EEE 204 Introduction to Embedded Systems

Asst. Prof. Dr Mahmut AYKAÇ

INTRODUCTION

Course Content and Schedule

1. **Introduction to Embedded Systems (Lecture 1)**
2. Number Systems and Data Formats **(Lecture 2)**
3. Microcomputer Organization **(Lecture 3 and 4)**
4. Assembly Language Programming on Embedded Processor **(Lecture 5, 6 and 7)**
5. Fundamentals of Interfacing **(Lecture 7)**
6. Embedded Programming Using C **(Lecture 7 and 8)**
7. Introduction to Interrupts **(Lecture 9)**
8. Pull up/down resistors, timers and counters **(Lecture 10)**
9. Analog to Digital Converters **(Lecture 11)**

Reference Books & Grading

1. Introduction to Embedded Systems Using Microcontrollers and the MSP430, Manuel Jiménez, Rogelio Palomera, Isidoro Couvertier
2. Embedded Circuit Desing Using MSP430 series, Chris Nagy
3. Embedded System Design using MSP430 LaunchPad Development Kit, Texas Instrument Company
4. Embedded Systems Design Using the MSP430FR2355 LaunchPad, Brock J. LaMeres

Score = Midterm Exams (20%+20%) + Laboratory 20% + Final Exam 40%

Embedded System?

There are many definitions for this. Here are some…

- An **embedded system** is a special-purpose computer system designed to perform one or a few dedicated functions with real-time computing constraints include hardware, software and mechanical parts. OR….
- An **embedded system** is a microprocessor-based system that is built to control a function or range of functions and is not designed to be programmed by the end user in the same way that a PC is. OR…
- An **embedded system** can be broadly defined as a device that contains tightly coupled hardware and software components to perform a single function

Early Forms of Embedded Systems

The concept of an embedded system is as old as the concept of a an electronic computer, and in a certain way, it can be said to precede the concept of a general purpose computer. If we look a the earliest forms of computing devices, they adhere better to the definition of an embedded system than to that of a general purpose computer.

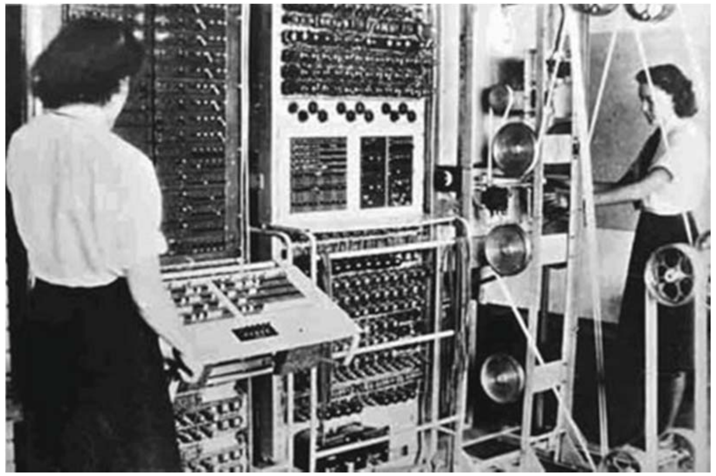


Fig. 1 Control panel and paper tape transport view of a Colossus Mark II computer, used to break encrypted teleprinter German messages during WorldWar II

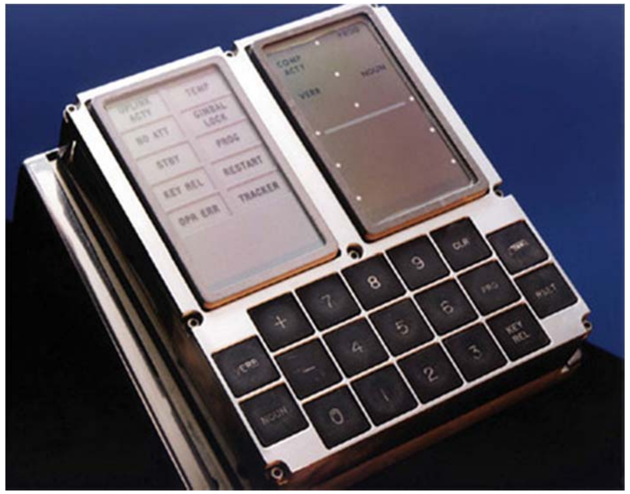


Fig. 2 Apollo Guidance Computer (AGC), was part of the guidance and navigation system used by NASA in the Apollo program for various spaceships.

