

## Dossier TIPE:

# **Un exemple d'utilisation de la logique floue.**

Tuton Sophie  
2014-2015

# Sommaire

<b>Préface</b> .....	<b>p1</b>
----------------------	-----------

## **Partie I: Le système (p2 à p18)**

Introduction à la partie I: explication du fonctionnement général.....	p3
I. L'algorithme général.....	p3
II. La logique floue dans l'algorithme de régulation.....	p5
II.1. Les 4 lois de régulation et les ensembles flous considérés.....	p5
II.2 Les fonctions d'appartenance aux ensembles.....	p5
II.3 Les opérateurs: union, intersection et implication.....	p8
II.3.a. Les opérateurs de Zadeh.....	p8
II.3.b Un exemple.....	p9
II.4 L'obtention d'une décision via l'algorithme de logique floue.....	p10
III/ L'interface.....	p12
III.1. Les composants et leurs caractéristiques.....	p12
III.1.a. Python.....	p12
III.1.b. La carte Arduino.....	p12
III.1.c Les capteurs LM335A.....	p13
III.2. Le dialogue entre les composants.....	p15
III.3 L'algorithme régissant l'interface (Arduino).....	p16
IV. L'algorithme principal.....	p17
IV.1 La sauvegarde des données.....	p17
IV.2 L'algorithme de régulation.....	p19

## **Partie II: Les expériences (p21 à p26)**

Introduction à la partie II.....	p22
I. L'enceinte expérimentale.....	p23
II. Le calcul des déperditions thermiques.....	p24
III. Les résultats.....	p25
III.1 Les 10 expériences.....	p25
III.2 Une expérience "bonus".....	p27

<b>Annexe</b> .....	<b>p30</b>
- Les courbes obtenues lors des 10 premières expériences.....	p30
- Bibliographie.....	p43

# Préface

La logique floue, ou *fuzzy logic* en anglais, utilise des quantités nuancées contrairement à la logique classique qui utilise le vrai et le faux. Il s'agit en fait d'une extension de la logique classique, fondée par Lotfi Zadeh en 1965.



Selon L.Zadeh, la logique floue **"c'est cette capacité qui fait que l'homme peut déchiffrer une écriture peu soignée, comprendre une élocution déformée et focaliser son attention sur une information qui soit pertinente..."**

La logique floue est maintenant utilisée dans beaucoup de domaines, par exemple dans:

- l'automatisme: avec l'automatisation de **métro** de Sandai en 1988 ou même l'automatisation de hauts **fourneaux** à Fos-sur-Mer et à Dunkerque en 1990,. Plus récemment la logique floue est utilisée dans des **machines à laver** et des **autocuiseurs** Panasonic, la **caméra** vidéo Sanyo, les **freins ABS** .... etc

- la robotique car la logique floue peut permettre la réaction à un environnement mal connu, à la **reconnaissance de forme** ou de mot..

- l'environnement puisque la logique floue est fréquemment utilisé pour le contrôle des **stations d'épurations**.

- la médecine avec l'**aide au diagnostic**.

# Partie I: Le système

## Introduction: explication du fonctionnement général

L'**objectif** est de créer un chauffage dont le fonctionnement se base sur la logique floue. Ce chauffage regule la température d'une enceinte en fonction du caractère économe de l'utilisateur. Pour cela on a conçu un algorithme de régulation sous Python, une interface ordinateur-capteurs et l'enceinte expérimentale.

Voici le **fonctionnement global** du système:

Le programme de régulation est lancé depuis un ordinateur. Une durée notée  $t_c$  (appelée temps de boucle), le caractère économe, la température de consigne et les caractéristiques de l'enceinte sont écrits dans un fichier de cet ordinateur – le caractère économe est décrit par une note sur 10 (plus la note est élevée, plus l'utilisateur est économe) -.

Le programme fonctionne en boucle d'une durée  $t_c$ . A chaque boucle: à partir des températures intérieures et extérieures obtenues via deux capteurs, l'algorithme calcule un temps de chauffe ( $< t_c$ ) et l'envoie à un Arduino (microcontrôleur). Ce dernier déclenche l'alimentation du résistor chauffant puis l'arrête dès que le temps de chauffe calculé est écoulé.

## I/L'algorithme général

Les données d'entrée de l'algorithme sont : la température extérieure  $T_{ext}$ , la température intérieure  $T_{int}$ , mesurées par les capteurs. Mais il y a également la température de consigne  $T_c$ , la note du caractère économe  $E$  et le temps de boucle  $t_c$  stockés dans un fichier modifiable de l'ordinateur nommé `fichier_val`\*.

Tous les  $t_c$ , l'algorithme représenté sur l'organigramme à la page suivante s'exécute. Il y a tout d'abord le calcul des déperditions thermiques noté  $Perdue$  (cf. Partie II / II.). Si ces déperditions sont négatives alors l'algorithme renvoie le message d'erreur: "Erreur, le systeme est un chauffage" et il n'y aura pas de chauffe. Dans la cas contraire l'algorithme de logique floue est appliqué et retourne une puissance de chauffe  $P_{chauff}$ . Cette puissance de chauffe est ensuite convertie en un temps de chauffe  $t$  (inférieur à  $t_c$ ) à partir de  $t_c$  et  $P_{max}$  : c'est le temps durant lequel la résistance chauffante sera alimentée.

Puis il y a la mise à jour des données correspondant aux températures grâce aux capteurs: on recommence une nouvelle boucle.

A la page suivante: l'**organnigramme du fonctionnement général de l'algorithme**.

---

### \*`fichier_val`:

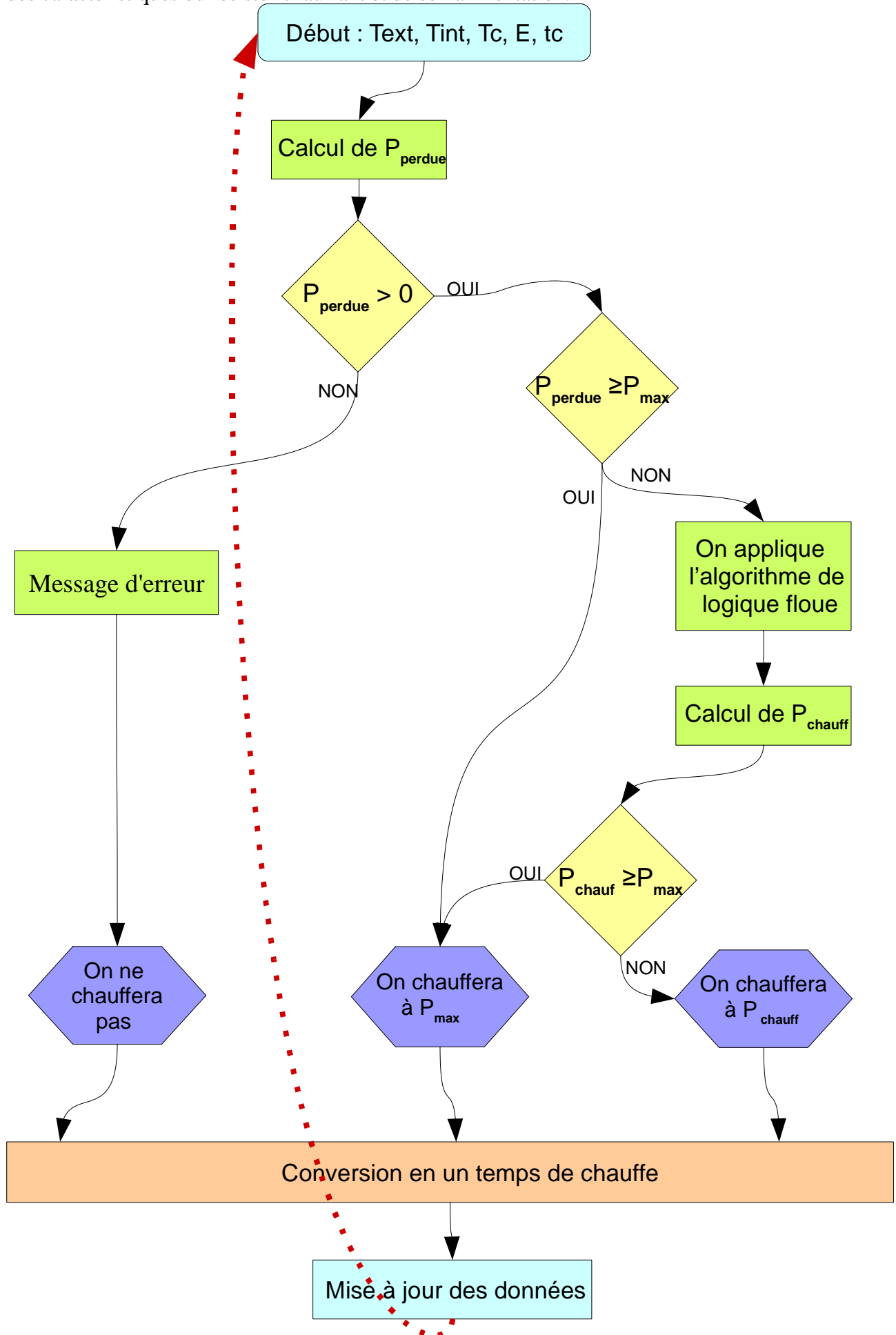
Fichier dans lequel sont stockées des valeurs modifiables par l'utilisateur telles que les caractéristiques de l'enceinte expérimentale, les caractéristiques de la résistance chauffante, la température de consigne, la note du caractère économe, le temps de boucle...etc

### \*\*`Pmax`

Il s'agit de la puissance de la résistance chauffante. Celle-ci est calculée dans le `fichier_val` à l'aide



des caractéristiques du résistor chauffant et de son alimentation.



## II/La logique floue dans l'algorithme de régulation.

### II.1.Les 4 lois de régulation et les ensembles.

L'algorithme de régulation est basé sur les lois qualitatives de régulation suivantes:




Loi n°1: (S'il fait **bon**) OU (qu'il fait **froid** et que l'utilisateur est **très économe**) *alors je ne compense que les pertes thermiques.*

Loi n°2: S'il fait **froid** et que l'utilisateur est **économe** *alors je chauffe.*

Loi n°3: S'il fait **froid** et que l'utilisateur n'est **pas économe** *alors je chauffe vraiment plus fort.*

Loi n°4: S'il fait **chaud** *alors je ne chauffe plus.*

Ces lois de régulations font appel à 3 variables linguistiques\*: **la température**, **la note du caractère**, **la puissance de chauffe** intervenant dans les ensembles dit flous suivant:

Ensemble	Couleur associée (dans cette sous partie seulement)	Intervention	Variable linguistique concernée
Bon, chaud,froid		Caractérise la chaleur intérieure	La température
Pas économe, économe, très économe		Correspond au caractère économe de l'utilisateur	La note du caractère
Ne pas chauffer, compenser les pertes thermiques, chauffer, chauffer vraiment plus fort		Caractérise l'action permettant de réguler	La puissance de chauffe

### II.2 Les fonctions d'appartenance aux ensembles

A chaque ensemble considéré on associe une fonction d'appartenance\*\* décrite la page suivante

**\*Variable linguistique:**

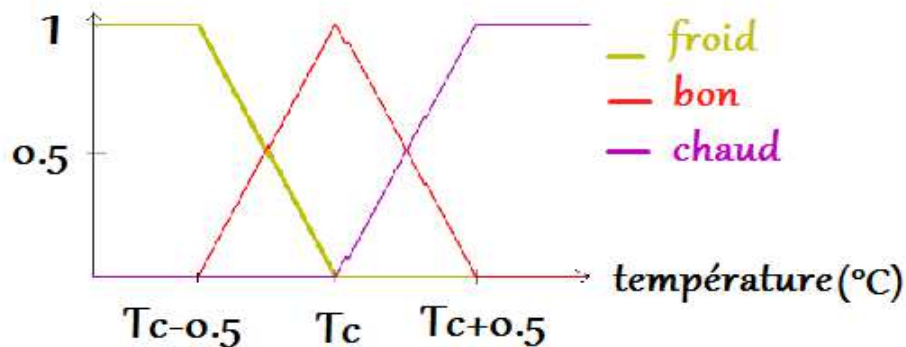
Les variables sont qualifiées de "linguistiques" car on les désigne par leurs noms.

**\*\*Fonction d'appartenance**

C'est une fonction qui à toute valeur d'entrée  $x$  associe son degré d'appartenance à l'ensemble considéré. Elle prend des valeurs comprises entre 0 et 1.

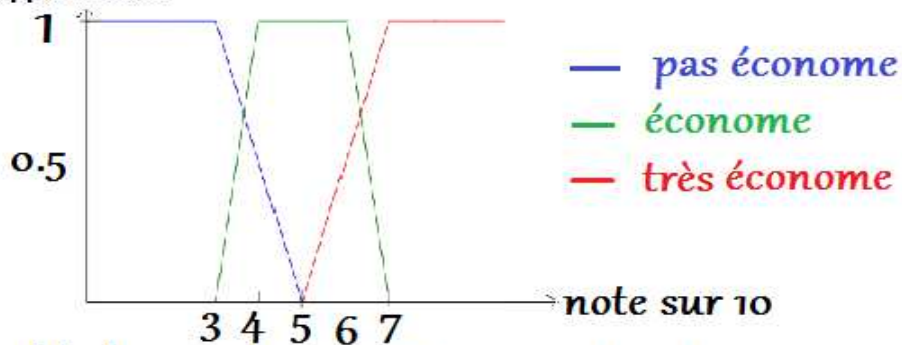
### ● Variable linguistique: Température

degré d'appartenance



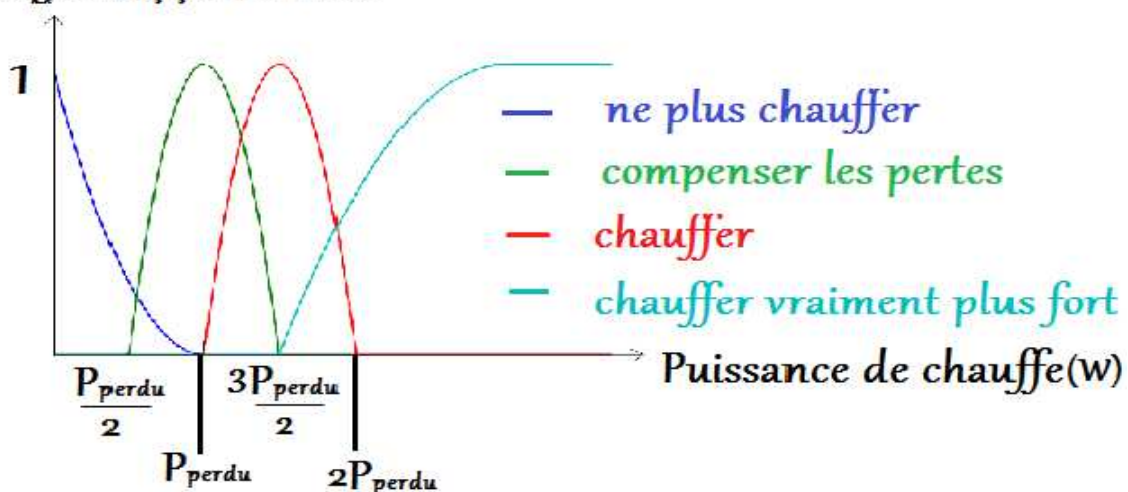
### ● Variable linguistique: Caractère de l'utilisateur

degré d'appartenance



### ● Variable linguistique: Puissance de chauffe

degré d'appartenance



Elles ont été créées à partir de quelques points caractéristiques. Ces points ont permis le calcul des équations des courbes tracées via Python. Ainsi pour chaque valeur de variable on peut obtenir son degré d'appartenance à chaque ensemble.

