PGT302 – Embedded Software Technology

PART 1
Introduction to the Embedded World

Objectives for Part 1

- Need to DESCRIBE and DISCUSS the following topics:
  - Embedded systems
  - Embedded software
  - Embedded hardware platform
  - Embedded software development
- Why this is important?
  - Money is always a good motivation (other than the fact that it is COOL to be good at this!)
  - Many applications need this: smart-{fill-in-this-spot-with-anything-you-can-think-of}
  - They pay people who can develop such thing :p

Embedded Systems

- Can be defined from many point of view
- "An embedded system is an application that contains at least one programmable computer (typically in the form of a microcontroller, a microprocessor or digital signal processor chip) and which is used by individuals who are, in the main, unaware that the system is computer-based." - Michael J. Pont, Embedded C
- "An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts." - Wikipedia entry on Embedded System 20140910

Embedded Systems (cont.)

- Check out the following statement:

  "One very unfortunate aspect of embedded systems is that the terminology surrounding them is not very consistent. For every word, there are four or five subtly different meanings. You will just have to live with this problem." - An Embedded Software Primer, David E. Simon
- Different point of view → Different meaning
- Remember 7 blind men and the elephant!

Embedded Systems (cont.)

- The following is summarized from my site:
  http://azman.unimap.edu.my/dokuwiki/doku.php#embedded_system
- “An embedded system is a combination of hardware and software components that is usually meant to be part of (or embedded into) a larger system, but is still technically a standalone system and is primarily designed for a specific purpose”
- Focus on the ‘symbiosis’ between hardware and software (both are equally important!)

Embedded Software

- Based on my definition: the software component of an embedded system
- Sometimes, also known as *firmware*
- Based on application / hardware requirements:
  - Can be a bare-bone application
  - Can be a mini-OS capable of multitasking
  - Can be a full-blown OS
- No rules... just need to work as intended

Embedded Software

- Embedded Hardware provides functionality
  - Without functionality, software cannot perform
  - Hardware interrupt → events (software)!
  - Software is only as fast as hardware can execute
- Embedded Software provides features
  - Without features, hardware seems dull (not doing anything fancy)
  - Quad-core hardware running single thread vs. single-core hardware running multiple-thread
- To develop good software, know your hardware!

Embedded Hardware Platform

- Most of the time – microcontrollers!
- What is a microcontroller?
  - A microprocessor that emphasizes self-sufficiency (cost effective @single-chip solution!)
- Shows that COST is THE determining factor!
- Two distinguished platform:
  - Intel 8051 core: long-time industrial standard cost-effective 8-bit microcontroller (still VERY relevant)
  - ARM core: current leader in consumer electronics, a 32-bit RISC microcontroller core (ARMv8 introduces 64-bit support!)

Embedded Hardware Platform (cont.)

- Intel 8051 core: part of MCS-51 family of 8-bit control-oriented microcontrollers
- 2 timers (3 in 8052), 2 interrupt pins, dual-function UART lines (total 5 interrupt sources)
- 4 x 8 I/O lines, 4 register banks mapped to internal RAM, bit-addressable RAM
- Still popular today – various manufacturers, enhanced design (faster, more features)
- C compilers available – but assembly is preferred for time-critical systems

Embedded Hardware Platform (cont.)

- ARM core – based on reduced instruction set computing (RISC) architecture
- Targets portable, low-power devices
- Licensed out (as silicon core design) to many manufacturers by ARM Holdings (IP owner)
- 32-bit processing and 32-bit address space
- Some supports Java bytecodes
- ARMv8-A (2011) supports 64-bit
- Currently the most popular processor for consumer devices - 95% smartphones (ARM: Media fact sheet 2016)

Embedded Hardware Platform (cont.)

- Alternative Platforms:
  - Field Programmable Gate Array (FPGA)
  - Digital Signal Processor (DSP)
- FPGA
  - Meant for hardware prototyping of logic circuits
  - Now powerful enough as stand alone processing element
- DSP
  - Processors specifically designed for digital signal processing
  - Basically, a lot of hardware multiply-accumulate (MAC) circuits

Embedded Software Development

- Just like any software development... just remember – KNOW YOUR HARDWARE!
- Cross-compilers
  - Build machine: builds compiler
  - Host machine: hosts/runs the compiler
  - Target machine: intended platform of the binary
  - Common: Build & host are the same machine
  - Cross-canadian: separate host/build machine
- Again, know your hardware!
  - … and the (software) binary format required

Embedded Software Development (cont.)

- According to Jack Ganssle (ganssle.com):
  - 80% of all embedded systems are delivered late
  - Bugs are the #1 cause of late projects
  - New code generally has 50 to 100 bugs per 1000 lines
  - Traditional debugging is the slowest way to find bugs
- Drivers are usually the toughest part – the part where hardware and software touches!
  - The most probable cause of OS crashes

Embedded Software Development (cont.)

- Development Language
  - Either assembly or C
  - Forget the rest!
- Assembly is hardcore
  - Platform specific, hard to master
  - Long development time if coding from scratch
- C used to be high-level (still is, relatively)
  - Available on (almost?) all popular platforms
  - Good programmers can still ‘see’ the assembly in C codes...

Embedded Software Development (cont.)

- Testing / debugging
  - Board development sometimes takes longer
  - Use simulators / emulators (model accuracy)
    - Simulators usually simulate functionalities
    - Emulators usually emulate hardware processes
  - Can also test peripherals before deciding
- Implementing
  - ‘downloading’ to platform (bootloaders?)
  - flashing NVRAM (used to be burning ROMs)
  - Sometimes require specific software tool (board with proprietary protocol – not many nowadays?)
  - Open is preferable? (e.g. Android vs Nokia?)

The End
of Part 1