Today

- Course perspective and logistics
- Embedded systems introduction

Course Perspective #1: Mostly About Software

- The purpose of an advanced class is to tackle an area in depth
  - This course is primarily about embedded software
  - SW is primary focus on labs, exams, etc.
- In contrast 5780 is a basic course and tries to give a broad overview of microcontroller system issues

Course Perspective #2: Holistic

- Can't just look at an embedded system as a collection of parts
- Many important issues involve the whole system
  - Debugging
  - Security
  - Timeliness
  - Power and energy use

- Q: Why focus on a holistic view of embedded software?
- A: You are extremely valuable if you:
  1. Have a deep understanding of both the HW and SW sides of embedded system design, and how they interact
  2. Can see the big picture about a software design in order to spot potential problems and opportunities
- What does "extremely valuable" mean?

Prereqs and Expectations

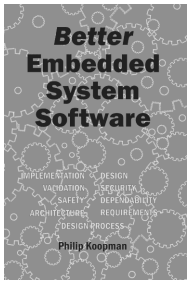
- Everyone should already:
  - Be able to write and debug C programs
  - Understand basic systems concepts – interrupts, device interfacing, etc.
    - From CS/ECE 5780, CS 4400, CS 5460, …
- CS folks need to be willing to learn:
  - Breadboarding
  - Logic analyzer use
  - How to read vendor reference manuals
- ECE folks need to be willing to learn:
  - Software engineering material
  - System-level software thinking
  - A bit of programming language and compiler material

Course Components

- Lecture
  - I expect good attendance
  - If attendance is too bad I start giving pop quizzes
- Labs
  - Embedded programming projects
  - Bulk of your time will be spent on these
- Homework
  - Pretty minimal – handful of assignments
- Exams
  - 1 midterm, 1 final

Textbook

- Better Embedded System Software, by Phillip Koopman
- Also: Get a good book on C programming
  - Course web page lists one
  - There are other good ones, we can talk about this…



Labs

- Tuesdays, 3:40-5:00 in the ECE Digital Lab
  - MEB 2265
- Programming environment is CodeWarrior IDE which runs on Windows
  - Free download if you want to run it on your machines
- Programming target is a ColdFire development board from FreeScale
  - Nice 32-bit CPU
- You work in groups of two
  - Best to have one CS person, one ECE person!
- Determines 50% of your course grade

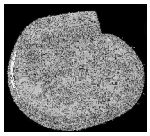
To Do

- Get on the cs5785 course mailing list
  - See https://sympa.eng.utah.edu/sympa
  - One list for all course sections
  - I'll assume everyone reads this
  - Mail to this list goes to all subscribers
  - To mail just me and the TA use
    - teach-cs5785@list.eng.utah.edu
- Look for a number starting with 2* on the back of your Ucard
  - If this number isn't there, you need a new card
  - The 2* indicates a modern card that contains the RFID chip that will get you into the lab

- Questions?

Embedded Systems

- Account for >99% of new microprocessors
  - Consumer electronics
  - Vehicle control systems
  - Medical equipment
  - Etc.



Definitions of "Embedded System"

1. A special-purpose computer that interacts with the real world through sensing and/or actuation
2. A computer that you don't think of as a computer
3. Almost any computer that isn't a PC
4. …

More definitions

- Microprocessor: A regular CPU
- Microcontroller: A system on chip that contains extra support for dealing with the real world
  - Analog to digital and digital to analog converters
  - Pulse width modulation
  - Networks: serial, I2C, CAN, USB, 802.15.4, etc…
  - General-purpose I/O pins
  - Lots of interrupt lines
  - Low-power sleep modes
  - Voltage / frequency scaling
  - Temperature / vibration resistance
  - Onboard volatile and nonvolatile RAM
  - What else?

Embedded Characteristics

- Close interaction with the physical world
  - Often must operate in real time
- Constrained resources
  - Memory
    - SRAM, DRAM, flash, EEPROM, …
  - Energy
  - CPU cycles
  - Pins
  - Flash memory read / write cycles
  - What else?

