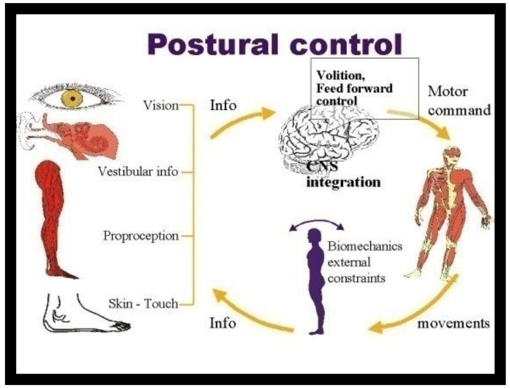
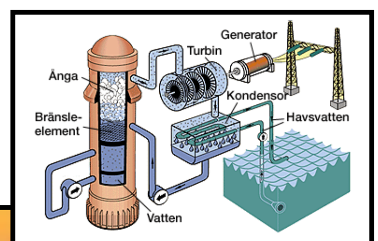
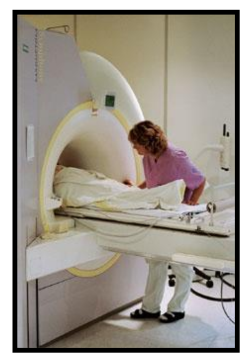
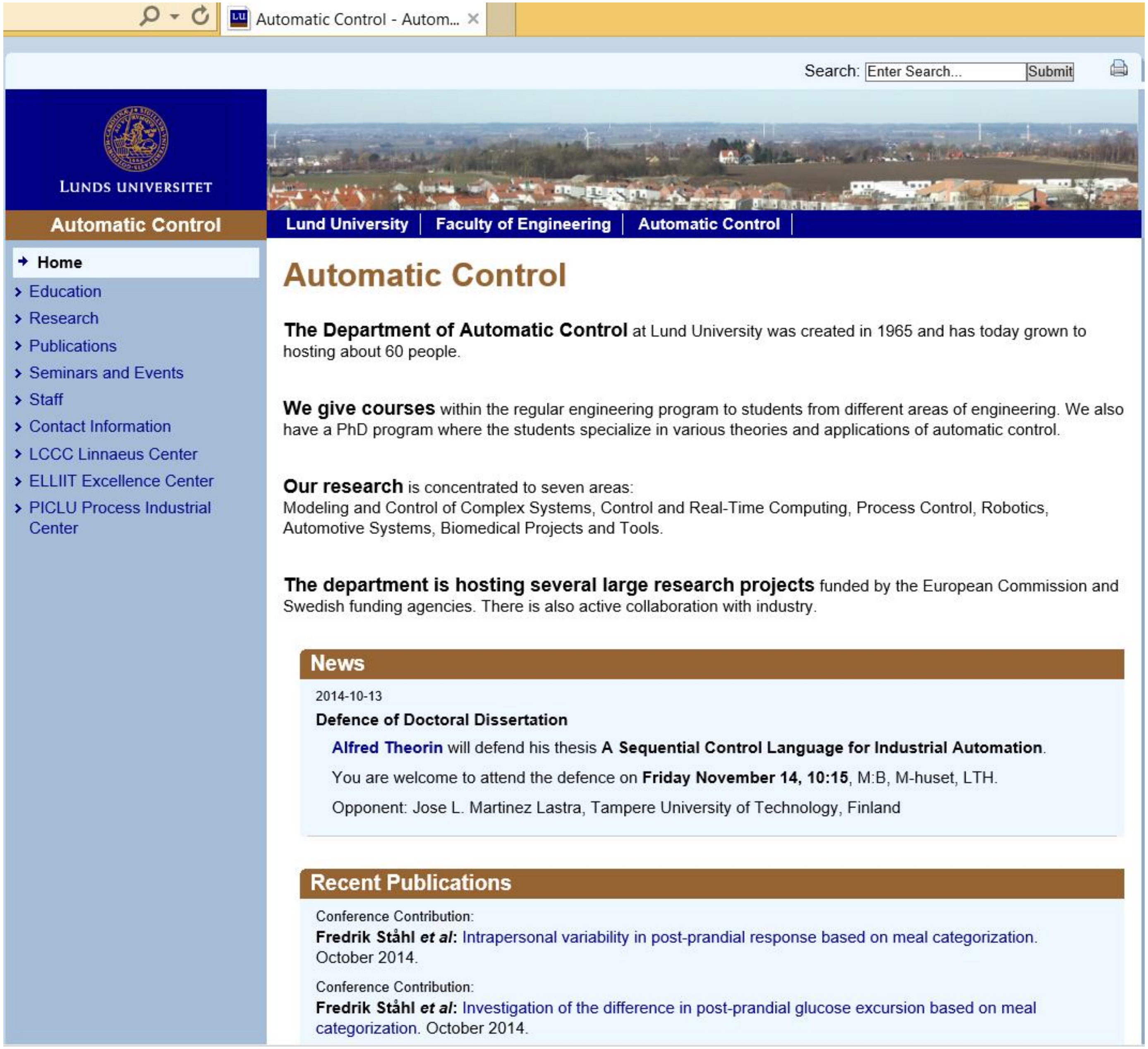


Vad är Reglerteknik?
Gästföreläsning i Datorer i System

Anders Robertsson

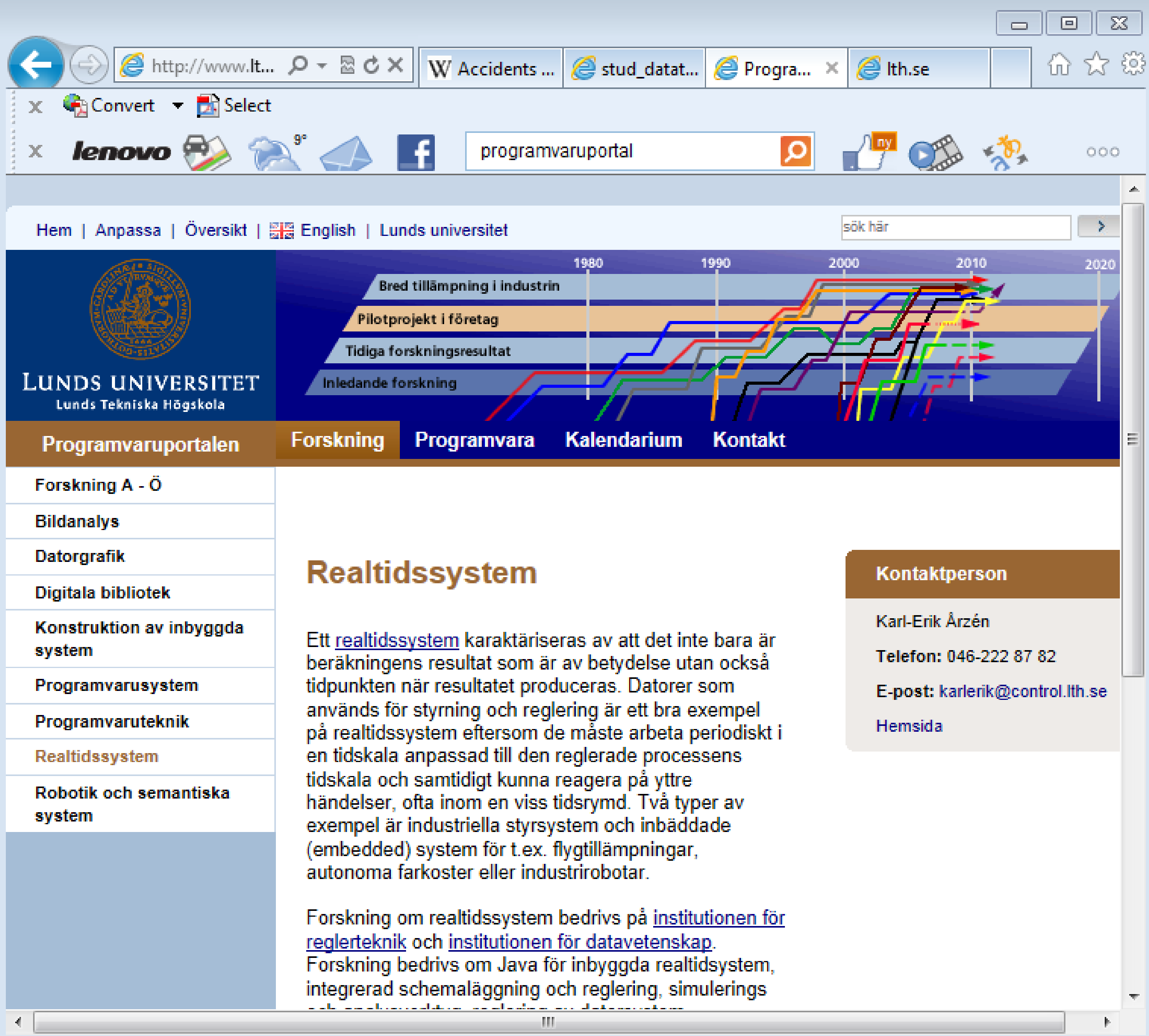






Material and slides from colleagues and courses at

http://www.control.lth.se



Hem | Anpassa | Översikt | English | Lunds universitet

LUNDS UNIVERSITET
Lunds Tekniska Högskola

Programvaruportalen | Forskning | Programvara | Kalendarium | Kontakt

- Forskning A - Ö
- Bildanalys
- Datorgrafik
- Digitala bibliotek
- Konstruktion av inbyggda system
- Programvarusystem
- Programvaruteknik
- Realtidssystem
- Robotik och semantiska system

Realtidssystem

Ett realtidssystem karaktäriseras av att det inte bara är beräkningens resultat som är av betydelse utan också tidpunkten när resultatet produceras. Datorer som används för styrning och reglering är ett bra exempel på realtidssystem eftersom de måste arbeta periodiskt i en tidskala anpassad till den reglerade processens tidskala och samtidigt kunna reagera på yttre händelser, ofta inom en viss tidsrymd. Två typer av exempel är industriella styrsystem och inbäddade (embedded) system för t.ex. flygtillämpningar, autonoma farkoster eller industrirobotar.

Forskning om realtidssystem bedrivs på institutionen för reglerteknik och institutionen för datavetenskap. Forskning bedrivs om Java för inbyggda realtidssystem, integrerad schemaläggning och reglering, simulerings

Kontaktperson

Karl-Erik Årzén

Telefon: 046-222 87 82

E-post: karlerik@control.lth.se

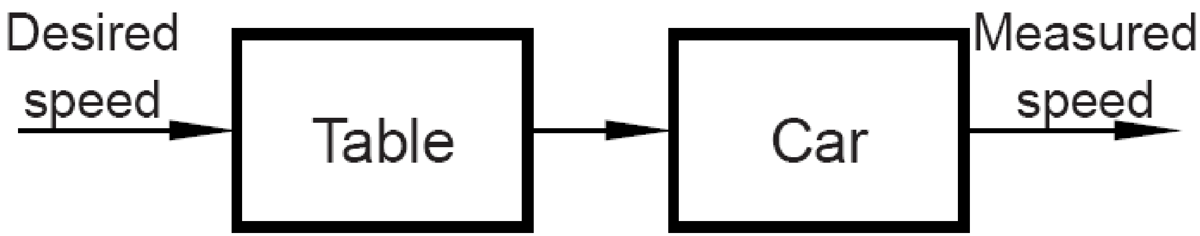
Hemsida

http://www.lth.se/programvaruportalen/

Översikt

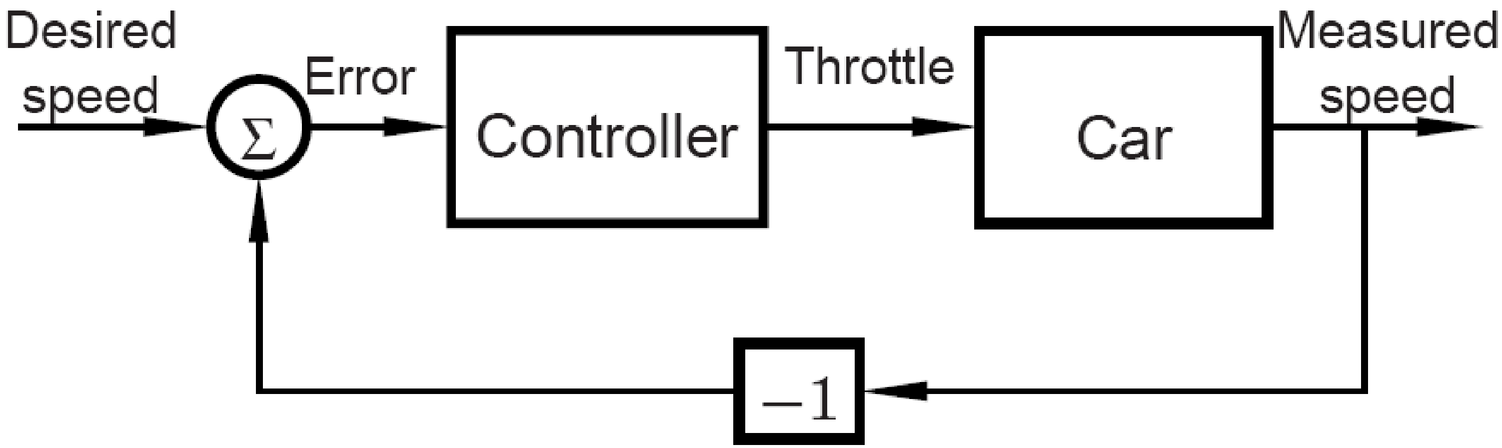
- Inledande exempel: Automatisk Farthållare
- Var finns Reglerteknik?
- Hur gör man?
- Varför Reglerteknik på D?
- Vilka kurser finns?
- Hur kopplar det till våra andra kurser?
- Exempel
  - Webserverreglering
  - ACTORS projektet (EU-projekt med resursallokering)

Farthållare: "Öppen loop"



- Open loop
- Problems?

Farthållare: Återkoppling



- Closed loop
- Simple controller:
  - Error > 0: increase throttle
  - Error < 0: decrease throttle

Återkoppling (Feedback)

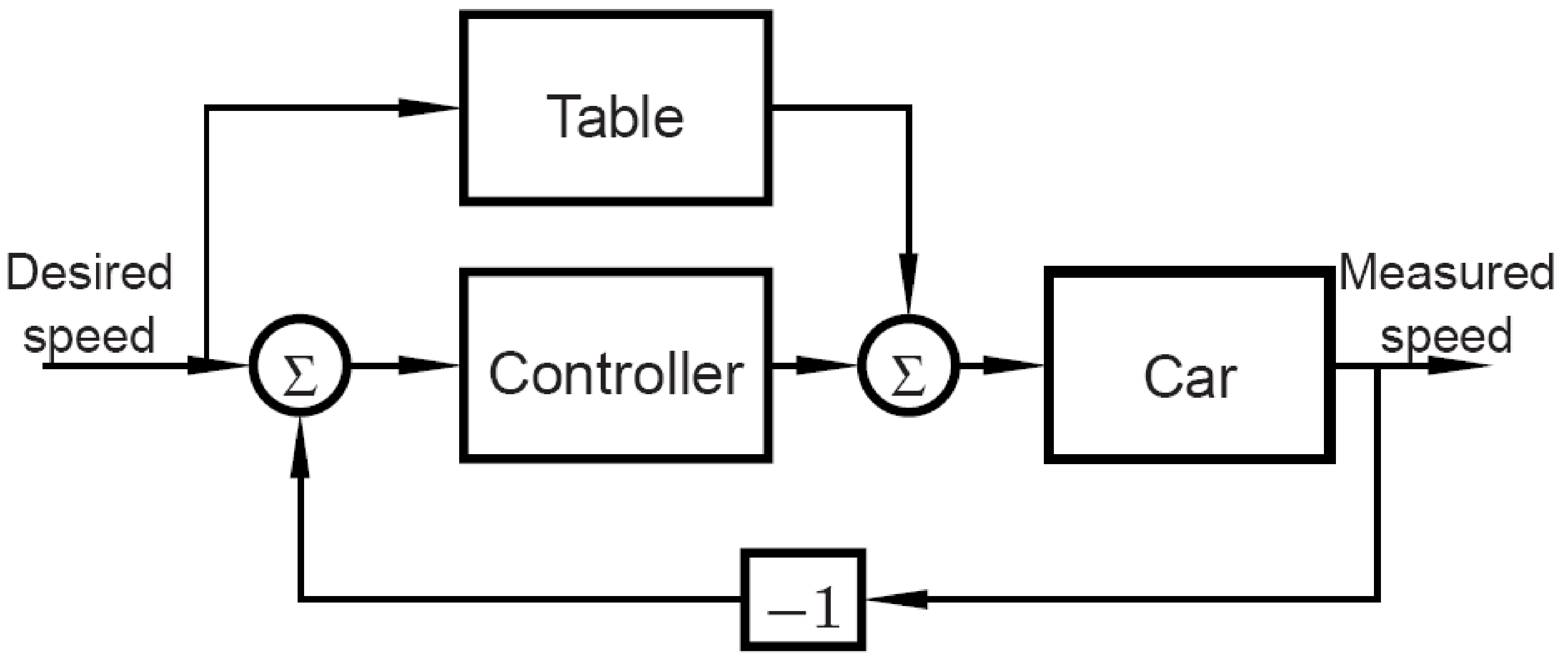
En mycket kraftfull princip

+ Minskar inverkan av störningar
+ Minskar känslighet för processvariationer
+ Kräver inte detaljerade modeller
- Kan förstärka mätbrus
- Kan ge upphov till instabilitet

Framkoppling (öppen loop)

