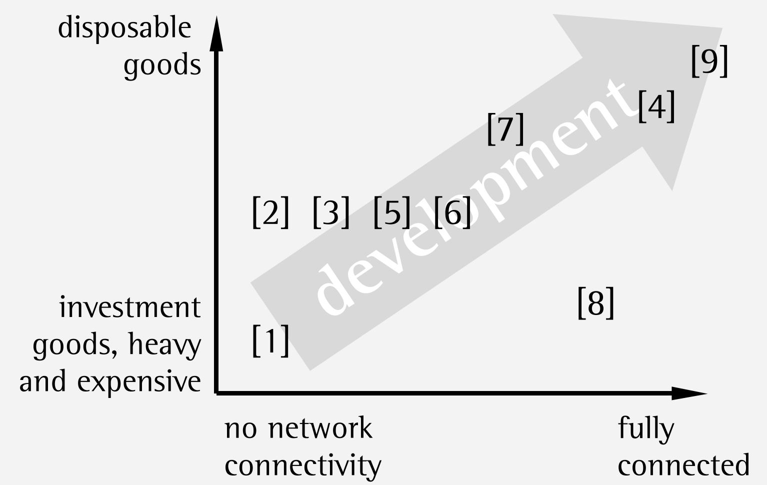
Lecture on Sensor Networks

Sensor networks - motivation



Historical Development

[1] IBM S 3/60 (1960)
[2] Apple II (running VisiCalc)/IBM PC/C 64 (1980)
[3] 486er PC, Amiga and modem, acoustic coupler, BTX (minitel in France) (mid 80ies)
[4] Cell phones become bulk article (end of 80ies, beginning of 90ies)
[5] Pentium class PCs, Datex-J, soon replaced by Internet (90ies)
[6] Boring PC-era (getting smaller, faster), increasingly „always-on“ (mid 90ies)
[7] GPRS capable PDAs, vanishing borders between PDA and cell phone (late 90ies)
[8] Connected car
[9] Smart Dust

Sensor networks - motivation

Applications for Sensor Networks

Monitoring the integrity of buildings and building automation

Changes in structural integrity, developing over years or after earthquakes, could be detected earlier. Sensors which are built into walls or the concrete could accomplish this task without any power supply or network connection. The sensor nodes would only have to "wake up" between large intervals like minutes or hours working for years or even decades. The dynamics of collapsing buildings could be analyzed after the event using data that may have been sent during the collapse. In particular our ESB nodes can be woken up by timer events as well as vibration or tilt events.

Each light bulb in or outside buildings could function as sensor node. A broken bulb could e. g. trigger a switched off neighbor to switch its own state. No complex and expensive wiring or control wires would be necessary. Bulbs could also be triggered by activity in a room with motion sensors like the ones used for the ESB nodes. Furthermore, every bulb could be part of a distributed alarm device measuring movements in a building. Simply unplugging the system would not be trivial anymore as there would no longer be a single point of failure.

Sensor nodes could serve as thermostats with no wiring overhead. Every room could be controlled independently.

Sensor networks - motivation

Applications for Sensor Networks

Early discovery of catastrophes like forest fires

By scattering sensor nodes from an airplane over a forest a so called ad-hoc network is built up autonomously. Heat-sensing nodes can signal events like fires over the network. This enables the early detection of forest fires which is crucial for efficient fire fighting.

Medical surveillance and remote diagnosis

In the future, long term measurements of vital functions might be possible with the help of tiny sensors which could be implanted under the skin, swallowed etc. Small, fully encapsulated and disposable video sensors which can be swallowed do already exist. They are able to send images from a person's interior for about 24 hours with no surgery necessary.

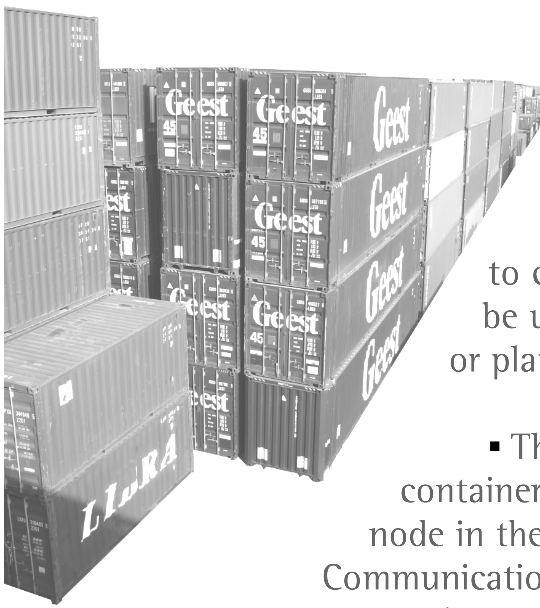
Burglary prevention

Safety for buildings and other territory without any installations. Surveillance of railroads in order to prevent crashes with animals and humans may be an application.

