SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Modelo de plano de Ensino em conformidade com as RESOLUÇÕES Nº 30/2011 (Normas Gerais da Graduação) e Nº 07/2020 (Atividades Acadêmicas Remotas Emergenciais), do CONSELHO DE GRADUAÇÃO

FACULDADE DE ENGENHARIA ELÉTRICA
COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

PLANO DE ENSINO

1. IDENTIFICAÇÃO

COMPONENTE CURRICULAR: Sistemas Embarcados II				
UNIDADE OFERTANTE: Faculdade de Engenharia Elétrica				
CÓDIGO: FEELT39015		**PERÍODO/SÉRIE:** 6º		**TURMA:** U
CARGA HORÁRIA			**NAT UREZA**	
TEÓRICA: 2	**PRÁTICA:** 2	**TOTAL:** **4**	**OBRIGATÓRIA: (X)**	**OPTATIVA: ( )**
PROFESSOR(A): Éder Alves de Moura				**ANO/SEMESTRE:** 2020/4
OBSERVAÇÕES: Serão ofertadas 15 vagas para o curso Engenharia de Controle e Automação e 20 vagas para o curso de Engenharia Mecatrônica.				

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 u lizar 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 disciplina é composta de componentes teóricas e práticas. Entretanto, será enfatizado os aspectos práticos da mesma, com fim na implementação de aplicações e configuração do ambiente Linux para o desenvolvimento de aplicações embarcadas. Ao final, é esperado que o discente tenha capacidade de configurar e desenvolver aplicações para sistemas embarcados que tenham o Linux como Sistema Operacional de base. Também, destaca-se a necessidade de programar aplicações concorrentes e com capacidade de comunicação via troca de pacotes TCP e UDP, bem como a configuração do Linux para a disponibilidade de serviços, como: SSH, FTP e servidores Web.

Para a presente componente curricular, a ser ministrada em formato remoto, no âmbito do período de Atividades Acadêmicas Remotas Emergenciais (AARE), serão adotadas aulas em duas modalidades distintas de comunicação: síncrona (todos os alunos conectados simultaneamente em uma sala de aula virtual, sob a regência do professor) e assíncrona (contemplando atividades remotas *off-line*).

- Atividade síncrona: Aulas expositivas e de interação entre o professor e os alunos para sanar dúvidas por meio das plataformas *Google Meet* ou *Microsoft Teams*.

- Atividade assíncrona: A disciplina será organizada no Microsoft Teams, onde serão disponibilizados os materiais, tais como os slides das aulas, listas de exercícios, apostilas, vídeos, códigos de programas, ou links para acessá-los, quando disponível em fontes externas. As atividades assíncronas serão divididas em duas partes: atividades semanais e um projeto final. As atividades semanais terão o prazo de uma semana para serem entregues, indo do dia da aula até o dia anterior da aula da próxima semana, com exceção da primeira e última semanas. O projeto final deverá ser desenvolvido pelo aluno/grupo, com a apresentação de um relatório final em formato de artigo.

Cronograma

As atividades síncronas acontecerão nas terças-feiras das 09h50 às 11h30, no link disponibilizado na sala virtual da disciplina, ficando o professor disponível on-line para apresentação expositiva e o saneamento de dúvidas. A Tabela 1 apresenta o cronograma com as atividades propostas.

Tabela 1 – Atividades síncronas com 18 ha (2 ha/semana)

Semana	Data	Conteúdo	Tópicos do Programa
01	22/10/2020	Linux: Configuração e Uso	1; 2; 8.
02	27/10/2020	Linguagens de Programação: C++ e Python	9; 10; 11.
03	03/11/2020	Kernel, funcionalidades de configuração	3; 6; 7; 12.
04	10/11/2020	Customização de distribuições Linux	4; 5.
05	17/11/2020	Ferramentas de programação aplicações concorrentes e comunicação entre processos	10; 11.
06	24/11/2020	Desenvolvimento de aplicações com interfaces gráficas	2; 6; 13.
07	01/12/2020	Linux como servidor de recursos	14; 15.
08	08/12/2020	Gerenciamento de Rede e segurança	1.
09	15/12/2020	Apresentação do trabalho final	~

A Tabela 2 apresenta o cronograma das atividades assíncronas programas.

Tabela 2 – Atividades assíncronas com 54 ha (6 ha/semana)

Semana	Data	Data	Conteúdo
01	22/10/2020	26/10/2020	Linux: Configuração e Uso
02	27/10/2020	03/11/2020	Linguagens de Programação: C++ e Python
03	03/11/2020	09/11/2020	Kernel, funcionalidades de configuração
04	10/11/2020	16/11/2020	Customização de distribuições Linux
05	17/11/2020	23/11/2020	Ferramentas de programação aplicações concorrentes e comunicação entre processos
06	24/11/2020	30/11/2020	Desenvolvimento de aplicações com interfaces gráficas
07	01/12/2020	07/12/2020	Linux como servidor de recursos
08	08/12/2020	14/12/2020	Gerenciamento de Rede e segurança
09	15/12/2020	19/12/2020	Apresentação do trabalho final

Em resumo, a ***carga horária*** será dividida em:

- Síncrona: 2 ha/semana (total: 18 ha)
- Assíncrona: 6 ha/semana (total: 54 ha)
- Somando: 18 ha/semana (total 72 ha)

Para o desenvolvimento das atividades propostas, será necessária a instalação, na própria máquina, de um ambiente virtualizado para o desenvolvimento de software. Serão adotadas ferramentas *open source* ou sem custos.

---

7. AVALIAÇÃO

A avaliação consistirá de três grupos de atividades:

- Atividade semanal – Lista de perguntas: Após cada aula, o aluno terá o prazo de uma semana para responder um questionário online disponível na plataforma Moodle da disciplina, com questões relativas aos temas abordados em sala e estudo dos recursos didáticos indicados. Esta atividade valerá 3,0 pontos cada.

- Atividade semanal – Implementação: Após cada aula, o aluno deverá executar em ambiente virtualizado ou de programação devidamente indicado, um conjunto de atividades práticas, com questões relativas aos temas abordados em sala e estudo dos recursos didáticos indicados. Esta atividade valerá 5,0 pontos cada.

- Projeto final: Consistirá do desenvolvimento de uma atividade em grupo, ao longo do semestre, de uma aplicação de controle ou de Internet das Coisas, de sistemas reais ou virtuais, por meio da configuração e programação de um sistema Linux embarcado (real ou virtual). A entrega do trabalho consistirá de vídeo demonstrativo, produzido pelos alunos e relatório descritivo. Esta atividade valerá 28,0 pontos.

Tabela 3 – Distribuição de pontos.

Avaliação	**Pontuação Individual**	**Número**	**Pontuação**
Atividade semanal: Perguntas	3,0	9	27,0
Atividade semanal: Implementação	5,0	9	45,0
Projeto final	28,0	1	28,0
Total			***100,0***

8. BIBLIOGRAFIA

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.
3. MOLLOY, Derek. Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux. New York: John Wiley & Sons, 2016.
4. SALVADOR, Otavio; ANGOLINI, Daiane. Embedded Linux development with Yocto. Project. Birmingham: Packt Publishing, 2014.
5. YIU, J. The Defini ve Guide to ARM Cortex-M3 and Cortex-M4 Processors. 3a ed. [s.l.]: Newnes/Elsevier, 2014.

BIBLIOGRAFIA COMPLEMENTAR:

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

16. 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 em: _____