Chapter 1
The Real-Time Environment

Objectives

What are you about to learn?

Knowledge Objectives

- Know the definition of a real-time system
- Understand a simple model for a real-time system
- Know about the different forms of requirements for real-time systems (functional, temporal, data collection, dependability)
- Understand real-time systems classification

Skill Objectives

- Compute temporal requirements like maximum rise time or AD-converter sampling frequencies from a given set of parameters
- Compute dependability attributes like reliability or availability from a given set of parameters

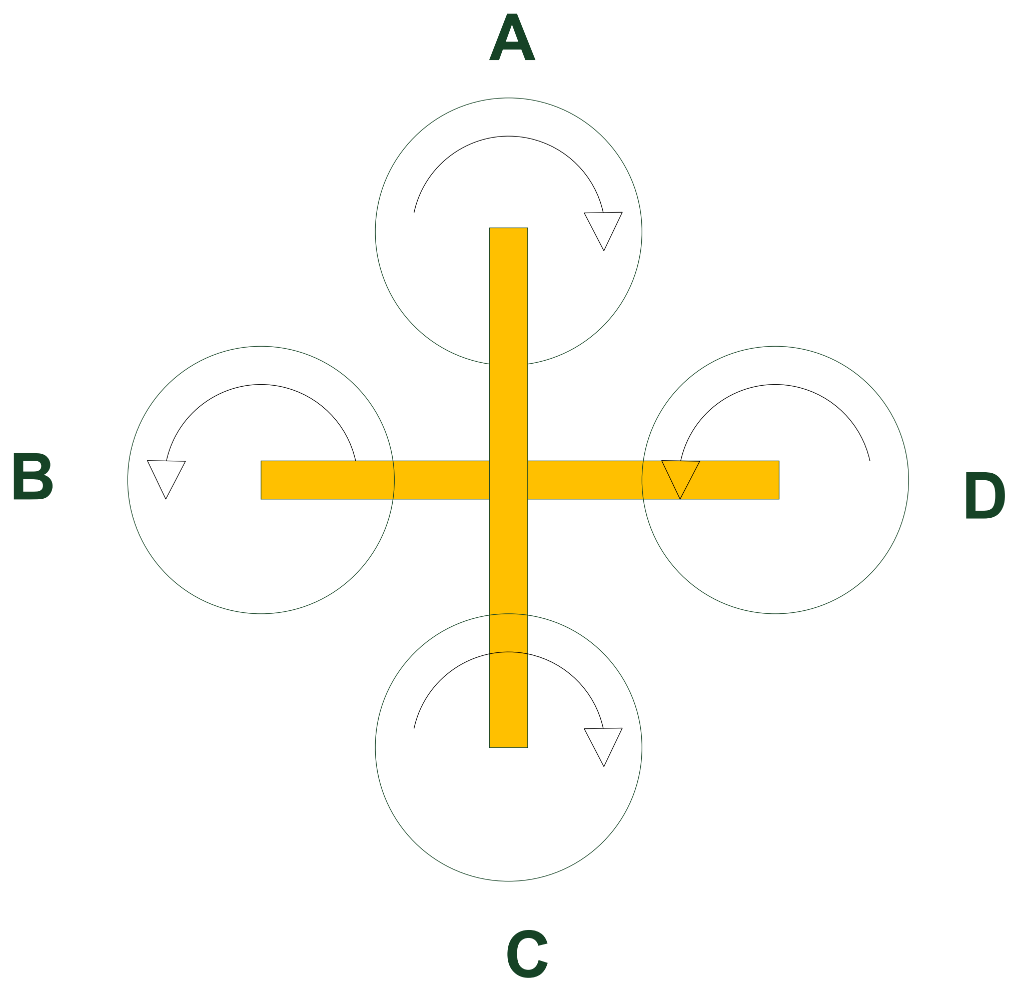
1.1 Example Real-Time Systems

Quadrocopter

A quadrocopter is a flying real-time system.

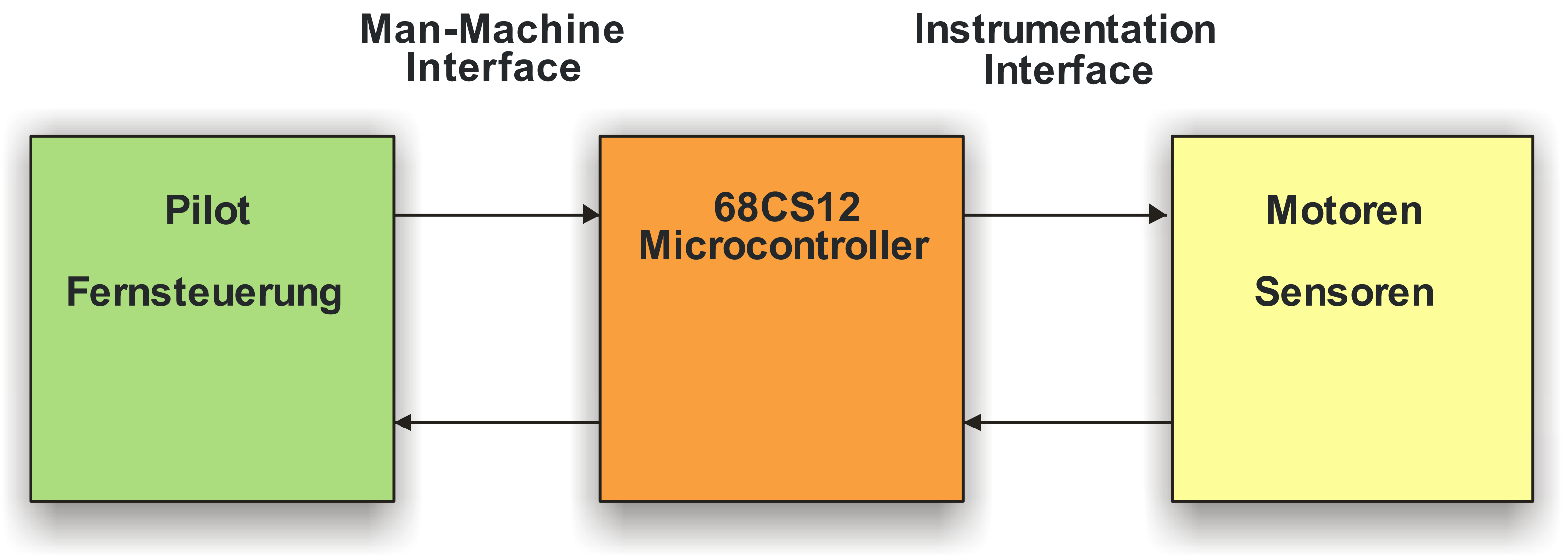
1.1 Example Real-Time Systems

Four propellers define force vector which controls vessel movement.



