18-648: Embedded Real-Time Systems

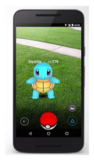
Raj Rajkumar

Lecture #1

Today's "High Tech" is significantly about…

Embedded, real-time, mobility and connectivity

- Smartphones, tablets and wearable devices
- Modern software environment (Linux, Java → Android)



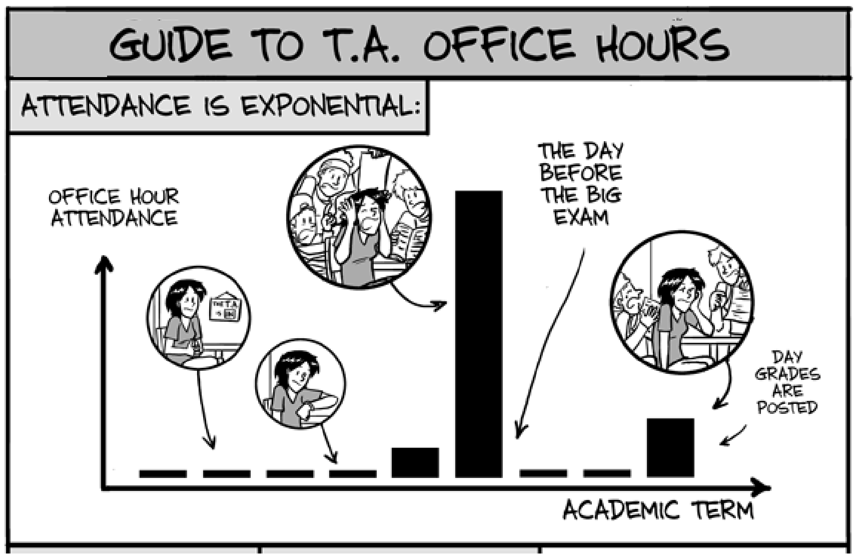
Introducing Your Instructor

Raj Rajkumar
Professor, ECE and RI
CIC 2309
x8-8707
rajkumar@cmu.edu
www.ece.cmu.edu/~raj
Office Hours: Wednesday 12:30pm – 1:30pm
(or by appointment)

Teaching Assistants

• *Name:*	**Isaac Manjares**
E-mail:	imanjarr@andrew.cmu.edu
Office Hours:	*TBA*
• *Name*:	**Aishwarya Prem Renu**
E-mail:	apremren@andrew.cmu.edu
Office Hours:	*TBA*

Thought for the Day


GUIDE TO T.A. OFFICE HOURS
ATTENDANCE IS EXPONENTIAL:
OFFICE HOUR ATTENDANCE
THE DAY BEFORE THE BIG EXAM
DAY GRADES ARE POSTED
ACADEMIC TERM

Course Logistics

- Course Management Assistant
  - **Michelle Mahouski**, Hamerschlag Hall (HH) 1112
  - *Phone*: 412-268-4951
  - *E-mail*: mmahousk@andrew.cmu.edu
- Lectures
  - Mon/Wed 10:30am -12:20pm US East Coast time
  - Porter Hall 125C
- Labs
  - A **few** evening recitations (likely Thursday evenings 7-9 pm)
- Textbook: *None*
  - Any necessary readings and handouts will be given in class and/or
  - Available online on CMU Canvas (or Piazza)

Pre-Requisites

- Pre-requisites
  - CS 211: Fundamental Data Structures and Algorithms
  - ECE 240: Fundamentals of Computer Engineering
  - CS 213: Introduction to Computer Systems

    or

    ECE 347: Introduction to Computer Architecture
- Knowledge of C programming
  - knowing *only* Java could actually be harmful!
    - Need to know, understand and manipulate "pointers" explicitly
- Basic knowledge of digital logic
  - basic gates and their truth tables
- Assembly-language programming
  - the ARM instruction set is useful in 18-648
  - knowledge of instruction sets of other microprocessors (x86, XScale, PowerPC, 680x0, MIPS) is a sufficient starting point.