- Voici les fonctions d'appartenance associées aux ensembles **carcatérisant la chaleur intérieure**. Les données sont en entrée: la température de consigne  $T_c$  et la valeur de la

température intérieure (ici  $k$ ); et en sortie: le degré d'appartenance de la température intérieure à l'ensemble

```
def app_froid(k,Tc):
    if k<=Tc-0.5:
        froid=1.
    elif k>Tc:
        froid=0.
    else :
        froid=-2.*(k-Tc).
    return(froid)

def app_chaud(k,Tc):
    if k<=Tc:
        chaud=0.
    elif k>Tc and k<Tc+0.5:
        chaud=2.*(k-Tc)
    elif k>=Tc+0.5:
        chaud=1.
    return(chaud)

def app_bon(k,Tc):
    if k<Tc-0.5:
        bon=0.
    elif k>=Tc+0.5:
        bon=0.
    elif k>Tc-0.5 and k<Tc:
        bon=2.*(k-Tc+0.5)
    elif k>=Tc and k<Tc+0.5:
        bon=-2.*(k-Tc)+1.
    return(bon)
```

Ces fonctions sont écrites dans le fichier nommé **fuzzy\_temperature**

- Voici les fonctions d'appartenance associées aux ensembles caractérisant le caractère de l'utilisateur: Les données sont en entrée: la valeur de la note du caractère économe de l'utilisateur (ici  $k$ ); et en sortie: le degré d'appartenance de la note à l'ensemble.

```
def app_Pas_econome(k):
    if k<=3.:
        degre_appartenance=1.
    if k>3. and k<=5.:
        degre_appartenance=-0.5*(k-3.)+1.
    if k>5:
        degre_appartenance=0.
    return(degre_appartenance)

def app_Econome(X):
    if k<=3. or k>7.:
        degre_appartenance=0.
    if k>3. and k<=4.:
        degre_appartenance=k-3.
    if k>4. and k<=6.:
        degre_appartenance=1.
    if k>6. and k<=7.:
        degre_appartenance=-(k-6.)+1.
    return(degre_appartenance)

def app_Tres_econome(X):
    if k<=5.:
        degre_appartenance=0.
    if k>5. and k<=7.:
        degre_appartenance=0.5*(k-5.)
    if k>7.:
        degre_appartenance=1.
    return(degre_appartenance)
```

Ces fonctions sont écrites dans le fichier nommé **fuzzy\_econome**

- Voici les fonctions permettant de définir les fonctions d'appartenance associées aux ensembles caractérisant l'action de chauffer. Les données sont: en entrée: les déperditions thermiques (ici P) en Watt et la liste des puissances correspondant aux abscisses (X) en Watt; et en sortie: la liste des degrés d'appartenance correspondant aux ordonnées (Y).

```
def Chfbcp(X,P):
    Y=[]
    for k in X:
        if k<=(3./2.)*P:
            Y.append(0.)
        elif k>3*P:
            Y.append(1.)
        else:
            Y.append((-4./(9.*P**2.))*(k-(3./2.)*P)*(k-(9./2.)*P))
    return(Y)
```

```
def Chf(X,P):
    Y=[]
    for k in X:
        if k < P or k > 2.*P:
            Y.append(0.)
        else:
            Y.append((-4.)*1./P**2. * (k-P) * (k-2.*P))
    return(Y)
```

```
def Continue(X,P):
    Y=[]
    for k in X:
        if (-4)*1/P**2 * (k-P/2) * (k-3*P*1./2.) < 0:
            Y.append(0)
        else:
            Y.append((-4.)*1/P**2. * (k-P/2) * (k-3*P*1./2.))
    return(Y)
```

```
def Refroid(X,P):
    Y=[]
    for k in X:
        if k>P:
            Y.append(0.)
        else:
            Y.append((((k-P)/P)**2))
    return(Y)
```

### Attention !!!

Chfbcp ↔ Chauffer vraiment plus fort  
Continue ↔ Compenser les pertes  
thermiques

Refroid ↔ Ne pas chauffer

Chf ↔ Chauffer

Ces fonctions sont écrites dans le fichier nommé **puissance**.

## II.3.Les opérateurs: union, intersection et implication

### II.3.a) Les opérateurs de Zadeh

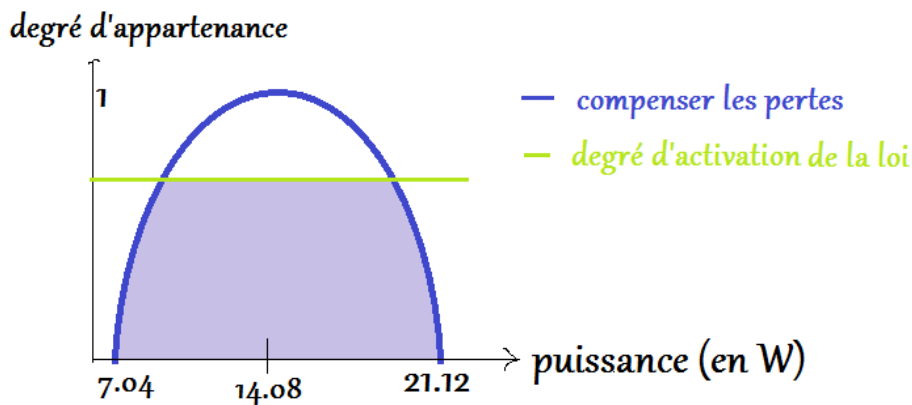
Outre les ensembles flous, les lois utilisent également l'union (OU), l'intersection (ET) et



comprise entre 0 et 1.

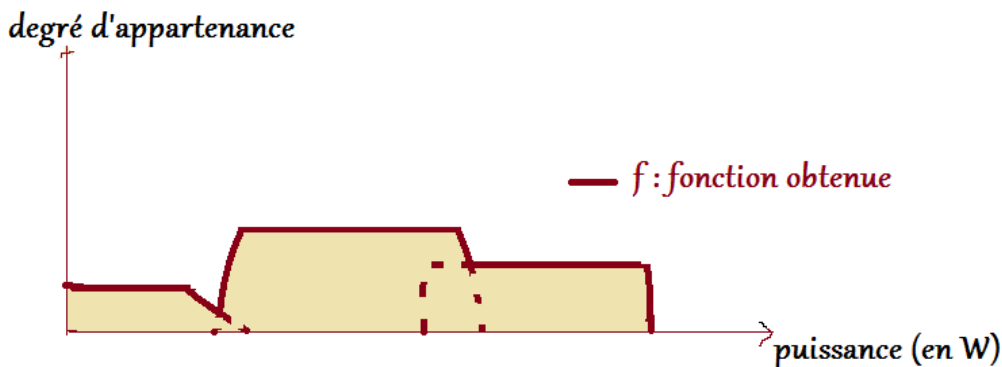
**\*\*Prédicat:**

Un prédicat est une proposition écrite dans la première partie de la loi (entre le "Si" et "Alors").



## IV.4. L'obtention d'une décision via l'algorithme de logique floue

On applique cette méthode sur les 4 lois. Puis on réunit les fonctions obtenues pour chaque lois. On obtient, par exemple, la figure ci-dessous.



Ensuite, on fait l'union (maximum) de toutes les courbes obtenues, puis on calcule le centre de gravité de la surface sous la courbe obtenue.

Ce centre de gravité est calculé entre discrétisant l'abscisse en une suite de  $x_i$ . On pose  $m_i = f(x_i)$  et  $x_G$  le centre de gravité. On a alors:

$$x_G = \frac{\sum_{i=1}^n x_{G_i} \cdot m_i}{\sum_{i=1}^n m_i}$$

Rappels:

- On refait ce calcul toutes les  $t_c$  secondes pour obtenir une décision de chauffe après mise à jour des températures.
- La puissance obtenue sera traduite en un temps de chauffe inférieur à  $t_c$

```

from fuzzy_temperature import*
from fuzzy_econome import*
from puissance import*

def ma_logique_floue(T,Tc,E,X,Pperdue):

    #LOIS 1: Si Bon ou (Froid et très econome) alors compense les pertes
    '''Calcul du degré d activation'''
    Degre_activ1=max(app_bon(T,Tc),min(app_froid(T,Tc),app_Tres_econome(E)))
    L1=[]
    for k in Continue(X,Pperdue):
        L1.append(min(k,Degre_activ1))

    #LOIS 2: Si Froid et Econome alors Chauffe plus fort
    Degre_activ2=min(app_froid(T,Tc),app_Econome(E))
    L2=[]
    for k in Chf(X,Pperdue):
        L2.append(min(k,Degre_activ2))

    #LOIS 3: Si Froid et Pas econome alors chauffe vraiment plus fort
    Degre_activ3=min(app_froid(T,Tc),app_Pas_econome(E))
    L3=[]
    for k in Chfbcp(X,Pperdue):
        L3.append(min(k,Degre_activ3))

    #LOIS 4: Si Chaud alors Chauffe moins.
    Degre_activ4=app_chaud(T,Tc)
    L4=[]
    for k in Refroid(X,Pperdue):
        L4.append(min(k,Degre_activ4))

    Y=[]
    i=0
    for k in X:
        m=max(L1[i],L2[i],L3[i],L4[i])
        Y.append(m)
        i=i+1

    '''Calcul du centre de gravité'''
    masse_totale=0.
    somme_de_produit=0.
    i=0
    for k in X:
        masse_totale=masse_totale + Y[i]
        somme_de_produit=somme_de_produit + k*Y[i]
        i=i+1
    centre_de_gravite = somme_de_produit/masse_totale
    return(centre_de_gravite)

```

A la page suivante: l'algorithme de logique floue Cette fonction est écrite dans le fichier

**logique\_floue**

Pour la **conversion de la puissance en un temps**: on calcule le pourcentage de puissance que cela donne par rapport à la puissance de la résistance chauffante. Le temps de chauffe correspondra au même pourcentage, mais par rapport à tc. Ensuite la valeur est arrondie de sorte qu'elle soit multiple de 10ms. Le temps obtenu est alors le temps de chauffe.

Pour contrer les troncatures diminuant le temps de chauffe (floor) ou l'augmentant (ceil) il faut



alterner les deux fonctions.

## III/L'interface

### III.1. Les composants et leurs caractéristiques

Pour l'étude de l'interface, on va tout d'abord expliquer les dialogues entre les principaux composants de la régulation. Ces derniers sont:

- ✓ l'ordinateur exécutant l'algorithme programmé sous Python
- ✓ la carte Arduino (cf III.1.b) exécutant l'algorithme d'interface
- ✓ les 2 capteurs de température: LM335A (un capteur extérieur et un intérieur)
- ✓ la Commande de chauffage ( lorsque qu'elle est allumée, le résistor chauffe par effet Joule)

#### III.1.a) Python

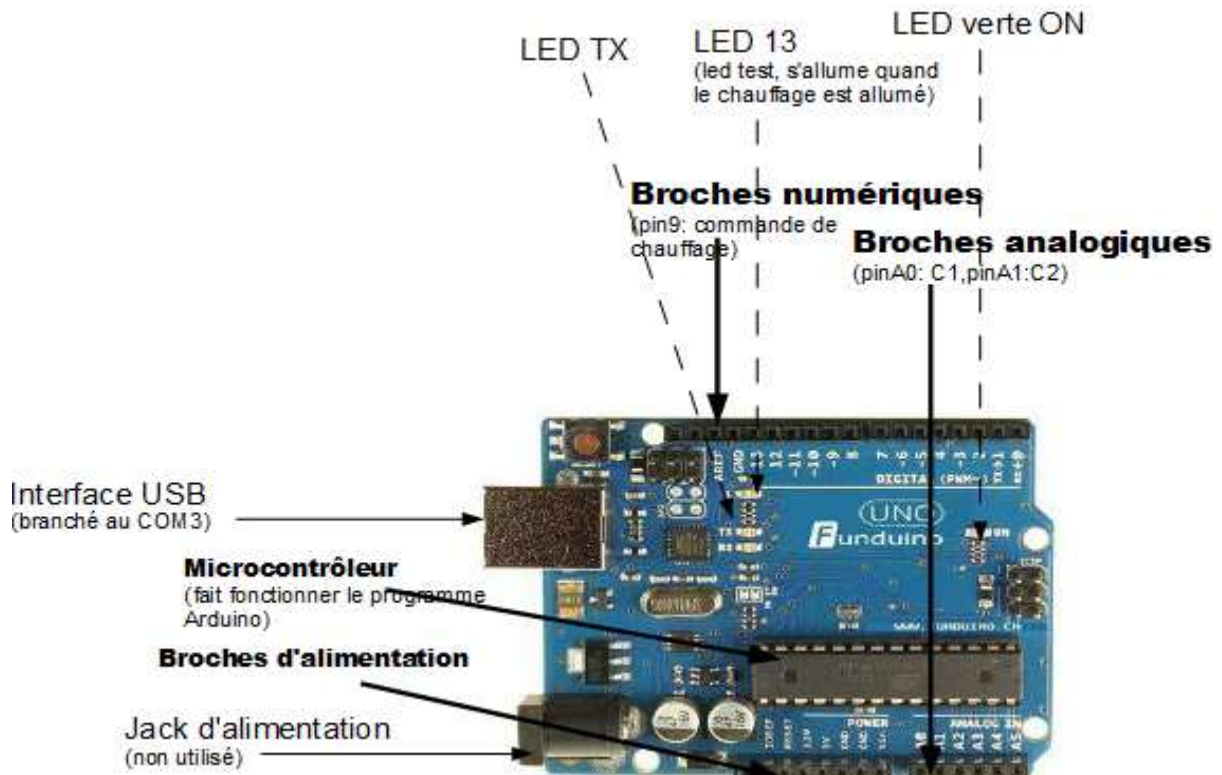


Le langage Python est utilisé avec l'interpréteur **Spyder de version 2.7**. Seules les bibliothèques: numpy, serial, pylab et time ont été utilisées (installées par défaut).

#### III.1.b) La carte Arduino

Arduino est un circuit imprimé en licence libre sur lequel se trouve un microcontrôleur que l'on peut programmer pour analyser ou produire des signaux électriques.

En réalité, la carte utilisée est une copie d'un Arduino Uno, appelée **Funduino Uno**. La carte est représentée ci-dessous avec ses caractéristiques à la page suivante.



Caractéristiques de la carte Arduino uno (identique au Funduino Uno):

-Micro contrôleur : ATmega328

**-Tension d'alimentation interne = 5V**

-tension d'alimentation (recommandée)= 7 à 12V, limites =6 à 20 V

-Entrées/sorties numériques : 14 dont 6 sorties PWM

-Entrées analogiques = 6

-Courant max par broches E/S = 40 mA

-Courant max sur sortie 3,3V = 50mA

-Mémoire Flash 32 KB dont 0.5 KB utilisée par le bootloader

-Mémoire SRAM 2 KB

-mémoire EEPROM 1 KB

-Fréquence horloge = 16 MHz

-Dimensions = 68.6mm x 53.3mm

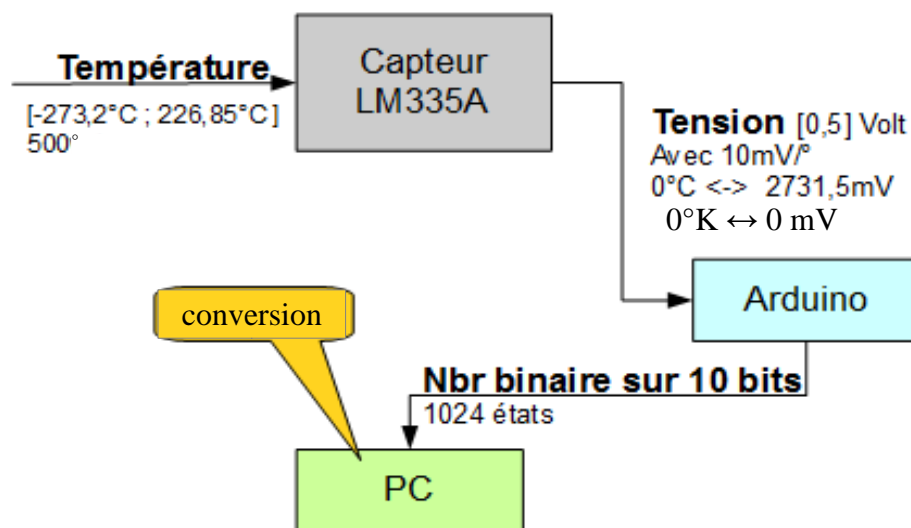
**-La carte s'interface au PC par l'intermédiaire de sa prise USB.**

-La carte s'alimente par le jack d'alimentation (utilisation autonome) mais peut être **alimentée par l'USB** (en phase de développement par exemple).