1.1 Example Real-Time Systems

The quadrocopter real-time system consists of a number of components:

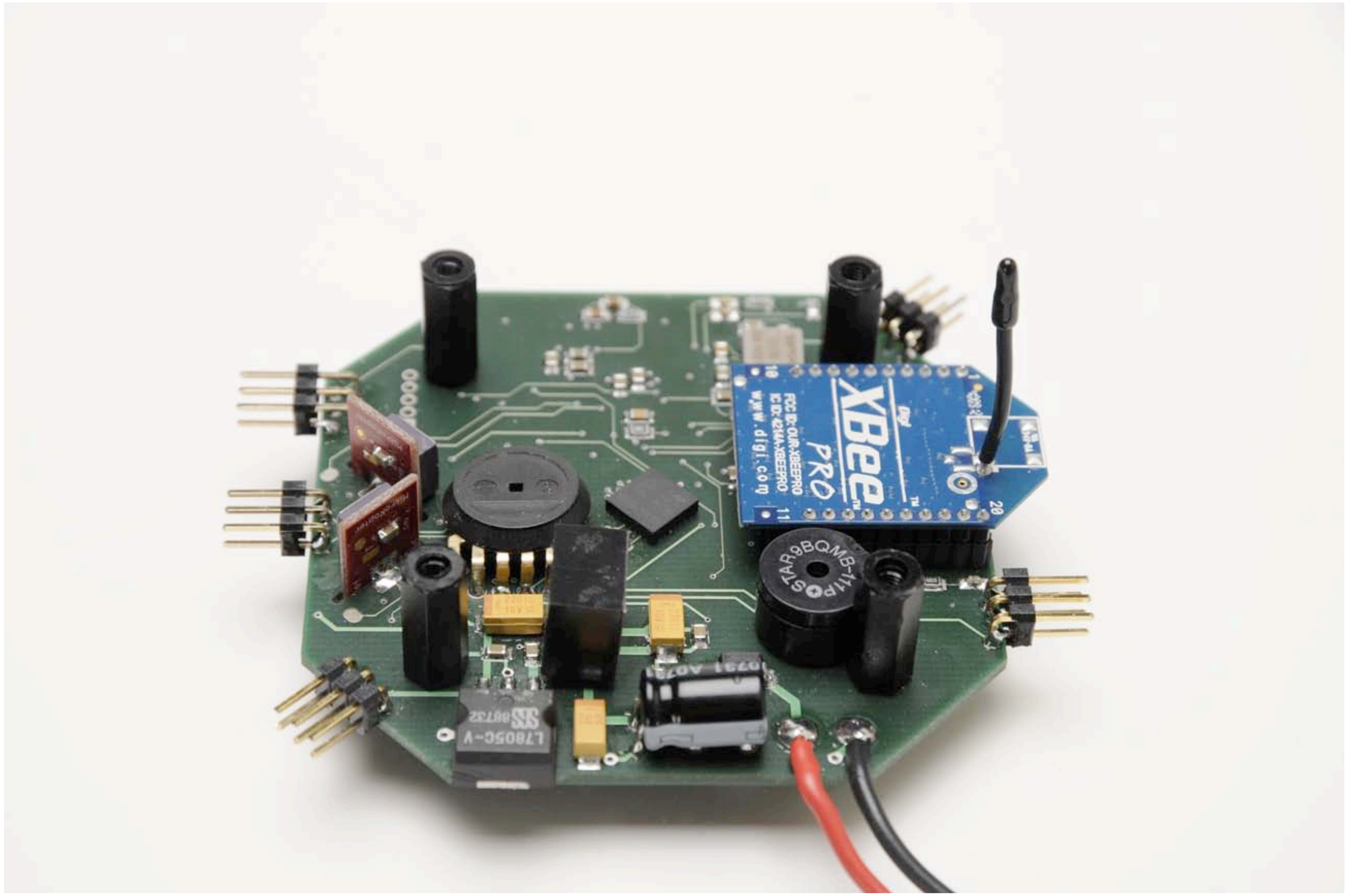


There are a number of sensors:

- Gyroscopes for three axis
- Acceleration sensors for three axis
- Pressure sensor for barometric height measurement
- Power supply voltage measurement for battery status
- Optionally more (GPS, ultrasonic distance, laser distance, video)

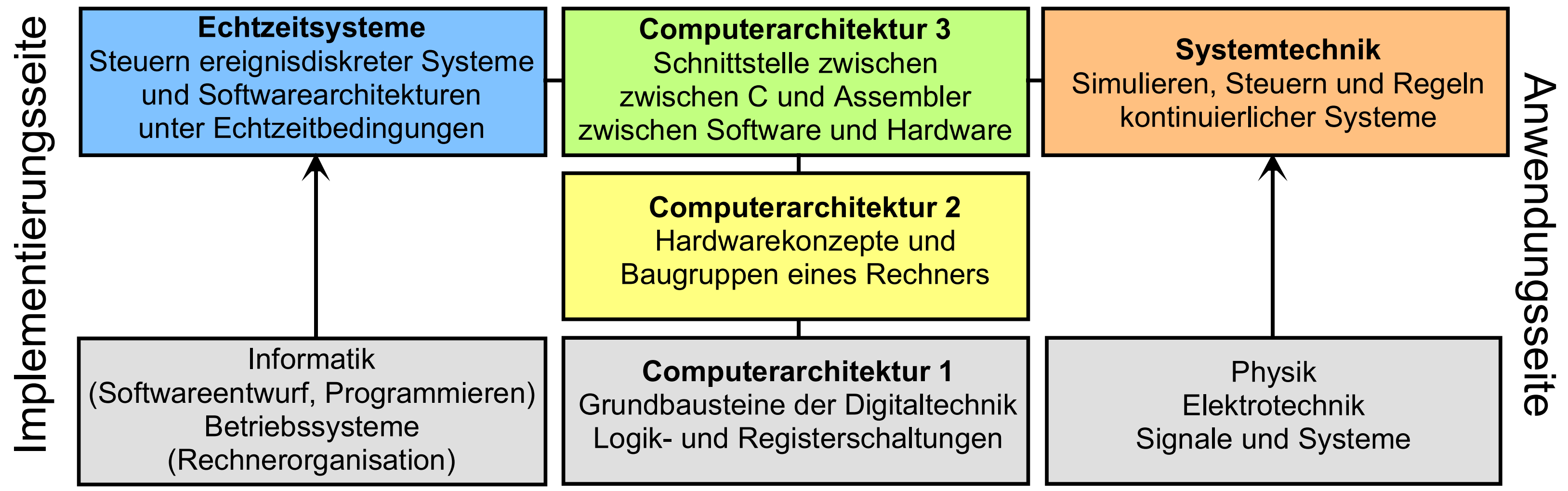
1.1 Example Real-Time Systems

The quadrocopter microcontroller looks like this:



1.1 Example Real-Time Systems

We need a broad know-how to understand such a system:


Implementierungsseite
Echtzeitsysteme
Steuern ereignisdiskreter Systeme und Softwarearchitekturen unter Echtzeitbedingungen
Computerarchitektur 3
Schnittstelle zwischen zwischen C und Assembler zwischen Software und Hardware
Systemtechnik
Simulieren, Steuern und Regeln kontinuierlicher Systeme
Computerarchitektur 2
Hardwarekonzepte und Baugruppen eines Rechners
Informatik
(Softwareentwurf, Programmieren)
Betriebssysteme
(Rechnerorganisation)
Computerarchitektur 1
Grundbausteine der Digitaltechnik
Logik- und Registerschaltungen
Physik
Elektrotechnik
Signale und Systeme
Anwendungsseite

- Electronics: to develop the microntroller board and power controllers
- Operating systems: we have developed a real-time operating system
- Software architecture: we develop software in layers and simulate most o fit on a work station, without the real hardware
- Simulation and control engineering: a quadrocopter can only fly because a real-time computer system precisely controls the four engines and measures accelerations

1.1 Example Real-Time Systems

Engine control

Consider combustion engine with injection valve. The start point of fuel injection must be precise within 0.1° of the measured angular crankshaft position.

Crankshaft turns with 6000 rpm; that means 10 ms for a 360° rotation. Precision of 0.1° transforms into a temporal accuracy of 3 μs.

The fuel injection is realized by opening a piezo-electric valve that controls the fuel flow from a high-pressure reservoir into the cylinder. The latency between giving the valve the “open” command and the actual point in time when the valve opens is in the order of hundreds of μs, and changes considerably due to environmental conditions (e.g. temperature).

