Pimpri Chinchwad Education Trust's

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

SECTOR NO. 26, PRADHIKARAN, NIGDI, PUNE 411044

An Autonomous Institute Approved by AICTE and Affiliated to SPPU, Pune

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Curriculum Structure and Syllabus

Honors in

Embedded Programming and Automotive Communication Networks

(Regulations 2023)

Effective from Academic Year 2026-27
(Offered under PCCoE and KPIT MoU)

Institute Vision

To be one of the top 100 Engineering Institutes of India in coming five years by offering exemplarily Ethical, Sustainable and Value-Added Quality Education through a matching ecosystem for building successful careers.

Institute Mission

1. Serving the needs of the society at large through establishment of a state-of-art Engineering Institute.
2. Imparting right Attitude, Skills, Knowledge for self-sustenance through Quality Education.
3. Creating globally competent and Sensible engineers, researchers and entrepreneurs with an ability to think and act independently in demanding situations.

EOMS Policy

“We at PCCOE are committed to offer exemplarily Ethical, Sustainable and Value Added Quality Education to satisfy the applicable requirements, needs and expectations of the Students and Stakeholders.

We shall strive for technical development of students by creating globally competent and sensible engineers, researchers and entrepreneurs through Quality Education.

We are committed for Institute’s social responsibilities and managing Intellectual property.

We shall achieve this by establishing and strengthening state-of-the-art Engineering Institute through continual improvement in effective implementation of Educational Organizations Management Systems (EOMS).”

Course Approval Summary

Board of Studies - Department of E&TC Engineering

Sr. No.	Name of the Course	Course Code	Page number	Signature and stamp of BoS chairman
1	Foundations of Computer Systems	BET25HN31	10	
2	Foundations of Computer Systems Lab	BET25HN32	12	
3	Embedded Programming	BET25HN33	13	
4	Embedded Programming Lab	BET25HN34	15	
5	Embedded Linux	BET26HN35	17	
6	Embedded Linux Lab	BET26HN36	19	
7	Automotive Communication Networks	BET27HN37	21	
8	Automotive Communication Networks Lab	BET27HN38	22	

Approved by Academic Council:

Chairman, Academic Council
Pimpri Chinchwad College of Engineering, Pune

INDEX

-

Preface

With the increasing integration of intelligent systems across industries, embedded systems and hardware–software co-design has emerged as critical pillars of modern technological innovation. From electric vehicles and aerospace systems to consumer electronics and industrial automation, embedded technology enables high-performance, safety-critical, and energy-efficient solutions. The rapid adoption of AI, IoT, real-time data processing, edge computing, and autonomous mobility has further accelerated the need for skilled engineers capable of designing optimized systems at both hardware and software levels.

To support industry demand and foster domain-focused academic specialization, this Honors program has been designed to provide students an opportunity to develop advanced, application-oriented competencies in Embedded Systems, Embedded Linux, Real-Time Systems, and Automotive Communication Networks. The curriculum emphasizes hands-on learning, industry-grade tooling, open-source technologies, and project-based skill development to prepare students for high-impact roles in automotive, and embedded system companies.

This Honors program offers an academic pathway to complement core engineering studies with specialized expertise, bridging the gap between academic foundations and industry-ready skills in the rapidly evolving embedded systems ecosystem.

Objectives of Honors Degree

1. To enable students to build specialized knowledge in embedded systems and hardware–software integration aligned with contemporary industry needs.
2. To provide hands-on experience using real-world development tools, platforms, and methodologies used in automotive and embedded industries.
3. To enhance employability through a combination of domain-specific competencies such as microcontroller programming, embedded Linux, RTOS, and automotive networking.
4. To develop practical engineering skills through project-based learning, system-level design, debugging, and performance optimization.
5. To create a strong academic and research foundation for students aspiring to pursue advanced studies or careers in embedded systems, automotive technologies, or related interdisciplinary domains.

