

Wireless Sensor Network Application for Fire Hazard Detection and Monitoring

Elias S. Manolakos, Evangelos Logaras, and Fotis Paschos

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
Panepistimioupolis, Ilissia, 15784 Athens, Greece
{evlog, eliasm}@di.uoa.gr

Abstract. Hazard detection systems are sophisticated tools that can help us detect and prevent environmental disasters. The role of a well designed environmental hazard detection system based on a Wireless Sensor Network (WSN) is to continuously monitor and report the environment's status by sampling relevant physical parameters (e.g. temperature), but at a rate that can be adapted dynamically to the criticality of the current situation, so that precious energy is conserved as much as possible and communication bandwidth is not wasted, both preconditions that need to be met for a scalable WSN application. We have designed and built a small-scale prototype of such a WSN system for fire detection and monitoring based on inexpensive in-house developed wireless sensor nodes. These nodes combine an AVR Butterfly microcontroller demonstration kit with an Xbee wireless Zigbee transceiver. The emphasis of the work reported here is on the software designed for the fire hazard detection application. We discuss the embedded computing strategy developed for the in-field sensor nodes that allows them to adjust their mode of operation (i.e. their sampling and reporting rates) dynamically and in an autonomous manner depending on the area prevailing conditions. We also discuss the functionality of the software running on the central node (PC) that is used to initialize the WSN system, synchronize nodes, monitor their status by maintaining an active registry, adjust parameters at any time, inspect real-time plots of the incoming temperature reports of selected nodes to monitor emerging trends and patterns etc. Several examples of the end-to-end system's use are also presented and discussed.

Keywords: Fire hazard detection, wireless sensor network, embedded systems AVR Butterfly, Zigbee.

1 Introduction

Significant advances in embedded hardware and software technologies are driving down rapidly the cost of Wireless Sensor Networks (WSN) and allow large-scale WSN deployments for environmental monitoring applications to become a reality. There exist already several ongoing WSN projects in Europe and worldwide for assessing habitat changes, monitoring the evolution or predicting the risk of catastrophic phenomena such as floods [1], fires [2], earthquakes [3], volcano eruptions [4], etc.

Modeling wildfire behavior is a very challenging problem due to, among other reasons, the large number of time-varying parameters governing this complex phenomenon. However, since it is a very useful tool for fire management, significant efforts have been devoted over the last decade towards the development of effective fire evolution prediction models [5], [6]. In most cases, these models are designed to simulate fire spread and estimate linear intensity and flame length, but not temperature. In few cases, especially for modeling crown fire initiation, certain models have proposed sound algorithms to predict, for example, the time-to-temperature profiles above a spreading flame front [7].

However, since low cost digital temperature sensors (that can measure usually up to ~ 120 °C) have become commodity items, it becomes feasible to deploy WSN based systems for distributed fire detection and monitoring. Such a system can contribute to the on-time fire detection via distributed signal processing and data fusion algorithms and also help us calibrate periodically computer models used to predict the evolution of a fire front under different prevailing wind conditions in the affected area. These capabilities can be particularly important in the so called Wildland Urban Interface (WUI) regions surrounding our modern cities, where the catastrophic impact of a rapidly spreading fire can be tremendous to human lives and property, as it has been recently experienced in Australia, Los Angeles CA, USA and Athens, Greece.

Fire spread modeling depends on many factors, such as ground morphology, combustion fuel, moisture and most importantly on the wind speed and direction which may be changing rapidly. Nevertheless, there exist several software tools (such as FARSITE [8], FSE [9] etc.) that consider a geographical region of interest as a grid of “cells” and use algorithms to predict for every cell the time of arrival of the front of an erupted fire.. They are almost all based on the seminal work of Rothermel [10] which, given the type of fuel and the wind conditions, can estimate several local fire related properties, such as the flame length, fire intensity, etc for each cell when visited by the evolving fire front.

We have developed and present here a WSN-based end-to-end system prototype for reliable fire hazard detection and notification. Our system is based on really inexpensive sensor nodes developed in-house by students, combining an AVR Butterfly (BF) demonstration kit [11] with an XBee (XB) [12] ZigBee [13] wireless transceiver [12] on the same PCB (Printed Circuit Board). The BF is equipped with an Atmel Mega169 [16] 8-bit microcontroller with 16KB flash memory. The embedded software design allows for each sensor node to utilize different sampling and reporting rates depending on its current mode of operation (related to the assessed criticality of the detected event). This choice maximizes system alertness while also minimizing power consumption, which is an important parameter for outdoor WSN environmental monitoring type applications. Sensor nodes can change modes of operation, either autonomously, depending on the sensed temperature, or under operator control. An end-to-end fire monitoring application has been developed and tested in which a software component running at the host-PC can be used to monitor the status of the WSN, initiate and terminate sensor nodes operation, log received temperature reports, plot the reported values from selected nodes in real-time, issue commands to change the mode of operation or the parameters of selected nodes, identify nodes that got silenced or return back to operation after a silence period etc.

