ANNA UNIVERSITY

NON-AUTONOMOUS COLLEGE

AFFILIATED TO ANNA UNIVERSITY

M.E., EMBEDDED SYSTEM TECHNOLOGIES

REGULATIONS 2025

PROGRAMME OUTCOMES (POs)

PO1	An ability to independently carry out research / investigation and development work to solve practical problems.
PO2	An ability to write and present a substantial technical report / document.
PO3	Students should be able to demonstrate a degree of mastery in embedded system technologies.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

PSO1	Graduates will be proficient in integrating hardware, software, and networking technologies to deliver optimized embedded solutions.
PSO2	Graduates will be able to apply advanced embedded system technologies to drive research, foster innovation, and support product development.

ANNA UNIVERSITY, CHENNAI

POST GRADUATE CURRICULUM (NON.AUTONOMOUS AFFILIATED INSTITUTIONS)

Programme: M.E., Embedded System Technologies **Regulations:** 2025

Abbreviations:

BS – Basic Science (Mathematics, Physics, Chemistry)
ES – Engineering Science (General (**G**), Programme Core (**PC**), Programme Elective (**PE**))
SD – Skill Development
SL – Self Learning
OE – Open Elective

L – Laboratory Course
T – Theory
LIT – Laboratory Integrated Theory
PW – Project Work
TCP – Total Contact Period(s)

Semester I

S. No.	Course code	Course title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.	ET25101	Design of Embedded Systems	T	3	0	0	3	3	ES (PC)
2.	ET25102	Programming Embedded Systems	LIT	3	0	2	5	4	ES (PC)
3.	ET25103	Microcontroller Based System Design	T	3	0	0	3	3	ES (PC)
4.	ET25C01	IoT for Smart Systems	T	3	0	0	3	3	ES (PC)
5.	ET25104	VLSI Design and Reconfigurable Architecture	T	3	1	0	4	4	ES (PC)
6.	ET25105	Technical Seminar	-	0	0	2	2	1	SD
Total Credits							**20**	**18**	

Semester II

S. No.	Course Code	Course title	Type	Periods Per Week			TCP	Credits	Category
				L	T	P			
1.	ET25201	Real Time Operating System	T	3	1	0	4	4	ES (PC)
2.	ET25202	Embedded System Networking	T	3	0	0	3	3	ES (PC)
3.	ET25203	Embedded Control System	T	3	1	0	4	4	ES (PC)
4.	---	Programme Elective - I	T	3	0	0	3	3	ES (PC)
5.	---	Industry Oriented Course I	--	1	0	0	1	1	SD
6.	ET25204	Embedded System Laboratory – I	L	0	0	4	4	2	ES (PC)
7.	ET25205	Industrial Training	--	--	--	--	--	2	SD
8.	---	Self-Learning Course	--	--	--	--	--	1	SD
Total Credits							**19**	**20**	

Semester III

S. No.	Course code	Course Title	Type	Periods Per Week			TCP	Credits	Category
				L	T	P			
1.	---	Programme Elective II	T	3	0	0	3	3	ES (PE)
2.	---	Programme Elective III	T	3	0	0	3	3	ES (PE)
3.	---	Programme Elective IV	T	3	0	0	3	3	ES (PE)
4.	---	Open Elective	--	3	0	0	3	3	-
5.	---	Industry Oriented Course II	--	1	0	0	1	1	SD
6.	ET25301	Project Work I	--	0	0	12	12	6	SD
Total Credits							**25**	**19**	

Semester IV

S. NO.	Course Code	Course Title	Type	Periods Per Week			TCP	Credits	Category
				L	T	P			
1.	ET25401	Project Work II	--	0	0	24	24	12	SD
Total Credits							**24**	**12**	

Programme Elective Courses (PE)

S. No.	Course code	Course title	Periods Per week			Total Contact Periods	Credits
			L	T	P		
1.	ET25001	Wireless And Mobile Communication	3	0	0	3	3
2.	ET25002	Data Driven Embedded Systems	3	0	0	3	3
3.	ET25003	Advanced Embedded Processor	3	0	0	3	3
4.	ET25004	DSP Based System Design	3	0	0	3	3
5.	ET25005	Automotive Embedded System	3	0	0	3	3
6.	ET25006	Embedded Linux	3	0	0	3	3
7.	ET25007	Autonomous Vehicle	3	0	0	3	3
8.	ET25008	Computer Vision	3	0	0	3	3
9.	ET25009	Digital Twin	3	0	0	3	3
10.	ET25C02	Machine Learning and Deep Learning	3	0	0	3	3
11.	ET25010	Wireless Sensor Networks	3	0	0	3	3
12.	ET25011	Embedded Computing	3	0	0	3	3
13.	ET25012	Embedded Systems Security	3	0	0	3	3
14.	ET25013	Robotics and Automation	3	0	0	3	3
15.	ET25014	Reconfigurable Processor and SoC Design	3	0	0	3	3
16.	ET25015	MEMS and NEMS Technology	3	0	0	3	3
17.	ET25016	Embedded System for Bio Medical Applications	3	0	0	3	3
18.	PX25C02	Electric vehicles and power management.	3	0	0	3	3
19.	ET25017	Edge Data Analytics	3	0	0	3	3
20.	ET25C03	Python Programming for Machine Learning	3	0	0	3	3
21.	ET25018	Embedded Device Driver Programming	3	0	0	3	3

Semester I

ET25101	Design of Embedded Systems	L	T	P	C
		3	0	0	3

Course objectives:

- Understand embedded architectures, components, and communication protocols.
- Analyze real-time requirements and hardware/software co-design approaches.
- Design and develop embedded applications using modeling tools and EDLC frameworks.

Introduction to Embedded Systems: Features of embedded systems, processor selection, memory organization, timers, watchdogs, and RTC. Development tools (IDE, assembler, linker, debugger, emulator) and basics of functional safety standards.
Activities: Install, configure, and create a simple project with a basic IDE to learn project structure.

Communication Protocols: I/O ports, buses, interrupts, and service mechanisms. Serial protocols (RS232, RS485-MODBUS, USB, I2C, CAN) and wireless protocols (Wi-Fi, Bluetooth, ZigBee). Introduction to device drivers.
Activities: Write a pseudo-code or library examples of initializing and using a peripheral (LED, LCD, sensor)

Real Time Systems: Structure and characteristics of real-time systems, run-time estimation, task scheduling, and performance measures. Fault tolerance, reliability, evaluation methods, and clock synchronization.
Activities: Design a simple traffic light controller or temperature monitoring system with periodic tasks.

Hardware/Software Design Approaches: Embedded software modeling using UML diagrams. Hardware/software partitioning, co-design, co-synthesis, and comparison of single vs. multi-processor architectures. Parallelism and modeling tools like Papyrus/Cameo.
Activities: For a typical simple embedded system examples, students can plan which functions should be implemented in hardware vs. software.

Embedded System Application Development: Phases of Embedded Development Life Cycle (EDLC). Target architecture selection for control- and data-dominated systems. Case studies: digital camera, adaptive cruise control, and mobile phone software.
Activities: Group activity - Discuss examples of embedded systems in daily life.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (5%), Assignments (10%), Review of Question Papers (IES, GATE, SSC Questions) (20%), Projects (20%), Flipped Class (5%), Internal Examinations (40%).

References:

1. Rajkamal. (2011). *Embedded system: Architecture, programming, design*. Tata McGraw Hill.
2. Krishna, C. M., & Shin, K. G. (Year unknown). *Real-time systems*. McGraw-Hill International Editions.
3. Das, L. B. (2013). *Embedded systems: An integrated approach*. Pearson.
4. Douglass, B. P. (2011). *Real-time UML workshop for embedded systems*. Elsevier.
5. Staunstrup, J., & Wolf, W. (Year unknown). *Hardware/software co-design principles and practice*. Springer.
6. Shibu, K. V. (Year unknown). *Introduction to embedded systems*. Tata McGraw Hill.

	Description of CO	PO	PSO
CO1	Describe embedded architectures and tools.	PO1(2) PO2(2) PO3(1)	PSO1(3) PSO2(1)
CO2	Design embedded solutions with I/O and protocols.	PO1(2) PO2(1) PO3(1)	PSO1(3) PSO2(2)
CO3	Evaluate real-time requirements, scheduling, and reliability.	PO1(3) PO2(1) PO3(2)	PSO1(2) PSO2(3)
CO4	Model embedded designs with UML and co-design methods.	PO1(2) PO2(2) PO3(1)	PSO1(3) PSO2(2)
CO5	Develop embedded applications using EDLC and case studies.	PO1(3) PO2(2) PO3(2)	PSO1(3) PSO2(3)

ET25102	Programming Embedded Systems	L	T	P	C
		3	0	2	4

Course objectives:

- Learn fundamentals of C, Embedded C, and Python programming for embedded applications.
- Gain proficiency with GNU C toolchain in Linux for coding, debugging, and optimization.
- Develop skills in modular programming, libraries, and practical application design.

Basic C Programming: Overview of C program development, structured programming, data types, operators, and program control. Functions, arrays, and fundamentals for building embedded applications.

Laboratory Session:

- Basic programming with 8-bit MCUs, C and Assembly coding on 8051/other 8-bit MCUs with peripherals, Functions, Arrays and menu driven code

Embedded C: Structured coding practices, modular development using headers and ports, and object-oriented features in C. Real-time constraints handling through delays, loop/hardware timeouts, and timing mechanisms.

Laboratory Session:

- I/O Programming with 8-bit Microcontrollers – interfacing with serial ports, LCD, sensors, PWM, and motor control.

C Programming Tool-Chain in Linux: Compilation stages, preprocessing, and GCC usage. Debugging with GDB, build automation using Make, profiling with gprof, GNU binary utilities, and libraries for efficient development.

Laboratory Session:

- Write a simple C program that prints " Hello World" and use GCC flags to understand preprocessing, compilation, assembly and linking
- Write a program with nested loops or recursive function and use gprof to analyze time spent in each function

Python Programming: Introduction, Parts of Python Programming Language, Control Flow Statements, Functions, Strings, Lists, Dictionaries, Tuples and Sets.

Laboratory Session:

- Write a python code to perform the following:
  1. a loop to simulate an LED blinking 10 times
  2. if and else code to check the status of the temperature
  3. Function code to simulate PWM by printing "ON" and "OFF" for a given duty cycle

Modules, Packages and Libraries In Python: Creating and using modules/packages. Practical applications using Python libraries for math, plotting, GUI, imaging, and networking to support embedded solutions.

Laboratory Session:

- Python code to create a custom module with to perform a specific task like ON/OFF

- Program to plot simulated sensor data over time
- Program to send a data over a network socket

Weightage: Continuous Assessment: 50%, End Semester Examinations: 50%

Assessment Methodology: Quiz (5%), Assignments (10%), Review of Question Papers (IES, GATE, SSC Questions) (20%), Projects (20%), Flipped Class (5%), Internal Examinations (40%).

References:

1. Deitel, P., & Deitel, H. (2016). *C how to program* (8th ed.). Pearson Education Limited.
2. Pont, M. J. (2002). *Embedded C*. Addison-Wesley (an imprint of Pearson Education).
3. Von Hagen, W. (2006). *The definitive guide to GCC* (2nd ed.). Apress Inc.
4. Gowrishankar, S., & Veena, A. (201?). *Introduction to Python programming*. CRC Press, Taylor & Francis Group.
5. Mueller, J. P. (2018). *Beginning programming with Python for dummies* (2nd ed.). John Wiley & Sons Inc.

	Description of CO	PO	PSO
CO1	Develop a C/Embedded C coding for microcontrollers.	PO1(2) PO2(1) PO3(2)	PSO1(3) PSO2(2)
CO2	Build a interface and program microcontroller peripherals to validate functionality.	PO1(2) PO2(1) PO3(2)	PSO1(3) PSO2(2)
CO3	Use GNU C toolchain in Linux to develop and optimize embedded software.	PO1(2) PO2(2) PO3(2)	PSO1(3) PSO2(3)
CO4	Implement microcontroller solutions in C/Embedded C and Python.	PO1(3) PO2(2) PO3(2)	PSO1(3) PSO2(3)
CO5	Design microcontroller applications with C/Embedded C.	PO1(3) PO2(2) PO3(2)	PSO1(3) PSO2(3)

ET25103	Microcontroller Based System Design	L	T	P	C
		3	0	0	3

Course objectives:

- Understand the architecture and programming of PIC and ARM/RISC processors.
- Explore peripherals, memory management, and DSP implementation in microcontrollers.
- Develop embedded system applications using PIC and ARM platforms.

PIC Microcontroller: Architecture, memory organization, addressing modes, instruction set, PIC programming in Assembly & C I/O port, Data Conversion, RAM & ROM Allocation, Timer programming, Programming practice in MP-LAB.

Activities: Install, configure, and create a simple project with a MPLAB IDE to learn project structure.

Arm Architecture: Architecture, memory organization, addressing modes, The ARM Programmer's model Registers, Pipeline, Interrupts, Coprocessors, Interrupt Structure.

Activities: Perform a coding exercise to configure a timer interrupt on ARM and learn how the vector table directs execution of ISR.

Peripherals of PIC and Arm Microcontroller:

PIC: ADC, DAC and Sensor Interfacing Flash and EEPROM memories.

ARM: I/O Memory, EEPROM, I/O Ports, SRAM, Timer, UART Serial Communication with PC, ADC/DAC Interfacing.

Activities: Perform a coding exercise to read analog input from a potentiometer or sensor via ADC and control LED brightness using DAC/PWM output

ARM Microcontroller Programming: ARM general Instruction set, Thumb instruction set, Introduction to DSP on ARM, Implementation example of Filters.

Activities : Perform a coding exercise with and without Thumb mode and to implement moving average filter in C language.

Design with PIC and Arm Microcontrollers: PIC applications include gate signal generation for converters/inverters, motor control, appliance control, frequency measurement, and standalone data acquisition.

ARM examples cover basic ASM/C programs such as loops, lookup tables, block copy, subroutines, and error detection with Hamming code.

Activities: Students collaboratively design and prototype small subsystems using PIC and ARM microcontrollers, applying to specific application areas.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (5%), Assignments (10%), Review of Question Papers (IES, GATE, SSC Questions) (20%), Projects (20%), Flipped Class (5%), Internal Examinations (40%).

References:

1. Furber, S. (2010). ARM system on chip architecture. Addison Wesley.
2. Sloss, A. N., Symes, D., Wright, C., & Rayfield, J. (2007). ARM system developer's guide: Designing and optimizing system software. Elsevier.
3. Mazidi, M. A., McKinley, R. D., & Causey, D. (2008). PIC microcontroller and embedded systems using assembly and C for PIC18. Pearson Education.
4. Iovine, J. (2000). PIC microcontroller project book. McGraw Hill.
5. Hohl, W. (Year unknown). ARM assembly language fundamentals and techniques. CRC Press.
6. Kamal, R. (2012). Microcontrollers: Architecture, programming, interfacing & | system design. Pearson.

