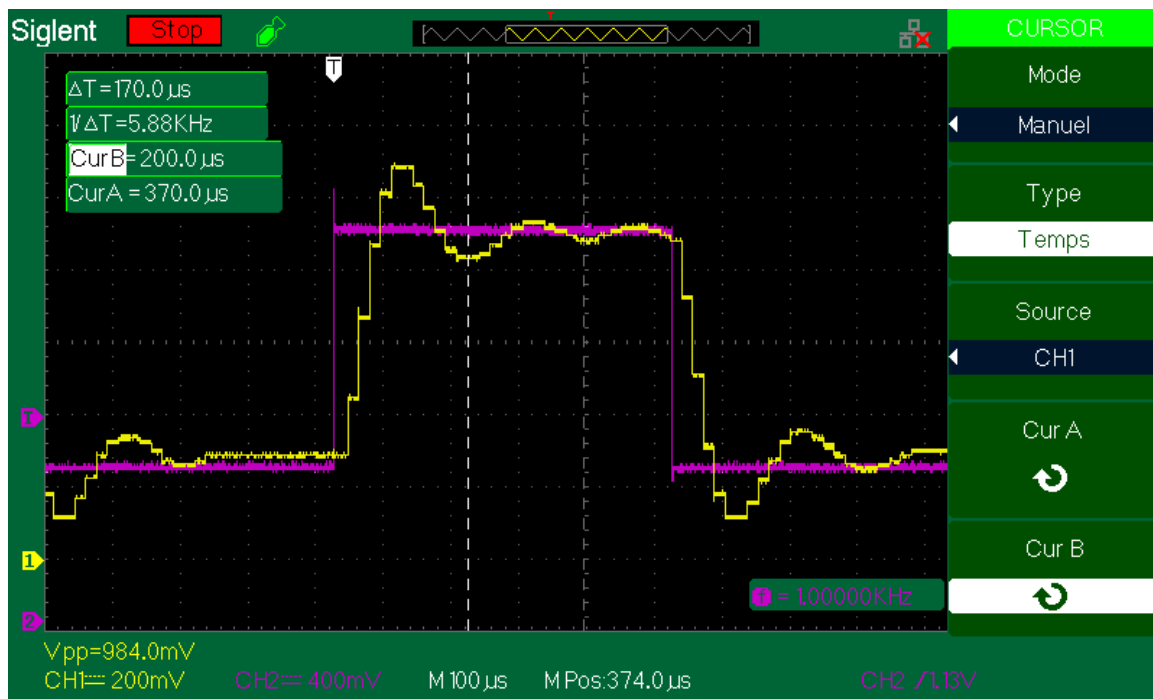
#define InAna 34

short x[3];
short y[3];
unsigned char n;

void setup() {
    dac_output_enable(DAC_CHANNEL_1);
    n = 2;
    y[n-1] = 0;
    y[n-2] = 0;
}

void loop() {
    x[n] = analogRead(InAna)/16;
    y[n] = 0.2596 * x[n] + 1.398 * y[n-1] - 0.6577 * y[n-2];
    dac_output_voltage(DAC_CHANNEL_1, y[n]);
    y[n-2] = y[n-1];
    y[n-1] = y[n];
}
```

période d'échantillonnage : $T_e=17\mu s$ $T_0=170\mu s$ GBF : signal carré $\sim 1\text{ kHz}$



La fréquence échantillonnage n'étant pas suffisamment élevée par rapport à f_0 , le calcul des coefficients par équivalence de la dérivation conduit à une réponse d'une médiocre précision.

4-2 Par la transformation bilinéaire

Aussi appelé transformation par équivalence de l'intégration

$$p \Leftrightarrow \frac{2}{T_e} \frac{1-Z^{-1}}{1+Z-1}$$

Ce qui donne l'équation de récurrence :

$$y_n = 0,084971 \cdot (x_n + 2x_{n-1} + x_{n-2}) + 1,55193y_{n-1} - 0,89181y_{n-2}$$

$$y_n = 0,084971x_n + 0,16994x_{n-1} + 0,084971x_{n-2} + 1,55193y_{n-1} - 0,89181y_{n-2}$$

Le programme correspondant :

