Real-Time Systems

Course Introduction

Karl-Erik Årzén & Martina Maggio

17 January 2017

Lund University, Department of Automatic Control

www.control.lth.se/course/FRTN01

Real-Time Systems

A *real-time system* is a computer system that has to respond to externally generated events or inputs within a finite and specified time period

All control systems are real-time systems

Most real-time systems are *embedded systems*, i.e, the computer is an embedded, integrated part of some equipment or machinery

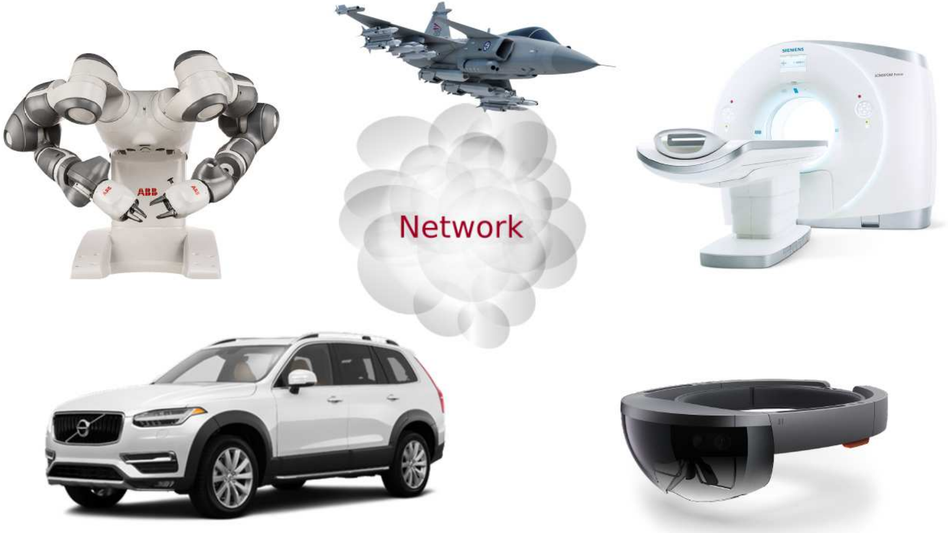
Embedded Systems

Embedded systems are by far the largest computer sector by volume

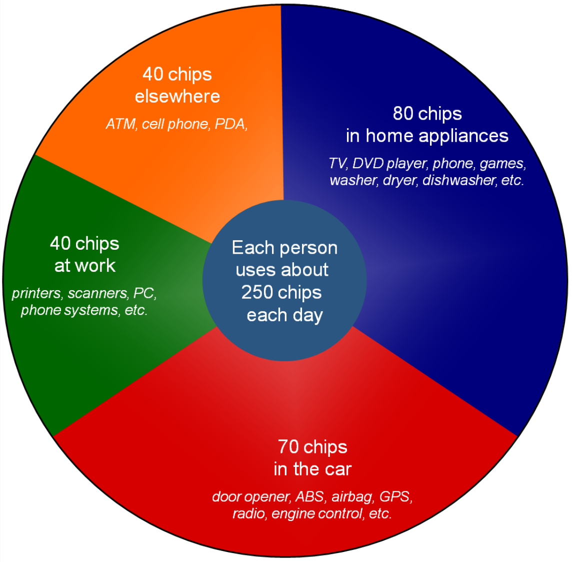
A large part of all embedded systems are control systems with hard/soft real-time constraints

- Vehicles
- Telecom
- Process & manufacturing industry
- Intelligent buildings
- ...