+ Minskar inverkan av mätbara störningar
+ Medger snabba refersensvärdesändringar utan att introducera något reglerfel
- Kräver goda modeller
- Kräver stabilt system

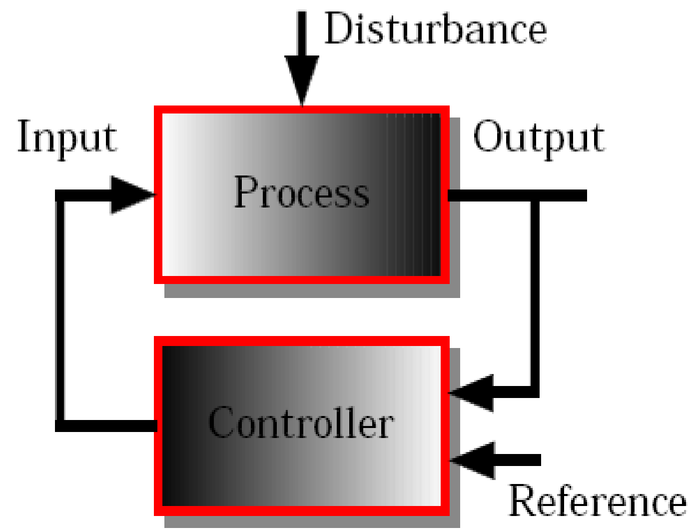
Framkoppling + Återkoppling



Use of **models** and **feedback**

Activities:

- Modeling
- Analysis and simulation
- Control design
- Implementation



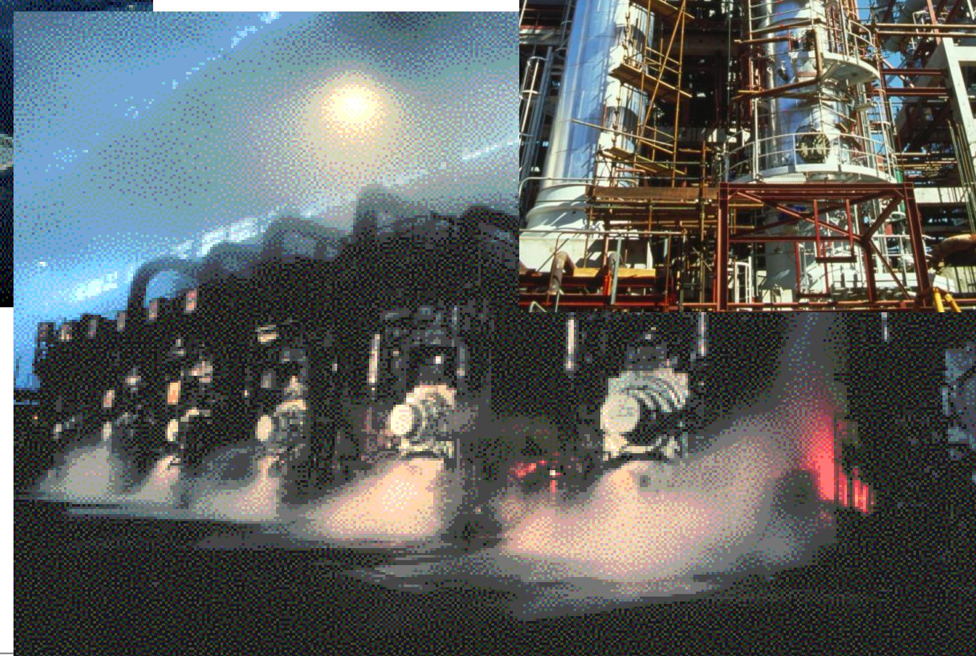
Var finns reglerteknik?

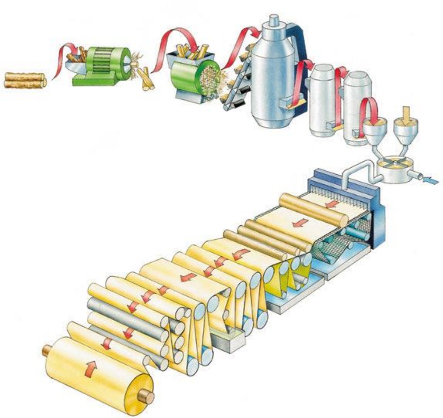
Var finns reglerteknik?

Precis överallt

Power Generation and Distribution

Process Control





Buildings

Design &
Energy Analysis

Windows &
Lighting

Natural
Ventilation

Indoor
Environment

Elevators

Safety

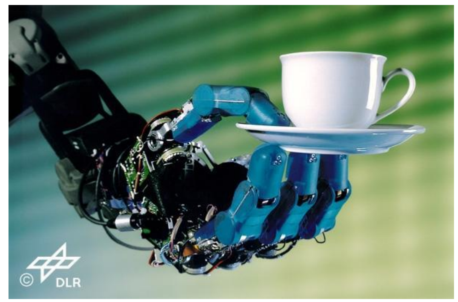
HVAC

Vibration
damping

Sensors, Networks,Communications, Controls

Slide from UTRC

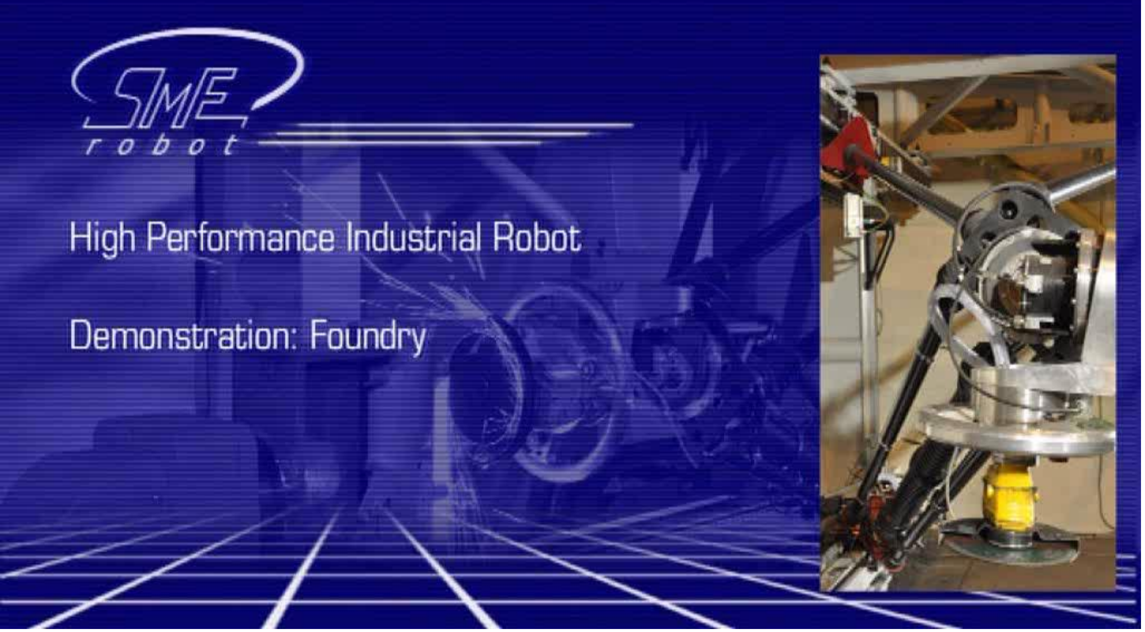
Manufacturing robotics



Stora och små leksaker

http://video.google.se/videoplay?docid=1210345008392050115&ei=tznoSrXwKqDQ2wLP1I2PDw&q=humanoid+robot&hl=sv&client=firefox-a#

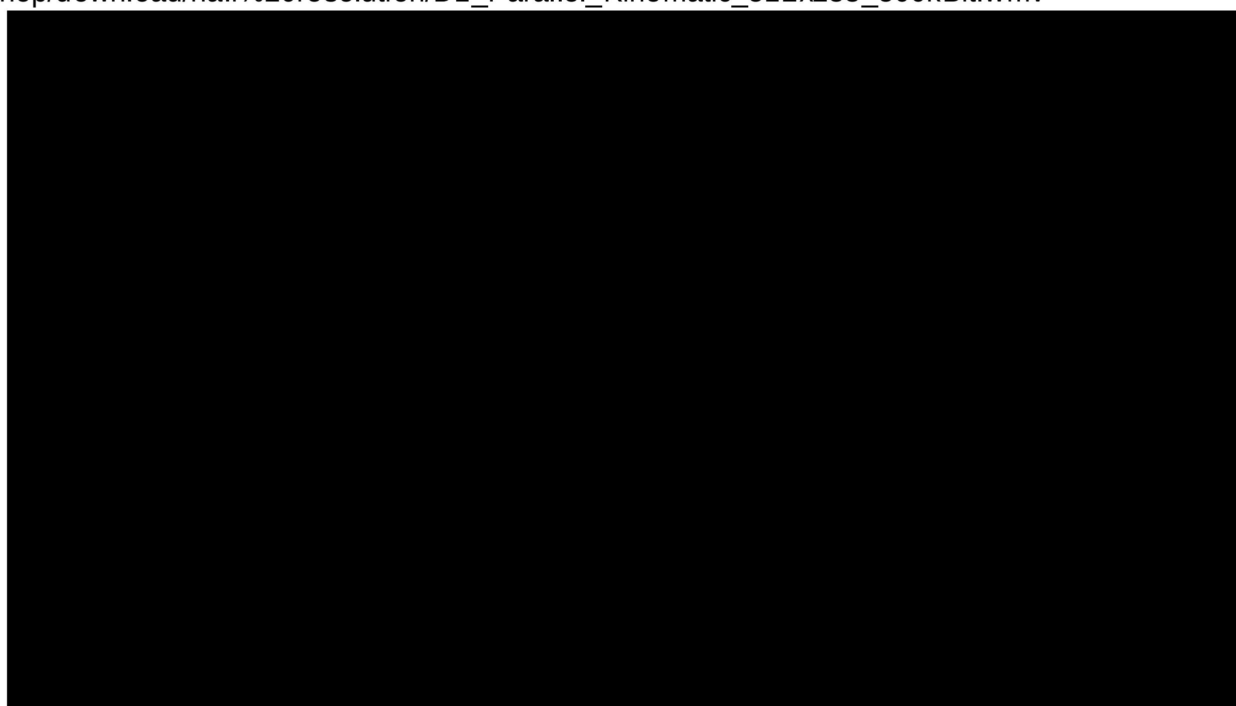
http://www.youtube.com/watch?v=W1czBcnX1Ww



http://www.smerobot.org

http://www.smerobot.org/15_final_workshop/download/half%20resolution/D1_Parallel_Kinematic_512x288_500kBit.wmv

The Fanta Challenge

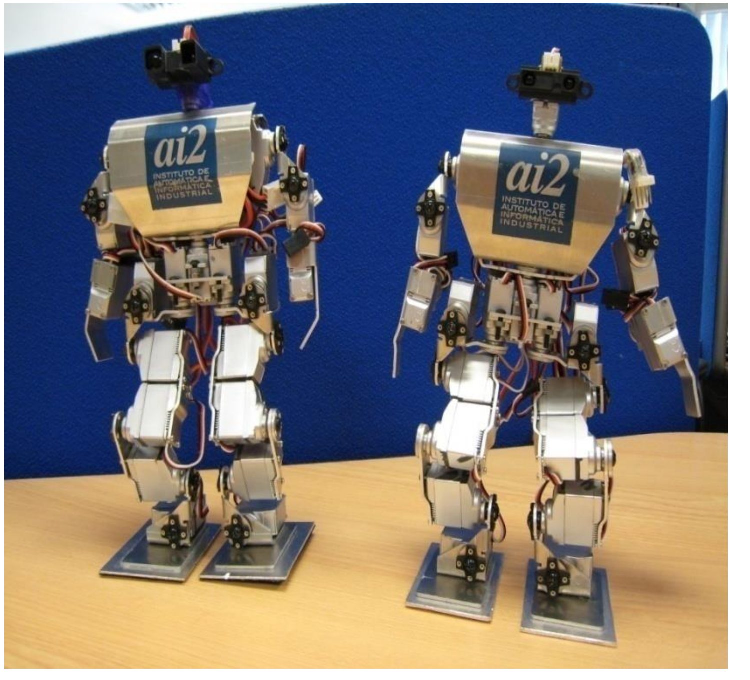


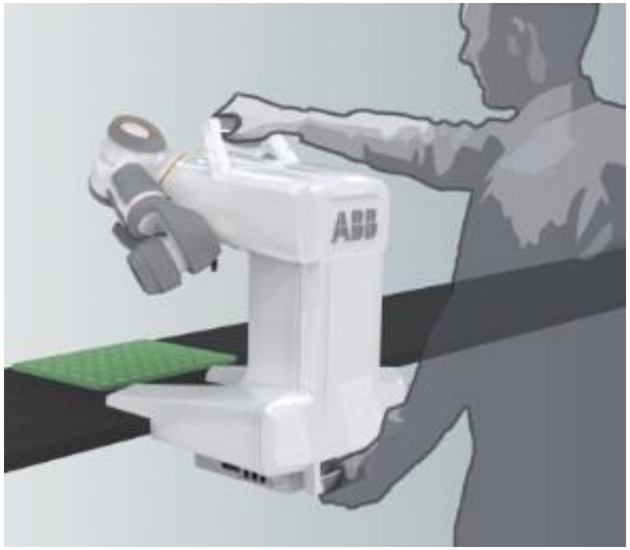
http://www.youtube.com/watch?v=SOESSCXGhFo

Dart- and ball-catching robot

http://www.youtube.com/watch?v=Fxzh3pFr3Gs

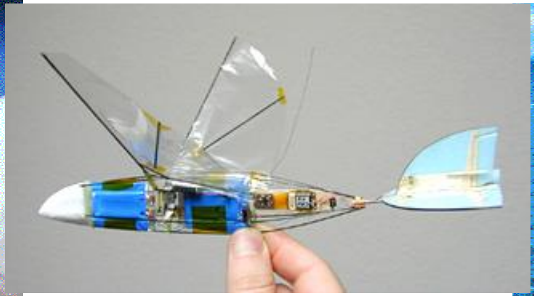
Stora och små leksaker





http://www.youtube.com/watch?v=7JgdbFW5mEg&list=PL13509A9A50E93865

Vehicles



Automotive

Strong technology driver

Engine control

Power trains

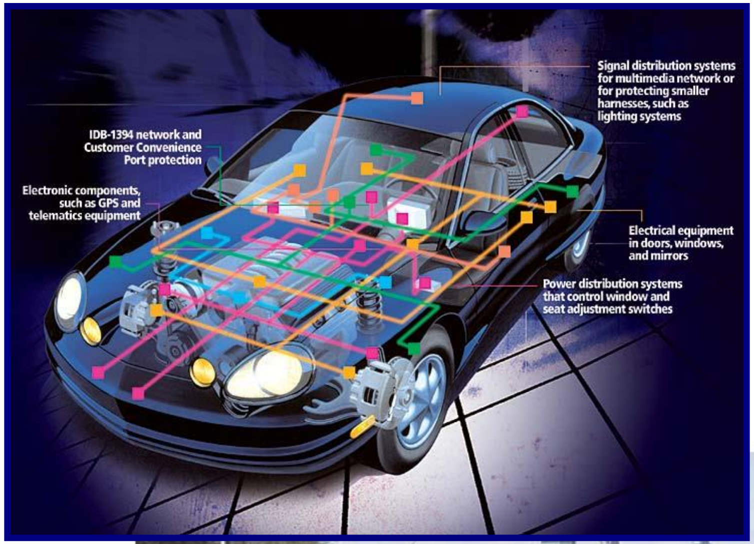
Cruise control

Adaptive cruise control

Traction control

Lane guidance assistance

Traffic flow control



Automotive

Strong technology driver

Engine control

Power trains

Cruise control

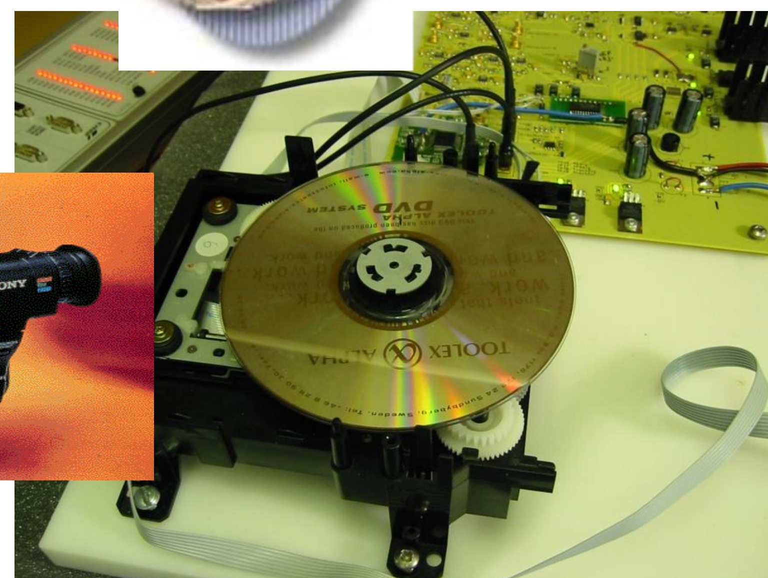
Traction control

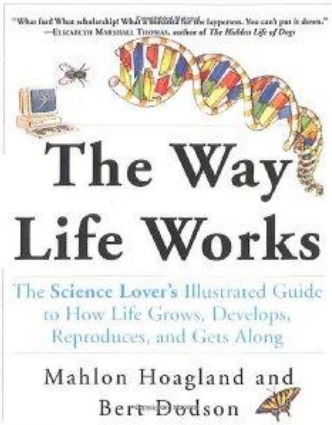
Road-Tire Friction Estimation for AFS Vehicle Control

