

An aerial photograph of a city, showing buildings, streets, and green spaces, with a semi-transparent overlay. The text is centered over the image.

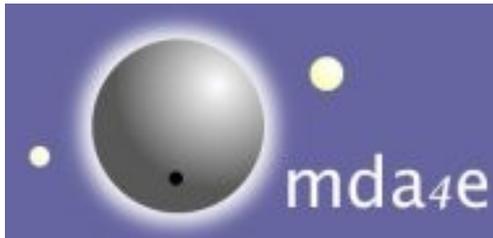
# **Eclipse**

für die kleinen Dinge  
des (modernen) Lebens

# Agenda

- Vorstellung
- Zielsysteme
- Software
- Modellierung
- Übertragung zum Gerät

# Vorstellung



[www.mda4e.org](http://www.mda4e.org)



EUROPÄISCHE UNION

Europäischer Fonds  
für Regionale Entwicklung



**Fachhochschule  
Dortmund**

University of Applied Sciences

**Ingenieurbüro Dr. Kahlert**  
Software-Engineering & Automatisierungstechnik

# Projektpartner mda4e

- itemis GmbH & Co. KG  
Spezialist für MDSD Beratung
- FH Dortmund, FB Informations- und Elektrotechnik  
Prof. Dr. Burkhard Igel
- Ingenieurbüro Dr. Kahlert  
Hersteller für Entwicklungswerkzeuge für eingebettete Systeme

# Agenda

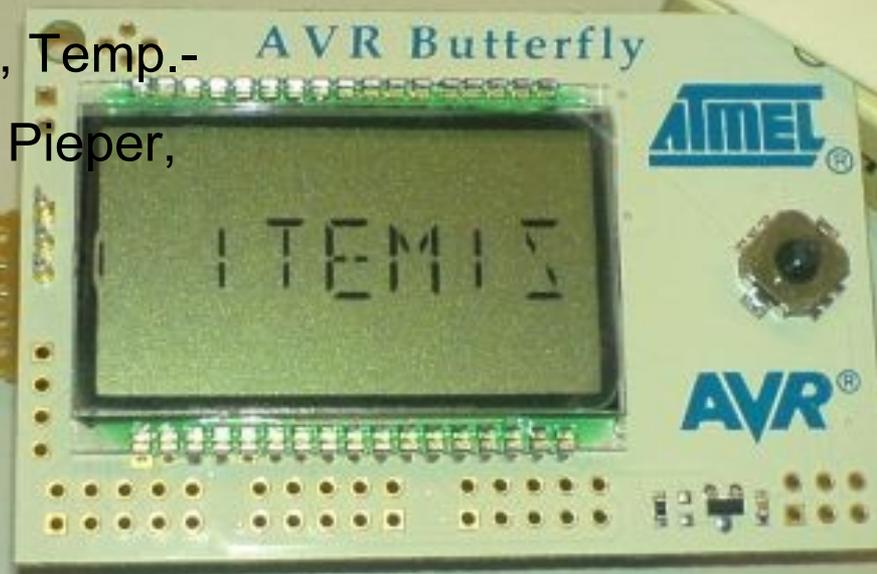
- Vorstellung
- Zielsysteme
  - AVR Butterfly
  - D071
  - STK500
  - Minimalsysteme
- Software
- Modellierung
- Übertragung zum Gerät