The prototype we have developed and tested was certainly small-scale, but the embedded software running on each node has been designed and implemented in embedded C and with a larger scale application in mind. If a wide-area WSN of temperature sensors is deployed in a field, it becomes feasible to use it as a mechanism to calibrate periodically fire spread prediction computer models (such as those implemented in FSE, FARSITE etc) and improve substantially their prediction performance. To achieve this goal we need to be able to correlate periodically temperatures reported by in field sensors with temperature predictions generated by computer simulation models for the same location and time step. However, the fire spread simulators give us only the spatiotemporal field of fire related parameters (e.g. flame length or fire intensity). To bridge the gap we have also developed a statistical signal processing method that allows us to transform the flame length field into a corresponding expected temperatures field for the same area [14]. This estimated temperatures field can now be compared to the WSN node temperatures field and the data assimilation system loop be closed by adjusting the predictive model parameters accordingly in order to reduce the prediction error. This work has been performed during our participation to the SCIER project [15], co-funded by the European Commission under the 6th framework of R+TD.

The rest of the paper is organized as follows: In Section 2 we present the sensor nodes architecture and functionality. We also present briefly the in-house hardware design of the sensor node and then in more detail the node's operation modes, the possible transitions and the embedded software design. In Section 3 we shift our attention to the design of the central node that controls the network and discuss the capabilities of a user friendly GUI tool developed for this purpose. In Section 4 we present representative cases of the use of the system in the field with data collected during our application field testing. Finally our findings are summarized and work in progress is outlined in Section 5.

2 Sensor Node Design

2.1 Sensor Node Hardware

We describe next the 2-layer PCB that we have developed in house to host the AVR BF and the XB modules and construct a low cost WSN node. The communication between the XB and the BF is achieved through an RS-232 serial connection. There is also an RS-232 serial port on the board through which a PC connection can be established. The serial connection can be active between the XB and BF or the BF and the PC. On board jumpers determine each time the connection mode. The BF can be programmed and exchange data with a PC through this port. A 9V battery is packed with the board to supply power. External power is an option too, but does not favor portability.

The sensor node is extensible since the user has access to two 8-bit I/O ports of the AVR microcontroller through appropriate connectors on the board. The microcontroller [16] is equipped with an 8-channel 10-bit ADC unit which is used for temperature sampling and a UART unit which implements the RS-232 serial communication with the XB module and the PC. A 512 bytes EEPROM and an 1KB

SRAM memory are available on-chip, while the clock rate can reach up to 16MHz providing an 1MIPS/MHz processing power. For the needs of our project the AVR chip was clocked at 8MHz.

2.2 Sensor Node Modes of Operation

The main tasks of the sensor node are: a) to collect temperature samples according to a predefined sampling period (SP) and b) to send the collected data back to the central node of the network according to a reporting period (RP). These two time periods are defined independently by the user, but the RP must be greater than the SP. Several samples may be collected and processed by the node, before a report (usually a summarizing statistic) is sent to the main node.

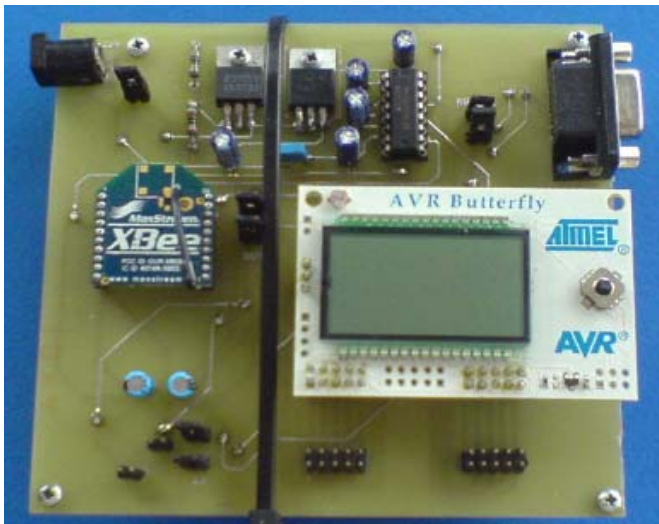


Fig. 1. The PCB of the wireless sensor node that includes an AVR Butterfly kit and an XBee ZigBee transceiver

The proper selection of the aforementioned period times is critical to the power dissipation of the node. For our application it is more power efficient to process a small number of samples and send a summarizing statistic to the central node since sending all measurements would drain the battery of the nodes and lead to network congestion. The AVR microcontroller on the sensor node is able to handle major processing tasks while in the same time sustaining a low power profile with its sleep mode features. The software of the sensor node is handled by the BF and the communication with the XB module is achieved through serial RS-232 connection between the two devices. The SP of the node must be adapted to the conditions in which is exposed, e.g. small SP on days with high temperatures.

The sensor node has 4 operation modes (states) which are:

- *Initialization Mode (I)*: In this state the central node performs initialization tasks on the network (e.g. sensor node discovery and registration etc.).
- *Low Risk Mode (LR)*: the node transits to this state after the initialization mode, when there is no obvious fire hazard and the sampling and reporting periods have their maximum values.
- *High Risk Mode (HR)*: When several temperature samples are higher than a predefined threshold (called Low Risk Maximum Temperature Threshold, LR_MAXT), the node transits to this mode and the time periods (report and sampling) become smaller. If the temperature falls quickly below the threshold, then we had a false alarm and the node transits back to Low Risk Mode.

