COMP 4550 – Real-Time Systems

Course Description

Calendar entry

An introduction to the theory and practice of real-time systems. Topics include the design of real-time systems, scheduling, event based processing, and real-time control. This course may not be held for credit if a student has previously completed both of ECE 4240 and ECE 3760. Prerequisites: COMP 3430 and COMP 3370.

General Course Description

Computers are digital. The world is analog and obeys the laws of Physics. Within a completely digital system the software developer is in full control of a simulation. As soon as the software starts interacting with the real-world, the developer needs to consider the analog constraints of the system. That's where real-time systems come in: applying algorithms and design techniques that guarantee the timing of analog interactions so we can correctly sense and manipulate the real-world. From simple interactions like keyboard and mouse input to communications to controlling the motion of a vehicle, real-time systems give us the tools we need to code at the interface between analog and digital worlds.

Detailed Prerequisites

Before entering this course, a student should be able to:

- (3430) Identify the critical section(s) requiring mutually exclusive access in a piece of code that will be run concurrently. Propose solutions that avoid and/or prevent deadlock.
- (2280) Explain how device driver code interacts with hardware to provide I/O functionality.
- (2280) Identify inappropriate coding within an ISR and explain why system/device driver implementations need to avoid such coding pitfalls.
- (3370) Use interrupts and memory mapped I/O to implement the transfer data between a device and memory independent of the CPU.
- (2280) Use logical operations to mask a binary value's individual bits to 0 or 1 and compare bits between binary values.

Course Goals

By the end of this course students will:

- Write code that sense aspects of the real-world.
- Write code that manipulate aspects of the real-world.

- Design and implement applications that make real-time decisions based on current goals and feedback.
- Determine the prioritization and scheduling of several real-time tasks concurrently executing within and outside interrupt service routines.
- Implement core kernel functionality needed to support real-time tasks.

Learning Outcomes

Event Driven Input Processing

Students should be able to:

1. Select sampling frequencies based on input hardware characteristics.
2. Implement hysteresis algorithms to filter unstable digital inputs (e.g., buttons).
3. Design and implement finite state machines that include time passed as an input for state transitions.
4. Implement time-based input capture to extract measurements such as signal frequency, time between bits, and RPM.
5. Implement analog input capture to extract measurements such as light intensity and temperature.
6. Design and implement applications that determine actions/output changes based on these input sources.

Controlling External Devices

Students should be able to:

1. Explain how voltage levels can be used control analog outputs such as the RPM of a motor and the strength of a magnet.
2. Implement pulse-width modulation (PWM) to vary the voltage level applied to analog device.
3. Apply PWM to the generation of signals such as a sine wave and a stream of bits on a network.
4. Compare and contrast open- versus closed-loop control algorithms.
5. Implement a Proportional, Integral, Derivative (PID) algorithm to ensure correctly timed control of an analog output based on command and feedback input changes.

Real-time Scheduling

Students should be able to:

1. Compare and contrast soft versus hard real-time deadlines.
2. Compare and contrast periodic versus aperiodic tasks.
3. Use a hardware timer to determine the exact amount of time that has passed between two events.

4. Use a hardware timer to implement a system heartbeat as the basis for determining the next task to perform.
5. Analyze the interplay between interrupts to determine the priorities required to ensure deadlines are met, critical sections are protected, and issues such as priority inversion and deadlock are prevented.
6. Perform rate monotonic and deadline monotonic analysis on a set of tasks to determine if a real-time schedule will meet all task deadlines.
7. Explain how dynamic algorithms such as Earliest Deadline First and Least Slack Time schedule real-time tasks.