A sensor signal indicates the point in time when the valve has actually opened to provide the ability to compensate for the latency jitter. The duration between the execution of the output command and the start of opening the valve is measured in each engine cycle, to compensate for the latency jitter in the next cycle.

Airbag System

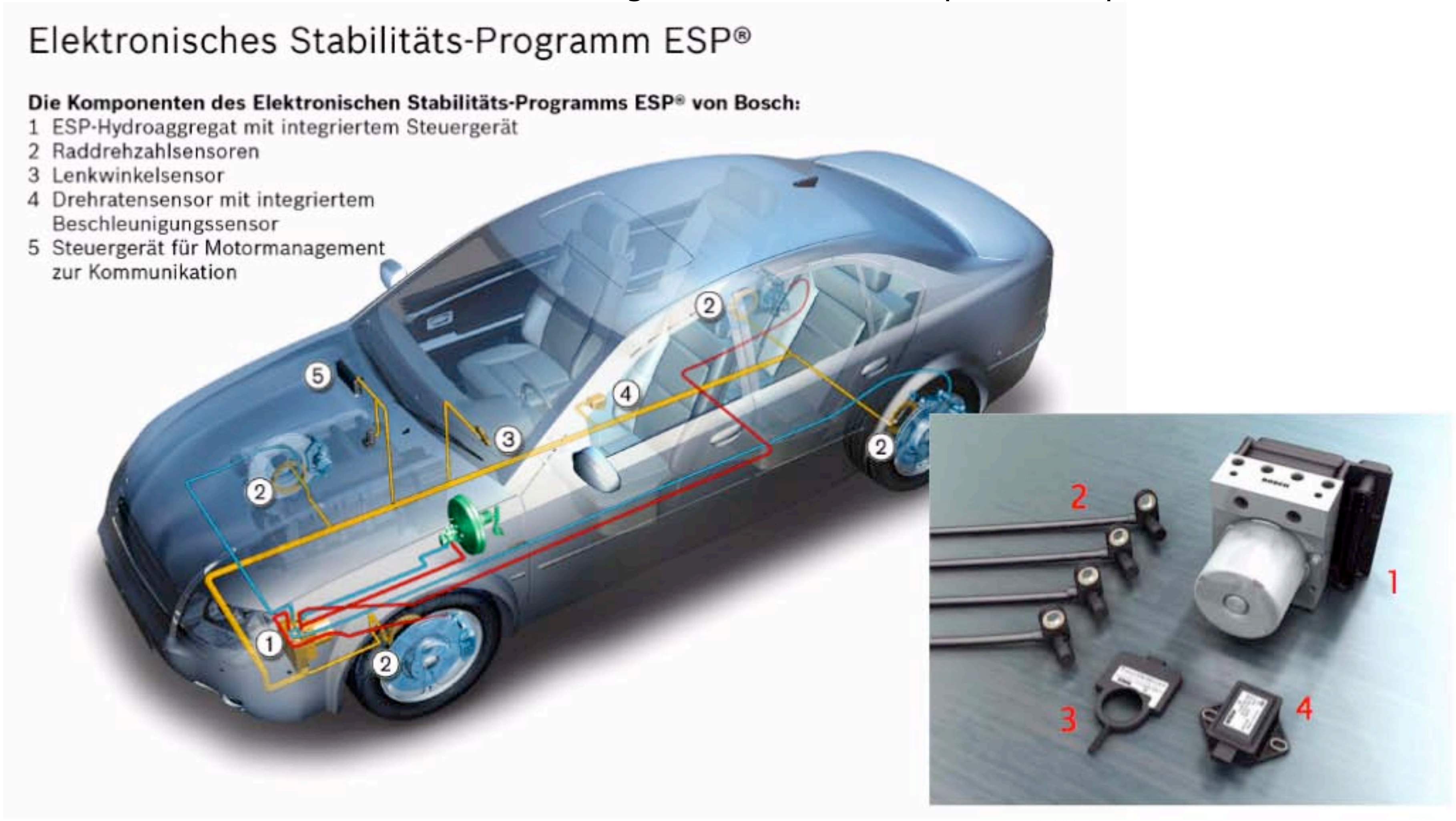
An air bag system serves to protect car passengers colliding with the dashboard or steering wheel in case of a crash. At least one sensor detects a crash condition, and an ECU ignites some solid propellant that rapidly generates inert gas. The gas inflates the airbag.

1.1 Example Real-Time Systems

Typically, the decision to deploy an airbag in a frontal crash is made within 15 to 30 milliseconds after the onset of the crash, and both the driver and passenger airbags are fully inflated within approximately 60-80 milliseconds after the first moment of vehicle contact.

Electronic Stability Program (ESP)

Prevents car from uncontrolled skidding. Performs 25 computations per second.



Elektronisches Stabilitäts-Programm ESP®

Die Komponenten des Elektronischen Stabilitäts-Programms ESP® von Bosch:

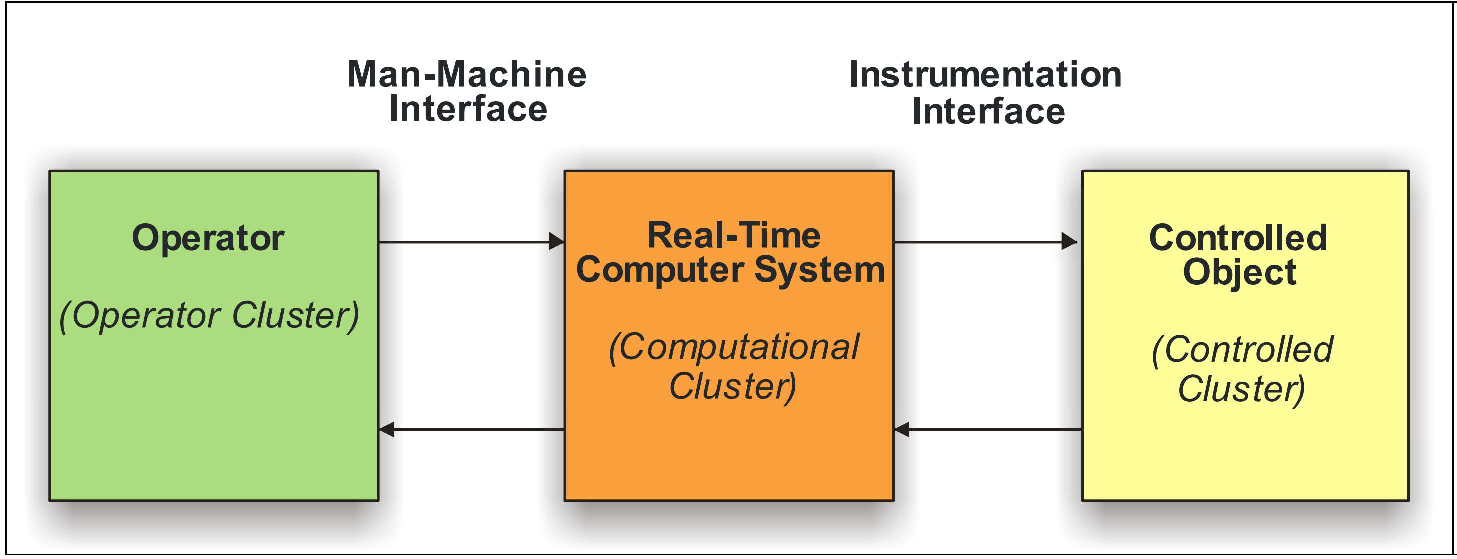
1 ESP-Hydroaggregat mit integriertem Steuergerät
2 Raddrehzahlsensoren
3 Lenkwinkelsensor
4 Drehratensensor mit integriertem Beschleunigungssensor
5 Steuergerät für Motormanagement zur Kommunikation

1.2 When is a Computer System Real-Time?

A real-time computer system is a computer system in which the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical instant at which these results are produced.

