SIXTH FRAMEWORK PROGRAMME
PRIORITY 2
INFORMATION SOCIETY TECHNOLOGIES

GoodFood
Food Safety and Quality Monitoring with Microsystems



Deliverable D12- WP7
" System Requirements and Design"

Partner Responsible of the deliverable: MIDRA
Other Partners: SYNAPSIS, HTA, NMRC, EADS, SSSA
Report Version: Final
Report Preparation Date: 31/07/04
Classification: Public
Contract Start Date: 01/01/2004
Duration: 42 months
Project Co-ordinator: CNM (CSIC)

TABLE OF CONTENTS OF DELIVERABLE D12

1 EXECUTIVE SUMMARY 4

2 SYSTEM REQUIREMENTS FOR M18 PILOT SITE AT MONTEPALDI 5

2.1 SYSTEMS FOR VINEYARD MONITORING: 5
2.2 SYSTEMS FOR CELLAR MONITORING 6
2.3 ALARM OR ALERTS 6
2.4 ET_C CALCULATION 7
2.4.1 ET_O calculation 8
2.4.2 K_C calculation 9
2.4.3 ET_C calculation 9

3 SYSTEM DESIGN: WINE CHAIN MONITORING 10

3.1 PILOT-SITE'S SENSORS TECHNICAL SPECIFICATIONS 10
3.1.1 Vineyard 11
3.1.2 Cellar 28
3.1.3 Contribution from other WP to Wine Chain 33

4 HIGH-LEVEL SYSTEM DESIGN ARCHITECTURE 34

4.1 HIGH-LEVEL SYSTEM DESIGN ARCHITECTURE 34
4.2 CONTEXT 35
4.3 INTERFACES 37
4.4 SUBSYSTEMS 37
4.4.1 Wireless Sensor Network 38
4.4.2 The Stand Alone Tools 39
4.4.3 Video Monitoring Tools 40
4.4.4 The AmI Application level 40
4.5 THE DATA WORKFLOW 44
4.6 THE HIGH LEVEL WORKING OF THE AMI-GRID PLATFORM 46
4.7 THE SECURITY IN COMMUNICATION HIGH LEVEL 47
4.7.1 TinySec for TinyOS 48
4.8 PRESERVATION OF OPENNESS AND EXPANDABILITY OF ARCHITECTURE 50
4.8.1 Software Abstraction Layers 50
4.8.2 Tiny OS Porting 52
4.8.3 Custom 868 MHz RF Wireless Platform 53
4.8.4 Specification Overview of the nRF905 53
4.8.5 Antenna 54
4.8.6 Chip Antenna Options 54
4.8.7 Stub/Whip Antenna Options 55
4.8.8 Embedded Antenna Solution 56
4.8.9 Preservation of openness and expandability of architecture versus ZigBee and IEEE 1451 Standards 56
4.8.10 Wireless sensor networks 61
4.8.11 Embedded Operating Systems: TinyOS portability issues 67

5 THE SATELLITE COMPONENTS 70

5.1 THE WIRELESS SENSORS NETWORKS 70
5.1.1 The Tiny-OS operative system 70
5.1.2 Node Architecture 73

5.1.3 Sensors to network node interfaces ... 74
5.1.4 Configuration of "AmI Compliant" interface ... 78
5.1.5 Communication protocol ... 83
5.1.6 The node data processing ... 93
5.1.7 Node to gateway communication protocol ... 96

6 THE AMI GRID SOFTWARE ARCHITECTURE ... 98

6.1 THE COMMUNICATION PROTOCOL ... 98
6.1.1 State of the art ... 101
6.1.2 Gateway to Protocol Handler Communication Protocol ... 103
6.1.3 The AmI communication protocol stack ... 110
6.2 THE DATA MODEL AND STORAGE ... 113
6.3 THE SOFTWARE MODULES AND DEPLOYMENT ... 116
6.4 THE DATA MANAGEMENT: STORAGE, TRANSPORT, QUERY AND PRESENTATION ... 118
6.4.1 Transport ... 118
6.4.2 Data Storage ... 119
6.4.3 Resource Management Storage ... 120
6.4.4 Querying ... 120
6.4.5 Presentation ... 121

7 DESIGN OF SOFTWARE INTERACTION INTERFACES ... 122

7.1 HUMAN-MACHINE INTERFACE ... 122
7.1.1 Common Features ... 122
7.1.2 Observational Data Monitoring ... 124
7.1.3 Resource Management GUIs ... 126
7.1.4 Analysis GUI ... 127
7.2 INTERFACE USABILITY ... 127



Deliverable D12

“System Requirements and Design”

1 Executive Summary

This deliverable (D12-WP7D2: System requirements and design) is a result of Task 7.1: Analysis of user's needs and system requirements, Task 7.2: System design – Wine chain monitoring and Task 7.3: System design – AmI infrastructure. Although not all WP7 partners (MIDRA, SYNAPSIS, HTA, NMRC, EADS, SSSA) are formally involved in all the three tasks, all of them have contributed to this deliverable. It is structured in seven chapters (including this introductory executive summary as Chapter 1).

Chapter 2 (System Requirements for M18 Pilot Site at Montepaldi) starts off with the description of the systems for vineyard and cellar monitoring, continues with the description of the desired alarms and alerts, and finishes off with the calculation of the plant evapotranspiration (ET).

Chapter 3 (System Design: Wine Chain Monitoring) lists and describes the specifications of all the sensors which could be usefully implemented and integrated in the vineyard. Sensors are classified in commercially available, available from other research projects, or to be developed.

Chapter 4 (High-level System Design Architecture) starts off with a general introduction and then covers the design architecture in the following sections: context, interfaces, subsystems, data workflow, high level working of AmI Grid platform, security. It finishes off with some considerations on the preservation of openness and expandability of the architecture.

Chapter 5 (The Satellite Components) describes in details the components of the Wireless Sensors Network: the Operating System, the Nodes, the Interfaces, the Communication protocols and the Data Processing.

Chapter 6 (The AmI GRID software architecture) describes it in four sections: the Communication protocol, the Data Model and Storage, the SW modules and deployment, and the Data Management.

Chapter 7 (Design of software interaction interfaces) consists in one section describing the design of the Human-Machine Interface and a second short one with some early considerations on the interface usability.

To sum up, the system architecture, as proposed in this deliverable, mainly consists of a Wireless Sensor Network monitoring infrastructure. The Network is composed by a set of nodes, which are embedded systems capable of wireless communication. Each node is also equipped with a set of sensors, and uses the wireless communication infrastructure to route the sensed values to a gateway node; the gateway is responsible to transfer these data to the AmI CORE System, for high level analysis functions. In particular, the AmI CORE System implements a Knowledge Discovery in Database subsystem, for data analysis; a Decision Support System will be also developed, intended to help user in taking the right decision on the base of analysis results.

2 System Requirements for M18 Pilot Site at Montepaldi

As from user's needs expressed in Task 7.1, and according to the Pilot Site's sensors technical specifications for vineyard and cellar monitoring (Task 7.2), the System Requirements for the M18 Pilot Site are reported here, specifying subsystems to be deployed and major technical requirements.

2.1 Systems for vineyard monitoring:

Table 1

Implemented subsystems:	1 Wireless Sensor Network; 1 Stand-alone meteorological station; 2..4 Stand-alone video monitoring tools;
Wireless Sensor Network for vineyard monitoring:	15..20 battery operated nodes (*motes + sensors*); 1 Gateway integrating a GSM/GPRS module;
Nodes' typologies:	5..7 full functional nodes, implementing up to 10 sensors *(soil temperature and moisture, at two different levels, differential leaf temperature, xylem flux, diametric stem growth, air temperature and humidity)*; 8..10 basic functional nodes, implementing up to 4 sensors *(soil temperature and moisture, at two different levels)*; Up to 6 "dummy nodes" with no sensors, deployed to improve multi-hop communication performances;
Mote-to-mote communication:	868 MHz multi-hop wireless communication;
Expected nodes' battery life:	6 months (excluding full functional node implementation);
Video monitoring tools:	1..2 imaging units deployed for grapes' colour monitoring; 1..2 imaging units deployed for vineyard overview monitoring; Up to 10 still images captured daily; Integrated Gateway with GSM/GPRS module; Solar cells power supply; Outdoor water proof and not condensing cases;
Meteo station stand-alone unit:	Up to 5 sensors implemented *(Air temperature and humidity, wind direction and speed, rainfall, PAR, global radiation)*; Integrated Gateway with GSM/GPRS module; Solar cells power supply; Outdoor water proof and non condensing case;

2.2 Systems for cellar monitoring

Table 2

Implemented subsystems:	1 Wireless Sensor Network;
Wireless Sensor Network for vineyard monitoring:	15..20 Battery operated nodes; 1 Gateway connected to a DSL communication interface;
Nodes' typologies:	3..5 nodes for must analysis, implementing up to 2 sensors, when available as prototypes in small factor form *(ethylic alcohol, glucose)* ; 5..10 nodes for barrique analysis, implementing up to 4 sensors; 1..2 nodes for cellar temperature and humidity monitoring; Up to 6 "dummy nodes" with no sensors, deployed to improve multi-hop communication performances;
Mote-to-mote communication:	868 MHz multi-hop wireless communication;
Expected nodes' battery life:	6 months;

2.3 Alarm or Alerts

1. Vineyard
    a. Drought stress
        i. variables required: *soil moisture*
        ii. alarm activated for soil water content ≤ 30% of soil available water
        iii. alarm rule - depending on the growing state of the plant:
            1. NOT IRRIGATE
            2. IRRIGATE

    b. pathogen attack
        i. variables required: *see forecast models*

2. Cellar
    a. Temperature during yeast alcoholic fermentation process
        i. variables required: *must temperature*
        ii. alarm activated for temperature ≥ 32°C or ≤22°C
        iii. alarm rule
            1. ACTIVATE COOLING-HEATING SYSTEM

    b. Sluggish or stuck alcoholic fermentation
        i. variable required: *glucose*, *ethylic alcohol*, *temperature*

ii. function required: flow graph of glucose *vs* time ($\Delta_{(glucose)}$/time)
iii. alarm activated for $\Delta_{(glucose)}$/time = 0 or very low
iv. alarm rule:
1. fermentation process has completed – NO ACTION
2. sluggish or stuck fermentation - ACTIVATE YEAST ACTIVATION SYSTEM
a. heat must (22-24°C) for 2 days
b. rouse yeast
c. make yeast starter (remove old yeast and re-inoculate young yeast) - only if the alcohol level of the must is ≤ 9%

c. Temperature during malolactic fermentation process
i. variable required: *must temperature*
ii. alarm activated for temperature ≥ 22°C or ≤16°C
iii. alarm rule
1. ACTIVATE COOLING-HEATING SYSTEM

3. Ageing (barrel)
a. SO_2
i. variables required: *loose* and *total* SO_2
ii. alarm activated for [loose SO_2] ≤ 25 mg/l
iii. alarm rule:
1. ADD SO_2

b. Acetic acid
i. variables required: *acid acetic* (AA)
ii. alarm activated for AA ≥ 0.5 g/l
iii. alarm rule:
1. CONTROL TEMPERATURE (≤ 15°C)
2. DO NOT AIR WINE
3. ADD SO_2

2.4 ET_C calculation

Potential evapotranspiration (ET_O or PET) is the sum of evaporation and plant transpiration, that is the amount of water that could be evaporated and transpired from a reference crop, usually a cool season grass, if there was plenty of water available.
The crop coefficient K_C is the fraction of water that a non-water stressed crop uses in relation to that of ET_O.

2.4.1 ET_O calculation

CIMIS Equation

The CIMIS Equation used is a version of the Modified Penman Equation by Pruitt and Doorenbos (*Proceeding of the International Round Table Conference on "Evapotranspiration", Budapest, Hungary. 1977*). It also employs a wind function developed at the University of California Davis. The input variables used in the CIMIS equation and the steps required to calculate ETo are described below.

Variables Required

i. RH = Mean hourly canopy relative humidity (%) – from plant sensor
ii. Rn = Mean hourly net radiation (Wm^{-2}) – from meteo station
iii. T = Mean hourly leaf temperature (°C) - from plant sensor
iv. U_2 = Mean hourly wind speed at 2 meters (ms^{-1}) - from meteo station
v. Z = Elevation of the station above mean sea level (m)

Steps

1. **Saturation vapor pressure (kPa)**

   $es = 0.61078 * \exp(T * 17.27/ (T + 237.3))$

2. **VPD - Vapor pressure deficit**

   VPD = es - ea (kPa) (where ea=es*RH/100)

3. **Δ - Slope of the saturation vapor pressure vs. air temperature curve at the average hourly air temperature (kPa/°C)**

   $\Delta = (4099 * es)/(T + 237.3)^2$

4. **Barometric pressure (kPa=atm/10)**

   $P = 101.3 - 0.0115 * Z + 5.44 * 10^{-7} * Z^2$

5. **γ - Psychrometer constant (kPa/°C)**

   $\gamma = 0.000646 (1 + 0.000946*T) P$

6. **W - Weighting function**

   $W = \Delta/(\Delta + \gamma)$

7. **FU2 - Wind function**

   For Rn<=0 (nighttime)

$FU_2 = 0.0125 + 0.0439U_2$

For Rn>0 (daytime)
$FU_2 = 0.030 + 0.0576U_2$

8. **NR - Convert Rn from Wm^{-2} to mm**

   NR = Rn/(694.5 (1-0.000946*T))

9. **Hourly ETo is approximately equal to RET**

   $RET = W*NR + (1-W)VPD * FU_2$

10. **Daily ETo equals the sum of 24 hours RET (mm)**

2.4.2 K_C calculation

Larry E. Williams (2001)

The crop coefficient K_C is dependent upon the stage of vine growth, degree of cover, height and canopy resistance, and varies throughout the season. Larry E. Williams proposes a computational mode based on degree-days accumulation and row spacing.

Variables Required

i. T_{MAX} = Daily maximum leaf temperature (°C) – from plant sensor
ii. T_{MIN} = Daily minimum leaf temperature (°C) – from plant sensor
iii. RS = Row spacing (m)

Steps

1. **degree-days calculation (°C)**

   $DDs = \Sigma[(T_{MAX} + T_{MIN})/2 - 10]$

2. **crop coefficient**

   For row spacing of 2.74 m, where x is DDs
   $K_C = 0.61 * [1 + e^{-(x + 525)/301}]^{-1}$

2.4.3 ET_C calculation

$ET_C = ET_O * K_C$

3 System Design: Wine Chain Monitoring

3.1 Pilot-Site's Sensors Technical specifications

This section reports the specifications about the sensors to be implemented and integrated in the vineyard. The following features have been considered as a metric to compare and assess the compatibility with the general objectives of the project:

- products availability,
- sensitivity,
- state of development,
- suitability for the pilot-site and AmI-compatibility,
- technique to release data,
- power supply and consumption,
- size and weight
- off-the-shelf product availability,

Soil analysis

Table 3

Sensor	**Deployment**
Soil temperature (20 and 50 cm depth)	Most of the nodes
Soil moisture (20 and 50 cm depth)	Most of the nodes

Air and meteorological parameters

Table 4

Sensor	**Deployment**
Air Temperature	On the meteo station
Air humidity	On the meteo station and on some nodes
Wind direction and speed	On the meteo station
Rainfall	On the meteo station
PAR	On the meteo station
Global radiation	On the meteo station

Plant analysis

Table 5

Sensor	**Deployment**
Differential leaf temperature	On some nodes
Xylem flux	On some nodes
Diametric stem growth	On some nodes
Area video monitoring	With a stand-alone still image camera
Peronospera (fungal disease)	With a stand-alone analysis equipment
Oidium (fungal disease)	With a stand-alone analysis equipment
Tignola (*eupoecilia ambiguella* – parasitic)	With a stand-alone capturing system and a still image camera

Cellar Monitoring

Table 6

Sensor	**Deployment**
Ethylic Alcohol	For must analysis
Glucose	For must analysis
Temperature	For must analysis
Malic Acid	For must analysis
Temperature	For barrique analysis
Loose SO_2	For barrique analysis
Total SO_2	For barrique analysis
Acetic Acid	For barrique analysis
Cellar Temperature	On a single node
Cellar Humidity	On a single node

3.1.1 Vineyard

3.1.1.1 Node: Soil

Each node will have to monitor two fundamental soil parameters:

- Soil Temperature
- Soil Moisture

These two parameters will be determined at two different depth levels. Both temperature and moisture will be implemented using commercially available sensors

Temperature sensor

A possible implementation is based on the TMP175 which is Two-Wire, serial output temperature sensor available in an SO-8 package. Requiring no external components, the device is capable of reading temperatures with a resolution of

0.0625 °C. The TMP175 feature a Two-Wire interface that is SMBus-compatible, with the TMP175 allowing up to 27 devices on one bus.

Table 7

Name of the Manufacturer	Texas Instruments TMP175
Sensors	Temperature
Measurement Range, Accuracy and Resolution	**Range** Temperature -40 to +125 °C **Accuracy** ±1.5 °C (max) from -25 °C to 85 °C ±2.0 °C (max) from -40 °C to 125 °C **ResolutionTemperature** 0.0625 °C 9- to 12-Bits
State of Development	Commercially available
AmI compatibility	Compatible though the interfacing of the station to the mote or to the gateway needed to be investigated
Data output	Two-Wire Serial Interface
Power Supply (V)	Device: from 2.7 VDC to 5.5 VDC
Size (Length×Width ×Height)	Device size is: 5 mm × 5.80 mm × 1.75 mm (0.197 x 0.228 x 0.069 inch)
Cost (Euro)	Device price: 0.85 $

Alternative Temperature Sensor

Table 8

Manufacturer	Supply Voltage	Temperature Range/Accuracy	Model	Interface
Hukseflux (www.hukseflux.com)	-	-60 to 150 degrees Celsius	ST01	Thermistor Based PT100
Cambell Scientific (www.cambellsci.com)	-	-35 to 50 degrees Celsius	107	Thermistor Based PT100
Novalynx (www.novalynx.com)	9-24V <5mA	-40 to 60 degrees Celsius /+-0.4	210-TP1	Output 0-5V or 4-20mA
Pessl Intruments (www.metos.at)	4.57V-7V 200uA	-30 to 99 degrees Celsius / +-0.5	MD503	1 to 4kHz
Met One Instruments (www.metone.com)		-50 to 50 degrees Celsius / +-0.1	063-1	Thermistor Based

Moisture Sensor

Table 9

Manufacturer	Supply Voltage	Range/Accuracy	Model	Interface
Novalynx (www.novalynx.com)	12V 40mA	0-100% moisture (Linear)	250-124	0-5V /4-20mA or 0-1mA
Stevens Water (www.stevenswater.com)	7-30V 20mA to 40mA	Temperature:-10 to +65 degrees Centigrade /+-0.6 Moisture: Completely dry to fully saturated /+-0.03	Hydro Probe Sensor	0-2.5 or 0-5V 4 channel output
Dynamax (www.dynamax.com)	5-15V DC 19 mA	Completely dry to fully saturated /+-1%	ML2x	0-1V
Phytech (www.phytech.co.il)	5.5-36V	0-100 %	S-SMS 0-2V	0 to 2V

3.1.1.2 Air Meteorological Station

System Requirements highlight the need of a single system for the monitoring of the most important air parameters. Such a parameter are:

- Temperature
- Humidity
- Wind Speed and Direction
- Rainfall
- PAR
- Global radiation

Products availability:

The sensors to measure temperature, humidity, wind direction rainfall PAR and global radiation can all be measured using the commercially available weather stations. There are several manufacturers offering different range of sensors integrated in the station. These stations are more or less software controlled, able to communicate with the base station (computer) using cable, modem, GSM and even radio frequency. These stations are able to harvest power using solar panel. As a result it eliminates the need of long power cable and also large batteries. Possible custom solution will be also taken into account based on the implementation of proprietary hardware and software. Moreover custom solution will be based on the integration of off the shelf commercially available sensor.

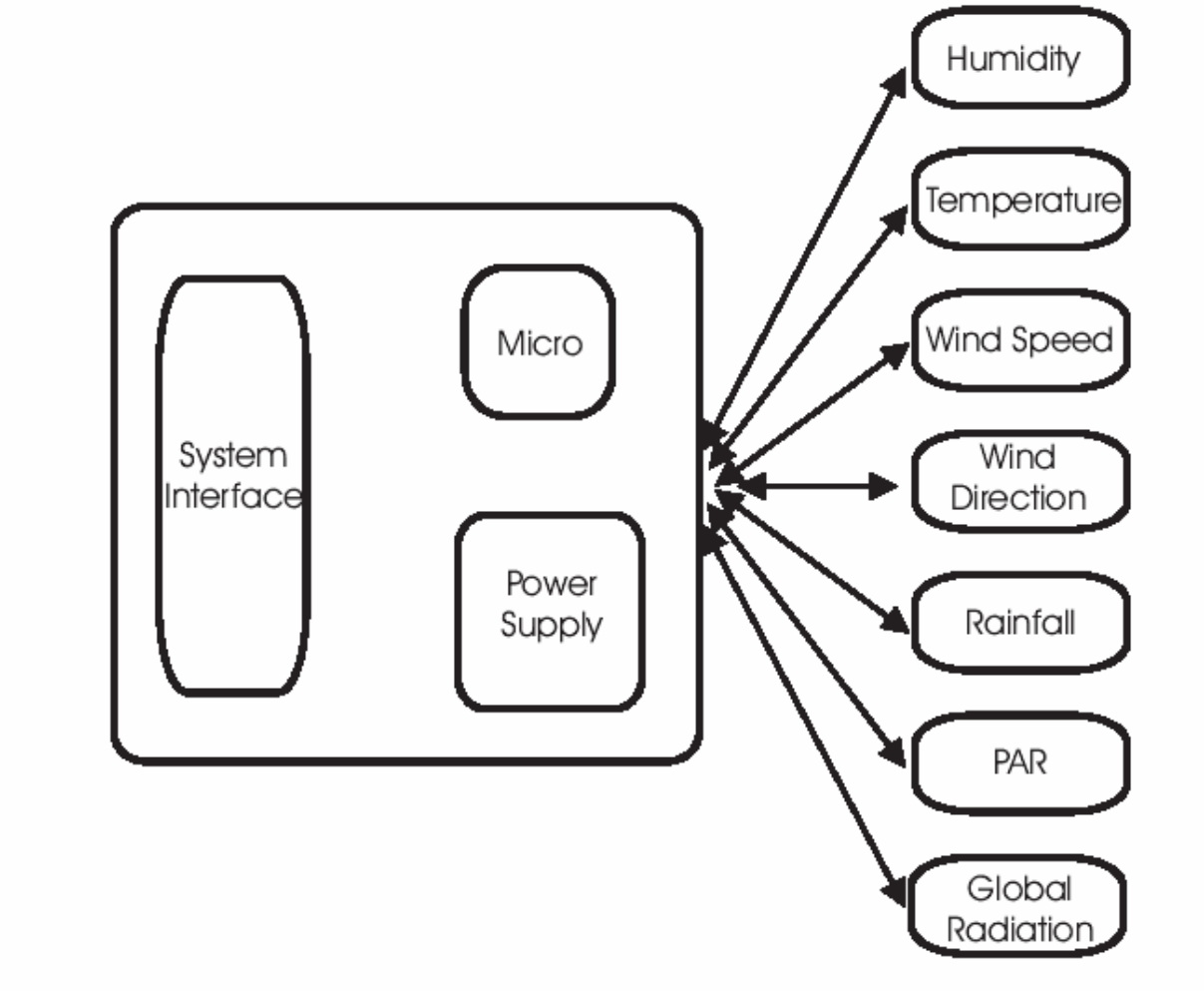


Figure. 1- Meteorological Station Block Diagram

Commercially available Meteorological Base Station 1

Table 10

Name of the Manufacturer	GroWeather® Standard from DAVIS (7450)		
Sensors	Anemometer	Temp. & Humidity	Rain
Measurement Range, Accuracy and Resolution	**Range** Wind Speed: 4 to 280 km/hr Wind Direction: 0° to 360° Wind Run: 0 to 1999.9 km Wind chill: -92° to +37°C **Accuracy** Wind Speed: ±5% Wind Direction: ±7% Wind Run: ±5% Wind chill: ±2°C **Resolution** Wind Speed: 1 km/hr Wind Direction 1° (0° to 355°) Wind Run: 0.1 km Wind chill: 1°C	**Range** Temperature: -45° to 60°C Relative Humidity: 0-100% Dew Point: -73° to 60°C **Accuracy** Temperature: ±0.5°C Relative Humidity: ±3% Dew Point: ±2°C **Resolution** Temperature: 0.1° Relative Humidity: 1% Dew Point: 1°C	**Range** Daily Rainfall 0 to 819 mm Total Rainfall 0 to 9999mm Rate Rainfall 0 to 999mm/hr **Accuracy** Rainfall ±4%, ±1 rainfall count between 0.2mm-50.0mm/hr; ±5%, ±1 rainfall count between 50.0mm-100.0mm/hr Rate of Rainfall: ±5% **Resolution** 0.2 mm
Size and Weight (Length×Height×Width)	470mm × 191mm × 121mm 1.332 kg	83mm × 38mm × 89mm 425 g	16.5 cm diameter × 24 cm high Collection Area 200 cm^2 1 kg
State of Development	The sensors can be easily integrated to the station and measurements can be done using software. The station requires solar power kit (optional) and other components like System Shelter, Rain Collector Shelf, etc.		
AmI compatibility	Compatible though the interfacing of the station to the mote or to the gateway needed to be investigated		
Data output	Through software digital data		
Power Supply (V) Consumption (W)	For the Console: 16mA (typical) at 10 to 16 V (100mA when display is illuminated)		
Size and Weight (Length×Width ×Height)	Console: 133mm × 137mm × 76mm Sensor Interface Module (SIM): 145 mm × 149 mm × 32 mm Display: 112 mm × 48 mm Total Weight: 1.53 kg		
Cost (Euro)	Complete system price: 1179 plus VAT and carriage		

Alternative Meteorological Base Station 2

Table 11

Name of the Manufacturer	GroWeather® Standard from DAVIS (7450)		
Sensors	Anemometer	Temp. & Humidity	Rain
Measurement Range, Accuracy and Resolution	**Range** Wind Speed: 4 to 280 km/hr Wind Direction: 0° to 360° Wind Run: 0 to 1999.9 km Wind chill: -92° to +37°C **Accuracy** Wind Speed: ±5% Wind Direction: ±7% Wind Run: ±5% Wind chill: ±2°C **Resolution** Wind Speed: 1 km/hr Wind Direction 1° (0° to 355°) Wind Run: 0.1 km Wind chill: 1°C	**Range** Temperature: -45° to 60°C Relative Humidity: 0-100% Dew Point: -73° to 60°C **Accuracy** Temperature: ±0.5°C Relative Humidity: ±3% Dew Point: ±2°C **Resolution** Temperature: 0.1° Relative Humidity: 1% Dew Point: 1°C	**Range** Daily Rainfall 0 to 819 mm Total Rainfall 0 to 9999mm Rate of Rainfall 0 to 999mm/hr **Accuracy** Rainfall ±4%, ±1 rainfall count between 0.2mm-50.0mm/hr; ±5%, ±1 rainfall count between 50.0mm-100.0mm/hr Rate of Rainfall: ±5% **Resolution** 0.2 mm
Size and Weight (Length×Height×Width)	470mm × 191mm × 121mm 1.332 kg	83mm × 38mm × 89mm 425 g	16.5 cm diameter × 24 cm high Collection Area 200 cm^2 1 kg
State of Development	The sensors can be easily integrated to the station and measurements can be done using software. The station requires solar power kit (optional) and other components like System Shelter, Rain Collector Shelf, etc.		
AmI compatibility	Compatible though the interfacing of the station to the mote or to the gateway needed to be investigated		
Data output	Through software digital data		
Power Supply (V) Consumption (W)	For the Console: 16mA (typical) at 10 to 16 V (100mA when display is illuminated)		
Size and Weight (Length×Width ×Height)	Console: 133mm × 137mm × 76mm Sensor Interface Module (SIM): 145 mm × 149 mm × 32 mm Display: 112 mm × 48 mm Total Weight: 1.53 kg		
Cost (Euro)	Complete system price: 1179 plus VAT and carriage		

3.1.1.3 Node: Air

As illustrated I the System Requirements users require that each node has to monitor both the:

- Relative Humidity
- Temperature

These two parameters will be monitored using a commercially available single digital sensor by Sensisirion.

Table 12

Name of the Manufacturer	SENSIRION: SHT11	
Sensors	Temperature	Humidity
Measurement Range, Accuracy and Resolution	**Range** Temperature -40 to +123.8 °C **Accuracy** Temperature ±0.5 °C @25 °C **Resolution** Temperature typical 0.01 °C (14 bit) **Response time** Temperature from 5 s min. to 30 s max. [1/e (63%)]	**Range** Relative Humidity: 0 to 100% RH **Accuracy** Relative Humidity: ±3.5% **Resolution** Relative Humidity typical 0.03%RH (12 bit) **Response time** Relative Humidity 4 s [1/e (63%) slowly moving air]
State of Development	Commercially available	
AmI compatibility	Compatible though the interfacing of the console to the mote or to the gateway.	
Data output	digital data.	
Power Supply (V) Consumption (A)	Device: from 2.4 VDC to 5.5 VDC Measuring: 550 □A; Average: 28 □A; Sleep: 0.3 □A [@25 °C VDD=5V] Output peak current: on 4 mA; Tristated 10□A [@25 °C VDD=5V]	
Size and Weight (Length×Width ×Height)	Device size is: 7.42 mm × 4.88 mm × 2.5 mm (0.29 x 0.19 x 0.1 inch) Device weight: 100 mg	
Cost (Euro)	Device price: 17.69	
Image of the Device		

Sensirion offers digital humidity sensor systems with integrated temperature sensors.
These sensors are developed with an innovative technology: CMOSens®, this technology lets to integrate two sensors and readout circuitry on one single chip.
The digital humidity and temperature sensor is fully calibrated and offers excellent long-term stability and ease of use at very low cost and user friendliness. Each SHT11 is individually calibrated in a precision humidity chamber with a chilled hygrometer as reference. The calibrated coefficients are programmed into the OPT memory.
These coefficients are used internally during measurements to calibrate the signals from the sensors.

3.1.1.4 Node: Plants

Four different sensors will be implemented directly on the plant.

- Diametric Growth Sensor (Stem)
- Differential Temperature Sensor (Leaf)
- Xylem Flux Sensor

Diametric Growth Sensor (Stem)

Table 13

Manufacturer	Supply Voltage	Range / Accuracy	Model	Interface
Dynamax (www.dynamax.com)	-	0 to 5mm/0.050 mm	DEX20	+/- 5 mV
Phytech (www.phytech.co.il)	10-30V	0 to 5mm Linear Sterns: 4 to 25mm	SD-5M	4 to 20mA or 0 to 20mA

Differential Temperature Sensor (Leaf)

Differential temperature sensor is not available commercially though it can be derived and integrated from existing precision temperature sensors.

Table 14

Name of the Manufacturer (Type of sensor)	Calex Electronics Limited (EL series, Infrared)	Phytech Limited (LT-2M, Thermistor)	Pessl Instruments GmbH (MD502, SMT160-30)
Measurement Range, Accuracy, Stability, Sensitivity	*Accuracy* ±1% of reading or ±1°C whichever is greater *Repeatability* ±0.5% of reading or ±0.5°C whichever is greater	*Range* 5-50 °C Accuracy ±0.2°C *Stability* ±0.5°C over 1 year period	*Range* -30-99°C *Accuracy* 0.5°C (-30 to +99 °C)
State of Development	Need to use two sensors to get differential temperature	Need power adapter operates on 10-30 V DC	Need minimum development as the sensor can be used directly with processor
AmI compatibility	Compatible though one of the problem is large supply voltage	Compatible though needs large supply voltage	Better compatibility as output is interpretable by a microprocessor without A-D converter
Data output	*Analog output:* 4-20 mA (linear with temp)	*Analog Output:* 0-2 V or 4-20 mA (optional)	Both analogue and digital. Output is a square wave with a well defined Temperature dependent duty cycle.
Power Supply (Volt)	*Supply voltage:* 24 V DC (28 V DC max.)	*Supply voltage:* 15 V DC ± 10%	Supply voltage: 4.57 - 7 V DC Supply current: 200 µA max.
Size and Weight (Diameter× Height)	18mm × 85mm	(Length × Width × Height) 35mm×15mm×15 mm	(Length × Width × Height) 25.4mm×12.7mm×5.08mm
Cost (Euro)	162 to 266 plus VAT and carriage	576 to 601 plus VAT and carriage	9.48 for SMT160-30 in HEC package plus VAT and carriage

Xylem Flux Sensor

Xylem flux sensor also known as Sap flow sensor is a commercially available sensor. Several flux sensors are available from different manufacturers namely

Dynamax, Environmental Measuring Systems, Phytech, Tyco Greenspan, etc. The parameters for different products are shown in the following table.

Table 15

Name of the Manufacturer	*Dynamax Limited*	Environmental Measuring Systems	PhyTech Limited	
			RS-SF-5PR	RS-SF-8PR
Sensitivity	System Range ±2.5 mV Resolution 0.33 µV to 0.1 µV	**Accuracy** 0.3 % of the full scale	Approximate range of ± 3 ml/h is determined with the use of stem simulator	
State of Development	Requires data logger (Flow32 Sap flow system or Flow4-DL system)	Requires data logger and software	For 4 to 10mm diameter stem	For stem diameter > 15mm
AmI compatibility	Yes. But its necessary that the algorithms used in the system - be made available to the designer, for the interface of the Mica Mote. The supply of algorithms will be the discretion of the company making the sensors.		The sensors can be connected to any data logger and do not require specific software.	
Data output	Analog from the sensor. Digital data output through software when connected to the system	Analog from the sensor. Digital data output through built-in data logger	Analog output 0-2 V or 4-20 mA Data logger supplied to extract data.	Analog output 0-2 V or 4-20 mA Data logger supplied to extract data.
Power Supply Consumption	Sensors 2.3 to 9 V 0.05 to 13 W	12 V	12 V @ 1 A	9V @ 0.45 A
Size and Weight (Diameter × Height)	**Sensors** A wide range of size from (2.1~3.5 × 35) to (150~165 × 900) Data logger *Flow32* 43 × 35 × 16 cm^3 11.5 kg *Flow4* 16.5 × 30 × 34.3 cm^3 7 kg	**Sensors** A range of sensors diameters 8-12mm and 12-18mm *Data logger* 43 × 33 × 20 cm^3 8 kg box, 12 kg cables, 2 kg sensors and accessories	30 × 35 × 40 mm^3	Probe 1mm × 10mm Sensor 70mm × 10mm Signal conditioner: 120 × 70 × 37 mm^3
Cost (Euro)	-	-	577 to 611 plus VAT and carriage	-

3.1.1.5 Video Monitoring

IMAGE ACQUISITION PURPOSES:

As a consequence of the User's Needs, exposed in the "sensor matrix" (vineyard), digital image acquisition and storage is requested for the following purposes:

- To monitor the colour variation of the grapes;
- To obtain a visual feedback of the plants growing, and to have a general sight of the vineyard;

Moreover, the image storage and visual analysis could be significant in order to compare and correlate the impact of sensed parameters with the vine growth and grapes' ripening.

