UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Engenharia Elétrica
Av. João Naves de Ávila, 2121, Bloco 3N - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4701/4702 - www.feelt.ufu.br - feelt@ufu.br

PLANO DE ENSINO

1. IDENTIFICAÇÃO

Componente Curricular:	Sistemas Digitais para Mecatrônica						
Unidade Ofertante:	Faculdade de Engenharia Elétrica						
Código:	FEELT49081	Período/Série:	8		Turma:	VB / VD	
Carga Horária:					Natureza:		
Teórica:	30	Prática:	30	Total: 60	Obrigatória:	(X)	Optativa: ( )
Professor(A):	Fábio Vincenzi Romualdo da Silva				Ano/Semestre:	2025/01	
Observações:							

2. EMENTA

Desenvolvimento de sistemas embarcados microprocessados, integração com serviços em rede ou nuvem, interfaces homem/máquina (HMI).

3. JUSTIFICATIVA

Essa disciplina aplica os conceitos teóricos do desenvolvimento de software, de sistemas computacionais conectados e tecnologias eletrônicas para o desenvolvimento de sistemas embarcados. Esses são sistemas computacionais completos e independentes, desenvolvidos para uma tarefa específica e que estão presentes em diversas áreas e aplicações de engenharia.

4. OBJETIVO

Objetivo Geral:

Desenvolvimento de sistemas embarcados com hardware de complexidade média ou alta, com foco em comunicação e interatividade com o usuário, em geral executando sistemas operacionais de alto nível.

Objetivos Específicos:

1. Uso de Linux embarcado ou de sistema operacional equivalente. Construção e aplicação de imagens;
2. Criação de interfaces homem máquina através de toolkits gráficos;
3. Integração e uso de stacks diversos de comunicação;
4. Programação com linguagem de script ou de desenvolvimento rápido;
5. Utilização de serviços em nuvem para automação e controle;
6. Interfaceamento com periféricos de comunicação e informação (GPS, modems, Bluetooth, WiFi, entre outros);
7. Utilização de redes com e sem fio no processo de integração;
8. Atualização de firmware no campo (OTA - Over the Air);
9. Segurança em sistemas embarcados.

5. PROGRAMA

1. Linux Embarcado

- 1.1. Breve histórico sobre UNIX
- 1.2. Linux
- 1.3. Por que utilizar Linux em sistemas embarcados?
- 1.4. Anatomia de um sistema embarcado
- 1.5. Considerações sobre armazenamento
- 1.6. Distribuições Linux para sistemas embarcados

2. Processadores
   - 2.1. Processadores Stand-Alone
   - 2.2. Processadores Integrados (Systems on Chip)
   - 2.3. Outras Arquiteturas
   - 2.4. Plataformas de hardware
3. O Kernel Linux
   - 3.1. Background
   - 3.2. Kernel
   - 3.3. Construção
   - 3.4. Sistemas construtores de Kernel
   - 3.5. Kernel customizados e documentação
   - 3.6. Inicialização
   - 3.7. Fluxo de controle
   - 3.8. Inicializando subsistemas
4. Inicialização do espaço do usuário
   - 4.1. Sistemas de arquivos Root
   - 4.2. O processo de inicialização
   - 4.3. Disco RAM inicial
   - 4.4. Utilizando initramfs
   - 4.5. Shutdown
5. Bootloaders
6. Device Driver
7. Subsistemas MTD
   - 7.1. Introdução
   - 7.2. Partições
   - 7.3. Utilitários
   - 7.4. Conceitos sobre Device Driver
   - 7.5. Módulos
   - 7.6. Métodos
   - 7.7. Device Driver e GPL
8. Ambiente de Desenvolvimento Embarcado
10. Ferramentas de Desenvolvimento
    - 10.1. GNU Debugger
    - 10.2. Ferramentas de Tracing e Profiling
    - 10.3. Utilitários Binários
    - 10.4. Técnicas de Depuração de Kernel
11. Ferramentas de depuração para aplicações em Linux embarcado
12. Linux e Sistemas em Tempo Real
    - 12.1. O que é um sistema de Tempo Real
    - 12.2. Preempção do Kernel
    - 12.3. Real-Time Kernel Patch
    - 12.4. Análise de desempenho do Real-Time Kernel
13. Ferramentas de desenvolvimento para IHM
    - 13.1. Compilação cruzada
    - 13.2. Desenvolvimento de interfaces gráficas
14. Sistemas GSM e GPS
    - 14.1. Definição de sistemas GSM
    - 14.2. Definição de sistemas GPS

