VAct Technologies
innovate your thoughts...

Course Title: "QNX Neutrino RTOS"

1. Course Summary: QNX training provides comprehensive instruction on the real-time operating system, equipping individuals with essential skills for developing and maintaining embedded systems and applications. It covers topics such as system architecture, software development, and

troubleshooting, enabling professionals to excel in the field of embedded systems development.

2. Pre-Requisite: Embedded systems and real-time operating systems (RTOS), along with basic programming skills in languages like C and C++.

3. Target Audience: Entry Level and Mid-Level

4. Software, Hardware & Network Requirements:

- 8GB or 16GB of RAM preferably Linux systems.
- QEMU, QNX SDP 8.0, QNX Suite of tools, QNX Momentics IDE
- RapberryPi 4b+ or above (1 each for 2 partcipants), Monitors with HDMI cables, USB keyboards, USB Mouse
- Raspberry Pi Imager or any other programmer with all permissions to read, write
- USB to TTL Serial Adapter (Optional)

6. Learning Objectives:

At the end of the training, the participants will be able to: They will have gained comprehensive knowledge and practical skills in QNX real-time operating systems, enabling them to develop and manage embedded systems with efficiency and precision

Course Content with Lab/Hands-on Activity (day wise break-up):

Day 1 - Introduction to Real-Time Programming, Understanding Real-Time Systems

- Real-time vs. General Purpose Systems: Discuss the fundamental differences between real-time systems, which are designed to respond to events within strict time constraints, and general-purpose systems.
- Importance of Determinism and Predictability: Explain why determinism and predictability are critical in real-time systems, as they ensure that tasks are completed within specified timeframes

Introduction to QNX

- History and Background: Explore the history of QNX as an operating system and its evolution over time.
- Key Features and Advantages: Highlight the features that make QNX suitable for real-time applications, such as its microkernel architecture and reliability.
- Supported Hardware Platforms: List the hardware platforms and devices that are compatible with QNX.

Real-Time Application Development Life Cycle

- Requirements Analysis: Discuss the initial phase of software development, where the specific requirements and constraints of the real-time application are identified.
- System Design: Explain how to design a system architecture that meets real-time requirements, including task scheduling and resource allocation.
- Implementation: Detail the process of coding and building the real-time application on the QNX platform.
- Testing and Validation: Discuss testing methodologies and validation techniques to ensure that the application meets its real-time performance goals.
- Maintenance: Highlight the ongoing maintenance and updates required for real-time systems to remain reliable and efficient.

Day 2 QNX Fundamentals

QNX Architecture

- Microkernel Architecture: Describe the microkernel architecture of QNX, emphasizing its benefits for real-time systems.

- Process and Thread Management: Explain how QNX manages processes and threads efficiently.

- Interprocess Communication (IPC): Detail the IPC mechanisms in QNX for communication between processes and threads.

QNX Neutrino RTOS

- Kernel Services: Explore the core services provided by the QNX Neutrino real-time operating system, including memory management and resource allocation.

- File Systems: Discuss the file systems supported by QNX and their usage in real-time applications.

- Memory Management: Explain how QNX manages memory, including virtual memory and real-time memory allocation.

- Networking Stack: Overview the networking capabilities of QNX and their relevance to real-time applications.

QNX Development Tools

- QNX Momentics IDE: Introduce the integrated development environment used for QNX application development.

- Compilers and Cross-Compilers: Discuss the compilers available for QNX and their role in cross-compilation.

- Debugging Tools: Explore debugging tools such as GDB, DDD, and the QNX System Profiler for diagnosing issues in real-time applications.

Day 3 - Real-Time Application Development on QNX

Building QNX Real-Time Applications (3hrs)

- Creating a QNX Project: Explain the steps to set up a QNX project, including configuring build settings.
- Writing and Compiling Code: Guide students on writing code for real-time applications and compiling it for deployment.
- Deploying Applications: Describe the process of deploying real-time applications on QNX.

Multi-Threaded Programming on QNX (3hrs)

- Creating and Managing Threads: Teach how to create and manage threads in a multi-threaded QNX application.
- Synchronization Mechanisms: Explain the use of mutexes and semaphores for thread synchronization.
- Thread Priority and Scheduling: Detail how to set thread priorities and manage scheduling for real-time performance.

Inter-process Communication (IPC) (2hrs)

- Message Passing: Demonstrate how to implement message passing between processes and threads.
- Shared Memory: Explore the use of shared memory for efficient data sharing in real-time applications.
- Sockets and Networking: Discuss socket-based communication and networking in QNX.

Day 4 - Debugging and Analysis Tools

Debugging Techniques (3hrs)

- Using GDB for QNX: Provide in-depth guidance on using the GNU Debugger for QNX application debugging.
- Tracing and Logging: Explain how to implement tracing and logging in real-time applications for debugging purposes.
- Remote Debugging: Show how to set up and use remote debugging for QNX applications.

QNX System Profiler (2hrs)

- Profiling Real-Time Applications: Describe how to use the QNX System Profiler for profiling and analyzing real-time applications.
- Analyzing CPU and Memory Usage: Teach students how to analyze CPU and memory usage patterns.
- Identifying Performance Bottlenecks: Explain techniques for identifying and addressing performance bottlenecks in real-time systems.

Memory Analysis (3hrs)

- Memory Leak Detection: Discuss methods for detecting memory leaks in real-time applications.
- Dynamic Memory Allocation Profiling: Explore tools and techniques for profiling dynamic memory allocation.

Day 5 - Real-Time Performance Optimization

Profiling and Optimization (2hrs)

- Identifying Performance Hotspots: Show how to identify areas of code that impact performance.
- Code Profiling Tools: Introduce various code profiling tools available for QNX.
- Optimization Strategies: Discuss strategies for optimizing real-time performance, including algorithm optimization and resource management.

Real-Time Safety and Reliability (2hrs)

- Error Handling and Recovery: Explain the importance of error handling mechanisms for maintaining system reliability.

- Redundancy and Failover Mechanisms: Discuss redundancy and failover strategies to ensure continuous operation in critical real-time systems.

Day 6 Capstone Projects

Project 1: Real-Time Data Acquisition and Processing (3hrs)

- Description: Students develop a real-time data acquisition system on QNX, focusing on efficient data transfer and processing using IPC. Performance monitoring and optimization techniques are applied.

Project 2: QNX-Based Embedded System (2hrs)

- Description: Students design an embedded system using QNX for a specific application (e.g., IoT device). Emphasis is placed on optimizing resource usage and real-time performance, with remote debugging and monitoring capabilities.

Project 3: Real-Time Control System (3hrs)

- Description: Students create a real-time control system using QNX, implementing thread synchronization and prioritization for precise control. They analyze system performance and optimize for minimal latency.