Institut Supérieur de l'Electronique et du Numérique

PROJET A3 – TRONC COMMUN

2022-2023

SUPERVISION ET REGULATION DE LA TEMPERATURE D'UNE HABITATION

- DESCRIPTION DETAILLEE -

Version 12.0 mise à jour le vendredi 3 mars 2023

BREST :	P.-J. BOUVET	Pierre-jean.bouvet@isen-ouest.yncrea.fr
	J.-B. PIERROT	jean-benoit.pierrot@isen-ouest.yncrea.fr
	F. LEGRAS	legrasf@me.com
	H. BENALLAL	hafsa.benallal@isen-ouest.yncrea.fr
NANTES :	N. BEAUSSE	nils.beausse@isen-ouest.yncrea.fr
CAEN :	N. ABDALLAH SAAB	nadine.abdallah-saab@isen-ouest.yncrea.fr
	S. LE GLOANNEC	simon.le-gloannec@isen-ouest.yncrea.fr

TABLE DES MATIERES

1 INTRODUCTION

La société "PEDAGIX SOFT", spécialisée dans le développement de logiciels pédagogiques, vous demande de réaliser un pupitre de commande sur micro-ordinateur pour le système conçu par la société "PEDAGIX". Ce pupitre de commande devra mettre en œuvre les fonctionnalités suivantes :

- Relevé des températures intérieure et extérieure via une liaison USB
- Visualisation de la température intérieure et extérieure
- Régulation de la température intérieure basée sur une température de consigne en fonction de différentes lois :
  - Tout ou rien
  - Régulateur PID
- Envoi de commande de chauffage au système (déduit du système de régulation)
- Visualisation du témoin de chauffe

La société "PEDAGIX SOFT" ne possède pas les modules de simulations "mini-environnement" et "échanges thermiques" de la société "PEDAGIX". Elle devra donc mettre en place un système permettant de valider ses propres développements.

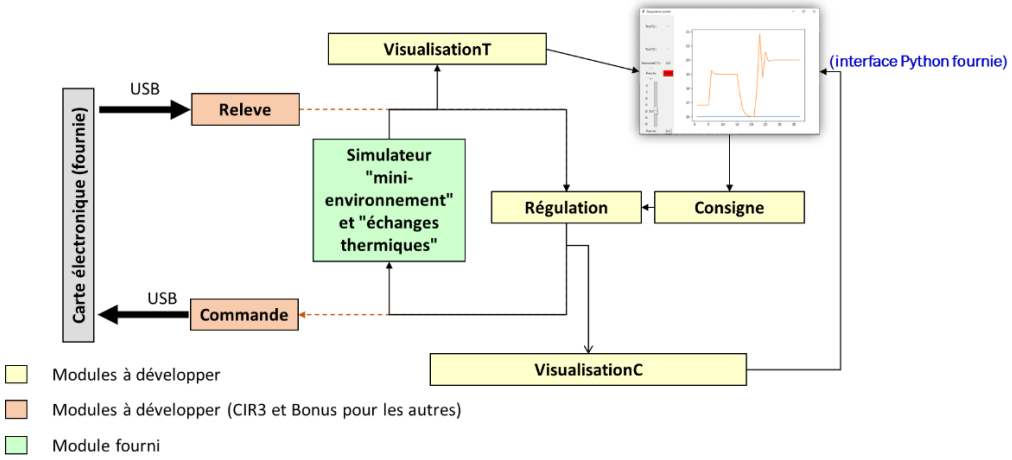


FIGURE 1 : SYNOPTIQUE FONCTIONNEL

2 DESCRIPTION DES BLOCS

2.1 RELEVE DES TEMPERATURES

Le relevé s'effectue via une liaison USB 2.0 entre la carte électronique et le micro-ordinateur de régulation :

- interface de transfert FPGA vers USB par le circuit FT245R de la société FTDI
- protocole de transmission des températures :
  - températures relatives numérisées (SOT) extérieures et intérieures transmises alternativement sur 6 octets.
  - période 4 ms représentant 10 secondes dans la réalité.
  - codage de la température relative numérisée SOT sur 12 bits (codage du capteur conservé) :

Température absolue en °C = -39,64 + 0,04 x SOT

  - format de transfert des températures relatives numérisées SOT : SOT décomposé en quartet et transmis sur 3 octets pour assurer la synchronisation.

	Octet	Bit							
		7	6	5	4	3	2	1	0
Température extérieure	1er	0	0	0	0	SOT Bit 11	SOT Bit 10	SOT Bit 9	SOT Bit 8
	2ème	0	0	0	1	SOT Bit 7	SOT Bit 6	SOT Bit 5	SOT Bit 4
	3ème	0	0	1	0	SOT Bit 3	SOT Bit 2	SOT Bit 1	SOT Bit 0

	Octet	Bit							
		7	6	5	4	3	2	1	0
Température intérieure	1er	1	0	0	0	SOT Bit 11	SOT Bit 10	SOT Bit 9	SOT Bit 8
	2ème	1	0	0	1	SOT Bit 7	SOT Bit 6	SOT Bit 5	SOT Bit 4

	3ème	1	0	1	0	SOT Bit 3	SOT Bit 2	SOT Bit 1	SOT Bit 0

2.2 INTERACTION AVEC L'INTERFACE PYTHON

Une interface utilisateur a été développée en langage Python à votre intention comme le montre la Figure 2.

