Team 44Tech

Cristian Iuliu Pop
cristip0p@yahoo.com
Ioana Romelia Bratie
ioanabratie@yahoo.com
Mircea-Dan Gheorghe
byrealmircea@yahoo.com
Omar Choudary
choudary.omar@gmail.com

Mentor:
Decebal Popescu
decebal@csit-sun.pub.ro
Index

Abstract .................................................................................................................................................. 2
1. System Overview ............................................................................................................................... 4
  1.1.1 System Components .................................................................................................................. 4
  1.1.2 Innovations .................................................................................................................................. 4
  1.1.3 Market Analysis ............................................................................................................................ 5
  1.1.4 Design Methodology .................................................................................................................... 5
  1.1.5 Team Organization ....................................................................................................................... 5
2. Implementation and Engineering Considerations .................................................................................. 6
  2.1.1 System Utilization ........................................................................................................................ 6
  2.1.2 Sentries ........................................................................................................................................ 8
  2.1.3 Network Specification .................................................................................................................. 9
  2.1.4 Smart Device Application ............................................................................................................ 12
  2.1.5 eBox Server Application .............................................................................................................. 13
  Sound data processing algorithm ........................................................................................................... 13
  2.1.6 Web Service .................................................................................................................................. 15
  2.1.7 Web Interface ............................................................................................................................... 15
  2.1.8 Costs ............................................................................................................................................. 16
  2.1.9 Tools ............................................................................................................................................. 16
  2.1.10 Testing ....................................................................................................................................... 17
3. Summary ............................................................................................................................................... 18

Abstract

The year 2005 might not be considered one of the best years neither for Romania, nor for other countries. Floods, hurricanes, soil erosion and fires have devastated many regions all over the Globe.

Saving the forest would have definitely changed this situation. Forests act as an important shield against floods. They reduce the greenhouse effect, the global warming process and settle the soil, preventing the erosion.

Deforestation has many causes, but one of the most important is illegal logging. In our country 40% of the trees that are cut are a result of it and this percent rises up to 80% in some countries of South East Asia, Latin America and Africa. Theft is somewhat an easy thing to do, considering that forests occupy a large area that cannot be supervised.

So far, the only way one could see if trees were being cut was to walk through the forest and check for thieves or for the stubs they left behind. The process needed either many workers or a lot of time. Fires were usually discovered too late, when no one could do anything about it, not to mention that there was no way of preventing them. Our solution offers a new way of handling that: few workers and a system that does the entire job in real-time. It comes to satisfy the need of an automatic and reliable way to detect and prevent factors that could destroy the forest. This can be done using a network of sensors placed in the forest, which gather information about cutting noises, temperature, humidity, pressure and carbon monoxide in the area. Data is processed and the forester is informed about problems like fire or illegal logging.
Forest Watcher is a security application and a fast response is expected from human operators. But as research confirms, humans act poorly in stress conditions, and they can take irrational decisions without proper training and experience. The system is designed as a user oriented solution, trying to overcome this problem. All user interaction will be reduced to a minimum and information will come in form of images and sounds, reducing the risk of confusion and delays in taking an action.

We designed a flexible solution by using a modular approach both for hardware and software aspects. Each module can be upgraded and new functionality can be added. The scalability issue has been resolved by using a multi-tier architecture and by making Forest Watcher a service oriented application (SOA).
1. System Overview

1.1.1 System Components

The solution consists of two main parts:

**The Area** is represented by a maximum of 200ha of forest. It includes a *Network of Sentries*, *Mobile Units* and a *Central Unit*.

- **Network of Sentries** is a mesh of maximum 200 sensors placed in the forest that communicate with the Central Unit using wireless technology.
- **Mobile Units (MU)** are portable devices (Pocket PC, PDA) used by the forester to receive alerts from the Central Unit.
- **Central Unit (CU)** is the brain of the solution, built on an eBox II device with Windows CE operating system. It runs an intelligent data processing application and synchronizes with the Forest Watcher Server System.

**The Forest Watcher Server (FWS)** is a system that unifies multiple Areas. It consists of three modules: *Web interface*, *Web Service* and *Data store*. These modules can be deployed on different machines for scalability.

