

The background of the slide is a faded, aerial photograph of a city, showing various buildings and streets. A semi-transparent, light-colored rectangular area is overlaid on the center of the image, containing the main text.

# **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

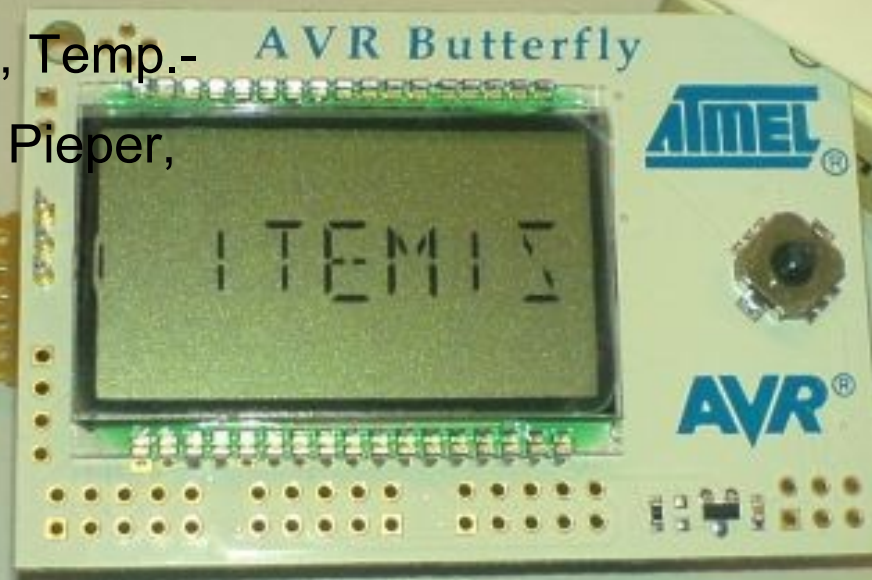
- Vorstellung
- Zielsysteme
  - AVR Butterfly
  - D071
  - STK500
  - Minimalsysteme
- Software
- Modellierung
- Übertragung zum Gerät