# Worum es nicht geht



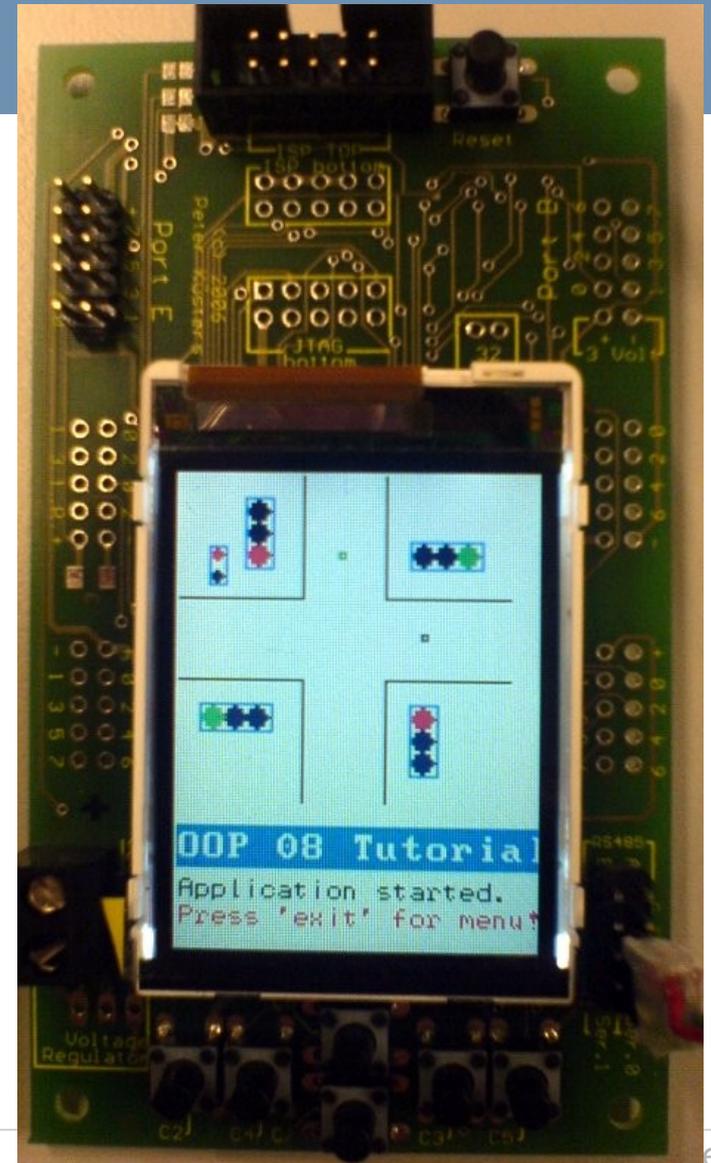
# AVR Butterfly

- Prozessor: ATmega169
- 16 kB FLASH-ROM
- 512 Byte RAM
- LCD, Lichtsensor, Temp.-  
Sensor, Joystick, Pieper,  
RS232