1.1.2 Innovations

Our solution represents a hybrid between an original idea, embedded systems, telemetry, artificial intelligence and the newest Microsoft technologies. Most of surveillance system developed so far is not addressed to large and wild areas, like forests. Those who do are not suitable for common forest owners, who don't need to spend money on a system too complicated for their needs. Forest Watcher comes with a new way of supervising such areas: real-time, ready-to-use data, no redundant (and expensive) information (like video) and easiness of use, without need of specialists.
1.1.3 Market Analysis

Romania has over 6.4 million ha of forest, estimated to 1.350 million cubic meters of wood. More than 25% of those represent private property, and more forests will become private in the following years. Considering this fact, the government issued a law that states that the private forests have to be defended against illegal logging and hunting [1].

The market segment which our solution addresses to is represented by the forest owners, forester associations and security companies that are affected by this law and to other international organizations that have forests in custody. The solution is localizable and customizable, being suitable for any region that concerns surveillance.

Considering that electronic components are cheaper everyday we appreciate that protecting one hectare of forest will cost less than 30$. At the same time the value of one hectare of forest can rise up to 30000$. We can say that you spend 1$ to protect 1000$.

1.1.4 Design Methodology

We used SCRUM, an Agile Project Management technique, which allows us to take care of aspects like visibility, inspection and adaptation without the burden of complex management schemes.

The starting point of this model is the Project Backlog (an ordered list of items that should be accomplished). All the work is done in Sprints, initiated with a Scrum meeting, when the project backlog is revised and a Sprint Backlog is developed with the team’s help.

During the Daily Scrum each member’s work is inspected [2].

Being a security solution by definition, we tried to apply the STRIDE model to determine possible threats and try to mitigate them. This model categorizes the threats in six classes: Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service and Elevation of privilege [3].

1.1.5 Team Organization

Communication is a key element, as the SCRUM model encourages a self organizing team and self organizing members. The team members have specific roles to maximize the use of their skills and experience: Ioana - mathematical and algorithm aspects and software programming (C#, Databases); Omar - networking and software programming (C# on embedded systems); Mircea - hardware aspects and low-level software (C on Controllers); Cristian - management (SCRUM master), low-level software and eBox II hardware aspects. Even if we designated these roles the team works together when decisions are to be made and during the Daily scrums, when team members learn from each others’ experience.
2. Implementation and Engineering Considerations

2.1.1 System Utilization

Installation
First, a team equipped with Pocket PCs deploys the network of sentries, and the eBox is placed in an adequate environment with Internet access (e.g. a foresters’ house). The Internet connection can be provided by a GPRS or satellite link where there is no mobile phone coverage.

Each sentry is mounted on a tree (taking in consideration radio visibility with sentries or central unit near it). The sentries are mounted starting from a nearby location of the central unit. After it is mounted, it receives an activation command and position from the Pocket PC of the worker (with GPS module). The sentry stores its position and relays it to the Central Unit.

A new area account is created by the administrator in the Forest Watcher Web Service allowing the eBox to send data.

The sentries will start to send data and synchronize timers to avoid high network load, and the eBox will start to receive telemetry.

Operation
The forester will be trained into using the mobile smart client; which will be deployed on mobile units (Pocket PC, PDA) equipped with GPS and Bluetooth modules. This application is easy to use and requires little or no user interaction. By default the system displays the map of the area and the Forest Condition (FCON) indicator.

The Forest Condition (FCON) level is a fast mean of seeing the threat level in an area or globally. It has four levels:

4 (green). Normal operation: no or small possibility of threats
3 (yellow). Medium alert: possibility of fire based on humidity and temperature levels, sound activity near the border of the network, sentry malfunction – timeout in sending information
2 (orange). High alert: threat is detected in the perimeter of the surveillance area (cutting of the trees detected or monoxide and temperature levels exceed the normal values)
1 (red). Extreme alert: multiple threats (theft) distributed throughout the perimeter, certain fire detected
If the FCON changes state, the locations that determined the change will be shown on the map. The fastest way to the nearest threat site and an arrow indicating the direction that the forester should go will be calculated and displayed.

