Université de Douala
École Supérieure Technique La Salle

Systèmes Temps Réel

Fiches de travaux dirigés

Classe : **IIA5**

Pr Moffo Lonla Bertrand

Année académique 2026–2027

Table des matières

Mode d'emploi **2**

TD 1 — Notions fondamentales sur les systèmes temps réel **3**

TD 2 — Ordonnancement classique et critères de performance **4**

TD 3 — Ordonnancement temps réel : RM, DM, EDF, LLF **5**

TD 4 — Ressources partagées et synchronisation **7**

TD 5 — RTOS du compilateur CCS C et environnement Proteus **9**

TD 6 — Études de cas avec le RTOS de CCS **12**

TD 7 — Conception de l'asservissement de température **14**

TD 8 — Révision générale : sujet type d'examen **17**

Mode d'emploi

Chaque fiche correspond à un chapitre du cours. Les exercices sont à préparer **avant** la séance de TD ; la séance est consacrée à la correction et aux difficultés rencontrées. Pour les chronogrammes, on applique les conventions du cours : temps discret, à priorité égale on garde la tâche en cours puis l'indice le plus faible ; au tourniquet, une arrivée est placée dans la file avant la tâche dont le quantum expire.

Fiche	Thème	Séance prévue
TD 1	Notions fondamentales sur les STR	travail personnel (séances 1–2)
TD 2	Ordonnancement classique et critères de performance	séance 3 (05/12/2026)
TD 3	Ordonnancement temps réel : RM, DM, EDF, LLF	séance 3 (05/12/2026)
TD 4	Ressources partagées et synchronisation	séance 5 (19/12/2026)
TD 5	RTOS du compilateur CCS C et Proteus	séance 5 (19/12/2026)
TD 6	Études de cas RTOS	séance 6 (09/01/2027)
TD 7	Conception de l'asservissement de température	séance 7 (16/01/2027)
TD 8	Révision générale : sujet type d'examen	séance 9 (30/01/2027)

TD 1 — Notions fondamentales sur les systèmes temps réel

Exercice 1 — Définitions

Cours

1. Définir : système temps réel, système de contrôle-commande, système embarqué, tâche, processus, ordonnanceur.
2. Un système temps réel est-il un système rapide ? Justifier par un exemple.
3. Citer les trois catégories de systèmes temps réel en indiquant ce qui caractérise chacune et en donnant un exemple de chaque.
4. Quelles sont les caractéristiques attendues d'un système temps réel ?

Exercice 2 — Choisir la (ou les) bonne(s) réponse(s)

QCM

1. Un système temps réel dur est : (a) un système rapide ; (b) un système prévisible ; (c) un système où toutes les échéances sont respectées ; (d) un système « bare-metal ».
2. Un système où les échéances peuvent occasionnellement être manquées, le résultat tardif restant utile, est : (a) *hard* ; (b) *soft* ; (c) *firm*.
3. Dans une voiture, une tâche critique est : (a) le contrôle de la climatisation ; (b) le contrôle du freinage (ABS) ; (c) la commande des vitres.
4. Un processus peut passer par les états : (a) Élu → Bloqué → Prêt ; (b) Bloqué → Élu → Prêt ; (c) Élu → Prêt → Bloqué.
5. Les algorithmes d'ordonnancement hors ligne construisent la séquence : (a) avant l'exécution ; (b) après ; (c) pendant.

Exercice 3 — Classification de systèmes

Pour chacun des systèmes suivants, indiquer la classe de contraintes (dur, ferme, souple) et un ordre de grandeur des échéances : (1) régulation de vitesse d'un moteur à courant continu ; (2) affichage du prix à une pompe à carburant ; (3) visioconférence ; (4) relais de protection d'une ligne haute tension ; (5) pacemaker ; (6) régulation de température d'un four à micro-ondes ; (7) distributeur automatique de billets ; (8) décodage d'un flux vidéo de télévision numérique.

Exercice 4 — Super-boucle ou RTOS ?

Un PIC doit : scruter un bouton d'arrêt d'urgence (réaction en moins de 5 ms), mesurer une température toutes les 100 ms (conversion 50 µs), calculer un PID toutes les 500 ms (1 ms de calcul), et rafraîchir un afficheur LCD (8 ms pour tout l'écran).

1. On écrit une boucle infinie appelant successivement les quatre traitements, sans attente. Quel est le pire délai de réaction au bouton ? Le cahier des charges est-il respecté ?
2. Proposer deux améliorations.

TD 2 — Ordonnancement classique et critères de performance

On rappelle : temps de rotation = fin − arrivée ; temps d'attente = rotation − durée ; temps de réponse = premier accès au processeur − arrivée ; débit = nombre de processus / (fin du dernier − première arrivée).