Sensor networks - motivation

Applications for Sensor Networks

Business applications



- Stock-keeping with connected temperature and humidity sensors contained in packages which can signal high humidity or exceeding temperature. Nodes being added to packages could at the same time be used to carry further information like recipes for food. They could be used to track and authenticate goods as it is already done or planed with the help of RF-IDs.

- These sensor nodes could also enable the administration of containers in container harbors. Every container would represent a node in the sensor network and could remember its content reliably. Communication over longer distances would be done hop-by-hop from one container to the next extending their range significantly.

- A collection of containers would represent the database itself and would thus always be consistent. Ships could easily identify their correct load and a container could even report a missing neighbor.

Sensor networks - motivation

Applications for Sensor Networks

Agriculture

Sensor nodes could be added to the seed. Dry areas could be identified easily by directing a query to the sensor network. Cattle would also be equipped with sensor nodes in order to track them (that saves on sheep dogs).

Pollution control

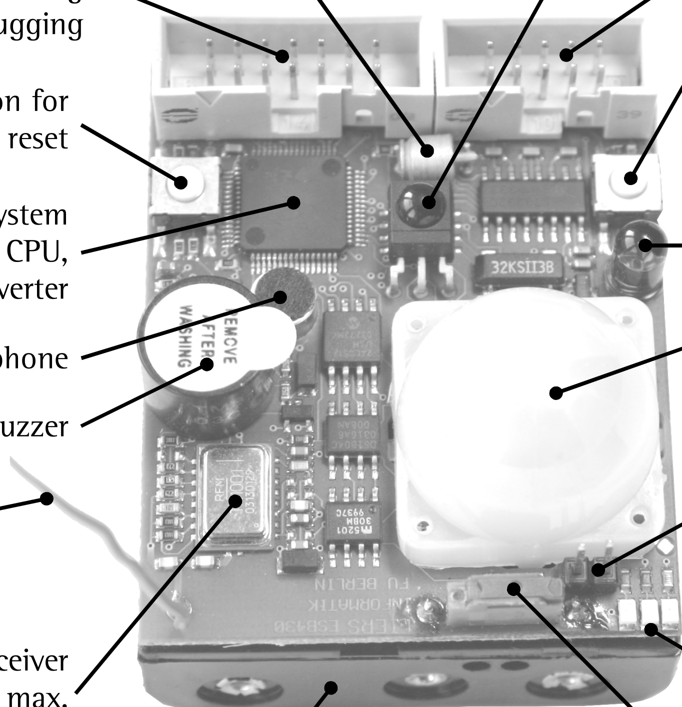
Surveillance of waters: A remote sensor network could be connected via GPRS and a cell-phone with the conventional telephone network and deliver (sparse) data over long distances. Actually the ESB nodes are equipped with a link to a cell-phone. The firmware includes a function to send SMS messages.

Nodes could also be scattered over an industrial site to detect leakages of gas or chemicals and alert in an early state.

Dam protection could be accomplished by including sensors into the dams or between sandbags. Early detection of intruding water could be used to strengthen the dam accordingly.

Sensor networks - motivation

Introduction to the ESB (Electronic Sensor Board)



Memory Organization

Introduction to the ESB (Electronic Sensor Board)

16 addresses for sub-routines	0xFFE0-0xFFFF	Interrupt Vectors
Is written once prior to initialization, however it can be changed in chunks of 512 bytes during operation	0x1100-0xFFDF	ca. 60 kByte Flash-ROM for firmware, programs, data, tables If there is enough energy left, the application itself can write data here!
Two small blocks	0x1000-0x10FF	2x128 Byte Flash-ROM
Programmed via scatt.-fl.	0x0A00-0x0FFF	Boot-Loader ROM (fix)
Only 2kB fast RAM	0x0200-0x09FF	RAM (for variables, stack)
No real memory behind those addresses, but connected with the "outside world" (memory-mapped)	0x0100-0x01FF	16-Bit periphery (Memory mapped) — only word-wise (16 Bit) reading
	0x0000-0x00FF	8-Bit periphery (Memory mapped) — only byte-wise (8 Bit) reading

16 Bit Multiplications

Introduction to the ESB (Electronic Sensor Board)

Multiplications are not included into the core of the MSP430. However, there is a hardware multiplier, which is addressed via the mapped memory for 16 Bit periphery (0x100-0x1FF) just like every other external device (e.g. the light emitting diodes). TI denotes the four types of multiplications with MPY, MPYS, MAC and MACS.