Application Examples



Embedded Systems

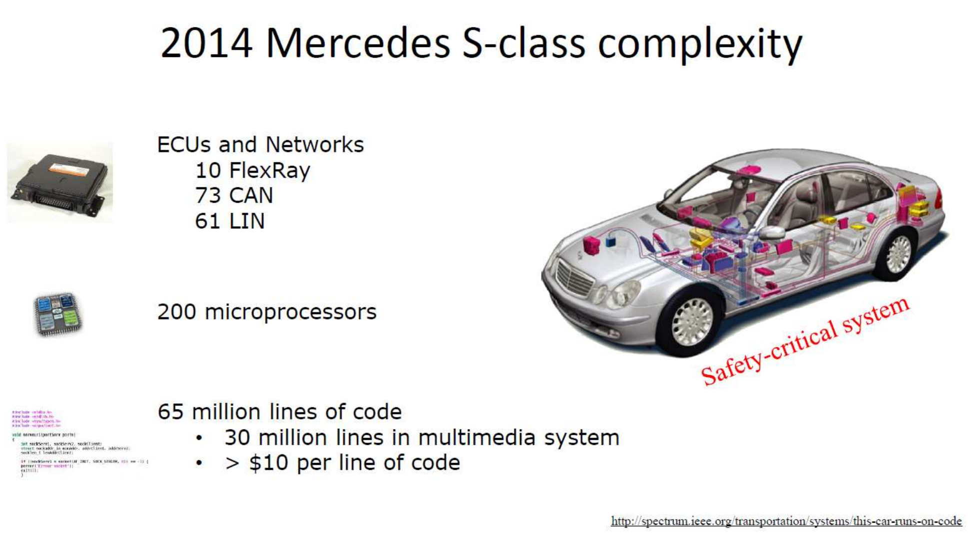


[Illustration courtesy of J. Sifakis]

Example - Car Industry

- A Volvo S80 contains $>$ 50 computers (ECUs) and several communication networks
- Most of them for various control applications
- 25–30% of the price
- Software the largest part of the cost
- Strong connections between control and software
    - e.g., climate control system: 25,000 lines of C code

Example - Car Industry 2



2014 Mercedes S-class complexity

ECUs and Networks
    10 FlexRay
    73 CAN
    61 LIN

200 microprocessors

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>

void serveur(portServ port)
{
    int sockServ1, sockServ2, sockClient;
    struct sockaddr_in monAddr, addrClient, addrServ2;
    socklen_t lenAddrClient;