Emergency Mode (EM): if the temperature continue to rise above the next predefined threshold (called High Risk Maximum Temperature Threshold, HR_MAXT), then the node transits to this mode, where a fire in the region is considered a certainty. The sampling and reporting periods get their minimum value.

The central node must be aware of the state of each sensor node, so every time a sensor node changes its state, it informs the central node with a report message.

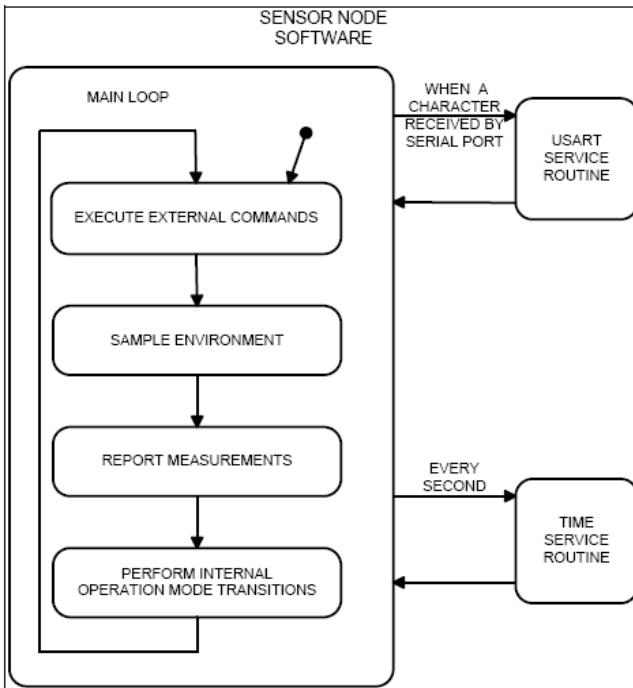


Fig. 2. The Statechart of the operation of the sensor node

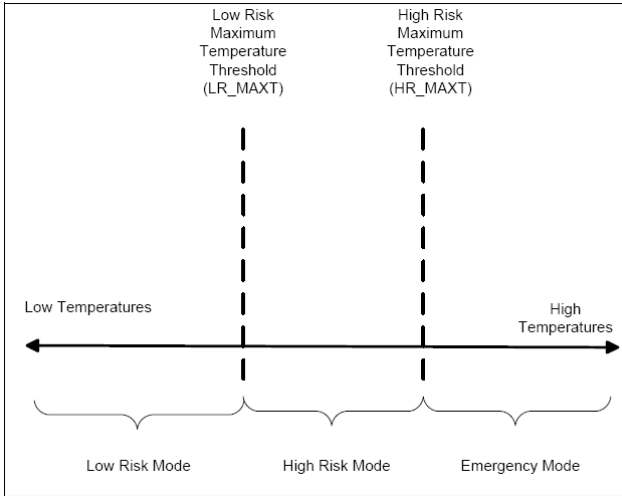


Fig. 3. The temperature range and associated thresholds of each mode of operation

The system operator may manually change the state of a sensor node or the value of the temperature thresholds and periods by issuing instruction messages to the network or to any individual node through the central node. The sensor node is able to receive these messages and reply back to the central node with reporting messages about its current state and the measured temperature. In Figure 2 can see that the first operation in the main loop that is being executed by the BF is to check for any incoming command messages. After that the node executes its normal operations, which are temperature sampling and reporting of a statistic of collected measurements. There is also a time service routine, that interrupts the main loop once every second and is used for time counting.

In Figure 4 we can see the possible transitions between the operation modes of the sensor node. The conditions triggering a mode transition are shown on the edges.. The transitions depicted with dashed lines are initiated by an instruction message from the central node.

After the initialization mode the node transits to the Low Risk Mode and starts taking temperature samples. The node can return to initialization mode only if the central node has sent the proper initialization mode instruction message. Every other transition occurs according to how the maximum value of the collected temperature samples compares to and the temperature thresholds shown in Fig. 3 at the end of a reporting period (RP). It is possible to use a different statistic (e.g. the median or the mean value) but the MaxValue has been found more effective in conducted experiments. The node can also be set up so that an immediate transition to the appropriate mode is performed as soon as an out of range sample is collected. This more reactive type of operation minimizes the system response delay at the expense of increasing its energy consumption (due to more mode transitions per unit time on average). There is also the case where the user forces a node to transit to a desired mode through a proper instruction message issued by the central node. These transitions are indicated with dashed arrowed lines in Figure 4.