Birth of Modern Embedded Systems

The beginning of the decade of 1970 witnessed the development of the first microprocessor designs. By the end of 1971, almost simultaneously and independently, design teams working for Texas Instruments, Intel, and the US Navy had developed implementations of the first microprocessors.

The i4004 recognized as the *first commercial, stand-alone single chip microprocessor,* was launched by Intel in November 1971.

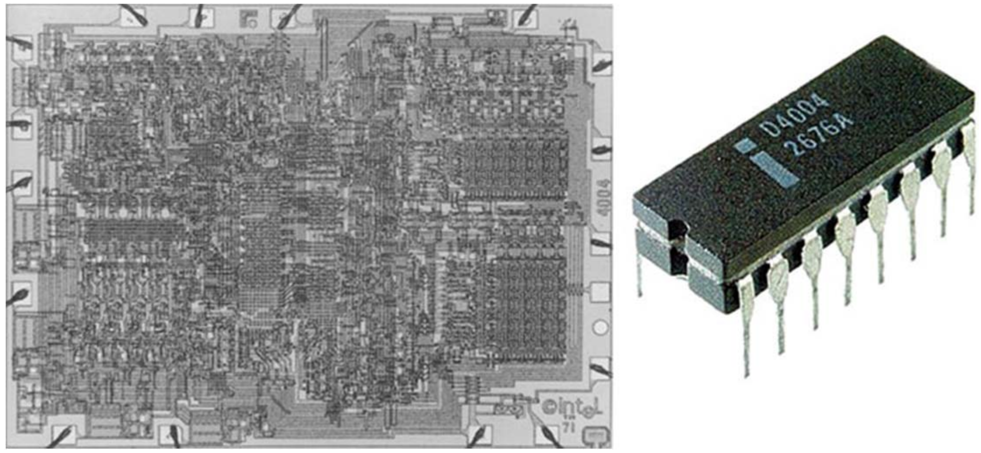
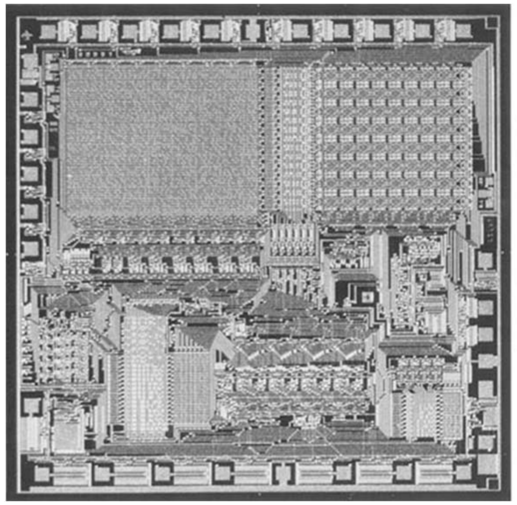


Figure Intel 4004

Birth of Modern Embedded Systems

Gary Boone from Texas Instruments was awarded in 1973 the patent of the *first single-chip microprocessor architecture* for its 1971 design of the TMS1000. This chip was a 4-bit CPU that incorporated in the same die 1K of ROM and 256 bits of RAM to offer a complete computer functionality in a single-chip, making it the first microcomputer-on-a-chip (also known as **microcontroller**).



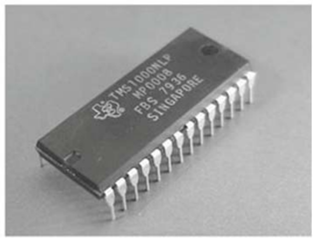


Figure TMS1000

Evolution of Modern Embedded Systems

- Microprocessor designs soon evolved from 4-bit to 8-bit CPUs. By the end of the 1970s, the design arena was dominated by 8-bit CPUs and the market for microprocessors-based embedded applications had grown to hundreds of millions of dollars.
- The list of initial players grew to more than a dozen of chip manufacturers that, besides Texas Instruments and Intel, included Motorola, Zilog, Intersil, National Instruments, MOS Technology, and Signetics, to mention just a few of the most renowned.

