ECE/CS 5785: Advanced Embedded Software

Credits and Contact Hours: 3.0 Credit Hours
15 weeks: Two 80-minute lectures + one 2-hour lab per week

Instructor’s Name: John Regehr

Text Book(s) and/or Required Material:

Catalog Description: This course is about designing and implementing reliable and efficient embedded software, with a bias toward whole-system issues. Students must be proficient in C programming, and complete a number of embedded programming projects in C. The course covers topics including embedded software architectures, digital signal processing, feedback control, real-time scheduling, verification and validation, embedded network protocols, and issues in creating safety-critical embedded systems.

Prerequisites:
- C- or better in ECE/CS 5780/6780: Embedded System Design; and
- Full major status in Computer Science or Computer Engineering

Designation: Elective

Contribution of Course to Meeting the Requirements of ABET Criterion 5: Engineering sciences and engineering design.

Specific Outcomes of Instruction: Students learn the following:
1. Hands-on programming experience with an embedded development board (the board varies; the 2014 version of this course uses the BeagleBone Black)
2. How to choose a processor, operating system, programming language, etc., for a new embedded project
3. How to rapidly acquire knowledge from a large pile of reference manuals
4. How “embedded C” differs from regular C code, and how to avoid pitfalls of embedded C programming
5. How embedded systems differ in fundamental ways from standard desktop computing systems
6. How to design and implement embedded systems software that reliably interfaces with hardware devices and with interrupt handlers
7. How to design and implement embedded system software that reliably meets time constraints
8. How to design and implement embedded system software that communicates with other computers

Relationship of the Course to the Program Outcomes:
(a) An ability to apply knowledge of mathematics, science, and engineering. The course requires students to reason about computer programs and their interaction with hardware platforms. An example is to verify that the peak memory usage of a system fits into the available storage. The importance of making good engineering tradeoffs (for example,
between different kinds of processors or different software architectures) is emphasized throughout the course.

(b) An ability to design and conduct experiments, as well as to analyze and interpret data, and debug and analyze software. Experiment design in a formal sense is not emphasized at all. However, system (hardware+software) debugging is of utmost importance in this course, and the relationship between debugging and the scientific method is specifically emphasized.

(c) An ability to design a system, component, process or software package to meet desired needs. This outcome is actually a good summary of the intended outcome of CS/ECE 5785. The entire point of the course is to design a HW/SW system that meets many conflicting objectives such as cost, size, power, reliability, and performance.

(d) An ability to function on multidisciplinary teams. Students complete projects, and are encouraged to mix EE and CS students on each team. This mixing is not enforced, but teams with mixed disciplines tend to have an easier time overall.

(e) An ability to identify, formulate, and solve engineering problems. As stated above, the entire purpose of this course is to understand, write down, and meet a collection of (usually) conflicting engineering objectives.

(f) An understanding of professional and ethical responsibility. Safety critical systems are discussed but the material is focused on how to create them, not on their broader interactions with society.

(g) An ability to communicate effectively. Typically, most exam questions are in essay form. No other written or oral communication is required.

(h) The broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context. The primary focus of this course is technical. More could certainly be done to address the role of safety critical systems in society.

(i) A recognition of the need for, and an ability to engage in life-long learning. The course explicitly encourages students to become self-sufficient in their acquisition of knowledge. The course lectures deliberately cover advanced topics, leaving the students to gather basic information from a large pile of technical documentation in order to complete the laboratory assignments. Lab assignments are presented as high-level goals, as opposed to detailed lists of steps to be performed.

(j) A knowledge of contemporary issues. The contemporary issues discussed are limited to issues such as hardware trends, software verification trends, etc.

(k) An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice. The course explicitly focuses on the large and increasing role of advanced tools and techniques in the creation of embedded systems.

Topics Covered in the Course:
ARM and ColdFire architectures; Tools and tool chains; Embedded C, Advanced C; Interrupts, inline assembly, intrinsic; MISRA C and other language subsets; RAM limits and embedded C extensions; Advanced interrupts; Debugging embedded systems; Advanced threads; Energy and power; Testing embedded systems; Safety critical systems; Embedded wired networks; CAN bus; Embedded wireless networks; Digital signal processing; DSPs; Real time intro and cyclic executives; Priority scheduling; Response time analysis; Real-time guarantees over CAN; Timing analysis – computing WCET; Feedback control – PID, PID tuning, and PID implementation