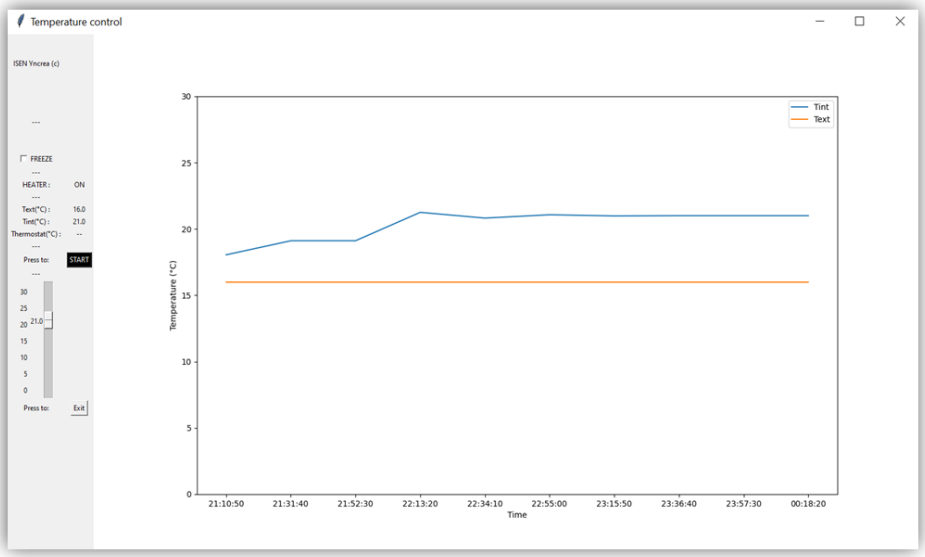


FIGURE 2 : INTERFACE PYTHON

Les données qui apparaissent sur cette interface sont lues dans les fichiers `consigne.txt` et `data.txt` (mise à jour toutes les secondes) comme le montre la Figure 3:

- La consigne doit être comprise entre 5°C et 40°C Cette donnée (type réel) est stockée dans le fichier `consigne.txt`. N'oubliez pas de cliquer sur le bouton Valider pour enregistrer la donnée dans le fichier.
- Les données concernant la température extérieure (type réel), la température intérieure (type réel) et le témoin de chauffe (type chaine de caractères : « true » ou « false ») sont lues dans le fichier `data.txt`. Le témoin de chauffe doit être allumé quand la puissance de chauffage est positive et être éteint quand la puissance de chauffage est nulle. Le fichier `data.txt` comporte trois lignes qui correspondent respectivement aux trois données précitées.

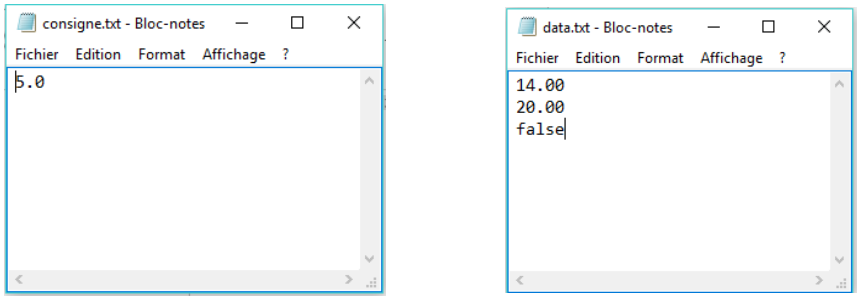


FIGURE 3: EXEMPLE DE FICHIERS CONSIGNE.TXT ET DATA.TXT

Afin d'être sûr de ne pas avoir de problèmes de concurrence entre les programmes, les accès en lecture ou écriture aux fichiers `consigne.txt` et `data.txt` sont protégés par des verrous. Il n'est pas possible de lire et écrire en même temps dans un fichier. C'est ce qu'on appelle l'exclusion mutuelle. Ce mécanisme de verrou est mis en œuvre avec deux fichiers : `.verrouConsigne` et `.verrouData` La présence d'un fichier verrou indique qu'une opération de lecture ou écriture est en cours sur le fichier de données correspondant. L'absence de verrou autorise la lecture ou l'écriture.

2.3 REGULATION DE LA TEMPERATURE INTERIEURE

2.3.1 TOUT OU RIEN

C'est le système le plus simple à mettre en place. On lance une commande de chauffage uniquement si l'on se trouve en dessous de la valeur de la consigne. Le chauffage sera fixé à 40 % lorsqu'il est en marche et à 0 s'il est éteint.

