University Politehnica of Bucharest

Pi-CoPS

Pipe Contamination Prevention System

Windows Embedded Student ChallengE

Final Report

Team Members

Ştefan Bucur
stefan.bucur@gmail.com

Răzvan Tătăroiu
razvan784@gmail.com

Mihail-Alexandru Balan
mihai.balan@gmail.com

Vlad Ureche
vlad.ureche@gmail.com

Team Mentor

Professor Nicolae Ţăpuş Ph.D.
ntapus@cs.pub.ro

1 Abstract

We live on a planet that is 75% covered with water and it would be apparently difficult to imagine that the water shortage could be a problem. However, one must take into consideration that less than 1% of Earth's water is drinkable, and unfortunately the pollution of water sources has become a serious concern–over 20000 waterbodies across the U.S. alone have reported problems [3]. Organizations around the world are continuously trying to find improvements to the existing water treatment and distribution systems. Millions of dollars are invested into research and environmental education programs, and the recent technological innovations are opening more ways to address this public threat.

Statistics [5] show, for example, that around 50 million Americans drink from water supplies contaminated in unsafe concentrations with toxins. Even in the most advanced cities of the world, deteriorating, out-of-date plumbing are sometimes causing the water to become unhealthy to drink. In the recent geo-political context, we should not even exclude a potential terrorist action which, by contaminating the water distribution system of a major metropolis, would lead to an unprecedented disaster.

In this context, *Pi-CoPS* (Pipe Contamination Prevention System) is committed to bringing a higher degree of security in every community that is concerned with the safety of its tap drinking water, whose quality is vulnerable to accidental infiltrations or even malicious actions.

Pi-CoPS is an automated system that relies on a multi-level sensor network connected to the main water pipes in an urban area. As the sensors are acquiring data regarding various water quality parameters, the system processes it in real time, and takes action to prevent the spreading of contaminants to the population, while opening alternate paths to supply clean water, if possible. Pi-CoPS can also provide authorities and the public with a valuable selection of information, while maintaining fast and efficient network communication.

The entire network was designed with three main features in mind: *fault tolerance*, *responsiveness and scalability*. Pi-CoPS' *innovative asset* is its possibility to prevent the contamination spreading by blocking or rerouting the water flow. Even in the case of the data network failure, the devices continue to operate and provide protection, without any remote intervention.

Data is continuously processed by devices attached to sensors, in order to achieve a fast local response in case of emergency. They can take measures automatically if programmed to do so or the network is down, or they can forward the information and receive commands in order to provide an efficient overall reaction. During emergency, the intervention teams are announced and supplied with the exact location of the source of pollution, so they could fix the problem and put back the water distribution in normal state.

The system can be easily extended as the distribution network grows or the security requirements increase. The sensor number and placement is decided by the pipe infrastructure topology and the degree of safety the water company is planning to offer. In this context, Pi-CoPS provides assistance through a management application in projecting and testing device placements, and also monitor the real-time information received from the sensors.

2 System Overview

2.1 Pi-CoPS at a Glance

Pi-CoPS is designed to integrate with the existing water distribution infrastructure, providing automatic intervention in case of contamination, additional support for data acquisition, processing, real-time intervention, and data collection and storage (see Figure 1).

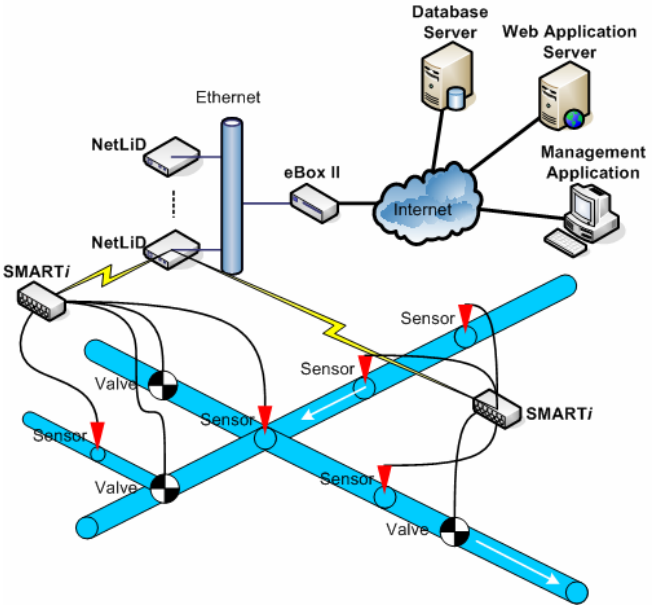

Figure 1 System Overview

The information acquired by the *sensors* attached to the pipes is processed by *Sensor Monitoring and Automatic Real Time Intervention* devices (SMART*i*). They acquire the data, process it, and get prepared to take measures in case of emergency by closing water valves. If the network is down or they were programmed to do so, SMART*i*-s will take all the measures automatically. This is the case for most situations that involve simple network topologies. However, in a more complex situation, SMART*i*-s can be programmed to take action only after receiving explicit commands from the upper layer devices. Paragraph 3.1.3 describe in detail the high level issues involved in the decision process.

The data acquired by SMART*i-s* is forwarded to *Network Linking Devices* (NetLiDs). A NetLiD collects data from multiple SMART*i*-s in one area (up to 255), and forwards it to the *eBox II*. The eBox can synthesize the information received from the entire network and takes decisions regarding water flow control and sends them back to the SMART*i*-s for execution. The eBox stores relevant statistics in a *SQL database* hosted on a dedicated server, and provides all the information requested by the *management application* and the public access ASP.NET *web site* and *web service*.

2.2 Commercial Feasibility and Potential Customers

There is a high demand in the field of environmental protection for more advanced solutions to water quality preservation. Even in the most advanced cities of the world, pollution and deteriorating, out-of-date plumbing are sometimes causing the water to become unhealthy to drink. Giving just a few examples taken from an official study [2], one can enumerate Atlanta, with a poor distribution system, Fresno which has no real water source protection, and some others just from the United States.

Pi-CoPS offers a large scale solution guided by the principle that clean water should be a certitude for *everyone*. Thus, The Pi-CoPS system targets implementation by large water distribution companies that already maintain an extensive pipe infrastructure. Implementation costs are proportional to the detection and intervention capabilities of the deployed system, and

should be viewed as a reasonable and necessary investment by any community interested in bringing a high level of safety regarding the quality of its drinking water, whether it can be jeopardized by accidents or malicious actions.

2.3 Innovative Approaches and System Features

Pi-CoPS' objective is to *eliminate the risk of contamination spreading* in the water distribution system of a city or any other community. This critical task is accomplished in an *original approach* by providing **automatic intervention** in case pollutants are detected, even in case of communication or hardware failure. The **distributed system** characteristics of the Pi-CoPS network allow every device to continue functioning based on its current assignments and fail-safe plan, even in case of communication line or higher-ranking device failure.

