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.		Real Time Operating System	T	3	1	0	4	4	ES (PC)
2.		Embedded System Networking	T	3	0	0	3	3	ES (PC)
3.		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.		Embedded System Laboratory – I	L	0	0	4	4	2	ES (PC)
7.		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.		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.		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.		Wireless And Mobile Communication	3	0	0	3	3
2.		Data Driven Embedded Systems	3	0	0	3	3
3.		Advanced Embedded Processor	3	0	0	3	3
4.		DSP Based System Design	3	0	0	3	3
5.		Automotive Embedded System	3	0	0	3	3
6.		Embedded Linux	3	0	0	3	3
7.		Autonomous Vehicle	3	0	0	3	3
8.		Computer Vision	3	0	0	3	3
9.		Digital Twin	3	0	0	3	3
10.		Machine Learning and Deep Learning	3	0	0	3	3
11.		Wireless Sensor Networks	3	0	0	3	3
12.		Embedded Computing	3	0	0	3	3
13.		Embedded Systems Security	3	0	0	3	3
14.		Robotics and Automation	3	0	0	3	3
15.		Reconfigurable Processor and SoC Design	3	0	0	3	3
16.		MEMS and NEMS Technology	3	0	0	3	3
17.		Embedded System for Bio Medical Applications	3	0	0	3	3
18.		Electric vehicle and power management.	3	0	0	3	3
19.		Edge Data Analytics	3	0	0	3	3
20.		Python Programming for Machine Learning	3	0	0	3	3
21.		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)