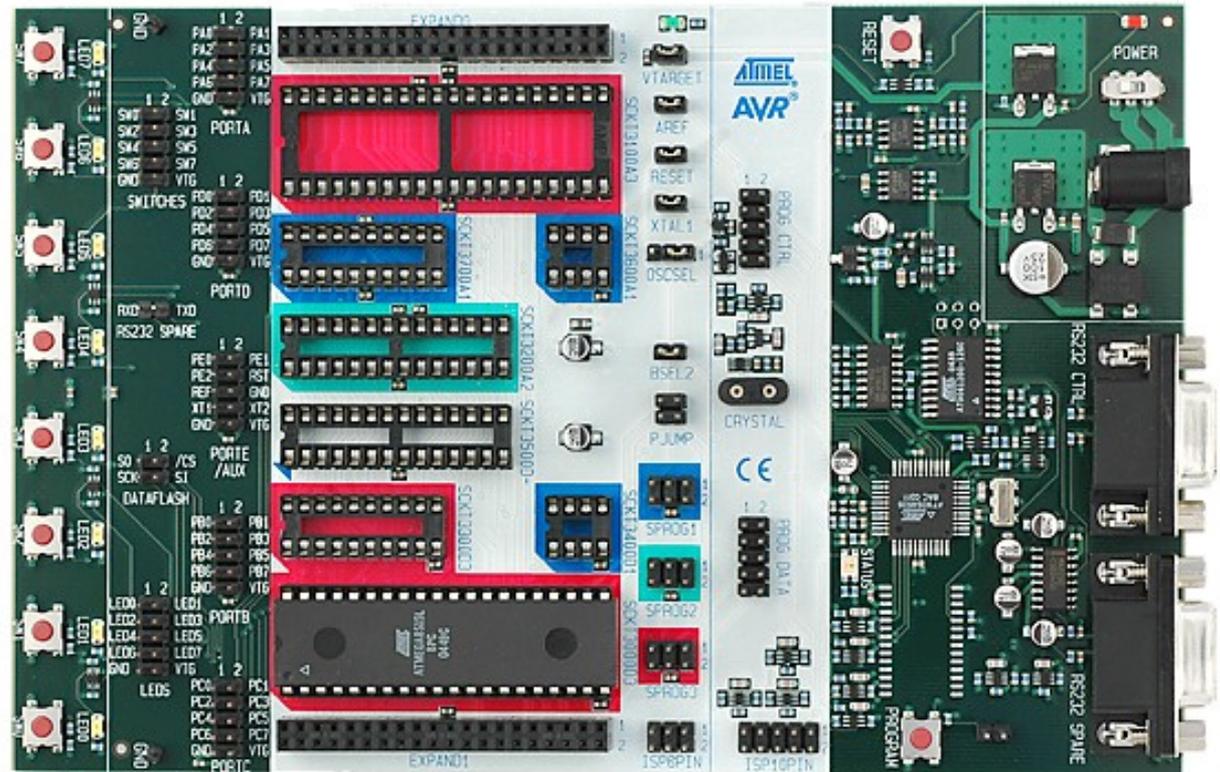
# Display 3000 D071

- Prozessor: ATmega128
- 128 kB FLASH-ROM
- 4kB RAM
- 2,1" TFT-Farbdisplay
- Taster
- herausgeführte I/O-Leitungen
- 2x RS-232
- Optional: CAN-Bus

