Real Time and Embedded Systems

by

Dr. Lesley Shannon

Email: lshannon@ensc.sfu.ca

Course Website: http://www.ensc.sfu.ca/~lshannon/courses/ensc351

Simon Fraser University

Slide Set: 10

Date: November 24, 2011

Slide Set Overview

- Final Project Report
  - Final Demo
  - Group Report
  - Individual Report

- Real Time software design issues

- Embedded System software design issues

Final Demo

Final Demo

- The final demo is worth 6%
  - 4% functionality
  - 2% understanding
- Everyone needs to know what their code is doing and how it works
- Be sure your code is up and running ***before*** the demo time slot begins

Final Demo

- During the final demo, we will be checking to see:
  - That the user can interact with the game properly (timely user interface responses)
  - Screen updates are timely, without glitching
    - This includes game scores, etc
  - The game logic functions properly plus the speed and quality of the AI functions
  - The system is stable
    - In other words, incorrect/invalid key presses won't cause the system to crash

Final Demo

- During the final demo, we will be checking to see (cont'd):
  - The corner cases on the display and gpio interaction
  - Demonstration of built in error handling
    - What are you expecting?
    - How will you handle it? How will you handle the unexpected?
  - Demonstrations and descriptions of any special features of your project
- PLUS individual responses to questions on your design

Group Report

Group Report

- A complete document of strictly technical information
  - It should allow another person to determine how your design works so they can modify it or maintain it
  - Remember we'll be looking at your code, so use a defined coding standard (XP) and ***COMMENT*** it
    - **Send your project source code attached to a WebCT email message to the TA that marked your milestones (you do not need to provide a hardcopy of the source code as part of your report).**
- Group report maximum is 10 pages
  - No books please =)

Group Report

- Use the following structure for your final report
  - Introduction
    - Give an overview of the project
    - Make it brief (a paragraph, two at the most)
    - Include your team number and members and group divisions
  - System Overview (examples of ***SOME*** of what to include)
    - A Task/Stream diagram delineating who worked on what
    - A table/figure illustrating all the functions in your design and their associated tasks
    - A users' manual

Group Report

- Use the following structure for your final report (cont'd)
  - Outcome
    - Results (how well it works or not)
    - Suggestions for future work (improvements)
    - What types of Robustness/Reliability are built into the code?
  - Description of Your Implementation
    - What do each of the functions do in your system?
    - How do they work together (shared data structures, etc)?

Group Report

- Use the following structure for your final report (cont'd)
  - Description of your AI Algorithms
    - How do you achieve the longest length/highest score per unit of time? How do you search for food?
    - An algorithm flow chart (tie it into your code)
    - A description of how the corner cases are handled (extra algorithm flow charts can be used if necessary)
  - Description of how the "push button" and "Timer 2" Drivers work
    - Algorithm description and flow chart
    - REMEMBER: In all cases, we are using this document and discussion to understand the code

Group Report

- Use the following structure for your final report (cont'd)
  - Description of the extra (bonus) features
  - Multi-threading options

Individual Report

Individual Report

- This is for ***you*** to describe ***your own*** contribution to the project
  – Talk about the pain and anguish if you need to vent =)
  – Talk about what worked/what did not and how you made it work
  – Talk about any problems with group dynamics
- Remember this project is a significant component of your grade so make it clear to me that you were a working contributor

Individual Report

- The structure for the project report is
  - Brief Introduction
    - Team number, members and your partner(s)
    - List the tasks you worked on personally
  - What you did:
    - What functions did you act as navigator/driver?
    - How did you ensure integration?
    - What were the challenges/what did you learn
    - Etc…

Individual Report

- The structure for the project report is
  - Community contributions for the term:
    - On the bulletin board
    - In the lab
  - Course feedback:
    - Did the project timeline work (demos, deadlines)?
    - Does the grading structure work? Suggestions?
    - Did you like the open lab concept for the tutorials and bi-weekly demos?
    - How did the lectures work? Did they work with the labs?