Embedded Programming and Automotive Communication Networks

Objectives:

1. Introduce students to the architecture, tools, and development workflows used in embedded systems.
2. Develop proficiency in programming microcontrollers and configuring peripheral interfaces such as GPIO, UART, SPI, I2C, ADC, and timers.
3. Provide knowledge of Embedded Linux, including kernel, bootloaders, device drivers, and Yocto-based system customization.
4. Enable understanding of automotive communication protocols such as CAN, LIN, and Automotive Ethernet, along with system-level diagnostics.
5. Foster skills in real-time system design, task scheduling, interrupts, and multithreading using RTOS environments.
6. Encourage industry-oriented project development integrating hardware, firmware, operating systems, and communication networks.

Outcomes: After completion of this program, students will be able to:

1. Design and develop embedded applications using microcontroller peripherals with efficient and optimized C/C++ implementations.
2. Configure and deploy Embedded Linux systems, including kernel modules, device drivers, and Yocto-based builds on target hardware.
3. Implement real-time applications using RTOS concepts such as task scheduling, synchronization, interrupts, and IPC mechanisms.
4. Build and evaluate automotive communication systems using CAN, LIN, and Ethernet with proper debugging, testing, and fault handling.
5. Integrate hardware and software components to design end-to-end embedded solutions for automotive and digital systems.
6. Use industry-relevant tools such as GDB, CMake, Google Test, Yocto, QEMU, and Python-based automotive scripting environments for development and validation.

Curriculum Structure

Curriculum structure

			Credit Scheme				Teaching Scheme(Hours/Week)			Evaluation Scheme and Marks						
Course Code	**Semester**	**Course Name**	**L**	**P**	**T**	**Total**	**L**	**P**	**T**	**FA 1**	**FA 2**	**SA**	**TW**	**PR**	**OR**	**Total**
BET25HN 31	V	Foundations of Computer Systems	3			3	3			20	20	60				100
BET25HN 32	V	Foundations of Computer Systems Lab		2		2		4					25		25	50
BET25HN 33	V	Embedded Programming	3			3	3			20	20	60				100
BET25HN 34	V	Embedded Programming Lab		2		2		4					25		25	50
BET26HN 35	VI	Embedded Linux	3			3	3			20	20	60				100
BET26HN 36	VI	Embedded Linux Lab		2		2		4					25		25	50
BET27HN 37	VII	Automotive Communication Networks	3			3	3			20	20	60				100
BET27HN 38	VII	Automotive Communication Networks Lab		2		2		4					25		25	50
Total			**12**	**8**	**0**	**20**	**12**	**16**	**0**	**80**	**80**	**240**	**100**	**0**	**100**	**600**

Abbreviations:
1 Lecture hour = 1 Credit 2 Lab Hours = 1 Credit 1 Tutorial Hour = 1 Credit Abbreviations are: L-Lecture, P-Practical, T-Tutorial, H- Hours, FA-Formative Assessment, SA-Summative assessment TW –Term work, OR – Oral, CR- Credits

Final year B.Tech. schemes will not be applicable to this Honors Course.

Course Syllabus Semester - V

Program:	B. Tech. (E&TC-Honors)			Semester:		V	
Course:	Foundations of Computer Systems			Code:		BET25HN31	
	Teaching Scheme (Hrs./Week)			Evaluation Scheme and Marks			
Credit	Lecture	Practical	Practical	FA		SA	Total
				FA1	FA2		
3	3	-	-	20	20	60	100

Prior Knowledge of:
Basic understanding of C/C++ programming, operating system concepts is essential.

Course Objectives:

1. Explore tools and techniques of writing efficient and optimized codes.
2. Understand computer organization and architecture.
3. Describe the principles of computer architecture.
4. Understand the fundamentals of system software and role of operating system in achieving parallel processing.
5. Understand the ways of Power and Performance Optimization in a computer system

Course Outcomes:
After the completion of the course, the students will be able to:

1. Evaluate code efficiency and apply optimization strategies using GNU toolchain.
2. Design software workflows using SDLC and testing frameworks for quality assurance.
3. Analyze CPU–memory interactions to assess system performance.
4. Compare CISC, RISC, ARM architectures to justify architecture selection.
5. Examine OS-level resource management to evaluate system performance.