	Description of CO	PO	PSO
CO1	Explain PIC/ARM architecture and instruction sets	PO1(2) PO2(2) PO3(2)	PSO1(3) PSO2(2)
CO2	Develop a program PIC/ARM in assembly and C.	PO1(2) PO2(1) PO3(3)	PSO1(3) PSO2(2)
CO3	Compare 8-, 16-, and 32-bit RISC.	PO1(2) PO2(2) PO3(2)	PSO1(2) PSO2(3)
CO4	Implement DSP applications on ARM processors.	PO1(3) PO2(1) PO3(3)	PSO1(3) PSO2(3)
CO5	Design embedded applications using PIC/ARM.	PO1(3) PO2(2) PO3(3)	PSO1(3) PSO2(3)

ET25C01	IoT For Smart Systems	L	T	P	C
		3	0	0	3

Course objectives:

- Understand IoT technologies, architectures, and communication methods.
- Learn about embedded processors, sensors, and IoT platforms.
- Explore IoT applications, data analytics, and security aspects.

Introduction to Internet of Things: Definition, elements, and characteristics of IoT. Architectural stack, enabling technologies, challenges, and hardware platforms like Arduino, Raspberry Pi, and ESP boards.
Activities: Students explore the definition, elements, architecture, and hardware platforms of IoT by designing a small-scale IoT prototype using embedded boards.

IoT Architecture: Reference models and architectures, node structure (sensing, processing, communication, power). Networking topologies, IoT standards, cloud and fog computing, and Bluetooth-based solutions.
Activities: Case study to construct a IoT reference architectures and node structure

Protocols And Wireless Technologies for IoT : Overview of protocols: NFC, SCADA, RFID, Zigbee, MIPI family, GSM/CDMA, LTE, and 5G small cells. Wireless standards for IoT including WiFi, Bluetooth, ZigBee, UWB, LoRa, 6LoWPAN, Thread vs. Matter, and proprietary systems.
Activities: Group activity to perform the selection of protocols for a typical use case.

IoT Subsystems: IoT services and attributes: analytics, dependability, interoperability, security, and maintainability. Platforms for data analytics, visualization, virtualization, and IoT application development with emphasis on privacy and security.
Activities: Compare the performance of various data analytics & visualization tools.

Case Studies: Applications of IoT in industrial systems, smart homes, cities, grids, vehicles, EV charging, agriculture, environment, productivity, and defense.
Activities: Each student group can prepare a short report/presentation on different use cases.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (5%), Assignments (10%), Review of Question Papers (IES, GATE, SSC Questions) (20%), Projects (20%), Flipped Class (5%), Internal Examinations (40%).

References:

1. Bahga, A., & Madisetti, V. (2015). *Internet of things: A hands-on approach*. Universities Press.
2. Greengard, S. (2015). *The internet of things*. The MIT Press.
3. Raj, P., & Raman, A. C. (2017). *The internet of things: Enabling technologies, platforms, and use cases*. CRC Press.
4. Cirani, S., Ferrari, G., Picone, M., & Veltri, L. (2018). *Internet of things: Architectures, protocols and standards*. Wiley.
5. Madisetti, V., & Bahga, A. (2014). *Internet of things: A hands-on approach*.

	Description of CO	PO	PSO
CO1	Explain IoT fundamentals, ecosystem, and technologies.	PO1(1) PO2(2) PO3(1)	PSO1(3) PSO2(1)
CO2	Analyze IoT architectures, nodes, and standards.	PO1(2) PO2(1) PO3(1)	PSO1(3) PSO2(1)
CO3	Evaluate IoT protocols, wireless tech, and trends.	PO1(3) PO2(1) PO3(2)	PSO1(2) PSO2(3)
CO4	Integrate IoT subsystems in application development.	PO1(2) PO2(1) PO3(2)	PSO1(3) PSO2(2)
CO5	Examine IoT case studies for innovative solutions.	PO1(3) PO2(3) PO3(2)	PSO1(1) PSO2(3)

ET25104	VLSI Design and Reconfigurable Architecture	L	T	P	C
		3	1	0	4

Course Objectives

- Learn fundamentals of sequential circuits, CMOS concepts, and IC fabrication.
- Understand reconfigurable processors, SoC architectures, and analog VLSI design.
- Gain practical skills in HDL programming and digital system modeling.

Introduction to Advanced Digital System Design: Modeling and design of synchronous and asynchronous sequential circuits. Includes vending machine controller design, hazards (static, dynamic, essential), and methods for hazard-free circuit implementation.
Activities: Perform a simulation to design combination circuits using ModelSim / Xilinx Vivado / Quartus and verify with testbench.

CMOS Basics & IC Fabrication: MOSFET scaling, transistor models, CMOS logic design, BiCMOS, low-power techniques, and fabrication methods. Emphasizes stick diagrams, layout design rules, and IC implementation basics.
Activities: Study MOSFET logic design, scaling effects, and apply stick diagrams with layout rules to connect theory and fabrication.

ASIC and Reconfigurable Processor and Soc Design: ASIC design flow, programmable ASICs, reconfigurable processor architecture, SoC overview, embedded FPGA, and applications such as DC motor control.
Activities: Compare traffic light controller implementation across ASIC simulation, FPGA/reconfigurable processor, and SoC co-design.

Analog VLSI Design: CMOS op-amp design (two/three-stage), high-speed/frequency op-amps, Super MOS, analog primitive cells, and introduction to FPAA concepts.
Activities : Design a two-stage CMOS op-amp using a circuit simulator and measure its key parameters.

HDL programming : VHDL-based digital design: structural, dataflow, behavioral modeling. Logic synthesis and simulation for adders, multipliers, ALUs, shift registers, and test benches.
Activities: Design a 4-bit ALU in VHDL with add, subtract, increment, AND, OR operations, and verify using an automated test bench.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (5%), Assignments (10%), Review of Question Papers (IES, GATE, SSC Questions) (20%), Projects (20%), Flipped Class (5%), Internal Examinations (40%).

References:

1. Givone, D. G. (2002). Digital principles and design. Tata McGraw Hill.
2. Nurmi, J. (Ed.). (2007). Processor design system-on-chip computing for ASICs and FPGAs. Springer.
3. Gaillardon, P.-E. (2015). Reconfigurable logic: Architecture, tools, and applications (1st ed.). CRC Press.
4. Ismail, M., & Fiez, T. (Year unknown). Analog VLSI signal and information processing. McGraw Hill International Editions.
5. Dally, W. J., Harting, C., & Aamodt, T. M. (2015). Digital design using VHDL: A systems approach. Cambridge University Press.

	Description of CO	PO	PSO
CO1	Model synchronous/asynchronous circuits and remove hazards.	PO1(2) PO2(1) PO3(2)	PSO1(3) PSO2(2)
CO2	Examine CMOS issues, low-power methods, and layouts.	PO1(3) PO2(1) PO3(2)	PSO1(2) PSO2(3)
CO3	Evaluate ASIC vs. reconfigurable SoC architectures.	PO1(3) PO2(2) PO3(1)	PSO1(2) PSO2(3)
CO4	Implement analog VLSI blocks for high-frequency use.	PO1(2) PO2(1) PO3(3)	PSO1(1) PSO2(2)
CO5	Simulate and verify digital circuits with HDL.	PO1(2) PO2(2) PO3(2)	PSO1(3) PSO2(2)

Semester II

ET25201	REAL TIME OPERATING SYSTEM	L	T	P	C
		3	1	0	4

Course objectives:

- To review and analyze fundamental operating system concepts
- To understand the principles and architecture of Real-Time Operating Systems (RTOS)
- To study real-time system models, scheduling algorithms, and kernel design issues

Review of Operating Systems: Basic Principles - Operating System structures – System Calls – Files – Processes – Design and Implementation of processes – Communication between processes – Introduction to Distributed operating system – Embedded operating systems

Activities: Identify system calls, process states, and IPC mechanisms for a simple embedded application and compare embedded OS with general-purpose OS.

Overview of RTOS: RTOS Task and Task state –Multithreaded Preemptive scheduler- Process Synchronization- Message queues– Mail boxes -pipes – Critical section – Semaphores – Classical synchronization problem – Deadlocks.

Activities Simulate pre-emptive task scheduling and solve classical synchronization problems using semaphores and message queues.

Real Time Models and Languages: Event Based – Process Based and Graph based Models – Real Time Languages – RTOS Tasks – RT scheduling - Interrupt processing – Synchronization – Control Blocks – Memory Requirements.

Activities: Model a real-time system using event-based and process-based approaches and test schedulability using fixed-priority scheduling algorithms.

Real Time Kernel: Principles – Design issues – Polled Loop Systems – RTOS Porting to a Target – Comparison and Basic study of various RTOS like – VX works – Linux supportive RTOS – C Executive.

Activities: Analyze RTOS kernel components and compare features of VxWorks, Linux-based RTOS, and µC/OS for different application scenarios.

Case Studies: Discussions on Basics of Linux supportive RTOS – uCOS-C Executive for development of RTOS Application – Case study – Automotive Systems.

Activities: Design an RTOS-based application by defining tasks, priorities, and inter-task communication, and analyze a real-world RTOS case study.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Silberschatz, Galvin, Gagne" Operating System Concepts",11th Ed,John Wiley,2020
2. Charles Crowley, "Operating Systems-A Design Oriented approach" McGraw Hill,2009
3. Raj Kamal, "Embedded Systems- Architecture, Programming and Design" Tata McGraw Hill,2013.
4. Karim Yaghmour," Building Embedded Linux System",O'reilly Pub,2008
5. MukeshSighal and N G Shi "Advanced Concepts in Operating System", McGraw Hill,2000

	Description of CO	PO	PSO
CO1	Explain fundamental operating system concepts and analyze their relevance to embedded and real-time systems.	-	-
CO2	Analyze RTOS task management, scheduling algorithms, and synchronization mechanisms for deterministic system behavior.	PO1(3) PO2(3) PO3(2)	PSO1(3) PSO2(2)
CO3	Apply real-time models, scheduling theories, and interrupt handling concepts to evaluate real-time system feasibility.	PO1(2) PO2(3) PO3(2) PO4(2)	PSO1(3) PSO2(2)
CO4	Compare real-time kernel architectures and RTOS implementations such as VxWorks, Linux-based RTOS, and µC/OS for specific applications.	PO1(2) PO2(2) PO4(2) PO5(1)	PSO1(2) PSO2(3)
CO5	Design a basic RTOS-based application by identifying tasks, priorities, and inter-task communication using real-world case studies.	PO1(2) PO2(2) PO3(3)	PSO1(3) PSO2(2)

ET25202	EMBEDDED SYSTEM NETWORKING	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce the fundamentals of embedded networking,
- To explain embedded ethernet technologies and protocols
- To familiarize students with the building blocks of digital instrumentation
- To enable understanding of the design and integration of networked embedded systems

Embedded Process Communication with Instrument Bus: Embedded networking: Introduction – Cluster of instruments in System: Introduction to bus protocols – comparison of bus protocols – RS 232C, RS 422, RS 485 and USB standards – embedded ethernet – MOD bus, LIN bus, CAN bus and ARINC.

Activities: Students match wired protocols (UART, SPI, I²C, CAN) with applications using live polls or chat-based responses, followed by a short discussion.

Embedded Ethernet: Elements of a network – Inside Ethernet – Building a Network - Hardware options – Cables, Connections and network speed – Ethernet controllers – Inside the internet protocol – Exchanging messages using UDP and TCP – Email for Embedded systems using FTP – Keeping devices and network secure.

Activities Instructor shows an Ethernet frame; students identify fields (MAC, IP, payload) and explain data flow using screen annotation or chat.

Wireless Embedded Networking: Wireless sensor networks – Introduction – Node architecture – Network topology -Localization – Time synchronization – Energy efficient MAC protocols – SMAC – Energy efficient and robust routing – Data centric routing - WSN Applications - Home Control - Building Automation - Industrial Automation

Activities: Students choose the most suitable wireless technology (Wi-Fi, Bluetooth, Zigbee, LoRa) for given use cases via breakout rooms and present justification.

Building System Automation: Sensor Types & Characteristics: Sensing Voltage, Current, flux, Torque, Position, Proximity, Accelerometer - Data acquisition system- Signal conditioning circuit design- microcontroller based & PC based data acquisition – microcontroller for automation and protection of electrical appliances – processor based digital controllers for switching Actuators: Stepper motors, Relays –System automation with multi-channel Instrumentation and interface

Activities: Students collaboratively construct a digital instrumentation system block diagram using an online whiteboard (Jamboard / Miro).

Communication For Large Electrical System Automation: Data Acquisition, Monitoring, Communication, Event Processing, and Polling Principles, SCADA system principles – outage management– Decision support application - substation automation, extended control feeder automation, Performance measure and response time, SCADA Data Models, need, sources, interface

Activities: Students design a networked measurement and control system (e.g., smart energy meter) and explain components, networking method, and control strategy.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Mohammad Ilyas And ImadMahgoub, 'Handbook of sensor Networks: Compact wireless and wired sensing systems', CRC Press,2005
2. Peter W Gofton , "Understanding Serial Communication", Sybes International, 2000
3. Jan Axelson 'Embedded Ethernet and Internet Complete', Penram publications, 2003
4. Krzysztof Iniewski, "Smart Grid, Infrastructure & Networking", TMCGH,2012
5. Control and automation of electrical power distribution systems, James Northcote-Green, Robert Wilson, CRC, Taylor and Francis, 2006

	Description of CO	PO	PSO
CO1	Explain the fundamentals of wired and wireless embedded networking techniques and communication protocols.	-	-
CO2	Analyze embedded Ethernet architectures, frame structures, and protocol stacks for embedded applications.	PO1(3) PO2(3) PO3(2) PO5(1)	PSO1(3) PSO2(2)
CO3	Apply wireless embedded networking technologies for selecting suitable solutions in IoT and industrial applications.	PO1(2) PO2(3) PO3(2) PO5(2)	PSO1(3) PSO2(3)
CO4	Explain the fundamental building blocks of digital instrumentation and programmable measurement systems.	PO1(2) PO2(2) PO4(1) PO5(1)	PSO1(2) PSO2(3)
CO5	Design a basic networked embedded system for programmable measurement and control of electrical/electronic devices.	PO1(2) PO2(2) PO3(3)	PSO1(3) PSO2(2)

ET25203	EMBEDDED CONTROL SYSTEM	L	T	P	C
		3	1	0	4

Course objectives:

- To introduce fundamental concepts of control systems and model-based design techniques for digital control implementation.
- To provide knowledge of advanced control strategies for induction, BLDC, and SRM motor drives using modern processors and power electronics.
- To expose students to intelligent control techniques and real-time implementation aspects using FPGA, ARM, and AI-based controllers.

Control System Fundamentals: Overview of open loop control - closed loop control - analysis of simple control loops - stability - time and frequency domain specifications of control system performance - simple approaches for controller design – discretization - practical realization of a control loop.

Activities: Students analyze a given real-world system (e.g., temperature or speed control) to identify open-loop and closed-loop components and discuss stability and performance parameters.

Model Based Control System: Model Based Control System Design - discrete systems, notion of state, Finite State Machines, Extended State Machines, Model based design, code generation, verification and validation, HIL, MIL, SIL, PIL. Performance assessment of control algorithms on the target implementation architectures.

Activities Students map the complete model-based design flow (modeling → code generation → MIL/SIL/PIL/HIL) for a simple control application using a block diagram.

Induction Motor Control: Types- Speed control methods-PWM techniques- VSI fed three-phase induction motor- Fuzzy logic Based speed control for three phase induction motor-FPGA based three phase induction motor control.