MPY: 1. Operand unsigned Multiplication	0x130
MPYS: 1. Operand signed Multiplication	0x132
MAC: 1. Operand unsigned Mult. a. Add.	0x134
MACS: 1. Operand signed Mult. a. Add.	0x136
2. Operand signed/unsigned	0x138

RESLO: 16 LSW of the result	0x13A
RESHI: 16 MSW of the result	0x13C
SUMMEXT: Sum Extension	0x13E

Sensor networks - motivation

Future Batteries

Energy measured in watt hours / gram

Lithium-Ions in chemical batteries:	0.3
Methanol in fuel cells	3.0
Tritium in nuclear batteries	850.0
Polonium-210 in nuclear batteries	57,000.0

We assume a degree of efficiency of about 50% for the fuel cell and only about 8% for all radio active isotopes. It still holds true that 0.08 x 57,000 > 0.5 x 0.3. Today, a problem with the nuclear batteries is that they deliver too small amounts of energy over a period of several years.

Sensor networks - motivation

Connect ESB <-> PC

Change to console 1-4 using Alt+F1-F4

login: sensor <RETURN>
password: . <RETURN>

Console 4: Connect the parallel port with the PC's local IP-network

```
msp430-gdbproxy msp430
```

Console 3: Open a serial terminal for entering commands and reading the ESB's output

```
minicom
```

Console 2: Console needed for programming

```
cd tmp/msp430/userapp
nano src/userapp.c
```
(advanced users may use emacs)

Console 1: console for compiling and flashing programs

```
cd tmp/msp430/userapp
```

`make`	compile application and firmware
`make flash`	automatically flash binary via gdbproxy (does not work the first time – repeat as necessary)

Sensor networks - motivation

ESB terminal commands

Some of the basic features of the ESB node can be used with a simple terminal application which allows to enter commands for the nodes that are transmitted over the serial connection. These commands are executed by the ESB immediately or returned values are sent back as ASCII strings.

Sampled values or those which have to be entered (e. g., for defining thresholds) usually range from 0-4095 as 12 Bits are provided by the AD/DA-converter.

LED commands

`rlr/rlg/rly`	`read state of LED red/green/yellow` `0=OFF / 1=ON`
`slr/slg/sly x`	`set state of LED red/green/yellow` `x=0 (OFF) / x=1 (ON)`
`swr/swg/swy`	`toggle state of LED red, green or yellow`

beeper commands

`rbp/sbp`	`read/set beeper state 0=OFF / 1=ON`
`sbp`	`set beeper state 0=OFF / 1=ON`

Sensor networks - motivation

ESB terminal commands

energy control

rvb/rve	read voltage battery/extern
rbl/rel	read threshold for battery/extern voltage
sbl/sel	set threshold for battery/extern voltage

microphone readings

rmc	read current value of microphone voltage
rmm	read counter for microphone voltage
rma	read average value of microphone voltage
rmi/smi	read/set sensitivity of microphone (noise is below 60)

temperature readings

rtt	read temperature value
rtl/rth	read low/high temperature threshold
stl/sth	set low/high temperature threshold
rte/ste	read/set temperature alarm enable bit the alarm wakes up the ESB based on the predefined threshold (stl/sth) 0=ALARM OFF / 1=ON

Sensor networks - motivation

ESB terminal commands

vibration and tilt sensor

rms	read counter for vibration sensor (accumulated value)
rvs	read both counter and tilt sensor value

light sensor

rls	read counter for light sensor
rll/rul	read lower/upper threshold for "light-alarm"
sll/sul	set lower/upper threshold for "light-alarm" - this wakes the ESB up if the sampled brightness is in the interval

radio transceiver

rrp	read current AD value of the transceiver (is never zero - why?)
rtp/stp xx	read/set transceiver transmission power range for x within [00,99]
rfr/sfr xxxx (xxxx < 4096)	read/set transceiver reception theshold (since the transceiver always samples value, e.g., noise, signal must be above the thresh.