Detailed Syllabus:

Unit	Description	Duration
1.	**Programming Fundamentals and Tools** Introduction to GNU Toolchain (GCC, GDB, Makefiles), Concepts of C and C++ programming, coding standards and code analysis, writing efficient and optimized codes, use of GenAI tools for writing, debugging and testing codes.	9
2.	**Software Engineering** Introduction, Software Development Life Cycle, Requirement gathering and analysis, Use Case Modeling. Software Testing and Quality Assurance – levels of testing, testing techniques.	9
3.	**Introduction to Computer Organization** Overview of computer organization, Brief history of computer development; Hardware Components - CPU: structure, function, and types, Memory - main memory, virtual memory, caching, I/O devices - peripherals, interfaces; Memory Management – memory organization, Virtual memory - concept, advantages, and disadvantages, Caching - concept, types, and cache hierarchies; Input/Output Systems – peripherals and Interfaces.	9
4	**Introduction to Computer Architecture** Overview and history of computer architecture, CISC, RISC, and ARM Architectures, Bus Architecture - Bus types: system bus, I/O bus, Bus protocols - synchronous and asynchronous buses; Parallel Processing and Multi-Core Architecture.	9
5	**System Software** Introduction to Operating System - Operating System Concept, Roles, and Operations, process, thread, and memory management, process scheduling, memory allocation, and I/O management, booting, login, and system calls, types of operating system.	9
	Total	**45**

Text Books:

1. Harris, S., & Harris, D. (2021). Digital Design and Computer Architecture: RISC-V Edition. Morgan Kaufmann.
2. Patterson, D. A., & Hennessy, J. L. (2017). Computer Organization and Design RISC-V Edition: The Hardware/Software Interface (5th Ed.). Morgan Kaufmann.
3. Bryant, R. E., & O'Hallaron, D. R. (2016). Computer Systems: A Programmer's Perspective (3rd Ed.). Pearson. School of Computing Science

Reference Books:

1. Kernighan, B. W., & Ritchie, D. M. (1988). The C programming language. prentice-Hall.
2. Deitel, P., & Deitel, H. (2016). C++ how to Program. Pearson.
3. Stallman, R., Pesch, R., & Shebs, S. (1988). Debugging with GDB. Free Software Foundation.
4. Bast, R., & Di Remigio, R. (2018). CMake Cookbook: Building, testing, and packaging modular software with modern CMake. Packt Publishing Ltd.
5. Hayes, J. P. (2022). Computer architecture and organisation. Tata McGraw Hill.
6. Deitel, H. M., Deitel, P., & Choffnes, D. R. (2003). Operating systems. Prentice Hall.

Program:		B. Tech. (E&TC-Honors)		Semester: V			
Course:		Foundations of Computer Systems Lab		Code: BET25HN32			
	Teaching Scheme (Hrs./Week)			Evaluation Scheme and Marks			
Credit	Lecture	Practical	Tutorial	TW	OR	PR	Total
02	--	04	-	25	25	--	50

Prior knowledge of:
C/C++ programming skills, familiarity with Linux command-line tools, and basic debugging concepts are essential.

Course Objectives:

1. Apply coding guidelines and best practices to write highquality code.
2. Use GDB to identify and fix errors in C/C++ programs
3. Understand the basics of CMake and its applications in building C/C++ projects.
4. Use the Google Test framework to write and execute unit tests for C/C++ code.
5. Utilize Generative AI tools to improve code writing, debugging and testing efficiency.
6. Explore simulation tools to model and analyze computer systems and architectures.