System requirements:

Images should be acquired and stored about every:

- 30 minutes (plants and full vineyard monitoring),
- daily (grapes monitoring).

Time delay between image acquisition and availability on the user side has not been considered a crucial parameter

Outdoor requirements:

- The image acquisition systems should be enclosed in fully water-proof and not condensing cases.

Communication interface for data transfer:

- Should be wireless (no wired communication infrastructure is supposed to be available in the vineyard).

These systems are to be considered as stand-alone and will not interact directly with the wireless sensor network nodes.
Therefore an AmI-grid compliant wide-area communication interface will be integrated.

Image resolution and quality:

Will be selected according both User requirements for the specific application, and technical requirements: as a first evaluation, the VGA (640x480 pixels) or QVGA (320x240 pixels) resolutions, with 10 or 16 bits of colour depth should match the expected results.

Especially for the grape's close monitoring, particular attention will be paid to the colour balancing; moreover, daily acquisitions should be performed at a fixed time, to store images acquired with similar light conditions, anyway considering the possible effects of different weather conditions.

System architecture:

The proposed system architecture and interactions with the other subsystems are represented in the following scheme:

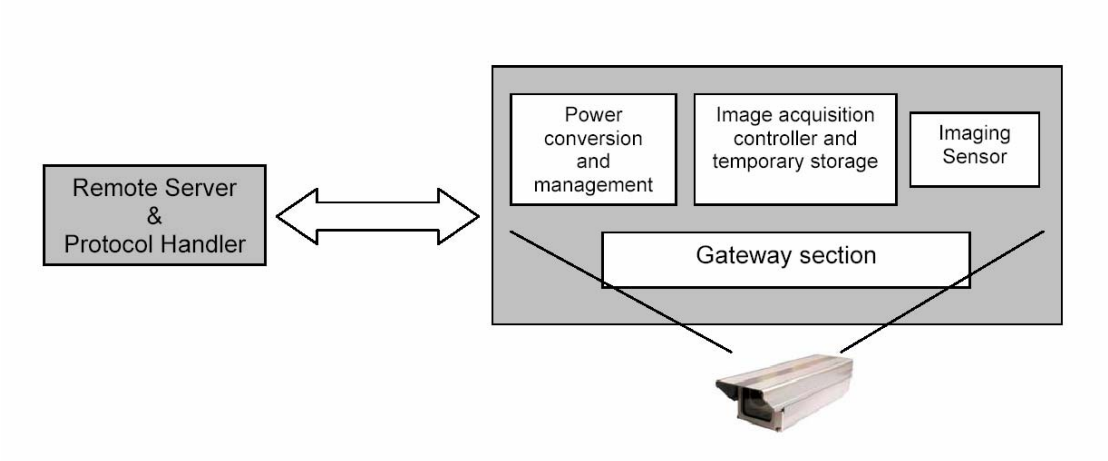


Figure 2: Video Monitoring tools Architecture

The image acquisition equipments can be considered as stand-alone subsystems, integrating all the necessary components and interfaces for capturing, compressing and transferring digital still images:

Imaging sensor: a low cost CMOS sensor and the necessary optical lens; to improve flexibility, remotely controlled zoom and camera orienting functionalities could be implemented;

Image acquisition controller and temporary storage: a microcontroller-based board and companion imaging co-processor for hardware level image compression; an additional SRAM memory module could be integrated if temporary image buffering is necessary;

Power conversion and management: all the necessary components to provide regulated 3V or 5V DC power to the other sections.

Gateway section: an AmI-grid compliant communication interface implementing all the protocols for data exchange with the remote Protocol Handler. As from the User's Needs and Requirements and Pilot Site specifications, no wired communication infrastructure is supposed to be available. Using an on-board digital image compression (JPEG) the data size for image transferring should be about 15 – 20 Kbytes; so, the GPRS connectivity should match both user's and technical requirements.

Image sensor technologies:
Low cost CMOS imaging technology should offer the expected results in terms of image size and colour depth. The adopted architecture is derived from the

commercially available webcams, providing extremely integrated, low power and low-cost components.
Moreover, the 2nd generation imaging modules developed for the top-level mobile phones are now available, with increased performances in terms of image resolution and quality: the main benefit will be represented from the all-in-one packages, integrating the CMOS image sensor and the optical lens in a single component.
Adopting such architecture, companion hardware co-processors are usually supplied from many manufacturers, and can be integrated to provide a transparent low level control of the imaging sensor and a hardware-level high speed image compression (usually in JPEG compatible formats): in this case, no further data elaboration is necessary for viewing the images with a PC or with most of the commercially available PDAs.

Compared to CCD sensors, the CMOS technology still offers a lower quality, but provides the following advantages:

- reduced power consumption;
- lower voltage requested (usually 3V);
- simpler integration and less additional circuitry requirements;
- lower cost.

3.1.1.6 Pathogenic sensors

The most dangerous pathogens of the grape can be identified in the following species:

- The two moths of the grapes
- Peronospora
- Uncinula necator
- Botrytis

Moth of the grapes

The two moths are Lepidoptera, *Lobesia botrana* (Figure 1) and *Eupoecilia ambiguella*.
The harmfulness of these moths is due to their target: the bunch of grapes. The moth eat before the flower, after the grape until the maturation. Furthermore this is not the worst effect of the moth, in fact the damage on the grape due to the grub is a possible cause of the Botrytis, a worse pathogen for the grapes.
The grapes is exposed to the moth three times for years. In the case of the *Lobesia botrana*, for example, it winters in April as chrysalis (see Fig 2). The mating is during the twilight at a temperature over 15° degree. The moth lays the grub in the bud of the grape flower. When they born eat the flowers. A second attach of the *Lobesia botrana* is in middle of June when start the second coming out of moths of the grub in the grapes.
From this moment the laying of the eggs. The damage of this second attach is stronger of the first because they attach the little grape with consequently

emptying. But the worst attack is due to the moths were born from this laying because the moth of this generation attacks the mature grapes making the better condition for the start of the Botrytis. The laying of eggs of this generation of moth hide in gorge of the bark until the next Springer.
Actually for low diffusion of these pathogens any measure is taken. Consequently it is important to monitor the moth diffusion using, for example, several pheromone traps for the male. If the concentration of insect is too high an intervention with pesticide is necessary. An alternative to the traps, at the moment, is the visual control







Figure 3: Moth of the grapes

Peronospora (*Plasmopara viticola*, downy mildew)

The peronospora is the most dangerous pathogen of the grapes. It is caused by a microscopic fungus that attacks the leaf, the buds and the grapes. The peronospora propagates asexually in summer by forming sporangia and zoospores, while in autumn sexual propagation starts and diploid oospores are produced in host tissues. Primary infection occurs when water-borne sporangia or zoospores, produced by germinating oospores, are splashed onto wet vine foliage. Then motile zoospores move to stomata and penetrate the stomatal aperture via a germ tube. After an incubation period sporangiophores and sporangia are produced and grow out through stomatal pores during warm

humid nights. Secondary infection follows wind or water-splash dispersal of these sporangia. Dependent on weather several circles of secondary infections can take place during one season. The effects of this fungus is a typical "oil stain" in the upper side of the leaf. The main causes of this pathogen are temperature and humidity.

Monitoring level for the pathogenic sensing
In fact an empirical method to decide when start an antiperonospora measure, the rule of the three ten, is based on these two factors. When it is rained 10 mm in 24-48 hours, the minimum temperature is 10 degree and the buds are long 10 cm it is time to take antiperonospora measure. After the first peronospora infection it is enough that the leaf are wet for 2 hours to have another infection.



Figure 4: Peronospora

Uncinula necator (Powdery mildew):

The powdery mildew is visible under naked eye as a white-greyish downy grouth that may cover the leaves, young shoots and bunches. The fungus cannot be grown in artificial media (obligate parasite).
In the shoots, brown spots with a white-greyish powder originating deformations
in the younger ones, and yellow brownish spots with fibrous-looking edges in the mature ones. In the leaves, light brown spots in the veins of the leaf (lower surface) originating deformations with wavy and crinkly aspect; greyish-white dusty outer covering, on the leaf surfaces.
In the clusters, death and fall of flowers and young berries and splitting of the developing ones (the susceptibility ceases when the sugar contents reaches 12-15%); powdery appearance similar to the leaves.
The fungus overwinters, resisting to the adverse conditions as sexual fructifications (cleistothecia) and also as dormant mycelia, protected by the bud cales (main form of maintenance). The infections take place from this mycelium and also from the sexual spores; (ascospores).
The superficial mycelia that forms, penetrates in the host epidermal cells through haustoria; externally conidio phores are formed which produce assexual spores (conidia) in chains, originating the secondary infections; in autumn, cleistothecia

are formed from the subtle mycelia that covers the leaves and the immature tips of the shoots.



Figure 5: Uncinula necator

Botrytis (Grey rot, Grey mould)

The conidiophores and conidia form a greyish mould on the affected organs; the sclerotia are easily visible as black pustules that appear in the elongated spots in the shoots, in autumn. The fungus can be isolated and grown in artificial nutritive media. Necrotic spots triangular in shape formed at the edge of the leaves; brown elongated spots, in the young shoots (that may break), turning black at the maturation due to the formation of sclerotia; total or partial destruction of the bunches before flowering and later on, infection through the wounds, by contact between berries or by infection of the rachis and peduncule. The fungus may overwinter as a saprophyte in dead plant organs or in the bud scales mainly as sclerotia. The infection takes place from these, forming new mycelium and conidia which will germinate in the young shoots, leaves and bunches (direct infection) or through wounds or dead organs (floral pieces); in maturing bunches, infection takes place through the wounds or by contact between berries. In autumn, the unripened shoots show elongated whitish spots, producing black sclerotia. The infection takes place after the bud bursting under high humidity conditions and suitable temperatures (15-20 °C.); deficient aeration (dense vegetation) is very favourable to the disease development. The ascospores and conidia, particularly from infected ripen bunches, may be carried

by the wind or insects. Chemical preventive measures with appropriate fungicides, at the beginning of the vegetation (protection of young s hoots) and at the maturation (protection of the bunches). The spots caused by *Plasmopara viticola*, particularly in the tender organs, either shoots, leaves or young bunches but, on those spots appears a white grainy pubescence, whereas in *B. cinerea* the pubescence is dusty, greyish and the leaf spots are usually triangular near the edges.

Botrytis (Grey rot, Grey mould)

The conidiophores and conidia form a greyish mould on the affected organs; the sclerotia are easily visible as black pustules that appear in the elongated spots in the shoots, in autumn



Figure 6: Botrytis

The fungus can be isolated and grown in artificial nutritive media. Necrotic spots triangular in shape formed at the edge of the leaves; brown elongated spots, in the young shoots (that may break), turning black at the maturation due to the formation of sclerotia; total or partial destruction of the bunches before flowering and later on, infection through the wounds, by contact between berries or by infection of the rachis and peduncule. The fungus may overwinter as a saprophyte in dead plant organs or in the bud scales mainly as sclerotia. The infection takes place from these, forming new mycelium and conidia which will germinate in the young shoots, leaves and bunches (direct infection) or through wounds or dead organs (floral pieces); in maturing bunches, infection takes place through the wounds or by contact between berries.

State of the art

The principal systems to forecast the insurgence of these pathogens are based, like the rule of three ten for the peronospora's case, on the monitoring and elaboration of the agro-meteorological parameters.
Some studies have been made on models of simulation for the insurgence of pathogens in the vineyard (1-3). Some software for the elaboration of the agro-meteorological parameters already exist. A good example is the **Metwin II and MicroLink** (http://www.goelgroup.com/pessl.html). Such software is able to forecast potential infection of the main pathogenic of the grapes like:
Downy mildew (Plasmopara viticola); Powdery mildew (Uncinula necator); Grey mould (Botrytis cinerea).

Conclusions

In lack of direct sensors for the measurement of the mentioned pathogens, a possible fruitful approach consists in using and developing forecasting models bases on indirect measurements of meteorological conditions, such as: air temperature, air humidity, wind direction and speed, rainfall.
This system is valid and widely used. In the GoodFood context, a dedicated software will be developed to increase the prevision level of the pathogens' onset and consequently to intervene with aimed measures, allowing a remarkable economic saving (use of pesticides just in case of real and incipient hazard risk) and an improvement of the quality and safety of the product.

3.1.2 Cellar

3.1.2.1 Cellar temperature and humidity

Sensirion offers digital humidity sensor systems with integrated temperature sensors. These sensors are developed with an innovative technology: CMOSens®, this technology let to integrate two sensors and readout circuitry on one single chip. The digital humidity and temperature sensor is fully calibrated and offers excellent long-term stability and ease of use at very low cost and user friendliness. Each SHT11 is individually calibrated in a precision humidity chamber with a chilled hygrometer as reference. The calibrated coefficients are

programmed into the OPT memory. These coefficients are used internally during measurements to calibrate the signals from the sensors.

Table 16

Name of the Manufacturer	SENSIRION: SHT11	
Sensors	Temperature	Humidity
Measurement Range, Accuracy and Resolution	**Range** Temperature -40 to +123.8 °C **Accuracy** Temperature ±0.5 °C @25 °C **Resolution** Temperature typical 0.01 °C (14 bit) **Response time** Temperature from 5 s min. to 30 s max. [1/e (63%)]	**Range** Relative Humidity: 0 to 100% RH **Accuracy** Relative Humidity: ±3.5% **Resolution** Relative Humidity typical 0.03%RH (12 bit) **Response time** Relative Humidity 4 s [1/e (63%) slowly moving air]
State of Development	Commercially available	
AmI compatibility	Compatible though the interfacing of the console to the mote or to the gateway.	
Data output	digital data.	
Power Supply (V) Consumption (A)	Device: from 2.4 VDC to 5.5 VDC Measuring: 550 μA; Average: 28 μA; Sleep: 0.3 μA [@25 °C VDD=5V] Output peak current: on 4 mA; Tristated 10μA [@25 °C VDD=5V]	
Size and Weight (Length×Width ×Height)	Device size is: 7.42 mm × 4.88 mm × 2.5 mm (0.29 x 0.19 x 0.1 inch) Device weight: 100 mg	
Cost (Euro)	Device price: 17.69	
Image of the Device		

Further output of the performed state-of-the-art analysis of existing and commercially available sensors to monitor cellar's environmental parameters (temperature and humidity), some suitable products fitting well the already defined AmI compliant features have been identified. (Need to re-write this section)

Temperature sensors:

Table 17

Name of the Manufacturer (sensor)	Maxim (DS1626, 8 pin μ-MAX Package)	Texas Instrument (TMP121, SOT23-6 Package)	National (LM74CIM, SO8-Package)
Sensitivity	Range: from –55°C to 125°C Accuracy: ±0.5°C from 0°C to +70°C Resolution: is user configurable from 9 to 12 bits, this last corresponding to temperature increments of 0.0625 °C	Range: from –40°C to 125°C Accuracy: ±1.5°C from -25°C to +85°C Resolution: 12 bits, that corresponding to temperature increments of 0.0625 °C	Range: from –55°C to 150°C Accuracy: ±1.25°C from -10°C to +65°C Resolution: 12 bits, that corresponding to temperature increments of 0.0625 °C
State of Development	Available on the market.	Available on the market.	Available on the market.
AmI compatibility	Low Power sensor (less than 5mW), digital and serial output. No I2C output.	Low Power sensor (less than 5mW), digital and serial output. No I2C output.	Low Power sensor (less than 5mW), digital and serial output. No I2C output.
Data output	Digital output: is a 3-wire serial interface	Digital Output: SPI-Compatible Interface	Digital Output: SPI/Microwire Compatible Interface
Power Supply (Volt)	Supply voltage: 2.7V to 5.5V	Supply voltage: 2.7V to 5.5V	Supply voltage: 2.7V to 5.5V
Size (mm×mm)	Size: 3×5 mm	Size: 3×3 mm	Size: 5×5 mm
Cost (Euro)	3÷4	1	1

Humidity sensor

Table 18

Name of the Manufacturer (sensor)	Sensirion (SHT15 humidity+temperature, 8 pin LCC)	Honeywell (HIH-3610, 3 Pin SIP Package)
Sensitivity	Humidity sensor Range: 0÷100% RH Accuracy: ±2% from 10 to90% RH Resolution: 12 bits, that corresponding to humidity increments of 0.03% RH Temperature sensor Range: from -40°C to 125°C Accuracy: ±0.4°C from +5°C to +40°C and ±1°C from -20°C to +70°C Resolution: 14 bits, that corresponding to temperature increments of 0.01 °C	Range: 0÷100% RH Accuracy: ±2% from 0 to 100% RH (with Vsupply = 5 V) Sensibility: 30.680mV / %RH
State of Development	Available on the market.	Available on the market.
AmI compatibility	Low Power sensor (less than 5mW), digital and serial output. No I2C output.	Low Power sensor (less than 5mW), No digital and I2C output.
Data output	Digital output: is a 2-wire serial interface	Analog Output: from 0.8 to 3.9V
Power Supply (Volt)	Supply voltage: 2.4V to 5.5V	Supply voltage: 4V to 5.8V
Size (mm×mm)	Size: 7.5×5 mm	Size: 9.5×4.5 mm
Cost (Euro)	15-20	< 20

3.1.2.2 Fermentation of must

Ethylic alcohol

Concerning the ethylic alcohol measurement no AmI compliant sensors have been identified. The only instruments available on the market as alternative to the laboratory analysis are breathaliser and disposable biosensor

Glucose:

A wide number and range of portable devices are available on the market for glucose detection, but they are mainly biosensors aimed to measure glucose in

the blood of diabetic people. Moreover, these biosensors are disposable sensor and therefore unsuitable for a continuous monitoring. In alternative, there are dedicated kits for chemical analysis that use amperometric systems for the determination of the glucose concentration (e.g. perBacco from Biofutura s.r.l.; http://www.biofutura.com).

Temperature:

Table 19

Name of the Manufacturer	Texas Instruments TMP175
Sensors	Temperature
Measurement Range, Accuracy and Resolution	**Range** Temperature -40 to +125 °C **Accuracy** ±1.5 °C (max) from -25 °C to 85 °C ±2.0 °C (max) from -40 °C to 125 °C **ResolutionTemperature** 0.0625 °C 9- to 12-Bits
State of Development	Commercially available
AmI compatibility	Compatible though the interfacing of the station to the mote or to the gateway needed to be investigated
Data output	Two-Wire Serial Interface
Power Supply (V)	Device: from 2.7 VDC to 5.5 VDC
Size (Length×Width ×Height)	Device size is: 5 mm × 5.80 mm × 1.75 mm (0.197 x 0.228 x 0.069 inch)
Cost (Euro)	Device price: 0.85 $

Malic acid:

For the analysis of malic acid there are not any AmI compliant sensor available on the market and at present the measurements are realised by using chemical kits for analysis based on amperometric systems (Biofutura s.r.l.).

3.1.2.3 Ageing of Wine

Loose SO_2:

At present, there are any sensor for the measurement of the sulphur dioxide in wet phase. There are only sulphur dioxide sensor for gas. As possible solution, if the barrel is a closed system, it could be possible to perform indirect measurements taking advantage of the fact that the wet phase is in chemical equilibrium with the gaseous phase. Following this scenario, a suitable sulphur

dioxide sensor is SO_2-AF (SO_2-BF, SO_2-AE) from Alphasense (http://www.alphasense.com).

Total SO_2:

Analougus to Loose S02

ACETIC ACID

Any AmI compliant sensor for the analysis of acetic acid have been identified; also in this case, the measures are performed by using dedicated kits for chemical analysis using potentiometric systems for the determination of the volatile acidity concentration. (Biofutura s.r.l.)

3.1.3 Contribution from other WP to Wine Chain

During the GoodFood Meeting held in Cork at the end of June the WP 1-6 offer their contribution in terms of expertise as sensors supplier.

Comments on possible contribution are described in the following.

WP1. ANTIBIOTIC DETECTION

Antibiotic detection in viticulture is absolutely not necessary, as the use of antibiotics in agriculture is forbidden.

WP2. HERBICIDES AND FUNGICIDES

Atrazine is not used as a herbicide in viticulture, but only for field crops and ornamentals (e.g. corn, wheat, …). Moreover, triclorophenols are the most important component of some fungicides (e.g. procloraz against *Oidium spp.*) employed in field agriculture (e.g. cereals) but not in viticulture.

WP2. PESTICIDES

We are greatly interested in pesticides detection in viticulture**.** WP2 list of molecules will be helpful to verify their use in viticulture (e.g. Metalaxil, Penconazol, Lufenuron, Metidathion, etc.) and the possibility to find them into the final product (wine).

WP3. OCHRATOXIN

Ochratoxin is one of the fundamental aspects for modern viticulture and traceability in wine-chain.
First, our experience and knowledge suggest the importance of development of specific ochratoxin sensor for must and wine (in fact, legal limit for wine chain are referred to these two products). Our suggestion is to follow this approach.
Second, predictive meteorological models are both attractive but unpractical, since different fungi can grow with the same meteorological conditions. On the other side, optimal climatic conditions could not always improve fungi

development, due to other environmental sub-optimal factors. For this reason, virtual sensor comparison with field data (actual sensor) will be important, but only when more information and knowledge about ochratoxin production be will be available (see OCHRAWINE).

WP5. ALCOHOLIC DEGREE
We are greatly interested in continuous alcoholic degree measurement in wine. Please, let inform us when the description of the technical specifics will be available.

WP5. CO2 EMISSION, HUMIDITY AND ALCOHOL MEASUREMENTS
CO2 emission: **important** for must fermentative process monitoring.
Humidity into wine: not relevant.

WP5. GASCHROMATOGRAPHY
We suggest Mr. Siciliano to contact us as soon as possible, in order to decide which parameters should be measured by their system.
Indicatively, this is a list of the possible molecules to be monitored both in must and wine:

- malic acid
- polyphenols
- loose and total SO2
- acetic acid
- ethylic alcohol
- anthocyanins
- tartaric acid

4 High-level System Design Architecture

4.1 High-level System Design Architecture

Many human activities can greatly benefit from the convergent impact of a number of enabling technologies referred to as Ambient Intelligence (AmI). First prototypes of AmI systems have been developed, but the realisation of true AmI calls for much additional research of multidisciplinary teams. Most common applications of AmI span from travel and tourism to the in-home or office automation, but a lot of futuristic scenarios may be identified. One of this is the application of AmI concept in the agro-food field. It is aimed to guarantee security, quality and reliability of a complete food-chain (from land to market) through an infrastructure with the following major components (Figure 7):

- a distributed wireless network of sensors, actuators and portable micro-systems at each step of the food-chain in a large-scale distributed for the control of raw materials, the improvement of food

processing, the improvement of food logistics during transport, storage and vending, the control of the quality and the detection of possible adulteration of final product;

- an efficient, reliable and secure communication infrastructure;
- a software architecture oriented to the application level built with databases, knowledge base and intelligent systems for storing and elaborating safety, quality and traceability related information;
- intelligent and intuitive interfaces, able to support the acquisition and information spreading of healthiness, quality and traceability of food.

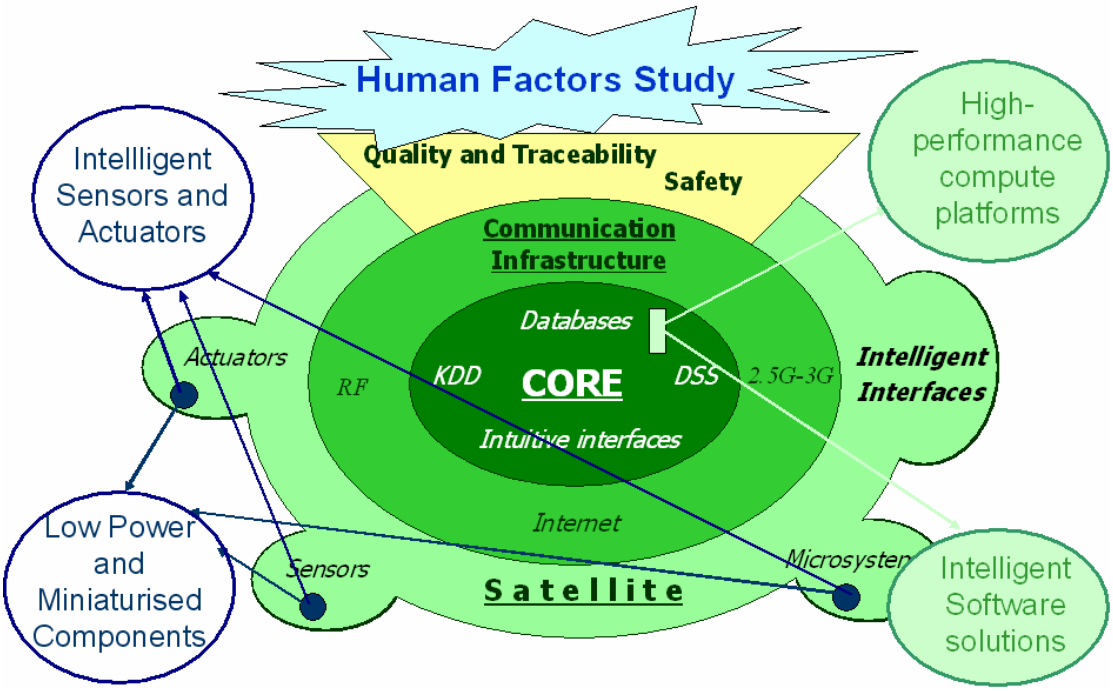


Figure 7: Ami Architecture

4.2 Context

The general high-level overview of system architecture proposed for the M18 pilot site implementation is briefly described.
The proposed architecture is suitable for monitoring the process in both the cellar and the vineyards.

The system is organized in two main groups of blocks:

The *Remote Data Acquisition and Analysis Tools*
- Wireless sensor monitoring networks;
- Stand alone image video monitoring tool

- Stand-alone tools (data sensing equipments, such as weather stations and portable analysis instruments).

The *Data Management and Protocol Handling Unit*

- Protocol Handler Unit
- Central Storage & Processing Unit

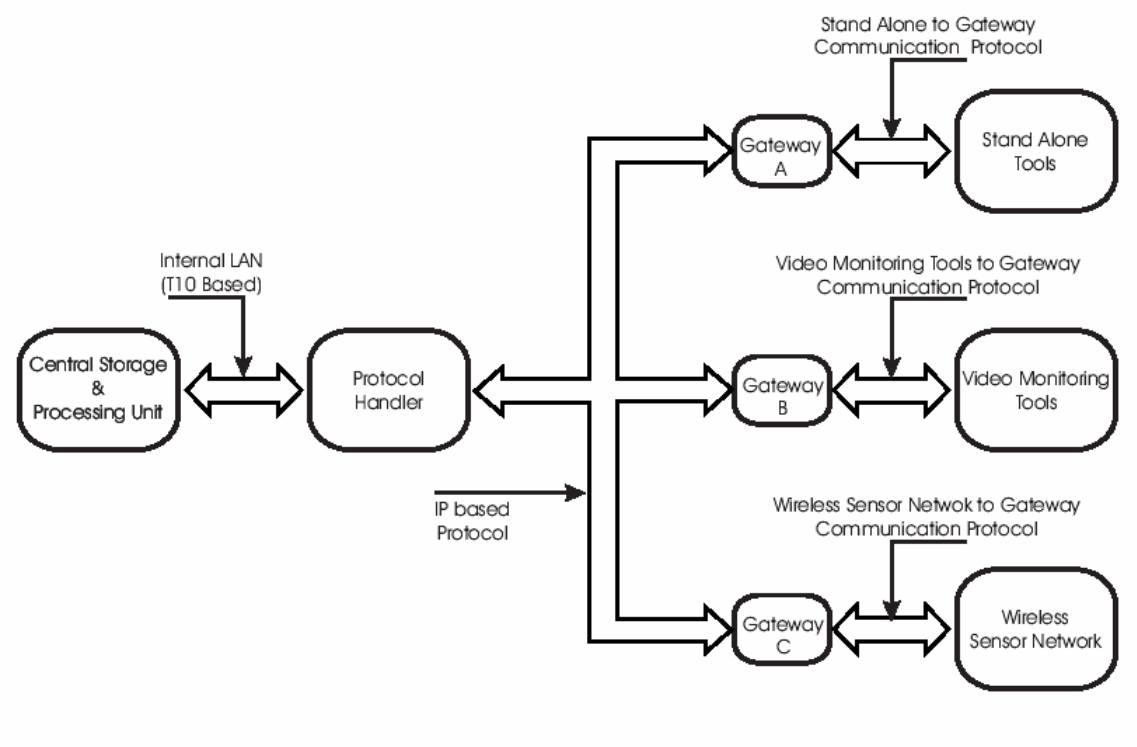


Figure 8: High Level System Architecture

Each of the Remote Data Acquisition and Analysis Tools is directly connected to a single stand alone Gateway making use of a specific Custom Communication Protocol.
The Gateway acts both as a data collector and interface with the protocol handler, managing, for each Remote Data Acquisition and Analysis Tools, the communication from and through the Protocol Handler Unit.
Each of the Gateways, making use of specific physical communication layer (GSM/GPRS, LAN, ADSL, etc.), is properly interfaced to the Central Storage and Processing Unit, through a custom defined IP based communication protocol, designed to match the specific operational requirements.
The Protocol Handler module is designed to implement all the communication procedures and protocols, in order to provide transparent operations between all the different remote subsystems and human to machine software interfaces.
User interfaces, including data mining and processing support components, will be generally provided through the central storage and processing unit; direct user interaction with local subsystems are also allowed.
The architecture described above has been devoted to the M18 trial Site in Montepaldi, but is delivered in order to ensure system up-scaling and down-scaling.

Moreover the architecture is delivered as an Open one, in a way that new Stand Alone Tools and communication protocol, as well as different physical communication media, could be simply integrated in the described structure when available and requested by the user.
The proposed architecture is intended to be compatible with different food chain and different process to be monitored.
Specific communication protocol, interfaces and communication layer could be implemented to make sure the correct applicability to different process.
Each of the Remote Data acquisition and Analysis Tools, here described as a simple block, is a subsystem whose architecture and functionality will be described in the following.

4.3 Interfaces

As described in the previous paragraph, the system makes use of different interfaces to enable data and command exchange among the different subsystems.
The system makes use of hardware/software or only software interface and, in both the cases, custom or Standard based communication protocols will be implemented.
In the high-level block diagram of system architecture a certain number of interfaces are outlined:

- Central Storage Unit versus Protocol Handler interface:
  This interface is based on a standard LAN communication system or equivalent. Standard communication protocols will be implemented.
- Protocol Handler to Gateways interfaces:
  The physical layer Interface is based on a generic IP network communication protocol. The protocol handler is designed to communicate with the gateways through his standard LAN interface. On the gateway side, the communication interfaces will be implemented making use of one of the following possible solutions:
  a) GPRS/EDGE data modules for outdoor applications;
  b) Ethernet controllers for indoor LAN/WLAN equipped environments;
  c) ADSL bridge + Ethernet, or PSTN controllers for other indoor environments Custom handshaking procedure will be implemented.
  . General equipment (stand-alone tools, Video Monitoring tools, Wireless Sensor Network) to gateway interface:
  Those interfaces are custom, low-level interfaces both software and hardware. Specific communication protocol for data handshaking will be implemented (IEEE 1451 as well as Custom Protocols will be taken into account).

4.4 Subsystems

Figure 8 shows the M18 trial site system level architecture. Three different type of data acquisition subsystems will be implemented in the trial site.

At least two Wireless Sensor Network subsystems (one for the vineyard and one for the cellar), up to ten Video Monitoring Stand Alone Tools for the colour variation monitoring of the grapes and a suitable number of Stand Alone Tools will be implemented in the M18 Trial Site.
Those subsystem are :

- The Wireless Sensor Network.
- The Stand Alone Tools.
- The Video Monitoring Tools.

Each of those subsystems is connected to the Protocol Handler through its proper Gateway making use, as described in the previous paragraph, of specific Interfaces and Communication Protocols or on well defined and known standards or custom developed.

4.4.1 Wireless Sensor Network

This will be the main Subsystem for both the vineyard and cellar monitoring application. The Wireless Sensor Networks (WSN) subsystem will be implemented by making use of a grid of nodes that communicate and exchange data using radio links.
The routing is implemented on the base of the multi-hop communication protocol implemented in TinyOS Operating System.

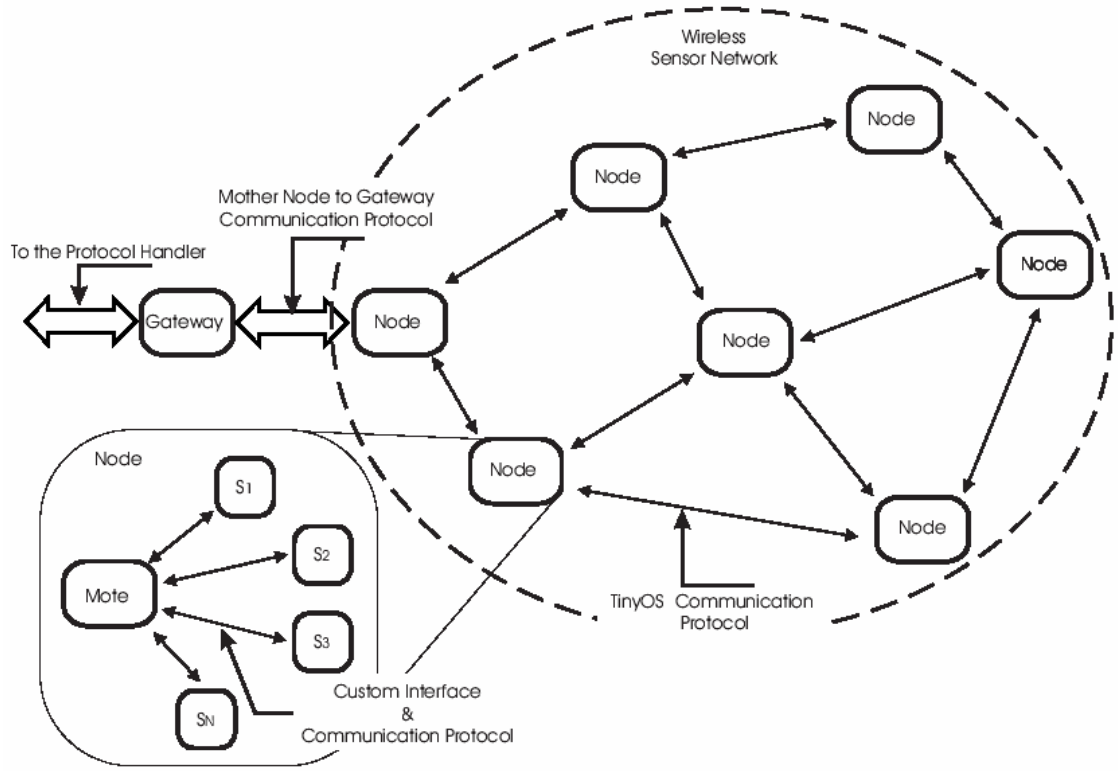


Figure 9:Wireless Sensor Network Architecture

A variable number of sensors (up to 16) will be implemented in the node in order to provide the necessary monitoring functionalities.
Data read from the sensors are internally routed, through the nodes of the network, to the Mother Node, the node directly connected to the Gateway, and the Gateway will provide to transfer them to the Protocol Handler.
As described in the previous paragraph, the Protocol Handler will supply data to the Central Storage and Processing Unit for post processing.
A Mother Node to Gateway Communication Protocol will be implemented in order to ensure the data transfer among the Mother Node and the Gateway.
This Protocol will apply a custom version of the TinyOS data format.
As showed in Fig. xy each node will be put into operation making use of a Mote (a battery supplied board with a microcontroller, a memory unit for data storage and a transceiver for communications) and the requested number of sensor.
The use of TinyOS Operating System will promise to develop application independent from the specific hardware platform (Mote).