Individual Report

- Note the maximum for the "meat" of the individual report is 2 pages
  - You can add up to an extra half page for the community contributions and course feedback sections (for a total of 2.5 pages maximum)

Slide Set Overview

- Real Time software design issues
- Embedded System software design issues

Do you recall the definitions/characteristics of Real Time and Embedded Systems?

Real Time Systems Definition

Real Time Systems Definition

- A system that responds in a timely and predictable way to unpredictable external stimuli arrivals.
- Must respond quickly to meet each tasks deadlines
- Often requires simultaneous processing more than one event
  - ***all deadlines should still be met***
- Predictability/Reliability: react to ***all*** possible events in a predictable way.

Embedded Systems Definition

Embedded Systems Definition

- A computing system that is embedded in a product
- Its primary function is not computing a general computing platform
- Uses a combination of hardware and software to perform the required tasks
  - aka Hardware/Software Codesign
- Typically thought to not be "plugged in", so there is some alternative power source requirement

Real Time Systems

- Systems have to meet soft and hard deadlines
  - Soft Deadlines ***should*** be met
    - They are important for correct operation, meaning that numerical calculations may be incorrect
    - If you fail to meet the deadline, the system should not crash
  - Hard Deadlines ***must*** be met
    - Failure to meet deadlines will result in a fatal crash
    - The system will fail

Real Time Systems

- Examples of soft deadlines

- Examples of hard deadlines

Real Time Systems

- Types of hardware platforms
  - Desktop systems
    - Possibly multi-threaded
  - Embedded systems
    - May or may not include an O/S

Real Time Systems

- Desktop systems typically have
  - An O/S
  - Relatively unlimited memory (hard-drive space)
  - The ability to access the internet
  - Easy power access (not battery operated, but maybe a battery back up)
- Desktop systems may have
  - A network cluster
  - High end processors/lots of RAM

Real Time Systems

- Let's assume a desktop based system is used to implement a banking database
  - What are your assumptions?

Real Time Systems

- Let's assume a desktop based system is used to implement a banking database
  - What are your top priority concerns?

Real Time Systems

- Embedded Real Time Systems typically have
  - Limited memory capacity
  - Battery powered restrictions
  - A timer
  - “Unique peripherals”

Real Time Systems

- Let’s assume you are using an embedded real time system in an airplane
  - What are your assumptions?

Real Time Systems

- Let's assume you are using an embedded real time system in an airplane
  - What are your top priority concerns?

Real Time Systems

- Multi-threaded Real Time systems
  - Can use priority to help ensure threads run in the correct order
  - Also use scheduling algorithms
    - First come, First serve
    - Shortest Job Next
    - Round Robin
  - Not always the sensible choice
    - Significant overhead is incurred when multi-threaded designs are run on single-threaded processors

Embedded Systems

- These systems are comprised of a combination of hardware and software
- One of the largest growing market shares in the computer industry (10-25%)
- A very active area of research
  - Compilers, Architecture and Synthesis of Embedded Systems (CASES) being one of the big conferences and the Embedded Systems Conference

Embedded Systems

- Very hard to design well

- Typically looking for the “good enough” solution
  - Not the “best” solution as would be used for high end ASICs such as general purpose processors

- Designers not only have to worry as much about the computing problem being solved, but also the environment in which the processing takes place

Embedded Systems

- Design Methodology:
  - Specify System Requirements
  - Specify Application Platform (Co-specification)
  - Partition and Map Application Tasks to the Platform Resources
  - Schedule Execution Order
  - Model System Functionality (Co-simulate)
  - Do I meet System Requirements (re-specify platform/re-partition)
  - Implement (Code) Tasks
  - Verify Functionality (Co-verification)
  - Do I meet System Requirements? (re-specify platform/re-partition)