Master thesis work by

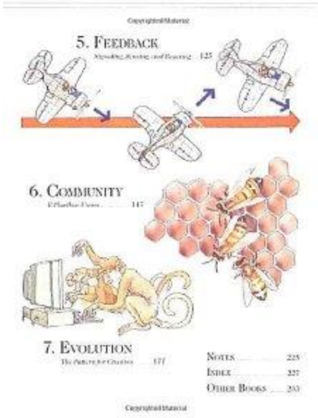
Andreas Andersson

Consumer Electronics





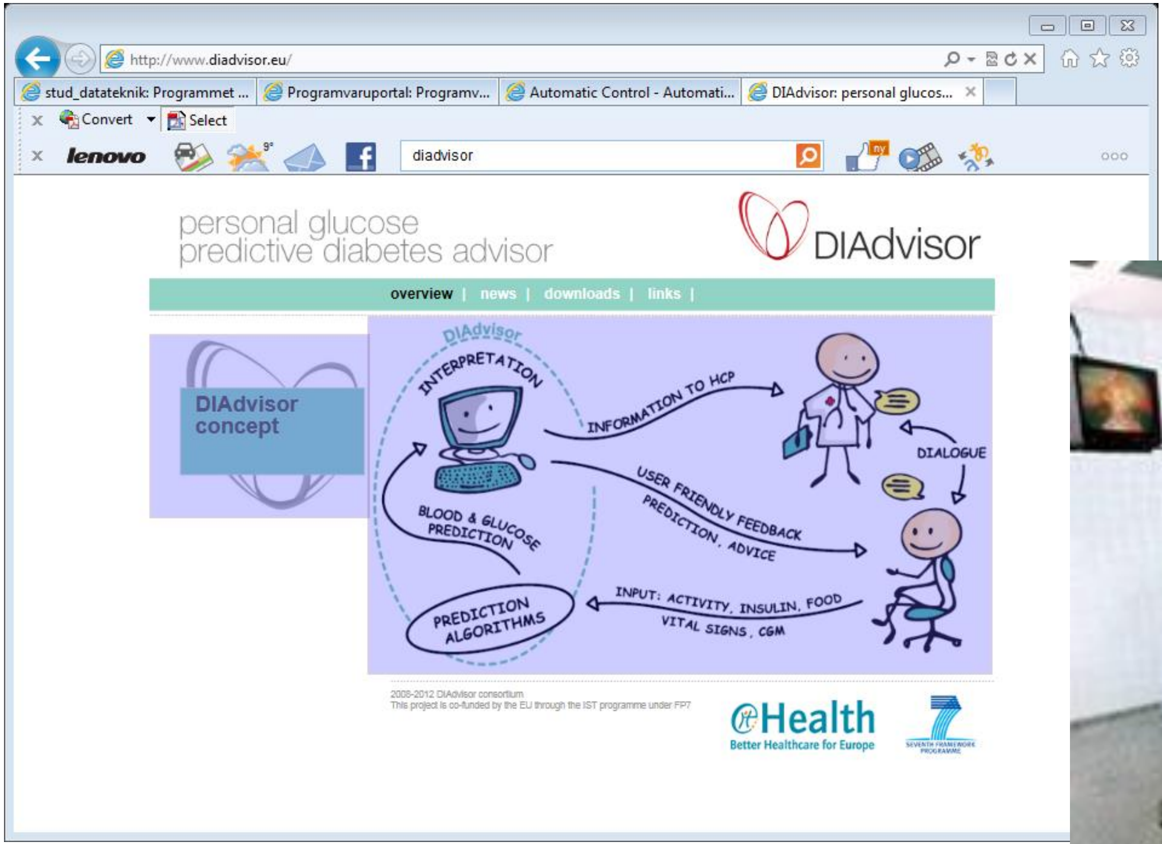
Biology



Feedback is a central feature of life. The process of feedback governs how we grow, respond to stress and challenge, and regulate factors such as body temperature, blood pressure, and cholesterol level. The mechanisms operate at every level, from the interaction of proteins in cells to the interaction of organisms in complex ecologies.

Mahlon B Hoagland and B Dodson The Way Life Works Times Books 1995

Control in medical applications



http://www.intuitivesurgical.com/

http://www.diadvisor.eu/

http://www.youtube.com/watch?v=

Error control of software systems [L.Sha]

- *The idea* behind error control of software is to use ideas similar to the ideas used in feedback control in order to *detect malfunctioning software* components and, in that case *fall back on, a well-tested core software component* that is able to provide the basic application service with guarantees on performance and safety.

- Provide techniques and tools that support making the semantic assumptions of each software component explicit and machine checkable.

Error control of software systems [L.Sha]

- *The idea* behind error control of software is to use ideas similar to the ideas used in feedback control in order to *detect malfunctioning software* components and, in that case *fall back on, a well-tested core software component* that is able to provide the basic application service with guarantees on pe

- Provide techniqu making the semantic assumptions of each software component explicit and machine checkable.

Reglerteknik - Den Dolda Tekniken

- Används överallt
- Mycket framgångsrikt
- En förutsättning för många produkter och system
- Inget som det talas så mycket om
  - Utom när det går snett!
- Varför?
  - Lättare att marknadsföra saker som man kan ta på än principer, metoder och idéer

http://www.youtube.com/watch?v=K4h9qJ6NG9E

JAS – the sequel

http://www.youtube.com/watch?NR=1&v=mkgShfxTzmo

http://www.youtube.com/watch?v=OVr6QJzW094

Stockholm water festival 1993

Hur gör man?

Hur bestämmer vi regulatorn?

- Vi väljer regulatorns struktur och parametrar så att det återkopplade systemet får önskat **dynamiskt** beteende.

- "Att få saker och ting att bete sig som man vill"

Differentialekvationer

$$\frac{dx}{dt} + 3x(t) = 1$$
$$x(0) = 0$$

$$\frac{dx}{dt} - 4x(t) = 1$$
$$x(0) = 0$$

Vilken lösning är "snäll" och vilken lösning är "elak",

d v s stabil resp. instabil?

Differentialekvationer

$$\frac{dx}{dt} + 3x(t) = 1$$

$$x(0) = 0$$

$$\frac{dx}{dt} - 4x(t) = 1$$

$$x(0) = 0$$

$$x(t) = \frac{1}{-3} \cdot \left(e^{-3 \cdot t} - 1\right)$$

$$stabil$$

$$x(t) = \frac{1}{4} \cdot \left(e^{4 \cdot t} - 1\right)$$

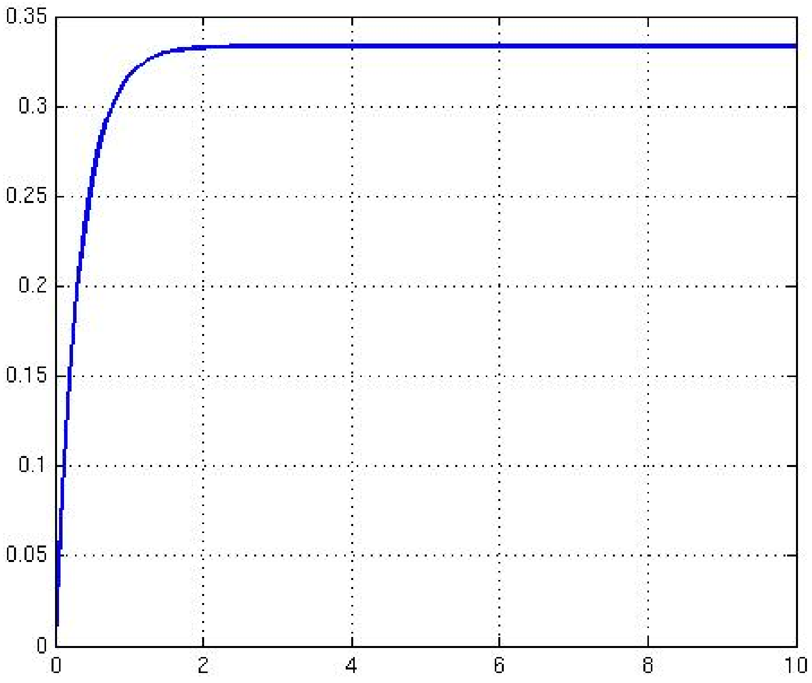
$$instabil$$

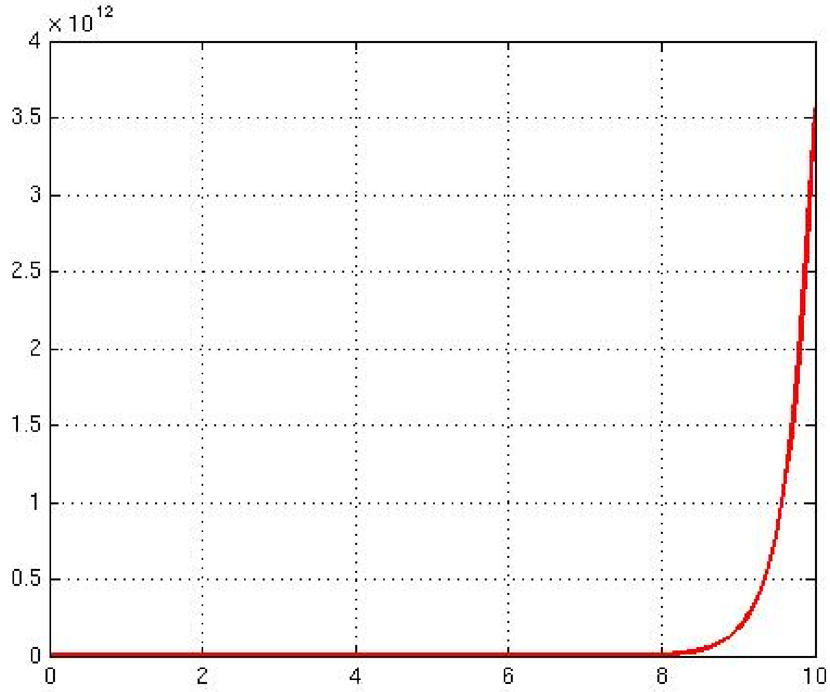
Differentialekvationer

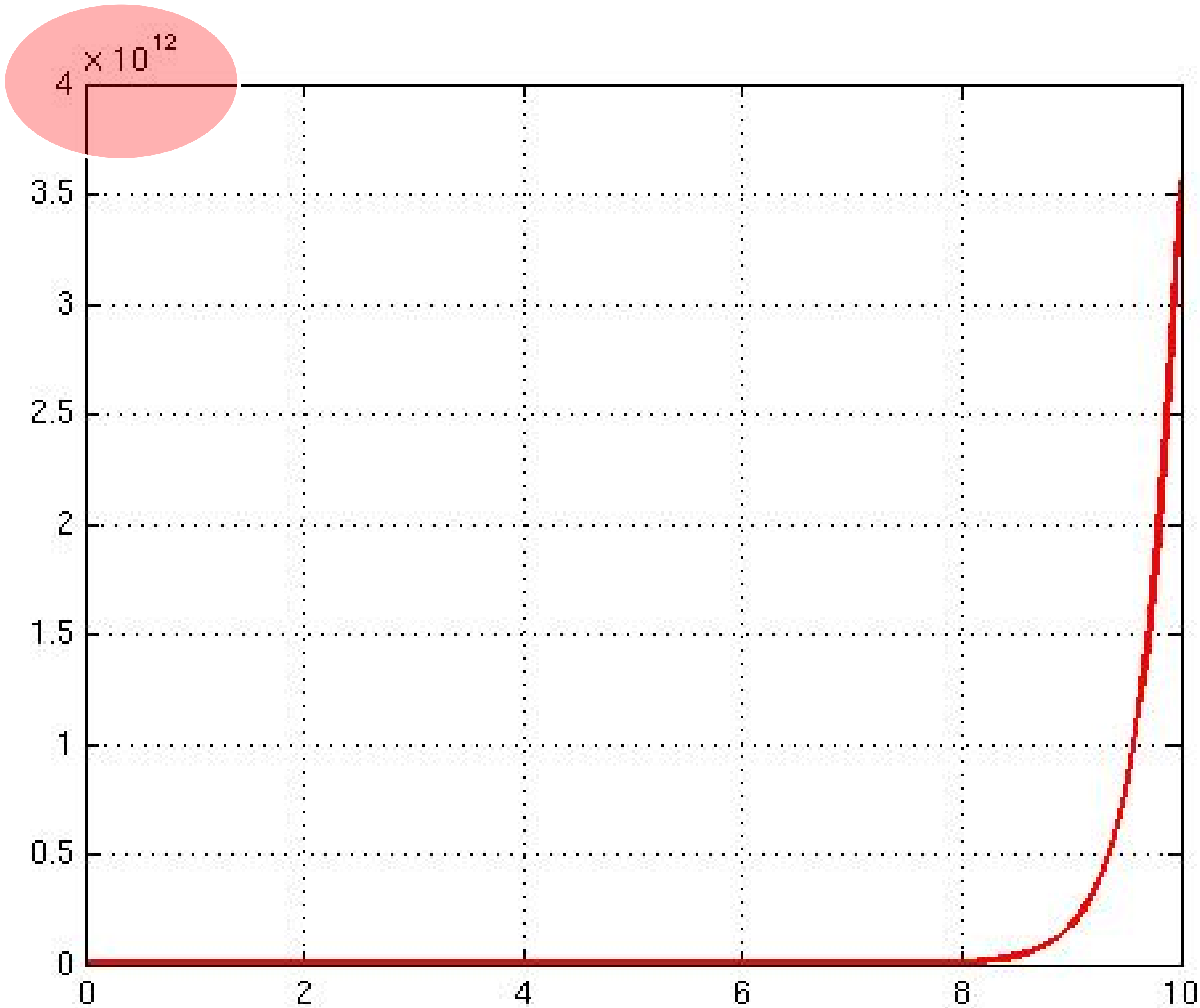
$$x(t) = \frac{1}{-3} \cdot (e^{-3 \cdot t} - 1)$$

$$x(t) = \frac{1}{4} \cdot (e^{4 \cdot t} - 1)$$

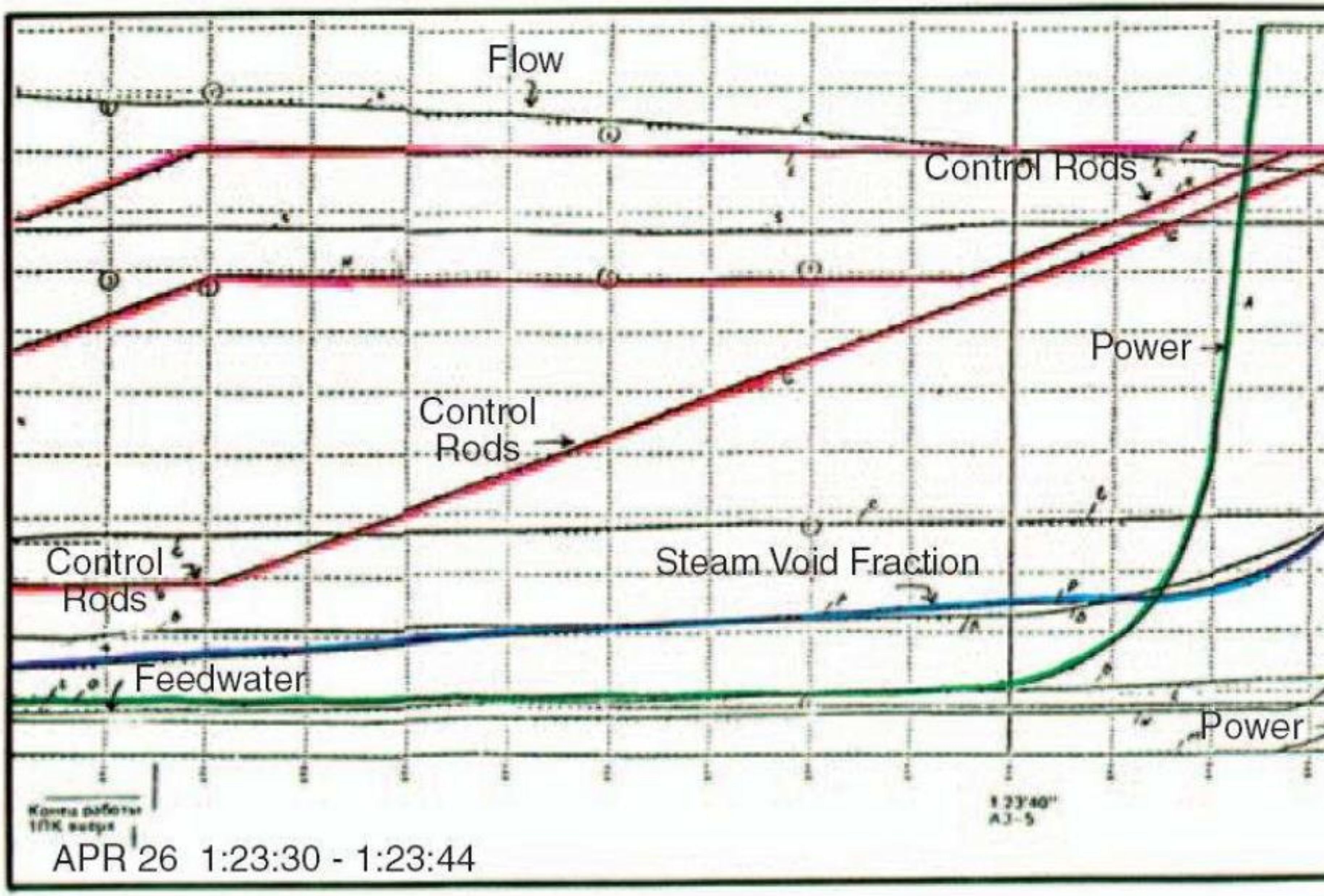
stabil *instabil*







× 10^12
4
3.5
3
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10


Flow
Control Rods
Power
Control Rods
Steam Void Fraction
Control Rods
Feedwater
Power
APR 26 1:23:30 - 1:23:44

$\sim e^{2t}$

Figure 2. *Chernobyl nuclear power plant shortly after the accident on 26 April 1986.*

Reglerteknik AK på en minut

Du bestämmer!