Activities: Students compare different PWM techniques for induction motor speed control and justify the choice for a given operating condition.

BLDC Motor Control: Overview of BLDC Motor -Speed control methods -PWM techniques- ARM processor based BDLC motor control- ANN for BLDC Motor control and operation.

Activities: Students select an appropriate BLDC motor control method and controller (ARM/ANN-based) for an application such as EV or drone propulsion and explain their choice.

SRM Motor Control: Overview of SRM Motor -Speed control methods -PWM techniques- FPGA based SRM motor control- DNN for SRM Motor control and operation.

Activities: Students discuss how FPGA-based control and DNN can improve SRM performance and list advantages over conventional control methods.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Tim Wiscott, “Applied control for embedded systems”, Elsevier Publications, 2011,1st Edition.
2. Jim Ledin, “Embedded control systems in C/C++”, CRC Press, 2003, 1st Edition.
3. Vedam Subramanyam, “Electric Drives – Concepts and Applications”, Tata McGraw- Hill publishing company Ltd., New Delhi, 2002
4. K. Venkataratnam , “Special Electrical Machines”, Universities Press, 2014.
5. Karl Johan Astrom, Bjorn Wittenmark, “Computer Controlled Systems”, Dover Publications, 2011.

6. Ron Sass and Anderew G.Schmidt, “Embedded System design with platform FPGAs: Principles and Practices”, Elsevier, 2010.
7. J. W. Valavano, “Embedded Microcomputer Systems: Real-time Interfacing”, Thompson Asia, 2011.

	Description of CO	PO	PSO
CO1	Analyze open-loop and closed-loop control systems using time-domain and frequency-domain performance specifications.	PO1(3) PO2(3) PO4(2)	PSO1(2) PSO2(1)
CO2	Apply model-based control system design methodologies including state models, FSMs, and validation techniques such as MIL, SIL, PIL, and HIL.	PO1(3) PO2(3) PO3(2) PO4(2)	PSO1(3) PSO2(2)
CO3	Design and evaluate induction motor speed control schemes using PWM techniques and intelligent controllers.	PO1(3) PO2(2) PO5(2) PO6(1)	PSO1(3) PSO2(3)
CO4	Analyze and implement BLDC motor control strategies using ARM-based platforms and AI techniques such as ANN.	PO1(2) PO2(2) PO3(3) PO5(2)	PSO1(2) PSO2(3)
CO5	Develop control strategies for SRM drives using FPGA-based architectures and advanced learning methods such as DNN.	PO1(2) PO3(2) PO5(3)	PSO1(3) PSO2(3)

ET25204	EMBEDDED SYSTEM LABORATORY – I	L	T	P	C
		0	0	4	2

Course objectives:

- To provide hands-on experience using workbenches, software tools, and hardware processor boards along with supporting peripherals.
- To develop skills in algorithm design and programming through practical implementation using software tools and platforms.
- To encourage effective use of open-source and commercial hardware–software tools through structured experiments that reinforce concepts learned in theory courses.

Sl.No	EXPERIMENT DETAIL	EQUIPMENT/ SUPPORTS REQUIRED
1.	Programming exercises to understand the GPIO with required external or onboard features.	Any embedded hardware like Arduino/ESP32/STM32/Raspberry Pi development Boards with peripherals; Board Support Software Tools, peripherals with interface
2	Timers/Interrupts/ Serial port programming exercises	
3.	PWM Generation/ Motor Control/ADC/DAC Programming exercises	
4.	Programming Exercise to understand the concepts of BLE, WiFi and other communication protocol useful in IoT applications	
5.	Programming exercise to understand the concepts of Webserver – MQTT – HTTP	
6	Simple Home automation programming exercise using IoT	
7.	Simulation of RTOS a) Creation of Simple Task b) Priority Assignment c) Queues, Mutex and Binary Semaphore Counting Semaphore and Software Timers	1.PC with atleast i3 configuration with Windows or Linux OS. 2.Free version of Microsoft Visual Studio - VSCODE software with FreeRTOS installed 3.Tools and Software Installation Support. https://www.freertos.org/Documentation/01-FreeRTOS-quick-start

8.	RTOS Programming in Microcontroller / Processor.	Open Source RTOS or any commercial RTOS can be used and the example programs can be demonstrated in any 8bit or higher bit microcontrollers like Arduino, ESP32, LPC controllers, etc.,

	Description of CO	**PO**	**PSO**
CO1	Experiment and demonstrate with simulators, in programming processor boards, processor interfacing/ designing digital controllers	PO1(3) PO2(2) PO3(3) PO5(3)	PSO1(3) PSO2(2)
CO2	Design & simulate Arithmetic, Logic programs, Filters, Signal analysis with simulators/experiments, in programming processor boards, processor interfacing/ Tools	PO1(3) PO2(3) PO3(2) PO4(2)	PSO1(3) PSO2(2)
CO3	Develop real time solution for embedded applications.	PO1(2) PO2(3) PO3(3) PO6(1)	PSO1(3) PSO2(2)
CO4	Program and compile in various tools & software domains.	PO1(2) PO2(2) PO5(2) PO6(1)	PSO1(2) PSO2(2)
CO5	Improved Employability and entrepreneurship capacity due to knowledge up gradation on recent trends in commercial embedded processors and its programmable interfacing..	PO1(1) PO6(1) PO7(1)	PSO1(3) PSO2(3)

Programme Elective Courses

ET25001	WIRELESS AND MOBILE COMMUNICATION	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce the fundamentals of wireless channel planning, mobile radio propagation, and impairments affecting wireless communication systems.
- To explain equalization, diversity techniques, and wideband code division multiple access for reliable wireless transmission.
- To provide an understanding of wireless multiple access techniques and IP-based communication in mobile networks..

The Cellular Concept: System Design Fundamentals: Introduction, Frequency Reuse, Channel Assignment Strategies, Handoff Strategies-Prioritizing Handoffs, Practical Handoff Considerations, Interference and system capacity –Co channel Interference and system capacity, Channel planning for Wireless Systems, Adjacent Channel interference, Power Control for Reducing interference, Trunking and Grade of Service, Improving Coverage & Capacity in Cellular Systems-Cell Splitting, Sectoring.

Activities: Allocate frequencies on a cellular map and simulate handoff scenarios to study interference and coverage.

Mobile Radio Propagation - Large-Scale Path Loss: Introduction to Radio Wave Propagation, Free Space Propagation Model, Relating Power to Electric Field, Diffraction-Fresnel Zone Geometry, Knife edge Diffraction Model, Multiple knife-edge Diffraction, Scattering, Outdoor Propagation Models-Longley-Ryce Model, Okumura Model, Hata Model, Indoor Propagation Models-Partition losses, Partition losses between Floors, Log-distance path loss model, Ericsson Multiple Breakpoint Model, Attenuation Factor Model, Signal penetration into buildings, Ray Tracing and Site Specific Modelling.

Activities Measure or simulate signal strength over distance and apply diversity techniques to improve reliability.

Mobile Radio Propagation: Small –Scale Fading and Multipath: Small Scale Multipath propagation-Factors influencing small scale fading, Doppler shift, Impulse Response Model of a multipath channel-Relationship between Bandwidth and Received power, Small-Scale Frequency Domain Channels Sounding, Parameters of Mobile Multipath Channels-Time Dispersion Parameters, Coherence Bandwidth, Doppler Spread and Coherence Time, Types of Small-Scale Fading-Fading effects Due to Multipath Time Delay Spread, Flat fading, Frequency selective fading, Fading effects Due to Doppler Spread-Fast fading, slow fading, Fundamentals of Equalization, Training A Generic Adaptive Equalizer, Equalizers in a communication Receiver, Linear Equalizers, Nonlinear Equalization

Activities: Simulate multiple users transmitting simultaneously and observe how spreading codes separate signals.

Wideband Code Division Multiple Access: CDMA system overview -air interface –physical and logical channel–speech coding, multiplexing and channel coding – spreading and modulation: frame structure, spreading codes-uplink-downlink – physical layer procedures: cell search and synchronization-establishing a connection-power control- handover-overload control.

Activities: Compare TDMA, FDMA, and CDMA methods in a table and discuss best-fit scenarios.

IP Mobility Framework: Challenges of IP Mobility -Address Management -Dynamic Host Configuration Protocol and Domain Name Server Interfaces –Security – Mobility-Based AAA Protocol -IP Mobility Architecture Framework -X Access Network -IPv6 Challenges for IP Mobility.

Activities: Configure a small IP-based mobile network using a simulator and analyze mobility and QoS issues.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Theodore, S. Rappaport, “Wireless Communications, Principles, Practice”, 2nd Ed., 2002, PHI.
2. Andrea Goldsmith, “Wireless Communications”, Cambridge University Press, 2005.
3. Principles of Wireless Networks –Kaveh PahLaven and P. Krishna Murthy, 2002, PE
4. Gottapu Sasibhushana Rao, “Mobile Cellular Communication”, Pearson Education, 2012.
5. Schiller, Jochen H, “Mobile communications”, Pearson, 2nd Edition, 2007.
6. William Stallings, “Wireless Communication and Networking”, PHI, 2003.

	Description of CO	PO	PSO
CO1	Understand Cellular communication concepts	PO1(3) PO2(2) PO3(1)	PSO1(2) PSO2(1)
CO2	Explain the mobile radio propagation	PO1(3) PO2(2) PO3(1)	PSO1(3) PSO2(2)
CO3	Perceive the wireless network different type of MAC protocols	PO1(2) PO2(2) PO3(2)	PSO1(3) PSO2(3)
CO4	Analyse the Equalization and Diversity	PO1(2) PO2(3) PO3(2) PO5(1)	PSO1(2) PSO2(3)
CO5	Build the Wireless multiple access and IP	PO1(2) PO2(2) PO3(3)	PSO1(3) PSO2(3)

ET25002	**DATA DRIVEN EMBEDDED SYSTEMS**	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce Python programming and libraries for data acquisition, manipulation, and visualization.
- To develop embedded applications on ESP32, including interfacing with sensors and real-time data processing.
- To understand and implement data transfer methods, industrial communication protocols, and APIs for system integration.

Python Programming for Data Driven Embedded Systems: Installing Python – Basics – Tuple – List – Set – Dictionary – NumPy – Pandas - Data Structures – Import, Export and Save Data – Data Manipulation – Basic Plotting and Visualization.

Activities: Perform basic Python exercises to manipulate data using lists, dictionaries, NumPy, and Pandas, and visualize results with plots.

ESP32 Embedded Programming and Development Environment: Programming Tools Concepts - Thonny IDE - Esptool - working with files – First Steps in Programming – Handling GPIO - LED flasher as alarm system simulator - Analog Signal Generation – Interrupts and Timers.

Activities Program the ESP32 to blink LEDs, read/write files, and handle GPIO, interrupts, and timers using Thonny IDE.

Accessing Sensor Data: Acquisition of measurement and sensor values – Voltage measurement and linearity correction - Linearization by limitation of the value range - Linearization of the ADC input - Voltage measurement – Data Access from Touch sensor, temperature sensor, Air pressure and altitude measurement, Hall Sensor.

Activities: Acquire and process data from sensors like temperature, pressure, touch, and Hall effect, including ADC linearization and calibration.

Methods of Data Transfer: Data transfer methods – Python Modbus communication - OPC UA - WebSocket – RFID wireless data transmission – MQTT protocol – Concepts of Application Program Interfaces (APIs) - REST – GraphQL-gPRC.

Activities: Implement data communication using Python via Modbus, MQTT, OPC UA, REST, and simulate simple API-based data exchange.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Donald Norris , “Python for Microcontrollers: Getting Started With MicroPython”, Tab Books, 2016
2. Dr Günter Spanner , “MicroPython for Microcontrollers - Projects with Thonny-IDE, uPyCraft-IDE, and ESP32”, Elektor Publication, 2021.
3. Nicholas h Tollervey, “Programming with Micropython: Embedded Programming with Microcontrollers and Python”, O'Reilly Media, 2017.

E-Resources:

1. https://docs.micropython.org/en/latest/esp32/quickref.html
2. https://apmonitor.com/dde/index.php/Main/CourseSchedule

	Description of CO	PO	PSO
CO1	Apply Python programming and data manipulation techniques using NumPy and Pandas for embedded system applications	PO1(3) PO2(2) PO3(2)	PSO1(2) PSO2(3)
CO2	Develop and deploy firmware for ESP32-based systems using tools such as Thonny IDE and Esptool.	PO1(3) PO2(2) PO3(3)	PSO1(3) PSO2(3)
CO3	Interface and acquire data from various sensors and apply basic signal linearization techniques.	PO1(3) PO2(2) PO5(3)	PSO1(3) PSO2(2)
CO4	Implement various data communication protocols including Modbus, MQTT, OPC UA, and REST APIs for real-time embedded applications.	PO1(2) PO2(3) PO5(2) PO6(1)	PSO1(2) PSO2(3)
CO5	Design and demonstrate a functional data-driven embedded system integrating sensing, computing, and data transfer.	PO1(2) PO2(2) PO3(3)	PSO1(3) PSO2(3)

ET25003	ADVANCED EMBEDDED PROCESSOR	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce embedded system architecture and ARM Cortex-M processor fundamentals.
- To understand AMBA protocols (AHB, APB, AXI) and their role in SoC interconnects.
- To learn interfacing techniques for connecting ARM processors with external devices and networks.
- To explore on-chip peripherals, memory systems, and their integration in embedded applications.
- To provide an overview of high-performance processors like RISC-V, vector processors, and GPUs.

Introduction to ARM processor based Embedded Systems: Introduction to embedded system architectures – Elements of SoC Solutions - Overview of chip design specifications, Cycle time, Die area, Power performance Introduction to Arm Cortex – M processor architecture – Arm Cortex – M programmer's model – Exception and interrupts – Memory protection unit

Activities: Analyze the ARM Cortex-M programmer's model and identify exception and interrupt handling through block diagrams and discussion.

Advanced Microcontroller Bus Architecture (AMBA): Introduction to Interconnects – Building an Interconnect protocol – Interconnecting multiple requesters and completers AHB Lite Interconnect Protocols – APB Interconnect Protocol- AXI Interconnect protocol Case studies of AMBA in real-world applications: Design and verification of AMBA-based systems - Performance analysis and optimization of AMBA-based designs **Activities** Draw and explain AHB, APB, and AXI interconnect protocols and discuss their real-world applications in groups..
ARM Cortex Interface With External World: External Bus Architectures – Serial interfacing techniques: UART'S, RS-232, RS-485, SPI, QUAD SPI, I2C, I2S, CAN, USB – Network: LANs, wired LANs, Wi-Fi, Bluetooth, Zigbee, LoRa and LoRaWAN. **Activities:** Simulate or diagram serial and network interfacing techniques (UART, SPI, I2C, CAN, Wi-Fi, Bluetooth, LoRa) and explain data flow.
Peripherals and Memory System: GPIOs – AHB UART – AHB TIMER – ADC and DAC – AHB SPI – Case Study IP Peripheral Integration with Cortex – M1 Case Studies: Industrial Robotics - Collaborative Robots (Cobots) Introduction to Memory System – Soc memory – Cache memory – Brief overview of Volatile & Non-Volatile memory **Activities:** Demonstrate integration of GPIO, timers, ADC/DAC, and SPI peripherals with Cortex-M in case-study scenarios such as industrial robots.
HIGH PERFORMANCE PROCESSORS : Introduction and Architecture of RISC V, Vector processor, GPU - Applications. **Activities:** Compare RISC-V, vector processors, and GPUs in terms of architecture and applications through a group presentation.
Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%
Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. René Beuchat, Andrea Guerrieri, Sahand Kashani, “Fundamentals of System-on-Chip Design on Arm Cortex-M Microcontrollers”, Arm Education media, 2021.
2. Steve Furber, ‘ARM system on chip architecture’, Addision Wesley,2010.
3. Alexander G. Dean, “Embedded Systems Fundamentals with Arm Cortex-M based Microcontrollers”, Nucleo-F091RC Edition, 2021
4. David J. Greaves, “Modern System-on-Chip Design on Arm”, Arm Education media, 2021
5. David Patterson and Andrew Waterman, “The RISC-V Reader: An Open Architecture Atlas”, Strawberry Canyon, 2017.

