Plano de Ensino - Sistemas Embarcados

Listagem | Oferta

Código: 206181 | **Créditos:** 4 | **Turma:** A

Locais e horários: Lab SS, 3as e 5as feiras – 16:00 às 17:50

Objetivos

- Conhecer os fundamentos básicos de sistemas embarcados;
- Compreender o processo de desenvolvimento de software para um sistema embarcado;
- Conhecer as técnicas de desenvolvimento de aplicações e drivers para sistemas embarcados.

Ementa

- Introdução aos sistemas embarcados
- Introdução ao OS Linux
- Desenvolvimento para sistemas embarcados
- Inicialização de sistemas embarcados
- Subsistema de I/O
- Recursos de sistemas I
- Introdução ao RTOS
- Recursos de sistemas II
- Introdução ao LKM
- Introdução aos device drivers

Programa

1. Introdução aos sistemas embarcados
2. Introdução ao OS Linux
    1. Obtendo informações do sistema
    2. Principais comandos
    3. Estrutura de diretórios
    4. Instalação padrão de programas
3. Desenvolvimento para sistemas embarcados
    1. Conceito de cross-platform
    2. Componentes e suas funções (host, target, interface de comunicação, etc)
    3. Processo de geração de imagens
    4. Utilização de makefiles
4. Inicialização em Sistemas Embarcados
    1. Ferramentas do target

    2. Transferência de imagens
    3. Ferramentas para o Target System
    4. Transferência de imagens
    5. Cenários de boot do target
    6. Seqüência de inicialização do target
5. Subsistema de I/Os.
    1. Conceito básico de I/Os
    2. O subsistema de I/Os
    3. Modo de transferência serial e em blocos
    4. Funções de um subsistema de I/O
6. Recursos do sistema I
    1. Processos, sinais e threads
    2. Comunicação e sincronismo entre processos
    3. Programação para redes (sockets)
7. Introdução ao RTOS
    1. Características dos sistemas RT
    2. Conceitos de Hard Real-Time e Soft Real-Time
    3. Conceito de latência e Jitter
    4. Introdução ao framework Xenomai
8. Recursos de sistema II
    1. Tarefas
    2. Alarmes
    3. Mutex
    4. Semáforos
    5. Variáveis condicionais
    6. Fila de mensagens
9. Introdução ao LKM
    1. Introdução
    2. Módulos do kernel
    3. Desenvolvimento de um módulo
    4. Comandos do modutils
10. Introdução aos device drivers
    1. Conceitos básicos sobre device driver
    2. Estrutura de um device driver
    3. Funções de um device driver
    4. Instalação e carregamento de um device driver

Método de Ensino

Aulas divididas de acordo com a técnica Pomodoro, com 3 intervalos de 25 a 30 minutos, intercalados com descansos de 5 minutos:

- 1º intervalo: aula expositiva do professor;
- 2º intervalo: alunos responderão a perguntas teóricas propostas pelo professor;
- 3º intervalo: alunos responderão a perguntas práticas propostas pelo professor.

Dentro do 2º e do 3º intervalos, cada aluno deverá desenvolver ao máximo suas respostas, **aproveitando o tempo disponível à exaustão, como esperado na técnica Pomodoro**. Todas as respostas deverão ser armazenadas em um repositório GitHub público, no mesmo dia em que as questões forem colocadas.
Será utilizada a placa Raspberry Pi em toda a disciplina. Caberá ao aluno decidir qual modelo de Raspberry Pi utilizar, de acordo com suas necessidades no projeto final.

Avaliação

- Dois testes (40% da nota final);
- Projeto de fim de curso (30% da nota final);
- Pontos de controle (20% da nota final).
- Respostas dos ciclos Pomodoro em sala de aula (10% da nota final).

Os alunos formarão duplas para montar um projeto final de maior dificuldade, cobrindo os tópicos vistos em sala de aula. Os projetos serão propostos pelos alunos, e será apresentado um protótipo em funcionamento e o relatório do mesmo, na forma de relatório científico com formatação IEEE, contendo os seguintes tópicos:

1. **Resumo (1 ponto):** Apresentar o projeto de forma geral e em poucas palavras.
2. **Introdução (1 ponto):** Descrição do problema a ser resolvido e revisão bibliográfica do assunto.
3. **Desenvolvimento (5 pontos):** Solução, com justificativa baseada em fundamentos teóricos. O desenvolvimento deverá conter a descrição do hardware e do software:
    1. **Descrição do hardware (2 pontos)**, incluindo a BOM (*bill of materials*) e a montagem (diagrama de blocos, esquemático, descrição textual etc.).
    2. **Descrição do software (3 pontos)**, apresentando e justificando o algoritmo desenvolvido. Esta seção do relatório NÃO CONSISTE em simplesmente replicar o código, e sim em explicar como ele funciona, com base em fluxogramas, pseudo-códigos etc. O código deverá ser apresentado como um apêndice do relatório.
4. **Resultados (2 pontos):** Explicar os experimentos definidos para validar o projeto proposto, seguido de análise crítica dos resultados esperados e obtidos. Em caso de divergências, apontar as possíveis causas.
5. **Conclusão (1 ponto):** Retomar sucintamente os principais pontos do relatório: descrição do problema, solução utilizada e resultados obtidos.

Os pontos de controle serão prévias do projeto final. Cada dia de atraso na apresentação dos pontos de controle acarreta na perda de 0,5 pontos da nota do mesmo. Serão esperados os seguintes resultados para cada ponto de controle:
PC1 Proposta do projeto: justificativa, objetivos, requisitos, benefícios, revisão bibliográfica.
PC2 Protótipo funcional do projeto, utilizando as ferramentas mais básicas da placa de desenvolvimento, bibliotecas prontas etc.
PC3 Refinamento do protótipo, acrescentando recursos básicos de sistema (múltiplos processos e *threads*, *pipes*, sinais, semáforos, MUTEX etc.).
PC4 Refinamento do protótipo, acrescentando recursos de Linux em tempo real.

A entrega de todos os relatórios será feita por *email*, EM FORMATO PDF. Envie SOMENTE UM ARQUIVO contendo o relatório. Em caso de cópias (integrais ou parciais), ambos os relatórios ficarão com nota ZERO. Os diferentes *templates* com formatação IEEE estão disponíveis em:

- Word: http://goo.gl/ghgLsR
- Unix LaTeX: http://goo.gl/p3ExfQ
- Windows LaTeX: http://goo.gl/gAxi96

Bibliografia

Básica:

- Mitchell, M., Oldham, J. & Samuel, A., Advanced Linux Programming, Editora: Newriders, 2001.
- Li, Q. & Yao, C., Real-Time Concepts for Embedded Systems, Editora: CMP Books, 2003.

Complementar:

- Kopetz, H., Real-Time Systems - Design Principles for Distributed Embedded Applications, 2o Edição. Editora Springer.
- Bovet, D. P. & Marco Cesati, M., Understanding the Linux Kernel, 3o Edição, Editora: O'Reilly, 2005.
- Corbet, J., Rubini, A. & Kroah-Hartman, G., Linux device drivers, 3o Edição, Editora: O'Reilly, 2005.
- Karim Yaghmour, Building Embedded Linux Systems, Publisher: O'Reilly, 2003.
- David Kalinsky, Introduction to Real-Time Operating Systems, White paper Xenomai API, www.xenomai.org.