WHY IS SENSOR DATA SO HARD TO GET?

CAROLINA FORTUNA
SENSORLAB, JOŽEF STEFAN INSTITUTE
JOŽEF STEFAN INSTITUTE

- founded in 1949
- leading Slovenian national research organization in the areas of natural sciences and technology:
  - > 500 researchers (total stuff > 800)
  - 200 on-going international projects, FP6 (>20 still running), FP7 (~30) and EURATOM (33)
OUTLINE

What is a “sensor”?  
Sensor nodes vs computing devices  
Sensor deployments  
Sensor nodes  
Middleware  
Sensor data and meta-data

SensorLab experience, technology and testbeds
WHAT IS A “SENSOR”? 

The meaning of the word varies across communities 

1. Physics/material science – refers to a material which changes its conductive properties according to a physical stimulus 

2. Electronics/computer networking – refers to a small device that features sensors, a processing unit and a communication module 

3. Computer science – refers to software/hardware artifacts which are able to observe something
WHAT IS A “SENSOR”?

In this talk we understand by:

- **sensor** a material or passive device which changes its conductive properties according to a physical stimulus
- **sensor node** a small device that features sensors, a processing unit and a communication module
SENSORY NODES VS COMPUTING DEVICES

Diminishing maintenance costs:

- Integrating sensors into personal computing devices such as phones/laptops
- Efficient remote configuration and management
- Disposable

What is a “sensor”? Sensor nodes vs computing devices

Sensor deployments

Sensor nodes

Middleware

Sensor (meta)data

SensorLab

Personal devices

Sensor nodes
The next revolution of the Internet is not going to be built on manual input of information by 500 million or a billion users. Rather, there is much greater potential in connecting computers to sensors so that valuable new information can be created automatically without human data entry. Much like the early days of the Internet ... the next generation of sensor networks can monitor our environment and deliver relevant information – automatically.

1 Pete Hartwell, How a Physically Aware Internet Will Change the World, Mashable, October 13, 2010.
ASPECTS OF THE PHYSICAL WEB

Web of Things - WoT (also SensorWeb or The Physical Web):
- integrating embedded devices to the Web using standards like HTML, XML, RSS …

Internet of Things - IoT:
- world-wide network of heterogeneous smart objects (sensors, actuators, RFID, MEMS …) based on standard communication protocols

Wireless Sensor Network - WSN:
- wireless network of spatially distributed nodes, which jointly observe certain phenomena
WIRELESS SENSOR NETWORKS (WSN)

W Communication links are established over radio channel (frequency, propagation, PHY)

S Sensing of physical information about environment (calibration, sampling, accuracy, processing)

N Sensed data is transferred to sink via network (routing, sleep, self-*)

Traditionally: low complexity, low power, small size/weight, long life, autonomous, short range, low cost …

WSN Node = Sensor + Microcontroller + Communication Module + Power Source
CURRENT STATE IN WSN AREA

Research
- Well developed field with many degrees of freedom
- Complex, large-scale, resource constrained systems
- Focus is on intra network communications

Development
- Solutions are tailored to specific application requirements
- Standard compliance (interoperability)
- Focus is on communication between WSN and Internet, to facilitate IoT/WoT

Business
- Slowly taking off (evolving standards, endless proprietary solutions multi-vendor interoperability)
- Interference and congestion in ISM bands
- Huge savings are promised (e.g. smart grids)
WIRELESS SENSOR NODES

Traditionally: low complexity, low power, small size and weight, long life, autonomous, short range, low cost ...

Standardized wireless communication protocols

- IEEE 802.15.4 - Low Rate WPAN (PHY & MAC)
- ZigBee, 6LoWPAN, WirelessHART, Dash7, Wavenis ...
- IEEE 802.15.1 – Bluetooth (BT 3.0 Low Energy Mode)
- IEEE 802.11x – Wi-Fi
- Other dedicated communication technologies (Ethernet, GPRS, ...) in WSN concept – gateways

Proprietary wireless communication protocols

- Z-Wave, ANT, MiWi, SimpliciTI, DigiMesh …
WHY WSNS ARE DIFFERENT?

- WSNs are destined to wide variety of applications.
- Asymmetric, highly directional information flow (data fusion).
- Energy is highly constrained.
- WSN may have huge amount of nodes.
- Application run-time is extremely long.
- Data aggregation (and network control) may be centralized, decentralized or hierarchical.
- Measurements reporting can be periodical, triggered by external event or on request by sink node.
PROGRAMING SENSOR NODES

- tool chain: IDE, compiler, debugger
- microcontroller is programmed and executes the code
- radio chip is not programmed, but controlled by microcontroller, usually via SPI which sets/reads registers
- compiled code is loaded to the microcontroller using bootloader or JTAG
- protocol stack may be precompiled and available through API or available as library
- operating system (not needed for simple tasks)
- virtual machine (optional)
Try to imagine a "world littered with trillions" of wireless sensors. Now try to imagine the *problems* getting even a few *thousand* of them to work together in any kind of *intelligible* way…¹

¹ John Cox, *Turning the world into a sensor network*, NetworkWorld, August 11, 2010.
## EUROPEAN DEPLOYMENTS

<table>
<thead>
<tr>
<th>Testbed</th>
<th>N</th>
<th>Sensors</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN</td>
<td>52</td>
<td>Temperature, Humidity, Visible light</td>
<td>ETH Zurich</td>
</tr>
<tr>
<td>Piaggio</td>
<td>40</td>
<td>Temperature, IR light, Ultrasound</td>
<td>Centro di Ricerca &quot;E. Piaggio&quot;</td>
</tr>
<tr>
<td>Powerbench</td>
<td>28</td>
<td>None</td>
<td>TU Delft</td>
</tr>
<tr>
<td>TWIST</td>
<td>204</td>
<td>Temperature, Humidity, Visible light</td>
<td>TU Berlin</td>
</tr>
<tr>
<td>UBoonn Testbed</td>
<td>30</td>
<td>Temperature, Humidity, Visible light, IR light</td>
<td>UBoonn</td>
</tr>
<tr>
<td>ART-WiSe</td>
<td>30</td>
<td>Temperature, Visible light, Sound, RSSI readings</td>
<td>ISEP-IPP</td>
</tr>
</tbody>
</table>

According to the [CONET Testbed Federation Survey, July 3 2010](http://example.com).
US DEPLOYMENTS

CMU: SensorAndrew*
- campus-wide testbed
- Firefly nodes
- Unknown scale

* According to web site on Oct 2010 and tech report from 2008.
US DEPLOYMENTS

Berkley: Motescope*
- Soda Hall, the Computer Science building
- Permanent testbeds for research, development and testing
- 78 Mica2DOT nodes

MIT: Senseable City Lab**
- Sensor nodes built into the wheels of bikes
- Unknown number

*According to web site visited on Oct 2010.

