Today

◆ **Administrative stuff**
  - Perspective taken by this course
  - Expectations and prerequisites
  - Components of the course

◆ **Embedded systems intro**
  - What are they?
  - What is it that they do?
  - Why do we care?
Course Perspective #1: Software-Centric

- 5780/6780 is a basic course and tries to give a broad overview of microcontroller system issues, especially low-level interfacing.

- This class is about building embedded software:
  - What it does
  - How it does it
  - How to build it
  - How to make sure it works
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?
Another view: You are extremely valuable if you…

1. Are really good at something
2. Can talk to people who are really good at things you’re not good at
3. Can work on a team to accomplish goals that are too large to accomplish alone

We will spend a lot of time working on these skills this semester
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:**
  - How to think about software
Course Components

◆ Lecture
  ➢ I expect good attendance
  ➢ If attendance is too bad I start giving pop quizzes

◆ Homework
  ➢ Pretty minimal – handful of assignments

◆ Group programming assignments
  ➢ These will take up the bulk of your time in this course

◆ Exams
  ➢ 1 midterm, 1 final
Good Books

- Better Embedded System Software, by Phillip Koopman
- The C Programming Language 2e, Kernighan and Ritchie
Labs

- ECE digital lab is available for us to work in
- No regularly schedule lab time
- But: I will be meeting with each group each week outside of class
- You’ll work in groups of 3 or 4
- Assignments will run on Raspberry Pi boards
  - Small ARM-based development boards
  - Runs full Linux
  - You will each buy one
To Do

◆ Get on the cs5785 course mailing list
  - See https://sympa.eng.utah.edu/sympa
  - One list for all course sections
  - 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
More ToDo

◆ Order a Raspberry Pi
  ➢ $35
  ➢ Do this right away! There is a shipping delay
  ➢ Use Element14 / Newark
  ➢ Find links at the course web page or the Raspberry Pi site:
    ➢ http://raspberrypi.org/

◆ Order or find these accessories:
  ➢ Micro-USB charger
  ➢ 4 GB or larger SD card
  ➢ HDMI-to-whatever cable
  ➢ USB keyboard, mouse
More ToDo

- Read the Launch Interceptor Program specification
- Register for a Github account
  - [http://github.org/](http://github.org/)
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. …
Is smartphone and tablet programming “embedded programming”?
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
  - Embedded 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 / radiation 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
  ➢ Computers 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
  ➢ 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 mostly ignore it

◆ 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
Why do we care?

- Embedded systems are amazingly useful
- Your life depended on dozens of them already today
  - What were they?
- By mid-2012, 30 billion ARM processors had shipped
  - > 4 per person on Earth!
- These trends are growing in importance
- You are the people who will design, develop, test, and maintain these systems for the next 40 years