Embedded Systems and PNT Education At Stanford University

November 6, 2008

Max Klein
EE PhD Candidate, Teaching Fellow
Stanford University STARLab
maxklein@stanford.edu

Background

- 3 Major PNT Classes
  - AA272C: Intro to Satellite Navigation (Winter)
  - EE292C/EE192C Embedded Systems Engineering (Spring)
  - AA272D: Advanced topics in navigation (Fall)

- All are project-based courses

- We feel that project-based classes are the best way to teach Systems Engineering

- This presentation focuses on EE292C/192C

<u>Goal</u>

What do our students need education in to be more effective in industry

What is an Embedded System?

- No standard (e.g. IEEE, etc) definition
- A system in which a processor/microcontroller/computer is embedded to perform a specific task or tasks
  - Often have sensing, processing, storage, and communication
- Although task-specific, general purpose processor are often used
  - ASICs enhance performance, but cost time and money to develop
- Blurry, moving line between ES and PCs

Motivation

- Embedded Systems are extremely common in everyday life
  - Cell phones, MP3 Players, home appliances, vehicle navigation, collision detection, etc
- Real Time ES support critical applications, with huge stakes
  - Aircraft control, nuclear power plant management, targeting and fire control systems, medical applications
- Building these systems is very rewarding, and is a truly interdisciplinary activity
  - Requires *both* hardware and software
  - Hardware interfaces to the real world
    - Sensors and actuators
  - Software controls this real-world interaction
  - End product is a tangible item that does something *for* you
- The ES industry is huge, and growing each year
  - About 98% of processors shipped are used in embedded systems (embedded.com)

EE192C: What is it?

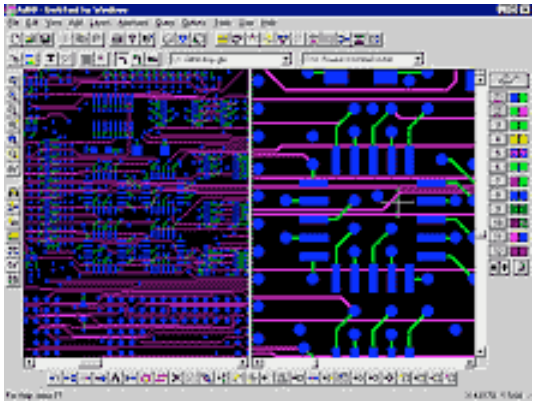
- Hands on
  - Class starts by programming an existing hardware development kit
  - During the second half of the course, students design the hardware itself

- Generalized
  - Some assignments are done the hard way
    - The knowledge students gain will apply to implementing subsystems of future protocols, even when not directly supported in hardware
  - We must have a platform to learn on, however the principals are generic
    - We use an inexpensive, relatively simple 8-bit RISC microcontroller
    - Future work with ARM, XSCALE, etc is just an increased set of registers and peripherals

- Gadget Class
  - We're not <u>using</u> gadgets, but rather designing them!
    - Goal is to create systems to fill a gap in what's available COTS
    - They won't be pretty to start with, so cooperation suggested with Product Design

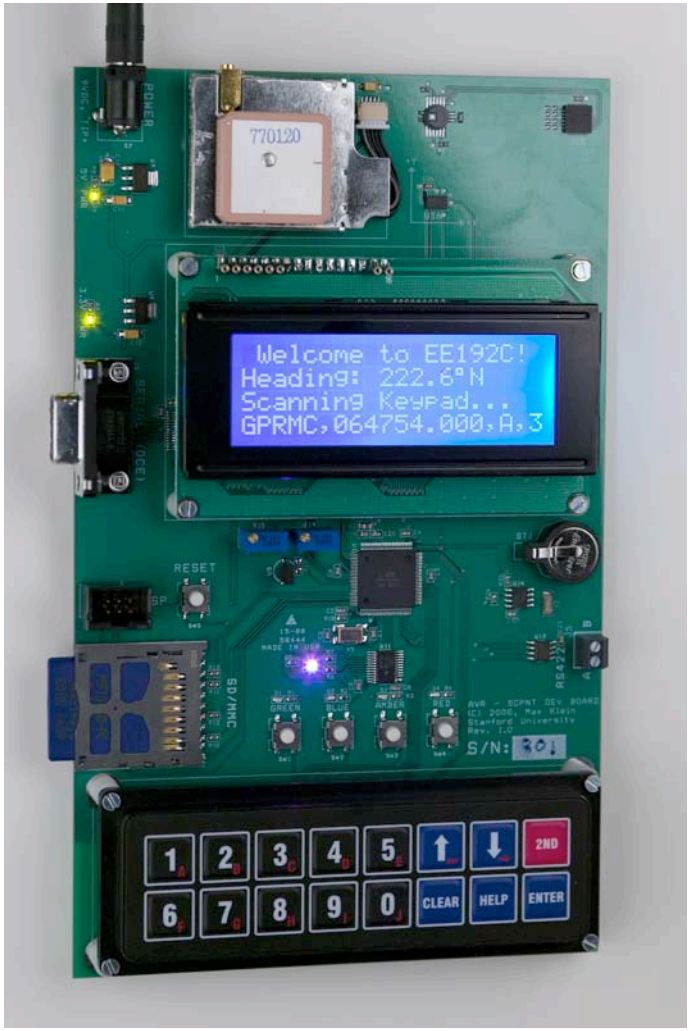
Course Flow

- We start by programming known-good hardware
  - This programming requires an understanding of the underlying hardware
  - Easier to debug when either the software OR hardware is known functional
  - Cover the actual hardware and electronics of the peripherals that are utilized in software
  - The midterm projects are software-only, developing an integrated systems utilizing most of the available peripherals in the development kit