```
#include <driver/dac.h>
#include <Arduino.h>

#define InAna 34

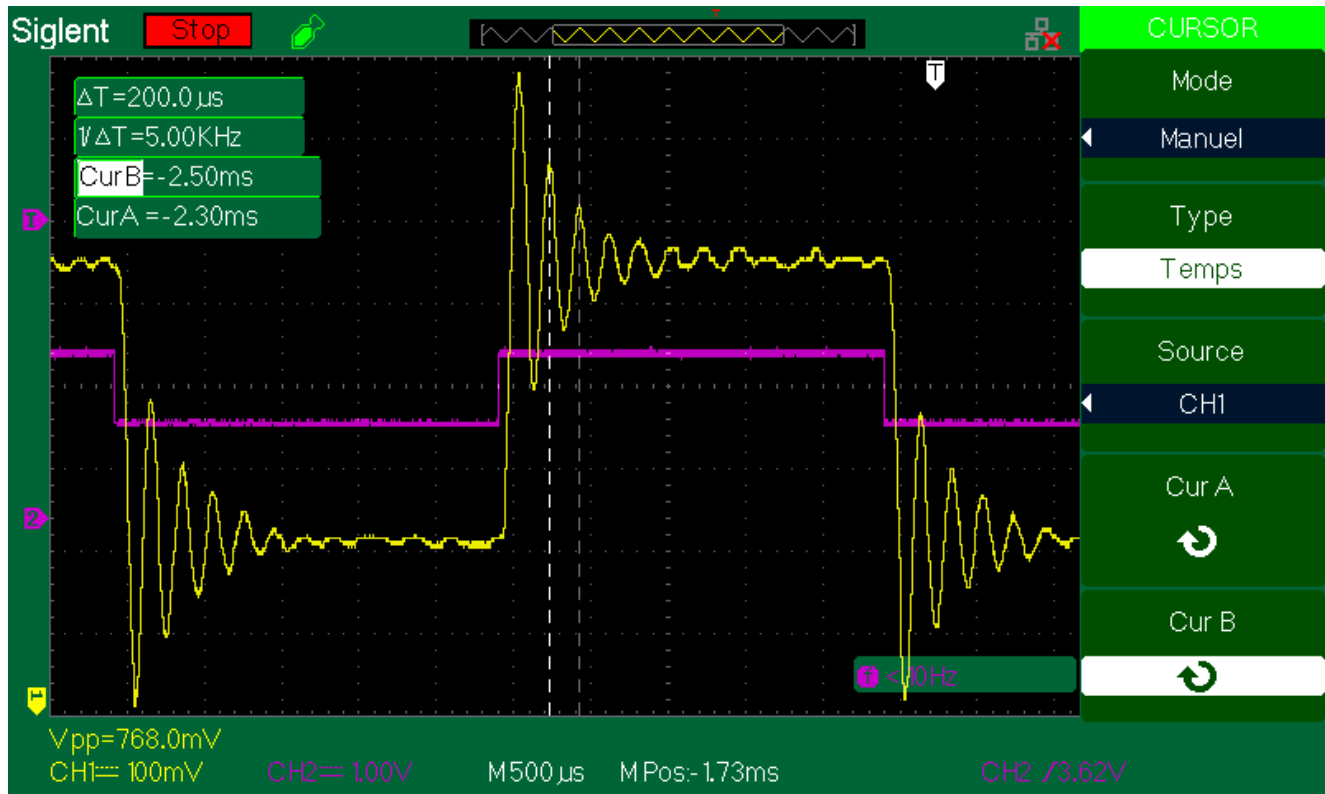
float x[3];
float y[3];
unsigned char n;

void setup() {
    dac_output_enable(DAC_CHANNEL_1);
    n = 2;
    y[n-1] = 0;
    y[n-2] = 0;
    x[n-2] = 0;
    x[n-1] = 0;
}

void loop() {
    x[n] = analogRead(InAna)/16;
    y[n] = 0.08497 * x[n];
    y[n] += 0.16994 * x[n-1];
    y[n] += 0.08497 * x[n-2];
    y[n] += 1.5519 * y[n-1];
    y[n] -= 0.8918 * y[n-2];
    dac_output_voltage(DAC_CHANNEL_1, y[n]);
    y[n-2] = y[n-1];
    y[n-1] = y[n];
    x[n-2] = x[n-1];
    x[n-1] = x[n];
}
```

Période d'échantillonnage : $T_e = 20\mu s$ $T_0 = 200\mu s$

GBF : signal carré $\sim 200\text{Hz}$ $\text{ampl} = 0,43\text{V}$ $\text{offset} = 0,75$



On vérifie la réponse à un échelon d'un système numérique du 2ième ordre (voir cours) pour $m=0,1$.

On peut comparer les deux transformations (la transformation bilinéaire est plus performante)

5 Calcul des coefficients

<http://www.iowahills.com/Index.html>