CSE 466 – Software for Embedded Systems

- **Instructor:**
  - Bruce Hemingway
  - CSE 464, Hours: MW 10:30-11:30
  - bruceh@cs.washington.edu

- **Teaching Assistants:**
  - Waylon Brunette
    - CSE 603, Hours: TTh 2:30-4:30
    - wb@cs.washington.edu
  - Trevor Hamilton
    - trevah@cs.washington.edu
    - CSE 603, Hours: TTh 2:30-4:30

---

CSE 466 – Software for Embedded Systems

- **Class Meeting Times and Location:**
  - Lectures: EE 045, MWF 9:30-10:20
  - Lab: CSE 003, T – Section A, 2:30-5:20
  - Th – Section B, 2:30-5:20

- **Exams**
  - I: Wednesday, 27 April, EE 045, 9:30-10:20
  - II: Friday, 27 May, EE 045, 9:30-10:20
  - Final demo: Wednesday, 8 June, CSE Atrium, 8:30-10:20

---

Embedded systems

- Some embedded systems include an operating system, but many are so small and specialized that the entire logic can be implemented as a single program.

---

Embedded system – from the web

- **Definitions**
  - A device not independently programmable by the user.
  - Specialized computing devices that are not deployed as general purpose computers.
  - A specialized computer system which is dedicated to a specific task.
  - An embedded system is preprogrammed to perform a narrow range of functions with minimal end user or operator intervention.

---

Embedded system – from the web

- **Examples**
  - Virtually all appliances that have a digital interface – watches, microwaves, VCRs, cars – utilize embedded systems.
  - A computer system dedicated to controlling some non-computing hardware, like a washing machine, a car engine or a missile.
  - Examples of embedded systems are medical equipment and manufacturing equipment.
  - While most consumers aren't aware that they exist, they are extremely common, ranging from industrial systems to VCRs and many net devices.
**What is an embedded system?**

- Different than a desktop system
  - Fixed or semi-fixed functionality (not user programmable)
  - Different human interfaces than screen, keyboard, mouse, audio
  - Usually has sensors and actuators for interface to physical world
  - May have stringent real-time requirements
- It may:
  - Replace discrete logic circuits
  - Replace analog circuits
  - Provide feature implementation path
  - Make maintenance easier
  - Protect intellectual property
  - Improve mechanical performance

**What do these differences imply?**

- Less emphasis on
  - Graphical user interface
  - Dynamic linking and loading
  - Virtual memory, protection modes
  - Disk and file systems
  - Processes
- More emphasis on
  - Real-time support, interrupts (very small OS, if we’re lucky)
  - Tasks (threads)
  - Task communication primitives
  - General purpose input/output
  - Analog/digital/digital-analog converters
  - Timers
  - Event capture
  - Pulse-width modulation
  - Built-in communication protocols

**What is an embedded system? (cont’d)**

- Figures of merit for embedded systems
  - Reliability – it should never crash
  - Safety – controls things that move and can harm/kill a person
  - Power consumption – may run on limited power supply
  - Cost – engineering cost, manufacturing cost, schedule tradeoffs
  - Product life cycle – maintainability, upgradeability, serviceability
  - Performance – real-time requirements, power budget

**Capacity**

- Assume:
  - 8 MHz processor @ one instruction/cycle
  - Assume fan runs between 30Hz and 60Hz
  - Assume 256ms period on speed control PWM, with 1ms resolution.
- What percent of the available cycles are used for the temperature controller?
  - [total instructions in one second] / (8MInstr/sec)
- How much RAM do you need?
- How much ROM do you need?

**Example: a temperature controller**

**Resource analysis of temp controller**

<table>
<thead>
<tr>
<th>Task: Tachometer (external interrupt)</th>
<th>Task: TempControl (periodic, soft constraint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>now = getTime();</td>
<td>if (Temp &gt; setpoint) Thi++;</td>
</tr>
<tr>
<td>period = then - now; overflow?</td>
<td>if (Temp &lt; setpoint) Thi--;</td>
</tr>
<tr>
<td>then = now;</td>
<td>if (period &lt; min</td>
</tr>
<tr>
<td>return;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task: FanPWM (periodic, hard constraint)</th>
<th>Task: Main</th>
</tr>
</thead>
<tbody>
<tr>
<td>count++;</td>
<td>THi = 0;</td>
</tr>
<tr>
<td>if (count == 0) GP0 = 1;</td>
<td>setup timer for 1ms interrupt;</td>
</tr>
<tr>
<td>if (count &gt; Thi) GP0 = 0;</td>
<td>setup timer for 100ms interrupt;</td>
</tr>
<tr>
<td>return;</td>
<td>while (1) ;</td>
</tr>
</tbody>
</table>

**Total Instructions/Sec = 8260**, at SMIPS, that’s only 0.1% utilization! Other resources? local variables, stack
Class logistics – see course web

- [http://www.cs.washington.edu/education/courses/cse466/05sp/](http://www.cs.washington.edu/education/courses/cse466/05sp/)
- Class structure
- Business matters
- Grading
- Syllabus
- What we’ll be doing

Class structure

- **Lecture**
  - Closely linked to laboratory assignments
  - Cover main concepts, introduced laboratory problems
- **Lab**
  - Implementation of two projects
  - Lab reports due prior to 30 minutes of start of next lab section
- **Exams**
  - Two, based on lecture, lab, and reading
- **Final demo**
  - During scheduled final time – participation required
- **Reading and textbook**
  - Title: *Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers*
  - Author: Tammy Nørgaard
  - CoursePak: CSE466 – Communications bldg, Rm B-042

Business Matters

- Lecture slides will be on line after class (links in several places)
- Get the CoursePak for CSE466 ($30.90, Communications B-042)
- Random lab partner assignments, changed mid-quarter
- Sign up for CSE466 mailing list

Grading

- **Lab reports:**
  - Demonstration(s) required
  - Brief answers to questions embedded in assignment
  - Sometimes hand-in code
  - Do with your partner
- **Distribution:**
  - Labs: 40%
  - Exams: 30% (27 April and 27 May)
  - Demo: 10%
  - Class Participation: 20%

CSE466 Lab Projects

- Two multi-week projects
  - Four lab assignments each
  - Different lab partners
- **First project**
  - Familiarize with microcontroller
  - Learn how to interface various devices
  - Testing and debugging
  - Basic communication between chips and between chip and PC
- **Second project**
  - Wireless communication
  - Embedded operating system
  - Real-time issues
  - Testing and debugging
  - Emergent behavior of a collection of devices

CSE466 Lab Projects (cont’d)

- **Project 1 – USB device**
  - Platform: ATmega16 AVR microcontroller
  - Accelerometer and push-button used to control a tri-color LED
  - Connects sensor and actuator to PC through USB port
- **Color Controller**
  - Tilt the controller to change color mix
  - Accelerometer senses movement of ball – tilting
  - Push button activates sensing
CSE466 Lab Projects (cont’d)

- Project 2 – Ad hoc wireless network (“flock”)
  - Platform: UC Berkeley wireless sensor nodes (UCB “motes”)
  - Sound generation coordinated with neighbors and time of day
  - Emergent behavior between different nodes
  - **Flock-III**
    - Install in Allen Center atrium for pleasing auditory display
    - Modify birdsong using techniques from Evolutionary Computation
    - Generate sound using Yamaha FM synthesis Ring-tone IC
    - Birds sing, have color, sense light