Chapitre 9

Applications Automatiques

9.1 Introduction

Ce chapitre final a pour vocation de concrétiser l'ensemble des compétences acquises tout au long du cours en les inscrivant dans le cadre applicatif de l'automatique et du contrôle-commande. Il s'agit non seulement de maîtriser la théorie des régulateurs numériques, mais surtout de savoir les implémenter efficacement sur une architecture embarquée contrainte, en tirant parti des mécanismes offerts par le noyau temps réel. La structure en deux semaines permet une progression naturelle : d'abord l'étude approfondie d'un composant algorithmique fondamental qu'est le régulateur PID numérique, puis sa mise en œuvre au sein d'un projet intégrateur complet qui mobilise l'ensemble des briques matérielles et logicielles abordées précédemment.

9.2 Conception et implémentation d'un régulateur PID numérique

9.2.1 Fondements théoriques du régulateur PID

La première semaine débute par un rappel des principes fondamentaux du régulateur PID, qui demeure l'algorithme de contrôle le plus répandu dans l'industrie en raison de sa simplicité conceptuelle et de son efficacité pour une large classe de systèmes. Nous établirons les trois termes constitutifs de la commande : l'action proportionnelle (P), qui réagit à l'erreur instantanée et assure la rapidité de réponse ; l'action intégrale (I), qui élimine l'erreur statique en accumulant l'erreur passée ; et l'action dérivée (D), qui anticipe les variations futures en réagissant à la vitesse de variation de l'erreur, améliorant ainsi la stabilité et le dépassement. Nous distinguerons les deux formes algorithmiques principales que sont la forme parallèle classique et la forme série (ou interactive), et nous discuterons des avantages respectifs de chaque structure pour une implantation sur microcontrôleur.

9.2.2 Discrétisation et implémentation numérique

Le passage d'une forme continue à une forme numérique nécessite une discrétisation soigneuse. Nous étudierons les méthodes les plus adaptées à l'embarqué, principalement la transformation bilinéaire (ou méthode de Tustin) et la méthode d'Euler avant pour l'intégrateur. L'implémentation algorithmique sera détaillée sous forme de code structuré, en utilisant une représentation en virgule fixe ou en virgule flottante selon les capacités du microcontrôleur cible. Nous insisterons sur l'importance de la période d'échantillonnage, qui doit être choisie de manière cohérente avec la dynamique du système à contrôler et avec les

capacités de calcul disponibles. Une attention particulière sera portée à l'organisation du code : encapsulation du régulateur dans une structure de données dédiée, fonctions d'initialisation, de mise à jour des gains, et de calcul de la commande.

9.2.3 Problématiques de mise au point et limitation de l'intégrale

La mise au point d'un régulateur PID en conditions réelles constitue une étape critique qui sera abordée en détail. Nous présenterons les méthodes empiriques de réglage, notamment la méthode de Ziegler-Nichols en boucle fermée, ainsi que des approches plus systématiques comme le placement de pôles ou l'optimisation sous contraintes. Une problématique centrale sera approfondie : celle de la limitation de l'intégrale, communément appelée anti-windup. Nous analyserons le phénomène de saturation de l'actionneur, qui conduit à une accumulation indésirable du terme intégral lorsque l'erreur persiste en régime saturé, provoquant des dépassements importants et une dégradation de la stabilité. Plusieurs stratégies de compensation seront présentées et comparées : la simple désactivation de l'intégration en cas de saturation, la méthode dite de « back-calculation » qui soustrait l'erreur de saturation à l'entrée de l'intégrateur, ainsi que l'approche par « conditional integration » qui suspend l'intégration tant que la commande reste saturée.

9.2.4 Intégration dans une architecture RTOS

L'implémentation du régulateur PID ne saurait se réduire à un simple calcul algorithmique ; elle doit s'inscrire dans une architecture logicielle temps réel cohérente. Nous verrons comment structurer le système autour de plusieurs tâches collaboratives. Une tâche dédiée à l'acquisition des capteurs sera chargée de lire périodiquement la mesure à contrôler, en utilisant soit un timer matériel pour générer l'interruption d'échantillonnage, soit un timer logiciel du RTOS. Une tâche de calcul de la commande exécutera l'algorithme PID à chaque période d'échantillonnage, en s'appuyant sur des mécanismes de synchronisation comme les sémaphores binaires pour déclencher le traitement à intervalle régulier. Enfin, une tâche d'application de la commande pilotera l'actionneur (moteur, vanne, chauffage, etc.) via une sortie PWM ou un convertisseur numérique-analogique. L'utilisation de files de messages permettra de transmettre les données entre les tâches de manière robuste et non-bloquante. Nous discuterons également des critères de priorisation des tâches dans un tel système : la tâche d'acquisition doit bénéficier d'une priorité élevée pour ne pas manquer d'échantillons, tandis que la tâche de calcul, bien que critique, peut accepter une priorité légèrement inférieure tant qu'elle respecte ses échéances.