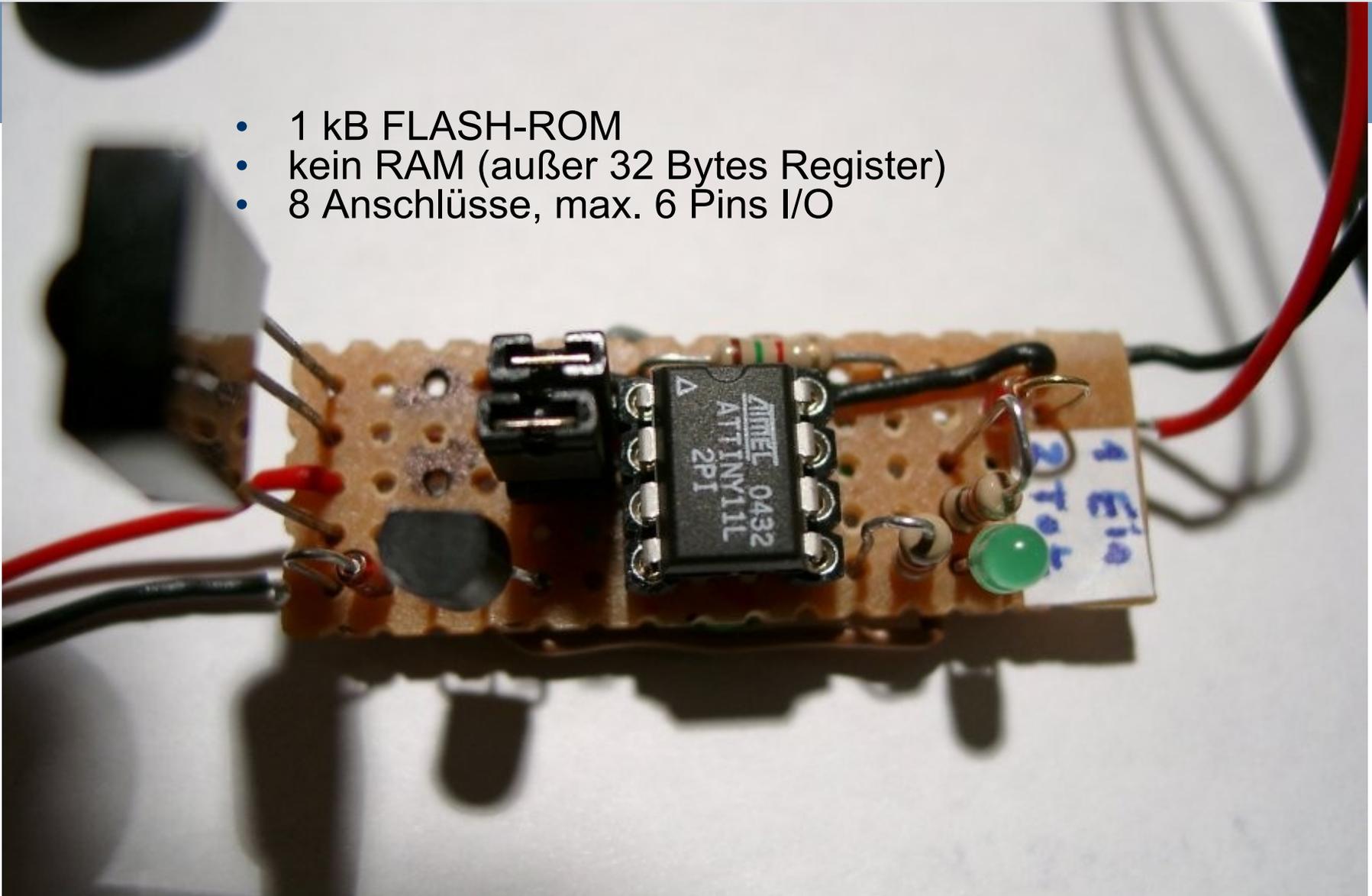


# STK 500

- Evaluationsboard für AVR-Mikrocontroller



- 1 kB FLASH-ROM
- kein RAM (außer 32 Bytes Register)
- 8 Anschlüsse, max. 6 Pins I/O



# Agenda

- Vorstellung
- Zielsysteme
- **Software**
- Modellierung
- Übertragung zum Gerät

# Software: Compiler und Zusatztools

- AVR-GCC Crosscompiler / Assembler
- AVR-LIBC: optimierte Libraries
- diverse Tools für Binärformat
- Simulavr – Simulator
- avr-gdb – Debugger
- AVRDude zum flashen
- Ferner: Echtzeit-Multitaskingbetriebssysteme (z.B. FreeRTOS, AvrX)

- CDT
- AVRDUde

The screenshot shows the Eclipse IDE interface. The top toolbar contains an AVR icon with a green arrow pointing to it. The Project Explorer on the left shows a tree view of the project structure, including 'include' and 'main.c'. The main editor window displays the source code for 'main.c', which includes various headers and defines. The bottom panel shows the 'Tasks' tab with 'AVRDUde Output' displayed, showing the progress of flashing the firmware to the AVR microcontroller.

```
#include <string.h>
2
3
4
5 #ifndef GCC_MEGA_AVR
6 /* EEPROM routines used only with the WinAVR compiler. */
7 #include <avr/eeprom.h>
8 #endif
9 /* Scheduler include files. */
10 #include "FreeRTOS.h"
11 #include "task.h"
12 #include "uiTask.h"
13 #include "glcd-Display3000-211.h"
14 #include "queue.h"
15
16 #include "main.h"
17 #include "serial.h"
18
19 #include "events.h"
20 #include "safeDemoTask.h"
21
22
23
```

