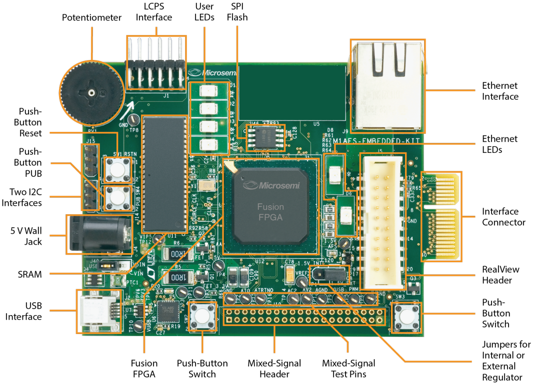


Introduction to real-time systems

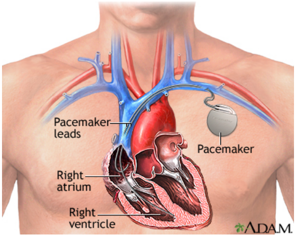
Kizito NKURIKIYEYEZU, Ph.D

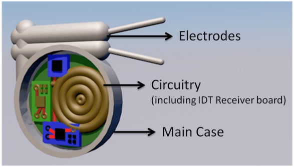
TAB 1. Embedded system programming paradigms

Bare metal[1]	■ simple processors ■ simple devices ■ few operations ■ you already know this	
RTOS[2]	■ devices with multitasking ■ strict deadlines ■ powerful processors ■ complex devices	
Embedded Linux[3]	■ very complex application ■ file-systems,networking ■ Pretty UI	

[1] https://www.embeddedrelated.com/thread/5762/rtos-vs-bare-metal
[2] https://en.wikipedia.org/wiki/Real-time_operating_system

Why real-time systems?





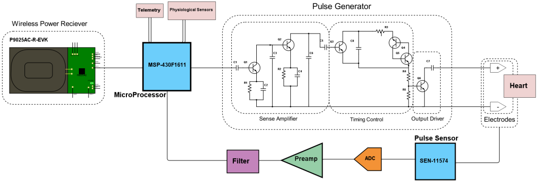


FIG 1. A pacemaker is a small, battery-operated device. This device senses when your heart is beating irregularly or too slowly. It sends a signal to your heart that makes your heart beat at the correct pace. In general, a hear pacemaker contains a small micro-controller and electrodes that connect the heart tot the generator. The electrodes carry the electrical message to the heart. A defective pacemaker can cause more harm

Why real-time systems?

- How can we prove that an unmanned aerial vehicle (UAV) will brake quickly enough if it encounters an object on its path?
- The possibility of life-or-death decisions being taken by an UAV not under the direct control of humans needs to be taken seriously
- In short, how do you know that a UAV military drone will work as expected?

FIG 2. General Atomics MQ-9 Reaper The MQ-9 is capable of remotely controlled or autonomous flight operations and is primarily for the United States Air Force (USAF).

[1] https://en.wikipedia.org/wiki/General_Atomics_MQ-9_Reaper

[2] What is the implication of drone warfare? The following link debates their

Why real-time systems?

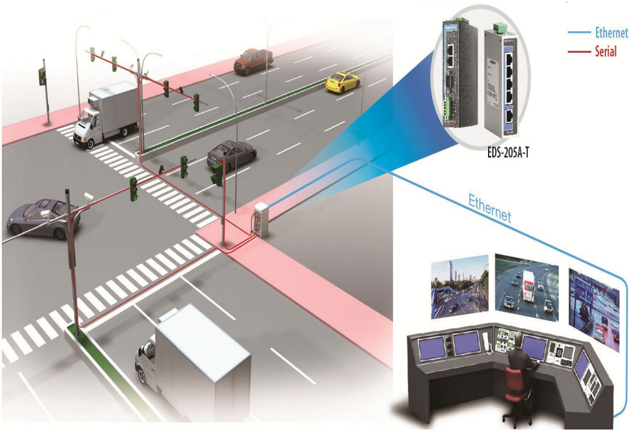


FIG 3. Traffic lights—How do you guarantee that cars won't clash into each other?

Why real-time systems?

