Embedded Systems

Course Overview:

dridhOn one of the best embedded training institutes in Bangalore will make you an eminent embedded systems engineer and makes you work in top companies. This embedded systems training will make your work in real-world projects and makes you an expert in all three components of the embedded system such as hardware, application software, real-time operating system (RTOS). We will award you embedded system course certification after completion of the course that makes you work in large organizations in the world.

Training Features:

- 64 hours of blended learning
- 64 hours of Online self-paced learning 16 hours of instructor-led training
- Five lesson-end knowledge checks
- 1 real-life course-end project 20+ assisted practices on all modules
- Industry-recognized course completion certificate

Delivery Mode:

- Online Live Virtual Instructor Led Training

Target Audiance:

dridhOn embedded system training joined by,

- IT professionals
- Fresh graduates who are interested build their career in embedded system can take this course
- Degree pursuing individuals can take this embedded systems course

Key Learning Outcomes:

Skills covered in embedded system course,

- Introduction to embedded system and applications
- Interfacing of tools and peripherals
- Learning of 8051 microcontroller system
- I/O programming
- C – The dominant language among embedded systems
- Simulation of an embedded system using Keil
- Source control using online repositories
- Creating your own Real Time Operating System (RTOS)

Certification Details:

- Complete at least 85 percent of the course or attend one complete batch
- Successful completion and evaluation of the project

dridhOn

GETTING STARTED:

- What is C?
- Data Types
- Variables
- Naming Conventions for C Variables
- Printing and Initializing Variables

CONTROL FLOW CONSTRUCTS:

- if
- if else
- while
- for
- Endless Loops
- do while
- break and continue
- switch
- else if

THE C PRE PROCESSOR:

- #define
- Macros
- #include
- Conditional Compilation
- #ifdef
- #ifndef

MORE ON FUNCTIONS:

- Function Declarations
- Function Prototypes
- Returning a Value or Not
- Arguments and Parameters
- Organization of C Source Files
- Extended Example

BIT MANIPULATION:

- Defining the Problem Space
- A Programming Example
- Bit Wise Operators
- Bit Manipulation Functions
- Circular Shifts

STRINGS & ARRAY:

- Fundamental Concepts
- Aggregate Operations
- String Functions
- Array Dimensions
- An Array as an Argument to a Function
- String Arrays
- Example Programs

POINTERS (PART 1):

- Fundamental Concepts
- Pointer Operators and Operations
- Changing an Argument with a Function Call
- Pointer Arithmetic
- Array Traversal
- String Functions with Pointers
- Pointer Difference
- Prototypes for String Parameters
- Relationship Between an Array and a Pointer
- The Pointer Notation *p++

POINTERS (PART 2):

- Dynamic Storage Allocation -malloc
- Functions Returning a Pointer
- Initialization of Pointers
- gets -a Function Returning a Pointer
- An Array of Character Pointers
- Two Dimensional Arrays vs. Array of Pointers
- Command Line Arguments
- Pointers to Pointers
- Practice with Pointers
- Function Pointers

STRUCTURES:

- Fundamental Concepts
- Describing a Structure
- Creating Structures
- Operations on Structures
- Functions Returning Structures
- Passing Structures to Functions
- Pointers to Structures
- Array of Structures
- Functions Returning a Pointer to a Structure
- Structure Padding

STRUCTURE RELATED ITEMS (UNION):

- typedef-New Name for an Existing Type
- Bit Fields
- unions
- Non-HomogeneousArrays
- Enumerations

FILE I/O:

- System Calls vs. Library Calls
- Opening Disk Files
- fopen
- I/O Library Functions
- Copying a File
- Character Input vs. Line Input
- scanf
- printf
- fclose
- Servicing Errors -errno.h
- Feofo

ridhOn

SCOPE OF VARIABLES:

- Block Scope
- Function Scope
- File Scope
- Program Scope
- The auto Specifier
- The static Specifier
- The register Specifier
- The extern Specifier
- The Const Modifier
- The Volatile Modifier

INTRODUCTION TO EMBEDDED:

- What is Embedded Systems?
- Difference b/w Micro processor & Micro
- Controller
- CISC Vs RISC
- Architecture of8,16,32-bitProcessor
- Software Used, Compilation, Debugging
- Example Programs (LCD, RELAY, STEPPER MOTOR)
- Embedded Software Life Cycle TestingSPI, ADC,
- Serial Communication, Protocols (I2C , CAN, Ethernet)

Linux OS Architecture:

- Linux Features
- Linux Kernel Source Directory Structure
- Linux Kernel Components
- User Mode Vs Kernel Mode
- System Initialization –Booting Process

Introduction to Linux tools, compilers and utilities:

- Introduction To Makefile
- How to write Makefile to compile programs on Linux
- Building static and dynamic libraries
- LABs

Kernel compilation:

- Importance of Makefiles
- Procedure to recompile the kernel
- LAB

Detail study of Linux OS components:

- Process Management
- Process Control Block (PCB)
- Types Of Processes
- States Of Process
- How to Create Process?
- Process Scheduling
- LABs

Thread Management:

- What is Thread?
- Thread Control Block (TCB)
- User-level Vs Kernel level Threads
- How to create and cancel threads?
- Thread Scheduling
- Process Vs Threads
- LABs

Interrupt Management:

- What are interrupts?
- Types Of Interrupt
- Interrupt Handling
- Interrupt Service Routine (ISR)
- Interrupt Latency

Signal handling

- What are signals in Linux OS?
- Signal Implementation
- Signal Handling
- LABs

Inter-Process communication (IPCs):

- Introduction To Inter-process communication mechanism
- Pipes, Message Queue and Shared Memory
- Semaphores and Mutex
- LABs

Socket programming on Linux:

- TCP/IP and UDP socket programming
- LABs

Linux Kernel and Device driver Programming:

- Linux File system and System call interface
- Introduction To System Call Mechanism
- Significance Of System Calls
- LABs

Memory Management Unit:

- Segmentation and Paging
- Swapping and demand to page
- malloc(), kmalloc() and free()

Linux Kernel and Device driver Programming:

- Module Basics
- Introduction to Modules
- Writing Your first kernel module
- Statically linked vs Dynamically linked
- Exporting symbols from modules
- The kernel symbol table
- Concurrency in the kernel
- Module Parameters
- Version dependency
- LABs

An introduction to device drivers:

- Role of the Device Drivers
- Role of Virtual file system
- Classes of devices
- Registering a character device driver
- File operations and ioctls
- Reading and writing into char devices
- LABs

Block Device Drivers:

- Registering block driver
- File operations and ioctls
- Handling requests
- Write RAM type of disk driver
- LABs

Network Device Drivers:

- The net_device structure in detail
- Packet transmission
- Packet reception
- Simulating a network device
- LABs

Embedded Linux On ARM9:

- Intro to the target board (Samsung mini2440– ARM 9)
- Introduction to ARM9 architecture

Installation of cross compilation tool chain:

- Installing Sources, Patching
- Installing the GCC toolchain
- Set Cross Compiling Environment
- LABs

Porting Linux on ARM9:

- Recompilation and flash Kernel on ARM9 board
- LABs

The Root Filesystem:

- Busybox
- A Small Application example
- Flashing the new root filesystem
- Compilation procedure
- LABs

Write device driver for ARM9 board:

- Services on Board
- Compiling and setting up services
- An example of service – LABS

Structure and implementation of open-source RTOS:

- RTOS Source Organization
- Configuration Of RTOS
- Implementation Of RTOS

Port RTOS on ARM Board:

- Steps for porting RTOS On ARM7
- LAB(Demonstration)- Port Open Source RTOS On ARM Board

Real-Time Operating Fundamentals:

- Task Management
- Multitasking
- Context Switching
- Inter-Process/Task Communication (IPC)
- LABs (Demonstration)

References and Guideline for Linux based embedded system Skills developed after completion of course:

- Learn about Key principles of Linux OS
- Expertise on a device driver for the target board
- Porting Linux on advanced cross-platform i.e. ARM 9
- Get good expertise on Linux based embedded system