Course Outcomes:
At the end of Laboratory work, the students will be able to:

1. Apply coding standards and optimize implementations using static analysis.
2. Diagnose program failures using GDB and justify debugging choices.
3. Construct modular build systems using CMake for scalable software.
4. Develop automated unit tests and evaluate correctness using GoogleTest.
5. Simulate multi-core/memory systems and construct a performance-optimized project.

Detailed Syllabus:

Expt. No.	List of Experiments
1	Writing C/C++ programs considering coding guidelines and code analysis. Use of Generative AI tools for efficient code writing.
2	Debugging code with GDB. Use of GenAI tools for efficient code debugging.
3	Understanding CMake for cross-platform project configuration and builds.
4	Performing unit testing with Google Test framework. Use of GenAI tools for testing.
5	Simulate CPU execution, memory management, multi core task execution and bus communication using tools like Multi2Sim etc.
6	Development of Minor project in group adhering to the learned concepts.

Reference Books:

1. Kernighan, B. W., & Ritchie, D. M. (1988). The C programming language. prentice-Hall.
2. Deitel, P., & Deitel, H. (2016). C++ how to Program. Pearson.
3. Stallman, R., Pesch, R., & Shebs, S. (1988). Debugging with GDB. Free Software Foundation.
4. Bast, R., & Di Remigio, R. (2018). CMake Cookbook: Building, testing, and packaging modular software with modern CMake. Packt Publishing Ltd.
5. GoogleTest User's Guide: https://google.github.io/googletest/
6. CMake: A Powerful Software Build System: https://cmake.org/
7. Multi2Sim A Heterogeneous System Simulator: http://www.multi2sim.org/

Program:	B. Tech. (E&TC-Honors)			Semester:	V		
Course:	Embedded Programming			Code:	BET25HN33		
	Teaching Scheme (Hrs./Week)			Evaluation Scheme and Marks			
Credit	Lecture	Practical	Practical	FA		SA	Total
				FA1	FA2		
3	3	-	-	20	20	60	100

Prior Knowledge of:
microcontroller basics, digital circuits, and C programming fundamentals is essential.

Course Objectives:

1. Understand the fundamentals of embedded systems and interpret technical documents.
2. Understand and explain the build and development tools used in embedded systems design.
3. Understand the function and program the microcontroller peripherals.
4. Write interrupt service routines while using microcontroller peripherals.
5. Understand RTOS internals and performance metrics.

Course Outcomes:

After the completion of the course, the students will be able to:

1. Analyze system constraints to choose suitable embedded architecture.
2. Construct build pipelines and evaluate memory mapping decisions.
3. Develop peripheral-based applications interfacing with I/O, timers, ADC, UART, SPI, I2C.
4. Design interrupt-driven workflows and optimize ISR latency.
5. Implement multitasking systems and evaluate real-time scheduling performance.

Detailed Syllabus:

Unit	Description	Duration
1.	**Introduction to Embedded Systems** What are Embedded Systems and characteristics, building blocks of Embedded Systems, various applications of Embedded Systems, Difference between Desktop Vs Embedded application development process, role of microprocessor and microcontroller in Embedded System, Understanding Block Diagram, Schematic Diagram, Datasheet, User manual, Application Notes	9
2.	**Embedded Systems Development Tool Chain** Build and development tools, Build Process, role of Linker, understanding object files, Elf Generation / Compilation/ Linker Error Resolution, tools used to extract binary code from elf, converting to different formats like plain binary, S-record, Concept of Make file and Complex Make file, Big Endian vs Little Endian, Memory mapping, understanding Startup code, Reset Code, Bootloader	9
3.	**Microcontroller Peripherals and Programming** Understanding and programming STM32 microcontroller for GPIO, UART, Timer, PWM, ADC, UART, I2C and SPI Protocol	9
4.	**Microcontroller Peripheral Interrupts and Programming** Interrupt Vector Table, Interrupt Priorities, Nested Interrupts, Writing ISR in assembly, and Embedded C, interrupt latency, Write interrupt service routines for various peripherals and operations	9
5.	**Real Time Operating System** Introduction to Real-Time Concepts, RTOS Internals & Real Time Scheduling, Performance Metrics of RTOS, Task Specifications, Schedulability Analysis, Application Programming on RTOS, Configuring and Porting of RTOS	9
	Total	**45**

Text Books:

1. Mazidi, M. A., McKinlay, R., & Causey, D. (2017). The STM32 Arm Programming for Embedded Systems: Using C and STM32F4xx (1st Ed.). Pearson.

