Companies - Looking for trained engineers?

EMBISYS LABS
key to success

Get Trained
Get Interviewed
Get Employed

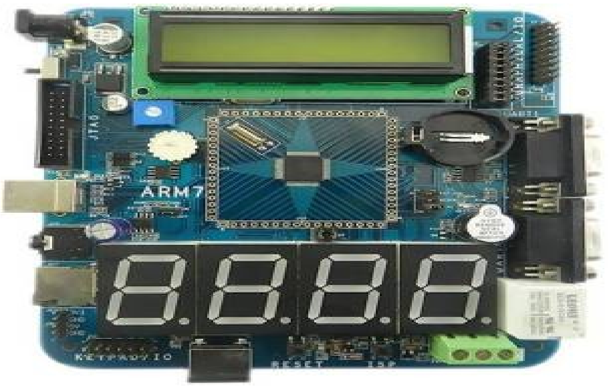
About Embisyslabs @ Bangalore

Embisys Labs is the Embedded Systems consultancy that contributes the Training and development in the areas of Embedded Technologies. We strive for perfection in whatever we do by providing high quality Training, Development and Solutions in Embedded Systems,Embedded Formware, Embedded linux and Linux Device Drives on various embedded target board for our Engineers and customers. At the company level Embisys Labs focuses on innovative embedded project like Embedded IOT, Robotics, Embedded Linux, Linux device drivers projects .

Why Training in Embisyslabs

- Maximum 6 to 8 Participants in one Batch.
- Indivisual Attention to each Participant.
- High Quality practical/application Oriented Training.
- Genuine Placement Assistance.
- ARM 7 Controller for Embedded C Practicals.
- Cortex-A8 Processor for Embedded Linux Practicals.
- Beaglebone and Raspberry-Pi for Driver Practicals.
- Flexible and Convenient time Slots for Classes.
- Experience and co-operative Trainers.

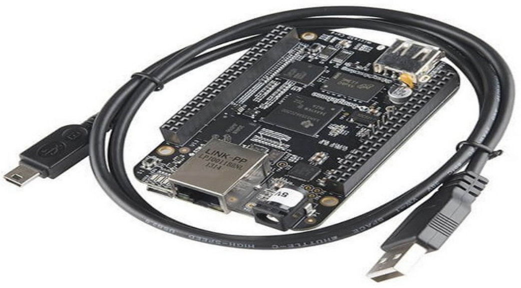
ARM 7 Board



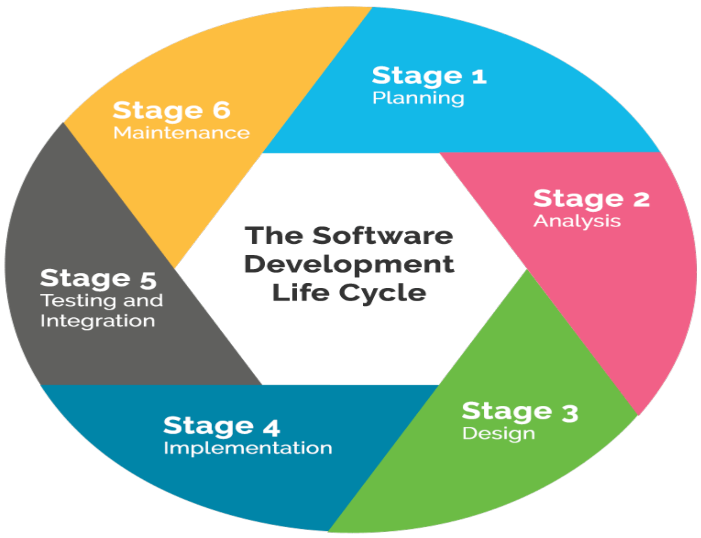
Training and Practicals Process

- Classes 5-Days a week for Weekdays Batch
- Theory(1 1⁄2 -2 hrs.) and practical (3 hrs.)
- Classes 2-Days for a Weekend Batch(Sat & Sun)
- Theory(2 1⁄2 -3 hrs) and practical (3hrs.)
- Daily theory and lab assignments
- Module wise theory and lab exams
- Interviews & Project Guidance
- Repeatation classes will be conducted as required.