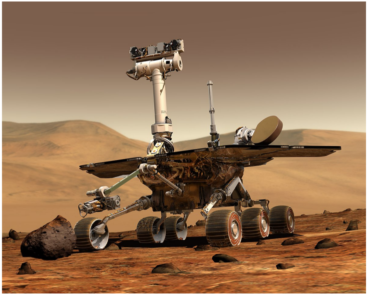


FIG 4. Artist's conception of NASA's Mars Exploration Rover on Mars. It's mission almost failed due priority inversion.

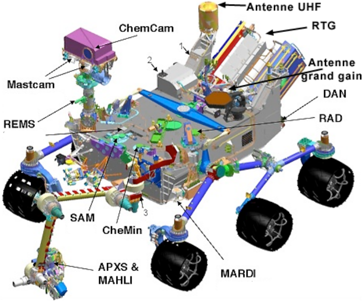


FIG 5. Instrumentation of the Mars Rover

[2] http://www.cs.cornell.edu/courses/cs614/1999sp/papers/pathfinder.html

Example —A real-time control system

- Inputs are excitations and outputs are corresponding responses
- Inputs and outputs may be digital or analog
- Inputs are associated with sensors, cameras, etc.
- Outputs with actuators, displays, etc.

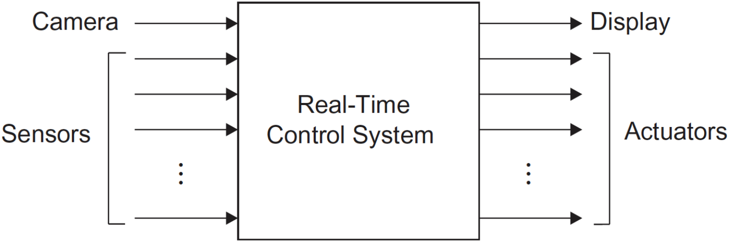


FIG 6. A real-time control system including inputs from a camera and multiple sensors, as well as outputs to a display and multiple actuators

Representation of real-time system

- A sequence of jobs to be scheduled and performance to be predicted
- Ignores the usual fact that the input sources and hardware under control may be highly complex

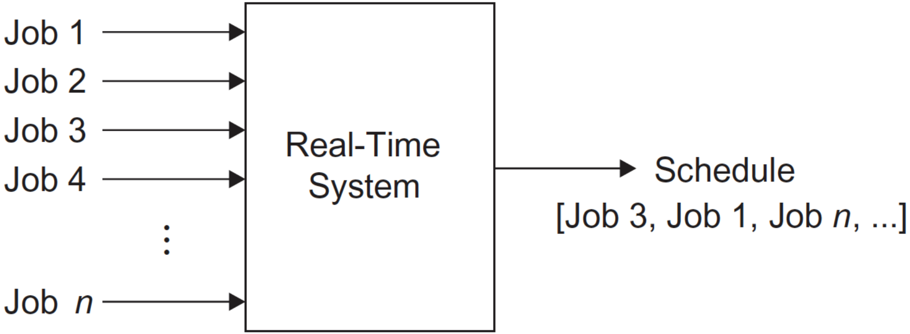


FIG 7. A classic representation of a real-time system as a sequence of schedulable jobs.

Response Time

The time between the presentation of a set of inputs to a system and the realization of the required behavior, including the availability of all associated outputs, is called the response time of the system

- How fast and punctual does it need to be? —Depends on the specific real-time system
- But what is a real-time system if every system somehow has deadlines and a response time?

Real-Time System

- A real-time system is a computer system that must satisfy bounded response-time constraints or risk severe consequences, including failure
- Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced.
- Real-time systems respond to external events in a timely fashion and the response time is guaranteed

Failed System

- A failed system is a system that cannot satisfy one or more of the requirements stipulated in the system requirements specification

Real-time system

- Responding to external events includes
  - Recognize when an event occurs
  - Perform the required processing
  - Output the result within a given time constraint
- Timing constraints include
  - Finish time
  - Both start time and finish time
- Example: DVD player
  - Controlling system —DVD player must decode both the video and audio streams from the disc simultaneously
  - Controlled system —Remote control is viewed as a sensor to feed pause and language selection events into DVD player

Degrees of “Real-Time”

Soft Real-Time System

A soft real-time system is one in which performance is degraded but not destroyed by failure to meet response-time constraints

Firm Real-Time System

A firm real-time system is one in which a few missed deadlines will not lead to total failure, but missing more than a few may lead to complete or catastrophic system failure

Hard Real-Time System

A hard real-time system is one in which failure to meet even a single deadline may lead to complete or catastrophic system failure.