14.3. Tipos de aplicação

15. Aplicações para Sistemas Embarcados

15.1. Comunicação Serial

15.2. IHM de dados com interface serial

15.3. Interação com redes GSM, comandos AT

15.4. Geração de informações de posicionamento

15.5. Interação com sistemas em nuvem (AWS)

15.6. Construção de Gateway MQTT com interface para sistemas em nuvem (AWS)

15.7. Reconhecimento facial em nuvem

6. METODOLOGIA

A presente componente curricular possui carga horária total de 60 horas / 72 ha (hora-aula, com duração de 50 min) com predominância de aulas presenciais e algumas aulas alocadas no modo assíncrono para completar a carga horária obrigatória. Para tal:

- Serão ministradas 2 ha semanais relativas à carga horária teórica (**totalizando 36 ha**) e mais 2 ha semanais relativas à carga horária prática (**totalizando 36 ha**);
- As aulas serão expositivas/interativas com uso de projetor, quadro, exemplificação por meio de software e experimentos e demais materiais, conforme os cronogramas apresentados no Quadro 1, referente às aulas teóricas, e Quadro 2, referente às aulas práticas;

Horário de Atendimento

Bloco 3N – Sala 3N213. Segunda-feira das 8h00 às10h00.

Atendimento pelo Telegram podendo ocorrer a qualquer dia/horário da semana, dependendo da disponibilidade. Todo aluno matriculado deverá cadastrar-se no grupo da disciplina, intitulado SEMB II - 2025/01, criado no software Telegram, por meio do link: https://t.me/+WBxWgMl-zok1NjYx

Quadro 1 – Cronograma de aulas teóricas.

Semana	Conteúdo
01	Linux com plataforma para o desenvolvimento de sistemas embarcados
02	Linux/RTOS: 1) Biblioteca de programação padrão Glibc. 2) Operações Atômicas, 3) Padronização Internacional de Programação de Sistemas Embarcados 4) Portabilidade de Código 5) Tipos Primitivos, 6) Pré-Processador, 7) Nomes de Variáveis. 8) Exemplos de Aplicação em protocolos industriais.

03	Linux/RTOS: 1) Especificação de paralelismo 2) Problema de seção crítica 3) Mecanismos básicos de exclusão mútua: soluções em software puro, desabilitação de interrupções, spin-lock. 4) Mutex: implementação de mutex 5) Semáforos: implementação de semáforos 6) Monitores: implementação de monitores 7) Mensagens 8) Deadlock
04	Linux/RTOS: Multiprocessamento e programação concorrente
05	Linux/RTOS: Multiprocessamento e programação concorrente
06	Programação em script com JavaScript
07	Desenvolvimento de interface gráfica para sistemas embarcados com Programação de baixo código para aplicações orientadas a eventos/IHM
08	Compilação cruzada e toolchains
09	Implementação de Servidores em Nuvem
10	Comunicação MQTT com a nuvem
11	Configuração de aplicações e atualização OTA
12	Linux como servidor de recursos: FTP / SSH / Web
13	Criando distribuições customizadas
14	Programação de módulos para o Kernel do Linux
15	Projeto Final
~	Projeto Final
~	Projeto Final
~	Projeto Final

Quadro 2 – Cronograma de aulas de laboratório.

Semana	**Conteúdo**
01	Linux com plataforma para o desenvolvimento de sistemas embarcados
02	Linux/RTOS: 1) Biblioteca de programação padrão Glibc. 2) Operações Atômicas, 3) Padronização Internacional de Programação de Sistemas Embarcados 4) Portabilidade de Código 5) Tipos Primitivos, 6) Pré-Processador, 7) Nomes de Variáveis. 8) Exemplos de Aplicação em protocolos industriais.
03	Linux: 1) Especificação de paralelismo 2) Problema de seção crítica 3) Mecanismos básicos de exclusão mútua: soluções em software puro, desabilitação de interrupções, spin-lock. 4) Mutex: implementação de mutex 5) Semáforos: implementação de semáforos 6) Monitores: implementação de monitores 7) Mensagens 8) Deadlock

