AG43 Examen final Automne 2014

Durée : 1h30

Exercices 1 et 2 à rédiger sur feuille 1, 3 et 4 sur feuille 2.

Modification boucle de gestion d'une chaudière.

La boucle principale de gestion d'une chaudière est la suivante :

```
Iterer
        k = get_tempo()
        tempos[k] = 300

        t_ext = temp(0)
        t_eau = temp(1)
        dif = calc_consigne(t_ext) - t_eau

        Si1 dif > 50
                Alors1 n = 3
                Sinon1 si2 dif > 15
                        Alors2 n = 2
                        Sinon2 Si3 dif > 0
                                Alors3 n = 1
                                Sinon3 n = 0
                        Fin si3
                Fin si2
        Fin si1

        tant que tempos[k] > 0
                resistance(n)
        fin tant que

Fin iterer
```

Rappels :

- `tempos[]` est un tableau de 8 entiers servant de temporisations. Chaque élément non nul de ce tableau est décrémenté 10 fois par seconde par la fonction de type interruption `timer()`.
- `get_tempo()` est une fonction qui retourne l'indice du premier élément nul (donc inutilisé) du tableau `tempos[]`.
- `temp(numéro_sonde)` est une fonction qui retourne la température en dixièmes de Kelvin mesurée par diverses sondes. Dans le cas présent, la valeur 0 est attribuée à la sonde

qui mesure la température à l'extérieur de la maison, et 1 à la sonde mesurant la température du départ eau chaude de la chaudière.

- `calc_consigne()` est une fonction qui, à partir de la température extérieure calcule et retourne la consigne de température nécessaire au niveau du départ eau chaude.

- `resistance(n)` est une fonction qui commute n résistances (maximum 3). `resistance(0)` éteint toutes les résistances.

La boucle est parcourue toutes les 30 secondes, et le nombre n de résistances chauffantes à actionner dépend de la différence entre la température mesurée de l'eau et la consigne. Les résistances sont de type monostable, il faut les commander régulièrement (au moins une fois par seconde).

Question 1 : réduction de l'erreur due au caractère proportionnel de cette régulation

On souhaite améliorer cette boucle en introduisant une intégrale par rapport au temps. Pour ce faire, on modifie totalement la boucle, en tenant compte des points suivants :

- On n'évalue plus la différence entre la température mesurée de l'eau et la consigne, on se contente de les comparer.

- On parcourt la boucle une fois toutes les 10 mn.

- On dispose d'un compteur de résistances appelé n, qui indique le nombre de résistances à actionner.

- Si l'on trouve une température de l'eau inférieure à celle de la consigne, on incrémente n, sinon on le décrémente.

- La chaudière comporte 3 résistances (inchangé, n ne dépassera pas la valeur 3).

Modifier la boucle fournie de manière à respecter les points précédents. Inutile de déclarer les variables.

Question 2 : Réduction de la surchauffe occasionnée par cette seconde méthode, par prédiction de la température atteinte par l'eau

Lorsque l'on utilise la méthode précédente, la température de l'eau va inévitablement dépasser la valeur de la consigne, car il faut 30 mn pour couper toutes les résistances à partir du moment où elles seraient toutes en fonctionnement. On décide donc de prendre en compte la cinétique de la température.

On mesure la variation de la température de l'eau entre deux bouclages, et on prédit la température de l'eau pour le bouclage suivant en additionnant cette variation à la température actuelle de l'eau. Si l'on constate que l'on risque de dépasser la valeur de consigne, on diminue le nombre de

résistances actionnées, sinon on l'augmente ou on le laisse inchangé le cas échéant. La boucle sera parcourue toutes les 10 mn. Modifier la boucle fournie et déclarer les nouvelles variables.

Thermostat programmable - Gestion de l'horloge

Un thermostat programmable comporte une sonde de température, une horloge temps réel et une interface de commande (émetteur radio de télécommande ou relais de puissance).

Nous nous intéresserons plus particulièrement à l'horloge. Le fonctionnement, bien que sommairement décrit est à peu près le suivant.

Le programmateur fonctionne sur piles, il est donc hors de question de laisser en fonctionnement le microcontrôleur. En fait, l'horloge temps réel interne se contente de générer une impulsion toutes les minutes. Cette impulsion réveille le microcontrôleur qui est en mode sommeil (consommation réduite, programme arrêté), et provoque le déclanchement d'une interruption. A chaque réveil, le microcontrôleur réalise les tâches suivantes :

- Mise à jour de la mémoire horloge qui comporte les heures, les minutes et les jours.
- Mesure de la température et mémorisation.
- Gestion du programme (à partir de la programmation mémorisée des températures).

La routine d'interruption gère l'horloge interne, mesure la température et la mémorise dans la variable `theta`.

L'horloge est un tableau de 3 octets noté `date`, tel que `date[2]` représente les minutes, `date[1]` les heures de 0 à 23, et `date[0]` le jour, numéroté de 0 à 6, 0 étant le Lundi.

On dispose de la fonction `get_temp()` ne nécessitant pas d'argument et retournant la température exprimée de dixièmes de Kelvin.

A chaque appel (c'est-à-dire toutes les minutes) la routine d'interruption, appelée `inter_gestion()`, met à jour la date et la température.

Question 3 : Gestion de l'horloge interne

Ecrire la procédure de type interruption appelée `inter_gestion()`. Déclarer toutes les variables, globales ou locales.

Thermostat programmable - Transmission des ordres

Le thermostat doit pouvoir envoyer deux ordres distincts (Marche et Arrêt) à l'appareil de chauffage. Dans le cas d'un thermostat déporté, on bénéficie de la possibilité de le soustraire à l'influence du rayonnement direct du radiateur. Le radiateur comporte un récepteur qui le met en marche ou le coupe selon l'ordre reçu. Pour éviter l'interférence possible entre plusieurs thermostats, chaque thermostat est apparié à son récepteur, et la communication de l'ordre se fait en plaçant en préambule un octet d'appairage qui identifie l'émetteur auprès de son récepteur. Il est donc nécessaire qu'il y ait transmission de l'information d'identification avec l'ordre de commande.

Nous nous intéresserons à la transmission série de ces informations, et plus particulièrement à l'émission d'un octet.

Description de l'émission série d'un octet.

On dispose de l'opération décalage à droite d'un octet, notée **>>**, qui décale d'un cran vers la droite tous les bits d'un octet (exemple : a = >>a, ou bien a ← >>a), le bit de poids faible étant perdu, et le bit de poids fort remplacé par un zéro. Pour savoir si le bit de poids faible est à 1, on effectue l'opération logique ET notée **&** entre l'octet à transmettre et la valeur 1 (valeur binaire sur un octet : 00000001). Si cette opération donne comme résultat 0, on sait que le bit de poids faible est à 0.

En utilisant l'opération décalage et l'opération ET logique on peut transmettre successivement les 8 bits de l'octet en commençant par le bit de poids faible. On dispose de la variable TX qui sert à la transmission, et de la fonction `attente()` qui permet de maintenir TX à la valeur à transmettre un certain temps, conforme à la vitesse de transmission. La transmission d'un 1 se fait en écrivant TX=1, et d'un 0 en écrivant TX=0.

Pour assurer la synchronisation du récepteur, la transmission d'un octet nécessite la transmission d'un bit de démarrage, qui sera toujours 1, suivie ensuite des 8 bits de l'octet à envoyer, et se termine par la transmission d'un bit d'arrêt qui sera toujours 0.

Question 4 : Procédure d'envoi d'un octet

Ecrire l'algorithme de la procédure Send(a) qui réalise l'envoi sériel de l'octet a.