Degrees of "Real-Time"—Examples

- Hard real-time software systems have a set of strict deadlines, and missing a deadline is considered a system failure —airplane sensor and autopilot systems, spacecrafts and planetary rovers.
- Firm real-time systems treat information delivered/computations made after a deadline as invalid. Like soft real-time systems, they do not fail after a missed deadline, and they may degrade QoS if a deadline is missed —financial forecast systems, robotic assembly lines.
- Soft real-time systems try to reach deadlines but do not fail if a deadline is missed. However, they may degrade their quality of service in such an event to improve responsiveness. —audio and video delivery software for entertainment (lag is undesirable but not catastrophic).

Real-Time Punctuality

Real-time punctuality means that every response time has an average value, t_R, with upper and lower bounds of $t_R + \epsilon_U$ and $t_R - \epsilon_L$, respectively, and $\epsilon_U, \epsilon_L \rightarrow 0^+$

- In all practical systems, the values of ϵ_U and ϵ_L are nonzero, though they may be very small
- The nonzero values are due to cumulative latency and propagation-delay components (hardware/software)
- Such response times contain jitter[4] within the interval $t \in [-\epsilon_L, +\epsilon_U]$

[4] jitter is the amount of variation in response time

Where Do Deadlines Come from?

- Deadlines are based on the underlying physical phenomena of the system under control
- Punctuality is another measure related to response times for example in periodically sampled systems with high sampling rates (e.g., in audio and video signal processing)
- Examples:
  - An elevator door is automatically operated and it may have a capacitive safety edge for sensing possible passengers between the closing door blades
  - Thus, the door blades can be quickly reopened before they touch the passenger and cause discomfort or even threaten the passenger's safety
  - What is the required system response time from when it recognizes that a passenger is between the closing door blades and starting to reopen the door?

Where Do Deadlines Come from?

This response time consists of five independent latency components:

1. Sensor[5] ($t_{SE_{min}=5ms}$, $t_{SE_{max}=15ms}$)
2. Hardware ($t_{HW_min=1\,\mu s}$, ,$t_{HW_max=2\,\mu s}$)
3. System software[6] ($t_{SS_min=16\,\mu s}$, $t_{SS_max=48\,\mu s}$)
4. Application software [7]($t_{AS_min=0.5\,\mu s}$, $t_{HW_max=0.5\,\mu s}$)
5. Door drive[8] ($t_{DD_min=300\,ms}$, $t_{HW_max=500\,ms}$)

Consequently:

- minimum response time—$t_{min} = 305ms$
- maximum response time—$t_{max} = 515ms$

[5] Infrared beam sensors used to detect a person and prevent the doors from closing if a person or an object blocks the doorway

[6] System Software is a set of programs that control and manage the operations of computer hardware. It also helps application programs to execute correctly. Example: Operating system, programming language, Communication software, etc.

[7] Application Software acts as a mediator between the end-user and System

Event

Any occurrence that causes the program counter to change non-sequentially is considered a change of flow-of-control, and thus an event

Release Time

The release time is the time at which an instance of a scheduled task is ready to run, and is generally associated with an interrupt

Taxonomy[9] of Events

- An event can be either synchronous or asynchronous
  - Synchronous events are those that occur at predictable times in the flow-of-control
  - Asynchronous events occur at unpredictable points in the flow-of-control and are usually caused by external sources
- Moreover, events can be periodic, aperiodic or sporadic
  - A real-time clock that pulses regularly is a periodic event
  - Events that do not occur at regular periods are called aperiodic
  - Aperiodic events that tend to occur very infrequently are called sporadic

TAB 2. Various Types of Events

Type	Periodic	Aperiodic	Sporadic
Synchronous	Cyclic code	Conditional branch	Divide-by-zero (trap) interrupt
Asynchronous	Clock interrupt	Regular, but not fixed-period interrupt	Power-loss alarm

[9] Taxonomy is the practice and science of categorization or classification

Deterministic systems

- For any physical system, certain states exist under which the system is considered to be out of control
- The software controlling such a system must therefore avoid these states
- In embedded real-time systems, maintaining overall control is extremely important
- Software control of any real-time system and associated hardware is maintained when the next state of the system, given the current state and a set of inputs, is predictable