04	RTOS: 1) Especificação de paralelismo 2) Problema de seção crítica 3) Mecanismos básicos de exclusão mútua: soluções em software puro, desabilitação de interrupções, spin-lock. 4) Mutex: implementação de mutex 5) Semáforos: implementação de semáforos 6) Monitores: implementação de monitores 7) Mensagens 8) Deadlock
05	Multiprocessamento e programação concorrente
06	Multiprocessamento e programação concorrente
07	Desenvolvimento de interface gráfica para sistemas embarcados com Programação de baixo código para aplicações orientadas a eventos
08	Linux como servidor de recursos: FTP / SSH / Web
09	Desenvolvimento de aplicações Web no backend
10	Comunicação MQTT com a nuvem
11	Criando distribuições customizadas
12	Criando distribuições customizadas
13	Programação de módulos para o Kernel do Linux
14	Configuração de aplicações e atualização OTA
15	Projeto Final
~	Projeto Final
~	Projeto Final
~	Projeto Final

7. AVALIAÇÃO

As avaliações serão constituídas de seminários e apresentação de projeto. Eles poderão ser individuais ou em grupo, dependendo do número de alunos matriculados:

· **Seminários**
· **Trabalhos Práticos**

Distribuição da Pontuação da disciplina:

- Seminários/Projeto Final: 40 pontos

- Trabalhos Práticos: 60 pontos

Avaliação de recuperação: Será oferecida uma avaliação de recuperação para os discentes que não obtiverem o rendimento mínimo para aprovação e com frequência mínima de 75% na disciplina, conforme Resolução CONGRAD nº 46/2022, Cap. II, Seção III.

A avaliação de recuperação será composta por uma prova extra (teórico/prática), no valor de 100 pontos e em substituição às atividades teóricas, abordando qualquer conjunto de conteúdos ministrados ao longo do semestre.

Os estudantes que realizarem a atividade de recuperação e forem aprovados (nota superior a 60 pontos) terão limitada a sua nota final em 60 pontos.

8. BIBLIOGRAFIA

Básica

1. ALMEIDA, R.; MORAES, C.; SERAPHIM, T. **Programação de sistemas embarcados: desenvolvendo software para microcontroladores em linguagem C.** Rio de Janeiro: Elsivier, 2016.
2 . MATTHEW, Neil.; STONES, Richard. **Beginning Linux programming.** [s.l.]: Wiley, 2007.
MOLLOY, Derek. **Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux.** New York: John Wiley & Sons, 2016.
3. SALVADOR, Otavio; ANGOLINI, Daiane. **Embedded Linux development with Yocto Project.** Birmingham: Packt Publishing, 2014.
4. YIU, J. **The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors.** 3a ed. [s.l.]: Newnes/Elsevier, 2014.

Complementar

1. BACKES, André. **Linguagem C: completa e descomplicada.** Rio de Janeiro: Elsevier, 2013.
2. BARR, Michael; MASSA, Anthony. **Programming embedded systems: with C and GNU development tools.** O'Reilly Media, 2006.
3. GRENNING, James W. **Test Driven Development for Embedded C.** [S. l.]: Pragma c Bookshelf, 2011.
4. KLEMENS, Ben. **21st Century C: C ps from the New School.** [S. l.]: O'Reilly Media, 2015.
5. HOOK, Brian. **Write portable code: An Introduytion to Developing Software for Multiple Platforms.** [S. l.]: No Starch Press, 2005.
6. HYDE, Randall. **Write great code: understanding the machine.** v. 1. [S. l.]: No Starch Press, 2012.
7. MONTGOMERY, Stephen L. MISRA. **C: Guidelines for the Use of the C Language in Critical Systems** 2012. [S. l.]: Misra, 2013.
8. PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional.** 8. ed. Porto Alegre: McGraw-Hill, 2016.
9. SINK, E. **Version Control by Example.** [S. l.]: Pyrenean Gold Press, 2011.
10. TANENBAUM, Andrew S. **Organização estruturada de computadores.** São Paulo: Pearson, 2013.
11. WHITE, E. **Making Embedded Systems: Design Patterns for Great Software.** [S. l.]: O'Reilly Media, 2014.

9. APROVAÇÃO

Aprovado em reunião do Colegiado realizada em: ____/____/_____

Coordenação do Curso de Graduação: _____

Documento assinado eletronicamente por **Fabio Vincenzi Romualdo da Silva**, **Professor(a) do Magistério Superior**, em 28/06/2025, às 09:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **6463324** e o código CRC **4BAB56E3**.