Cautionary Note on Assembly Programming

- If you do not know assembly language programming, you may need to spend extra effort in coming up to speed as soon as you can
  - A recitation/lecture will be provided

Lab Outline

- 4 Lab projects
  - All labs will be in groups of 3
  - All group members must be present for project demonstrations and will be asked questions
  - Most labs will comprise more than one part
    - Start *early* on each lab
    - The sheer amount of work required for each lab will ensure that ***if*** you start late, you ***will not*** be able to complete.

Miscellaneous Lab Information

- Introducing…. hacking the kernel on a modern tablet and building some simple apps
- Real-world tools: industrial grade (means there will be bugs!)
- You may spend (waste) time because of relative "newness" problems
- Most labs cannot be done in a day or even two.
  - Plan *and* execute early.
  - Last-minute pleas for deadline extensions will be summarily and unceremoniously dismissed.
- Pick your group-mates wisely – always!

Tentative Course Schedule (1 of 2)

Date	Day	Week	Class Activity	Notes
August				
28	Mon.	1	Course Overview	**Classes begin**
30	Wed.		Intro to Embedded Real-Time Systems	
September				
04	Mon.	2	**Labor Day; No Classes**	
06	Wed.		Uniprocessor Scheduling – I	
11	Mon.	3	Uniprocessor Scheduling – II	**Lab #1 handout**
13	Wed.		Device and Memory Management	
18	Mon.	4	Linux and RT-Extensions	
20	Wed.		Performance Optimization	**+ Quiz #1**
25	Mon.	5	Resource Reservations	
27	Wed.		Resource Sharing – I	**Lab #2 handout**
October				
02	Mon.	6	*cont'd.*	
04	Wed.		Resource Sharing – II	**+ Quiz #2**
09	Mon.	7	Power-Aware Scheduling – I	
11	Wed.		Power-Aware Scheduling – II	**Lab #3 handout**
16	Mon.	8	Aperiodics – Fixed-Priority Servers	
18	Wed.		Aperiodics – Dynamic-Priority Servers	**+ Quiz #3**
23	Mon	9	Multiprocessor Scheduling – I	
25	Wed		Multiprocessor Operating Systems	

Tentative Course Schedule (2 of 2)

Date	Day	Week	Class Activity	Notes
30	Mon.	10	Power-Aware Multiprocessor Scheduling	
November				
01	Wed.		Real-Time Communications	**+ Quiz #4**
06	Mon.	11	Overload Management	**Lab #4 handout**
08	Wed.		Hierarchical Scheduling	
13	Mon.	12	Case Studies – I	
15	Wed.		Case Studies – II	**+ Quiz #5**
20	Mon.	13	Feedback Control Theory	
22	Wed.		*Thanksgiving Break; No Classes*	**Have a good break!**
27	Mon.	14		
29	Wed.		Signal Processing	
December				
04	Mon.	15	System-Level Considerations	
06	Wed.		Advanced Topics – I	**Dec 9: Last Day of Classes**
11-15	?		Final Examination	**The exam date and time are picked by the registrar's office**

Final Exam Date

- The final date for the exam will be picked by the registrar's office.
  - The instructor has <u>no</u> control over this date.
  - Do NOT plan *any* travels <u>before</u> you know your final exam schedule.
  - The exam will ***NOT*** be administered for your convenience at an earlier or later date.

Where Could *You* End Up?

- **Automotive systems**
  - perhaps designing and developing "drive-by-wire" systems
- **Telecommunications**
- **Consumer electronics**
  - Smartphone/tablet makers
  - MP3 devices, integrated cellular/walkman/PDA/kitchen sink
  - Home appliances
  - Internet appliances: your washer will be on the internet more than you are!
- **Defense and weapon systems**
- **Process control**
  - gasoline processing, chemical refinement
- **Automated manufacturing**
  - Supervisory Control and Data Acquisition (SCADA)
- **Aerospace applications**
  - Satellite communications

Goals of the Course

High-Level Goals

1. Understand the scientific principles and concepts behind embedded real-time systems, and
2. Obtain hands-on experience in programming embedded real-time systems.