# Worum es nicht geht



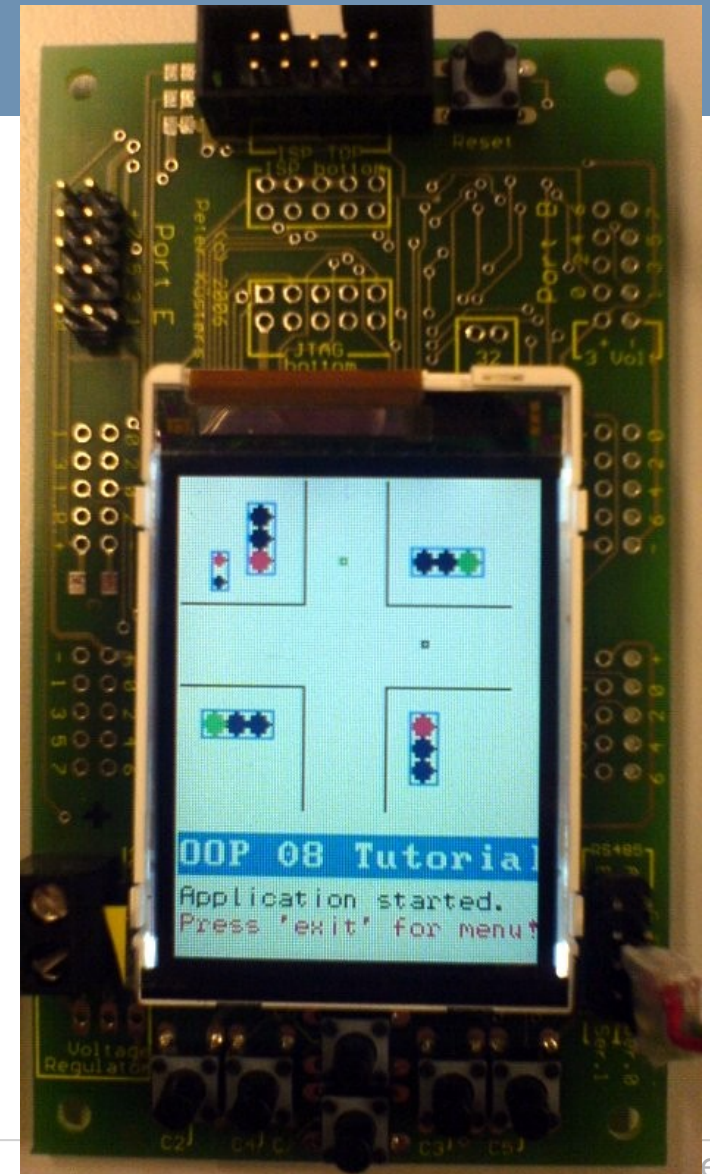
# AVR Butterfly

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



# Display 3000 D071

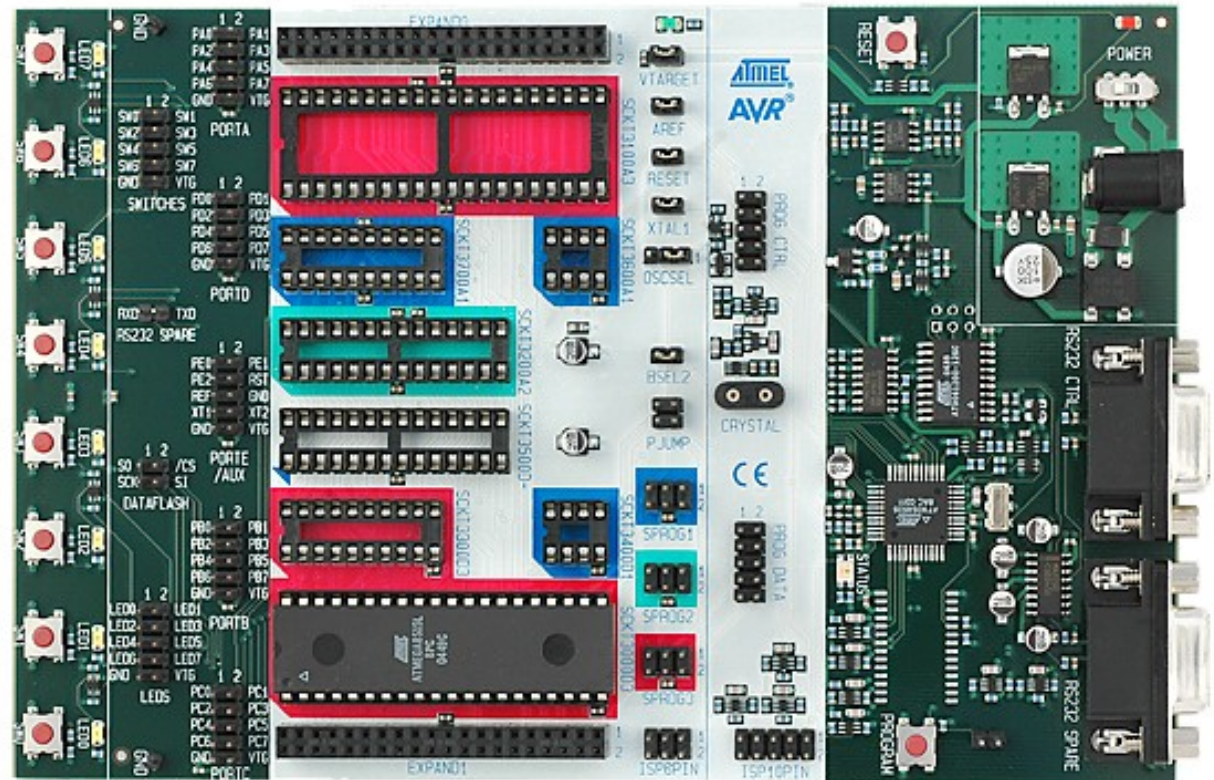
- Prozessor: ATmega128
- 128 kB FLASH-ROM
- 4kB RAM
- 2,1" TFT-Farbdisplay
- Taster
- herausgeführte I/O-Leitungen
- 2x RS-232
- Optional: CAN-Bus