4.4.2 The Stand Alone Tools

The Stand Alone Tools

The High Level Architecture of the Trial Site foresees the use of some Stand Alone Tools that will communicate, directly, with the Protocol Hendler through a proper Gateway, making use of a specific Communication Protocol.
The Stand Alone Tools will consist of some highly complex multifunctional sensors such as DNA sensor for pathogenic diagnostic or meteorological station and similar.

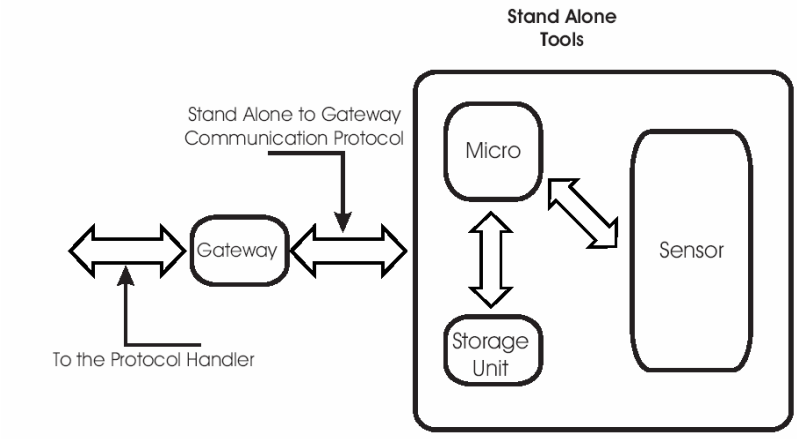


Figure10: Stand Alone Tools Block Architecture

Such a sensor can not be connected directly to the WSN due to their complexity, power consumption and data size that have to be transfer to the Central Storage and Processing Unit.

Stand Alone Tools, typically, will be implemented making use of the a Micro controller (devoted to manage both the functionality of the sensor and the data exchange with the Gateway), the sensor and a Storage Unit used to accumulate data from the sensors making them available when necessary.
Data pre-elaboration will be also possible inside the Stand Alone Tools.

4.4.3 Video Monitoring Tools

Video Monitoring Tools is another Stand Alone Tools included in the M18 Trial Site. Up to ten Video Monitoring Tools will be deployed in the trial site.
As described in the specific paragraph dedicated to the sensor specification this Subsystem consists of an Imaging Sensor connected directly to a Video Processor. No Storage Unit for image storage will be available on the PCB.
The Imaging Sensor will be a low cost CMOS imager controlled by a video processor dedicated to process and elaborate the images.
The output Image will be in a standard format and will be transmitted to the Gateway using a specific Video Monitoring to Gateway Communication Protocol.
The Gateway will provide image transmission to the he Protocol Handler.
Fig. 11 illustrates the general Architecture of the Video Monitoring Tools and highlights the connection with the Gateway.

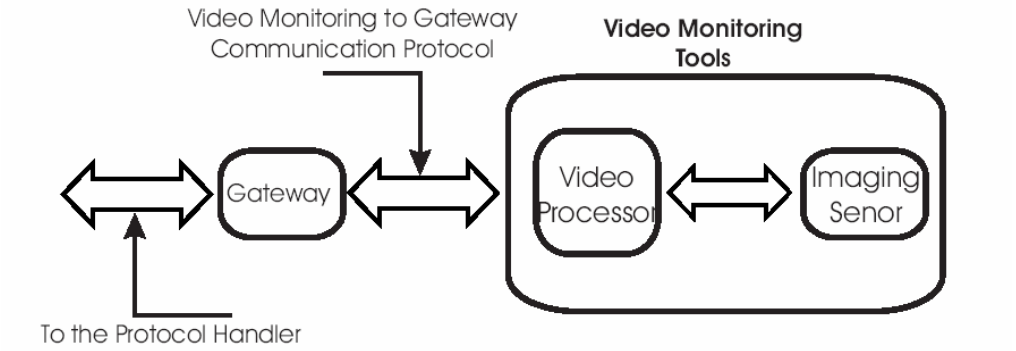


Figure 11: Video Monitoring Architecture

4.4.4 The AmI Application level

As described in the previously section, a Sensor Web is composed of a set of sensors. One group of related sensors is controlled by a micro device. In the GoodFood AmI infrastructure this device is a MICA Mote. A Mote device is steered by an operating system called Tiny OS. All the Mote devices are interconnected through a wireless network with a given topology. There is a particular Mote that represents the interface to the AmI Core. This Mote is called MICA Mote Gateway (GW Mote).

As shown in Figure 12, the GW Mote is connected through a wireless connection to a PC which is called Protocol Handler (PH). The PH is responsible for wrapping raw data that comes from the Sensor Web or low level requests to the Motes into a structured stream of information that travels between the Sensor Web and the high level AmI Core.
The AmI Core is connected to the PH through a bus that contains the logic to distribute information coming from the Sensor Web to the related AmI modules. As examples for the AmI Core modules:

- A KDD (Knowledge Discovery in Database) system for analysing structured information with data mining techniques to find some interesting correlations.
- A DSS (Decision Support System) that helps the AmI framework in taking any sort of decisions according to the knowledge acquired with the analysis of the stored information.
- The file system for storing data structured in files.
- A database for storing and retrieving data in a more structured way according to a database model.

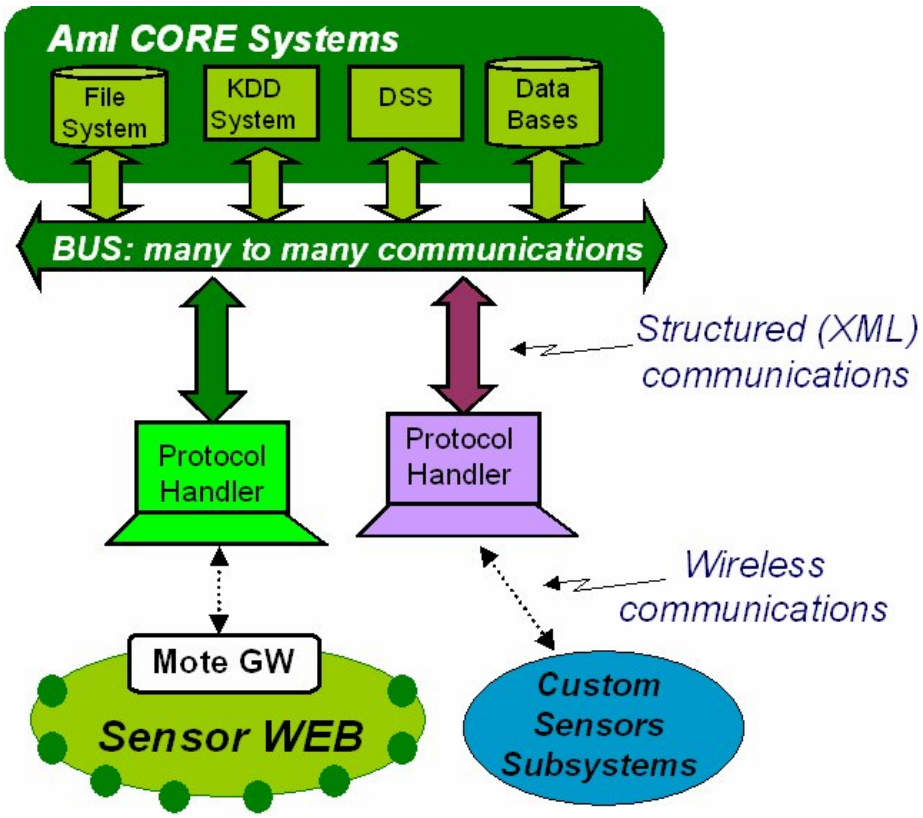


Figure 12

Actually the bus should be seen as a logical layer that treats information coded in the high level, XML/HTTP based protocol. This information is represented by:

- resource management related information exchange, global sensor data retrieval and (possibly) request of actions to be taken by the system actuators.
- result Data produced by the sensor subsystem following a query Command Message posted by the AMI Core.
- notifications of information that is actively produced by the sensor subsystem when significant events occur in its environment.

A bus architecture is preferred to simple peer-to-peer connections because it focuses on the presence of a logical layer that matches the many-to-many communication pattern, governs access to structured information streams according to the different responsibilities of the communication partners and adds a further degree of modularity because the AmI Core can be extended with other systems.
Figure 12 shows an interface oriented view of the AmI framework. In this view, the PH exposes one interface to the lower level sensor system which can be the Sensor Web or another sensor subsystem.

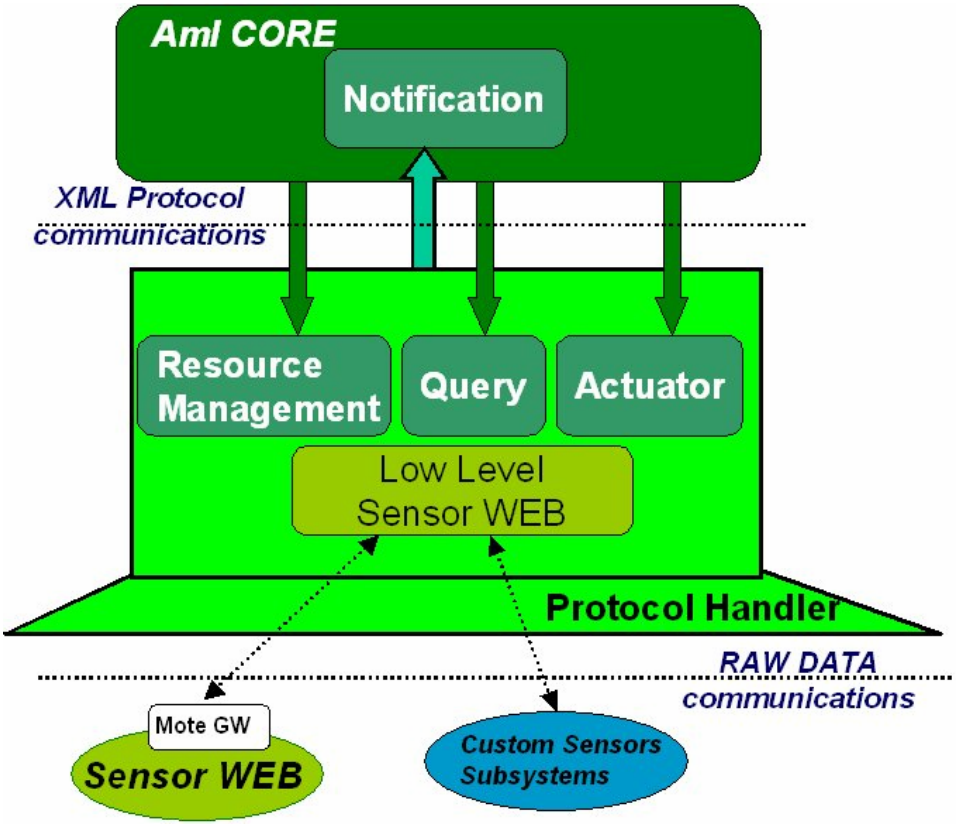


Figure 13

This interface implements bidirectional communication of raw data represented between the sensor subsystem and PH. The term raw data is referred to the protocol and not to the semantics of a communication content. This means that a floating point measure (either directly measured or evaluated by an arithmetic operator), a configuration command message, a notification all are sent or received in a binary representation and not, for example, in a text representation (like XML) which would be too verbose.
PH and the AmI Core expose a set of interfaces for communicating between each other. Interactions with these interfaces are performed through the high level protocol based on XML/TCP standards that will be implemented for the AmI framework.

- The Resource Management interface is used by the AmI Core for remotely querying or changing the configuration of the sensor subsystem. The implementation of the Resource Management interface could be adapted from already existing tools such as resource management tools used in GRID environment.
- The Query interface is used by the AmI Core for sending queries on the data of the sensor system environment.
- The Notification interface exposed by the AmI Core is thought to be the entry point for the notification of events occurring in the sensor subsystem. It is used by the PH to forward notifications coming from the sensor subsystem.
- The Query and the Notification interfaces are the main gateways for the information that feeds the AmI Core analysis systems such as KDD and DSS and storage systems like databases.

As an example of how the entire structure could be extended, a further interface, the *Actuator* interface is shown. This interface could be used by the AmI Core to send commands to the PH who forwards them (translated to raw commands) to the actuators of the sensor system. As shown, it is possible to extend the information flow among the different layers of the architecture by adding interfaces to the responsible modules.
A PH could be provided with meta information regarding its interfaces to allow the AmI Core to query the actual capabilities of the underlying sensor system. A standard mechanism like web services could be used for this purpose avoiding any conflicts with the choice made for the high level XML based transport protocol.
The shown architecture aims at the following targets:

- **Minimization of the communication overhead** inside the Sensor Web. All the data sent or received among the Motes and from the GW Mote to the PH is raw data. This is meant to reduce the overhead, in terms of amount of data, due to a higher level, structured protocol. In this way the following points are optimised:
  - communication Performance – the messages sent are smaller thus requiring less network time.



  - energy Consumption – communication requires much more energy than computation on a MICA Mote.
  - cost – cost of communication is evaluated on a traffic basis.
  - computation resources – no computation power is used for transforming raw messages into higher level protocol messages.

- **Modularity**. Whole sensor subsystems, possibly created with different low level technologies, can be plugged into the AmI framework by simply supplying a dedicated implementation of the PH. At the topmost level other systems (like data bases or analysis systems) can easily be added because the bus architecture will be designed with flexibility as a main requirement.
- **Scalability**. Decoupling the low level communication from the high level protocol handling and assigning the high level protocol operations to a machine, the PH, that is typically a scalable machine (for example a fully featured PC) makes it possible to manage consistent quantitative changes in the sensor subsystem.
- **Robustness**. The Sensor Web is composed of devices that communicate through a wireless mechanism. Moreover the single devices are often exposed to risks related to their positioning or to extreme weather conditions. Isolating the Sensor Web and the lower level related communications from the AmI Core and its higher level protocol, through the use of the PH, introduces a further level of robustness between a safe environment (AmI framework) and a less safer one (Sensor Web).

4.5 The Data Workflow

The logical exit point of the sensor network towards the higher levels of the architecture, is represented by a microcomputer with a special connection to the Protocol Handler (PH).
The PH is a full blown personal computer that accepts instructions from the AmI framework to interact with the lower level sensor network and receives observed data that has to be forwarded, using the bus-like communication infrastructure to the framework tools for storage, analysis and monitoring.
The key factors that have to be discussed when focusing on data modelling, transport and storage regard the protocol that wraps low level sensor data for transport across the AmI bus and the data structures used for storing the information inside the AmI databases.
The protocols used for communications inside the sensor network have to consider many aspects related to energy consumption, wireless specific transmission safety, routing and localization.
Thus the specification of the lower level inter-node communication protocol is out of the scope of this discussion and is left to the specification effort of the sensor network domain experts.
The interface between the lower level protocol and the higher level protocol is located in the PH.

The PH is responsible for wrapping and unwrapping the low level protocol, using a more structured and more general higher level protocol that will be used for transport on the AmI bus and possibly for storage into the AmI databases.
In particular, the high level protocol should consider the following tasks:

- wrapping of the resource management information coming from "plug&play" sensors.
- Unwrapping of queries from the AmI framework systems or from human operators and related to resource management or data retrieval.
- Wrapping observed data coming from the sensors.
- Unwrapping of commands generated by AmI framework systems or by human operators and directed to the sensor network actuators.

Concerning the main requirements of the high level protocol, it should be:

- based on open standards,
- quite general and extensible since Good Food applications will focus on a great variety of different scenarios.

Moreover there should be tools for automating most of the tedious aspects related to parsing and binding of the protocol to other proprietary *back-end* protocols such as databases or GUIs.
The data produced by a sensor network will be streamed into the AmI layer according to the schema shown in Figure 14.

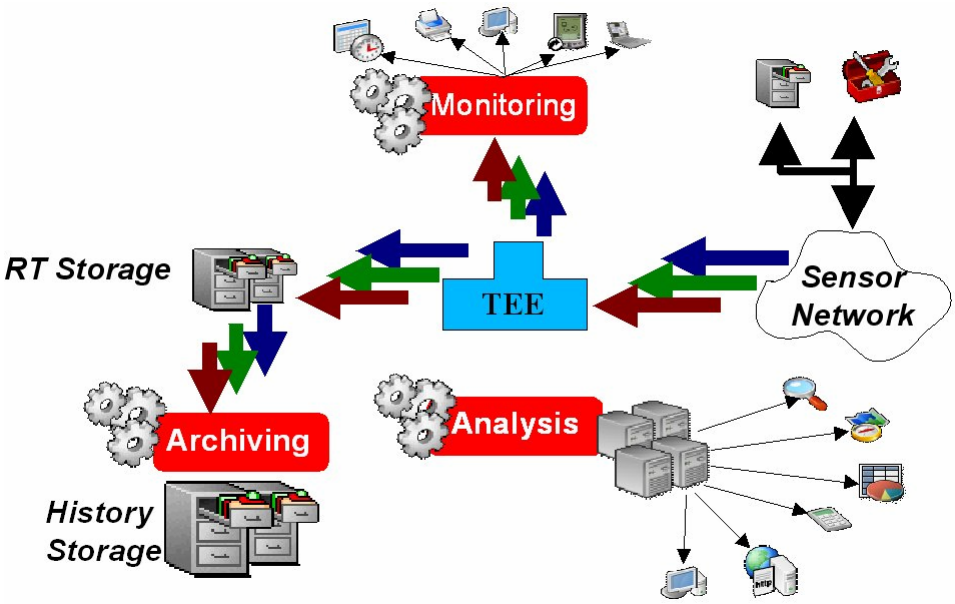


Figure 14

The real-time data flow is sent to a Real Time Monitoring (RTM) process that is able to visualize the data coming from a sensor or simple combinations of that data that are produced by operations who aggregate the data in both ways intra sensor or inter sensors.
The data is then streamed into a Real-time Storage that makes it persistent. However since the data produced by a Sensor Network might grow indefinitely, it useful to provide a historical

storage system where the data can be flushed periodically for long term storage. This operation is called *Archiving*. Archiving is a complex operation that involves the functional transformation of the data along with a timing policy that drives the operation.
Analysis involves both the storage systems (real-time and historical) and is intended to model the study of the acquired data to obtain cross comparisons among long time periods, decision and planning support, finding new patterns in natural phenomena, economically exploitation of the results and quality assurance. Finally all this information can be published on different presentation platforms.

4.6 The high level Working of the AmI-GRID platform

As shown in the previous section, the application level of the GoodFood AmI architecture includes a set of processing systems for monitoring, archiving and analyzing the data coming from the sensor WEB and from other sensor subsystems.
The Real Time Monitoring (RTM) process is able to visualise the data coming from a sensor or simple combinations of that data that are produced by operations who aggregate the data in both ways intra sensor or inter sensors. The data, that is fed into the RTM process, can be aggregated functionally (average, sum, factor) or chronologically (every hour, daily...). In particular, for the RTM system alarm/alert values can be defined, that is, the critical threshold values for the monitored parameters. The RTM process implements, moreover, the user interfaces to present the data and to notify other devices or the operators in case of topic events. The RTM process is also responsible for streaming the data into a Real-time Storage (RTS) to make the data persistent.
The Archiving process is responsible for defining:

- what kind of data, stored in the RTS, has to be archived,
- the archiving timing rules,
- the aggregation functions to perform for implementing consumptive archiving.

The destination of the data processed by the archiving system is a History Storage System (HSS).
The Analysis system contains:

- tools to implement a Knowledge Discovery in Databases (KDD) process to analyse and cross compare both real-time and historical data on the two storage systems, RTS and HSS;
- a set of Decision Support Systems (DSS) to value the results of the KDD process in terms of quality assurance, activity planning, resource optimization;
- a set of user interfaces to publish the results.

The information which will be processed in the Analysis system comes from various sources:

- RTS;

- HSS;
- Knowledge base derived from agro-food professionals.

4.7 The security in communication high level

Wireless networks, in general, are more vulnerable to security attacks than wired networks, due to the broadcast nature of transmission medium. Wireless sensor networks are even more vulnerable since the nodes are often placed in a hostile or dangerous environment where they are not physically protected.
A sensor network should not leak sensor readings. In many applications, nodes communicate highly sensitive data.
Since node data contain some confidential information, the content of all messages in the network must be encrypted. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers possess, hence achieving confidentiality.
Authenticated broadcast requires an asymmetric mechanism; otherwise any compromised receiver could forge messages from the sender. Asymmetric mechanisms have high computation, communication, and storage overhead, which make their usage on resource-constrained devices impractical.
This can be overcome by introducing asymmetry through a delayed disclosure of symmetric keys, resulting in an efficient broadcast authentication scheme.
A suite of security building blocks is optimized for resource-constrained environments and wireless communication. Security Protocol for Sensor Networks (SPINS) has two secure building blocks:

Secure Network Encryption Protocol (SNEP) provides data confidentiality, two-party data authentication, and data freshness.

µTESLA (the micro version of the Timed, Efficient, Streaming, and Loss-tolerant Authentication Protocol) provides authenticated broadcast for severely resource-constrained environment.

These protocols are practical even on minimal hardware, and can be used for building higher level protocols.
Unfortunately the above mentioned solution for the problem is only applicable if the sensorweb can be treated isolated from the overall system architecture. And even if only a secure communication between the sensor nodes has to be solved depending on the other requirements like spoofing, stealing, moving and so on additional features have to be included. For example protection against stealing could be solved with a clear and known topology of the sensorweb and/or the signal transmission delay – this requires in the worst scenario that all sensors have to know their exact location in the overall topology. In general it can be said that if more security has to be added more resources (storage, computing power, transmission power etc.) are needed in every node. In the following some aspects are listed that require a clear statement how they will be handled in the system. As far as possible a solution is presented or at least outlined.

4.7.1 TinySec for TinyOS

The TinySec link layer encryption mechanism provides a number of features that can be used to cope with quite a number of the above mentioned problems. Access control, integrity and confidentiality can be solved with TinySec. But there will be a high-level management layer for the setup and base configuration of the sensorweb. Every wine yard will be treated individually, which requires a different key, which will be statically assigned to the nodes and the gateways – a software download is necessary to use a node in a different environment. Another solution would be the possibility to load only a wine yard specific key onto the node.

Operational Scenario
The usages of the sensorweb should be done in two steps to guarantee security at a higher level. First the sensorweb will be setup and after this has been finished the production phase can start with the collection of data. The setup phase will be initiated by the gateways and its primary purpose is to collect the topographical information of the sensorweb. After this phase the exact topography of the sensorweb is known and every possible (required) communication path is defined. Every node knows exactly from which nodes it receives information and towards with node or gateway it has to send either the received or its own data. After this setup the gateway sets all nodes to the operational mode and starts data collection. Mobile nodes are possible in this scenario but can only communicate with the gateways (other solutions are possible and have to be discussed) which allow them to collect data on the field and transfer the additional data over either a gateway or directly in the core AmI system.
The actual specification of the TinySec package only specifies the type of algorithm that has to be used. There is the possibility to select the algorithm that fits the specific needs. (see TinySec: User Manual, Chris Karlof et. al. January 21, 2004)

End-to-End Security: Due to the fact that the TinySec package will be used and the limited resources end-to-end security will be solved on a wine yard specific or sensorweb restricted level. That means every node has the ability to de- and encipher the information that will be passed through them. The information about the communication path will be sent in plain which allows the collection of topology data but keeps the data secret.

Ease of sensor installation: Since the sensors will be wine yard specific the installation in the previously described operational scenario is very easy. The two modes (setup and operation) make the installation very easy and allow the reconfiguration or recollection anytime.

Topology: The gateway could store the overall topology but this is not necessary since the nodes know their communication partners (fixed during setup phase). The gateway can start anytime a sensor lookup/recovery to check

whether all sensors (or a specific sensor) are still functioning or have been moved / removed. This recovery process increases the availability and stability of the overall system.

Sensor functions: What information from the sensors (collected data) or the topology is necessary and can be transmitted plain? Data that are collected by the sensors are confidential but how are the location and node information treated? Tbd.

Sensor management: Online and offline management of the sensors has to be provided. Therefore reconfiguration of the topology initiated by the gateway or the management station must be possible.

Mobile sensors: Mobile sensors can collect information but do not use the sensorweb for the communication of the collected data. This communication is only possible through the gateway nodes which act as a control node for the whole topology. The concept of setup und operational phases would require too much overhead to integrate a mobile node into the network. Mobile nodes should be as flexible as possible in usage and position. Usually a mobile node is better equipped than a standard "field" node.

Management station: The management of the topology as well as the security of the data will be done centrally on the protocol station, the central storage unit or a dedicated station.

Gateway communication: The gateway acts only as the management station for the topology and the communication setup. It is able to initiate reconfiguration and key-changes requested by the management station.

Tempest: All information on the nodes has to be treated confidential and a simple solution for this problem would be the inclusion of a smart-card chip on the nodes. Also a software solution – storing only ciphered data – is also reasonable but requires an additional node-specific key.

Algorithms: Due to the fact that in most cases symmetric algorithms will be used the key management will be handled centrally. That means on every installation (e.g. wine yard) a central management station will be necessary. This station can and will be used also as the node that communicates with the next higher system. This can be the protocol handler or the central storage unit. The algorithm will be selected to fulfil the required needs, that could be synchronization, (node) computing power, compression, data to transmit and so on. The chosen hardware defines the constrains and their implications.

Based on the system architecture and the requirements on confidentiality, integrity, authenticity, authority and non-repudiation of data and communication as well as the resource restrictions a solution for all mentioned aspects can be fixed.
As long as the TinySec package will be uses only small changes are necessary – these modifications to the package will still maintain the interface compatibility

to the current version. Interoperability can only be provided if the algorithms won't be changed.
The management scenarios or currently under construction and will be discussed in the next few days.
The overall data confidentiality and integrity concept has a great impact on the possible data processing functions in the whole data path from the sensor to the governmental institution. Where can the data be viewed, processed, changed and so on.
Apart from the sensorweb also the AmIBus (as port of the AmI Grid) has to provide security functions. These functions act as basic middleware security and needs to be compatible and interoperable with the TinySec security. If this is not possible well defined security gateways have to be included to maintain a true end-to-end security from the sensor to the system management.

4.8 Preservation of openness and expandability of architecture

4.8.1 Software Abstraction Layers

In a traditional system, well established layers of abstractions are available. Independent applications at endpoints communicate point to point through routers.
The interface between the system and the user's application is given by an API (application programming interface).
To increase portability of application among various systems, common standards came into beginning. In the area of software under the aspect of abstracting the underlying operating system, two representatives are:

POSIX:
POSIX was designed as a standard environment to enable the portability of applications software and to some extent people. This portability of applications software is achieved through the specification of a set of services that every POSIX conforming application can expect to exist on a conforming platform.

ACE:
The ADAPTIVE Communication Environment (ACE) is an object-oriented (OO) toolkit that implements fundamental design patterns for communication software. ACE provides a rich set of reusable C++ wrappers and frameworks that perform common communication software tasks across a range of OS platforms, including Win32 (i.e., WinNT3.5.x/4.x, Win95, and WinCE), most versions of UNIX (e.g., SunOS 4.x and 5.x, SGI IRIX 5.x and 6.x, HP-UX 9.x, 10.x, and 11.x, DEC UNIX 4.x, AIX 4.x, Linux, SCO, UnixWare, NetBSD, and FreeBSD), real-time operating systems (e.g., VxWorks, Chorus, LynxOS, and pSoS), and MVS OpenEdition. A single source tree is used for all these platforms and porting ACE to other platforms is relatively easy.
A lot of other abstraction layers and standards exits, covering aspects of interoperability, communication between systems, and so on.

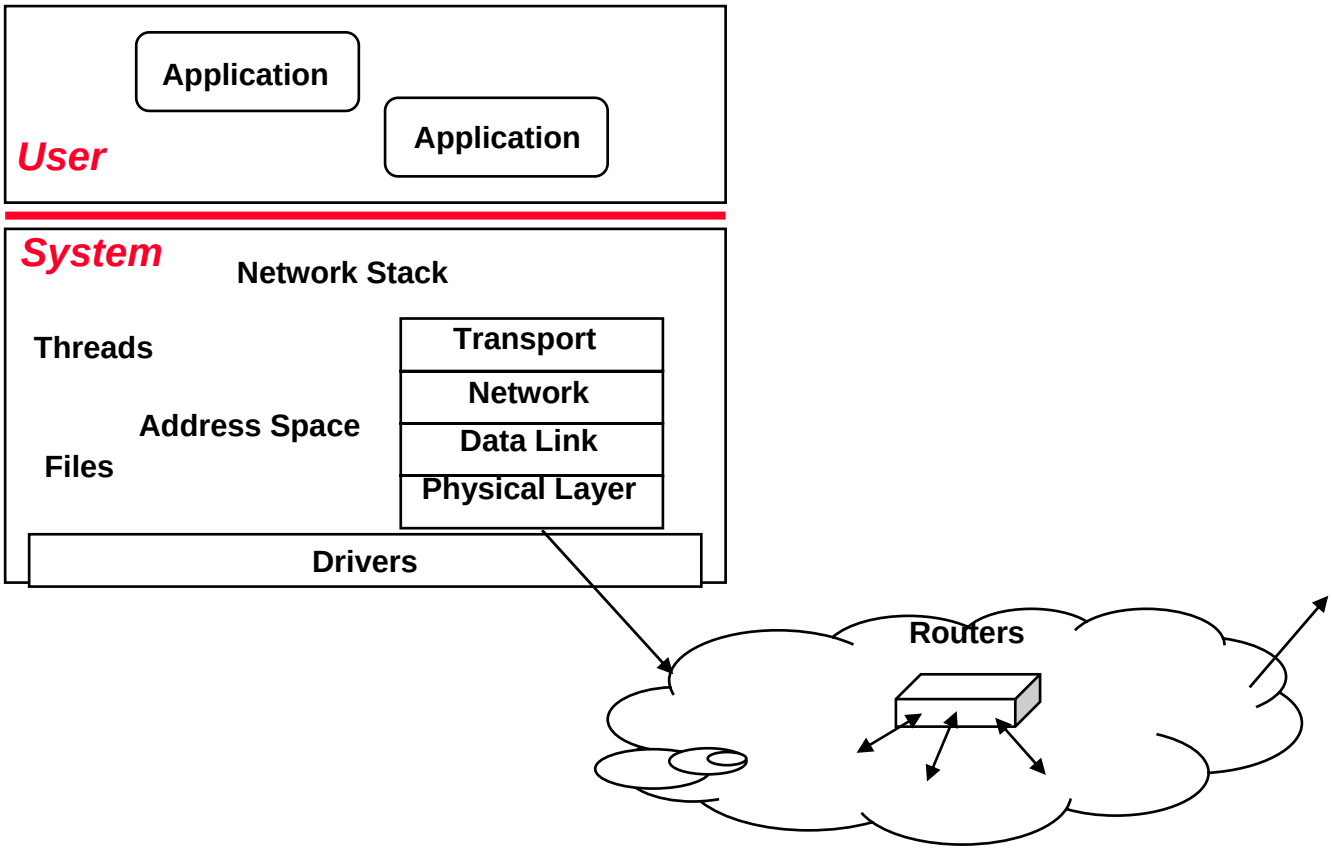


Figure 15

In a sensor network however, the following special requirements have to be considered:

- Highly constrained resources in processing, storage, bandwidth and power
- Applications spread over many small nodes
- Need for a self-organizing collectives
- Highly integrated with changing environment
- Network communication is fundamental
- Concurrency intensive in bursts: streams of sensor data and network traffic
- Robustness

The TinyOS is build to handle the above requirements. Its main concepts are:

- Two-level scheduling model: threads + events
- Components: Commands, event handlers, frames (storage) and tasks (concurrency)

- Storage model: Frame per component, shared stack, no heap
- Very lean multithreading
- Efficient layering

A typical TinyOS application looks like:

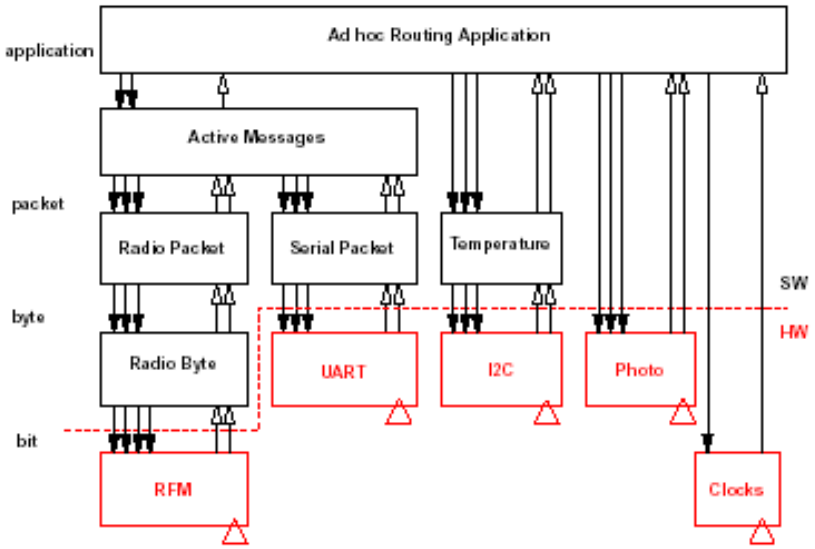


Figure 16

The TinyOS architecture defines interfaces between the components, allowing the replacement of a component by another one if the same interface is supported. Normally, a user of TinyOS just replaces the application component. However, it is also possible to exchange other components to fulfil special requirements.

Conclusion:
TinyOS is targeted as the operating system for wireless sensor nodes where little resources are available. Its main focus is the handling of messages, coming from sensors or other nodes.
TinyOS is not only an operating system but also defines abstraction layers in the form of interfaces between components.
With the selection of TinyOS as the operating system for the sensor nodes, it is possible to implement all the Goodfood requirements. It even seems that with TinyOS it is possible to change the standards for the wireless communication and to introduce/implement standards for interfacing the various sensors.

4.8.2 Tiny OS Porting

Recent work in the department of computer science in UCC has considered the porting of the TinyOS operating system onto the 25mm transceiver module. This

work has been carried out in collaboration with NMRC in the project D-Systems funded by Enterprise Ireland project (ATRP/02/227b).
The work describes version 0.2 of the libraries composed to support TinyOS V1.1 on the 25mm NMRC transceiver module. So far the support is limited to Active messages via radio and access to LEDs. The research of porting version 0.2 libraries to the 25mm transceiver module has concluded that:

- Active messages can only be transferred via radio;
- Communication support is embedded in the RadioCRCPacket component;
- The nRF2401 transceiver can be initialized for ShockBurst Mode;
- Data packets may be sent via Serial Peripheral Interface of the microcontroller;
- Pins E1, A0 and E0 can be toggled using the component LedsC;

The report also gives a tutorial on Obtaining, Installing and Running TinyOS for the 25mm module using the Linux RedHat 9 operating system.

