Lycée TOUCHARD WASHINGTON

Informatique embarquée

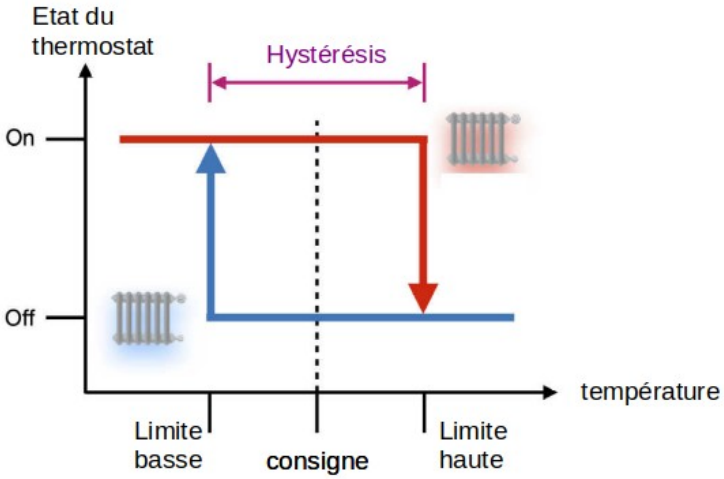
TD n°3

CIEL 1
2023-2024

Gestion du Thermostat

1- Mise en situation

Un thermostat électronique régule la température en suivant un principe simple. L'utilisateur définit une température de consigne. Lorsque la température atteint cette consigne plus la moitié de l'hystérésis, le chauffage s'éteint. Il se rallume lorsque la température descend à la température de consigne moins la moitié de l'hystérésis.



2- Description des entrées/sorties du système

BP1	Bouton +
LED1 (rouge)	Chauffage
SW On	Réglage consigne

BP2	Bouton -
LED2 (verte)	Repos
SW Off	En fonctionnement

3- Création du projet

Créez un projet intitulé **Thermostat** en utilisant **PlatformIO** pour l'environnement NetBeans. Une fois le projet ouvert dans **NetBeans**, copiez les fichiers nécessaire pour gérer l'afficheur et le thermomètre dans votre projet.

Ajoutez deux nouveaux fichiers, **thermostat.h** et **thermostat.cpp** pour recevoir la librairie de gestion du thermostat.

Ajoutez le fichier **main.cpp**, il contiendra les fonctions **setup()** et **loop()** de votre application.

4- Création de la librairie Thermostat

Créez les fonctions suivantes :

a) **InitialiserThermostat**

Cette fonction configure les boutons-poussoirs pour ajuster la consigne de température et initialise les LED pour refléter l'état du système. À la fin de l'initialisation, les LED sont éteintes. Cette fonction ne prend aucun argument en entrée et ne retourne rien.

b) **FixerConsigne**

Cette fonction prend en paramètre la consigne actuelle et renvoie un nombre réel représentant la consigne modifiée. Lorsque l'utilisateur appuie sur le bouton **+**, la consigne augmente de 0,5 degrés Celsius. Lorsqu'il appuie sur le bouton **-**, la consigne diminue de 0,5 degrés Celsius. L'affichage de la consigne se fait sur l'écran, similaire à l'affichage de la température. Un délai de 200 millisecondes est prévu entre chaque pression sur une touche.

c) **RegulerTemperature**

Cette fonction reçoit la consigne de température, la température actuelle (deux nombres réels), ainsi que la valeur de l'hystérésis (un nombre entier). Elle retourne un booléen : vrai si le chauffage est en marche, faux sinon. Cette fonction agit sur les LED et sur la co valeur du paramètre de retour conformément aux règles de régulation décrites dans le contexte initial.

5- Codage du programme principal

Complétez les fonctions **setup()** et **loop()** dans le fichier **main.cpp** présent à la page suivante.

La première fonction est responsable de l'initialisation de l'afficheur, du capteur de température, du thermostat, ainsi que de la configuration des entrées pour l'interrupteur.

La seconde fonction vérifie d'abord la position de l'interrupteur. Si l'interrupteur est en position de régulation, elle obtient la température actuelle, l'affiche sur l'afficheur OLED, puis appelle la fonction de régulation de température. Si l'interrupteur est en position de réglage de la consigne, elle appelle simplement la fonction FixerConsigne() pour ajuster la consigne de température sans réguler la température.

Pour ne pas bloquer le programme pendant 5 secondes on propose d'utiliser la fonction millis() qui retourne le nombre de millisecondes écoulé depuis la mise en route du système.

Pour terminer, testez le paramètre de retour de la fonction **RegulerTempérature()** et, en fonction de sa valeur, agissez sur le contact du Relais1 qui utilise la broche GPIO27.

main.cpp

```
#define INTERVAL 5000 // mesure de la température toutes les 5s
#define HYSTERESIS 1

float temperature = 0;
float consigne = 0;
unsigned long precedent;

void setup()
{
   precedent = millis() - INTERVAL ;  // initialisation pour voir le temps écoulé

}

void loop()
{
   if (/* Test de la position de l'interrupteur */)
   {
      // Réglage de la consigne
   }
   else
   {

      unsigned long courant = millis(); // attente de 5 secondes sans blocage
      if (courant - precedent >= INTERVAL)
      {
         precedent = courant;
         // Fonctionnement normal

      }
   }
}
```