innovations
for high
performance
microelectronics

Configurable Sensor Nodes for AAL Applications

Peter Langendörfer

Krzysztof Piotrowski, Anna Sojka, Frank Vater

IHP
Im Technologiepark 25
15236 Frankfurt (Oder)
Germany

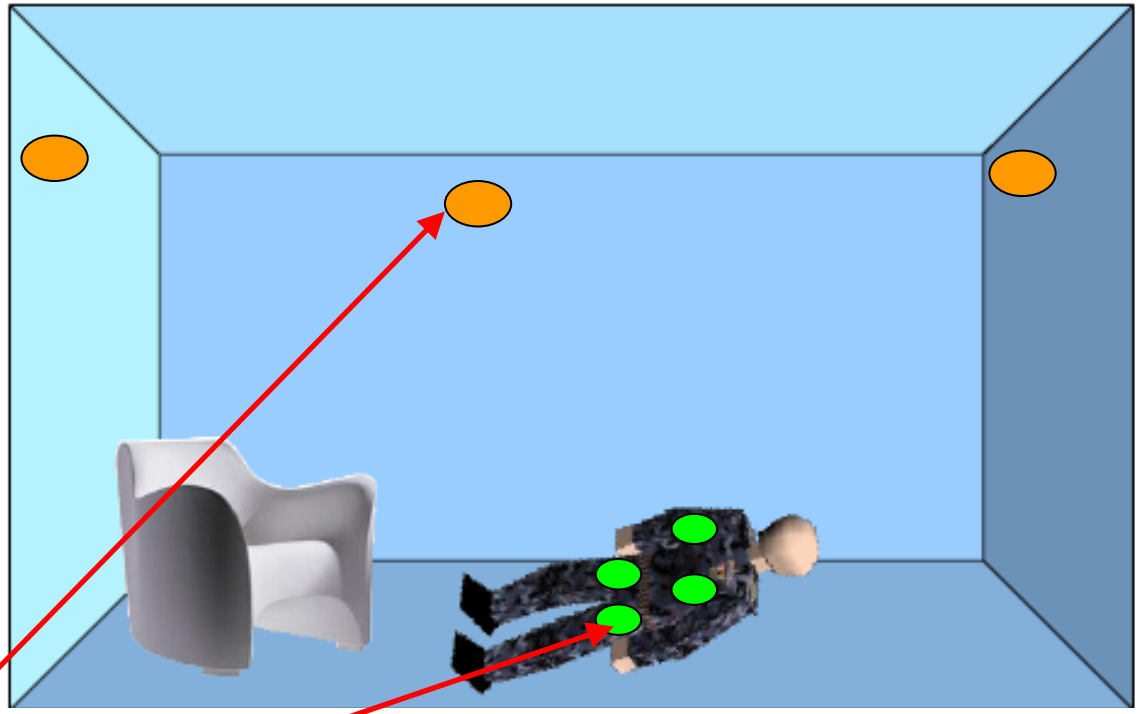
Some background

Outline

- **Introduction and Motivation**
- **Configuration approach: solutions and challenges**
- **First all IHP vital data monitoring node**
- **Life Demo: featuring Lars Wolf**
- **Conclusions**

Example – AAL Application

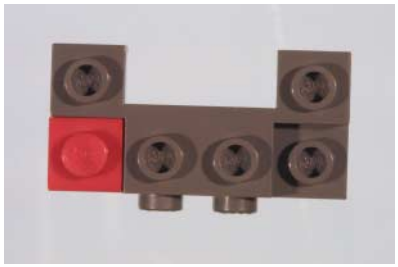
- **AAL application for monitoring vital parameters and fall detection**
- **Requirements:**
  - **Monitor vital parameters**
  - **Fall detection**
  - **Positioning**
  - **Strong protection of user data**
- **Two different kinds of WSN required**
  - **Fixed installation of large nodes**
  - **Small body area network (BAN)**

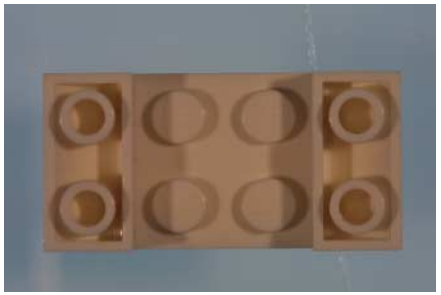


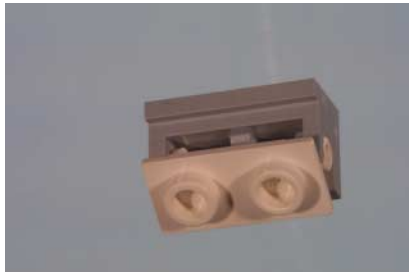
What we all would like to have

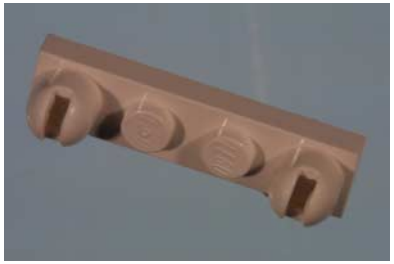
Lego like interfaces

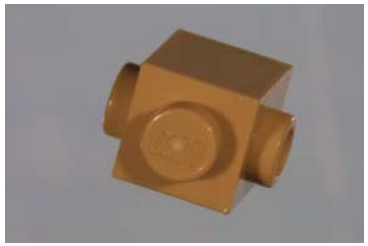
But even the Lego world
becomes more and more complex