Exercice 1 — Comparaison de FIFO, SJF, SRTF et tourniquet

séance 3

On considère cinq processus :

Processus	P_1	P_2	P_3	P_4	P_5
Arrivée	0	1	2	3	4
Durée	8	4	9	5	2

1. Tracer le chronogramme pour FIFO, SJF (non préemptif), SRTF (préemptif) et le tourniquet de quantum $q = 3$.
2. Pour chaque algorithme, calculer les temps de rotation, d'attente et de réponse de chaque processus et leurs moyennes, ainsi que le débit.
3. Classer les algorithmes. Lequel minimise le temps d'attente moyen ? le temps de réponse moyen ? Quel inconvénient a SJF/SRTF ?

Exercice 2 — Priorités fixes, avec et sans préemption

séance 3

Quatre processus (priorité 1 = la plus forte) :

Processus	P_1	P_2	P_3	P_4
Arrivée	0	1	2	4
Durée	4	3	5	2
Priorité	3	1	4	2

1. Tracer le chronogramme et calculer les temps de rotation, d'attente et de réponse, sans préemption puis avec préemption.
2. Combien de changements de contexte dus à une préemption observe-t-on ? Conclure sur l'intérêt de la préemption pour un STR.

Exercice 3 — Choix du quantum

Au tourniquet, un changement de contexte coûte $s = 0{,}1$ ms. Quel pourcentage du temps processeur est perdu pour un quantum de 1 ms ? de 10 ms ? Quels sont les inconvénients d'un quantum trop grand ? trop petit ?

TD 3 — Ordonnancement temps réel : RM, DM, EDF, LLF

Toutes les tâches sont périodiques, indépendantes, synchrones (réveil initial à $t = 0$) ; $\tau_i(C_i, T_i)$ désigne une tâche à échéance sur requête ($D_i = T_i$), $\tau_i(C_i, D_i, T_i)$ une tâche à échéance contrainte.

Exercice 1 — Test de Liu et Layland non concluant

séance 3

Soit $\tau_1(1, 4)$, $\tau_2(2, 5)$, $\tau_3(2, 10)$.

1. Calculer le facteur d'utilisation U et l'hyperpériode H. Que permet de conclure le critère de Liu et Layland pour RM ?
2. Calculer le pire temps de réponse de chaque tâche par l'analyse du temps de réponse. Conclure.
3. Tracer le chronogramme RM sur $[0, H]$ et vérifier le résultat. Tracer ensuite le chronogramme EDF. Quelle est la différence ?
4. Expliquer pourquoi on dit que la condition de Liu et Layland est suffisante mais pas nécessaire.

Exercice 2 — RM, EDF et LLF à pleine charge

séance 3

Soit $\tau_1(3, 6)$, $\tau_2(2, 8)$, $\tau_3(3, 12)$.

1. Calculer U et H. Que peut-on dire de l'ordonnançabilité par EDF ? par RM avec Liu et Layland ?
2. Montrer par l'analyse du temps de réponse que RM n'est pas valide ; vérifier sur le chronogramme.
3. Tracer les chronogrammes EDF et LLF sur $[0, H]$. Compter les préemptions. Quel est le taux d'occupation du processeur ?

Exercice 3 — Échéances contraintes : RM, DM et EDF

Soit $\tau_1(C = 1, D = 2, T = 5)$, $\tau_2(C = 2, D = 4, T = 4)$, $\tau_3(C = 2, D = 7, T = 10)$.

1. Calculer U et la densité $\Delta = \sum C_i/D_i$. Que conclure ?
2. Donner l'ordre des priorités RM et DM. Calculer les temps de réponse pour chacun.
3. Tracer les chronogrammes RM, DM et EDF sur $[0, 20]$. Conclure.

Exercice 4 — Durée maximale d'une tâche

On considère le jeu de tâches ordonnancé par RM :

Tâche	τ_1	τ_2	τ_3	τ_4	τ_5
C_i	1	2	3	2	x
$T_i = D_i$	10	15	20	30	40

1. Déterminer la valeur entière maximale de x garantie par le critère de Liu et Layland.
2. Déterminer la valeur entière maximale exacte de x par l'analyse du temps de réponse. Commenter.

Exercice 5 — Ordonnancement non préemptif (noyau coopératif)

Le jeu $\tau_1(1,4)$, $\tau_2(2,6)$, $\tau_3(3,12)$ est exécuté par un noyau **coopératif** à priorités RM (une tâche lancée va jusqu'à sa fin).

1. Tracer le chronogramme sur $[0, 24]$.
2. Calculer le pire blocage B_i de chaque tâche et majorer son temps de réponse. Le jeu est-il ordonnançable ?
3. Que se passerait-il si C_3 valait 4 ?

Exercice 6 — Questions de cours

1. Sur quel critère est basé l'algorithme RM ? DM ? EDF ? LLF ? Lesquels sont à priorités fixes ?
2. RM peut-il ordonnancer des tâches apériodiques ? Comment les prendre en compte ?
3. Pourquoi LLF est-il rarement utilisé malgré son optimalité ?

TD 4 — Ressources partagées et synchronisation

Exercice 1 — Condition de course

séance 5

Deux tâches A et B partagent une variable `x`, initialement nulle. Chacune exécute une fois `x = x + 1;`, traduit par le compilateur en `LOAD R, x` ; `ADD R, 1` ; `STORE x, R` (chaque tâche a son propre registre `R`, sauvegardé dans son contexte). Le noyau est préemptif.

