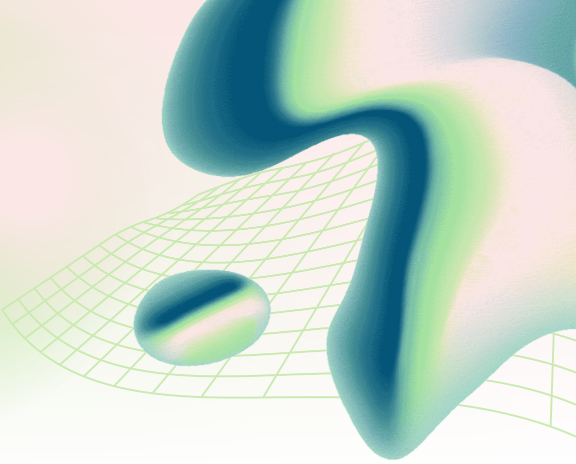
# MASTERING EMBEDDED SYSTEMS

A 5-Month Journey with C, C++, Linux, and Raspberry Pi



# Embedded Systems Development

**Become an Embedded Systems Expert in Just 5 Months!**

Embark on a hands-on journey to master C Programming, C++, System Programming, Linux Internals, Device Drivers, and more with real-world projects. Whether you're aiming to work in embedded software development or want to learn how to interact with hardware systems at a low level, this course will guide you step-by-step.

## About Spectrum Technologies

### Empowering Careers Through Quality Technical Education

Spectrum Technologies is a premier technical training institute dedicated to bridging the gap between academic learning and industry requirements. Since our inception, we have been committed to providing world-class training in cutting-edge technologies, with a special focus on Embedded Systems, IoT, Linux, and Full-Stack Development.

## Course Highlights

- **24-Week Duration** - Comprehensive learning journey
- **Intermediate Level**- Perfect for aspiring embedded engineers
- **95% Completion Rate**- Proven track record of success

## What Makes This Course Unique

- End-to-End Embedded Linux Coverage
- Strong Hardware–Software Integratio
- Kernel-Level Depth with Practical Focus
- Industry-Oriented, Project-Based Learning
- Interview & Placement Readiness

## Course Outcomes

- Strong Foundation in C and C++ Programming
- Readiness for Professional Roles
- Efficient Linux System-Level Application Development
- Kernel Analysis and Interview Preparation
- Design and Development of Kernel Modules and Device Drivers
- Interfacing Embedded Peripherals
- Practical Skills for Embedded Product Development
- Building Linux Systems
- Proficiency in Linux Build Systems
- Placement Readiness

## Career Opportunities

- Embedded Software Engineer
- Firmware Developer
- IoT Engineer
- Linux Device Driver Developer
- Embedded Linux Engineer
- System Software Engineer
- Hardware Design Engineer
- Embedded Systems Architect
- Automotive Embedded Engineer
- BSP (Board Support Package) Developer

# Detailed Curriculum

## Module 1: C Programming ( 4 Weeks )

**Build a Rock-Solid Foundation in C Programming – The Backbone of Embedded Systems Development**

**Key Topics:**

- **Fundamentals**: C history, data types, operators, control statements
- **Advanced Concepts**: Pointers, multi-level pointers, function pointers
- **Data Structures**: Arrays, strings, structures, unions
- **Memory Management**: malloc, calloc, realloc, free
- **File Operations**: Binary files, file I/O
- **Preprocessor**: Macros, header files, conditional compilation

- **Best Practices**: Debugging with GDB, command-line arguments, volatile/const keywords

**What You'll Build:**

- **Mini Project**: Complete C-based application (e.g., student management system)
- **Assignments & Exercises** to practice key concepts

**Tools:**

- GCC Compiler, VS Code, GDB Debugger, Makefile, Valgrind

## Module 2: C++ Programming ( 3 Weeks )

**Master Object-Oriented Programming & Modern C++ Features**

**Key Topics:**

- **OOP Fundamentals**: Classes, objects, inheritance, polymorphism
- **Modern Features**: Templates, exception handling, lambda expressions
- **STL Mastery**: Vectors, maps, sets, iterators
- **Memory**: Smart pointers (unique_ptr, shared_ptr), RAII principles
- **Design Patterns**: Common patterns in embedded systems

**What You'll Build:**

- **Project 1**: Template-based calculator
- **Project 2**: Real-time data processing application using STL

**Tools:**

- G++ Compiler, VS Code, CMake, GDB

## Module 3: System Programming in Linux Using C ( 3 Weeks )

**Develop Robust System-Level Applications Using Linux APIs**

**Key Topics:**

- **Linux Architecture**: System calls, kernel vs. user space
- **File I/O**: System calls (open, close, read, write)
- **Process Management**: fork(), exec(), waitpid()
- **Signals & Timers**: POSIX timers, signal handling
- **IPC Mechanisms**: Shared memory, semaphores, message queues
- **Multithreading**: POSIX threads, mutexes, condition variables
- **Network Programming**: Socket programming, TCP/UDP

**What You'll Build:**

- **Project 1**: TCP client-server application
- **Project 2**: Multi-threaded producer-consumer application

