On a Mission to Transform Talent

talent sprint

EMBEDDED SYSTEMS

COURSE CURRICULUM

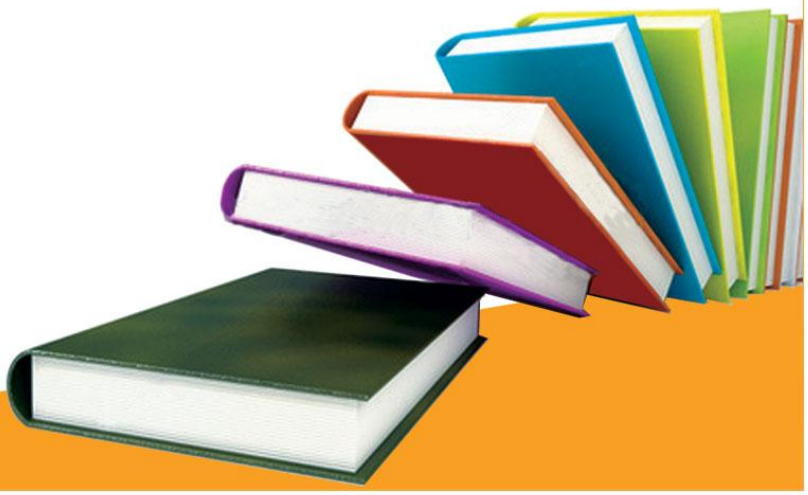

Table of Contents

Embedded Systems Course Curriculum

Course: Embedded Systems (Duration: 14 Weeks)

Course Overview:

This program is designed by TalentSprint to prepare aspirants for a career as an Embedded Systems professional.

In this course, the fundamentals of embedded system hardware and firmware design will be explored.

Issues such as embedded processor selection, hardware/firmware partitioning, circuit design, circuit layout, circuit debugging, development tools, firmware architecture, firmware design, and firmware debugging will be discussed.

The architecture and instruction set of the microcontroller will be discussed, and a wire wrapped microcontroller board will be built and debugged by each student.

Audience:

Engineering graduates (ECE & EEE) with a good consistent academic record in SSLC, HSLC and graduation, with 0-1 years of work experience.
The participants will possess good computer usage skills - using computers for creating documents, sending emails etc.

- The participants will be shortlisted based on an Aptitude test.
- The final selection will be based on personal interview.

Course Objectives:

On completion of this course, successful participants will be able to:

- Perform effectively as entry level Embedded Systems professionals.
- Develop and maintain applications written using Embedded C.
- Independently design and develop a hardware platform encompassing a microcontroller and peripherals.

Assessment Strategies:

Assessment methodology is based on Kirkpatrick's model of measuring learning effectiveness (Level 2 & 3). The participants will be assessed using tests, examinations, viva-voce and written assignments to evaluate the degree of achievement of the objectives. Specifically the following assessment mechanisms will be used.

- Continuous assessments during training to track learning.
- Periodic written and/or online tests for review.
- End of module written and/or online examinations.
- End of course project.

Certification:

On successful completion, the participant will be awarded a Certificate in Embedded Systems Engineering Foundation.

Module 1: Basic Electronics and PCB Software Overview (Duration: 1 Week)

Objectives:

At the end of this module, the successful participants will be able to:

- Understand the Number system, Boolean algebra, Logic Gates, Adders, Flip-Flops, Counters, Registers, Multiplexers and Schmitt triggers.
- Understand Digital circuits are electric circuits based on a number of discrete voltage levels.
- Understands basics of Electronic Components like Resistor, Capacitor, Diode, Transistor and IC's etc.
- Use PC Based electronic design software to View and Draw, Schematic and PCB.
- Tackle problems with EMI and EMC and address problems at the Board level.

Lessons

1. Number Systems
2. Boolean Algebra
3. About Basic Electronic Components
4. Data sheets, power supplies, voltage regulators. Thermal considerations, heat sinks, parts kits.
5. Introduction to Embedded Systems Laboratory and equipment.
6. Logic probes, voltmeters and oscilloscopes; Debugging using logic analyzers, state and timing information.
7. Designing with tolerances and margins, part variations and substitutions, reliability/part count.
8. Interfacing different logic families, fan out, signal buffering, noise margins, pull ups/pull downs.
9. Overview of board development process, wire wrapping and soldering.
10. Schematics and wiring diagrams, recommended practices, CAD tools.
11. Board layout considerations, signal integrity (noise, crosstalk, etc.) and decoupling techniques.
12. Manufacturing and test engineering, PCB design, ground and power planes, EMI, EMC.