1. Quelles valeurs finales de `x` sont possibles ? Donner un entrelacement pour chacune.
2. Même question si chaque tâche exécute l'instruction deux fois.
3. Proposer trois façons de protéger la section critique sur un PIC18 : avec le RTOS de CCS, sans RTOS, et dans un noyau préemptif.

Exercice 2 — Producteur / consommateur

séance 5

Une tâche `Acquisition` produit des mesures qu'une tâche `Archivage` écrit sur carte SD, à travers un tampon circulaire de N cases.

1. Écrire les deux tâches en pseudo-code avec des sémaphores. Donner leurs valeurs initiales et le rôle de chacun.
2. Que se passe-t-il si le producteur exécute `P(mutex)` avant `P(places)` ?
3. On ajoute une seconde tâche d'acquisition (deux producteurs). Faut-il modifier la solution ? Et si l'on avait un seul producteur et un seul consommateur, le sémaphore `mutex` serait-il indispensable ?

Exercice 3 — Inversion de priorité, PIP et ICPP

séance 5

Quatre tâches à priorités fixes (1 = la plus forte) partagent le sémaphore S :

Tâche	Priorité	Réveil	Séquence d'exécution
τ_1	1	3	1 unité normale, 2 unités en section critique sur S, 1 unité normale
τ_2	2	4	4 unités normales
τ_3	3	2	2 unités normales
τ_4	4	0	1 unité normale, 4 unités sur S, 1 unité normale

1. Tracer le chronogramme sans protocole. Calculer le temps de réponse de τ_1 et sa durée de blocage. Identifier l'inversion de priorité.
2. Même question avec l'héritage de priorité (PIP).
3. Même question avec le plafond de priorité immédiat (ICPP). Quel est le plafond de S ?
4. Comparer les temps de réponse de τ_2 et τ_3 dans les trois cas. Commenter.

Exercice 4 — Analyse du temps de réponse avec blocage

Trois tâches RM : $\tau_1(C = 2, T = 10)$, $\tau_2(C = 3, T = 15)$, $\tau_3(C = 5, T = 30)$, $D_i = T_i$. τ_1 et τ_3 partagent la ressource R_A (τ_1 l'utilise 1 unité, τ_3 2 unités) ; τ_2 et τ_3 partagent R_B (τ_2 : 1 unité, τ_3 : 2 unités). Le protocole est PCP.

1. Donner les plafonds de R_A et R_B et les facteurs de blocage B_1, B_2, B_3.
2. Calculer les pires temps de réponse avec et sans blocage. Le système est-il ordonnançable ?
3. Vérifier avec le test de Liu et Layland étendu au blocage.

Exercice 5 — Interblocage

Les tâches `Affiche` et `Journal` utilisent toutes deux le LCD et l'UART :

```
Affiche :  P(sLCD) ; P(sUART) ; ... ; V(sUART) ; V(sLCD)
Journal :  P(sUART) ; P(sLCD) ; ... ; V(sLCD) ; V(sUART)
```

1. Montrer qu'un interblocage est possible. Quelles conditions de Coffman sont réunies ?
2. Proposer deux corrections.

TD 5 — RTOS du compilateur CCS C et environnement Proteus

Microcontrôleur : PIC18F45K22, quartz 16 MHz, PLL ×4 ($F_{\text{osc}} = 64$ MHz).

Exercice 1 — Analyse d'un programme multitâche

séance 5

On considère le programme `leds_rtos.c` du cours (quatre tâches `task_B0` à `task_B3`, périodes 250 ms, 500 ms, 1 s, 2 s, `minor_cycle = 10ms`).

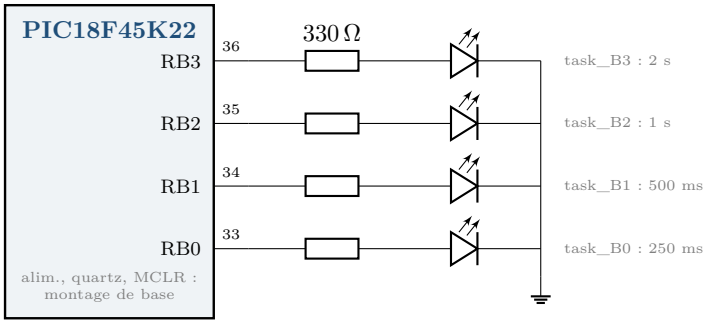


FIGURE 5.1 – Montage du programme `leds_rtos.c` (le montage de base — alimentation, quartz 16 MHz, MCLR — n'est pas représenté).

1. Commenter chaque ligne de la configuration (`#use delay`, `#use rtos`, `#task`).
2. Quelle est la fréquence de clignotement de chaque LED ? Tracer l'allure des quatre signaux sur 4 s.
3. Quelle est l'hyperpériode du système ? Combien de tâches sont exécutées dans le cycle mineur commençant à $t = 2$ s ?
4. Peut-on donner à `task_B0` une période de 25 ms ? de 300 ms ? un paramètre `max = 20ms` ? Justifier.
5. Quel état de tâche (prêt, élu, bloqué) correspond à une tâche CCS dont la date d'exécution n'est pas encore échue ? Le RTOS de CCS est-il préemptif ?

