# UNIVERSITÀ DI PISA


INTERNET OF THINGS

## SMART AGRICOLTURE SYSTEM

Project Documentation
a.y 2019-2020

Laura Lemmi

# Index

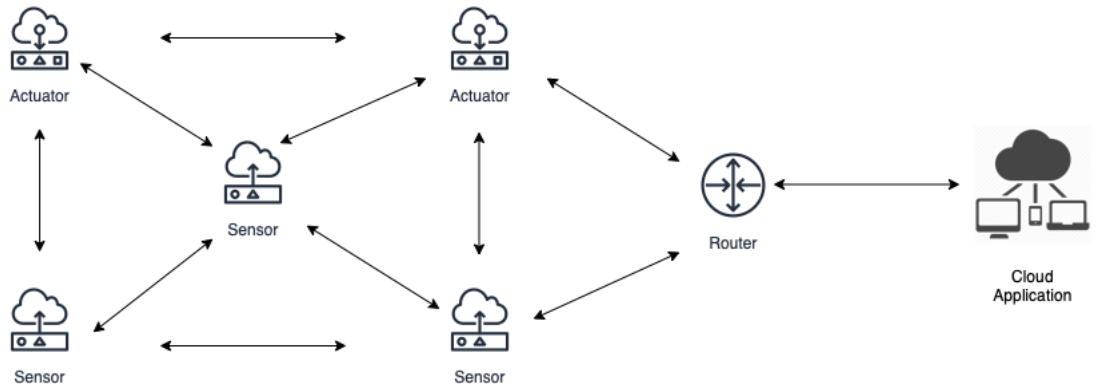
# 1. INTRODUCTION

This project aims to provide the user with a system capable of monitoring the conditions of a vegetable garden, acting automatically based on the detected values and allowing the verification of the status and the remote control through a user interface.
The system must be able to detect temperature and humidity data, trigger the switching on or off the light and sprinkler devices. The status of the devices must be automatically set by the system or by the user through the interface, based on the configuration status.

# 2. ARCHITECTURE

The system is composed by:

- Sensors: they collect data from the physical environment and send them to the cloud application.
- Actuator: they control and set the status of environmental devices.
- Border Router: it allows the communication between the application and the IoT devices.
- Cloud Application: it allows the interaction with the devices to obtain detected data and set environmental devices status.



## 3. FUNCTIONALITIES

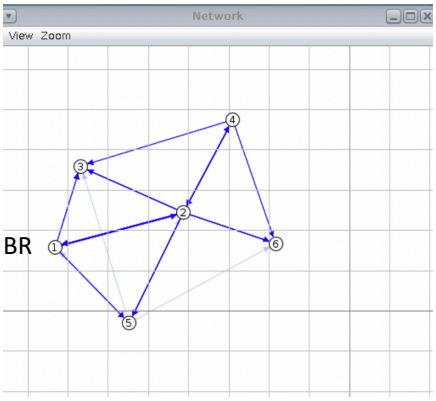
The system if formed of a set of devices (sensors and actuators) which, on startup, are placed by the application in the “default” area; the user has been given the possibility to change the latter in order to divide the nodes in different zones.

Each device may act as:

- Sensor
- Actuator
- Sensor and Actuator.

The device type must be set by the user once is turned on through the serial line exposed by it. After the type selection the device tries to register with the application by sending a CoAP request to the server and then, periodically, it monitors its resources and reports the latter values to the observers (the cloud application).

The network on which the operations and results shown in this document are based is formed by 5 nodes on which the custom program has been loaded, and a border router. The interaction between BR and the other nodes can be a multi-hop communication, which also allows the connection of the devices with the outside application through the former BR.



## 3.1 Resources

The resources offered by the **sensors** (acting as CoAP servers) to the application are:

- Temperature "coap://[nodeAddress]:5683/temperature"
- Humidity "coap://[nodeAddress]:5683/humidity"

They are both observable and their values are periodically reported to the application. The last 5 monitored values of each resource are maintained in the application data structures (associated to the related device) and their average is shown to the user on request. Those values may be also used for sending POST requests to the actuators in case of automatic management.