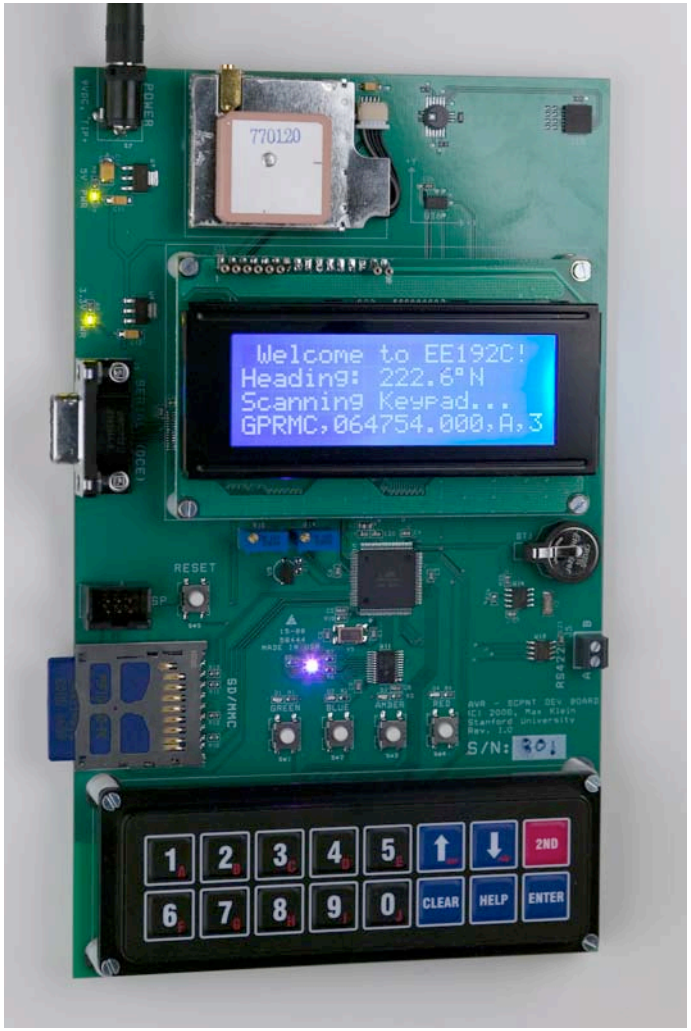
- In the second half of the class, students design hardware
  - Students understand the software implications of how they implement the hardware
  - Hardware designed beginning with a functional description, all the way through component selection and PCB layout

Development Board: Basics



- Atmel 8-bit AVR Microcontroller
  - 256KB Flash, 8KB SRAM
  - 16MHz, “Single-cycle” RISC
- GPIO Buttons, LEDs
- PWM Tri-Color LED
- Matrix Keypad
- Character LCD
- RS232 Port
- SD / MMC Slot
  - Includes 1GB SD Card
  - SPI

Development Board: PNT



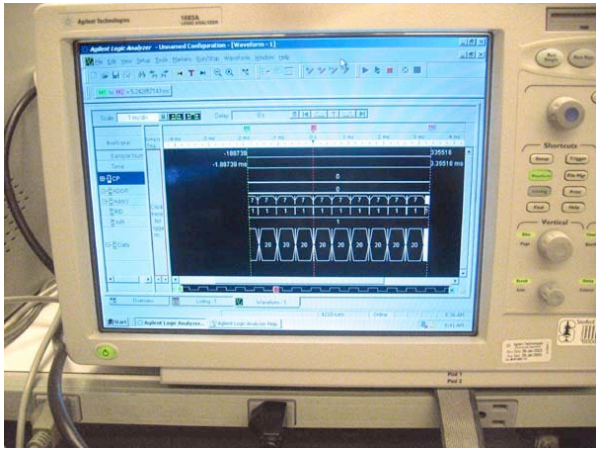
- GPS Receiver
  - NMEA over RS232
- LIS302 Accelerometer
  - 3-axis linear, +/-2g
  - I2C Interface
- 6352 2-Axis Magnetometer
  - “Digital Compass”
  - SPI Interface
- SCP1000-D11 Pressure Sensor
  - 10cm air column resolution
  - I2C Interface
- Real Time Clock
  - I2C Interface
- RS485 Header
  - Future IMUs

Course Outline

- Basics Review
  - Number Systems, binary and hexadecimal math
- Microcontroller Overview
  - System Blocks
  - Registers, memory, and address space
  - How a processor can interact with external devices
- General Purpose I/O
  - Outputs (LEDs)
  - Inputs (Buttons, Keypad)
- Communication Methods
  - RS-232
  - Character LCDs
  - I2C / TWI
  - SPI
- Solid State Storage
  - Compact Flash / IDE
  - Secure Digital / MMC

Course Outline (cont.)

- Timers, Interrupts, and Real Time Systems
  - Definition of hard vs soft real time
  - IRQs and ISRs
  - Preemption
  - Real Time Operating Systems
- Mid-term Project: Software Design
- System Design
  - Requirements specification
  - Hardware selection
    - ASICs
    - Power Budgets, Batteries, chargers
  - Software implications of hardware choices
  - OrCAD PCB Design Suite
    - Capture CIS for schematics
    - Layout Plus for PCB routing
    - Lots of practical information and hints for PCB layout
      - Avoiding noise, power planes, analog and digital separation
- Final Project: Hardware Design



Thank You

Your Thoughts?