**-Convertisseur analogique-numérique (CAN) fonctionne sur 10bits**

### III.1.b) Les capteurs LM335A

Le **capteur LM335A** peut considérer des températures dans l'intervalle [-273,2°C ; 226,85°C ].  
Voici son fonctionnement:



La **température** est traduite par le capteur en une **tension** comprise entre 0 et 5V avec la formule suivante:

$$V_{lm335A} = 10(T - 273,15)$$

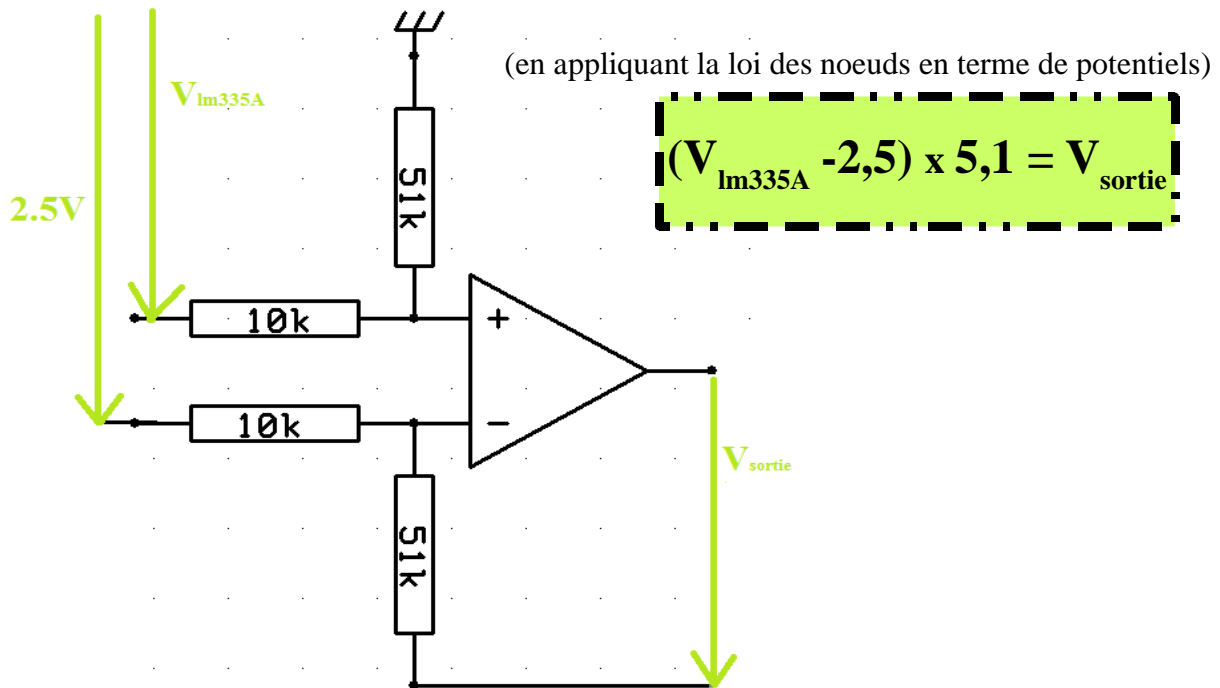
Tension aux bornes du capteur, en mV

Température au niveau du capteur, en °C

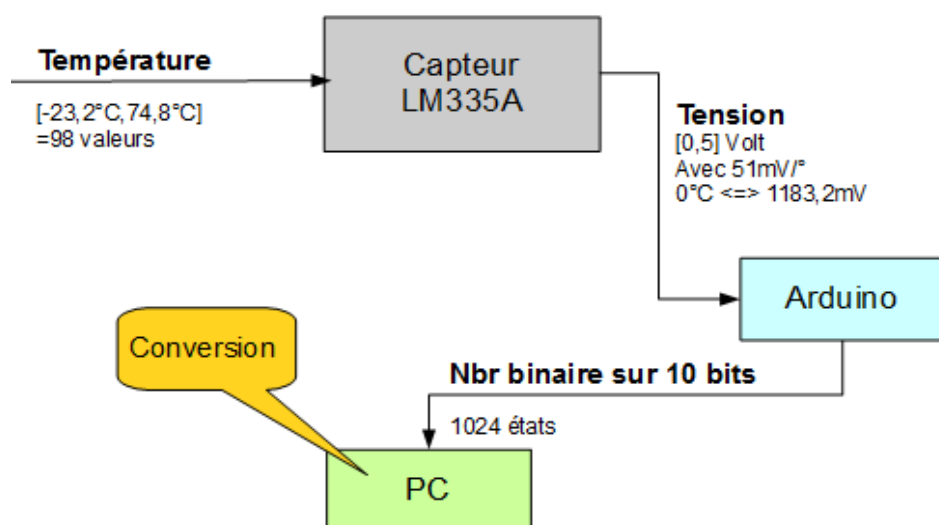
Puis la tension est convertie sur 10 bits par l'Arduino (qui peut donc coder  $2^{10}=1024$  états) et envoyée sur le port série de l'ordinateur. L'ordinateur convertira ensuite le **nombre binaire** obtenu en **température**.

Cependant, la plage de température comporte 500 valeurs discrètes et l'Arduino peut coder en tout 1024 états, la résolution de mesure est donc de  $500/1024$ , c'est-à-dire légèrement inférieure à  $0,5^\circ$ .

La résolution étant grossière\*, des modifications ont été faites. Il y a eu l'ajout d'un amplificateur opérationnel différentiel représenté sur le schéma ci-dessous fait à partir de ExpressSCH\*\*: le but était de diminuer la plage de température et de la recentrer sur l'intervalle des températures utilisées. On a alors diminué la tension obtenue par le capteur de 2500mV, puis multiplié la différence obtenue par un gain de 5.1, ce qui donne le schéma et la formule suivante:



Et voici les modifications engendrées sur le fonctionnement global de la capture des températures:



\*Résolution grossière

La résolution est qualifiée de grossière lorsque le pas de mesure (ici  $0,5^\circ$ ) est trop élevé.

### **\*\*ExpressSCH**

ExpressSCH est un logiciel permettant de saisir des schémas électroniques.

Ce qui donne la formule suivante:

$$V_{\text{sortie}} = (T + 1183,2) * 51$$

Tension aux bornes du capteur, en mV

Température au niveau du capteur, en  $^\circ\text{C}$

On a alors une plage de température de 98 valeurs pour 1024 états, donc:

**Résolution de mesure:**  $98/1024 = 0,1^\circ$

La fonction qui à un nombre binaire associe la température en  $^\circ\text{C}$  est la suivante:

```
def conversion(B):  
    mvB=B*5000./1024.  
    celsiusB= (mvB -1183.2)/51.  
    return(celsiusB)
```

Explication: Pour appliquer la conversion du nombre binaire en température, il faut tout d'abord le convertir en tension (ici en mV en multipliant le nombre par 5000/1024), puis convertir cette tension en degré Celsius (en soustrayant par l'ordonnée à l'origine 1183,2 puis en divisant le tout par le coefficient directeur 51).

Cette fonction se trouve dans le fichier **conversion**

## **III.2. Le dialogue entre les composants**

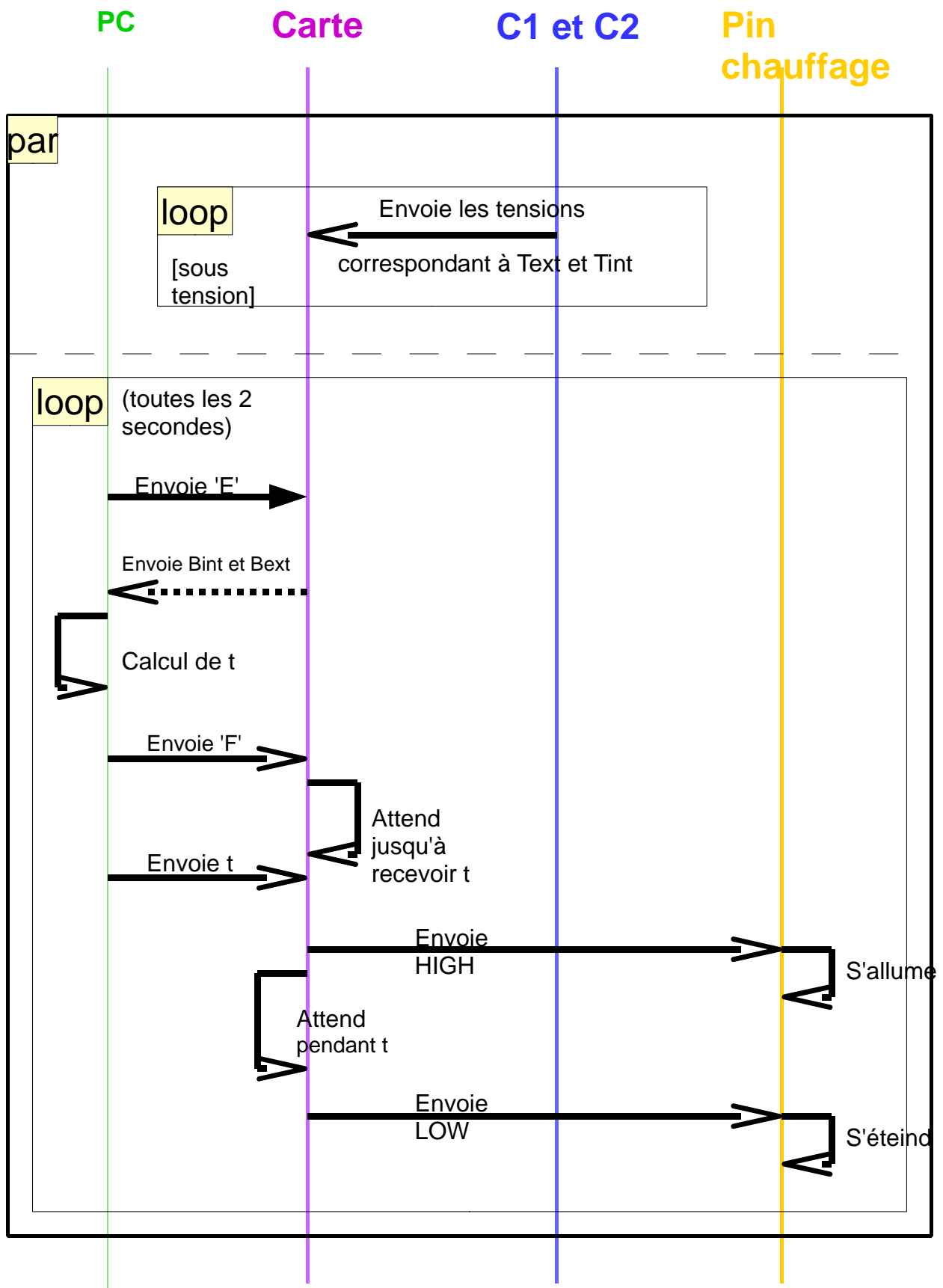
Le dialogue entre les composants correspond à deux boucles s'exécutant en parallèle:

- La première correspond à l'envoi des tensions - correspondant à Text et Tint – par les capteurs C1 et C2 à la carte Arduino. Elle s'effectue dès lors que le système est sous tension.

- La seconde s'effectue à chaque temps de boucle tc (égale à 2secondes ici).

L'ordinateur envoie un mot de commande 'E' à la carte Arduino, en réponse celle-ci va lui retourner les nombres binaires Bint et Bext correspondant aux tensions obtenues à l'autre boucle. Puis l'ordinateur calcule le temps de chauffe, envoie le mot de commande 'F' à la carte Arduino et lui envoie ensuite le temps de chauffe. L'Arduino déclenche alors la commande de chauffage (le résistor est alors alimenté), attend que le temps de chauffe s'écoule (l'effet Joule permet la chauffe), puis arrête la commande de chauffage (le résistor n'est plus alimenté).

**A la page suivante:** l'organigramme du dialogue entre les différents composants



## III.3 L'algorithme régissant l'interface: l'Arduino

► **Code couleur utilisé pour l'algorithme du microcontrôleurs (en C):** *noir italique* ou **gras** pour les commentaires, **gris gras** pour la définition des broches, **violet** pour ce qui fait référence à l'interface Arduino-ordinateur, **cyan** pour ce qui fait référence à l'interface Arduino-Capteurs, **bleu foncé** pour ce qui fait référence à l'interface Arduino-Commande de chauffe et noir simple pour le reste du code.

**//definition des broches utilisees**

**#define LED 13**

**#define CAPTEUR\_Int\_C1 0**

**#define CAPTEUR\_Ext\_C2 1**

**#define CHAUFFAGE 9**

void setup(){

pinMode(LED,OUTPUT);

pinMode(CHAUFFAGE,OUTPUT);

Serial.begin(9600);

}

void loop(){ **//boucle infinie**

if (Serial.available()) { *//si le port série est disponible*

char lecture=Serial.read(); *//je regarde si Python veut quelque chose*

**//Les temperatures**

if ( lecture == 'E') { *//Si Python demande les valeurs, Arduino exécute*

float B=analogRead(CAPTEUR\_Int\_C1); *//il lit la température sur le capteur*

float Bext=analogRead(CAPTEUR\_Ext\_C2); *//il lit la température sur l'autre capteur*

Serial.println(B);

Serial.println(Bext); *//il envoie à Python les nbr correspondant aux températures*

}

**//Le temps de chauffe**

if ( lecture == 'F' ) { *//il récupère le temps de fonctionnement du chauffage*

int dure=0;

char temps[10];

memset(temps, '\0',11);

int i=0;

while (Serial.available()< 1){ *// il se bloque ici tant que Python n'a pas envoyé le temps*

}

while (Serial.available()> 0){ *// il est ici quand Python a envoyé le temps*

temps[i] = Serial.read(); *//il lit le temps comme un tableau de caractère*

delay(10);

i++;

}

dure=atoi(temps); *// il convertit le tableau de caractère en entier*

digitalWrite(LED,HIGH); *//il allume la led*

digitalWrite(CHAUFFAGE,HIGH); *//il déclenche la commande de chauffage*

delay(dure); *//il attend que le temps de chauffe (ici dure) s'écoule*

digitalWrite(CHAUFFAGE,LOW); *//il éteint le chauffage*

digitalWrite(LED,LOW); *//il arrete la commande de chauffage*

}

}

}

## IV. L'algorithme de régulation:

### IV.1 La sauvegarde des données

A chaque temps de boucle les résultats -  $P_{perdu}$ ,  $T_{int}$ ,  $T_{ext}$ , temps de chauffe - sont sauvegardés dans un fichier stocké sur l'ordinateur. Deux fonctions ont été créées dans ce but:

- La première, nommée **preparation\_sauvegarde** ( ), sert à préparer le fichier de sauvegarde. La voici:

```
L2 def preparation_sauvegarde(E):  
L3     date=strftime('%d_%m_%y_%H%M%S',localtime())  
L4     nom=str(E)+'_'+date+'.txt'  
L5     fichier_save=open(nom,'a')  
L6     fichier_save.write('Pperdue\tTint\tText\ttemps_de_chauffe\t\t\n')  
L7     fichier_save.close()  
     return(nom)
```

Explication:

#L2: On récupère l'heure est la date sous le format : 30\_07\_2014\_\_14h39m45s

#L3: On lui rajoute txt pour faire un nom unique pour le fichier de sauvegarde

#L4: On crée (car il n'existe pas) puis ouvre le fichier de sauvegarde en mode "append"

#L5: On écrit l'entête du fichier de sauvegarde correspondant à la chaîne de caractères:  $P_{perdue}$   $T_{int}$

$T_{ext}$  temps\_de\_chauffe t

#L6: On referme le fichier de sauvegarde.

#L7: On retourne le nom du fichier préparé.

- La seconde fonction nommée **sauvegarde** ( ) enregistre les valeurs dans le fichier de sauvegarde:

```
L1 def sauvegarde (k, nom, Pperdue, liste_Pperdue, T, liste_temperature_int, Text,  
L2 liste_temperature_ext, temps_mult, liste_temps_mult, t, liste_temps):  
L3     liste_temps.append(k*t)  
L4     liste_temperature_int.append(T)  
L5     liste_temperature_ext.append(Text)  
L6     liste_Pperdue.append(Pperdue)  
L7     liste_temps_mult.append(temps_mult)  
L8     k=int(k)  
L9     fichier_save=open(nom,'a')  
L10    A_sauvegarder=str(liste_Pperdue[k-1])+'\t'+str(liste_temperature_int[k-1])  
L11    +'\t'+str(liste_temperature_ext[k-1]) + '\t' + str(liste_temps_mult[k-1]) +  
L12    '\t'+str(liste_temps[k-1]) + '\n'  
L13    fichier_save.write(A_sauvegarder)  
L14    fichier_save.close()
```

Explication:

#L1 à L7: On ajoute dans chaque liste, le nouvel élément

#L8: On prend la partie entière de k (nbr de boucles déjà exécutées)

#L9: On ouvre le fichier nom en mode "append". Il s'agit du fichier de sauvegarde

#L10 à L12: A\_sauvegarder correspond à la chaîne de caractères des valeurs:  $P_{perdue}$   $T_{int}$   $T_{ext}$

temps\_de\_chauffe t

#L13: On écrit les valeurs: Pperdue Tint Text temps\_de\_chauffe t à la suite du fichier déjà ouvert

#L14: On ferme le fichier de sauvegarde. LA SAUVEGARDE EST TERMINE.