When FCON changes to high or extreme, the Web Service will raise an alert and the administrator or owner of the forest will be announced by SMS or email, with the possibility of seeing the exact cause using the Web or WAP interface from any Internet connected device.

If forest health, meteorological or other type of research is required over the Areas, the Web interface can provide the user with charts or maps of temperature, pressure, monoxide and humidity levels.

For each owner or administrator, the Web Service provides through the Web interface a Global indicator (Global FCON) based on each Area’s FCON.

**Failure Scenarios**

*Internet failure on a CU*

When an eBox ceases to transmit over a specified period of time, the Web Service modifies the FCON for that area to yellow and a special alert is raised to the administrator. If the failure is not power related the eBox will continue to function disconnected, keeping a local cache of the logs. When the internet connection is available the system will automatically synchronize with FWS.

*eBox II turns off*

If the power supply of the eBox II fails, or simply the eBox II is turned off (from the switch) our system will only be affected during the power off time. After the eBox II is restarted, our Windows CE image and FW application are automatically loaded and the system will have full functionality.

*Sentry malfunction*

If a sentry is out of power, or it is damaged, the network will still be able to communicate due to the adaptability of the DSR protocol used to route packets between the sentries. They will immediately discover new routes for destinations that used the bad sentry as an intermediate. After the failure, the eBox II will notice the malfunction of the sentry and an alert will be sent to the Mobile Units in order to solve the problem. After the sentry is replaced, the new one will be quickly integrated into the network.

**Maintenance**

The system requires regular maintenance actions:

The sentries must be operational all the time and can fail due to environmental conditions or power module malfunction (e.g. if the sentries run on batteries, they must be changed periodically).

The eBox must be connected to Internet to report to the Forest Watcher Server and a reliable power source must be provided.
### 2.1.2 Sentries

The data acquisition system of this solution is represented by the sentries. These are modular sensors organized into an ad-hoc network that are continuously scanning the premises.

For maximum versatility we adopted a modular concept: each part of the node can be replaced with a different component. The sentries can be upgraded or used in a different environment using the same architecture.

The node is made out of 4 customizable modules:

#### Base module

This module is the brain of the node. Its first task is to process analog data from the sensors. The second task is to act as a member of the ad-hoc network (transmitting or relaying data) and the last and maybe most important is power management. In our case we used Microchip’s PIC 18F4550 microcontroller that runs at 48 MHz, and an additional external 128kB I²C bus EEPROM memory used as buffer for the sound samples and DSR Routing algorithm. This MCU has the advantage of high speed, multiple analog to digital inputs, 20kB of RAM and USART, combined with µA power saving function, all in a cheap capsule [4].

#### Sensor module

This module receives input from the outside world. For sound detection of a chainsaw, axe or gunshot we used a high sensibility microphone and amplifier. Temperature, pressure, carbon monoxide and a humidity sensor are used to monitor the environment. They can be used for prevention and a precise detection of fire.
## Transceiver

To use a standard means of communication a Class-A Bluetooth device was selected for this module. The use of a Bluetooth device instead of simpler transceivers had a big impact in the development process of the software application on the MCU. The tradeoff was in choosing a more expensive piece of equipment but having more functionality already implemented in hardware.

The PromiESD Bluetooth module has a range of 100m and a low power consumption of about 7 mA. It connects to the MCU using the hardware USART and uses an “AT” command set (like most PSTN modems).

The same module is used on the CU (the eBox), connected on a COM port through a voltage-level adapter. A protocol stack driver has been implemented in C language with little or no modification from the DSR Simulator.

The range of the module can be extended using a dipole antenna (200m) or using a directional patch antenna (1.2km) [5].

## Power module

The most important part of a remote sensor it’s its capacity to function a long time unattended. The node can be powered up by any kind of energy source. The Power Selector will detect if it has continuous energy (from a solar cell, power line, etc) or has only a power cell attached, indicating the power mode that the base module should adopt.

To prolong the lifespan of the energy cell the node will work using an internal timer to schedule the data acquisition system. The calculated time was about 15 minutes to record sound and relay data to the CU. Other sensors can be activated more frequently and the data kept locally or sent over to the CU only when an emergency rises.

The time divisions were calculated to avoid network overloading and build a low cost system.