9.3 Projet intégrateur – réalisation d'un système de contrôle-commande

9.3.1 Présentation du projet et cahier des charges

La seconde semaine est entièrement dédiée à un projet intégrateur dont l'objectif est de mettre en œuvre l'ensemble des compétences acquises dans un cas concret. Plusieurs déclinaisons seront proposées pour couvrir différents champs d'application de l'automatique. Le premier scénario possible est la réalisation d'un système de régulation de vitesse de moteur à courant continu : un codeur incrémental ou un capteur à effet Hall fournit la mesure de vitesse, une

commande PWM pilotée par un PID ajuste la tension d'alimentation du moteur, et une interface utilisateur (potentiomètre ou liaison série) permet de fixer la consigne. Le deuxième scénario propose un thermostat connecté : une sonde de température (DS18B20 ou thermistance) alimente le régulateur qui commande un élément chauffant par relais ou par variateur, avec affichage local sur écran LCD et communication vers une interface de supervision. Le troisième scénario, plus ambitieux, consiste en un petit véhicule autonome : la régulation de vitesse sur chaque roue, l'asservissement en direction, et éventuellement la poursuite de ligne ou l'évitement d'obstacles, offrent un cadre riche pour appliquer les concepts temps réel.

9.3.2 Architecture logicielle modulaire

L'un des objectifs pédagogiques majeurs de ce projet est la conception d'une architecture logicielle modulaire et maintenable. Nous guiderons les apprenants dans la structuration du code en couches distinctes : une couche matérielle (Hardware Abstraction Layer) qui encapsule les accès aux périphériques (GPIO, timers, UART, I2C, PWM) ; une couche middleware regroupant les drivers des capteurs et actionneurs ; une couche applicative contenant le régulateur PID et la logique de contrôle ; et une couche de supervision assurant l'interface utilisateur et la communication. Cette séparation des préoccupations facilite la réutilisation du code, les tests unitaires et la maintenabilité. Nous insisterons également sur l'importance d'une configuration centralisée des paramètres, rendue accessible soit par compilation, soit par configuration dynamique via une liaison série.

9.3.3 Orchestration des tâches sous RTOS

L'organisation temporelle du système repose sur l'utilisation systématique des services du RTOS. Nous définirons avec précision la liste des tâches nécessaires : une tâche d'acquisition périodique des capteurs, une tâche de calcul du PID déclenchée à la fin de chaque acquisition, une tâche de commande des actionneurs, une tâche de communication pour l'échange de données avec une interface de supervision, une tâche de supervision locale (gestion d'écran, de boutons), et éventuellement une tâche de journalisation des données. Pour chaque tâche, nous déterminerons sa priorité, sa période d'exécution (si elle est périodique), et ses interactions avec les autres tâches. Les mécanismes de synchronisation et de communication seront mis en œuvre de manière rigoureuse : sémaphores binaires pour le déclenchement périodique, mutex pour la protection des ressources partagées (comme l'affichage ou les données globales), et files de messages pour le passage des consignes, des mesures et des paramètres de réglage.

9.3.4 Validation du comportement temps réel

La dernière partie du projet est consacrée à la validation rigoureuse du comportement temps réel du système final. Nous aborderons les méthodes de test spécifiques aux systèmes temps réel : vérification du respect des échéances (deadlines) pour les tâches critiques, mesure de la latence d'interruption et de la gigue (jitter) sur la période d'échantillonnage, analyse des temps d'exécution des tâches via des sorties sur des broches GPIO observables à

l'oscilloscope. Nous introduirons des techniques de traçage logiciel (tracing) permettant de visualiser la séquence d'exécution des tâches et de détecter d'éventuels phénomènes d'inversion de priorité ou de blocages (deadlocks). Enfin, des tests de robustesse seront menés : injection de charges de calcul supplémentaires pour observer le comportement en surcharge, simulation de défauts sur les capteurs pour valider les mécanismes de sécurité, et tests de longue durée pour vérifier l'absence de dérive mémoire ou de comportements indéterminés.

9.3.5 Synthèse et perspectives

Le projet intégrateur se conclut par une phase de synthèse au cours de laquelle les apprenants présentent leur réalisation, discutent des choix d'architecture effectués, des difficultés rencontrées et des solutions apportées. Cette étape permet de consolider les acquis en confrontant la théorie aux contraintes pratiques de la mise en œuvre. En ouverture, nous évoquerons les perspectives offertes par les techniques plus avancées du contrôle-commande embarqué : commande prédictive, commande adaptative, fusion de capteurs par filtrage de Kalman, et déploiement sur des architectures multi-cœurs. L'objectif est de donner aux apprenants les clés pour poursuivre leur exploration dans le vaste domaine des systèmes embarqués temps réel appliqués à l'automatique, en leur fournissant une base solide tant sur le plan théorique que pratique.