2.3.2 REGULATEUR PID (PROPORTIONNEL INTEGRAL DERIVE)

Un régulateur Proportionnel Intégral Dérivé (PID) est un organe de contrôle permettant d'effectuer une régulation en boucle fermée d'un système industriel. C'est le régulateur le plus utilisé dans l'industrie. Il permet de contrôler un grand nombre de procédés. Nous vous proposons d'utiliser ce type de régulateur pour contrôler le chauffage.

2.3.2.1 FONCTIONNEMENT :

Comme son nom l'indique, le régulateur PID est composé de trois termes :

- Proportionnel (P) : évalue la différence entre la mesure et la consigne
- Intégral (I) : évalue la somme des différences précédentes
- Dérivé (D) : évalue la rapidité de changement de la différence

La valeur utilisée pour la régulation PID est la somme des ces trois termes :

$$PID = P + I + D$$

Le poids accordé à ces différents termes permet d'optimiser la régulation du système. On peut dans certains cas mettre l'un des poids à zéro.

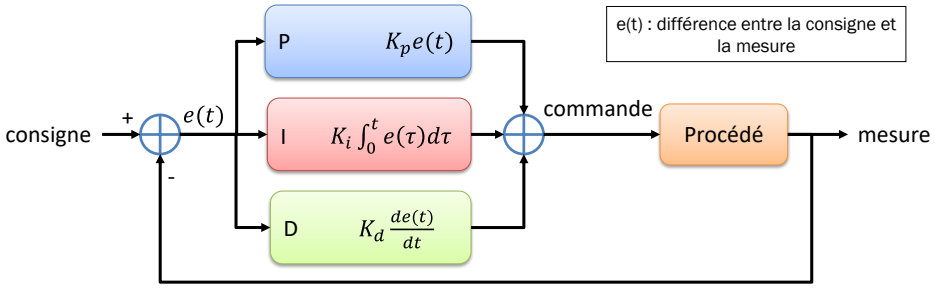
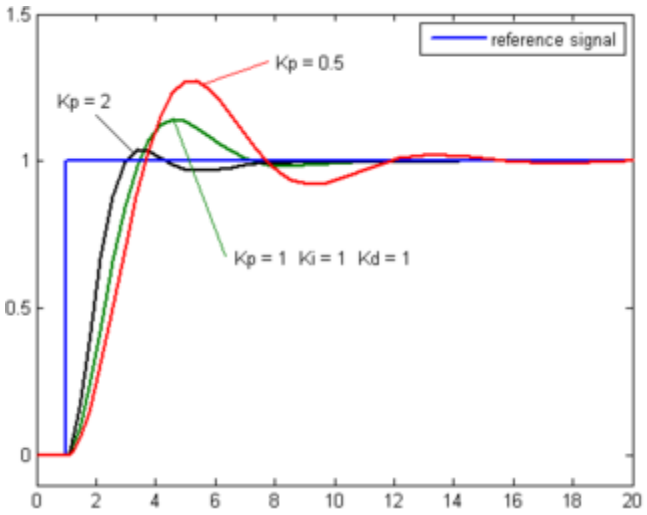


FIGURE 4: ALGORITHME DU REGULATEUR PID

- Terme Proportionnel :

$$P = K_p e(t)$$



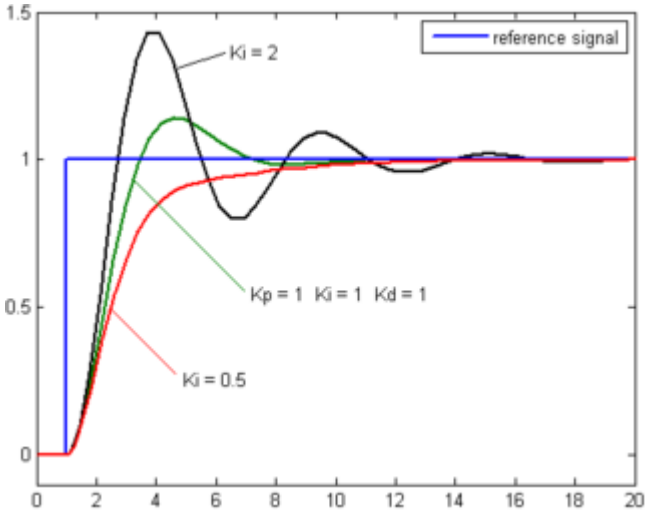
$e(t)$ est la différence entre la consigne et la mesure (dépendant du temps t).

K_p est le gain proportionnel permettant d'ajuster le poids du terme proportionnel.

Un fort gain proportionnel entraîne une réaction forte en sortie. S'il est trop fort le système ne convergera pas. S'il est trop faible, la convergence sera très lente.