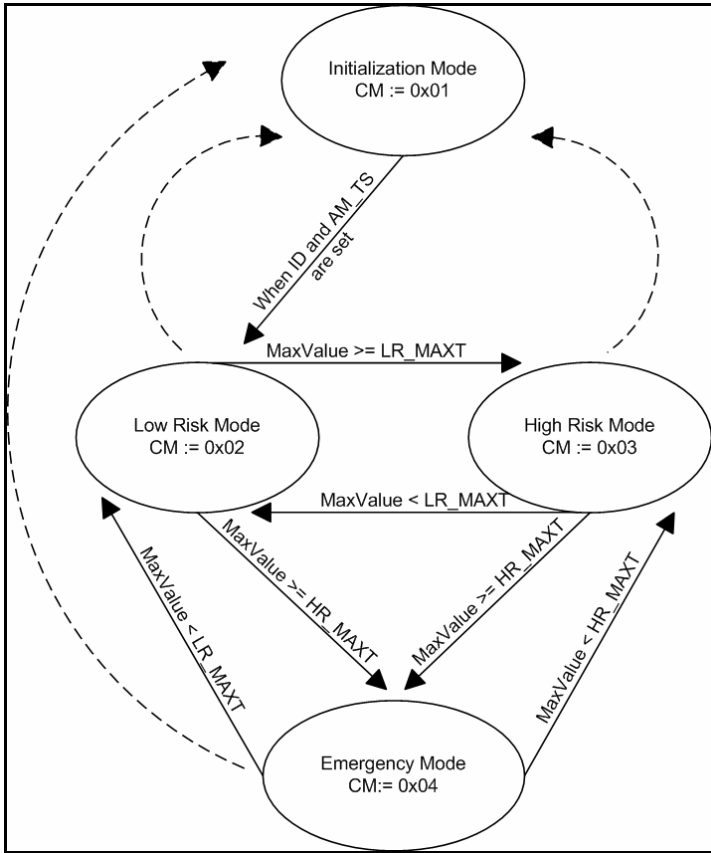


Fig. 4. The FSM describing sensor node modes and possible transitions

3 Central Node Design and Operations

The central node needs extra processing and storing capabilities than the AVR microcontroller on the sensor node can support. The solution that we adopted was to bundle together a sensor node and a PC into a central node. The central node, from a hardware aspect, is similar to the sensor node and uses its serial port to connect to the PC. The XB module is connected directly to the serial port while the AVR microcontroller is not used.

The operator of the network, communicates with the sensor nodes through a GUI software tool running on the PC. The hardware of the central node's board acts like a gateway between the RS-232 protocol used for communication with the PC and the ZigBee protocol used for wireless communication with the sensor nodes of the network.

As mentioned earlier, the operator has the ability to issue instruction messages through the central node to the sensor nodes. The reception and execution of an instruction message is the first action in the main loop of the sensor node as indicated

by the StateChart of Figure 2. The purpose of these messages is to change on demand the operation mode (state) or the value of one of the parameters that affect the functionality of the sensor nodes (e.g. the temperature thresholds).

We have defined a messaging scheme that is implemented on top of the ZigBee protocol, which uses its own predefined messaging system. All the details about routing and addressing messages to the network are handled by the ZigBee protocol, which is realized on each node by the XB device. Our messaging scheme is realized by the software running on the AVR microcontroller on each node.

As every message transmitted over the network, the instruction messages have a unique message identification code (which is 0x04) followed by the code of the parameter to be updated and its new value that the user wants to assign. The instruction messages may be broadcasted to all sensor nodes or addressed to specific nodes only. As mentioned, the addressing mechanism is handled by the ZigBee protocol.

Table 1. The message format of an instruction sent by the central node to sensor nodes

<i>Instruction code</i>	<i>Parameter code</i>	<i>Parameter value</i>
0x04	Code of the parameter we want to change	New value of the parameter
1 byte	1 byte	1 byte

The most important tunable parameters a sensor node uses are the temperature thresholds and its current mode of operation. They are initialized by the central node who can also update them during the system's operation. Other parameters are the address of the node and the current time (relatively to a global time reference).

The PC connected to the central node keeps track of all the sensor nodes connected to the network. The topology of the nodes is a logical star network, since all the instructions and reports are transmitted from or received at the central node. When the central node must communicate with a sensor node that is out of its immediate range, the message may reach this remote node through multi-hopping, assuming that each sensor node in the network (including the gateway) is within range of at least one other node. This routing scheme is supported by the ZigBee protocol and is suitable for outdoor WSN applications with a large number of sensors.

The main functions of the central node are listed below:

- *Network initialization:* after its activation, the central node starts tracking all the available sensor nodes. For each node discovered, the central node registers its address. The ZigBee protocol defines the global (64-bit) and the network (16-bit) address for each network node. The global address is unique and assigned by the manufacturer of each ZigBee transceiver. The network address may be changed dynamically during the operation of the network. In our application we chose to assign our own addresses to the sensor nodes and avoid retransmission of large bit sequences of the global or the network address. Each node discovered is assigned a decimal number as its node address, starting from 0. Every address is associated to the global address of the node's XBee module. After the registration of the all available sensor nodes, the central node

broadcasts the current time, so every sensor node is properly synchronized. Upon reception of the time reference, all the sensor nodes transit to the Low Risk mode and start sampling their environment.