Ces fonctions se trouvent dans le fichier **Sauvegarde**

Ainsi, tout au début de l'algorithme (en dehors de la boucle de régulation) on fait appel à la fonction **preparation\_sauvegarde** pour créer le fichier de sauvegarde, puis à chaque boucle de régulation on fera appel à la fonction **sauvegarde** pour enregistrer chaque valeur.

Mais, l'algorithme enregistre également l'allure des courbes de régulation tous les 10 tours de boucle grâce au ligne de code suivante, se trouvant dans l'algorithme principal:

```
L1 Eco=str(E)+' '
L2 '''Au bout de 10boucles on sauvegarde les graph'''
L3 if r==10:
L4     print "sauvegarde des graph"
L5     date=strftime('%d %m %y %H%Mmin%Ss',localtime())
L6     nom_graph=Eco+date+'__graph'+'.png'
L7     savefig(nom_graph)
L8     r=0
```

Explication: r est un incrément que l'on réinitialise tous les 10 tours de boucle.

Donc lorsque que r=10 (L3), on affiche que la sauvegarde des graphes est en cours (L4).

On récupère la date d'aujourd'hui (L5) et crée le nom du graphe avec le caractère économe et la date d'aujourd'hui (L6). Puis on sauvegarde le graphe sous ce nom (L7) et on réinitialise le compteur r (L8).

## IV.2 L'algorithme de régulation.

Voici les fichiers de fonctions importés avec les fonctions dont ils sont porteurs:

Nom des fichiers	<b>fuzzy_econome</b>	<b>fuzzy_temperature</b>	<b>Puissance</b>
Nom des fonctions appartenant au fichier	app_Pas_econome app_Econome app_Tres_econome	app_froid app_bon app_chaud	Chfbcp Chf Continue Refroid
Partie explicative	<i>Partie I: II.2</i>	<i>Partie I: II.2</i>	<i>Partie I: II.2</i>

Nom des fichiers	<b>logique_floue</b>	<b>conversion</b>	<b>sauvegarde</b>
Nom des fonctions appartenant au fichier	ma_logique_floue	conversion	preparation_sauvegarde sauvegarde
Partie explicative	<i>Partie I: II.4</i>	<i>Partie I: III.I.b</i>	<i>Partie I: IV.I</i>

Rappel: Le fonctionnement global a été expliqué dans la Partie I.I.

► **Code couleur utilisé pour l'algorithme de régulation (Python):** en noir *italique* ou **gras** les commentaires, en **vert** ce qui fait référence à la logique floue, en **violet** ce qui fait référence à l'interface Arduino-ordinateur, en **rose** ce qui fait référence à la conversion des nombres binaires en



températures, en **rouge** ce qui fait référence à la sauvegarde de données, en **orange** ce qui fait référence à l'affichage des données sur la console Python, en **marron** ce qui fait référence à l'affichage graphique des données, en **bleu** ce qui fait référence à la conversion de la puissance en un temps de chauffe, en **cyan** ce qui fait référence aux déperditions thermiques, et en noir simple le reste du code.

```
'''importation des bibliothèques connues'''
```

```
from pylab import*
```

```
import serial
```

```
from time import*
```

```
from numpy import*
```

```
'''importation des fonctions créées'''
```

```
from puissance import*
```

```
from fuzzy_temperature import*
```

```
from fuzzy_econome import*
```

```
from conversion import*
```

```
from sauvegarde import*
```

```
from logique_floue import*
```

```
from fichier_val import*
```

```
'''Preparation du fichier de sauvegarde: création du fichier et de l'entete'''
```

```
nom=preparation_sauvegarde(E)
```

```
'''initialisation du programme'''
```

```
Arduino=serial.Serial('COM3',9600,writeTimeout=0.1) #ouverture d'Arduino
```

```
start=0 #temps initial: compteur à 0
```

```
k=0 #tour de boucle
```

```
r=0 #tour de boucle réinitialisé tous les 10 tours
```

```
liste_temperature_int=[0] #liste permettant d'afficher le graphique
```

```
liste_temperature_ext=[0] #liste permettant d'afficher le graphique
```

```
liste_temps=[0] #liste permettant d'afficher le graphique
```

```
liste_Pperdue=[0] #liste permettant d'afficher le graphique
```

```
liste_temps_mult=[0] #liste permettant d'afficher le graphique
```

```
print("initialising")
```

```
grid()
```

```
ylim([-10,50])
```

```
ylabel('Temperature (C)')
```

```
xlabel('temps (s)')
```

```
line1,=plot(liste_temps,liste_temperature_int)
```

```
line2,=plot(liste_temps,liste_temperature_ext)
```

```
ion()
```

```
memo=0 #memorisation de l'horloge
```

```
alt=0 #sert a alterner le ceil et floor pour le calcul du temps_multiple de 10ms
```

```
'''boucle de régulation'''
```

```
while 1:
```

```
    if clock()-start>=(t-0.2): # correspond au 2 sleep(0.1)
```

```
        print "Temps écoulé=",clock()," sur ",Dur,"secondes"
```

```
        print "Carctère économe:", E," Tc=",Tc
```

```
        print "clock()-start=",clock()-start
```

```

o=clock()
print "delta_clock=",clock()-memo
memo=o
if o > Dur: #Dur: duree en seconde de l'experience
    break
try:
    k=k+1
    r=r+1
    print "k=", k, " r=", r
    '''On accede aux temperatures interieures et exterieures'''
    Arduino.write('E') #On demande a Arduino de fournir les temperatures
    sleep(0.1) #j'attends 0.100secondes
    B=Arduino.readline().decode('utf-8')[:-2] #On lit la première ligne sur port série
    Bext=Arduino.readline().decode('utf-8')[:-2] #On lit la deuxième ligne sur le port série
    B=float(B)
    Bext=float(Bext)
    T=conversion(B)
    Text=conversion(Bext)
    print "Tint=",T," Text=",Text

    '''Calcul de la puissance perdue'''
    Pperdue=-A*(Text-Tc)
    print "Pperdue=",Pperdue,"W", " Pmax=",Pmax,"W"
    X=linspace(0.,Pmax,1000.)

    '''cas 1: Pperdue positif'''
    if Pperdue>0: #Lorsque la puissance perdue est positive
        '''cas 1.1: Pperdue positif est supérieur à la puissance du chauffage'''
        if Pperdue>=Pmax:
            #Si la puissance est plus grande que la puissance maximale
            print('Resistance trop faible pour compenser les pertes... ERREUR')
            print('Nous allons donc chauffer à la puissance maximale',Pmax)
            #on chauffe à la puissance maximale ( donc tout le temps )
            temps_multiple=t

        '''cas 1.2: Perdue positif et inferieur à la puissance de la resistance'''
    else:
        # lorsque la puissance est inferieure à la puissance maximale on effectue le
        programme d'algorithme floue
        centre_de_gravite= ma_logique_floue (T, Tc, E, X, Pperdue)
        print "Il faut chauffer a la puissance", centre_de_gravite,"W"
        if centre_de_gravite>=Pmax:
            print "mais cela est trop grand alors on chauffe à ",Pmax,"W"
            centre_de_gravite=Pmax
            pourcentage_de_temps_de_puissance = centre_de_gravite*100/Pmax
            #correspond au pourcentage de temps !
            print"le poucentage de puissance est", pourcentage_de_temps_de_puissance,

            "% "

            temps_de_chauffe = pourcentage_de_temps_de_puissance*t/100.
            #En seconde
            if alt==0 : #alterne le ceil et le floor
                temps_multiple=ceil(temps_de_chauffe/(10*10**(-3)))*10*10**(-3)

```

```

        #on prend le multiple de 10ms
        alt=1
    else : #alterne le ceil et le floor
        temps_multiple=floor(temps_de_chauffe/(10*10**(-3)))*10*10**(-3)
        #on prend le multiple de 10ms
        alt=0
        print "la puissance de chauffe est"
        ,temps_multiple*Pmax/t,"W"

'''cas 2 : Pperdue negatif'''
else:
    print("Erreur, le systeme est un chauffage")
    print('Nous allons donc chauffer à la puissance maximale',Pmax)
    #on ne chauffe pas
    temps_multiple=0

    print "on chauffe durant ",temps_multiple,"secondes"
    temps_mult=str(temps_multiple*1000)
    print"temps_mult=",temps_mult
    Arduino.write('F') #F= je t'envoie les températures
    sleep(0.100)
    Arduino.write(temps_mult)
    start = clock()

    '''mise à jour et tracé des graphiques'''
    xlim([0,k*t]) #cadrage des abscisses
    line1.set_xdata(liste_temps) #mise à jour des abscisses
    line1.set_ydata(liste_temperature_int) #mise à jour des ordonnées
    line2.set_xdata(liste_temps)
    line2.set_ydata(liste_temperature_ext)
    pause(0.001)
    draw() #on retrace le graphique avec les mises à jour

Eco=str(E)+'__'
'''Au bout de 10 boucles on sauvegarde les graphes'''
if r==10: #si p/10 est un entier, cad si p est un multiple de 10
    print "sauvegarde des graph"
    date=strftime('%d_%m_%y__%Hh%Mmin%Ss', localtime())
    nom_graph=Eco+date+'__graph'+'.png'
    savefig(nom_graph)
    r=0
    print "#####"
'''On sauvegarde les valeurs t,T,Text,Perdue'''
sauvegarde (k, nom, Pperdue, liste_Pperdue, T,liste_temperature_int, Text,
            liste_temperature_ext, temps_mult, liste_temps_mult, t, liste_temps)
except Arduino.SerialTimeoutException:
    print("Data could not be read")

Arduino.close()
print("Fin du Programme")

```

# Partie II: Les expériences

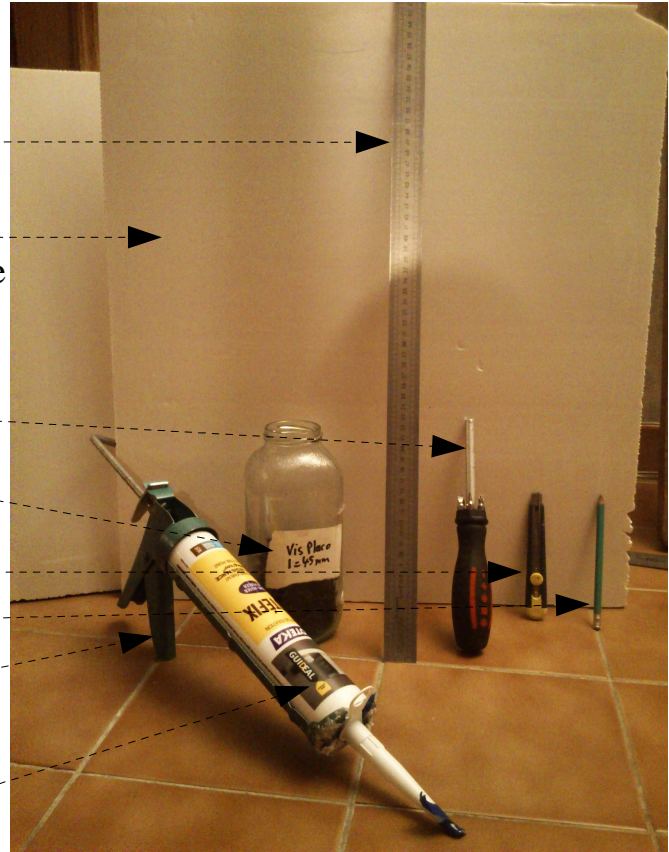
**Introduction:** Les expériences ont été permises par la création d'une enceinte expérimentale peu coûteuse. Il a fallu également procéder à un calcul des déperditions thermiques en fonction des caractéristiques de cette enceinte pour pouvoir faire fonctionner l'algorithme créé au préalable (vue dans la Partie I).

Les données sauvegardées sur l'ordinateur grâce à l'algorithme sous Python ont ensuite été exploitées à l'aide d'Excel.

## I/ L'enceinte expérimentale

### Matériels utilisés:

- une règle
- 4 Panneaux de polystyrène extrudé URSA XPS NW1 de taille 1250x600x20.
- Tounevis
- 24 vis placô de longueur 45mm
- un couteur
- un crayon
- un pistolet à colle
- 1 tube de mastic de fixation Cotéfix 310ml



### Protocole:



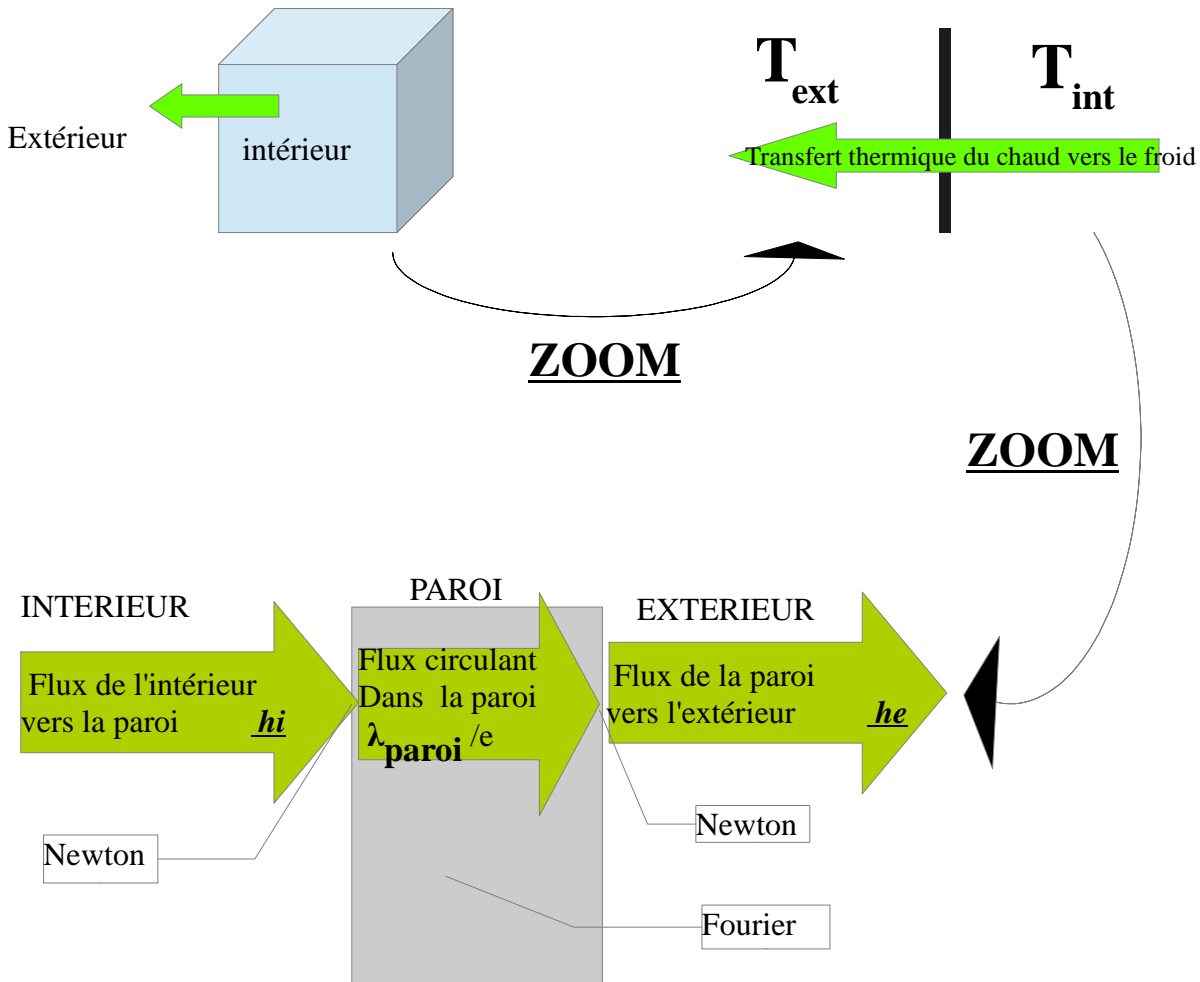
1. On **coupe** soigneusement 6 planches de polystyrène de 60x60x60cm.
2. On **colle** avec le mastic 4 planches ensemble (1 tranche sur le côté d'une face) pour former les côtés du cube.
3. On les **maintient** en place grâce à des vis auto-perçantes.
4. On **colle et visse** la planche correspondant au plafond du cube.
5. On rajoute du mastic dans les coins et sur les bords pour qu'il n'y ait pas de trou.

