University of Pisa

Laurea Magistrale in Artificial Intelligence and Data Engineering

Diy Smart Garden

Internet of Things course project

Academic year 2020 – 2021

GitHub repository: https://github.com/Matteo-Castrignano/Diy-Smart-Garden

Castrignano Matteo

1. Introduction

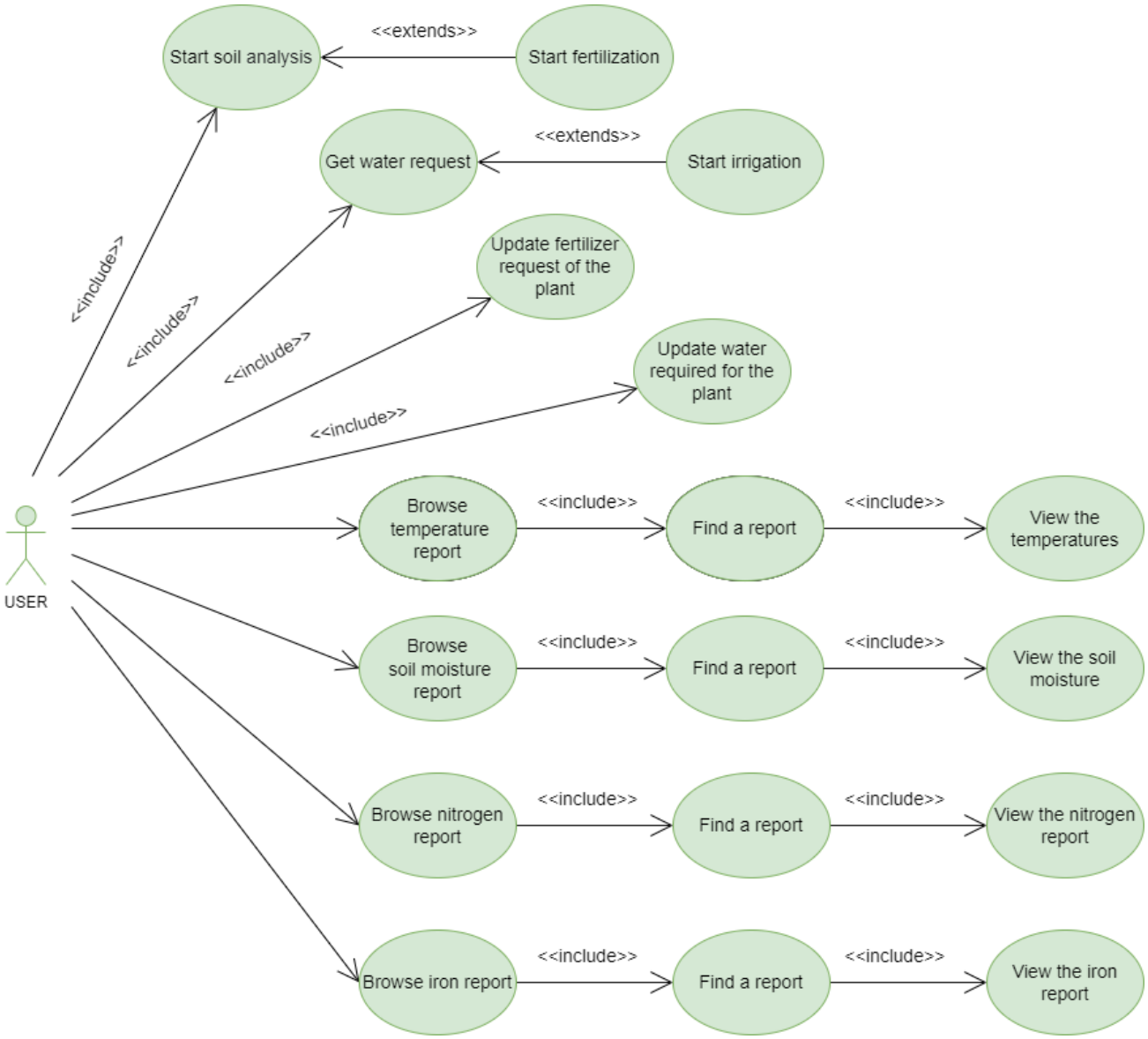
People nowadays increasingly need to grow a few small plants in their homes but often not take care of them optimally due to lack of knowledge and lack of time. To cope with this problem, a small application was created which, through a series of sensors, can provide the user with information on irrigating and fertilizing the plants. In particular, the application makes use of four sensors to be positioned near the plants we want to monitor, in particular:

- Temperature sensor: monitors the temperature of the external environment
- Soil moisture sensor: monitors the soil moisture and together with the previous sensor data necessary to understand when to water the plants.
- Soil iron monitoring sensor: iron is an indispensable nutrient for plants necessary for their growth. According to the needs of the plants, it must be present in the following quantities:
    - Little supplied: between 0.7% and 1.8%
    - Average supplied: between 1.8% and 2.6%
    - Well supplied: greater than 2.6%
- Soil nitrogen monitoring sensor: Nitrogen is another nutrient to monitor for a plant's growth. According to the needs of the plants, it must be present in the following quantities:
    - Little supplied: between 0.5 and 1
    - Average supplied: between 1 and 1.5
    - Well supplied: greater than 1.5

Periodically these sensors will collect the related data which will then be invited and processed by an application implemented in java. The user can insert into the system the necessary quantities of nutrients and water of his own plants, whose indications are very often found on the packages. Then it will be up to the application when to carry out the irrigation and fertilization. These two activities can be automated by installing two actuators:

- Irrigation actuator: opens and closes a small water valve.
- Fertilizer actuator: releases small quantities of fertilizer

2. Use cases



3. Implementation

The application was completely developed in java, while for the development of the sensors the C language was adopted.

The data from the sensors are collected by the application, which manages it and stores it in a MySQL database. As for the encoding of the data, the JSON was chosen, as it provides a light encoding of the data and given its simplicity, it is human readable, and this allows easy interpretation by those who read it. The data are sent by the sensors with two fields, one is the sensor id, the other is the value returned by the sensor, and are stored in the database adding a timestamp.

4. COAP Network

On the CoAP network has been decided to simulate the soil iron monitoring sensor, the soil nitrogen monitoring sensor and the actuator for fertilization. The latter, it makes use of LEDs to simulate the turned-off state (RED LED) and the turned-on state (GREEN LED).
The CoAP network is entirely simulated with Cooja.

5. MQTT Network

On the MQTT network has been decided to simulate the temperature sensor, the soil moisture sensor which include the actuator for irrigation.
The MQTT network is deployed using the 3 real sensors from the testbed, one is used to develop the RPL border router instead the other two are used to develop the soil moisture sensor and the temperature sensor.

6. The collector

The Collector is responsible of the connection to CoAP devices, of receiving updates from the MQTT Broker and the management database operations.
It provides to the user a CLI with the following commands:

1. Start fertilizes analyse
2. View irrigation requirement
3. View sensor log
4. View actuator log
5. Change water and fertilizer parameter
6. Start fertilizing
7. Start irrigation
0. Exit

7. Service manual

To run our application is necessary to execute the following steps:

1. Open contiki and develops, in this order, the rpl border router, the coap mote and the mqtt motes.
2. Create the tunslip interface.
3. Open the shh connection with the testbeds and in one of them develops a rpl border router instead in the other develops the mqtt mote.
4. Start the simulation.
5. Launch the java application and try it!