Université Djilali BOUNAAMA, Khemis Miliana
مليانة خميس بونعامة جيلالي جامعة
Faculté des Sciences et de la Technologie
والتكنولوجيا العلوم كلية

Série d'exercices 9: Partie B : Applications Automatiques

Exercice 1 : Discrétisation d'un PID continu

Un PID continu est donné par :

$$U(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt}$$

On utilise une période d'échantillonnage T_s. Discrétisez ce PID par la méthode d'Euler avant (pour l'intégrale et la dérivée). Donnez l'équation récurrente du PID numérique.

Exercice 2 : Anti-windup simple

Un actionneur est saturé entre $u_{min}=-10$ et $u_{max}=10$. Le PID calcule u_{pid}.
L'intégrale $I_n=I_{n-1}+K_iT_se_n$. Proposez une méthode anti-windup (back-calculation) simple qui ajuste l'intégrale quand la sortie est saturée.

Exercice 3 : PID numérique sur microcontrôleur

Écrire une tâche RTOS (pseudo-code) qui exécute un PID à 100 Hz, lit un capteur via ADC, calcule la commande, et l'envoie au PWM. Gérer l'anti-windup.

Exercice 4 : Architecture RTOS pour thermostat connecté

Énoncé
Décrivez l'architecture logicielle (tâches, synchronisation, communication) d'un thermostat connecté avec :

- Capteur de température (I2C)
- Chauffage (PWM)
- Écran LCD (affichage)
- Communication WiFi (envoi données)
- Bouton de réglage consigne

Exercice 5: Validation temps réel d'un véhicule autonome

Un véhicule utilise 3 tâches :

- Tâche A : odométrie (1 kHz)
- Tâche B : contrôle PID direction (100 Hz)
- Tâche C : détection obstacle (10 Hz)

On mesure les temps d'exécution :
$T_A = 0.3ms$, $T_B = 1.2ms$, $T_C = 5ms$.
Le système est-il temps réel (sous réserve que les délais soient respectés) ? Proposez une ordonnancement.

Université Djilali BOUNAAMA, Khemis Miliana
مليانة خميس بونعامة جيلالي جامعة
Faculté des Sciences et de la Technologie
والتكنولوجيا العلوم كلية

Solution Série 9

Exercice 1 :

Exercice 2 :

```
u_pid = Kp*e + I + Kd*(e - e_prev)/Ts;
u = saturate(u_pid, umin, umax);
if (u != u_pid)
   I -= K_aw * (u_pid - u);
else
   I += Ki*Ts*e;
```

pas préempter → inversion résolue.

Exercice 3 :

```
void task_pid(void *arg) {
  float e, e_prev = 0, I = 0, u;
  const float Kp=2, Ki=1, Kd=0.5, Ts=0.01;
  const float umax=255, umin=0;
  TickType_t last_wake = xTaskGetTickCount();

  while(1) {
    // Lecture capteur
    float mesure = read_ADC();
    e = consigne - mesure;

    // Calcul PID
```

```
        float u_pid = Kp*e + I + Kd*(e - e_prev)/Ts;
        u = saturate(u_pid, umin, umax);

        // Anti-windup
        if (u != u_pid)
            I -= (u_pid - u); // simple correction
        else
            I += Ki*Ts*e;

        // Commande actionneur
        set_PWM_duty(u);

        e_prev = e;
        vTaskDelayUntil(&last_wake, pdMS_TO_TICKS(10));
    }
}
```

Exercice 4

Tâches (sous FreeRTOS) :

1. **Acquisition** (période 1 s) : lit capteur → place mesure dans queue `temp_queue`.
2. **Contrôle PID** (période 0.5 s) : lit consigne (variable partagée protégée par mutex), lit dernière mesure, calcule commande → envoie via queue `cmd_queue`.
3. **Actionneur** : attend sur `cmd_queue`, applique PWM.
4. **Affichage** (période 0.5 s) : lit mesure et consigne → met à jour LCD.
5. **WiFi** (période 5 s) : envoie température et état via TCP/UDP.
6. **Bouton** (interruption + tâche) : modifie consigne avec mutex.

Synchronisation : queues pour données échantillonnées, mutex pour consigne, sémaphore binaire pour interruption bouton.

Exercice 5

Charge processeur :

- A : $0.3 \times 1000 = 300$ µs/ms=30%
- B : $1.2 \times 100 = 120$ µs/ms=12%
- C : $5 \times 10 = 50$ µs/ms=5%

Charge totale = $30+12+5=47\%$ → OK (pas de saturation).

Mais respect des échéances :

- A doit être exécutée toutes les 1 ms → possible si priorité haute.
- B toutes les 10 ms, C toutes les 100 ms.

Ordonnancement RMS (Rate Monotonic) :
Priorité : A (période 1 ms) > B (10 ms) > C (100 ms).
Vérification : $CA=0.3/1=0.3$, $CB=1.2/10=0.12$, $CC=5/100=0.05$.
Test de Liu & Layland : $0.3+0.12+0.05=0.47<0.69$ (pour 3 tâches) → faisable.

Donc le système est temps réel avec cette affectation de priorités.