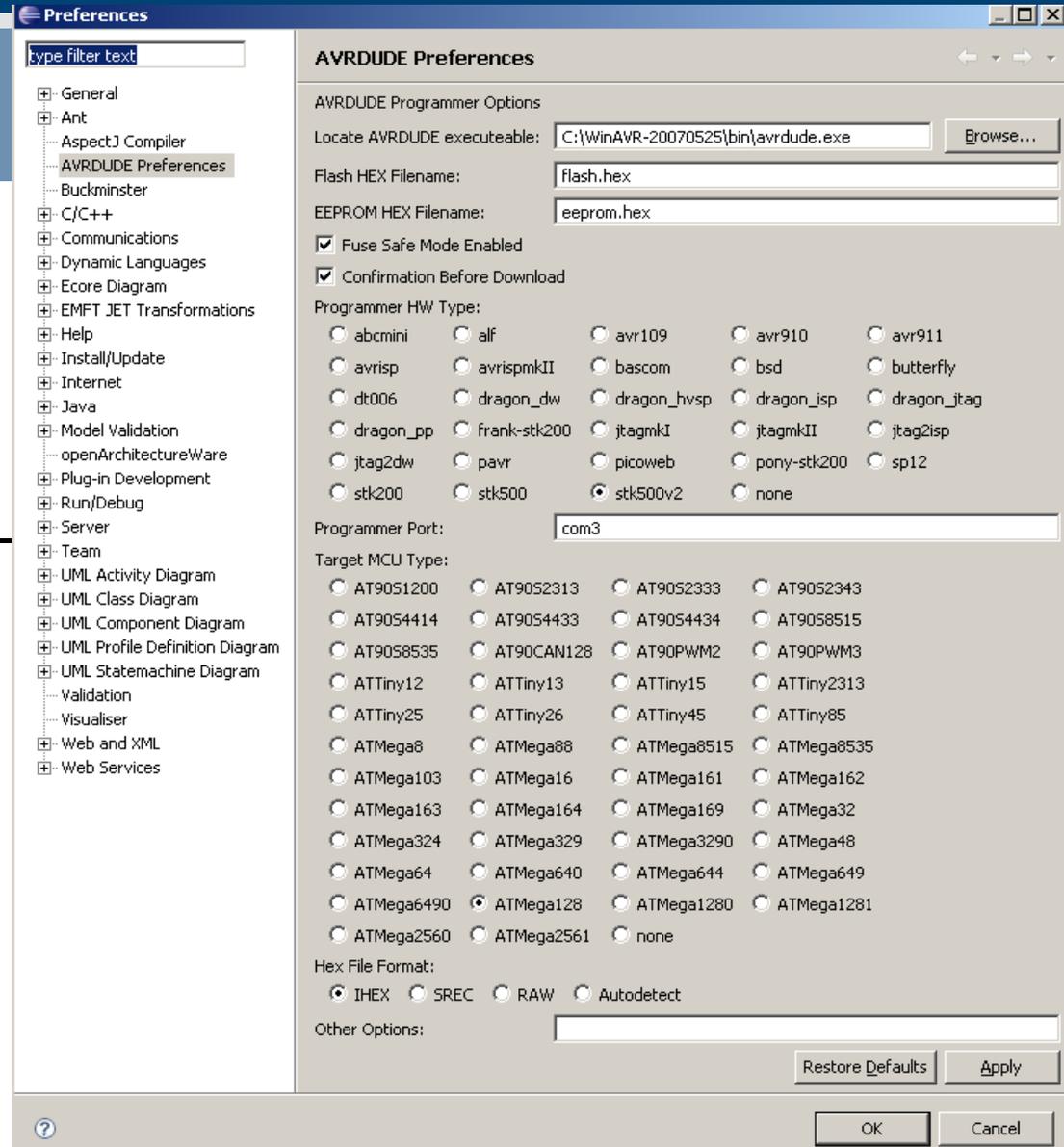
AVRDUDE Output

```
avrdude.exe: NOTE: FLASH memory has been specified, an erase cycle will be performed
    To disable this feature, specify the -D option.
avrdude.exe: stk500v2_cmd(): failed to send command
avrdude.exe: erasing chip
avrdude.exe: reading input file "C:\OOP2008\workspaces\test\D071.base\Safe\flash.hex"
avrdude.exe: input file C:\OOP2008\workspaces\test\D071.base\Safe\flash.hex auto detected as Intel He
avrdude.exe: writing flash (25624 bytes):

Writing | ##### | 100% 3.58s
```