	Description of CO	PO	PSO
CO1	Demonstrate about basic concepts of embedded system	PO1(3) PO2(2) PO5(1)	PSO1(2) PSO2(1)
CO2	Design a System-on-chip (SoC) architectures based on AMBA	PO1(3) PO2(3) PO3(3) PO6(2)	PSO1(3) PSO2(2)
CO3	Integrate and configure on-chip peripherals such as GPIOs and memory components in ARM-based SoCs.	PO1(3) PO2(3) PO5(3)	PSO1(3) PSO2(3)
CO4	Design a macro level system of industrial robotics based on Arm processor	PO1(3) PO2(3) PO3(2) PO6(1)	PSO1(3) PSO2(3)
CO5	Describe the architecture and application domains of high-performance processors.	PO1(2) PO2(1) PO6(1)	PSO1(1) PSO2(2)

ET25004	DSP BASED SYSTEM DESIGN	L	T	P	C
		3	0	0	3

Course objectives:

- To familiarize the various architectures of DSP system
- To perform analysis of DSP architectures and to learn the implementation of DSP system in programmable hardware
- To learn the details of DSP system interfacing with other peripherals

Representation of DSP System: Single Core and Multicore, Architectural requirement of DSPs - high throughput, low cost, low power, small code size, embedded applications. Representation of digital signal processing systems - block diagrams, signal flow graphs, data-flow graphs, dependence graphs. Techniques for enhancing computational throughput - parallelism and pipelining.

Activities: Draw and compare block diagrams, signal-flow graphs, and data-flow graphs for a simple DSP system and discuss parallelism and pipelining.

DSP Algorithms: DSP algorithms - Convolution, Correlation, FIR/IIR filters, FFT, adaptive filters, sampling rate converters, DCT, Decimator, Expander and Filter Banks. DSP applications. Computational characteristics of DSP algorithms and applications, Numerical representation of signals-word length effect and its impact, Carry free adders, Multiplier.

Activities: Manually compute and compare the steps of FIR filtering and FFT for a given input signal to understand computational characteristics.

System Architecture: Introduction, Basic Architectural Features, DSP Computational Building Blocks, Bus Architecture and Memory, Data Addressing Capabilities, Address Generation MODULE, Programmability and Program Execution, Features for External Interfacing. VLIW architecture. Basic performance issue in pipelining, Simple implementation of MIPS, Instruction Level Parallelism, Dynamic Scheduling, Dynamic Hardware Prediction, Memory hierarchy. Study of Fixed point and floating-point DSP architectures

Activities: Analyze a DSP processor architecture diagram and identify memory, bus, addressing modes, and pipelining features.

Architecture Analysis on Programmable Hardware: Analysis of basic DSP Architectures on programmable hardware. Algorithms for FIR , IIR, Lattice filter structures, architectures for real and complex fast Fourier transforms, 1D/2D Convolutions, Winograd minimal filtering algorithm. FPGA: Architecture, different sub-systems, design flow for DSP system design, mapping of DSP algorithms onto FPGA.

Activities: Map a basic FIR or FFT algorithm onto an FPGA block diagram and explain the design flow.

System Interfacing: Examples of digital signal processing algorithms suitable for parallel architectures such as GPUs and multi-GPUs. Interfacing: Introduction, Synchronous Serial Interface CODE, A CODEC Interface Circuit, ADC interface.

Activities: Illustrate the interfacing of a DSP processor with ADC, CODEC, or GPU and explain the data flow.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Sen M Kuo, Woon Seng S Gan, "Digital Signal Processors – Architectures, Implementations, and Applications", Pearson Education, 2005.
2. Peter Pirsch, "Architectures for Digital Signal Processing", John Weily, 2007
3. DSP Processor and Fundamentals: Architecture and Features. Phil Lapsley, JBier, AmitSohan, Edward A Lee; Wiley IEEE Press
4. K. K. Parhi - VLSI Digital Signal Processing Systems - Wiley – 1999.
5. Rulph Chassaing, "Digital signal processing and applications with C6713 and C6416 DSK", Wiley, 2005
6. Keshab K Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation", student Edition, Wiley, 1999.

	Description of CO	PO	PSO
CO1	Evaluate the DSP system using various methods.	PO1(3) PO2(3) PO4(1)	PSO1(2) PSO2(1)
CO2	Design algorithm suitable for different DSP applications.	PO1(3) PO2(3) PO3(3) PO4(2)	PSO1(3) PSO2(2)
CO3	Explain various architectures of DSP system.	PO1(3) PO2(2) PO3(1)	PSO1(3) PSO2(3)
CO4	Implement DSP system in programmable hardware.	PO1(3) PO2(3) PO3(2) PO5(3)	PSO1(3) PSO2(3)
CO5	Build interfacing of DSP system with various peripherals.	PO1(3) PO2(2) PO5(3)	PSO1(1) PSO2(2)

ET25005	AUTOMOTIVE EMBEDDED SYSTEM	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce the fundamentals of automotive embedded systems including engine control, sensors, actuators, and ECUs.
- To develop understanding of vehicle management, diagnostics, communication protocols, and safety standards used in modern automobiles.
- To familiarize students with embedded system applications in electric, connected, and autonomous vehicles.

Basic of Electronic Engine Control Systems: Overview of Automotive systems, fuel economy, air-fuel ratio, emission limits and vehicle performance; Automotive microcontrollers- Electronic control MODULE- Hardware & software selection and requirements for Automotive applications – open-source ECU- RTOS - Concept for Engine Management-Standards; Introduction to AUTOSAR and Introduction to Society SAE- Functional safety ISO 26262- Simulation and modeling of automotive system components- Introduction to software defined vehicles.

Activities: Students draw a simple ECU block diagram and label components (sensors, actuators, MCU, RTOS, communication).

Sensors and Actuators for Automotives: Review of sensors- sensors interface to the ECU, conventional sensors and actuators, Modern sensor and actuators - camera-radar-gps/gnss, LIDAR sensor- smart sensors- MEMS/NEMS sensors and actuators for automotive applications.

Activities: Provide a list of sensors and automotive functions; students match each sensor to its correct application.

Vehicle Management Systems: Electronic Engine Control-engine mapping, air/fuel ratio spark timing control strategy, fuel control, electronic ignition- Adaptive cruise control - speed control-anti-locking braking system-electronic suspension - electronic steering , Automatic wiper control- body control system ; Vehicle system schematic for interfacing with EMS, ECU. Energy Management system for electric vehicles- Battery management system , power management system-electrically assisted power steering system- Adaptive lighting system- Safety and Collision Avoidance. Integrated vehicle health monitoring system.

Activities: Students create a basic flowchart for systems like ABS, cruise control, or engine management.

Onboard Diagnostics and Telematics: On board diagnosis of vehicles -System diagnostic standards and regulation requirements Vehicle communication protocols Bluetooth, CAN, LIN, FLEXRAY, MOST, KWP2000 and recent trends in vehicle communications- Navigation- Connected Cars technology – Tracking- Security for data communication- dashboard display and Virtual Instrumentation, multimedia electronics- Role of IOT in Automotive systems- automotive ethernet SOME-IP.

Activities: Students prepare a comparison table of CAN, LIN, FlexRay, and MOST based on speed, cost, and application.
Electric Vehicles: Electric vehicles –Components- Plug in Electrical vehicle-Charging station – Aggregators- Fuel cells/Solar powered vehicles- Autonomous vehicles. **Activities:** Students draw and explain the block diagram of an electric vehicle including battery, BMS, motor, and charger.
Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%
Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. William B. Ribbens ,"Understanding Automotive Electronics", Elseiver, 8th Edition, 2017
2. Ali Emedi, Mehrdedehsani, John M Miller , "Vehicular Electric power system-land, Sea, Air and Space Vehicles" Marcel Decker, 2004.
3. L.Vlacic,M.Parent,F.Hirashima,"Intelligent Vehicle Technologies", SAE International,2001.
4. Jack Erjavec, Jeff Arias," Alternate Fuel Technology-Electric, Hybrid & Fuel Cell Vehicles", Cengage ,2013.
5. Ronald K Jurgen, "Electronic Engine Control technology", SAE International. 2004.
6. Tom Denton, "Automotive Electricals / Electronics System and Components", 3rd Edition, 2018.
7. Uwe Kiencke, Lars Nielsen, "Automotive Control Systems: For Engine, Driveline, and Vehicle", Springer; 2005.
8. Automotive Electricals Electronics System and Components, Robert Bosch Gmbh, 2014.
9. Automotive Hand Book, Robert Bosch, Wiley Publishers, 2022.
10. Ronald K. Jurgen, "Automotive Electronics Hand Book", McGraw-Hill, 2000. .

	Description of CO	PO	PSO
CO1	Explain the architecture, standards, and functional principles of automotive embedded and engine control systems.	-	-
CO2	Identify and analyze automotive sensors, actuators, and their interfacing with electronic control units (ECUs).	PO1(3) PO2(3) PO3(2) PO5(2)	PSO1(3) PSO2(3)
CO3	Describe the operation and control strategies of vehicle management systems including safety and driver-assistance features.	PO1(2) PO2(3) PO3(3)	PSO1(3) PSO2(3)
CO4	Apply knowledge of onboard diagnostics, in-vehicle networks, and telematics for vehicle communication and monitoring.	PO1(2) PO2(2) PO4(2) PO5(3)	PSO1(2) PSO2(3)
CO5	Demonstrate understanding of embedded systems used in electric, connected, and autonomous vehicle technologies.	PO1(2) PO2(2) PO5(3)	PSO1(3) PSO2(2)

ET25006	EMBEDDED LINUX	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce the fundamentals of Linux operating systems and command-line tools for embedded applications.
- To provide knowledge of embedded Linux architecture, cross-development toolchains, and system boot processes.
- To familiarize students with Linux programming, cross-compilation, hardware interfacing, and basic device driver development.

Linux Fundamentals: Introduction to Linux: A brief History - Features and Advantages of Linux - System and Software Features - Linux's Copyright - The Design Philosophy of Linux - Differences between Linux and Other Operating Systems - Hardware Requirements - Source of Linux Information - Obtaining and Installing Linux: Distributions of Linux - Installing Linux. Working with Linux: Logging in and Logging Out - Linux File System - Directory and File Commands - Other Useful Linux Commands - File Access Permissions - Pipes and Filters - Text Editors - Working with GNOME.

Activities: Students draw a simple ECU block diagram and label components (sensors, actuators, MCU, RTOS, communication).

Cross-Development Toolchain: History of Embedded Linux - Embedded Linux Vs Desktop Linux - Types of Hosts - Types of Host/Target Development Setups - Types of Host/Target Debug Setups - Types of Boot Configurations - System Memory Layout. User space - Architecture of Embedded Linux - Linux Kernel Architecture - Linux Start-Up Sequence. GNU Cross Platform Toolchain.

Activities: Provide a list of sensors and automotive functions; students match each sensor to its correct application.

Running Linux on Embedded Boards: Embedded Boards and their Features - Exploring Embedded Linux System: Different Raspberry Pi Boards and their comparison - Embedded Linux Introduction - Managing Linux Systems - Using Git for Version Control - Using Desktop Virtualization. Programming on the Raspberry Pi: Scripting Languages - Dynamically Compiled Languages - C and C++ on the RPi - Overview of Object- Oriented Programming - Interfacing to the Linux OS - Improving the Performance of Python.

Activities: Students create a basic flowchart for systems like ABS, cruise control, or engine management.

Cross-Compilation and Interfacing to the Raspberry PI: Cross-Compilation and the Eclipse IDE: Setting Up a Cross-Compilation Toolchain - Cross- Compilation

Using Eclipse - Building Linux. Interfacing to the Raspberry Pi Busses: Introduction to Bus Communication - I2C - SPI - UART - Logic-Level Translation

Activities: Students prepare a comparison table of CAN, LIN, FlexRay, and MOST based on speed, cost, and application.

Introduction to Linux Device Drivers: Device Driver Basics: User Space and Kernel Space - Driver Skeletons - Errors and Message Printing - Module Parameters - Building First Module. Character Device Drivers: Concept behind Major and Minor - Introduction to Device File Operations - Allocating and Registering a Character Device - Writing File Operations.

Activities: Students draw and explain the block diagram of an electric vehicle including battery, BMS, motor, and charger.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, and Philippe Gerum, "Building Embedded Linux Systems", O'Reilly Media Inc., 2008.
2. P. Raghavan, Amol Lad and Sriram Neelakandan, "Embedded Linux System Design and Development", Auerbach Publications, Taylor & Francis Group, 2006.
3. Derek Molloy, "Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux", John Wiley & Sons, Inc., 2016.
4. John Madieu, "Linux Device Drivers Development: Develop customized drivers for embedded Linux", Packt Publishing, 2017.

	Description of CO	PO	PSO
CO1	Explain Linux OS concepts, file systems, permissions, and commonly used commands.	-	-
CO2	Describe the architecture and startup sequence of embedded Linux systems and cross-development environments.	PO1(3) PO2(2) PO4(1) PO5(2)	PSO1(3) PSO2(2)
CO3	Develop and execute Linux applications on embedded platforms such as Raspberry Pi using C/C++ and scripting languages.	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(3)
CO4	Apply cross-compilation techniques and interface embedded Linux systems with hardware communication buses.	PO1(2) PO2(3) PO3(3) PO5(2)	PSO1(3) PSO2(3)
CO5	Understand the structure and working of Linux device drivers and implement basic character device drivers.	PO1(3) PO2(3) PO5(1)	PSO1(3) PSO2(2)

ET25007	AUTONOMOUS VEHICLE	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce the fundamentals, architectures, and dynamics of autonomous ground, aerial, and underwater vehicles.
- To provide knowledge of perception, localization, navigation, and path-planning techniques used in autonomous systems.
- To familiarize students with safety standards, communication technologies, and emerging trends in autonomous mobility.

Fundamentals of Autonomous Ground Vehicles: Evolution of autonomous driving systems - SAE levels of autonomy - Vehicle dynamics and control basics - Drive-by-wire systems: steering, braking, throttle - Embedded systems architecture in automotive applications

Activities: Students map real vehicles (Tesla, Waymo, ADAS cars) to SAE autonomy levels and identify required subsystems.