Exercice 2 — Recherche d'erreurs

séance 5

Relever toutes les erreurs (de syntaxe, d'utilisation du RTOS ou de conception temps réel) du programme suivant, puis le corriger.

```
#include <18F45K22.h>
#use delay(clock=64MHz, crystal=16MHz)
#use rtos(timer=0, minor_cycle=10ms)
int8 sem;

#task(rate=25ms, max=20ms)
void Lecture()
{
    while (input(PIN_A1) == 1) ;            // attend un appui
```

```
    rtos_msg_send(Affichage, read_adc());
}

#task(rate=100ms, max=5ms)
void Affichage()
{
    output_b(rtos_msg_read());
    delay_ms(500);                          // laisser le temps de voir
}

void main()
{
    rtos_wait(sem);
    rtos_run();
}
```

Exercice 3 — Écriture d'un programme : chenillard pilotable

séance 5

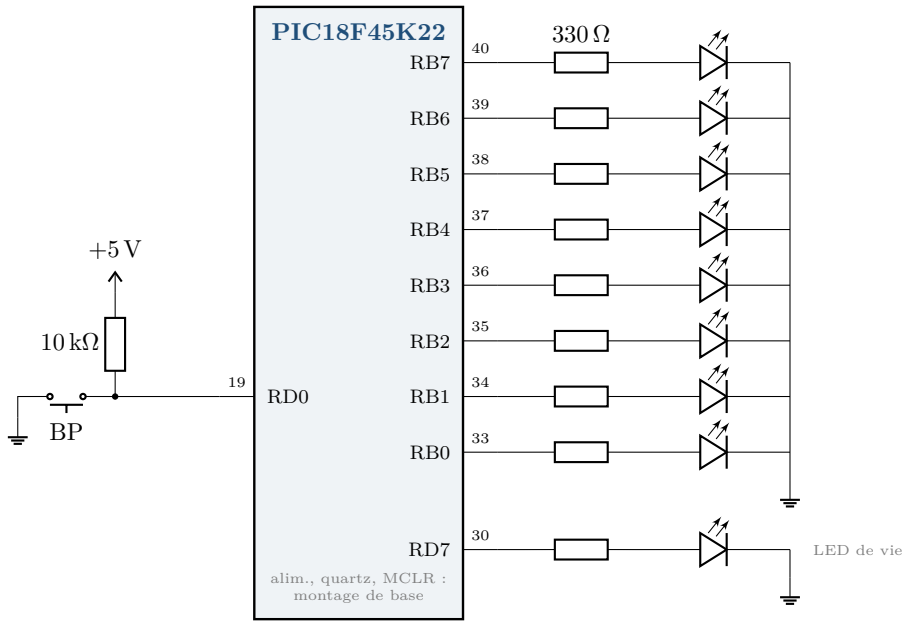


FIGURE 5.2 – Montage du chenillard pilotable.

Huit LED sont reliées au PORTB, un bouton (actif bas) à RD0 et une LED de vie à RD7. Écrire un programme CCS à trois tâches :

- `Live` : fait clignoter RD7 (période 1 s) ;
- `Chenillard` : allume une seule LED à la fois et la décale toutes les 100 ms, dans le sens courant ;
- `Bouton` : scrute RD0 toutes les 20 ms ; à chaque appui, envoie à `Chenillard` un message lui demandant d'inverser le sens.

Préciser le cycle mineur choisi et justifier.

Exercice 4 — Dimensionnement temporel

Un programme CCS comporte les tâches suivantes (durées mesurées par `rtos_stats`) :

Tâche	`rate`	durée mesurée
`Acquisition`	5 ms	120 µs
`Filtrage`	10 ms	800 µs
`Commande`	20 ms	1,5 ms
`Affichage`	200 ms	3,2 ms
`Liaison`	100 ms	2 ms

1. Proposer `minor_cycle` et les paramètres `max`.
2. Calculer la charge processeur U.
3. Que se passe-t-il au cycle mineur commençant à $t = 200$ ms ? Les périodes sont-elles respectées ? Proposer une amélioration.

Exercice 5 — Mise en œuvre sous Proteus

On s'appuie sur le schéma de l'exercice 3. Décrire les étapes permettant de simuler sous Proteus le programme de l'exercice 3 : composants à placer, réglages du microcontrôleur, fichier à charger, instruments utiles pour vérifier les périodes des tâches.