By the end of the course, you must be able to

- Understand the "big ideas" in embedded real-time systems
- Obtain direct hands-on experience on both hardware and software elements commonly used in embedded real-time system design
- Understand basic real-time resource management theory
- Understand the basics of embedded real-time system application concepts such as signal processing and feedback control
- Understand, and be able to discuss and communicate intelligently about
  - I/O and device driver interfaces to embedded processors with networks and multimedia cards
  - OS primitives for concurrency, timeouts, scheduling, communication and synchronization

The Big Ideas

- HW/SW Boundary
- Non-processor-centric view of architecture
- Bowels of the operating system
  - specifically, the *lower* half of the OS
  - Concurrency
- Real-time scheduling
- Analyzability
  - how do you “know” that your drive-by-wire system will function correctly?
- Application-level techniques
  - signal processing, control theory

Grading Criteria

- **Assignments**
  - 200 points from 5 quizzes (50 points each, we drop the lowest score)
  - 200 points from 1 final exam
  - 600 points from 4 projects
    - Currently, 100 from Project 1, 160 from Project 2, 160 from Project 3 and 180 from Project 4
- **Relative Grading Scale**
  - No more than 33% of students will get an 'A'.
  - No more than 47% of students will get a 'B'.
  - Less than 70% in aggregate points will score a 'D'.
  - The others will get a C.
- **Late Policy**
  - A penalty of 10% per day (including Saturday and Sunday)
  - Last day to turn in everything is the *last official day of class for the semester*

Questions and Feedback

- Grades will be available **online** on Canvas or piazza (https://canvas.cmu.edu/courses/1249)
  - Log in using your andrew id and password
- Send **e-mail** to the instructor (`rajkumar@cmu.edu`), the TAs and the course management assistant (mmahousk@andrew.cmu.edu) from the web-page.
- **Post** to the class discussion board from the web-page.
- **Talk to Raj or any of the TAs** if you have any concerns regarding *any*thing related to the course.
- Raj will want to **talk with you** if *you* are **not** doing well in class.
- You can provide **anonymous feedback *anytime*.**
  - Non-anonymous feedback is also welcome in person or by other means.

WHAT ARE EMBEDDED REAL-TIME SYSTEMS?

What are **Embedded Systems**?

- Anything that uses a microprocessor but isn't a general-purpose computer
  - PDAs
  - Settop boxes
  - Televisions
  - Video Games
  - Refrigerators
  - Cars
  - Planes
  - Elevators
  - Remote Controls
  - Alarm Systems
  - Smartphones?
- The user “sees” a smart (special-purpose) system as opposed to the computer inside the system
    - “how does it do that?” “it has a computer inside it!”
    - “oh! BTW, it does not or cannot run Windows or MacOS!”
  - the end-user typically (typically) does not or cannot modify or upgrade the internals

Why are **Embedded Systems** important?

- **Engineering reasons**
  - Why does a satellite need a Windows prompt ?
  - Does the McDonald's POS (point-of-sale) terminal need MacOS?
  - Any device that needs to be controlled can be controlled by a microprocessor
- **Market reasons**
  - The general-purpose computing market is in billions of US $
  - The embedded systems market is also in billions of $
  - In year 2000, about $2,700 of every car went into its electronics
  - In year 2015, expected to be more than a third of every car's value
- **Pedagogical reasons**
  - General-purpose system designers specialize
  - HW vs. SW
- Embedded system designers are often
  - "Jack of many trades"
  - Need to know hardware, software, and some combination of networking, control theory and signal processing
  - business models

What are **Real-Time** Systems?