2. Noviello, C. (2019). Mastering STM32 – Second Edition. Leanpub.
3. Prasad, K. V. K. (2003). Embedded/Real-Time Systems: Concepts, Design and Programming Black Book. Dreamtech Press.

Reference Books:

1. STM32F401 - PDF Documentation: https://www.st.com/en/microcontrollersmicroprocessors/stm32f401/documentation.html
2. FreeRTOS Documentation: https://www.freertos.org/Documentation/00-Overview

Program:		B. Tech. (E&TC-Honors)		Semester: V			
Course:		Embedded Programming Lab		Code: BET25HN34			
	Teaching Scheme (Hrs./Week)			Evaluation Scheme and Marks			
Credit	Lecture	Practical	Tutorial	TW	OR	PR	Total
02	--	04	-	25	25	--	50

Prior knowledge of:

Hands-on experience in writing and compiling C code, using embedded toolchains, and reading datasheets is essential.

Objectives:

1. Interpret and Analyze Technical Documents.
2. Utilize build and development tools and processes to design, develop, and implement projects using the STM32 microcontroller.
3. Write efficient C code to program and interface with various STM32 peripherals, including GPIO, UART, Timer, PWM, ADC, I2C, and SPI protocols.
4. Develop interrupt service routines to handle peripheral operations and events, ensuring efficient and reliable system performance.
5. Design and implement multitasking operations using the FreeRTOS.

Outcomes:

At the end of Laboratory work, the students will be able to:

1. Interpret schematics and datasheets to validate system design decisions.
2. Configure cross-compilation toolchains and debug firmware on target boards.
3. Construct driver-level programs interacting with MCU peripherals.
4. Develop ISR-enabled applications and diagnose interrupt-related faults.
5. Create FreeRTOS-based applications using tasks, queues, and semaphores.

Detailed Syllabus:

Expt. No.	List of Experiments
1	Interpret and understand various technical documents, including Block Diagrams, Schematic Diagrams, Datasheets, User Manuals, and Application Notes related to the STM32.
2	Understanding build and development tools and processes involved in working with the STM32.
3	Program the STM32 microcontroller to utilize its peripherals, such as GPIO, UART, Timer, PWM, ADC, I2C, and SPI protocols.
4	Write interrupt service routines to handle peripheral operations and events.
5	Perform multitasking operations using FreeRTOS, by creating tasks and using inter process communication and synchronization mechanisms.
6	Development of Minor project in group adhering to the learned concepts.

Reference Books:

1. STM32F401 - PDF Documentation: https://www.st.com/en/microcontrollersmicroprocessors/stm32f401/documentation.html
2. FreeRTOS Documentation: https://www.freertos.org/Documentation/00-Overview

Course Syllabus Semester - VI

Program:	B. Tech. (E&TC-Honors)					Semester:	VI
Course:	Embedded Linux					Code:	BET26HN35
	Teaching Scheme (Hrs./Week)			Evaluation Scheme and Marks			
Credit	Lecture	Practical	Practical	FA		SA	Total
				FA1	FA2		
3	3	-	-	20	20	60	100

Prior Knowledge of:
OS fundamentals, basic Linux commands, and C programming is essential.

Course Objectives:

1. Understand Linux OS Fundamental.
2. Design and Develop POSIX Thread-Based Applications.
3. Understand the principles and terminologies of Embedded Linux based on Yocto.
4. Apply the BSP philosophy to configure and develop Yocto Project for application development.
5. Develop and Debug Embedded Linux Device Drivers.