Cortex-A8 Beaglebone



Software development methodologies and Project life Cycle



Evaluate
Design
Develop
Analyze
Project Management

MODULE 1: C AND DATA STRUCTURE PROGRAMMING

CH1. GETTING STARTED

- Why C Programming Language
- History & Features
- Compilation Model
- How to Compile & Run a C program
- Strategy of Desinging a Program

CH2. FUNDAMENTALS OF PROGRAMMING

- Variables & Constants
- Keywords & Data Types
- Identifires & Rules
- I/O Functions

CH3. OPERATORS AND CLASSIFICATIONS

- Arithmetic Operators
- Bitwise Operators
- Logical Operators
- Increment Operators
- Decrement Operators
- Relational Operators
- Conditional Operators

CH4. CONTROL FLOW STATEMENTS

- Sequential statements
- Decision making statements
- if,else,nested-if
- break,switch

CH5. LOOPING STATEMENTS

- For Looping
- While Looping
- Do—While Looping
- Continue Looping

CH6. C PRE-PROCESSOR

- File inclusion
- Macro substitution
- Conditional Compilation
- #ifde, #ifndef

CH7. ARRAYS AND STRING

- Definition and Declaration of Array
- Definition and Declaration of String
- Memory Layout &accessing Array Elements
- String Library Functions
- Two dimensional Arrays

CH8. POINTERS [PART 1]

- Definition &Declaration of Pointer
- Indirect Access using Pointers
- Pass by Reference
- Rela. b/w Arrays and Pointers
- Type Casting
- Pointer to an Array
- Array of Pointers

CH15. FILE INPUT/OUTPUT

- System Calls vs. Library Calls
  I/O Library Functions

CH9. FUNCTIONS AND ITS TYPE

- Why Functions ?
- Function Declarations
- Function Prototypes
- Returning a Value or Not
- Arguments and Parameters
- Function Pointers
- Recursion and Recursive function

CH10. SCOPE and LIFETIME 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

CH11. POINTERS [PART 2]

- Dynamic Storage Allocation -
- malloc(),calloc(),realloc(),free()
- Functions Returning a Pointer
- An Array of Character Pointers
- Two Dim.Arrays vs. Array of Pointers
- Command Line Arguments
- Pointers to Pointers
- Use of Function Pointers

CH12. SERCHING &SORTING

- Linear Searching
- Binary Searching
- Bubble sorting
- Selection Sorting

CH13. 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
- # pragma Definition and its use

CH14: STRUCTURE RELATED (UNION)

- Why Union is called Memory Saving Concept
- Difference between Union and Structure
- Typedef - New Name for an Existing Type
- Bit Fields Memory Saving Concept

- Standard Input/Output Descriptors
- fopen(),fread(),fwrite(),fclose()
- Character Input vs. Line Input
- fscanf(),fprintf(),fclose()
- fgtes(),fputs(),fgetc(),fputc()

- Enumerations and its use
- Const and Volatile Modifier
- Volatile and Const Volatile Modifier

MODULE 2: DATA STRUCTURE USING C PROGRAMMING

CH1. INTRODUCTION TO DATA STRUCTURE
- Why data structure ?
- Definition and Classification
- Primitive and Non Primitive

CH2: STACK PROGRAMMING
- What is Stack?
- Push operatins insert an item from top end of Stack
- Pop operatins delete an item from top end of Stack
- Implementation of Stack using Array Pointer
- Implementation of Stack using Pointer

CH3. CH3: QUEUE PROGRAMMING
- What is Queue?
- Insert an item from front end of Queue
- Delete an item from rear end of Queue
- Implementation of Queue using Array Pointer
- Implementation of Queue using Pointer

CH4 . CH4: LINK LIST PROGRAMMING
- Singly link lists
- Circular link lists
- Double link list
- Implement Below Assignment for All types of Link List
- Insert an item from Front End of Link Lis
- Insert an item from Rear End of Link List
- Insert an item at Specific Postion of Link List
- Delete an item from Front End of Link List
- Delete an item from Rear End of Link List
- Delete an item at Specific Postion of Link List
- Reverse the link list
- Find the middle node of link list