For the prototype we have used a Li-Ion 9V battery and to simulate the solar cells we used a conventional power source (power adapter).

### 2.1.3 Network Specification

#### Network communication

The sentries in the network communicate via Bluetooth with the PromiESD device as described in the hardware section. For the communication of a sentry with another device that is not on its wireless range our solution has implemented a routing protocol specially designed for Mobile Ad-hoc Wireless Networks (MANETS). The routing protocol is an implementation of Dynamic Source Routing (DSR) protocol.

#### DSR protocol

This protocol is the piece of the network that gives to a sentry the path to another device. It has 2 base modes of functionality:
- **Route Discovery**: in this mode, the sentry finds out the route to a destination device (sentry, eBox, PDA).
- **Route Maintenance**: in this mode the sentry verifies that its entries in the routing table are correct, and that the intermediate sentries in each path are still there.

These functionalities are implemented through the exchange of 3 messages:

- **Route Request**: is initiated by the sentry that needs the path to a destination device but it doesn’t have a path to that device in its routing table. In the way from the initiator to the target device, each intermediate sentry puts its address in the message. This is the “source routing” method of the protocol.

- **Route Reply**: is initiated by the sentry destination that receives a Route Request for its address. When the Route Request arrives here at the destination, in the message there is a list with all the addresses of all the intermediate nodes that the message passed through its way from initiator to destination. The target sentry only has to reverse the list in order to send the reply to the initiator sentry.

- **Route Error**: this message is initiated by any node that has to send a message to one of its neighbors but discovers that the communication can’t be established [6].

**Data flow**

The sentries need to send their data (sound waves, temperature info, etc…) to the eBox in order to maintain the knowledge database updated. They also need to send this type of data from the CU to a Mobile Unit.

The data (sound waves, etc…) needs to be encapsulated in order to be used by the DSR protocol. This data has to be also encapsulated to be used by the Bluetooth device. The destination device has to remove the encapsulation and send the data to the application that needs it. For all of this we have implemented a network protocol, named FWProtocol, very similar to the ISO OSI and TCP/IP but without the Internet layer.
This data flow is shown in the next figure:

**DSR encapsulation**

As described earlier, each message is encapsulated in one of the 4 types of messages: RREQ, RRPLY, RERR and RSOURCE.

**ESDPRO encapsulation**

At this level, the message received from the Network layer is encapsulated in a frame so the Bluetooth target device knows when a message finishes and when the next one starts. In this encapsulation a CRC code is sent for the reliability of data. Also in this encapsulation the addresses of the sending sentry and the neighbor sentry (the Bluetooth pair) device are introduced.

**Bluetooth connection**

Each sentry has a Bluetooth chip inside for communication. For the communication between neighbors, each sentry has 2 functioning modes:

A. **Initial Mode**: In this mode the sentry is able to observe new sentries that can be added to the network. In this mode each sentry stays 24 hours from the startup. In this mode the sentry follows the next steps:

1. Sends a discovery signal for 5 minutes so the other sentries can observe the presence of the new sentry
2. Goes into the passive mode. In this mode the device can be connected by any other Bluetooth device that knows its address. This mode consumes very little power (only 6mA), so it’s the base mode of the sentry.
3. Every 5 minutes it goes into listening mode (for 30 seconds) to observe the possible presence of new neighbor devices. If such a device is observed it sends a **Hello** packet to the discovered device in order to establish a neighbor relationship between the two devices.

**B. Normal Mode**: After the 24 hours, the sentry goes into passive mode (same as A.2), and stays like this until the sentry is restarted (from a command of the Pocket PC).

When a sentry has to send a message (it can be originated here or can be a DSR message that has to be forwarded) it searches its neighbor table for the neighbor address and initiates a Bluetooth connection with that neighbor. If the connection fails, it puts that message in the send buffer and waits until the connection with that Bluetooth device is possible. If after 5 attempts (each attempt is delayed with the double of the last period of time) the connection fails then the message is discarded (erased from the send buffer). If it was a DSR forwarded message, then a RERR message is sent back to the source neighbor.

