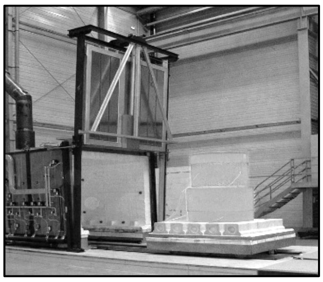
TD 1 : surveillance de température d'un four industriel

A- Présentation

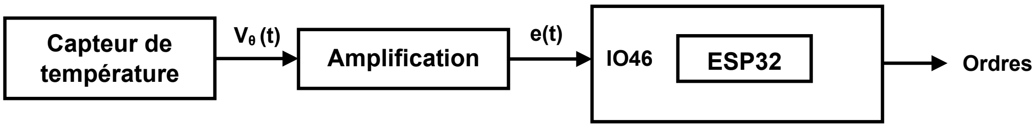


Une entreprise industrielle souhaite mettre en place un système de surveillance de température dans un four industriel. Le capteur utilisé délivre un signal analogique **e(t)** mesurant la température interne.

Cependant, ce signal est pollué par du bruit haute fréquence provenant des équipements environnants (moteurs, relais…). Pour assurer une lecture fiable, on décide d'implémenter un filtre passe-bas numérique de premier ordre dans une carte de traitement ESP32.

Le signal contient une composante utile basse fréquence (variation lente de température **< 5 Hz**), mais aussi des parasites de haute fréquence dus au bruit électromagnétique des onduleurs (composantes de **100 Hz** et **200 Hz** observées via analyse spectrale par FFT).

Objectif : concevoir un **filtre passe-bas numérique du 1er ordre** permettant de conserver l'évolution thermique utile tout en supprimant le bruit parasite.



Spécifications techniques :

Composantes parasites observées	100 Hz et 200 Hz
Signal utile	< 5 Hz
Fréquence de coupure visée	Fc = 10 Hz
Carte de traitement	ESP32
Fréquence d'échantillonnage proposée	Fe = 1000 Hz

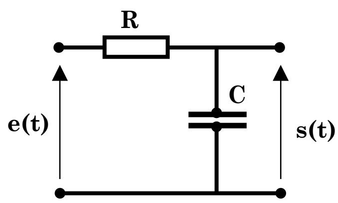
ESP32

B- Analyse du spectre et justification du choix de la fréquence d'échantillonnage

- **Question 1 :** Justifier pourquoi un filtre avec **fc =10 Hz** et pourquoi les parasites à **100 Hz** et **200 Hz** peuvent être supprimés sans altérer la mesure utile.
- **Question 2 :** Rappeler le **théorème de Shannon**.
- **Question 3 :** Conclure sur le choix de **Fe = 1000 Hz.**

C- Modélisation du filtre analogique

L'objectif est de dimensionner un filtre passe-bas passif de premier ordre, d'en déterminer la fonction de transfert, puis d'établir son équation différentielle afin de le transposer en filtre numérique.



- **Question 4 :** Exprimer la fonction de transfert du filtre **H(jω)** en se basant de schéma ci-après et la mettre sous la forme canonique suivante : $H(j\omega) = \frac{G}{1 + \tau j\omega}$

  Que vaut les expression **G** et **τ** ?
- **Question 5 :** Calculer les valeurs numériques de **G** et de **τ** sachant que la fréquence de coupure du filtre **fc = 10 Hz**

En négligeant les conditions initiales, on peut approximer l'expression suivante : $\mathbf{j\omega\ \underline{X}} \Leftrightarrow \frac{\mathbf{d\,x(t)}}{\mathbf{dt}}$. Cela montre que, dans le domaine fréquentiel, la multiplication par jω correspond à une dérivation dans le domaine temporel

- **Question 6 :** En utilisant cette approximation, exprimer l'équation différentielle sous la forme suivante :
  $\mathbf{T}\ \frac{\mathbf{d\,s(t)}}{\mathbf{dt}} + \mathbf{s(t)} = \mathbf{K\,e(t)}$, que vaut les expressions de **T** et **K** ?

D- Transposition vers le numérique

L'étape suivante consiste à transposer le filtre analogique en filtre numérique en utilisant l'approximation d'Euler explicite, qui permet de discrétiser l'équation différentielle à l'aide d'une relation entre échantillons successifs du signal.

$$\boldsymbol{y(t) = \frac{d\,x(t)}{dt} \quad \Rightarrow \quad y_{(n)} = \frac{x_{(n)} - x_{(n-1)}}{Te}}$$

On note : $\boldsymbol{s_{(n)} = s\ (t = \mathrm{n}.T_e)}$, $\boldsymbol{e_{(n)} = e\ (t = nTe)}$, $\boldsymbol{n \in \mathbb{N}}$. Avec Te est la période d'échantillonnage (temps de calcul)

- **Question 7 :** Exprimer l'équation du filtre discrétisé sous la forme $\boldsymbol{s_{(n)} = \mathbf{b}\ s_{(n-1)} + \mathbf{a}\,.\,e_{(n)}}$ en explicitant les termes **a** et **b**.
- **Question 8 :** Compléter l'algorithme ci-dessous.

```
// Filtrage numérique sur ESP32 / Arduino

    float a = ............ ;                // insérer Paramètres du filtre

    float b = ............;                 // insérer Paramètres du filtre

    float sn = 0, sn1 = 0, en = 0;          // Initialisation des paramètres


void loop() {

     en = analogRead(46 );                  // Lire le signal analogique par le CAN (broche 46)

     sn =..........................................;   // Appliquer le filtre

     ............................................;    // Mémoriser la sortie précédente

     delay(............);                   // Te = 1 ms

}
```