The resources offered by the **actuators** (acting as CoAP servers) to the application are:

- Light "coap://[nodeAddress]:5683/light"
- Sprinkler "coap://[nodeAddress]:5683/sprinkler"

They are both observable and their values are periodically reported to the application which used them to understand, based also on the sensor resources monitoring, whether their statuses should be changed.

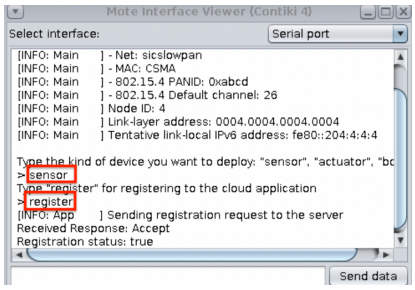
In addition, a last resource is exposed by each node to allow the user to unregister the latter from the application:

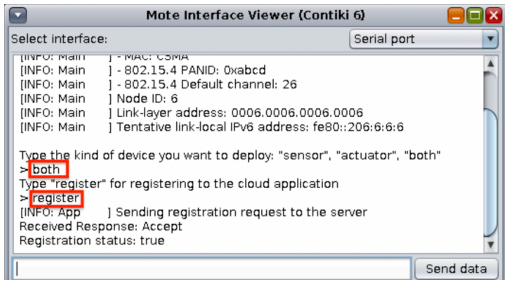
- Unregister "coap://[nodeAddress]:5683/unregister".

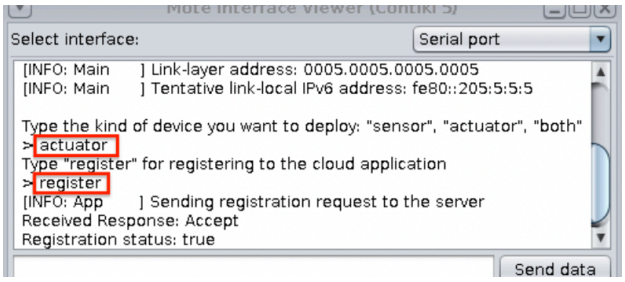
## 3.2 Cooja Simulation

Each node, except for the border router, can act as sensor, actuator, or both.
At the startup the nodes running the custom program, connect with the border router and wait for the user input to select the device type:

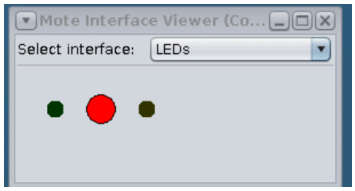
- sensor
- actuator
- both



Both sprinklers and lights are switched off at startup.




After the right input (node type and “register”), the node tries to register with the cloud application, which exposes the “Registration” resource, through the CoAP protocol:

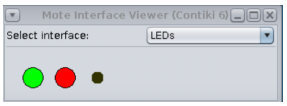
`"coap://[fd00::1]:5683/registration"`.

By sending a GET request to the endpoint: `"coap://[nodeAddress]:5683/.wellknown/core"`, the application obtains all the information of each device needed to implement the functionalities described. The application assigns an identifier to each registered device, in order to uniquely identify each exposed resource; that ID can be used by the user to get/set information and parameters of each.

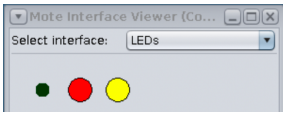
```
[SERVER]: Handling Registration Request
SRC ADDR: fd00:0:0:0:205:5:5:5
The area default already exists. Just add the device.
Device Area set to: "default" for resource: sprinkler
Resource: sprinkler, Resource observable: true

The area default already exists. Just add the device.
Device Area set to: "default" for resource: light
Resource: light, Resource observable: true

[SERVER]: Device Registered
```

After this phase, the node, based on its type, periodically monitors the resources, and reports the data to the observers.
Each actuator shows its status through the leds:

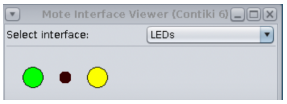
- RED: sprinkler and light off
- RED + GREEN: light off and sprinkler on




- RED + YELLOW: sprinkler off and light on




- YELLOW + GREEN: sprinkler and light on.



