## Topic 1 Notes

### Introduction to Realtime Systems

An embedded system is "a specialized computer system that is part of a larger system or machine. Typically, an embedded system is housed on a single microprocessor board with the programs stored in ROM. Virtually all appliances that have digital interfaces (e.g., watches, microwaves, VCRs, cars) utilize embedded systems" Many embedded systems are real-time systems
Real-time system is a system whose specification includes both logical and temporal correctness requirements.
Logical Correctness: Produces correct outputs. It can be checked, for example, by Hoare logic.
Temporal Correctness: Produces outputs at the right time. In this course, we spend much time on techniques and technologies for achieving and checking temporal correctness.

### Terminologies

a. Hard real-time — systems where it is absolutely imperative that responses occur within the required deadline, e.g. A flight control system
b. Soft real-time — systems where deadlines are important but which will still function correctly if deadlines are occasionally missed. E.g. Data acquisition system
c. Firm real-time — systems which are soft real-time but in which there is no benefit from late delivery of service
d. A single system may have hard, soft and firm real-time subsystems. In reality many systems will have a cost function associated with missing each deadline.
e. Time-aware — system makes explicit reference to time (eg. open vault door at 9.00
f. Reactive — system must produce output within deadline (as measured from input)
g. Control systems are reactive systems
h. Required to constraint input and output (time) variability, input jitter and output jitter control
i. Time-triggered — computation is triggered by the passage of time
j. Release activity at 9.00
k. Release activity every 25ms – a periodic activity
l. Event-trigger — computation is triggered by an external or internal event
m. The released activity is called sporadic if there is a bound on the arrival interval of the event
n. The released activity is called aperiodic if there is no such bound

### Misconceptions about Real-Time Systems

i. There is no science in real-time-system design.

ii. Advances in supercomputing hardware will take care of real-time requirements.The old "buy a faster processor" argument…

iii. Real-time computing is equivalent to fast computing. Only to ad agencies. To us, it means PREDICTABLE computing.

iv. Real-time programming is assembly coding, We would like to automate (as much as possible) real-time system design, instead of relying on clever hand-crafted code.

v. "Real time" is performance engineering. In real-time computing, timeliness is almost always more important than raw performance …

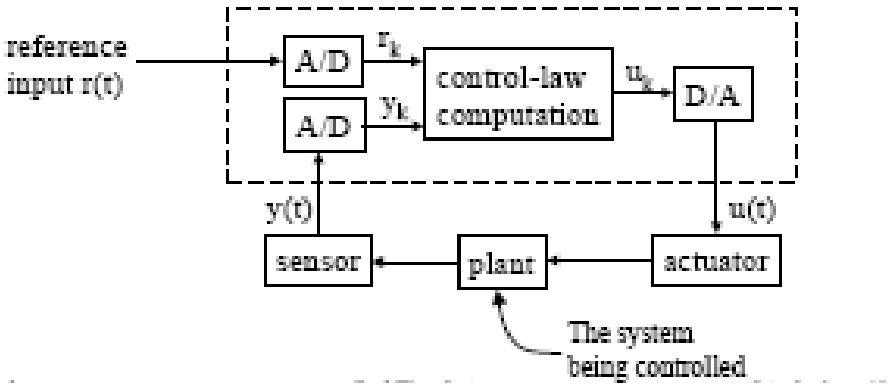
vi. "Real-time problems" have all been solved in other areas of CS or operations research. OR people typically use stochastic queuing models or one-shot scheduling models to reason about systems.

vii. In other CS areas, people are usually interested in optimizing average-case performance.

**Real time applications**

Many real-time systems are control systems

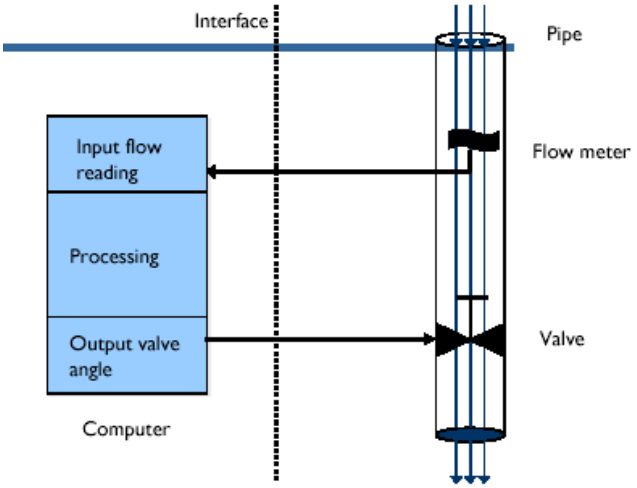
Example 1: A simple one-sensor, one-actuator control system




Pseudo-code for this system:

```
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;
od
```

T is called sampling period. T is a key design choice. Typical range for T: seconds to milliseconds.