6. On **pause** délicatement le tout sur la planche du bas et on **attend** 48 heures.

Au final, on a un cube dont une face peut être retirée. Cette face amovible sert à placer la résistance chauffante, son alimentation, et le capteur de température intérieure.

## II/ Le calcul des déperditions thermiques

Voici la situation dans laquelle on est: nous avons une paroi d'épaisseur  $e$ , qui sépare l'air (=fluide) à la température  $T_{int}$  et l'air à la température  $T_{ext}$ .



On néglige les ponts thermiques et on suppose que la température est uniforme à l'intérieur et à l'extérieur de l'enceinte.

Pour calculer la déperditions thermiques; on a besoin:

- des coefficients de transfert thermique de surface, ou coefficient d'échange superficiel ( $h_i$  et  $h_e$ ). Ils sont obtenus (en fonction de l'inclinaison des parois) via les valeurs de résistance superficielle  $1/h_i$  et  $1/h_e$  données par le site <http://www.techniques-ingenieur.fr>
- de la conductivité thermique des parois,  $\lambda_{pol} = 1/R$  avec  $R$  la résistance thermique que l'on retrouve dans les caractéristiques fournies par le site [www.ursa.fr](http://www.ursa.fr)

On compte calculer un ordre de grandeur des pertes thermiques objectives lorsque la résistance chauffante est éteinte ( ni puit ni source ).

Pour cela on va appliquer la loi phénoménologique de Fourier:  $\vec{j}_{th} \geq - \lambda_{pol} \vec{\text{grad}} T$ , et la loi de Newton

(cas d'un contact avec un fluide) et on utilise l'égalité suivante:

$$P_{\text{perdu}} = \int (j \cdot dS) \\ = \text{somme des } j \cdot S \text{ pour chaque paroi}$$

On simplifie l'écriture en posant A, une constante homogène à des  $\text{W.K}^{-1}$ , telle que  $P_{\text{perdu}} = A(T_{\text{ext}} - T_c)$

$$P_{\text{perdu}} = A(T_{\text{ext}} - T_c)$$

Température intérieure (en K) →  $T_c$   
 Température extérieure (en K) →  $T_{\text{ext}}$   
 $\text{W.K}^{-1}$  → A  
 Puissance perdue par transfert d'énergie via les parois (en W) →  $P_{\text{perdu}}$

$$A = \frac{S_{\text{verticales}}}{\left( \frac{1}{h_{i\text{verticale}}} + \frac{1}{h_{e\text{verticale}}} + \frac{e}{\lambda_{\text{pol}}} \right)} + \frac{S_{\text{horizontale}}}{\left( \frac{1}{h_{i\text{horizontale ascendante}}} + \frac{1}{h_{e\text{horizontale ascendante}}} + \frac{e}{\lambda_{\text{pol}}} \right)} + \frac{S_{\text{horizontale}}}{\left( \frac{1}{h_{i\text{horizontale descendante}}} + \frac{1}{h_{e\text{horizontale descendante}}} + \frac{e}{\lambda_{\text{pol}}} \right)}$$

Les données sont récapitulées dans ce tableau:

La résistance thermique	<b><math>R = 0,60 \text{ m}^2 \cdot \text{K} \cdot \text{W}^{-1}</math></b>
les résistances superficielles	<b><math>R_{si\_vertic} = 0.11 \text{ W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}</math>  <math>R_{se\_vertic} = 0.11 \text{ W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}</math>  <math>R_{si\_horiz\_ascend} = 0.09 \text{ W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}</math>  <math>R_{se\_horiz\_ascend} = 0.09 \text{ W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}</math>  <math>R_{si\_horiz\_descend} = 0.17 \text{ W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}</math>  <math>R_{se\_horiz\_descend} = 0.17 \text{ W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}</math></b>
L'épaisseur des parois	<b><math>e = 0,02 \text{ m}</math></b>
La surface d'une paroi en contact avec l'air	<b><math>S = (L - 2 \cdot e)^2 = (0,6 - 2 \cdot 0,02)^2 = 0,56^2 = 0,3136 \text{ m}^2</math></b>

Application numérique:

$$A = 2,2653 \text{ W} \cdot \text{K}^{-1}$$

En réalité les données sont stockées dans le fichier nommé **fichier\_val** où le calcul de A est également effectué.

### III/ Les résultats

#### III.1 Les 10 expériences

Les expériences ont été réalisées sur environ 1h avec l'enceinte expérimentale décrite précédemment, avec un résistor de faible inertie, de résistance  $R = 1000 \text{ Ohms}$ , alimenté par le secteur (tension  $U = 230 \text{ V}$ ). Sa puissance est donc  $P_{\text{max}} = U^2 / R = 53 \text{ W}$ .

Celle-ci est en réalité calculée dans le **fichier\_val** où sont stockées les valeurs de R et U. La température de consigne était de  $26^\circ \text{C}$  pour chaque expérience.



Une expérience a été réalisée pour chaque caractère économe, les résultats obtenus sont les courbes de l'évolution de la température intérieure en fonction du temps (cf Annexes). Les grandeurs caractéristiques de chaque expériences sont les suivantes:

Caractère économe (note /10)	Retard (+ ou - 2s)	Temps de montée (au bout duquel la température $T_c = 26^{\circ}\text{C}$ est atteinte)	Dépassement maximal / temps associé (+ ou - 0,1°C)		Température de départ (+ ou - 0,1°C)
0	74s	526s	1,5°C	725s	19,2°C
1	80s	568s	1,6°C	772s	19°C
2	94s	574s	1,5°C	784s	18,6°C
3	98s	574s	1,4°C	794s	19°C
4	100s	764s	0,9°C	838s	18,1°C
5	110s	1012s	0,4°C	1158s	18,1°C
6	124s	1142s	0,4°C	1308s	18,8°C
7	132s	2502s	0,3°C	2894s	18,7°C
8	108s	2528s	0,1°C	2904s	18,2°C
9	162s	2470s	0,2°C	2994s	18,8°C
10	180s	2472s	0,1°C	2650s	18,3 °C

**Observation1:** Le retard augmente progressivement avec le caractère économe:  
Ex: Il a augmenté de plus de 100s lorsqu'on est passé de  $E=0$  à  $E=10$ .

**Observation2:** Le temps de montée est d'autant plus long que le caractère économe est élevé:  
Ex: Il est multiplié par 2 de  $E=0$  à  $E=5$  et par 5 entre  $E=0$  et  $E=10$ .

**Observation3:** Le dépassement décroît Avec la note du caractère économe:  
Ex: il est divisé par 5 lorsqu'on passe de  $E=0$  à  $E=5$ .  
**Observation4:** le dépassement pour  $E=10$  correspond au pas de mesure du capteur de température.

**Analyse1:** le temps de réaction du système est d'autant plus long que l'utilisateur est économe (on chauffe moins fort au début)

**Analyse2:** plus l'utilisateur est économe Plus le système est lent (moins on chauffe par tour de boucle)

**Analyse 3:** plus l'utilisateur est économe, moins d'énergie sera "gachée"

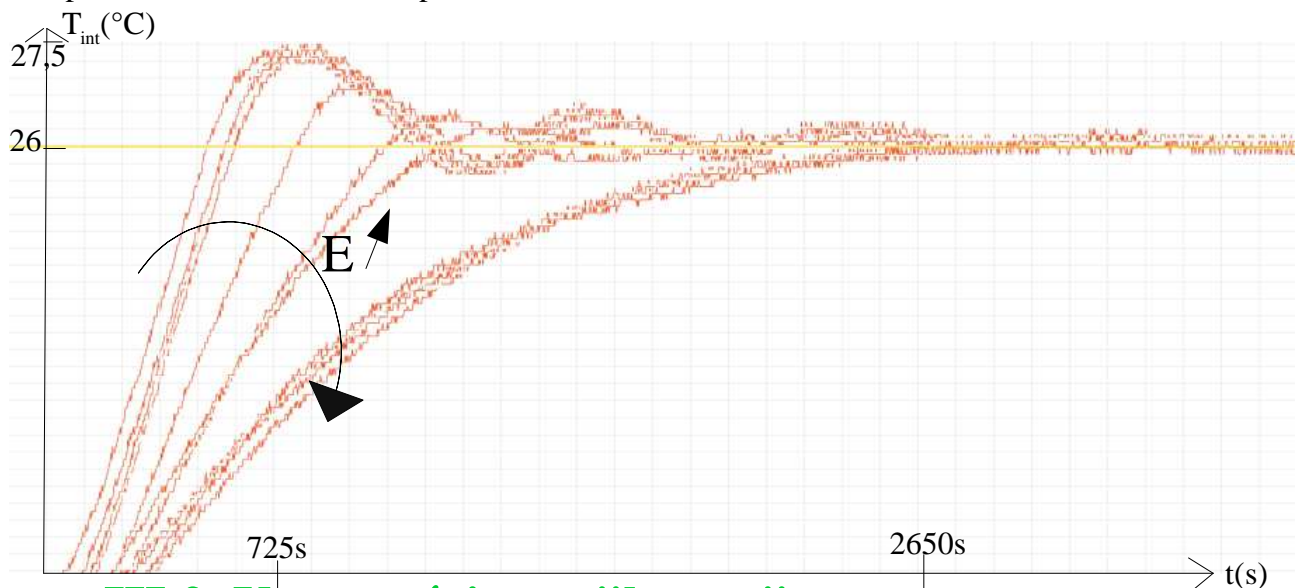
Mais on observe quelques valeurs particulières ne correspondant pas à l'observation générale. Ces petits défauts peuvent être expliqués par les limites de l'expérience: la température extérieure n'a pas été maîtrisée, c'est pourquoi ses variations et sa valeur de départ diffèrent selon les expériences. On observe sur le tableau ci-dessous que les **températures de départ** appartiennent à l'intervalle  $[18,1^{\circ}\text{C}; 19,2^{\circ}\text{C}]$  il y a donc un écart maximal de **1,1°**. Cependant, la **température extérieure moyenne** appartient à l'intervalle  $[18^{\circ}\text{C}; 18,7^{\circ}\text{C}]$  donc avec un écart maximal de **0,7°**. Alors que les **variations de la température extérieure** sont tout de même très proches: elles appartiennent à l'intervalle  $[0,4^{\circ}; 0,7^{\circ}]$ , un écart maximal de **0,3°** de différence de variation a été observé.

Le retard peut être modifié si le résistor servant de radiateur a été **précédemment allumé**. Ces résultats peuvent d'ailleurs justifier les petits décalages observés (**en gras souligné dans le tableau**)

précédent)

Caractère économe (note /10)	Température de départ (+ ou - 0,1°C)	Température extérieure moyenne (+ ou - 0,1°C au maximum)	Variation de température extérieure (+ ou - 0,1°C)
0	19,2°C	18,7°C	0,6°
1	19°C	18,6°C	0,6°
2	18,6°C	18,3°C	0,6°
3	19°C	18,5°C	0,5°
4	18,1°C	18°C	0,7°
5	18,1°C	18,1°C	0,6°
6	18,8°C	18,6°C	0,4°
7	18,7°C	18,4°C	0,4°
8	18,2°C	18,2°C	0,6°
9	18,8°C	18,4°C	0,6°
10	18,3 °C	18,7°C	0,7°

Voici l'allure de la superposition des courbes de régulation (observable une par une dans l'Annexe).  
Température en fonction du temps:



### III.2. Une expérience "bonus"

Afin de vérifier que le système fonctionne également lors d'un fort changement de température, une expérience commençant à une température intérieure de 5,1° (toujours avec une température de consigne de 26°C) et augmentant jusqu'à 17,5°C a été réalisée. Le caractère économe choisi était de 6. Voici les grandeurs obtenues:

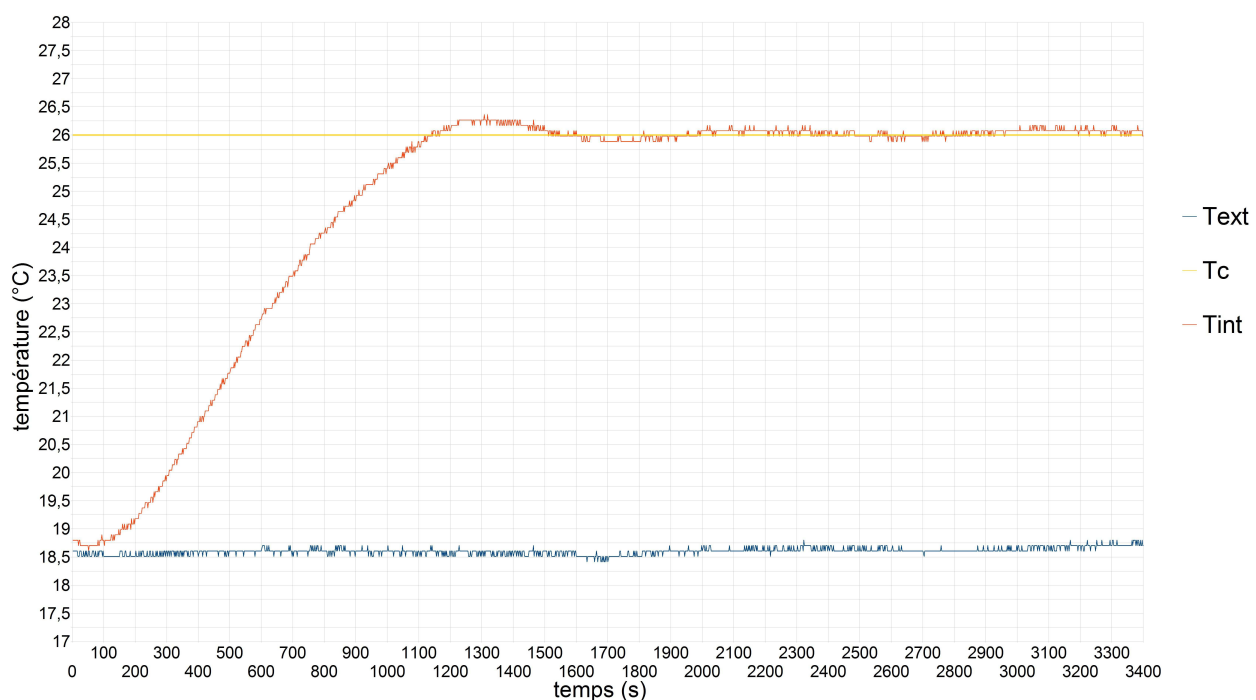


Caractère économe (note /10)	Retard (+ ou - 2s)	Température de départ (+ ou - 0,1°C)	Température extérieure moyenne (+ ou - 0,1°C au maximum)	Variation de température extérieure (+ ou - 0,1°C)	Temps de montée (au bout duquel la température $T_c = 26^\circ\text{C}$ est atteinte)	Dépassement maximal / temps associé (+ ou - 0,1°C)
6	76s	5,1°C	8,0°C	12,4°	4312s	0,3°C

Pour mieux comparer les résultats, voici la courbe de régulation de la première expérience pour  $E=6$ , puis celle de l'expérience bonus ( à la page suivante ) :

#### Evolution temporelle de la température

$E=6$   $T_c=26^\circ\text{C}$



#### ★ Remarque 1: Le retard.

On remarque que le retard est beaucoup plus faible que dans l'expérience précédente: cela peut être dû au fait qu'il faut chauffer beaucoup plus que dans la 1<sup>ère</sup> expérience, et que la température extérieure augmente alors que dans la première expérience la température extérieure est constante au début de l'expérience.