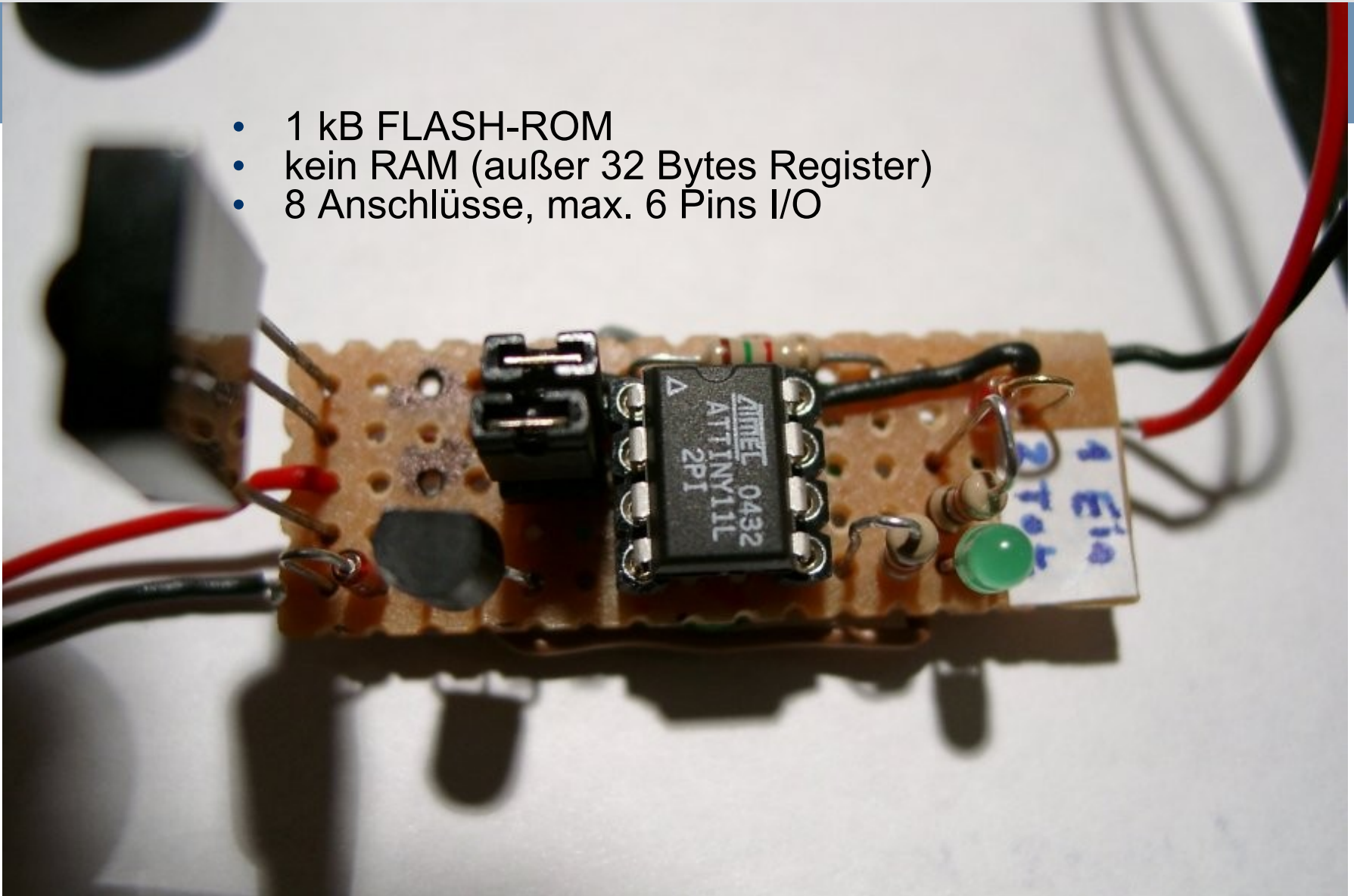


# STK 500

- 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

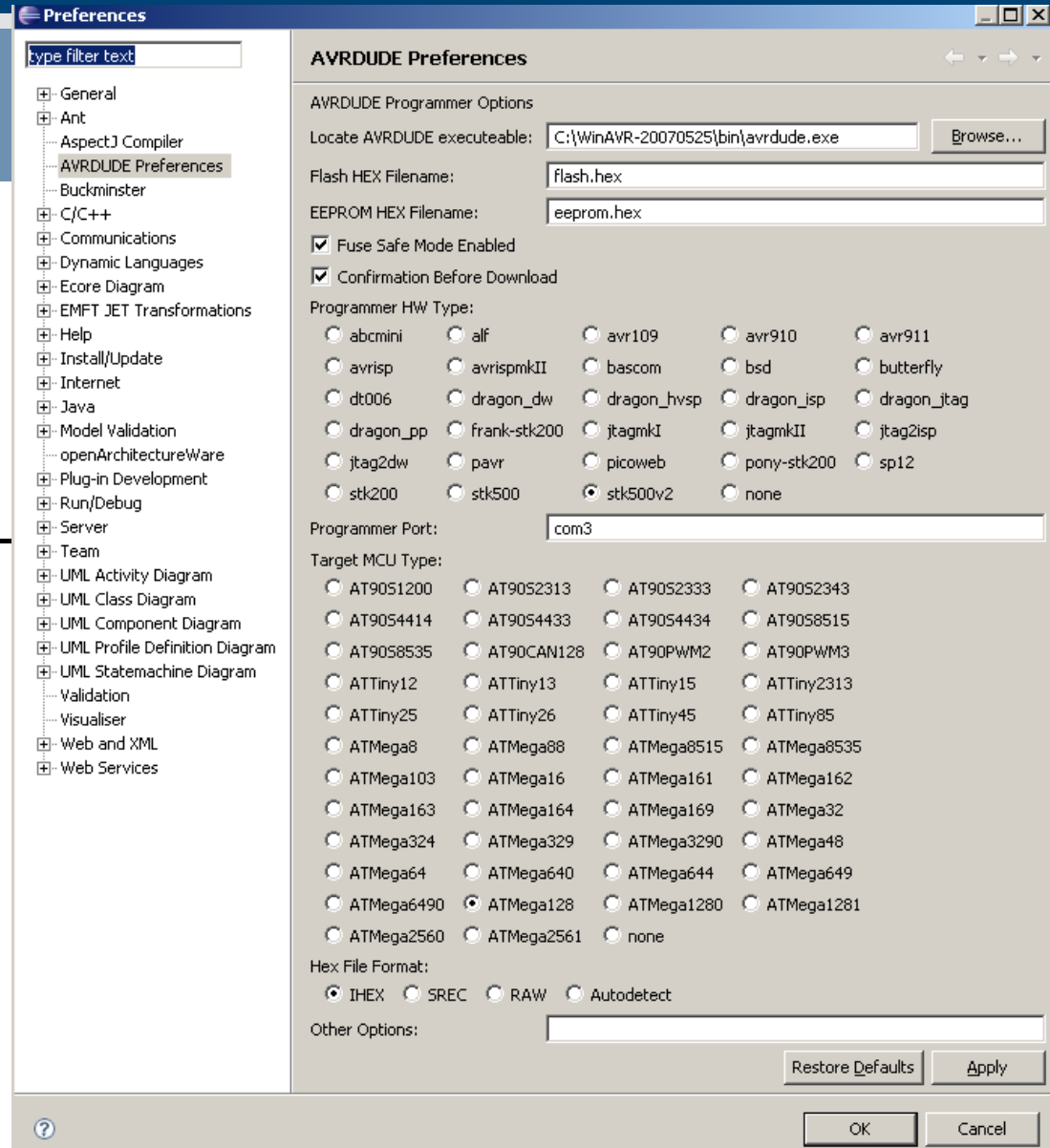
- AVR-GCC Crosscompiler / Assembler
- AVR-LIBC: optimierte Libraries
- diverse Tools für Binärformat
- Simulavr – Simulator
- avr-gdb – Debugger
- AVRDude zum flashen
- Ferner: Echtzeit-Multitaskingbetriebssysteme (z.B. FreeRTOS, AvrX)