4.8.3 Custom 868 MHz RF Wireless Platform

A custom version of the RF-microcontroller platform is now being designed. This module is based on the nRF905 RF transceiver from Nordic VLSI (http://www.nvlsi.no/), which works in the range from 430-928MHz as opposed to the nRF2401 2.4GHz transceiver used in the current module. The transceiver can also be used in conjunction with the Atmel microcontroller and should eventually support TinyOS. The nRF905 is available in a 32-pin 5x5mm QFN package, which has the same footprint size to that of the 24-pin 5x5mm QFN nRF2401 package used in the previous module.

4.8.4 Specification Overview of the nRF905

- 430-928MHz operation
- 100KHz frequency resolution @ 430MHz
- 200KHz frequency resolution @ 862-928MHz
- 1.9 - 3.6V voltage supply
- 0 - 50Kbit/sec data rate
- Programmable up 10dBm output power
- CRC and address decode
- ShockBurst mode

- 12.5mA Peak RX current
- Clock recovery on data
- -40 to +85°C temperature range

4.8.5 Antenna

The biggest disadvantage with the change in frequency selection is that the size of the antenna is increased. The options for the antenna are:

- chip antenna
- stub/whip antenna
- embedded loop antenna

4.8.6 Chip Antenna Options

The nRF2401 used a surface mounted chip antenna whose dimension allowed integration within the 25mm without any protrusion. The 868MHz chip antennas are about twice the size of the 2.4GHz equivalents, the Tricome (http://www.tricome.com/) version measuring 30mm in length (Figure 17) and an alternative, the MTX GreenChip™ (http://www.matrix.es/) slightly shorter ar around 25mm (Figure15). Due to the connector system format being implemented it is not possible to mount such an antenna within the 25mm frame so it would protrude slightly at best.



Figure 17: Tricome 868MHz Antenna

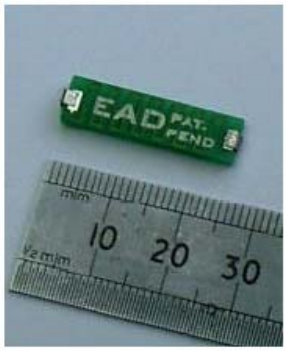


Figure18: MTX GreenChip™ 868 MHz Antenna

4.8.7 Stub/Whip Antenna Options

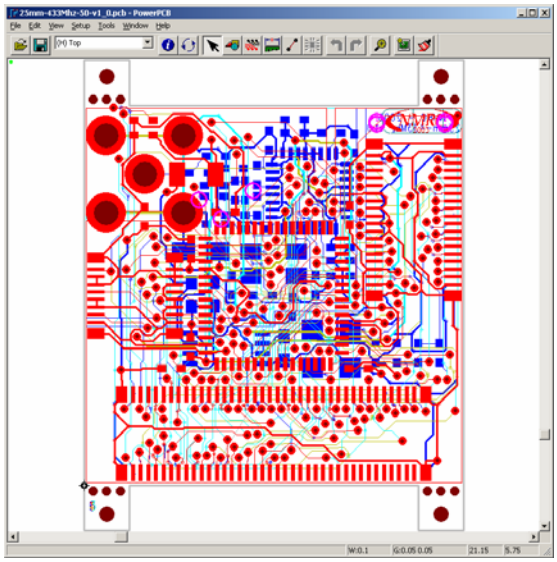


Figure 19: Incorporation of stub antena SMA footprint in the 25mm transceiver

Integration of a stub or whip antenna has the disadvantage that a fairly large area is required for its footprint and again the antenna will protrude from the module.
Figure 19 shows a possible floor plan for the modified module with the SMA connector for the antenna visible on the top left hand corner. The whip antennas range in height from around 40mm to 80mm, some examples are shown in Figure 20



Figure 20:¼ Wave whip antennas

4.8.8 Embedded Antenna Solution

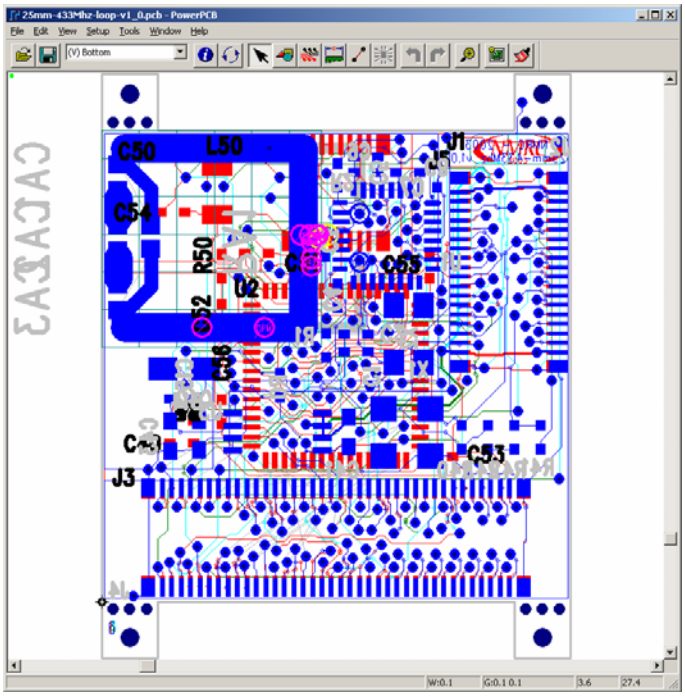


Figure 21: Embedded loop antenna floor plan

Perhaps the most interesting solution in terms of antenna volume is the incorporation of an embedded loop antenna into either the top or bottom PCB layer. The performance of such an antenna is however likely to be affected by shielding, noise effects of other layers and components in proximity thus limiting transmission range. The embedded option will however be investigated and an initial floorplan with the antenna in the top left corner is shown in Figure 21.

4.8.9 Preservation of openness and expandability of architecture versus ZigBee and IEEE 1451 Standards

The general architecture described in the previous section has been chosen in order to ensure the operability of the AmI system within the agreed project duration, but also to demonstrate the feasibility within the first 18 months. Therefore commercially available modules will be used for the implementation of the AmI system. However, the market situation is changing rapidly in this domain. It is therefore strongly recommended to preserve the openness of the architecture and its expandability towards novel solutions. Within this section a complementary approach will be presented, which focus on the use of existing standards and open source software, suitable to demonstrate the openness of the overall system architecture.

It starts with an overview over existing and future standards concerning sensor to networks interfaces, with emphasis on the interoperability and ´plug & play´ capabilitities This overview is followed by an extensive review of the ZigBee/802.15.4 options for wireless sensors, which is followed by a brief conclusion.

4.8.9.1 General considerations

Sensors present a extremely wide range of transduction principles. Similarly, sensors output can present many different of signals. The measurement can provide a modulated impedance value (resistive, capacitive and inductive sensors), or the sensor can actively generate a signal (for instance thermoelectric sensors). In the first case, analogue signal circuitry is needed to convert this impedance modulation to an electrical signal, typically voltage or current. In most cases, though, the signal amplitude will be low. Amplification is needed in most cases. To enhance resolution and profiting from the low pass nature of most measurands, proper filtering is also a must. In some cases, sensors are resonant so an oscillator is built and the output signal is frequency modulated. Signal demodulation or time measurement are an option in those cases. We may also found that the output signal is an AM signal (for instance LVDT sensors or modulated IR gas sensors).
In those cases, synchronous demodulation (Lock-in amplifiers) are needed. Despite these considerations, basic signal conditioning like amplification and filtering is common to a large family of sensors. We may also mention that controlled excitation is also required by many different sensors. In summary the signal conditioning requirements for sensors show a large diversity and are mostly designed case by case.

However, proper signal conditioning is just the first step in the signal pathway. In the digital world, and in particular in the framework of the Ambient Intelligence Vision, sensors signals should be acquired and converted to digital form, for processing, storage and transmission.
The key element in a ´smart sensor´ is the microprocessor, or more specifically, the microcontroller. This element contains a microprocessor core fitted with a variety of peripherals for interfacing to the external world including, but not only, A/D converters. Once the signal is in a digital form, communication capabilities are essential.

There is an increasing interest in smart sensors as a direct result of the need to communicate sensor information in distributed measurement systems. Unfortunately, numerous protocols have been defined for data communication: industry and consumer standards have been developed to satisfy different requirements.
Interoperability should allow the implementation of various interfaces across different networks for network independent operation. True transducer "plug&play" capability is the vision. The family of 1451 standards tries to address these problems.

4.8.9.2 Introduction to IEEE 1451.4

The amount of existing sensor and networking technologies and the cost of transitioning to the digital network standards defined in 1451.1, 1451.2 and 1451.3 are issues that the proponents of the IEEE 1451 standard are addressing. Analog sensors, with their existing wiring and the requirements for wide bandwidth analogue measurements are being addressed by 1451.4.
The proposed IEEE 1451.4 will establish a standard that allows those output-output, mixed-mode transducers to communicate digital information with a Transducer Electronic Data Sheet (TEDS).

A TEDS is added to a traditional two-wire, constant-current excited transducer with built-in amplifier or to other systems using an extra wire. When a transducer first comes online, it will go into digital communication mode and send out TEDS data for self-identification and control to the NCAP, using the same two wires used for analogue transmission.
Then it goes into analogue mode for normal operation.

The standard describes the following:

- A Transducer containing a Mixed-Mode Interface (MMI) and the TEDS.
- The MMI which is a two-wire, master-slave, multidrop, serial connection.
- The TEDS which is fixed and dynamic data contained in one or more memory nodes on the MMI.
- A Template which is a software object that define the specific properties for different sensor types. It describes the data structure of TEDS, is implemented in the Description Language and resides in the Transducer block (T-block).
- A Description Language to describe the functionality of the transducer.
- A T-Block located in the NCAP that handles the interpretation of the TEDS data for the end user.

A smart TEDS sensor as defined by IEEE P1451.4 includes a mixed-mode interface that
accommodates both an analogue signal (for measurement signal) and a serial digital channel for accessing the digital TEDS information. There are two types of mixed-mode interfaces, defined in the standard – Class 1 and Class 2 interfaces.

Class 1 interfaces are primarily intended for constant-current powered piezoelectric transducers (accelerometers, microphones, etc.) and define a scheme for sequentially switching between analogue mode and digital TEDS mode on a single pair of transducer wires.
Constant-current powered transducers, generally referred to as integrated electronics for piezoelectric transducers (IEPE), incorporate internal signal conditioning powered by a constant current that is sourced by the measurement system on the signal wires.

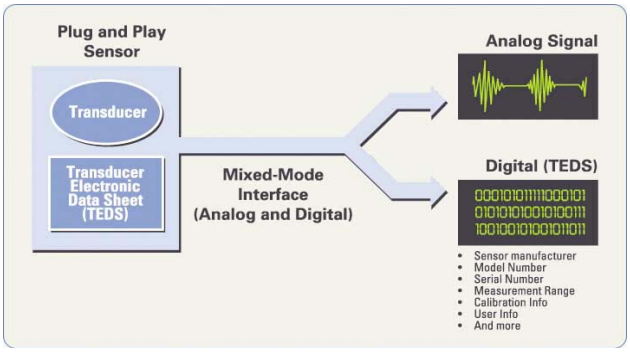


Figure 22:. IEEE P1451.4 defines a mixed mode interface

Class 1 transducers take advantage of this de facto analogue standard by adding the TEDS with a switch that is controlled by the direction of the current source, as diagrammed in Figure 20. By reversing the direction of the current, the instrumentation system switches the sensor into digital TEDS mode.

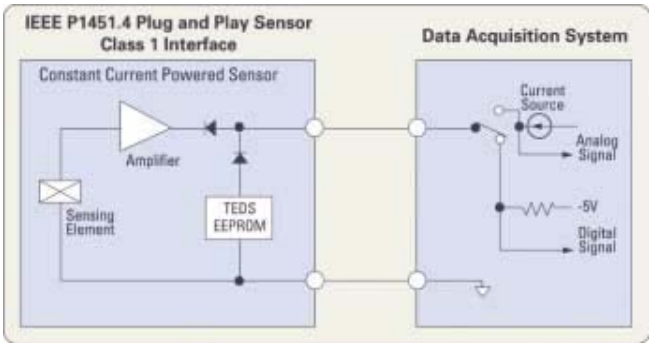


Figure 23:.Mixed-mode interface, Class 1 -shared wires for IEPE sensors.

Most sensor types will implement a form of the Class 2 interface, which separates the digital TEDS interface from the analogue sensor output. The analogue input/output of the transducer is left unmodified, and the 2-wire TEDS interface is added in parallel to the analogue interface. Using this approach, you can implement plug and play with virtually any type of amplified or unamplified sensor or actuator, including thermocouples, RTDs, thermistors, bridge sensors, electrolytic chemical cells, and 4-20 mA current loop sensors. In fact, with the 'add-on' approach of Class 2, it will be very easy to retrofit existing sensors, with a variety of packaging options. Figure 24 illustrates an example implementation of a Class 2 mixed mode interface with a bridge transducer.
The proposed IEEE 1451.4 interface is being defined with consideration for simplicity and low cost and will be compatible with the IEEE 1451.1 network interface standard. One of the possible implementations for a 1451.4 compatible NCAP, which includes an IEEE 1451.1 object model with an IEEE 1451.4 T-block is shown in Figure 4.

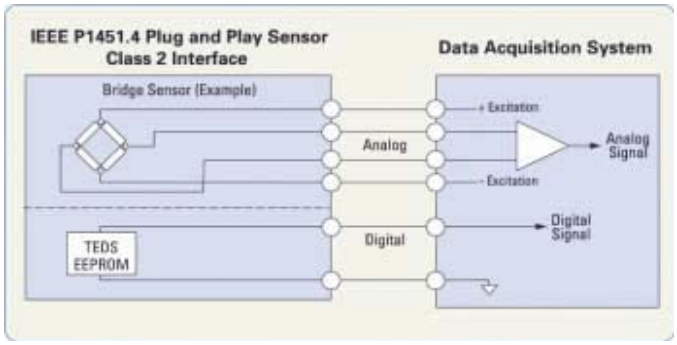


Figure 24:.Mixed-mode interface, Class 2 -independent digital TEDS connection

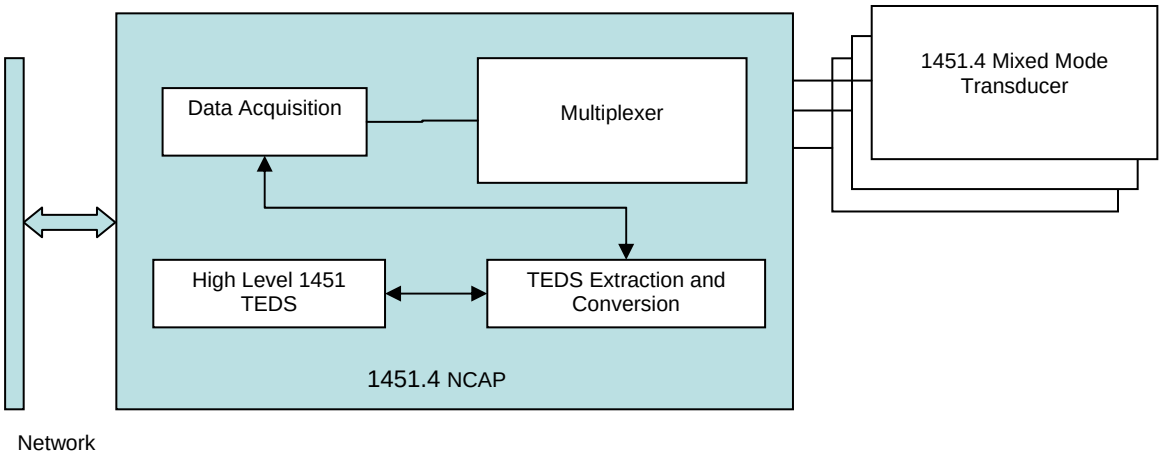


Figure 25:.1451.4 Compatible NCAP

Advantages and Disadvantages of IEEE 1451.4

The main advantage of IEEE 1451.4 in relation to the previous IEEE 1451.2 is that if offers analogue transmission, which makes it compatible with traditional sensors and systems. Furthermore, the number of wires needed for the communication decreases (from 10 to 2-4) and it offers the possibility of having a multidrop configuration. The definition of TEDS templates also makes possible to cover a broad range of sensor types and applications while also keeping memory usage to a minimum.
The main disadvantage is that its implementation is more complicated that the one from IEEE 1451.2 and of course that it is not approved yet. IEEE 1451.4 needs to implement a signal conditioning module that converts the analog output from the sensor into a digital signal that can be sent to the network. If the sensors have already a digital output then 1451.2 is the best option.

4.8.10 Wireless sensor networks

Despite the interest of this standardization effort, currently there is a strong interest in wireless sensor networks. By using electromagnetic waves as their transmission medium, wireless systems avoid the limitations of wired networks and offer competitive advantages in terms of costs, flexibility and ease of use. The costs associated with installing, maintaining, troubleshooting and upgrading wiring have escalated, while costs for wireless technology have continued to drop. Most failures in any network occur at the connectors: wireless sensors eliminate these problems. Wireless solutions provide Improved Flexibility. Without the constraints of wires, plant managers can better track goods and materials and easily reconfigure the network and transducer locations. Unfortunately, IEEE 1451.5 is still under development with basic discussions going on. (see last section)

4.8.10.1 ZigBee and Bluethoot Standards

The most widely used standards today were originally designed for wireless office and local area networks: among them IEEE 802.11, and Bluetooth. The latter one has been developed by a group of companies (www.bluetooth.org), and has been labelled by IEEE as 802.15.1. Both these standards use (but not only) the ISM 2.4 GHz band. A comparison of the properties of different standards is presented below. The standard IEEE 802.15.4 (also known as ZigBee) appears in the first column and it is the most suited choice for sensor networks. This will be described in the next section.

Table 20: Comparison of different IEEE 802 standards.

	802.15.4	802.15.1	802.11b	802.11g	802.11a	802.15.3	802.15.3a
Standard Version/status	IEEE approved	V1.1 (Low-rate)	IEEE approved	IEEE approved	IEEE approved	Draft	Draft IEEE 802.15.3a
Max. data rate	250 kbps 40 kbps 20 kbps	1 Mbps	11 Mbps	54 Mbps	24 Mbps mandatory; 54 Mbps optional	11 Mbps (QPSK) 55 Mbps	110 Mbps (10m) 200 Mbps (4m)
Avg. distance	10-30 m	10 m	100 m	100 m	50 m	10 m	10 m
Frequency allocation	868-868.6 MHz; 902-928 MHz; 2400-2483.5 MHz	2.4 GHz (ISM)	2.4 GHz (ISM)	2.4 GHz (ISM)	5 GHz UNII (5.15-5.35 + 5.725 – 5.825) GHz	2.4 GHz (ISM) 2.4 – 2.4835 GHz	3.1 – 10.6 GHz
Channel bandwidth	0.3 MHz; 0.6 MHz (2 MHz spacing); 2 MHz	1 MHz	25 MHz	25 MHz	20 MHz	15 MHz	Min. 500 MHz Max. 7.5 GHz
Number of RF channels	1; 10; 16	79	3	3	12 US. 8 EU 4 Japan	5	1 – 15
Modulation type	BPSK; OQPSK	GFSK	11 Mbuad QPSK	OFDM (64+CCK (legacy)	COFDM PSK QPSK, 16 QAM	DQPSK 16/32/64 QAM	BPSK, QPSK
Spreading	DS-SS	DS-FH	CCK	OFDM	OFDM		Multiband
Maximum allowed RF power	US: 1W+6dB antenna gain (FCC 15.247); EU: (868 MHz) ERC70-03E: 25mW if duty cycle < 1% in 1 hour.	0 dBm 20 dBm	US: 30 dBm EU: 20 dBm Japan: 10 dBm	US: 30 dBm EU: 20 dBm Japan: 10 dBm	50 mW; 250 mW; 1-watt (depending on the used channels within the band)	US: 50 mV/m EU: 100 mW² EIRP (ETS 300-328) Japan: 10 mW (ARIB STD-T66)	-41.3dBm/MHz (Max average EIRP over entire band = 0.562 mW) (FCC First Report and Order: Part 15 ET Docket 98-153)
Required receiver sensitivity	-85 dBm PER < 1%	-70 dBm BER < 10^{-3}	–76 dBm	From –76 dBm to –74 dBm	From –82 dBm (6 Mbps) to – 65 dBm (54 Mbps)	From –82 dBm (DQPSK) to – 68 dBm (64 QAM)	
Approx. power consumption	< BT	BT (–40 – 100mW)	–4BT	–4BT	–6BT		(–2-3BT)
Approximate cost	~ 0.5 BT	BT (~ $5)	~ 4BT	–4BT	–5BT		~1 to 2 BT

4.8.10.2 ZigBee for wireless sensor networks

There are a multitude of standards that address mid to high data rates for voice, PC LANs, video, etc. However, up to now there has not been a wireless network standard that meets the unique needs of sensors and control devices. Sensors and controls do not need high bandwidth but they do need low latency and very low energy consumption for long battery lives and for large device arrays.

There are a multitude of proprietary wireless systems manufactured today to solve a multitude of problems that also do not require high data rates but do require low cost and very low current drain. These proprietary systems were designed because there were no standards that met their requirements. These legacy systems are creating significant interoperability problems with each other and with newer technologies

ZigBee main features

The ZigBee physical layer was designed to accommodate the need for a low cost yet allowing for high levels of integration. The use of direct sequence allows the analogue circuitry to be very simple and very tolerant towards inexpensive implementations. The media access control (MAC) layer was designed to allow multiple topologies without complexity.
The power management operation does not require multiple modes of operation. The MAC allows a reduced functionality device (RFD) that need not have flash nor large amounts of ROM or RAM. The MAC was designed to handle large numbers of devices without requiring them to be "parked". The network layer has been designed to allow the network to spatially grow without requiring high power transmitters. The network layer also can handle large amounts of nodes with relatively low latencies.

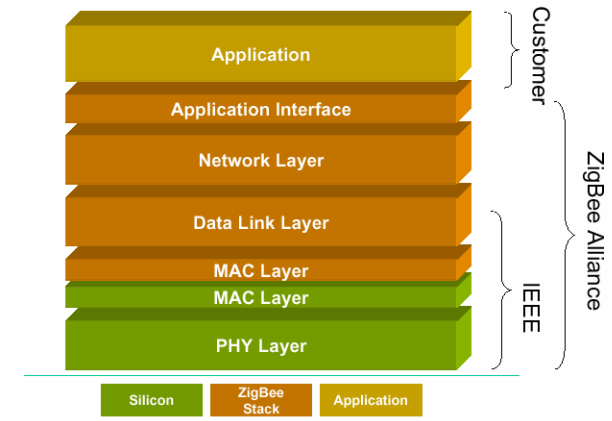


Fig.26: ZigBee and the seven layers ISO Model

ZigBee is poised to become the global control/sensor network standard. Wireless sensors are expected to be the main driver of this technology. Among the various applications we may find.

- SCADA (supervisory control an data acquisition): manufacturing, shipping, warehouse,
- Precision agriculture: irrigation, wineries
- Healthcare: home health monitoring, in-hospital monitoring
- In-building: HVAC, lighting, security
- Homeland Security: shipping, transportation

It has been designed to provide the following features:

- **Low power consumption:** Users expect batteries to last many months to years! Consider that a typical single family house has about 6 smoke/CO detectors. If the batteries for each one only lasted six months, the home owner would be replacing batteries every month! .Bluetooth has many different modes and states depending upon your latency and power requirements such as sniff, park, hold, active, etc.; ZigBee/IEEE 802.15.4 has active (transmit/receive) or sleep. Application software needs to focus on the application, not on which power mode is optimum for each aspect of operation. Even mains powered equipment needs to be conscious of energy. Consider a future home with 100 wireless control/sensor devices,
  - Case 1: 802.11 Rx power is 667 mW (always on)@ 100 devices/home & 50,000 homes/city = 3.33 megawatts
  - Case 2: 802.15.4 Rx power is 30 mW (always on)@ 100 devices/home & 50,000 homes/city = 150 kilowatts
  - Case 3: 802.15.4 power cycled at .1% (typical duty cycle) = 150 watts.
- **Low cost** (device, installation, maintenance): Low cost to the users means low device cost, low installation cost and low maintenance. ZigBee devices allow batteries to last up to years using primary cells (low cost) without any chargers (low cost and easy installation). ZigBee's simplicity allows for inherent configuration and redundancy of network devices provides low maintenance.
- **High density of nodes per network:** ZigBee's use of the IEEE 802.15.4 PHY and MAC allows networks to handle any number of devices. This attribute is critical for massive sensor arrays and control networks.
- **Simple protocol, global implementation:** ZigBee's protocol code stack is estimated to be about $1/4^{th}$ of Bluetooth's or 802.11's. Simplicity is essential to cost, interoperability, and maintenance. The IEEE 802.15.4 PHY adopted by ZigBee has been designed for the 868 MHz band in Europe, the 915 MHz band in North America, Australia, etc; and the 2.4 GHz band is now recognized to be a global band accepted in almost all countries.

Battery drain and effective data rate

The Packet length can affect battery drain. Typically the shorter the packet the quicker the device can go to sleep. Bluetooth is a slotted protocol. Communication can occur either in 625 µS, 1875 µS or 3125 µS slots. Fig.4 shows that the effective data rate based on the transmissions speeds stated in Bluetooth v1.1 and IEEE 802.15.4 draft 18, using the 250 kb/s rate. The general trend is that at larger packet sizes the effective data rate approaches the raw data rate. The peaks for the Bluetooth rate are a result of the three slot sizes, when a packet becomes too big for one slot it must increment to the next slot even though it does not fill the whole slot allocation.

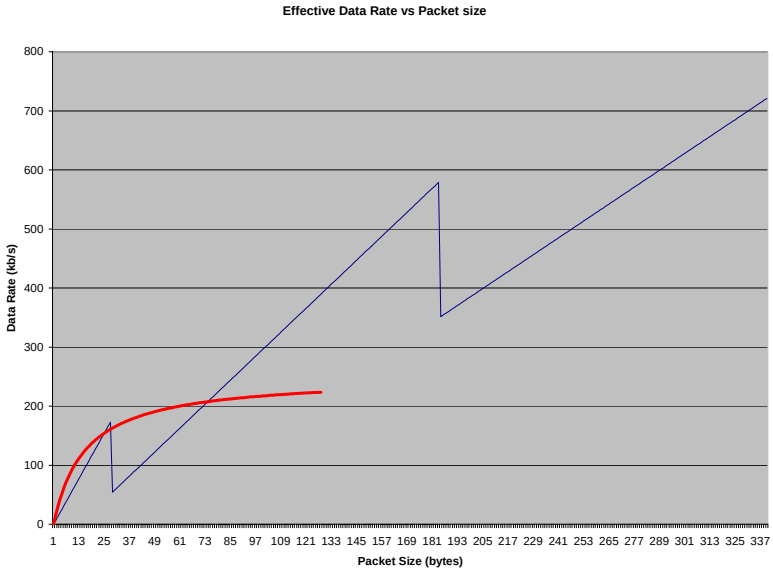


Fig.27: Effective data rate vs. packet size based on theoretical values with no retransmissions (red: ZigBee, blue: Bluetooth) (Is it possible to use two different symbols rather than colour as in black and white the colours have no meaning)

IEEE 802.15.4 is designed for small packets so it is no surprising that it is more efficient at those small packets resulting in a higher effective rate despite its lower raw data rate.
From this graph we can see that for packets less than 75 bytes ZigBee has a higher effective data rate than Bluetooth. Having a lower rate for small packets means that Bluetooth needs longer transmit and receive times and therefore current drain is higher for small data packets.
Although these numbers do not represent retransmissions or multiple devices requesting the bandwidth; the author believes that the same traits will be exhibited in other cases.

Power consumption:

One of the most important factors for a sensor based information network is the battery operation of the sensor system. According to the mode of operation of

the sensor system the ZigBee offers several advantages in terms of power consumption. The Bluetooth standard is primarily designed to maximize ad-hoc functionalities whereas ZigBee is designed to optimize slave power requirements. The IEEE 802.15.4 (ZigBee) standard uses a CSMA-CA protocol which means that the end nodes only when they have data to send. The benefits are:

- No waiting for polling (however they must wait for a clear channel which shouldn't be a problem in low duty cycle networks such as with sensor and control devices)
- Current drain is substantially reduced over a polling protocol that must poll to maintain latencies even though the majority of the time the device needed be polled

Table 21: Timing and interface features of ZigBee and Bluetooth

Timing issues			Interface issues		
	ZigBee	*Bluetooth*		*ZigBee*	*Bluetooth*
New slave enumeration	30ms typically	3s, typically 20s	*Transmission mode*	DSSS (Direct Sequence Spread Spectrum)	FHSS (Frequence Hopping Spread Spectrum)
Sleeping slave changing to active	15ms typically	3s typically	**Modulation**	11 chips/symbol	1600 hops/s
Active slave channel access time	15ms typically	2ms typically	*Transmission rate*	62.5 K symbols/s	1 M Symbol/s
			Symbol size	4 bits/ symbol	1 bit/symbol
			Peak Information Rate	128 Kbit/s	108-723 kbit/s

ZigBee devices can be quickly attached to the network for the exchange information. Once detached they can be put in a 'deep sleep' mode of operation in order to achieve a very long battery life (about 2 years or more for 'normal' batteries). Bluetooth devices require about 100 times more energy for this operation. The power model is based on a regular charging like mobile phones.

4.8.10.3 Smart Wireless Sensors Architecture

A possibility to fuse the concept of interoperability given by the IEEE1451 family and the advantages of wireless sensing is illustrated in figure 6.

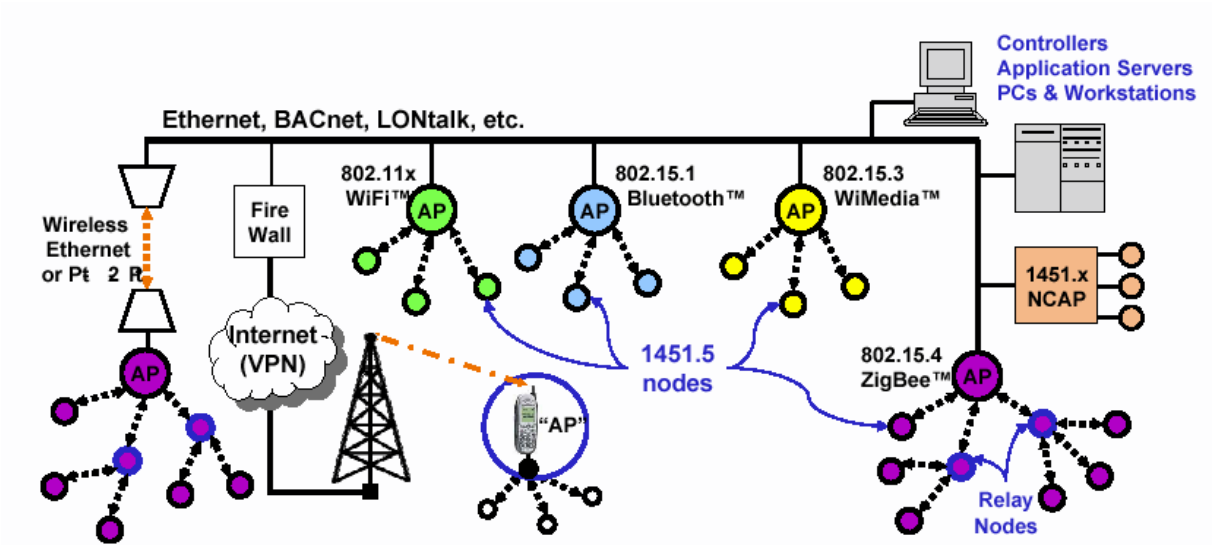


Figure 28: Different option for wireless sensor networking.

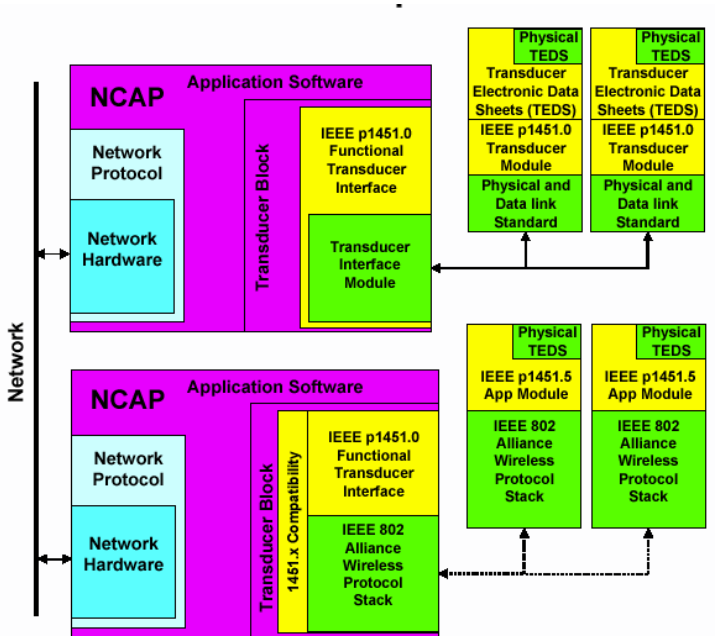


Figure 29: A wired (top) and wireless (bottom) STIMs connect to their respective NCAPs and to the network.

The figure shows how many different sensors would connect to a single network based on Ethernet, BACnet, LONtalk, using different wireless standard protocols. The basic approach is not to reinvent the wheel, but to use already developed standards, while keeping other concepts from 1451 family to assure the interoperability between different smart sensors (such as for instance the TEDS) This is a proposal by Motorola to the 1451.5 working group. As far as we know, no ultimate decision on this concept has been done.
There would be wireless STIMs (w-STIM) and wireless NCAPs (w-NCAP) sharing the same protocol. For the rest of the network a 1451.5 w-NCAP will appear as

its wired counterpart. A more detailed view of this possibility is shown in Figure 29.

4.8.11 Embedded Operating Systems: TinyOS portability issues

The GoodFood AmI working group has recognised the importance of deeply embedded operating systems for wireless sensor networks. Among the different options available, TinyOS is the stronger player due to its very low footprint and because it is open source. TinyOS is a project created by the Computer Science Department at University of California, Berkeley. TinyOS is primarily designed for programming sensor nodes in sensor networks. The concept of TinyOS is to provide the basic surroundings for an application and thus allowing the application to be executed on an embedded platform. The application must be written in the NEST programming platform, which is a large set of C macros. A set of macros have been designed to use the hardware on the boards, e.g. timers, temperature-sensitive resistors, light-sensitive resistors, LEDs, speakers and microphones. Sensor networks are an essential part of the targeted embedded platforms for TinyOS. TinyOS was originally developed for the AVR platform, but it has been also implemented for the MIPS and it is also running on Motorola HCS08 and Texas Instruments MSP430, in the Telos mote platform.
Telos is the next-generation mote platform for extremely low power, high data-rate, sensor network applications designed with the dual goal of fault tolerance and development ease.
Designed at the University of California, Berkeley, by TinyOS developers, the Telos platform offers seamless vertical integration between the hardware and the TinyOS operating system. Moteiv provides the service of producing and distributing the open-source design in both low and high volumes for wireless sensor network applications. The Telos mote boasts the first 802.15.4 (ZigBee) radio and an integrated on-board antenna providing up to 50 meter range.
Telos is part of a line of low cost motes with various integrated sensors including temperature, relative humidity, magnetic field, and occupancy
detection.

