Introduction to Embedded System

Unit 1(continued)

Topics to be discussed

- Embedded system architecture
  - Architectural Structures
- System Model
- Standards
  - Programming Languages and Standards
  - Standards and Networking
  - Multiple standards-based device

Embedded system architecture

- Architectures – abstraction of embedded device
- Generalization of the system- doesn’t have detailed implementation info such as h/w circuit design s/w source code.
- Architectural level - > components are represented as elements
- Embedded architecture:
  - Elements of embedded system
  - Interacting with embedded system
  - Properties of individual elements
  - Interactive relationship between elements

Embedded system architecture

- Architecture-level information is physically represented in the form of *structures*.
- Structures: a ***snapshot*** *of the system's hardware and software at design time and/or at run-time, given a particular environment and a given set of elements.*
- An architecture is typically made up of more than one structure.
- All structures are inherently related to each other.
- Next, a table is shown that:
  - Summarizes some of the most common structures that can make up embedded architectures
  - Shows generally what the elements of a particular structure represent and how these elements interrelate.

Architectural Structures

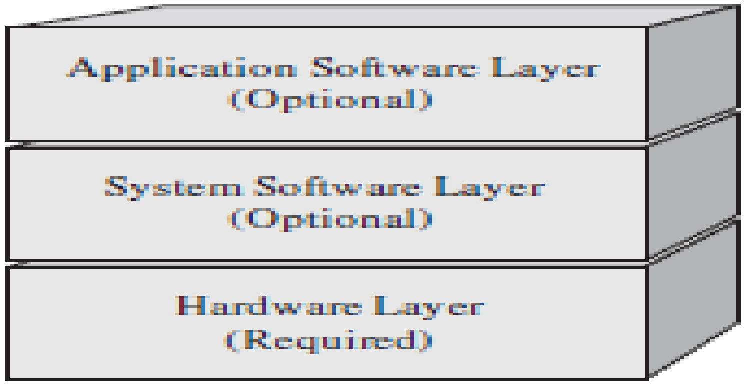
Structure Types*			Definition
Module			Elements (referred to as modules) are defined as the different functional components (the essential hardware and/or software that the system needs to function correctly) within an embedded device. Marketing and sales architectural diagrams are typically represented as modular structures, since software or hardware is typically packaged for sale as modules (i.e., an operating system, a processor, a JVM, and so on).
	Uses (also referred to as subsystem and component)		A type of modular structure representing system at runtime in which modules are inter-related by their usages (what module uses what other module, for example).
		Layers	A type of Uses structure in which modules are organized in layers (i.e., hierarchical) in which modules in higher layers use (require) modules of lower layers.
		Kernel	Structure presents modules that use modules (services) of an operating system kernel or are manipulated by the kernel.
		Channel Architecture	Structure presents modules sequentially, showing the module transformations through their usages.
		Virtual Machine	Structure presents modules that use modules of a virtual machine.
	Decomposition		A type of modular structure in which some modules are actually subunits (decomposed units) of other modules, and inter-relations are indicated as such. Typically used to determine resource allocation, project management (planning), data management (encapsulation, privitization, etc.).
	Class (also referred to as generalization)		This is a type of modular structure representing software and in which modules are referred to as classes, and inter-relationships are defined according to the object-oriented approach in which classes are inheriting from other classes, or are actual instances of a parent class (for example). Useful in designing systems with similar foundations.

Architectural Structures(Cntd..)