Sensor networks - motivation

ESB terminal commands

miscellaneous

rcf	read configuration
rid/sid xxxx	read/set ID of ESB with x in [0000,0255]
raf/saf	read/set announce flag (?)
rmr/smr	read/set receiver's phone number for short message service
rsm/ssm	read/set sensor mask
dea	erase complete EEPROM
flx	start broadcasting userapp. (binary)

timer function

rt5	read 5 milliseconds timer
rdd/rct	read date/time of realtime clock
sdd dd-mm-yyyy	set date of realtime clock (sdd 31-12-2005)
sct hh:mm:ss	set time of realtime clock (sct 23:59:30)
sat hh:mm:ss	set alarm time. This is the time when the node should wake up
rce/sce x	read/set alarm time enable bit x=0 (OFF)/ x=1(ON)

Sensor networks - motivation

ESB terminal commands

infrared transceiver

```
sir xxyy       send RC-5 code via the IR sensor diode
               xx (hex) is the MSB (most significant bit)
               yy (hex) is the LSB (least significant bit)
               Bit 11  :    toggle bit (stable bit=key is
                            kept pressed/changing bit=key
                            was pressed again)
               Bit 6-10:    5 address bit (TV=0)
               Bit 0-5 :    6 code bits   (16=volume++)
               (0 << 11)|(0<<6)|(16)

rir            read last two bytes received by IR diode
```

memory functions

```
reb xxxx       read byte from address xxxx
rer xxxx yyyy  read bytes between xxxx and yyyy
web xxxx yy    write byte yy at xxxx
```

Sensor networks - motivation

Energy consumption

Important variables:

```
P_RX    4.5mA           energy consumption RECEIVING
P_TX    12.0mA          energy consumption SENDING
P_CL    12.0mA          BASIC consumption w/o radio
P_SL    8uA(0.008mA)    energy needed for SLEEP mode
```

[capacity (watt) = current (ampere) x voltage (volt)]

Rough estimation for energy consumption and sensor lifetime:

Let’s assume that each sensor should wake up once a second, measure a value and transmit it over the network.

a) Calculations needed: 5,000 instructions (for measurement and preparation for sending)
b) Time to send information: 50 bytes for sensor’s data, another 250 bytes for forwarding foreign data
c) Energy needed to sleep for the rest of the second (Sleep-Modus)

Sensor networks - motivation

Energy consumption

(a) Time for calculations and energy consumption

MSP430 running at 8Mhz clock rate => one cycle takes 1/(8 x 10^6)s
1 instruction needs an average of 3 cycles => 3/(8 x 10^6)s, 5,000 instructions 15/(8 x 10^3)s.
15/(8 x 10^3)s x 12mA = 180/8,000 = 0.0225mAs (milli Ampere seconds)

(b) Time for sending data and energy consumption

The radio frequency unit sends with 19,200 baud (here approx. by 19,200 bits/s)
1 bit takes 1/19,200s. We have to send 50 bytes (own measurement) and we have to forward 250 bytes (external data): 250 + 50 = 300 which takes 300 x 8/19,200s
300 x 8/19,200s x 24mA (energy basic+sending) = 3mAs

(c) Energy consumed while sleeping

Time for calculations 15/8,000 + time for transmission 300 x 8/19,200s = ca. 0.127s
Time for sleep mode = 1s – 0.127 = 0.873s
Energy consumed while sleeping 0.008mA x 0.873s = 0.007mAs

Sensor networks - motivation

Energy consumption

Total amount of energy per second and resulting lifetime

The ESB needs to be supplied with 4.5V, so we need 3 x 1.5V AA batteries.

(0.0225 + 3 + 0.007) x 4.5V = ca. 4.5V x 3.03mAs (= mWs (milli Watt seconds))
Energy of 3 AA batteries: ca. 3 x 1.5 x 2,300mAh = 4.5 x 2,300 x 60 x 60mWs
Total lifetime: 4.5 x 2,300 x 60 x 60/4.5 x 3.03 = ca. 32 days

Critical review

- Battery suffers from leakage current (loosing about 10% energy/year)
- Small network (forwarding takes only 250 Byte)

most important: Only sending was taken into account, not receiving!
If we listen into the channel rather than sleeping 0.007mA has to be replaced by (12+4.5)mA which results in a lifetime of 2,300 x 60 x 60/19.5225 = ca. 5 days