IoT Projects - 2021/2022

May 20, 2022

1 General Rules, Grading and Deadlines

Grade composition of the entire course:

- 25 out of 30 points are assigned based on the written exams
- up to 4 points are assigned based on the home projects done during the hand-on activities
- up to 4 points are assigned based on final projects

General rules:

- Projects are NOT mandatory. One student can decide not to take any project; in this case, the maximum grade he/she can get will be 25/30.
- Projects can be developed in groups of maximum 2 people.
- The project deadline is September 1st 2022

IMPORTANT 1: ONLY THE PROJECTS REGISTERED ON THE ONLINE FORM WILL BE CONSIDERED. LATE REGISTRATION WILL NOT BE ACCEPTED.

IMPORTANT 2: THE DELIVERY IS ONE AND ONLY ONE.

IMPORTANT 3: REGISTRATION IS NOT BINDING. IF YOU REGISTER FOR A PROJECT AND THEN DECIDE AFTERWARDS YOU DON'T WANT TO DELIVER IT, THAT'S OK.

2 Proposed Projects

The following projects proposal are thought of being implemented with the tools seen during the hands-on lectures.

You have to deliver the following items:

- Complete source code of the project
- A self-explanatory log file showing that your project works. Try to be as detailed as possible when preparing the log file (i.e., use debug/print statements in all the crucial phases of your project)
- Project report (max. 3 pages), which summarizes your approach in solving the problem, including figures when needed. Don't include source code in the project report.

Projects will be evaluated based on the rationale and technical depth of the design choices, the correctness and the organization of the source code, and the project report's clarity.

2.1 Project 1. Smart Bracelets

You are requested to design, implement and test a software prototype for a smart bracelet. The bracelet is worn by a child and her/his parent to keep track of the child's position and trigger alerts when a child goes too far. These bracelets are becoming more and more popular, with some commercially available prototypes on the market.

The operation of the smart bracelet couple is as follows:

1. **Pairing phase**: at startup, the parent's bracelet and the child's bracelet broadcast a 20-char random key used to uniquely couple the two devices. The same random key is pre-loaded at production time on the two devices: upon reception of a random key, a device checks whether the received random key is equal to the stored one; if yes, it stores the address of the source device in memory. Then, a special message is transmitted (in unicast) to the source device to stop the pairing phase and move to the next step.

2. **Operation mode**: in this phase, the parent's bracelet listen for messages on the radio and accepts only messages coming from the child's bracelet. The child's bracelet periodically transmits INFO messages (one message every 10 seconds), containing the position (X, Y) of the child and an estimate of his/her kinematic status (`STANDING`, `WALKING`, `RUNNING`, `FALLING`).

3. **Alert Mode**: upon reception of an INFO message, the parent's bracelet reads the content of the message. If the kinematic status is FALLING, the bracelet sends a FALL alarm, reporting the position (X, Y) of the children. If the parent's bracelet does not receive any message, after one minute from the last received message, a MISSING alarm is sent reporting the last position received.

Continue in the next page with Requirements.

2.1.1 Requirements

1. Implement the prototype with the O.S. of your choice (including application logic, message formats, etc...). The X, Y coordinates can be random numbers, and the kinematic status should be randomly selected according to the following probability distribution:

$$P(STANDING) = P(WALKING) = P(RUNNING) = 0.3$$

and

$$P(FALLING) = 0.1$$

2. Simulate your implementation with 2 couples of bracelets at the same time. Your simulation should demonstrate that your code follows the design requirements. If you simulate using TOSSIM, you can emulate that a node goes out of range by turning it off (i.e., calling `mote.turnOff()` in python). In Cooja, you can simply move a node out of the communication range of the other one.

Optional Attach your Tossim simulation to Node-red: alarm messages should be transmitted on the serial port, and their output should be readable on the Node-Red dashboard.

2.2 Project 2. The (hidden) Terminal

The hidden terminal problem is common to all wireless communication technology. In particular, the IEEE 802.11 (WiFi) standard uses a virtual carrier sensing methodology to overcome the problem: the RTS/CTS exchange can be used to inform hidden terminals of the presence of transmissions, thus helping in avoiding collisions.

In this project, you are requested to implement a lightweight version of the RTS/CTS protocol in a Wireless Sensor Network operating on the IEEE 802.15.4 protocol. In particular, the following points should be implemented:

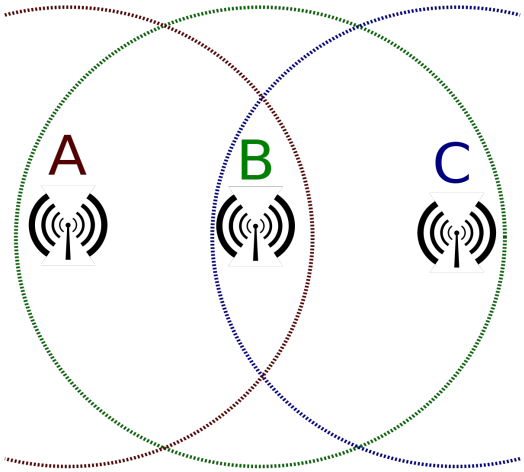

Figure 1: Hidden terminal example.

- Use the WSN simulator of your choice (TOSSIM or COOJA) to simulate a network with 1 base station and 5 nodes, with at least 2 hidden terminals.
- Implement a baseline transmission protocol: each node randomly transmits data to the base station. Each node transmits sequentially numbered packets according to a non-deterministic traffic distribution (for instance: Poisson distribution with a parameter Lambda at your choice).
- At the base station, compute each node's Packet Error Rate. You should tune simulation parameters so that the hidden terminal problem is highlighted.

- Implement the RTS/CTS mechanism. Each node willing to transmit a packet transmits first an RTS message to the base station. Nodes receiving the RTS message will stop their transmission for a fixed time of X seconds. The base station replies with a CTS message. Nodes receiving the CTS message will stop their transmission for a fixed time of X seconds.

- At the base station, compute each node's Packet Error Rate in this case. Change the value of X to Y and repeat the test. (X and Y of your choice)

Final report The final report must include the full TinyOS code, the log of the simulation (.log file in case of TOSSIM and some relevant screenshots for Cooja) and a report that explains the approach and the assumptions done.