Instructor

Instructor: Lawrence Lo (lawlo@uw.edu)
Assistant: Joe Schilz (joseph.schilz@gmail.com)

Preferred mode of contact: via the Canvas messaging tool.

Class Sessions

Monday evenings, 6-9pm (Pacific Time)

Classes are held in-person in Bellevue, and webcast via Zoom.

Address: 2445 140th Avenue N.E., Ste. B-100, Bellevue, WA 98005-1879

Zoom link: https://washington.zoom.us/my/embsys

Grading and Attendance

This is a pass/fail course that is dependent on performance *and* participation.

Classroom students must attend at least 60% of sessions, in-person, to be eligible to pass the course.

Online students must participate online, or in person, for at least 60% of sessions to be eligible to pass the course.

All students must complete a minimum of 80% on total assignments to be eligible to pass the course. Individual assignments will have prescribed weights and due dates (see below).

Textbooks

The main textbook for this course is:

Samek, Miro. Practical UML Statecharts in C/C++, 2nd Edition. Newnes. 2008. (PDF available at http://www.state-machine.com/psicc2.)

In addition we will use the following books as reference:

Gomaa, Hassan. Real-Time Software Design for Embedded Systems. 1st Edition. Cambridge University Press. 2016. (Online version available at UW library.)

Yiu, Joseph. The definitive guide to ARM Cortex-M3 and Cortex-M4 processors. Newnes/Elsevier. 2014. (Online version available at UW library.)

Course Summary

Students continue to develop the skills learned in the first two courses, while learning to determine the limitations of hardware and software in an embedded system (real-time requirements, computation limits, etc.), analyze the different scheduling algorithms and optimize the usage of memory. In addition, students learn how to develop and integrate optimizations within a system and gain a detailed understanding of power management, reliability, safety critical and simulation. Upon completion of the program, students have a firm understanding of real world issues and design/optimization methods and techniques.

Students may have an opportunity for interactive work, forming collaborative groups to solve problems. This is a more advanced class in which design issues and concepts will be discussed.

Lesson Plan (Subject to Change):

Module 1 (4/8)

Introduction to software design.

Eclipse/GNU toolchain setup.

Module 2 (4/15)

Essential C++ for embedded systems.

Introduction to course project.

Module 3 (4/22)

State-machine design and implementation.

Introduction to QP statechart framework.

Module 4 (4/29)

Object-oriented design.

Application framework on top of QP.

Module 5 (5/6)

Optimization of embedded systems with interrupts and DMA.

Design and optimization of UART driver.

Module 6 (5/13)

Command console.

Integration with WiFi expansion board (SPWF04) using UART.

Module 7 (5/20)

Integration with MEMS sensor expansion board (IKS01A1) using I2C.

Integration with TFT display expansion board using SPI.

Module 8 (6/3)

Debugging, profiling and optimization techniques.

Error handling strategies and robustness.

Module 9 (6/10)

Reactive software architecture.

Putting it together.

Module 10 (6/17)

Project presentation.

<u>Assignment Plan (Subject to Change):</u>

Assignment 1 (due 4/15, 10%)

Development environment setup. Discussion.

Assignment 2 (due 4/22, 10%)

C++ classes for LED patterns and a simple LED controller.

Assignment 3 (due 4/29, 10%)

LED controller state machine.

Assignment 4 (due 5/6, 10%)

Traffic light controller.

Assignment 5 (due 5/13, 10%)

Button input driver.

Assignment 6 (due 5/20, 10%)

IoT.

Final Project (due 6/15, 40%)