A real-time computer system is always part of a larger system, the real-time system.

We decompose a real-time system into a set of subsystems called clusters.



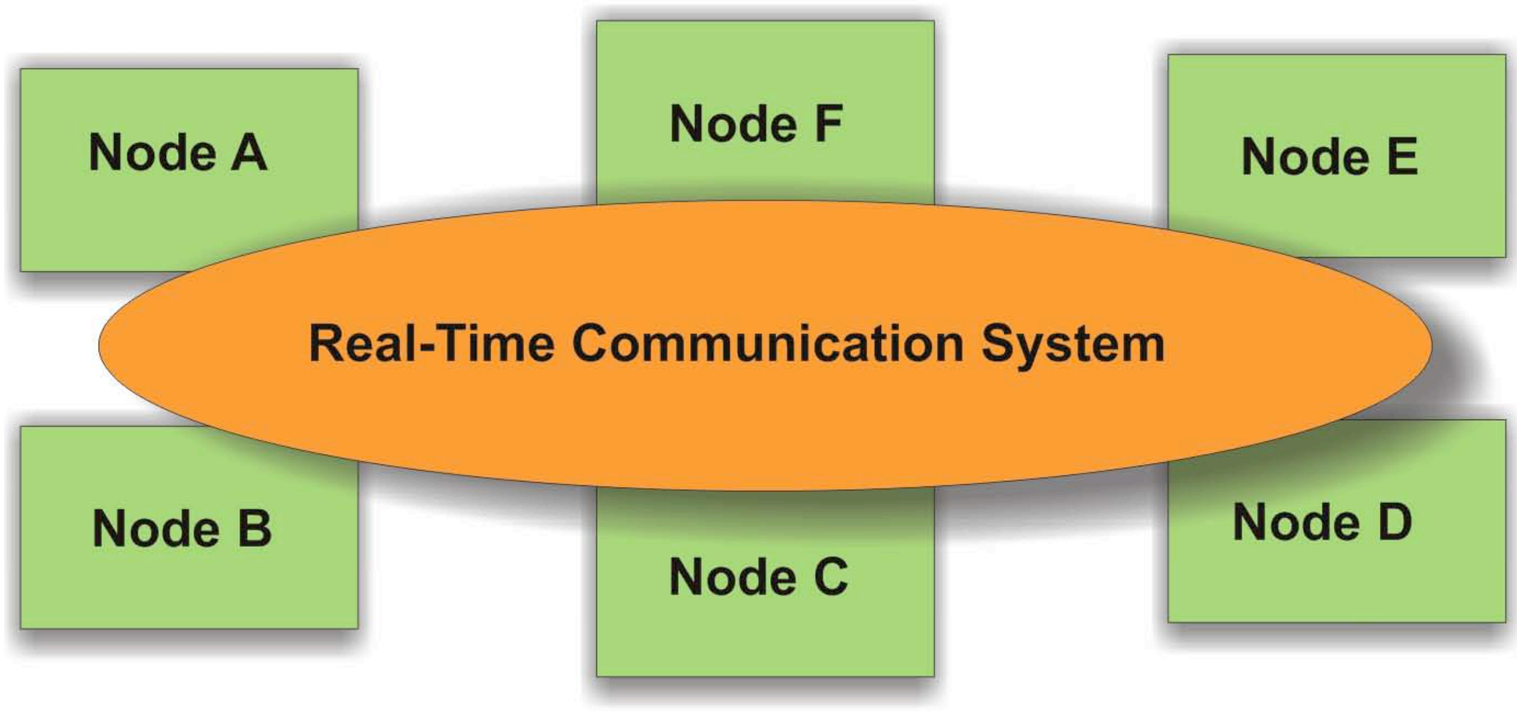
Man-Machine-Interface: keyboard, mouse, screen

Instrumentation-Interface: sensors and actuators, transforming physical signals into a digital form and vice versa

The controlled object and the operator form the environment of the real-time computer system. Not every node has both, a man-machine interface and an instrumentation interface.

1.2 When is a Computer System Real-Time?

A distributed real-time computer system consists of a set of computer nodes interconnected by a real-time communication network.



A real-time computer system must react to stimuli from the controlled object or operator within time intervals dictated by the environment.

The instant at which the result must have been produced is called a deadline.

1.2 When is a Computer System Real-Time?

Soft deadline: result has some utility passed this instant in time.

Firm deadline: result has no or very little utility past this instant in time.

Hard deadline: catastrophic failure when result is produced past this instant in time.

Hard real-time computer system or safety-critical real-time computer system:

- A real-time system that must meet at least one hard deadline.

Soft real-time computer system:

- A real-time computer system for which no hard deadline exists.

1.3 Functional Requirements

Functional Requirements are grouped into

- Data collection requirements
- Direct digital control requirements
- Man-machine interaction requirements

Data Collection

A controlled object (car, industrial plant) changes its state as a function of time.

Current state: values of all state variables at this moment

Example: speed of car, position of switches on dashboard, position of piston in a cylinder

We are normally only interested in a subset of state variables significant for our purpose:

- A significant state variable is called a real-time (RT) entity.

Every RT entity is in the sphere of control (SOC) of a subsystem, i.e., it belongs to a subsystem that can change the value of that RT entity.

Outside that SOC the value of that entity can only be observed, not modified.

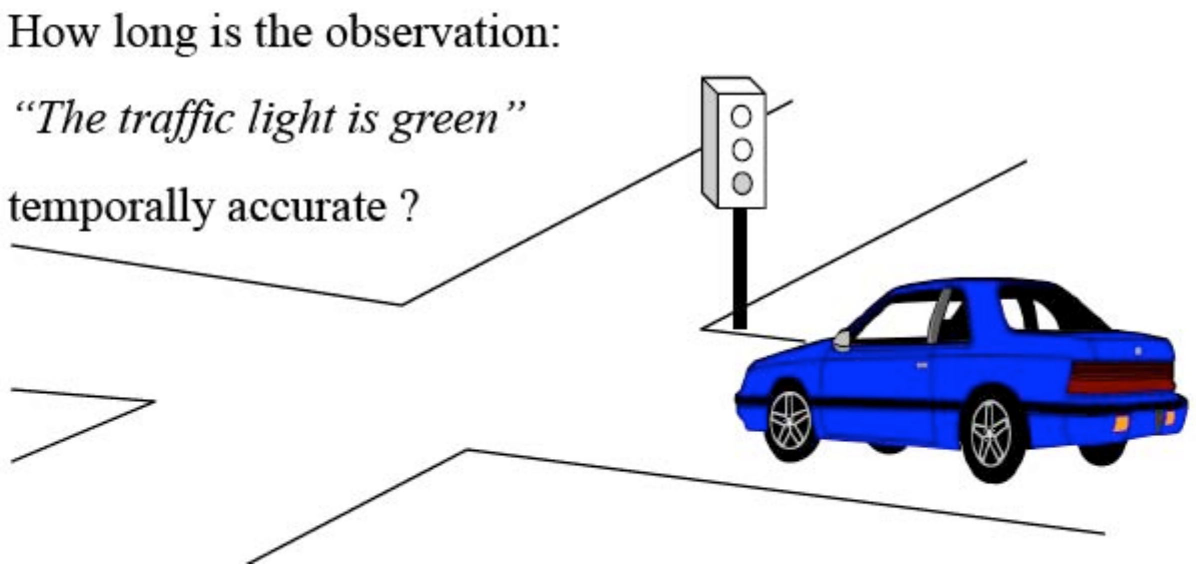
1.3 Functional Requirements

Thus, we have the following functional requirements:

- Observation of the RT entities in a controlled object
- Collection of these observations

An observation of an RT entity is represented by a real-time (RT) image.