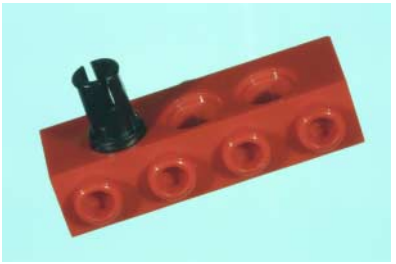




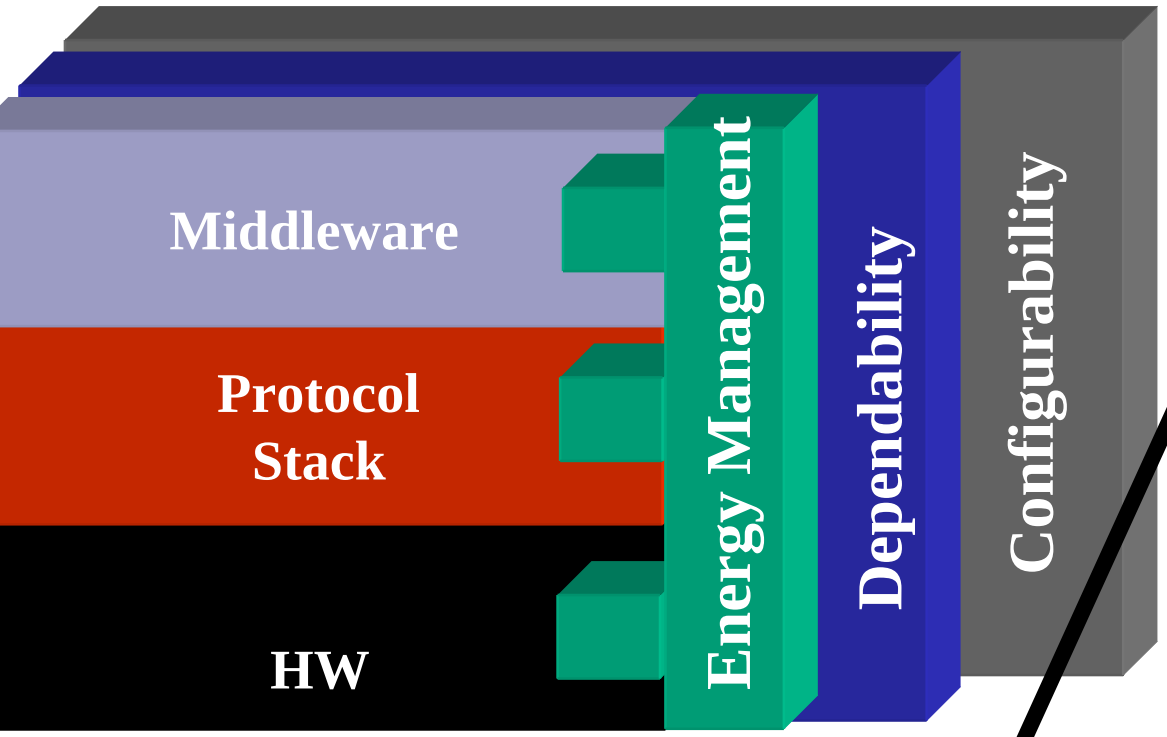




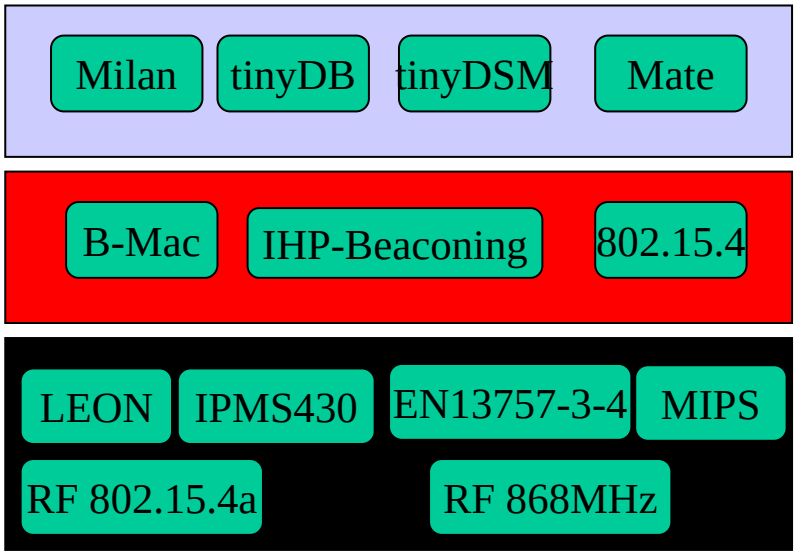




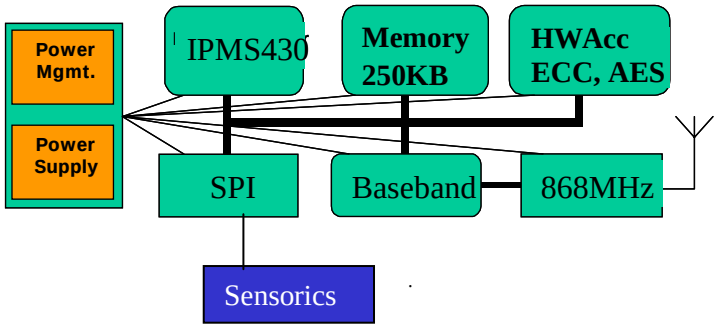
Sensor Node Configuration Approach



Development



