Real Time & Embedded Systems

- Your lecturers:
  - Prof. J Sventek, weeks 1-5, joe@dcs.gla.ac.uk
  - Dr. C. Perkins, weeks 6-10, csp@csperkins.org
- Venues and times
  - Tuesdays, 15:00-16:00, [F171]
  - Wednesdays, 12:00-13:00, [F171]
  - Thursdays, 12:00-13:00, [Boyd Orr 407/LT A]
- Required textbook – "Real-Time Systems" by Jane W. S. Liu, ISBN 0-13-099651-31
- Web site: **http://www.dcs.gla.ac.uk/people/personal/joe/03-04RTES.html**

Real Time & Embedded Systems

- Course overview:
  - Lectures 1-10 – theory of real-time systems, covering scheduling and resource management
  - Lectures 11-20 – the pragmatics of building real-time systems with available operating systems and network stacks
- Two assessed problem sets in weeks 1-5
- Assessed programming exercise in weeks 6-10
- 25% of grade derived from assessed course work; 75% of grade derived from your exam mark

Real Time & Embedded Systems

Week beginning	Tue, 15:00-16:00	Wed, 12:00-13:00	Thu, 12:00-13:00
12 January	No meeting	Lecture 1	Lecture 2
19 January	Q&A	Lecture 3	Lecture 4
26 January	Q&A	Lecture 5	Lecture 6
2 February	Q&A	Lecture 7	Lecture 8
9 February	Q&A	Lecture 9	Lecture 10
16 February	Lecture 11	Lecture 12	Lecture 13
23 February	Lecture 14	Lecture 15	Lecture 16
1 March	Individual work on programming assignment		
8 March	Q&A	Lecture 17	Lecture 18
15 March	Q&A	Lecture 19	Lecture 20

Real Time & Embedded Systems Weeks 1-5

Topic	Lectures	Pre-Reading
Typical Real-Time Applications	1	Chapter 1
Hard vs. Soft RT Systems	1	Chapter 2
Reference Model of RT Systems	2	Chapter 3
Commonly Used Approaches to RT Scheduling	3	Chapter 4
Clock-driven Scheduling	4	Chapter 5
Priority-driven Scheduling of Periodic Tasks	5-6	Chapter 6
Scheduling Aperiodic and Sporadic Jobs in Priority-Driven Systems	7-8	Chapter 7
Resources and Resource Access Control	9-10	Chapter 8

Real Time & Embedded Systems Weeks 6-10

Topic	Lecture	Pre-Reading
Real-Time Support in Operating Systems	11	Chapter 12
Scheduling in Practice	12	
Operating System Support for Concurrency	13	
Introduction to Real-Time Communications	14	Chapter 11
Real-Time Communications on IP Networks	15	
Network Quality of Service	16	
Real-Time on General Purpose Systems	17	Chapter 10
Real-Time Embedded Systems	18	
Low-level Programming	19	
Review of Major Concepts	20	

Typical Real-Time Applications

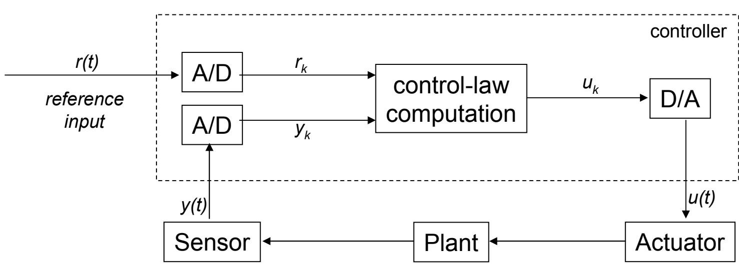
- A real-time system is required to complete its work and deliver its services on a timely basis
- The computers and networks that run real-time applications are often hidden from view (embedded) – successful, embedded RT systems are not seen by the user
- Some RT systems are safety critical – i.e. if they do not complete on a timely basis, serious consequences result
- Therefore, it is crucial that one be able to validate RT systems – i.e. provide a rigorous demonstration that the system has the intended timing behaviour