Example for accuracy interval:



If the information „traffic light is green" is used outside its accuracy interval, a collision may occur.

1.3 Functional Requirements

The set of all temporally accurate RT images of the controlled object is called the real-time database.

The RT database must be updated whenever an RT entity changes its value.

The update can be performed periodically with a fixed period (time-triggered (TT) observation).

The update can be performed immediately after a state change in the RT entity, which constitutes an event (event-triggered (ET) observation).

Signal conditioning

Signal conditioning encompasses all processing steps to obtain meaningful measured data from physical sensor data (raw data element, e.g. a voltage). Typical processing steps:

- filtering and averaging
- plausibility checks
- calibration and transformation

Agreed data element: a data element judged to be a correct RT image of the corresponding RT entity.

1.3 Functional Requirements

Alarm monitoring

Alarm monitoring serves to detect abnormal behaviour by continuous observation of RT entities.

Example: rupture of a pipe in a chemical plant will cause many RT entities (pressures, temperatures, liquid levels) deviate from their normal operating ranges.

The rupture will lead to a set of correlated alarms, which is called an alarm shower.

The computer system must detect and display these alarms, and must try to identify the primary event (the initial cause of the alarm).

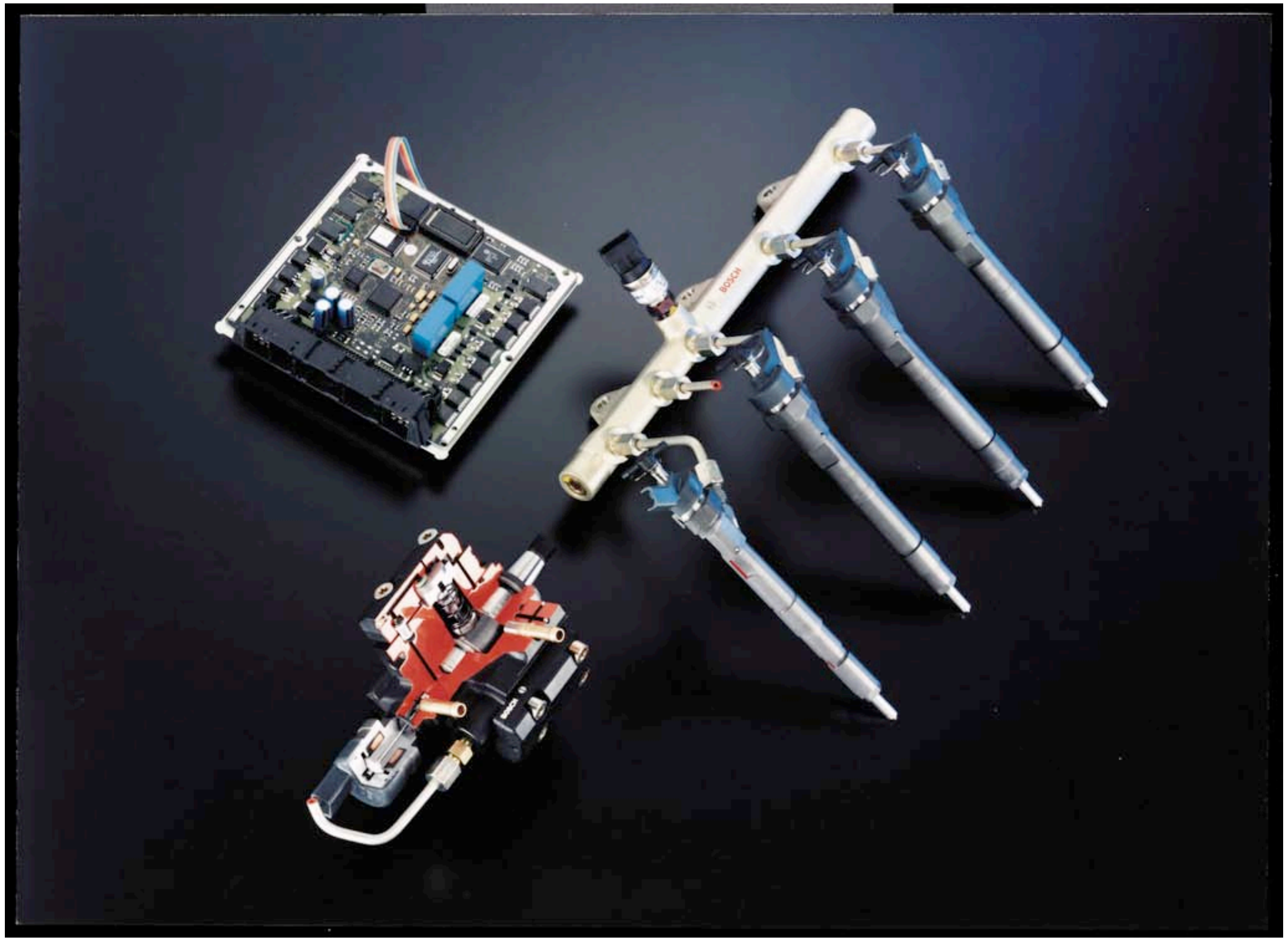
Alarms are logged with the exact time at which they occur. The time stamp helps to eliminate secondary alarms.

A situation that occurs infrequently but is of great concern when it does is called a rare-event situation. Rare-event performance validation is a great challenge (e.g. nuclear plants).

1.3 Functional Requirements

Direct Digital Control

Many real-time computer systems must calculate set points for the actuators and control the controlled object directly. Example: engine controller



1.3 Functional Requirements

Control applications are highly regular, consisting of an infinite sequence of control periods:

- Sampling of the RT entity
- Execution of the control algorithm to calculate a new set point
- Output of set point to actuator

Design of control algorithms is topic of field of control engineering.

1.3 Functional Requirements

Man-Machine Interaction

A real-time computer system must inform the operator of the current state of the controlled object, and must assist the operator in controlling the machine or plant object.

The man-machine interface in safety-critical real-time systems is of utmost importance. A bad design can result in catastrophic failures.

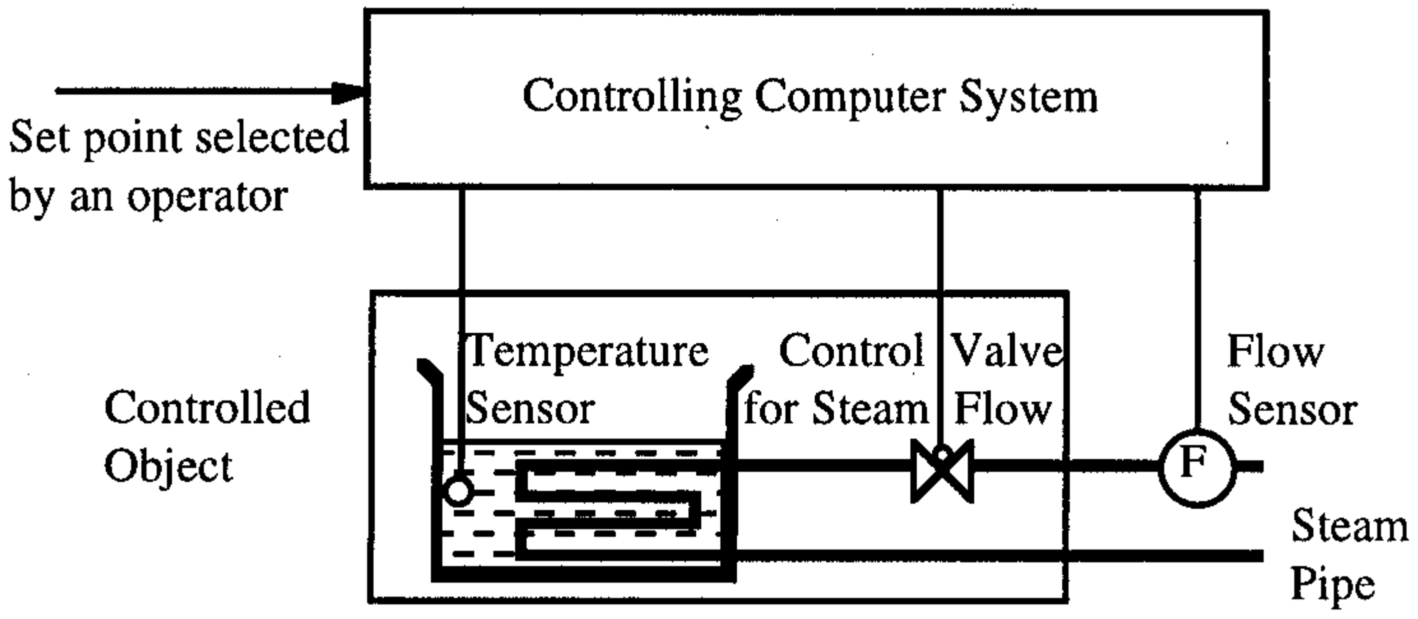
Automotive examples:

- Dashboard
- Joystick
- Steering wheel
- Brake, accelerator pedal

This topic justifies a lecture on its own, and is not dealt with any further here.

1.4 Temporal Requirements

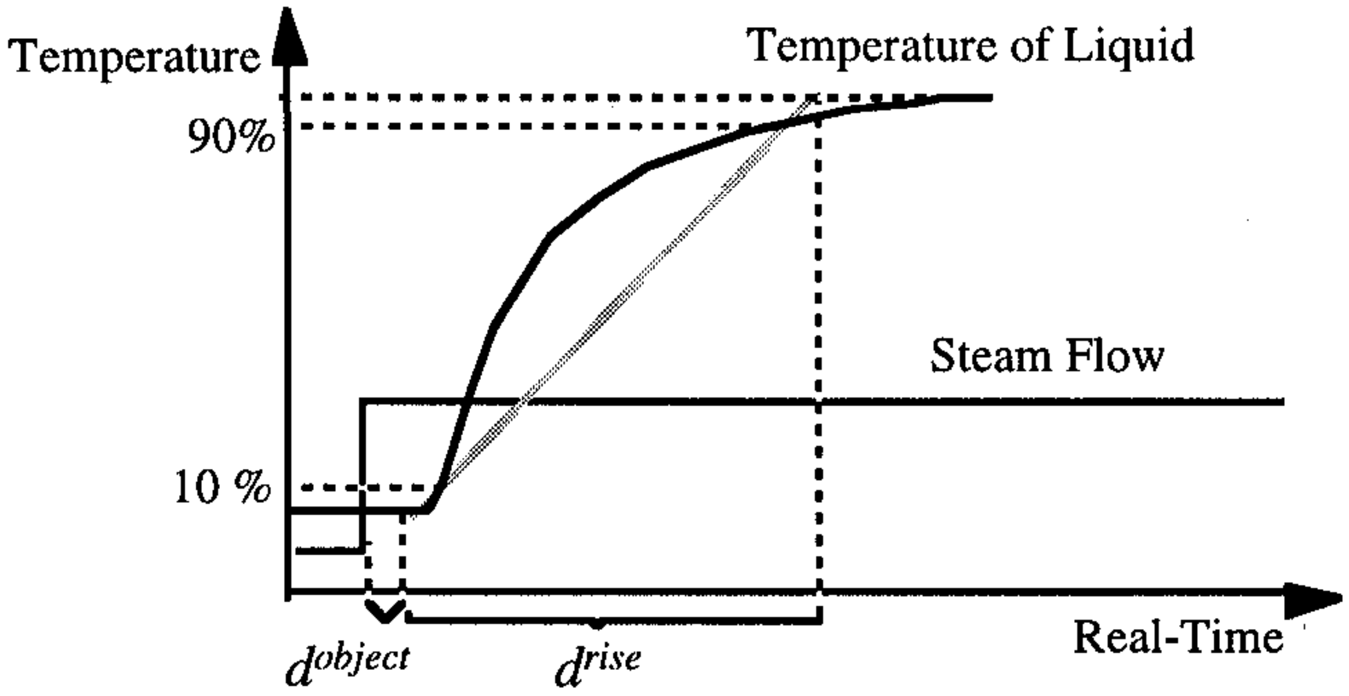
- Most stringent temporal demands come from control loops, e.g. control of fast mechanical processes such as automotive engines
- Requirements at the MMI are less stringent (human perception delay is between 50-100 ms)
- Example for a simple control loop: control steam flow through pipe such that liquid temperature remains at set point.


Controlling Computer System
Set point selected by an operator
Controlled Object
Temperature Sensor
Control Valve for Steam Flow
Flow Sensor
F
Steam Pipe

1.4 Temporal Requirements

Behavior of liquid temperate when steam flow is increased instantly by a fixed amount: response function of the temperature.

Response function characterized by two temporal parameters: object delay d^{object} or process lag and rise time d^{rise} of the temperature



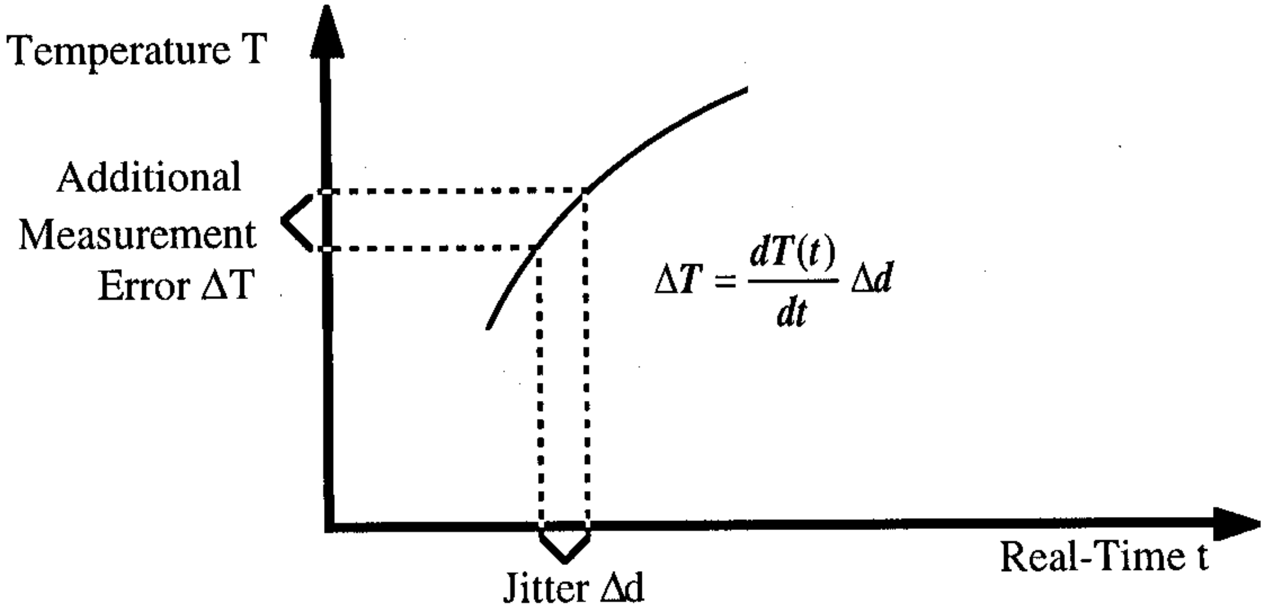
Controlling computer system samples temperature of vessel periodically with sampling period d^{sample}. The sampling frequency is $f^{sample} = 1/ d^{sample}$.

