Smart Sensor Architecture

Smart Sensor System Architecture in Mimosa project

Iiro Jantunen / Nokia Research Center

Tutorial Ambient Intelligence,
Toulouse, March 10, 2006
Contents

• What is a smart sensor system?
  • Transducer, analog electronics, ADC, digital electronics

• Open architecture for smart sensors
  • BluLite
  • Public SSI protocol
  • nanoUDP/nanoIP networking
  • Open APIs for developing applications for using smart sensors

• Mimosa architecture for smart sensors
  • Wired & wireless BluLite or RFID connection
  • Many sensors in many devices over many radio technologies
From sensors to smart sensors

Passive sensor (photodiode)

Sensor

Active sensor (ST 3-axis accelerometer)

Wireless smart sensor
Smart Sensor Systems

- Combine functions of sensors and interfaces
  - Sensing
  - Amplification
  - Signal conditioning
  - AD conversion
  - Bus interfacing

- Include higher level functions
  - Self-testing
  - Auto-calibration
  - Data processing and evaluation
  - Context awareness
  - Communications

- Modularity and/or integration

- Transducer
  - physical reality → electrical measurable

- Amplifier & filtering
  - sensor impedance, signal strength and quality

- Analog-digital conversion

- Microcontroller / DSP / ASIC
  - Digital signal processing
  - Communications to outside world

- Networking
  - Serial interface (USB, SPI, MMC)
  - Radio interface (optional)
Transducer

- Converts between physical properties
- The resulting property can be
  - Electrical capacitance
  - Current
  - Voltage
  - Optical power
- Electrical measurand needs an amplifier/filtering circuit to provide electrical power, impedance etc.
- Optical measurand needs a optoelectrical transducer with the same properties

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Typical/common techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration</td>
<td>Piezoelectric, capacitive</td>
</tr>
<tr>
<td>Displacement, position, proximity</td>
<td>Reluctance, optoelectronic, ultrasonic, radar</td>
</tr>
<tr>
<td>Flow</td>
<td>Pressure difference</td>
</tr>
<tr>
<td>Force</td>
<td>Piezoresistive</td>
</tr>
<tr>
<td>Humidity</td>
<td>Resistive, capacitive</td>
</tr>
<tr>
<td>Location</td>
<td>GPS</td>
</tr>
<tr>
<td>Time</td>
<td>Clock signal</td>
</tr>
<tr>
<td>Sound, pressure</td>
<td>Capacitive</td>
</tr>
<tr>
<td>Radiation</td>
<td>Optoelectronic</td>
</tr>
<tr>
<td>Gas concentration</td>
<td>Tuned laser &amp; optoelectronic</td>
</tr>
</tbody>
</table>
## Non-ideal behavior of sensors

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Sensor Design</th>
<th>Sensor Interface</th>
<th>MCU/DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonlinearity</td>
<td>Consistent</td>
<td></td>
<td>Reduce</td>
</tr>
<tr>
<td>Drift</td>
<td>Minimize</td>
<td></td>
<td>Compensate</td>
</tr>
<tr>
<td>Offset</td>
<td></td>
<td>Calibrate</td>
<td>Calibrate/reduce</td>
</tr>
<tr>
<td>Time dependence of offset</td>
<td>Minimize</td>
<td></td>
<td>Auto-zero</td>
</tr>
<tr>
<td>Time dependence of sensitivity</td>
<td></td>
<td></td>
<td>Auto-range</td>
</tr>
<tr>
<td>Nonrepeatability</td>
<td>Reduce</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross-sensitivity to temp and strain</td>
<td></td>
<td>Calibrate</td>
<td>Store value and correct</td>
</tr>
<tr>
<td>Hysteresis</td>
<td>Predictable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low resolution</td>
<td>Increase</td>
<td>Amplify</td>
<td></td>
</tr>
<tr>
<td>Low sensibility</td>
<td>Increase</td>
<td>Amplify</td>
<td></td>
</tr>
<tr>
<td>Unsuitable output impedance</td>
<td></td>
<td>Buffer</td>
<td></td>
</tr>
<tr>
<td>Self-heating</td>
<td>Increase cooling</td>
<td></td>
<td>Reduce power use</td>
</tr>
<tr>
<td>Unsuitable frequency response</td>
<td>Modify</td>
<td>Filter</td>
<td></td>
</tr>
<tr>
<td>Temperature dependence of offset</td>
<td></td>
<td></td>
<td>Store value and correct</td>
</tr>
<tr>
<td>Temp. dependence of sensitivity</td>
<td></td>
<td></td>
<td>Store value and correct</td>
</tr>
</tbody>
</table>