- Terme Intégral :

$$I = K_i \int_0^t e(\tau)\, d\tau$$

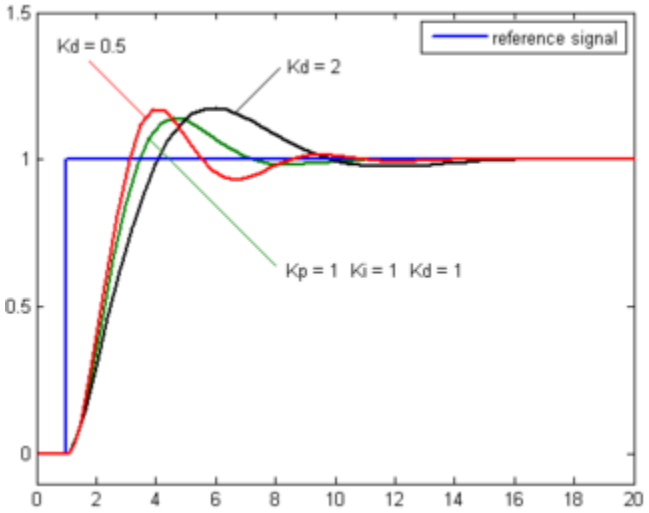


K_i est le gain intégral. Ce terme prend en compte la différence entre la mesure et la consigne ainsi que le temps pendant lequel cette différence existe.

Ce terme accélère la convergence et élimine l'erreur résiduelle introduite par le terme proportionnel.

- Terme Dérivé :

$$D = K_d \frac{de}{dt}$$



K_d est le gain dérivé. Ce terme détermine la pente de la différence par rapport au temps.

Ce terme ralenti les changements du régulateur, ce qui est important lorsque l'on s'approche de la convergence mais ce terme amplifie le bruit.

La complexité de la mise en place d'un système de régulation PID réside dans les points suivants :

- Faire le lien entre le PID et la commande de chauffage. Il faut donc manipuler le PID de façon à avoir une valeur qui variera entre 0 % (pas de chauffage) et 100 % (chauffage maximum)
- Trouver les gains qui optimisent la régulation (manuellement ou en utilisant une méthode telle que celle de Ziegler-Nichols, voir sur Internet)

2.4 ENVOI D'UNE COMMANDE DE CHAUFFAGE

L'envoi d'une commande est réalisé via une liaison USB 2.0 avec le micro-ordinateur de régulation :

- interface de transfert USB vers FPGA par le circuit FT245R de la société FTDI
- protocole de commande du chauffage :
  - codage de la puissance de chauffage sur 7 bits :

puissance en % = (PUIS / 127) x 100

  - codage de chaud /froid sur 1 bit :
    - 0 = chauffage
    - 1 = froid
  - format de transfert de la commande de chauffage :

	Bit							
	7	6	5	4	3	2	1	0
Commande chauffage	0 = chauffage 1 = froid	PUIS Bit 6	PUIS Bit 5	PUIS Bit 4	PUIS Bit 3	PUIS Bit 2	PUIS Bit 1	PUIS Bit 0

2.5 FIN DU PROGRAMME

Lorsque l'utilisateur rentre la valeur de consigne minimum (5°c), **on demande à ce que la régulation s'arrête et que votre programme se termine**. Bien entendu afin de prévenir tout problème pratique (incendie, etc…), vous devez penser à mettre la puissance de chauffage à 0% avant de quitter.

2.6 SIMULATEUR DU "MINI-ENVIRONNEMENT" ET "ECHANGES THERMIQUES"

La société "PEDAGIX" n'a pas mis sa carte électronique à disposition, il est donc impossible de tester la partie régulation sur le système réel avant la phase d'intégration. Il est ainsi nécessaire de simuler les modules "mini-environnement" et "échanges thermiques" informatiquement.

2.6.1 PRINCIPE

Afin de simplifier le développement de votre programme, un simulateur de l'environnement et des échanges thermiques vous est proposé, vous le trouverez sur l'intranet dans la section ressources sous le nom `simulateur.zip`.

Ce simulateur prend en entrée une puissance de chauffage (entre 0 et 100) et ressort une température intérieure et extérieure. En se basant sur une durée de simulation de 10 s de l'environnement, le programme calcule une température intérieure (calculée à partir de la valeur de chauffage) ainsi que la température extérieure (initialisée au début du programme et qui reste constante).

Le simulateur garde une trace du temps écoulé, de la puissance de chauffage demandée, de la température extérieure et de la température intérieure calculée, en écrivant à chaque appel, toutes ces valeurs dans un fichier `trace.txt`. La formule utilisée par le simulateur est la suivante :

$$T_{int}(t+1) = \frac{T_{ext}(t) + R_{GF}\frac{P_{max}\cdot P(t)}{127} + T_{int}(t)\cdot K_r}{K_r + 1}$$

Avec :

$$\begin{cases} P_{max} = 3200 \\ \Delta T = 10 \\ C_{TH} = 73746.9 \\ R_{GF} = 0.00686 \\ K_r = \dfrac{C_{TH}\cdot R_{GF}}{\Delta T} \end{cases}$$

2.6.2 FONCTIONS

2.6.2.1 CONSTRUCTION DU SIMULATEUR :

```
struct simParam_s* simConstruct(temp_t temperature)
```

Cette fonction construit une structure de simulateur de type `simParam_s` à partir d'une température extérieure et intérieure initiales stockées dans la variable structurée `temperature`. La fonction initialise également le fichier `trace.txt`. La structure créée est retournée par la fonction.