US DEPLOYMENTS

Harward, BBN: CitySense*

- 100 wireless sensors deployed across a city
- Sensor nodes are embedded PC, 802.11a/b/g interface, and various sensors for monitoring weather conditions and air pollutants
- open testbed

FEDERATION OF SENSOR DEPLOYMENTS

Pachube*
- 3700 sensor nodes, over 9400 data streams (April 2010)
- Sensor data and meta-data
- Open to upload/download

Sensorpedia*
- Similar to Pachube, limited testing Beta

Global Sensor Network*
- Framework for federated testbeds
- Used in the Swiss Experiment
- Closed

* According to web site visited on Oct 2010.
WHAT SENSORS ARE MOUNTED ON THE NODES?

- **Temperature**: 21%
- **N axis accelerometer**: 13%
- **Light**: 10%
- **RGB LED**: 8%
- **Humidity**: 6%
- **Pressure**: 5%
- **GPS**: 5%
- **Acoustic / sound**: 4%
- **Microphone**: 4%
- **Camera**: 4%
- **N axis magnetometer**: 4%
- **IrDA**: 2%
- **ECG**: 2%
- **Button**: 2%
- **Motion**: 1%
- **Photodiode**: 1%
- **Seismic**: 1%
- **Speaker**: 1%
- **Ultrasonic sound**: 1%
- **Vibration**: 1%
- **Switch**: 2%
- **Motion**: 1%
- **Switch**: 2%
- **Switch**: 2%
- **Switch**: 2%
- **Switch**: 2%
- **Switch**: 2%
- **Switch**: 2%
### WHAT TO SENSE?

<table>
<thead>
<tr>
<th>absorbed dose</th>
<th>current density</th>
<th>frequency</th>
<th>mass</th>
<th>sound intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>absorbed dose rate</td>
<td>direction</td>
<td>gesture</td>
<td>mass density</td>
<td>specific energy</td>
</tr>
<tr>
<td>acceleration</td>
<td>direction of motion</td>
<td>GPS info</td>
<td>molar energy</td>
<td>specific entropy</td>
</tr>
<tr>
<td>activity</td>
<td>dose equivalent</td>
<td>heart rate</td>
<td>molar entropy</td>
<td>specific heat capacity</td>
</tr>
<tr>
<td>alcohol</td>
<td>duration</td>
<td>heat capacity</td>
<td>molar heat capacity</td>
<td>specific volume</td>
</tr>
<tr>
<td>altitude</td>
<td>dynamic viscosity</td>
<td>heat flux density</td>
<td>moment of force</td>
<td>step frequency</td>
</tr>
<tr>
<td>amount of substance</td>
<td>electric charge</td>
<td>humidity</td>
<td>motion</td>
<td>stress</td>
</tr>
<tr>
<td>amount of substance</td>
<td>electric charge</td>
<td>illuminance</td>
<td>percentage</td>
<td>substance presence</td>
</tr>
<tr>
<td>concentration</td>
<td>electric charge</td>
<td>image</td>
<td>permeability</td>
<td>surface tension</td>
</tr>
<tr>
<td>angle</td>
<td>electric conductance</td>
<td>image</td>
<td>permeability</td>
<td>surface tension</td>
</tr>
<tr>
<td>angular acceleration</td>
<td>electric currency</td>
<td>inductance</td>
<td>permittivity</td>
<td>switch state</td>
</tr>
<tr>
<td>angular velocity</td>
<td>electric current</td>
<td>irradiance</td>
<td>plane angle</td>
<td>temperature</td>
</tr>
<tr>
<td>area battery charge</td>
<td>electric field strength</td>
<td>kerma</td>
<td>power</td>
<td>thermal conductivity</td>
</tr>
<tr>
<td>blood glucose level</td>
<td>electric flux density</td>
<td>length</td>
<td>presence</td>
<td>time</td>
</tr>
<tr>
<td>blood oxygen level</td>
<td>electric potential</td>
<td>location</td>
<td>pressure</td>
<td>torque</td>
</tr>
<tr>
<td>blood pressure</td>
<td>difference</td>
<td>luminance</td>
<td>proximity</td>
<td>velocity</td>
</tr>
<tr>
<td>body fat percentage</td>
<td>energy density</td>
<td>luminous flux</td>
<td>quantity of heat</td>
<td>volume</td>
</tr>
<tr>
<td>capacitance</td>
<td>entropy</td>
<td>luminous intensity</td>
<td>radiance</td>
<td>wave number</td>
</tr>
<tr>
<td>catalytic activity</td>
<td>exposure</td>
<td>magnetic field strength</td>
<td>radiant flux</td>
<td>wind speed</td>
</tr>
<tr>
<td>Application / (Meta) Data</td>
<td>JSI tested: Cyc KB RDF, OWL, CycL Jena, Racer, Cyc Weka, PISA</td>
<td>JSI preferred: CycKB RDF, OWL, CycL JSI analytic tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Existing technology:</td>
<td>Possible technologies</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CSIRO, CESN, W3C SSN, Ontosensor… XML, RDF, RDF-S, OWL, SensorML … Jena, Racer, Cyc Weka, PISA</td>
<td>None for now</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSI tested:</td>
<td>Squawk, .NET micro, Darjeeling</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSI preferred:</td>
<td>TinyOS, Contiki, CooCox</td>
<td>Versatile Sensor Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtual Machine</td>
<td></td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Existing technology:</td>
<td>Possible technologies</td>
<td>Custom code, Contiki</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mate, VM*, SwissQM, CVM, …</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSI tested:</td>
<td></td>
<td>Crossbow Imote2, Mote Works, eKo kit, TI eZ430-RF2500, Microchip PICDEM Z, SunSPOT, Arduino/ Libelium (Xbee), proprietary HW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSI preferred:</td>
<td></td>
<td>Versatile Sensor Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td></td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Existing technology:</td>
<td>Possible technologies</td>
<td>Custom code, Contiki</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MantisOS, FreeRTOS, eCOS, uOS, RETOS, SOS, …</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSI tested:</td>
<td></td>
<td>Crossbow Imote2, Mote Works, eKo kit, TI eZ430-RF2500, Microchip PICDEM Z, SunSPOT, Arduino/ Libelium (Xbee), proprietary HW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSI preferred:</td>
<td></td>
<td>Versatile Sensor Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Node</td>
<td></td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Existing technology:</td>
<td>Possible technologies</td>
<td>Versatile Sensor Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mica, FireFly, WASPMote, BTNode, SquidBee, Iris, Cricket, Ember, Medusa MK-2, BSN, Stargate, Sensinode, PushPin, SpotON, …</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSI tested:</td>
<td></td>
<td>Crossbow Imote2, Mote Works, eKo kit, TI eZ430-RF2500, Microchip PICDEM Z, SunSPOT, Arduino/ Libelium (Xbee), proprietary HW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSI preferred:</td>
<td></td>
<td>Versatile Sensor Node</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## COMPONENTS AND CLASSIFICATION

- Sensors + Microcontroller + Communication Module + Power Source

### Classification:
- adapted/augmented general-purpose computers
- embedded sensor modules
- system on chip (SoC) solutions

<table>
<thead>
<tr>
<th>Power Source</th>
<th>Comm. Module</th>
<th>CPU &amp; Memory</th>
<th>Sensor (Actuator)</th>
</tr>
</thead>
</table>
CLASSIFICATION BY PROCESSING AND COMMUNICATION CAPABILITY