1.4 Temporal Requirements

Rule of thumb: sampling period should be less than one-tenth of the rise time d^{rise} of controlled object to obtain quasi-continuous behavior ($d^{sample} < d^{rise}/10$).

The computer system calculates the new value for the control variable, and outputs it to the control valve after some delay, called the computer delay $d^{computer}$. The computer delay should be less than the sampling period d^{sample}.

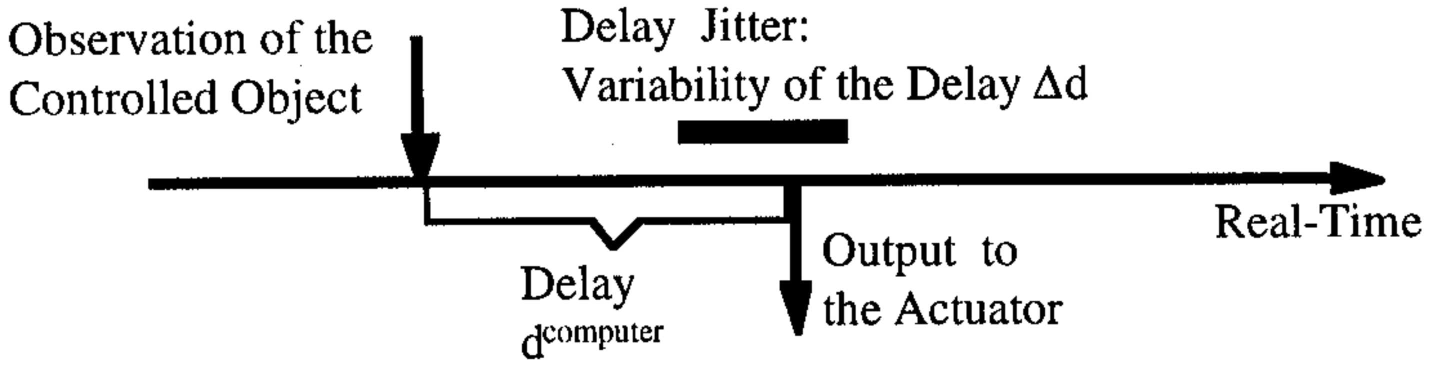
The jitter $\Delta d^{computer}$ is the difference between the maximum and minimum computer delay. It should be small compared to the computer delay $d^{computer}$ (μs to ms). The jitter can be interpreted as causing an additional error of the temperature ΔT.


Temperature T
Additional Measurement Error ΔT
$\Delta T = \frac{dT(t)}{dt} \Delta d$
Jitter Δd
Real-Time t

1.4 Temporal Requirements

The dead time $d^{deadtme}$ is the time interval between the observation of the RT entity and the start of a reaction of the controlled object due to a computer action.
The dead time is $d^{object} + d^{computer}$.

Dead time reduces stability of the control loop, and should thus be as small as possible.


Observation of the Controlled Object
Delay Jitter: Variability of the Delay Δd
Real-Time
Output to the Actuator
Delay $d^{computer}$

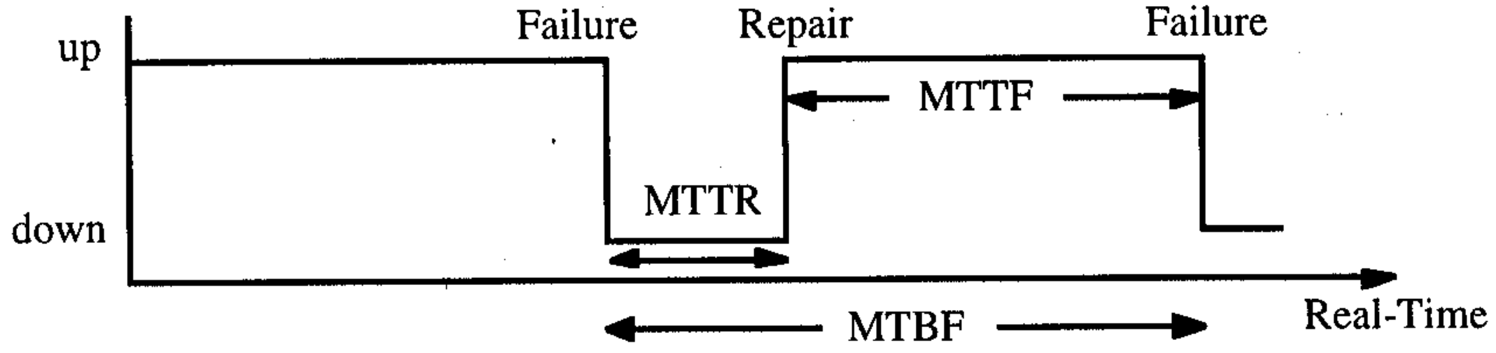
Hard real-time applications are, by definition, safety-critical. Errors must therefore be detect-ed within a short time with high probability. The required error detection latency must be in the order of magnitude of the sampling period of the fastest control loop.

1.5 Dependability Requirements

Verlässlichkeit

Dependability is determined by:

- Reliability
- Safety
- Maintainability
- Availability
- Security



Reliability

Reliability R(t) is the probability that a system will provide the specified service until time *t*, given that the system was operational at time t_0.

If a system has a constant failure rate λ *failures/hour*, then *R(t)* is given by

$$R(t) = exp(-\lambda\,(t-t_0)) \qquad \text{(with t and } t_0 \text{ given in hours)}$$

1.5 Dependability Requirements

The inverse of the failure rate $1/\lambda = MTTF$ is called the Mean-Time-To-Failure (in hours).

Safety

Safety is reliability regarding critical failure modes (malign failure modes). Examples of malign failures:

- airplane crash due to a failure in the flight-control system
- automobile accident due to a failure in the computer controlled braking system

Hard real-time systems must have a failure rate conforming to ultrahigh reliability requirements (*MTTF* 10^{-9} *failures/hour* or less). Example: one such failure per one million cars per year, when car is operated one hour per day.

Maintainability

Maintainability is a measure of the time required to repair a system after the occurrence of a benign failure (non safety-critical failure). It is measured by the probability M(d) that the system is restored within a time interval *d* after the failure.

Similar to reliability, there is a repair rate μ ; the inverse of the repair rate $1/\mu = MTTR$ is called the Mean-Time-To-Repair (in hours).
Reliability and maintainability are in conflict to each other (e.g. soldered connection vs. plug). Many replaceable units improve maintainability and reduce reliability. In automotive market typically OEMs decide for reliability at the cost of maintainability.

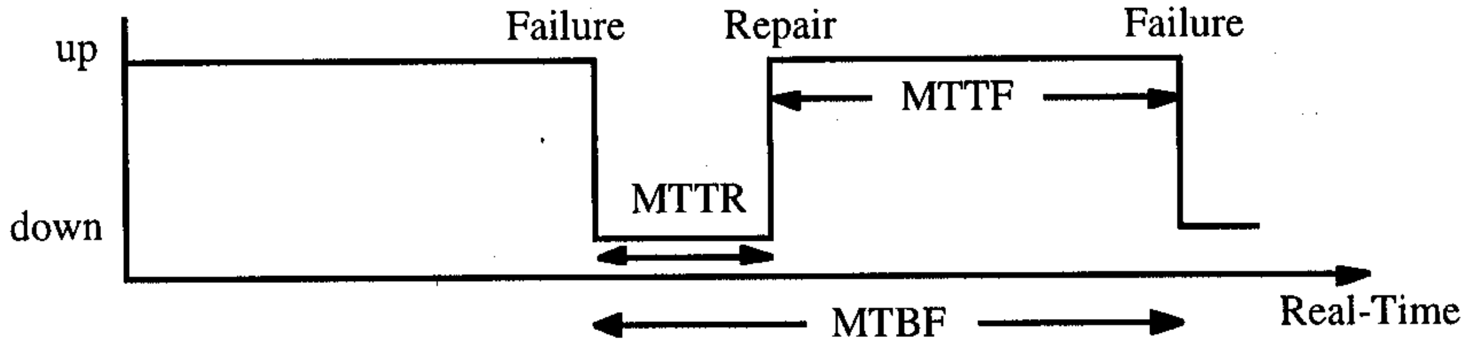
Availability

Availability is a measure of the delivery of correct service with respect to the alternation of correct and incorrect service. It is measured by the fraction of time the system is ready to provide the service.