2.6.2.2 FONCTION DE SIMULATION :

```
temp_t simCalc(float puissance, struct simParam_s* param_ps);
```

La fonction réalise une simulation de 10 secondes d'échange thermique en prenant en entrée une valeur de puissance représentée par la variable `puissance` et retourne un nouveau couple de températures intérieure et extérieure sous la forme d'une structure de type `temp_t`. La fonction met également à jour le fichier `trace.txt`.

2.6.2.3 FONCTION DE DESTRUCTION DU SIMULATEUR :

```
void simDestruct(struct simParam_s* param_ps);
```

Cette fonction détruit la structure de simulateur.

2.6.3 EXEMPLE D'UTILISATION

- Tout d'abord il faut extraire de l'archive `simuleur.zip`, les fichiers sources `simulateur.c` et `simulateur.h`.
- Vous pouvez ensuite ouvrir le fichier test_simulateur.c qui montre comment utiliser le simulateur.
- Vous pouvez ensuite compiler puis exécuter le code test de la façon suivante :

```
gcc -c simulateur.c

gcc -c test_simulateur.c

gcc simulateur.o test_simulateur.o -o test_simumateur.exe
```

3 COTE PRATIQUE

3.1 ENVIRONNEMENT DE TRAVAIL

Pour la partie simulation, l'environnement de travail n'est pas imposé, veuillez-vous reporter aux consignes données dans le module « Algorithme et Langage C ».

Pour la partie connexion à la carte électronique, les drivers USB sont prévus pour fonctionner sous Windows uniquement. **Un développement sous windows (MinGW) devient ainsi nécessaire**.

Afin de faciliter la recette on vous demande d'organiser les fichiers de votre projet de façon à avoir une architecture identique à la Figure 5. Attention toute architecture différente pourra entrainer un malus lors de la recette.

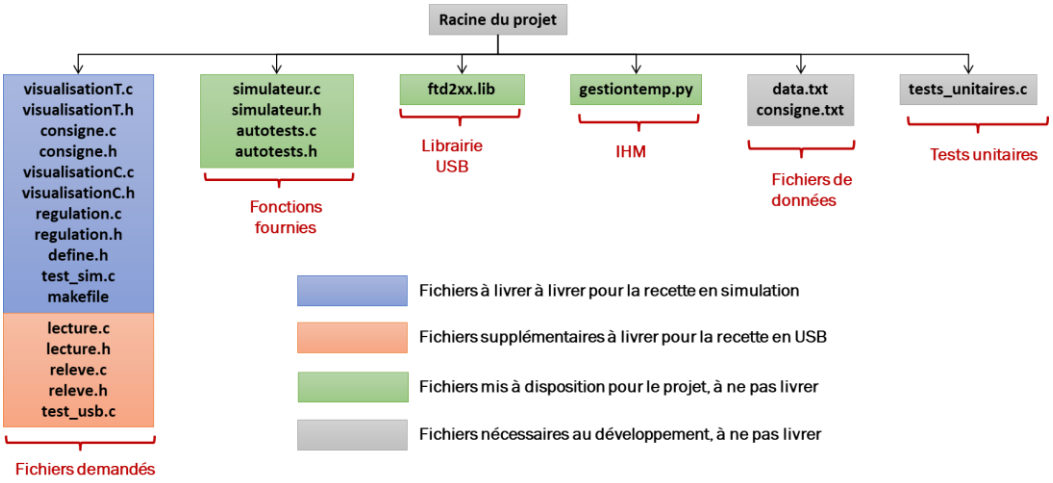


FIGURE 5 : ARCHITECTURE DES FICHIERS (OBLIGATOIRE)

Pour utiliser les drivers USB de la carte FTDI, on vous demande de suivre le guide d'installation donné sur l'intranet. Pour la compilation, il vous sera nécessaire d'inclure les fichiers suivants :

```
#include <windows.h>
#include "ftd2xx.h"
```

De même lors de l'édition de lien il faut ajouter la librairie ftd2xx.lib[1] :

[1] La librairie à utiliser pour Windows 64 bits : CDM v2.12.36.4 WHQL Certified.zip\i386\ftd2xx.lib.

```
gcc toto1.o toto2.o ftd2xx.lib –o toto.exe
```

Attention si l'architecture des fichiers représentée en Figure 5 n'est pas respectée, la commande précédente échouera.

3.2 UTILISATION DE LA CARTE ELECTRONIQUE

3.2.1 INSTALLATION

La carte doit être alimentée en 5V DC et reliée à votre PC au moyen d'un cordon mini USB branché sur le port « USB COM » comme le montre la Figure 6.