- Systems where **timing** constraints must be satisfied for functional correctness
  - Correct outputs at the wrong times are not sufficient and can lead to failure
- Examples:
  - Motion control (trains, automobiles, ships/boats, planes, helicopters, spacecraft, elevators …)
  - Process control (gas refinement, chemical manufacture)
  - Manufacturing (automated factories, packaging, bottling)
  - Telecommunications
  - Consumer electronics (video game consoles, mobile phones, set-top boxes, TVs, …)
- The system must be "responsive" to changing needs

What Are You Going to Learn?

- **Hardware**
  - I/O, memory, buses, devices, control logic, interfacing hardware to software
- **Software**
  - C and assembly, some device drivers, some lowlevel OS issues
  - Concurrency
  - Lots of real-time scheduling theory
- **Software/Hardware interactions**
  - Where is the best place to put functionality hardware or software?
  - What are the costs:
    - Performance and memory requirements (RAM and/or ROM)
- **Integration of hardware and software**
  - Programming and logic design
  - Algorithms, mathematics and *common sense*
- **Energy management**
  - Battery is a big constraint in mobile devices

ECE Embedded Systems Faculty

- **Phil Koopman**: building robust embedded systems
- **Bruce Krogh**: control systems, hybrid systems, automotive systems
- **Radu Marculescu**: performance evaluation and power-efficient computing
- **Diana Marculescu**: power-aware computing, VLSI logic
- **Priya Narasimhan**: reliability, security, embedded middleware
- **Raj Rajkumar**: embedded real-time operating systems, Quality of Service (QoS), networking, real-time scheduling theory, cyber-physical systems.
- **Anthony Rowe**: sensor networks, smart grids, cyber-physical systems.
- **Ed Clarke**: model checking and formal methods for verification

Embedded Systems Course #1 in ECE

18-342: Fundamentals of Embedded Systems

This practical, hands-on course introduces students to the basic building-blocks and the underlying scientific principles of embedded systems. The course covers both the hardware and software aspects of embedded procesor architectures, along with operating system fundamentals, such as virtual memory, concurrency, task scheduling and synchronization. Through a series of laboratory projects involving state-of-the-art processors, students will learn to understand implementation details and to write assembly-language and C programs that implement core embedded OS functionality, and that control/debug features such as timers, interrupts, serial communications, flash memory, device drivers and other components used in typical embedded applications. Relevant topics, such as optimization, profiling, digital signal processing, feedback control, real-time operating systems and embedded middleware, will also be discussed. This course is intended for INI students.

- Pre-requisites: 18-240
- Co-requisites: None.

Embedded Systems Course #2 in ECE

18-349: Embedded Real-Time Systems

This practical, hands-on course introduces the various building blocks and underlying scientific and engineering principles behind embedded real-time systems. The course covers the integrated hardware and software aspects of embedded processor architectures, along with advanced topics such as real-time, resource/device and memory management. Students can expect to learn how to program with the embedded architecture that is ubiquitous in cell-phones, portable gaming devices, robots, PDAs, etc. Students will then go on to learn and apply real-time principles that are used to drive critical embedded systems like automobiles, avionics, medical equipment, the Mars rover, etc. Topics covered include embedded architectures (building up to modern 16/32/64-bit embedded processors); interaction with devices (buses, memory architectures, memory management, device drivers); concurrency (software and hardware interrupts, timers); real-time principles (multi-tasking, scheduling, synchronization); implementation trade-offs, profiling and code optimization (for performance and memory); embedded software (exception handling, loading, mode-switching, programming embedded systems). Through a series of laboratory exercises with state-of-the-art embedded processors and industry-strength development tools, students will acquire skills in the design/implementation/debugging of core embedded real-time functionality.

- Pre-requisites: 18-240 and 15-213
- Co-requisites: None.