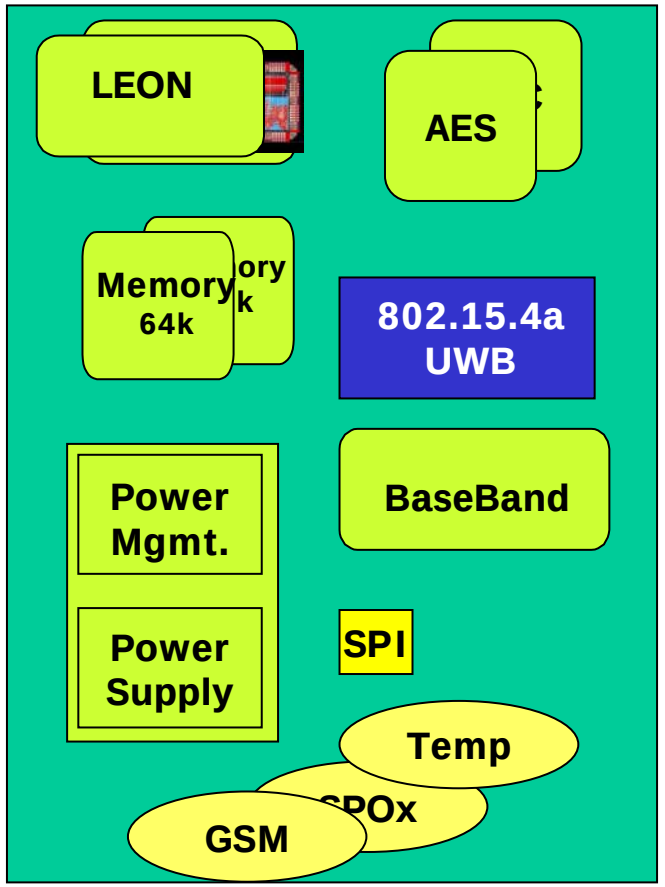
Runtime



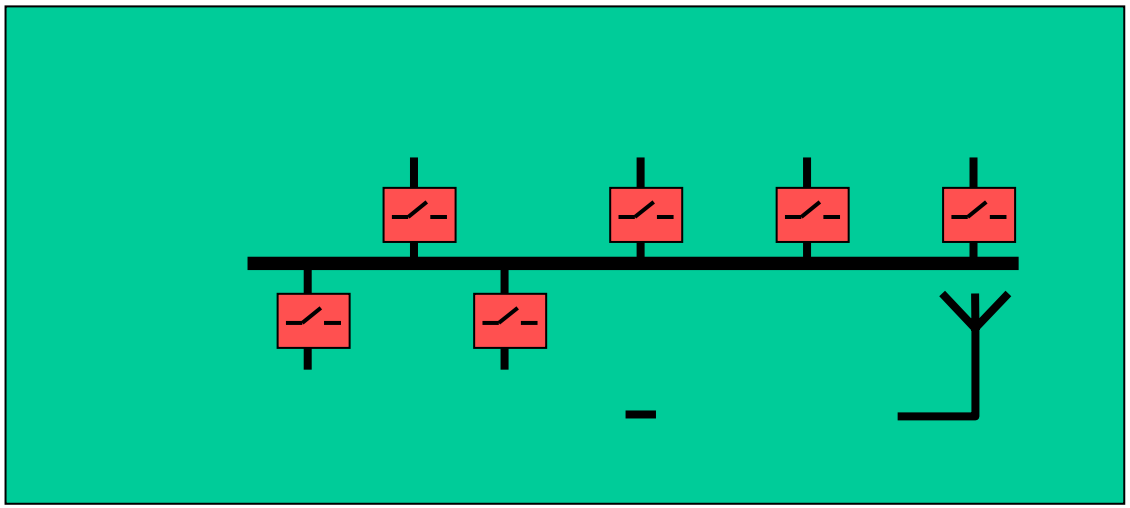
Compiling complex ASICs: Positioning node

- **Fixed Point node:**
  - -32 bit µC,
  - -UWB for communication and positioning