From R. Frank: *Understanding Smart Sensors*, 2nd ed.
Amplifying and analog filtering of sensor output

- Amplifier provides signal strength and sensor impedance
- Filtering needed to suppress noise
- If needed, quite complex functions can be done with analog electronics, but many functions are better done in digital electronics
  - easier
  - cheaper
  - need less room

Issues:
- Signal conditioning
- Signal transmission
- Data display
- Operating life
- Calibration
- Impedance of sensor and system
- Supply voltage
- Frequency response
- Filtering
Analog-digital conversion (ADC)

- The measurement must usually be changed to digital form for
  - further processing and calculations
  - storing to memory
  - sending the data
  - or just an economical reason
- ADC often included in MCU:s, e.g. MSP430F1xx
  - Is the resolution (8 or 12-bit) enough?

Issues:
- Sample rate
- Resolution (8, 12, 16-bit)
- Accuracy
- Power consumption
- Price
Digital signal processing

• More complex signal processing
  • summing many sensors (e.g. 3-axis accelerometers)
  • pattern recognition, e.g. step counters, speech recognition

• Can be done in
  • **ASIC**, for cheap mass-production
  • **DSP**, for high-speed number crunching
  • **MCU** (microcontroller), medium level calculations, control software

**Issues:**

• Fixed-point vs. floating-point DSP
• Data precision
• Speed
• Power usage
• Price
• Internal ADC
• Need for flexible programming: MCU vs. DSP
• Programming languages: Assembly vs. C/C++ or Java
Microcontroller

- Software controlling a smart sensor system, e.g.
  - measurements
  - communications
  - data memory
  - real time clock
- Usually includes
  - ADC (DAC)
  - timers
  - serial ports (SPI, I2C, UART)
  - flash memory
- Power-efficient, cheap, flexible
- Also called MCU or µC

MSP430F169 by Texas Instruments

- Low supply-voltage 1.8 - 3.6 V
- Ultra-low power consumption:
  - Active: 330 µA at 1 MHz, 2.2 V
  - Standby: 1.1 µA
  - Off (RAM retention): 0.2 µA
- Wake-up from standby in less than 6 µs
- 16-Bit RISC, 125-ns instruction cycle
- Program memory 60 kB (flash), RAM 2048 B
- 8-channel 12-Bit ADC with internal reference, sample-and-hold and autoscan
- Dual channel 12-Bit DAC with synchronization
- 16-Bit Timer_A & Timer_B with 3/7 CCR
- On-chip comparator
- 2 serial interfaces: UART or SPI or I2C™
- Supply voltage supervisor/monitor
- Brownout detector
- Bootstrap loader
- Serial onboard programming
Networking sensors over radio

- RFID
- Bluetooth
- UWB
- Low End
- WLAN
- Cellular

Internet

GPRS/3G network
BluLite (or Bluetooth Low End Extension)

- Low End Extension for Bluetooth is designed to complement Bluetooth by creating a wireless solution that allows small devices that are limited in battery power, size, weight and cost to have a wireless connection with mobile terminals without adding yet another radio to mobile terminals (as ZigBee).
- Optimized for irregular data exchange between Bluetooth enabled mobile terminals and button cell battery powered small devices.
- Concept assumes two device classes:
  - Dual-mode (Bluetooth 1.2 with Low End mode) for terminals
  - Stand-alone (Low End mode alone) for sensors and enhancements
BluLite radio

Channels of the proposed system

IEEE 802.11b channel in North America and Europe

Bluetooth channels

Low End Extension channels

One default initialization channel, non-overlapping with Bluetooth
Two secondary initialization channels, for jamming resistance
24 Unicast channels for user data
BluLite radio parameters with 1 Mbps

- Physical bit rate 1 Mbps
- Frequency band 2.4 GHz (ISM band)
- Duplex TDD
- Co-existence of multiple devices
  - Connection setup channel CSMA
  - Data delivery FDMA
- Jamming avoidance FDMA
- PDU payload
  - Byte aligned, variable length, max 255 bytes
- Bit rate excluding PHY and MAC overheads
  - Uni-directional with ARQ, max 890 kbps
  - Bi-directional with ARQ, max 2 x 471 kbps
Basic functions of BluLite radio

1. **ADVERTISE**: Makes the local device visible and connectable to all remote devices within reach. Low-power protocols optimized for this state. A possibility for an application dependent trade-off between connection set-up delay and the power consumption.

2. **SCAN**: Returns the addresses and short description of the advertising remote devices within reach.

3. **CONNECT**: Establishes a point-to-point connection with an advertising remote device.