Är lösningen till

$$\frac{dx}{dt} - 4x(t) = u(t)$$

stabil eller instabil ?

Anm. $u(t)$ är en styrsignal som vi kan "välja fritt"...(?!)

Reglerteknik AK på en minut

Välj t.ex

$$u(t) = 1 - Kx(t)$$

Ger det slutna systemet

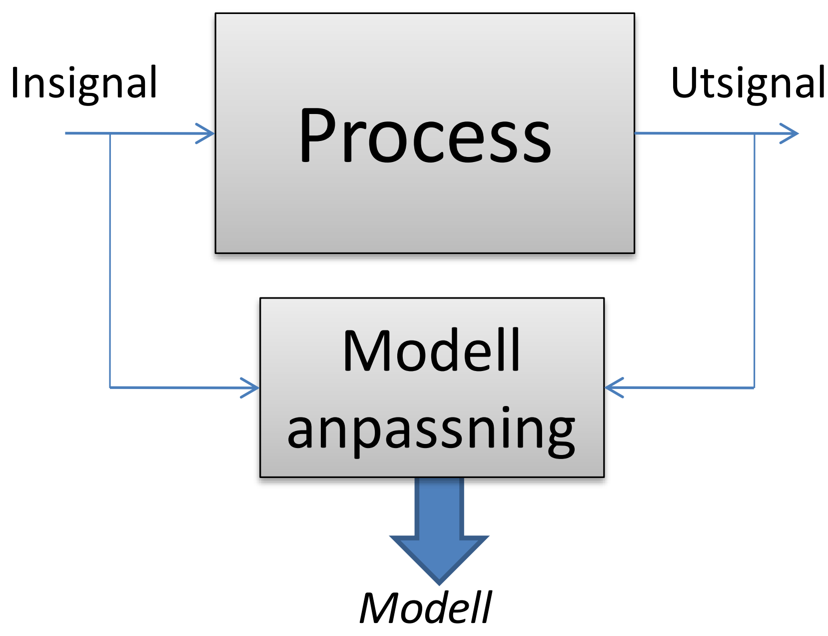
$$\frac{dx}{dt} - 4x(t) + Kx(t) = 1$$

Om K > 4 så blir vårt slutna system stabilt!

Med hjälp av K så bestämmer vi hur snabbt det återkopplade systemet blir **(dvs vi modifierar dynamiken**)

Var kommer modellerna ifrån?

- Antingen genom fysikaliskt modellbygge
  - – Grundlagar från fysik, kemi, mekanik, ……
- … eller genom systemidentifiering



Fysikaliskt Modellbygge

- Balansekvationer (dynamiska samband)
  - Flödesbalanser
    - Upplagring / tidsenhet = inflöde - utflöde + produktion / tidsenhet
    - T.ex. massbalans, partikelbalans, energibalans, strömbalans
  - Intensitetsbalanser
    - Ändring per tidsenhet = drivande storhet - belastande storhet
    - T.ex. Kraftbalans, momentbalans, spänningsbalans
- Konstitiva relationer (statiska samband)
  - Ohms lag (spänning - ström över motsånd)
  - Ventilkarakteristika (tryckfall - flöde genom ventil)
  - Bernoullis lag (vätskenivå - utströmmningshastighet I en tank)
  - Allmänna gaslagen (temperatur - tryck i en gastank)
  - ….

Frekvensanalys

- Ger en beskrivning hur systemet reagerar för olika frekvenser

”Låg frekvens - långsamma tidsförlopp”

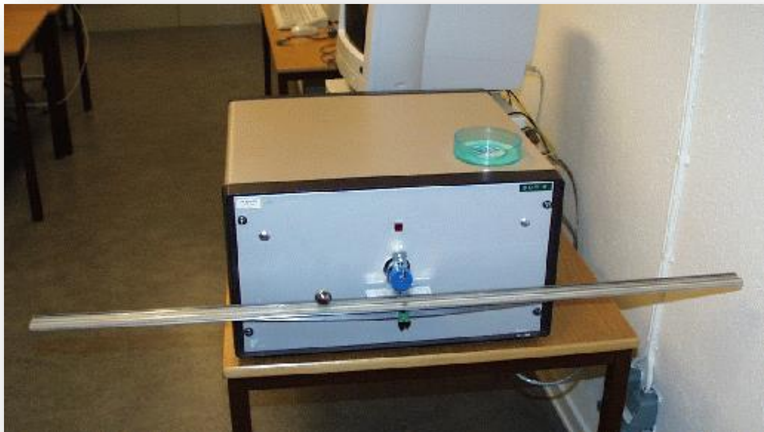
”Hög frekvens – snabba tidsförlopp”

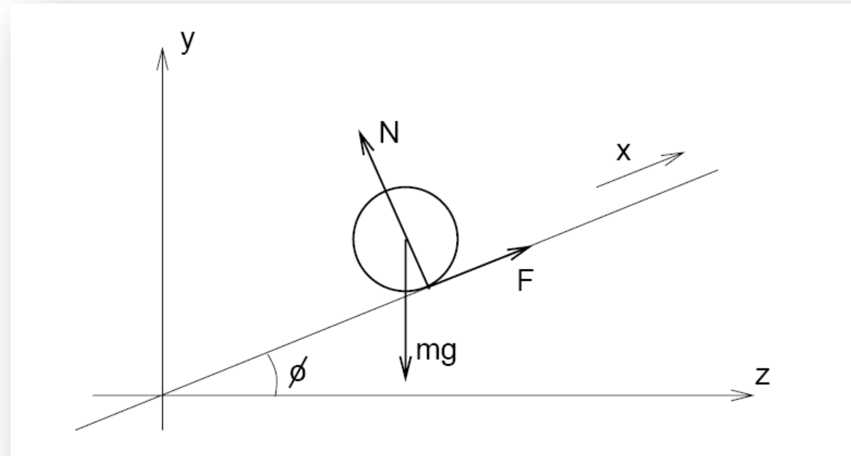
Resonansfrekvenser?

Universell Teori

- Oavsett om det är ett mekaniskt, elektriskt, kemitekniskt, …. system som vi vill reglera så är det samma sorts matematiska modeller som används
- Teorin är helt generell!
  - Principer
  - Metoder
  - Verktyg
- Kurser för F, E, D, C, M, I, Pi, K, B, W, N
- Mycket bred arbetsmarknad
  - Ericsson, ABB, Gambro, Tetra Pak, Haldex, Volvo, Konsulter, …..

Exempel: Kulan på Bommen





Kraftbalansekvationer → olinjär differentialekvation → approx med linjär diffekvation

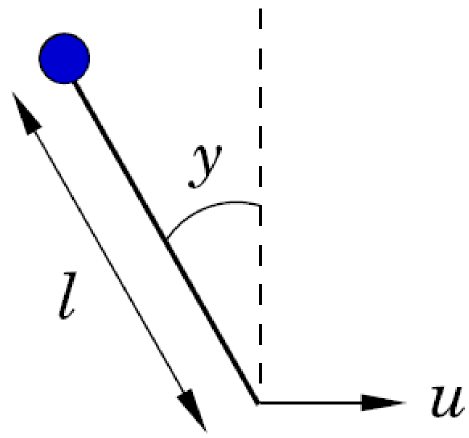
$$m(\ddot{x} - x\dot{\phi}^2) = -mg\sin\phi - \frac{2}{5}m\ddot{x}.$$

→

$$m\ddot{x} = -mg\phi - \frac{2}{5}\ddot{x}$$

[Control of a ball on a beam, adjusted Td-parameter]

Exempel: Inverterad Pendel



En enkel andra-ordningens differentialekvation ges av

$$\frac{d^2y}{dt^2} = \omega_0^2 \sin y + u\,\omega_0^2 \cos y$$

där $\omega_0 = \sqrt{\frac{g}{l}}$ är systemets/pendelns egenfrekvens.

Exempel: Inverterad Pendel

Om man **linjäriserar** kring övre jämviktsläget får man tillståndsmodellen

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 \\ \omega_0^2 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ \omega_0^2 \end{pmatrix} u$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} x$$

Instabilt linjärt system

Om man linjäriserar systemet runt det nedre jämnviktsläget och lägger till en dämpningsterm proportionell mot vinkelhastigheten så får man ett (oscillativt) stabilt linjärt system

Exempel: Inverterad Pendel

Om man **linjäriserar** kring övre jämviktsläget får man -
tillståndsmodellen

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 \\ \omega_0^2 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ \omega_0^2 \end{pmatrix} u$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} x$$

Instabilt linjärt system

Stabiliserande reglering dock känslig för fördröjningar...

Exempel: Inverterad Pendel

- Bra reglering kräver att man både återkopplar från vinkel och vinkelhastighet
- Vinkelhastighet?
  - Sensor som mäter vinkelhastighet
  - Mäter enbart vinkel och beräknar vinkelhastighet genom en enkel differensapproximation

$$\frac{dv}{dt} = \frac{v(k) - v(k-1)}{T_s}$$

  - Filter som utnyttjar att man har en modell av pendeln

Exempel: Segway

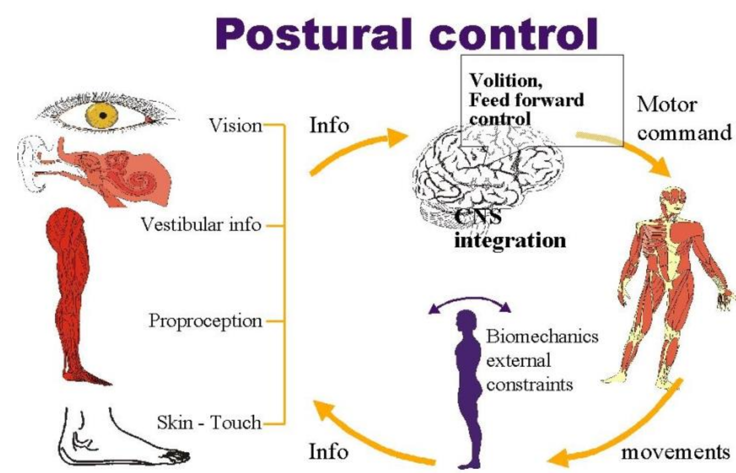
Segway Varianter

http://www.noob.us/miscellaneous/introducing-the-enicycle/

http://www.dailymail.co.uk/sciencetech/article-1215741/Honda-unveils-Segway-style-unicycle-travels-direction-want.html

http://abcnews.go.com/Video/playerIndex?id=7283602

Andra inverterade pendlar



Olinjär Dynamik

- Kan ge upphov till mycket!

http://www.youtube.com/watch?v=lXyG68_caV4

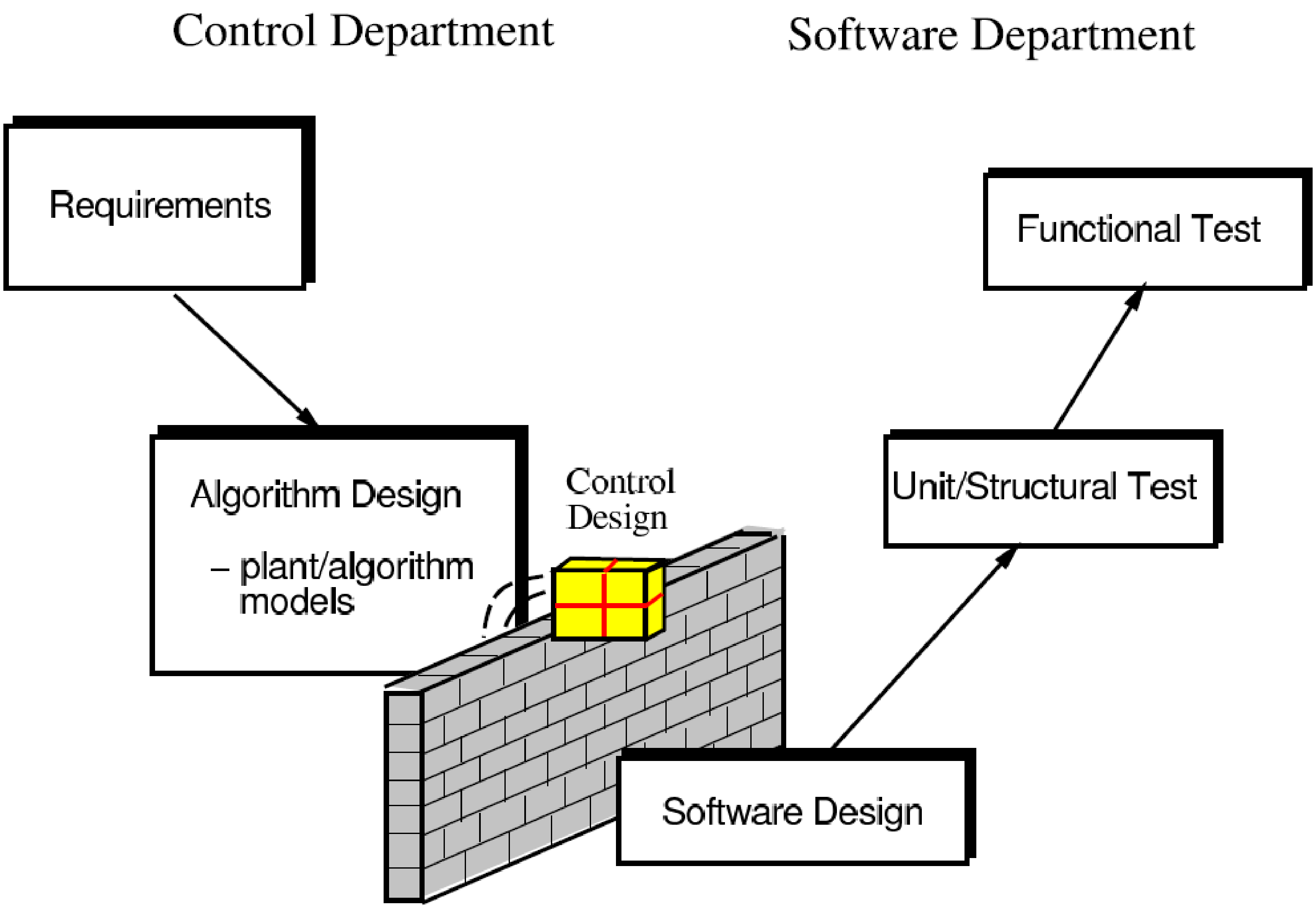
"Limit cycles"

Varför reglerteknik på D?

Väldigt naturligt

- De flesta reglersystem implementeras idag i programvara eller programmerbar hårdvara
- Programvaruingenjörer måste ha förståelse för dynamik och återkoppling för att kunna bygga "styrbara" system
  - Bra möjligheter i Lund

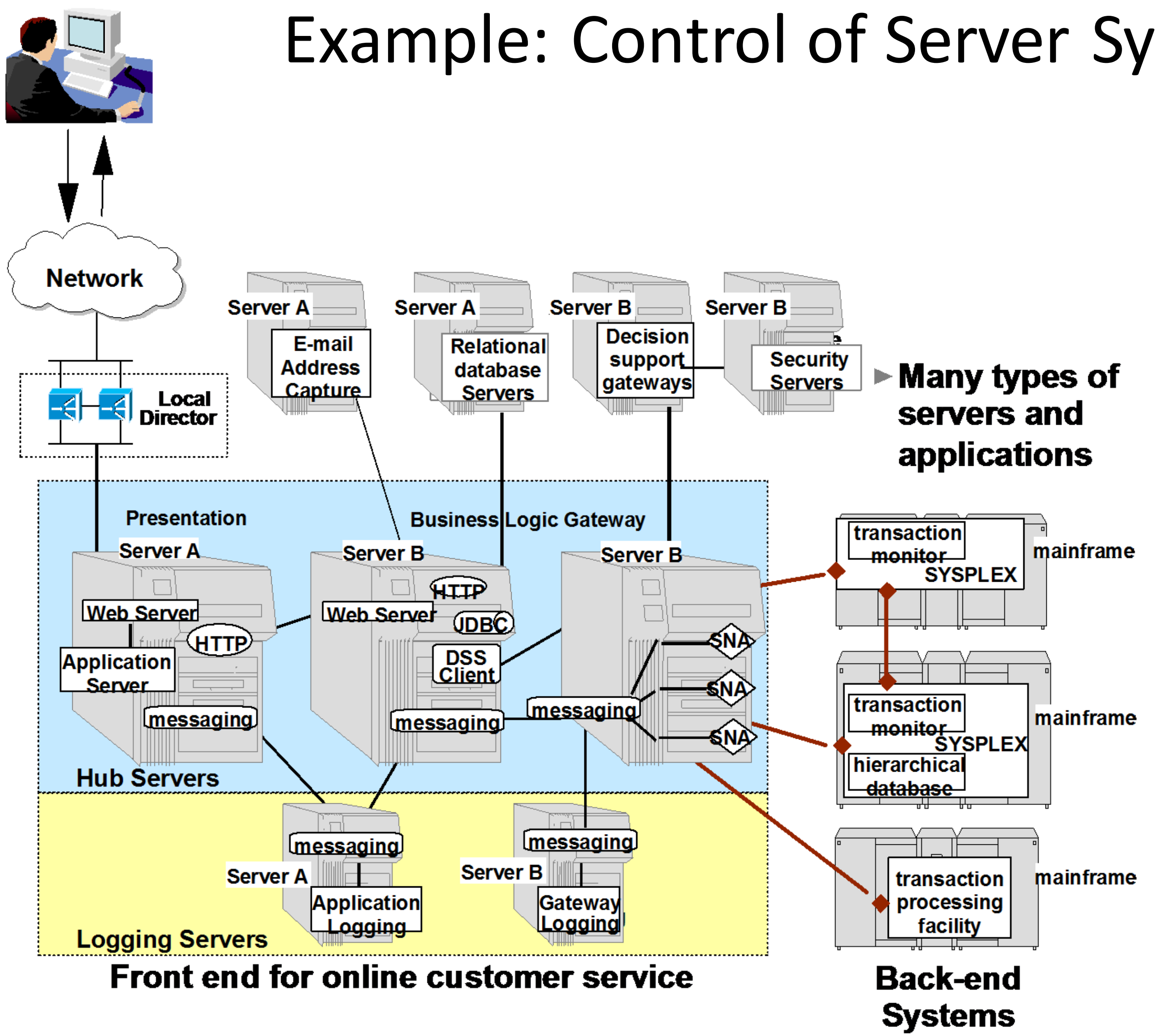
Om inte .....