Evolution of Modern Embedded Systems

- Remarkable parts include the Intel 8080 that eventually evolved into the famous 80×86/Pentium series, the Zilog Z-80, Motorola 6800 and MOS 6502. The evolution in CPU sizes continued through the 1980s and 1990s to 16-, 32-, and 64-bit designs, and nowadays even some specialized CPUs crunching data at 128-bit widths. In terms of manufacturers and availability of processors, the list has grown to the point that it is possible to find over several dozens of different choices for processor sizes 32-bit and above, and hundreds of 16- and 8-bit processors.

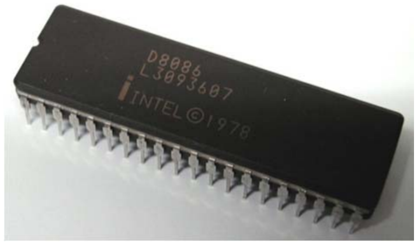


Fig.1 Intel 80x86

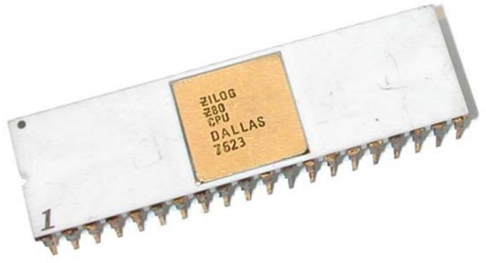


Fig.2 Zilog Z-80

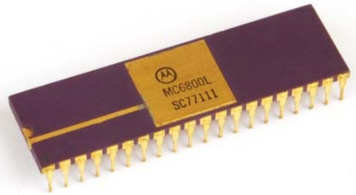


Fig.3 Motorola 6800

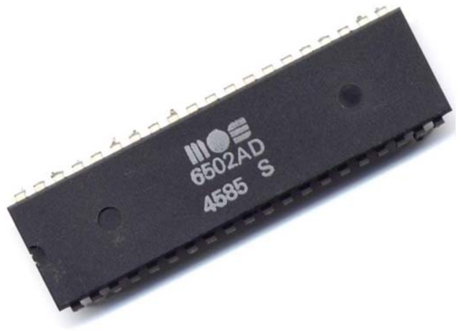


Fig.4 MOS 6502

Contemporary Embedded Systems

- To better illustrate the case, consider the application illustrated in Figure, corresponding to a generic multimedia player. The system provides audio input/output capabilities, a digital camera, a video processing system, a hard-drive, a user interface (keys, a touch screen, and graphic display), power management and digital communication components.