C and Data Structures Hands-on Assignments in Class Room

1. More than Hundred Subjective Questions in C and Data Structure Programming
2. More than Hundred Objective Questions in C and Data Structure Programming
3. Two Mini Projects on C and Data Structure Programming Modules
4. Class Room Test based on C and Data Structure Programming Modules

MODULE 3: ARM7TDMI-S and EMBEDDED C PROGRAMMING

CH1. INTRODUCTION TO ARM PROCESSOR
- Intro. to Cortex-A and Cortex-M Series Controller
- Why Embedded C Programming
- Why Assembly Programming
- Difference betwwen C And Embedded C Programming

CH2. ENVIRONMENT SETUP AND ITS USE
- IDE installation (Keil uVision / ARM GCC)
- Downloading and installation of IDE, Flash Magic
- Setting up hardware (development board)

CH3. ARM ARCHITECTURE [LPC2148]
- Overview of ARM7 architecture
- Features of LPC2148 microcontrollers

CH7. ARM LPC2148 UART
- Difference between UART/USART
- UART module overview
- UART Frame structure
- UART Baudrate Calculation
- Sending and receiving data using UART
- U0RBR (UART0 Receive Buffer Register)
- U0THR (UART0 Transmit Holding Register)
- U0DLL and U0DLM (UART0 Divisor Latch Registers)
- U0LCR (UART0 Line Control Register)

CH8. LPC2148 PWM PROGRAMMING
- PWM functionality and applications
- LPC2148 supports 2 types of PWM

- Pin configuration and pin descriptione
- Memory map of LPC2148
- Flash memory and RAM details
- Internal block diagram and functional overview

CH4. ARM GPIO PROGRAMMING
- Understanding ports and pins
- Configuring pins as input/output
- Pin Function Select Registers
- Fast and Slow GPIO Registers
- IOxPIN (GPIO Port Pin value register)
- IOxSET (GPIO Port Output Set register)
- IOxDIR (GPIO Port Direction control register)
- IOxCLR (GPIO Port Output Clear register

CH5. LPC2148 INTERRUPT HANDLING
- Interrupt controller in LPC2148
- Configuring and enabling interrupts
- Writing ISR (Interrupt Service Routine)
- Fast IRQ (highest priority)
- Non-Vectored IRQ (low priority
- Vectored IRQ (medium priority)

CH6. LPC2148 TIMER / COUNTER
- Timer features in LPC2148
- Timer Counter(TC) and Prescale Register(PR)
- What is a Match Register
- What are Capture Registers
- Prescale (TxPR) Related Calculations
- Setting up & configuring Timers

- 7 match registers inside the PWM block
- Configuring and Initializing PWM
- PWM Prescale (PWMPR) Calculations
- DC Motor Speed Control Using PWM
- LED Dimming Using PWM

CH 9. LPC2148 ADC PROGRAMMING
- ADC features in LPC2148
- ADC0 has 6 channels &ADC1 has 8 channels
- Steps for Analog to Digital Conversion
- ADxGDR (ADCx Global Data Register)
- AD0STAT (ADC0 Status Register)
- ADxCR (ADC Control Register)

CH 10. LPC2148 I2C PROGRAMMING
- I2C overview
- I2C-Bus Configuration
- I2C Operating modes
- I2C Master Transmitter mode
- I2C Master Transmitter mode
- I2C Implementation and operation
- I2C Register description
- I2C Programming

CH 11. LPC2148 SPI PROGRAMMING
- SPI overview
- SPI data transfer format
- SPI data to clock phase relationship
- SPI Master operation
- SPI Slave operation
- SPI Register description
- SPI Programming

Embedded C and ARM7 Hands-on Assignments in Class Room

1. All Peripherials Program with Keil C Simulator

2. All Peripherials Program on ARM Board (LPC 2148)

3. Class Room Test based on Embedded C and ARM Architectures

MODULE 4: UNIX and LINUX SYSTEM PROGRAMING

CH1. INTRODUCTION TO UNIX/LINUX
- Histoty of Unix/Linux
- Linux Layered Architecture
- Type of Kernels
- Micro and Monolithic kernel
- Different types of kernel structure
- Linux Bootup Sequence