- 125 types of sensor nodes identified in research community
- 118 types of processing units
  - the most popular microcontroller (Atmel ATmega128L, 8 bits)
  - the most popular DSP (Coolflux DSP NxAH1200, 24 bits)
  - FPGAs are exceptions (Xilinx XC3S200 Spartan-III)
- 112 types of communication interfaces
  - the most popular radio frequency communication module (TI / Chipcon CC2420, 2.4 GHz)
  - the most popular optical communication module (IrDa transceivers)

Overview as of May 2010
CLASSIFICATION BY PROCESSING UNITS

- Most of the nodes feature one pre-determined processing unit (Mica mote)

- multiple processing units (UCLA Medusa MK-2)
CLASSIFICATION BY CLOCK FREQUENCY AND MEMORY

According to observed clock frequencies, Flash, and ROM memories processing units can be divided into:

- average and low capability units (BSN node)
- more advanced units (basically based on ARM architecture) (Intel PXAx)

<table>
<thead>
<tr>
<th>Clock frequency</th>
<th>Flash and ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>average and low capability</td>
<td>4-32 MHz</td>
</tr>
<tr>
<td>advanced units</td>
<td>32-600 MHz</td>
</tr>
</tbody>
</table>
CLASSIFICATION BY COMMUNICATION SPEED

- **high data-rates (i.e. > 1000 bps) (Stargate)**
  - IEEE 802.11 units
  - Adapted general-purpose computers (Wi-Fi routers)
    - usually constant power supply
  - adapted general-purpose computers (smartphones) and embedded sensor nodes
    - rapidly consume battery power
- **medium data-rates (i.e. ~1000 bps) (Smart-its)**
  - Bluetooth radios
    - usually do not require constant power supply and can last on battery power for a while
- **low data-rates (i.e. ~100-250 bps) (Sentio)**
  - data-rates lower than 100 bps (PushPin)
  - data-rates of only few bps (SpotON)
  - dedicated RF interfaces

Overview as of May 2010
CLASSIFICATION BY ANTENNA

- 44% external antenna (whip),
- 36% onboard antennas (PCB, chip),
- 17% external and onboard antennas at the same time,
- 3% of the nodes do not have antenna, but are equipped with optical or wired communication interfaces.
WHICH PROCESSORS AND HOW ADVANCED?

Processing units
Overview as of May 2010
WHICH COMMUNICATION MODULES AND FREQUENCIES?

- CC24x: 26%
- CC10x: 18%
- Bluetooth: 14%
- TR1000: 11%
- Xbeex: 7%
- nRFx: 5%
- TR1001: 3%
- ROK10100x: 3%
- AT86RF230: 3%
- 802.11x: 4%
- wired serial connection: 3%

Frequency bands distribution:
- 315 MHz
- 433 MHz
- 868 MHz
- 915 MHz
- 2.4 GHz

Overview as of May 2010
<table>
<thead>
<tr>
<th>Sensor Node</th>
<th>Operating System</th>
<th>Virtual Machine</th>
<th>Application / (Meta) Data</th>
<th>JSI tested</th>
<th>JSI preferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing technology: Mica, FireFly, WASPMote, BTNode, SquidBee, Iris, Cricket, Ember, Medusa MK-2, BSN, Stargate, Sensinode, PushPin, SpotON, …</td>
<td>MantisOS, FreeRTOS, eCOS, uOS, RETOS, SOS, …</td>
<td>Mate, VM*, SwissQM, CVM, …</td>
<td>CSIRO, CESN, W3C SSN, Ontosensor… XML, RDF, RDF-S, OWL, SensorML … Jena, Racer, Cyc, Weka, PISA, …</td>
<td>Cyc KB RDF, OWL, CycL Jena, Racer, Cyc Weka, PISA</td>
<td>Versatile Sensor Node</td>
</tr>
<tr>
<td>Existing technology: Crossbow Imote2, Mote Works, eKo kit, TI eZ430-RF2500, Microchip PICDEM Z, SunSPOT, Arduino/Libelium (Xbee), proprietary HW</td>
<td>JSI tested: Squawk, .NET micro, Darjeeling</td>
<td>JSI preferred: None for now</td>
<td>JSI tested: TinyOS, Contiki, CooCox</td>
<td>JSI preferred: Custom code, Contiki</td>
<td>JSI preferred: Versatile Sensor Node</td>
</tr>
<tr>
<td>Sensor Node</td>
<td>Existing technology:</td>
<td>JSI tested:</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------</td>
<td>-------------</td>
<td>----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mica, FireFly, WASPMote, BTNode, SquidBee, Iris, Cricket, Ember, Medusa, MK-2, BSN, Stargate, Sensinode, PushPin, SpotON, ...</td>
<td>Cyc KB, RDF, OWL, CycL, Jena, Racer, Cyc, Weka, PISA</td>
<td>None for now</td>
<td>Versatile Sensor Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Existing technology:</td>
<td>JSI tested:</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MantisOS, FreeRTOS, eCOS, uOS, RETOS, SOS, ...</td>
<td>Squawk, .NET micro, Darjeeling</td>
<td>Custom code, Contiki</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtual Machine</td>
<td>Existing technology:</td>
<td>JSI tested:</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mate, VM*, SwissQM, CVM, ...</td>
<td>TinyOS, Contiki, CooCox</td>
<td>None for now</td>
<td>Contiki</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td>Existing technology:</td>
<td>JSI tested:</td>
<td>JSI preferred:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCSIRO, CESN, W3C SSN, Ontosensor, XML, RDF, RDF-S, OWL, SensorML ... Jena, Racer, Cyc, Weka, PISA, ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application / (Meta) Data</td>
<td>Existing technology:</td>
<td>JSI tested:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CycKB RDF, OWL, CycL Jena, Racer, Cyc, Weka, PISA</td>
<td>Squawk, .NET micro, Darjeeling</td>
<td>None for now</td>
<td>Versatile Sensor Node</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MIDDLEWARE

Is a software layer which provides an abstraction of whatever lies under for use by whatever lies above.

Under the middleware can lay hardware resources of various types or software resources such as firmware, drivers and sometimes also the operating system.

“The main purpose of middleware for sensor networks is to support the development, maintenance, deployment, and execution of sensing-based applications”\(^1\)

EMBEDDED OPERATING SYSTEM

- OS running on limited devices with restricted capabilities
- Restricted to narrow applications
  - industrial controllers, robots, networking gear, gaming consoles, metering, sensor nodes…
- Architecture and purpose of embedded OS changes as the hardware capabilities change (i.e. mobile phones)

What is a “sensor”? Sensor nodes vs computing devices Sensor deployments Sensor nodes Middleware Sensor (meta)data SensorLab
CLASSIFICATION

Heavy-weight processes with memory protection

Lightweight multi-threading, preemption

Event-driven system

Simple control loop

Original, untested

Research, testbeds

Commercial, Industrial system

What is a “sensor”?