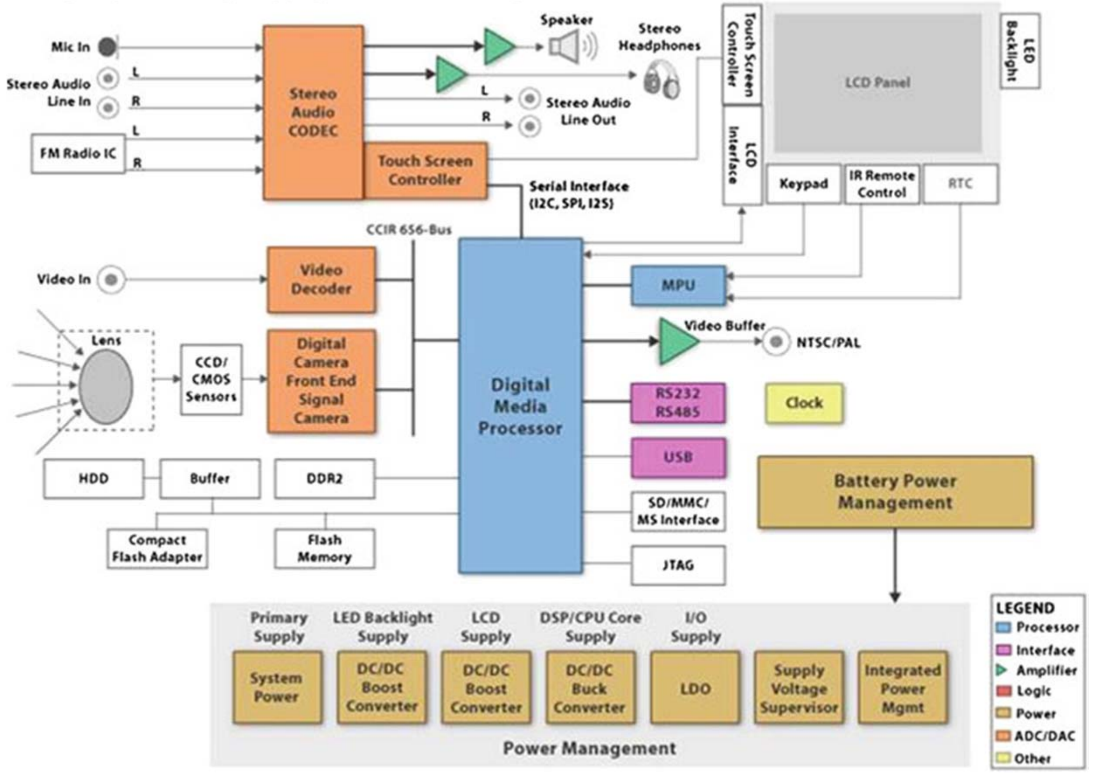


Figure Generic multi-function media player

Contemporary Embedded Systems (Examples)

A ventilator is a machine that provides mechanical ventilation by moving breathable