University of Pisa

School of Engineering

Master of Science in Artificial Intelligence and Data Engineering

---

Smart Greenhouse monitoring and management IOT system

Massimo Valentino Caroti
Simone Landi
Academic year 2021-2022

INDICE

INTRODUCTION

dsa asdsdA Diesel generator is a fossil-fuel powered generator employed to supply electrical power. Diesel generating sets are used in places without connection to a power grid, or as emergency power-supply if the grid fails. Electrical power is the most important source for electronic devices. In industrial systems, electrical power is needed to keep control systems running for 24 hours. Stable power source should also be taken care to prevent devices from being damaged.

The objective of this project is to design and implement an IoT telemetry and control system for diesel generators used in a data center. The system is composed of nodes that keep track of the energy generated by the generator, its fuel level and its temperature with their sensors. Sampled data are sent by the monitors, either using MQTT or CoAP, to a collector, which saves them in a database for data collection and analysis. The collector can also detect anomalies in the measurements and handle them.

DESIGN

OVERVIEW

dsa asdsdA Diesel generator is a fossil-fuel powered generator employed to supply electrical power. Diesel generating sets are used in places without connection to a power grid, or as emergency power-supply if the grid fails. Electrical power is the most important source for electronic devices. In industrial systems, electrical power is needed to keep control systems running for 24 hours. Stable power source should also be taken care to prevent devices from being damaged.

MQTT NODE

dsa asdsdA Diesel generator is a fossil-fuel powered generator employed to supply electrical power. Diesel generating sets are used in places without connection to a power grid, or as emergency power-supply if the grid fails. Electrical power is the most important source for electronic devices. In industrial systems, electrical power is needed to keep control systems running for 24 hours. Stable power source should also be taken care to prevent devices from being damaged.

COAP NODE

dsa asdsdA Diesel generator is a fossil-fuel powered generator employed to supply electrical power. Diesel generating sets are used in places without connection to a power grid, or as emergency power-supply if the grid fails. Electrical power is the most important source for electronic devices. In industrial systems, electrical power is needed to keep control systems running for 24 hours. Stable power source should also be taken care to prevent devices from being damaged.

COLLECTOR

dsa asdsdA Diesel generator is a fossil-fuel powered generator employed to supply electrical power. Diesel generating sets are used in places without connection to a power grid, or as emergency power-supply if the grid fails. Electrical power is the most important source for electronic devices. In industrial systems, electrical power is needed to keep control systems running for 24 hours. Stable power source should also be taken care to prevent devices from being damaged.

DATABASE

dsa asdsdA Diesel generator is a fossil-fuel powered generator employed to supply electrical power. Diesel generating sets are used in places without connection to a power grid, or as emergency power-supply if the grid fails. Electrical power is the most important source for electronic devices. In industrial systems, electrical power is needed to keep control systems running for 24 hours. Stable power source should also be taken care to prevent devices from being damaged.

DATA ENCODING

dsa asdsdA Diesel generator is a fossil-fuel powered generator employed to supply electrical power. Diesel generating sets are used in places without connection to a power grid, or as emergency power-supply if the grid fails. Electrical power is the most important source for electronic devices. In industrial systems, electrical power is needed to keep control systems running for 24 hours. Stable power source should also be taken care to prevent devices from being damaged.

GRAFANA

For data visualization, Grafana was used to display most of the information presented above.

The structure of Grafana is divided into two sections, one for general control of all greenhouses and a section in which the data of the greenhouses can be viewed individually.

To visualize the status of each greenhouse, we created a Grafana dashboard. One greenhouse per node, three monitoring panels for each node:

- the moisture panel shows the soil moisture trend. The moisture can only decrease, when it reaches the threshold and the operator activates the irrigation system, the moisture returns to normal.
- the sunlight panel shows the most recent measurement made by the sensor placed outside the greenhouse.
- the temperature panel shows the temperature trend. Whenever the temperature exceeds the threshold, you can see that subsequent measurements are reduced by the actuator until the temperature returns to normal.

An example is shown in the figure below:

Regarding the general monitoring section: a graph for each measurement using data from all greenhouses over the last month. These graphs provide a broader view in which we monitor monthly water consumption for irrigation, ventilation and lighting system use.

TESTING

The dongles used are IoT devices without sensing capabilities, so we need to simulate the sensors using the C language. To simulate measurements, we start a process for each type of sensor: sunlight intensity, soil moisture, and temperature inside the greenhouse.

- For sunlight intensity, we created a function that randomly simulates the light outside the greenhouse, so the sensor makes measurements by generating a random value within a dynamic range.

- For soil moisture, initially the sensor generates a random value that is decremented with each new measurement.

- For the temperature inside the greenhouse, we created a function that simulates, with randomness, the temperature inside the greenhouse, then the sensor performs measurements by generating a random value within a dynamic range.

In addition, the alarm system is also simulated. For all measurements we define thresholds, when at least one of the measurements made by the sensor are below the threshold value, the red LED on the sensor is turned on and the corresponding actuator is activated.
Specifically:

- For soil moisture the MinThresholds is 40, when the measurement is below this threshold, the red led is turned on and remains so until an operator pushes the irrigation button. Once pressed the led is turned off (if there aren’t other alert) and at the next measurement the soil moisture will have a value greater than the threshold.