Sensor nodes vs computing devices

Sensor deployments

Sensor nodes

Middleware

Sensor (meta)data

SensorLab
MAIN DESIGN FEATURES OF EMBEDDED OS

- Scheduling
- Memory management
- Kernel
- (Re)Programming
- Footprint

Additionally, when selecting which one to use:
- Protocol stack support
- Virtual machine support
- Development status
- Documentation and community support
**EMBEDDED OS CLASSIFICATION BY SCHEDULING MODEL**

- **Event driven model (Contiki, TinyOS, SOS)**
  - No locking - only one event running at a time
  - One stack – reused for every event handler
  - Requires less memory
  - Synchronous vs. asynchronous events

- **Thread driven model (FreeRTOS, eCOS, Nut/OS, eCOS)**
  - Each thread has its own stack
  - Thread stacks allocated at creation time (Unused stack space wastes memory)
  - Locking mechanisms - to prevent modifying shared resources
EMBEDDED OS CLASSIFICATION BY SYSTEM IMAGE

- **Monolithic (TinyOS, FreeRTOS, eCOS, uC/OS-II, Nut/OS)**
  - One system image: (kernel) + modules + application compiled together
  - Efficient execution environment (optimization at compilation)
  - High energy costs for updating

- **Modular (Contiki, SOS)**
  - Static image: (kernel) + loadable component images
  - Lower execution efficiency (no global optimization at compilation time)
  - Updates are less expensive (smaller size) - energy and time

What is a “sensor”? Sensor nodes vs computing devices | Sensor nodes | Middleware | Sensor (meta)data | SensorLab
## Embedded OS Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>eCOS</td>
<td>Thread, preempt</td>
<td>Multiple stacks, static</td>
<td>Yes</td>
<td>Monolithic, no</td>
<td>variable</td>
<td>lwIP, TCP/IP</td>
<td>FreeBSD</td>
<td>(yes)</td>
<td>Yes/(yes)</td>
<td>yes</td>
</tr>
<tr>
<td>uC/OS-II</td>
<td>Thread, preempt</td>
<td>Multiple stacks, static</td>
<td>Yes</td>
<td>(Monolithic, no)</td>
<td>variable</td>
<td>uC/TCP-IP</td>
<td>(no)</td>
<td>Yes/(yes)</td>
<td>limited</td>
<td>yes</td>
</tr>
<tr>
<td>FreeRTOS</td>
<td>Thread, preempt</td>
<td>(Multiple stacks, static)</td>
<td>Yes</td>
<td>(Monolithic, no)</td>
<td>variable</td>
<td>lwIP</td>
<td>(no)</td>
<td>Yes/(yes)</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Nut/OS</td>
<td>Thread, preempt</td>
<td>Multiple stack, Dynamic</td>
<td>Yes</td>
<td>Monolithic, (no)</td>
<td>variable</td>
<td>BTNut, Nut/Net (TCP/IP)</td>
<td>(no)</td>
<td>Yes/(yes)</td>
<td>limited</td>
<td>yes</td>
</tr>
<tr>
<td>TinyOS</td>
<td>Event, (thread)</td>
<td>Single stack, Static</td>
<td>No</td>
<td>Monolithic, wireless</td>
<td>variable</td>
<td>CC100, CC2420, TinyBt, serial</td>
<td>yes</td>
<td>Yes/yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SOS</td>
<td>Event</td>
<td>Single, dynamic</td>
<td>Yes</td>
<td>Modular,</td>
<td>variable</td>
<td>message</td>
<td>(no)</td>
<td>No/no</td>
<td>limited</td>
<td>yes</td>
</tr>
<tr>
<td>Contiki</td>
<td>Event, thread</td>
<td>Yes</td>
<td>Modular, wireless</td>
<td>variable</td>
<td>yes</td>
<td>Yes/(no)</td>
<td>yes</td>
<td>Yes/yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
PROTOCOL STACK

Communication protocol

- defines the functions that have to be implemented and services that have to be provided by the parties involved in the information exchange.

In computer and sensor networks, protocols are organized as a stack and the number of layers in the stack is standard specific.

Communication protocol standards for sensor networks

- IEEE802.15.4, ZigBee, ISA100.11a, WirelessHART, 6LoWPAN, Bluetooth, IEEE802.11
## PROTOCOL STACK IMPLEMENTATION

Software (within the OS) | On microcontroller | Hybrid
---|---|---

<table>
<thead>
<tr>
<th>Communication Standard</th>
<th>Protocol Stack Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE 802.15.4</td>
<td>“Implementation of IEEE 802.15.4 protocol stack for Linux”</td>
</tr>
<tr>
<td>ZigBee</td>
<td>Z-Stack, Open-ZB, FreakZ, Microchip Stack</td>
</tr>
<tr>
<td>6LoWPAN</td>
<td>NanoStack2.0, Mantus, μIPv6, BLIP</td>
</tr>
<tr>
<td>IEEE 802.11</td>
<td>smxWiFi</td>
</tr>
<tr>
<td>ISA100.11a</td>
<td>NISA100.11a</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>TinyBT, Axis OpenBT, BlueZ, Affix</td>
</tr>
</tbody>
</table>
VIRTUAL MACHINE

A virtual machine is a software implementation of a machine and provides a level of abstraction over the physical machine.

VM for embedded systems
- Replace the operating system
- Add extra functionality to the operating system (memory management)
- Provide a friendlier application development environment
CLASSIFICATION OF VIRTUAL MACHINES

- System VM
  - virtualize hardware resources and can run directly on hardware.
  - In embedded systems they implement functions of the OS and completely replace it
  - Squawk, .NET Micro

- Application (process) VM
  - typically run on top of an OS as an application and support a single process.
  - Mate, Darjeeling, VM*, SwissQM, CVM, DVM
## VIRTUAL MACHINE COMPARISON

<table>
<thead>
<tr>
<th>Name</th>
<th>OS</th>
<th>ASVM</th>
<th>Platform</th>
<th>Memory</th>
<th>Multi thread</th>
<th>Supported PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mate</td>
<td>TinyOS</td>
<td>no</td>
<td>Rene2 and Mica</td>
<td>1KB RAM 16KB ROM 7.5KB ROM, 600B RAM</td>
<td>yes</td>
<td>TinyScript</td>
</tr>
<tr>
<td>.NET Micro</td>
<td>With/without OS TinyOS</td>
<td>No (?)</td>
<td>Imote2</td>
<td>300KB RAM 512MB of flash memory</td>
<td>yes</td>
<td>C#</td>
</tr>
<tr>
<td>Darjeeling</td>
<td>TinyOS, Contiki, FOS</td>
<td>No (?)</td>
<td>Tnode, Tmote Sky, Fleck3/Fleck3B</td>
<td>2K RAM</td>
<td>Yes</td>
<td>Java subset</td>
</tr>
<tr>
<td>Squawk</td>
<td>With (Solaris, Windows, MAC OS X, linux systems) /without</td>
<td>Yes (?)</td>
<td>SunSpots Java Card 3.0</td>
<td>Core 80KB RAM Libraries: 270 Kb flash</td>
<td>yes</td>
<td>Java mostly</td>
</tr>
<tr>
<td>VM*</td>
<td>OS*</td>
<td>yes</td>
<td>Mica, ongoing work: Telos, XYZ, Stargate and handheld devices</td>
<td>6kb code 200 bytes data (depends on the app req)</td>
<td>no</td>
<td>Java</td>
</tr>
<tr>
<td>SwissQM</td>
<td>TinyOS</td>
<td>Yes</td>
<td>Mica2 and tmote sky</td>
<td>33kb flash and 3kb SRAM(on Mica2)</td>
<td>Yes</td>
<td>Subset of Java (37 instructions + 22 specific)</td>
</tr>
<tr>
<td>CVM</td>
<td>Contiki</td>
<td></td>
<td></td>
<td>8 RAM 1344 ROM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ARE VIRTUAL MACHINES NECESSARY?

