Embedded & Real-Time Systems EEP522A (2023)

Andrew N. Sloss

October 29, 2022

Abstract

A modern introduction to Embedded & Real-Time Systems. The course length is ten weeks, i.e., 1-quarter. The course is about *exploration*. The goal is that the students will appreciate Embedded & Real-Time Systems by learning how to characterize hardware and software. The characterization will involve being introduced to a hardware platform, exploring real-time requirements, computation limits, analysis of different scheduling algorithms, and memory configuration. In addition to characterization, some more complex subjects in Embedded & Real-Time Systems will be explored, including power management, reliability, safety-critical systems, and simulation. Upon completing the program, students will have a firm understanding of real-world issues and characterization methods and techniques for Embedded & Real-Time Systems. The course is 100% project-based with a presentation.

1 Recommended book

- **Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux**
  - *1st Edition*

2 Prerequisites

The course assumes a basic understanding of Computer Science and Electrical Engineering terms, programming language fundamentals, and fundamental electronics. The overarching aim of the study is to provide a set of useful problem-solving tools for Embedded Systems.

3 Target board

This year's target board will be the *Raspberry Pi 4* with 1GB of RAM memory. All students will receive a board, SD card, power supply, and heat sinks.

4 Outline

The course has two flows. One flow is the exploration (the coursework), and the second flow covers a set of in-depth topics. The investigation is about characterizing a development board, i.e., determining the limitations and capabilities by attempting to build something. The second flow covers the following specific topics.

1. Introduction & Characterization
2. Architecture, Fundamentals
3. Architecture, Memory
4. System level, Power management
5. System level, Critical Systems I - Reliability

6. System level, Critical Systems II - Safety-Critical
7. System level, Simulation
8. Future, What happens next in Embedded?

At a high level, the course divides into **Architecture** and **System level**. Architecture deals with processor design and memory complexity, e.g., caches, pipeline, instructions, etc. The System-level covers more general areas of embedded, e.g., power management, critical systems, co-design, etc. Finally, a future discussion looks at what happens next in Embedded Systems.

We will discuss the Scientific Method & Causality as it pertains to Embedded Systems.

5 Course structure

Each week will comprise an interactive session, where new developments will come up, and students are encouraged to ask questions. This year's plans include recording the topics, and students are encouraged to watch, investigate further, and question where appropriate.

6 Course work

The coursework involves three activities plus a presentation. The first activity is to initialize an embedded system board, characterize the board, and finally build a project with the knowledge gained. This exploration includes getting a device to boot as an embedded system, setting up communication channels, and attaching sensors (or actuators).

Characterization is about determining the theoretical board capabilities and comparing those with practical experience.

7 Lessons

7.1 Lesson: Introduction & Characterization

Introduce Embedded & Real-Time Systems subject, as well as hardware-software characterization. Characterization involves determining whether a target board is adequate for a specific application.

Students should be able to provide the correct hardware-software configuration. This lesson will introduce Embedded & Real-time systems, basic system design, exceptions and handling exceptions (including interrupts), and the type of kernel/executives required for a specific solution. The later part of the lesson will introduce Real-Time and the different methods to determine whether a problem can be schedulable through Rate Monotonic (RMS) or Earliest Deadline First scheduling (EDF).

7.1.1 Structure

1. Introduction to Embedded & Real-Time Systems, and the Characterization Theme
2. Applications, Tasks, Threads, and Processes
3. Exceptions, Interrupts, Stacks, Context Switch, Ready-List, and Scheduler
4. Deterministic (Real-Time) and non-deterministic behavior
5. System Level Characterization
6. Scheduling algorithms
7. Re-entrant/non-re-entrant kernel
8. Nested and non-nested interrupt handling
9. Introduce scheduling algorithms
   (a) Shortest-job-first

   (b) First-come-first-serve
   (c) Round-robin
   (d) Priority-scheduling

10. Introduce real-time scheduling design
   (a) Ask the question, is the system schedulable?
   (b) Rate Monotonic scheduling
   (c) Earliest Deadline First Scheduling

7.2 Lesson: Fundamentals

The fundamentals lesson will explain two critical areas in Embedded Systems: the boot/initialization process and the constraints on development tools and environments for Embedded & Real-Time systems.

Introduce the hardware boot process. The boot process will involve reviewing the order and the specific hardware pieces required to initialize an Operating System. On the software side, students will be potentially re-introduced to compilers, debuggers, and, more specifically, introduced to hardware-assist debugging/trace for Embedded & Real-Time development.

7.2.1 Structure

1. Cover a complete boot process of a simple device
2. Revisit compilers, debuggers, and architectures
3. Cover compiler optimizations
4. Hardware assist tools
   (a) JTAG Probe
   (b) Hardware trace
   (c) Logic Analyzers
   (d) Oscilloscopes

7.3 Lesson: Memory

Provide an overview of caches and introduce the different forms of memory management concerning Embedded & Real-Time Systems. Covering simple micro-controllers to the more advanced symmetric multi-core systems.