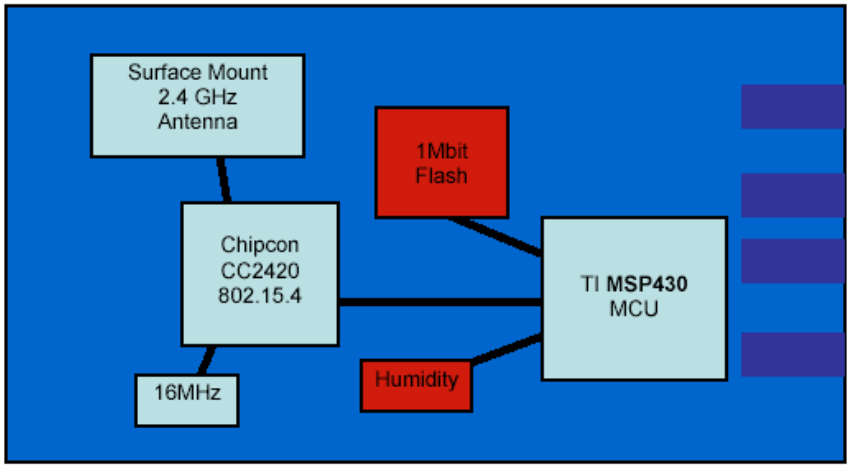


Figure 30: Block Diagram of the Telos mote.

The implementation for a new platform would mean to reprogram the assembler code and the code that handles the specific drivers used. If TinyOs had a hardware abstraction layer (HAL) this reprogramming would be much easier since the operating system and the hardware layer would become independent.
A HAL is a layer of programming in an operating system that functions as an interface between a system's hardware and software, providing a consistent hardware platform on which to run applications. When a HAL is employed, applications do not access hardware directly but access the abstract layer provided by the HAL. Like APIs, HALs allow applications to be device-independent because they abstract information from such systems as caches, I/O buses and interrupts and use this data to give the software a way to interact with the specific requirements of the hardware on which it is running. HALs are usually programmed in a high level language like C.

4.8.11.1 Implementing ZigBee with TinyOS

The goal of the IEEE 802.15.4 standard for low rate wireless personal area networks (LR-WPANs) was to define a standard to provide ultra-low complexity, low-cost, and extremely low power wireless connectivity among inexpensive fixed, portable and moving devices. It is designed to address applications with relaxed throughput and latency requirements.

This standard just defines the Physical Layer and Medium Access Control (MAC) sublayer. The Logic Link Control (LLC) Layer is defined by the IEEE 802.2, as is represented in Figure 28. The upper layers are beyond the scope of the standard, even though some exemplary networks are described.

The only recommendation that exists regarding the upper layers of IEEE 802.15.4 is the one by the ZigBee Alliance. The ZigBee Alliance is an association

of companies working together to define all the ISO/OSI layers on the basis of IEEE 802.15.4.

TinyOS is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools

In principle, there should be no problem to implement ZigBee on the top of TinyOS, it only will define a new network protocol that could be used by the services. A deeper study need to be done on ZigBee to see the difficulties of implementing their protocols on TinyOS, though it would be necessary to access the proprietary information.

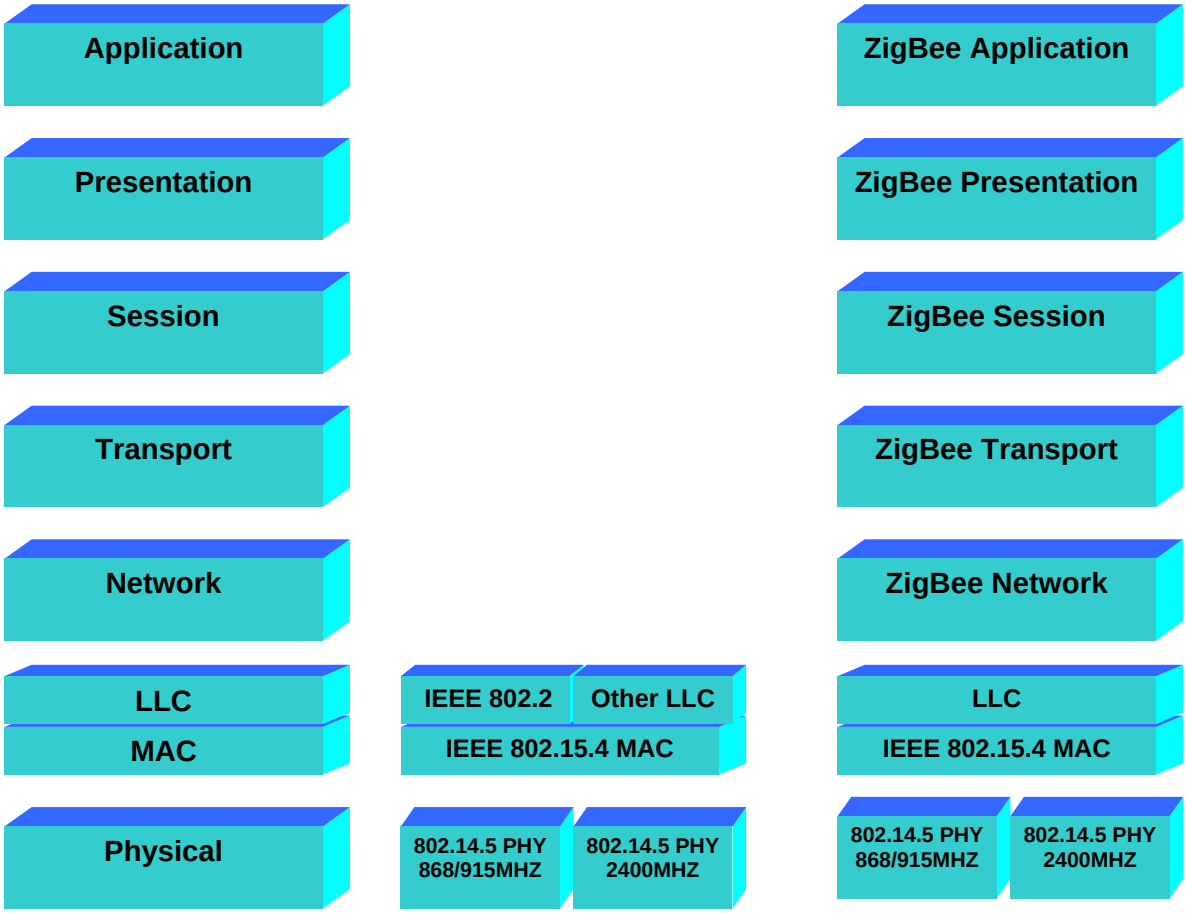


Figure 31: IEEE Layers (left), IEEE 802.15.4 Layers (middle) and ZigBee Layers (right)

4.8.11.2 Conclusion

In order to preserve the openness of the architecture and its scalability is important to adhere to existing standards for sensor networking and for low-data rate , low-power consumption wireless networks. Among the many different options, the family of standards IEEE 1451.X has been identified as a major body of interest for smart sensor networking. A number of hardware/software objects like TEDS, STIM and NCAP, have been identified as key concepts for truly "plug & play" capabilities. IEEE 1451.2 defines a wired interface connection consisting in 10 wires. Other options with less wiring cost are about to appear in IEEE 1451.4 or fully wireless in IEEE 1451.5. We advocate being aware of these developments specially for final solution adopted in the project.

Concerning the wireless transmission of information, we have compared IEEE 802.15.4 (ZigBee) to IEEE 802.15.1 (Bluetooth). The results show that the former is much more adequate for very low power sensor nodes.

From the embedded software point of view, the use of TinyOS is a very interesting alternative because it is open source. Moreover in recent times, ports of this operating systems to a variety of microcontrollers are appearing, avoiding the danger of being restricted to a particular vendor.

5 The Satellite Components

5.1 The Wireless Sensors networks

5.1.1 The Tiny-OS operative system

Applications that need to meet these requirements are not just complex to design, but they are also like to be tailored on the specific hardware used, resulting in solutions that are not very flexible and scalable. Researchers at the University of California, Berkeley, have been extensively investigating sensor networks and embedded systems. One of the major projects, developed with the support of DARPA and Intel, has lead to the release of TinyOS, an open source operating system for MICA motes (v. par. 2.3)

TinyOS has been distributed under an open source license since Oct 2001 (ver. 0.5.1). The last stable version is the 1.1, which was released in Oct 2003 and has several major improvements over previous versions.

Installation packages for TinyOS are available for Linux and Windows (under cygwin environment or similar). The distribution package includes operating

system core components, libraries, a compiler and a debugger to develop application using the nesC language (the same that has been used to code TinyOS). Complete documentation and reference manuals are included in the package and a tutorial is provided to help novice users to get started with TinyOS and MICA mote programming.

Along with the development tools, the distribution package includes TOSSIM, a discrete event simulator for TinyOS sensor networks. This tool uses a PC framework to simulate the code written to run on motes before real deployment. This allows users to debug, test, and analyze algorithms in a controlled and repeatable environment, reducing the overall time required for the development of large sensor network applications.

TinyOS, differently from other OS for embedded systems, was expressly designed to meet the stringent constraints imposed by the use of microcontrollers with limited memory and processing capabilities. The main advantage in the use of an operative system is that it provides an abstraction layer between applications and the underlying hardware. This implies that applications development cycles can be speed up by using a rich set of functionalities that are pre-built and are optimized to make an efficient use of the hardware. More in detail, TinyOS is based on the concept of 'components' that are very similar to the concept of classes in the Object Oriented Programming Paradigm.

TinyOS' objects can be classified in three main categories:

- Hardware abstraction components. This is a set of software modules that encapsulate the functionalities of the hardware components present on the node. Examples of this are the components that allow to control the radio chip or to interact with the UART module.
- Synthetic hardware components. These components use a certain number of hardware abstraction components to emulate the presence of more sophisticate hardware that is not present on the node. An example is the

RadioByte, a component used to access the radio link without having to deal with the low level detail of the communication protocol (i.e. the Data Link layer: bit encoding, medium access control, etc).

- High level software components. Under this category fall all the other software components, likes the ones used for packet routing, control and application specific task.

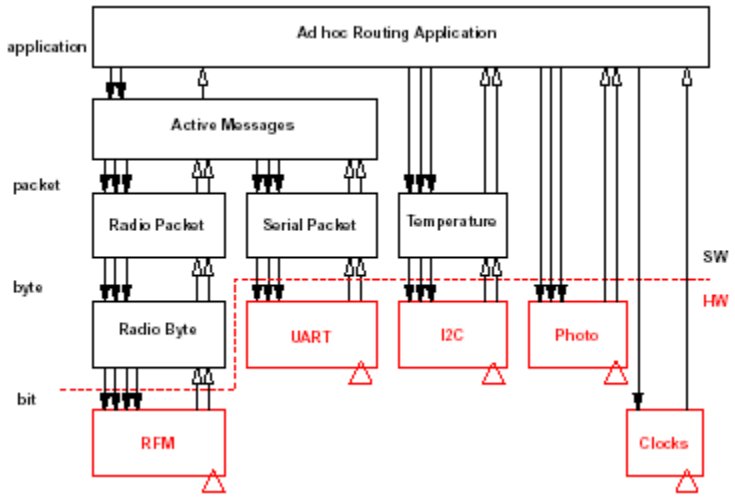


Figure 321: A routing application built as composition of TinyOS components

By using the TinyOS component model, it's possible to access a rich set of ready-to-use modules or build new modules that enclose and extends the functionality of lower level modules. This programming paradigm allows to produce applications that are efficient, flexible and scalable. Applications can benefit from enhancements of the node's hardware without need to be redesigned, in fact only the components that directly deal with the new hardware need to be changed. All the other higher-level components can be left unchanged.

Another major feature of TinyOS is the system support for concurrent tasks, a feature that allows the microcontroller (which doesn't include a parallel computing architecture) to serve multiple data flows at the same time. Thanks to this concurrency support, it's possible to run long processing tasks in the background and let the microcontroller to quickly respond to hardware interrupts when requested.

TinyOS also comes with a set of subsystems that can be used to include advanced functionalities in user applications. The major modules are:

- Multi-hop Routing: The TinyOS-1.1 release and later include library components that provide ad-hoc multi-hop routing for sensor network applications. The implementation uses a shortest-path-first algorithm with a single destination node (the root) and active two-way link estimation. The data movement and route decision engines are split into separate

- components with a single interface between them to permit other route-decision schemes to be easily integrated in the future. Use the multi-hop

router is essentially transparent to applications (provided they correctly use the interface).

- TinySec: a link layer encryption mechanism which is meant to be the first part in a suite of security solutions for tiny devices. The core of TinySec is an efficient block cipher and keying mechanism that is tightly coupled with the Berkeley TinyOS radio stack. The main aims of TinySec are:

- Access Control: only authorized nodes should be able to participate in the network.
- Integrity: a message should only be accepted if it was not altered in transit. This prevents, for example, man-in-the-middle attacks where an adversary overhears, alters, and re-broadcasts messages.
- Confidentiality: unauthorized parties should not be able to infer the content of messages.

- Bombilla: a bytecode interpreter that runs on TinyOS. The conciseness of a high-level instruction set allows users to write programs and dynamically install them in a network much more quickly than possible with binary images. Additionally, Bombilla can automatically forward programs through a network; a new forwarding Bombilla program installed on a single mote will eventually run on every mote in the network.

- MatchBox: a system that is designed to provide a simple filing system for mote-based applications. The system has been designed to be reliable and to make an efficient use of hardware resources (especially RAM) available on MICA motes.

- TinyDB: a query processing system for extracting information from a network of TinyOS sensors. Unlike existing solutions for data processing in TinyOS, TinyDB does not require you to write embedded C code for sensors. Instead, TinyDB provides a simple, SQL-like interface to specify the data to extract, along with additional parameters, like the rate at which data should be refreshed. Given a query specifying data of interests, TinyDB collects that data from motes in the environment, filters it, aggregates it together, and routes it out to a PC. TinyDB does this via power-efficient in-network processing algorithms.

- Xnp: a set of software modules that allows for network reprogramming of MICA motes. By using Xnp it's possible to write a new application, compile it and send the executable image to a remote mote using the wireless network infrastructure. Once the new code is received by the mote, it is used to replace the old firmware so that the node can execute the new application.

5.1.2 Node Architecture

For the architecture of the sensor node, the following points have to be taken into considerations:

1. Environment conditions in the wine yard are heavy. A good packaging has to be foreseen.
2. Power consumption of the sensor node is affected by the following facts:
   - Power needs of the connected sensors (which may be the biggest part)
   - Send measurements to Gateway
   - Serve as router for other sensor nodes
   - Serve as router for wireless devices

   A lot of power saving may be done by a synchronized, time controlled measurement and communication strategy.
3. The electrical and logical interface for the various sensors may be too expensive.
   By defining some types of sensor nodes and using integrated sensors (e.g. one type of sensor node just measures air temperature & humidity and soil temperature & humidity in two depth), cost savings may are possible.
4. Restricted computing resources (as memory and CPU power) restrict the data processing capability of the sensor nodes. Especially when adding security to data transition, CPO performance may run short.

The next figure shows the general configuaration of the WSN Node for wine yard.
The WSN node will be implemented in a stackable configuration including the Communication Board (Mote) and up to 16 sensors connected to the mote through a Sensor to Node Interface.**Errore.**

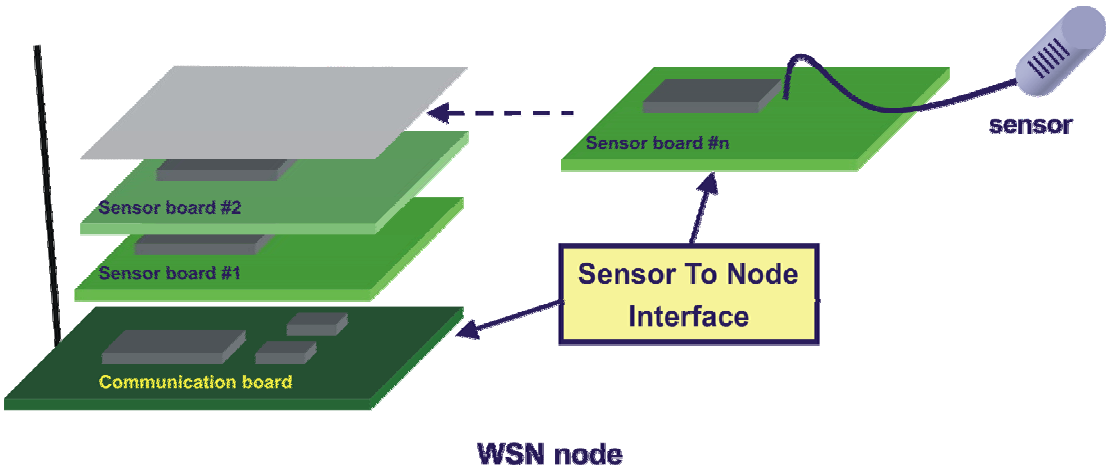


Figure 33 – Wireless Sensor Network Implementation

Each of the Sensor to Node Interface Board will make use of a Programmable System on a Chip (PSoC) device.

PSoC includes a certain number of analog programmable circuits and a microcontroller.
This architecture will enable a Plug & Play approach for the sensor connection between the Mote and the sensors.
Sensor to Node interface will be described in the following paragraph.
By using a solar panel with a 12V power supply, enough power will be available for all the tasks of a sensor node.

5.1.3 Sensors to network node interfaces

In the AmI vision an "AmI Compliant Sensor" is a sensor that can be easily connected to whichever node of the network and that is able to work without changing any configuration parameters in the network (*plug and play* sensors).
In order to achieve this goal, a well defined interface has to be developed.
Because of the complexity of a well defined plug and play interface (in terms of protocol and descriptors) it is not realistic to suppose that all the partners involved in the sensors development activities, have the competence required to implement this interface. In particular, most of the developed sensors could have a simple analogical output. Moreover, the commercial available sensors are obviously NOT AmI Interface compliant.
This work proposal is based on an **intermediate hardware interface layer** that allows to make a generic sensor "AmI Compliant" and with "plug and play" features.
Specific requirements, as flexibility and adaptability of this interface, can be achieved by using a device with programmable analogical and digital blocks and with on chip micro-controller (e.g. Programmable System on Chip (PSoC) produced by Cypress, http://www.cypress.com). Using this type of device it is possible to connect both analogical and digital output sensors with the nodes, implementing also descriptors and intelligent power management.

5.1.3.1 A brief description of programmable interfaces

The programmable system interfaces are designed to replace multiple traditional MCU-based system components with one, low cost single-chip programmable component. This device includes configurable blocks of analog and digital logic, as well as programmable interconnect (all reconfigurable in run time). This architecture allows the user to create customized peripheral configurations, to match the requirements of each individual application.

DEMANDED FEATURES:

- High speed microprocessor
- Low power
- 3.0 to 5.25 V Operating voltage
- Analogical to digital converter (at least 12 bit)
- Digital to analogical converter (at least 6 bit)

- Programmable Gain Amplifiers
- Programmable Filters and Comparators
- Timers, Counters and PWMs
- Watchdog and Sleep Timers
- On-Chip Precision Voltage Reference
- Programmable Clocking
- Flash Program Storage
- SRAM Data Storage
- EEPROM
- Configurable Interrupt

5.1.3.2 "AmI Compliant" basic interface design

The main important feature required for achieving the foreseen goal is the possibility to change the hardware configuration of analogic circuitry, modifying some software parameters. This allows to easily interface analog (conditioned or not) sensor as well as digital sensor to the network node by configuration of some parameters.
The configuration of the programmable interface will be the fundamental step to make "AmI Compliant" a generic sensor. In fact, all the information useful to describe correctly the sensor must be inserted in the interface's memory. These information are used for the right configuration of the analogic/digital modules of the interface system and also to provide the support for "Plug and Play" features.

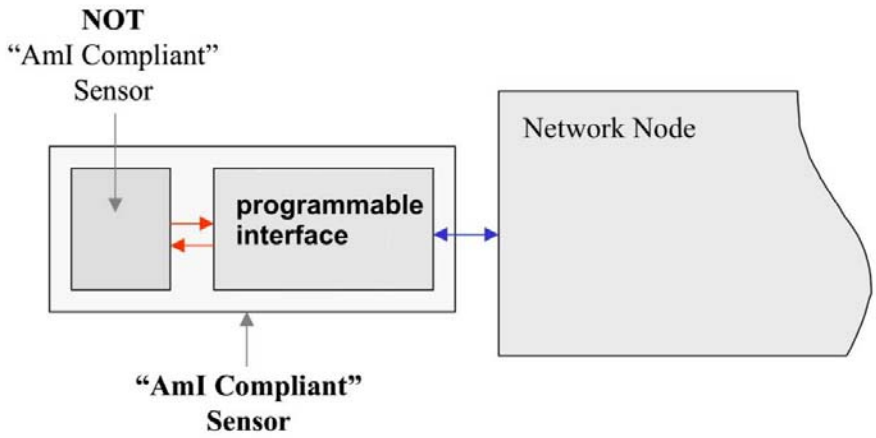


Figure 34. Sensor and "AmI Compliant" Sensor

The information must be structured in a well defined way as "Descriptors"; the interface will transmit this "Descriptor" to the network node to update automatically the AmI system.

5.1.3.3 Architecture overview

The generic sensor NOT "AmI Compliant" will be connected to the node trough the interface.
Concerning the power, the node will supply directly the interface but not the sensor. The sensor will be supplied by interface, using internal buffered DAC.
The timing for the turning on of the sensor will be managed directly by interface (on the basis of the configuration parameters).
In addition, the acquisition of the sensor output will be done directly by interface; the results of the conversion will be sent to the node following the protocols that will be defined.
The hardware internal architecture of programmable device will be selected on the basis of the type of sensor to interface; the selection will be self-done after the right loading of the sensor parameters configuration. Three type of sensor could be necessary:

- Conditioned Sensor
- Unconditioned Sensor
- Digital Sensor (serial interface:rs232 - SPI - Microwire)

For each type of sensor a specific architecture will be required.

Conditioned Sensor

In the case of a conditioned sensor the output is usually 0-5V (or similar), relative low impedance does not require any amplification. The output of the sensor is referred to the ground (no differential output).

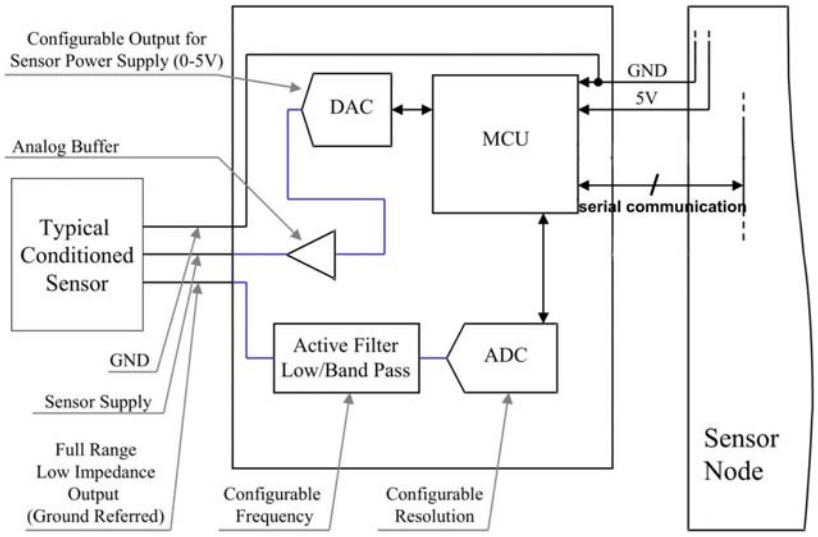


Figure 35: Conditioned sensor configuration.

Therefore the acquisition module should be composed by an analog filter (if necessary), with configurable characteristic frequency (by configuration parameter), and by an Analog to Digital Converter (with selectable resolution).

Unconditioned Sensor

In the case of a unconditioned sensor the output is usually very low (in the range of mV), the output impedance is relatively high and an amplification stage must be provided. The output of the sensor can be also differential.

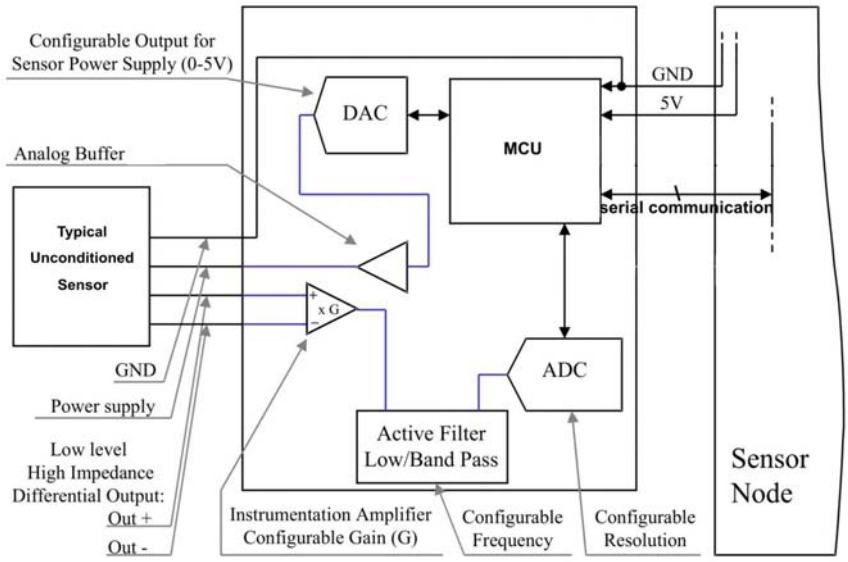


Figure 36. Unconditioned sensor configuration.

Therefore the first stage of acquisition module will be a differential amplification stage: the gain will
be automatically selected on the basis of the configuration parameters loaded in the programmable interface. After the first stage of amplification, a filter stage is present in order to reduce the present noise: the filter will have configurable characteristic frequency (by configuration parameter). The last module is the Analog to Digital Converter (with selectable resolution).

Digital Sensor

In the case of digital sensor the digital configurable blocks of the programmable interface could be used to implement hardware peripheral control. The digital protocol implemented could be the following:

- SPI
- Microwire

- RS232

The support of digital sensor could be useful in particular to support commercial available digital sensors. In addition to these three protocols, other serial protocols could be supported.

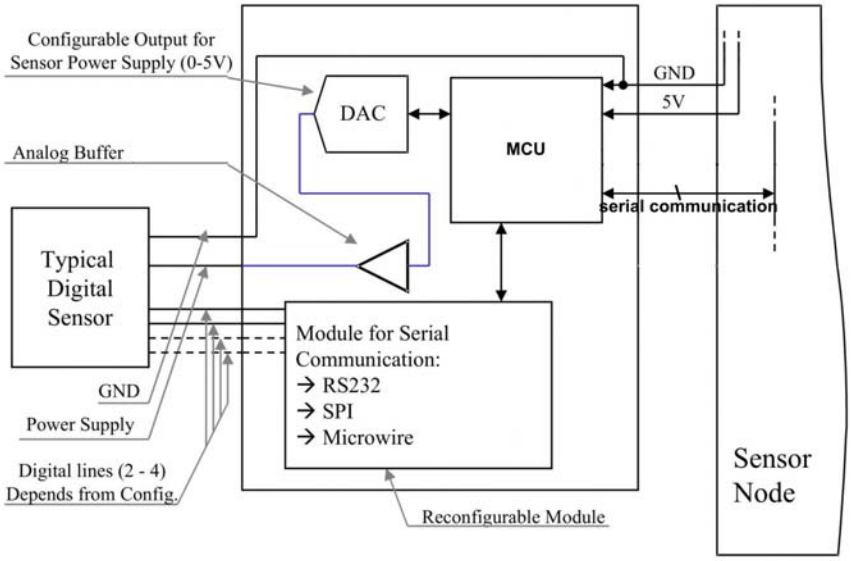


Figure 37. Digital sensor configuration.

5.1.4 Configuration of "AmI Compliant" interface

The design and realization of the programmable interface will be carried out by WP7 partners.
The configuration of the programmable interface must be made easily by sensor supplier (all WPs) in order to make "AmI Compliant" the developed sensors (or other commercial sensors).

Each type of sensor must have a dedicated configuration. A programmable interface, configured for a type of sensor could be applied to all sensor of the same type. The interface configuration must be done easily and quickly before to connect it to the sensor. In this way a reliable tool for the configuration step will be provided. The tool will be a software for the data entry (configuration parameters) and generation of "descriptors"; these descriptors will be loaded into the programmable interface using some programmer devices (developed *ad hoc*).
For the sensor to node interface it is desirable to have a module fitting the 25mm form factor of the NMRC nodes that provides support for a wide variety of sensors without the need to redesign the PCB for each new sensor. In order to support a wide range of sensors the interface must contain a degree of analogue programmability for the necessary signal conditioning and analogue to digital conversion requirements for a particular sensor. For these reasons it is proposed

to fabricate an interface module based around the PSoC™ devices from Cypress Microsystems (http://www.cypressmicro.com/).

5.1.4.1 PSoC Features

The PSoC™ family consists of many Mixed Signal Array with On-Chip Controller devices and the aim is to design a module to interface between a selection of sensors and the Atmel ATMega128L microcontroller-RF unit. The features of the PsoC devices that are desirable in the context of GoodFood are:

- configurable blocks of analog and digital logic;
- a fast CPU to manage control and manage digital data interfacing to the microcontroller-RF module;
- a range of digital interfacing options, UART, SPI, I2C;
- Flash program memory for retention after power down;
- SRAM data memory to buffer incoming sensor data;
- Configurable IO to enable a generic solution;
- highly stable, instrumentation-quality analogue performance for reliable sensor data;
- 14-bit analog-to-digital converters (up to four independent converters are available in a single PSoC device);
- low noise, input leakage and voltage offset;
- In system serial programming so that devices can be reprogrammed multiple times after soldering in the module;
- 3.0 to 5.25V operation voltages are compatible with device voltages supplied to other modules.

5.1.4.2 Device Selection

A range of 33 PSoC devices are available in the CY8C27x device family. In selection of a device the form factor of the 25mm platform must be taken into account as shown in Table 22. The 48 pin MLF package gives the best pin density per volume area or volume, however, the 100 pin TQFP gives the maximum number of IO and therefore the most powerful solution leading to initial selection of this device. At a later stage this device may perhaps be found to contain unnecessary resources in which case a smaller less powerful device will be selected in order to prolong battery life.
Initial floorplanning has shown that the top of range CY8C29866-24AXI in the 100 pin TQFP package as shown in Figure 38.

Table 22: PSoC package options

Package	Maximum Package Dimensions (mm)		
	Length	Width	Height above PCB
28 PDIP	36.19	7.62	4.82
28 SSOP	10.40	8.10	2.00
44 TQFP	12.00	12.00	1.80
48 SSOP	16.00	10.67	2.79
48 MLF	7.10	7.10	1.00
100 TQFP	16.00	16.00	1.20

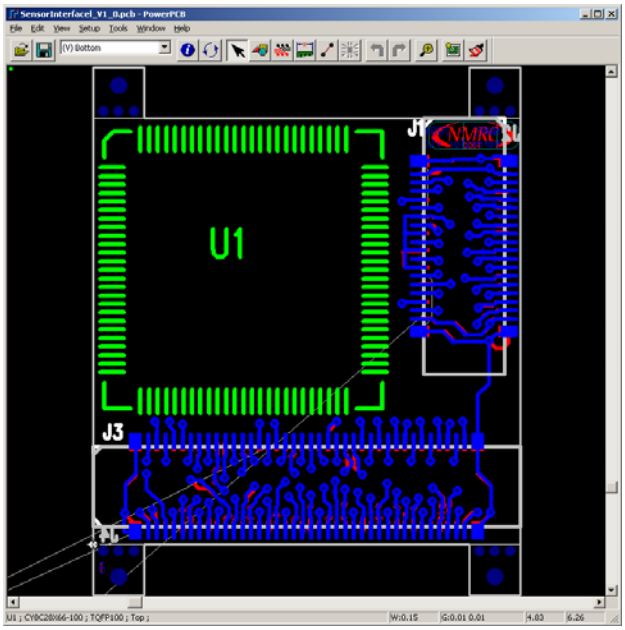


Figure 38: Initial Floorplanning of the PSoC in the 25mm platform

5.1.4.3 CY8C29866-24AXI Specifications

This section gives an overview of the specifications of the CY8C29866-24AXI, for full datasheet consult http://www.cypress.com .

Features of Harvard Architecture Processor

- M8C Processor Speeds to 24 MHz
- 8x8 Multiply, 32-Bit Accumulate
- Low Power at High Speed
- 3.0 to 5.25 V Operating Voltage

- Operating Voltages Down to 1.0V Using On-Chip Switch Mode Pump (SMP)
- Industrial Temperature Range: -40°C to +85°C

On-Chip Memory

- 32K Bytes Flash Program Storage 50,000
- Erase/Write Cycles
- 2K Bytes SRAM Data Storage
- In-System Serial Programming (ISSP□)
- Partial Flash Updates
- Flexible Protection Modes
- EEPROM Emulation in Flash

Programmable Pin Configurations

- 25 mA Sink on all GPIO
- Pull up, Pull down, High Z, Strong, or Open Drain Drive Modes on all GPIO
- Up to 12 Analog Inputs on GPIO
- Four 40 mA Analog Outputs on GPIO
- Configurable Interrupt on all GPIO

The Digital System

The Digital System is composed of 16 digital PSoC blocks. Each block is an 8-bit resource that can be used alone or combined with other blocks to form 8, 16, 24, and 32-bit peripherals, which are called user module references. Digital peripheral configurations include those listed below.

- PWMs (8- to 32-bit)
- PWMs with Dead band (8- to 16-bit)
- Counters (8- to 32-bit)
- Timers (8- to 32-bit)
- UART 8-bit with selectable parity (up to 4)
- SPI master and slave (up to 4)
- I2C slave and master (1 available as a System Resource)
- Cyclical Redundancy Checker/Generator (8- to 16-bit)
- IrDA (up to 4)
- Pseudo Random Sequence Generators (8- to 16-bit) The digital blocks can be connected to any GPIO through a series of global busses that can route any signal to any pin. The busses also allow for signal multiplexing and for

- performing logic operations. This configurability frees your designs from the constraints of a fixed peripheral controller.

The Analog System

The Analog System is composed of 12 configurable blocks, each comprised of an opamp circuit allowing the creation of complex analog signal flows. Analog peripherals are very flexible and can be customized to support specific application requirements. Some of the more common PSoC analog functions are listed below.

- Analog to digital converters (up to 4, with 6 to 14-bit resolution, selectable as Incremental, Delta Sigma, and SAR)
- Filters (2, 4, and 6 pole band-pass, low-pass, and notch)
- Amplifiers (up to 4, with selectable gain to 48x)
- Instrumentation amplifiers (up to 2, with selectable gain to 93x)
- Comparators (up to 4, with 16 selectable thresholds)
- DACs (up to 4, with 6- to 9-bit resolution)
- Multiplying DACs (up to 4, with 6- to 9-bit resolution)
- High current output drivers (four with 40 mA drive as a Core Resource)
- 1.3V, 1% voltage reference (as a System Resource)
- Modulators, Correlators & Peak detectors

Additional System Resources

System Resources, some of which have been previously listed, provide additional capability useful to complete systems. Additional resources include a multiplier, decimator, switch mode pump, low voltage detection, and power on reset.
Brief statements describing the merits of each system resource are presented below.