Perception, Localization and Path Planning: Sensors: LiDAR, radar, cameras, ultrasonic sensors - Sensor fusion techniques - Computer vision for autonomous driving (lane detection, traffic sign recognition, obstacle detection) - GPS/INS integration, SLAM, HD maps - Path planning algorithms: A*, RRT, trajectory optimization

Activities: Students match sensors to functions and draw a simple path-planning solution (A* or RRT) for a given road scenario.

Autonomous UAVs (Unmanned Aerial Vehicles): UAV types and configurations (fixed-wing, rotary, hybrid) - Flight dynamics and control systems - Embedded avionics: autopilot controllers, IMU, GPS integration - Path planning and navigation in 3D space - UAV applications: surveillance, delivery, disaster management - Safety, regulations, and communication protocols for UAVs

Activities: Students draw and explain a UAV system block diagram including sensors, autopilot, communication, and control loops.

Autonomous Underwater Vehicle: Remotely Operable Vehicles (ROV) – design and stability – components of ROV – AUV – Gliders – construction – buoyancy driven – Control strategies, AUV – construction – components – control strategies - applications.

Activities: Students prepare a comparison table of ROVs, AUVs, and gliders based on control, power, buoyancy, and applications.

Safety, Standards and Future Directions: Functional safety: ISO 26262, redundancy, fault tolerance - Cybersecurity in autonomous systems - Testing and validation: simulation, hardware-in-the-loop (HIL), scenario-based testing - V2X communication: V2V, V2I, cooperative driving - Ethical, legal, and societal implications of autonomous systems - Case studies: Tesla Autopilot, Waymo, UAV swarm systems, AUV exploration missions

Activities: Group discussion on Tesla Autopilot, Waymo, UAV swarms, or AUV missions focusing on safety, ethics, and standards.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Ishwar K. Sethi (Ed.), Autonomous Vehicles and Systems: A Technological and Societal Perspective, River Publishers, 1st Edition, 2023
2. Thomas Bräunl, Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems, Springer, 4th Edition, 2022
3. P.G. Fahlstrom, T.J. Gleason, Introduction to UAV Systems, Wiley, 5th Edition, 2022.
4. Hans-Leo Ross, Functional Safety for Road Vehicles: New Challenges and Solutions for ISO 26262, Springer, 2nd Edition, 2021.
5. Narayan Panigrahi, Smita Tripathy, ”Design Principles of Autonomous Systems: UAV, UGV, and AUV,CRC Press (Taylor & Francis Group), 1st Edition, 2025.
6. Sabiha Wadoo,”Autonomous Underwater Vehicles: Modeling, Control Design and Simulation”,T & F India / CRC Press Taylor & Francis, 1st Edition, 2019.

	Description of CO	PO	PSO
CO1	Explain the evolution, levels of autonomy, vehicle dynamics, and embedded architectures of autonomous ground vehicles.	-	-
CO2	Describe perception, sensor fusion, localization, and path-planning techniques used in autonomous navigation.	PO1(3) PO2(3) PO4(2) PO5(3)	PSO1(3) PSO2(3)
CO3	Analyze the design, dynamics, control, and applications of autonomous UAV systems.	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(3)
CO4	Explain the construction, buoyancy principles, control strategies, and applications of ROVs and AUVs.	PO1(3) PO2(3) PO3(2) PO5(2)	PSO1(3) PSO2(3)
CO5	Understand safety standards, cybersecurity, V2X communication, and societal implications of autonomous systems through real-world case studies.	PO1(2) PO2(2) PO7(3)	PSO1(2) PSO2(2)

ET25008	COMPUTER VISION	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce the fundamentals of digital image processing and computer vision principles.
- To develop understanding of image features, object detection techniques, and vision algorithms.
- To provide exposure to practical computer vision applications using OpenCV and real-world case studies.

Introduction to Computer Vision: Digital Image Processing – Various Fields that use Image Processing – Fundamentals Steps in Digital Image Processing – Components of an Image Processing System. Applications of Computer Vision – Recent Research in Computer Vision. Introduction to Computer Vision and Basic Concepts of Image Formation: Introduction and Goals – Image Formation and Radiometry – Geometric Transformation – Geometric Camera Models – Image Reconstruction from a Series of Projections.

Activities: Students draw and label the basic steps of an image processing and computer vision pipeline for a given application.

Image Processing Concepts and Image Features: Image Processing Concepts: Fundamentals – Image Transforms – Image Filtering – Colour Image Processing – Mathematical Morphology – Image Segmentation. Image Descriptors and Features:

Texture Descriptors – Colour Features – Edge Detection – Object Boundary and Shape Representation – Interest or Cornet Point Detectors – Histogram Oriented Gradients – Scale Invariant Feature Transform.

Activities: Given sample images, students identify edges, textures, corners, and shapes using conceptual explanations.

Image Processing with OPENCV: Introduction to OpenCV and Python: Setting up OpenCV – Image Basics in OpenCV – Handling Files and Images – Constructing Basic Shapes in OpenCV. Image Processing in OpenCV: Image Processing Techniques – Constructing and Building Histograms – Thresholding Techniques.

Activities: Instructor shares a simple OpenCV Python code; students predict output and explain each processing step.

Object Detection: Models and types – Importance of Object Detection. The Working: Inputs and outputs – Basic Structure – Model Architecture Overview – Object Detection on the Edge. Use Cases and Applications: Video Surveillance – Self-driving Cars. Embedded Boards: Connecting Cameras to Embedded Boards – Simple algorithms for processing Images and Videos.

Activities: Students prepare a flowchart explaining the working of an object detection system (input → model → output).

Applications and Case Studies: Applications: Machine Learning algorithms and their Applications in Medical Image Segmentation – Motion Estimation and Object Tracking – Face and Facial Expression Recognition – Image Fusion. Case Studies: Face Detection – Object Tracking – Eye Tracking – Handwriting Recognition with HoG.

Activities: Students discuss one application (face detection, medical imaging, tracking) and explain the technique used.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Rafael C Gonzalez and Richard E Woods, “Digital Image Processing”, 4th Edition (Global Edition), Pearson Education Limited, 2018.
2. Manas Kamal Bhuyan, “Computer Vision and Image Processing - Fundamentals and Applications”, CRC Press, 2020.
3. Alberto FernándezVillán, “Mastering OpenCV 4 with Python”, Packt Publishing, 2019.

4. Adrian Rosebrock ,“Practical Python and Open CV: Case Studies”, 3rd Edition, PyImage Search, 2016.

	Description of CO	PO	PSO
CO1	Explain the fundamentals of image formation, digital image processing, and computer vision systems.	-	-
CO2	Analyze image features, descriptors, segmentation, and transformation techniques.	PO1(3) PO2(3) PO3(2) PO4(2)	PSO1(3) PSO2(3)
CO3	Apply OpenCV tools to perform basic image processing and analysis tasks.	PO1(2) PO2(2) PO3(3)	PSO1(2) PSO2(3)
CO4	Describe object detection models, architectures, and their deployment in embedded and edge systems.	PO1(2) PO2(3) PO3(3) PO5(2)	PSO1(3) PSO2(3)
CO5	Interpret and apply computer vision techniques to real-world applications and case studies.	PO1(2) PO2(2) PO6(2)	PSO1(2) PSO2(2)

ET25009	DIGITAL TWIN	L	T	P	C
		3	0	0	3

Course objectives:

- Introduce students to Digital Twin concepts, lifecycle, and enabling technologies.
- Enable students to develop physics-based and data-driven models of physical systems.
- Provide hands-on experience with open-source tools for Digital Twin development.
- Build skills in data integration, visualization, and analytics.
- Encourage project-based learning for real-world Digital Twin applications.

Introduction to Digital Twins: Evolution: Digital Models - Digital Shadows - Digital Twins - Definitions, characteristics, and taxonomy - Lifecycle, value chain, and reference architectures - Enabling technologies: IoT, simulation, AI, cloud, edge - Digital Twin vs Simulation vs Cyber-Physical Systems -Industry use cases: Manufacturing, Smart Cities, Healthcare, Robotics

Activities: Case study discussion: Digital Twin of a Smart Factory or Autonomous Robot

System Modeling for Digital Twins: Physical system abstraction and decomposition - Physics-based vs data-driven modeling - Model fidelity, validation, and uncertainty - Co-simulation concepts and model integration. **Activities:** Build a simple physics-based model (e.g., motor, thermal system) and validate against simulated data.
Data, Sensors, and IoT Integration: Sensors and data acquisition for Digital Twins - Real-time data pipelines: MQTT, REST APIs - Edge vs Cloud Digital Twins- Data quality, latency, and synchronization. **Activities:** Simulate sensor data streaming and connect to a simple Digital Twin model.
Simulation, Analytics, and Visualization: Real-time simulation and co-simulation - State estimation, prediction, and data-driven twins using ML/DL - Visualization dashboards (3D, real-time) - Human-in-the-loop Digital Twins **Activities:** Integrate simulation, sensor data, and dashboard visualization for a mini Digital Twin.
Digital Twin Architecture & Capstone Project: Reference architectures (ISO 23247, RAMI 4.0) - Deployment strategies, scalability, and cybersecurity - Ethics, data governance, and sustainability - Digital Twin maturity levels. **Project Work / Activities (PBL):** Define problem, system modeling, data integration, visualization, validation, and demonstration – Smart Energy Meter twin – Mobile Robot.
Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%
Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Grieves, M., & Vickers, J. “Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior”, Springer, 1st Edition, 2017.
2. Sabri, S., et al. “Digital Twin: Fundamentals and Applications”, Academic Press, 1st Edition, 2021.
3. Surjya Kanta Pal et al., “Digital Twin – Fundamental Concepts to Applications in Advanced Manufacturing”, CRC Press, 1st Edition, 2020.
4. Shyam Varan Nath & Pieter van Schalkwyk., “Building Industrial Digital Twins: Design, Develop, and Deploy Digital Twin Solutions for Real-World Industries”, Packt Publishing, 1st Edition, 2022.
5. Dhanaraj, R.K., et al., “Digital Twin for Smart Manufacturing: Emerging Approaches and Applications”, Springer, 1st Edition, 2022.

6. Vijay Raghunathan, Santanu Deb Barma," Digital Twin: A Complete Guide For the Complete Beginner", Kindle Edition, 2019

	Description of CO	PO	PSO
CO1	Explain the fundamental concepts, lifecycle, and taxonomy of Digital Twin systems.	-	-
CO2	Develop physics-based and data-driven models of physical systems suitable for digital twinning.	PO1(3) PO2(3) PO3(2) PO4(2)	PSO1(3) PSO2(2)
CO3	Integrate sensor and IoT data streams into a Digital Twin model for real-time monitoring and simulation.	PO1(2) PO2(2) PO3(3)	PSO1(2) PSO2(3)
CO4	Apply simulation, analytics, and visualization tools to design and implement Digital Twin applications.	PO1(2) PO2(3) PO3(3) PO5(2)	PSO1(3) PSO2(3)
CO5	Plan, design, and demonstrate a functional Digital Twin system for a selected real-world or industrial application through a project-based approach.	PO1(2) PO2(3) PO5(3)	PSO1(3) PSO2(3)

ET25C02	MACHINE LEARNING AND DEEP LEARNING	L	T	P	C
		3	0	0	3

Course objectives:

- Familiarizing students with the fundamental concepts of machine learning and neural networks.
- Enabling students to acquire knowledge about pattern recognition.
- Motivating the students to apply deep learning algorithms for solving real life problems.

Foundations of Machine Learning in Embedded System: Various paradigms of learning problems, Forms of Learning: Supervised, Semi-supervised, and Unsupervised algorithms, Reinforcement Learning, Machine Learning Terminologies and Model Evaluation: Confusion Matrix, Accuracy, Precision, Recall, F1-Score, the curse of dimensionality, training, testing, validation, cross-validation, overfitting, underfitting, early stopping, regularization, bias and variance, Introduction to TinyML and Edge AI, Challenges in real-time ML deployment: latency, memory, and power.

Activities: Students analyze a small dataset and compute accuracy, precision, recall, and F1-score using a confusion matrix, then discuss overfitting vs underfitting.
Advanced Machine Learning Techniques: Feature Engineering, Dimensionality Reduction, Classifiers: K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Decision Trees, Naïve Bayes, Binary Classification, Multi-class Classification, Clustering, Ensemble learning, Meta Learning, Foundation to Quantum Machine Learning, Low-power classification methods for Embedded controllers. **Activities:** Implement and compare KNN and SVM classifiers on a low-dimensional dataset to observe accuracy vs memory trade-offs for embedded systems.
Neural Networks and Model Training: Differences between Biological and Artificial Neural Networks - General Architecture, Multi-layer neural network, Linear Separability, Hebb Net, Perceptron, Adaline, Standard Back propagation, Training Algorithms for Pattern Association - Hebb rule and Delta rule, Hetero associative, Auto associative, Kohonen Self Organising Maps, Learning Vector Quantization, Gradient descent, Boltzmann Machine Learning, Model training vs inference on embedded systems, Lightweight Neural Network deployment strategies - quantization and pruning **Activities:** Train a simple perceptron or small neural network and apply quantization to compare model size and inference accuracy.
Deep Learning Architectures: Convolutional Neural Networks: Convolution layers, Pooling layers, Fully connected layers, Advanced CNN models: Alexnet, VGGnet, ResNet, Google net, Handling overfitting in CNN: drop out, transfer learning, data augmentation, Recurrent Neural Networks and Autoencoders, State, Structure of RNN Cell, LSTM and GRU, Time Distributed Layers, Autoencoders: Convolutional Autoencoders, Denoising Autoencoders, Variational Autoencoders, Lightweight DL architectures for embedded systems: SqueezeNet, MobileNetV3, and Tiny-YOLO. **Activities:** Fine-tune a lightweight CNN (MobileNet or SqueezeNet) on a small image dataset and evaluate inference speed and accuracy.
Generative Models, Deployment and Real-World Applications: Generative Adversarial Networks (GANs): The Discriminator, Generator, Deep Convolutional GANs (DCGANs), Introduction to BERT Transformer, Real-world applications in NLP, Computer Vision, Healthcare, and Finance, Fundamentals of Federated Learning, Federated Learning in IoT-based embedded networks, Deployment frameworks: TensorFlow Lite, PyTorch Mobile. **Activities:** Deploy a pre-trained TensorFlow Lite or PyTorch Mobile model on an edge device or emulator and measure latency and memory usage.
Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. ShaiShalev-Shwartz and Shai Ben-David, “Understanding Machine Learning”, 1st Edition, Cambridge University Press, 2017.
2. Ian Good fellow, Yoshua Bengio and Aaron Courville, “Deep Learning”, MIT Press,2016.
3. J. S. R. Jang, C. T. Sun, E. Mizutani, “Neuro Fuzzy and Soft Computing - A Computational Approach to Learning and Machine Intelligence”,1st Edition, PHI learning, 2012.
4. Trevor Hastie, Robert Tibshirani and Jerome Friedman, “The Elements of Statistical Learning”,2nd Edition, Springer Series, 2009.
5. P. Warden and D. Situnayake , “TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers”, O’Reilly Media Publishers, 2020.
6. Christopher M. Bishop and Hugh Bishop, Deep Learning: Foundations and Concepts, Springer Nature, 2023
7. Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer,2016.