4. **CONNECTED**: Provides point-to-point bi-directional data delivery with error detection, ARQ, segmentation, role switch and a low-activity mode. An evolution path to point-to-multipoint, requiring additions only in master capable devices.
Simple Sensor Interface (SSI) protocol

• A simple protocol for reading smart sensors over, e.g., BT LEE
• Also provided for RFID sensor tags
  • Memory map of sensor data on
  • Compatible with ISO 18000-4
• Support for multiple sensors on multiple devices
• Support for data polling or streaming
• Developed in co-operation with Suunto, Vaisala, Mermit, CEA-LETI, Oulu University and Ionific
• Development of the specification keeps backward compatibility (v0.4-)
• For specifications, source code and discussion forum, visit http://ssi-protocol.net

• Client-server architecture
  • Terminal has client software
  • Sensor unit has server software
• Client can
  • poll sensors
  • ask the server (Observer) to stream data to client (Listener)
  • read and modify the configuration of the server
• Commands have two forms
  • Capital letter ("N"), no checksum
  • Low case ("n"), the payload has a CRC checksum
## SSI v1.0 command base

<table>
<thead>
<tr>
<th>Command</th>
<th>Dir.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensor discovery</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q / q</td>
<td>0x51 / 0x71</td>
<td>→ Query</td>
</tr>
<tr>
<td>A / a</td>
<td>0x41 / 0x61</td>
<td>← Query reply</td>
</tr>
<tr>
<td>C / c</td>
<td>0x43 / 0x63</td>
<td>→ Discover sensors</td>
</tr>
<tr>
<td>N / n</td>
<td>0x4E / 0x6E</td>
<td>← Discovery reply</td>
</tr>
<tr>
<td><strong>Configuration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z / z</td>
<td>0x5A / 0x7A</td>
<td>→ Reset SSI device</td>
</tr>
<tr>
<td>G / g</td>
<td>0x47 / 0x67</td>
<td>→ Get configuration data for a sensor</td>
</tr>
<tr>
<td>X / x</td>
<td>0x58 / 0x78</td>
<td>← Configuration data response</td>
</tr>
<tr>
<td>S / s</td>
<td>0x53 / 0x73</td>
<td>→ Set configuration data for a sensor</td>
</tr>
<tr>
<td><strong>Read data</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R / r</td>
<td>0x52 / 0x72</td>
<td>→ Request sensor data</td>
</tr>
<tr>
<td>V / v</td>
<td>0x56 / 0x76</td>
<td>← Sensor data response</td>
</tr>
<tr>
<td>D / d</td>
<td>0x44 / 0x64</td>
<td>← Sensor data response with one byte status field</td>
</tr>
<tr>
<td><strong>Streaming data</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O / o</td>
<td>0x4F / 0x6F</td>
<td>→ Create sensor observer</td>
</tr>
<tr>
<td>Y / y</td>
<td>0x59 / 0x79</td>
<td>← Sensor observer created</td>
</tr>
<tr>
<td>K / k</td>
<td>0x4B / 0x6B</td>
<td>→ Delete sensor observer</td>
</tr>
<tr>
<td>L / l</td>
<td>0x4C / 0x6C</td>
<td>← Request sensor listener</td>
</tr>
<tr>
<td>J / j</td>
<td>0x4A / 0x6A</td>
<td>→ Sensor listener created</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E / e</td>
<td>0x45 / 0x65</td>
<td>† Error messages</td>
</tr>
<tr>
<td><strong>Free data</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F / f</td>
<td>0x46 / 0x66</td>
<td>† Free data for custom purposes</td>
</tr>
</tbody>
</table>
Example SSI command: Discovery reply

- Sensor device answers to Discovery command sent by a mobile terminal
- Discovery Reply message contains information about one or more sensors.
- Each sensor is identified with:
  - Sensor Id – 2 bytes
  - Description – 16 byte ASCII
  - Unit – 8 byte ASCII
  - Type – 1 byte
  - Scaler - signed 1 byte
  - Min – minimum sensor reading value
  - Max – maximum sensor reading value
- The reply can carry either one sensor per N command or many depending on buffer size
- UART or nanoIP frame provides the information about the length of a single message

<table>
<thead>
<tr>
<th>Addr</th>
<th>N/n</th>
<th>Sensor Id 1</th>
<th>Sensor desc.</th>
<th>Unit</th>
<th>Type</th>
<th>Scaler</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>16</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
nanoIP networking