air into and out of the lungs, to deliver breaths to a patient who is physically unable to breathe, or breathing insufficiently

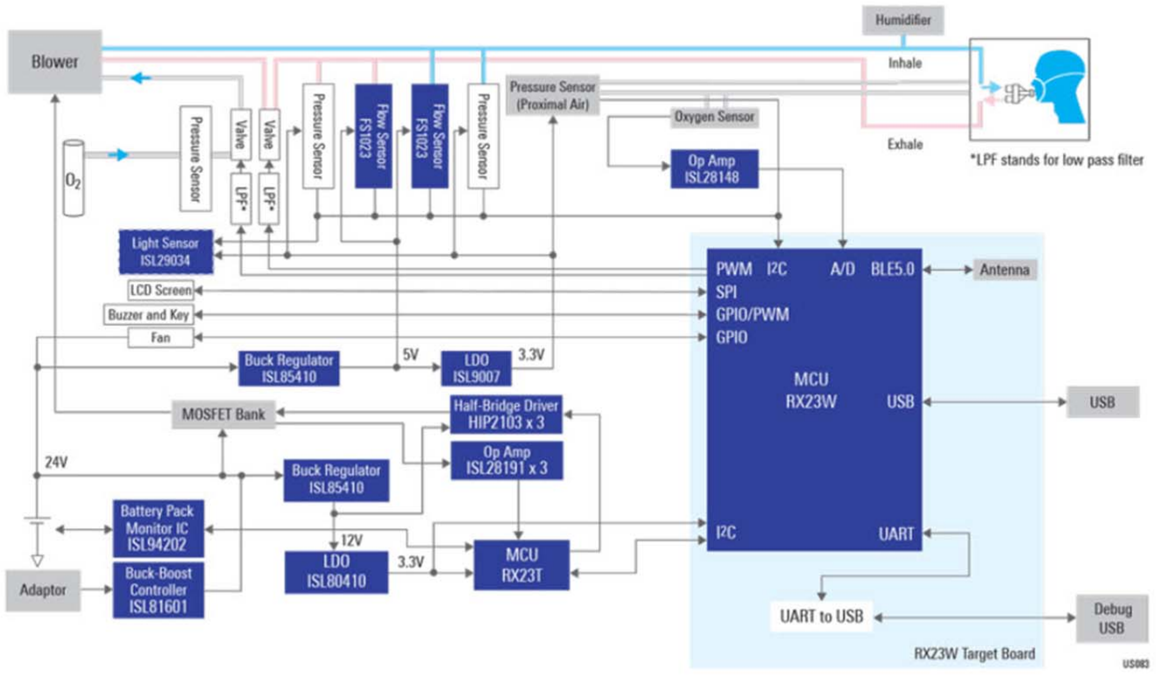
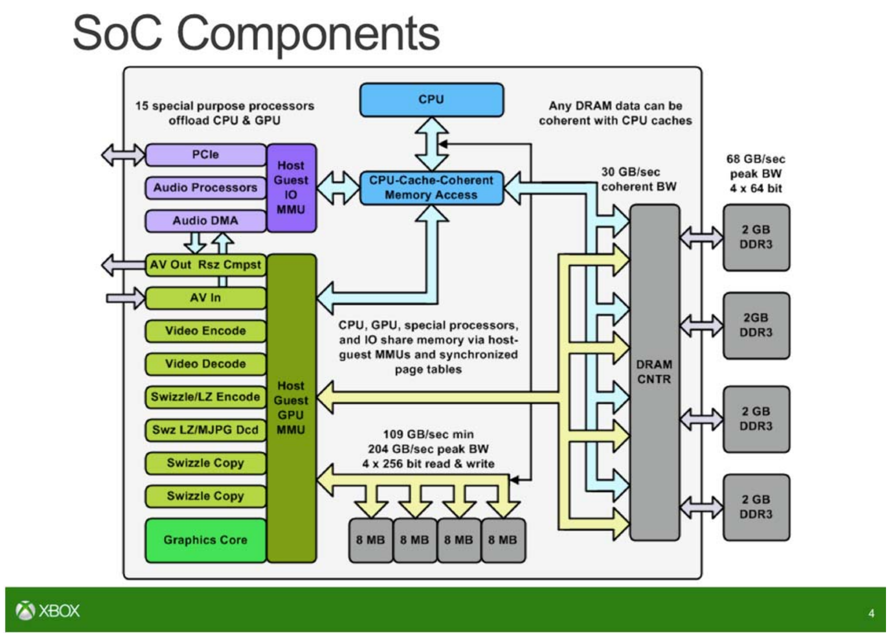


