CSE 474 – Introduction to Embedded Systems

- **Instructor:**
  - Bruce Hemingway
  - CSE 464, Office Hours: 11:00-12:00 p.m., Tuesday, Thursday
  - or whenever the door is open
  - bruceh@cs.washington.edu

- **Teaching Assistants:**
  - Cody Ohlsen, Kendall Lowrey and Ying-Chao (Tony) Tung

CSE 474 – Software for Embedded Systems

- **Class Meeting Times and Location:**
  - Lectures: EEB 037, TTh 2:30-4:20
  - Labs: EEB 345, 24/7 access

- **Exams**
  - Midterm: Tuesday, Feb. 7, EEB 037, 2:30-4:20
  - Final: Tuesday, March 14, 2017, 4:30-6:20 pm, EEB 037
CSE 474 – Software for Embedded Systems

- Grading Policy
  - There will be two exams, as shown on the class schedule.
  - Lab reports: Demo usually required, sometimes questions

- Ratios:
  - Lab: 40%
  - Exams total: 40%
  - Class Participation: 20%

Recommended Textbook (not required):

The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors, Third Edition

By Joseph Yiu

Newnes; 3 edition (November 1, 2013)

U Bookstore doesn’t have it.
Other Textbook (not required):

- **Embedded Systems: A Contemporary Design Tool**
  Paperback – 2009
  by [James K Peckol](#)

U Bookstore has used hardback.

---

**Embedded systems**

[Images of various embedded systems devices]
Embedded systems

What are Embedded Systems?

- Anything that uses a microprocessor but isn't a general-purpose computer
  - Smartphones
  - Set-top boxes
  - Televisions
  - Video Games
  - Refrigerators
  - Cars
  - Planes
  - Elevators
  - Remote Controls
  - Alarm Systems

- The user "sees" a smart (special-purpose) system as opposed to the computer inside the system
  - "how does it do that?"
  - "it has a computer inside it!"
  - "oh! BTW, it does not or cannot run Windows or MacOS!"

- the end-user typically does not or cannot modify or upgrade the internals
What Are You Going to Learn?

- **Hardware**
  - I/O, memory, busses, devices, control logic, interfacing hardware to software
- **Software**
  - Lots of C and assembly, device drivers, low level real-time issues, scheduling,
  - Concurrency: interrupts
- **Software/Hardware interactions**
  - Where is the best place to put functionality hardware or software?
  - What are the costs:
    - performance,
    - memory requirements (RAM and/or FLASH ROM)
- **Integration of hardware and software courses**
  - Programming, logic design, architecture,
  - Algorithms, mathematics and *common sense*

Where Could You End Up?

- **Automotive systems**
  - perhaps designing and developing “drive-by-wire” systems
  - self-driving vehicles
- **Telecommunications**
- **Consumer electronics**
  - cellular phones, MP3 devices, integrated cellular/tablet/kitchen sink
  - Set-top boxes and HDTV
  - Home appliances
  - Internet appliances
    - your washer will be on the internet more than you are!
- **Defense and weapon systems**
- **Process control**
  - gasoline processing, chemical refinement
- **Automated manufacturing**
  - Supervisory Control and Data Acquisition (SCADA)
- **Space applications**
  - Satellite communications
Goals of the Course

- **High-Level Goals**
  1. Understand the scientific principles and concepts behind embedded systems, and
  2. Obtain hands-on experience in programming embedded systems.

**By the end of the course, you should be able to**
- Understand the "big ideas" in embedded systems
- Obtain direct hands-on experience on both hardware and software elements commonly used in embedded system design.
- Understand the basics of embedded system application concepts such as signal processing and feedback control
- Understand, and be able to discuss and communicate intelligently about
  - embedded processor architecture and programming
  - I/O and device driver interfaces to embedded processors with networks, multimedia cards and disk drives
  - OS primitives for concurrency, timeouts, scheduling, communication and synchronization

The Big Ideas

- **HW/SW Boundary**
- **Non processor centric view of architecture**
- **Bowels of the operating software**
  - specifically, basic real-time operation with interrupts
  - Concurrency
- **Real-world design**
  - performance vs. cost tradeoffs
- **Analyzability**
  - how do you “know” that your drive-by-wire system will function correctly?
- **Application-level techniques**
  - signal processing, control theory
  - semaphores, locks, atomic sections
What is an Embedded System?

• Computer purchased as part of some *other* piece of equipment
  – Typically dedicated software (may be user customizable)
  – Often replaces previously electromechanical components
  – Often no “real” keyboard
  – Often limited display or no general purpose display device
• But, every system is unique there are always exceptions

CPU: An All-Too-Common View of Computing

• Measured by:
  – Performance
An Advanced Computer Engineer's View

- Measured by: Performance
  - Compilers matter too...

An Enlightened Computer Engineer's View

- Measured by: Performance, Cost
  Compilers & OS matters
An Embedded Computer Designer's View

- Measured by: Cost, I/O connections, Memory Size, Performance

![Diagram of a computer system]

An Embedded Control System Designer's View

- Measured by:
  Cost, Time to market, Cost, Functionality, Cost & Cost.

![Diagram of a control system]
A Customer View

- Reduced Cost
- Increased Functionality
- Improved Performance
- Increased Overall Dependability

Why Are Embedded Systems Different?