More Characteristics

- Concurrent
  - Easy to make concurrency errors
  - Hard to find and fix them
- Often lack:
  - Virtual memory
  - Memory protection
  - Hardware supported user-kernel boundary
  - Secondary storage
- Have to be developed rapidly
- Cost sensitive
  - Per-unit cost often dominates overall cost of a product

Important Difference

- Unlike PC software, embedded software is developed in the context of a particular piece of hardware
  - This is good:
    - App can be tailored very specifically to platform
    - In many cases writing portable software is not a concern
  - This is bad:
    - All this tailoring is hard work

What Do Embedded Systems Do?

- 5 main kinds of functionality:
  - Digital signal processing
  - Open loop and closed loop control
  - Wired and wireless networking
  - User interfacing
  - Storage management
- Most embedded systems do 1-4 of these
- Which apply to:
  - Cell phone?
  - LinkSys home router?
  - Cruise control?
  - Stoplight?

Digital Signal Processing

- Idea:
  - Operate on discrete approximations of continuous signals
- Origins in the 1960s and 70s:
  - Radar and sonar
  - Space program
  - Oil exploration
  - Medical imaging
- Far broader applicability today

More DSP

- Applications:
  - Telecom: Compression, echo control, wireless
  - Audio: Music, speech generation and recognition
  - Echo location: Radar, sonar, medical, seismology
  - Image processing: Compression, feature recognition, manipulation
- You could take years of courses on DSP
  - Extremely broad topic
  - Extensive theoretical underpinnings

Control

- Idea
  - Make stuff happen in the world
- Open loop control
  - No feedback
  - E.g. toaster, stoplight
- Closed loop control
  - Uses feedback to adjust output
  - E.g. thermostat, cruise control
- You could take years of courses on control
  - But you better enjoy differential equations…

Networking

- Idea
  - Processors want to talk to each other
- Differences from PC networking
  - Communication is often local
    - E.g. "unlock the driver's side door"
  - Specialized protocols
    - Often not TCP/IP
  - Topology may be fixed
  - Often low-bandwidth
    - Faster networks not necessarily better
  - Wireless increasingly important
  - Packets can have real-time deadlines

User Interfacing

- Idea
  - Present functionality directly to humans
- Modes:
  - Visual – screens
  - Tactile – keyboards
  - Aural – sounds, speech recognition
- This aspect of embedded systems shouldn't be ignored
  - Bad interfaces kill people
    - E.g. anesthesia, radiation therapy
- But we will ignore it anyway
  - Doesn't really fit in with rest of course
  - We have a UI course if you're really interested

Storage

- Idea
  - Make today's huge persistent storage devices available to embedded applications
- Sometimes embedded storage is special-purpose
  - Car needs to remember if passenger-side airbag is enabled or disabled
- But often, general-purpose storage management can be embedded
  - iPods, digicam flash cards, etc. use standard filesystems

Embedded System Requirements

- Two basic flavors
  - Functional – What the system does
    - We just talked about this
  - Non-functional (or para-functional) – Important properties not directly related to what the system does

Example Non-Functional Requirements

- Energy-efficient
- Real-time
- Safety critical
- Upgradeable
- Cost sensitive
- Highly available or fault-tolerant
- Secure

- These issues cut across system designs
  - Important (and difficult) to get them right
  - We'll be spending a lot of time on these

Crosscutting Issues

- Energy efficiency
  - Must run for years on a tiny battery (hearing aid, pacemaker)
  - Unlimited power (ventilation control)
- Real-time
  - Great harm is done if deadlines are missed (process control, avionics, weapons)
  - Few time constraints (toy)

More Crosscutting Issues

- Safety critical
  - Device is safety critical (nuclear plant)
  - Failure is largely irrelevant (toy, electric toothbrush)
- Upgradability
  - Impossible to update (spacecraft, pacemaker)
  - Easily updated (firmware in a PC network card)