**Tools:**

- GCC, GDB, Valgrind, strace, ltrace

## Module 4: Linux Internals and Interfacing ( 2 Weeks )

**Deep Dive into Linux Kernel Architecture**

**Key Topics:**

- **Kernel Architecture**: Task struct, kernel components
- **Memory Management**: Paging, swapping, slab allocators
- **Interrupts & Concurrency**: Hardware interrupts, softirq
- **Debugging**: printk, ftrace, kernel panic analysis

**What You'll Do:**

- **Practical Exercises**: Analyze /proc and /sys files
- **Kernel Debugging**: Learn techniques for analyzing kernel issues

**Tools:**

- Linux kernel source, dmesg, strace, ftrace

## Module 5: Linux Device Driver Programming ( 3 Weeks )

**Write Professional Linux Kernel Modules and Device Drivers**

**Key Topics:**

- **Kernel Modules**: Module structure, file operations, device registration
- **Character & Block Drivers**: Request queue, cdev, bio structure
- **GPIO Framework**: GPIO descriptor interface, interrupts
- **I2C & SPI Drivers**: Subsystems, device tree bindings
- **Interrupt Handling**: request_irq(), tasklets, workqueues

**What You'll Build:**

- **Project 1**: Character device driver
- **Project 2**: GPIO LED driver with interrupts
- **Project 3**: I2C sensor driver for temperature monitoring

**Tools:**

- Linux kernel headers, GCC, Make/Kbuild, insmod/rmmod

## Module 6: Device Driver Programming with RPi-4 (2 Weeks)

**Apply Driver Development on Real Raspberry Pi Hardware**

**Key Topics:**

- **Raspberry Pi Setup**: Architecture, GPIO layout, hardware components
- **GPIO Drivers**: LED blink driver, button input with interrupts
- **I2C & SPI**: Sensors, EEPROM interfacing, display drivers
- **Interrupt Handling**: GPIO interrupt configuration
- **Timers**: Kernel timers, periodic tasks

**What You'll Build:**

- **Project 1**: Multi-peripheral driver system (LED + button + sensor)
- **Project 2**: System monitoring application

**Hardware:**

- Raspberry Pi 4, sensors, breadboard, jumper wires

**Tools:**

- Raspberry Pi OS, cross-toolchain, dtc, i2c tools, spi tools

## Module 7: Digital System Peripherals and Interfacing ( 2 Weeks )

**Master Communication Protocols Used in Embedded Systems**

**Key Topics:**

- **I2C, SPI, UART**: Protocols, master/slave, addressing, multi-slave config
- **CAN Bus**: Automotive applications, error detection, SocketCAN
- **USB & PCIe**: Device classes, configuration space, enumeration

**What You'll Build:**

- **Project**: Multi-protocol communication system using I2C, SPI, and UART

## Module 8: Embedded Device Driver with RPi-4 ( 3 Weeks )

**Build Production-Grade Embedded Systems with Professional Driver Integration**

**Key Topics:**

- **Advanced Module Development**: Sysfs attributes, DMA, error handling
- **Power Management**: Runtime PM, CPU frequency scaling
- **Performance Optimization**: Zero-copy techniques, buffer management
- **Testing & Validation**: Unit testing, integration testing

**What You'll Build:**

- **Capstone Project**: Multi-interface embedded driver system with real-time data acquisition

**Hardware:**

- Raspberry Pi kit, sensors, peripherals

## Module 9: Linux Build System and Toolchains ( 1 Week )

**Master the Tools and Techniques for Building Embedded Linux Systems**

**Key Topics:**

- **Build Systems**: Makefiles, GCC flags, cross-compilation
- **Toolchains**: GCC, binutils, glibc, kernel headers
- **Automation**: Autoconf, Automake, CMake

**What You'll Build:**

- **Project 1**: Cross-compile and deploy applications to Raspberry Pi
- **Project 2**: Build custom minimal root filesystem

**Tools:**

- GCC toolchains, Make, CMake, Busybox, Buildroot

## Module 10: Yocto Linux System ( 2 Weeks )

**Create Custom Embedded Linux Distributions Using Yocto**

**Key Topics:**

- **Yocto Fundamentals**: Architecture, OpenEmbedded, BitBake
- **Image Creation**: Custom images, rootfs, kernel customization
- **Advanced Features**: SDK generation, CI/CD integration

**What You'll Build:**

- **Final Project**: Custom Yocto Linux distribution for Raspberry Pi with device drivers, applications, and minimal/full images

**Tools:**

- Yocto Project, BitBake, devtool, Toaster, QEMU

---

## Why Choose This Course?

- **Hands-On Learning**: Work with real hardware (Raspberry Pi, sensors, peripherals).
- **Industry-Relevant Skills**: Learn embedded systems development, Linux internals, and device driver programming.
- **Expert Instructors**: Learn from professionals with years of experience in embedded systems.
- **Capstone Project**: Build a complete embedded system with a real-time application, multi-protocol communication, and web-based monitoring.

**Ready to get started? Enroll today and build the embedded systems of tomorrow!**