	Description of CO	PO	PSO
CO1	Evaluate different paradigms of machine learning to determine their suitability for embedded systems.	PO1(3) PO2(3) PO3(2) PO5(2)	PSO1(2) PSO2(1)
CO2	Interpret the architecture and training mechanisms of neural network models suitable for embedded deployment.	PO1(3) PO2(2) PO3(2) PO4(2)	PSO1(2) PSO2(2)
CO3	Apply feature selection, dimensionality reduction, and classification techniques for efficient pattern recognition.	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(2)
CO4	Analyze the design and performance of deep learning models including CNNs, RNNs, and autoencoders in embedded systems.	PO1(3) PO2(3) PO3(3) PO5(2)	PSO1(3) PSO2(3)
CO5	Demonstrate the use of generative models and deployment frameworks in real-world embedded machine learning applications.	PO1(2) PO2(3) PO3(3)	PSO1(2) PSO2(3)

ET25010	WIRELESS SENSOR NETWORKS	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce the architecture, protocols, and design principles of Wireless Sensor Networks.
- To enable students to analyze MAC, routing, transport, and security mechanisms in WSNs with energy and performance constraints.
- To provide exposure to modern open-source tools and simulators for modeling, analyzing, and evaluating WSN applications.

Wireless Sensor Network Architecture: Introduction to wireless sensor networks-Challenges, Comparison with ad hoc network, Node architecture and Network architecture, design principles, Service interfaces, Gateway, Short range radio communication standards-IEEE 802.15.4, Zigbee and Bluetooth. Physical layer and transceiver design considerations.

Activities: Students compare WSN and ad hoc network architectures and identify suitable short-range communication standards for a given application.

Data Link Layer: MAC protocols – fundamentals, low duty cycle protocols and wakeup concepts, contention based protocols, Schedule-based protocols - SMAC, BMAC,TRAMA, Link Layer protocols – fundamentals task and requirements, error control, framing, link management, Naming and addressing – address assignment, unique, Content-based and geographical addressing.

Activities: Analyze and compare SMAC and BMAC protocols based on energy efficiency and delay using a simple timing diagram.

Network Layer: Routing protocols – Requirements, Taxonomy - Data-centric routing – SPIN, Directed Diffusion, Energy aware routing, Gradient-based routing – COUGAR, ACQUIRE, Hierarchical Routing – LEACH, PEGASIS, Location Based Routing – GAF, GEAR, Data aggregation – Various aggregation techniques, Localization and positioning – Properties, Approaches, Mathematical basics for single hop and multi-hop environment.

Activities: Students map different routing protocols (LEACH, PEGASIS, GEAR) to application scenarios such as environmental monitoring or smart agriculture.

Transport Layer: Transport Protocol, Coverage and deployments - Sensing models, Coverage measures, Random deployments: Poisson model, Boolean sensing model, general sensing model, Coverage determination, grid deployment, Reliable data transport, Single packet delivery, Block delivery, Congestion control and rate control – in-network processing, Time synchronization – Issues and protocol – Sender/Receiver, Security – protocols and Key Distribution Techniques.

Activities: Solve numerical problems on coverage models and discuss congestion control strategies using in-network processing examples.

Tools For WSN: Contiki-NG: Overview, architecture, network stack, and features; supported protocols and IPv6/6LoWPAN support with RPL. Cooja Simulator: Simulation of sensor networks with Contiki-NG, node types, radio models, and event scheduling. Overview of RIOT OS.

Activities: Simulate a small WSN using Contiki-NG and Cooja to observe packet delivery and energy consumption.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Hossam Mahmoud Ahmad Fahmy, “Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis”, Springer, 1st Edition, 2016.
2. Nandini Mukherjee, Sarmistha Neogy, Sarbani Roy, “Building Wireless Sensor Networks: Theoretical and Practical Perspectives”, CRC Press, 1st Edition, 2015.
3. Holger Karl , Andreas willig, “Protocol and Architecture for Wireless Sensor Networks”, John Wiley Publication, 2006.
4. Kazem Sohraby, Daniel Minoli and TaiebZnati, “Wireless Sensor Networks Technology Protocols and Applications”, John Wiley & Sons, 2007.
5. Paolo Santi, “Topology Control in Wireless Adhoc and Sensor Networks”, John Wiley & Sons, 2005.
6. Waltenegus Dargie, Christian Poellabauer, “Fundamentals of Wireless Sensor Networks Theory and Practice”, John Wiley and Sons, 2010
7. Yingshu Li, My T. Thai,Weili Wu, “Wireless Sensor Networks and Applications”, Springer, 2008.

	Description of CO	PO	PSO
CO1	Explain the architecture, design challenges, and communication standards used in Wireless Sensor Networks.	-	-
CO2	Analyze data link layer protocols and addressing mechanisms with respect to energy efficiency and scalability.	PO1(3) PO2(3) PO3(2) PO4(2)	PSO1(3) PSO2(1)
CO3	Compare and evaluate WSN routing, data aggregation, and localization techniques for different application scenarios.	PO1(3) PO2(3) PO3(2)	PSO1(3) PSO2(1)
CO4	Examine transport layer mechanisms, coverage models, synchronization, and security techniques in WSNs.	PO1(3) PO2(3) PO3(2) PO5(2)	PSO1(2) PSO2(1)
CO5	Use modern open-source tools to simulate and evaluate the performance of Wireless Sensor Networks.	PO1(2) PO2(2) PO3(2)	PSO1(1) PSO2(3)

ET25011	EMBEDDED COMPUTING	L	T	P	C
		3	0	0	3

Course objectives:

- Understand the principles of networked and distributed embedded computing systems, including security and real-time aspects.
- Apply Java, smart card, and Android technologies for developing embedded and distributed applications.
- Design basic distributed and real-time embedded system solutions using modern frameworks and tools.

Network Infrastructure: Broad Band Transmission facilities –Open Interconnection standards – networking devices Network diagram –Network management – Network Security – Cluster computers.

Activities: Draw and analyze a network diagram for an embedded system with security and management components.

Java Technology for Embedded Systems: Basic concepts of Java - IO streaming – Object serialization – Networking – Threading – RMI – distributed databases –– Advantages and limitations of Internet – Web architecture for embedded systems – security model for embedded systems.

Activities: Develop a simple Java program demonstrating multithreading or socket-based communication.

Smart Card Techniques: Smart Card basics – Java card technology overview – Java card Types - Card components - smart card microcontrollers - Contactless Cards - Smart Card Operating Systems– smart card Security Techniques.

Activities: Analyze the architecture and security features of a Java smart card through a case example.

Android Framework: Android SDK – Access to Hardware - Framework development - Peer-to-Peer communication- Android security design and architecture –overview of automotive android – case study.

Activities: Design a basic Android application that accesses a hardware feature or simulates peer-to-peer communication.

Developing Distributed Real-Time System Applications: Developing MATLAB Real-Time Targets - Using the xPC Target - Building various Distributed Real Time Applications.

Activities: Demonstrate or simulate a simple distributed real-time application using MATLAB/xPC Target.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Amitava Gupta, Anil Kumar Chandra & Peter Luksch, “Original: Real-Time and Distributed Real-Time Systems: Theory and Applications”, CRC Press, 2016.
2. Hermann Kopetz, “Real-Time Systems: Design Principles for Distributed Embedded Applications”, 3rd Edition, Springer, 2022.
3. Wolfgang Rankl & Wolfgang Effing, “Smart Card Handbook”, 4th Edition, Wiley, 2010.
4. Reto Meier & Ian Lake, “Professional Android”, 4th Edition, Wiley, 2018.
5. Joshua “ Android hacker’s Handbook” John Wiley & Sons, 2014
6. Paul J. Deitel & Harvey M. Deitel, “Java How to Program” , 11th Edition, Pearson, 2018.
7. Maarten van Steen & Andrew S. Tanenbaum, “Distributed Systems”, 4th Edition, 2023.

	Description of CO	PO	PSO
CO1	Explain network infrastructure components, protocols, and security mechanisms used in embedded systems.	-	-
CO2	Develop Java-based embedded and distributed applications using threading, networking, and RMI concepts.	PO1(3) PO2(2) PO3(3) PO5(2)	PSO1(3) PSO2(3)
CO3	Analyze smart card architectures, operating systems, and security techniques for secure embedded applications.	PO1(3) PO2(2) PO3(2)	PSO1(3) PSO2(2)
CO4	Use Android framework features to access hardware and develop secure embedded and automotive applications.	PO1(2) PO2(2) PO3(3) PO5(3)	PSO1(3) PSO2(3)
CO5	Design and implement basic distributed real-time system applications using MATLAB/xPC real-time tools.	PO1(2) PO2(2) PO3(3)	PSO1(3) PSO2(3)

ET25012	EMBEDDED SYSTEMS SECURITY	L	T	P	C
		3	0	0	3

Course objectives:

- To introduces the foundational concepts and practical techniques for securing embedded systems
- To explore the vulnerabilities, attack surfaces, and countermeasures in hardware, firmware, and communication interfaces of embedded platforms.

Fundamentals of Embedded Systems and Security: Embedded systems: architecture and applications; Characteristics of embedded systems; Threat landscape for embedded systems; Security goals: confidentiality, integrity, availability, authenticity; Security models and policies; Attack surfaces in embedded systems; Real-world case studies of embedded system breaches.

Activities: Identify a real-world embedded system breach and map the violated security goals (CIAA).

Firmware and Software Security: Firmware reverse engineering and vulnerability analysis; Secure coding practices for embedded software; Common vulnerabilities: buffer overflows, format string attacks; Memory protection mechanisms: DEP, ASLR, MPU; Firmware update mechanisms and secure bootloaders; Secure firmware deployment: digital signatures, encryption; Anti-debugging and obfuscation techniques

Activities: Analyze a sample embedded C code to identify vulnerabilities such as buffer overflow and suggest fixes.

Communication and Network Security in Embedded Systems: Protocols used in embedded systems (CAN, SPI, I2C, Modbus, Zigbee); Wireless embedded systems vulnerabilities (BLE, Wi-Fi); Cryptographic techniques for embedded communication; Lightweight cryptography and constraints of embedded devices; Secure communication protocols (TLS, DTLS, IPsec for embedded); Secure key management in embedded systems; Intrusion detection in embedded networks.

Activities: Design a secure communication flow diagram for an embedded protocol (CAN/Zigbee) highlighting encryption and key management.

Hardware Vulnerabilities and Countermeasures: Hardware attack models: invasive, non-invasive, semi-invasive; Side-channel attacks: power analysis, timing attacks, EM analysis; Fault injection attacks: voltage glitching, clock glitching, laser attacks; Physical tampering and reverse engineering; Hardware-based security primitives: PUFs (Physically Unclonable Functions); Trusted Platform Modules (TPMs) and secure boot; Design-for-security in hardware.

Activities: Analyze a side-channel or fault-injection attack scenario and propose suitable hardware countermeasures.

System Level Security and Case Studies: Embedded operating system - Security lifecycle management - Embedded operating system security - Secure boot and measured boot - Remote attestation; Supply chain threats and mitigation – Overview of IoT and CPS security - Case studies - IoT- Hackable Cardiac Devices, Mirai botnet, Owlet WiFi Baby Heart Monitor.

Activities: Present a case study analysis of an IoT attack (e.g., Mirai) identifying attack vectors and defense mechanisms.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. David Kleidermacher & Mike Kleidermacher, “Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development”, Elsevier / Newnes, 1st Edition, 2012.
2. Ross J. Anderson, “Security Engineering: A Guide to Building Dependable Distributed Systems”, John Wiley & Sons, 3rd Edition, 2021.

3. Debdeep Mukhopadhyay & Rajat Subhra Chakraborty, “Hardware Security: Design, Threats, and Safeguards”, CRC Press / Chapman & Hall, 1st Edition, 2014.
4. Mohammad Tehranipoor & Cliff Wang, “Introduction to Hardware Security and Trust”, Springer, 1st Edition, 2011.

	Description of CO	PO	PSO
CO1	Understand the fundamentals of embedded systems and recognize the associated security challenges	PO1(3) PO2(2) PO3(1) PO6(2)	PSO1(2) PSO2(2)
CO2	Gain practical knowledge of securing embedded firmware/software and protecting it against exploitation.	PO1(3) PO2(3) PO3(3) PO5(2)	PSO1(3) PSO2(3)
CO3	Understand how to secure data in transit and protect communication interfaces in embedded platforms.	PO1(2) PO2(3) PO3(3)	PSO1(3) PSO2(2)
CO4	Analyze threats to hardware components and understand the mechanisms to secure them.	PO1(3) PO2(3) PO3(2) PO5(2)	PSO1(3) PSO2(3)
CO5	Integrate various aspects of embedded systems security into a system-level understanding and analyze real-world attacks and defenses.	PO1(2) PO2(3) PO3(3)	PSO1(3) PSO2(3)

ET25013	ROBOTICS AND AUTOMATION	L	T	P	C
		3	0	0	3

Course objectives:

- To understand the fundamentals of robotics and automation, including robot components, actuation, sensing, and embedded systems.
- To analyze robot motion, kinematics, and path planning techniques using mathematical and computational approaches.
- To introduce Robot Operating System 2 (ROS2) concepts for modeling, simulation, and basic robot application development.

Introduction to Robotics & Automation: Overview of Robotics & Automation – Principles and Strategies of Automation System –Hardware and software for Automation- Embedded Processors for Automation-Different Types of Robots – Various Generations of Robots - Asimov's Laws of Robotics – Key components of a robot - Design Criteria for Selection of a Robot – Role of embedded system in Robotics and Automation - Recent trends.

Activities: Identify a real-world robotic system and map its components, control strategy, and application domain.

Sensors and Drive Systems: Hydraulic, Pneumatic And Electric Drive Systems – Understanding how motor power, current torque, friction co-efficient affect the design of a Robot - Determination of Motor HP and Gearing Ratio – Variable Speed Arrangements. Sensors – Classification based on sensing type (including Optical, Acoustic, Magnetic) - Proximity Sensors – Ranging Sensors – Speed & Displacement Sensing - Tactile Sensors – Vision Sensing - Smart Sensors - MEMS sensors.

Activities: Calculate motor power and gearing for a given robotic application and justify sensor selection.

Manipulators and Grippers: Introduction to Manipulators - Joints and Degrees of Freedom - Construction of Manipulators – Manipulator Dynamics and Force Control – Electronic and Pneumatic Manipulator Control Circuits – End Effectors – Various types of grippers – Design Considerations.

Activities: Analyze the degrees of freedom and gripper type for a specified industrial robot task.

Kinematics and Path Planning: Kinematic Equations – Forward and Inverse Kinematics - Solution Of Inverse Kinematics Problem – Jacobian based Velocity Kinematics– Various Path Planning Algorithms – Hill Climbing Techniques - Robot Operating System - Simulation and modeling of a simple Path Planning application. Introduction to SLAM.

Activities: Solve a simple forward/inverse kinematics problem and simulate a basic robot path.

Introduction to ROS2 Programming: Overview of Robot Operating System 2 (ROS2) and its evolution from ROS – ROS2 architecture and middleware (DDS) – Installation and workspace setup – ROS2 file system and package structure – Nodes, topics, publishers, subscribers, messages, and services – Parameters and launch files – ROS2 communication model (QoS concepts – basic overview) – Introduction to ROS2 tools (rqt, rviz2, ros2 CLI) – Simulation using Gazebo / Ignition – Developing a simple ROS2 application (publisher–subscriber example).