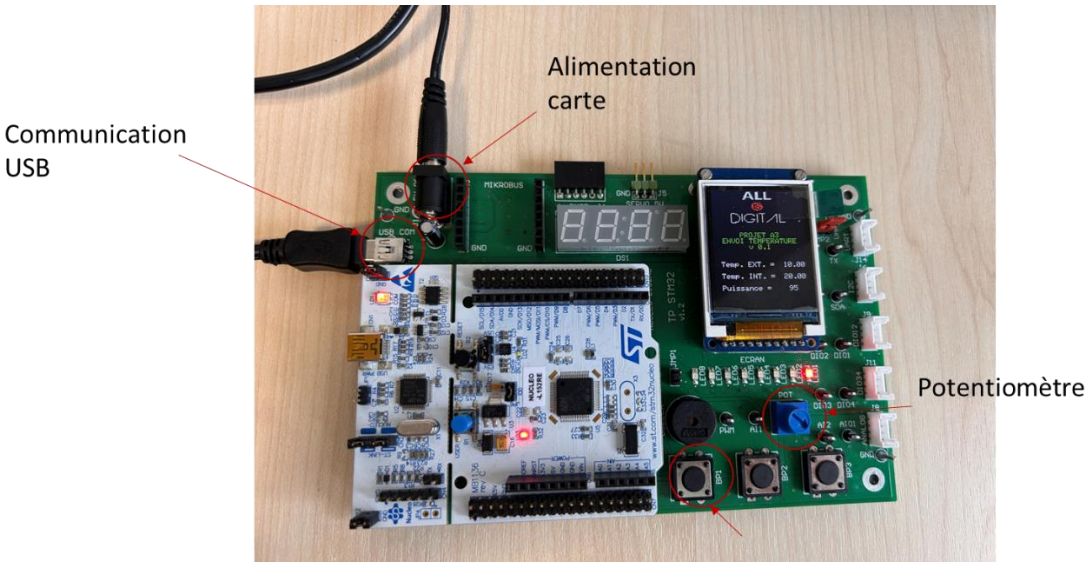


FIGURE 6 : CARTE ELECTRONIQUE DE TEST

Il est également possible d'alimenter la carte au moyen d'un second câble mini-USB relié à votre PC en utilisant le port « USB PWR »

3.2.2 PARAMETRAGE DRIVER

Lors de l'initialisation de la liaison USB au moyen de la bibliothèque FTD2XX.lib, **il est nécessaire d'effectuer les paramétrages suivants** :

- Débit de la communication (*baudrate*) fixé à **115200** bauds
- Caractéristiques des données
  - **8 bits par mots**
  - **1 bit de stop**
  - **Pas de parité**
- Contrôle de flux
  - **Pas de contrôle de flux**
  - **Autres paramètres fixés à 0**

3.2.3 UTILISATION

Un appui sur le bouton BP1 permet de changer de mode.

- Led1 clignotante (défaut) : Température fixe $T_{ext} = 10°C$ et $T_{int} = 20°C$
- Led8 allumée et Led1 clignotante : les températures sont modifiables au moyen du potentiomètre « POT » avec $T_{int} = T_{ext} + 5\ °C$

3.3 IHM PYTHON

3.3.1 INSTALLATION

- Télécharger python sur https://www.python.org/downloads/
- Lancer l'installation en pensant à cliquer sur « add python.exe to PATH »

- Ouvrir un terminal (sous windows « invite de commande ») et taper les commandes suivantes

```
python -m pip install -U numpy

python -m pip install -U matplotlib
```

3.3.2 UTILISATION

Le principe d'utilisation de l'IHM est donné en Figure 7.