This feature is supported by Pi-CoPS along with other services and functions that permit the water company to control almost every aspect of the system:

- **Real-time monitoring of the water** flowing through the pipes of the distribution system by means of a large number of sensors placed in strategic locations.
- **Reliable data acquisition and storage** is possible by implementing a scalable and fault tolerant data network featuring intelligent hubs (of which the most capable are implemented with eBoxes), a storage server, and remote administration and monitoring points, where engineers can examine in detail various aspects of the entire system.
- **Statistics and on-demand real-time information from sensors** are carried out by the front-end services located on the servers: a web application that offers basic statistical and real time information over the Internet, and a specialized desktop application that, apart from providing advanced monitoring and visualization, is helping the water company engineers to efficiently design and implement the Pi-CoPS sensor network over the existing water pipes infrastructure.

2.4 Design Methodologies and Team Collaboration

A spiral methodology was used for our system development, as it benefited from its rapid prototyping, and parallelism in design and build activities. In the initial phase, the project requirements were established and the main components of the system were delimited. After an initial functional prototype was obtained, the development life-cycle of each component consisted of series of design adjustment, implementation and testing in the new conditions. During every iteration, new functionality and more testing was added to the Pi-CoPS system.

The team kept connected through an on-line collaborative document editing application (Wiki). All document drafts were edited there and all ideas and options were discussed. All specifications for system components, interfaces and protocols are also stored there. A SubVersion source control system is used to share and edit the source code for all software modules. A Gantt chart was used to schedule and synchronize each task and delimit the work each member had to carry out. In addition, weekly meetings took place in order to evaluate the progress of each member.

3 Implementation and Engineering Considerations

3.1 System Architecture

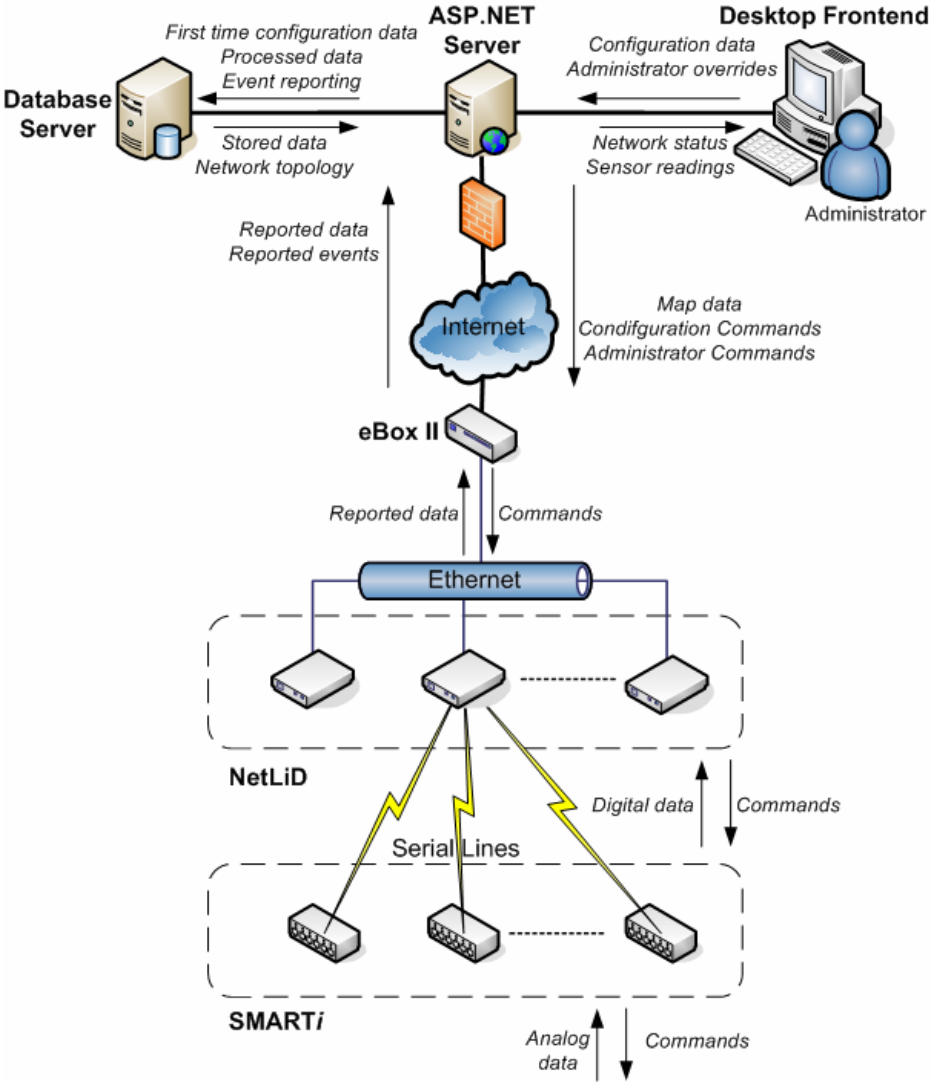

Figure 2 System Architecture

The next sections detail the design and functionality of each component in the system. The diagram depicted in Figure 2 presents an overview of the way each component interacts with the others.

Pi-CoPS is mainly a *distributed system*, as its processing power and decision-taking responsibility is spread across all of its devices, so that it is not a single-point-of-failure structure. It has however a clear hierarchical structure that tends to concentrate around a central node. Data flow towards this node is not possible when the network is down, but the *critical task of protecting the public* by shutting down the water flow in case of pollution is not affected in a significant way.