- Digital clock dividers provide three customizable clock frequencies for use in applications. The clocks can be routed to both the digital and analog systems. Additional clocks can be generated using digital PSoC blocks as clock dividers.
- A multiply accumulate (MAC) provides a fast 8-bit multiplier with 32-bit accumulate, to assist in both general math as well as digital filters.
- The decimator provides a custom hardware filter for digital signal, processing applications including the creation of Delta Sigma ADCs.
- The I2C module provides 100 and 400 kHz communication over two wires. Slave, master, and multi-master modes are all supported.
- Low Voltage Detection (LVD) interrupts can signal the application of falling voltage levels, while the advanced POR (Power On Reset) circuit eliminates the need for a system supervisor.

- An internal 1.3V voltage reference provides an absolute reference for the analog system, including ADCs and DACs. The reference's 1% resolution over voltage and temperature satisfies most system requirements.
- An integrated switch mode pump (SMP) generates normal operating voltages from a single 1.2V battery cell, providing a low cost boost converter.

The pinout of the 100 pin CY8C29866-24AXI device is shown in Figure 36.

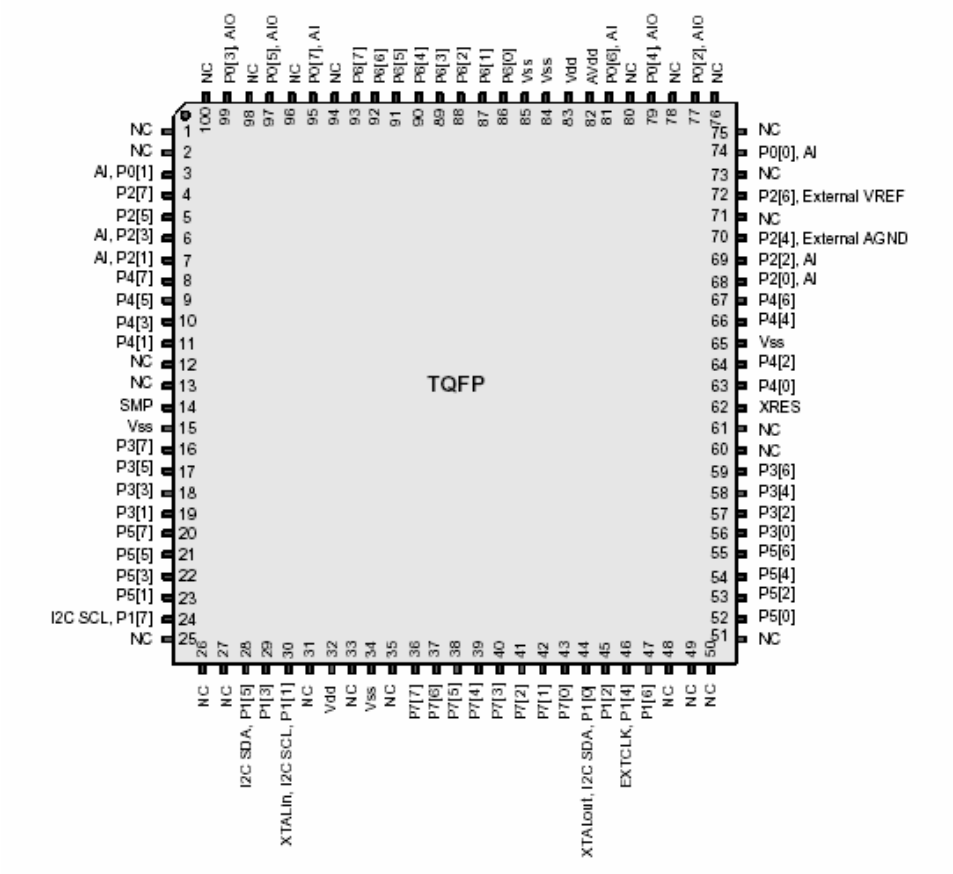


Figure 39: Pinout of the CY8C29866-24AXI

5.1.5 Communication protocol

5.1.5.1 Node to sensor

One of the main issues in the system design is the identification of the optimum communication protocol between the sensors and the mote, which aims is the implementation of plug & play sensors.

As described in the previous each sensor will be connected through an intermediate layer for signal conditioning and subsystem control (PSOC).
Custom solution as well as standard based communication protocol could be implemented. A decision on the approach that will be followed will be carried out when the previously described hardware platform will be available.
The custom approach foresees the adoption of a standard communication interface (as simple as I^2C or SPI) and the development of a specific protocol, able to implement the required plug & play features.
To this aim some description structres for the installed sensors have to be introduced
On the other hand the integration of a fully compliant standard communication protocol (see IEEE 1451.x) could lead to a high level solution.
For example IEEE 1451.x standards provide a complete hardware and communication protocol definition, with a complete set of sensor descriptors.
The bus architecture (stackable connection of the sensor boards to the mote) defined in the sec. 5.1.2 (Node Architecture) doesn't seem to be fully compliant with the application of the IEEE1451.x standards.
One possible solution in order to implement one of the 1451 standard could be its partial customization.
In particular the integration of 1451.2 standard with the I^2C architecture (enabling the bus communication) will be one of the possible solution that will be tested in the developing phase.

System requirement

The choice between the described approach is mainly based on the satisfaction of some basic system requirements.
At present, a fruitful work is carrying out in the identification and definition of these requirements:

- Max number of sensors for each node: 16
- Max distance from node to sensor: 1m
- The chosen architecture is a stack of board with integrated on them the mote (transceiver, MCU) and a series of board with a PSoC and the connected sensors. These modules are connected through a bi-directional bus
- Plug&Play is the insertion or the movement of a board with its integrated PSoC and sensors. The modality of P&P is: 1) turn off the node; 2) connection (or movement) of the board; 3) turn on the node; 4) Transfer of sensor's data sheet from PSoC to mote.
- Measurement procedure is turn on the sensor for a warm up time and sampling the output at regular intervals.

- Descriptors:
    - ID sensor: *a unambiguous identification number, different for each sensor unit*
    - ID Type of sensor: *a number that identifies the type of the sensor (as "model" i.e. LM35 or TMP121)*
    - Physical or Chemical measured quantity*: definition of sensor classes i.e. Temperature Sensor or Pressure Sensor)*

- Order of magnitude of measured quantity: *sensor of the same class can have different measuring range; for each class of sensor must be used the relative International Standard (MKS) unit of measure.*
- Data format
- Timing Information: *a measurement can be instantaneous, with a fix or variable delay; a sensor can also contain historical data.*

I²C Features

In nearly every system there is some intelligent control (usually a single-chip microcontroller), general-purpose circuits (like LCD drivers, remote I/O ports, RAM, EEPROM, or data converters) and application-oriented circuits (such as digital tuning and signal processing circuits for radio and video systems, or DTMF generators for telephones with tone dialling). To exploit these similarities to the benefit of both systems designers and equipment manufacturers, as well as to maximize hardware efficiency and circuit simplicity, Philips developed a simple bi-directional 2-wire bus for efficient inter-IC control. This bus is called the Inter IC or I2C-bus.

Here are some of the features of the I2C-bus:

- Only two bus lines are required; a serial data line (SDA) and a serial clock line (SCL).
- Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exist at all times; masters can operate as master-transmitters or as master-receivers.
- It's a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer.
- Serial, 8-bit oriented, bi-directional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, or up to 3.4 Mbit/s in the High-speed mode.
- On-chip filtering rejects spikes on the bus data line to preserve data integrity. The number of ICs that can be connected to the same bus is limited only by a maximum bus capacitance of 400 pF.

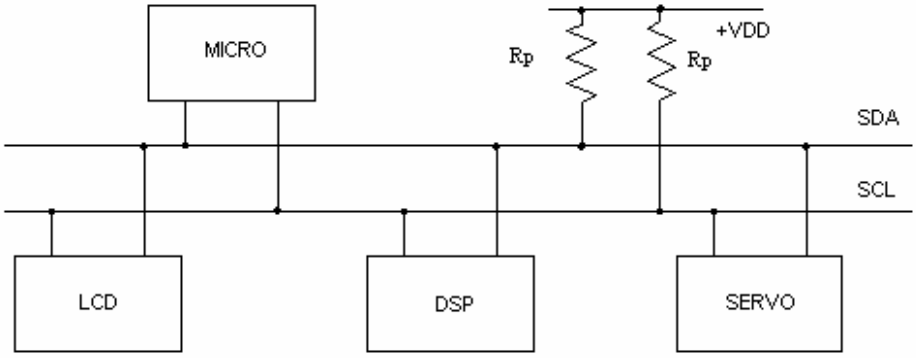


Figure 40: Connection of different devices to the I2C bus

Pull-up resistors Rp maintain the SDAH and SCLH lines at a HIGH level when the bus is free and ensure the signals are pulled up from a LOW to a HIGH level within the required rise time. An example of the connection of different devices to the I2C bus is shown in Figure 40.

IEEE 1451 standards family

Since 1993, IEEE and NIST have been developing different standards for smart sensors:

- IEEE 1451.1 "Network Capable Application Processor Information Model (NCAP)"
- IEEE 1451.2 "Transducer to Microprocessor Communication Protocol and Transducer Electronic Data Sheet (TEDS) Formats"

One of the key focuses of this family of standards is the interoperability of transducer sensors in different network protocols. An architecture for this interoperability is defined where the glue between the smart sensor and the network is provided by the NCAP. The smart sensor (STIM in the terminology of the standard) provides an standardised connection to the NCAP.

This standard family is an ongoing effort and there are two more that have not been fully accepted yet and there are under the ballot procedure

- IEEE 1451.3 "Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems"
- IEEE 1451.4 "Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet Formats"

There are even currently two more under development

- IEEE 1451.5 "Wireless Communication Protocols and Transducer Electronic Data Sheets (TEDS) Formats"
- IEEE 1451.0 "Family Interoperability"

A number of terms and acronyms have appeared as a consequence. The TEDS is a machine-readable specification of the characteristics of the sensor, with the intention to easy sensor installation and replacement. The STIM (Smart Transducer Interface Module) includes the TEDS and supporting electronics based usually in a microcontroller to interface the transducer to the NCAP (Network Capable Application Processor). The NCAP is a device that supports a network interface, application functionality, and capabilities to communicate with the STIM. A transducer independent interface (TII) is a 10-wire digital communication interface that allows the NCAP to obtain sensor readings as well as request of TEDS data. In fact, the TEDS (Transducer Electronic Data Sheet) is

a key concept to ensure true plug & play capabilitites for the sensors. They are described in more detail later on.

Communications between the STIM and the NCAP

Communications between the STIM and NCAP are defined by a 10-wire physical specification and a set of protocols to access the TEDS information, read sensors, and set actuators. Table 23 shows the digital lines specified in the Transducer Independent Interface (TII).

Basic communications between the NCAP and STIM require four lines (DCLK, DOUT, DIN, and NIOE) [6]. DCLK is driven by the NCAP. The data transfers are based on SPI-like (serial peripheral interface), bit-transfer protocol. The NCAP drives the NTRIG line to initiate a measurement or action, and the STIM uses the NTRACK line to acknowledge that the requested function has been performed. The STIM can notify the NCAP of any exception conditions by use of the NIO_INT line. IEEE 1451.2 defines a set of status registers to support notification of standard exceptions such as hardware errors, busy channels, STIM power cycle, calibration failure, and self-test failure.

Table 23 Signal and Control lines for the TII

Line	Description
DOUT	Data transport from STIM to NCAP
DIN	Address and data transport from NCAP to STIM
DCLK	Positive-going edge latches data on both DIN and DOUT
NIOE	Signals that the data transport is active and delimits data transport framing
NTRIG	Performs triggering function
POWER	Normal 5 V power supply
COMMON	Signal common or ground
NACK	Serves two functions: Trigger acknowledge and data transport acknowledge
NSDET	Used by the NCAP to detect the presence of the STIM (Hot-plug)
NIMT	Used by the STIM to request service of the NCAP

The NCAP can trigger an individual channel, or all transducer channels at once. In the latter case, there are TEDS fields provided to specify timing offsets between the STIM's channels and to determine when each measurement or actuation has occurred relative to the single trigger acknowledgment. The draft proposes that the slowest channel be the reference channel and that all the offsets be specified relative to this channel
The data transport function interacts with the trigger function in the way that it shall be inactivated before the trigger is asserted. A data transport frame shall begin by the NCAP sending an address to the STIM. The complete address specifies whether data shall be written to or read from the STIM, and which channel and function are involved.

The CHANNEL_ZERO functions, write global transducer data and read global transducer data, result in writing to or reading from the data structures of all implemented channels concatenated together, in order, beginning with channel one. This allows access to all transducers without the addressing overhead associated with accessing each individual channel. There are no delimiters between channel data structures. The data for each channel shall be parsed according to the data models recorded in each Channel TEDS.

There are seven transducer channel types that differ from another in the way they responds to triggering:

- A *sensor* measures some physical parameter on demand and returns digital data representing that parameter. A new data set shall be sampled once for each triggering event. The data set available to be read shall be the data set acquired as a result of the most recent trigger event.
- An *actuator* causes a physical or virtual action to occur that shall be related to the data set sent to the actuator. The actuator state changes to match the data set most recently sent to it when a triggering event occurs.
- A *buffered sensor* differs from a sensor in that the value is returned from the previous trigger. That means the response is immediate, but the time of the reading is uncertain.
- A *data sequence sensor* acquires data continuously, with sampling times under control of the STIM. Triggering selects a data set from this stream of continuous measurements and makes it available to be read by the NCAP. The data set acquisition timing need not be periodic.
- A *buffered data sequence sensor* is similar to the data sequence sensor, except that it makes the data in the holding buffer available. The trigger acknowledge is coincident with the availability of the previous acquired data set. If another trigger occurs after the holding buffer is read but before the acquisition of the next sample of data is complete, the acknowledge signal is not returned until the data acquisition process is complete.
- An *event sequence sensor* is the same as the data sequence sensor, but it doesn't return any data. For this sensor the time of the event is the relevant information.
- The *general transducer* is a category that allows for the presence of channels that behave differently from the above types. These types shall implement the same functions required of all channels, but this standard does not specify

behaviour with respect to triggering or data written to or read from such a channel.

TEDS

The TEDS are scalable and extensible electronic data sheets that allows transducer manufacturers to *burn* in specific fields about their product such as manufacture date, version information, and calibration specifics, etc., into a

small nonvolatile memory associated with the transducer hardware. With a requirement of only 256 bytes of memory for the mandatory data, the TEDS is fully scaleable. TEDS makes possible the self-identification of transducers to the system or network. This capability is important for maintenance, diagnostics, and determining mean-time before-failure characteristics.

Since the transducer manufacture data in the TEDS always goes with the transducer and this information is electronically transferred to a NCAP or host, human errors associated with manual entering of sensor parameters into the host are eliminated. Because of this feature of the TEDS, upgrading transducers with higher accuracy and enhanced capability or replacing transducers for maintenance purpose is simply "plug and play".

The TEDS are defined in IEEE 1451.2 and then redefined in IEEE 1451.3 and IEEE 1451.4 to accomplish the specific necessities in these standards. Here TEDS from 1451.2 and 1451.4 will be summarized since these are the most interesting for our application.

The TEDS structure is divided into:

- Meta-TEDS (one per STIM), which contains data common to all the transducers connected to the STIM. The field contains information such as overall description of the TEDS data structure, worst case STIM timing parameters, etc.
- Channel TEDS (one per STIM channel), with information for one specific transducer in a STIM. Each transducer has its associated Channel TEDS containing information such as physical units, uncertainty, upper/lower range limits, warm-up time, etc.
- Calibration TEDS (one per STIM channel) including information such as calibration parameters, and calibration interval.
- Application-specific TEDS (multiple per STIM), which are for application specific use by end-users.
- Extension TEDS (multiple per STIM), which is reserved for implementing future extension to P1451.2.

P1451.4 TEDS

The P1451.4 TEDS shall be a subset of the 1451.2 TEDS with the goal to minimize the size of nonvolatile memory. The key consideration for TEDS design are: relevant information that help the user, "plug-and-play" functionality, support for all transducers, openness for individual needs and compatibility to 1451.2. The P1451.4 TEDS shall include the following categories:

1.Identification Parameters
a. Manufacturer name
b. Model number
c. Series number
d. Revision number
e. Date code

2.Device Parameters
a. Sensor type
b. Sensitivity
c. Bandwidth
d. Units

3.Calibration Parameters
a. Last calibration date
b. Correction engine coefficients

4.Application Parameter
a. Channel identification
b. Channel grouping
c. Sensor location and orientation

The most recent proposed TEDS include a 256-byte non-human-readable compact-TEDS and additional human-readable TEDS. The compact-TEDS enables the minimum implementation of a P1451.4 TEDS for transducers which have limited memory or size.

In order to cover a broad range of sensor types and applications while also keeping memory usage to a minimum, the IEEE P1451.4 TEDS utilizes the concept of templates that define the specific properties for different sensor types. The draft standard also allows manufacturers to define custom subtemplates that can be used instead of, or in addition to, the standard templates to accommodate specialized parameters and requirements.

The Status of the P1415.4:

One of the controversial issues of the new existing standards 1451.1 and 1451.1 is the definition of a TII consisting of 10 wires. From the industrial side there has been a strong push towards the definition of an standard with minimal wiring needs (2-3 wires) and with the possibility to be compatible with analog current loop transmission (mostly used in already working installations). The 1451.4 standard tries to address these points. The next paragraphs briefly review the status of this standard.

Almost two years ago, the preliminary IEEE P1451.4 "Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet Formats" was presented as a standard defining a mechanism for adding self-identification technology to traditional analog-mode sensors and actuators.

Since then, there have been many advances in the proposed IEEE P1451.4 standard, and the availability of sensors, data acquisition hardware, software, and related tools incorporating IEEE P1451.4 has increased in the last months [3, 4, 5].

The goal was to approve a final version of the standard by the end of 2003, but nowadays it is in the final approval process.

Conclusion:

The actual system requirements do not exclude any of the two identified protocols. I^2C has the advantage of being a standard used in many devices, therefore it does not introduce additional complications. The descriptor is not defined and consequently it is customisable for the specific requirement of the project. This protocol could be a limit for eventual future developments.
1451.2 requires a greater use of programmable interface, a greater work for the software development (both node and programmable interface) and has a very large descriptor. 1451.2 has the advantage that covers a more wide field of cases.

5.1.5.2 Node to node

The peer-to-peer communication protocol involving the sensor network nodes could be effectively described by resorting to the stack layer approach, that is attributing to each layer a set of functions related to the same communication aspect and, hence, developing standard interfaces between layers.
According to the widely accepted ISO/OSI taxonomy, the following layers have to be considered within our investigation, since they involve link-to-link communications:

- Physical Layer;
- Medium Access Control (MAC) Layer;
- Network Layer.

Though this approach allows easy substitution or integration among different layers in a modular way, however the data passing between layers is not well defined, leading to a suboptimal scheme.
This problem can be faced by means of the so called *cross-layer* approach, where basically the Physical and MAC layers knowledge of the wireless medium is shared with higher layers, in order to provide efficient methods of allocating network resources and applications over an IP-based network.
This leads to a design philosophy that blurs the lines between layers and attempts to optimize across layer functionality.

Generalizing the above idea, we can refer to a cross-layer *adaptability*, in which all the network functions are allowed to pass information between functions and to adapt simultaneously by reconfiguring the protocols parameters or procedures. As a result, WSN is able to dynamically tackle with the traffic load (congestion) and QoS requirements variations, the network topology changing and the channel conditions (fading and collisions) as well.



Physical Layer (RF communications)

This layer functionalities are mainly involved with the electromagnetic communications setting up and maintaining. To this end, several issues might be clarified.

First of all, the choice of the proper spectrum, whose range is likely to be within VHF up to UHF with a narrowband communication. Strictly connected with this topic there are the power management issues, that is selection of transmitted power level and its control along with the communication to match the requested performance together with avoiding power wasting. In particular, being the radio transceiver able to sample the level of background noise when there is no active transmission, it can, therefore, adjust the transmission strength to improve the overall efficiency. As a consequence, the transceiver can be either in active state (both transmitting and receiving) or in idle mode, depending on the software capability to quickly and predictably turn radio power on or off by means of a time slotting, which has to be provided by means of a synchronization protocol via pilot symbols. Therefore, the access scheme is a Time Division Multiple Access (TDMA) scheme.
Finally, the signalling scheme is to be characterized, once a certain target packet error rate has been fixed. Our system relies on a digital modulation as the Pulse Amplitude Modulation (PAM), thanks to its good complexity/performance trade-off.

MAC Layer

This layer is in charge of managing and controlling the communications across the physical medium, of assembling and disassembling frames and, moreover, of performing error detection and addressing function.
The first aim is accomplished by firstly assigning to each node a unique identification number by means of including a silicon serial number, that is a low-cost ROM device with a minimal electronic interface and no power requirements. Then an access scheme can be developed by resorting to the Carrier Sensing Multiple Access (CSMA) algorithm, according to which, whenever a node has data to send, it senses the channel for a carrier transmitted by any other station. If it detects that the medium is idle, the transmission is allowed, while, in the opposite case, after a random waiting time, it begin a new contention phase. However, due to the finite propagation delay of electromagnetic signals, a collision may occur if several nodes begin transmitting after having erroneously detected that the physical medium is free. Then, a CSMA scheme endowed with a collision detection mechanism (CSMA/CD) is needed, provided that there are no coupling between transmitted and received power levels, or, simply, an exponential back-off algorithm to randomized the transmission beginning.
Obviously, the performance strictly depends on the accurateness of the timing procedures.
Moreover, several powerful channel coding schemes are taken into account to provide the expected reliability. Depending on the transmission and coding rates, it is possible to derive the communication throughput, that is the rate effectively provided to data exchange.

Network Layer

The routing algorithm we have applied takes advantage of the multi-hopping approach with the aim of minimizing the battery consumption, together with providing a high end-to-end delivery efficiency.

In particular, we develop a reactive and location-aware routing protocols, in which, basically, the sender select from its neighbours the closer to the gateway (thanks to an ID to location mapping) and deliver to it the data packet, and so on, until the destination has been reached.
This protocol is integrated with the Physical layer, since the power management is taken into account by awakening the sleeping destination node at every forwarding attempt. Moreover, the routing protocol is joined with the MAC layer, as if a negative acknowledgement is received or, equivalently a timeout is expired, the forwarding procedure is set up again.

5.1.6 The node data processing

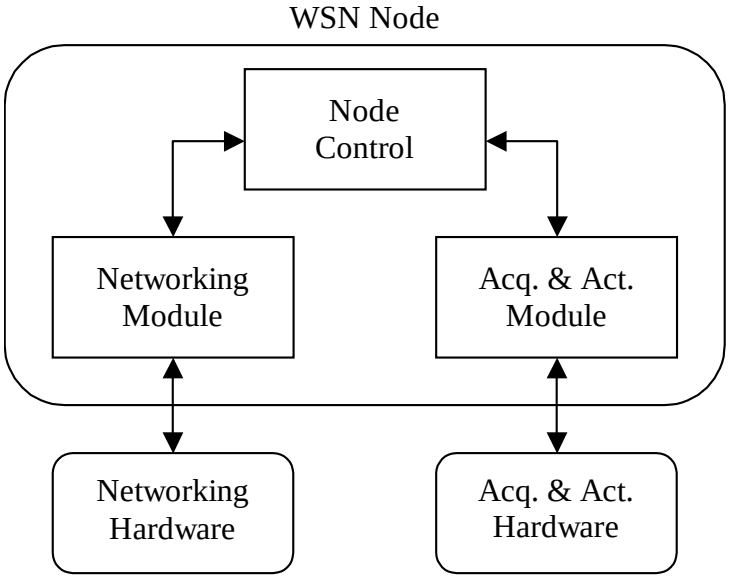


Fig 41: schematization of the main software modules governing a network node

Each node within a Wireless Sensor Network must be able to perform two basic tasks: read measurements from its connected sensors and transmit data to the gateway. From a software architecture perspective, these functionalities are enclosed in two software modules that can be referred as "Networking Module" and "Acquisition & Actuation Module". See Figure 38
The "Networking Module" is a software element responsible to manage the wireless communication with other nodes. This module implements the communication stack described in the "Node to Node Protocol" paragraph. The networking module defines the routing policies used to transfer packets in a multi-hop network and manages the low-level details related to packet format and physical channel access. At the lowest level, the networking module interacts

with the hardware used to communicate with other nodes. This is mainly represented by the radio transceiver, but not limited to it since the node might be equipped with other hardware for different kind of communications (IrDa, RS-232, I2C, …)
The "Acquisition & Actuation" module is responsible for the interaction with sensors and actuators that might be connected to the node.

This module implements the "Node to Sensor Communication Protocol" that enables the use of smart plug&play sensors, as described in section 2.1.3.1 of this document. Similarly to the networking module, at the lowest level, the Acquisition & Actuation module controls the actual hardware used for data acquisition and actuation.
The functionalities of the two software modules described above are coordinated by a third module, the "Node Control" which is used to manage every other aspects not strictly related to data acquisition or communication. In its most basic form, the "Node Control" has to take the measurements from the "Acquisition & Actuation" module, complete them with a few other information related to the node itself (e.g. node's ID, node's location, battery voltage….) and instruct the "Networking Module" to send a data packet to the gateway.
Some of the working parameters of the node (e.g. the sampling rate) need to be programmable by the user or by higher levels of the AmI infrastructure, so the "Node Control" module should be also able to interpret and process commands coming from the network.

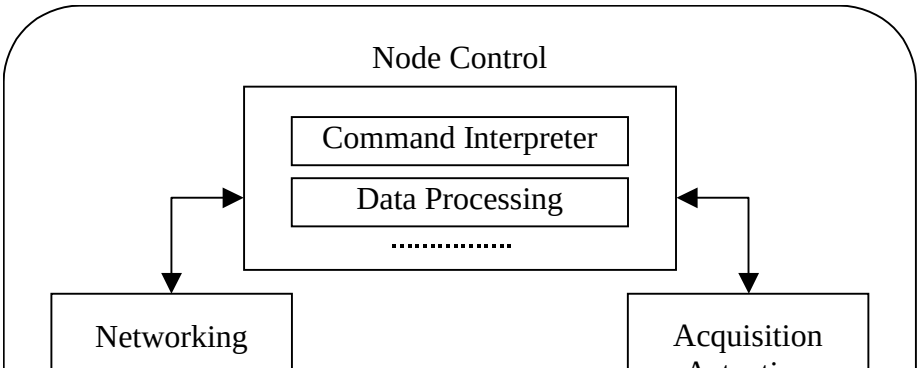


Fig. 42: among other functionalities, a node must be able to process commands and perform some kind of data elaboration.

The ability to process and interpret commands is enclosed in the "Command Interpreter" sub-block outlined in fig. XX. The same figure also highlights a further block referred as "Data Processing" which has been thought as a way to provide the node with extended processing capabilities. Elaboration algorithms that a node could be requested to perform fall under three main categories:

a) Sensor Compensation.

In most cases, basic signal correction will be needed at the node level to compensate undesired cross-sensitivities on the measurements, like temperature or humidity compensation. In special cases, the sensors will provide only partial information due to insufficient selectivity (this is the typical case for low cost chemical sensors). In this cases multivariate data elaboration, from a chemical multisensor array is necessary. At this phase of the system design, it's not clear if this kind of compensation should be performed by the sensors themselves (smart sensors), by the node or by higher level of the Ami architecture. The inclusion of a "Data Processing" module serves as an entry point for the execution of this elaboration if the node will be requested to perform compensation on sensor readings.

b) Event detection.

In some cases, sensors will be working as detectors. Drift robust event detection algorithms integrated in the node will enable the communication only in cases of interest while disregarding ´normal´ data. The Data Processing module can trigger notifying events when the value of a single sensor exceed a predefined threshold or when special patterns are detected in data from multiple sensors.

c) Advanced data reduction algorithms.

In the search for ultra low power consumption, transmission of information from node to node has to be kept to a minimum. It can be easily demonstrated that communication power consumption overwhelms computing power consumption. For this reason is recommended that nodes do only communicate meaningful information and only on demand raw data. Signal elaboration at the sensor node is a pre-requisite for efficient power management. Moreover, AmI nodes will be inherently multisensor systems, so some kind of sensor data fusion is desired at the node level. The microprocessor could combine partial information from different sources to meaningful, high-level information.

The Data Processing module is intended as a component providing the node with advanced data processing capabilities that will be investigated in later stages of the Ami infrastructure development. It can be reasonably assumed that the deployment of the WSN will consist of approximately three phases in which the system will operate differently. During the first phase (test phase), the data will be simply routed from the sensors to the gateway without any additional elaboration to facilitate the debug of the Networking and the Acquisition & Actuation modules. The second phase will start when the sensors prove to give correct measurements and the wireless network can reliably transfer such information to the gateway. During this phase, the users will familiarize themselves with the system and the higher levels of the AmI architecture (e.g. the KDD system) will start analyzing data to find interesting correlations. Data coming from the sensors still need to be transferred unprocessed so that both the users and the AmI infrastructure can perform a sound analysis on them. A the end of this phase, the user, supported by the AmI infrastructure, should be able to define if there are useful elaboration that can be performed on sensor data in order to make them more meaningful. For example, the user might want to aggregate a multitude of temperatures values in a few significant parameters (average, min and max). Although this kind of operations can be easily performed by high-level software tools that operate on values stored in a

database, it would be preferable if such elaboration could be performed locally by the network nodes. Data processing, like the aggregation of more values, can significantly reduce the size and the number of data packet exchanged within the sensor network. Since data transmission require much more energy than elaboration, the average battery life of nodes could be extended by enabling such kind of data processing within each node.

5.1.7 Node to gateway communication protocol

Sensor readings are collected by nodes and transferred to a 'Mother Node' using the wireless network infrastructure. The 'Mother Node' routes the packets received from the WSN Nodes to the gateway, which transfer them to the higher-level AmI components through an IP-based network. In a similar way, queries and commands originated by high-level Ami components are transferred to the WSN using the WSN gateway and the 'Mother Node'. In other words, the "WSN Gateway" and the 'Mother Node' together serve as a bridge between the IP-based network and the Wireless Sensor Network. By using this bridge, the high-level AmI components can communicate with the wireless sensor nodes and vice versa.

IN ORDER TO PROVIDE THE DESIRED FUNCTIONALITIES, THE WSN GATEWAY AND THE MOTHER NODE WILL BE CONNECTED USING A WIRED LINK AND AN APPROPRIATE HARDWARE INTERFACE.

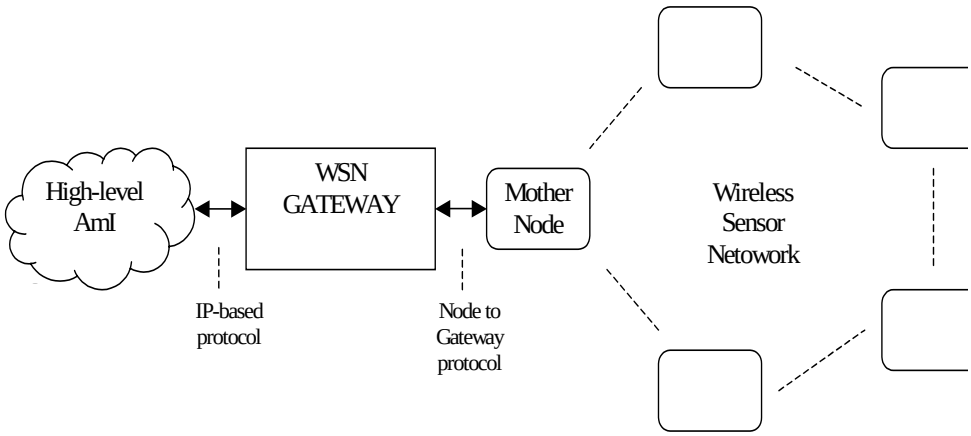


Fig. 43: the WSN Gateway in conjunction with the 'Mother Node' serve as bridge between the High-level AmI components and the Wireless Sensor Network.

Both the Gateway and the Mother Node are embedded systems based on a microcontroller, so the communication link can be implemented using one of the I/O interfaces available to both the systems (USART, SPI, I2C or a custom bus using GPIO lines).
In addition to the physical link, a communication protocol is required to guarantee a sound communication between the two subsystems. Since the mother node is identical to the other WSN nodes, the TinyOS built-in networking stack can be use to implement the node to gateway communication protocol

Each WSN node implements a multi-hop communication protocol that relies on a mechanism capable to route packets that originate from other nodes and are directed to some other point of the network. The same routing scheme can be adopted by the 'Mother Node'. The only difference respect to other WSN nodes is that the 'Mother Node' posses two communication interfaces and the routing algorithm need to select the appropriate interface depending where the packets goes.

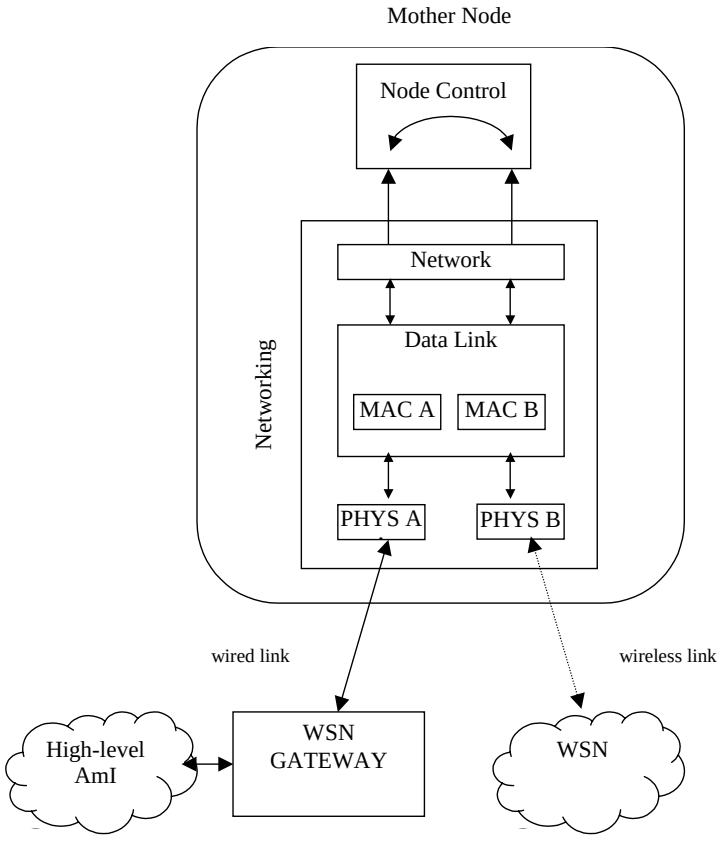


Figure 44: the 'Mother Node' can use the same networking stack implemented on common WSN nodes to route packets from the WSN to the gateway

Figure 44 show a diagram of the software and hardware configuration used by the Mother Node to route packets from the WSN Gateway to the WSN and vice versa.
The Networking module is implemented using some of the networking capabilities of the TinyOS operating systems, which allows to transparently access different communication resources. For example, a same data packet can be indifferently transmitted using the wireless radio or the USART: the differences between the two devices are hidden in the lower levels of the communication stack (MAC and PHYSICAL).
According to this schema, routing a packet from one interface (e.g. the radio) to another (e.g. the USART) become extremely simple and effortless.
The only requirement for this approach is to have the gateway being able to handle communication packets in the same format used inside the WSN.