TD 6 — Études de cas avec le RTOS de CCS

Exercice 1 — Étude de cas 1 : générateur de nombres aléatoires

séance 6

Huit LED sont reliées au PORTB d'un PIC18F45K22 (64 MHz), un bouton-poussoir (actif bas) à RD0, une LED à RD7 ; un terminal est connecté à l'UART1 (9600 bauds). Quatre tâches :

- `Live` : toutes les 200 ms, fait clignoter RD7 ;
- `Generator` : incrémente en continu une variable de 0 à 255 (toutes les 1 ms) ; lors d'un appui sur le bouton, ou lorsque la tâche `Terminal` lui transmet le caractère `'N'`, envoie la valeur courante aux tâches `Display` et `Terminal` par messages ;
- `Display` : lit le nombre dans sa file et l'envoie aux LED du PORTB ;
- `Terminal` : transmet à `Generator` les caractères reçus de l'UART, et envoie au PC les nombres reçus de `Generator`.

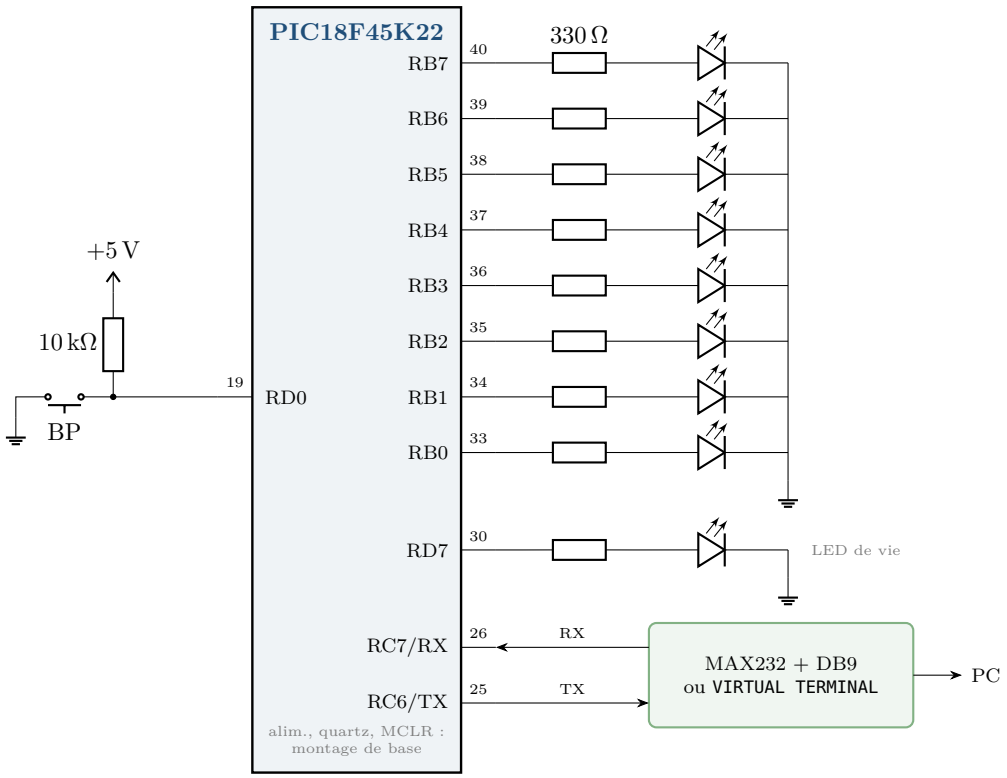


FIGURE 6.1 – Montage de l'étude de cas n° 1 (montage de base non représenté).

1. Représenter le graphe des tâches et des messages. Préciser la taille des files.
2. Pourquoi le nombre affiché peut-il être considéré comme aléatoire ?

3. À 9600 bauds (8N1), combien de temps prend l'émission d'un caractère ? de la chaîne `"255\r\n"` ? Pourquoi ne faut-il pas appeler `printf` directement dans une tâche quand `minor_cycle = 1ms` ? Proposer une solution.
4. Écrire le programme complet.
5. Décrire la validation sous Proteus.

Exercice 2 — Étude de cas 2 : voltmètre avec liaison série RS232

séance 6

La tension d'un capteur LM35 (10 mV/°C) est lue sur AN0 ($V_{\text{ref}} = 5$ V, CAN 10 bits) et envoyée sur la liaison série vers un PC. Trois tâches :

— `Live` : toutes les 200 ms, fait clignoter la LED de RD7 ;

— `Get_voltage` : toutes les 2 s, lit AN0, calcule la tension en mV et la température, et les range dans des variables partagées ;

— `To_RS232` : toutes les secondes, envoie la tension et la température au PC.

1. Proposer un schéma électronique (liaison PC par MAX232 ou `COMPIM`/`VIRTUAL TERMINAL` sous Proteus).
2. Calculer la résolution en tension et en température. Quelle tension et quel code numérique pour 37 °C ?
3. Pourquoi un sémaphore est-il demandé pour l'accès aux données ? Dans quel cas serait-il réellement indispensable avec ce RTOS ?
4. Choisir `minor_cycle` et les paramètres `max` sachant qu'une trame fait environ 26 caractères. Écrire le programme.
5. La température n'étant mise à jour que toutes les 2 s, que voit-on sur le terminal ? Est-ce gênant ?