3.1.1 Sensor Monitoring and Automatic Real Time Intervention devices (SMART*i*-s)

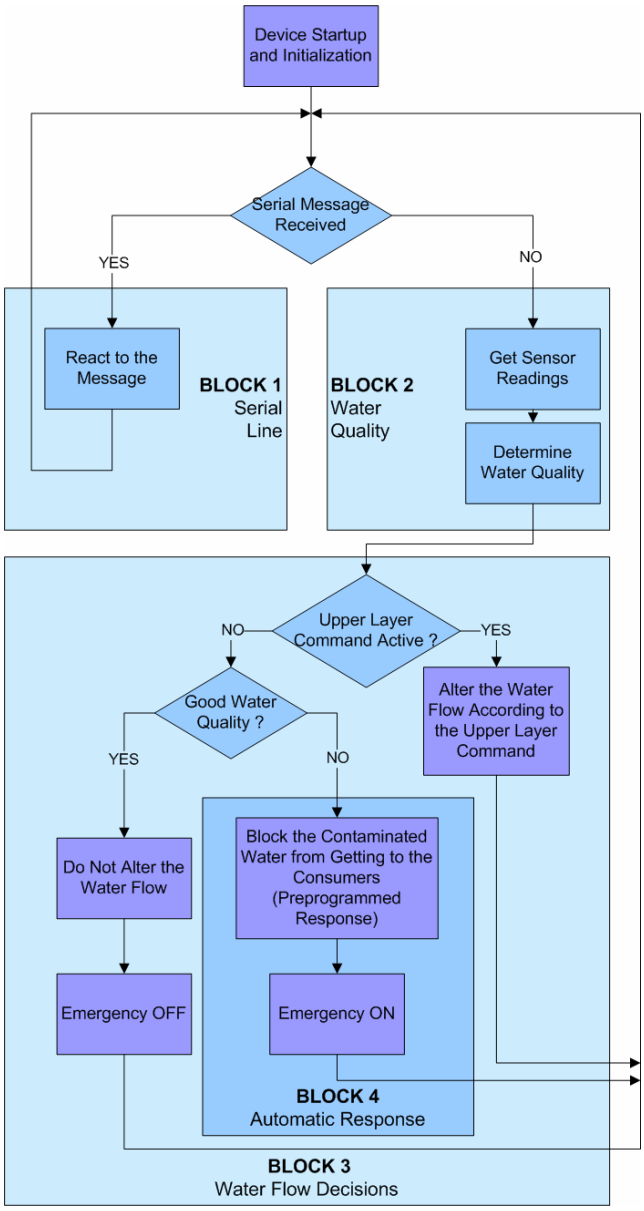

Figure 3 SMART*i* Workflow Diagram

SMART*i*-s are the base of the hierarchy, being the simplest and the most numerous modules. They perform three basic operations:

- Monitoring water quality using sensors
- Communicating the sensor readings to upper-level devices (NetLiD-s)
- Driving devices that alter the water flow: valves, automated taps

Figure 3 depicts a simplified diagram of the general SMART*i* workflow process. Communication with the upper layer is made through a serial line, over which SMART*i*-s receive commands and responds with data or status. (Block 1) During a SMART*i* configuration process, every sensor is assigned a range of readings that correspond to accepted water parameters. When the sensor readings are sampled, the SMART*i* analyses the results and determines whether the water parameters fit the accepted range of readings. (Block 2) If not, an emergency situation is reached.

There are two actions that are being taken when the SMART*i* is in an emergency situation: (Block 3)

- Automatic Response (Block 4) is the preprogrammed action to be taken when an emergency situation occurs, and is not situation-specific but general. **When an emergency occurs the most important fact is to keep consumers safe, therefore over-reacting is normally better than not reacting at all**. It should be noted that this action is a preprogrammed response that occurs a few milliseconds after the detection of the pollutants, and will be suppressed in at most 4 seconds by a situation and topology specific command issued by the eBox.
- Upper layer command issued by the eBox after applying the water routing algorithm to the situation in the field.

The **sensors** mounted on the water pipes are chosen according to the type of contaminants that need to be supervised. For the purpose of testing, some simple, easy to manufacture sensors were considered.

The *water flow speed sensor* is a small encased rotating paddle wheel connected to a magnet. It outputs electrical pulses with a rate proportional to the speed of the water through the pipe. The SMART*i* has a special Counter input that can measure the frequency directly.

The *conductivity sensor* can detect many contaminants. Pure water is a very good electrical insulator, but this is hardly the case in practice. Whenever excess salts are present in the water, its conductivity rises and that is be easily detected with a sensor made from two inert electrodes and an amplifier.

The *transparency sensor* only detects severe pollution but is nevertheless useful, especially as a demonstrator. It is also very simple as it only consists of a light emitter and a light detector enclosed in a black box.

The *laser sensor* also uses light, but it measures how much it scatters rather than how much of it is absorbed. It's much more sensitive and complex. It is only partly complete due to unavailability of some components.

3.1.2 Network Linking Devices (NetLiD-s)

NetLiD-s collect data from SMART*i*-s and process it statistically. All SMART*i*-s are polled once every second. After being processed, data from the SMART*i*-s is sent to the higher level once every minute in case no pollution is detected.

NetLiD-s also keep a map of the pipes in the area they service. When one or more SMART*i*-s detect pollution, the NetLiD will possibly override decisions taken by the SMART*i*-s in an attempt to offer increased protection or provide an alternate supply route for clean water.

A company employee can request a real-time view of the readings on a sensor. Since sending all data from all sensors managed by NetLiD-s would quickly overload the network, an elegant solution was employed: on request, the NetLiD can forward data as it receives it from specific SMART*i*-s once a second.

3.1.3 The eBox II

The main processing unit of the Pi-CoPS system is represented by the eBox. The eBox runs a custom built Microsoft Windows CE image. They provide the second level of data processing and decision taking, as well as the primary level on which communication with the central servers occurs.

The operating system image was customized in order to get the smallest footprint and preserve the computing power to the tasks related to the functioning of the Pi-CoPS system. The image was built from scratch, and included support for wired internet connectivity, .NET Compact Framework 2.0, remote application installation, various network tracking utilities as well as the custom management solution.

The functions of the eBox are as follows:

- During normal operation (no special events – i.e. no pollution occurred) it packs and send synthetic data to the Global Storage Server (see section 3.1.4) over a custom designed Web Service.
- In case a contamination occurs and is reported to the eBox, it takes measures to limit and/or reroute pollution to the evacuation system, by examining the distribution and Pi-Cops network topology. After issuing the necessary commands to the SMART*i*-s, it also monitors the evolution of the sent commands and alerts the central server if malfunction is detected (device timeouts, actuators unresponsiveness, etc.)

Water Routing

In case pollution is detected, the network map is examined and the shortest path to a "sink point" is determined. A sink point can be a point where water naturally leaves the distribution system or a (large) consumer where water quality is rather unimportant (i.e. a thermal point).This path is determined by taking into consideration the length of the pipes the polluted water passes through and the number of ramifications that pipes have (a longer path but with fewer ramifications is preferred – as fewer valves need to be closed). On the other hand, the algorithm can take into consideration the option of totally isolating the polluted area if the distance to any sink point is too big (the specific distance can be set when initially deploying the system). This is tuned on the initial setup of the system.

In this latter case, commands for routing the polluted water to a specific point can still be issued centrally, from an administrator, and the system will carry out the commands. Also, based on readings from the sensors the eBox determines the location of the pollution source. The accuracy of the location is strictly related to the number and the precision of the sensors on the considered network segment.

The Necessity of the eBox

The eBox has been chosen to be used instead of a PC or a more lightweight device due to the following reasons:

- The eBox was designed to operate under industrial conditions (a wide temperature range: -20°C … +60 °C, and very permissive humidity conditions) and is less susceptible to hardware failure than a PC, which has more components in its structure.
- By optimizing data transfers (reporting only the information that is actually necessary) the eBox can easily handle a network of 256 fully-loaded NetLiD-s at a rate of one pollution event per second. We also simulated larger events that were "detected" by a significant number of virtual sensors.
- Its dimensions make it suitable for being placed in inaccessible spots (on telegraph posts, or underground, etc.).
- An authorized engineer could directly connect to it and carry out configurations through a friendly user interface.

