Instituto Superior Técnico

Department of Computer Science and Engineering (DEI)

Software for Embedded Systems 2014/2015

Class Project
Sensor Network for Fire Detection

1 Introduction

Fire is a calamity that every year puts lifes and properties at risk. Better than systems to put out the fires are systems that prevent those fires or at least give an early warning of an iminent fire. One such detection system can be made by using wireless sensor networks.

By deploying a great number of sensors that detect smoke or favorable condition to the apearance of fires and by wireless connecting those sensors to a central place we can built a fire detection system that can give an immediate warning in case of fire.

2 Objective

Design an embedded system to be easily deployed in a forest and which can monitor temperature, humidity and detect smoke. That embedded system must be part of a large network which communicates data of its sensor readings to a server.

3 System specifications

The sensor network will use MICAz motes running the operating system TinyOS. The motes will be programmed in nesC.

There will be two types of nodes:

- Sensor Nodes;
- Routing Nodes.

Each node in the sensor network must be designed according to the functional specifications and non-functional specifications.

3.1 Functional specification

Each Sensor Node has an humidity sensor, a temperature sensor and a smoke detector. Each Sensor Node has also a GPS sensor device to detect its position.

Since nodes will be dispersed on a large field, they must use a Radio Frequency (RF) module to communicate. Thus, both the Sensor Nodes and the Routing Nodes have a RF module.

The Server will be a personal computer with a RF module. The Server will receive the values from the Sensor Nodes and will store all values received in file. Each line in the file must contain the timestamp of the date of the measurement, the id of the Sensor Node and the message sent in plain text.

Only a few nodes are close enough to the Server to communicate directly with it. Thus, messages sent to the Server, from far away nodes, must be routed through the different nodes until a transmitting node is sufficiently near the Server for it to receive the message.

Each Sensor Node will always communicate with the same Routing Node. If there are two or more Routing Nodes in the range of a Sensor Node, which Routing Node a Sensor Node communicates with is automatically defined when the nodes are deployed. Thus, when a Sensor Node want to send a message it always communicates with the same Routing Node. The Routing Node then sends the message to another Routing Node until the message arrives at the Server. The communication protocol used by the Routing Nodes is the flooding one where each Routing Node sends all the messages that it receives unless it has already sent that particular message. Thus, the Routing Nodes have a more powerful RF module to communicate longer distances

Note that the Server will always be close enough to at least one Routing Node. Also, each Sensor Node will be deployed close enough to at least one Routing Nodes. And each Routing Node will be deployed close enough to at least another Routing Node. Thus, there is always some path from a Sensor Node to the Server.

Values of humidity and temperature are taken by the Sensor Node every *Tmeasure* minutes and sent to the Routing Node. The Routing Node will then send the messages to other Routing Nodes until the message reaches the Server. When the Sensor Node detects smoke it immediately sends an alert message to the Routing Node and the message is routed immediately to the Server.

You must design your own network dissemination protocol instead of using the one provided by TinyOS.

Each Sensor Node has an id different from all the other nodes. When the Sensor Node is deployed it must register in the network by getting its Routing Node and sending to the Server its GPS coordinates.

3.2 Non functional specifications

Each Routing Node can handle a maximum of 100 Sensor Nodes. There is no limit on the number of Routing Nodes.

Since the nodes will not be connected to the power grid they must be designed taken into account power consumption.

The network must be designed with redundancy in mind. If one or more nodes fail, the network must continue to function. Whenever possible, failing of nodes must be prevented/warned.

The nodes must be easy to configure and easy to deploy.

The size of the ROM and RAM needed for your program should be minimal.

The system must be designed according to the best practices studied in Software for Embedded

Systems.

4 Simulator

In the first step of the design of the sensor network, an application that simulates the nodes, and their interaction with one another, must be designed in a host machine. This simulation must model as accurately as possible the physical implementation.

For the simulation it will be used TOSSIM (TinyOS SIMulator) which is a discrete event simulator for TinyOS. The simulation will thus consists of two modules:

- the sensor node programmed in nesC;
- the python script for the simulation.

4.1 Sensor and Routing nodes

The nodes will be programmed in nesC using the API for the MICAz mote. Each component must be clearly defined and well documented. Both Routing and Sensor nodes will be programmed in the same nesC source files. Routing nodes will have ids between 1 and 99. Sensor nodes will have ids equal or greater than 100.