Exercice 3 — Projet de synthèse : analyse des tâches

Le projet de synthèse fait tourner dans le même PIC18F45K22 un générateur de nombres aléatoires, l'asservissement PID de la température d'un four et une LED clignotante. Les tâches sont déclarées ainsi :

```
#task(rate=200ms, max=10us)          Live_
#task(rate=10ms, max=20us, queue=1)  Rand_Gen
#task(rate=10ms, max=30us, queue=1)  Display
#task(rate=10ms, max=40us, queue=1)  Terminal_D
#task(rate=1ms, max=500us)           PID_Crt
#task(rate=10ms, max=500us)          Display_LCD
```

1. Calculer la charge processeur maximale. Quel `minor_cycle` choisir ?
2. Critiquer la période de `PID_Crt` sachant que la constante de temps du four est de l'ordre de la minute. Proposer une valeur.
3. `Display_LCD` écrit deux lignes de 16 caractères ; un caractère prend environ 50 µs avec le pilote `lcd.c`. Le paramètre `max` est-il réaliste ? Proposer une correction.
4. Le partage de la consigne entre `Terminal_D` et `PID_Crt` « ne nécessite aucune synchronisation ». Dans quelles conditions est-ce vrai ?

TD 7 — Conception de l'asservissement de température

On reprend le four piloté par un PIC18F45K22 à 64 MHz du chapitre 5 du cours.