### 2.1.4 Smart Device Application

The Mobile Unit can be used to observe the sentries situation from anywhere in the covered area (where the sentries can communicate). It can also be used to find the shortest path from the actual position (where the Mobile Unit is) to the alerting sentry (if there is one). The interface for this application is shown in the figure.

In case of emergency, the application notifies the user by specific sounds and simple graphic announcements on the screen. It automatically changes its operating mode to highlight the threat area, computes the fastest way to get there taking into consideration level or ground obstacles marked on an internal map. It also displays the direction the user should take to reach the threat area, ETA and provides a voice guide.

**Map**: shows the map of the sentry's Area.
**Path**: shows the shortest path to the alerting sentry.
**Threat Status**: shows the alert level of all the sentries. This is sent by the expert system on the eBox that examines all the sentries and calculates the FCON level of the forest taking in consideration all the danger factors: temperature, humidity, sound type.
**Action Buttons**: they are used to change between the different types of maps (sentries map, temperature map) and to create a shortest path from the Mobile Unit to the selected sentry.
2.1.5 eBox Server Application

The eBox Server application has three functions: telemetry, data processing and data service.

The telemetry module

This module acquires data from the sentry network using the FWProtocol drivers. These drivers connect to the network via the Bluetooth module mounted on the COM port of the eBox. The data is split into two classes: sound information and simple sensor values. Simple sensor values as temperature, humidity, monoxide and pressure are converted in floating point values in a ready to use measuring system. Localization is later provided by the client applications (e.g.: converting Celsius to Fahrenheit). More complex sound data is prepared for processing (level adaptation), passed through a Fast Fourier Transform, data filtering (some frequencies are cutoff) and then passed through a neural network trained to recognize several classes of sounds. These algorithms are described later in the report. This module is considered to be time critical and is being written as a DLL using C language.

The data processing module

Stores logs of data from the telemetry module and prepares them for the smart client application. It computes the FCON indicator for the entire area using an uncertain calculus algorithm and provides a real time map of the distribution of threats. Data is presented in form of vectors with associated values of the different parameters to the mobile clients only when a threat is detected, to minimize network traffic.

The data service module

Provides connectivity with the mobile units through the FWProtocol drivers and it uses an ADO.NET disconnected recordset to send data logs to the Forest Watcher Web Service through an Internet connection. The data is sent over a secure connection (using SSL) and the eBox must authenticate with a unique key first (the key is a hash from its hardware serial numbers). The last two modules are being written as a standalone service in C# and pinvoke is used for the connection with the DLL.

Sound data processing algorithm

The telemetry module is the engine that processes the input received from the network of sentries. Sound processing has two main parts:
The first one is to apply Fast Fourier Transform on the sample sound received from the sentries. The advantage would be that now the modified signal is independent of phase, therefore easier to analyze.

The second is to pass previously obtained data through a neural network, which would determine the nature of the processed sound, in particular if it is a chainsaw noise or not. This can be extended for many other sounds just by training the network with different inputs.

**Neural network**

A network consists of units and directed weighted links (connections) between them. In analogy to activation passing in biological neurons, each unit receives a net input that is computed from the weighted outputs of prior units with connections leading to this unit. Picture shows a small network.

An activation function first computes the net input of the unit from the weighted output values of prior units. It then computes the new activation from this net input (and possibly its previous activation). The output function takes this result to generate the output of the unit.

Knowledge is distributed throughout the net and is stored in the structure of the topology and the weights of the links.

We have built a 3 layers neural network to satisfy the needs of our application. First layer represents the input, an array of numbers, in our case the Fast Fourier Transform of the sample sound. The next one would be an intermediate, hidden layer, and the third one is the output layer.

For training, we used the back propagation algorithm, where the weights of the links are calculated using the formula:

\[
\Delta w_{ij} = \eta \delta_i o_i \\
\delta_i = \begin{cases} 
 f'_j (\text{net}_i)(i_i - o_i) & \text{if unit } j \text{ is an output unit} \\
 f'_j (\text{net}_i) \sum_k \delta_k w_{jk} & \text{if unit } j \text{ is a hidden unit}
\end{cases}
\]

The network was organized by (automated) training method, using a visual tool: Stuttgart Neural Network Simulator (SNNS). Training patterns were used for learning, validation and testing. After the learning process ended, a C code file was generated with the topology.