Activities: Demonstrate a ROS 2 publisher–subscriber example and visualize data using ROS2 tools.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Khushdeep Goyal & Deepak Bhandari, “Automation And Robotics”, S.K. Kataria & Sons, 3rd Ed., 2012
2. T. Bräunl, “Embedded Robotics: From Mobile Robots to Autonomous Vehicles with Raspberry Pi and Arduino”, Springer, 4th Ed., 2022.
3. John J. Craig, Introduction to Robotics: Mechanics and Control, Pearson, 3rd Ed.Pearson Education India, 2008.
4. Peter Corke, Robotics, Vision and Control: Fundamental Algorithms in MATLAB, Springer, 1st ed. Springer, 2011.
5. Francisco Martín Rico, A Concise Introduction to Robot Programming with ROS2, Taylor & Francis/CRC, 1st Ed., 2022.

	Description of CO	PO	PSO
CO1	Explain the principles of robotics and automation and identify suitable robots for industrial and service applications.	-	-
CO2	Analyze and select appropriate sensors, actuators, and drive systems for robotic system design.	PO1(3) PO2(3) PO3(2)	PSO1(3) PSO2(2)
CO3	Describe manipulator structures, grippers, and apply basic force and motion concepts.	PO1(3) PO2(2) PO5(1)	PSO1(3) PSO2(2)
CO4	Solve forward and inverse kinematics problems and explain path planning and SLAM concepts.	PO1(3) PO2(3) PO3(2) PO5(2)	PSO1(2) PSO2(3)
CO5	Develop and simulate a simple robotic application using ROS 2 tools and communication concepts.	PO1(2) PO2(3) PO9(3)	PSO1(3) PSO2(3)

ET25014	RECONFIGURABLE PROCESSOR AND SoC DESIGN	L	T	P	C
		3	0	0	3

Course objectives:

- To understand the fundamentals of reconfigurable processors, FPGAs, and SoC architectures.
- To analyze and design FPGA-based systems including soft-core and hard-core processors using hardware/software co-design approaches.
- To apply reconfigurable processor and SoC concepts to real-time embedded applications.

Introduction: Introduction to reconfigurable processor- Reconfigurable Computing- Programming elements and Programming Tools for Reconfigurable Processors, ASIC design flow- Hardware/Software Co-design- FPAA Architecture overview- recent trends in Reconfigurable Processor & SoC.

Activities: Analyze a reconfigurable processor and identify its programming elements and potential applications.

FPGA Technologies: FPGA Programming technology - Alternative FPGA architectures: MUX Vs LUT based logic blocks – CLB Vs LAB Vs Slices- Fast carry chains- Embedded RAMs- Routing for FPGAs- Circuits and Architectures for Low-Power FPGAs- Physical Design.

Activities: Compare LUT-based and MUX-based FPGA architectures and evaluate routing and embedded RAM usage in a given design.

FPGA Architecture: FPGA architecture overview- Challenges of FPGA processor design-Opportunities of FPGA processor design- Designing Soft Core Processors – Designing Hardcore Processors –hardware/software co-simulation- FPGA to multi core embedded computing- FPGA based on-board computer system.

Activities: Case study - Design a simple soft-core processor using FPGA simulation tools and analyze hardware/software co-design trade-offs.

Reconfigurable SoC Processors: SoC Overview –Architecture and applications of Virtex II pro ,Zynq-7000, Excalibur, Cyclone V - A7, E5- FPSLIC- Multicore SoCs.

Activities: Compare Virtex II Pro, Zynq-7000, and Cyclone V SoCs in terms of architecture and multicore capabilities.

Reconfigurable Processor and SoC Applications: Reconfigurable processor based DC motor control- digital filter design- mobile phone development- High Speed Data Acquisition -Image Processing application-controller implementation for mobile robot- Crypto-processor.

Activities: Group Discussion.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Nurmi, Jari (Ed.) "Processor Design System-On-Chip Computing for ASICs and FPGAs" Springer, 2007.
2. Ian Grout , “Digital system design with FPGAs and CPLDs” Elsevier, 2008 Joao Cardoso, Michael Hübner, "Reconfigurable Computing: From FPGAs to Hardware/Software Codesign", Springer, 2011.
3. Ron Sass and Anderew G. Schmidt, “Embedded System design with platform FPGAs: Principles and Practices”, Elsevier, 2010.
4. Steve Kilts, "Advanced FPGA Design: Architecture, Implementation, and Optimization" Willey, 2007
5. Pierre-Emmanuel Gaillardon, “Reconfigurable Logic: Architecture, Tools, and Applications”, 1st Edition, CRC Press , 2015.

	Description of CO	PO	PSO
CO1	Explain the principles, programming tools, and architectures of reconfigurable processors and FPGAs.	-	-
CO2	Describe FPGA technologies, logic blocks, routing, and low-power design techniques.	PO1(3) PO2(3) PO3(2)	PSO1(3) PSO2(2)
CO3	Analyze and design FPGA architectures, including soft-core and hard-core processor implementations.	PO1(3) PO2(3) PO5(3)	PSO1(3) PSO2(2)
CO4	Understand the architecture, features, and applications of reconfigurable SoC processors such as Zynq-7000 and Cyclone V.	PO1(3) PO2(3) PO3(2) PO5(2)	PSO1(2) PSO2(3)
CO5	Implement reconfigurable processor or SoC-based solutions for real-world applications like motor control, digital filtering, image processing, or crypto-processors.	PO1(2) PO2(3) PO3(3)	PSO1(3) PSO2(3)

ET25015	**MEMS and NEMS TECHNOLOGY**	L	T	P	C
		3	0	0	3

Course objectives:

- To provide fundamental knowledge of MEMS and NEMS devices, their principles, and practical applications.
- To develop understanding of modeling, fabrication techniques, and operation of electrostatic, piezoelectric, thermal, and magnetic sensors and actuators.
- To introduce MEMS packaging, system integration, and microfabrication methods for the development of functional MEMS/NEMS devices.

Introduction: Definition and overview of MEMS and NEMS-History and evolution of microsystems. Intrinsic characteristic of MEMS-Scaling laws in miniaturization. Semiconductor conductivity and resistivity – Stress and strain analysis – Flexural beam bending- Torsional deflection.

Activities: In-class discussion with problem-solving (e.g., calculate scaling effects, analyse mechanical stress/strain).

Microfabrication: Semiconductors, Metals, Metal Alloys, thin metal films, Polymers, Glass and Quartz substrates, diamond, piezoelectric and magnetic compounds, Microfabrication and Micromachining of Micro devices, Bulk Micromachining-Surface micro-machining-High-Aspect-Ratio (LIGA) Technology Photolithography, Etching,Sputtering, Evaporation.

Activities: Students create 3D or CAD models of a micromachining process (e.g., photolithography, LIGA).

Sensors and Actuators – I: Principle, material, design and fabrication of parallel plate capacitors as electrostatic sensors and actuators – Applications – Inter digitated Finger capacitor – Comb drive devices – Micro Grippers – Micro Motors.

Activities: Demonstrate a comb drive or interdigitated finger capacitor using cardboard, 3D printed parts, or simple circuit setups.

Sensors and Actuators – II: Piezoelectric effect – cantilever piezo electric actuator model – properties of piezo electric materials Applications. Thermal Sensing and Actuation – Magnetic Actuators – Micro magnetic components – Actuation using Shape Memory Alloys – Piezo resistive sensors.

Activities: Students present on Shape Memory Alloy actuators or magnetic MEMS devices.

Packaging: Types of packaging, Ceramic, Metal, Molded plastic, Hermetic packaging, Die-attach process, Interconnects, Encapsulation.

Activities: Groups design posters on MEMS Packaging Techniques.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Chang Liu, “Foundations of MEMS”, Pearson International Edition, 2nd Edition, 2012.
2. Stephen D. Senturia, “Microsystem Design”, Kluwer Academic Publishers, 1st Edition, 2001.
3. Marc J. Madou, “Fundamentals of Microfabrication and Nanotechnology”, 3rd Edition, CRC Press, 2011.
4. Tai-Ran Hsu,”MEMS and Microsystems: Design, Manufacture, and Nanoscale Engineering”, 2nd Edition, Wiley, 2008.

	Description of CO	PO	PSO
CO1	Understand and comprehend the functional properties of MEMS/NEMS devices..	PO1(3) PO2(3) PO3(3) PO8(2)	PSO1(2) PSO2(1)
CO2	Analyze the semiconductor materials and suitable techniques for Microfabrication of MEMS devices	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(2)
CO3	Design and analyze electrostatic based MEMS devices.	PO1(3) PO2(3) PO5(3)	PSO1(3) PSO2(2)
CO4	Design and analyze Piezoelectric and Piezoresistive based MEMS devices	PO1(3) PO2(3) PO3(3) PO5(2)	PSO1(2) PSO2(3)
CO5	Understand the various packaging technologies of MEMS/NEMS	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(3)

ET25016	EMBEDDED SYSTEM FOR BIO MEDICAL APPLICATIONS	L	T	P	C
		3	0	0	3

Course objectives:

- To understand the fundamentals of biomedical engineering, bio-signals, and biomedical system components.
- To analyze and design wearable health devices and embedded systems for medical applications.
- To apply embedded system concepts for medical image processing, diagnostic applications, and healthcare monitoring devices.

Introduction to Biomedical Engineering: Origin of bio potential and its propagation- Resting and Action Potential – Bio signals characteristics- Types of electrodes - Types of transducers and applications-Bio-amplifiers- Types of recorders- components of a biomedical system.

Activities: Observe a sample ECG/EEG signal and identify its characteristics using simulation software or recorded datasets.

Wearable Health Devices: Concepts of wearable technology in health care- Components of wearable devices- Biosensors- Blood glucose sensors - Head worn- Hand worn- Body worn-pulse oxymeter- Cardiac pacemakers – Hearing aids and its recent advancements-wearable artificial kidney.

Activities: Compare and analyze different wearable biosensors and wearable devices for monitoring vital signs.

Embedded System for Medical Image Processing: Introduction to embedded image processing . ASIC vs FPGA - memory requirement-, power consumption-parallelism - Design issues in VLSI implementation of Image processing algorithms - interfacing. Hardware implementation of image processing algorithms: Segmentation and compression.

Activities: Simulate a simple image segmentation or compression algorithm on MATLAB or FPGA tool for biomedical images.

Embedded System for Diagnostic Applications: ICCU patient monitoring system – ECG-EEG-EMG acquisition system-MRI scanner - CT scanner- Sonography.

Activities: Analyze ECG/EEG/EMG datasets to identify abnormalities and discuss system requirements for acquisition.

Case Study: Respiratory measurement using spirometer- IPPB MODULE for monitoring respiratory parameters - ventilators- -Defibrillator- Glucometer-Heart-Lung machine.

Activities: Examine a ventilator or glucometer system and map its embedded system components, sensors, and data flow.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Leslie Cromwell, “Biomedical Instrumentation and Measurement”, Prentice Hall of India, New Delhi, 2007.
2. John G.Webster, “Medical Instrumentation Application and Design”, 3rd Edition, Wiley India Edition, 2007
3. Khandpur R.S, Handbook of Biomedical Instrumentation, Tata McGraw Hill, New Delhi, 3rd Edition, 2014.
4. L.A Geddes and L.E.Baker, Principles of Applied Biomedical Instrumentation, 3rd Edition, John Wiley and Sons, Reprint 2008.
5. Richard S.Cobbold, Transducers for Biomedical Measurements; Principle and applications- John Wiley and sons, 1992.

	Description of CO	PO	PSO
CO1	Explain bio-potentials, signal characteristics, transducers, electrodes, and biomedical system components.	-	-
CO2	Describe the principles, components, and functioning of wearable health devices including pacemakers, pulse oximeters, and biosensors.	PO1(3) PO2(3) PO3(2)	PSO1(3) PSO2(2)
CO3	Analyze embedded systems for medical image processing, hardware implementation of algorithms, and VLSI design issues.	PO1(3) PO2(3) PO5(3)	PSO1(3) PSO2(2)
CO4	Demonstrate embedded systems applications in diagnostic devices such as ECG, EEG, EMG, MRI, CT, and sonography.	PO1(3) PO2(3) PO3(2) PO5(2)	PSO1(2) PSO2(3)
CO5	Develop conceptual designs or analyze case studies for real-life biomedical devices including ventilators, glucometers, defibrillators, and heart-lung machines.	PO1(2) PO2(3) PO3(3) PO4(3)	PSO1(3) PSO2(3)

PX25C02	Electric Vehicles and Power Management	L	T	P	C
		3	0	0	3

Course Objective:

This course aims to equip students with a comprehensive understanding of electric vehicle (EV) architecture, propulsion systems, and energy storage technologies. It focuses on power management strategies that optimize performance, efficiency, and battery life under varying load and driving conditions. Students will also explore smart charging, regenerative braking, grid integration, and the role of power electronics in advancing sustainable transportation systems.

Introduction to Electric Vehicles (EVs)

EV evolution and classification (BEV, HEV, PHEV, FCEV) - EV configurations and architecture -Vehicle dynamics: tractive effort, energy consumption, gradability - Driving cycles: EPA, NEDC, WLTP, and custom profiles

Activities:

Quiz on BEV, HEV, PHEV, FCEV features

Team design activity: Design its architecture (battery, motor, converter, controller layout)

Electric Propulsion and Drive Systems

Types of electric motors used in EVs: PMSM, BLDC, IM, SRM - Motor characteristics and drive selection - Motor control strategies: Field-Oriented Control (FOC), Direct Torque Control (DTC) - Regenerative braking – Torque Vs. Speed Control

Activities:

Group Discussion/Debate: Types of electric motors used in EVs (Pros/cons in EVs ,Torque-speed curve sketch ,strengths/limitations and real-world use of each motor.)

Simulation: Run and observe simulations for **Field-Oriented Control (FOC) vs. Direct Torque Control (DTC)**

Energy Storage Systems and Battery Management

Battery types: Li-ion, NiMH, Lead-acid – comparison and characteristics - Battery modeling and performance parameters - Battery Management Systems (BMS): SoC, SoH, cell balancing - Supercapacitors and hybrid storage systems

Activities:

Technical seminar: Compare Li-ion, NiMH, and Lead-acid batteries based on key parameters

Modelling: Estimate required battery size, discharge rate, and range for an EV

Power Electronics and Energy Conversion

DC-DC converters, inverters, and onboard chargers - Bidirectional power flow and energy recuperation - Charging infrastructure: AC vs DC, wireless charging, V2G systems - Safety, thermal management, and EMI issues

Activities:

Simulation: Simulation or flowchart design during drive and regenerative braking (e.g., motor ↔ battery ↔ grid)

Group Discussion/Debate: Compare AC charging vs DC fast charging (e.g., speed, efficiency, cost, grid impact)

Power Management Strategies and Future Trends

Power flow optimization and energy scheduling in EVs - Smart energy management using AI and predictive algorithms - Integration with smart grids and renewable sources - Trends in EV policy, standardization, and sustainable mobility

Activities:

Case Study: Analyze and optimize power flow between battery, motor, regenerative braking, and auxiliary systems during different drive conditions

Team design activity: Propose an EV charging solution using solar/wind, including storage and grid backup.