The programmed node must adhere to the specifications described in Section 3.

4.2 Server

The server will be implemented as a node in the network with id equal to 0 (zero). The Server must adhere to the specifications described in Section 3.

4.3 Network

For nodes to be able to communicate with each other a network topology must be defined. In TOSSIM this is done by using two types of data:

- topology data;
- noise data.

The topology file to be used should consist of a set of lines where each line is of the form:

- <node1> <node2> <gain>

For example, if we have:
1 2 -55
this means that when node 1 transmits, node 2 receives the signal at a gain of -55dBm.

The noise trace file should consist of a set of lines where each line has a single numerical value. The number of lines should be at least 100. This will allow the creation of a statistical model of the hardware noise floor.

Both of these files will be read by the python script.

4.4 Debugging

In order to assert the correct functionality of the simulator, some debugging functionality must be present. Therefore, while the system is running, it must be possible to do what is described in Section 3.1 and also:

- simulate the event of a fire;
- simulate the malfunction of a Routing Node;
- simulate the malfunction of each module in the Sensor Node;
- check the content of the file where the logs are stored.

4.5 User interaction with the simulator

Define a set of commands to be used in the terminal where the simulation will run to interact with the simulation.

The commands will allow to control the server and to issue debugging orders to the simulation.

This can be done by always waiting for a command before a defined number of events in the simulator.

4.6 Testing

Taking into account the system specifications (see Section 3), prepare a procedure to test the simulator of the system. This procedure should be the one to be used in the project presentation (see Section 9)

Use a number of nodes that allow for full testing of the simulator. Their locations (network topology) should be set in order to simulate the case when a message from a node to the server passes by at least two other Routing Nodes.

5 Hardware modules

The necessary hardware modules must be chosen according to the specifications of the system. Search the web and choose possible hardware modules to implement the main required functions. Whenever possible, relevant technical specification of the modules should be obtained (e.g. type of input/output, power consumption). Note that the system consists of two different types of nodes.

When it's not possible to find a suitable hardware component a custom hardware component can be specified. In that case it must, at least, be specified:

- funcionality;
- inputs and outputs.

The software implementation used in the simulator to interact with the different hardware modules must match the specifications of actual hardware modules (e.g. type of input/output).

6 Support Material

All support material is on the course page. It's advisable for the students to go through the tutorials. In particular the TOSSIM tutorial.

7 What should be delivered

The submission of the project will consist in uploading a zip file through the fenix system. The zip file must consist of a report, the source code of the nodes and the source code of the simulator.

7.1 Report (no more than 10 pages)

The report should be in pdf format and must include at least the following items with the rationale for them when appropriate:

- General description of the system and enumeration of its features;
- System architecture (hardware and software architecture of the sensor node and the hardware/software interaction);
- Description of the different components of the sensor node;
- Network architecture (in particular how the sensor nodes interaction with each other);
- Deployment procedure and description of the node initialization;
- User manual of the simulator;
- Main problems found in implementing this system and solutions for those problems (when found);
- List of specifications not met and reasons for it.

7.2 Source code of the sensor node and the simulator

All files necessary to run the simulation, including the source code of the nodes and of the simulator, must be in a zip file. The zip file must also contain a text file describing all the steps to run a simulation.

8 Project milestones

March, 09 - Project presentation in Tagus (Lab class).

March, 11 - Project presentation in Alameda (Lab class).

Next weeks - Project development.

May, 1, at 23h59 - Deadline for submission of project in fenix.

Next two/three weeks - Project presentations and evaluations. Evaluations of "competitors" projects.

9 Project assessment

After the submission of the project, groups will be associated in clusters. The evaluation will occur on the lab classes and will have the following phases:

1. Project presentation – Presentation of the project, including the demonstration of the simulator and the presentation of the main architecture, design, and features of the system. (Oral presentation, no slides required.)

2. Assessment of the projects of the other groups in the cluster – Students must participate in all the presentations of their cluster to clarify any issues related with the projects of the other groups of their cluster.

3. Evaluation report – Short report in pdf (one page max.) about all the projects of the cluster – including own project of the group – stating what the group considers to be the main features and weaknesses of the projects of the cluster. These reports must be submitted in fenix no more than 48 hours after the end of the presentations of the cluster.

4. Review meeting – Review evaluation meeting with each group and the teacher. Therefore each group must participate in two evaluation sessions – Project presentation (all the groups of the cluster), and review meeting (group by group).