Control Department
Software Department
Requirements
Algorithm Design
– plant/algorithm models
Control Design
Functional Test
Unit/Structural Test
Software Design

Väldigt naturligt

- De flesta reglersystem implementeras idag i programvara eller programmerbar hårdvara
- Programvaruingenjörer måste ha förståelse för dynamik och återkoppling för att kunna bygga "styrbara" system
  - Bra möjligheter i Lund
- Återkoppling används mer och mer internt i hårdvaru och programvarusystem för att förbättra prestanda och robusthet
  - TCP/IP
  - Reglering av serversystem
  - Resurshantering i mobiltelefoner

Example: Control of Server Systems



Multi-tier systems of Web browsers, business logic and databases

Feedback at various levels

Queue Control

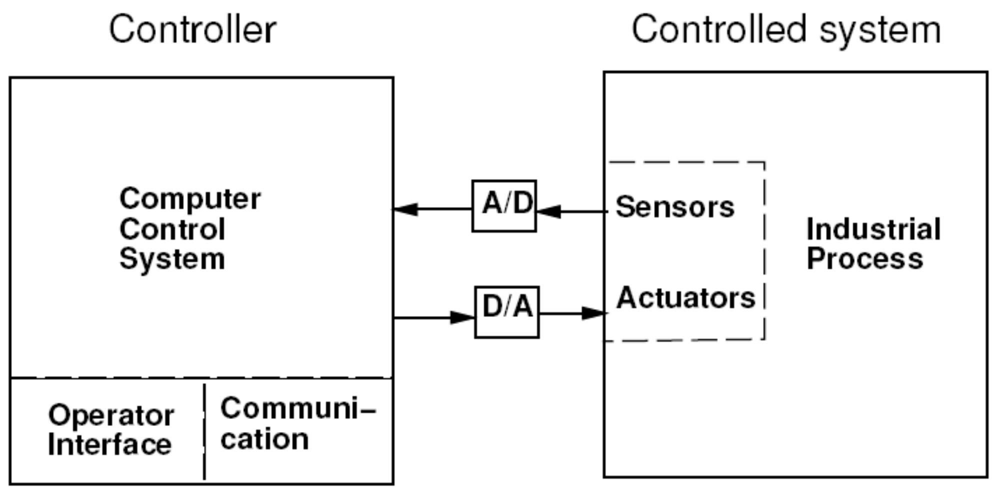
IBM, HP, Microsoft, Amazon, ….

Challenges:

- Modeling formalisms (DES, ODEs, queuing theory, …)
- Design of software and computing systems for controllability

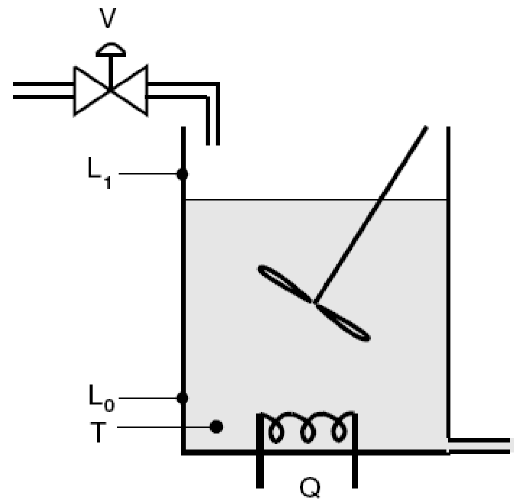
Real-Time Control Systems

Many real-time systems are real-time control systems.



- control algorithms
- process presentation
- operator communication
- data communication

A Typical Control Problem: The Buffer Tank



Raw material buffer + heating

Goals:

- Level control: open V when level below L_0, keep the valve open until level above L_1
- Temperature control: PI-controller

Typical Characteristics

- Parallel activities.
- Timing requirements – more or less hard.
- Discrete and analog signals.
- Continuous (time-driven) control and Discrete (event-driven), sequential Control

All control systems have these characteristics.

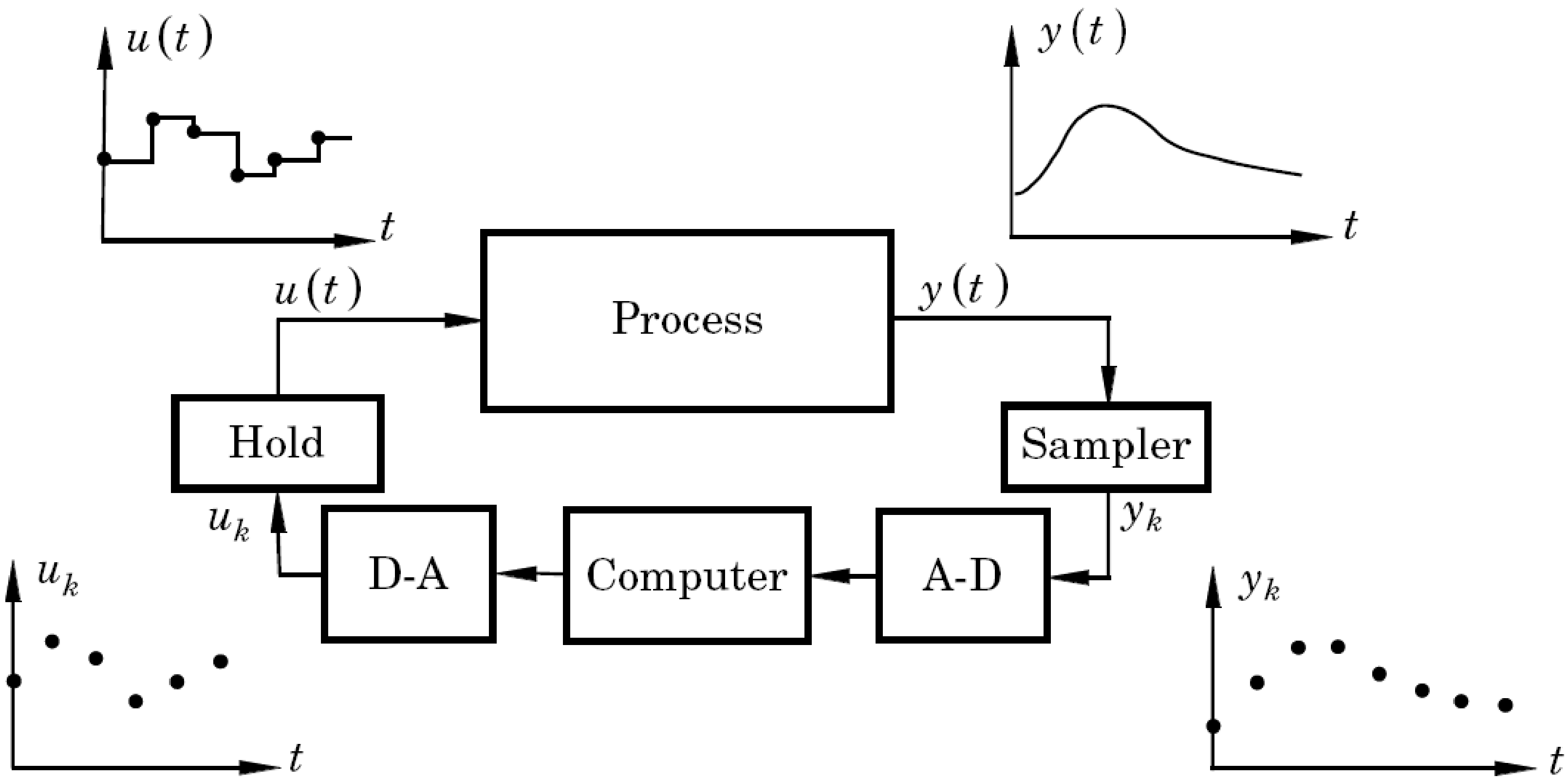
Sampling - Control - Actuation

Frequently:

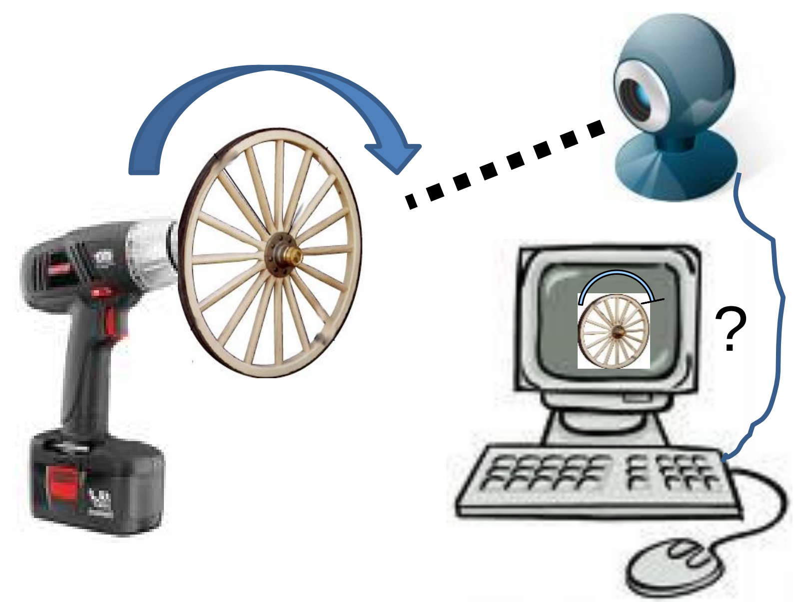
- Sampling of measured signal $y(t)$
- Calculation of control signal (software algorithm)
- Actuation of calculated control signal $u(k)$

In most cases periodically, i.e. driven by a clock (time)

Sampled-data control systems

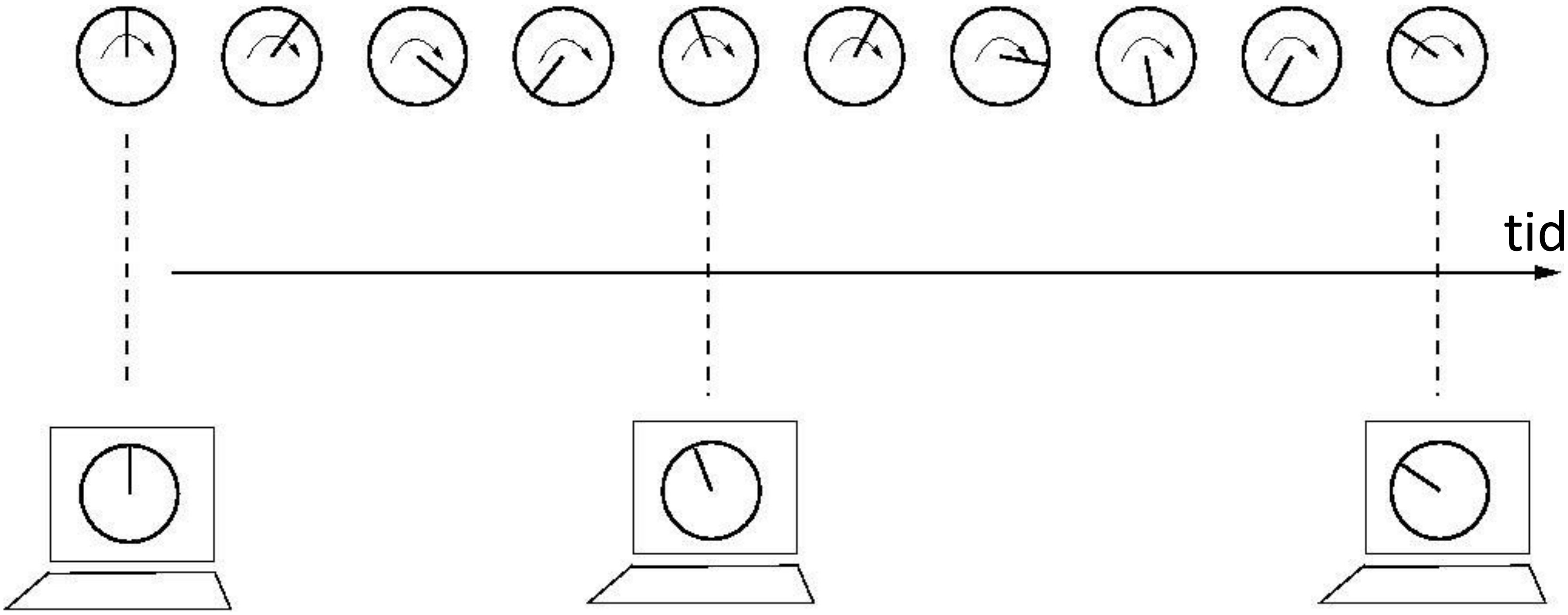


Varför rör sig hjulen på dilligensen ibland åt “fel håll” på film?

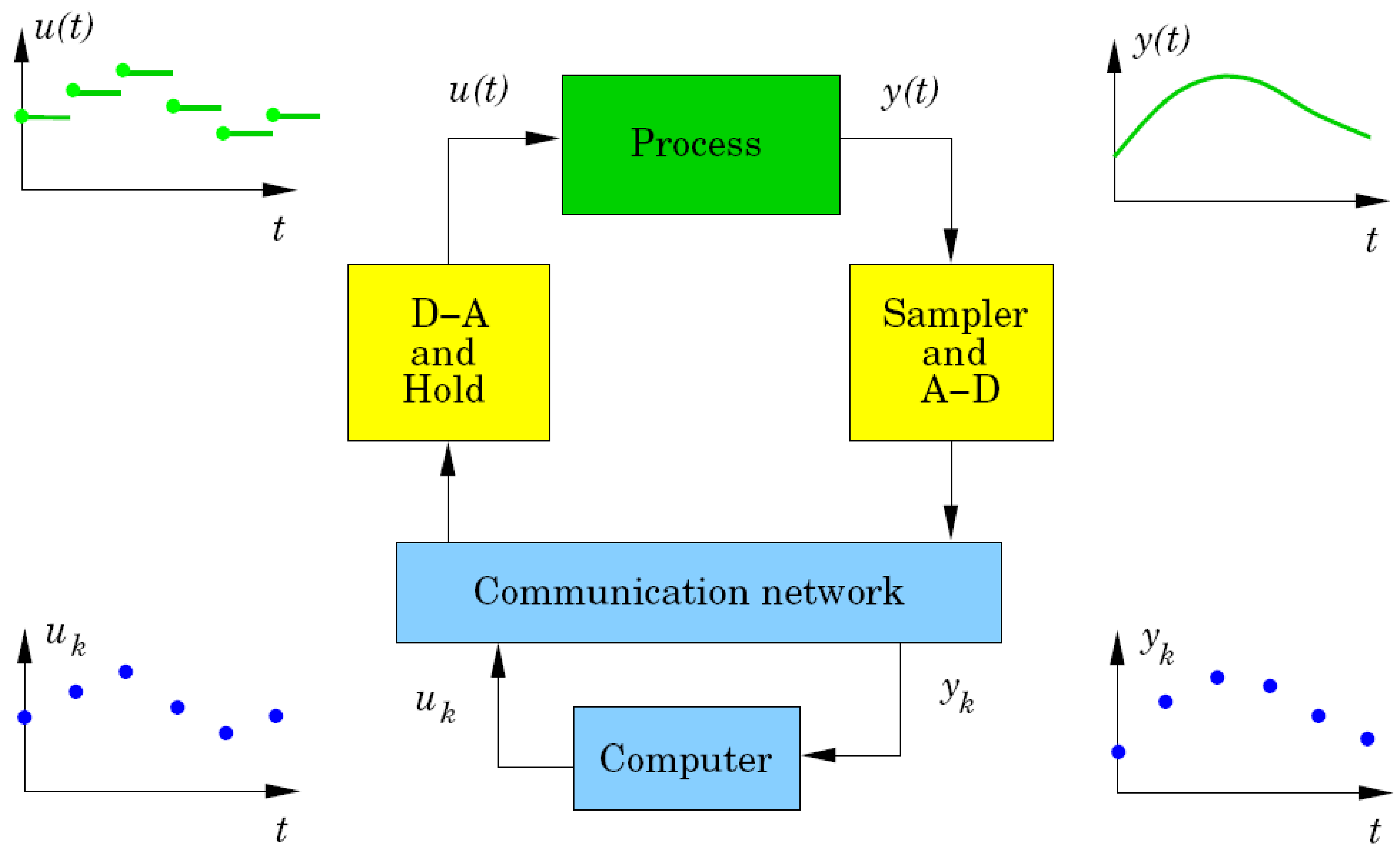


Sampling av signaler

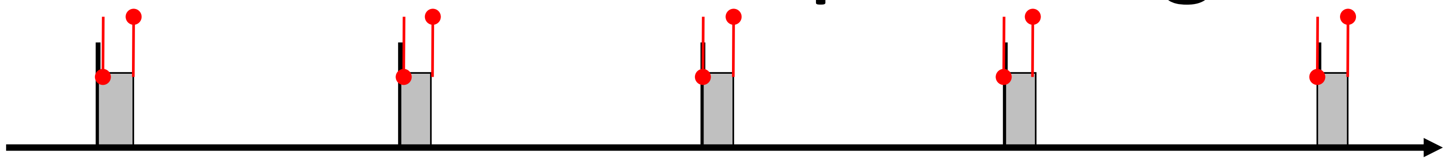
- Vikning /alias :
  - *För långsam sampling* ger upphov till FALSKA frekvenser



Networked control systems



Control Loop Timing



- Classical control normally assumes periodic sampling
  - too long sampling interval or too much jitter give poor performance or instability
- Classical control assumes negligible or constant input-output latency (from sampling to actuation)
  - if the latency is small compared to the sampling interval it can be ignored
  - if the latency is constant it can be included in the control design
  - too long latency or too much latency jitter give poor performance or instability
- Not always possible to achieve with limited computing resources that are shared with other applications

Real-Time Control Systems

Two types of real-time control systems:

1. Embedded Systems
    - dedicated control systems
    - the computer is an embedded part of some piece of equipment
    - microprocessors, real-time kernels, RTOS
    - aerospace, industrial robots, vehicular systems, ...
2. Industrial Control Systems
    - distributed control systems (DCS), programmable logic controllers (PLC), Soft-PLCs
    - hierarchically organized, distributed control systems
    - process industry, manufacturing industry, ...

