# Bringing Embedded Software Closer to Computer Science Students

Jogesh K. Muppala
Dept. of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
muppala@cse.ust.hk

# Talk Outline

- Introduction
- Background
- The Course
  - Course Topics and Structure
  - Hands-on Laboratory Exercises
  - Course Projects
- Students
  - Student background
  - Student feedback
- Conclusions

# Introduction

- Embedded systems growing in importance, widespread deployment and complex with time
- Software becoming an important component of embedded systems
- Training manpower on the design and implementation of embedded software becoming more important

# Introduction

- Computer science students often stay away from embedded systems:
  - Perception as a “hardware” course
  - Missing out on participating in a dynamic field
  - How can we “correct” this misperception?
- There is indeed lot of scope for CS students to participate in this field without getting too “involved” with the hardware

# Background

- CS students tend to be trained well on abstractions, especially about creating formal abstractions of computational processes and to relate layers of abstractions to each other (Sztipanovits et al. [16])
- Drastic rethinking needed on the way embedded software is developed, putting emphasis on the timing aspects of software (E. A. Lee [10])
- Model-driven approach (Sangiovanni-Vincentelli et al. [15], Sztipanovits et al. [16])

# Background

- What is so special about embedded software?
- Does this conglomeration of diverse topics deserve a special course?
  - Why not cover the related topics in the courses dealing with those specific topics?
  - How do we address the overlap of topics with the other courses?
  - Why not make embedded a point of emphasis in these other courses?
- Example of such an integration: Prof. Gary Nutt's OS course for Small Computer Systems [13]

# Course Topics and Structure

- Three major areas emphasized in the course:
  - Embedded software development
  - Real-time and embedded operating systems (RTOS)
  - Embedded software engineering
- Striking a good balance between theory and practice

# Course Topics and Structure

- Introduction
  - Introduction to Embedded Systems
  - Examples of Embedded Systems
  - Embedded System Characteristics
- Embedded Systems Architecture
  - Hardware Fundamentals: Processors, Memory, Bus, etc.
  - Software: OS, Application Software
- Embedded Software Development
  - Hosts and Targets
- Interrupts
  - Introduction to Interrupts
  - Interrupt Handlers and Interrupt Service Routines
- Embedded Software Architectures
- Real-Time Operating Systems (RTOS)
  - Review of Operating Systems Basics
    - Tasks, Processes and Threads
    - Task Scheduling: Rate Monotonic Scheduling, Priority Inversion
    - Task Synchronization and Coordination
    - Intertask Communication
    - Memory Management
  - Example RTOS: mC/OS-II, Windows CE, Embedded Linux
- Embedded Software Engineering
  - Basics of Software Engineering
  - Software Engineering Models
  - Unified Modeling Language (UML)
  - Software Testing
- Testing and Debugging Embedded Systems

# Hands-on Laboratory Exercises

- Major goals:
  - Introduce students to various IDEs and embedded environments
  - Give an overview of various techniques for embedded software development including aspects of RTOS
  - Preparing the students for course project
- Hands-on laboratory component concentrated mainly on the use of several real-time OS and integrated development environments
  - Mainly Windows CE, Platform Builder, Embedded Visual C++
  - General computer laboratory used
  - Dedicated embedded software laboratory using Ebox-II from ICOPTech and Intel PXA255/PXA 270 boards

# Hands-on Laboratory Exercises

- Specialized equipment for embedded laboratory:
  - Ebox-II from ICOPtech

  - Intel PXA255/PXA270 based embedded development boards from Emdoor Inc., Shenzhen

# Hands-on Laboratory Exercises

- Typical set of laboratory exercises include:
  - Introduction to Platform Builder
  - Advanced PB and debugging features
  - Application development using eVC++ and VS 2005
  - Threads and thread synchronization in Win32 API
  - Interprocess communication using message queues, MSMQ, events
  - Priority scheduling and priority inversion issues
  - Memory leaks and detection

# Course Project

- Team project with teams of up to 3 students
- Main emphasis on demonstrating the use of techniques learnt in the course
- Students felt that this experience illustrated to them that a whole new arena of small device programming was easily accessible to them and provided them with alternate avenues for future career

# Course Projects

- Automatic "Dim Sum" Ordering System
- On-line Retail Management System (RMS)
- Stock Manager
- Podcast CE
- Video Surveillance System
- “Bomberlady” – an embedded linux game
- NewStation
- Real Time Operating System – USTOS

# Student Background

- A good mix of students in their junior and senior year of undergraduate studies
- A good mix of students from the computer engineering and computer science stream
- Background includes courses on:
  - Computer programming including OO
  - Computer architecture and organization
  - Operating Systems
  - Practically no software engineering

# Student Feedback

- More hands-on labs, experience with dedicated hardware and embedded development platforms, rather than the general purpose PC
- Most popular: embedded development, and RTOS
- Least popular: software engineering
- More in-depth coverage of only two or three major RTOS appreciated
- Overlap with other courses should be minimized
- UML not well appreciated: lack of suitable case-studies

# Conclusions

- Embedded systems software is an interesting topic deserving a dedicated course
- Our experience is only one point of reference
  - More expeirence sharing needed
- Greater consensus on embedded systems and software curricula:
  - Where do different courses fit in the overall picture?
  - What are the suitable set of courses?