# AVRDude

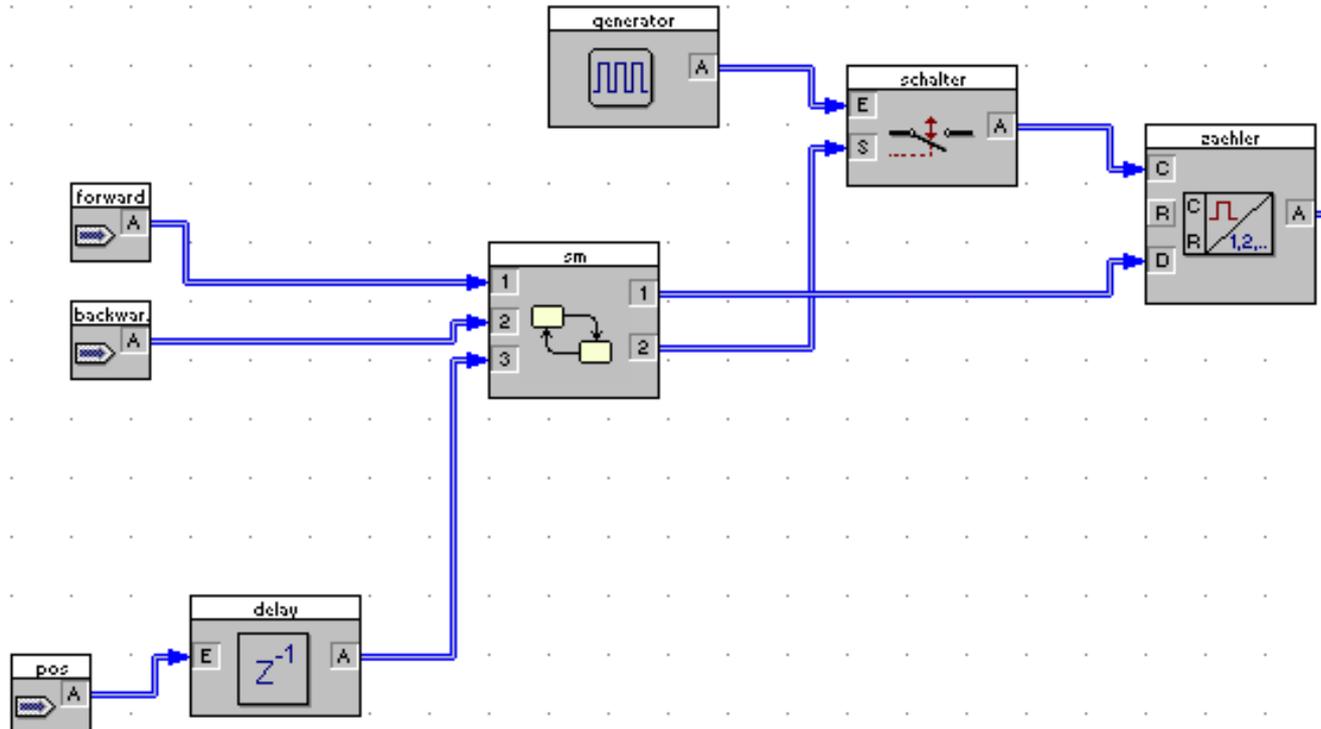
- Programmierung der Controller über den AVR-Button
- Einstellung der Parameter über Preferences



# Agenda

- Vorstellung
- Zielsysteme
- Software
- **Modellierung**
- Übertragung zum Gerät

# Beispiel: Blockschaltssysteme ("Boris")



Quelle: WinFACT/BORIS

# Statemachine mit TopCaseD-UML-Editor

The screenshot displays the Eclipse IDE interface for editing a State Machine (UML diagram) for a CD player. The main workspace shows a state machine diagram with the following elements:

- States:** Off, Stop, Play, Pause, TrayOpen, and a Pseudostate On.
- Transitions:**
  - Off to On (Power On)
  - On to Off (Power Off)
  - On to Stop (Start Transition ON)
  - Stop to Play (Play)
  - Play to Stop (Stop from Play)
  - Play to Pause (Pause)
  - Pause to Play (Continue)
  - Pause to Stop (Stop from Pause)
  - TrayOpen to Stop (Close Tray)
  - TrayOpen to Off (Open Tray)
  - On to On (Track Minus On State)
  - On to On (Track Plus in On State)
- Initial State:** On

The right sidebar (Outline) shows the following structure:

- <<initialState>> <<Pseudostate>> Start
- <<State>> Off
- <<State>> TrayOpen
- <<State>> On
- <<Activity>> checkCd
- <<Activity>> stop
- <<Region>>
  - <<Pseudostate>>
  - <<State>> Stop
  - <<State>> Play
  - <<State>> Pause
  - <<Pseudostate>> On History
  - <<Transition>> Start Transition ON
  - <<Transition>> Play
  - <<Activity>> play
    - <<Call Operation Action>> Call Play
    - <<Trigger>> Play
    - <<Transition>> Pause

The bottom of the IDE shows the Problems view (empty) and the Properties view for the selected state:

Property	Value
Info	
derived	false
editable	true
last modified	30.01.07 15:34
linked	false