We have chosen this type of algorithm due to the inherent fault tolerance. Furthermore, neural nets can be made tolerant against noise in the input: with
increased noise, the quality of the output usually degrades only slowly (*graceful performance degradation*) [10].

### 2.1.6 Web Service

Forest Watcher Server solution is a service oriented application (SOA). It is designed to resolve real world issues as scalability, interoperability and security. The web service has several functionalities:

- Data acquisition from different areas (It synchronizes with the eBox servers that are using disconnected recordsets.)
- Data processing (For each area it builds threat, average temperature, humidity, carbon monoxide and pressure charts and distribution maps. Based on uncertain calculus algorithm [11], it calculates the Global FCON indicator).
- Real time warnings (Using SMS, email and telephony, the system announces the owner and administrators of the forest when the FCON changes above a selected level and it presents the possible reasons.)
- AAA (Authentication, Authorization, Accounting) (It restricts users from accessing data that is not meant for them and keeps logs of access.)

### 2.1.7 Web Interface

The Forest Watcher web site represents the interface that permits the access of forest owner or administrator to a range of facilities:

- forest map visualization, with sentries position and parameters
- multiple map types: temperature, humidity, pressure
- threats visualization, described in a natural language
- Global threat level visualization
- customization of threat alerts, according to user preferences
- access to logs and charts about previous situation in the area
- localization of data (automatic conversion of the measuring units, country map based on an initial user profile)

All this is at only one click from user, making the entire process very easy to manage. To improve the accessibility from all mobile devices a WAP interface is also available.

In order to protect information, only registered users can access data, and only for the areas they own or administrate. They have to login before entering the system.

The web site has been developed using ASP.NET 2.0, and will be hosted on an IIS server. Many of its functionalities would be available due to its interconnectivity to the web service previously described.
A standalone client has been taken in consideration, but we gave up the advantage of fast response time to portability and mobility.

### 2.1.8 Costs

To demonstrate the functionality we first had to work with a test board, and then build a prototype. The following table displays the costs of the components. However, mass produced the costs of the sentries are expected to be considerably lower. Furthermore, using SMD technology the dimensions will be smaller.

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM335Z Temperature sensor</td>
<td>0.5$</td>
</tr>
<tr>
<td>SemiGAS07 Carbon Monoxide sensor</td>
<td>7.0$</td>
</tr>
<tr>
<td>SMTHS10 Humidity sensor</td>
<td>10.0$</td>
</tr>
<tr>
<td>Promi-ESD-01 Class-A Bluetooth communication module</td>
<td>60.0$</td>
</tr>
<tr>
<td>PIC 18F4550-I/SP Microchip microcontroller</td>
<td>6.5$</td>
</tr>
<tr>
<td>24C1025-i/p I2C EEPROM (1Mb)</td>
<td>2.6$</td>
</tr>
<tr>
<td>max3238 Voltage-level adapter (3.3V)</td>
<td>2.6$</td>
</tr>
<tr>
<td>7805 Voltage regulator 5V</td>
<td>0.3$</td>
</tr>
<tr>
<td>LM1117T-3.3. Voltage regulator 3.3V</td>
<td>1.0$</td>
</tr>
<tr>
<td>Power Cell Li-Ion Cell</td>
<td>~10$</td>
</tr>
<tr>
<td>Other Transistor, Capacities, Resistors, Connectors, Led</td>
<td>~5$</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>~105$</td>
</tr>
</tbody>
</table>

### 2.1.9 Tools

**Platform Builder**

In order to create, build and deploy the Windows CE image of our project to the eBox II we have used the Platform Builder program that was shipped aside with it. To design our image we have used the New Platform Wizard with ICOP_eBoxII-50DVS BSP. Here we have selected the **Gateway design** template with the following subsets:

- **File Server** in the Gateway extensions
- **Firewall, Network Bridging, Network Utilities, Reference Gateway, SNTP** (to synchronize all the clocks in the network), **IPv6** (for future compatibility), **PPPoE** in Networking and Communications.

