Caracterização da Unidade Curricular / Characterization of the Curricular Unit

Designação da Unidade Curricular (UC) / Title of Curricular Unit (CU): Sistemas de Microprocessadores / Microprocessor Systems

Área científica da UC / CU Scientific Area: Eletrónica e Automação / Electronics and Automation

Semestre / Semester: 3º

Número de créditos ECTS / Number of ECTS credits: 6

Carga horária por tipologia de horas / Workload by type of hours: TP: 45; OT: 6; O: 9

Carga letiva semanal / Weekly letive charge: 3h

Objetivos de aprendizagem (conhecimentos, aptidões e competências a desenvolver pelos estudantes):

Conceção de sistemas embebidos de pouca complexidade baseados em microcontroladores. Na vertente hardware são estudados uma arquitetura de processador real e os periféricos que o circulam. Na vertente software é estudada a interação com o hardware e o desenvolvimento de aplicações reais.

Os estudantes que terminam com sucesso esta unidade curricular serão capazes de:

- Analisar estruturas de hardware baseadas em microcontroladores.
- Desenvolver software de sistema interface com o hardware.
- Desenvolver software de aplicação com a máquina.
- Testar e depurar o software realizado, usando as ferramentas adequadas, sobre hardware real.
- Instalar programas em memória ROM e FLASH.
- Definir e interpretar especificações de sistemas embebidos.
- Integrar periféricos e desenvolver os respetivos gestores.
- Integrar pacotes de software genéricos.
- Configurar o sistema operativo de acordo com o hardware as necessidades da aplicação.
- Configurar um sistema autónomo, que inicie a sua operação a partir da ligação da energia.

Intended learning outcomes (knowledge, skills and competences to be developed by the students):

Design of low-complexity embedded systems based on microcontrollers. On the hardware side, a real processor architecture and the peripherals that surround it are studied. In the software aspect, interaction with hardware and the development of real applications are studied.

Students who successfully complete this curricular unit will be able to:

- Analyze hardware structures based on microcontrollers.
- Develop system software to interface with hardware.
- Develop application software with the machine.
- Test and debug software performed, using appropriate tools, on real hardware.
- Install programs in ROM and FLASH memory.

- Define and interpret specifications for embedded systems.
- Integrate peripherals and develop their respective managers.
- Integrate generic software packages.
- Configure the operating system according to the hardware and application needs.
- Configure an autonomous system, which starts its operation when the power is connected.

Conteúdos programáticos:

1. A Sustentabilidade como tema institucional

1.1. Os conceitos da Sustentabilidade

1.2. A definição e enquadramento da aplicação dos ODS

1.3. A concretização da sustentabilidade pela via das arquiteturas e tecnologias usadas em sistemas embebidos e respetivas aplicações