#### ★ Remarque 2: Le temps de montée

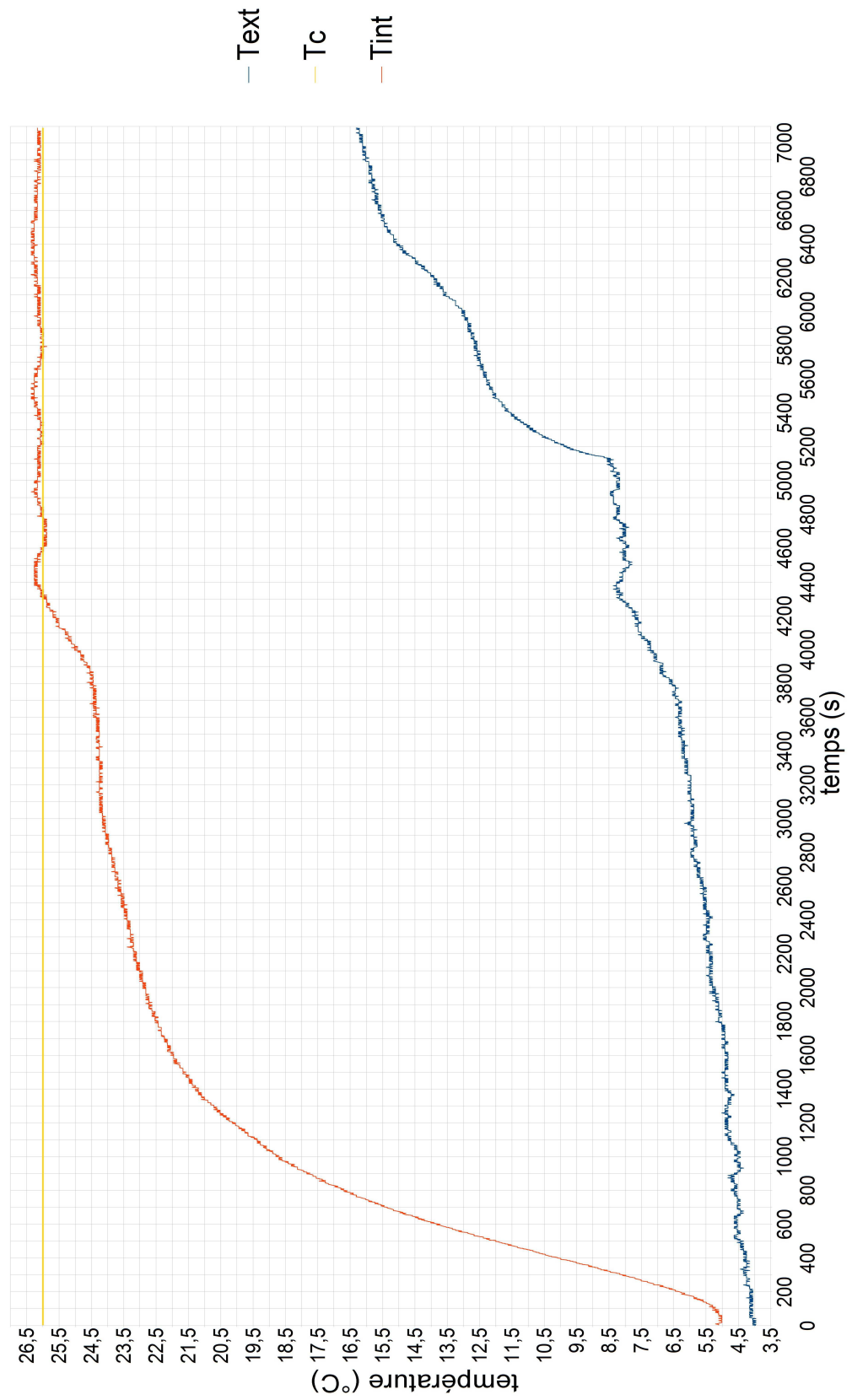
Le temps de montée est deux fois plus élevé dans cette expérience que dans la 1<sup>ère</sup> expérience. Ceci reste cohérent car la variation de température intérieure doit être largement supérieure (de  $7,2^\circ$  pour l'expérience 1 contre  $20,9^\circ$  pour l'expérience 2). Cependant il ne faut pas négliger l'augmentation de la température extérieure pour l'expérience 2.

#### ★ Remarque 3: Le dépassement

Le dépassement maximal s'effectue juste après l'atteinte de la température de consigne. Ce dépassement est identique dans les 2 expériences !

## Evolution temporelle de la température

E=6 Tc=26°C



### **Conclusion:**

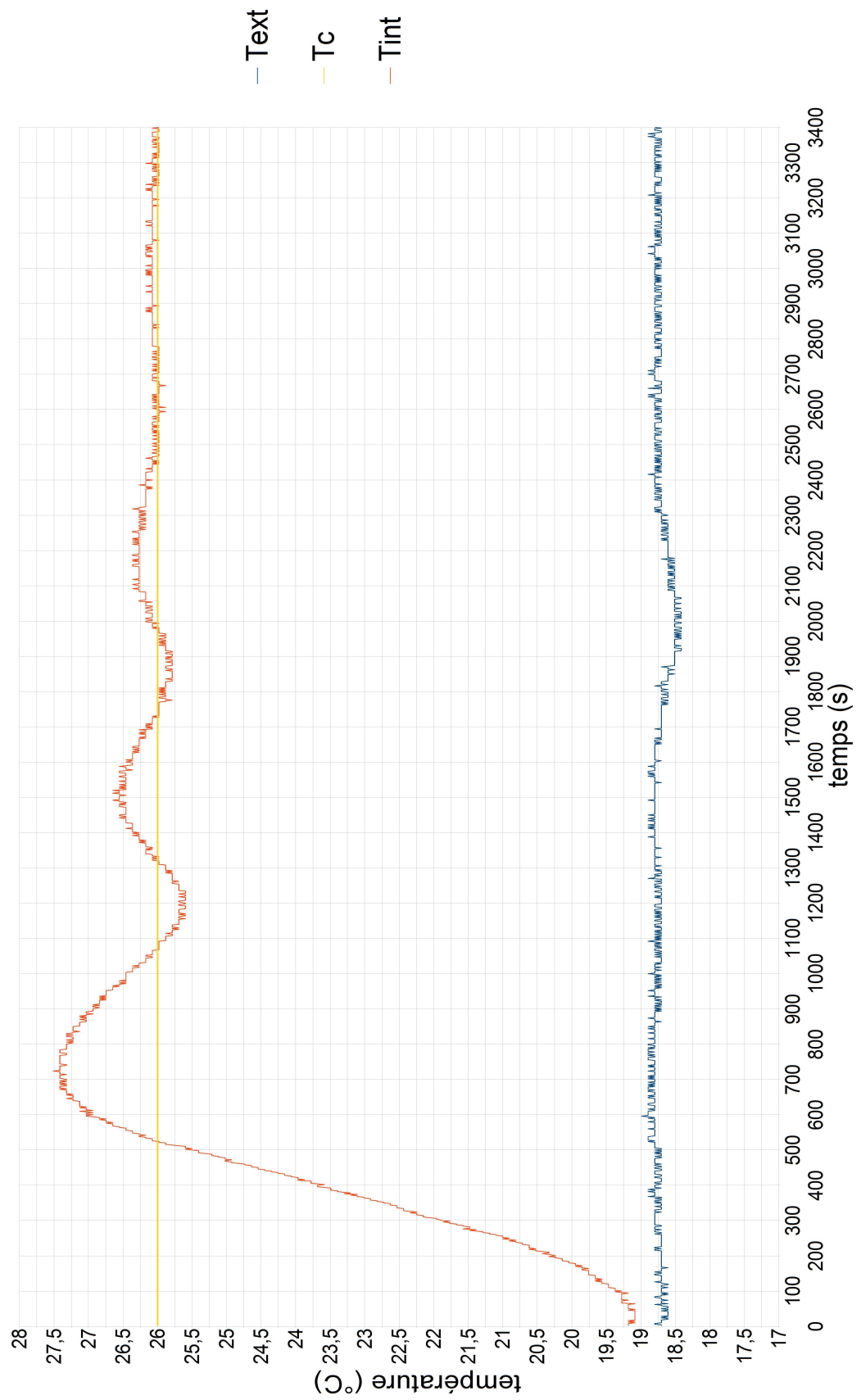
D'autres expériences n'ont pu être faites: il aurait été pertinent de reproduire l'expérience 2 pour les autres caractères économes, mais ceci est bien trop difficile avec les moyens du bord et le temps à disposition. J'ai tout de même fait le choix d'exposer cette expérience car elle met en évidence la possibilité de réguler dans des conditions inhabituelles.

On aurait également pu réaliser des expériences avec différents types d'enceinte, faire en sorte que le système soit "auto adaptatif" et créer une interface graphique pour simplifier l'utilisation du système (à l'aide de Tkinter par exemple).

### **Annexe:** Les courbes obtenues lors des 10 premières expériences

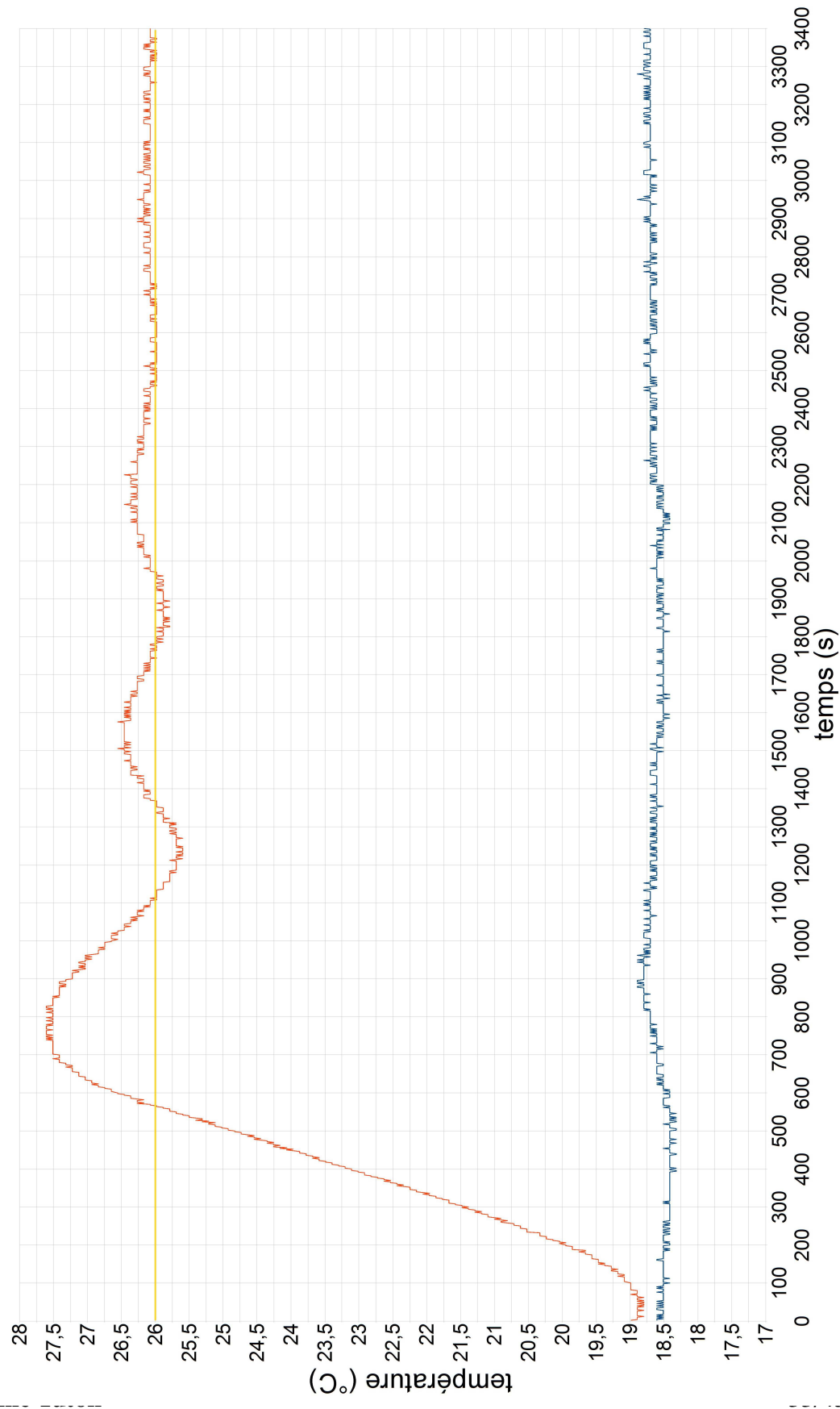
## Evolution temporelle de la température