The screenshot displays the Eclipse IDE interface for an AVR project. The Project Explorer on the left shows the file structure, including source files like main.c and various header files. The main editor window shows the source code for main.c, which includes headers for string.h, avr/eeprom.h, FreeRTOS.h, task.h, uiTask.h, glcd-Display3000-211.h, queue.h, main.h, serial.h, events.h, and safeDemoTask.h. The console window at the bottom shows the output of the AVRDUDE command, indicating that the flash memory has been specified and an erase cycle will be performed. The output also shows that the input file C:\OOP2008\workspaces\test\D071.base\Safe\flash.hex was auto-detected as Intel Hex and successfully written to the flash (25624 bytes) in 3.58 seconds.

- CDT
- AVRDUDE

# AVRDUde

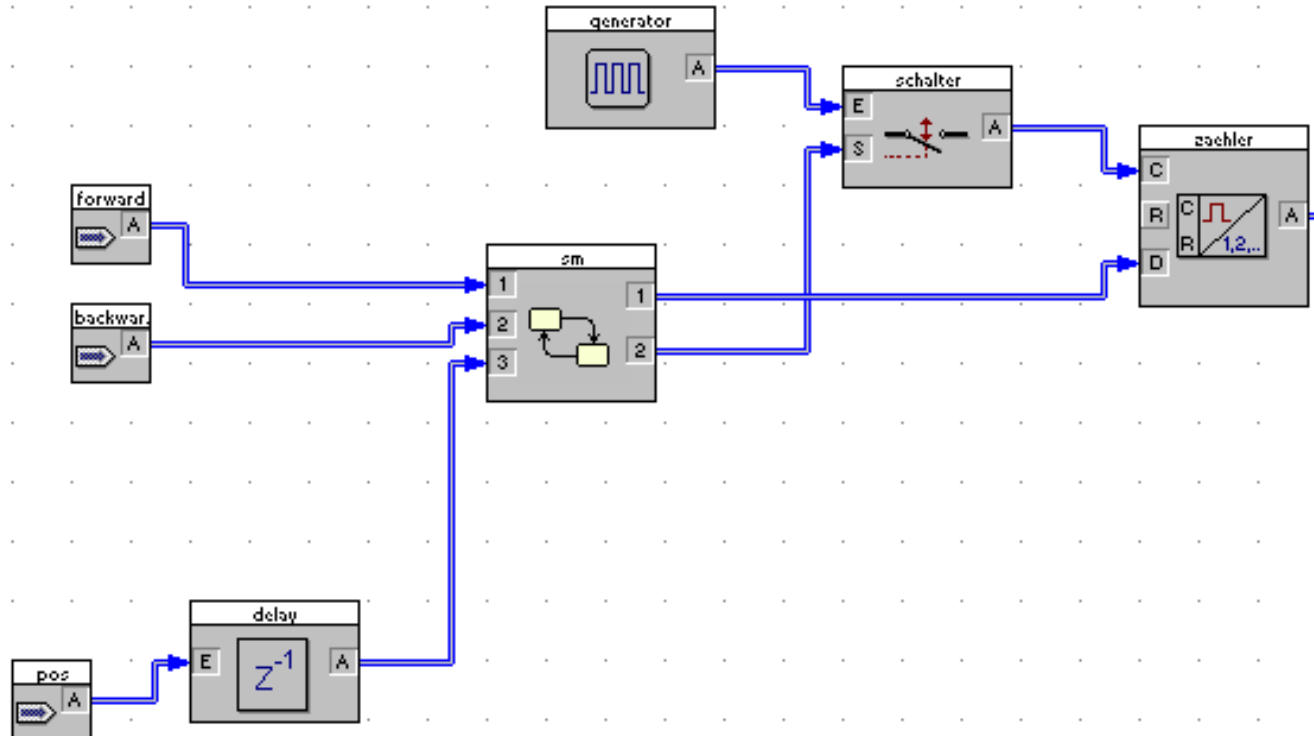
- Programmierung der Controller über den AVR-Button
- Einstellung der Parameter über Preferences