Course Outcomes:

After the completion of the course, the students will be able to:

1. Analyze kernel subsystems to determine resource allocation strategies.
2. Develop multithreaded applications and evaluate synchronization efficiency.
3. Configure Yocto environment and justify build customizations.
4. Construct board-specific Linux images and validate bootloader configurations.
5. Develop kernel modules and debug drivers for GPIO, I2C, SPI subsystems.

Detailed Syllabus:

Unit	Description	Duration
1.	**Introduction to Linux** Introduction to Embedded Operating Systems and Linux, Bootloader, Kernel, Root File System, Process Management, Inter-process Communication & Synchronization, Memory Management, I/O subsystem.	9
2.	**POSIX Thread Programming** POSIX Thread Programming, POSIX Semaphores, Mutexes, Conditional Variables, Barriers, Message Queues, Shared Memory, Debugging and Testing of Multi-threaded Applications, Socket Programming.	9
3.	**Embedded Linux based on Yocto** Introduction to the Yocto Project, setting and configuration of build environment, Yocto Project architecture and components, OpenEmbedded Build System, BitBake Build Engine, Poky	9
4.	**Board Support Packages and Application Development** BSP Philosophy, building with a BSP, Prebuilt Binaries and configuration files, Recipe Files, creating a Yocto Project BSP, Creating Bootable Media Images. Setting Up a Yocto Project ADT, Preparing for Application Development and building an application and launching with QEMU.	9
5.	**Embedded Device Drivers** The Embedded Linux Software Eco-System, Linux Kernel Modules and Module Programming, Char Device Drivers, Kernel Internals: Dynamic memory allocations, Handling Delays, Timers, Synchronization, Locking, I/O Memory and Ports, Interrupts, Deferred Executions, Driver De-bugging Techniques, Drivers for GPIO, I2C, and SPI, Pseudo Filesystems (procfs, sysfs).	9
	Total	**45**

Text Books:

1. Hallinan, C. (2011). Embedded Linux Primer: A Practical Real-World Approach. Pearson Education India.
2. Streif, R. J. (2016). Embedded Linux Systems with the Yocto Project. Prentice Hall Press.
3. Kroah-Hartman, G., & Rubini, A. (2005). Linux Device Drivers (3rd Ed.). O'Reilly Media.

Reference Books:

1. Yocto Project Documentation – (OpenEmbedded).
2. POSIX Thread Programming texts – e.g., “Programming with POSIX Threads” by Nichols, Buttlar & Pekarik.
3. Linux Kernel Module Programming Guide.

Program:		B. Tech. (E&TC-Honors)		Semester: VI			
Course:		Embedded Linux Lab		Code: BET26HN36			
	Teaching Scheme (Hrs./Week)			Evaluation Scheme and Marks			
Credit	Lecture	Practical	Tutorial	TW	OR	PR	Total
02	--	04	-	25	25	--	50

Prior knowledge of:
Experience with Linux shell, compilation of C programs, and basic knowledge of hardware interfaces is essential.

Objectives:

1. Demonstrate Proficiency in understanding and applying Linux Commands.
2. Design and Implement POSIX Thread-Based Applications.
3. Configure and Customize Embedded Linux using Yocto.
4. Develop, Build, and Deploy embedded Linux applications on Embedded Hardware.
5. Understand the process of developing character and GPIO drivers.

Outcomes:
At the end of Laboratory work, the students will be able to:

1. Execute system-level operations and analyze runtime behavior using shell tools.
2. Build multi-threaded applications and resolve concurrency bottlenecks.
3. Configure, compile, and deploy custom Yocto images on target hardware.
4. Launch applications on QEMU/BeagleBone and validate kernel boot logs.
5. Develop character drivers and integrate them into a complete embedded project.