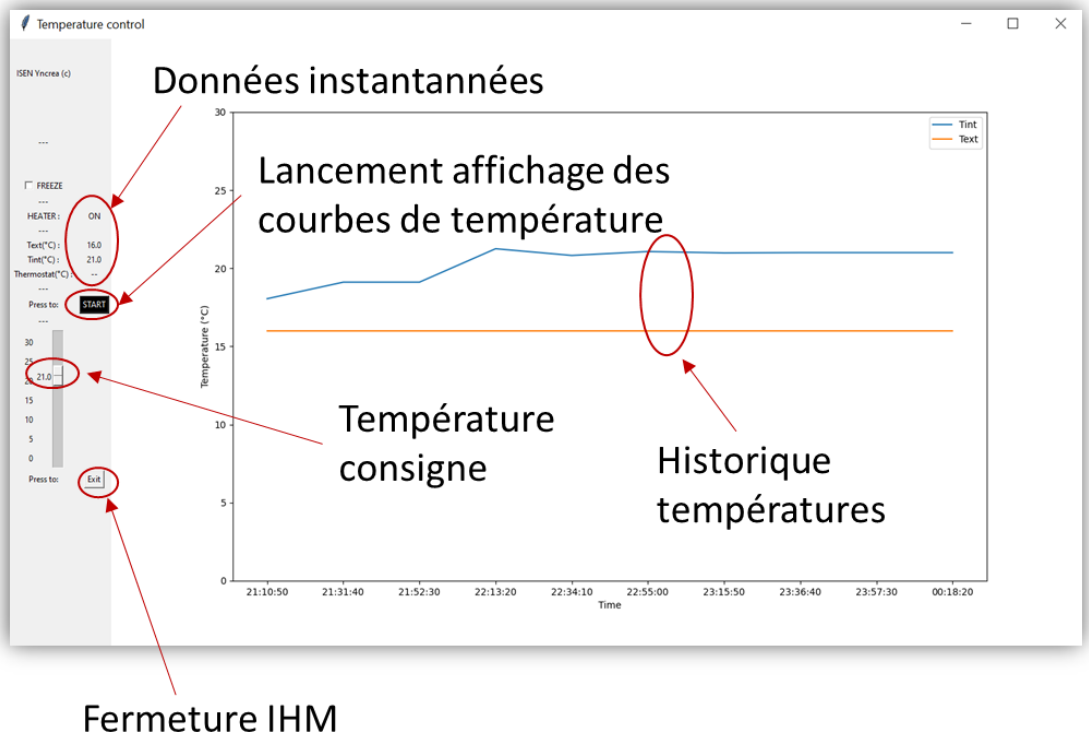


FIGURE 7 : UTILISATION DE L'IHM

4 EVALUATION

Le barème prévisionnel du projet est décrit dans le Tableau 1.

Métrique	Coefficient
Recette programme	3
Audit code	1
QCM	1
Total	**5**

TABLEAU 1 : BAREME PREVISIONNEL

4.1 RECETTE

4.1.1 MODALITES

Le barème de la recette de votre programme est indiqué dans le Tableau 2. La recette de votre programme sera effectuée le dernier jour entre 15:00 et 18:00. Vous devrez rendre avant 15 :00 le jour de la recette une archive contenant vos sources que vous nommerez sources_xx_yy.zip (où xx et yy désignent les initiales respectives du binôme). Les fichiers à inclure sont les suivants :

- Fichiers sources à rendre minimum
  - Attention, le contenu est imposé (voir ci-dessous)
  - Les prototypes des fonctions sont également imposés (voir annexe)

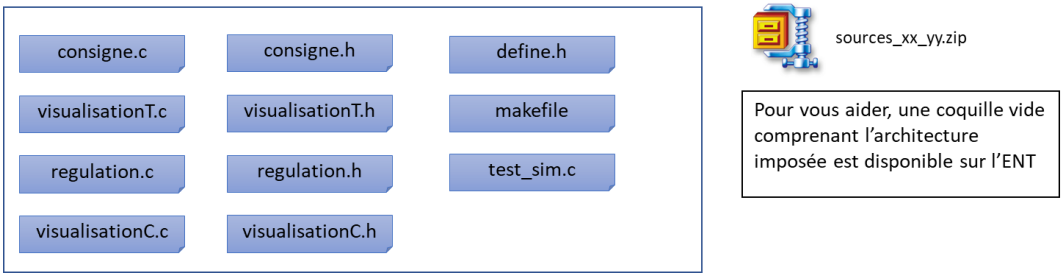


FIGURE 8 : ARCHIVE POUR TESTS AUTOMATIQUES

- Fichiers sources supplémentaires pour les tests USB :
  - Fichiers précédents
  - releve.c
  - releve.h
  - commande.c
  - comande.h
  - test_usb.c

#	Nom du test	Type	A3 hors CIR3	CIR3
1	Visualisation températures	Automatique	2,5	2
2	Lecture consigne	Automatique	2,5	2
3	Visualisation Commande	Automatique	2,5	2
4	Régulation ToR	Automatique	2,5	2
5	Régulation PID	Automatique	4	4
6	Programme global simulation	Manuel	4	4
7	Releve USB	Manuel	1	1
8	Commande USB	Manuel	1	1
9	Programme global USB	Manuel	Bonus	2
Total			**20**	**20**

TABLEAU 2 : BAREME RECETTE PREVISIONNEL

Tout retard lors de la livraison des sources sera systématiquement sanctionné (l'heure du réseau faisant foi).

Lors de la recette, un outil anti-plagiat sera utilisé pour vérifier qu'il n'y a pas eu de recopie entre étudiants ou avec du code sur internet. Attention, toute distance informationnelle inférieure à 0.5 entre 2 codes entrainera au minimum une pénalité de 50% pour chaque auteur. **Nous rappelons que l'utilisation d'outils de génération de codes par intelligence artificielle de type ChatGPT est totalement proscrit et sera sévèrement sanctionné** (0 à la recette et signalement auprès de la direction des études en vue d'un éventuel conseil de discipline)

Si lors de la recette, les test automatiques ou manuels ne peuvent être effectués (erreur de compilation ou d'exécution), un repassage plus tard dans l'après-midi peut être proposé, une pénalité (substantielle) sera alors appliquée.