# 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 the state machine diagram with states: Off, Stop, Play, Pause, and TrayOpen. Transitions are labeled with events and actions, such as "Power On" leading to "Start Transition ON", "Play" leading to "Play", "Pause" leading to "Pause", and "Track Minus On State" leading to "H".

The right sidebar shows the Outline view, which lists the state machine elements in a tree structure:

- <<initialState>> <Pseudostate> Start
- <State> Off
- <State> TrayOpen
- <State> On
- <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 and Properties views. The Properties view displays the following information:

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 displays the Eclipse IDE interface with three panels. The left panel shows the Package Explorer with a project structure including 'src-gen' and 'c\_code'. The middle panel shows the Java source code for 'AbstractDevice.java', which uses a switch statement to handle various events like 'POWERSWITCHPRESSED', 'TRAYOPEN', 'STOP', 'TRACKPLUS', and 'TRACKMINUS'. The right panel shows the corresponding C code for 'AbstractDevice.c', which uses a similar switch statement and includes guard clauses to ensure state transitions are valid. The status bar at the bottom indicates 'Writable', 'Smart Insert', and '59:1'.

```

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;
        }
    }
}
    
```

```

#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;
        }
    }
}
    
```

# Simulation (Java)

```

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 simulation is running on ATmega169, with the current line being Ln 262, Col 1. The system tray shows the Start button, AVR Studio - \\Galeria..., LCD Properties, and the system clock at 15:09.

The screenshot shows the Eclipse IDE in a debug configuration for an AVR microcontroller. The main window displays the source code for `safeDemoTask.c` at line 161, where a static array of port characters is defined. The disassembly window on the right shows the corresponding assembly instructions, all of which are `0xffff`. The console at the bottom shows the GDB prompt and the result of the `var-evaluate-expression` command, which is `80026`.

**Debug Console Output:**

```
(gdb)
232-var-evaluate-expression var72
232^done,value="80026"
(gdb)
```

**Registers Window:**

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

**Source Code Snippet:**

```
157 void addTrafficLight(unsigned portCHAR type, signed portCHAR ori, unsigned portCHAR x,
158                     unsigned portCHAR y)
159 {
160     unsigned portCHAR radius = (type == 0 ? 4 : 2);
161     static unsigned portCHAR x1, x2, y1, y2;
162     // red:
163     outputHandles[nextOutputIndex] = createMacroitem(
164         mi_round_indicator1, 1, x*scaleX, y*scaleY, radius, 1);
165     uiq_macroitemCmd(outputHandles[nextOutputIndex++], 0, 0);
166     // yellow (cars only)
167     if (type == 0) {
168         outputHandles[nextOutputIndex] = createMacroitem(
169             mi_round_indicator1, 2,
170             x*scaleX + 2*radius * (ori==1?-1:ori==2?1:0),
171             y*scaleY + 2*radius * (ori==0?1:ori==2?-1:0),
172             radius, 1);
173     }
```

**Disassembly Snippet:**

```
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 ; ???
```

# 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**