Course Title: ECE 3140 / CS 3420: Embedded Systems
Author: Prof. Nils E. Napp, ECE
Authorship or Revision Date: Nov. 14, 2023
Credit Hours: 4 hours
Catalog Description:

An introduction to the design of embedded systems, with an emphasis on understanding the interaction between hardware, software, and the physical world. Topics covered include assembly language programming, interrupts, I/O, concurrency management, scheduling, resource management, and real-time constraints.

Course Frequency:

Offered every Spring semester.

Prerequisites:

ECE 2300/ENGRD 2300 required. ECE 2400/ENGRD 2140 strongly recommended but not required.

Corequisites:

N/A

Student Preparation Summary:

Programming: Students need to be familiar with basic computer programming. In particular, the students in the course are expected to (learn and) write code in the C programming language.
Computer organization: This course assumes that the students are familiar with the basic concepts of computer hardware organization, including processor pipelines and memory, and understand how instructions execute on a processor. Under this assumption, the course will discuss hardware-software interfaces and low-level programming.

Textbook(s) and/or Other Required Materials:

- We will supply you a development board (FRDM KL46Z), you are responsible to return it and replace broken boards.
- No required textbook; course notes distributed via CMS/Canvas.
- References
  - ARM® v6-M Architecture Reference Manual
  - ARM® Cortex™-M0+ Devices Generic User Guide
  - E.W. Dijkstra, Cooperating Sequential Processes, EWD-123, 1968
  - C.A.R. Hoare, Monitors: An Operating System Structuring Concept, STAN-CS-73-401, 1973
  - C.L. Liu and J.W. Layland, Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, J. of the ACM, Vol. 20, No. 1, 1973
  - G. Buttazzo, Hard Real-Time Computer Systems: Predictable Scheduling Algorithms and Applications, 3rd Ed., Springer, 2011

Class and Laboratory Schedule:

Lectures: Each week will have two lecture and the slides will be posted online for review.
Discussion sections: One 75-min discussion section per week. These will be smaller group lectures that cover additional material and help with labs.
Labs: Open lab help scheduled most days.

Assignments, Exams and Projects:

Exams/Quizzes: There are weekly graded quizzes related to lectures.
Lecture "attendance": We will grade lecture attendance by participation in lecture quizzes.

Labs: Lab assignments on programming an embedded system platform are due roughly every two weeks. The final lab assignment is an open-ended project where student groups choose a topic and build an embedded system using the C language on the FRDM K46Z board. Students are expected to work as groups of two on the lab assignments.

Course Grading Scheme: 55% Labs, 10% Project, 25% Quizzes, 10% Lecture participation/quizzes

Details List of Topics Covered:

- Introduction to assembly language
- Cortex-M instruction set architecture
- Assembly support for high-level languages
- Input-output mechanisms
- Concurrency control: mutual exclusion, locks, semaphores, condition variables
- Embedded bus protocols
- Operating system scheduling
- Aperiodic real-time scheduling
- Periodic real-time scheduling
- Priority inversion

Student Outcomes [ABET]:

1. Understand the function, basic structure, and operation of assembly language, and its relationship to machine code as well as high-level programming.
2. Comprehend how embedded microprocessors communicate with the external world through analog as well as digital I/O, and engineer programmatic responses of such microprocessors to external stimuli using polling, interrupts, etc.
3. Be capable of constructing correct synchronization primitives in a concurrent environment, and of reasoning about these primitives in terms of correctness, progress, and fairness.
4. Understand the concept of process scheduling, real-time and otherwise, and its applicability to constructing complex embedded solutions.
5. Acquire experience in modern agile programming techniques.
6. Be able to articulate one's technical reasoning in a succinct, comprehensive, accurate, and clear manner, orally as well as in writing.
7. Learn to seek, appreciate, and incorporate everyone's points of view and contributions.

Academic Integrity:

Students are expected to abide by the Cornell University Code of Academic Integrity with work submitted for credit representing the student's own work. Discussion and collaboration on assignments is permitted and encouraged, but final work should represent the student's own understanding. Course materials posted are intellectual property belonging to the author. Students are not permitted to buy, sell, or publicly post, any course materials without the express permission of the instructor. Such unauthorized behavior will constitute academic misconduct.