Beside these options, we have manually added the following options to the image:

- **.Net Compact Framework**: necessary to deploy Visual Studio 2005 applications
- **OS Dependencies for .NET Compact Framework 2.0**
- **Hive-based Registry**: necessary to save registry settings when the unit is turned off.
- **CAB File Installer/Uninstaller**

Also we have set the following two environment variables in order to make communication with Visual Studio .NET 2005:

- **IMGRAM128** set to 1
- **BSP_VS2005_CORECON** set also to 1
We decided to use Visual Studio .NET 2005 and Platform Builder for the programming of the eBox II, the internet Web Service and the Web Site in conjunction with eVC++ for time critical sections. Free software from Microchip (MP Lab, C18 Student edition) has been used for microcontroller programming. Matlab was used to simulate sensor data and implement algorithms during research. For neural network design, training and source code generation we used the Stuttgart Neural Network Simulator. Scripts were developed to interface the two programs. A software simulator was developed to test the functionality of DSR, the routing algorithm.

Proteus was used to develop circuit schematics and simulate microcontroller behavior. Several hardware adapters were built by the team to program the PIC Controller. For hardware developing we used testing boards (speeding up the process as the components were not soldered to a PCB). A software testing platform was developed with hardware resembling the prototype sentry and serial terminal capabilities for debugging purposes.

Communications were solved using Open VPN. All other services were handled through the VPN system: File Server (SMB), Visual Source Safe 2005 and NetMeeting. A backup FTP/Web collaboration server was used when the link to the server was unavailable. As management tools we used MS Project 2005 and MS Excel. Graphics were made in Visio 2005 and GIMP.

### 2.1.10 Testing

Applying the Scrum model, each member’s work was tested at each Daily scrum. First, each module was tested independently:

- The neural network software was linked with Matlab wave files or microphone input was used to generate sensor data.
- The DSR routing algorithm was tested using TCP/IP sockets.
- The Bluetooth devices were tested using HyperTerminal and a hardware interface built by the team. The results were very promising as it seems that even in a building the unit has radio visibility at 100 meters.
- The base unit was tested by building and programming the testing platform. The maximum speed we achieved from the PIC on the testing board was only 36 MHz; probably because of the testing board utilized (the connections are not soldered, but wrapped). Full speed was possible after the sentry prototype has been built.
- A preliminary version of the mobile smart client was tested both on eBox and a Pocket PC.
- At the end of each Sprint, all the work is put together to detect interoperability bugs between separate modules.

Real world tests and the stabilization phase are scheduled in the last sprint, when two nodes will be developed and deployed in our campus.
3. Summary

When we told our friends about our project they first saw it more like a science-fiction idea. We wanted a user-oriented, scalable and flexible solution to protect the forest. After the prototype was built, artificial intelligence algorithms were developed, network software was written, the Bluetooth communication range was checked, there all seemed more and more realistic. We even found a way to act as a team, be efficient and communicate more efficiently.

At this point we can say that Forest Watcher is no longer only an idea in a document, but it is even more than the project we had thought to initially. We started from a real life problem and we found a real life solution. We had a device that was meeting our requirements – the eBox II: small, with low power consumption, with ability to run unattended and at a low price. The rest of the solution was built upon this basis.

There is much work to do from now on, too, but we have adopted a strategy that proved to be efficient when organizing the team and the tasks.

Let's see once again what this project does: If someone cuts trees from the forest, the microphone records the chainsaw sound and sends it to the Central Unit. Here the neural network recognizes it and a message is sent to the forester. He finds out from his portable device where the problem is and he goes to fix it. In the same time the Central Unit updates the information from Forest Watcher Server, and the owner of the forest is informed about the situation and can act consequently.

Improvements can be made to a possible commercial version of the solution, especially regarding the sentries. A specially designed chip would do the entire job, being considerably smaller, cheaper, easier to replace and would consume less energy.

Further functionality can also be added, like monitoring wild life (we can look for bears for example, for a certain bird or for wounded animals). For that the sound data processing unit should be trained to recognize those specific sounds. Another field where we can add features is the data processing function of the Web Service, so that more complex analysis of data to be done using OLAP cubes or intelligent systems.

We think that applying this solution on a large scale will diminish floods and soil erosion saving lives and reducing material losses. We all deserve a safer place to live. Let’s do something about that!
References