Structure Types*			Definition
Component and Connector			These structures are composed of elements that are either components (main hw/sw processing units, such as processors, a Java Virtual Machine, etc.) or connectors (communication mechanism that inter-connects components, such as a hw bus, or sw OS messages, etc.).
	Client/Server (also referred to as distribution)		Structure of system at runtime where components are clients or servers (or objects), and connectors are the mechanisms used (protocols, messages, packets, etc.) used to intercommunicate between clients and servers (or objects).
	Process (also referred to as communicating processes)		This structure is a SW structure of a system containing an operating system. Components are processes and/or threads (see Chapter 9 on OSes), and their connecters are the inter-process communication mechanisms (shared data, pipes, etc.) Useful for analyzing scheduling and performance.
	Concurrency and Resource		This structure is a runtime snap shot of a system containing an OS, and in which components are connected via threads running in parallel (see Chapter 9, Operating Systems). Essentially, this structure is used for resource management and to determine if there are any problems with shared resources, as well as to determine what sw can be executed in parallel.
		Interrupt	Structure represents the interrupt handling mechanisms in system.
		Scheduling (EDF, priority, round-robin)	Structure represents the task scheduling mechanism of threads demonstrating the fairness of the OS scheduler.
	Memory		This runtime representation is of memory and data components with the memory allocation and deallocation (connector) schemes—essentially the memory management scheme of the system.
		Garbage Collection	This structure represents the garbage allocation scheme (more in Chapter 2).
		Allocation	This structure represents the memory allocation scheme of the system (static or dynamic, size, and so on).
	Safety and Reliability		This structure is of the system at runtime in which redundant components (hw and sw elements) and their intercommunication mechanisms demonstrate the reliability and safety of a system in the event of problems (its ability to recover from a variety of problems).

System Model

- It’s a primary architectural tool, introduces the major elements located within an embedded system design.
- All embedded systems share one similarity at the highest level; i.e. they all have at least one layer (hardware) or all layers into which all components fall.
- **<u>hardware layer</u>**
  - contains all the major physical components located on an embedded board,
- **<u>system & application software layers</u>**
  - contain all of the software located on and being processed by the embedded system.



Standards

- Standards dictate
  - how components should be designed,
  - what additional components are required in the system to allow for their successful integration and function.
- Standards can define the functionality that is specific to each of the layers of the embedded systems model,
- Standard can be classified as:
  - *market-specific* standards(intended for specific groups of embedded devices),
  - *general-purpose* standards,
  - or standards that are applicable to *both* categories(include networking standards and some television standards.).

Examples of standards implemented in embedded systems

Standard Type		Standard	Purpose
Market Specific	*Consumer Electronics*	JavaTV	The Java TV Application Programming Interface (API) is an extension of the Java platform that provides access to functionality unique to a digital television receiver, such as: audio video streaming, conditional access, access to in-band and out-of-band data channels, access to service information data, tuner control for channel changing, on-screen graphics control, media-synchronization (allows interactive television content to be synchronized with the underlying video and background audio of a television program) and application lifecycle control. (Enables content to gracefully coexist with television programming content such as commercials).[2-3] (See java.sun.com)
Market Specific *(cont.)*	*Consumer Electronics (cont.)*	ATVEF (Advanced Television Enhancement Forum) – SMPTE (Society of Motion Picture and Television Engineers) DDE-1	The ATVEF Enhanced Content Specification defines fundamentals necessary to enable creation of HTML-enhanced television content that can be reliably broadcast across any network to any compliant receiver. ATVEF is a standard for creating enhanced, interactive television content and delivering that content to a range of television, set-top, and PC-based receivers. ATVEF [SMPTE DDE-1] defines the standards used to create enhanced content that can be delivered over a variety of mediums—including analog (NTSC) and digital (ATSC) television broadcasts—and a variety of networks, including terrestrial broadcast, cable, and satellite.[2-6] (See www.smpte.org/ or www.atvef.com)

(Cntd.)

Market Specific *(cont.)*	*Medical Devices*	FDA (USA)	U.S. government standards for medical devices relating to the aspects of safety and/or effectiveness of the device. **Class I devices** are defined as non-life sustaining. These products are the least complicated and their failure poses little risk. **Class II devices** are more complicated and present more risk than Class I, though are also non-life sustaining. They are also subject to any specific performance standards. **Class III devices** sustain or support life, so that their failure is life threatening. Standards include areas of anesthesia (i.e., Standard Specification for Minimum Performance and Safety Requirements for Resuscitators Intended for Use with Humans, Standard Specification for Ventilators Intended for Use in Critical Care, etc.), cardiovascular/neurology (i.e., Intracranial pressure monitoring devices, etc.), dental/ENT (i.e., Medical Electrical Equipment – Part 2: Particular Requirements for the Safety of Endoscope Equipment, etc.), plastic surgery (i.e., Standard Performance and Safety Specification for Cryosurgical Medical Instrumentation, etc.) ObGyn/Gastroenterology (i.e., Medical electrical equipment – Part 2: Particular requirements for the safety of haemodialysis, haemodiafiltration and haemofiltration equipment, etc.), and so on.[2-13] (See www.fda.gov/)
		IEEE1073 Medical Device Communications	IEEE 1073 standards for medical device communication provide plug-and-play interoperability at the point-of-care, optimized for the acute care environment. The IEEE 1073 General Committee is chartered under the IEEE Engineering in Medicine and Biology Society, and works closely with other national and international organizations, including HL7, NCCLS, ISO TC215, CEN TC251, and ANSI HISB.[2-15] (See www.ieee1073.org/)

(Cntd.)