# Generierung von Java und C mit oAW

The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer (Left):** Shows a project named 'javacode/src-gen' containing a package 'de.itemis'. Under 'de.itemis', there are sub-packages 'AbstractDevice' and 'DeviceActions'. The 'AbstractDevice' package contains files 'AbstractDevice.java', 'DeviceActions.java', 'DeviceEvents.java', and 'DeviceStates.java'. Below this, there are system libraries and test files.
- Main Editor (Center):** Displays the generated Java code for 'AbstractDevice.java'. The code includes:
 

```
private DeviceStates currentState = DeviceStates.OFF;
private boolean terminated = false;

public void handleEvent(DeviceEvents event) {
    if (terminated) {
        throw new RuntimeException("this sm is terminated!");
    }
    switch (currentState) {
    case OFF:
        if (event == DeviceEvents.POWERSWITCHPRESSED) {
            this.powerOn(); // call Method powerOn
            currentState = DeviceStates.STOP;
            break;
        }
        break;
    case TRAYOPEN:
        if (event == DeviceEvents.OPENCLOSEPRESSED) {
            this.closeTray(); // call Method closeTray
            currentState = DeviceStates.STOP;
            break;
        }
        break;
    case STOP:
        if (event == DeviceEvents.TRACKPLUS) {
            if (!this.isNotLastTrack()) {
                break; // Guard
            }
            this.track_plus(); // call Method track_plus

            // shallow history - no new state
            break;
        }
        if (event == DeviceEvents.TRACKMINUS) {
            if (!this.isNotFirstTrack()) {
                break; // Guard
            }
        }
        this.track_minus(); // call Method track_minus

        // shallow history - no new state
        break;
    }
    if (event == DeviceEvents.PLAYPRESSED) {
        this.play(); // call Method play
        currentState = DeviceStates.PLAY;
    }
}
```
- Right Editor:** Displays the generated C code for 'AbstractDevice.c'. The code includes:
 

```
#endif
char terminated = FALSE;
char actionCallStatus = 0; // 0 = OK

uint8_t handleEvent( unsigned char event ) {
    if ( terminated )
    {
        return 1;
    }
    switch ( currentState ) {
    case OFF:
        if ( event == POWERSWITCHPRESSED ) {
            actionCallStatus = powerOn(); // call Method p
            currentState = STOP;
            break;
        }
        break;
    case TRAYOPEN:
        if ( event == OPENCLOSEPRESSED ) {
            actionCallStatus = closeTray(); // call Method c
            currentState = STOP;
            break;
        }
        break;
    case STOP:
        if ( event == TRACKPLUS ) {
            if ( ! isNotLastTrack() ) break; // Guard
            actionCallStatus = track_plus(); // call Meth
            // shallow history - no new state
            break;
        }
        if ( event == TRACKMINUS ) {
            if ( ! isNotFirstTrack() ) break; // Guard
            actionCallStatus = track_minus(); // call Meth
            // shallow history - no new state
            break;
        }
        if ( event == PLAYPRESSED ) {
            actionCallStatus = play(); // call Method play
            currentState = PLAY;
            break;
        }
        if ( event == POWERSWITCHPRESSED ) {
            actionCallStatus = powerOff(); // call Method p
            currentState = OFF;
        }
    }
}
```

At the bottom right of the IDE, there are status indicators: 'Writable', 'Smart Insert', and '59:1'.

# Simulation (Java)

```
Problems Declaration Search Properties Console X
ConsoleCdPlayer [Java Application] /usr/lib/jvm/java-1.5.0-sun-1.5.0
Signal (0) OPENCLOSEPRESSED
Signal (1) POWERSWITCHPRESSED
Signal (2) PLAYPRESSED
Signal (3) PAUSEPRESSED
Signal (4) STOPPRESSED
Signal (5) TRACKPLUS
Signal (6) TRACKMINUS
exit: exit      help: this text
$> 1
-->Event POWERSWITCHPRESSED
** Powering on...
$> 0
-->Event OPENCLOSEPRESSED
** Opening Tray...
$> 0
-->Event OPENCLOSEPRESSED
** Closing Tray...
$> 2
-->Event PLAYPRESSED
** Playing track 1...
$>
```

# Agenda

- Vorstellung
- Zielsysteme
- Software
- Modellierung
- Übertragung zum Gerät
  - Simulation