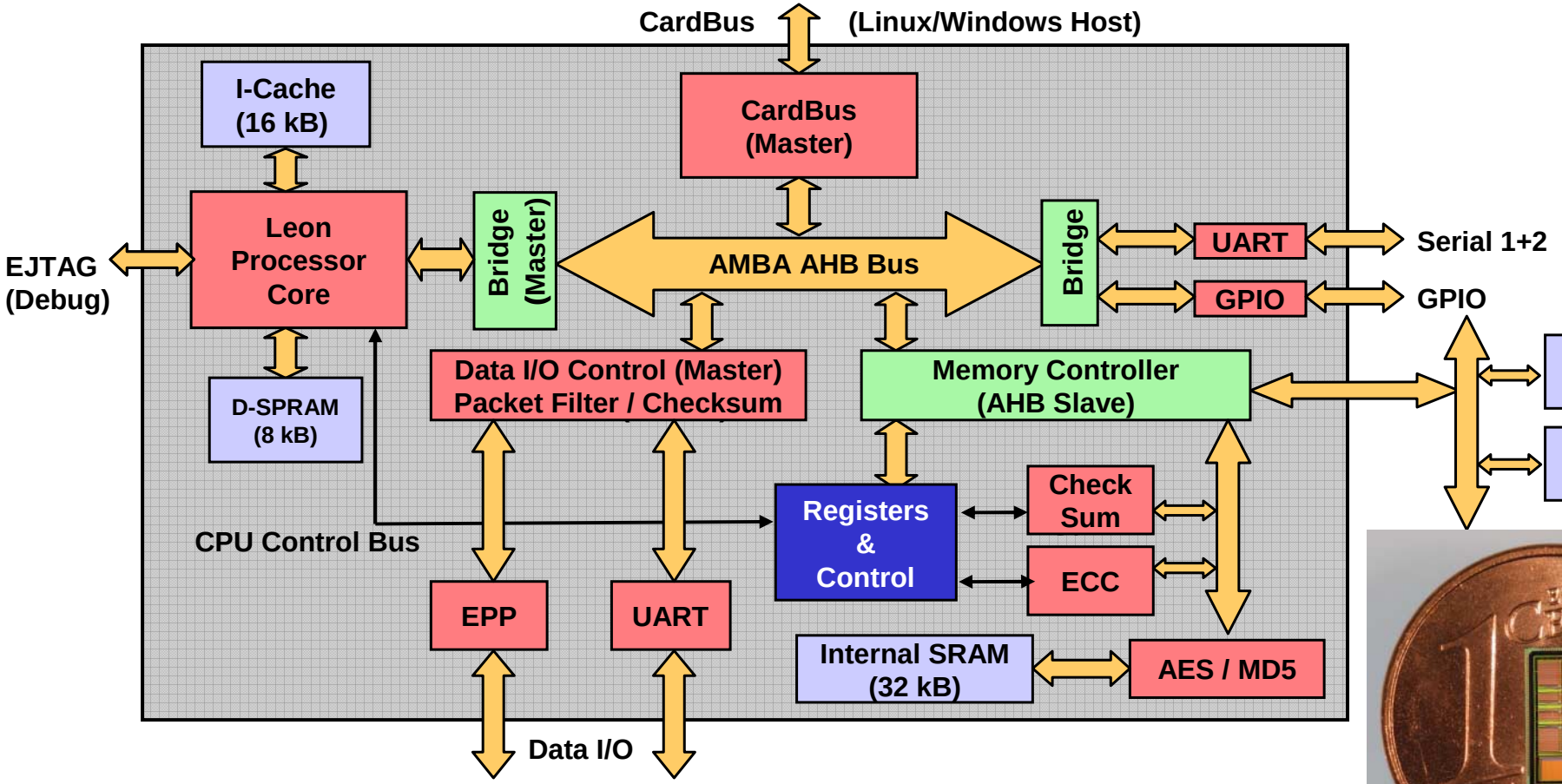
Library

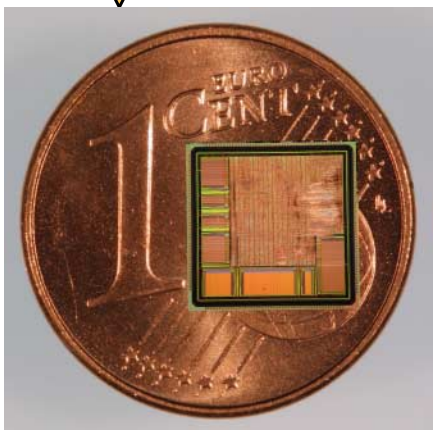


Node



Positioning node (approx.)

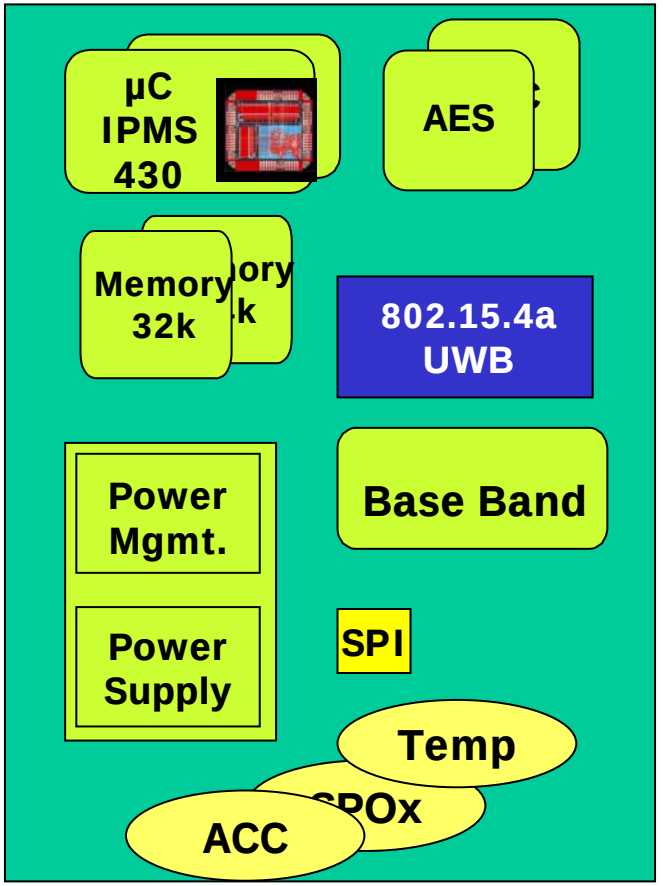




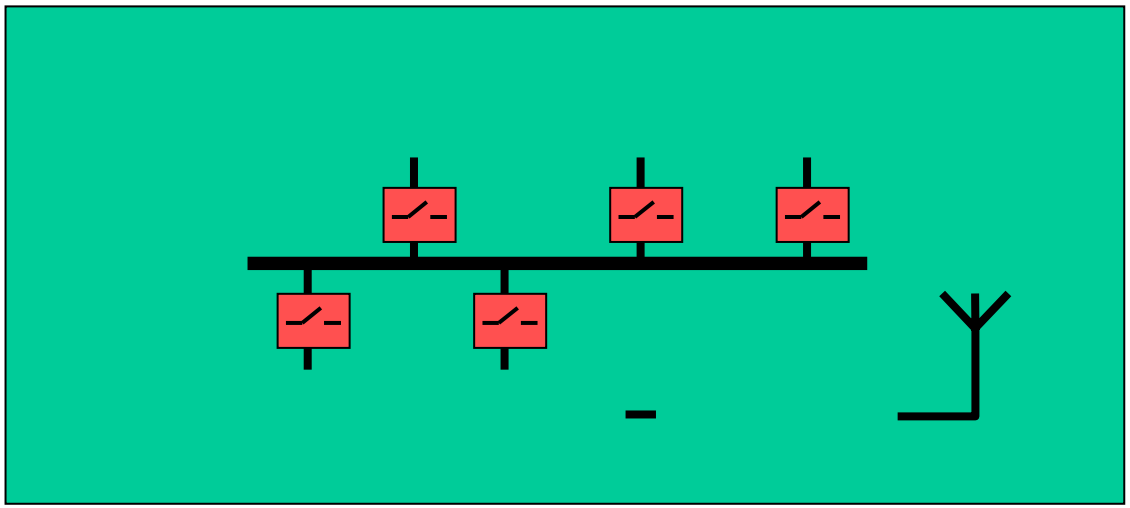
Compiling complex ASICs: vital data monitor

- **Body Area Node (BAN)**
  - -16 bit µC
  - -Communication interfaces (SPI, UWB)
  - -Encryption
  - -Fall sensor