Embedded Systems are Everywhere

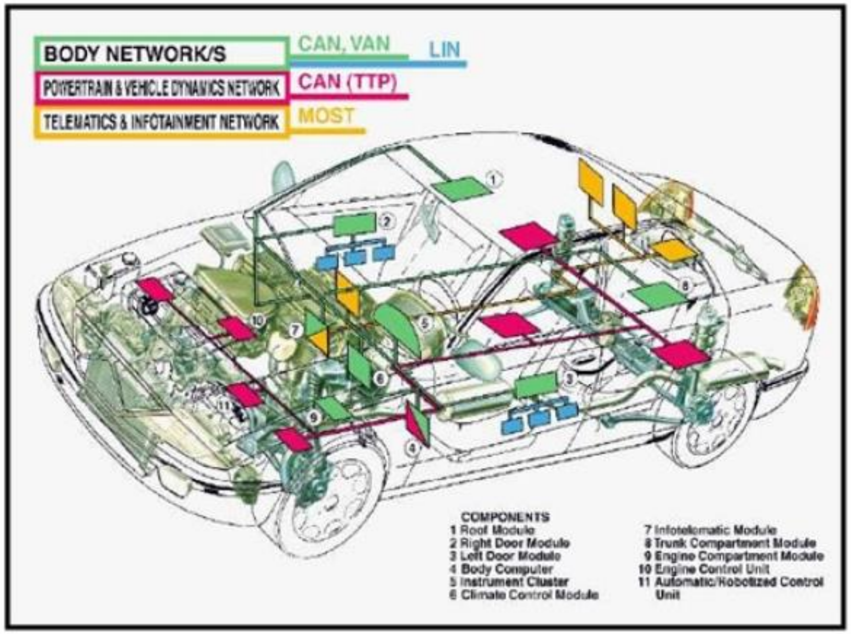
Electronic components integrate **software and hardware** jointly and specifically designed to provide given functionalities, which are often **critical**.

Example: Modern Cars

- **Embedded control systems in modern car** (brakes, transmission, engine, safety, climate, emissions, …)

40-100 ECUs in a new car

~ 2-5 miljon lines of code

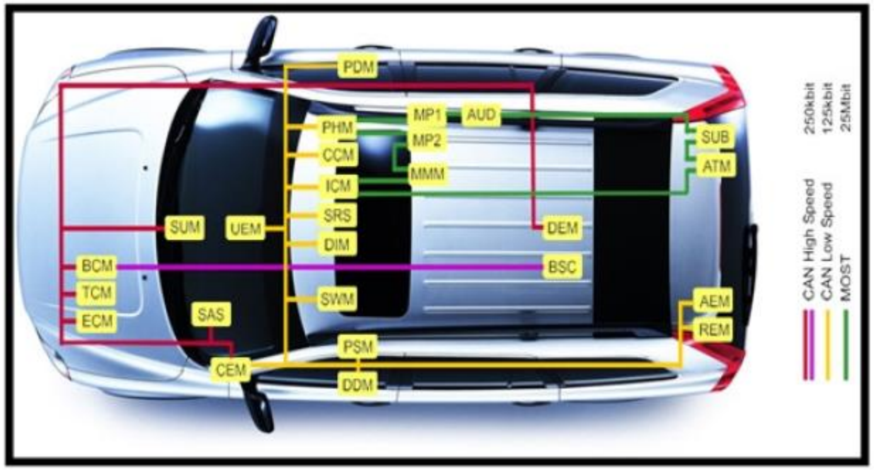


Example: Modern Cars

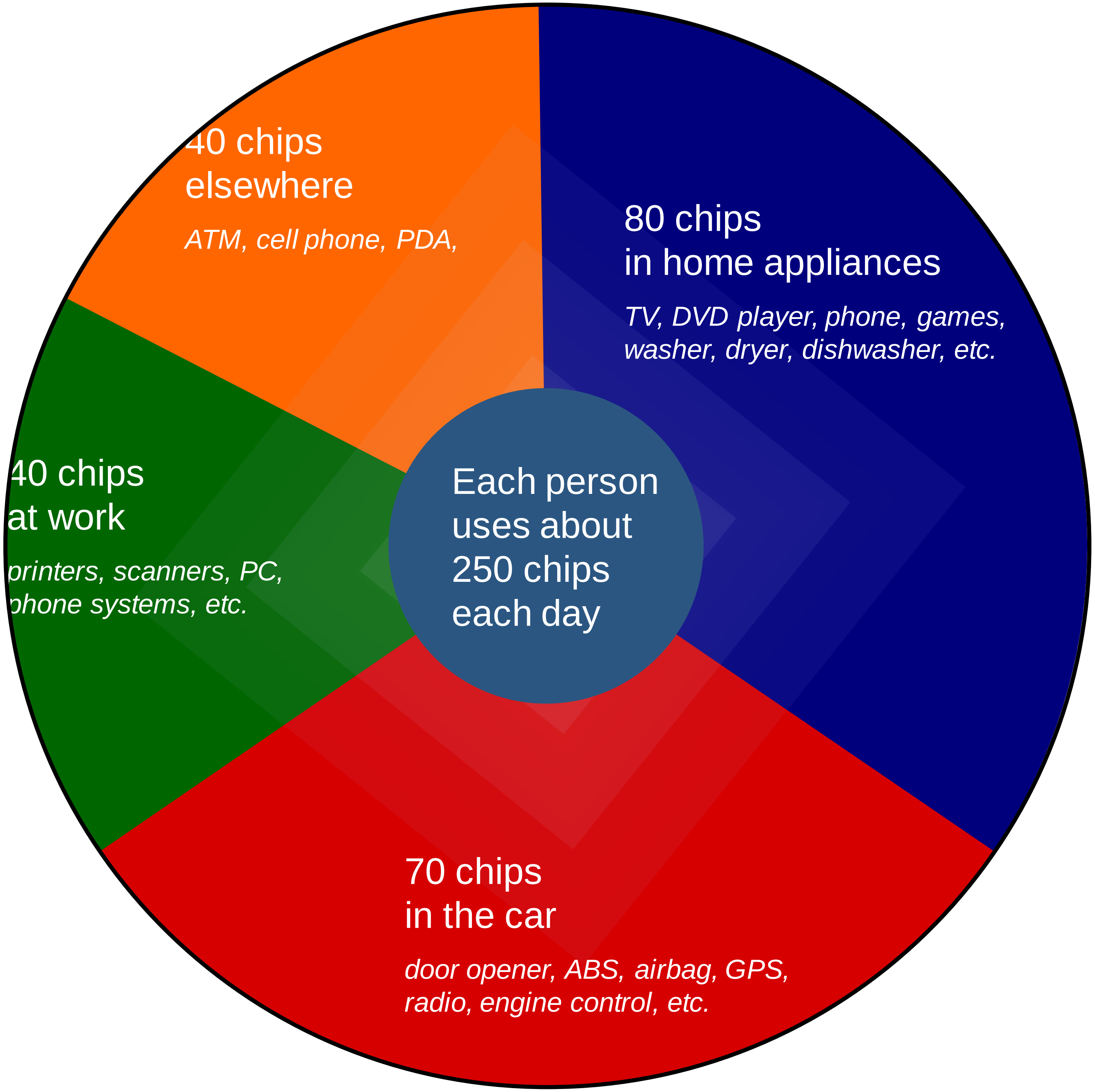
- Networked Systems

Volvo XC 90:

- **_3 CAN-buses_**
- **_+ other buses_**



Embedded Systems are Everywhere



1 billion transistors used per person each day (2008)

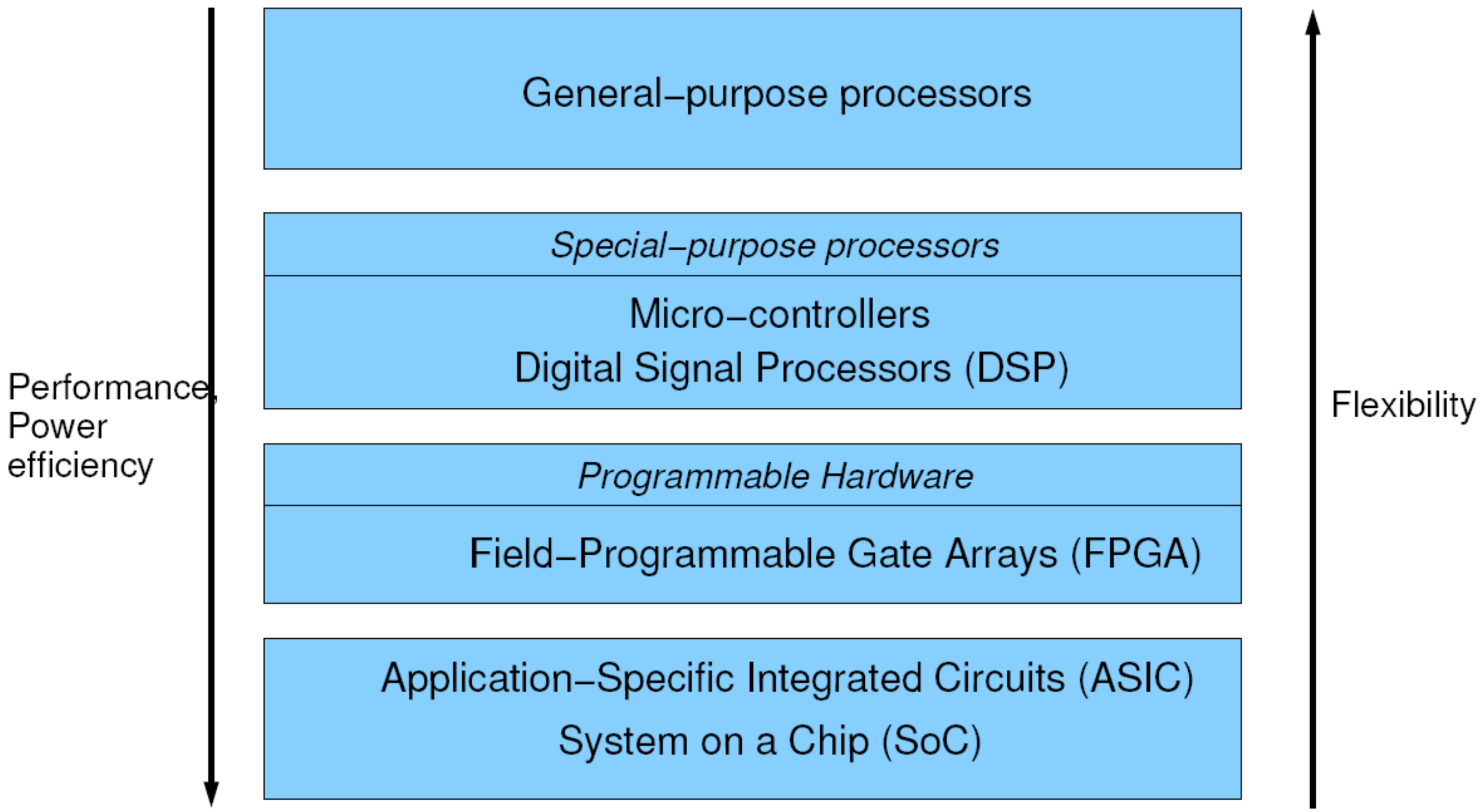
Embedded Control Characteristics

- Limited computing and communication resources
  - Often mass-market products, e.g., cars
  - CPU time, communication bandwidth, energy, memory, ...
- Autonomous operation
  - No human “operator” in the loop
  - Several use-cases and complex functionality
    - Often large amounts of software
  - Need for formal guarantees

Embedded Control Characteristics

- Limited resources $\Rightarrow$ Efficiency
  - Code-size efficiency
  - Run-time efficiency
  - Energy efficiency
  - Weight and size efficiency
  - Cost efficiency
- Autonomous operation $\Rightarrow$ Dependability
  - Reliability
  - Availability
  - Safety
  - Security
  - Maintainability

Vanliga implementationsmetoder



Microcontroller (MCU)

- ”Computer-on-a-chip”
- A microprocessor containing all the memory and interfaces required for a simple (control) application
  - Processor (4bit – 64bit)
  - IO interfaces, e.g., UARTs and AD/DA converters
  - Peripherals (e.g., timers, watchdogs)
  - Serial communications interfaces, e.g., I^2C, CAN
  - ROM, EPROM, or EEPROM (Flash) memory for program storage
  - RAM memory for data storage
  - Clock generator

Parallelism

The real world is parallel

Events may occur at the same time.

The work that has to be done to service an event is called the task associated with the event.

It is often natural to handle the different tasks independently during design.

Temperature Loop

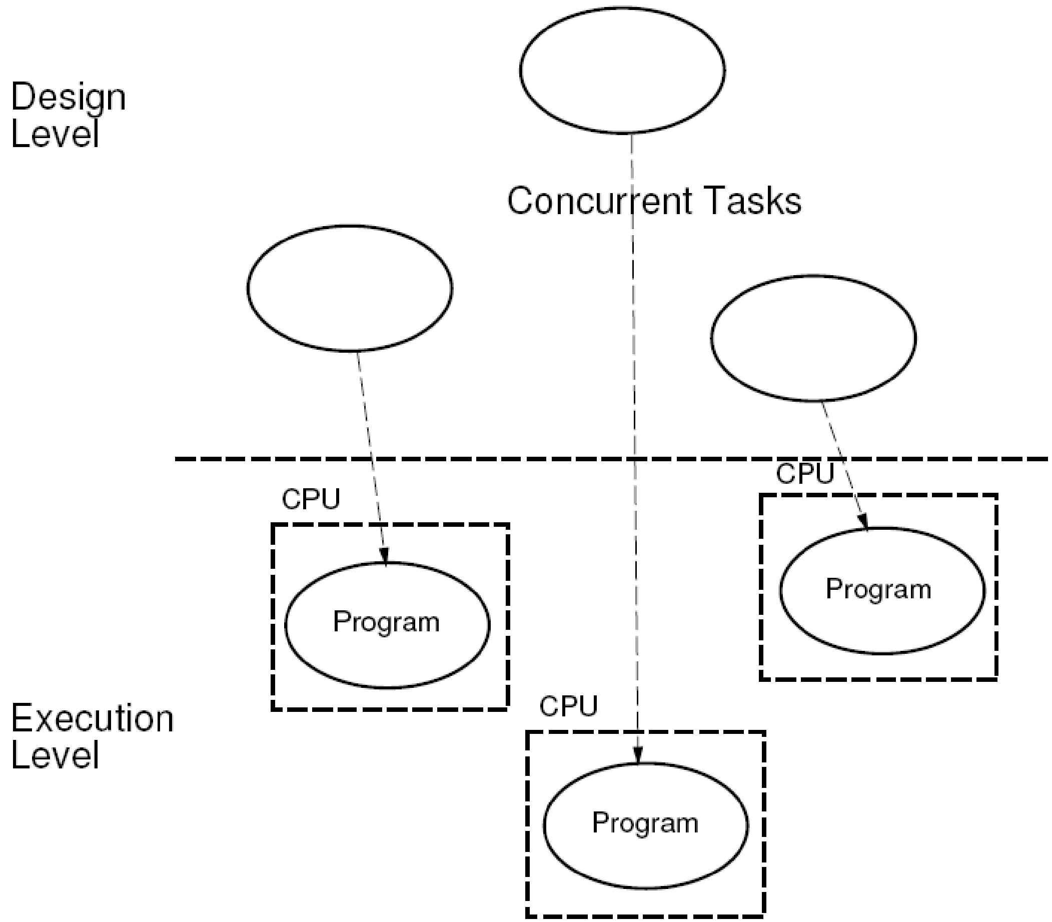
```
while (true) {
   Measure temperature;
   Calculate temperature error;
   Calculate the heater signal with PI−control;
   Output the heater signal;
   Wait for h seconds;
}
```

Level Loop

```
while (true) {
   Wait until level below L0;
   Open inlet valve;
   Wait unitil level above L1;
   Close inlet valve;
}
```