**Muiltirate control systems**

More complicated control systems have multiple sensors and actuators and must support control loops of different rates.

Example 2: Helicopter flight controller.

Do the following in each 1/180-sec. cycle:
validate sensor data and select data source;
if failure, reconfigure the system

Every sixth cycle do:
keyboard input and mode selection;
data normalization and coordinate transformation;
tracking reference update
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

Every other cycle do:
control laws of the inner pitch-control loop;
control laws of the inner roll- and collective-control loop

Compute the control laws of the inner yaw-control loop;

Output commands;

Carry out built-in test;

Wait until beginning of the next cycle

Note: Having only harmonic rates simplifies the system

**Siginal processing systems**

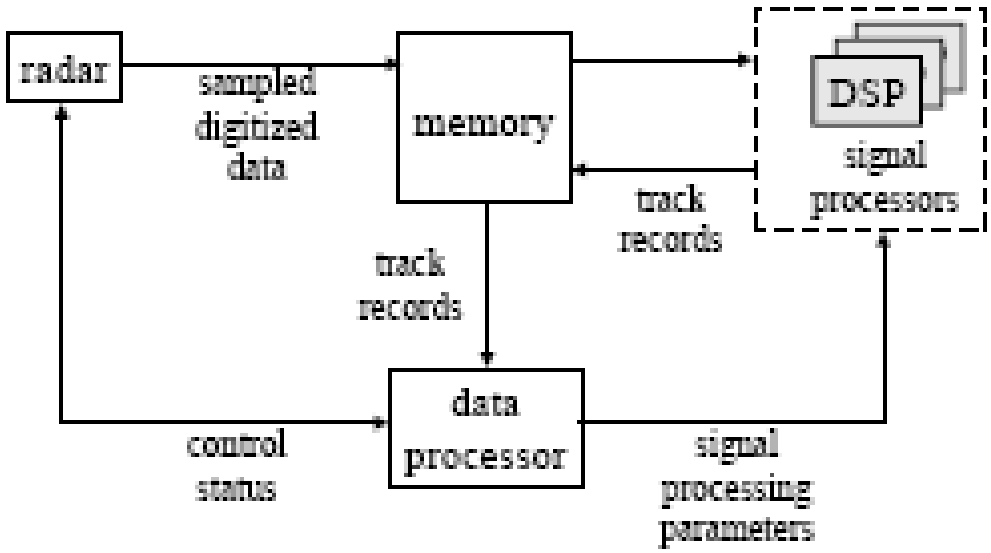
Signal-processing systems transform data from one form to another.

Examples: Digital filtering.

Video and voice compression/decompression.

Radar signal processing.

Response times range from a few milliseconds to a few seconds.




Other real time Applications

**Real-time databases.**

Transactions must complete by deadlines.

Main dilemma: Transaction scheduling algorithms and real-time scheduling algorithms often have conflicting goals. Data may be subject to absolute and relative temporal consistency requirements.

Overall goal: reliable responses

**Multimedia.**

Want to process audio and video frames at steady rates.

a. TV video rate is 30 frames/sec. HDTV is 60 frames/sec.
b. Telephone audio is 16 Kbits/sec. CD audio is 128 Kbits/sec.

**Other requirements:** Lip synchronization, low jitter, low end-to-end response times (if interactive).

**Typical Characteritics of real time systems.**

i. Event-driven, reactive.

ii. High cost of failure.

iii. Concurrency/multiprogramming.

iv. Stand-alone/continuous operation.

v. Reliability/fault-tolerance requirements.

vi. Predictable behavior.

**Real Time Operating Systems**

RTOS: specialized operating system for RTS Main responsibilities:

a. Process management
b. Resource allocation (processor, memory, network)

They may not include regular OS facilities such as file management, virtual memory, user/kernel level separation, etc.

Manage at least two priority levels:

a. Interrupt level, for processes that need fast response
b. Clock level, for periodic processes

Typical components: real-time clock, interrupt handler, scheduler, resource manager, dispatcher

**Real Time Programming Languages**

i. Assembly languages
ii. Sequential systems implementation languages — e.g. RTL/2, Coral 66, Jovial, C.
iii. Both normally require operating system support.
iv. High-level concurrent languages. Impetus from the software crisis. e.g. Ada, Chill, Modula-2, Mesa, Java.
v. No operating system support!
vi. We will consider:
vii. Java/Real-Time Java
viii. C and Real-Time POSIX (not in detail)
ix. Ada 2005