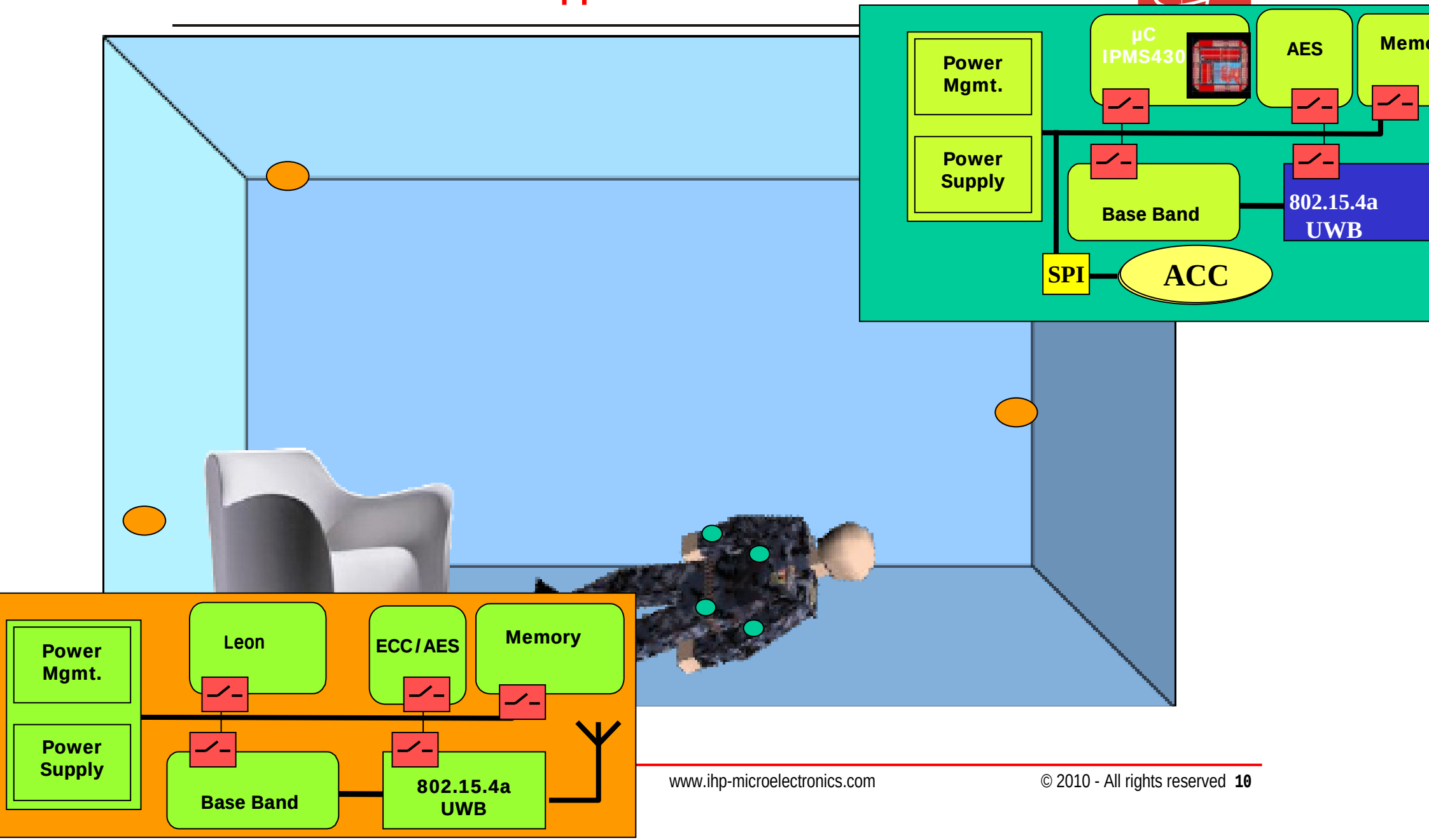
Library



Node



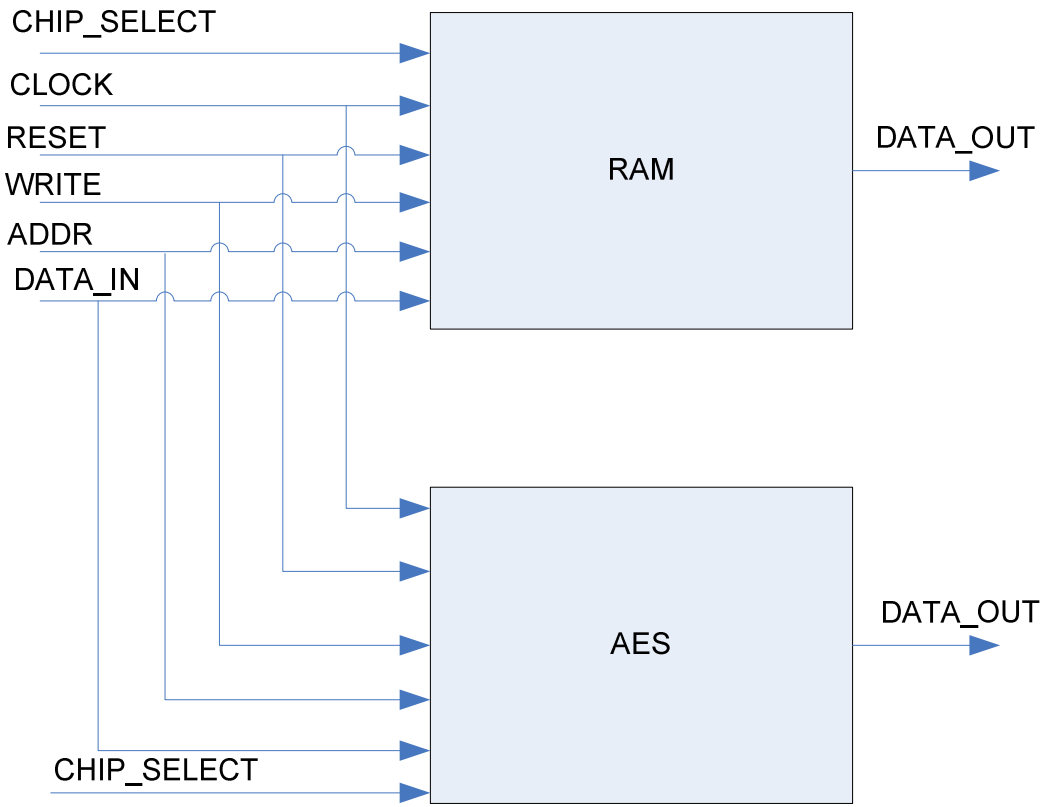
Sensor nodes for AAL- Applications: Hardware