General Purpose	*Networking*	HTTP (Hypertext Transfer Protocol)	A World Wide Web (WWW) protocol defined by a number of different RFCs, including RFC2616, 2016, 2069, 2109, and so on—Application Layer networking protocol implemented within browsers on any device, for example. (See http://www.w3c.org/Protocols/Specs.html)
		TCP (Transmission Control Protocol)/IP (Internet Protocol)	Protocol stack based on RFCs (Request for Comments) 791(IP) & 793 (TCP) that define system software components (more information in Chapter 10). (See http://www.faqs.org/rfcs/)
		IEEE (Institute of Electronics and Electrical Engineers) 802.3 Ethernet	Networking protocol that defines hardware and system software components for local area networks (LANs) (more information in Chapters 6 and 8). (See www.ieee.org)
		Bluetooth	Bluetooth Specifications are developed by the Bluetooth Special Interest Group (SIG), which allows for developing interactive services and applications over interoperable radio modules and data communication protocols (more information on Bluetooth in Chapter 10).[2-21] (See www.bluetooth.org)

(Cntd.)

General Purpose *(cont.)*	*Programming Languages*	pJava (Personal Java)	Embedded Java standard from Sun Microsystems targeted and larger embedded systems (more information in Section 2.1). (See java.sun.com)
		J2ME (Java 2 Micro Edition)	Set of embedded standards from Sun Microsystems targeting the entire range of embedded systems, both in size and vertical markets (more information in Section 2.1). (See java.sun.com)
		.NET Compact Framework	Microsoft-based system that allows an embedded system to support applications written in several different languages, including C# and Visual Basic (more information in Section 2.1).
	Security	Netscape IETF (Internet Engineering Task Force) SSL (Secure Socket Layer) 128-bit Encryption	The SSL is a security protocol that provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection, and is typically integrated into browsers and web servers. There are different versions of SSL (40-bit, 128-bit, etc.), with "128-bit" referring to the length of the "session key" generated by every encrypted transaction (the longer the key, the more difficult it is to break the encryption code). SSL relies on session keys, as well as digital certificates (digital identification cards) for the authentication algorithm. (See http://wp.netscape.com/eng/ssl3/ for version 3 (latest version at the time this book was written) of Netscape's SSL specification.
		IEEE 802.10 Standards for Interoperable LAN/MAN Security (SILS)	Provides a group of specifications at the hardware and system software layer to implement security in networks. (See http://standards.ieee.org/getieee802/index.html)

Programming Languages and Standards

- A programming language can introduce an additional component into an embedded architecture
- In embedded systems design, there is no single language that is the perfect solution for every system.
- Examples- Java and the .NET Compact Framework, ANCI C
- Because machine code is the only language the hardware can directly execute, all these languages need some type of mechanism to generate the corresponding machine code.
- This mechanism usually includes one or some combination of *preprocessing*, *translation*, and *interpretation*.

(Cntd...)

- Depending on the language, these mechanisms exist on the programmer’s ***host*** system (typically a nonembedded development system, such as a PC or Sparc station), or the ***target*** system (the embedded system being developed).