    if ((sockServ1 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }
```

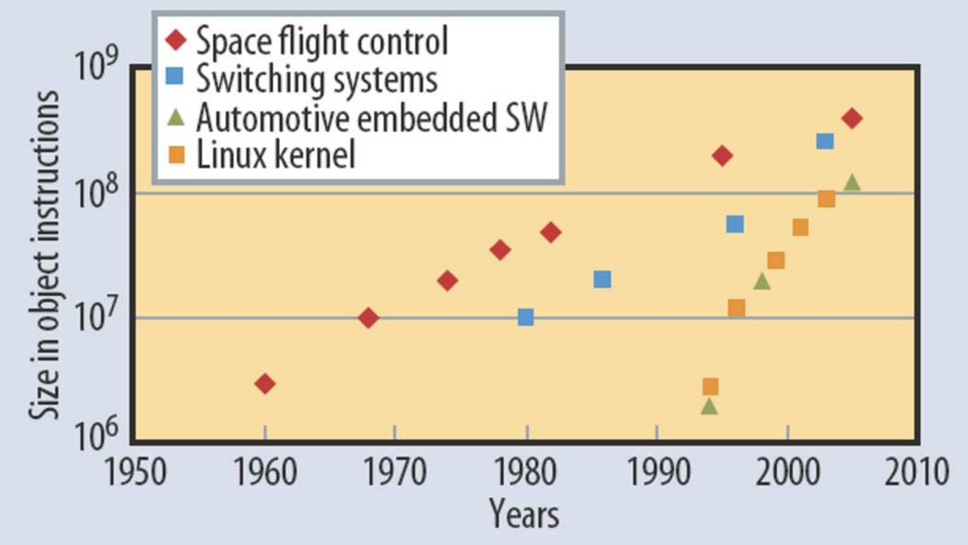
65 million lines of code

- 30 million lines in multimedia system
- > $10 per line of code

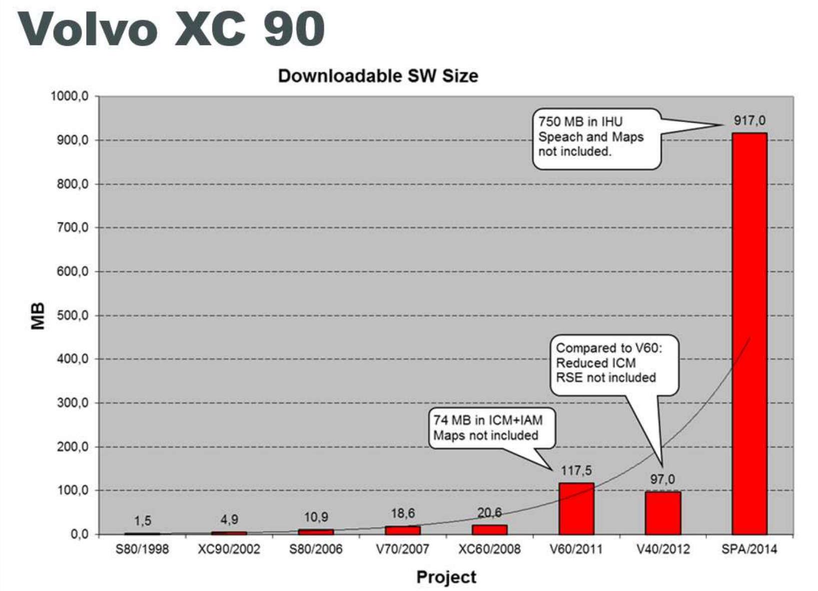
Safety-critical system

http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code

Software Size

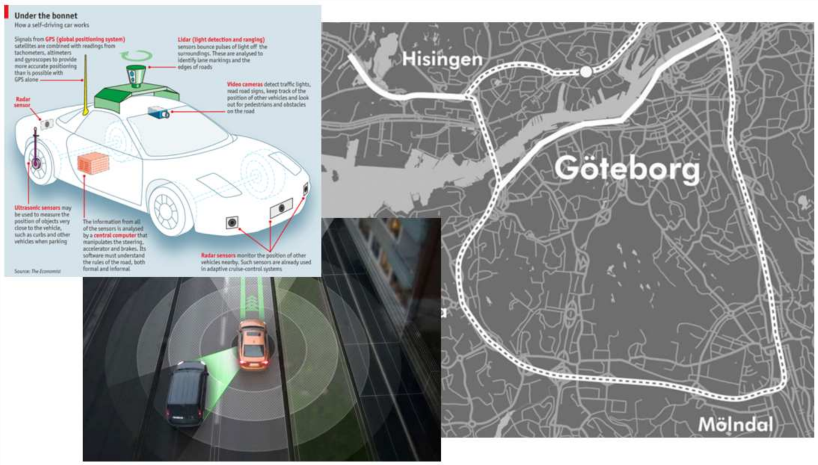


Software Size - Car Industry



Volvo XC 90

Autonomous Cars



Under the bonnet

How a self-driving car works

Signals from **GPS (global positioning system)** satellites are combined with readings from tachometers, altimeters and gyroscopes to provide more accurate positioning than is possible with GPS alone

Lidar (light detection and ranging) sensors bounce pulses of light off the surroundings. These are analysed to identify lane markings and the edges of roads

Video cameras detect traffic lights, read road signs, keep track of the position of other vehicles and look out for pedestrians and obstacles on the road

Radar sensor

Ultrasonic sensors may be used to measure the position of objects very close to the vehicle, such as curbs and other vehicles when parking

The information from all of the sensors is analysed by a **central computer** that manipulates the steering, accelerator and brakes. Its software must understand the rules of the road, both formal and informal

Radar sensors monitor the position of other vehicles nearby. Such sensors are already used in adaptive cruise-control systems

Source: *The Economist*

In 2017 Volvo will start running self-driving cars around Gothenburg

Lines of Code Comparisons

Pacemaker	80k
Space Shuttle	400k
Windows 3.1 (1992)	2.5M
US military drone	3.5M
Mars Rover	5M
Google Chrome	6M
Firefox	9M
Android	12M
Boeing 787	14M
Linux 3.1	15M
F-35 fighter jet	24M
Microsoft Office 2013	44M
Facebook	61M
High-end car	100M
Mouse genome	120M
Human genome	3,300B

Source:

www.informationisbeautiful.net/visualization/million-lines-of-code/

1M LOC = 18,000 pages of printed text

Example - Process Automation

- “Industrial IT” buzzword used by ABB
- Integration of automation and IT
    - software, distributed systems, WWW, e-commerce
- Focus on software rather than hardware

Real-Time Systems in Sweden

Real-Time and Embedded Systems have a very strong position in Sweden and in Lund

Research:

- LUCAS: Center for Applied Software Research at LTH
    - Computer Science and Automatic Control
- EASE: Industrial Excellence Center for Embedded Applications Software Engineering
- ELLIIT: The Lund-Linköping Initiative on IT and Mobile Communications
- WASP: Wallenberg Autonomous Systems and Software Program

Industry:

- embedded systems and embedded control systems of vital importance to Swedish industry (Ericsson, ABB, Volvo, Scania, SAAB, ...)

Course Aims

Study methods for design and implementation of computer control systems.

Focused on embedded control systems.

Two parts:

1. Real-time programming
2. Design and Implementation of Digital Control Systems

Programming Languages

Java as the main programming language.

However, not a Java course.

We assume basic knowledge of

- Java
- object-oriented programming concepts

Code examples written Modula 2 (very similar to C, Pascal) will be shown.

One laboratory session and some of the projects will use C

Relation to EDA040 Concurrent Programming