Assessment Weightage: Internal Assessment 1 – 15%; Internal Assessment 2 – 15%; Digital Assignments/Simulations (minimum 2) – 10%; Final Assessment – 60%

References:

1. "Advanced Electric Drive Vehicles", Authors: Ali Emadi, MehrdadEhsani, John M. Miller, Publisher:CRC Press, Published Year:2014
2. "Modern Electric Vehicle Technology", Authors: C.C. Chan, K.T. Chau, Publisher: Oxford University Press, Published Year: 2001
3. "Battery Management Systems for Large Lithium-Ion Battery Packs", Author: Philip Weicker, Publisher: SAE International, Published Year:2013
4. "Power Electronics for Electric Vehicles and Energy Storage Systems", Author:Shuai Jiang, Publisher: Springer, Published Year:2022
5. "Electric and Hybrid Vehicles: Technologies, Modeling and Control – A Mechatronics Approach", Authors: Amir Khajepour, M. SaberFallah, AvestaGoodarzi, Publisher:Wiley, Published Year:2014

CO	CO Description	PO	PSO1	PSO2
CO1	Understand the architecture of EV powertrains and identify the role of various subsystems in power and energy flow management.	PO1 (2) PO2 (2) PO3 (3)	3	3
CO2	Analyze battery systems, charging infrastructure, and battery management techniques to ensure safety, efficiency, and longevity of EVs.	PO1 (3) PO2 (1) PO3 (3)	3	3
CO3	Apply power electronics converters and control strategies for energy conversion, bidirectional charging, and regenerative braking in electric vehicles.	PO1 (3) PO2 (2) PO3 (3)	3	2
CO4	Evaluate real-time power management strategies under different driving conditions using performance metrics such as efficiency, thermal stability, and cost-effectiveness.	PO1 (3) PO2 (2) PO3 (3)	3	3
CO5	Design intelligent energy management systems using rule-based, fuzzy logic, and optimization-based algorithms for hybrid and battery electric vehicles.	PO1 (2) PO2 (2) PO3 (3)	3	3

ET25017	EDGE DATA ANALYTICS	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce edge computing paradigms, architectures, and hardware platforms for real-time and latency-sensitive applications.
- To understand data acquisition, preprocessing, and protocol handling in edge-based IoT systems.
- To explore lightweight machine learning models and frameworks for efficient edge deployment.
- To study edge analytics architectures using microservices, containers, and hybrid processing strategies.
- To address security, privacy, and performance optimization in distributed edge computing environments.

Fundamentals of Edge Computing and Analytics: Introduction to edge computing paradigms - fog computing, mobile edge computing (MEC), and cloudlet architectures. Edge vs cloud vs hybrid computing models. Edge computing hardware platforms: ARM processors, NVIDIA Jetson, Intel NUC, and FPGA-based solutions. Edge device capabilities and constraints. Real-time requirements and latency-sensitive applications in IoT ecosystems.

Activities: Compare different edge computing platforms (ARM, NVIDIA Jetson, Intel NUC, FPGA) and discuss their suitability for real-time IoT applications.

Edge Data Collection and Preprocessing: Sensor networks and IoT data acquisition at the edge. Data streaming protocols: MQTT, CoAP, and WebSocket. Edge-based data filtering, aggregation, and compression techniques. Data quality assessment and cleaning at source. Time-series data handling and synchronization. Edge gateways and protocol translation. Data standardization and interoperability challenges.

Activities: Simulate sensor data acquisition and perform basic edge-level filtering, aggregation, and time-series synchronization.

Machine Learning at the Edge: Lightweight machine learning algorithms for edge deployment. Model optimization techniques: quantization, pruning, and knowledge distillation. TensorFlow Lite, ONNX Runtime, and edge-specific ML frameworks. Real-time inference and model serving at edge nodes. Federated learning principles and edge-based collaborative learning. Adaptive algorithms for dynamic edge environments.

Activities: Deploy a lightweight ML model using TensorFlow Lite or ONNX Runtime on an edge device and evaluate inference latency.

Edge Analytics Architectures and Frameworks: Stream processing frameworks for edge: Apache Kafka Streams, Eclipse IoT, and Azure IoT Edge. Microservices architecture for edge applications. Container orchestration with Docker and Kubernetes at the edge. Event-driven architectures and serverless computing at edge nodes. Edge-cloud hybrid processing patterns and workload distribution strategies.

Activities: Design a microservices-based edge application using Docker/Kubernetes and demonstrate simple event-driven data processing.

Security, Privacy, and Performance Optimization: Security challenges in distributed edge environments. Encryption and secure communication protocols for edge devices. Privacy-preserving analytics and differential privacy techniques. Access control and device authentication in edge networks. Performance monitoring and optimization of edge analytics pipelines. Resource management and energy efficiency in edge computing. Edge device lifecycle management and over-the-air updates.

Activities: Analyze security and privacy requirements for an edge analytics pipeline and propose strategies for secure communication and energy-efficient operation.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Javid Taheri and Shuiguang Deng,"Edge Computing: Models, Technologies and Applications", IET Digital Library, 2021.
2. Rajkumar Buyya and Satish Narayana Srirama "Fog and Edge Computing: Principles and Paradigms", John Wiley & Sons, 2019.
3. K. Anitha Kumari, G. Sudha Sadasivam, D. Dharani, and M. Niranjanamurthy, "Edge Computing: Fundamentals, Advances and Applications", CRC Press, 2022.
4. Daniel Situnayake and Jenny Plunkett,"AI at the Edge: Solving Real-World Problems with Embedded Machine Learning", O'Reilly Media, 2023.
5. Javid Taheri, Schahram Dustdar, Albert Zomaya, and Shuiguang Deng, "Edge Intelligence: From Theory to Practice", Springer, 2023.

	Description of CO	PO	PSO
CO1	Understand and design edge computing systems for real-time IoT and embedded applications.	PO1(3) PO2(3) PO3(3) PO5(3)	PSO1(2) PSO2(2)
CO2	Collect, preprocess, and stream data efficiently on resource-limited edge devices.	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(2)
CO3	Deploy and optimize lightweight machine learning models on edge hardware.	PO1(3) PO2(2) PO3(3)	PSO1(3) PSO2(2)
CO4	Build secure and privacy-aware edge analytics solutions with proper authentication.	PO1(3) PO2(2) PO3(2) PO5(2)	PSO1(2) PSO2(3)
CO5	Assess performance and design scalable hybrid edge-cloud analytics architectures.	PO1(2) PO2(3) PO3(3) PO4(3)	PSO1(3) PSO2(3)

ET25C03	PYTHON PROGRAMMING FOR MACHINE LEARNING	L	T	P	C
		3	0	0	3

Course objectives:

- To understand and apply fundamental Python programming concepts including control structures, functions, recursion, and basic data structures.
- To develop skills in handling data using lists, dictionaries, files, and images, and in implementing programs using Python.
- To introduce machine learning concepts and enable students to formulate and implement basic machine learning solutions using Python through practice and exercises.

Introduction to Machine Learning and Python: Introduction to Machine Learning: Significance, Advantage and Applications – Categories of Machine Learning – Basic Steps in Machine Learning: Raw Data Collection, Pre-processing, Training a Model, Evaluation of Model, Performance Improvement

Introduction to Python and its significance – Difference between C, C++ and Python Languages; Compiler and Interpreters – Python3 Installation & Running – Basics of Python Programming Syntax: Variable Types, Basic Operators, Reading Input from User – Arrays/List, Dictionary and Set – Conditional Statements – Control Flow and loop control statements

Activities: Write a simple Python program and map the steps of a machine learning workflow using a real-world example.
Python Functions and Packages: File Handling: Reading and Writing Data – Errors and Exceptions Handling – Functions & Modules – Package Handling in Python – Pip Installation & Exploring Functions in python package – Installing the Numpy Library and exploring various operations on Arrays: Indexing, Slicing, Multi-Dimensional Arrays, Joining Numpy Arrays, Array intersection and Difference, Saving and Loading Numpy Arrays – Introduction to SciPy Package & its functions - Introduction to Object Oriented Programming with Python **Activities:** Perform file handling and array operations using NumPy and SciPy on a sample dataset.
Implementation of Machine Learning using Python: Description of Standard Datasets: Coco, ImageNet, MNIST (Handwritten Digits) Dataset, Boston Housing Dataset – Introducing the concepts of Regression – Linear, Polynomial & Logistic Regression with analytical understanding - Introduction to SciPy Package & its functions – Python Application of Linear Regression and Polynomial Regression using SciPy – Interpolation, Overfitting and Underfitting concepts & examples using SciPy. **Activities:** Implement and compare linear and polynomial regression models using a standard dataset.
Classification and Clustering Concepts: Introduction to ML Concepts of Clustering and Classification – Types of Classification Algorithms – Support Vector Machines (SVM) - Decision Tree - Random Forest – Introduction to ML using scikit-learn – Using scikit-learn, loading a sample dataset, Learning & prediction, interpolation & fitting, Multiclass fitting - Implementation of SVM using Blood Cancer Dataset, Decision Tree using data from csv. Types of Clustering Algorithms & Techniques – K-means Algorithm, Mean Shift Algorithm & Hierarchical Clustering Algorithm – Introduction to Python Visualization using Matplotlib: Plotting 2- dimensional, 3-dimensional graphs; formatting axis values; plotting multiple rows of data in same graph – Implementation of K-means Algorithm and Mean Shift Algorithm using Python. **Activities:** Train a classification model and perform clustering on a dataset, then visualize the results using Matplotlib.
Introduction to Neural Networks and Embedded Machine Learning: Introduction to Neural Networks & Significance – Neural Network Architecture – Single Layer Perceptron & Multi-Layer Perceptron (MLP) – Commonly Used Activation Functions - Forward Propagation, Back Propagation, and Epochs – Gradient Descent – Introduction to Tensorflow and Keras ML Python packages –

Implementation of MLP Neural Network on Iris Dataset – Introduction to Convolution Neural Networks – Implementation of Digit Classification using MNIST Dataset

ML for Embedded Systems: Comparison with conventional ML – Challenges & Methods for Overcoming – TinyML and Tensorflow Lite for Microcontrollers – on-Board AI – ML Edge Devices: Arduino Nano BLE Sense, Google Edge TPU and Intel Movidius.

Activities: Build a simple neural network for digit classification and discuss its deployment on an edge device.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. Mark Lutz, ”Learning Python, Powerful OOPs,O’reilly,2011
2. Zelle, John “M. Python Programming: An Introduction to Computer Science.”, Franklin Beedle& Associates, 2003
3. Andreas C. Müller, Sarah Guido, “Introduction to Machine Learning with Python”, O'Reilly,2016
4. Sebastian Raschka , VahidMirjalili, “Python Machine Learning - Third Edition”, Packt, December 2019.

	Description of CO	PO	PSO
CO1	Explain basic machine learning concepts, workflows, and implement fundamental Python programs.	-	-
CO2	Use Python functions, packages, and numerical libraries to process and analyze data.	PO1(3) PO2(2) PO3(2)	PSO1(3) PSO2(2)
CO3	Implement regression models and analyze overfitting and underfitting using standard datasets.	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(2)
CO4	Apply classification and clustering algorithms using scikit-learn and visualize results.	PO1(3) PO2(3) PO3(3) PO5(3)	PSO1(2) PSO2(3)
CO5	Develop basic neural network models and understand embedded machine learning and TinyML concepts.	PO1(2) PO2(2) PO3(3) PO4(2)	PSO1(3) PSO2(3)

ET25018	EMBEDDED DEVICE DRIVER PROGRAMMING	L	T	P	C
		3	0	0	3

Course objectives:

- To introduce Embedded C programming and the architecture of Embedded Linux systems.
- To understand and develop Linux device drivers, including peripheral interfaces and memory management techniques.
- To explore interrupt handling, real-time concepts, and synchronization mechanisms in embedded Linux and RTOS environments.

Introduction to Embedded C and Embedded Linux: Embedded C concepts-pointers-bitwise operations- Embedded Linux versus Desktop Linux- Architecture of Embedded Linux, Linux Kernel Architecture, Linux StartUp Sequence, GNU Cross-Platform Tool chain.

Activities: Analyze the Embedded Linux boot sequence and toolchain flow using a block diagram.

Basic Device Drivers: The role of device drivers in interacting with hardware. Types of Device Drivers: Character drivers, block drivers, and their characteristics. Boot loader, Driver concepts -Block & character driver distinction -Low level drivers, OS drivers etc -Writing character drivers.

Activities: Write and compile a simple Linux character device driver and observe its operation.

Embedded Device Driver: Linux Serial Driver, Ethernet Driver, I2C subsystem on Linux, USB Gadgets, Watchdog Timer, and Kernel Modules.

Activities: Study and modify an existing Linux driver (serial/I2C/USB) to understand data flow.

Memory Management and Peripheral Drivers: Dynamic memory allocation, memory mapping, and memory protection -Specific Driver Examples: Writing drivers for common peripherals like UART, timers, and storage devices - USB Device Basics - Writing a USB Driver.

Activities: Implement a simple peripheral driver using dynamic memory allocation and memory mapping.

Interrupts and Time Delays: Interrupt Handling: Interrupt service routines (ISRs), and interrupt handling mechanisms. -Writing interrupt driven drivers, Implementing bottom halves-Kernel Threads & Work Queues-Timers, Kernel timers, Jiffies , Timer interrupts- Debugging using printing, querying, watching and system defaults-Debugging tools.

Activities: Develop an interrupt-driven driver and analyze ISR and bottom-half execution.

Real Time OS and Drivers: RTOS concepts, task management, scheduling, synchronization, and interrupt handling - Interfaces to driver read, write, ioctl etc-Blocking and non-blocking calls, Synchronisation -Semaphores , mutexes ,spinlocks Proc & Sysfs interfaces.

Activities: Demonstrate task synchronization using semaphores or mutexes in a real-time driver scenario.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Quiz (10%), Assignments (40%), Internal Examinations (50%)

References:

1. John Madieu, "Linux Device Drivers Development: Develop customized drivers for embedded Linux", Packt Publishing, 1st Edition, 2017
2. Mohan Lal Jangir, Linux Kernel and Device Driver Programming, 2014, 1st Edition, University Science Press, India
3. Embedded Linux System Design and Development, P. Raghavan, Amol Lad, Sriram Neelakandan, Auerbach Publication, 2006.
4. Michael J. Pont, “Embedded C”, Pearson Education, 1st Edition, 2007
5. Christopher Hallinan, "Embedded Linux Primer: A practical Real-World approach", Prentice Hall, 2nd Edition, 2011.
6. Jonathan Corbet, Alessandro Rubini, Greg Kroah, "Linux Device Drivers", O'Reilly, 3rd Edition, 2005.

	Description of CO	PO	PSO
CO1	Explain Embedded C concepts and the architecture and boot process of Embedded Linux systems.	-	-
CO2	Understand and develop basic Linux device drivers, including character and block drivers.	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(2)
CO3	Implement Linux drivers for common interfaces such as serial, Ethernet, I2C, USB, and watchdog timers.	PO1(3) PO2(3) PO3(3)	PSO1(3) PSO2(2)
CO4	Apply memory management techniques and develop peripheral drivers with proper resource handling.	PO1(3) PO2(3) PO3(3) PO5(3)	PSO1(2) PSO2(3)
CO5	Develop interrupt-driven and real-time capable drivers using RTOS concepts and synchronization mechanisms.	PO1(3) PO2(3) PO3(3) PO4(2)	PSO1(2) PSO2(2)