Interface Issues

- **Strong need to combine individual hardware blocks**
  - **2nd µC or DSP**
  - **Protocol accelerators**
  - **Crypto accelerators**
- **Strong need to combine software components**
  - **Protocols & operating systems**
  - **Middleware & operating systems**
  - **Middleware & applications**
- **Providing access to specialized hardware for**
  - **Applications**
  - **Middleware & protocols**
  - **Operating system**

Hardware Blueprint: Memory-like Interface (MLI)



- **Data bus is 32 bit wide**
- **CHIP_SELECT-Signal is used to determine hardware block**

Hardware Blueprint: Software for AES integration

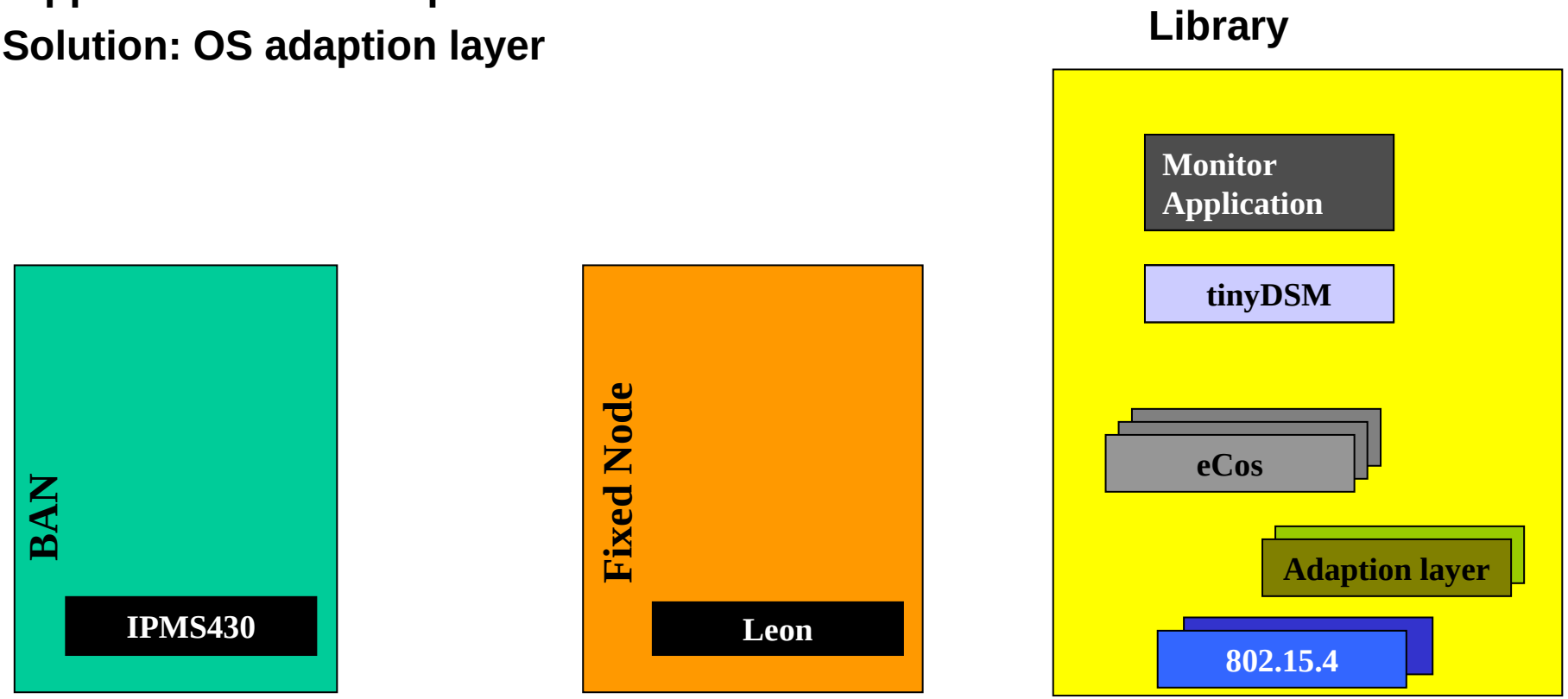
- **AES data block 128 BIT => 4 chunks á 32 Bit**
- **Commando word does not exist**
- **Instead: address bits select the function (`key, data, encryption/decryption`)**
- **Integration via Macro**

```
// AES base address
#define AESBASE 0x20200000 //memory mapped IO
//Write key
writeReg(0x2b7e1516, AESBASE + KEYBASE + 0);
writeReg(0x28aed2a6, AESBASE + KEYBASE + 1);
writeReg(0xabf71588, AESBASE + KEYBASE + 2);
writeReg(0x09cf4f3c, AESBASE + KEYBASE + 3);
writeReg(0x3243f6a8, AESBASE + DATABASE + ENCRYPTION + 0);
writeReg(0x885a308d, AESBASE + DATABASE + ENCRYPTION + 1);
writeReg(0x313198a2, AESBASE + DATABASE + ENCRYPTION + 2);
writeReg(0xe0370734, AESBASE + DATABASE + ENCRYPTION + 3);
//Wait for 70 Clock cycles or interrupt
//READ_OUT
```