- An open-source networking architecture
  - Minimal overheads
  - Wireless networking
  - Local addressing
- NanoIP makes use of the MAC address of underlying network technology rather than IP addresses
- Used with nanoUDP (User Datagram Protocol) or nanoTCP (Transmission Control Protocol)
  - nanoUDP does not provide reliability or ordering
  - nanoTCP provides retransmissions and flow control
- Mimosa uses nanoUDP
Open source SSI/nanoIP implementation

- SSI v0.4 over nanoUDP/nanoIP provided by Oulu University
- SSI v1.0 (nanoUDP/nanoIP) being finalized by Nokia Research Center, will be open source
- Works over BluLite (Bluetooth LEE)

- 1 byte protocol type (nanoIP)
- 4 bytes nanoUDP header
  - 2 byte payload + CRC length
  - 1 byte Source port (40 for SSI)
  - 1 byte Destination port (40 for SSI)
- n bytes
  - SSI payload
- 2 bytes
  - optional CRC checksum
Mimosa API’s for 3rd parties

- Back-end server
  - IP
  - Local MIMOSA SW Applications
    - UI_API
    - Context_API
    - Sensor_API
    - LC_API
    - MIMOSA Ambient User Interface Layer
    - MIMOSA Context Awareness Layer
    - MIMOSA Sensor Layer
    - MIMOSA Local Connectivity Layer

Java & C++ implementation
Sensor Management Board

- Controls the sensors (host)
- Controls the sensors and BTLE (Sensor Radio Node)
- Real time clock (with its own battery)
- Connectors for
  - LOCOS and/or CART
  - Sensor board (UART, SPI/I2C, general digital I/O, 8-channel analog)
  - JTAG programming interface
  - Debugging ports (UART)
  - IRQ ports
- MCU runs SSI/nanoIP and device drivers for sensors
Sensor Management Board

- Controls the sensors (host)
- Controls the sensors and BTLEE (Sensor Radio Node)
- Real time clock (with its own battery)
- Connectors for
  - LOCOS and/or CART
  - Sensor board (UART, SPI/I2C, general digital I/O, 8-channel analog)
  - JTAG programming interface
  - Debugging ports (UART)
  - IRQ ports
- MCU runs SSI/nanoIP and device drivers for sensors
Mimosa terminal

- Nokia 6630 phone as terminal
- Attached electronics (CART, LOCOS, SEMBO, RF) provide the Mimosa hardware functionality
- Java software on N6630 will provide the user interface, context awareness, sensor management etc.
- LOCOS board as motherboard of attached electronics
- SEMBO provides local sensor control
- SEPPPO (or other) sensor board connected to SEMBO with a standard interface
- RF part controlled with FPGA on LOCOS on baseband-module
- Separate RF module
Mimosa terminal - 2

Connection to RF board

Connection to phone

SEPPO

SEMBO

LOCOS

CART

Mimosa

Nokia
Sensor Radio Node

- Smart sensor which can be read over BT LEE connection
- EMMI board provides Bluetooth LEE for communications with mobile terminals
- SEMBO for controlling BT LEE, running the SSI server and sensor drivers
- LOCOS acts as motherboard
- SEPPPO (or other) as sensor hardware board via standard connector
  - UART
  - SPI / I2C
  - 8 analog channels
  - 15 unspecified digital I/O pins
# Sensors demonstrated on Mimosa platform

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Interface</th>
<th>Provider</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humidity, temperature, pressure</td>
<td>Digital</td>
<td>Nokia, (Sensirion, Intersema)</td>
<td>Weather station, SSI demonstration</td>
</tr>
<tr>
<td>Fat %</td>
<td>Amplified analog</td>
<td>Nokia</td>
<td>Fitness &amp; health</td>
</tr>
<tr>
<td>ECG</td>
<td>Amplified analog</td>
<td>Cardiplus</td>
<td>Fitness &amp; health, streaming data over SSI/nanoIP/BluLite</td>
</tr>
<tr>
<td>Lactate, glucose</td>
<td>Amplified analog</td>
<td>Fraunhofer-ISIT, Cardiplus, Åmic</td>
<td>Fitness &amp; health</td>
</tr>
<tr>
<td>Gyroscope</td>
<td>I2C</td>
<td>Fraunhofer-ISIT, CEA-LETI, STMicroelectronics</td>
<td></td>
</tr>
</tbody>
</table>
Weather station

- Demonstration of SSI use
- Weather information: Temperature, Humidity / dew point, and Pressure
- Intersema MS5534A pressure sensor
  - Pressure range 300-1100 mbar
  - Internal 15 Bit ADC
  - 6 coeff. software compensation on-chip
  - 3-wire serial interface
  - 1 system clock line (32.768 kHz)
  - 35 ms measurement time
- Sensirion SHT11 humidity sensor
Weather station

