Internet of Things, a.a. 2021-2022
Projects' proposals

Lecturers: Prof. Luciano Bononi, Prof. Marco Di Felice
{luciano.bononi, marco.difelice3}@unibo.it

May 3, 2022

General rules

- **HOW** The project can be done individually or in groups of maximum 3 units.
- **HOW** The project can implement one of the two drafts described below, or a new draft proposed by the student/group. In the first case (draft below), the system specifications can be customized/extended, however the overall complexity must not be reduced significantly. In the second case (draft from students), the proposal must be approved by Prof. Di Felice or by Prof. Bononi. Originality of the topic is not matter of evaluation. However, every project must implement the IoT data pipeline presented during the IoT course, i.e. data generation and processing on the edge device (e.g. the ESP32), data acquisition (e.g. via HTTP/MQTT/COAP protocols), data management (e.g. via time-series database), data visualization (e.g. via a `GRAFANA` dashboard), data processing and analytics.
- **HOW** The projects must be submitted exclusively via the Virtuale platform (Section: Project Submission). Submissions via emails or Teams will not be considered.
- **WHEN** Submit the project when ready. However, we will check the submissions on Virtuale at (the midnight of) these dates: 15 June 2022, 1 July 2022, 15 July 2022, 15 August 2022, 15 September 2022, 15 October 2022, 15 November 2022, 15 December 2022, 15 January 15 2022, 15 February 2022. All submissions found on Virtuale will be considered for the project discussion, that will be scheduled by the next two weeks; instructions about date and location of the discussion will be sent by email.
- **WHAT** Submit a zip file containing all the source code of the project and a technical report describing the project design and implementation. Sections of the report: Introduction, Project's Architecture, Project's Implementation, Results. The report must be written in English.

- **WHAT** In addition to the technical report, the student must prepare a presentation (with slides) of the project for the oral discussion. The slides must be submitted on Virtuale (Section: Project Presentation), at least one hour before the discussion.
- **WHAT** The oral discussion consists of the illustration of the project and a demo (using the students' devices). Questions about the theory of the course can be posed on topics related or unrelated to the project.
- **NOTE** All members of the group must be present during the discussion and must be aware of 100% of the project.

1 Proposal 1 (IoT-Based Indoor Air Quality Monitoring System)

The project consists in the deployment of an IoT application for smart home scenarios, including functionalities of IoT-based monitoring of indoor environmental parameters such as temperature, humidity and gas concentration, data collection and data forecasting. More in details, the system must include the following components:

- **IoT smart device**, constituted by a prototyping board (e.g. ESP32) connected to some environmental sensors. The set of sensors must include: outdoor temperature, outdoor humidity, gas leakage (e.g. carbon Monoxide) detection (see Section 1.2). The IoT device must support these functionalities:
  1. Acquisition of sensor values (temperature, humidity, gas concentration) every `SAMPLE_FREQUENCY` seconds.
  2. Transmission of sensor values to a remote back-end over a Wi-Fi connection, by using a messaging protocol among the ones presented during the course (MQTT, COAP, or HTTP). In addition to the sensor values, the IoT device must also transmit: (i) the WiFi signal strength of the wireless link to the router; (ii) the Air Quality Index (AQI), computed according to the details provided later in this Section; (iii) the GPS position and id of the device, both hard-coded in the firmware.
  3. Possibility to change at run time the protocol used to transmit the sensor, Wifi RSS and AQI data. At least two protocols among the ones presented during the course (MQTT, COAP, or HTTP) must be supported (e.g., CoAP and MQTT).
  4. Possibility to tune at run-time the configuration of these parameters: `SAMPLE_FREQUENCY`, `MIN_GAS_VALUE`, `MAX_GAS_VALUE`. The parameters' setting must be managed by using an IoT messaging protocol presented during the course (MQTT, COAP, or HTTP). Sup-

porting multiple protocols for the configuration change operations is not requested (while it is mandatory for the data transfer).