More Crosscutting Issues

- Cost sensitivity
  - A few % in extra costs will kill profitability (many products)
  - Cost is largely irrelevant (military applications)
- Availability / fault-tolerance
  - Must be operating all the time (pacemaker, spacecraft control)
  - Can shutdown at any time (cell phone)

More Crosscutting Issues

- Secure
  - Security breach extremely bad (smart card, satellite, missile launch control)
  - Security irrelevant (many systems)
- Distributed
  - Single-node (many systems)
  - Fixed topology (car)
  - Variable topology (sensor network, bluetooth network)

Ripped from the Headlines

- Since 2008, new vehicles in the US all have a sensor in each tire
  - Communicates with main ECU using wireless
- "The pressure sensors contain unique IDs, so merely eavesdropping enabled the researchers to identify and track vehicles remotely. Beyond this, they could alter and forge the readings to cause warning lights on the dashboard to turn on, or even crash the ECU completely."
- http://arstechnica.com/security/news/2010/08/cars-hacked-through-wireless-tyre-sensors.ars

Software Architectures

- Important high-level decision when building an embedded system:
  - What does the "main loop" look like?
- How is control flow determined?
  - What computations can preempt others, and when?
- How is data flow determined?
- Options:
  - Cyclic executive
  - Event-driven
  - Threaded
  - Dataflow
  - Client-server

Cyclic Executive

```
main() {
  init();
  while (1) {
    a();
    b();
    c();
    d();
}}
```

Advantages?
Disadvantages?

Historically, most embedded systems are based on cyclic executives

Cyclic Exec. Variations

```
main() {
  init();
  while (1) {
    wait_on_clock();
    a();
    b();
    c();
}}
```

```
main() {
  init();
  while (1) {
    a();
    b();
    a();
    c();
    a();
}}
```

Interrupt Driven

```
main() {
  while (1) { }
}
```

Or…

```
main() {
  while (1) {
    sleep();
  }
}
```

```
interrupt_handler() {
  …
}
```

Advantages?
Disadvantages?

Event Driven

```
main() {
  while (1) {
    event_t e =
      get_event();
    if (e) {
      (e)();
    } else {
      sleep_cpu();
}}}
```

```
interrupt_handler() {
  time_critical_stuff();
  enqueue_event
    (non_time_critical);
}
```

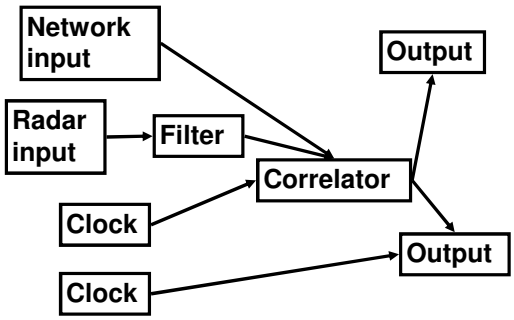
Advantages?
Disadvantages?

Threaded (using an RTOS)

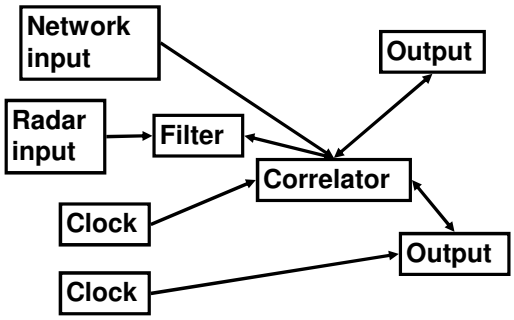
- Threads are usually sleeping on events
- Highest priority thread runs except when:
  - It's blocked
  - An interrupt is running
  - It wakes up and another thread is executing in the kernel

Advantages?
Disadvantages?

