Learning Module 2 Homework

Problem 1

Describe at least three advantages of a centralized embedded system over a distributed embedded system. Also, describe at least three disadvantages of a centralized embedded system over a distributed embedded system.

1. Centralized Embedded System Advantages over Distributed
    a) Designer chooses the location of the central processor
    b) Design can choose powerful CPU to perform all required functions
    c) Developer can use simplified software configuration and model
2. Distributed Embedded System Advantages over Centralized
    a) The system is modular which allows for the development, verification, and support of modules
    b) A distributed embedded system is flexible which means that new functionality can be added by adding new modules or nodes.
    c) A distributed embedded system is more diagnosable which allows for the identification and isolation of system problems.

Problem 2

Read *Recent Advances in In-vehicle embedded systems* and write a one-page summary including your own opinion of the material presented in the paper.

An embedded system is typically a micro-computer system with one or few dedicated functions, usually with real-time computation constraints. Generally, designers have the choice of two main families of digital device technologies. The first family consists of microcontrollers and DSPs, based on a pure software platform. Considering the increase of complexity of embedded electronic architecture, the development of it has to integrate different hardware and software units provided by different vendors, which raises the question of "composability". And the issue of functional design to the implementation perspective and back to integration and acceptance testing on vehicle level.

Automobile manufacturers, suppliers, and tool developers jointly develop an open and standardized automotive software architecture–AUTOSAR (AUTomotive Open System ARchitecture), with the objective of creating and establishing open standards for automotive E/E (Electrics/Electronics) architectures that will provide a basic infrastructure to assist with developing vehicular software, user interfaces, and management for all application domains. This includes the standardization of basic systems functions, scalability to different vehicle and platform variants, transferability throughout the network, integration from multiple suppliers, maintainability throughout the entire product life-cycle, and software updates and upgrades over the vehicle's lifetime as some of the key goals.

Designers need to define, evaluate, and choose car electronic architectures years in advance, but at that time the functions they will support are not completely known. Many automotive applications, including most of those developed for active safety and chassis systems, must comply with hard real-time deadlines and are also sensitive to the average latency of the end-to-end computations from sensors to actuators. Worst case analysis based on schedulability theory allows computing the contribution of tasks and messages to end-to-end latencies and provides the architecture designer with a set of values (one for each end-to-end path) on which he/she can check correctness of an architecture solution.

What's more, several new attracting features such as higher levels of parallelism are brought to the designers by multicore ECUs, which ease the respect of the safety requirements such as the ISO 26262 and the implementation of other automotive use-cases. With multiple CPUs, an ECU is turned into a highly integrated "networked system" microcosm, in which there exist complex interdependencies among those CPUs due to the use of shared resources even in partitioned scheduling. In this trend of upgrading to multicore ECUs, how to reuse the previous software generations and configurations becomes a major concern of automotive suppliers and manufacturers, as property changes can be costly involving many different departments and companies.

Problem 3

Read Embedded systems overhaul and write a one-page summary including your own opinion of the material presented in the paper.

The IBM Institute for Business Value has performed an extensive analysis of the automotive industry, identifying industry challenges and drivers of change as well as a road map to the future to help companies address the issues created by the quick evolution and growth of embedded systems. Increased competition is the top market factor that concerns automotive industry CEOs, outpacing the average across all industries by 12.1% Embedded systems provide features that are important to customers which is critical for competitiveness – design has always been important, but functional differentiation is on the rise. Upper middle class American needs are emerging based on personalized needs. Reuse of parts, subsystems, designs an d architectures around embedded software is a common goal for OEMs. However, engineers often find themselves reinventing the wheel because it takes too long to find what they need – the result of a hardware- driven approach to design and the absence of a modular strategy to promote commonality of components. Lastly, automotive software and system engineering skill sets are not mature enough to support the growing requirements for complex software.
Magna Vectrics, a newly founded division of auto parts supplier Magna International, is working on leading-edge technologies to support the implementation of "smart" air bags. The following questions are designed to help automotive executives assess their current approach to embedded systems and begin to understand where improvements may be necessary .

- How do your software and electronics-based features contribute to the positive perception of your brand?
- What percentage of your warranty costs are due to electronic system problems? • Do you have a business model in place to buy/sell software as a product?
- Are you working in a team-oriented, collaborative approach with your software development suppliers, providing access for everyone to one model for test cases and requirements?
- How much of your software code is transferable across programs, brands and OEMs without changes?
- Are you buying to build an architecture instead of buying to satisfy requirements?
- Are your software development processes as mature as your mechanical design processes?
- How much time do your development engineers spend transferring parameters from one tool to another?