Typical Real-Time Applications

- We will discuss several representative classes of RT Applications
  - Digital control
  - Optimal control
  - Command and control
  - Signal processing
  - Tracking
  - Real-time databases
  - Multimedia

Typical Real-Time Applications

- Digital Control



Typical Real-Time Applications

- Digital Control
  - A sampled data system
  - y(t) is the measured state of the plant
  - r(t) is the desired state of the plant
  - e(t) = r(t) – y(t) is the difference between desired and measured
  - A proportional, integral and derivative (PID) controller has the output u(t) of the controller that consists of three terms: one proportional to e(t), a second proportional to the integral of e(t), and a third that is proportional to the derivative of e(t)
  - Pseudocode for the controller

```
set timer to interrupt periodically with period T;
at each timer interrupt, do
        do analog-to-digital conversion to get y;
        compute control output u;
        output u and do digital-to-analog conversion;
end do;
```


Typical Real-Time Applications

- Selection of sampling period
  - ***Sampling period*** – the length of time T between any two consecutive instants at which y(t) and r(t) are sampled
  - Making T small better approximates the analog behaviour
  - Making T large means less processor-time demands
  - Must achieve a compromise
  - Perceived responsiveness – if users can provide input at any time t, then the response to the input can be as late as t+T; if T is too large, the user will perceive the system as sluggish
  - Want to keep the oscillation in the plant's response small and the system under control

Typical Real-Time Applications

- Selection of sampling period
  - ***Rise time*** – the amount of time that the plant takes to reach some small neighbourhood around the final state in response to a step change in the reference input
  - If R is the rise time, and T is the period, a good rule of thumb is that the ratio 10 <= R/T <= 20
- Multirate Systems – system is composed of multiple sensors and actuators, each of which require different sampling periods
- Usually best to have the sampling periods for the different degrees of freedom related in a harmonic way

Typical Real-Time Applications

Flight controller for a helicopter

- Validate sensor data and select data source; in the presence of failures, reconfigure the system
- Do the following 30-Hz avionics tasks, each once every 6 cycles:
  - Keyboard input and mode selection
  - Data normalization and coordinate transformation
  - Tracking reference update
- Do the following 30-Hz computations, each once every 6 cycles
  - Control laws of the outer pitch-control loop
  - Control laws of the outer roll-control loop
  - Control laws of the outer yaw- and collective-control loop
- Do each of the following 90-Hz computations once every 2 cycles, using outputs produced by the 30-Hz computations
  - Control laws of the inner pitch-control loop
  - Control laws of the inner roll- and collective-control loop
- Comput the control laws of the inner yaw-control loop, using outputs from the 90-Hz computations
- Output commands
- Carry out built-in-test
- Wait until the beginning of the next cycle

Typical Real-Time Applications

- PID controllers make three assumptions:
  - Sensor data give accurate estimates of the state-variables being monitored and controlled - noiseless
  - The sensor data gives the state of the plant – usually must compute plant state from measured values
  - All parameters representing the dynamics of the plant are known
- If any of these assumptions are not valid, digital controllers are often implemented as follows:

```
set timer to interrupt periodically with period T;
at each clock interrupt, do
        sample and digitize sensor readings to get measured values;
        compute control output from measured and state-variable values;
        convert control output to analog form;
        estimate and update plant parameters;
        compute and update state variables;
end do;
```


Typical Real-Time Applications

- Divide RT applications into the following four types according to their timing attributes:
  - Purely cyclic: every task executes periodically; its demands in (computing, communication, and storage) resources do not vary significantly from period to period
  - Mostly cyclic: most tasks execute periodically; the system must also respond to some external events (fault recovery and external commands) asynchronously
  - Asynchronous and somewhat predictable: most tasks are not periodic; the durations between consecutive executions of a task may vary considerably, or the variations in resource utilization in different periods may be large; these variations have either bounded ranges or known statistics
  - Asynchronous and unpredictable: applications that react to asynchronous events and have tasks with high run-time complexity

Typical Real-Time Applications

- Examples
  - Purely cyclic – most digital controllers and real-time monitors
  - Mostly cyclic – modern avionics and process control systems
  - Asynchronous, predictable –
  - Asynchronous, unpredictable – intelligent real-time control systems