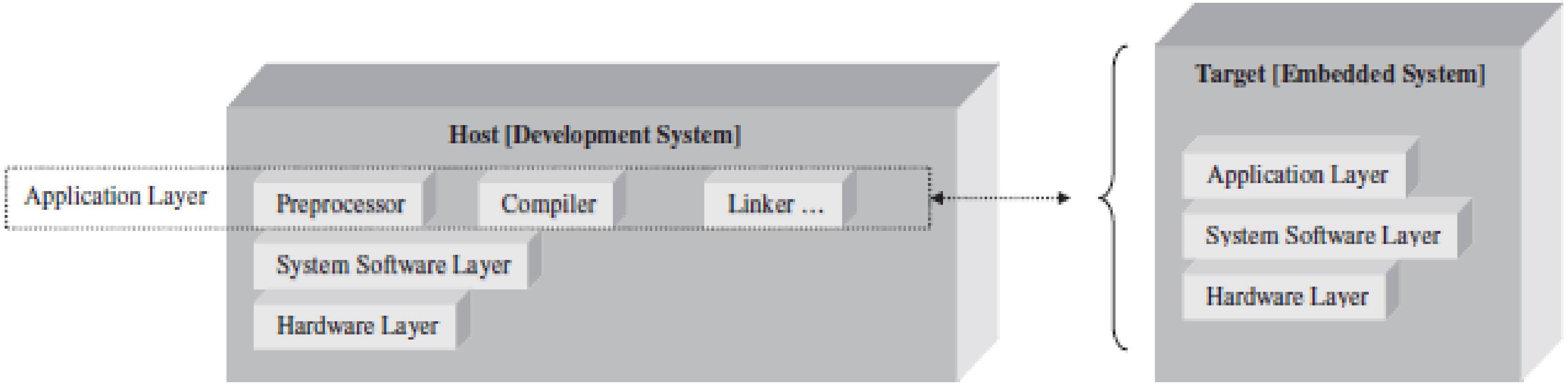
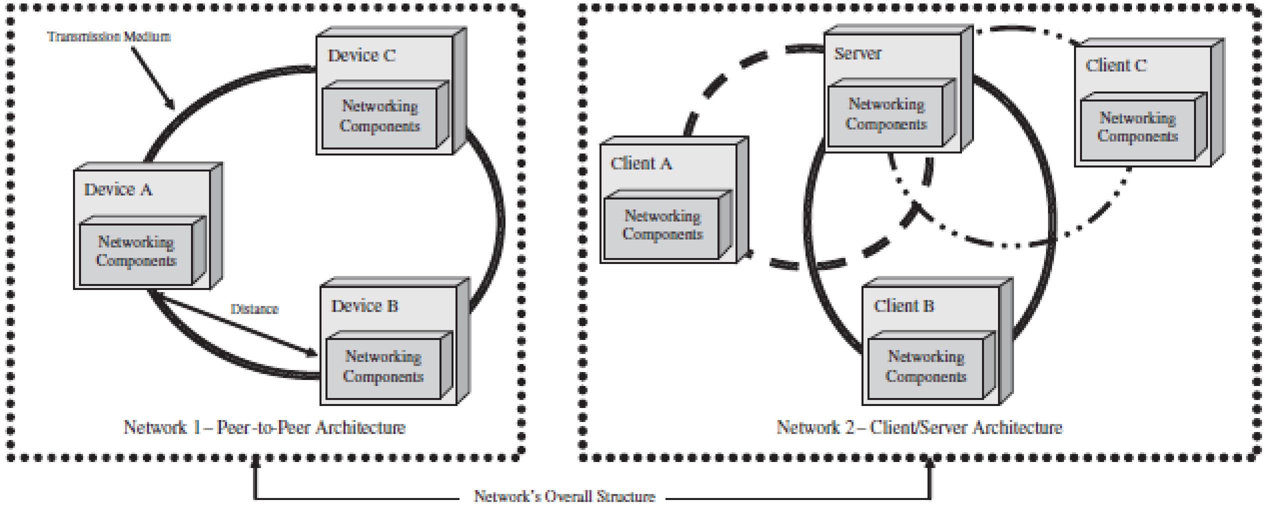


Figure 2-2: Host and target system diagram

Standards & Networking

- If an embedded system needs to communicate with any other system, whether a development host machine, a server, or another embedded device, it must implement some type of connection (networking) scheme.
- In order for communication to be successful, there needs to be a scheme that interconnecting systems agree upon.