Compiling complex software for heterogeneous systems

- **Different OS (tinyOS, Contiki,…) and the same application on the top?**

→ Solution: OS adaption layer



Software blueprint: tinyDSM OS adaptation layer, realisation

- **OS specific functionality realisations**
  - **Timers**
  - **Input/Output (flash, radio, etc.)**
  - **Task scheduling**
- **The tinyDSM middleware core integration**
  - **The OS adaptation layer allows easy integration**
- **tinyDSM is implemented in C**
  - **Uses a specified internal interface for the used OS functions**
  - **For each OS specific interface a wrapper is needed**
  - **The OS adaptation layer consists of a complete set of wrappers**

Software blueprint: Wrapper sample

Communication interface process task

- **implementation in the tinyDSM core**
  `void CorecommIntProcess_Task(){…}`
- **scheduling of task in tinyDSM known**
  `Corepost_commIntProcess();`

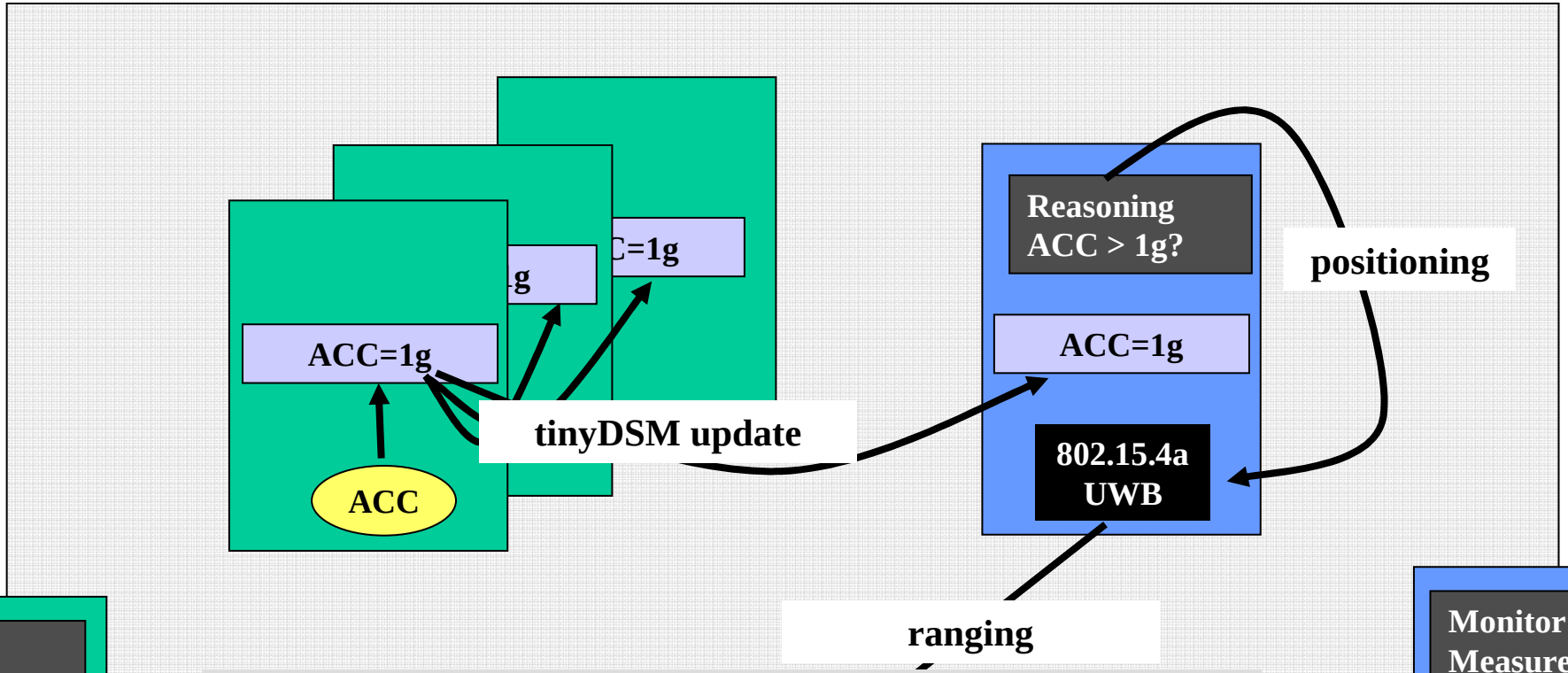
Contiki Wrapper

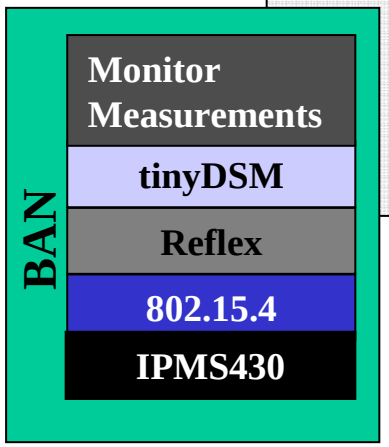
```
PROCESS(commIntProcess, "commIntProcess");
PROCESS_THREAD(commIntProcess, ev, data){
  PROCESS_BEGIN();
  CorecommIntProcess_Task();
  PROCESS_END();
}

void Corepost_commIntProcess(){
  process_post(&commIntProcess, 0x81, 0);
}
```