$E=0$   $T_c=26^\circ\text{C}$



## Evolution temporelle de la température

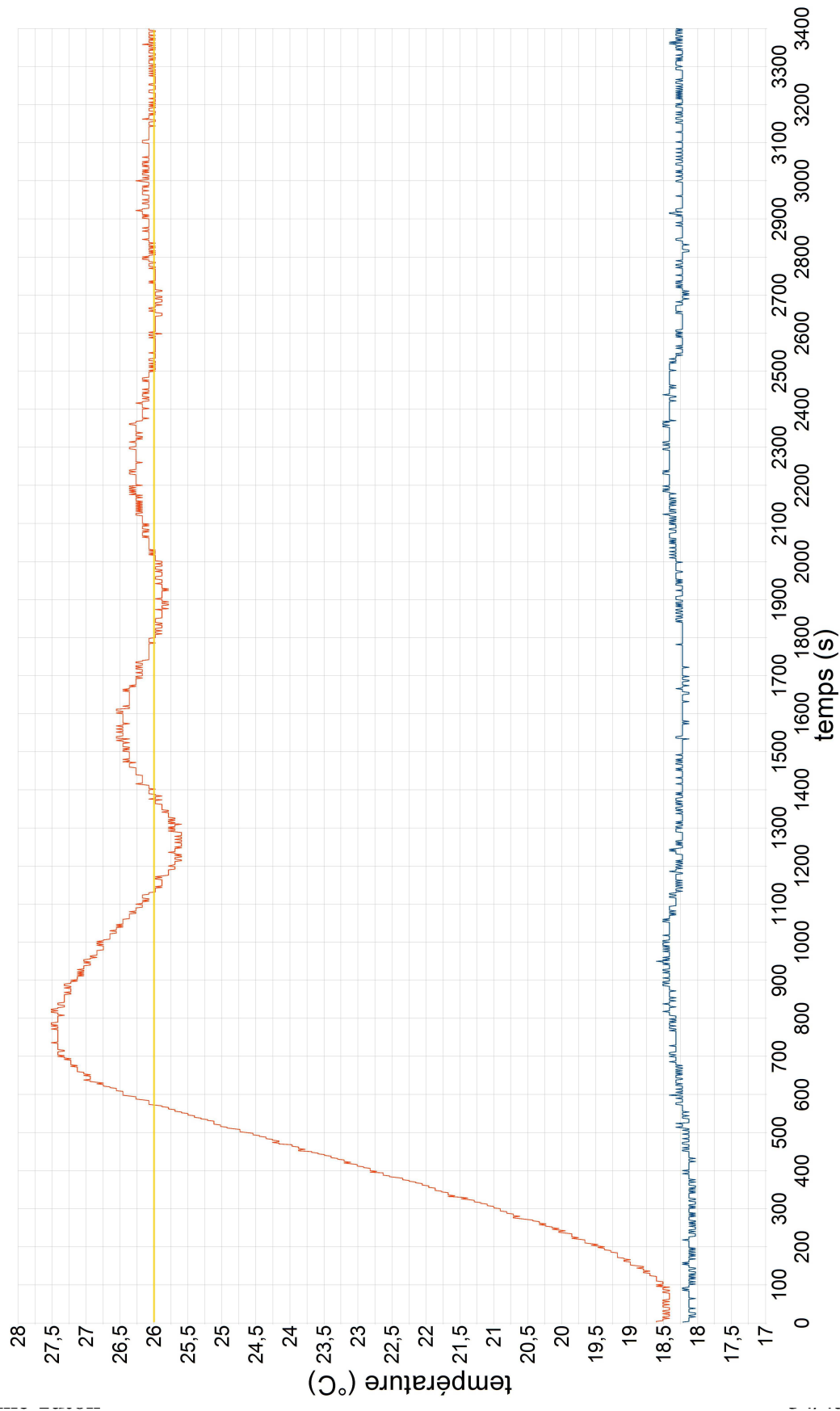
E=1 T<sub>c</sub>=26°C



— Text  
— T<sub>c</sub>  
— T<sub>int</sub>

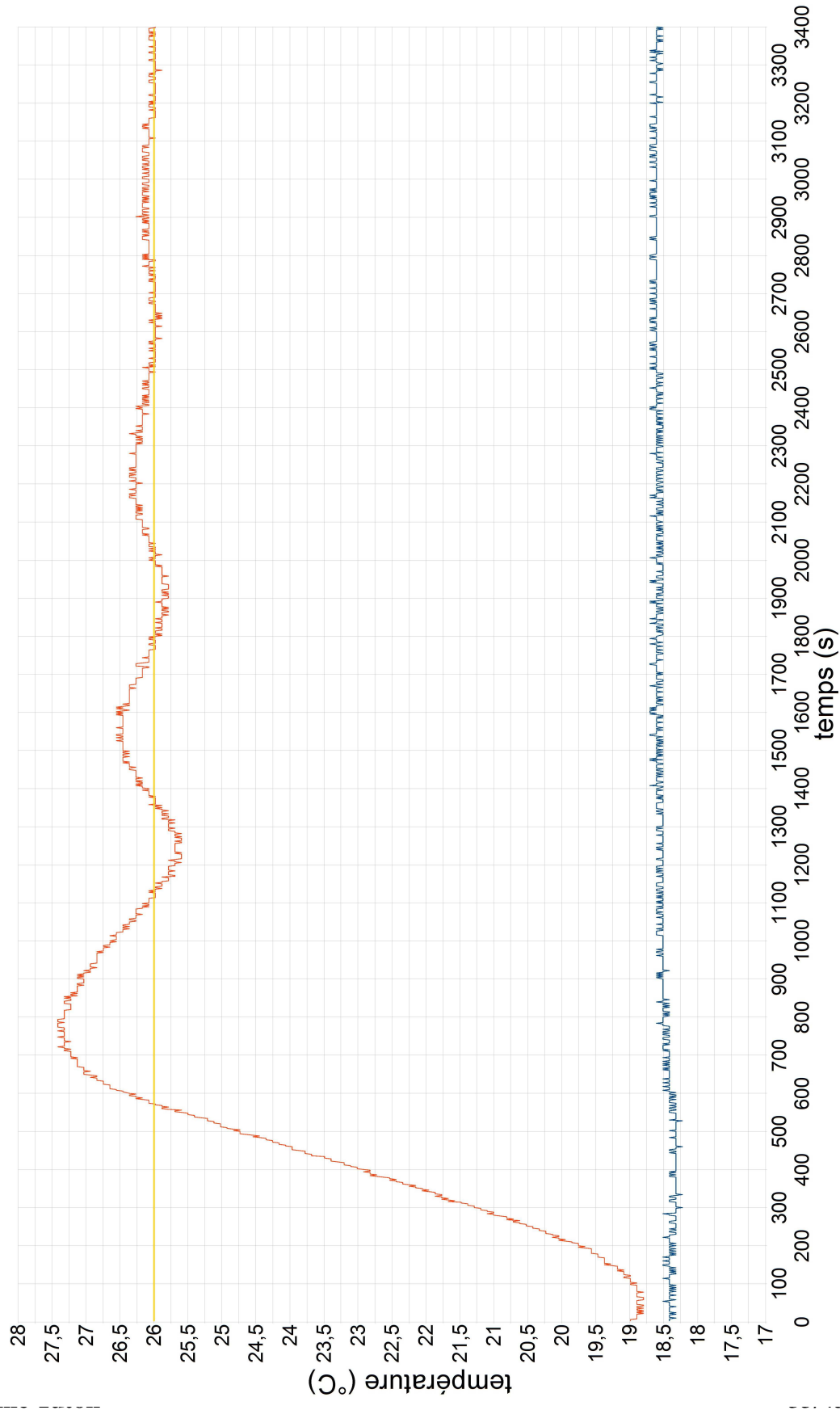
## Evolution temporelle de la température