4.1.2 TESTS AUTOMATIQUES

Nous vous mettons à disposition un fichier de fonctions (autotests.c/autotests.h) permettant de tester unitairement les différents blocs :

- testVisualisationT();
- testConsigne();
- testVisualisationC();
- testRegulationTOR();
- testRegulationPID();

Chaque fonction renvoie une valeur entre 1 (bloc fonctionnel) et 0 (bloc non fonctionnel). Un programme permettant de tester unitairement les différents blocs est le suivant :

```
int main(){
   float score1=0,score2=0,score3=0,score4=0,score5=0;
   score1 = testVisualisationT();
   score2 = testConsigne();
   score3 = testVisualisationC();
   score4 = testRegulationTOR();
   score5 = testRegulationPID();
   printf("----- Auto tests results: -----\n");
   printf("testVisualisationT\t:score=%g%%\n",score1*100);
   printf("testConsigne \t\t: score=%g  %%\n",score2*100);
   printf("testVisualisationC\t:score=%g %%\n",score3*100);
   printf("testRegulationTOR\t:score=%g  %%\n",score4*100);
   printf("testRegulationPID\t:score=%g  %%\n",score5*100);
   return EXIT_SUCCESS;
}
```

Attention les prototypes des fonctions principales de chaque bloc sont imposés (cf section 5).

4.2 AUDIT DU CODE

Lors de la recette, l'examinateur auditera votre code sur les points suivants

- Respect du cahier des charges
- Lisibilité du code
- Présence de commentaires
- Absence de variables globales
- Explication de certaines parties du code par les deux membres du binôme

4.3 QCM

Lors du jour de la recette, une épreuve théorique de validation des connaissances acquises sera effectuée sous la forme d'un questionnaire à choix multiple (QCM) d'une durée de 30 min. Ce QCM sera effectué en monôme et en ligne sur l'intranet.

5 ANNEXE : STRUCTURE ET FONCTIONS ET IMPOSEES POUR LES TESTS AUTOMATIQUES

5.1 FICHIER DE DEFINITION « DEFINE.H »

```
#ifndef DEFINE_H
#define DEFINE_H

typedef struct{
          float exterieure;
          float interieure;
} temp_t;

…

#endif
```

5.2 VISUALISATION DE LA TEMPERATURE

visualisationT.h

```
#include "define.h"
void visualisationT(temp_t maTemperature);
```

visualisationT.c

```
void visualisationT(temp_t maTemperature){
          …
          …
}
```

Attention, si le verrou est présent, la fonction ne doit modifie pas modifier le fichier `data.txt`.

5.3 RECUPERATION DE LA CONSIGNE

consigne.h

```
float consigne(float csgn);
```

consigne.c

```
float consigne(float csgn){
  …
  return csgn;
}
```

Attention, si le verrou est présent la fonction doit retourner le paramètre `csgn` passé en entrée de la fonction !

5.4 REGULATION

Ce bloc est un peu particulier car le prototype de la fonction régulation n'est pas imposé, ceci dans le but de laisser au libre choix des étudiants les paramètres d'entrées et de sorties du bloc. En revanche, il est demandé de fournir une fonction de test de ce bloc régulation nommée « `regulationTest` » dont le prototype est imposé :

regulation.h

```
float regulationTest(int regul,float csgn,float* tabT, int nT);
```

regulation.c

```
float regulationTest(int regul,float csgn,float* tabT, int nT){
          float cmd=100.0;
          …
          …
          return cmd;
 }
```

Cette fonction retourne la valeur de la commande de chauffage obtenue après nT appels à la fonction `regulation()` :

- `regul` : type de régulation (1:Tout ou Rien, 2:PID)
- `csgn` : température de consigne
- `tabT` : tableau de températures intérieures successives en entrée de la régulation
- `nT` : nombre d'entrées dans le tableau `tabT`

Pour la régulation PID, les gains sont fixés à Kp=1, Ki=0.1 et Kd=0.1. La loi de saturation entre la valeur de PID et la commande de chauffage est linéaire.

5.5 VISUALISATION DE LA COMMANDE DE CHAUFFAGE

visualisationC.h

```
void visualisationC(float puissance);
```

visualisationC.c

```
void visualisationC(float puissance){
          …
          …
}
```

Si le verrou est présent, la fonction ne modifie pas le fichier `data.txt`. Le paramètre puissance représente la puissance de chauffage en % allant de 0 à 100

5.6 CONSEILS

Une archive contenant tous les fichiers demandés et les prototypes associés vous est fournie sur l'intranet sous le nom minimal.zip.

Nous vous conseillons de partir de ces fichiers et de compléter au fur et à mesure les différentes fonctions.

Si vous n'avez pas réussi un bloc, commentez le code qui n'est pas fonctionnel mais laissez le squelette obligatoire de la fonction (coquille vide).