- Demonstration of SSI use
- Weather information: Temperature, Humidity / dew point, and Pressure
- Intersema MS5534A pressure sensor
- Sensirion SHT11 humidity sensor
  - Internal 14-bit ADC
  - Fully calibrated
  - Internal power regulation
  - Range
    - Humidity 0 – 100 %
    - Temperature -40 – 128°C
  - 2-wire serial interface (not I2C)
  - 11/55/210 ms for a 8/12/14bit measurement
Fat percentage

- Actually measures body water content (TBW)
- From TBW, fat-% and dehydration (ΔTBW) calculated
- 4-point measurement of impedance with 50 kHz signal
- Amplified analog output
  - Low-noise OPA2350 two-channel
- Also used for 2-point measurement of surface impedance, Galvanic Skin Response (GSR)
  - Skin stratum corneum humidity
  - Stress measurement
Lactate & glucose sensors

- Fraunhofer-Institut für Siziliziumtechnologie (Itzehoe, Germany) makes the sensor
- Åmic (Uppsala, Sweden) makes a needle array to penetrate outer layer of skin
- Measures the lactate or glucose content of interstitial fluid
- Lactate and glucose sensors differ only on the enzyme used
- Amplified analog connection to SEMBO
RFID sensor tag

- Virtual server on the terminal device answers to the Sensor API when connecting to RFID sensors.
- Virtual server translates the commands to the RFID reader to read a specified memory area of the tag.
- 0x53 (for ASCII “S”) in address 0x0C (Tag memory layout) defines the memory layout to be SSI compliant.
- Memory mapping designed for one or more sensors per tag.
- Memory layout designed to be convenient for SSI use.

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0x00 - 0x07</td>
<td>Tag ID</td>
</tr>
<tr>
<td>4</td>
<td>0x08 - 0x09</td>
<td>Tag manufacturer</td>
</tr>
<tr>
<td></td>
<td>0x0A - 0x0B</td>
<td>Tag hardware type</td>
</tr>
<tr>
<td>6</td>
<td>0x0C - 0x11</td>
<td>Tag memory layout. This defines tag to be SSI compliant sensor.</td>
</tr>
<tr>
<td></td>
<td>0x12 -</td>
<td>User data (layout defined by SSI).</td>
</tr>
</tbody>
</table>
## RFID memory mapping: Sensor data space

<table>
<thead>
<tr>
<th>Byte address</th>
<th>Field name</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x12 – 0x15</td>
<td>Sensor value</td>
<td>4 byte HEX</td>
<td>Variable sensor value</td>
<td>0x00000014</td>
</tr>
<tr>
<td>0x16</td>
<td>Type</td>
<td>1 b HEX</td>
<td>Describes the type of the Sensor value (0x00 = float, 0x01 = signed integer…)</td>
<td>0x01</td>
</tr>
<tr>
<td>0x17</td>
<td>Multiplier</td>
<td>1 b HEX</td>
<td>Sensor status. Bits 0 – 7 indicate if the sensor values are valid data (bit = 1) or not yet valid (bit = 0).</td>
<td>0x00</td>
</tr>
<tr>
<td>0x18</td>
<td>Status</td>
<td>1 b HEX</td>
<td>For future use. Could be, e.g., number of sensors.</td>
<td>0x01</td>
</tr>
<tr>
<td>0x19</td>
<td>Empty</td>
<td>1 b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1A – 0x29</td>
<td>Sensor description</td>
<td>16 b ASCII</td>
<td>Constant sensor description</td>
<td>“Temperature”</td>
</tr>
<tr>
<td>0x2A – 0x31</td>
<td>Unit</td>
<td>8 b ASCII</td>
<td>Constant unit description.</td>
<td>“C”</td>
</tr>
<tr>
<td>0x32 – 0x35</td>
<td>Minimum value</td>
<td>4 b HEX</td>
<td>Minimum value the sensor can provide.</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x36 – 0x39</td>
<td>Maximum value</td>
<td>4 b HEX</td>
<td>Maximum value the sensor can provide.</td>
<td>0x00000064</td>
</tr>
<tr>
<td>0x3A</td>
<td>Activate sensor</td>
<td>1 b</td>
<td>Bits 0 – 7 will be written by the reader to request the tag to write the sensor data</td>
<td>0x01</td>
</tr>
<tr>
<td>0x3B – 0x3D</td>
<td>Sensor control</td>
<td>3 b</td>
<td>Not defined yet. Could be, e.g., time needed before sensor value is valid.</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td>n bytes</td>
<td>Optional configuration data or 2nd sensor</td>
<td></td>
</tr>
</tbody>
</table>
Questions & Answers