In systems with constant failure and repair rates, the reliability (*MTTF*), maintainability (*MMTR*) and availability (*A*) measures are related by

$$A = MTTF\,/\,(MTTF+MTTR)$$

The sum of *MTTF* and *MTTR* is called the Mean Time Between Failures (*MTBF*).



A high availability can be achieved by high *MTTF* or by a short *MTTR*.

1.5 Dependability Requirements

Security

Security is the ability of a system to prevent unauthorized access to information or services.

Examples in the automotive context are:

- Ignition lock for theft avoidance
- Car access with lock system
- Protection against software reverse engineering and manipulation

There is no standardized quantitative measure for security.

1.6 Classification of Real-Time Systems

Hard Real-Time versus Soft Real-Time System

Characteristic	Hard Real-Time	Soft Real-Time (on-line)
Response time	Hard - required	Soft - desired
Peak-load performance	Predictable	Degraded
Control of pace	Environment	Computer
Safety	Often critical	Non-critical
Size of data files	Small/medium	Large
Redundancy type	Active	Checkpoint-recovery
Data integrity	Short-term	Long-term
Error detection	Autonomous	User assisted

Fail-Safe versus Fail-Operational

Fail safe: controlled object can enter a safe state where nothing bad can happen (e.g. all traffic lights switched to red, all trains stopped). Fail-safeness is property of controlled object, not of computer system.

Fail-operational: system must be still operational in case of failure (airplane, no safe state).

1.6 Classification of Real-Time Systems

Guaranteed-Response versus Best-Effort

We assume some peak load and a fault model. If we can by design guarantee adequate responses under all conditions, without reverting to probability arguments, we can speak of a system with a guaranteed response.

In case we cannot guarantee adequate response by design, we speak of a best effort system. It is very difficult to demonstrate that a best-effort design operates correctly under rare event scenarios.

Resource-Adequate versus Resource-Inadequate

Guaranteed response systems must have enough resources to handle a specified peak load and fault scenario. Due to economic reasons many non safety-critical real-time systems are based on the principle of resource inadequacy.

Event-Triggered versus Time-Triggered

In the event-triggered approach, all communication and processing activities are initiated whenever a significant change of state is noted. Notification to the computer system via interrupts. Requires dynamic scheduling strategy.

In the time triggered approach, all communication and processing activities are initiated at predetermined points in time. Only one interrupt, the periodic clock interrupt. In a distributed system there needs to be some notion of a synchronized global time.

1.6 Classification of Real-Time Systems

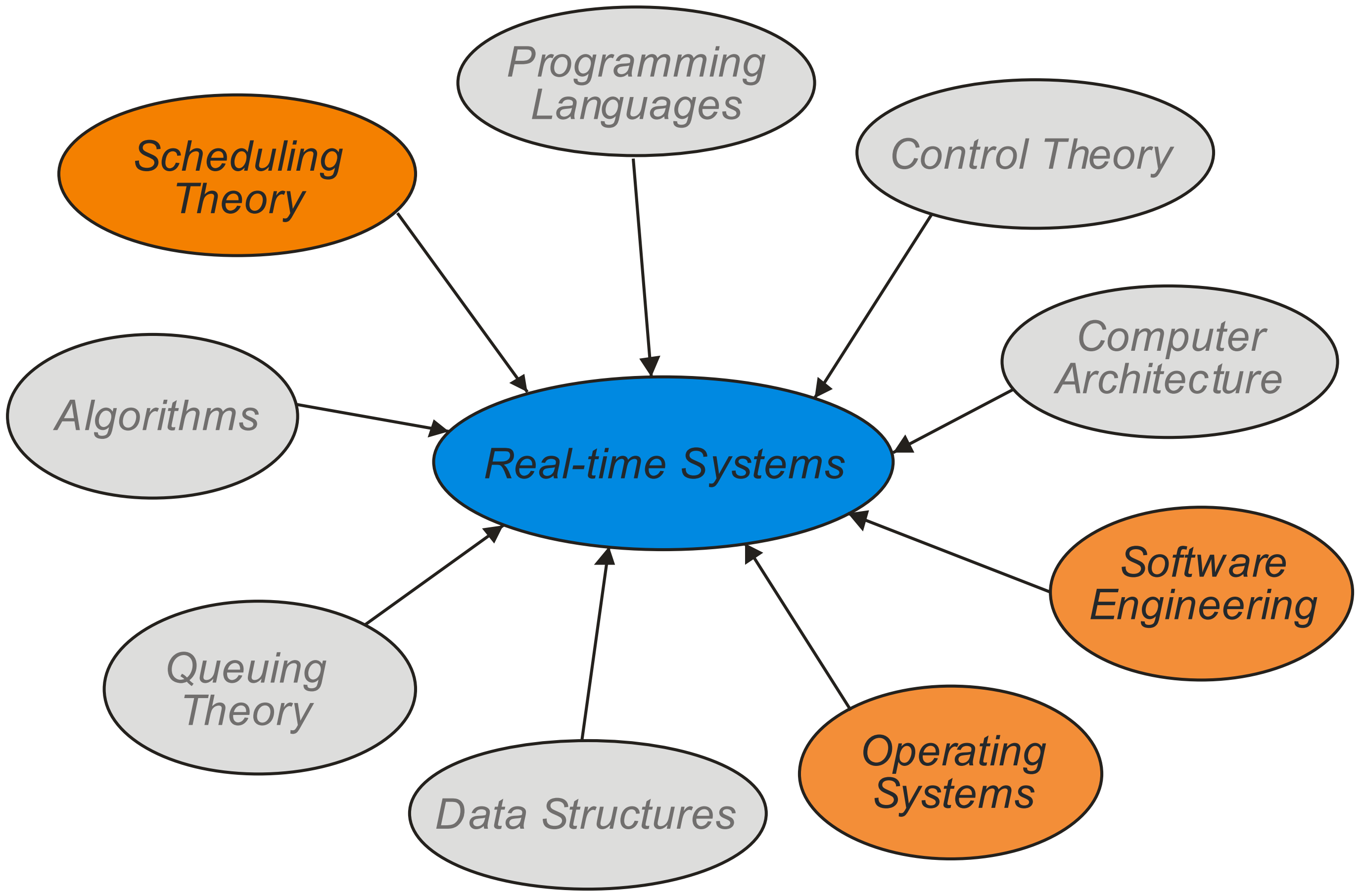
Product Automation versus Process Automation

In product automation, the real-time computer system is part of a product that is typically sold in large quantities. In product automation, the real-time computer system contributes significantly to total system cost. Development effort is spread over a large number of units. Examples: cameras, engine control.

In process automation, the number of units sold is rather small. The real-time computer system price typically is not significant compared to the cost of the rest of the system. Development effort directly contributes to system cost. Examples: chemical processing plant.

1.7 Real-Time System Design Issues

In this course, we will focus on the orange areas.



Points to Remember