Module 2: Embedded C Programming (Duration: 5 Weeks)

Objectives:

At the end of this module, the successful participants will be able to:

- Understand the basic components of a computer, write assembly and C language programs that perform I/O functions and implement simple data structures, manipulate numbers in multiple formats, and understand how software uses global memory to store permanent information and the stack to store temporary information.
- Design and implement at elementary data structures such as linked lists, stacks and queues.
- Understand how the computer stores and manipulates data (characters, integers, and fixed-point numbers) and how basic arithmetic and logical operations are performed.
- Learn techniques for implementing interrupt handler code in C, multi-module programming including applications containing a mixture of C and assembly language modules, and techniques for manipulating hardware registers and special function registers.

Lessons

1. C Basic data types
2. Programming constructs
3. Functions in C
4. Data Structures
5. Advanced topics
6. Overview of the C standard library
7. Embedded System Oriented Topics
8. MISRA C — Designing Safer C Programs
9. Basics of event driven programming

Module 3: Microcontroller Architectures (Duration: 6 Weeks)

Objectives:

At the end of this module, the successful participants will be able to understand:

- How the computer interacts with its environment.
- How an embedded system can be used to solve Electrical and Electronics problems.
- How hardware construction is performed on a breadboard and debugged using a millimeter (students learn to measure voltage and resistance).
- How software is developed for the micro-controller in assembly and C; 10 labs are simulated in IDE and then run on the real Targer xxxx Board. Software debugging occurs during the simulation stage. Verification occurs in both stages. 10 Labs are written in a combination of assembly and C.

Lessons

1. Embedded systems descriptions, definitions, and vocabulary.
2. Embedded system design considerations and requirements.
3. Processor selection and tradeoffs.
4. Microprocessor/microcontroller architectures and instruction sets, 8051 architecture.
5. Design cycle, planning a development project, derivation of requirements, tradeoffs.
6. Oscillators and reset circuits. Microprocessor supervisory circuits, watchdog timers.
7. Microcontroller peripherals, selection and interfacing. Core component circuitry (CPU, ROM, RAM).
8. 8051 timing diagrams, program read, data read, data write.
9. Port pin structure. Controlling port pins in asm. User interface design, human factors. Driving LEDs.
10. Timing requirements, propagation delay, setup, hold, rise/fall times, timing analysis. Clock skew.
11. Memory selection and interface, SRAM, NVRAM, DRAM, EPROM, EEPROM, Flash.
12. Switch debouncing in hardware and firmware, keypad decoding.
13. 8051 timers/counters. Interrupts and Interrupt Service Routines (ISRs).
14. Serial communication, RS-232/485, line drivers/receivers, charge pumps, terminal emulation, USB.
15. EEPROMs, I2C and synchronous serial communication.
16. LCDs
17. Analog-to-Digital Converters (ADCs), Digital-to-Analog Converters (DACs).
18. Motor control, stepper motors, DC motors, PWM, H-Bridges. Case study: hard disk drive.

Module 4: Overview of RTOS (Duration: 1 Week)

Objectives:

At the end of this module, the successful participants will be able to:

- Appreciate the use of multitasking techniques in real-time systems.
- Understand the fundamental concepts of real-time operating systems.
- Understand the features and structures of practical Implementations.

Lessons

1. Introduction to distributed embedded systems
2. Task manager
3. Interrupt management
4. Time management
5. Memory management Inter task communication and synchronization
6. pSOS system overview
7. pSOS real time kernel concepts
8. pNA network manager concepts
9. Input output system
10. Exploring pRISM environment
11. Overview of building a pSOS system application debugger

Module 5: Get Started with LabView Environment (Duration: 1 Week)
Objectives: At the end of this module, the successful participants will be able to: • Learn how to develop basic applications in the LabView graphical programming environment. • Create applications using a state machine design pattern, read and write data to file.
Lessons
1. Getting Started with LabView, Opening a New VI from a Template 2. Adding a Control to the Front Panel, Running a VI 3. Front Panel and Block Diagram Tools, LabView Documentation Resources 4. Customizing a User Interface from the Block Diagram 5. Configuring a VI to Run Continuously until the User Stops It 6. Customizing the Block Diagram Code 7. Hardware and Software Requirements 8. Acquiring a Signal in NI-DAQmx, All Controls and Indicators, All VIs and Functions 9. When to Use Other LabView Features