5. Computation of the AQI metric as follows:

$$AQI(t) = \begin{cases} 0 & \text{if } avg \geq \text{MAX\_GAS\_VALUE} \\ 1 & \text{if MIN\_GAS\_VALUE} \leq avg < \text{MAX\_GAS\_VALUE} \\ 2 & \text{otherwise} \end{cases} \tag{1}$$

where *avg* is the average of the gas sensor value, computed over the last 5 measurements (moving average window).

- **Data Proxy**, constituted by a software application running on a non-IOT device (can be your laptop or a Raspberry device). The back-end is in charge of: (i) acquiring all values transmitted by the IoT edge device by supporting the same messaging protocols of the device (e.g. MQTT, CoAP or HTTP); (ii) saving all data on the `INFLUX` database, setting the id and GPS position as tags, and the rest of data as field values.

- `Data Management System`, represented by `INFLUX` and `GRAFANA` tools. They can installed on the same device hosting the Data Proxy. The `INFLUX` tool must be configured in order to store all time-series produced by the IoT device. A `GRAFANA` dashboard must be setup, by creating a separate graph for each data flow available in `INFLUX`, i.e.: temperature, humidity, gas concentration, AQI, WiFi RSSI. An alert should be triggered (in `INFLUX` or `GRAFANA`) in case $AQI(t) \geq 1$.

- `Data Analytics`, represented by a separate software application, running on the same device hosting the Data Management System. The module is in charge of: (i) forecasting the value of temperature in the next X seconds; (ii) forecasting the value of humidity in the next X seconds; (ii) forecasting the value of the gas sensor in the next X seconds (X=defined by users or developers). The time-series of the predicted values must be saved on `INFLUX` and shown on the `GRAFANA` dashboard as well; e.g. by using two lines for the temperature graph, one related to raw measurements, one to predictions. One or multiple techniques for time-series forecasting can be developed and compared.

In addition to the software deployment, a focus of the project in on the performance evaluation of the data acquisition and analytics components. Regarding the first, the students must perform a performance evaluation computing the average delay and packet delivery ratio for each of the protocols supported by the IoT smart device for the data acquisition. Regarding the data analytics, the students must compute the average Mean Square Error (MSE) of the forecasting algorithm(s). The experimental results presented as graphs or tables (showing average values and confidence intervals) must be included in the technical report.

1.1 Additional Components

- +1 point. Develop an additional component to retrieve the average outdoor temperature value at your current location from an Open API Weather service. For instance, the `meteo-stat` Python library can be used (https://github.com/meteostat/meteostat-python). The outdoor temperature values must be stored through `INFLUX` and displayed through `GRAFANA`.

- +1 point. Develop a Bot Telegram component to receive alerts and periodic reports with average sensor values.

- (for Laude only). Deploy a Web dashboard to allow the system administration to monitor multiple air quality stations at the same time. Through the Web dashboard, the user must be able to register a new IoT air quality monitoring device, to configure its parameters, to visualize its position on a map, and to access the sensor data through a link to the `GRAFANA` dashboard.

1.2 Technologies

There are no constraints on the programming languages and libraries; students are free to deploy the system by using any programming language, and their choice is not matter of evaluation. Regarding the prototype, we suggest using the ESP32 board plus DHT11 sensor (for sensor/humidity) and MQ-2 sensor for gas monitoring. The latter are quite inexpensive devices. Algorithms for data forecasting can be decided by students (e.g. `FB Prophet`).

2 Proposal 3 (IoT Bridge Platform)

The project consists in the deployment of a software platform able to acquire sensor data flows using different protocols (MQTT, CoAP, HTTP) and with advanced functionalities of **protocol bridging** and of **data persistence**. The project is inspired by the PONTE[1] tool. It allows acquisition of IoT data flows in input, and creation of new/processed data flows in output. More specifically, the framework must be able to receive data from generic IoT data sources, supporting these protocols:

- **MQTT**: in this case, the user must set the broker configuration (e.g. network address, port, username, password) and the topic of interest;

- **CoAP**: in this case, the user must set the URL of the CoAP server, the port and the messaging mode (confirmable/non confirmable);

- **HTTP**: in this case, the user must set the URL of the HTTP server, the network port and the HTTP primitive (GET/POST).