Figure A ventilator system design

Contemporary Embedded Systems (Examples, cont…)

Popular video game consoles…



SoC Components
15 special purpose processors offload CPU & GPU
CPU
Any DRAM data can be coherent with CPU caches
PCIe
Audio Processors
Audio DMA
Host Guest IO MMU
CPU-Cache-Coherent Memory Access
30 GB/sec coherent BW
68 GB/sec peak BW 4 x 64 bit
AV Out Rsz Cmpst
AV In
Video Encode
Video Decode
Swizzle/LZ Encode
Swz LZ/MJPG Dcd
Swizzle Copy
Swizzle Copy
Graphics Core
Host Guest GPU MMU
CPU, GPU, special processors, and IO share memory via host-guest MMUs and synchronized page tables
109 GB/sec min 204 GB/sec peak BW 4 x 256 bit read & write
8 MB
8 MB
8 MB
8 MB
DRAM CNTR
2 GB DDR3
2GB DDR3
2 GB DDR3
2 GB DDR3
XBOX
4

Figure 1 Xbox One

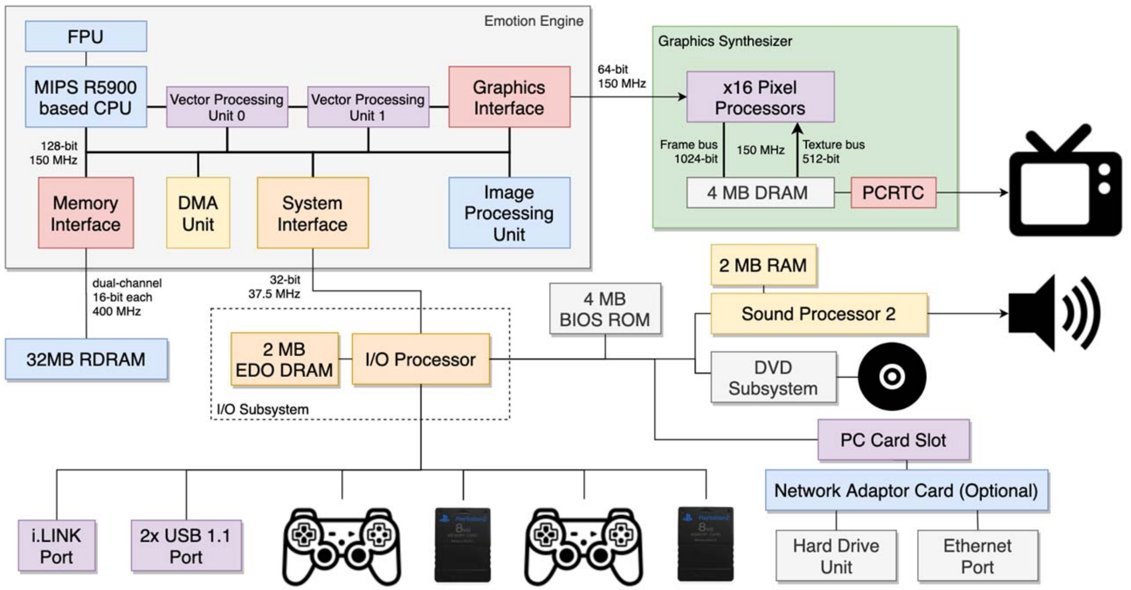


Figure 2 Play Station 2

Structure of an Embedded System

- Regardless of the function performed by an embedded system, the broadest view of its structure reveals two major, tightly coupled sets of components: a set of hardware components that include a central processing unit, typically in the form of a microcontroller; and a series of software programs, typically included as firmware, that give functionality to the hardware.

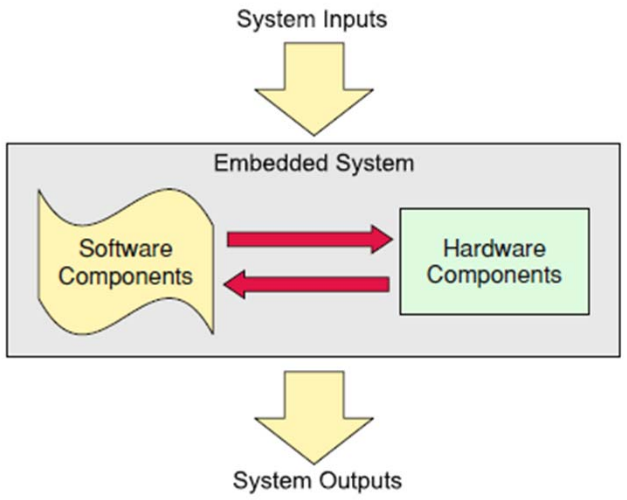


Figure General view of an embedded system

Structure of an Embedded System

- Typical inputs in an embedded system are process variables and parameters that arrive via sensors and input/output (I/O) ports.
- The outputs are in the form of control actions on system actuators or processed information for users or other subsystems within the application. In some instances, the exchange of input-output information occurs with users via some sort of user interface that might include keys and buttons, sensors, light emitting diodes (LEDs), liquid crystal displays (LCDs), and other types of display devices, depending on the application.

Hardware Components

When viewed from a general perspective, the hardware components of an embedded system include all the electronics necessary for the system to perform the function it was designed for.

Three core hardware components are essential in an embedded system:

1. The **CPU** executes software instructions to process the system inputs and to make the decisions that guide the system operation.
2. **Memory** stores programs and data necessary for system operation.
3. The **I/O ports** allows conveying signals between the CPU and the world external to it.

Hardware Components

In addition to core hardware components…

1. Communication ports for serial and/or parallel information
2. User interfaces to interact with humans
3. Sensors and electromechanical actuators
4. Data converters (Analog - to - Digital (ADC) and/or Digital - to - Analog (DAC))
5. Diagnostics and redundant components
6. System support components
7. Other sub-systems to enable functionality, that might include Application Specific Integrated Circuits (ASIC)

Software Components

The software components of an embedded system include all the programs necessary to give functionality to the system hardware.

The major components identified in a system software include:

1. **System Tasks**. The application software in embedded systems is divided into a set of smaller programs called *Tasks*. Each task handles a distinct action in the system and requires the use of specific *System Resources*. *(Application software such as Word, Excel, WinRAR etc.)*

2. **System Kernel**. The software component that handles the system resources in an embedded application is called the *Kernel*. System resources are all those components needed to serve tasks. These include memory, I/O devices, the CPU itself, and other hardware components *(Core of System Software such as Linux).*

3. **Services**. Tasks are served through *Service Routines*. A service routine is a piece of code that gives functionality to a system resource *(Device drivers such as Ethernet and Graphics card driver software)*.

