# Embedded and Ambient Systems (vimiac06)

## Topics

1. General structure of embedded systems: typical sensors (high complexity device choice vs. advantage / disadvantage of custom development), signal conditioning tasks, ADC and DAC types and their areas of use. Typical processing units (μP, μC, DSP, FPGA) their relationship to each other is the performance of the devices, the design time and the task that can be solved in terms of complexity (what task would be solved by which means).

2. Giant Gecko (EFM32-STK3700) developer card: typical sensors (brightness meter, LC metal sensor, touch sensor) (operating principle, what design guidelines are there, what μC peripherals are used). Principle of clock management on the Giant gecko processor, how it serves to reduce consumption.

3. Development environments and compilers: translation process (.c → obj → link). Some typical knowledge of translation interface (-D, -O0… -O3, -o, -mcpu, -I, -Wall, -std). make program and knowledge of makefile format: makefile rules (goal, prerequisite, instruction syntax, interpretation of simpler patterns)

4. Software architectures: aspects of software architecture design. The typical typical architecture of software architectures (sample code, schedule diagram, response time…) and their properties: cyclic program organization (and its cases), supplemented with interrupts, scheduled functions.

5. Interrupt Management: General principles for interrupt initialization and management. THE general hierarchy of interrupt authorization, principle of vector interrupt handling, Cortex Structure of the interrupt handler M3, ATmega128 and ADSP-BF357 (in principle, complex figures are not required). Here are some ways to specify interrupt handling functions in C (Cortex-M3, ATmega128, ADSP-BF537, ADSP21364). How to get functions a vector table, and how do we tell the compiler that they are interrupt functions?

6. Shared variables: formulation of a basic problem, for which variables is it critical, example, solution options. Double buffering. Dynamic memory usage in embedded systems (malloc function). Stack overflow. Robust programming (timeout, secure coding, structured program organization, type usage, redundancy).

7. Special C language elements: inline functions, bitfield structures, union data type, structured management of register arrays (direct access to memory areas), __attribute__ (eg: interrupt, always_inline, weak) and #pragma (eg once, interrupt, align) keywords, idiom recognition, #define specialties. Examples.

8. Portable code: what it means, why it is important, integer data types (stdint.h), library functions (what to look for, blocking / non-blocking). Virtualization: 1 and 2 operating model, hypervisor function, requirements for it.

8. Troubleshooting. Debugging features of an embedded system. Used for debugging tools: debugger, tracer, profiler, watchpoint. Debug built using JTAG port system block diagram, typical tasks implemented using JTAG. Typical basic debug options (GPIO, UART: redirect printf). Runtime measurement methods.