E=2 Tc=26°C



## Evolution temporelle de la température

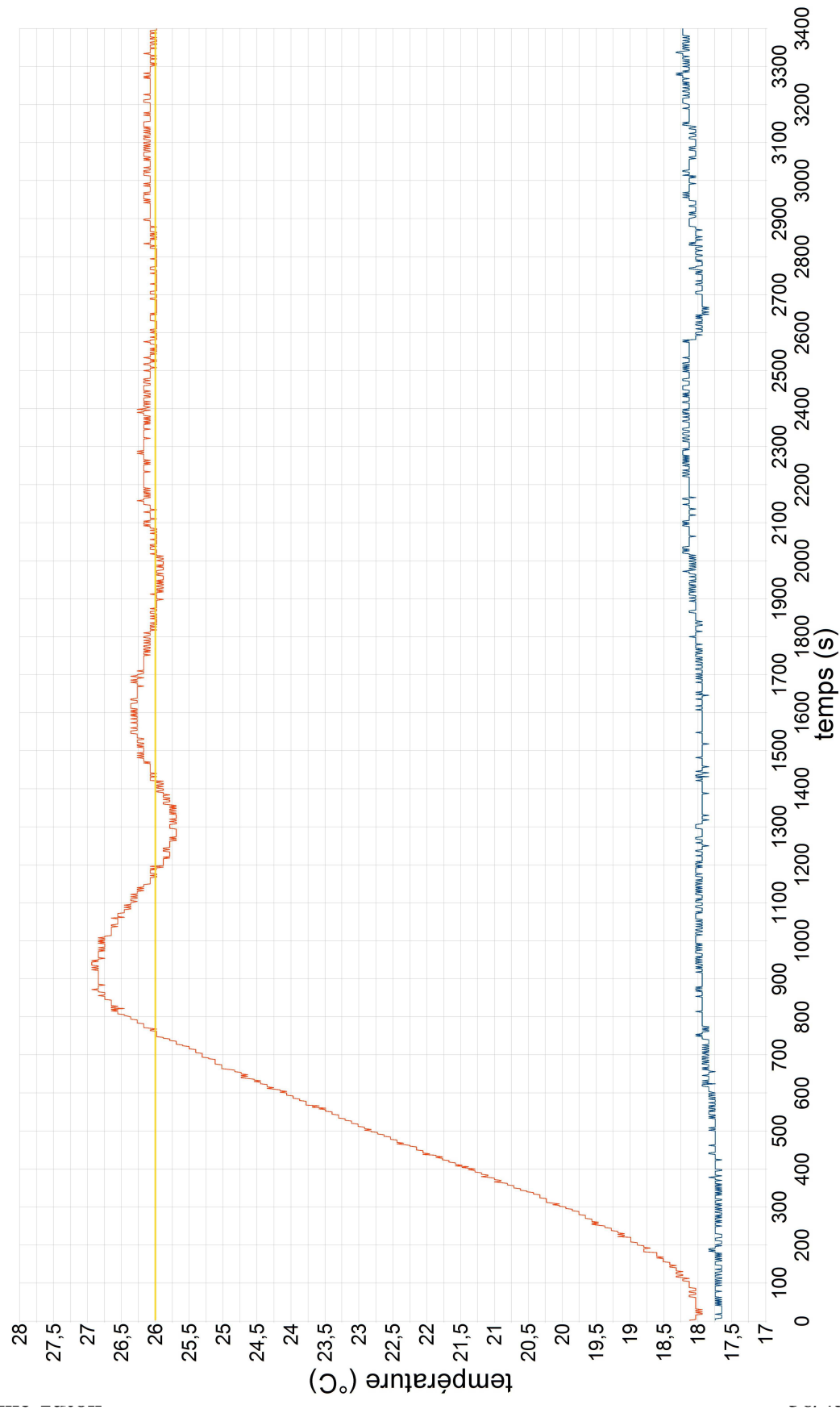
E=3 T<sub>c</sub>=26°C



— Text  
— T<sub>c</sub>  
— T<sub>int</sub>

## Evolution temporelle de la température

E=4 Tc=26°C



— Text

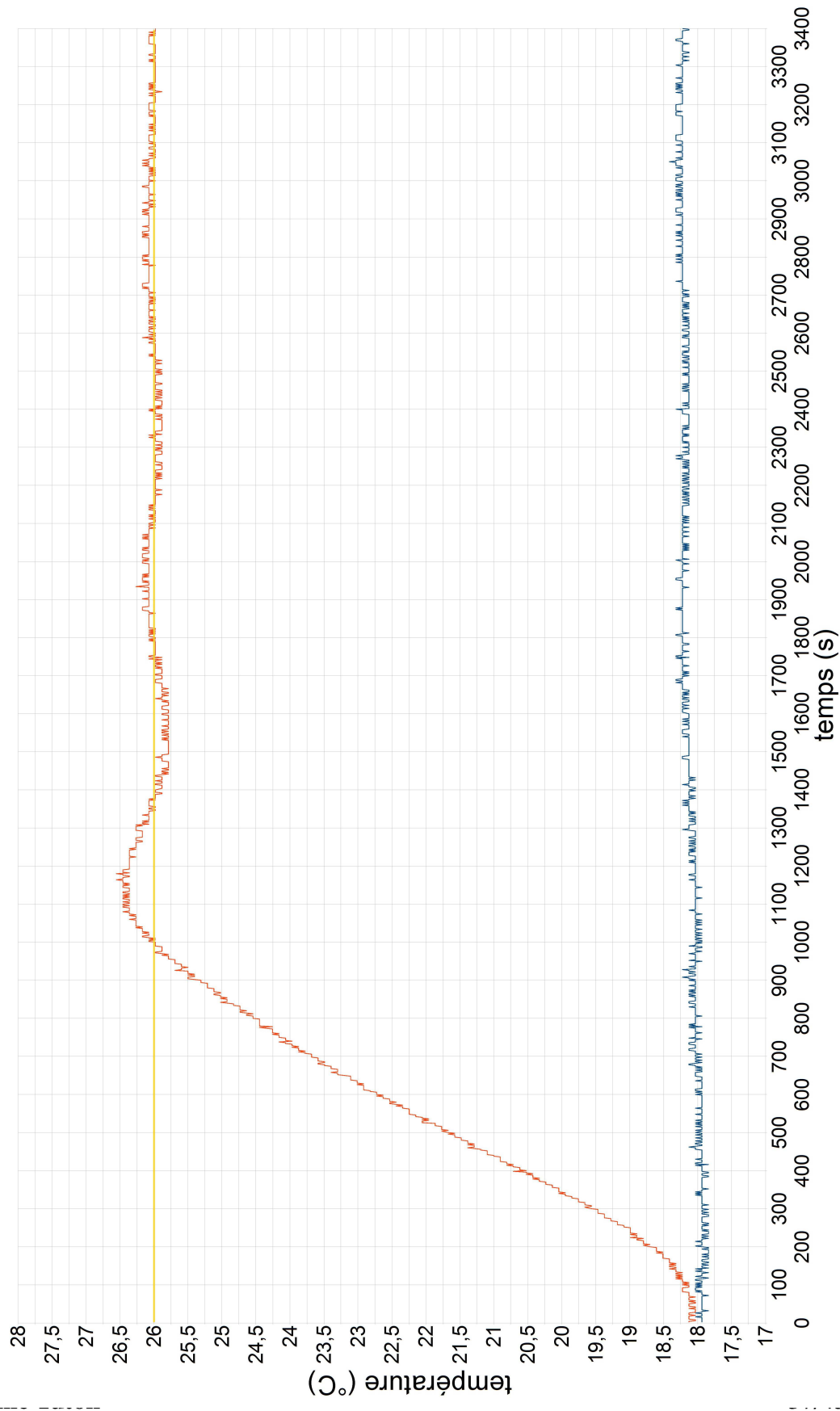
— Tc

— Tint



## Evolution temporelle de la température

E=5 Tc=26°C



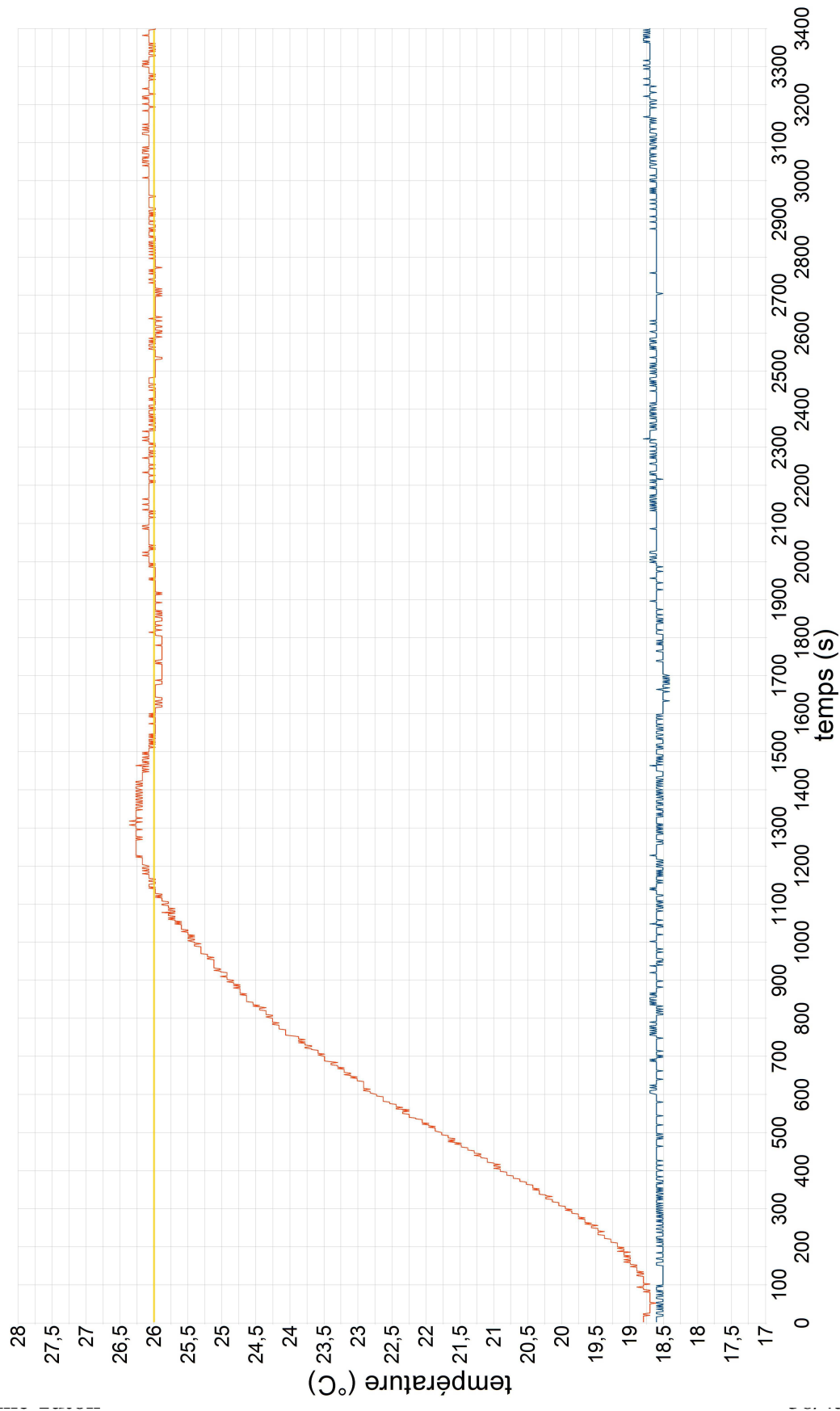
— Text

— Tc

— Tint

## Evolution temporelle de la température

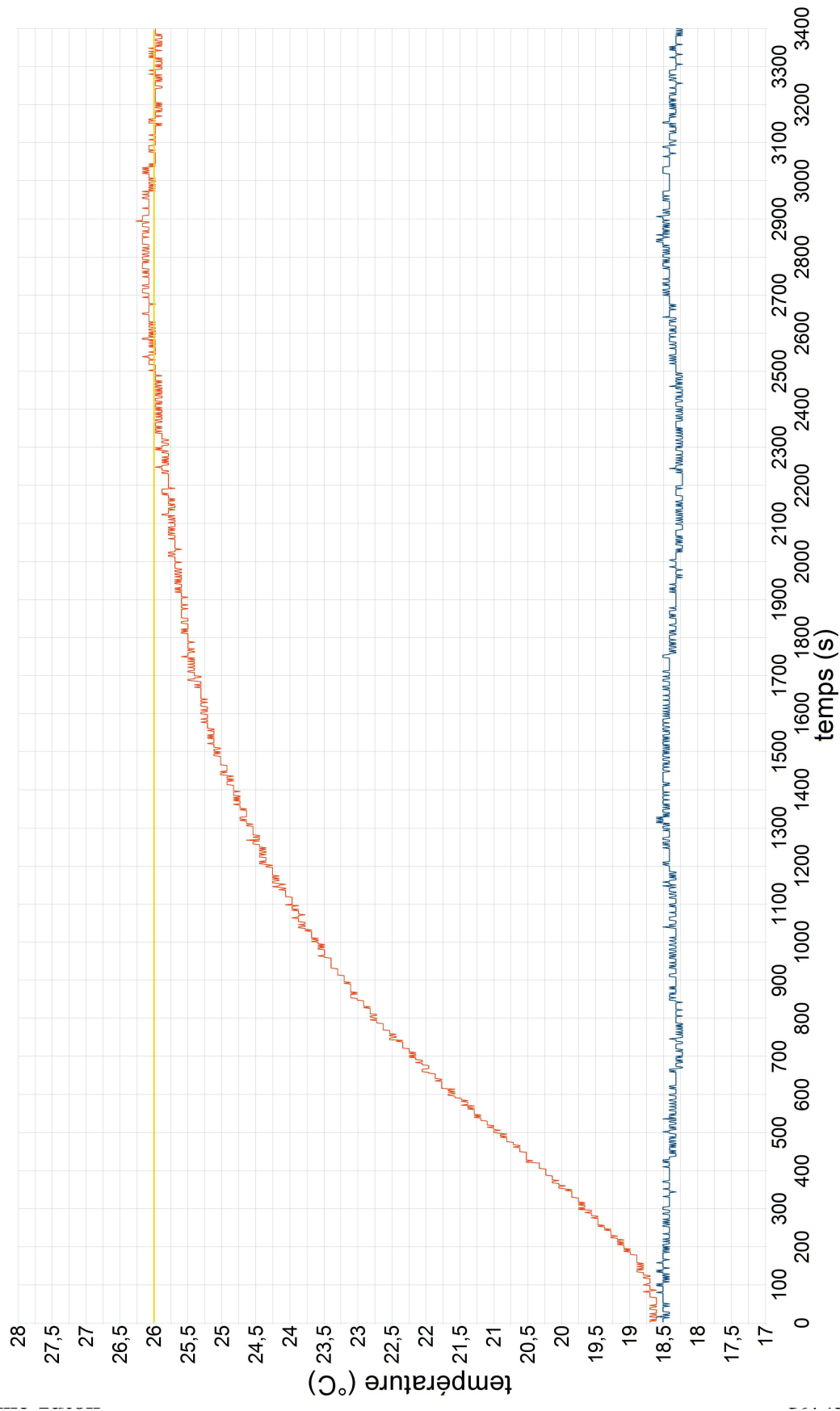
E=6 T<sub>c</sub>=26°C



— Text  
— T<sub>c</sub>  
— T<sub>int</sub>

## Evolution temporelle de la température

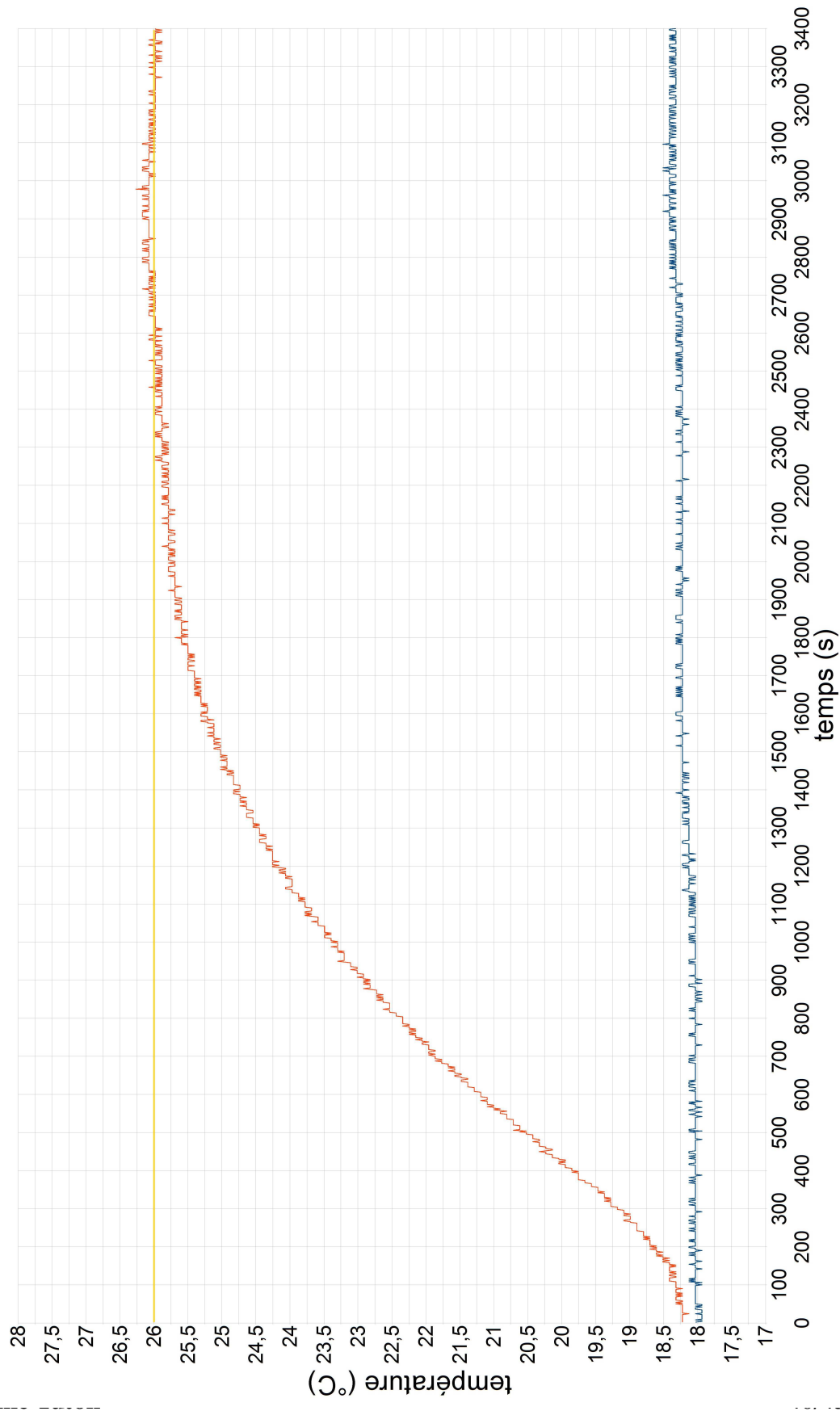
E=7 Tc=26°C



— Text  
— Tc  
— Tint

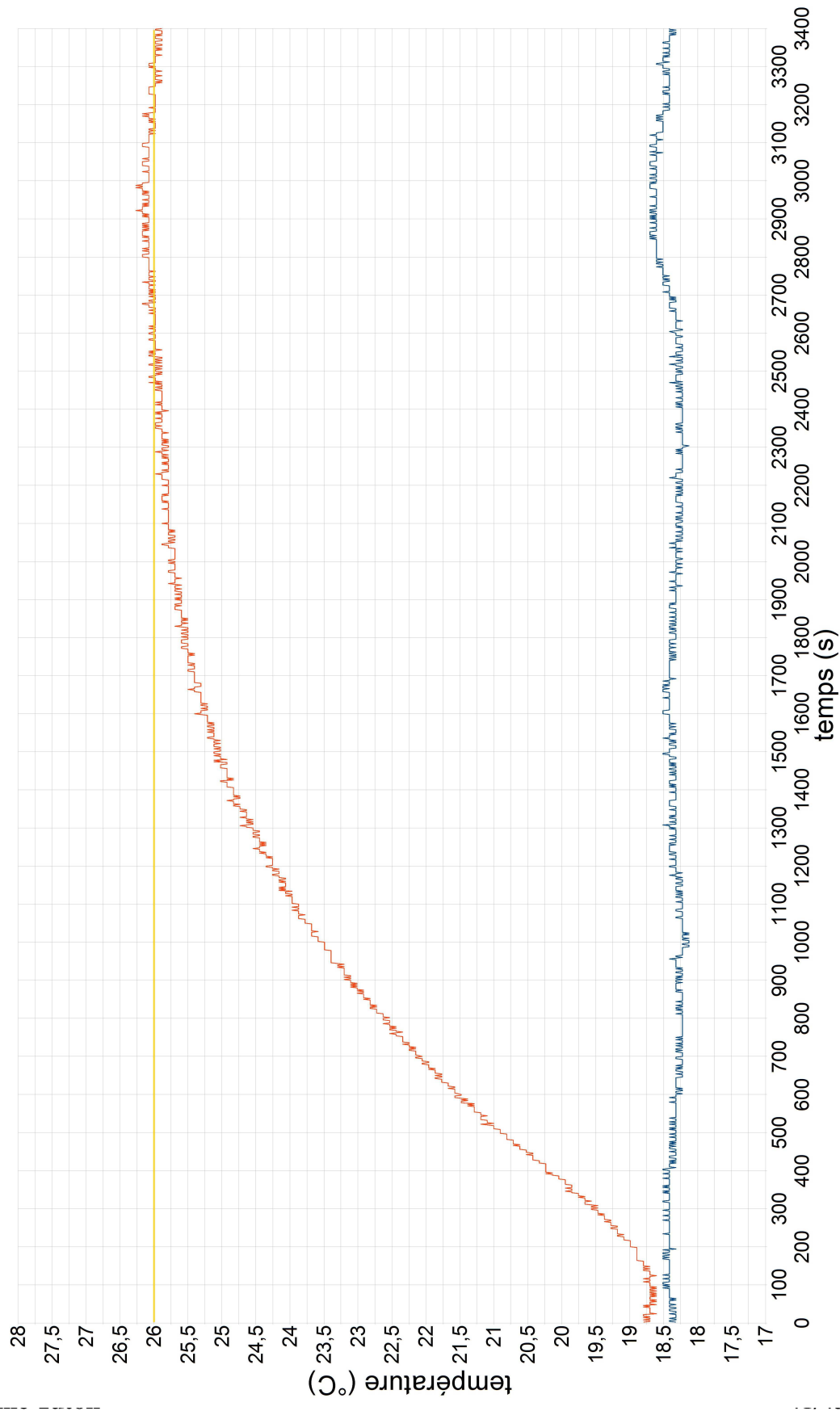
## Evolution temporelle de la température

E=8 Tc=26°C



## Evolution temporelle de la température

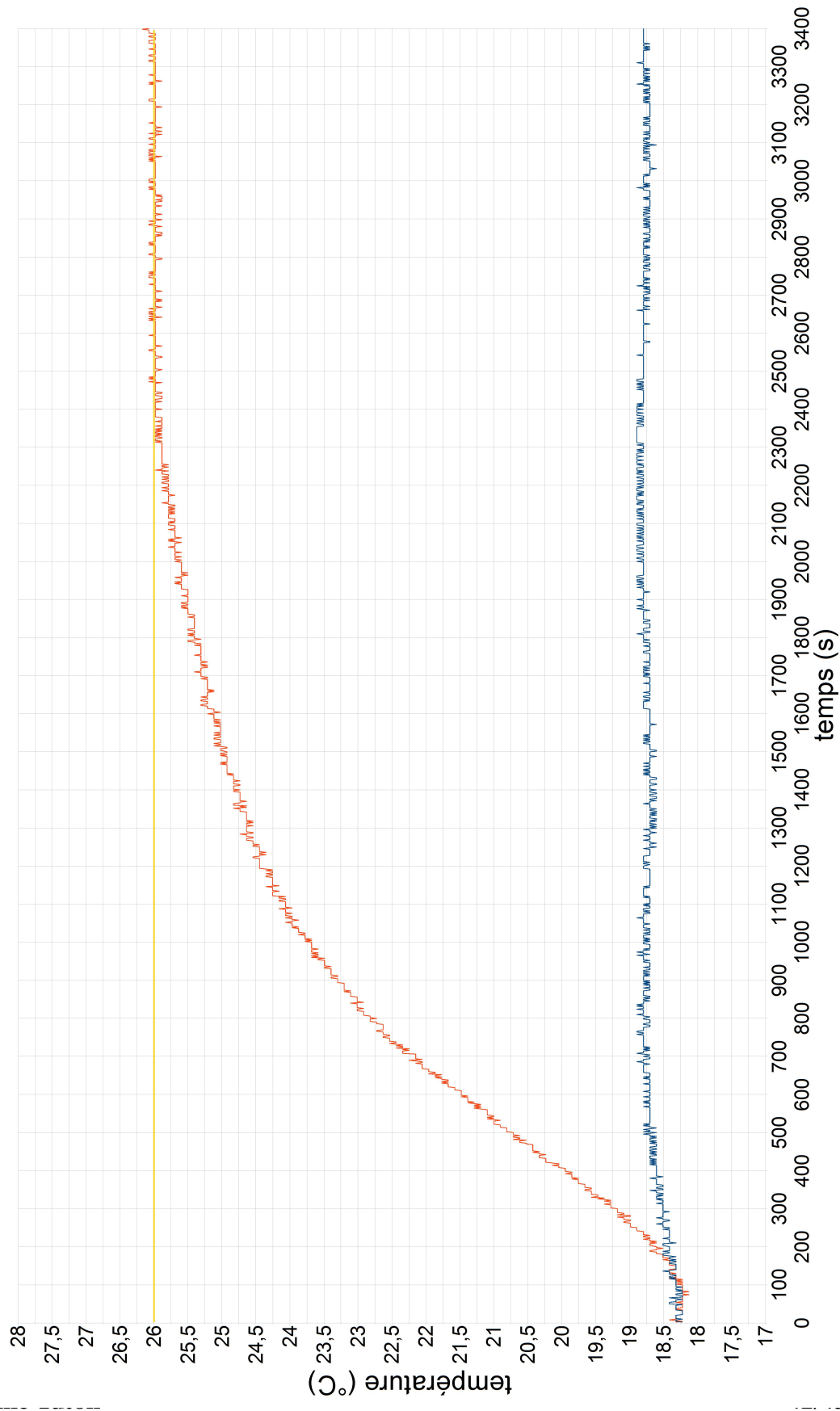
E=9 Tc=26°C



— Text  
— Tc  
— Tint

## Evolution temporelle de la température

E=10 Tc=26°C



— Text  
— Tc  
— Tint

## Bibliographie

- Antoine Cornuéjols 2013, Cours d'introduction à la logique floue (à AgroParisTech). Disponible sur le Web: <https://www.lri.fr/~antoine/Courses/AGRO/Cours-logique/Cours-IA-fuzzy-logic-2013x4.pdf>
- Cours d'introduction à la logique floue par Franck Dernoncourt. Disponible sur le Web: <http://fr.openclassrooms.com/informatique/cours/introduction-a-la-logique-floue>
- François CHEVRIE et François GUÉLY ,Cahier technique Schneider n° 191 *La logique floue*, Collection technique du Groupe Schneider, 1998 Disponible sur le Web: <http://www2.schneider-electric.com/documents/technical-publications/fr/shared/automatismes/automatismes-reseaux-information/ct191.pdf>
- Patrick REIGNIER, Thèse présentée à l'Institut National Polytechnique de Grenoble: *Pilotage réactif d'un robot mobile, étude du lien entre la perception et l'action*, Chapitre 5: *Le contrôle flou*. Disponible sur le Web: <http://tel.archives-ouvertes.fr/docs/00/04/63/77/PDF/tel-00005108.pdf>
- <http://www.ursa.fr/fr-fr/produits/ursa-xps/ursa-xps-niiii-et-nwi/pages/infos.aspx> (informations sur les caractéristiques du matériel isolant utilisé pour la réalisation de l'enceinte expérimentale)
- <http://www.techniques-ingenieur.fr/base-documentaire/construction-th3/pathologie-de-l-humidite-des-parois-pathologie-des-ponts-42241210/pathologie-de-l-humidite-paroi-simple-c7137/coefficients-d-echange-superficiel-ou-de-resistance-superficielle-c7137niv1> (pour obtenir les coefficients d'échange superficiel pour le calcul des pertes thermiques)
- <http://www.ti.com/lit/ds/snis160e/snis160e.pdf> (informations sur le capteur LM335A)
- G.Séguier , *L'électronique de puissance* , chapitre 5 Les gradateurs, Dunod technique, 1979 ( pour la réalisation de commutation de puissance à base de triacs )
- Louis Reynier, "C'est quoi Arduino ?" Disponible sur le Web: <http://www.louisreynier.com/fichiers/KesacoArduino.pdf>

Excel / ExpressPCB / Arduino / Spyder