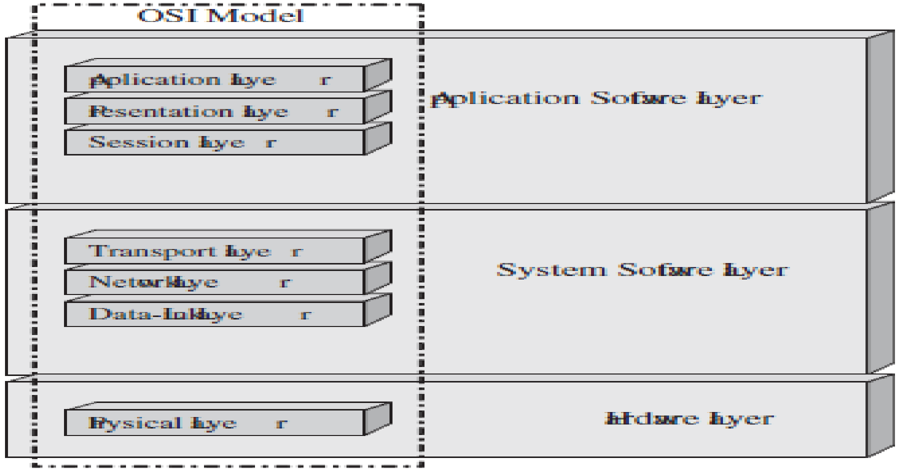


The Network’s Architecture

- The relationship between connected devices in a network determines the network’s overall architecture.
- The most common architecture types for networks :
  - **peer-to-peer architecture-**
    - no centralized area of control,
    - Every device on the network must manage its own resources and requirements.
  - **client/server architecture-**
    - a centralized device, server, manages most of the network’s requirements and resources.
    - other devices on the network, called clients, contain fewer resources and must utilize the server’s resource.
  - **Hybrid architecture-**
    - combination of the peer-to-peer and client/server architecture

Open Systems Interconnection (OSI) Model

- Universal networking model which demonstrate :
  - the dependencies between the internal networking components of an embedded system and the network's architecture,
  - the distance between connected devices,
  - and the transmission medium connecting the devices,
- All the required networking components in a device can be grouped together into the OSI model
- Created in the early 1980s by the International Organization for Standardization (ISO).



(Cntd…)

- Networking connection starts with data originating at the application layer of one device and flowing downward through all seven layers,
- Each layer adds a new bit of information to the data being sent across the network.
- Information, called the ***header*** is appended to the data at every layer (except for the physical and application layers) for peer layers in connected devices to process.
- In other words, the data is wrapped with information for other devices to unwrap and process.
- The data is then sent over the transmission medium to the physical layer of a connected device, and then up through the connected device's layers.
- These layers then process the data (that is, strip the headers, reformat, etc.) as the data flows upward.
- The functionality and methodologies implemented at each layer based on the OSI model are also commonly referred to as ***networking protocols***.

Multiple Standards-Based Device Example: Digital Television (DTV)

- Analog TVs process incoming analog signals of traditional TV video and audio content,
- Digital TVs (DTVs) process both incoming analog and digital signals of TV video/audio content, as well as application data content that is embedded within the entire digital data stream.
- **Why Digital Television (DTV) ?**
  - *According to Forrester Research, average consumer spends seven (7) hours a day watching TV, and TV is in 99% of U.S. homes.*
- The type of application data embedded is dependent on the capabilities of the DTV receiver itself.
- Depending on the type of receiver, DTVs can implement general-purpose, market-specific, and/or application-specific standards all into one DTV/set-top box (STB) system architecture design.

Examples of DTV standards

Standard Type	Standard
Market Specific	Digital video broadcasting (DVB) – multimedia home platform (MHP)
	Java TV
	Home audio/video interoperability (HAVi)
	Digital Audio Video Council (DAVIC)
	Advanced Television Standards Committee (ATSC)/Digital TV Applications Software Environment (DASE)
	Advanced Television Enhancement Forum (ATVEF)
	Digital Television Industrial Alliance of China (DTVIA)
	Association of Radio Industries and Business of Japan (ARIB-BML)
	OpenLabs OpenCable application platform (OCAP)
	Open services gateway initiative (OSGi)
	OpenTV
	MicrosoftTV
General Purpose	HTTP (hypertext transfer protocol) – in browser applications
	POP3 (post office protocol) – in e-mail application
	IMAP4 (Internet message access protocol) – in e-mail application
	SMTP (simple mail transfer protocol) – in e-mail application
	Java
	Networking (terrestrial, cable, and satellite)
	POSIX

DTV standards in the Embedded Systems Model