The eBox was thus found to be ideally suited to function as the manager of the whole PiCoPS network.

3.1.4 Global Storage Server

The data collected through the sensors infrastructure and processed by the NetLiD-s and the eBox is stored in a central data base server (which should be typically a dedicated machine) running SQL Server 2005.

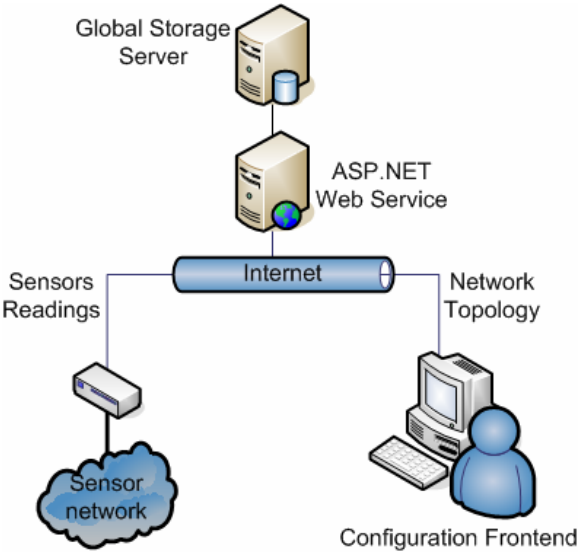

Figure 4 Position of the Global Storage Server in the System

This server is used as the main data source for both the network-pertinent data (distribution network topology and sensors types and locations), and network acquired data (sensors readings, system events). Although they belong logically to the same database, these two levels can be physically separated at the DBA level, in order to provide safety for sensitive data and increased performance for highly accessed tables.

This database is accessed by both the internal Pi-CoPS network and the company's administrative staff and customers through ASP.NET web services which provide the means of transparently authenticating the users and exchanging data.

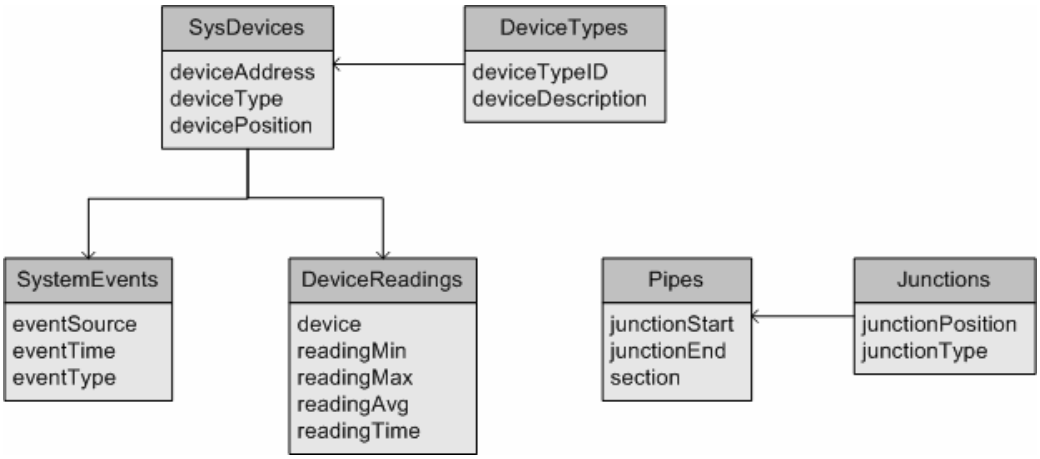

Figure 5 General Database Structure

During normal operation, data is regularly uploaded to the server and stored in the *DeviceReadings* table. The update interval is chosen so that relevant data is correctly stored and that no overflow occurs (this is typically 1 minute). For each reading the unique identifier of the source is stored, as well as the minimum, maximum and average values read in that given time span. In case a special event occurs (contamination detected, unresponsive device) records are inserted in the *SystemEvents* table, that track the evolution of that special event (event reported, actions taken, progress of actions, further sent alerts).

3.1.5 Desktop Front-end

The desktop front-end is represented by a rich Windows Forms management application that provides to the administration the necessary tools required to control every aspect of the Pi-CoPS system:

- A systematic view of the *device hierarchy*, either as a tree view, or as a customizable graph. The administrator can visually organize the sensor representations on the workspace, view and edit their settings, and see the current status reported by each sensor. The application keeps in touch with the entire physical hierarchy by the means of the eBox which intermediates all the communications.
- A *real time plot* of the sensor measured parameters, and statistical plots with previous values stored in the database. These visualizations are used by the engineers to discover flaws in the water distribution infrastructure, patterns in water quality parameters, or any other useful statistics.
- A *three dimensional representation* of the city-wide piping infrastructure correlates the sensor hierarchy with the actual physical placement of the sensors and the linking devices. By using the powerful capabilities of Microsoft Direct3D, the entire city map is displayed, and the pipes are displayed as a graph above the map. The sensor hierarchy detected through the eBox can be placed over the piping map, thus the management application can offer support for further sensor placement or pointing the exact failure points.
- *Simulation of various sensor placements and potential flaws* can be done by inserting emulated devices into the real sensor hierarchy. They act as real devices and can provide valuable information regarding a future Pi-CoPS network expansion.

The Figure 6 depicts the management application main window. The *distribution view* shows a simple water network implemented over a portion of the Bucharest map. Each device has its corresponding symbol and a small overlay that indicates its status. Each selected device can be identified in the *network tree structure* (on the left), and its *properties* displayed on the right. The graph below the distribution view illustrates *real-time water parameters* taken from the last selected sensor. Various operations can be performed directly with the contextual menus of the figured elements, or by accessing the menus and toolbars.