Since the communication protocol used by the higher-level of the AmI infrastructure won't be the same used within the WSN, it's clear that at some stage a conversion between the two protocols will be required.
That can be performed by the gateway or by more conveniently by the protocol handler.

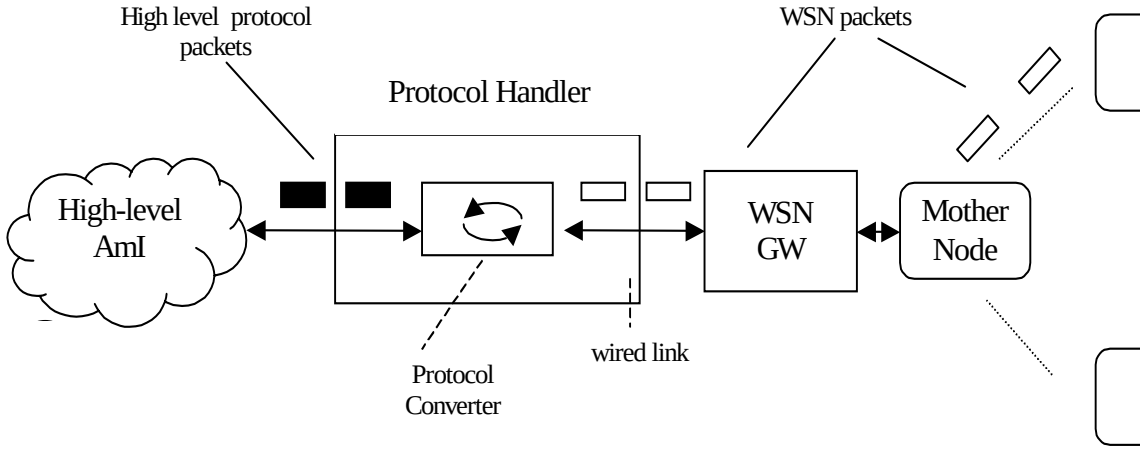


Fig. 45: the conversion of high-level protocol packet into simpler WSN packet can happen at different stages. Figure shows an example of such conversion performed by the Protocol Handler.

6 The AmI GRID software architecture

6.1 The Communication Protocol

In the GoodFood AmI Scenario the lower level sensors network is built of a set of heterogeneous sensors which are connected to a set of homogeneous platforms called the nodes of the network. These nodes are possibly made of microcomputers, such as the MICA Motes connected through a wireless communication network. The logical exit point of the network towards the higher levels of the architecture, is represented by a microcomputer, called Gateway and connected through a wireless connection to a PC which is called Protocol Handler (PH). The PH:

- interacts with the lower level sensor network sending it instructions received from the AmI framework and
- receives observed data from the sensors that has to be forwarded, using the bus-like communication infrastructure to the framework tools for storage, analysis and monitoring.
-

This section describes the protocol that wraps low level sensor data for transport, storage and management in the AmI Application level.
The interface between the lower level protocol and the higher level protocol is located in the PH.

The PH is responsible for wrapping and unwrapping the low level protocol, using a more structured and more general higher level protocol that will be used for transport on the AmI bus and possibly for storage into the AmI databases.
This section, thus, applies to the grayed area of Figure 46.

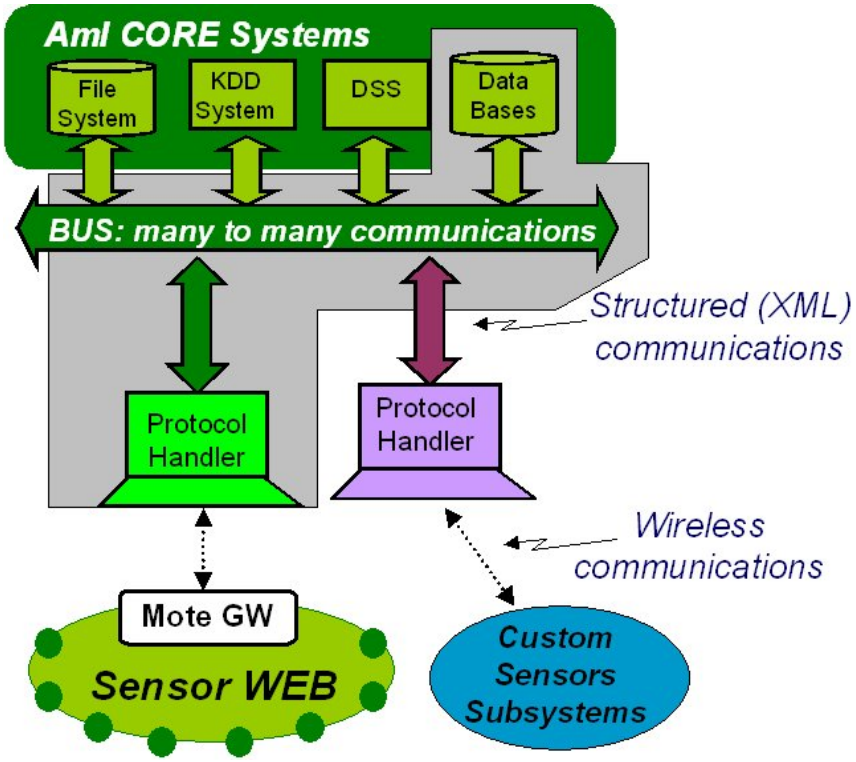


Figure 46

In particular, the high level protocol should consider the following tasks:

- Wrapping of the resource management information coming from plug&play sensors.
- Unwrapping of queries from the AmI framework systems or from human operators and related to resource management or data retrieval.
- Wrapping observed data coming from the sensors.
- Unwrapping of commands generated by AmI framework systems or by human operators and directed to the sensor network actuators.

The main requirements of the high level protocol should be based on open standards and it should be quite general and extensible since GoodFood applications will range over a great variety of different scenarios.
To accomplish the requirements related to the AmI framework stated above, XML (eXtensible Markup Language) has been chosen as the basis for the high level protocol. XML is an open standard for information transport and storage which is based on a standard ASCII representation of the information using mark up tags. Several tools for parsing, building, validating and transforming XML documents are publicly available and there exist also open source software products for

automating the creation of DOMs (Domain Object Model) that maps the information of an XML document into an object oriented memory model.
Choosing XML as a basis means that the designer of the protocol has to concentrate on developing a model consisting of XML tags organised in valid documents to represent all sort of information that travels on the AmI bus, that is stored in the AmI databases and that crosses the PH boundary in communications with the sensor networks.
The development of a new protocol that potentially faced all the tasks described above soon turned out to be a difficult task because of the complex abstractions that had to be made for keeping a good level of generality.
In effect the need for a particular domain expertise appeared to be necessary and the following key points had to be faced from a more concrete and specialized point of view:

- **Sensor Heterogeneity**: The huge quantity of heterogeneous sensors that could possibly be used in GoodFood scenarios has to be faced by a very robust way of modelling the sensor descriptions. Information on a big number of structurally and functionally very different sensors has to be kept in consistent databases or has to be displayed on sophisticated, integrated and well-designed graphical user interfaces.
- **Observation Heterogeneity**: Strictly related to sensor heterogeneity is the concept of observation heterogeneity. The observations differ among heterogeneous sensors because of the units of measure, data type and size (percentages, scalar values, vectorial values, multimedia values, colour tables), frequency of arrival. The observations can also differ among homogeneous sensors because of different scales, aggregation functions or timing issues. All these different observations have to be marshalled into a common model that is general enough to handle the intrinsic heterogeneity.
- **Node Sensor Relation**: There is a complex question regarding the relation among sensors and network nodes. In general nodes are the platforms to whom sensors are attached but there could be other relations such as a Node being itself a sensor or for example the usage of portable devices with sensors embedded into them as dynamic moving sensors.
- **Actuator Interface**: The protocol should consider a generic interface for interacting with the sensor network by sending a command to the sensors for querying new data or reconfiguration. But also the interaction with actuators should be considered as a possible extension of the sensor network. Actuators are devices that perform some action on the domain of the sensor network. They could be steered by a human operator or by an automated system that reacts on some predefined events. In any case a high level interface should be implemented that uses the high level protocol as a mean of transporting commands to the actuators.
- **Geo-location**: To solve problems related to the location of the sensors inside a geographic area or on complex geometries expertise on geographic information systems is required.

6.1.1 State of the art

The necessity to face all these questions lead us to a research activity that looked to analyze the state of the art concerning XML based specification of sensor networks models.

OGC
The first example of an XML specification that models sensor related application is the Open GIS Consortium's (OGC) Sensor Web Enabling infrastructure.
The infrastructure proposed by the OGC is based on two XML document specifications. GML (Geographic Markup Language) is a formalism for modelling geographic data in a very general way. It introduces the concepts of natural phenomena called features, observations, complex geometries, continuous and discrete coverages, geographic and time coordinates. SensorML (Sensor Markup Language) encompasses a set of documents for modelling different heterogeneous sensors and platforms in terms of functional specification, legal and usage specification, sensibility, measured phenomena, geographic location.
The OGC specification considers also the modelling of standard discovery interfaces and a mechanism for integrating the GML and SensorML documents into Web Services environment.

cGML
cGML has been developed by the Network Distributed Applications (NDA) Area of CRS (Center for Advanced Studies, Research and Development in Sardinia).
cGML aims at modelling a general way of describing cartography information. The target platform of cGML is the mobile world (palm tops, PDAs, cellular phones).

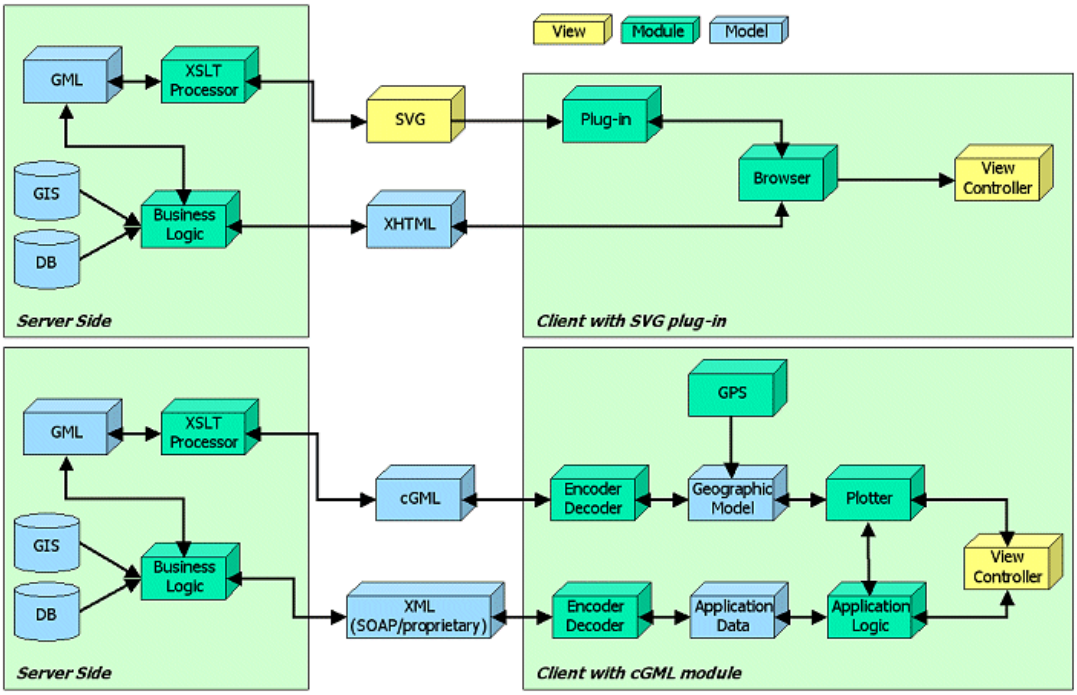


Figure 47: Possible architectures of cGML applications.

This imposes a set of important constraints related to energy consumption, communication bandwidth and size of data.
In this scenario cGML is based on a restriction of the OGC specifications. Only a set of the total specification has been considered for implementation: the one useful for designing urban cartographic maps. Moreover the tags suggested by the standard OGC specification have been customised to a more compact form to save some useful bytes for optimising communication and data size.
Figure 45 shows two architectures of a possible cGML based application. cGML like other XML based specification relies on the ease of creating XML-to-XML transformation through XSLT for transforming the data into other XML formalisms like Scalable Vector Graphics (SVG) for presentation.
cGML is not useful directly for itself. However it showed how the over-sized OGC specifications could be successfully tailored to specific application needs creating a sort of dialect of the original specification that matches more closely the proper system requirements.

TinyML:

A further refinement of the research activity took the focus to another specification called tinyML. This specification is also based on XML. The authors of this specification (Bill Kramer and Nate Ota at NERSC and Berkeley) were interested in creating a modelling language for targeting explicitly sensor networks (what they call Meta-data for Wireless Sensor Networks - mWSN) and removing some of the main limitations of the OGC specifications.
The authors state that the OGC specification is based on NASA's need for public distribution and thus the specification is very heavyweight and probably more well suited for addressing scenarios involving remote satellites, were there are

fewer sensors who produce a big quantity of data. Main criticism towards the applicability of OGC specification to sensor networks is expressed in the following terms:
"GML and SensorML are designed for high bandwidth TCP/IP links and the large data exchange due to the complex XML schemes are not constrained by bandwidth limitations. In wireless sensor networks bandwidth is one of the more critical topics to be faced.
"GML and SensorML are designed for individual sensor data collection. mWSNs often focus on sensor groups for such tasks as energy conservation through data aggregation and direct sensor querying.

"Unlike motes, the satellite-based platforms have a significantly higher energy budget, particularly in earth orbit. However, mWSN nodes would soon burn out their energy resources from transmitting/receiving such large data packets.
"GML and SensorML are designed only for data collection. However, as mWSNs progress into the control (actuation) realm, feedback loop capabilities will become increasingly important.
The fact that tinyML is energy aware (it considers energy as an important factor when describing sensors and sensor activity), that it has an actuator interface for programming the sensor network behaviour and that it is able to define virtual sensors that can easily provide aggregate observations makes this tool well suited for sensor network applications.
Finally the tinyML specification is already freely available in a prototype version. Because it is really small with respect, for example, to the huge OGC specification, it is easily modifiable and can be adapted with small effort to the specific application needs. The fact that its specification is based on XML makes tinyML also easy to maintain and to extend for future usage.

6.1.2 Gateway to Protocol Handler Communication Protocol

As described above, the interface between the Remote Data Acquisition(RDA) units and the higher level of the overall AmI system architecture is represented by the Protocol Handler (PH). PH is responsible for wrapping and unwrapping information that is exchanged between the two levels of the architecture for transforming it into the respective protocols.
The PH is responsible for:

- encapsulating the information coming from the RDA units towards the higher level of the AmI framework (Resource Management information, observational data, status notifications) into a robust, general and flexible high level protocol based on ***tinyML*** ;
- unwrapping the data produced by the AmI framework, to configure, query and update the RDA units status enabling also the translation into the protocol used for communication inside the RDA.

This section shows what kind of information is meant to be exchanged between the RDA and the higher level of the AmI framework and to outline a basic

structure of the used protocol between the Gateway of the RDA and the PH, called *GW2PH protocol*.

6.1.2.1 Resource Management Information

Resource Management information is related with the description of the RDA entities. In particular, each entity can produce information in a "Plug And Play (P&P)" manner. Moreover the Wireless Sensor Network (WSN) entities (due to the fact that up to sixteen sensors are connected to every node), may send messages stating the disconnection of a sensor from the WSN. Possibly, status information can be produced from sensors inside the WSN to notify their actual functional status. All this information is then sent to the gateway that forwards it to the PH.

6.1.2.2 P&P Connection

A message containing P&P information has to carry a set of predefined values describing the connected entity or sensor (hereafter also each P&P sensor connected to the nodes of the WSN will be indicated as an entity). This information should consider at least the following:

- **Identification:** information related to the identity of the entity that has to be produced. This information will reflect the hierarchical identification pattern described in Figure 1. With this pattern each entity has an identifier that is unique among all the entities of the same layer. The identifier of an entity at a higher level is built from the concatenation of the identifier of the entity with all the identifiers of the higher levels. Each entity has to produce its local identifier, which is then completed during the communication of the P&P message upward to the PH by attaching to it the identifier of all the entities encountered in the WSN tree.

- **Localization and Orientation**: a mechanism for providing localization information has to be provided; in every case, a correlation among entities and geographic position is required. This information is embedded into a P&P message and has to provide either a pair of relative/absolute coordinates or simply one absolute coordinate according to defined systems (geodetic, cartesian, or custom coordinate systems). Moreover, for pointing devices or optical sensors it could be interesting to know information about the orientation of the device.

- **Description**: descriptive information has to be supplied to identify the entity which has been connected. Examples of this sort of information are: Type, Vendor, Unity of Measure, Accuracy, Range, Hardware Equipment (Transmitter, Energy Source, Processor), Software Equipment (Operating System).

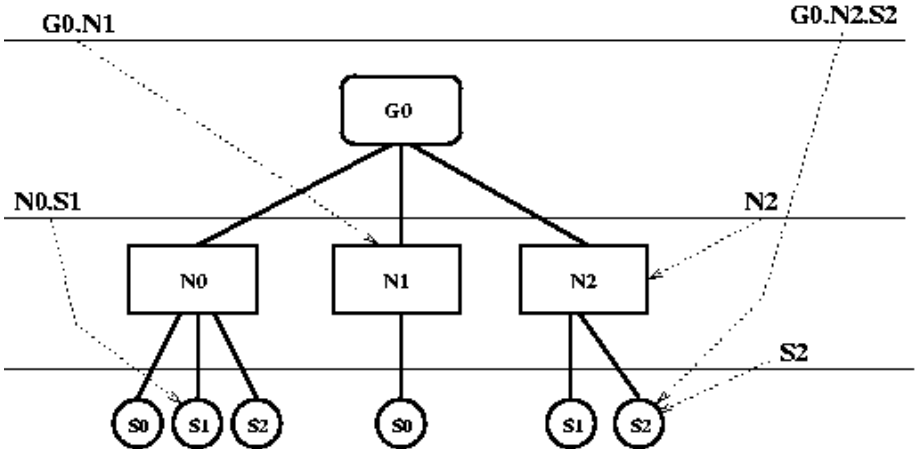


Figure 48: Example of hierarchical identification.

6.1.2.3 P&P Disconnection

Disconnection event is difficult to be implemented and in some cases it might be even impossible to produce a message stating the disconnection of an entity. However, when possible, the entity should try to produce a disconnection message that will be sent to the PH. In this way the resource management

databases can automatically be updated to keep track of the actual configuration of the RDA.
A message containing disconnection information could simply contain a header stating the type of the message and the identifier of the entity that has been disconnected.

6.1.2.4 Status update

For each type of the RDA entities a certain amount of status attributes can be monitored. This set of attributes has to be encoded in the GW2PH protocol. In this way an entity can automatically produce a message containing current values for those attributes. This message, along with its type description and the identifier of the entity, can be sent upward, through the PH node, to update the resource management information system.

6.1.2.5 Observational Data

Observational data is produced by the sensing activities produced by the RDA entities . A message describing observational data in the GW2PH protocol has to contain the following information:

- **Identifier:** an identifier of the entity that produces the data has to be inserted to enable higher level applications to correctly interpret the observation. Through this information the sensing entity descriptor can be retrieved and the observational data can be read according to the specification of the entity contained in this descriptor.

- **Time Stamp**: A time stamp has to be associated by the system to the observation. This is necessary to classify the information of the observation according to a chronological sorting. There are two main problems with the generation of time stamps to associate with the observation.

    - The first one is that there might be many RDA entities that are not able to produce time stamps. To face this problem a further responsibility could be added to the PH. This means that if an observation doesn't present a time stamp, the PH attaches to the observation a time stamp recording the time of arrival of the message containing it.
    - The second aspect is the difficulty of keeping all RDA units at least nearly synchronised. The PH could also serve as a centralization point for applying synchronization algorithms.

- **Data**: The data of the observation in a format that respects the definition given by the description of the entity stored inside the resource management database.

6.1.2.6 Functional Aggregation and Transformation

A query interface is provided to enable agents in the application layer of the AmI framework to request an observation from the RDA units. These observations may be some sort of aggregation or functional transformation of the observations produced by the RDA entities.
These requests are presented to the PH, from the higher level applications, in form of virtual sensors. The PH has to transform them into sequences of commands to be sent to the RDA.
The PH has to sent, to the processing enabled RDA entities, a command describing:

- **The function:** possibly an associative operator like MIN, MAX, SUM, AVG to be applied to the input data.

- **The span of members**: the set of RDA entities that participate in building the aggregate observation.

The shifting of these kind of operations down to the RDA enties can be made to implement energy saving strategies that tends to perform as much computation as possible locally in the RDA or in the WSN node; this is intended to save communication events, which are among the most energy wasting activities inside the WSN.

However, there could be many cases where this strategy would be a bad approach, for example when the *span of members* includes entities who are too far away from each other. To face these cases, a software Virtual Sensor Resolver can be installed inside the PH. In that way, the *function* is executed by the PH on the data coming from all the WSN entities included in the *span of members*.

6.1.2.7 Querying

Another interaction between the higher level of the AmI framework and the RDA entities is through the **Query Interface**. In particular, the systems of the AmI-GRID application level can post explicit queries to the RDA entities. Although the actual execution of these queries is always asynchronous at the lower levels of the architecture, they can be viewed as synchronous queries at the application level.
From the view point of the higher level systems, each query is posted in the form of a new virtual sensor.
For implementing the query mechanism, the query description stored inside an entity has to be modified.
To address the observational behaviour of a entity the PH has to configure and send to the RDA level the following information:

- **Identifier**: the entity to be queried.
- **Lifetime**: How long should the query last before expiring.
- **Resolution**: the time interval at which the query should produce new data.
-

6.1.2.8 Gateway-side communication interface

As previously mentioned, the Gateway is designed to provide a standardized and transparent interface to the upper level AmI framework.
The *Physical* and *Data Link* layers of the stack implemented on the Gateway side are selected according to the specific subsystem requirements: outdoor/indoor location, required bandwidth, power constrains. For the M18 pilot site implementation, a GPRS wireless data module should meet the requirements exposed for the subsystems deployed in the vineyard; for the cellar section, also cabled connectivity solutions (e.g. DSL) will be considered.
The *Network* and *Transport* layers will be implemented following the standard IP (RFC 791) and TCP (RFC 793) protocols; lower layers must therefore support IP based stack protocols. According to this approach, each Gateway will be assigned a (preferably static) valid IP address; consequently, a configuration procedure is required to the Gateway to notify the Protocol Handler of its assigned IP address, as a "Plug and Play" configuration attribute.
Custom application protocols will be implemented over the TCP-IP stack, either on Gateway and Protocol Handler side; more details are reported in the following sections.

The Gateway is responsible to keep the connection alive; alternatively, periodical connections can be managed by the Protocol Handler, in order to minimize the power consumption on the Gateway side. In this case, all the parameters concerning the connection availability must be provided by the Gateway as "Plug and Play" configuration attributes.

6.1.2.9 Gateway to Protocol Handler communication procedures

As discussed in the previous sections, the communication sessions established between Gateway and Protocol Handler can be classified as follows:

- Plug and Play/Resource Management Message: it is sent to the Protocol Handler to notify updates involving P&P attributes of Remote Data Acquisition (RDA) entities (e.g. new platforms and/or sensors are installed in a Sensor Network);
- Configuration/Resource Management Message: it is forwarded from the Protocol Handler to a Gateway whenever configuration parameters are edited on the user's side, to change settings of RDA entities;
- Observational Message: it contains sensor-originated data values, reported either from a physical or from a virtual sensor; it is forwarded from a Gateway to the Protocol Handler;
- Query Message: it is sent from the Protocol Handler to a Gateway, whenever a new data acquisition is requested; it is implemented on the Gateway side only if asynchronous data queries are supported by the specific subsystem.

A two-way handshaking approach will be implemented; an ACK message must be sent by the Gateway and the Protocol Handler whenever they receive a valid message. The data exchange procedures can then be indicated as follows:

the meaning of ACK message is to advise the counterpart that a formally valid message has been received; this could be very significant when the communication channel introduces considerable latencies, as for example in a GPRS TCP/IP session. Moreover, it is also meaningful when waiting for long delay operations (i.e. querying a sensor of a wireless sensor network could result in a very long delay response, depending on the power management procedures implemented and on the network topology).
Therefore, the ACK message is not a confirmation of a successfully operation, but only a response to a formally valid message received, exchanged between the Protocol Handler and a Gateway.
The ACK message will also contain a field indicating if the corresponding message has been correctly forwarded to the lower level subsystems, or not (i.e. because the requested operation is not supported or the command is unknown). Other message exceptions could be also originated from lower level subsystems (i.e. a configuration parameter has been rejected from a Platform or a Sensor object), but in this case a new message exchange session will be established reporting the message exception description.

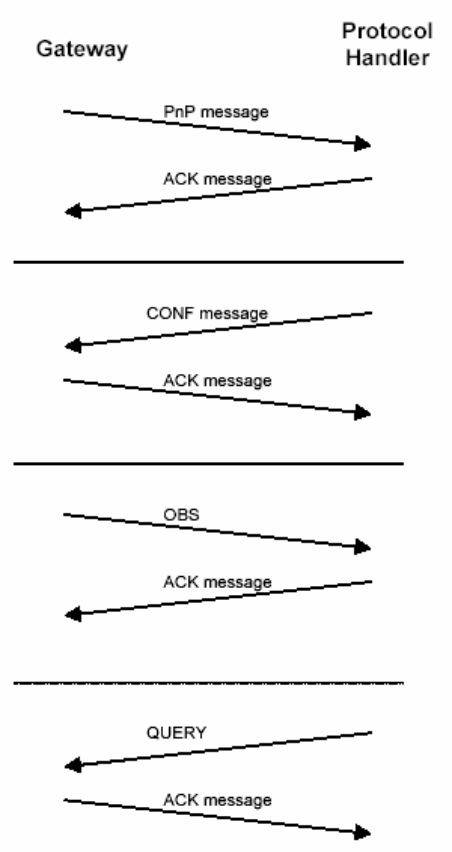


Figure 49

As specified in the following section, each message has a 16-bit identifier that allows to support multiple message exchange session simultaneously; the same identifier must be reported on the corresponding ACK message.
The custom protocol messages (over the TCP/IP stack protocol) exchanged with the Protocol Handler will be usually not translated or modified by the Gateway; anyway some fields could be analyzed to perform a formal check or to verify the type of the requested operation.
As exposed in the previous sections, an absolute time reference for exchanged data forwarded from the Gateway could be requested as a message field (Time Stamp); anyway, the implementation of a Real Time Clock section (or similar) in the Gateway will not be mandatory, in order to allow a simpler implementation in power-constrained environment: in this case, the Time Stamp field should be left blank by the Gateway, and will be properly filled by the Protocol Handler, considering not relevant the time delay for exchanging data between Gateway and Protocol Handler, with respect to the time accuracy requested.
The Gateway could also implement a message queue, to buffer ingoing and outgoing messages and to manage multiple message exchange sessions; the message queuing procedure should become useful with the increasing number of sensors controlled by a single Gateway.
Moreover, reducing data packet size and protocol overhead could be significant, where a wide-band communication channel is not available, and also where the connectivity costs depend on the size of exchanged data.

The protocols implemented over the TCP/IP stack for Gateway to Protocol Handler communication will be developed considering the specific subsystem requirement and functionalities.
Anyway, a minimum subset of commands should be implemented in every Gateway, in order to ensure a common data exchange basic implementation; a brief list of recommended commands to be implemented is here reported:

- **PING message**: a test message to verify the availability of a RDA entity; a PING_ ACK message is expected. It should be implemented for debugging scope.
- **PING_ACK message**: a response to a valid PING message; it must be sent from the destination entity reported in the PING message.
- **CONF message**: it is sent from any RDA entity to advise the high level AmI framework of its configuration settings; it could be sent asynchronously (as a P&P message) or in response to a CONF_REQ message.
- **CONF_REQ message**: it is sent from the AmI framework to request the configuration parameters of a RDA entity; a CONF message is expected in response.
- **CONF_SET message**: it is sent from the AmI framework to set user editable parameters of a RDA entity.

6.1.3 The AmI communication protocol stack

The communication protocol used in the AmI layer can be viewed as a three layer structure as shown in Figure 50.

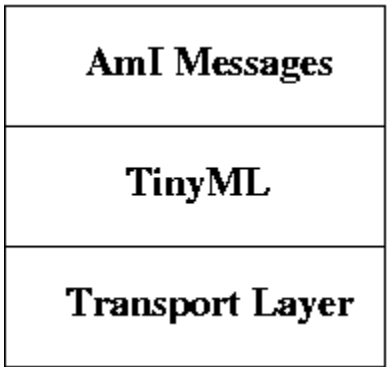


Figure 50

The undermost layer is an architecture specific standard transport protocol. The most suitable choice for this transport protocol is TCP because:

- It guarantees a connection-oriented, reliable communication pattern,

- its definition is based on open standard specifications (RFC),
- off-the-shelf implementations are freely available with open source distributions,
- decades of usage have proved that TCP is a very performing, flexible and stable solution for communications.

A further abstraction has been added in form of a second level layer. This layer offers a service that is responsible for serializing application specific objects. This layer has been introduced to decouple the definition of the messages used in the communication protocol from their content and to obtain a formalism that enables the programmer to automate the tedious task of transforming a message from its in-memory representation to its serialized representation.
The choice for the implementation of this protocol layer is tinyML.
As shown in the previous section tinyML is an XML based formalism that encodes some concepts related to Wireless Sensor Networks (WSN). It was proposed as a way to simplify OGC standards (SensorML and GML) to obtain a useful tool specialized for WSN.

The development team has adopted the tinyML specification as draft specification, a starting point for the development of a AmI/Goodfood adherent formalism that is inspired to the OGC standard proposals and that offers a good base for future improvements and generalization efforts.
The choice of using an XML based specification for message syntax enables the developing team to use automatic code generation tools that re-create the program code for containing message data automatically reducing the impact of modifications inside the topmost layer of the protocol to a simple compilation of the source code.
The objects modeled by the tinyML formalism are shown in the next section related to the Data Model, in form of UML class diagrams.
To show how the tinyML specification applies to the model of Figure 44 refer to Figure 51.

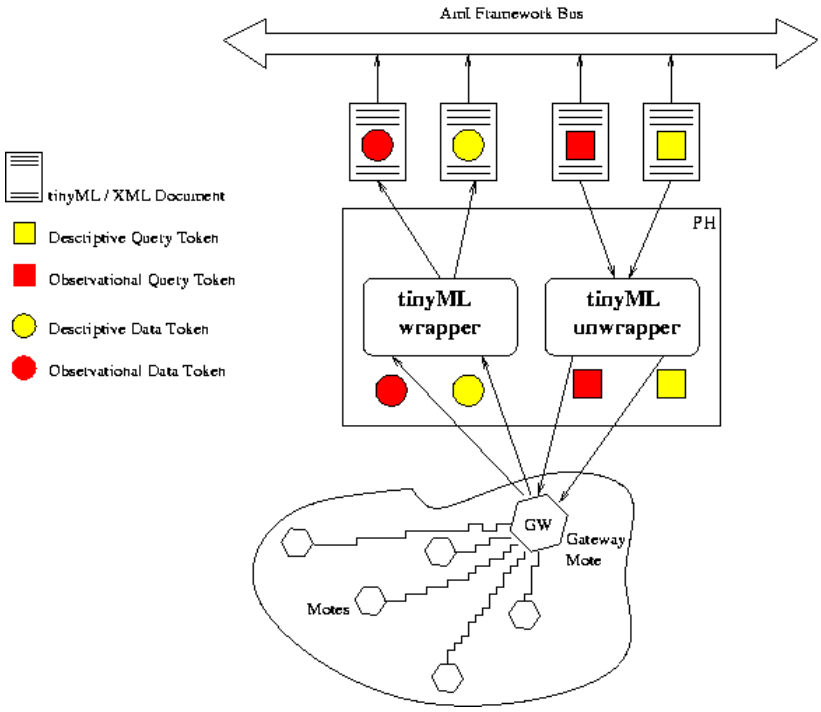


Figure 51: Applicability of tinyML

The top layer of the protocol architecture is defined through the concept of AmI Messages.
An AmI message is composed of two components: The Message Envelope (ENV) and the Message Content (CONT).

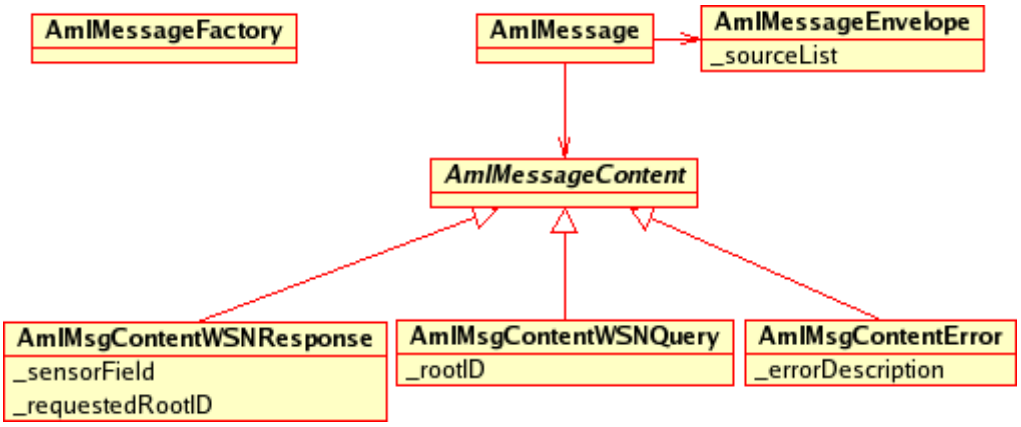


Figure 52: Example AmI Message classes

The ENV will contain information related to the message type, source and destination. The CONT will contain the actual information associated with the message, the payload. The payload is specific to a certain type of message.
Sender and Receiver of an AmI Message are responsible for parsing both ENV and CONT from their tinyML encoding to an in-memory DOM structure.

Figure 52 shows the UML class diagram of a minimal set of AmI Messages and their relation to the general architecture of the AmI Message layer implementation.
The shown concrete example messages represent the information to be sent for querying the structure of a Wireless Sensor Network (*AmIMsgContentWSNQuery*), for sending a response to such a query message (*AmIMsgContentWSNResponse*) and for communicating error conditions (*AmIMsgContentError*).
Figure 49 shows also a class called AmI Message Factory that is responsible for creating and destroying AmI Message objects.