Deterministic system

A system is deterministic, if for each possible state and each set of inputs, a unique set of outputs and next state of the system can be determined

Deterministic systems

- Event determinism means the next states and outputs of a system are known for each set of inputs that trigger events
- Thus, a system that is deterministic is also event deterministic
- However, event determinism may not imply determinism
- While it is a significant challenge to design systems that are completely event deterministic, it is possible to inadvertently end up with a system that is non-deterministic
- Finally, if in a deterministic system the response time for each set of outputs is known, then, the system also exhibits temporal determinism
- A side benefit of designing deterministic systems is that guarantees can be given that the system will be able to respond at any time, and in the case of temporally deterministic systems, when they will respond

CPU Utilization

- The final term to be defined is a critical measure of real-time system performance
- Because the CPU continues to execute instructions as long as power is applied, it will more or less frequently execute instructions that are not related to the fulfillment of a specific deadline
- The measure of the relative time spent doing non-idle processing indicates how much real-time processing is occurring

CPU Utilization Factor

The CPU utilization or time-loading factor, U, is a relative measure of the non-idle processing taking place

CPU Utilization Zones

- A system is said to be time-overloaded if $U > 100\%$
- Systems that are too highly utilized are problematic —additions, changes, or corrections cannot be made to the system without risk of time-overloading
- On the other hand, systems that are not sufficiently utilized are not necessarily cost-effective —The system was over-engineered and that costs could likely be reduced with less expensive hardware
- While a utilization of 50% is common for new products, 80% might be acceptable for systems that do not expect growth
- However, 70% as a target for U is one of the potentially useful results in the theory of real-time systems where tasks are periodic and independent

TAB 3. CPU Utilization Zones

CPU utilization [in %]	Zone Type	Typical Application
< 26	Unnecessarily safe	Various
26 – 50	Very safe	Various
51 – 68	Safe	Various
69	Theoretical limit	Embedded systems
70 – 82	Questionable	Embedded systems
83 – 99	Dangerous	Embedded systems
100	Critical	Marginally stressed s
> 100	Overloaded	Stressed system

Calculation of the CPU utilization

- U is calculated by summing the contribution of utilization factors for each task
- Suppose a system has$n \geq 1$ periodic tasks, each with an execution period of p_i
- If task i is known to have a worst-case execution time of e_i, then the utilization factor,u_i, for task i is given in Equation (1)

$$u_i = \frac{e_i}{p_i} \tag{1}$$

- The overall system utilization factor, U, is given by Equation (2)

$$\begin{aligned} U &= \sum_{n}^{i=1} u_i \\ &= \sum_{n}^{i=1} \frac{e_i}{p_i} \end{aligned} \tag{2}$$

- In practice, the determination of e_i can be difficult, in which case estimation or measuring must be used
- For aperiodic and sporadic tasks, u_i is calculated by

Example—calculation of CPU utilization

An individual elevator controller in a bank of elevators has the following tasks with execution periods of p_i and worst-case execution times of $e_i \in [1, 2, 3, 4]$ as shown in Table 5:

- Task 1—Communicate with the group dispatcher.
- Task 2—Update the car position information and manage floor-to-floor runs as well as door control.
- Task 3—Register and cancel car calls.
- Task 4—Miscellaneous system supervisions.

TAB 4. CPU utilization

i	e_i	p_i	$\frac{e}{p}$
1	17	500	0.0
2	4	25	0.1
3	1	75	0.0
4	20	200	0.1

The CPU utilization is computed as Equation (2)

TAB 5. CPU utilization

i	e_i	p_i	$\frac{e_i}{p_i}$
1	17	500	0.03
2	4	25	0.16
3	1	75	0.01
4	20	200	0.10
U			**0.31%**

Practical Real-Time Embedded Systems

- Aerospace
  - Flight control
  - Navigation
  - Pilot interface
- Automotive
  - Airbag deployment
  - Antilock braking
  - Fuel injection
- Household
  - Microwave oven
  - Rice cooker
  - Washing machine
- Industrial
  - Crane
  - Paper machine
  - Welding robot
- Multimedia
  - Console game
  - Home theater
  - Simulator
- Medical
  - Intensive care monitor
  - Magnetic resonance imaging
  - Remote surgery

The end