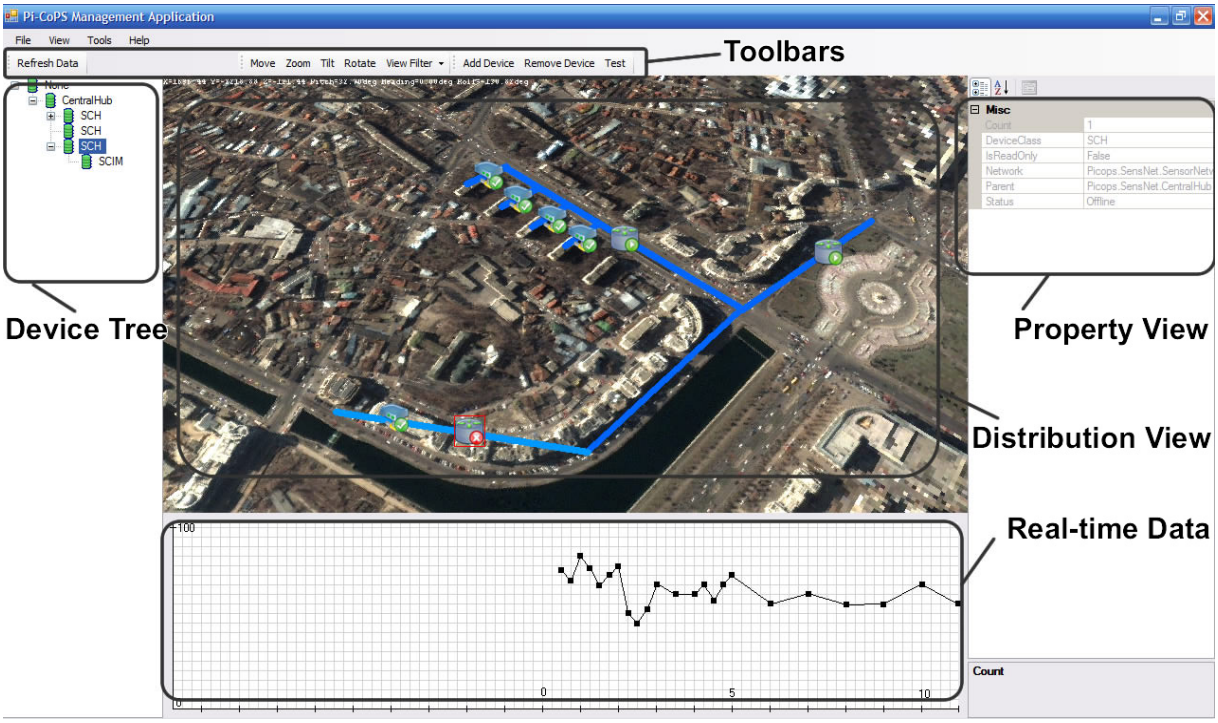

Figure 6 Management Application Main Window

3.1.6 Web Application Server

The web server runs ASP.NET 2.0 and hosts both the web services used for accessing the database (no direct connection exist between the "outer world" and the database sever in order to provide security for sensitive data) and the web application used to present in a filtered manner information pertaining network status and water quality statistics.

The web services are developed on top of the ASP.NET architecture using WSE 3.0 (Web Service Enhancements 3.0) to provide increased security (for communication between devices or other clients and the database server) and optimized transfer for binary files (the communication between the desktop front-end and the database). By using certificate based authentication users are given differentiated information in a secure way (i.e. company's customers or regular users can access far less and detailed information than the system administrator).

The web service used by the eBox in the network implemented a simpler authentication mechanism, as the API for WSE is not supported on the current version on .NET Compact Framework, nor by other third party libraries (as OpenNETCF). Instead, a custom made *Challenge-Response* authentication protocol is used, as short response times and high device responsiveness are essential.

3.2 Hardware Design and Implementation

3.2.1 Communication Media and Protocols

The Central Hub manages the Pi-CoPS network by means of an Ethernet connection. Each NetLiD also has an Ethernet interface and is assigned an IP address. The NetLiDs can be part of a city-wide LAN or can connect directly to the Internet. Pi-CoPS can thus use the existing Internet infrastructure within a city, to facilitate installation, or it can use a separate network which would be more secure and possibly more reliable.

Ethernet was chosen because it is a proven technology, and it can support large networks or give easy access to the Internet. Although Internet connectivity can sometimes fail, and a minimum latency cannot be guaranteed, the system is designed to tolerate that. Communication takes place over TCP, which increases the connection reliability.

Communication between a NetLiD and its SMART*i*-s does place a maximum limit on response time. Even if very fast communication is not required, a deterministic access scheme is mandatory and a higher degree of reliability is needed. SMART*i*-s are connected to a NetLiD on a 485 half-duplex serial bus [8]. Many alternatives exist to the 485 bus, such as CAN and 1-Wire(r). The industry-standard 485 was chosen because of its overall simplicity as well as flexibility.

The NetLiD acts as a bus master and polls the SMART*i*-s regularly. SMART*i*-s only respond to commands given by the NetLiD to their unique address. In case a pollution event arises, the NetLiD may also send commands to SMART*i*-s. A small amount of data, some 5 to 30 bytes, is contained in each packet. Packets that the SMART*i* sends contain sensor measurements, alarm flags and valve state. The NetLiD may command the opening or closing or certain valves. The protocol also provides error detection and packet retransmission with an ACK/NAK mechanism. By keeping the packets small, it is possible to use a modest speed of 115kb/s for the serial bus, which permits the use of long (1km) cables.

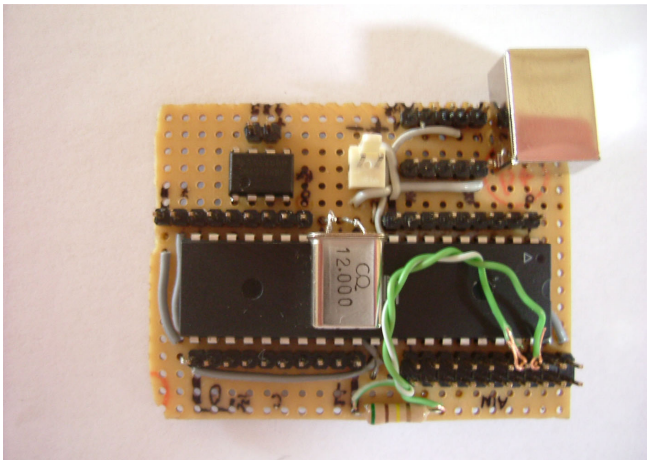

Figure 8 SMART*i* Initial Prototype

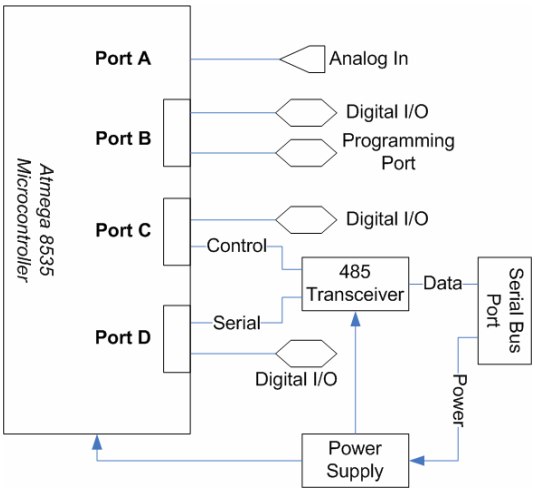

Figure 7 SMART*i* Block Diagram

3.2.2 SMART*i*

The SMART*i* is built around an ATmega8535 microcontroller [6]. Because the chip integrates most of the necessary resources, a solution with a very low component count could be designed, resulting in increased reliability. The SMART*i* does need to be the most reliable component in the Pi-CoPS architecture, because its failure renders a segment of the water network unprotected.