- *Data acquisition*: At the end of an RP, the central node accepts report messages from all the active sensor nodes and stores them to the PC memory. The frequency of reception of these messages depends on the mode of operation of each sensor node and is an adjustable parameter.
- *Temperatures plotting*: The GUI software running on the PC can plot in real time the temperatures reported by selected active nodes.
- *Data storing*: All temperature reports received by the central node as well as the current mode information of each node are stored in files on the PC for further processing.
- *Network surveillance*: The central node periodically tracks all the active sensor nodes and updates a registry every time a new node becomes active or an active node becomes inactive or gets out of range. If new nodes are added to the network the central node must initialize them without affecting the operation of the other nodes.

3.1 Central Node GUI Design

All the software of the central node is running in the PC connected to the XB module, in contrast with the sensor node where the software is running on the AVR microcontroller. For the development of the central node software we chose Matlab [17]. One of the main advantages of this choice, was the ease of development of the GUI.

Through Matlab we gain access to the serial ports of the PC and so we could easily produce and send any kind of data to the attached devices, the XB module in our case. The XB has its own microcontroller which accepts instructions through its serial port in the form of AT commands. The Matlab code constructs these commands and sends them through the serial port to the XB.

The main task of the central node, except the initialization of the network, is the reception of the reporting messages from the sensor nodes and their display in real time using the GUI, so that the user can easily understand the state of the network (measured temperatures and active nodes). The central node must be constantly aware of all the available active nodes on the network, so the creation of a sensor node registry was essential.

Upon activation of the network, the central node records the address (global address) and the parameters of all the active nodes. Also there is a record for each node with all the reporting messages it has sent. This information is available to the user anytime s/he wishes to inspect it. Integration of new nodes is possible after the activation of the network, since the central node searches periodically for changes in the structure of the network. Also, all the nodes send a confirmation message back to the central node upon reception of an instruction message. So the central node becomes aware of non-responding nodes and can refresh accordingly its registry. During a fire it is known that a sensor node may exhibit periods of silence without necessarily been burned, so it is important to be able to dynamically reintegrate such nodes into the network as soon as they become available.

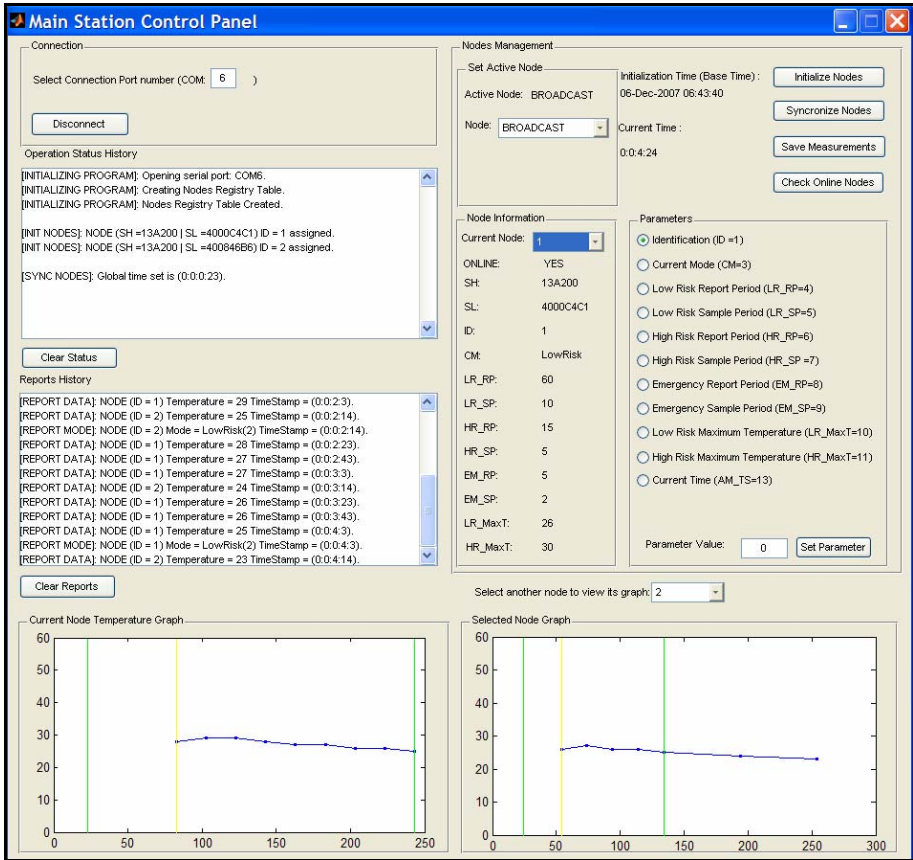


Fig. 5. The Graphical User Interface (GUI) used to control and monitor the system’s operation

In Figure 5 we can see the GUI of the central node that we have developed. In the upper left corner there is the button that connects the central node to the network. Also we can choose the serial port of the PC where the XB module is connected. In the “Report History” window there is an active listing of all the reporting messages received from the sensor nodes, while in the “Operation Status History” window messages can be found about the network operations. The user can inspect and change the parameters of any sensor node through the “Node Information” and “Parameters” windows. The node selected through the “Node” tab in the “Set Active Node” window is the receiver of all the instruction messages transmitted by the central node. We can see that there is also an option for broadcast-type transmission to all nodes. The current values of all the parameters of the selected node are presented in the “Node Information” window. All the tunable parameter names and codes are presented in the “Parameters” window. The user can update the default values for each parameter of the selected node. An important parameter for each node is also its current mode, which the user-operator can also change if so desired, by altering the value of the “CM” parameter. In the “Node Information” window the

current temperature threshold parameter values for the selected sensor node are given in degrees Celcius and the periods (sampling and reporting) in seconds. As we see the periods get smaller as the node transitions from the Low Risk towards the Emergency Mode.