- Trade-off between the resources needed and the services they provide

- Advantage
  - Reduce the distribution energy costs for software updates
    - VM code smaller than native machine code
    - Simpler reprogramming process

- Disadvantage
  - Additional overhead
    - Increased time and memory requirements for execution
    - Increased energy spent in interpreting the code
<table>
<thead>
<tr>
<th>Application / (Meta) Data</th>
<th>Operating System</th>
<th>Virtual Machine</th>
<th>Sensor Node</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Existing technology:</strong></td>
<td><strong>JSI tested:</strong></td>
<td><strong>JSI preferred:</strong></td>
<td><strong>JSI preferred:</strong></td>
</tr>
<tr>
<td>CSIRO, CESN, W3C</td>
<td>Cyc KB</td>
<td>CycKB</td>
<td>Versatile Sensor Node</td>
</tr>
<tr>
<td>SSN, Ontosensor...</td>
<td>RDF, OWL, CycL</td>
<td>RDF, OWL, CycL</td>
<td>Sensor Node</td>
</tr>
<tr>
<td>XML, RDF, RDF-S, OWL,</td>
<td>Jena, Racer, Cyc</td>
<td>Jena, Racer, Cyc</td>
<td></td>
</tr>
<tr>
<td>SensorML ...</td>
<td>Weka, PISA</td>
<td>Weka, PISA</td>
<td></td>
</tr>
<tr>
<td>Jena, Racer, Cyc,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weka, PISA, ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Existing technology:</strong></td>
<td><strong>JSI tested:</strong></td>
<td><strong>JSI preferred:</strong></td>
<td></td>
</tr>
<tr>
<td>Mate, VM*, SwissQM,</td>
<td>Squawk, .NET</td>
<td>None for now</td>
<td></td>
</tr>
<tr>
<td>CVM, ...</td>
<td>micro, Darjeeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Existing technology:</strong></td>
<td><strong>JSI tested:</strong></td>
<td><strong>JSI preferred:</strong></td>
<td></td>
</tr>
<tr>
<td>MantisOS, FreeRTOS,</td>
<td>TinyOS, Contiki,</td>
<td>Custom code,</td>
<td></td>
</tr>
<tr>
<td>eCOS, uOS, RETOS, SOS, ...</td>
<td>CooCox</td>
<td>Contiki</td>
<td></td>
</tr>
<tr>
<td><strong>Existing technology:</strong></td>
<td><strong>JSI tested:</strong></td>
<td><strong>JSI preferred:</strong></td>
<td></td>
</tr>
<tr>
<td>Mica, FireFly, WASPMote,</td>
<td>Crossbow Imote2,</td>
<td>Versatile</td>
<td></td>
</tr>
<tr>
<td>BTNode, SquidBee, Iris,</td>
<td>Mote Works, eKo kit,</td>
<td>Sensor Node</td>
<td></td>
</tr>
<tr>
<td>Cricket, Ember, Medusa</td>
<td>TI eZ430-RF2500,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MK-2, BSN, Stargate,</td>
<td>Microchip PICDEM Z,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensinode, PushPin,</td>
<td>SunSPOT, Arduino/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpotON, ...</td>
<td>Libelium (Xbee),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>proprietary HW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Application / (Meta) Data

**Existing technology:**
- CSIRO, CESN, W3C
- SSN, Ontosensor...
- XML, RDF, RDF-S, OWL, SensorML ...
- Jena, Racer, Cyc, Weka, PISA, ...

### Virtual Machine

**Existing technology:**
- Mate, VM*, SwissQM, CVM, ...

### Operating System

**Existing technology:**
- MantisOS, FreeRTOS, eCOS, uOS, RETOS, SOS, ...

### Sensor Node

**Existing technology:**
- Mica, FireFly, WASPMote, BTNode, SquidBee, Iris, Cricket, Ember, Medusa MK-2, BSN, Stargate, Sensinode, PushPin, SpotON, ...

### JSI tested:

- Cyc KB
- RDF, OWL, CycL
- Jena, Racer, Cyc
- Weka, PISA

### JSI preferred:

- CycKB
- RDF, OWL, CycL
- JSI analytic tools

### JSI tested:

- Squawk, .NET micro, Darjeeling

### JSI preferred:

- None for now

### JSI tested:

- TinyOS, Contiki, CooCox

### JSI preferred:

- Custom code, Contiki

### JSI tested:

- Crossbow Imote2, Mote Works, eKo kit, TI eZ430-RF2500, Microchip PICDEM Z, SunSPOT, Arduino/ Libelium (Xbee), proprietary HW

### JSI preferred:

- Versatile Sensor Node
SENSOR DATA AND META-DATA

Sensor data
- Measurements performed by sensors (pressure, luminance, noise, CO, CO2, …)
- Most often a stream of data (time series: value + time stamp)
- Sampling and transmission rates influence whether the data is real-time or not

Sensor metadata
- Data about the sensor nodes or the sensor network
- Providing context for data
- Enables machines to understand and reason
SENSOR DATA AND META-DATA

Sensor data
- Collection is done automatically by sensor nodes
- Processing and/or storage is performed on computing devices (centralized servers)

Sensor meta-data
- Collection is done mostly manually by operators
- Processing and storage is typically done on computing devices (centralized servers)
SENSOR DATA AND META-DATA

Sensor meta-data

- Can be organized in several ways, depending on the complexity of the application

- Design issues when building a meta-data management system
  - Vocabulary
  - Representation language
  - Processing engine
VOCABULARY

- A set of terms (the vocabulary in Pachube)

- A well defined set of words, perhaps even a taxonomy (SensorML)

- Ontology (CSIRO, Ontosensor)
  - “a formal, explicit specification of a shared conceptualization” (T. Gruber)

- Components:
  - Set of concepts
  - Set of relationships
  - Set of instances

- Based on the level of generality:
  - Domain ontologies
  - Upper ontologies
SENSOR ONTOLOGIES

- **CSIRO**
  - technical aspects (calibration, temporal resolution-sampling frequency, accuracy, what it measures)
  - information about access to the sensor for control and configuration
  - Structured into four clusters (groups) of concepts (Sensor, Feature, SensorGrounding, OperationModel)

- **OntoSensor**
  - map hierarchical concepts form SensorML to OWL
  - Includes concepts from SUMO and ISO 19155

- **CESN**
  - Uses Sensor ML vocabulary
  - Utilized for an application capable to reason about coastal storm

