IoT Projects - 2020/2021

Edoardo Longo
edoardo.longo@polimi.it
Fabio Palmese
fabio.palmese@polimi.it

March 17, 2021

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 2021

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.

WHEN YOU DELIVER THE PROJECT YOU CANNOT RE-DELIVER THE PROJECT FOR A BETTER GRADE. Send an email to `edoardo.longo@polimi.it` to deliver the project.

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.

Students willing to take the project assignment must choose among the project proposals listed in the following section or propose an original project, using the online form available at `https://forms.gle/UKPMdT7HdnM4eJoG7` by **May 28th, 2021**.

For original projects, write to `edoardo.longo@polimi.it` describing the project you intend to implement.

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. Keep your distance

You are requested to design and implement a software prototype for a social distancing application using TinyOS and Node-Red and test it with Cooja. The application is meant to understand and alert you when two people (motes) are close to each other. The operation of the software is as follow:

- Each mote broadcasts its presence every 500ms with a message containing the ID number.
- When a mote is in the proximity area of another mote and receives 10 consecutive messages from that mote, it triggers an alarm. Such alarm contains the ID number of the two motes. It is shown in Cooja and forwarded to Node-Red via socket (a different one for each mote).
- Upon the reception of the alert, Node-red sends a notification through IFTTT[1] to your mobile phone.

Use at least 5 motes. Start the simulation with all the mote far away from each other and move them with the mouse testing different configurations.

[1]IFTTT is a service to create conditional statements. It can be easily integrated with Node-Red, and it's an easy way to send notifications to your smartphone. This is a useful reference: `https://wiki.instar.com/Advanced_User/Node-RED_and_IFTTT/`, but don't be afraid to discover it by yourself. It's very easy and enjoyable.

2.2 Project 2. Smart Lights WSN - up to 4 points

Implement a network of smart lights, then program the network to visualize luminous pattern inside the predefined topology. Assign a fixed address to each node (e.g., use TOS_NODE_ID when using TinyOS), the routing will be defined a priori as described below.

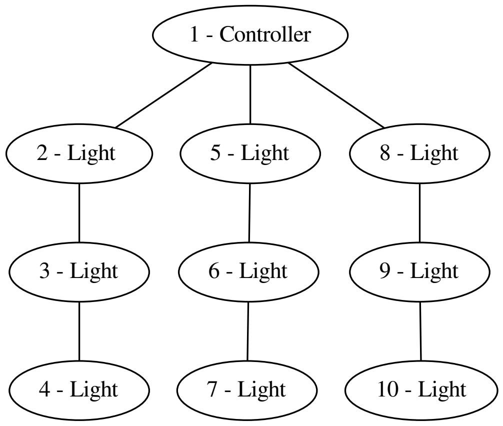

Figure 1: Network topology.

The requirements of this project are:

1. Create a topology with at least 10 nodes (1 controller node and 9 light nodes), as illustrated in Figure 1.

2. Addressing: use the same addressing idea as illustrated in Figure 1; for example, the controller will have the address 1 etc. You can modify addressing but they must always have a hierarchy to simplify the routing phase.

3. Routing: each node will have its predefined routing table. Multi-hop communication has to be considered (only nodes 2, 5, 8 are in the transmission range of the controller). The controller node knows that messages addressed to nodes from 2 to 4 have to be sent toward node 2, messages to node from 5 to 7 toward node 5 etc. Light nodes instead

have to forward packets not addressed to themselves. For example, when node with address N receives messages addressed to nodes with address greater than its own, it has to forward them to node N+1. While messages addressed to nodes with address lower than its own have to be forwarded to node N-1. Remember to take care of not re-forward messages to the same node from which the message is received.

4. The controller node is programmed with some pre-defined pattern to switch on the smart lights. It periodically sends unicast commands to the lights to switch them on and off.

5. Show at least 3 patterns (e.g. a cross switching on node 2,4,6,8,10; a triangle switching on node 6,4,7,10) that cyclically change. Remember that, from one time to the other, all the nodes have to be switched off before switching them on (it's ok to send a switch off message to every node instead of using broadcast packets).

You are free to use a different topology or different patterns from the proposed ones. Simulate the program in Cooja, showing the LEDs on the nodes.