2. Introdução aos Sistemas Embebidos

2.1. Definição e Características

2.2. História e Evolução dos Sistemas Embebidos

2.3. Aplicações e Exemplos

3. Linguagens de Programação para Sistemas Embebidos

3.1. Assembly

3.2. C/C++

3.3. Python

3.4. Outras Linguagens (e.g., Rust, Ada)

4. Single Board Computer (SBC)

4.1. Introdução aos SBC

4.2. Exemplos de SBC (e.g., Raspberry Pi, Arduino)

4.3. Comparação entre Diferentes SBCSistemas Operativos para Sistemas Embebidos

5. Sistemas Operativos em Tempo Real (RTOS)

5.1. Linux Embarcado

5.2. Outros Sistemas Operativos (e.g., FreeRTOS, Zephyr)

6. Firmware

6.1. Definição e Funções do Firmware

6.2. Desenvolvimento de Firmware

6.3. Atualização e Manutenção de Firmware

7. Comunicação para Sistemas Embebidos

7.1. I2C (Inter-Integrated Circuit)

7.2. SPI (Serial Peripheral Interface)

7.3. 1-Wire

7.4. Bluetooth

7.5. Outras Tecnologias de Comunicação (e.g., UART)

8. Interfaces Gráficas

8.1. Introdução às Interfaces Gráficas

8.2. Tecnologias de interfaces gráficas (LCD, OLED, TFT)

8.3. Écran Táctil

8.4. Ferramentas e Bibliotecas (e.g., Qt, GTK)

8.5. Desenvolvimento de Interfaces Gráficas para Sistemas Embebidos

9. Sensores e Atuadores Inteligentes

9.1. Tipos de Sensores

9.2. Tipos de Atuadores

9.3. Integração de Sensores e Atuadores em Sistemas Embebidos

10. Segurança em Sistemas Embebidos

10.1. Ameaças e Vulnerabilidades

10.2. Técnicas de Segurança

10.3. Boas Práticas de Desenvolvimento Seguro

11. Desenho de Hardware para Sistemas Embebidos

11.1. Componentes Básicos de Hardware

11.2. Design de Placas de Circuito Impresso (PCB)

11.3. Ferramentas de Design (e.g., KiCad, Eagle)

12. Fontes de Alimentação e Níveis de Alimentação

12.1. Tipos de Fontes de Alimentação

12.2. Conversores de Tensão

12.3. Proteção contra Sobrecarga e Curto-Circuito

13. Alimentação com Baterias e Painéis Fotovoltaicos (FV)

13.1. Tipos de Baterias

13.2. Gestão de Energia

13.3. Introdução aos Painéis Fotovoltaicos

13.4. Integração de Painéis FV em Sistemas Embebidos

14. Desenvolvimento de trabalhos com arquitetura AVR (utilizando a plataforma Arduíno com interfaces de I/O digital e analógica), explorando o projeto de software em assembly e C, com o apoio de ferramentas de simulação e de emulação. Implementação de pequenos projetos práticos que envolverão componentes de input e output em cenários de informática industrial, automação e telecomunicações.

15. Desenvolvimento de miniprojecto integrador.

Syllabus:

1. Sustainability as an institutional theme

1.1. The concepts of Sustainability

1.2. The definition and framework of SDG applications

1.3. The achievement of sustainability through architectures and technologies used in embedded systems and respective applications

2. Introduction to Embedded Systems

2.1. Definition and Characteristics

2.2. History and Evolution of Embedded Systems

2.3. Applications and Examples

3. Programming Languages for Embedded Systems

3.1. Assembly

3.2. C/C++

3.3. Python

3.4. Other Languages (e.g., Rust, Ada)

4. Single Board Computer (SBC)

4.1. Introduction to SBCs

4.2. Examples of SBC (e.g., Raspberry Pi, Arduino)

4.3. Comparison between Different SBCsOperating Systems for Embedded Systems

5. Real-Time Operating Systems (RTOS)

5.1. Embedded Linux

5.2. Other Operating Systems (e.g., FreeRTOS, Zephyr)

6. Firmware

6.1. Definition and Functions of Firmware

6.2. Firmware Development

6.3. Firmware Update and Maintenance

7. Communication for Systems Embedded Systems

7.1. I2C (Inter-Integrated Circuit)

7.2. SPI (Serial Peripheral Interface)

7.3. 1-Wire

7.4. Bluetooth

7.5. Other Communication Technologies (e.g., UART)

8. Graphical Interfaces

8.1. Introduction to Graphical Interfaces

8.2. Graphical interface technologies (LCD, OLED, TFT)

8.3. Touchscreen

8.4. Tools and Libraries (e.g., Qt, GTK)

8.5. Developing Graphical Interfaces for Embedded Systems

9. Intelligent Sensors and Actuators

9.1. Types of Sensors

9.2. Types of Actuators

9.3. Integration of Sensors and Actuators in Embedded Systems

10. Security in Embedded Systems

10.1. Threats and Vulnerabilities

10.2. Security Techniques

10.3. Good Secure Development Practices

11. Hardware Design for Embedded Systems

11.1. Basic Hardware Components

11.2. Printed Circuit Board (PCB) Design

11.3. Design Tools (e.g., KiCad, Eagle)

12. Power Supplies and Power Levels

12.1. Types of Power Supplies

12.2. Voltage Converters

12.3. Overload and Short-Circuit Protection

13. Power Supply with Batteries and Photovoltaic (PV) Panels

13.1. Types of Batteries

13.2. Energy Management

13.3. Introduction to Photovoltaic Panels

13.4. Integration of PV Panels in Embedded Systems

14. Development of work with AVR architecture (using the Arduino platform with digital and analog I/O interfaces), exploring software design in assembly and C, with the support of simulation and emulation tools. Implementation of small practical projects that will involve input and output components in industrial computing, automation and telecommunications scenarios.

15. Development of an integrative mini-project.