Detailed Syllabus:

Expt. No.	List of Experiments
1	Gain a comprehensive understanding of various Linux commands and their applications in different scenarios.
2	Learn to harness the power of POSIX threads and Inter-Process Communication (IPC) mechanisms to demonstrate multithreaded operations, ensuring efficient and synchronized execution of tasks
3	Set up and configure Embedded Linux using the Yocto Project.
4	Develop, build, and launch Embedded Linux applications on QEMU and BeagleBone Hardware modules, showcasing the integration of software and hardware components in embedded systems.
5	Leveraging kernel internals create character device drivers and develop drivers for GPIO.
6	Development of Minor project in group adhering to the learned concepts.

Reference Books:

1. Hallinan, C. (2011). Embedded Linux primer: a practical real-world approach. Pearson Education India.
2. Streif, R. J. (2016). Embedded Linux Systems with the Yocto Project. Prentice Hall Press.
3. Drivers, L. D. (3). Linux Device Drivers. Greg Kroah Hartman, Alessandro Rubini.

Course Syllabus Semester - VII

Program:	B. Tech. (E&TC-Honors)			Semester:		VII	
Course:	Automotive Communication Networks			Code:		BET27HN37	
	Teaching Scheme (Hrs./Week)			Evaluation Scheme and Marks			
Credit	Lecture	Practical	Practical	FA		SA	Total
				FA1	FA2		
3	3	-	-	20	20	60	100

Prior Knowledge of:
Basics of embedded systems, digital communication concepts, and microcontroller interfacing are essential.

Course Objectives:

1. Explain the importance and evolution of automotive networks.
2. Understand the Controller Area Network (CAN), Local Interconnect Network (LIN) protocols and their role in automotive networks.
3. Understand the role of the internet protocol and ethernet in advanced automotive systems.
4. Understand the role of Diagnostic over IP and SOME/IP in optimizing performance in automotive systems.

Course Outcomes:

After the completion of the course, the students will be able to:

1. Analyze automotive network topologies to justify protocol selection.
2. Develop and validate CAN-based communication considering error handling.
3. Construct LIN networks and evaluate behavior under fault scenarios.
4. Compare automotive Ethernet standards to optimize bandwidth and latency.
5. Implement TCP/UDP/SOME/IP communication flows and analyze diagnostics.

Detailed Syllabus:

Unit	Description	Duration
1.	**Introduction to Automotive Protocols** Overview of automotive networks and their importance, History and evolution of automotive networks, Types of automotive networks - CAN, LIN, FlexRay, MOST, Ethernet, Network topology and architecture, Introduction to OSI layers.	9
2.	**Controller Area Network** Introduction to CAN protocol, CANFD, CAN in automotive systems, CAN frame structure and message formatting, CAN bus topology and wiring, CAN error handling and fault tolerance. Introduction to CANXL. Introduction to CAN Database with DBC file.	9
3.	**Local Interconnect Network** Introduction to LIN protocol, LIN applications in automotive systems, LIN frame structure and message formatting, LIN bus topology and wiring, handling and fault tolerance.	9
4.	**Ethernet in Automotive Systems** Introduction to Ethernet protocol, Ethernet applications in automotive systems, Ethernet network topology, IEEE 100 BASE T1, IEEE 1000 BASE T1, IEEE 10 BASE T1S, Ethernet Frame, VLAN.	9
5.	**Internet Protocols and Ethernet Applications** Internet Protocol basics and properties, IPv4, IPv6, network and host address, TCP and UDP, Diagnostic over IP, SOME/IP.	9
	Total	**45**

Text Books:

1. Matheus, K., & Königseder, T. (2021). Automotive Ethernet. Cambridge University Press.
2. The “Automotive Ethernet – The Definitive Guide” (2nd Edition). Emerald / Elsevier.
3. “Vehicle Networking for Technicians: A Practical Guide to CAN, LIN, FlexRay and Ethernet Systems.” (2023) – TechPress.