Embedded Systems Course #3 in ECE

18-549: Embedded Systems Design

This course comprises a semester-long project experience geared towards the development of skills to design realistic and practical embedded systems and applications. Students will work in teams on an innovative project that will involve the hands-on design, configuration, engineering, implementation and testing of a prototype of an embedded system of their choice. Students will be expected to leverage proficiency and background gained from other courses, particularly with regard to embedded real-time principles and embedded programming. The project will utilize a synergistic mixture of skills in system architecture, modular system design, software engineering, subsystem integration, debugging and testing. From inception to demonstration of the prototype, the course will follow industrial project practices, such as version control, design requirements, design reviews and quality assurance plans. The initial lecture content will cover background material intended to complement the project work. The remainder of the course will consist of regular team presentations of key project milestones, current project status, a final project presentation and functional demonstrations of various subsystems, even as the entire prototype is being developed.

- Pre-requisites: 18-348 or 18-349
- Co-requisites: None.

Embedded Systems Course #4 in ECE

18-649: Distributed Embedded Systems

Embedded computers seem to be everywhere, and are increasingly used in applications as diverse as transportation, medical equipment, industrial controls, and consumer products. This course covers how to design and analyze distributed embedded systems, which typically consist of multiple processors on a local area network performing real time control tasks. The topics covered will include issues such as communication protocols, synchronization, real-time operation, fault tolerance, distributed I/O, design validation, and industrial implementation concerns. The emphasis will be on areas that are specific to embedded distributed systems as opposed to general-purpose networked workstation applications. This course assumes that students already know fundamental topics such as interrupts, basic I/O, and uniprocessor scheduling that are commonly taught in introduction-level embedded system courses such as 18-348 and 18-349. Any graduate student who has not taken one of the pre-requ isites is responsible for understanding relevant material necessary for this course. Additionally, all students are responsible for knowing or learning on their own intermediate-level programming in Java. Pre-requisites: 18-348 or 18-349 and senior or graduate standing.

- Pre-requisites: 18-348 or 18-349

Embedded Systems Course #5 in ECE

18-749: Fault-Tolerant Distributed Systems

The course provides an in-depth and hands-on overview of designing and developing fault-tolerant distributed systems. The course covers both the fundamental and advanced concepts of dependability, including replication, atomic multicast, group communication, consistency, checkpointing, transaction processing and fault injection, along with industrial standards and real-world practices for achieving high availability and fault-tolerance. Additional topics include the practical trade-offs and inter-relationships between fault-tolerance and other properties, such as real-time and performance. The lecture concepts are complemented through a semester-long hands-on project that involves the design, implementation and empirical evaluation of a distributed fault-tolerant, high-performance distributed system. To introduce students to the state-of-the-art technologies, the project emphasizes the use of object-oriented middleware, such as CORBA and EJB. 3 hrs. lec., 9 hrs. lab. Prerequisites: Experience in programming and senior or graduate standing.

- Pre-requisites: None.
- Co-requisites: None.

Embedded Systems Course #6 in ECE

18-848: Special Topics in Embedded Systems - Wireless Sensor Networks

The primary objectives of this course are to

1. Obtain a broad understanding of the technologies and applications for the emerging and exciting domain of wireless sensor networks, and
2. Get in-depth hands-on experience in designing and developing a real operational wireless sensor network system.

- Pre-requisites: 18-342 or 18-349.
- Co-requisites: None.

Cyber-Physical Systems

18-848: Special Topics in Embedded Systems - Networked Cyber-Physical Systems

Prof. Anthony Rowe

- This course studies the latest research in the emerging field of Cyber-Physical Systems. Cyber-Physical Systems are a class of networked embedded systems where sensing, computation and actuation are tightly integrated with the dynamics of the physical environment. This course will address the new challenges designers face as these systems are connected together forming large-scale networks that sense, monitor and control elements of the physical environment in real-time. We will study applications in the domain of safety-critical systems, smart grid technology, intelligent buildings, Supervisory Control and Data Acquisition (SCADA) systems used in manufacturing and utilities, automotive systems, and body sensor networks. This includes a broad range of topics related to networked embedded systems including: transducer hardware, operating systems, time synchronization, energy management, modeling, simulation and security. Networking topics will primarily focus on the principles behind deterministic embedded wired protocols, recent advances in Power-Line Communications (PLC) and wireless sensor networking.