Exercice 1 — Chaînes de mesure et d'action

séance 7

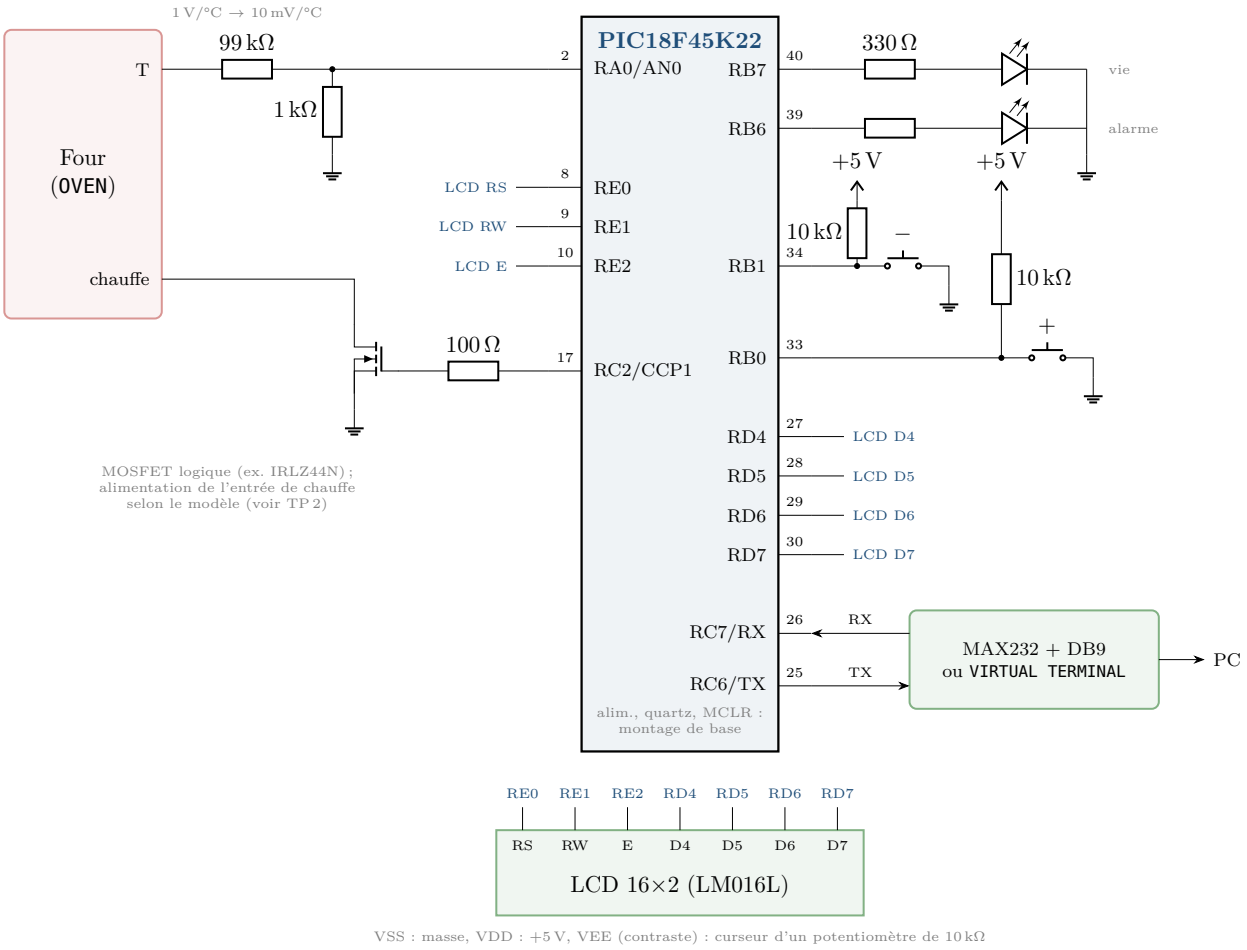


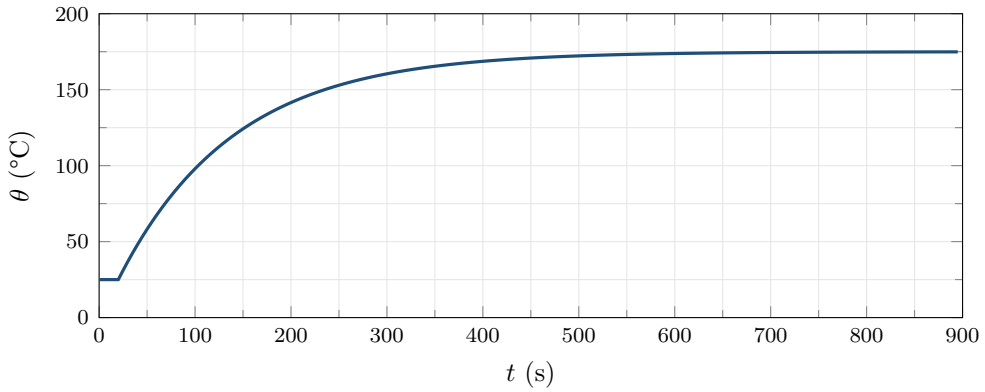
FIGURE 7.1 – Schéma du régulateur de température (cours, chapitre 5).

1. La température est convertie à raison de 10 mV/°C et lue par le CAN 10 bits ($V_{\text{ref}} = 5$ V). Donner la relation entre le code N et la température, la résolution et la plage de mesure. Quelle valeur de N correspond à la limite de sécurité de 260 °C ?
2. La PWM est produite par CCP1 avec le Timer2 (prédiviseur 1, 4 ou 16). Calculer $PR2$ et le prédiviseur pour une fréquence de 1 kHz. Quelle est alors la résolution du rapport cyclique ? Écrire les instructions CCS de configuration et l'instruction appliquant $u = 37{,}5$ %.

Exercice 2 — Identification et réglage du PID

séance 7

À partir de l'équilibre à 25 °C, on applique un échelon de commande de 0 à 50 % et on relève la réponse suivante :

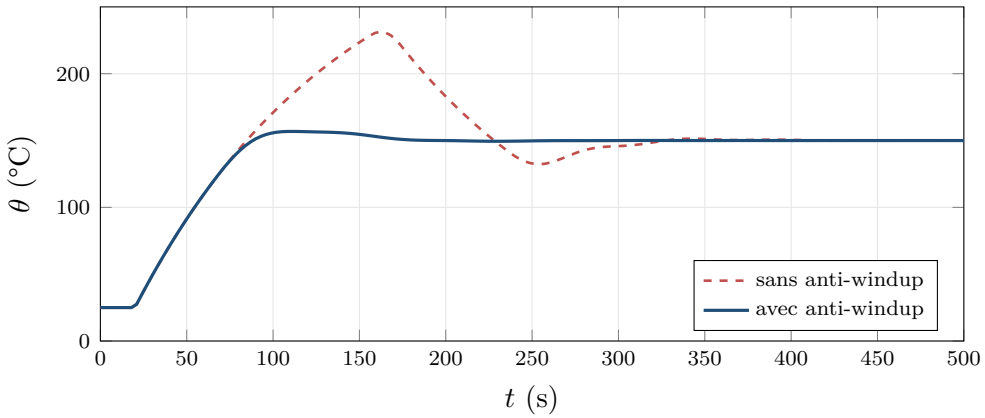


1. Identifier un modèle $Ke^{-\theta_r s}/(1+\tau s)$ par la méthode de Broïda (on relèvera $t_{28} \approx 59{,}4$ s et $t_{40} \approx 81{,}2$ s).
2. Calculer les réglages PID de Ziegler-Nichols et de Chien-Hrones-Reswick (0 % de dépassement).
3. Choisir la période d'échantillonnage T_e de la tâche PID. Justifier.
4. Écrire la loi de commande discrète en forme incrémentale et calculer q_0, q_1, q_2 pour le réglage CHR et $T_e = 0{,}5$ s.

Exercice 3 — Emballement de l'intégrale

séance 7

La figure ci-dessous compare les réponses obtenues avec le réglage ZN, avec et sans anti-windup, pour un échelon de consigne de 25 à 150 °C.



1. Relever le dépassement dans les deux cas. Expliquer le phénomène.
2. Écrire en C le calcul du PID (forme position) avec anti-windup par intégration conditionnelle.

Exercice 4 — Découpage en tâches et validation temporelle

séance 7

Le logiciel est organisé en sept tâches CCS : `Live` (200 ms, 10 µs), `Clavier` (50 ms, 40 µs), `Mesure` (100 ms, 200 µs), `PID` (500 ms, 900 µs), `Securite` (100 ms, 20 µs), `Afficheur` (250 ms, 2,5 ms), `Terminal` (10 ms, 60 µs).