- For the temperature inside the greenhouse the MaxThresholds is 30, the alarm is simulated by changing the color of the led to red. In addition, actuator commands are simulated by changing the way temperature samples are generated. In fact, at each measurement, the temperature is reduced by 1 degree Celsius to simulate the return of the greenhouse to a normal temperature by a ventilation system.

- For sunlight intensity, the MinThresholds is equal to 10, the alarm is simulated by changing the color of the LED to red but unlike other measurements, the activation of the actuator, in this case the UV lights, does not affect the values of the next measurements. This is to be more efficient and use artificial light only when sunlight is not enough for rapid growth.

The collector is implemented in Java and contains both MQTT and CoAP collectors.

COOJA

The test is performed by deploying all the components:MQTT broker, Java collector, database; on the same Ubuntu virtual machine. The IoT system is simulated by creating a single LLN in which a border router (node 1), three CoAP nodes (nodes 2, 3 and 4) and two MQTT nodes (nodes 5 and 6) are instantiated as Cooja motes.

The default configuration is used:

- the broker listens on port 1883
- the CoAP collector is reachable at [fd00::1]:5683
- the database is reachable at localhost:3306
- new samples are generated every 10 seconds

FOTO COOJA SIMULATION

FOTO TERMINALE COLLECTOR

FOTO TERMINALE particolare allert e bottoni

MYSQL DATABASE

All the data produced by the sensors are stored in a MySql Database by the Java Collector. The database runs locally on the same VM where the collector runs. The database is named SmartGreenHouse and contains 3 tables, one table for each measurement where measurement ID, IDdevice, measurement, Unit , Timestamp are contained

```
-- Table structure for table `temperature`
DROP TABLE IF EXISTS `temperature`;
CREATE TABLE `temperature` (
                             `sampleid` int(11) NOT NULL AUTO_INCREMENT,
                             `sample` float NOT NULL,
                             `unit` varchar(45) NOT NULL,
                             `machineid` varchar(45) NOT NULL,
                             `timestamp` datetime DEFAULT CURRENT_TIMESTAMP,
                            PRIMARY KEY (`sampleid`)
) ENGINE=InnoDB AUTO_INCREMENT=4920 DEFAULT CHARSET=latin1;

-- Table structure for table `light`
DROP TABLE IF EXISTS `light`;
CREATE TABLE `light` (
                             `sampleid` int(11) NOT NULL AUTO_INCREMENT,
                             `sample` float NOT NULL,
                             `unit` varchar(45) NOT NULL,
                             `machineid` varchar(45) NOT NULL,
                             `timestamp` datetime DEFAULT CURRENT_TIMESTAMP,
                            PRIMARY KEY (`sampleid`)
) ENGINE=InnoDB AUTO_INCREMENT=4632 DEFAULT CHARSET=latin1;

-- Table structure for table `humidity
DROP TABLE IF EXISTS `humidity`;
CREATE TABLE `humidity` (
                             `sampleid` int(11) NOT NULL AUTO_INCREMENT,
                             `sample` float NOT NULL,
                             `unit` varchar(45) NOT NULL,
                             `machineid` varchar(45) NOT NULL,
                             `timestamp` datetime DEFAULT CURRENT_TIMESTAMP,
                            PRIMARY KEY (`sampleid`)
) ENGINE=InnoDB AUTO_INCREMENT=4724 DEFAULT CHARSET=latin1;
```

1) create database : CREATE DATABASE smartgreenhouse;
2) select database : USE smartgreenhouse;
3) import dumb : SOURCE smartgreenhouse_dump.sql;

COLLECTOR

This module of the monitoring and control system collects data from IoT devices. These are its main functionalities:

- Process the received data and apply some modifications to one or more actuators, executing a control logic
- Store received data in a MySql database
- Output data to the user via a textual log

SIMULATION

Before deploying the collector check that the Mosquitto MQTT broker is running. To deploy the collector move into the IoT_Project/collector folder and run the following commands:

```
mvn clean install
mvn package
java -jar target/collector.iot.unipi.it-0.0.1-SNAPSHOT.jar
```

Open the Cooja simulator and import simulation.csc

To deploy the border router:

- add the socket on the border router (Tools -> Serial Socket (SERVER) -> Contiki
- start the serial socket
- use Tunslip6

```
make TARGET=cooja connect-router-cooja
```

- Start the simulation

DEPLOYMENT

To deploy the system on the dongles, flash the code on all the nodes:

- Border router:

  make TARGET=nrf52840 BOARD=dongle border-router.dfu-upload PORT= devttyACM0

- MQTT nodes:

  make TARGET=cc26x0-cc13x0 BOARD=/launchpad/cc2650 PORT=/dev/ttyACM0 NODEID=0x000x mqtt-sensor.upload

- COAP nodes:

  make TARGET=cc26x0-cc13x0 BOARD=/launchpad/cc2650 PORT=/dev/ttyACM0 NODEID=0x000x coap-sensor.upload

- After that power on all the nodes and use Tunslip6 to connect with the border router:

  make TARGET=nrf52840 BOARD=dongle connect-router PORT=/dev/ttyACM0