Cheating and Plagiarism

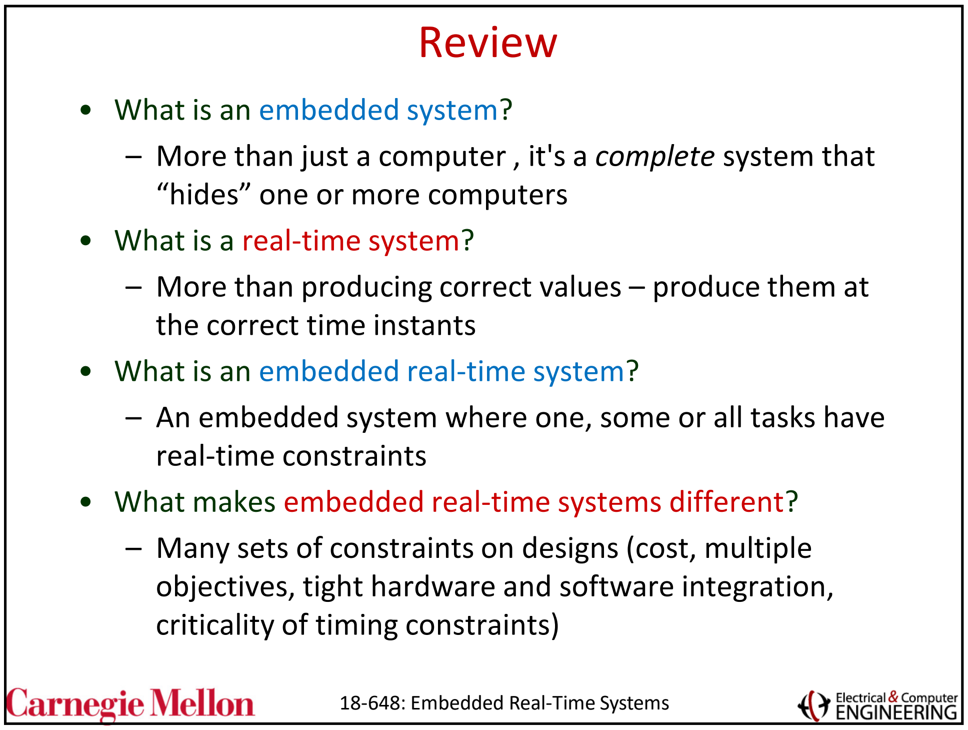
- Read the CIT policy regarding its cheating and plagiarism policy.
- Please follow it religiously! **Do NOT mess up your career**.
  - Many of you have come a long way.
  - It’s a *much* longer journey back home if you fail the course, go on probation, fail probation requirements, have to quit the program and go back (with all pride lost).

Mobile Phones and Laptops in Class

- You are allowed to take notes on your laptop, but you **must turn the sound off** so that you do not disrupt other students' learning.
- If you are doing anything other than taking notes on your laptop, please sit in the back row(s) so that other students are not distracted by your screen.
- Mobile phones must be turned off or be in "vibrate" mode.

Recording of Class Sessions

- Classroom **activities may be taped or recorded by a student for the personal, educational use** of that student or for all students presently enrolled in the class only
- Recordings may not be further copied, distributed, published or otherwise used for any other purpose without the express written consent of the instructor.



Review

- What is an embedded system?
  - More than just a computer , it's a *complete* system that “hides” one or more computers
- What is a real-time system?
  - More than producing correct values – produce them at the correct time instants
- What is an embedded real-time system?
  - An embedded system where one, some or all tasks have real-time constraints
- What makes embedded real-time systems different?
  - Many sets of constraints on designs (cost, multiple objectives, tight hardware and software integration, criticality of timing constraints)