IoT Projects - 2022/2023

Antonio Boiano
antonio.boiano@polimi.it

May 3, 2023

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 hands-on activities (3 challenges)
- up to 4 points are assigned based on the final project

General rules:

- Projects are NOT mandatory. One student can decide not to take any project/challenge
- Projects can be developed in groups of maximum 2 people.
- The project deadline is September 1st, 2023

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 `antonio.boiano@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.office.com/e/ZKvmmdurgP` by **June 4th, 2023**.

For original projects, write to `antonio.boiano@polimi.it` describing the project you intend to implement and wait for approval before filling the form.

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 (3-4 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. Lightweight publish-subscribe application protocol

You are requested to design and implement in TinyOS a lightweight publish-subscribe application protocol similar to MQTT and test it with simulations on a star-shaped network topology composed of 8 client nodes connected to a PAN coordinator. The PAN coordinator acts as an MQTT broker.

The following features need to be implemented:

1. Connection: upon activation, each node sends a CONNECT message to the PAN coordinator. The PAN coordinator replies with a CONNACK message. If the PAN coordinator receives messages from not yet connected nodes, such messages are ignored. Be sure to handle retransmissions if msgs get lost (retransmission if CONN or CONNACK is lost).

2. Subscribe: After connection, each node can subscribe to one among these three topics: TEMPERATURE, HUMIDITY, LUMINOSITY. In order to subscribe, a node sends a SUBSCRIBE message to the PAN coordinator, containing its node ID and the topics it wants to subscribe to (use integer topics). Assume the subscriber always use QoS=0 for subscriptions. The subscribe message is acknowledged by the PANC with a SUBACK message. (handle retransmission if SUB or SUBACK is lost)

3. Publish: each node can publish data on at most one of the three aforementioned topics. The publication is performed through a PUBLISH message with the following fields: topic name, payload (assume that always QoS=0). When a node publishes a message on a topic, this is received by the PAN and forwarded to all nodes that have subscribed to a particular topic.

4. You are free to test the implementation in the simulation environment you prefer (TOSSIM or Cooja), with at least 3 nodes subscribing to more than 1 topic. The payload of PUBLISH messages on all topics can be a random number.

5. The PAN Coordinator (Broker node) should be connected to Node-RED, and periodically transmit data received on the topics to Things-

peak through MQTT. Thingspeak must show one chart for each topic on a public channel.

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), the Node-Red code export (include also the details in case extra packages have been used), the ThingSpeak public channel URL, and a report that explains the approach and the assumptions done.

2.2 Project 2. LoraWAN-like sensor networks

Implement and showcase a network architecture similar to LoraWAN in TinyOS. The requirements of this project are:

1. Create a topology with 5 sensor nodes, 2 gateway nodes and one network server node, as illustrated in Figure 1.

2. Each sensor node periodically transmits (random) data, which is received by one or more gateways. Gateways just forward the received data to the network server.

3. Network server keeps track of the data received by gateways, taking care of removing duplicates. An ACK message is sent back to the forwarding gateway, which in turn transmits it to the nodes. If a node does not receive an ACK within a 1-second window, the message is re-transmitted.

4. The network server node should be connected to Node-RED, and periodically transmit data from sensor nodes to Thingspeak through MQTT.

5. Thingspeak must show at least three charts on a public channel.

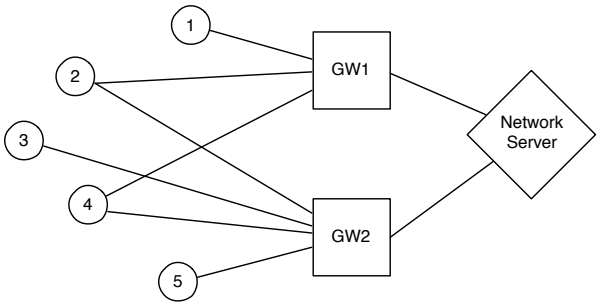

Figure 1: Network topology.

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), the Node-Red code export (include also the details in case extra packages have been used), the ThingSpeak public channel URL, and a report that explains the approach and the assumptions done.

2.3 Project 3. Beacon-enabled mode with ESP32 Clients

You are requested to implement using Wokwi and Node-Red a logic similar to the Beacon Enabled mode used by 802.15.4 protocol. The Superframe will have a Collision Access Part and a Collision Free Part. The Collision Access Part will be used by the clients willing to join the network to send an Association message, the Collision Free part will be used by the devices to transmit data. The communication is implemented using the MQTT protocol and a single topic for transmitting messages. At time 0 all the clients are Connected and Subscribed to one common topic (You can use one of the free public brokers available online i.e. broker.hivemq.com on port 1883, choose a personal topic hard to guess).

- Deploy on Node-Red an MQTT client which will act as PAN coordinator. The PAN coordinator client will send periodically to all the clients a beacon containing information on the superframe (i.e. Frame duration, collision-access part duration, collision-free part duration, information on the slot reserved for each client registered, duration of the Beacon Interval)

- Deploy on Wokwi 4 MQTT Clients using ESP32, which will act as Reduced-Function Devices (RFD). Each client should respect the slot assignment as specified in the beacon from the PANC.

    1. 2 Clients should use the DHT22 sensor and send the humidity percentage value randomly to one of the two actuator clients.
    2. 2 Clients will act as an actuator, they will receive the humidity value and based on the received value, will regulate the Luminosity level of an LED (using PWM).

- The RFD Clients must wait for a Beacon message to know the transmission interval structure (CAP, slots, sleep). For the first time a beacon is received, the RFD client should send a registration message to the PAN coordinator in the Collision Access Part. The PAN Coordinator will hence assign one slot for each node correctly registered. (if a new node has not a slot, sending registration will let the PANC add a slot in the CFP)

Final Report The final report must include the full Wokwi code for each client (4 different projects) as a .txt file, share also the project link of your 4 clients (4 links), the Node-Red code json export (include also the details in case extra packages have been used), and a report that explains the approach and the assumptions done. Describe your beacon division logic and how the slots have been defined. Deeply discuss on the Wokwi project logic by adding comments in the code and by commenting in the report. Also, describe the node-red nodes and their meaning.