Embedded Systems

- Significant Design Considerations:
  - Area
  - Power
  - Time & Performance
  - Environmental concerns
    - pressure/temperature/subatomic particles/outer space
  - Reliability & Fault Tolerance
  - Real Time deadlines
  - Health and safety issues
  - Other application specific design concerns

Embedded Systems Software

- Code space
  - Limited
    - These days you will probably be able to use a high-level language and not have to rely on assembly
    - In some cases, you'll still have to use assembly
    - You may have to be intelligent in your choices of data structures
      - e.g. arrays are faster than pointers
    - Some high-level languages are ***very*** inefficient
      - C is better than Java (probably better than C++)
    - Limited memory may mean limited stack space
      - Affects how you pass variables

Embedded Systems Software

- Code space
  - Load from the boot strap into RAM all at once
    - Don’t want to continuously have to upload different parts of the executable to cache
  - Often use “different” processor types
    - DSP processors/ASIPs provide you with instructions that you may want to leverage in your design
  - May have multiple processors of different types
    - This may effect how you partition your code over your processors (leverage the strengths of each processor)

Embedded Systems Software

- Software system resources
  - May not have an O/S
    - Even more likely to not have built in device drivers
  - Probably will have to include interrupts ("real" ones)
    - DSP processors/ASIPs provide you with instructions that you may want to leverage in your design
  - If you cannot change the hardware platform, you'll have to speed up the software
    - Use profiling to find the costly function and then use inlined assembly code

Fault Tolerance and Reliability

- Extra circuitry can be used to provide fault tolerance in the hardware
  - Protection in case hardware is damaged by the environment, burn in, or burn out

- Fault Tolerance can also be provided using the “voting” method
  - Have a calculation completed more than once and go with the majority result

Fault Tolerance and Reliability

- Reliability ensures predictable responses
  - Users require predictable responses from the system
  - They often require predictable response time, possibly within a set tolerance
- Designers may need to build in system diagnostics
  - In case of a system error/failure, the designer may wish to diagnose the system and fix it at runtime (without bringing it off line)
  - These issues lead us to consider safety concerns

Safety

- For designs that interact with people in a critical way, there are other concerns
- If a system's improper operation can put an individual's health or safety at risk, you must ***<u>over design</u>*** it
- Assume at some point in time, some part of the design may fail or become uncalibrated
  - It is your responsibility to ensure that this does not put anyone at risk

- A good example is to think of Civil Engineers and the bridges, parking structures, and roads they build

Safety

- What types of applications might you design for that would require to think about safety issues for the user?

Questions?

- Name 4 characteristics of Real Time and Embedded Systems?

- Describe why some systems are considered embedded systems but not real time systems?

Questions?

- Name 3 Real Time system applications and three Embedded Systems applications.

- Name two different scheduling algorithms for multi-threaded systems

Questions?

- What's the difference between hard and soft deadlines?

- What are some important considerations that arise from having a limited code size for embedded systems?

Questions?

- Describe the embedded system design methodology discussed in class.

A final example of reliable (unreliable?) systems

- Banks are not the only ones with private and secure information about individuals
- The government keeps lots of encrypted information from taxes to security files
  - Security and Reliability are key
- Remember what happened with the gun registry ‘02
  - $2M ballooning to $1-2B

What would be required for the gun registry system?

When spec’ing a system, what do you need to do?

- What are the user’s system requirements?
- What kind of hardware are you expected to use?
- What are your performance and reliability requirements?
  - Extra hardware for backups?
- What are your hard deadlines/soft deadlines?
- What are possible error situations?
  - How will you handle them?
- Decide what modules you will partition the design into.
- Are there any extra special (corner) cases/conditions to worry about?

AGILE programming and the gun registry problem

What would a unit test for the form submissions look like?

Real Time & Embedded Systems

- A final example for you to think on:
  - Let's say you are asked to design a cellphone, list the functions you support and the design concerns you have to consider.