[1] https://www.eclipse.org/ponte/

We assume that -regardless of the protocol- the data format is known and consists of a JSON document, with fields `sensorType` and `value` where the latter is the measured value produced by the sensor. The user can decide the action to apply to the IoT data flow in input, by selecting one of the three options below:

- **Data visualization**. Print the JSON documents acquired in real-time.
- **Data aggregation**. Print the statistical features of the sensory data, such as the maximum, minimum, average and standard deviation, computed every n observations, where n is a tunable parameters.
- **Data storage**. Store the data on the `INFLUX` database. In this case, the user must provide the URL of the device hosting the `INFLUX` instance (can be the same machine hosting the framework), the access token, name of the bucket, and additional tags. As a consequence of this action, the time-series of the flow are automatically sent to `INFLUX` and stored there.
- **Data bridging** towards other protocols. In this case, the user must specify the destination protocol (must be different from the acquisition protocol) as well as the above-mentioned protocol-specific parameters. The framework produces a data-flow in output using the target protocol. The framework must be orthogonal compared to the input/output protocols, i.e. be able to acquire data in input in any protocol and to produce an IoT flow in output in any other protocol.

All the functionalities must be enabled through a **CLI (Command Line Interface)**. Example of command:

```
./bridgeIoT -protocol mqtt -host 130.136.2.70

            -topic temperature -command visualize
```

Through the command above, the user is allowed to read data from an MQTT source and to display the JSON documents in real-time. Through the command:

```
./bridgeIoT -protocol mqtt -host 130.136.2.70

            -topic temperature -command aggregate -n 5
```

the user is allowed to read data from an MQTT source and to display the statistical values computed over a window of 5 measurements. Similarly, by issuing this command:

```
./bridgeIoT -protocol mqtt -host 130.136.2.70

    -topic temperature -influxconf influx.json  -command save
```

the user can read the data from an MQTT source and save them into the `INFLUX` database. The configuration parameters of `INFLUX` are within the file `influx.json`. Finally, through this command:

```
./bridgeIoT -protocol mqtt -host 130.136.2.70

  -topic temperature -command translate

                 -dstprotocol coap  -config coap.json
```

the user is allowed to read data from the MQTT source and to create an output data flow mapped to the CoAP protocol. The parameters of the CoAP connection are within the file `coap.json`.

2.1 Additional components

- +1 point. **Data Forecasting**: we consider an additional functionality beside the data visualization, storage and bridging described before. In this case, the platform produces in output a data flow using the same protocol of the input data flow, however, returning the next value of the time-series computed according to the ARIMA model. The statistical parameters of the ARIMA model (i.e. p, d, q parameters) are defined by the users and passed as input of the command.
- +1 point. **ESP32 integration**: The functionalities of the framework must be tested by considering input data-flows produced by an IoT prototyping board (e.g. ESP32 connected to a DHT11 sensor).
- (for Laude only). **Graphical user Interface**: the user commands can be issued through a GUI (beside the CLI).

2.2 Technologies

There are no constraints on the programming languages and libraries; students are free to deploy the system by using any programming language and are not evaluated for that.

3 Special Projects

Details must be discussed with Prof. Marco Di Felice or Prof. Luciano Bononi. Each proposal is for a single, highly-motivated student/group. The topics are related to research areas of the IoT:

- Evaluation of LoRa/LoRaWAN networks using the NS-3 simulator or a small scale testbed.
- Evaluation of BLE Mesh networks using ESP32 devices within a small scale testbed.
- W3C Web of Things (WoT): mapping BLE Mesh devices to the W3C WoT standard.

- TinyML, i.e. using Machine Learning (ML) tools (e.g. TensorFlow Lite) on micro-controllers.
- Federate Learning on IoT use-cases, i.e. testing FL algorithms for distributed training on IoT anomaly detection scenarios.