Pipeline-Driven (Dataflow)



Client-Server



Architecture Summary

- All of the architectures have significant advantages and disadvantages
  - Resource usage
  - Responsiveness
  - Safety
  - Fault tolerance
  - Maintainability
- Once an architecture is chosen, lots of other design decisions follow
- Very important to choose an appropriate architecture for a new system
- Architectures can be combined
  - But this is hard to get right

Choosing a CPU

- Issues:
  - Cost
  - Size
  - Pinout
  - Devices
  - Performance
  - Match to system workload
  - Memory protection
  - Address space size
  - Word size
  - User / kernel support
  - Floating point

CPU Options

- Create custom hardware
  - May not need any CPU at all!
- 4-bit microcontroller
  - Few nibbles of RAM
  - No OS
  - Software all in assembly
  - E.g. COP400, EM73201, W741E260, HD404358
  - Dying out?

More CPU Options

- 8-bit microcontroller
  - A few bytes to a few hundred KB of RAM
  - At the small end software is in asm, at the high end C, C++, Java
  - Might run a home-grown OS, might run a commercial RTOS
  - Still dominate both numbers and dollar volume
  - Two kinds:
    - Old style
      - CISC, designed for hand-written code
      - E.g. 68HC11, 6502, Z80, 8051
      - These are >20 years old and doing well
    - New style
      - RISC, designed as a compiler target
      - E.g. AVR, PIC

More CPU Options

- 16- and 32-bit microcontrollers
  - Few KB to many MB of RAM
  - Usually runs an RTOS
  - May or may not have caches
  - Wide range of costs
  - 16-bit: 68HC16, H8
  - 32-bit: ARM7, ARM9, ARM11, MIPS, MN10300, x86, PPC, ColdFire
  - Labs in this class will use ColdFire
  - Is 16-bit dying?
    - Has serious disadvantages compared to 32-bit but few advantages
  - New ARM "Cortex" processors designed to kill the 8-bit and 16-bit markets

More CPU Options

- 32- or 64-bit microprocessor
  - Basically a PC in a small package
  - Runs Win XP, Linux, or whatever
  - Relatively expensive in power and $$
- Many specialized processors exist
  - E.g. DSP – optimized for signal processing

Choosing a Language

- Issues:
  - Footprint
    - RAM, ROM
  - Efficiency
  - Debuggability
  - Predictability
  - Portability
  - Toolchain quality
  - Libraries
  - Level of abstraction
  - Developer availability
    - Anyone know Jovial? PL/1? Forth? BCPL?

Programming Languages

- Assembler
  - No space overhead
  - Good programmers write fast code
  - Non-portable
  - Very hard to debug
- C
  - Little space and time overhead
  - Somewhat portable
  - Good compilers exist

More Languages

- C++
  - Often used as a "better C"
  - Low space and time overhead if used carefully
  - Unbelievably complex
- Java
  - More portable
  - Full Java requires lots of RAM
  - J2ME popular on cell-phone types of devices
  - Bad for real-time!

Choosing an OS

- Issues very similar to languages
  - Footprint
    - RAM, ROM
  - Efficiency
  - Debuggability
  - Predictability
  - Portability
- Other issues
  - Process / thread model
  - Device support
  - Scheduling model
  - Price and licensing model

Real-Time OS

- Low end: Not much more than a threads library
- High end: Stripped-down version of Linux or WinXP
- Many, many RTOSs exist
  - They are quite easy to create
- Interesting RTOSs:
  - QNX
  - uClinux
  - uC/OS-II
  - VxWorks

Summary

- Embedded systems are highly diverse
- External requirements dictate
  - Choice of CPU, language, OS
  - Choice of software architecture
    - This is worth thinking about very carefully
- Very different experience developing embedded apps relative to desktop apps
- Embedded systems are:
  - Fun – They make stuff happen in the real world
  - Important – Your life depended on hundreds of them on the way to school today
  - Ubiquitous – More processors sold per year than people on earth

Assignment for Tuesday

- Find an embedded device that you can take apart such as an old
  - Cell phone, home router or hub, MP3 player, printer, …
- Has to be a device you don't care about!
  - If you can't find one, talk to me
- Open the device so you can see the main circuit board
- Identify as many parts as possible – search for part numbers on the web
- Talk about the device in class on Tues
  - Also hand in a short writeup – I'll mail about this

Assignment for Thurs

- A short quiz about C code is one the course web page
- Print it and write in the answers
- Bring it to class next Thurs