Other important buttons in Figure 5 are the “Synchronize Nodes” button with which the user can assign the current time to all sensor nodes of the network. Also, the user can force the sensor node registry to be updated through the use of the “Check Online Nodes” button. Updating the sensor node mode and discovering changes in the active nodes set are two operations performed periodically by the system which could, however, be forced by the user-operator at any time through the central node, if so desired.

The two graphic windows at the bottom display the current and past temperatures of any two nodes on the network. The left window displays the temperature of the selected node through the “Set Active Node” window, while the right window displays information about the node selected through the tab just above it. This is useful when the user-operator wants to visually inspect or report differences in real-time between a reference node and some other node. The user can also be informed about the time that every temperature report was generated and about the time a mode transition occurred in the selected node. Mode transitions are marked with colored vertical lines (entering LR, HR, ER modes with green, yellow, red color respectively).

4 Application Field Testing

Now that the functionality of the central and the sensor nodes has been described we can provide some examples of the system’s testing in the field.

The area where the system will be deployed is a very critical factor because it affects the communication range of the nodes. According to the datasheet of the XB module, in open areas this range can be up to 100m, while in urban areas the range is limited to approximately 30m. Keeping in mind the way the ZigBee protocol defines communication paths and routes messages through multi-hopping, in order to reach far nodes of the network every node must have in its range at least one other node and so their distance must be less than 30m. Upon its placement every sensor node must be activated through the buttons provided on the BF module.

After the activation of the sensor nodes, the network is controlled by the GUI of the central node. If there are no isolated or malfunctioning nodes then the central node must discover all the sensor nodes and report them to its GUI and the registry. The user must initialize the network through the “Initialize Nodes” button. In this way every sensor node is assigned an address (which is associated to the global address of its XB module). Then the “Synchronize Nodes” button must be activated in order to provide the correct time information to all nodes. If the user wants to add new nodes to the network after its activation, then s/he has to press the “Check Online Nodes” button in order to update the registry of the central node. If the user wants to remove a node from the network, again the “Check Online Nodes” button has to be pressed and the central node will mark this node as offline in the registry. The central node checks periodically for new online or offline nodes, but as described above the user can also force this procedure at anytime.

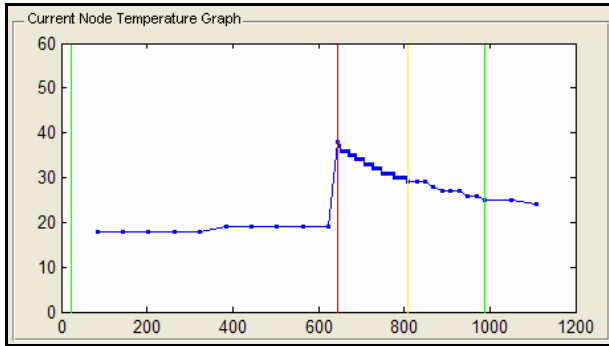


Fig. 6. Temperature reports received from a remote sensor node after a steep rise of its temperature. We observe the fast transition from Low risk to Emergency mode where the reporting period is much smaller. As the node cools down it enters the High Risk and finally the Low Risk mode and the reporting period is adjusted automatically back to normal.

If the user wishes to shutdown the network, then s/he must force all the sensor nodes to transit to initialization mode. In this way the nodes stop transmitting temperature reports, thus saving energy resources and are also in the proper state for a possible future reactivation of the network.

We can now present some usage scenarios of the system in real conditions. All reported tests took place at the University of Athens campus in the area surrounding our Department. In Figure 6 we raised the temperature of a sensor so that it transits directly from Low Risk to Emergency mode. The default RP in Low Risk Mode is 60 seconds, but if a sample exceeds the HR_MXAT threshold value, then the node immediately transits to Emergency mode. Same rule applies if a node has to transit from Low Risk to High Risk Mode. We can see that the RP is reduced to 15 seconds after the transition to Emergency mode (red vertical line) and then gradually transits to High Risk (30 seconds RP) and finally to Low Risk mode.

