Instituto Superior Técnico

Department of Computer Science and Engineering (DEI)

# Software for Embedded Systems 2013/2014

Class Project
**Sensor Network for Animal Husbandry**

## 1 Introduction

The challenges faced by modern agriculture have never been greater. There has always been a need for livestock producers to be able to observe their animals as often as possible. But managing a large quantity of animals while at the same time letting them be as loose as possible is not an easy task. Recently, we have seen the rise in the use of wireless sensor networks to help in the management of animals.

By putting a sensor node in an animal we can control their location, their iteration with each other, or their health, among other parameters. And we can have continuous, round the clock, observation of the animals and therefore we can intervene as soon as possible if some situation arises.

The purpose of this project is to design such sensor nodes in a network.

## 2 Objective

Design an embedded system to be deployed in an animal, as a collar, through which a person can monitor its location and control the amount of food intake. That embedded system must be part of a large network which communicates data of its sensor readings to/from a server.

## 3 System specifications

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

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

### Functional specification

Each sensor node has some sensor device to detect its position in a large field. The error should be less than 50 meters.

Throughout the field there are some feeding spots. In these feeding spots, when an animal enters, the sensor node in the animal communicates with the machine such that the machine will dispense the right amount of food to that particular animal. The feeding spot only communicates with the sensor node in the animal and dispenses the amount of food given by the sensor node. The feeding spot has no memory, whatsoever.

Since nodes will be dispersed on a large field, they must use a Radio Frequency (RF) module to communicate. To communicate with the sensor nodes in the animals, users have laptops that are able to communicate using RF radio. At a certain time, only a few nodes are close enough to the laptop to communicate directly with it. Thus, messages sent to the laptop, from far away nodes, must be routed through the different nodes until a transmitting node is sufficiently near the laptop for it to receive the message. The user with the laptop will always be close enough to at least one node.

Using the laptop, users should be able to:

- check the location of all the animals that are part of that cluster;
- check the previous known location of the animals that are not part of that cluster;
- check how much each animal of that cluster has eaten from the feeding spots;
- check how much food is left in the feeding spots;
- update the amount of food each individual animal should eat daily from the feeding spots;
- update the amount of food in each individual feeding spot.

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

### Non functional specifications

The system must be able to handle at least 10.000 animals and 100 feeding spots.

Since the sensor 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 sensor 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 sensor 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.

### Sensor node

The sensor node will be programmed in nesC using the API for the MICAz mote. Each component must be clearly defined and well documented.

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

### Laptop server

The server will be implemented in the same python script used in the simulation. The server must adhere to the specifications described in Section 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.

### 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 and also:

- simulate the proximity of an animal to a feeding spot and simulate its feeding. beginning of the simulation;

### 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.

### 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.

## 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).

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 sensor node and the source code of the simulator.

### 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);
- 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.

### Source code of the sensor node and the simulator

All files necessary to run the simulation, including the source code of the sensor node 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, 10 - Project presentation in Tagus (Lab class).

March, 12 - Project presentation in Alameda (Lab class).

Next weeks - Project development.

**May, 2, 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).