- **W3C SSN Incubator Group**
## REPRESENTATION LANGUAGES

<table>
<thead>
<tr>
<th>OWL2</th>
<th>OWL2 RL</th>
<th>OWL2 QL</th>
<th>OWL2 EL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lite</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OWL2**
- Defines vocabulary
- Organize Vocabulary in typed hierarchies
- Large no of classes and properties
- Losses tractability
- A class can be both collection and individual
- Maximal expressivity (while maintaining tractability)
- Classification hierarchies, simple constraints
- Property chaining, asymmetric, reflexive, disjoint properties
- Full: Large no of classes and properties
- Lite: Very large ontologies
- EL: Loses tractability
- QL: Large no. of individuals
- RL: Very large ontologies

**SensorML**
- Specific domain: sensors
- Sensor nodes vs computing devices
- Sensor deployments
- Sensor nodes
- Middleware
- (meta)data
- SensorLab

### What is a “sensor”?
- Sensor nodes vs computing devices
- Sensor deployments
- Sensor nodes
- Middleware
- (meta)data
- SensorLab

**Data Model**
- Graph, building block - triple
- Syntax

**Classification hierarchies, simple constraints**

**Property chaining**, asymmetric, reflexive, disjoint properties
PROCESSING ENGINE

Typically based on description logic

<table>
<thead>
<tr>
<th>Processor</th>
<th>Memory used on running</th>
<th>Memory used for the CSIRO ontology</th>
<th>Memory used for the Wine ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pellet</td>
<td>~11 Mb</td>
<td>~25 Mb</td>
<td>~34 Mb</td>
</tr>
<tr>
<td>Racer-Pro</td>
<td>~10 Mb</td>
<td>~16 Mb</td>
<td>~21 Mb</td>
</tr>
<tr>
<td>FaCT++</td>
<td>~20 Mb</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bossam</td>
<td>~9.2 Mb</td>
<td>~21 Mb</td>
<td>~90 Mb</td>
</tr>
<tr>
<td>Hoolet</td>
<td>~22 Mb</td>
<td>~28 Mb</td>
<td>~31 Mb</td>
</tr>
<tr>
<td>Cyc</td>
<td>~470 Mb</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
SEMANTIC SENSOR WEB

Knowledge Base
Ontology
Logic Rules

Semantic Annotations

Semantic Sensor Web

Inference Engine

Web Service
Publishers

Sensor Descriptions
Sensor Data
DETERMINING THE LEVEL OF COMFORT

Sensors measuring temperature, humidity and wind speed are located in the same place (e.g. a park).
DETERMINING THE LEVEL OF COMFORT

Rule:

\[
\text{TemperatureSensor}(\text{?S1}), \text{HumiditySensor}(\text{?S2}), \text{WindSpeedSensor}(\text{?S3}), \text{hasLocation}(\text{?S1}, \text{?L}), \\
\text{hasLocation}(\text{?S2}, \text{?L}), \text{hasLocation}(\text{?S3}, \text{?L}), \\
\rightarrow \\
\text{ComfortLevelSensor}(\text{?newVirtualSensor}), \text{hasLocation}(\text{?newVirtualSensor}, \text{?L})
\]

Query:

What are the sensors measuring comfort level at a specific location?

- without having to have the knowledge about all the other sensors needed for computing comfort level
<table>
<thead>
<tr>
<th>Application / (Meta) Data</th>
<th>Existing technology:</th>
<th>JSI tested:</th>
<th>JSI preferred:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CSIRO, CESN, W3C</td>
<td>Cyc KB</td>
<td>CycKB</td>
</tr>
<tr>
<td></td>
<td>SSN, Ontosensor…</td>
<td>RDF, OWL,</td>
<td>RDF, OWL, CycL</td>
</tr>
<tr>
<td></td>
<td>XML, RDF, RDF-S, OWL</td>
<td>CycL</td>
<td>CycL</td>
</tr>
<tr>
<td></td>
<td>SensorML …</td>
<td>Jena, Racer</td>
<td>None for now</td>
</tr>
<tr>
<td></td>
<td>Jena, Racer, Cyc,</td>
<td>Cyc</td>
<td>Custom code,</td>
</tr>
<tr>
<td></td>
<td>Weka, PISA, …</td>
<td>Weka, PISA</td>
<td>Contiki</td>
</tr>
<tr>
<td>Virtual Machine</td>
<td>Existing technology:</td>
<td>JSI tested:</td>
<td>JSI preferred:</td>
</tr>
<tr>
<td></td>
<td>Mate, VM*, SwissQM,</td>
<td>Squawk,.NET</td>
<td>Versatile Sensor Node</td>
</tr>
<tr>
<td></td>
<td>CVM, …</td>
<td>micro, Darjeeling</td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td>Existing technology:</td>
<td>JSI tested:</td>
<td>JSI preferred:</td>
</tr>
<tr>
<td></td>
<td>MantisOS, FreeRTOS,</td>
<td>TinyOS, Contiki,</td>
<td>Custom code,</td>
</tr>
<tr>
<td></td>
<td>eCOS, uOS, RETOS,</td>
<td>CooCox</td>
<td>Contiki</td>
</tr>
<tr>
<td></td>
<td>SOS, …</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Node</td>
<td>Existing technology:</td>
<td>JSI tested:</td>
<td>JSI preferred:</td>
</tr>
<tr>
<td></td>
<td>Mica, FireFly, WASPMote,</td>
<td>Crossbow Imote2,</td>
<td>Crossbow Imote2,</td>
</tr>
<tr>
<td></td>
<td>BTNode, SquidBee, Iris</td>
<td>Mote Works, eKo kit,</td>
<td>Mote Works, eKo kit,</td>
</tr>
<tr>
<td></td>
<td>Cricket, Ember, Medusa</td>
<td>TI eZ430-RF2500,</td>
<td>TI eZ430-RF2500,</td>
</tr>
<tr>
<td></td>
<td>MK-2, BSN, Stargate,</td>
<td>Microchip PICDEM Z,</td>
<td>Microchip PICDEM Z,</td>
</tr>
<tr>
<td></td>
<td>Sensinode, PushPin,</td>
<td>SunSPOT, Arduino/</td>
<td>SunSPOT, Arduino/</td>
</tr>
<tr>
<td></td>
<td>SpotON, …</td>
<td>Libelium (Xbee),</td>
<td>Libelium (Xbee),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>proprietary HW</td>
<td>proprietary HW</td>
</tr>
</tbody>
</table>
Commercial out-of-the-box solutions

- Crossbow Imote2, Mote Works, eKo kit
- TI eZ430-RF2500
- Microchip PICDEM Z
- SunSPOT
- Arduino/Libelium (Xbee)
BLUETOOTH AND SMART PHONE IN WSN

Bluetooth link between sensor node and smart phone
Video and voice capturing
Temperature, humidity, pressure, light monitoring
Proprietary Hardware
Java test application
HIERARCHICAL WSN FOR ENVIRONMENTAL MONITORING

Temperature and humidity monitoring

ZigBee based local sensor networks

GSM/GPRS interconnection with control center

Proprietary Hardware

Joomla extension WEB interface

• Customised data export (Chart, Table, GoogleMaps, XML, Database)

What is a “sensor”?