3.2.3 NetLiD

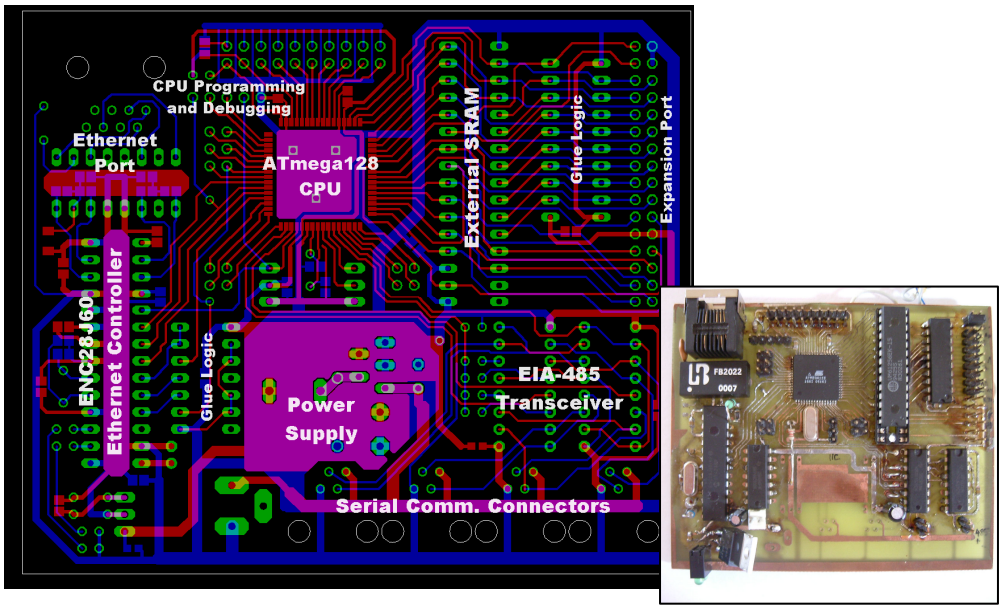

Figure 9 The PCB Layout and the Initial Prototype of the NetLiD

The NetLiD is powered by an ATmega128 microcontroller [7] running at 16 MIPS. A 64KB RAM was installed on the prototype (for development purposes, the maximum non-segmented amount was used). The NetLiD connects to a LAN or to the Internet by means of an ENC28J60 Ethernet controller. A PCB layout of the NetLiD is depicted in Figure 9, with the main components indicated. The NetLiD prototype board was built manually and its layout differs slightly from the final version, which will be professionally manufactured.

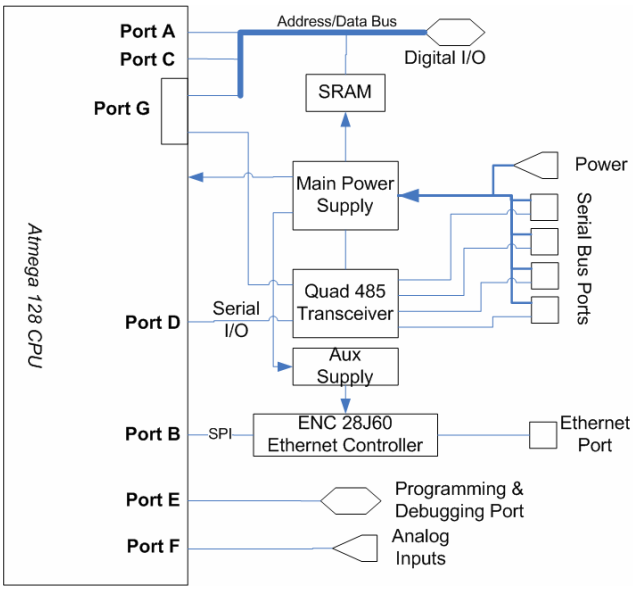

Figure 10 NetLiD Block Diagram

Selection of the CPU for the SMART*i* was based on several criteria, the most important being processing speed, hardware architecture, instruction set and ease of integration. A microcontroller was preferred because it integrates several resources such as UARTs, timers and memory. This simplifies the board design and increases reliability. The

necessary performance of the CPU was estimated and several candidates were then analyzed. An enhanced 8051 architecture with integrated Ethernet MAC was a significant candidate, but it was harder to set up and needed more external chips, some of them hard to source. The current CPU can be replaced by a newer, pin-compatible variant that would eliminate the need for an external SRAM chip.

The 485 interface on the NetLiD allows 4 bus segments to be logically connected, but electrically isolated. This improves electrical load balancing and assures that, in case a segment fails (either short- or open-circuit), the rest remain functional. The bus cables also contain, besides the 485 twisted pair, an additional pair for distributing power to SMART*i*-s in a fashion similar to Power over Ethernet.

3.3 Software Design and Implementation

3.3.1 eBox II Application

The eBox application is a .NET Compact Framework 2.0 application. It features a Windows Forms module used for *in-place* observation of network status and a resident module. The resident module processes data from the NetLiD-s and further relays it to the Database Server and also provides map analysis and decision taking in case an event is detected.

The application is built into the operating system image and it's set as a default start-up component in order to ensure minimal down time in case of power failure. In this respect, the only module of the application that loads by default is the Ethernet listener and the web service client. In case an on-field operator needs to have an overview of the network status he can connect an external display to the eBox and manually start the network status monitoring tool.

3.3.2 Pi-CoPS Management Application

The management application is a Windows Forms application, written in C# for .NET Framework 2.0. It also uses the Managed DirectX 2.0 Beta shipped with DirectX SDK (February 2006).

The Application Document

The application is basically designed as a Single Document Interface, where a document represents the city infrastructure, sensor and upper layer device placements, and any other custom application settings saved by the user. In a normal circumstance, only one document would be saved on the workstation, that is, the one that represents the city whose company implemented Pi-CoPS.

The document is an XML file that contains the serialization of every aspect of the application, with the exception of the city infrastructure (3D map and piping graph) which is held in the global storage server, along with the sensor readings.

Application Modules

The implementation has been divided in 5 .NET assemblies:

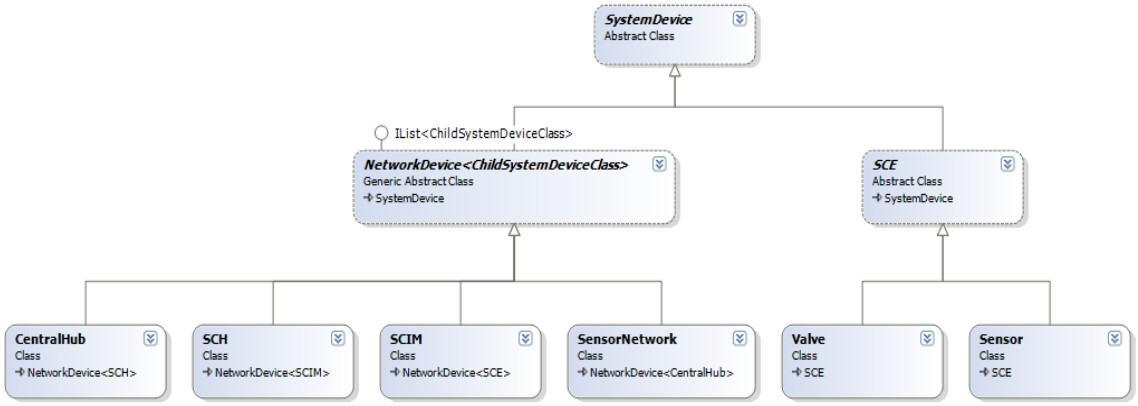

Figure 11 The Sensor Network Representation Class Hierarchy

- **SensNet** – contains the classes that encapsulate the sensor network hierarchy (see Figure 11), along with various helper classes and structures (not figured). Some class names differ from those of the network device they model, as the class hierarchy was designed in an earlier stage. An abstract, **`SystemDevice`** class was defined to ensure the common functionality of all its descendents and it includes, along with general status properties, support for external synchronization (connectivity with the eBox or with a virtual device, for simulation purposes), and a property interface for runtime visualization and editing in a **`PropertyGrid`** control. Its descendents provide explicit implementations and various instantiation means. The **`SensorNetwork`** class is the logical root of the class instances hierarchy, provides access through iterators at any level of the network and notifies through events when a change in the network occurs.
- **DistNet** – contains the classes that represent a pipe infrastructure. It is represented as an oriented graph, with **`Junction`**s as nodes and **`Pipe`**s as arcs. There are special nodes that represent the water sources (tanks or pumps) and the consumers (the buildings). The orientation of the arcs is given by the water flow. Each **`Pipe`** object contains a collection of **`SystemDevice`** objects that represent the sensors and valves installed on the physical pipe. If a water flow sensor is attached on each pipe of a junction, the application can verify the flow conservation on that junction and thus identify any possible water leaks.
- **DXControls** – represent a collection of controls, most of them using the Direct3D rendering capabilities. Each Direct3D control is based on a basic **`DXControl`** that provides Direct3D engine initialization and a generic rendering pipeline. There are two descendents of this control: **`DistributionView`** which displays the spatial view of the city map and distribution network, and **`DeviceView`** that allows the user to see the device tree as a graph. There is also a **`DeviceTree`** control, derived from the common control **`TreeView`**, and a **`DataMonitor`** control for monitoring the sensor parameters.
- **GraphRes** – provides a consistent access to the graphical resources that the application uses directly. They consist mainly of device icons, at various states and resolutions.

- **MainApplication** – contains the user interface and the main application logic. It instantiates the specialized controls, loads all the necessary data and maintains the interactions between all the controls.

3.4 Custom Tools

3.4.1 SMART*i* and Sensor Emulator

This PC software allows a high number of virtual SMART*i*-s to be connected to a real NetLiD in order to test its proper functioning under full load. It was vital during the development stages in the testing and debugging of NetLiD software, and will continue to be a useful tool as the software is being refined.

The Emulator allows one to modify the reported parameters in real time, simulating a contamination event, or can generate random events at specified intervals.

3.4.2 NetLiD Emulator

In a fashion similar to the SMART*i* Emulator, the NetLiD Emulator aids in the development of the eBox software. The main difference is that it communicates with the eBox via an Ethernet port and exchanges a larger quantity of data. In the real system, each NetLiD would have its own IP address, but the eBox can accept connections coming from the same IP address, allowing emulation of a very large number of NetLiDs on a single PC.

The NetLiD Emulator reads a map of a virtual water network and a file specifying which NetLiDs defined in the map to actually emulate. Functioning is then similar to that of the SMART*i* Emulator.

3.4.3 The Mipmap Builder

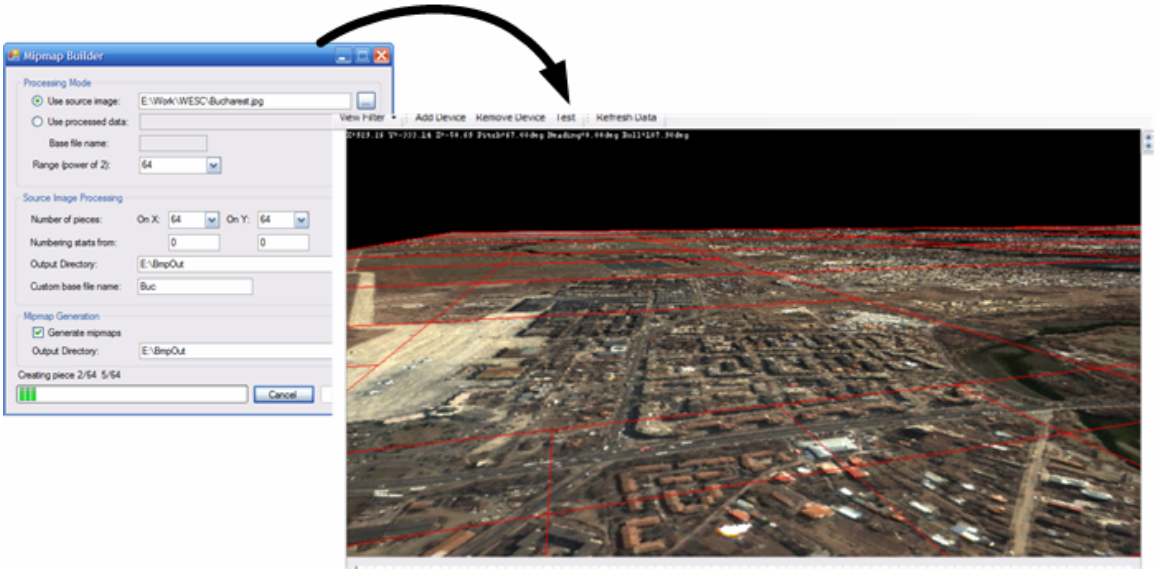

Figure 12 The Mipmap Builder and a View of The Generated Surface

The Mipmap Builder is a Windows Forms application that helps with the creation of the bitmap image components (*mipmaps*) that compose the 3D surface of the city land displayed in the Distribution View of the management application.

Due to the fact that a city map is very large and it would require a large amount of memory to completely load and display it, the Mipmap Builder breaks apart the original image into a large number of pieces, then it joins adjacent pieces in order to form images of a lower detail, and create a representation of the original image at a larger scale, and the process is recursively applied until a single mipmap of the original image remains.

3.4.4 485-PC Interface

Interaction between virtual SCIMs and a real SCH requires an interface between the PC running the emulator and the 485 bus on the SCH. It is realized with the SCIM prototype, which already has a 485 port needed for normal operation. Different software was programmed into the SCIM prototype that allows it to connect to the PC Parallel Port and relay data.