Software Components

Following figure illustrates the software structure in a typical embedded system

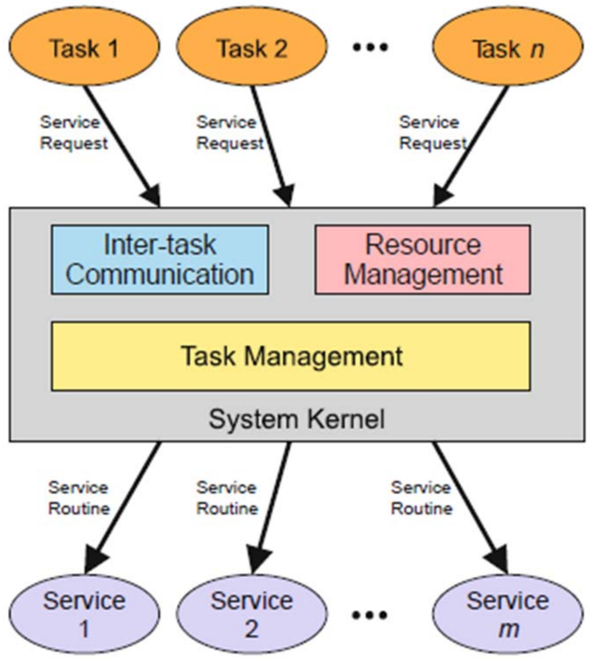


Figure Software structure in an embedded system

Classification of Embedded Systems

Small Embedded Systems

This class is typically centered around a single microcontroller chip that commands the whole application These systems are highly integrated, adding only a few analog components, sensors, actuators, and user-interface, as needed. These systems operate with minimal or no maintenance, are very low cost, and produced in mass quantities.

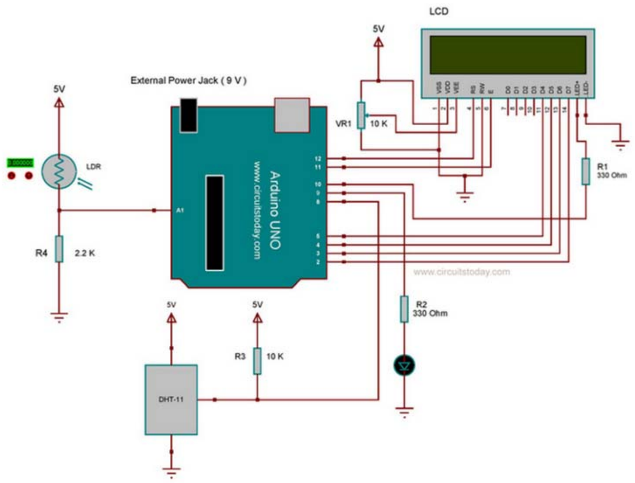


Figure LCD brightness control using Arduino UNO

Classification of Embedded Systems

Distributed Embedded Systems

These are typically board-level systems where, although robustness is not a critical issue, maintenance and updates are required, and include some means of systems diagnosis. Applications might require high-performance operations. Applications like video processors, video game controllers, data loggers, and network processors are examples of this categ

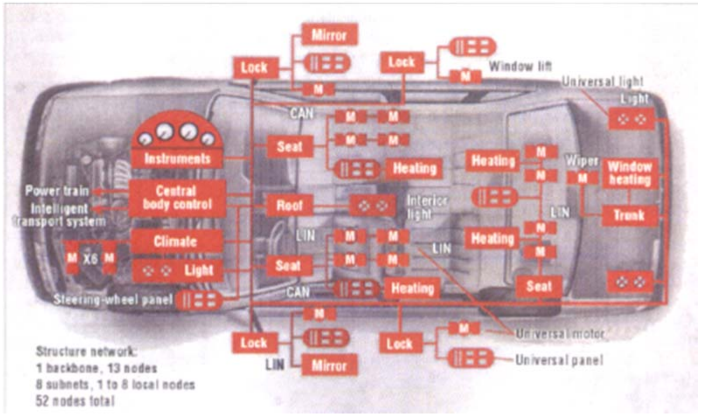


Fig.1 A car

Fig.2 A video game controller

Classification of Embedded Systems

High-Performance Embedded Systems

These systems usually require dedicated ASICS (**Application-Specific Integrated Circuits**), are typically distributed, might include DSPs **(Digital Signal Processors)** and FPGAs **(Field Programmable Gate Arrays)** as part of the basic hardware. They are produced in small quantities and their cost is very high. These are the type of embedded systems used in military and aerospace applications, such as flight controllers, missile guidance systems, and space craft navigation systems.