Four General Categories of Embedded Systems

- General Computing
  - Applications similar to desktop computing, but in an embedded package
  - Video games, settop boxes, wearable computers, automatic tellers
  - Tablets, Phablets
- Control Systems
  - Closed loop feedback control of real time system
  - Vehicle engines, chemical processes, nuclear power, flight control
- Signal Processing
  - Computations involving large data streams
  - Radar, Sonar, video compression
- Communication & Networking
  - Switching and information transmission
  - Telephone system, Internet
  - Wireless everything
Typical Embedded System Constraints

- Small Size, Low Weight
  - Handheld electronics
  - Transportation applications weight costs money
- Low Power
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- Harsh environment
  - Heat, vibration, shock
  - Power fluctuations, RF interference, lightning
  - Water, corrosion, physical abuse
- Safety critical operation
  - Must function correctly
  - Must not function incorrectly
- Extreme cost sensitivity
  - $.05 adds up over 1,000,000 units

Embedded System Design World-View

A complex set of tradeoffs:

- Optimize for **more than just speed**
- Consider **more than just the computer**
- Take into account **more than just initial product design**

**Multi-Discipline**
- Electronic Hardware
- Software
- Mechanical Hardware
- Control Algorithms
- Humans
- Society/Institutions

**MultiPhase**
- Requirements
- Design
- Manufacturing
- Deployment
- Logistics
- Retirement

**MultiObjective**
- Dependability
- Affordability
- Safety
- Security
- Scalability
- Timeliness
Embedded System Designer Skill Set

• Appreciation for multidisciplinary nature of design
  – Both hardware & software skills
  – Understanding of engineering beyond digital logic
  – Ability to take a project from specification through production
• Communication & teamwork skills
  – Work with other disciplines, manufacturing, marketing
  – Work with customers to understand the real problem being solved
  – Make a good presentation; even better write "trade rag" articles
• And, by the way, technical skills too...
  – Low-level: Microcontrollers, FPGA/ASIC, assembly language, A/D, D/A
  – High-level: Object oriented Design, C/C++, Real Time Operating Systems
  – Meta-level: Creative solutions to highly constrained problems
  – Likely in the future: Unified Modeling Language, embedded networks

Class logistics – see course web

- [https://courses.cs.washington.edu/courses/cse474/17wi/](https://courses.cs.washington.edu/courses/cse474/17wi/)
- Class structure
- Business matters
- Grading
- Syllabus
- What we’ll be doing
Class structure

- Lecture
  - Closely linked to laboratory assignments
  - Cover main concepts, introduce laboratory problems
- Lab
  - Work leading to implementation of a final project
  - Lab reports due by end of week
- Exams
  - Two, based on lecture, lab, and datasheet reading
  - Open datasheets, open notes
- Final demo
  - During last class time – participation required

Business Matters

- Lecture notes will be on line after class (links on Calendar page)
- You pick lab partner assignments, or we will
- Sign up for CSE474 mailing list
Grading

- Lab reports:
  - Demonstration(s) required
  - Brief answers to questions embedded in assignment
  - Sometimes hand-in code
  - Do with your partner
  - Both build hardware

- Distribution:
  - Labs: 40%
  - Exams: 40%
  - Class Participation: 20%

CSE466 Lab Content

- Arm Cortex M4 processor
  - Begin with basics and build
  - Do with your lab partner
  - You can work off-site

- Resources
  - Freescale Arm Processor
  - 320x240 Color LCD display
  - Switch, potentiometer
  - Accelerometer with gyroscope
  - Tri-color LED, NeoPixels, Bluetooth BLE
  - Learn how to interface various devices

- Final project
  - Heart-rate monitor– a mini ECG
  - LCD display of heart trace
  - Measure heart rate, basic arrhythmia detection
Freescale MK20DX256VLH7 processor

<table>
<thead>
<tr>
<th>Core</th>
<th>System</th>
<th>Memories</th>
<th>Clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM® Cortex®-M4</td>
<td>Internal and External Memories</td>
<td>SRAM 8 to 256 KB</td>
<td>Phase-Locked Loop</td>
</tr>
<tr>
<td>Debug</td>
<td>48/72/100/120 MHz</td>
<td>External Bus Interface (FlexBus)</td>
<td>Frequency-locked Loop</td>
</tr>
<tr>
<td>Interf.</td>
<td></td>
<td></td>
<td>Low-Power Frequency Oscillator</td>
</tr>
<tr>
<td>CoEIA</td>
<td></td>
<td></td>
<td>Internal Reference Oscillas</td>
</tr>
<tr>
<td>Security</td>
<td>Analog</td>
<td>Timers</td>
<td>Communication Interfaces</td>
</tr>
<tr>
<td>and Integrity</td>
<td></td>
<td></td>
<td>HMI</td>
</tr>
<tr>
<td>Cryptographic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authentication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voltage Regulator</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I²C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UART</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FlexTransceiver</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Texas-Motor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Freescale MK20DX256VLH7 processor is a high-performance microcontroller designed for various applications requiring efficient processing and reliable communication. It integrates a range of features, including an ARM Cortex-M4 core, extensive memory options, and a variety of interfaces for interfacing with other components and peripherals.
CSE466 Labs

- Final Project – Using the Teensy 3.1
- Build a Heart Rate Monitor

Assignment for Thursday:

- Review the K20 Sub-Family Reference Manual
  MK20DX256 Manual (8.0M PDF), for Teensy 3.1 (This manual has all the useful programing info)

  - Chapter 2: Introduction and Functional Modules

  Download here:

  https://courses.cs.washington.edu/courses/cse474/17wi/pdfs/K20P64M72SF1RM.pdf

  (link is on the Resources page...)