Sensor nodes vs computing devices

Sensor deployments

Sensor nodes

Middleware

Sensor (meta)data

SensorLab
REMOTE SENSING VIA UNMANNED AERIAL VEHICLE

RC controlled airplane
- glider + electric motor
- size = 268×118 cm
- weight = 1.8 kg
- payload weight < 0.4 kg

Multispectral camera
Online video transmission

Duplex communication
Ad-hoc sensor data retrieval

Autopilot
- GPS + compass + 3-axes accelerometer
- Ground station
<table>
<thead>
<tr>
<th>Sensor Node</th>
<th>Existing technology:</th>
<th>JSI tested:</th>
<th>JSI preferred:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mica, FireFly, WASPMote, BTNode, SquidBee, Iris, Cricket, Ember, Medusa, MK-2, BSN, Stargate, Sensinode, PushPin, SpotON, ...</td>
<td>Crossbow Imote2, Mote Works, eKo kit, TI eZ430-RF2500, Microchip PICDEM Z, SunSPOT, Arduino/ Libelium (Xbee), proprietary HW</td>
<td>Versatile Sensor Node</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Existing technology:</th>
<th>JSI tested:</th>
<th>JSI preferred:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MantisOS, FreeRTOS, eCOS, uOS, RETOS, SOS, ...</td>
<td>TinyOS, Contiki, CooCox</td>
<td>Custom code, Contiki</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual Machine</th>
<th>Existing technology:</th>
<th>JSI tested:</th>
<th>JSI preferred:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mate, VM*, SwissQM, CVM, ...</td>
<td>Squawk, .NET micro, Darjeeling</td>
<td>None for now</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application (Meta) Data</th>
<th>Existing technology:</th>
<th>JSI tested:</th>
<th>JSI preferred:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CSIRO, CESN, W3C SSN, Ontosensor... XML, RDF, RDF-S, OWL, SensorML ... Jena, Racer, Cyc, Weka, PISA, ...</td>
<td>Cyc KB RDF, OWL, CycL Jena, Racer, Cyc Weka, PISA</td>
<td>CycKB RDF, OWL, CycL JSI analytic tools</td>
</tr>
</tbody>
</table>

**JSI tested:**
- Cyc KB
- RDF, OWL, CycL
- Jena, Racer, Cyc Weka, PISA

**JSI preferred:**
- CycKB
- RDF, OWL, CycL
- JSI analytic tools
TESTED OPERATING SYSTEMS

Event driven OS:
- Contiki (On ST ARM Cortex-M3)
- TinyOS (Crossbow Mote)

Real time OS:
- CooCox (On ST ARM Cortex-M3)

Virtual machine:
- Squawk system level VM (On SunSpot)
- .Net Micro (On Crossbow Imote2)
COOJA SIMULATOR

Simulator platform: Linux, implemented in Java
Best choice for development of Contiki based WSN

Can be used in multiple development phases

- Validation of high level concepts (Java-based nodes)
- Simulations of Contiki core and user processes
- Firmware level simulations for selected platforms (MSP430) (with debugging capabilities such as break points, watches, logging and single stepping)
TINYOS ROUTING PROTOCOLS

Crossbow Mote Works

- Atmel ATmega128
- TinyOS
- nesC
- Cygwin

Run-time reprogramability
<table>
<thead>
<tr>
<th>Operating System</th>
<th>Virtual Machine</th>
<th>Sensor Node</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Existing technology:</strong> MantisOS, FreeRTOS, eCOS, uOS, RETOS, SOS, ...</td>
<td><strong>Existing technology:</strong> Mate, VM*, SwissQM, CVM, ...</td>
<td><strong>Existing technology:</strong> Mica, FireFly, WASPMote, BTNode, SquidBee, Iris, Cricket, Ember, Medusa MK-2, BSN, Stargate, Sensinode, PushPin, SpotON, ...</td>
</tr>
<tr>
<td><strong>JSI tested:</strong> Cyc KB RDF, OWL, CycL Jena, Racer, Cyc Weka, PISA</td>
<td><strong>JSI tested:</strong> Squawk, .NET micro, Darjeeling</td>
<td><strong>JSI tested:</strong> Crossbow Imote2, Mote Works, eKo kit, TI eZ430-RF2500, Microchip PICDEM Z, SunSPOT, Arduino/ Libelium (Xbee), proprietary HW</td>
</tr>
<tr>
<td><strong>JSI preferred:</strong> Versatile Sensor Node</td>
<td><strong>JSI preferred:</strong> Custom code, Contiki</td>
<td><strong>JSI preferred:</strong> None for now</td>
</tr>
</tbody>
</table>

*CSIRO, CESN, W3C SSN, Ontosensor… XML, RDF, RDF-S, OWL, SensorML … Jena, Racer, Cyc, Weka, PISA, …
PACHUBE AND SUNSPOT

Subscribed SunSpot sensor node description and stream to Pachube
AUTOMATIC TAGGING OF SENSOR META-DATA

Collect sensor description and data
- Automatic tagging

Use some of the existing rules and introduce new ones to perform reasoning using Cyc

Possible Applications
- Complex searching, with multiple constraints
  - Location, sensed features, tags
- Energy consumption monitoring combined with price plans for saving money

AUTOMATIC TAGGING OF SENSOR META-DATA

What is a “sensor”?

Sensor nodes vs computing devices

Sensor deployments

Sensor nodes

Middleware

Sensor (meta)data

SensorLab

<title>Ship - IAGO MAGADI IMO:8825808</title>
<feed>http://www.pachube.com/api/feeds/3826.xml</feed>
<status>live</status>
<location domain="physics" exposure="outdoor" disposition="mobile">
  <lat>45.61304</lat>
  <lon>13.78468</lon>
</location>
<data id="0">
  <tag>latitude</tag>
  <value minValue="0.0" maxValue="45.67596">45.61304</value>
</data>
<data id="1">
  <tag>longitude</tag>
  <value minValue="0.0" maxValue="13.80876">13.78468</value>
</data>
<data id="2">
  <tag>average_speed</tag>
  <value minValue="5.8" maxValue="7.3">6.7</value>
  <unit>knots</unit>
</data>

Individual: Sensor123
GAF Arg: 1

Mt: BaseKB
isa: Sensor
latitude: (Degree-UnitOfAngularMeasure 45.61304)
longitude: (Degree-UnitOfAngularMeasure 13.78)
sensorMeasures: Speed
KNOWLEDGE BASE FOR WIRELESS COMM.

Example of a Collection:
Constant: AccessNetwork.
In Mt: UniversalVocabularyMt.
  isa: SpatiallyDisjointObjectType.
  isa: QAClarifyingCollectionType.
In Mt: ComputerNetworkMt.
  genls: ComputerNetwork.

Example of an Individual:
Constant: IEEE802dot11Protocol.
In Mt: ComputerNetworkMt.
  isa: NetworkProtocol
  PhysicalLayerProtocol
  DataLinkLayerProtocol.
In Mt: ComputereseLexicalMt.
  acronymString: "IEEE802.11"
  "WiFi".