1. Calculer la charge processeur et l'hyperpériode. Choisir `minor_cycle`.
2. Vérifier qu'aucun cycle mineur ne déborde.
3. Quelles données sont partagées ? Par quelles tâches ? Quels mécanismes de protection proposer ?
4. La tâche `Terminal` envoie chaque seconde une trame de 23 caractères. Pourquoi ne pas utiliser `printf` ? Pourquoi la réception est-elle faite sous interruption ?

5. Calculer le pire délai entre le franchissement de 260 °C et la coupure de la chauffe. Le cahier des charges ($\leq$ 200 ms) est-il satisfait ? Proposer une amélioration.

TD 8 — Révision générale : sujet type d'examen

Durée : 2 heures — Documents non autorisés — Calculatrice autorisée — Barème indicatif sur 20 points

Exercice 1 — Questions de cours 5 pts

1. (1 pt) Définir : système temps réel, ordonnanceur. Un système temps réel est-il un système rapide ? Justifier.
2. (1 pt) Donner les états d'une tâche dans un RTOS et la condition de chaque transition.
3. (1 pt) Quel algorithme d'ordonnancement utilisent FreeRTOS et le RTOS de CCS C ? Lequel est préemptif ?
4. (1 pt) Qu'est-ce que l'inversion de priorité ? Citer deux protocoles qui la bornent.
5. (1 pt) QCM — L'ordonnancement EDF : (a) privilégie toujours les tâches de plus courte période ; (b) est un algorithme statique ; (c) privilégie toujours les tâches de plus proche échéance.

Exercice 2 — Ordonnancement RM et EDF 8 pts

On considère trois tâches périodiques indépendantes, synchrones, à échéance sur requête : $\tau_1(C = 1, T = 4)$, $\tau_2(C = 2, T = 6)$, $\tau_3(C = 3, T = 8)$.

1. (1 pt) Calculer le facteur d'utilisation et l'hyperpériode.
2. (1 pt) Que peut-on conclure du critère de Liu et Layland ? Expliquer pourquoi ce critère est suffisant mais pas nécessaire.
3. (2 pts) Montrer par l'analyse du temps de réponse que RM n'est pas valide.
4. (2 pts) Tracer le chronogramme RM sur l'hyperpériode et repérer l'échéance manquée.
5. (2 pts) Tracer le chronogramme EDF et montrer qu'il est valide. Calculer le taux d'occupation du processeur.

Exercice 3 — Programmation avec le RTOS de CCS 7 pts

Un PIC18F45K22 (64 MHz) surveille le niveau d'une cuve. Le capteur (0–5 V pour 0–200 cm) est sur AN0, une pompe est commandée par RC0, une LED de vie est sur RD7. Cahier des charges :

— `Live` : clignotement de RD7, période 1 s ;

— `Niveau` : toutes les 50 ms, mesure le niveau en cm et le range dans une variable partagée ;

— `Pompe` : toutes les 100 ms, met la pompe en marche si le niveau est inférieur à 50 cm et l'arrête au-dessus de 150 cm (hystérésis) ;

— `Rapport` : toutes les 2 s, envoie le niveau et l'état de la pompe sur l'UART (9600 bauds), trame d'environ 20 caractères.

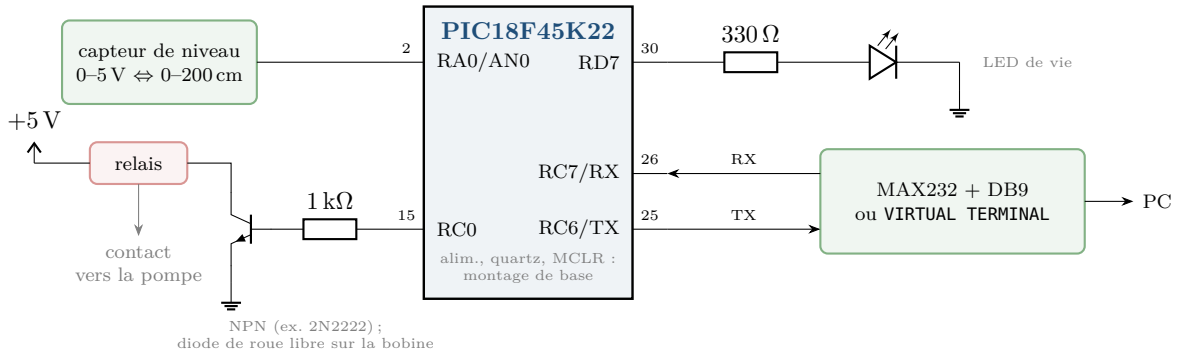


FIGURE 8.1 – Montage de la surveillance de cuve (montage de base non représenté).

1. (1 pt) Choisir `minor_cycle` et les paramètres `max`. Justifier.
2. (1 pt) Identifier les ressources partagées et le mécanisme de protection.
3. (4 pts) Écrire le programme complet.
4. (1 pt) Pourquoi la tâche `Pompe` ne doit-elle pas contenir `while (niveau < 150) ;` ?