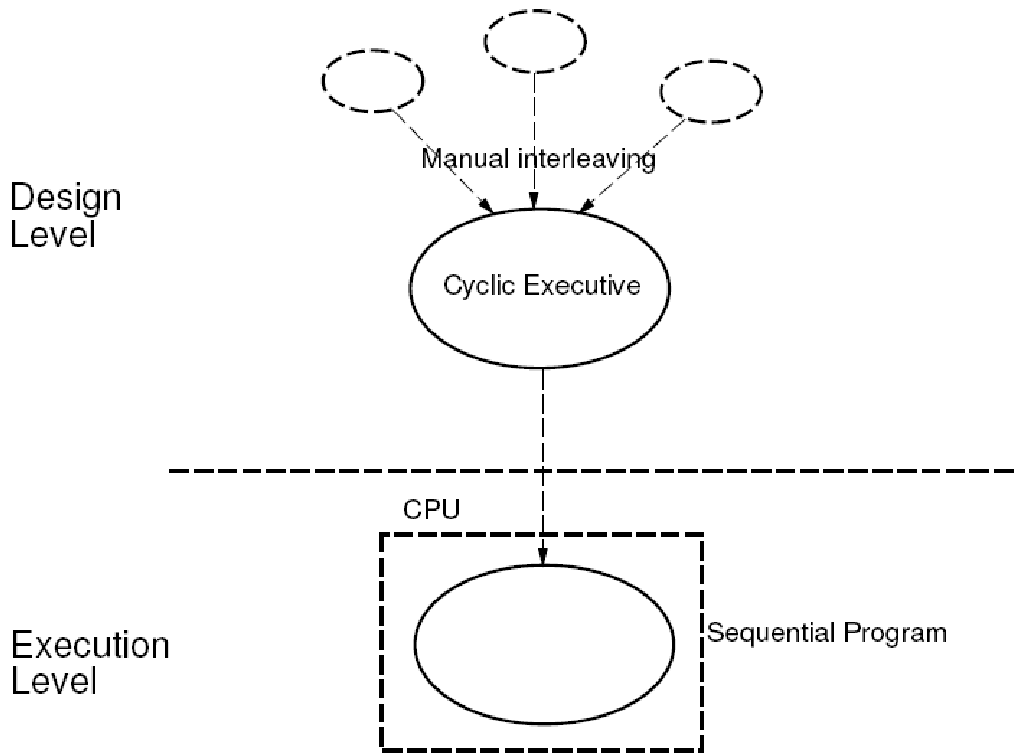
Paradigms

Parallel programming:



Paradigms

Sequential programming:



Interleaved temperature and level loops

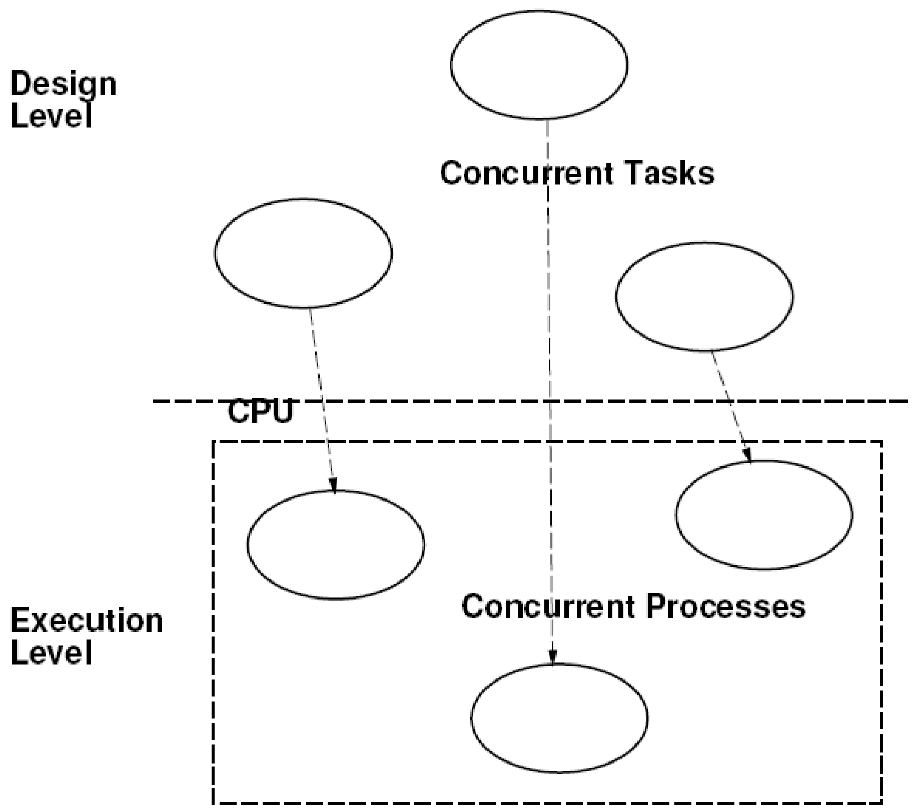
```
while (true) {
  while (level above L0) {
    Measure temperature;
    Calculate temperature error;
    Calculate the heater signal with PI−control;
    Output the heater signal;
    Wait for h seconds;
  }
  Open inlet valve;
  while (level below L1) {
    Measure temperature;
    Calculate temperature error;
    Calculate the heater signal with PI−control;
    Output the heater signal;
    Wait for h seconds;
  }
  Close inlet valve;
}
```

Complex and non user-friendly code

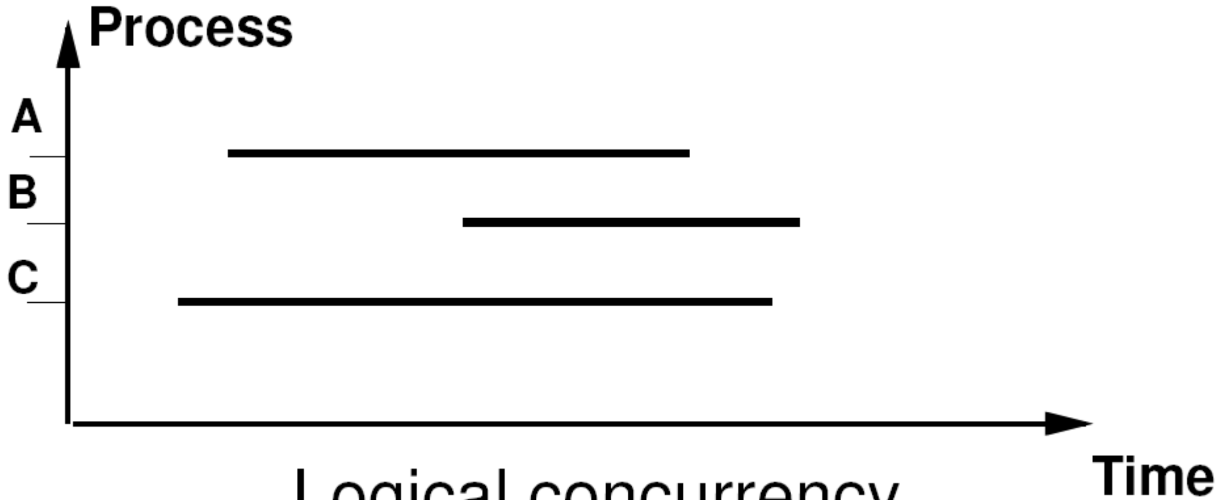
Can, however, often be automated.

Paradigms

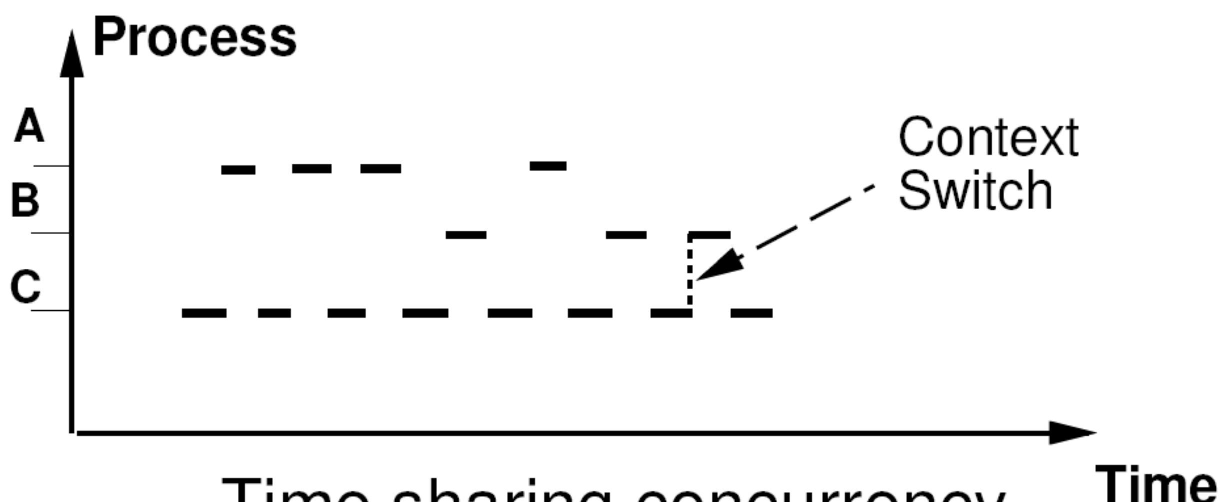
Concurrent programming:



The CPU is shared between the process (switches)

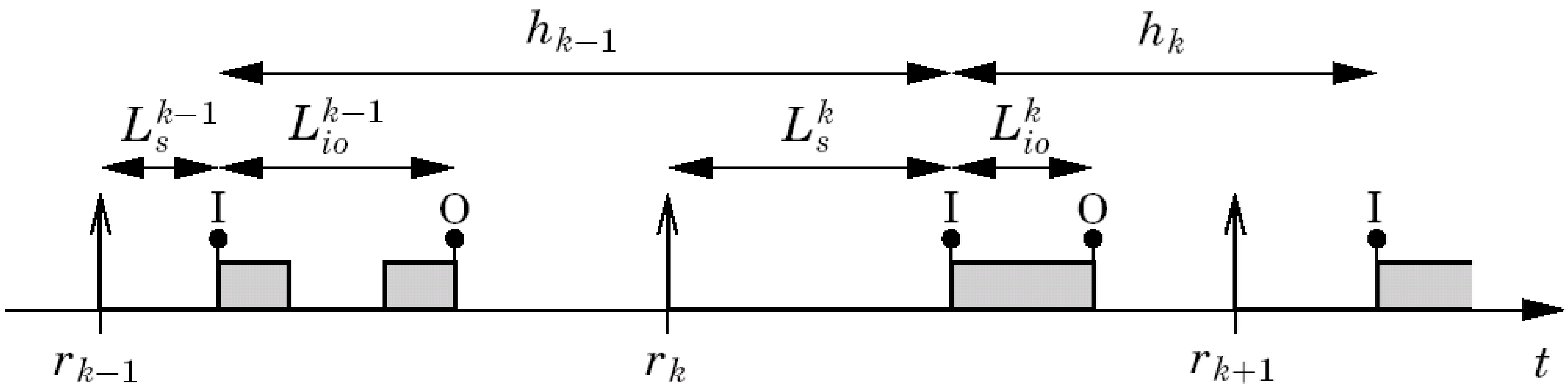


Logical concurrency



Time sharing concurrency

Networked Embedded Control Timing



- Embedded control often implies temporal non-determinism
  - resource sharing
  - preemptions, blocking, varying computation times, non-deterministic kernel primitives, …
- Networked control often implies temporal non-determinism
  - network interface delay, queuing delay, transmission delay, propagation delay, link layer resending delay, transport layer ACK delay, ...
  - lost packets

Vilka kurser finns?

Kurser för D

Årskurs 3

HT LP1	HT LP2	VT LP1	VT LP2
Reglerteknik AK			

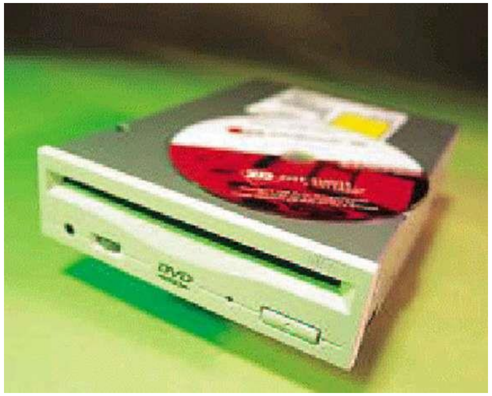
Årskurs 4-5

HT LP1	HT LP2	VT LP1	VT LP2
Flervariabel Reglering		Olinjär reglering, Systemidentifiering	Projekt i reglerteknik, Marknadsstyrda system
Realtidssystem, Prediktiv Reglering			

Specialiseringar:

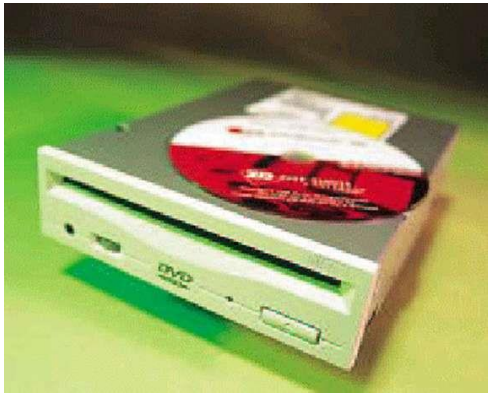
- Signaler, system och reglering
- Inbyggda system

Våra Kurser

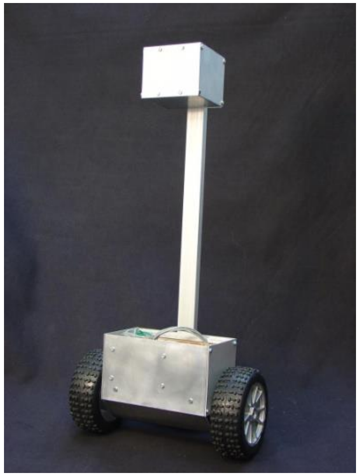


- **Flervariabel reglering**
  - Fortsättning på AK
  - Flervariabla system, designmetoder, optimering, begränsningar, …

Våra Kurser



- **Flervariabel reglering**
  - Fortsättning på AK
  - Flervariabla system, designmetoder, optimering, begränsningar, …



- **Realtidssystem**
  - Inbyggda system
  - Samplade reglersystem
  - Implementering av datorbaserade reglersystem
  - System med tidskrav
  - Projekt (PC/Java AVR/C)

Våra Kurser

- **Olinjär Reglering**
  - Självsvängningar, friktion, mättningar,
  - Olinjära designmetoder
- **Systemidentifiering**
  - Matematiska modeller från mätdata
  - Teknik, ekonomi, medicin, samhälle, miljö,..
  - Projekt

Våra kurser

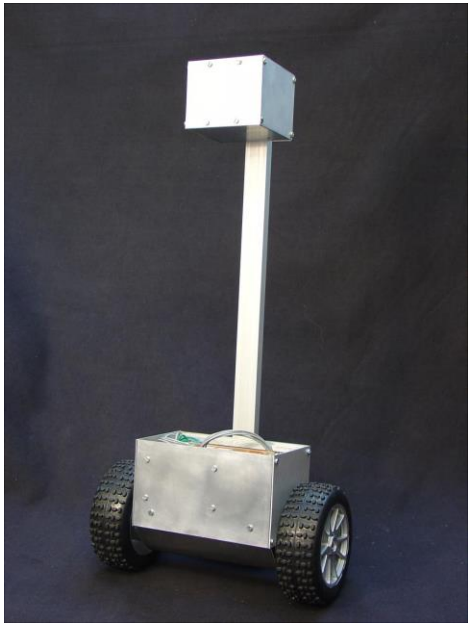
- **Prediktiv Reglering**
  - Regulatorer som automatiskt anpassar sig till ändrade förhållanden
  - Okända eller tidsvariabla system
  - Projekt
- **Projektkurs i Reglerteknik**
  - Ren projektkurs / många projektgrupper
  - 2005 och 2006: Mobila robotar – sensornätverk – trådlös kommunikation
  - 2009: Lego Mindstorm + Modelica
  - 2011: Quadcopter, android,…
  - Begränsat deltagarantal

- **Internationell Projektkurs**
  - Grupparbete
  - Gemensam större uppgift

  Lund + Kaiserslautern + företag

Hur kopplar det till andra kurser?