Introduce the complexities of memory systems. After two lectures, students will have a basic understanding of caches and, in particular, how different cache policies affect the overall system performance. Intertwined is the need to understand the other memory management techniques - linear, protected, and virtual memory. Students will have a good understanding of the trade-offs of using different memory management methods.

7.3.1 Structure

1. Memory layout
2. Caches Revisited
   (a) Cache lines
   (b) Cache lockdown
   (c) Virtual and Physical caches
3. Memory Protection

   (a) Protection regions

4. Memory Management Unit
   (a) TLB – software and hardware operations
   (b) Virtual Memory
   (c) Demand paging
   (d) Loading and unloading applications
   (e) Shared resources
5. Messaging Passing
   (a) Mailboxes (revisited with connection with memory design)
6. Direct Memory Access (DMA)
7. Brief overview of Virtualization, Containers and Serverless

7.4 Lesson: Power Management

Introduce the complexity of power management at the hardware level. Power management is essential for Internet-of-Things, Mobile devices, and tethered devices.

This lesson introduces power management as a subject. Explains the difference between power and energy efficiency equations and why understanding frequency is essential. Introduce mobile phone power demands, different techniques for saving power consumption, and some battery characteristics. And how power management covers everything from tethered to untethered devices.

7.4.1 Structure

1. Introduce power management
   (a) Tether and non-tethered (mobile)
   (b) Power state machine
2. Mobile Power Management
3. Introduce the Power and Energy equations
4. Processor features that help with power control
   (a) Voltage scaling
   (b) Frequency scaling
   (c) Wait-on-interrupt
5. Peripherals
   (a) Ethernet
   (b) LCD
   (c) Hard drives
6. Basic Battery Characteristics
   (a) Charging/discharging of batteries
   (b) History effects
7. Power Transmission
8. Power Harvesting
9. Discussion of the complexity of Power Management

7.5 Lesson: Critical Systems I Reliability

Introduce Reliability for Embedded Real-Time Systems and explain the difficulties hardware aging incurs.

Cover the exciting areas of why technology fails. The lesson introduces reliability as a subject. The class will describe Occam's razor, hardware aging, and the fundamental reliability equations. Explain different methods of recovery once a deadlock has occurred. Students will be aware of different embedded designs that can help reliability, such as voting, watchdog timers, and resetting.

7.5.1 Structure

1. Introduce the subject of reliability and its relationship with Safety-Critical
2. Explain Occam's Razor
3. Hardware aging and the bathtub failure curve
4. Discuss why systems fail.
5. Define the terms SLOC, MTTF, MTTR, and MTBF
6. Availability (Uptime)
7. Fault Tolerance
8. Revisit interrupt handling
9. Describe deadlocks
   (a) Deadlock identification
   (b) Deadlock recovery
   (c) Priority inversion
10. Hardware assistance
    (a) Watchdog timers (revisit)
11. Basic reset

7.6 Lesson: Critical System II Safety-Critical

Introduce the aspects of safety-critical for Embedded & Real-Time Systems.

The lesson introduces Safety-Critical and the consequences of failure. Safety-critical includes having a comprehensive understanding of the philosophy behind safety-critical and the consequences of bypassing the philosophy. Understand the meaning of the essential terms and the different methods to design a safety-critical system. The lesson will cover DO-178B as an example of a Safety-Critical standard. Also, MISRA-C as a programming standard for automotive.

7.6.1 Structure

1. Introduce the concept of safety-critical
2. Describe security, integrity, vulnerability, threat and risk
3. Complexity analysis
4. Consistency checking
5. Instrumenting code
6. Writing code to check for stack overflow and underflow
7. Redundancy (Byzantine algorithms)
8. MISRA C Standard
9. Example: safety-critical standard DO-178B

7.6.2 Lesson: Simulation

Introduce the aspects of Simulation for Embedded & Real-Time Systems and how a simulation is an essential tool for design and development.

Introduce and explore the concept of simulation, including the differences between simulation and emulation of systems. Be able to define cycle-accurate, cycle-approximate, and instruction-level modeling. Define the advantages and scope of simulation and hence discuss the problems that simulation introduces.

7.6.3 Structure

1. Introduce simulation modeling
   (a) Why?
   (b) How?
   (c) Cost?
2. Different types of simulation models
   (a) Cycle accurate modeling
   (b) Cycle approximate modeling
   (c) Instruction level modeling
3. Co-design/Co-simulation

7.6.4 Lesson: What happens next in Embedded?

Embedded Systems, as a subject, is not static. This lesson looks at what drives new Embedded Systems, including development strategies. The class covers applying new technologies to standard Embedded Systems and how these technologies will affect future decision-making.

1. Future of Embedded Systems
   (a) Machine Learning: Evolutionary Algorithms, Neural Networks, and Fuzzy Logic
   (b) Discussion around Embedded Applications