The students who have taken the Concurrent/Real-Time Programming course at Computer Science will recognize some parts of the first lectures

During the lectures we will also describe how real-time programming is performed with a conventional real-time programming language (Modula-2) and how a conventional real-time kernel (Stork) is implemented. (You do not have to program in Modula-2)

Deeper understanding and repetition

Students who have taken the Concurrent Programming course will do a special version of Lab 1 in which LJRT is used

Students that have taken the Concurrent Programming course must do a control-oriented project.

Staff

Karl-Erik Årzén
Course responsible
and lecturer

karlerik@control.lth.se
046-222 87 82

Martina Maggio
Course responsible
and lecturer

martina@control.lth.se
046-222 87 77

Mika Nishimura
Course administrator

mika@control.lth.se
046-222 87 85

Staff

Tommi Nylander

Teaching assistant

tommi@control.lth.se

Martin Heyden

Teaching assistant

martin.heyden@control.lth.se

Victor Millnert

Teaching assistant

victor@control.lth.se

Lectures

Lecture	Date	Time	Room	Topic	Lecturer
L1	Jan 17	8-10	M:E	Introduction	Both
LX	Jan 18	15-17	M:2112b	Extra: Introduction to Java	Martina
L2	Jan 19	10-12	M:D	Concurrent programming	Martina
L3	Jan 20	8-10	M:E	Process communication 1	Martina
L4	Jan 24	8-10	M:E	Process communication 2	Martina
L5	Jan 25	8-10	E:C	Interrupts and time	Martina
L6	Jan 27	8-10	M:E	Sampling of linear systems	Karl-Erik
L7	Jan 31	8-10	M:E	Input-output models	Karl-Erik
L8	Feb 2	10-12	M:E	Approx. of analog controllers, PID	Karl-Erik
L9	Feb 7	8-10	M:E	State feedback and observers	Karl-Erik
L10	Feb 9	10-12	M:E	Feedforward design	Karl-Erik
L11	Feb 14	8-10	M:E	Implementation aspects	Karl-Erik
L12	Feb 16	10-12	M:E	Scheduling theory	Martina
L13	Feb 21	8-10	M:E	Project specifications	Both
L14	Feb 28	8-10	M:E	Discrete-event control	Karl-Erik
L15	Mar 23	15-17	M:E	Real-time networks	Karl-Erik
L16	Mar 30	15-17	M:E	Hot research topics	Karl-Erik
LY	TBD	TBD	TBD	Extra: Repetition lecture	Both
L17	May 16	15-17	M:E	Project demos & oral presentations	-

Exercises

- Five computer exercises (C1–C5)
    - Weeks 2,3,4,5,6 in LP3
    - Tuesdays 10-12, 13-15, and 15-17
- One extra Java exercise (C0)
    - Week 1 in LP3
    - Friday Jan 20, 10-12
- Six problem-solving exercises (P1–P6)
    - Weeks 3,4,5,6 in LP3 and weeks 1,2 in LP4
    - Wed 8-10, Thu 8-10, Fri 10-12 in LP3
    - Tue 10-12, Thu 8-10 in LP4
- One extra Matlab exercise (P0)
    - Friday Jan 27, 10-12

All exercises are held in Department of Automatic Control, Lab A

Exercise Groups: Study Period 3

Computer Exercises:	**Teaching Assistant**
Tuesdays 10-12	Martin Heyden
Tuesdays 13-15	Victor Millnert
Tuesdays 15-17	Tommi Nylander

Problem-Solving Exercises:	**Teaching Assistant**
Wednesdays 8-10	Victor Millnert
Thursdays 8-10	Tommi Nylander
Fridays 10-12	Martin Heyden

In order to balance the load on the exercise groups you must register for the group that you would like to follow. Done via the course home page earliest on Thursday 19 January

Exercise Schedule

Exercise	Dates	Topic
C0	Jan 20	Extra: Introduction to Java
C1	Jan 24	Threads
P0	Jan 27	Extra: Control in Matlab
C2	Jan 31	Synchronization
P1	Feb 1-3	Sampling of systems
C3	Feb 7	Controller implementation
P2	Feb 8-10	Input-output models
C4	Feb 14	Graphical user interface
P3	Feb 15-17	State feedback and observers
C5	Feb 21	Prepare Lab 1