6.2 The Data Model and Storage

The Data Model used inside the AmI framework is guided by the choice made for the communication protocol middle layer, because this layer offers a natural interface to the application specific objects that have to be managed.
Thus, the Data Model closely matches the model proposed by the tinyML draft specification.
In this section we show how AmI/Goodfood specific objects and relations are modelled using UML class diagrams.
The tinyML architecture is based on the concept of *sensor field*. A sensor field is a collection of possibly heterogeneous *sensors* and *actuators* which are attached to a set of *platforms*.
There exists also the concept of a *super sensor field* which is meant to be a way to interconnect different sensor fields.
The platforms are equipped with processing units and with a radio link for network connections.
The sensors are classified in *basic sensors* and *virtual sensors*. The basic sensors are actual sensing units that measure some physical phenomena. The virtual sensors are abstractions that operate on the basic sensors to compute a transformed observation. For example an aggregation of related observations or a wrapping to another unit of measure can be achieved through appropriate virtual sensors.
There are two kinds of virtual sensors. *Platform virtual sensors* are associated only with concrete basic sensors attached to a physical platform. *Field virtual sensors* are virtual sensors that aggregate groups of (or possibly all) homogeneous sensors inside a field to compute aggregate transformations such as average, min, max.
Both, basic and virtual sensors can be queried not only for retrieving their descriptive information but also for producing observational data.
The actuator are considered to be devices that can be instructed to operate in a sensor field following a certain program. The same distinction between basic actuators and virtual actuators applies as for the sensor devices.

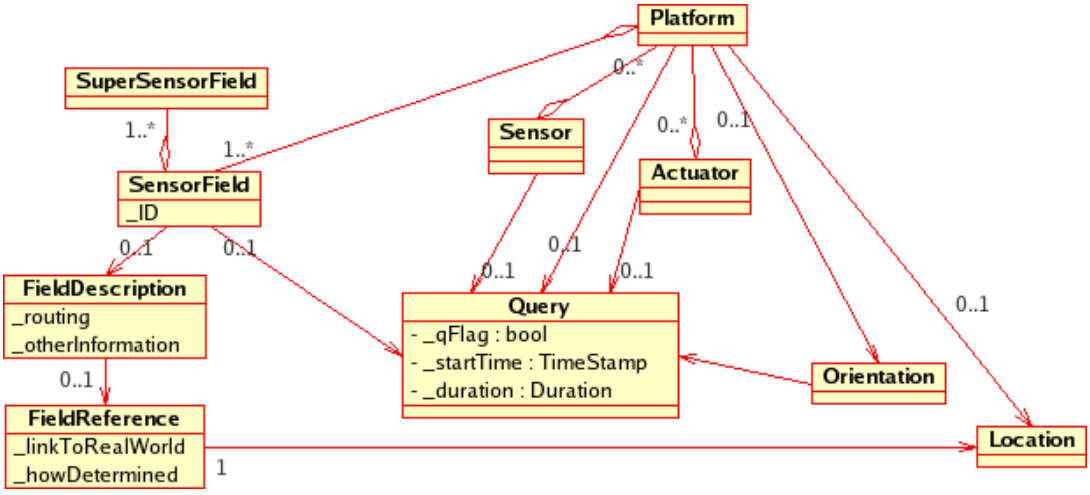


Figure 53: Rough Class Diagram of tinyML Sensor Field Model

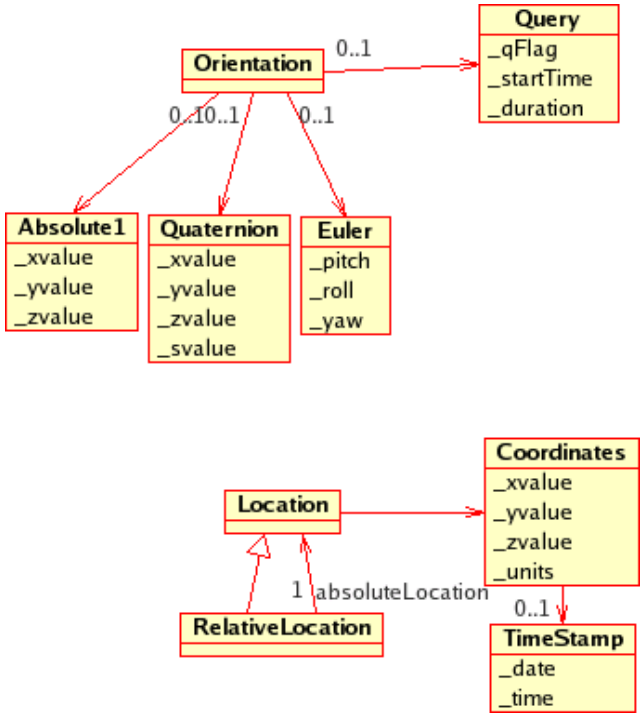


Figure 54: tinyML geometry objects

All the described entities carry information associated with positioning and orientation. Moreover there are detailed descriptive information associated with each entity and this information can be queried and modified by remote interactions. Figure 56 shows the UML class diagram for the tinyML draft specification for modelling the sensor field entity.
Figure 51 shows the tinyML proposal for modelling simple geometry operations like localization and orientation. Figure 53 shows a description of the tinyML Platform proposal.

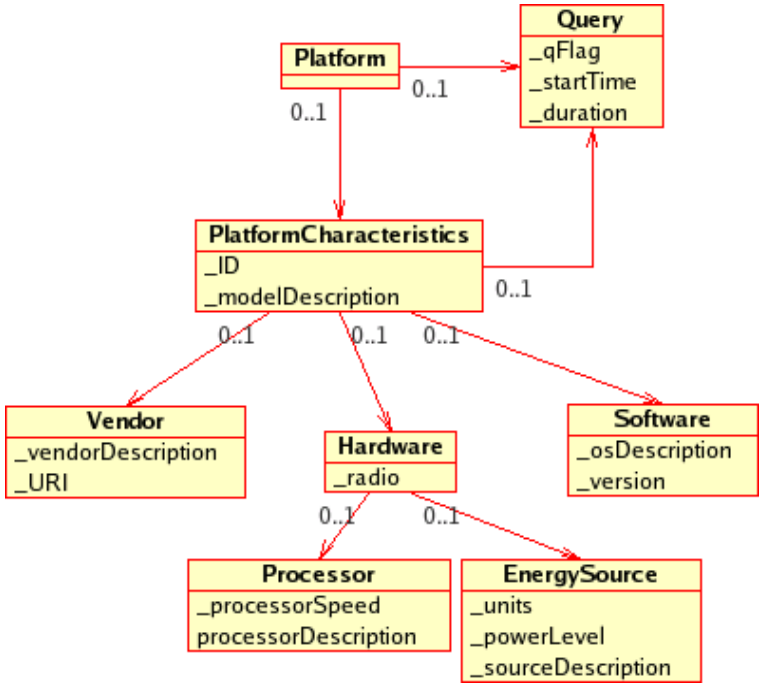


Figure 55: tinyML proposal for platform specification

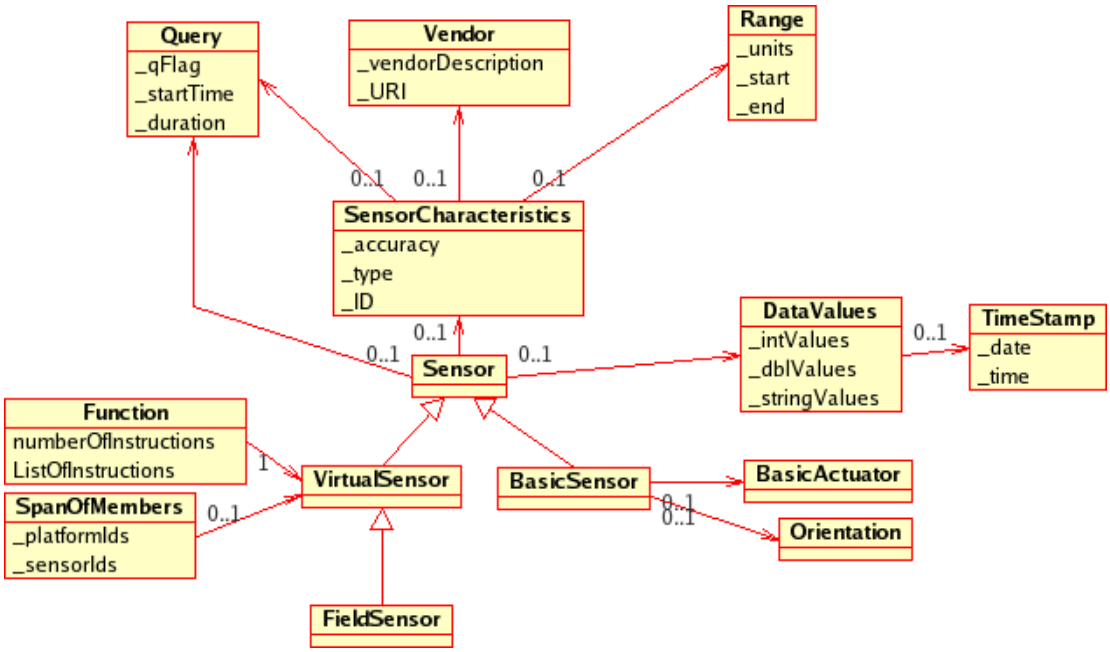


Figure 56 UML diagram for Sensor Model

As shown in Table 24, one wireless sensor network matches the tinyML concept of sensor field. The tinyML concept of Platform naturally matches the Motes of a sensor network.

Table 24: Relations between tinyML and GoodFood terminology

TinyML	Goodfood	Vineyard Pilot Site
Super Sensor Field	Food Chain	The Wine chain in Montepaldii
Sensor Field	The wireless sensor network at a chain step.	The vineyard, the cellar ...
Platform	Wireless sensor network node	MICA Motes
Sensor	Sensor	Leaf temperature, soil humidity, malic acid ...

The fact that all descriptive information associated with almost every component may be queried and modified, enables also the implementation of a plug&play mechanism useful to automatically update the information on the devices available in the sensor field also at the AmI framework level.
The actuator abstraction provided by tinyML enables also the future development of actuator interaction between the AmI framework and the sensor field.
Finally the generality of tinyML enables an easy adaptation of the software written for the AmI framework to a great variety of different sensor network requirements.

6.3 The Software Modules and Deployment

The high level architecture of the AmI framework is made up of a set of Software modules that are called ***AmIDevices***.
All AmIDevices are connected to a bus called ***AmIBus***. Communication among the AmIDevices is performed through writing and reading of ***AmIMessages***) on the AmIBus.
There are some AmIDevices that execute standard tasks like handling storage of the WSN or observational data inside databases or interfacing the high level AmI infrastructure with the WSN (i. e. The Protocol Handler PH).
These devices are called *Managers.*
They have to be executed at system start up and should stay on-line for the whole system life cycle.
A set of other AmIDevices, called *Clients*, is responsible for interfacing users in an interactive way to the AmI system.
Clients are mainly involved in the presentation of observed data and or status of the WSN, in querying the storage system for archived data for example to replay a monitoring session or for performing analysis of the data.
In Figure 55 a UML class diagram shows the relations among these classes of AmIDevices and a possible small set of concrete devices included in the AmI framework.

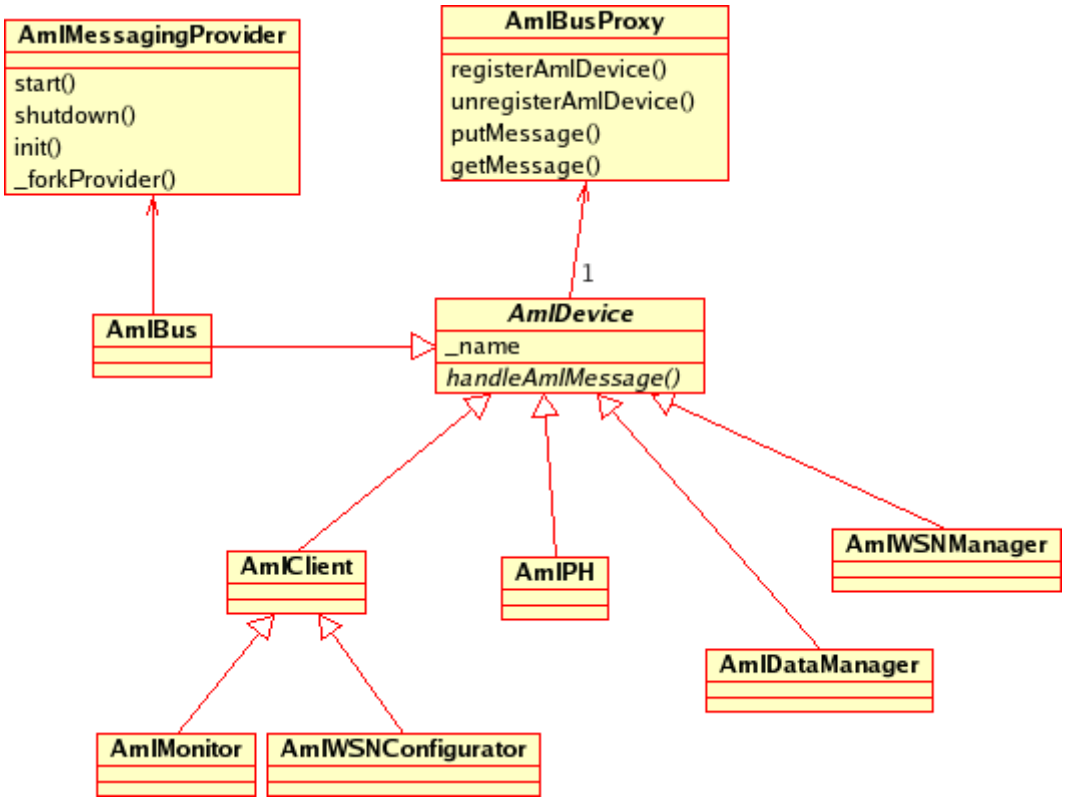


Figure 57: AmI UML class diagram with example MAnagers and clients

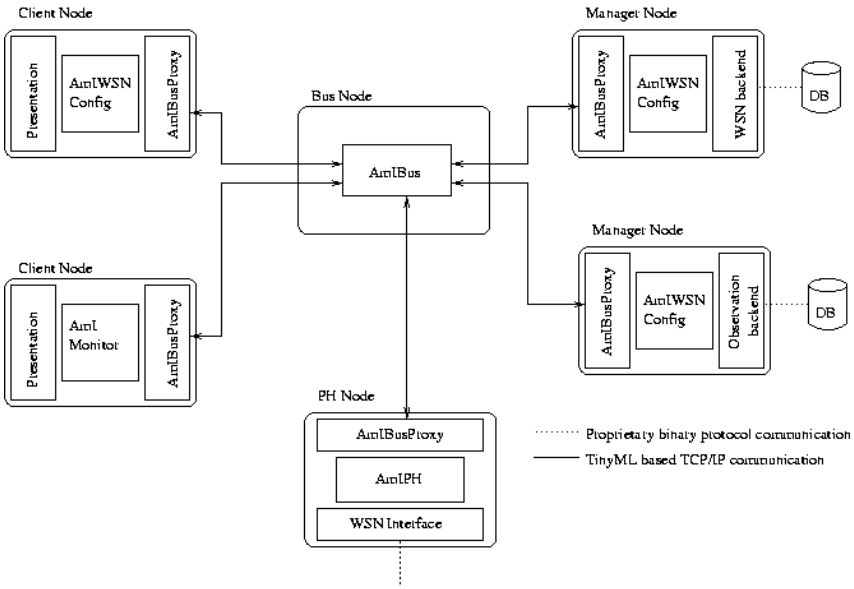


Figure 58

The chosen architecture, who decouples the system back-end from the front-end (user interfaces), enables a very flexible interaction between client and manager devices. Clients can always be started and put on-line by connecting to the

AmIBus. They simply access information by querying Managers or by registering themselves as consumer for messages of a specific topic on the AmIBus.

Figure 53 shows a possible instance of the AmI system architecture. Nodes in this figure represent processes. Each Node is composed of a set of software modules that perform node specific actions.
All Nodes include an ***AmIBusProxy*** interface module that links them to the AmIBus. This technique matches the well known OOP pattern called *Bridge*. Using a Bridge pattern enables a further level of independence from the underlying implementation details. In fact there is no need to change anything but the implementation of the *AmIBusProxy* module if the underlying communication technology (i. e. The implementation of the AmIBus Node) is modified.
Each Node has then a specific core software module that is responsible for implementing the tasks assigned to the module.
Finally each Node has another interface module that links them to their back-ends. Note that in this AmI architecture centric view, a Node back-end is the link between the Nodes and the system they manage.
The *WSN Interface* module in the PH is the interface to the wireless sensor network and in particular to the Gateway Node.
The Presentation interfaces in the client Nodes can be graphical user interfaces that show observational or resource management data to the end-user or intercept user interaction with graphical widgets for producing commands or query requests. But there could be also a software agent such as a Web Server that receives HTTP requests from remote users and queries the AmI system through an appropriate CGI-BIN technique to build up an HTML page containing a visualization of the requested data.
The back-ends of the Managers shown in Figure 56 are the links to the storage utilities for example JDBC.
Regarding the deployment of the various processes on actual hardware nodes the system architecture doesn't enforce particular restrictions. Many choices are possible and those have to be made a system setup analysing the various requirements related to robustness, performance, security and economical costs.

6.4 The Data Management: storage, transport, query and presentation

6.4.1 Transport

The transport of all sort of information among the AmIDevices is handled by a special AmIDevice called AmIBus. The information that has to be exchanged among the AmIDevices is wrapped in XML based documents respecting the tinyML syntax specification. A producer AmIDevice builds an AmIMessage object with the information it wants to put on the AmIBus. It then serialises the content of the message to a string representation in tinyML syntax. This textual representation is then written to the underlying communication infrastructure through the AmIBusProxy interface isolating the AmIDevice from the lower level technology details.



A particular role in this view is played by the PH AmIDevice. The PH is a process node that represents a gateway between the WSN and the AmI application level. It is responsible for wrapping into a tinyML compliant textual representation all information streaming from the WSN to the AmI application level devices. In particular:

- Observational data is forwarded to the AmIDevice responsible for controlling the data storage database and to possibly activated monitoring devices.
- Resource management information (status of a sensor or node, plug&play activity ...) is forwarded to the AmIDevice that manages the WSN and to WSN configuration/display client devices.

Finally the AmIBus device is also responsible for managing communication activity at the AmI Grid system level. Shutdown notification have to be produced for all connected AmIDevice objects or global error conditions have to be detected, broadcast and logged.

6.4.2 Data Storage

Storage regards both the resource management information on the WSN and the observational data. This data is stored in two different persistent databases. Each database is controlled by a specific AmIDevice. In particular the database storing resource management information is controlled by the *AmIWSNManager* whilst the database storing observational data is controlled by an AmIDevice called *AmIDataManager*.
The observational data is produced by the nodes inside the WSN and sent through a Gateway Node to the PH where it is wrapped and serialised according to the tinyML specification. The data is then sent by the PH to the AmIDataManager who tracks it by storing the values in the appropriate database resources.
The observational data can then be queried by client AmIDevices from the AmIDataManager through appropriate query and response AmIMessages.
Moreover the AmIDataManager contains also the logic to move the data from a real-time database to a historical database according to some (possibly configurable) policies.
These policies take several aspects into account. Some of these aspects are:

- **Timing.** Some rules have to be established for defining when a certain type of data has to be shifted from real-time storage to historical storage.
- **Structural modification**. The difference between the two kind of storage systems in terms of structure have to be taken into account.
- **Possible functional aggregation**. User needs could specify some functional aggregation (sum, average,min, max) of homogeneous data for saving valuable storage space.

- **Time aggregation**. For some data it might be useful to store only a subset of observed values (for example only one per day or week) since this could save a lot of storage space.

6.4.3 Resource Management Storage

Resource management data can be provided in two ways. One way is to have a WSN entity (sensor, actuator, mote ...), who plugs into the WSN, to communicate information on its functionality, positioning, orientation, energy resources and so on, through a low level protocol to the PH node. PH wraps this information into tinyML compliant messages that are then sent to the interested AmIDevices. This mode of operating can be viewed as a sort of plug&play mechanism.
Another way is to have a client AmIDevice called AmIWSNConfigurator interacting with a GUI front-end with a human operator that supplies the resource management information related to a newly connected WSN entity.
There could also be a mixed mode of operating where the WSN entity sends a subset of the required information and a client AmIDevice, which will be notified of the new presence, prompts the human operator for additional information.
In either case the sensors, motes or actuators have to continuously update their status by supplying information at system run-time.
This resource management data is sent to the AmIWNManager device who is responsible for storing it according to a specific WSN describing database schema that closely matches the tinyML formalism.
The resource management data can then be queried by client AmIDevices through appropriate query and response AmIMessages.
Every change in the WSN configuration has to be propagated to all client devices that in some way are interested in keeping track of the information regarding nodes, sensors and actuators in relation to their functional characteristics, activity, status, energy resources, positioning or orientation.
There are several ways of performing this task. The AmIWSNManager could broadcast on the AmIBus an update notification message. Every client that is interested in the WSN status could, once it receives the notification, query the manager for the new status. Another way is to make the manager broadcast directly the new status (either as a whole or by differences) on the AmIBus and all interested devices could directly update their knowledge with this message.

6.4.4 Querying

The observation activity of a sensor is defined by a Query object that is associated to it. As specified by the tinyML proposal a Query object has a duration attribute that establishes how long the sensor should keep responding to it by acquiring data and sending it to the Gateway mote. In effect this information is handled by the Platform (Mote) to which the sensor is attached because the Platform is actually the processing enabled entity in the WSN.
The query defining a sensor activity can be fed into the WSN by a human interaction AmIDevice at configuration time or it can be imposed by the sensor specification at connection time (plug&play). In either case it is necessary to

store this information along with the other data regarding a sensor into the WSN database.
For each virtual sensor it is also possible to define a sequence of instructions that build up the virtual observation from a set of concrete observations and the set

of sensors who contribute to it. This information, too, has to be stored in the WSN database.

6.4.5 Presentation

Presentation is performed by client AmIDevices. There are several purposes for presenting data:

- **Real-time monitoring:** each measure produced for a given observation is displayed on a graphical user interface when it is produced.
- **Play back of monitoring sessions**: complete monitoring session can be played back to find patterns in sensor behaviour.
- **Analysis**: data can be shown as a result of data-mining/knowledge discovery.

All these aspects need some presentation using user defined graphical interfaces. As an example of a graphical user interface Figure 59 shows a session of a real-time monitoring client AmIDevice.
The total display area is split in three panels. The left panel is used to locate the observed sensor, the right panel to describe the sensor characteristics (higher) and to display the monitored values (lower).
In the left panel of the screen there is a visual display of the sensor location.

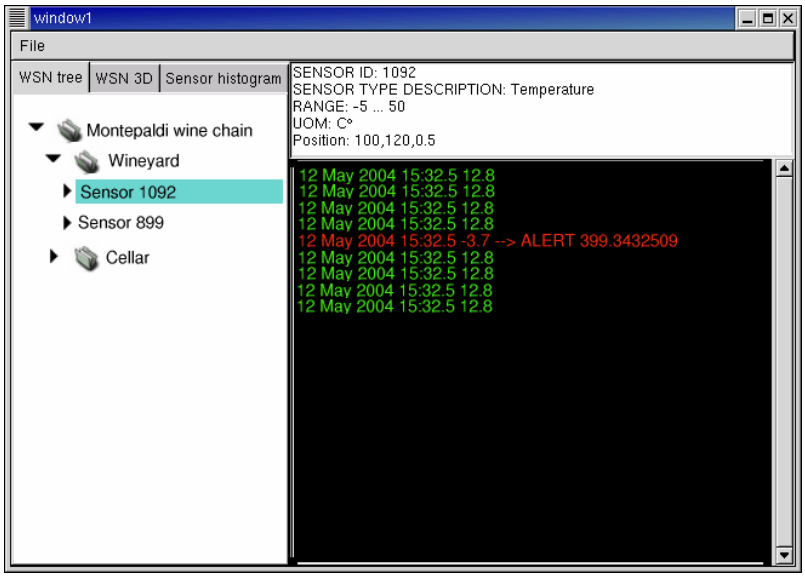


Figure 59: An example of graphical Monitoring client

Different choices are possible for that display ranging from a tree view (as shown in figure) to a 3D rendered view where the sensor is shown as a point in a 3D representation of the sensor field. Moreover, in this panel, some graphical representation (histograms, coloured particles, wind fans) of the measured observation values can be associated with the sensor.
such as a different colour for suspicious measures and operator to alert.
In the right side of the display the upper panel shows the complete description of the sensor that is being monitored. The lower panel instead, shows the values produced by the observational measures with, possibly, extra information

7 Design of software interaction interfaces

7.1 Human-Machine interface

At the higher levels of the AmI architecture user interfaces are required to enable the human operators to interact with the Remote Data Acquisition (RDA) level both in terms of resource and observational data management. Observational data is presented to the user during two specific activities: Monitoring and Analysis.
Monitoring is a mainly automatic activity that is used to control some attributes that influence the life of the specific food chain.
The *Analysis* activity is a computer aided Decision Support System designed as a management tool to assist the domain experts in the different food-chains to achieve their target levels of efficiency, yield and fruit quality, while reducing the risk of crop loss and minimising the use of chemicals and other inputs. In order to manage these targets, row data measured during the entire year, aggregation and correlation between parameters are necessary.
In this section we present a description of the Graphical User Interfaces (GUI) starting from a generalization effort based on the user requirements for a specific pilot site (Montepaldi Vineyard).

7.1.1 Common Features

Commonly the screen of a client *AmIDevice* is subdivided into displays. The number and type of displays can vary with the different types of clients. This organization is established by the programmer who designs the user interfaces for specific applications.
In any case, a graphical representation of the RDA structure based on the *tinyML* description made in section 3 is required.
This display is called ***RDA Display***.
The left display of the example interface in Figure 60 shows a representation of the RDA. For this information there are two suggested representations. One is based on a Tree widget (Figure 61) who shows a schematic view of the RDA based on its hierarchical relations.
Each element of the tree is selectable and multiple selection are possible. The selection of an element of the RDA Tree fires update events in the other displays that show related information.

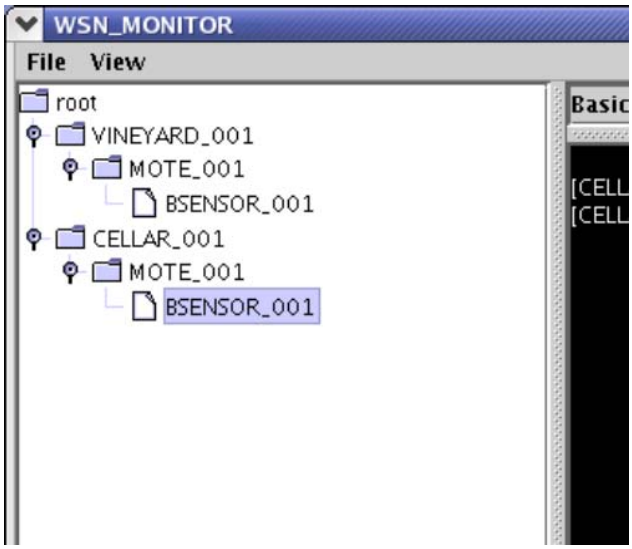


Figure 60: Example of a Tree representation of the RDA Display

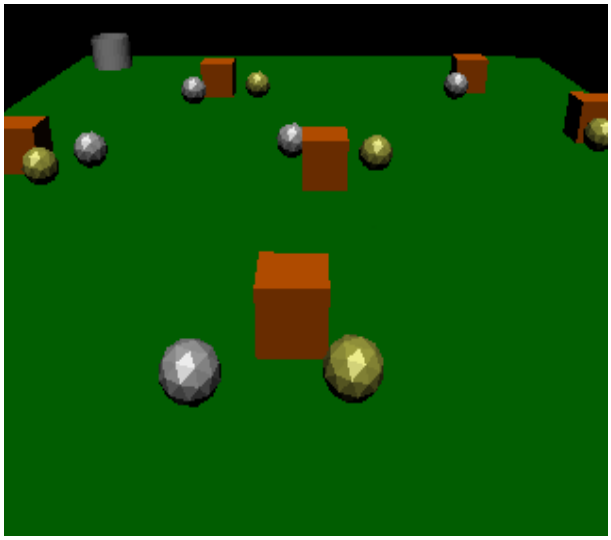


Figure 61: Virtual 3d view of the RDA(cubes = platforms; spheres = sensors; cylinder = GW)

A second representation is based on a schematic 3D view of the RDA (Figure 56) in which entities of the RDA are shown as graphical objects. In this case the graphical objects are selectable with same policies of the Tree representation but are in some way bound to a geographic location that is mapped in the virtual representation.
Another standard display is the Description Display that shows all information associated with a specific entity inside the RDA. This display will be based on simple textual representation of the descriptive information of the entity. An example is shown in the upper right display of the example interface of Figure 59

7.1.2 Observational Data Monitoring

In the case of Monitoring activity a default user interface is produced to enable an operator to observe real-time data that streams from the RDA to the higher level systems. For these activity an example of suggested interface is shown in Figure 62.
In this example the left display is an RDA Display. The right top display is a Description Display.
The right bottom display is the so called Monitor Display. In this display all the observational data produced by the entities selected in the RDA Display is shown along with the identifiers of the entities and the timestamps associated with the observations.
Like the RDA Display the Monitor Display has a 3D virtual representation of the monitored area. In this representation observational data can be shown as graphical effects on the virtual map (for example wind fans, isothermal curves, graphical particles and so on).
The graphical effect to be displayed should be associated with the entity descriptor through its type attribute.

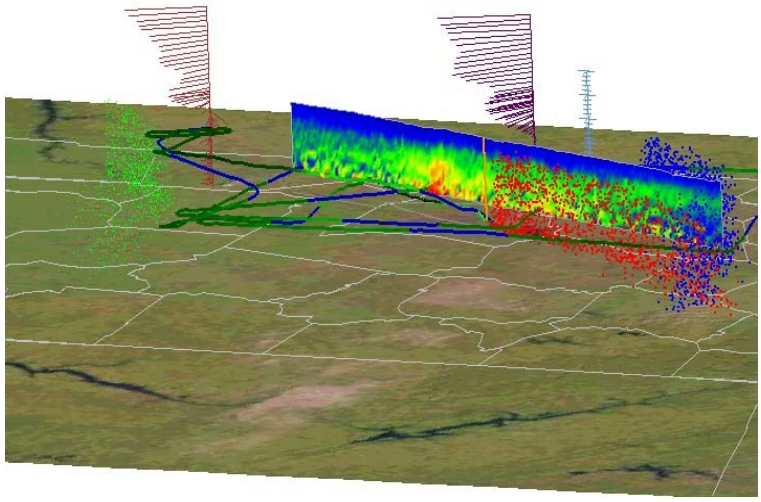


Figure 62: Example of monitored values represented in a 3d view.

The Monitoring Display should transparently manage basic and virtual sensors. Virtual Sensors are described in the previous section and are responsible for:

- creating aggregate observations from basic sensors.
- create functional approximations of non existing sensors.
- automatically monitoring of correlations among different (possibly heterogeneous) sensors.

The creation/configuration of virtual sensors are a resource management specific task that is described in the next section.
The following tables show all the observations related to the pilot site. The columns labelled with an associative operator show the aggregation functions that have to be monitored on the given observation.
Figure 60 shows aggregation functions that work in the time domain (i. e. Aggregations that work on a single sensor observations).
Also spatial data aggregations provide useful insights for the right "micromanagement".
Spatial data aggregation is obtained by combining through associative operators data produced by different but homogeneous sensors. Table 27 shows the aggregations that are required to be monitored in the vineyard of the Montepaldi pilot site.
Other observation that are required to be monitored are reported in Table 26. For these observation the development team has not yet identified basic sensors that are able to measure them.
In this case a suitable solution is to provide virtual sensors that can measure these observations by combining related observations.
In Table 26 the columns state what kind of basic sensors have to be combined in order to analyse the pathogen related observation inside the pilot site.

Table 24: The observations done in the pilot site with the required aggregations in the time domain.

	average	**min**	**max**	**sum**	**variance**
Soil analysis					
Soil temperature (20 and 50 cm depth)	x	x	x		
Soil moisture (20 and 50 cm depth)	x	x	x		
Air and meteo parameters					
Air measurement					
Air Temperature	x	x	x	x	x
Air humidity	x	x	x	x	x
Wind direction and speed					
Rainfall	x	x	x	x	x
Plant analysis					
Differential leaf temperature	x	x	x	x	x
Xylem flux	x	x	x	x	x
Diametric stem growth	x	x	x	x	x
Cellar monitoring					
Ethylic Alcohol					
Glucose					
Loose SO_2					
Total SO_2					
Acetic Acid					
Malic Acid					
Cellar Temperature	x	x	x		
Cellar Humidity	x	x	x		

Table 25: Spatial aggregations on the sensors of the vineyard.

	average	min	max	sum	variance
Soil analysis					
Soil temperature (20 and 50 cm depth)	x	x	x		
Soil moisture (20 and 50 cm depth)	x	x	x		
Air and meteo parameters					
Air Temperature	x	x	x	x	x
Air humidity	x	x	x	x	x
Wind direction and speed					
Rainfall	x	x	x	x	x
Plant analysis					
Differential leaf temperature	x	x	x	x	x
Xylem flux	x	x	x	x	x
Diametric stem growth	x	x	x	x	x

Table 26: Virtual sensors for approximating non-existing sensors

Virtual sensor	temperature	air relative humidity	rain fall	wind speed
Downy mildew	x	x	x	
Powdery mildew	x	x		
Grey mould	x	x		
Evapotranspiration	x			x

As a final task of the Monitoring Display an Alarm Generation System should be considered.
Each descriptor of a RDA entity carries information on what are considered to be suspicious measurements, upper and lower bound threshold values, exact procedures to be executed or operators to be alerted in case of alarm occurrence.
For example in the pilot site dangerous situations such as pathogen attack and drought stress in the vineyard; stuck of fermentation and suspicious high level of temperature, SO2 and acetic acid during aging have to be monitored.

7.1.3 Resource Management GUIs

The GUIs that will be provided for resource management are based on the same displays described in section 4.1.1. There will be an RDA Display that shows all the RDA entities.
By selecting a proper entity from the RDA Display the human operator will be prompted with a Description Display on the right top display of the GUI and with a Configuration Display in the right bottom display of the GUI.
The Configuration Display will be a form that can be filled in with the information an entity does not provide by itself (P&P).
Another particular use of the Configuration Display is related with the programming of virtual sensors.

For each virtual sensor there must be a field that collects information on the participating basic sensors and a field for inputing the code of the function that has to be executed to combine the participant sensors.

7.1.4 Analysis GUI

The analysis GUIs will have to be tailored to the specific algorithms that are applied to the stored data in order to obtain meaningful datasets and parameters.

In particular there should be some forms for submitting queries that will be forwarded to data manager *AmIDevice* and a Query Result Display that shows the queried data in form of simple textual report or as for the Monitoring Display in form of graphical 3D representations.

The same patterns used for aggregating data in the monitoring software has to be accessible also in the analysis patterns. In particular virtual sensor should continue to exist.

7.2 Interface Usability

Interface Usability will be measured by the users of the AmI software of the Montepaldi Vineyard pilot site on a demo prototype implementation of the architecture specified in sections 3 and 4 of this document.

The measurement of the quality will be performed as described in the Test Protocol Section of the D11 document.