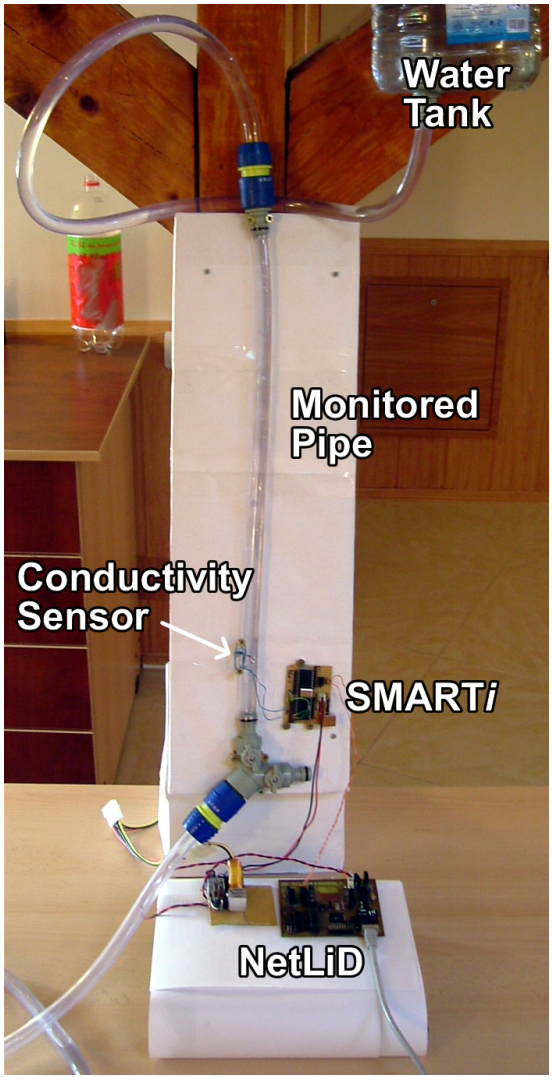

Figure 13 Pi-CoPS Testing Prototype

3.4.5 Microcontroller Programming and Debugging

A simple adapter was created to write microcontroller memories using a Parallel Port.

Software running on the microcontrollers (NetLiD and SMART*i*) sends debugging information via a serial port if instructed to do so, which is displayed on the PC using an application such as HyperTerminal. An adapter was assembled to link the microcontroller serial port and the PC serial port through a level converter.

3.5 Verification and Testing

The main part of the testing setup is depicted in Figure 13. It consists of a scaled-down water pipe on which a conductivity sensor and a valve are attached. They are also connected to a SMARTi which communicates with a NetLiD. The NetLiD is connected to an Ethernet switch (not shown), linking it to the eBox and a PC.

The SMARTi was tested and confirmed to respond instantaneously to pollution, with and without its link to the NetLiD.

The NetLiD was tested under full load using an array of 255 emulated SMARTi devices. Network load (for the serial link) was 66% under normal conditions (monitoring only). The NetLiD was instructed to send override commands to all the SMARTi devices.

Network load was 89% (highest possible load). This assures a 10% margin for packet retransmission in case of data errors. Pollution detection was simulated on some of the SMARTi devices, in varying percentages. The NetLiD responded correctly. Maximum response time was 3 seconds.

For the last test, the NetLiD and the eBox were connected to the Internet from two locations. Maximum round-trip response time (SMARTi-NetLiD-eBox and back) was under 4 seconds.

4 Summary

4.1 Project Current Status

A prototype implementation has been built and it is fully functional. Each component of the system has been tested and found to be conforming to its specified functionality under a large range of external conditions. By benefiting the rapid application development facilities offered by .NET Framework 2.0 and Visual Studio 2005, we were able to develop in short time the set of complex applications that automate many deployment, diagnostics and maintenance tasks.

Due to its nature, the system as a whole could only be physically tested on a small scale. This proved all modules function correctly. Additionally, we used the emulators to generate a large virtual sensor network and verify the performance of the eBox II system and the management application.

Currently, discussions with the local water distribution authority are carried out, in order to get access to a portion of a real water distribution network and test the Pi-CoPS system under the real life conditions.

4.2 Further Improvements

The Pi-CoPS system has been implemented with all the planned functionality; in addition it can be improved with some add-ons. Wireless access could be realized for devices that need to be placed in remote locations, where a cabling infrastructure would be difficult to execute.

Moreover, new sensors can be tested and interfaced to be included in the sensor array, as new pollutants are suspected to be able to infiltrate in the distribution piping. However, the 255 SMART*i* limit on each NetLiD is expected to be sufficient for the following decade.

4.3 Conclusion

Pi-CoPS addresses the high demand in the field of environmental protection for more advanced solutions to water quality preservation, as it brings a new approach to this vital mission: a fail safe, responsive, and scalable water quality management system deployed over the existing water supply infrastructure.

It provides to the distribution authority the complete solution for deploying, maintaining, and extending a distributed network of sensors and smart devices. If placed strategically and in sufficient number, they can assure a high degree of protection against virtually any type of detectable pollutant. They would also provide complementary assistance to the already existing filtering systems that implement water treatment and chemical analyses.

In addition, the system offers services for data collection and storage, powerful monitoring capabilities and public access for general information. In this context, the eBox II running the Windows CE 5.0 platform played a centralization role in the system, as it confirmed to be a stable and fast-response device.

By addressing the problem of water, the very essence of life, our system greatly impacts human life in a large variety of communities by providing the means of assuring a high standard of public safety.

5 References

Web References

[1] Our Most Precious Resource, Wasted
http://www.dustpro.com/DustParticles.htm

[2] NRDC Study Finds Safety of Drinking Water in U.S. Cities at Risk
http://www.nrdc.org/water/drinking/uscities.asp

[3] Clean Water Act Status Report
http://www.scorecard.org/env-releases/water/cwa-us.tcl

[4] The EPA's Pesticide-Protection Failure
http://www.nrdc.org/health/pesticides/olgpesticides.asp

[5] Wellness Goods – Don't Drink the Water
http://www.wellnessgoods.com/dontdrink.asp

[6] Atmel ATmega8535 Microcontroller Datasheet
http://atmel.com/dyn/resources/prod_documents/doc2502.pdf

[7] Atmel ATmega128 Microcontroller Datasheet
http://atmel.com/dyn/resources/prod_documents/doc2467.pdf

[8] Texas Instruments, 422 and 485 Standards Overview and System Configurations
http://www-s.ti.com/sc/psheets/slla070c/slla070c.pdf

[9] Managed DirectX 9.0 Tutorials
http://www.thehazymind.com/

[10] MSDN Web Service Resources
http://msdn.microsoft.com/webservices/

[11] Mikehall's Embedded WEblog
http://blogs.msdn.com/mikehall/

Books

[12] A. Santos Lobao, E. Hatton, *NET Game Programming with DirectX 9.0* – Apress, 2003

[13] Steve McConnell, *Code Compete, 2nd Edition* – Microsoft Press, 2004

[14] Gavin Powell, *Beginning Database Design* – Wrox Press, 2005

[15] Chris Hart, *Beginning ASP.NET 2.0* – Wrox Press, 2005

[16] Dhananjay Gadre, *Programming and Customizing the AVR Microcontroller* – McGraw Hill, 2000