P4	Feb 22-24	Discrete approximation, PID
P5	Mar 21-23	Fixed-point implementation
P6	Mar 28-30	Scheduling theory

Laboratory Sessions

- Three manadatory laboratory sessions, 4 hours each
- The preparatory assigments will be checked at the beginning of each lab
- Room: Department of Automatic Control Lab A

Lab	**Approx. dates**	**Topic**	**Sign-up**	**Responsible**
Lab 1	Feb 22–Mar 3	Control of ball and beam	TBD	Victor Millnert
Lab 2	Mar 6–10	Sequence control of bead sorter	TBD	Martin Heyden
Lab 3	Mar 20–31	Embedded control of rotating servo	TBD	Tommi Nylander

Lab 1

Implementation of a control system for the ball & beam process

- Cascaded PID controllers
- Java or Java/LJRT with Swing-based GUI
- Prepared during the computer exercises

Lab 2

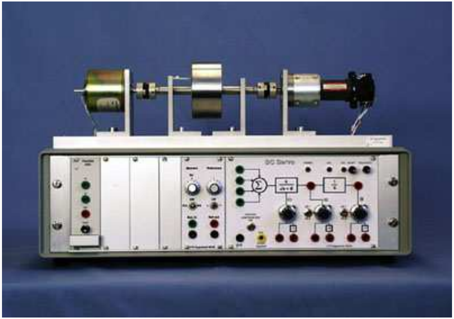
Sequence control of a bead-sorter process

- Discrete-event controller
- JGrafchart – a Java-based Grafcet editor and run-time system

Lab 3

Fixed-point implementation of a DC-servo controller

- State feedback controllers
- C on ATMEL AVR Mega16



Project

Projects are performed as team works with four persons per team (in special cases it is OK with smaller project teams).

Around 30 different projects to chose among:

- control of ball and beam process
- control of inverted pendulum
- control of helicopter process
- real-time kernel projects
- embedded system projects using ATMEL AVR and C
- Lego Mindstorm NXT projects
- etc.

If you are following the Predictive Control course it will be possible to do a joint project between the courses.

Students that have taken EDA040 Concurrent Programming course must do a control-oriented project

Project

Important dates:

- Feb 21, at Lecture 15: Presentation of available projects
- TBD: Deadline for team formation and project selection
- TBD: Deadline for suggested solution
- TBD: Deadline for project report (10–15 pages, English/Swedish)
- May 16, at Lecture 18: Project demos (mandatory)
- May 16: Oral presentations (mandatory)

Literature

- K.-E. Årzén, "Real-Time Control Systems", 2015. KFS.
- B. Wittenmark, K.J. Åström, K-E Årzén, "Computer-Control: An Overview", Educational version 2016. KFS.
- "Real-Time Systems – Problem Solving Exercises", 2015. KFS.
- "Real-Time Systems Formula Sheet". Online.

The 2014 versions are very similar and also possible to use.

Examination

Mandatory parts: Three laboratory sessions, project, written exam (5 hours).

The exam consists of 25 points and gives the grade Fail, 3, 4, or 5.

Accepted aid: The textbooks “Real-Time Control Systems” and “Computer Control: An Overview ”, standard mathematical tables and authorized “Real-Time Systems Formula Sheet”; pocket calculator.

Two exam opportunities in the spring:

- Apr 21 at 14-19
- Jun 3 at 8-13

Course History

-71-72 Control of LKAB iron ore crusher over modem, PDP 15

-73 “Computers in Control Systems”, PDP 15, assembler

-79 “Computers in Control Systems 2”, LSI-11, Concurrent Pascal

-81 Pascal + real-time kernel

-83 “Applied Real-Time Programming”, IBM PC, Modula 2

-86-87 CS course on real-time programming. Focus on robotics.

-89 “Computer Implementation of Control Systems”, VME 68020

-93 “Real-Time Systems”. CS course no longer a prerequisite.

-96 Windows NT, Pentium, InTouch

-98 PowerPC, Migration to Java started

-00 Java, Linux, PC

-03 ATMEL AVR microprocessors introduced

-07 More focus on digital control and embedded systems