Mini-Segway



An Example of an Embedded Control System

Sensors & Actuators

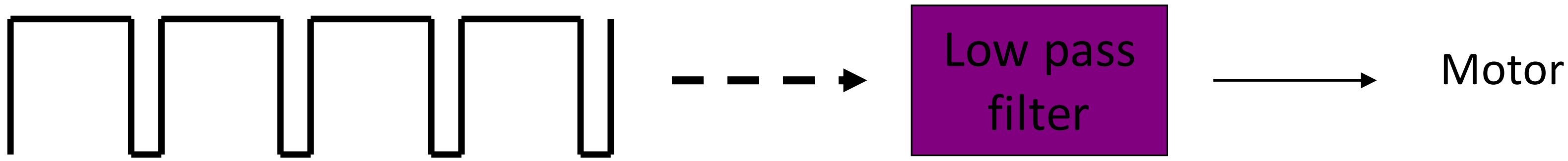
- Wheel encoders
- Accelerometer & Gyro
  - used together to measure the pendulum angle
  - accelerometer:
    - accurate static measurement of angle
    - does not give any useful information during acceleration of the Segway
  - gyro:
    - accurate dynamic measurement of angle velocity
    - integration to get angle
    - integrated value drifts over time due to measurement errors
  - complementary filter technique
    - block high-frequency parts of accelerometer signal
    - block low-frequency parts of the gyro signal
- Two DC motors

Processor

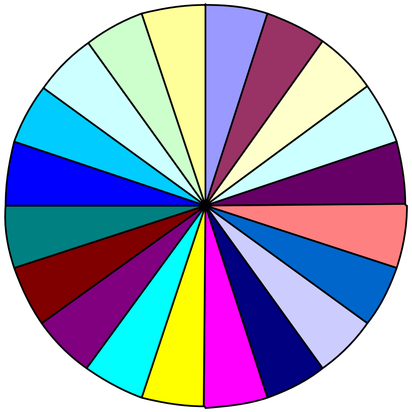
- ATMEL AVR Mega16
  - 16 kByte program memory
  - 2 kByte RAM
  - No hardware support for floating point
    - software emulation
  - AD converters
  - Digital outputs
  - RS 232 communication
  - No RTOS, no threads/tasks – only interrupts and timers
    - definutely no Java
  - Programmed in C with gcc compiler

IO interface

- Motors
  - Digital output
  - Pulse width modulation (PWM)



IO Interface



- Wheel encoders
  - optical sensor
  - bit counter that is incremented or decremented
  - interrupt generated when an 8-bit counter overflows
    - wheel position and velocity calculated
- Gyro
  - AD converter
  - 10 bits
- Accelerometer
  - sensor generates PWM signals
  - 16 bit counter to measure time intervals

Software Structure

- Event driven
- Interrupts:
  - Wheel encoders
  - AD converter
  - Timer interrupt for controller
  - Interrupts for sending and receiving over RS232 (UART)

Control Design

- State feedback controller
  - states: pendulum angle, angle velocity, wheel position, wheel velocity
  - velocities approximated using differences
- Design process:
  - model in continuous time (physics + experiment)
  - sampled to a discrete-time model (h = 16.4 ms)
  - design using LQ and poleplacement

Relaterade Kurser

- Matematik
  - Linjär algebra - Åk1VT2
    - Vektorer, matriser, linjära ekvationssystem, egenvärden
  - Tillämpad matematik ÅK2VT1, eller
    - Överföringsfunktioner, stegsvar, impulssvar, frekvensfuncktioner
  - Funktionsteori + System och Transformer
    - Djupare förståelse

Relaterade Kurser

- Programmering
  - Programmeringsteknik Åk1HT1-2
    - Grundläggande Java
  - Programmeringsteknik FK Åk1VT1
    - Objekt-orientering
  - Algoritmer, datastrukturer och komplexitet Åk2VT2
    - Datastrukturer
  - O-O modellering och diskreta strukturer Åk2HT1-2
    - Större o-o applikationer, UML, …
  - Realtidsprogrammering Åk3HT1-2
    - Trådprogrammering i Java (concurrent programming)
- Vill ni hålla på med inbyggda system så bör ni också läsa någon kurs i C eller C++

Relaterade Kurser

- Fysik Åk3HT2VT1
  - Matematiska modeller
- Datorer och digitalteknik
  - Digitalteknik Åk2HT1-2
    - Digitala kretsar, Boolesk algebra
  - Datorteknik Åk2VT1
    - Hur är en dator uppbyggd och hur fungerar den
- Elektronik Åk1HT2VT1-2
  - Analog elektronik, signaler och kretsar

Relaterade Kurser

- Kommunikation
  - Datorkommunikation, Åk2HT1
    - OSI, olika protokoll, …
- Kösystem Åk3VT2
  - Matematiska modeller av datorsystem
- Numerisk analys Åk3VT2
  - Integrationsmetoder, ODE, PDE
- Kognition Åk3HT2VT1
  - Människan i loopen, människan som regulator

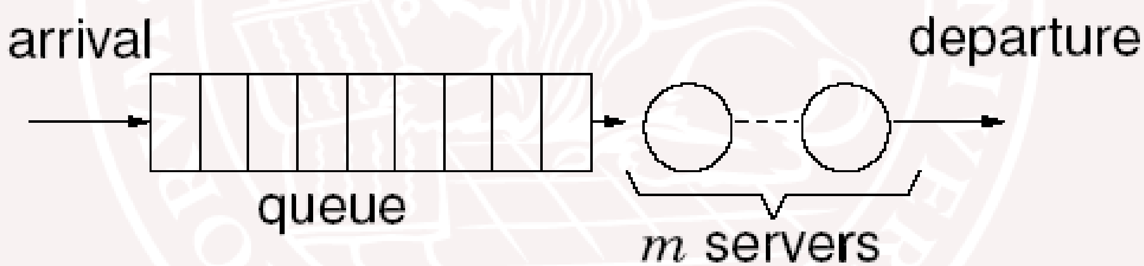
Two application examples

- Modeling and Control of Queueing systems
- EU-project ACTORS on Resource Allocation

Modeling of queueing-systems

- Discrete event models
- Queue theoretic model (Markov chains etc.)
- Flow models (cont. time / average models)
- Discrete time models

Queue models: A/B/m



A: Arrival process,   B: Service process,   m: number of servers

D (deterministic) M (exponential distr. - Poisson arrivals), GI (General indep) etc.

Modeling aspects

”Gain-scheduling”: (standard control principle):

- ”Choose among different control-parameters depending on *e.g.,* operating condition”.

- ”Good” model structure of corresponding computing system may change with work load (e.g., for server systems)
  - Flow models OK for high loads
  - DEDS-models feasible för low loads
    - Interpolation between different model structures?!
    - Transient vs steady-state behavior

Actuator Mechanisms

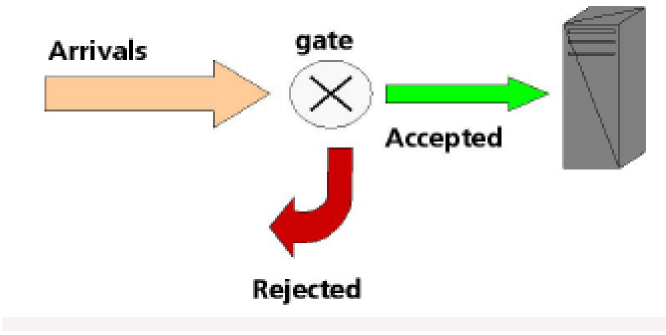
- *The difference between the service rate, μ, and the arrival rate, λ, determines the delay experienced by the requests.*

- Enqueue actuators: (*Changing the arrival rate)*
  - *Admission control mechanism*
  - *Change inter-arrival period of task ”upstream” in multitiered system*

- Dequeue actuator: *Changing the service rate:*
  - *Number of server threads*
  - *Quality adaptation*
  - *Dynamic voltage scaling*

Actuators - Implementation aspects

- Gate model:
  - *Call gapping* — accept first *u*(*kh*) calls in control interval
  - *Percent blocking* — **preserves** distribution



Crusial not to trigger network resend (lost package),
-"cmp Heracles and the hydra" [Sha ?]

Related reseach areas

Similarities/differences

of the different domains

- Traffic flow control
- Manufacturing and supply chains
- Communication networks
- Power networks

with respect to

- Where does the congestion appear?
- Routing?
- Available information (dest.)?
- Time/distance matters?
- ”Package dropping” – OK or not?
- Control action?

Example: Highway congestion in LA [Varaiya *et. al.*]

This image cannot currently be displayed.

Animation

Control of server systems

- Temporal control *locally* at server
  - Direct or ”indirect” objective (service provider vs. customer)

- Queue-management and load balancing
- Inherent nonlinearities
- Multi-tiered systems including large eCommerce systems

Example: Admission control

Objective:

- Good transient behavior for traffic changes
- Preserve good performance for overload situations

Measure of admission

- queue length
- average time
- **utilization**
- CPU load / energy consumption
- memory
- …

Adaptive Resource Management in ACTORS

Mikael Lindberg,
Vanessa Romero Segovia,
Anton Cervin, Karl-Erik Årzén

ACTORS:
Adaptivity & Control of
Resources in Embedded Systems

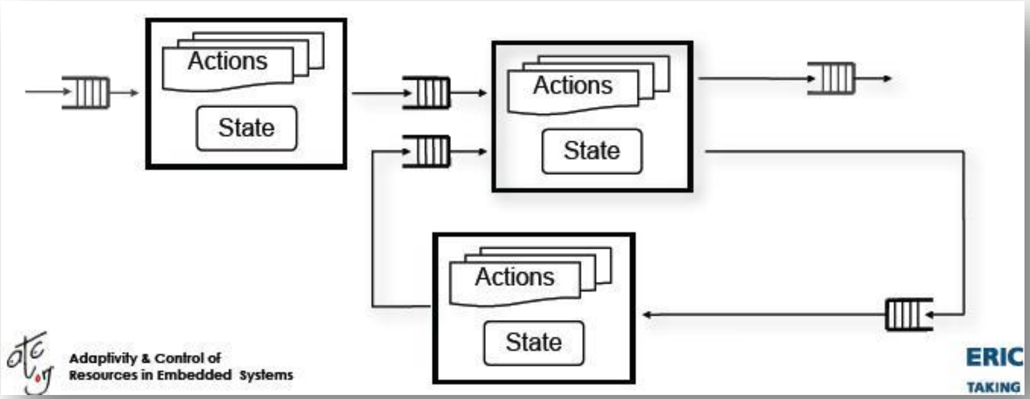
LUND
UNIVERSITY

Cellular Phones Tomorrow

- Multimedia streaming and processing increasingly important
  - Multiple simultaneous streams
- Large dynamic variations in use cases and QoS demands
  - Dynamic adaptation necessary
- Multicore architectures
  - Today in laptops and desktops, tomorrow in high-range embedded systems
  - Performance and power consumption reasons
  - Major challenge for hard real-time community
    - E.g. Worst-case timing analysis very conservative (breaks down)
- More advanced processors
  - E.g. ARM11
  - Powerful and complex instruction sets
  - Generation of efficient code an even higher challenge than today
- Open operating systems

ACTORS: Key Ingredients

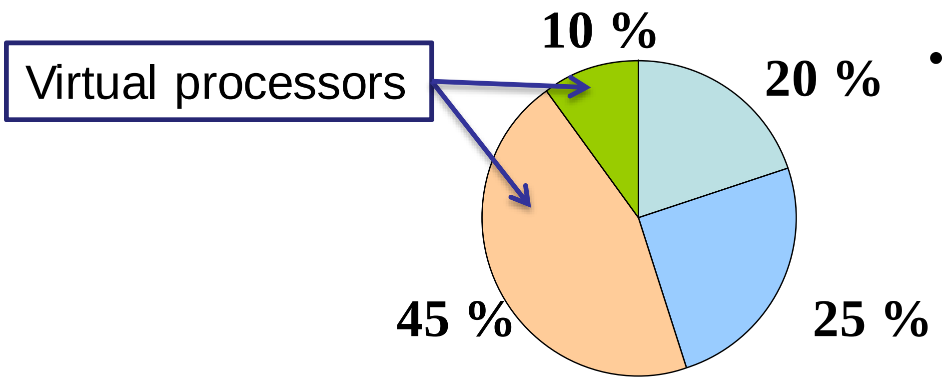
1. Data-Flow Programming in CAL



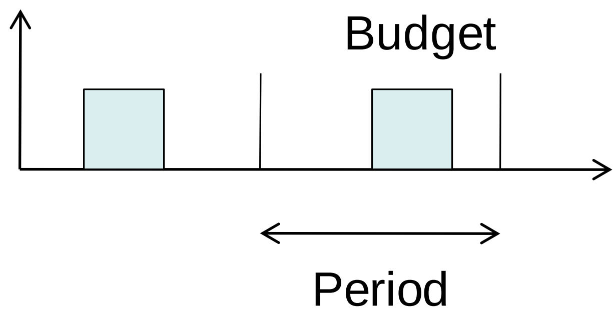
2. Feedback-Based Resource Management
   – Control how much CPU resources that are allocated to different applications based on feedback from resource utilization and achieved QoS

ACTORS: Key Ingredients

3. Reservation-Based Scheduling



- Periodic Bandwidth Servers
  - Constant Bandwidth Server



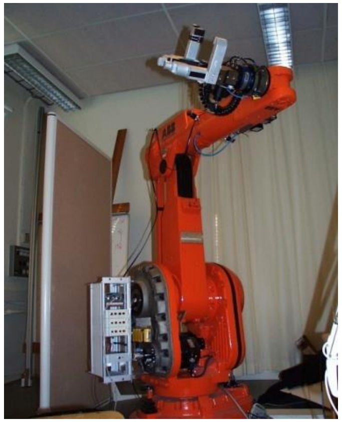
4. Multicore Linux Platforms
   – ARM 11

ACTORS Platforms

- ARM 11 multicore card with “new” Linux
  - SCHED_EDF scheduling class
    - Partitioned EDF scheduler
    - Support for hard partitioned CBS servers
      - Virtual processors
  - Linux CFS (Completely Fair Scheduler)
    - Limited support for reservations
- FPGA
- ARM multicore + FPGA
  - HW/SW partitioning

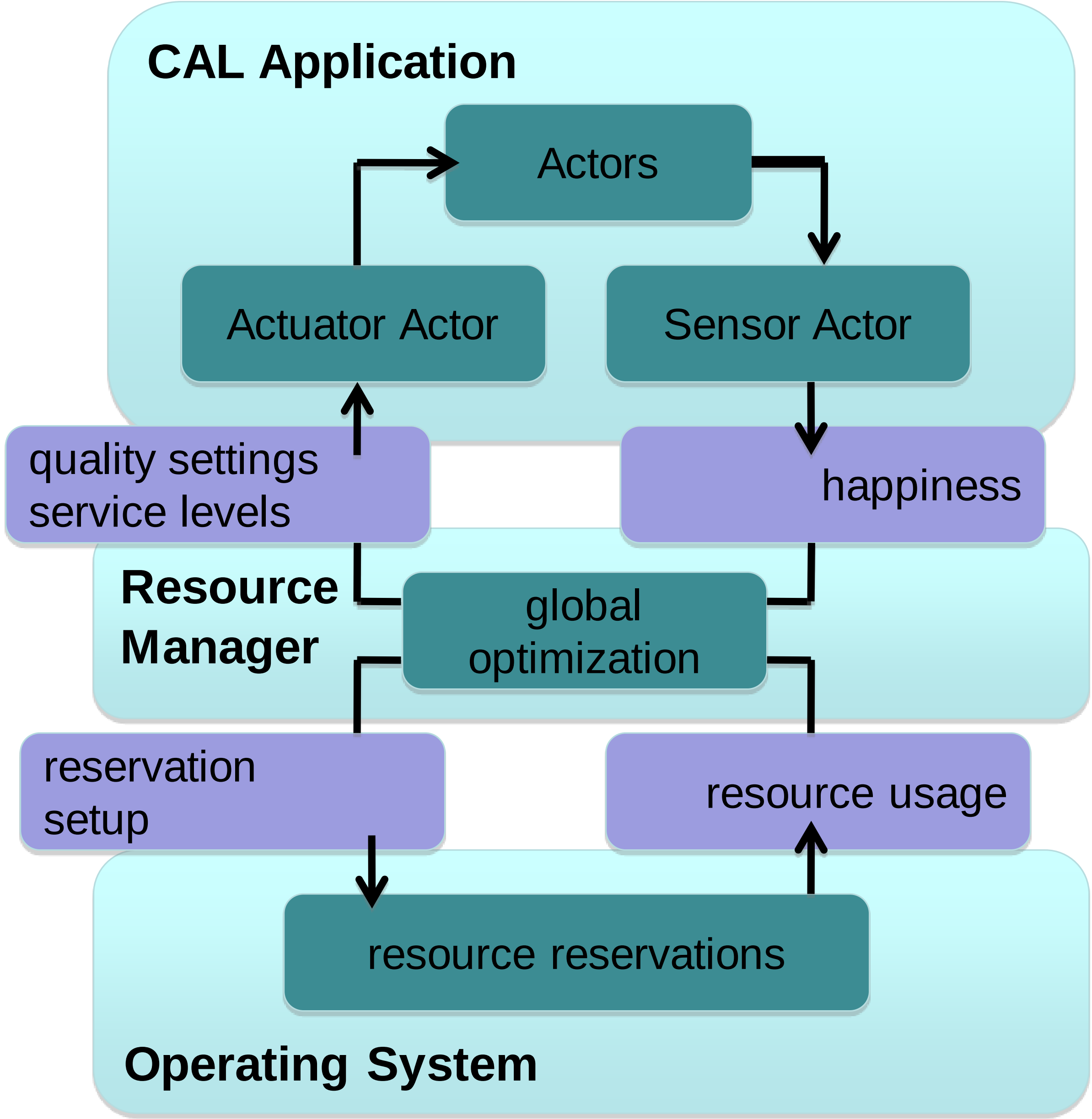
Three Demonstrators

- Media Streaming in Mobile Terminals
  - ARM11 / Linux
  - Conversational video
- Feedback Control
  - ARM11 / Linux
  - Dart-catching robot
- High-Performance Video
  - FPGA + ARM 11



Cast

- Ericsson
  - Coordinator (Johan Eker)
- Lund University
- TU Kaiserslautern
- Scuola Superiore Sant’Anna di Pisa
- EPFL - Lausanne
- AKAtech
  - Swiss SME in high-performance video
- Evidence
  - Italian SME in embedded hardware
- Xilinx US
  - Associated Partner


CAL Application
Actors
Actuator Actor
Sensor Actor
quality settings
service levels
happiness
Resource
Manager
global
optimization
reservation
setup
resource usage
resource reservations
Operating System

Välkomna till Reglerteknik i september 2016!!

Listen to Yoda!