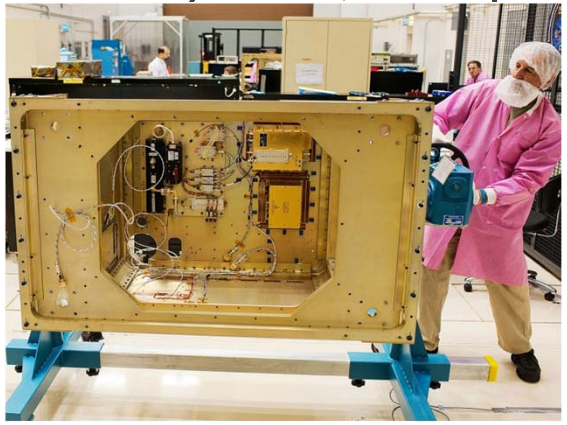


Figure From NASA's Space Communication and Navigation Program

The Life Cycle of Embedded Designs

Embedded systems have a finite lifespan throughout which they undergo different stages, going from system conception or birth to disposal and in many cases, rebirth. We call this the *Life Cycle of an Embedded System*. Five stages can be identified in the cycle: Conception or Birth, Design, Growth, Maturity, and Decline. Figure illustrates the sequence of these stages, and the critical phases that compose each stage.

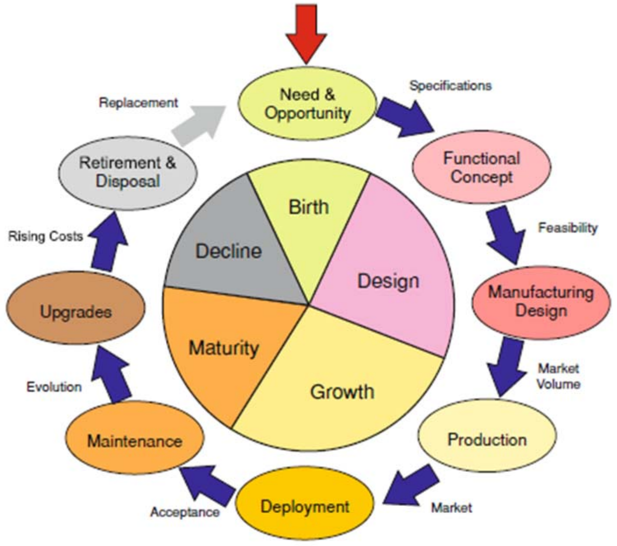
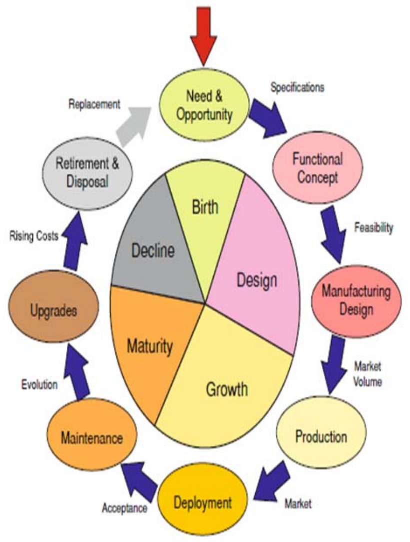


Figure Life cycle of an embedded system

The Life Cycle of Embedded Designs

- An embedded design is first conceived by the identification of a need to solve a particular problem, or the opportunity to apply embedded technology to an old problem.
- **In the design stage,** the system goes first through a phase of functional design, where proof-of-concept prototypes are developed. This phase allows to tune the system functionality to fit the application for which it was designed.
- **The growth stage** initiates with the production of the system to supply an estimated market demand.
- **In the maturity stage**, the product is kept functional and up to date. This involves providing technical support, periodic maintenance, and servicing.
- **In the decline stage**, system components become obsolete and difficult to obtain, driving the cost of maintenance, service, and upgrades to levels that approach or exceed those of replacing the whole system at once.



Design Constraints

These are systems with a high cost sensitivity to the resources included in a design due to the high volumes in which they are produced. Moreover, designs need to be completed, manufactured and launched in time to hit a market window to maximize product revenues. These constraints shape the design of embedded applications from beginning to end in their life cycle.

Functionality: The system ability to perform the function it was designed for.

Cost: The amount of resources needed to conceive, design, produce, maintain, and discard an embedded system.

Performance: The system ability to perform its function on time.

Size: Physical space taken by a system solution.

Power and Energy: Energy required by a system to perform its function.

Time to Market: The time it takes from system conception to deployment.

Maintainability: System ability to be kept functional for the longest of its mature life.