Reference Books:

1. CAN Protocols: https://www.bosch-semiconductors.com/products/ipmodules/can-protocols/
2. CAN Bus Explained https://www.csselectronics.com/pages/can-bussimple-intro-tutorial.
3. LIN standards and specifications: https://www.lin-cia.org/standards/

Program:		B. Tech. (E&TC-Honors)		Semester: VII			
Course:		Automotive Communication Networks Lab		Code: BET27HN38			
	Teaching Scheme (Hrs./Week)			Evaluation Scheme and Marks			
Credit	Lecture	Practical	Tutorial	TW	OR	PR	Total
02	--	04	-	25	25	--	50

Prior knowledge of:
microcontroller programming, and Linux-based tools is essential.

Objectives:

1. Design and implement a CAN-based system.
2. Implement a LIN-based system.
3. Integrate CAN communication with Enhanced Ethernet Switch.
4. Use Python scripts for CAN communication in ANDi.
5. Send and receive SOME/IP messages using Media converter.

Outcomes: At the end of Laboratory work, the students will be able to:

1. Implement CAN nodes and analyze protocol-level error responses.
2. Build LIN-based communication and validate frame scheduling.
3. Integrate CAN through Ethernet gateway and evaluate network throughput.
4. Automate CAN interaction using Python scripts for validation.
5. Send/receive SOME/IP messages and develop integrated automotive network project.

Detailed Syllabus:

Expt. No.	List of Experiments
1	Implement a CAN-based system. Investigate the effects of errors and faults on the CAN network and implement fault-tolerance mechanisms.
2	Implement a LIN-based system.
3	CAN communication using Enhance Ethernet Switch.
4	CAN communication using Python Script in ANDi.
5	Sending and Receiving TCP/UDP/ SOME/IP messages using Media converter.
6	Development of Minor project in group adhering to the learned concepts.

Reference Books:

1. CAN Protocols: https://www.bosch-semiconductors.com/products/ipmodules/can-protocols/
2. LIN standards and specifications: https://www.lin-cia.org/standards/
3. Technica Hardware and Software Document:
4. ANDi Reference Document:
5. TI flyer: https://www.ti.com/lit/pdf/slyt560
6. ArXiv paper: “A Simulation Environment and preliminary evaluation for Automotive CAN-Ethernet AVB Networks.” https://arxiv.org/abs/1409.0998

Vision and Mission of E&TC Department

VISION : To be recognized as a distinguished department in the field of electronics and telecommunication transforming students into competent technocrats by providing an Ethical, Sustainable and Value-Added Quality Education.

MISSION :

1. To create competent Electronics and Tele-communication engineers with Knowledge, Skill and Attitude by establishing a conducive learning environment.
2. To nurture technical competency, entrepreneurship skills and promote higher studies through the state-of-art facilities for building successful careers.
3. To facilitate research by engaging in projects of industrial requirement and national importance.
4. To impart Life skills, Ethical and Social values for self-sustainability.

Programme Educational Objectives (PEO's)

1. Establish a strong base in mathematics, basic sciences, and the fundamental principles of Electronics and Telecommunication Engineering for the students.
2. Equip students with a comprehensive understanding of Electronics and Telecommunication Engineering, enabling them to effectively comprehend, analyse, design, and to innovate practical solutions for real-world challenges.
3. Foster the development of effective communication skills, teamwork, and professional ethics among students, in order to meet the demands of employers and prepare them for higher studies and successful careers.
4. Promote social consciousness and a sense of responsibility among students, creating awareness about their commitment and obligations to society.

Program Outcomes (PO's)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics,natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply thesssse to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

1. **PSO1:** Ability to exhibit the competency to solve the problems related to Electronics & Telecommunications Engineering by applying advanced knowledge in the fields of VLSI, Embedded Systems, Signal Processing, Communication, Computing and Automation.
2. **PSO2:** Ability to design and analyse Electronics & Telecommunications systems using state of the art hardware and software tools to address the needs of the industry and society.
3. **PSO3:** Ability to build research and problem-solving attitude through Project based learning to address the societal, environmental, health and safety issues.