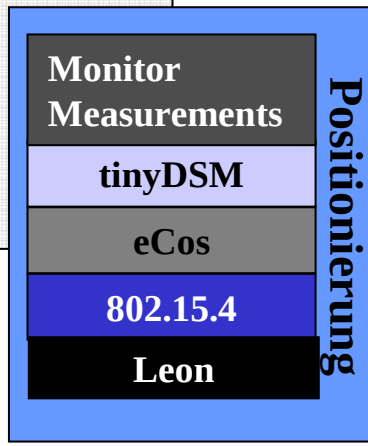
TinyOS Wrapper

```
task void commIntProcess(){
  CorecommIntProcess_Task();
}


void Corepost_commIntProcess(){
  post commIntProcess();
}
```

tinyDSM empowered event scripting





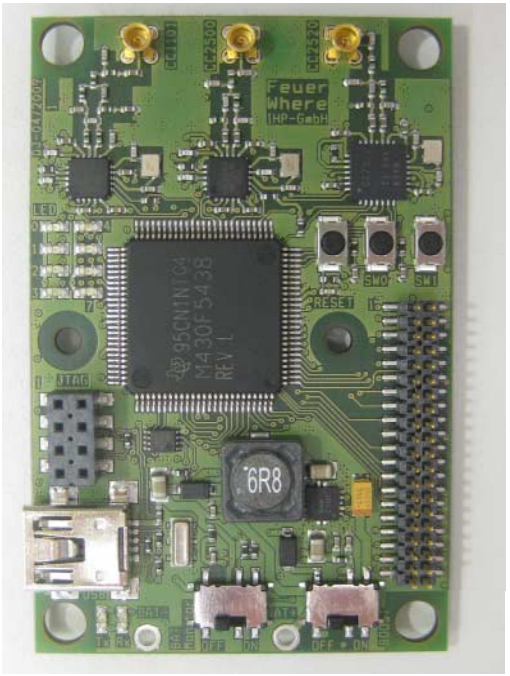


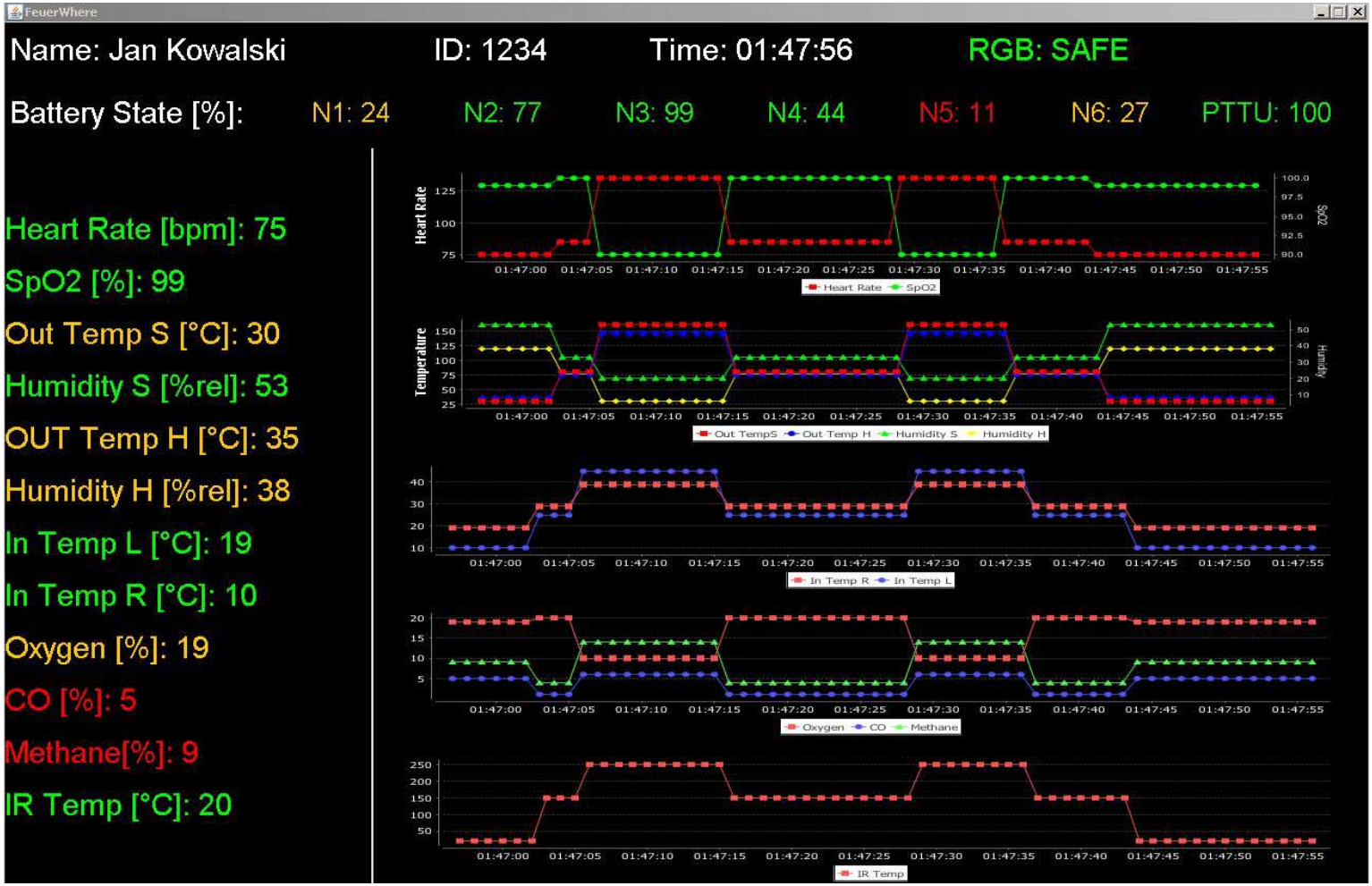

```
Event ACC IF acc > 1g trigger update()

Event ACC IF acc > 1g trigger positioning();
```

Distributed fall detection based on tinyDSM

Vital data monitoring for firefighters

IHPs Sensor Node for Vital Data Monitoring





IHP FeuerWhere Node

Conclusions

- **Plug‘n Play for sensor node hard- and software is a big challenge**
- **Some solutions exist and can be used to customize sensor nodes**
- **Tool support is an open research issue**
- **Hand crafted working solution demonstrated**

Thank you for your attention

Questions or comments

?

Visit us @ Hannover Messe; Hall 2; Booth D52

Contact me at:

langendoerfer@ihp-microelectronics.com

http://www.ihp-microelectronics.com/~langend