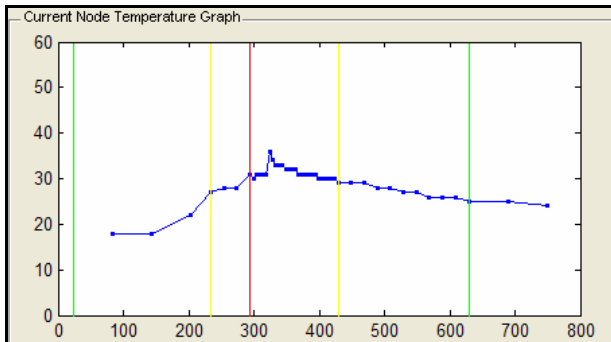


Fig. 7. Temperature reports received from a remote sensor node while increasing its temperature gradually and then cooling it down. Mode transitions are marked with vertical lines.

In Figure 7 we can clearly see that the remote sensor node passes from all the modes, of operation as the temperature gradually increases (yellow line for entering High Risk mode and red line for entering Emergency mode). Then again as we decrease the temperature the node returns back to Low Risk mode (green line) and the SP increases back to normal.

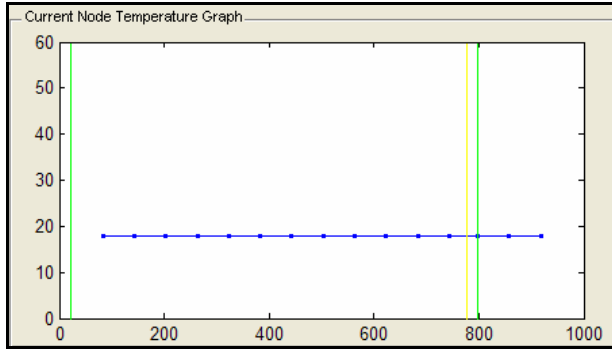


Fig. 8. Sensor node forced to enter the High risk mode returns back to Low Risk mode after confirming that the forced action is not supported by local temperature sampling

In Figure 8 we can see a situation in which while a node operates at the Low Risk mode and without any appreciable change in the sensed temperature from its environment it is forced to change its mode to High Risk, because the user-operator has sent it the corresponding instruction message using the GUI of the central node. This may have happened because the operator has already some reports from neighboring sensors entering the HR mode. We also see, however, that the node transits back to Low Risk mode at the end of the RP, since its local temperature measurements did not justify remaining for longer at the forced High Risk mode.

We have also mentioned that through the GUI the user can observe graphically and in real-time simultaneously the reports received by two nodes, through the two graphical windows shown at the bottom of the GUI (see Figure 5). In Figure 9 we can observe the temperature reports of two different nodes. We can see that every node is reporting back to the central node with each own reporting period (depending on its

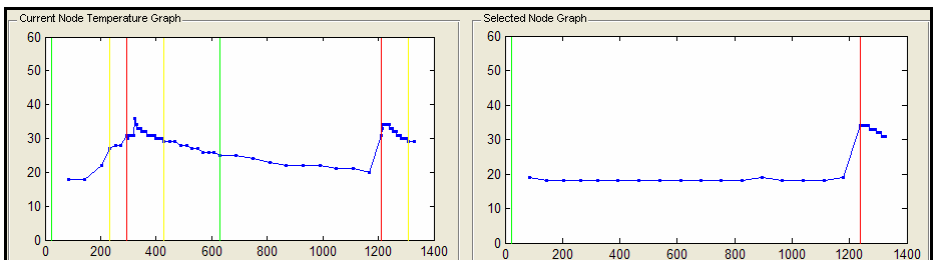
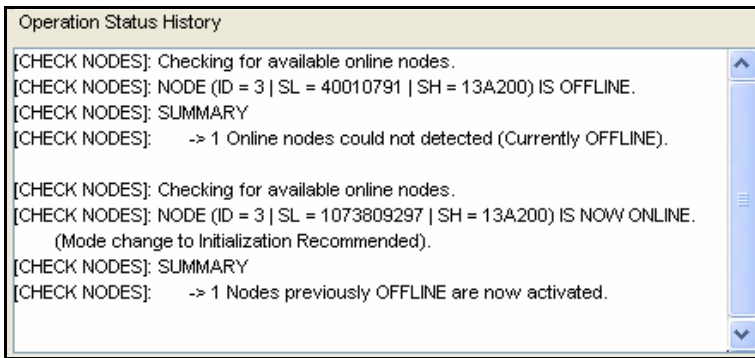


Fig. 9. Temperature report of two nodes presented in the GUI in real-time. The nodes are synchronized with the central node and use a common timeline.

mode), while the GUI can track and present in parallel all the mode transitions. We can also see that the time line is common for the two nodes, since they are synchronized with respect to the reference time provided by the central node.