CH2. FILE SYSTEM MANAGEMENTS
- File Systems – VFS
- File Systems Layouts
- Super Block & Inode Block
- Inode block Structure
- Device Special Files

CH4. PROCESS MANAGEMENTS
- Program and Process
- Process Control Block (PCB)
- States Of Process
- Mode of Execution
- User mode and Kernel mode
- Context Switching
- Scheduling & Priority

CH5. PROCESS RELATED PROGRAMMING
- Process Creation by fork() amd vfork()
- Why fork() not vfork()
- Creation and Destroying Zombie Process
- Creation of Orphan Process
- wait() and waitpid() calls

➢ Types of File ➢ File descriptor table ➢ System calls Sequence ➢ System Vs Function Calls ➢ File related System Calls ➢ open(),read(),write(),close() ➢ stat(),lstat(),dup() etc. **CH3. FILE LOCKING PROGRAMMING** ➢ File Control Operations ➢ Types of File Locking ➢ Advisory and Mandatory File locking ➢ fcntl() and flock()calls	➢ exit() and exec() ,sleep() calls ➢ Creating , synchronizing and performing multiprocessing concepts ➢ Setting and changing nice value and Prority no. **CH6:MEMORY MANAGEMENTS AND MMU** ➢ Memory Policy and Hirarchy ➢ Memory allocation Technique ➢ Physical memory &Virtual Memory ➢ Paging & Demand paging ➢ Memory Mapping using TLB ➢ Swap in & Swap out ➢ Internal & External Fragmentation

MODULE 5: LINUX INTERNALS AND IPCs(INTER PROCESS COMMUNICATION)

CH1. THREADS AND MULTI-THREAD CONCEPTS ➢ Threads on different O.S ➢ Why Threads in Linux ➢ Threads Vs Process ➢ Thread APIs ➢ Creation of Multithreading ➢ Performig Multiple operation using multi-threading **CH2. SIGNALS VS. INTERRUPTS** ➢ Sources of Signals ➢ Diffrents type of Signals ➢ Actions of Signals ➢ Receiving a Signal ➢ Handling a Signal ➢ Signal System Calls **CH3. USER AND DAEMON PROCESS** ➢ Creating a Daemon Process ➢ Characteristics of a Daemon ➢ Writing and Running Daemon	**CH4 . PRIMITIVE INTERPROCESS COMM (IPCS)** ➢ PIPES ➢ Creation of Half and Full-duplex Pipe ➢ Half and Full-duplex communication ➢ FIFO **CH5 . SYSTEMS V IPCs** ➢ Shared Memory ➢ Message Queues ➢ Semaphores **CH6 . NETWORK AND SOCKET PROGRAMMING** ➢ Description of ISO/OSI Model ➢ Types of IP Classes (A,B,C,D and E) ➢ Configuring IP address on Systems ➢ Network addresses and Host addresses ➢ Types of Socket ➢ UDP Connectionless Oriented Socket ➢ TCP/IP Connection Oriented Socket ➢ Iterative Server-Client Programming ➢ Concurrent Server- Client Programming ➢ One Server and Many client Programming

Linux Systems and IPCs Hands-on Assignments in Class Room
1. More than Hundred Subjective Questions in Linux Systems Programming
2. Two Mini Projects on Linux Systems Programming
3. Class Room Test based on Linux Systems Programming Modules

MODULE 6: KERNEL PORTING ON BEAGLE BONE BLACK

CH1. INTRODUCTION OF EMBEDDED LINUX
- Genesis of Linux project
- Embedded hardware for Linux systems
- Criteria for choosing the hardware

CH2.TOOLCHAIN AND SETUP
- What is Toolchain
- Toolchain Components
- Build Systems for Toolchain
- Toolchain Setup Environment
- Toolchain compilation and usage