Example of a Predicate:
In Mt: ComputerNetworkMt.
  isa: BinaryPredicate.
  arg1Isa: ComputerNetwork.
  arg2Isa: CharacterString.
  comment: "Specifies the version of the protocol."

<table>
<thead>
<tr>
<th>Sensor Node</th>
<th>Operating System</th>
<th>Virtual Machine</th>
<th>Application / (Meta) Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Existing technology:</strong></td>
<td><strong>Existing technology:</strong></td>
<td><strong>Existing technology:</strong></td>
<td><strong>Existing technology:</strong></td>
</tr>
<tr>
<td>Mica, FireFly, WASPMote, BTNode, SquidBee, Iris, Cricket, Ember, Medusa MK-2, BSN, Stargate, Sensinode, PushPin, SpotON, …</td>
<td>MantisOS, FreeRTOS, eCOS, uOS, RETOS, SOS, …</td>
<td>Mate, VM*, SwissQM, CVM, …</td>
<td>CSIRO, CESN, W3C SSN, Ontosensor… XML, RDF, RDF-S, OWL, SensorML … Jena, Racer, Cyc, Weka, PISA, …</td>
</tr>
<tr>
<td><strong>JSI tested:</strong></td>
<td><strong>JSI tested:</strong></td>
<td><strong>JSI tested:</strong></td>
<td><strong>JSI tested:</strong></td>
</tr>
<tr>
<td>Cyc KB RDF, OWL, CycL Jena, Racer, Cyc Weka, PISA</td>
<td>Jena, Racer, Cyc Weka, PISA</td>
<td>Squawk, .NET micro, Darjeeling</td>
<td>Squawk, .NET micro, Darjeeling</td>
</tr>
<tr>
<td>Jena, Racer, Cyc Weka, PISA</td>
<td>TinyOS, Contiki, CooCox</td>
<td>None for now</td>
<td>None for now</td>
</tr>
<tr>
<td>TinyOS, Contiki, CooCox</td>
<td>Custom code, Contiki</td>
<td>Versatile Sensor Node</td>
<td>Versatile Sensor Node</td>
</tr>
</tbody>
</table>

**JSI preferred:**
- CycKB RDF, OWL, CycL
- JSI analytic tools
- Custom code, Contiki
- Versatile Sensor Node
VERSATILE SENSOR NODE

Processing platforms
- ST ARM Cortex-M3

Comm. tech.
- ZigBee, 6LoWPAN, IEEE 802.15.4
- Bluetooth
- Wi-Fi
- GSM/GPRS
- Ethernet
- Sensor Network Protocol (SNP)
- Satellite

Sensors
- Temperature
- Humidity
- Luminance
- Color
- Reflectance
- Pressure/Force
- Camera
- Optical Detector
- GPS
- Sound
- Accelerometer
- Gas (O₂, CO₂, CO)
- Hall effect
- Motion, presence, range (IR, ultrasonic)
- Capacitive/inductive touch
- Gyroscope
- Compass
- ...

Actuators
- Pulse Width Modulation (Light, Motor)
- Switch, Relay
- Servo
- Alarm

What is a "sensor"?
Sensor nodes vs computing devices
Sensor deployments
Sensor nodes
Middleware
Sensor (meta)data
SensorLab
VERSATILE SENSOR NODE (VSN)

Modular platform for WSN (VSCore + VSRadio + VSApplication + VSPower = VSN)

• High processing power and low energy consumption
• Sensor node & gateway (multi-tier / IP) capability
• Battery, solar or external power supply
• Re-configurable radio

In collaboration with ISOTEL d.o.o.
VERSATILE SENSOR NODE

VSCore
- Analog and digital sensor/actuator interfaces
- Possibility to use operating system (real-time, event-driven)
- Multiple expansion options
- Open C/C++ code libraries
- Onboard memory

VSRadio
- 300-900 MHz, 2.4 GHz radio interface (all ISM bands)
- ZigBee, 6LoWPAN and other IEEE 802.15.4 based solutions

VSApplication
- Bluetooth, Wi-Fi, Ethernet, GSM/GPRS
- Sensors/actuators
- PoE
<table>
<thead>
<tr>
<th>Application / (Meta) Data</th>
<th>Existing technology: CSIRO, CESN, W3C SSN, Ontosensor… XML, RDF, RDF-S, OWL, SensorML … Jena, Racer, Cyc, Weka, PISA, …</th>
<th>JSI tested: Cyc KB RDF, OWL, CycL Jena, Racer, Cyc Weka, PISA</th>
<th>JSI preferred: CycKB RDF, OWL, CycL JSI analytic tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Machine</td>
<td>Existing technology: Mate, VM*, SwissQM, CVM, …</td>
<td>JSI tested: Squawk, .NET micro, Darjeeling</td>
<td>JSI preferred: None for now</td>
</tr>
<tr>
<td>Operating System</td>
<td>Existing technology: MantisOS, FreeRTOS, eCOS, uOS, RETOS, SOS, …</td>
<td>JSI tested: TinyOS, Contiki, CooCox</td>
<td>JSI preferred: Custom code, Contiki</td>
</tr>
<tr>
<td>Sensor Node</td>
<td>Existing technology: Mica, FireFly, WASPMote, BTNode, SquidBee, Iris, Cricket, Ember, Medusa MK-2, BSN, Stargate, Sensinode, PushPin, SpotON, …</td>
<td>JSI tested: Crossbow Imote2, Mote Works, eKo kit, TI eZ430-RF2500, Microchip PICDEM Z, SunSPOT, Arduino/ Libelium (Xbee), proprietary HW</td>
<td>JSI preferred: Versatile Sensor Node</td>
</tr>
</tbody>
</table>
TARGETED APPLICATION AREAS

- Smart infrastructures
  - Lighting
  - Electricity
  - Water
  - Gas

- Transportation
  - Logistics

- Smart House
  - Industrial Processes

- Military

- Agriculture
  - Environmental monitoring

- eHealth
  - Sport

- Security
  - Emergency

- Advertising
  - Marketing

- Social Networks
OUTDOOR TESTBED

Municipality in Gorica (Gorizia) equipped with sensors, estimated 830 sensor nodes

- Project started August 2010
- Equip public light poles with sensor nodes able to also actuate by dimming the intensity of the light
OUTDOOR TESTBED

Kostanjevica: 53 light poles
5 equipped with sensor nodes
OUTDOOR TESTBED

Miren: 109 light poles
20 equipped with sensor nodes
OUTDOOR TESTBED

First phase: up & running 25 nodes
Second phase: up & running 100 nodes (by end of 2010)
Third phase: up and running 830 nodes (2011)
PORTABLE DEMO
SUMMARY

What is a “sensor”?  
Sensor nodes vs computing devices  
Sensor deployments  
Sensor nodes  
Middleware  
Sensor data and meta-data  

SensorLab experience, technology and testbed
THANK YOU!

MOST OF THESE SLIDES ARE BASED ON WORK WHICH IS TO BE PUBLISHED

THE WORKING TITLE OF THE PAPER IS

“LARGE TESTBED FOR SENSOR NETWORKS: COMPONENTS AND INTEGRATION”

CAROLINA.FORTUNA@IJS.SI