The next function to demonstrate is the ability of the central node to track new active nodes or mark as offline nodes that have stopped transmitting or got for some reason out of the range of the network. It is well known that when a fire is passing the communication range of a sensor is affected and thus it may appear as offline temporarily. In Figure 10 we observe some messages in the “Operation Status History” window. In the 4th line we are informed about one node being not active anymore, while in the last line we are informed that this node has become active again. The central node through the use of its node registry can track previously active nodes that become active again, or register new nodes that join the network for the first time.



```

Operation Status History
[CHECK NODES]: Checking for available online nodes.
[CHECK NODES]: NODE (ID = 3 | SL = 40010791 | SH = 13A200) IS OFFLINE.
[CHECK NODES]: SUMMARY
[CHECK NODES]:    -> 1 Online nodes could not detected (Currently OFFLINE).

[CHECK NODES]: Checking for available online nodes.
[CHECK NODES]: NODE (ID = 3 | SL = 1073809297 | SH = 13A200) IS NOW ONLINE.
                (Mode change to Initialization Recommended).
[CHECK NODES]: SUMMARY
[CHECK NODES]:    -> 1 Nodes previously OFFLINE are now activated.
  
```

Fig. 10. Network status messages in the central node, tracking nodes that become inactive temporarily

5 Conclusions

An end-to-end system prototype for fire hazard detection and monitoring based on a WSN has been presented. The system was built using inexpensive sensor nodes that were designed in our laboratory (cost less than 50 euros/node). The embedded software for the sensor nodes allows them to adjust dynamically their mode of operation based on the local temperature trends sensed in their environment. Four modes of operation (Initialization, Low Risk, High Risk, Emergency) have been specified. As an approaching hazard makes the sensor nodes transition towards the Emergency mode, the sampling and reporting periods are decreasing so that the available energy is spent wisely when it is really needed and conserved during the rest of the time. A user friendly GUI has been designed for the central node (PC) that allows the user-operator to start, stop, synchronize, check the status of the network and change parameters (periods, thresholds) globally or for each individual node. The system has been tested in the field and examples of its usage are reported. Despite the fact that only a small-scale prototype has been built, the embedded software design was performed with network scalability in mind. By adjusting properly the reporting period a single central node can handle even large size networks without becoming a

communication bottleneck. Furthermore, the application design is modular so that multiple central nodes could be introduced if partitioning of the network control responsibilities is deemed necessary as the network grows bigger.

References

- [1] Chang, N., Guo, D.: Urban Flash Flood Monitoring, Mapping and Forecasting via a Tailored Sensor Network System. In: Proc. of the 2006 IEEE Int. conference ICNSC 2006, April 2006, pp. 757–761 (2006)
- [2] Doolin, D.M., Sitar, N.: Wireless Sensor for Wildfire Monitoring. In: Proc. of the SPIE, vol. 5765, pp. 477–484 (2005)
- [3] Suzuki, M., Saruwatari, S., Kurata, N., Morikawa, H.: A high-density earthquake monitoring system using wireless sensor networks. In: Proceedings of the 5th international conference on Embedded networked sensor systems, Sydney, Australia (2007)
- [4] Welsh, M., Werner-Allen, G., Lorincz, K., Marcillo, O., Johnson, J., Ruiz, M., Lees, J.: Sensor networks for high-resolution monitoring of volcanic activity. In: Proceedings of the 20th ACM symposium on Operating systems principles, Brighton, United Kingdom (2005)
- [5] Glasa, J., Halada, L.: Envelope theory and its application for a forest fire front evolution. *Journal of Applied Mathematics, Statistics and Informatics* (1), 27–37 (2007)
- [6] Asensio, M.I., Ferragut, L.: On a wildland fire model with radiation. *Int. Journal for Numerical Methods in Engineering* 54, 137–157 (2002)
- [7] Cruz, M., Butler, B., Alexander, M., Forthofer, J., Wakimoto, R.: Predicting the ignition of crown fuels above a spreading surface fire. *Int. Journal of Wildland Fire* 15, 47–60 (2006)
- [8] Finney, M.A.: FARSITE: Fire Area Simulator – Model Development and Evaluation. Forest Service, Research Paper, RMRS-RP-4 Revised (March 1998)
- [9] EUFIRELAB, a wall-less Laboratory for Wildland Fire Sciences and Technologies in the Euro-Mediterranean Region, Deliverable D-03-06, <http://www.eufirelab.org> (07/02/2008)
- [10] Rothemmel, R.C.: How to Predict the Spread and Intensity of Forest and Range Fires. USDA Forest Service Tech. Report, INT-143, Ogden, UT (1983)
- [11] ATMEL Corporation, AVR Butterfly Evaluation Kit User Guide, ATMEL (2005), http://www.atmel.com/dyn/resources/prod_documents/doc4271.pdf
- [12] MaxStream Inc., XBee OEM RF Modules Datasheet, Maxstream (2006), http://www.maxstream.net/hottag/index.phpht=/products/xbee/datasheet_XBee_OEM_RF-Modules.pdf
- [13] IEEE std. 802.15.4, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE (June 2006)
- [14] Manolakos, E.S., Manatakis, D.: Temperature field modeling and simulation of wireless sensor network behavior during a spreading wildfire. In: Proceedings of the 2008 European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland (August 2008)
- [15] SCIER: Sensor and Computing Infrastructure for Environmental Risks, <http://www.scier.eu>
- [16] ATMEL Corporation, 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash, ATMEL (2006), http://www.atmel.com/dyn/resources/prod_documents/doc2514.pdf
- [17] MATLAB and Simulink for Technical Computing, <http://www.mathworks.com>