Trabalho Prático Final de ATR
Professor: Armando Alves Neto
Dep. de Engenharia Eletrônica – Escola de Engenharia
Universidade Federal de Minas Gerais

Instruções:

1. Este trabalho prático tem por objetivo auxiliar a consolidação de conceitos vistos em classe, além da familiarização com a programação concorrente, dispositivos de entrada e saída, e comunicação entre processos.
2. Pode ser feito em dupla.
3. Data de entrega: **19/11/2022 (impreterivelmente)**
4. **Trabalhos copiados serão anulados.**

Descrição do problema:

Uma empresa de ensaios elétricos possui um banco com 30 motores que utiliza para diversos testes e treinamentos. Mas apesar da quantidade, e por questões de segurança, apenas 12 deles podem ser ligados simultaneamente. Além disso, dois motores em sequência não podem operar ao mesmo tempo (por exemplo, não se pode ligar os motores 1 e 2 ou 2 e 3, etc). Os motores são de corrente contínua, de modo que seu comportamento dinâmico pode ser representado pelo diagrama de blocos a seguir:

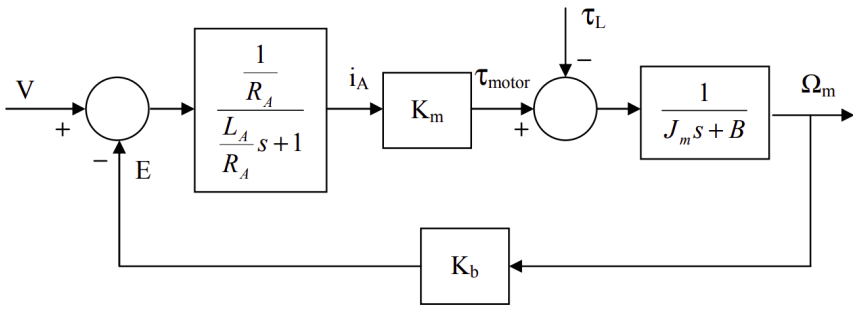

Figura 1: Diagrama de blocos de um motor c.c. do banco de motores.

Assim, temos que:

- V representa a tensão de armadura (entrada);
- R_A e L_A são a resistência e a indutância do circuto de armadura;
- i_A é a corrente de armadura;
- K_m é a constante de torque τ_{motor};
- τ_L é um torque de carga (perturbação);
- J_m e B são o momento de inércia e o atrito viscoso do motor;
- Ω_m é a velocidade de rotação do motor (saída);
- e K_b é a constante elétrica.

O que deve ser feito:

- Crie um programa que dispare 31 threads (30 threads *motor_thread_i* com i = 1,...,30 e uma thread *control_thread*):

  - Cada thread *motor_thread_i* deve simular periodicamente a equação dinâmica de um dos motores, dada pelo diagrama da Fig.1. O período dessa simulação deve ser de no mínimo 100ms. Escolha parâmetros iguais para todos os motores do banco, especificando-os no relatório final.
  - A *control_thread* deve efetuar o controle de velocidade dos motores a uma taxa de 200ms. Implemente uma lei de controle que lhe for mais simples, e que mantenha os motores funcionando a metade da velocidade máxima Ω_{max} por cerca de 1min. Essa velocidade será dada por meio de um sinal de referência $\widetilde{\Omega}_m$. Lembre-se da restrição de que apenas 12 motores podem ser ligados simultaneamente, assim, escolha 12 motores quaisquer para controlar. Assuma condições iniciais nulas e $\tau_L = 0$.

- Adapte o programa acima para que ele contenha mais duas novas threads, (*logger_thread* e *interface_thread*):

  - A *interface_thread* será responsável pela leitura de valores para as referências de velocidade dos motores do banco. Esses valores serão fornecidos pelo usuário via teclado, sendo posteriormente lidos pela thread de controle implementada na parte 1 do trabalho. Deverá ser fornecido a cada leitura o índice do motor $i = \{1, \ldots, 10\}$ e o valor correspondente de referência Ω^i_{ref} (lembrando que no máximo 12 motores podem ser ligados simultamente).
  - A *logger_thread* será responsável por salvar algumas informações do processo em disco. Para isso, ela deverá criar um arquivo "log.txt", onde salvará periodicamente (a cada segundo) os valores de velocidade de cada motor Ω, além do *time stamp*.
  - Toda ministragem de tempo deverá ser realizada por meio de dispositivos temporizadores.

- Por fim, ajuste o programa para que ele dispare um novo processo (via **fork()**, **multiprocess**, etc...) chamado *synoptic_process*:

  - O *synoptic_process* deve emular um sistema supervisório para a teleoperação do controle do sistema do banco. Esse processo deve trocar informações com a *soft-PLC_thread*, via interface de *socket* TCP/IP, mostrando na tela do computador os valores de Ω_i, e permitindo que se leia do teclado valores de Ω^i_{ref}.
  - Todas essas informações devem ser registradas em um arquivo denominado "historiador.txt". Aqui, a interface gráfica do sinótipo deve ser a mais simples possível, sem o uso de outras bibliotecas. Use o próprio terminal para das saídas das variáveis.

obs: Perceba que tanto as variáveis manipuladas V's quanto as variáveis controladas Ω_m's devem ser declaradas como variáveis globais, e consequentemente precisam ser protegidas pelo uso de diretivas do SO.

O que deve ser entregue:

- (50% da nota) Códigos contendo os arquivos para compilação do projeto (cpp, hpp, etc). Também pode ser utilizada a linguagem Python. Devem ser utilizadas apenas funções da STD/C++ ou libs Python padrão.
- (10% da nota) Um documento "readme.txt" com instruções de compilação e operação do sistema.
- (40% da nota) Relatório de descrição do trabalho em ".pdf".

Dica: para implementar o motor, o diagrama de blocos da Fig.1 pode ser descrito por meio das seguintes equações no domínio da frequência:

$$(L_A s + R_A)\tau_{motor}(s) = K_m V(s) - K_m K_b \Omega_m(s),$$
$$(J_m s + B)\Omega_m(s) = \tau_{motor}(s) - \tau_L(s),$$

correspondentes aos pólos elétrico e mecânico do motor. Considerando condições iniciais nulas, podemos representar essas equações no domínio do tempo (contínuo) como sendo:

$$L_A \dot{\tau}_{motor}(t) + R_A \tau_{motor}(t) = K_m v(t) - K_m K_b \omega_m(t),$$
$$J_m \dot{\omega}_m(t) + B\omega_m(t) = \tau_{motor}(t) - \tau_L(t).$$

Para simular o comportamento do motor, você deve descrever a variação de velocidade do motor $\dot{\omega}_m(t)$ em função da entrada de tensão $v(t)$ e usar um algortimo de integração numérica (Runge-Kutta, triangular, etc) para obter $\omega_m(t)$. Para fins de programação, utilize uma aproximação discreta simples do sistema, por exemplo:

$$L_A \frac{\tau_{motor}(k+1) - \tau_{motor}(k)}{\Delta T} = K_m v(k) - K_m K_b \omega_m(k) - R_A \tau_{motor}(k),$$
$$J_m \frac{\omega_m(k+1) - \omega_m(k)}{\Delta T} = \tau_{motor}(k) - \tau_L(k) - B\omega_m(k),$$

onde ΔT representa o intervalo de amostragem da simulação (quanto menor, mais próximo do tempo contínuo).