CH3. BOOTLOADER AND COMPILATION
- What is Loader
- What is Bootloader
- 1st and 2nd Stage Bootloader
- U-Boot Bootloader Porting on New Hardware.
- U-Boot Commands Lists
- Bootloader Cross-Compilation
- Downloading on Target board
- Bootloader commands and usage,
- Bootloader code customization, U-Boot.
- U-Boot Image for Target Board

CH4 . LINUX KERNEL AND COMPILATION
- Browsing Linux Kernel Source
- Visualizing Kernel Source Tree
- Cross-Compilation of Kernel Source
- Generating Kernel Image(uImage or zImage)
- Cross Compiling kernel with rootfs
- Cross Compiling Kernel without rootfs

CH5 . NFS AND TFTP SETUP ON SYSTEM
- Configuring NFS on Host Platform PC
- Configuring TFTP on Host Platform PC

CH6 . COMPILED IMAGES ON TARGET BOARD.
- Downloading and booting kernel image using rootfs over NFS
- Downloading and booting kernel image over TFTP

CH7 . PROGRAMMING FOR TARGET BOARD
- User Level Application Programming
- Device Driver Programming
- GPIO Interfacing Programming

MODULE 7: LINUX CHAR DEVICE DRIVER AND KERNEL PROGRAMMING

CH1: AN INTRO. TO DEVICE DRIVERS
- Role of the Device Drivers
- Splitting the kernel
- Classes of devices and modules
- Kernel Architecture or Model

CH2:BUILDING AND RUNNING MODULES
- Types of Modules in the kernel
- Writing Your first kernel module
- Module Related Commands
- Kernel Module vs Applications
- User space vs Kernel space
- Compiling Modules
- Loading and Unloading Modules
- Module Parameters

CH3: CHAR DEVICE DRIVERS
- Major and Minor Numbers
- The Internal Representation of Device Numbers
- Allocating and Freeing Device Numbers
- File Operations Data structure
- Driver methods and Function Pointers
- Char Device Registration
- The Cdev Structure
- The file and inode Structure
- Manual Creation of Device Files
- Automatic Creation of Device Files

CH4: MEMORY ALLOCATION TECHNIQUE
- The Real Story of kmalloc
- The Flags Argument
- Memory zones
- kmalloc and Friends

CH5: ADVANCED CHAR DRIVER OPERATIONS
- Inpout/Output Control (ioctl)
- User space, the ioctl system call
- The ioctl driver method
- Choosing the ioctl Commands
- Using the ioctl Argument

CH6:CONCURRENCY AND RACE CONDITION
- Concurrency and its Managements
- Semaphores and Mutexes
- Linux Semaphore Implementation
- Introduction to the Semaphore API
- Spinlocks Implementation
- Introduction to the Spinlock API
- Spinlocks and Atomic Context

CH7: INTERRUPT AND INTERRUPT HANDLING
- The Definition and Role of Interrupt
- Installing an Interrupt Handler
- Implementing a Handler
- Handler Arguments and Return Value
- Installing a Shared Handler
- Top and Bottom Halves
- Tasklets and Workqueues mechanisms

Device Driver and Beaglebone Hands-on Assignments in Class Room

1. Subjective Questions in Linux Device Driver Programming
2. Case Study on Linux Device Driver and Beaglebone Black
3. Class Room Test based on Linux Device Driver Modules

MODULE 8:RTOS :REAL TIME OPERATING SYSTEM

CH1: DIFFERENCE BETWEEN GPOS AND RTOS

- Introduction and Overview
- Components O.S
- Monolithic Vs Microkernel Architecture

CH2: REAL TIME MULTITASKING

- Task Basics Structure
- Task Control Block
- Task Creation
- Task States
- Task Status

CH3: INTER TASK COMMUNICATION(ITCs)

- Shared Memory
- Message Queues
- Pipes

CH4: SEMAPHORES and SYNCHRONIZATION

- Synchronization Problem
- Binary Semaphore
- Mutex Semaphore
- Mutual Exclusion Problem
- Priority Inversion
- Priority Inheritance

CH5: INTERRUPT AND EXCEPTION

- What is Interrupt and Signal
- What is Exception
- What is signal Handler
- What is Exception Handler

Email us: info@embisyslabs.com
www.embisyslabs.com
Contact us:+91-9972257855