# AVR Studio mit Display-Simulation

The screenshot displays the AVR Studio IDE interface during a simulation. The top menu bar includes File, Project, Build, Edit, View, Tools, Debug, Window, and Help. The toolbar contains various icons for file operations, simulation control, and debugging. The I/O View on the left shows a tree structure of hardware components, including the Processor (Program Counter, Stack Pointer, Cycle Counter, X-register, Y-register, Z-register, Frequency, Stop Watch) and I/O ATMEGA169 (AD\_CONVERTER, ANALOG\_COMPARATOR, BOOT\_LOAD, CPU, EEPROM, EXTERNAL\_INTERRUPT, JTAG, LCD, PORTA). The main editor window shows the source code for a simulated device, with the following C code:

```

char terminated = FALSE;
char actionCallStatus = 0; // 0 = OK

uint8_t handleEvent( unsigned char event ) {
    if ( terminated )
    {
        return 1;
    }

    switch ( currentState ) {
        case OFF:
            if ( event == POWERSWITCHPRESSED ) {

```

The Watch window at the bottom right shows the following data:

Name	Value	Type	Locat
input	2 '0'	char	R15
PowerSave	0 ''	char	0x01
gPowerSaveTimer	0 ''	uint8_t	0x01
g\$ECOND	20 '0'	uint8_t	0x01

The AVR LCD Visualizer at the bottom left shows a simulated AVR Butterfly board with a display displaying "PLAY 1". The status bar at the bottom indicates the current state: ATmega169, AVR Simulator, Auto, Running, Ln 262, Col 1, CAP, NUM, SCRL. The system tray shows the Start button, AVR Studio - \\Galeria..., LCD Properties, and the system clock at 15:09.

The screenshot displays the Eclipse IDE in a debug configuration for an AVR microcontroller. The main window shows the source code for `safeDemoTask.c` at line 161, where a static array of port characters is defined. The console window shows the execution of the command `var-evaluate-expression var72`, resulting in the value `80026`. The disassembly window shows the current instruction at memory address `0x0001389a`.

**Registers Window:**

Name	Value
r29	0
r30	0
r31	0
SREG	0
SP	0x00800000

**Disassembly Window:**

```

0x0001389a .word 0xffff ; ???
0x0001389c .word 0xffff ; ???
0x0001389e .word 0xffff ; ???
0x000138a0 .word 0xffff ; ???
0x000138a2 .word 0xffff ; ???
0x000138a4 .word 0xffff ; ???
0x000138a6 .word 0xffff ; ???
0x000138a8 .word 0xffff ; ???
0x000138aa .word 0xffff ; ???
0x000138ac .word 0xffff ; ???
0x000138ae .word 0xffff ; ???
0x000138b0 .word 0xffff ; ???
0x000138b2 .word 0xffff ; ???
0x000138b4 .word 0xffff ; ???
    
```

**Console Window:**

```

D071.base Standard [C/C++ Local Application] C:\WinAVR-20070525\bin\avr-gdb.exe (27.11.07 16:43)
(gdb)
232-var-evaluate-expression var72
232^done, value="80026"
(gdb)
    
```

# Übertragung zum Gerät

Verschiedene Möglichkeiten:

- ISP (In-System-Programmierung)
- Parallele Programmierung / High-Voltage
- JTAG (In-System-Programmierung, Debugging)
- Bootloader über beliebige Schnittstellen (seriell, parallel, USB, Netzwerk, CAN, Infrarot, ...)

# Bootloader

## Vorteil Bootloader:

- Software-Update liegt in der Hand des Programmierers
- Kein Programmiergerät notwendig
- “gefährliche” Änderungen (Fuses, Lock Bits) können unterbunden werden
- Updates sind im Feld (beim Anwender) möglich

# Fazit

- Eclipse bietet ein leistungsfähiges Framework auch für Embedded-Entwicklung
- Gute Integrierbarkeit externer Anwendungen über Plugins (Beispiel: AVRDUde)
- Tools für “große” Anwendungen können auch für Mikrocontroller als Ziel eingesetzt werden (CDT, gdb, Modellierungstools, oAW, ...)

Danke für Ihre Aufmerksamkeit