Hard vs. Soft Real-Time Systems

- Each unit of work that is scheduled and executed by a system is a ***job*** – e.g. computation of a control-law, computation of an FFT on sensor data, transmission of a data packet, retrieval of a file
- a set of related jobs which jointly provide some system function is a ***task*** – e.g. the set of jobs that constitute the "maintain constant altitude" task – i.e. keep an airplane flying at a constant altitude
- A job executes or is executed by the operating system
- Every job executes on some resource; each such resource is termed a ***processor***

Hard vs. Soft Real-Time Systems

- ***Release time*** – the instant of time at which a job becomes available for execution
- A job can be scheduled and executed at any time at or after its release time whenever its data and control dependency conditions are met
- If all jobs are released when the system begins execution, then these jobs "have no release time"
- ***Deadline*** – the instant of time by which a job's execution is required to be completed (also called ***absolute deadline***)
- ***Response time*** – the length of time from the release time of the job to the instant when it completes
- ***Relative deadline*** – the maximum allowable response time of a job
- absolute deadline = release time + relative deadline
- ***Completion time*** – the instant of time at which a job completes execution
- ***Timing constraint*** – any constraint imposed on the timing behaviour of a job

Hard vs. Soft Real-Time Systems

- Example: a system that monitors and controls several furnaces; after initialization, every 100 ms, the system:
  - samples and reads each temperature sensor
  - computes the control-law of each furnace in order to process the temperature readings and determine the flow rates of fuel, air, coolant
- The periodic control-law computations can be stated in terms of the release times of the control-law computation jobs J_0, J_1, …, J_k, …; if we assume that 20 ms are required for initialization, then J_k's release time is 20 + k x 100 ms
- Suppose each job must complete before the release of the subsequent job – i.e. J_k's relative deadline is 100 ms; then J_k's absolute deadline is 20 + (k + 1) x 100 ms
- Alternatively, each control-law computation may be required to finish sooner – i.e. the relative deadline is smaller than the time between jobs

Hard vs. Soft Real-Time Systems

- ***Tardiness*** – how late a job completes relative to its deadline; `0` if at or before its absolute deadline, otherwise,
  `completion time - absolute deadline`
- Common definitions for hard and soft RT constraints:
  - A timing constraint is hard if the failure to meet it is considered to be a fatal fault (e.g. failure to release a bomb on time causes civilians around a military target to be hit instead) – this definition is based upon the functional criticality of a job
  - The usefulness of the results of a job can be used to define hard vs soft RT – if the usefulness falls off abruptly at the deadline (or may even go negative), then it is a hard RT constraint
  - If a job must never miss its deadline, then it is a hard RT constraint; if the deadline can be missed occasionally with some acceptably low probability, then it is a soft RT constraint

Hard vs. Soft Real-Time Systems

- ***Validation*** – a demonstration by a provably correct, efficient procedure or by exhaustive simulation and testing.
- The timing constraint of a job is ***hard***, and the job is a hard RT job, if the user requires the validation that the system always meets the timing constraint.
- ***Statistical constraint*** – a timing constraint specified in terms of statistical averages
- If no validation is required, or only a demonstration that the job meets some statistical constraint is needed, then the timing constraint on the job is ***soft***
- Guaranteed vs best-effort
- An application/task with hard timing constraints is a hard RT application/task; a system containing mostly hard RT applications/tasks is a hard RT system

Hard vs. Soft Real-Time Systems

- There may be no advantage in completing a job with a hard deadline early – in fact, it is often advantageous/essential to keep jitter in the response times of a stream of jobs small
- Hard timing constraints can be expressed in many ways:
  - Deterministic – e.g. the relative deadline of every control-law computation is 50 ms; the response time of at most 1 out of 5 consecutive control-law computations exceeds 50ms *
  - Probabilistic – e.g. the probability of the response time exceeding 50 ms is less than 0.2
  - In terms of some usefulness function – e.g. the usefulness of every control-law computation is at least 0.8