## 3.3 Application

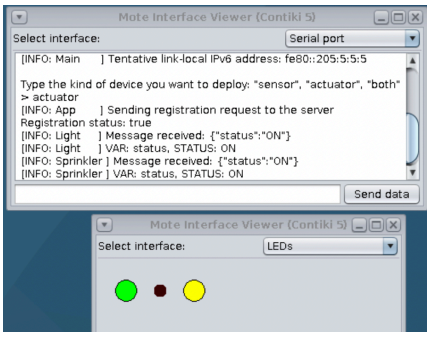
The application, thanks to the border router, is able to communicate with the WSN; registering the nodes it can then observe their exposed resources and changed the actuators status.
Regarding actuator resources, the application, for each, stores their periodically received status.
For the sensor resources instead, the application, for each, stores the last 5 measurements.
The devices are divided in areas, known only at application level: on startup they are all automatically placed in the “default” area, new placements can be setup by the user.

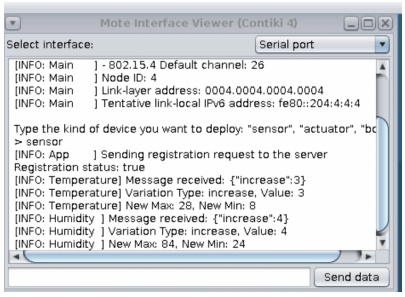
```
!getAreasList
        --      Get Areas Info  --
[ default ] :
{ID: 1, IP: fd00:0:0:0:202:2:2:2, type: sensor, res: humidity }
{ID: 2, IP: fd00:0:0:0:202:2:2:2, type: sensor, res: temperature }
{ID: 3, IP: fd00:0:0:0:204:4:4:4, type: sensor, res: humidity }
{ID: 4, IP: fd00:0:0:0:204:4:4:4, type: sensor, res: temperature }
{ID: 5, IP: fd00:0:0:0:203:3:3:3, type: actuator, res: sprinkler }
{ID: 6, IP: fd00:0:0:0:203:3:3:3, type: actuator, res: light }
{ID: 7, IP: fd00:0:0:0:205:5:5:5, type: actuator, res: sprinkler }
{ID: 8, IP: fd00:0:0:0:205:5:5:5, type: actuator, res: light }
{ID: 9, IP: fd00:0:0:0:206:6:6:6, type: sensor, res: humidity }
{ID: 10, IP: fd00:0:0:0:206:6:6:6, type: sensor, res: temperature }
{ID: 11, IP: fd00:0:0:0:206:6:6:6, type: actuator, res: sprinkler }
{ID: 12, IP: fd00:0:0:0:206:6:6:6, type: actuator, res: light }
```

The network can be managed:

- Automatically: the user can decide to set the management mode of a specific area to auto. In that case, if the application noticed that the average temperature / humidity values detected in the last 5 measurements are above/below the threshold, it changes the light / sprinkler status through a POST request to the actuators located in the same area. Based on that, also the range of randomly generated values for the measurement of the resource are changed. The thresholds, for each area, are set by the user at the creation of the latter.
  Given an area:
  - Temperature too high => Switch off the lights
  - Temperature too low => Switch on the light
  - Humidity too high => Switch off the sprinklers
  - Humidity too low => Switch on the sprinklers

```
Set Lights of area tomato to ON
Light 8 set to ON

Set Sprinklers of area tomato to ON
Sprinkler 7 set to ON
Set Lights of area tomato to ON
Lights of area tomato are already ON
```




- Manually: is the default configuration mode. The user, after having requested the average of the last measurement in a certain area, can decide to set ON or OFF the status of sprinklers/lights.

### 3.3.1 Operations

- "**!help**" the application prints all the commands allowed:

```
        --      This is the list of accepted command    --

!help                   -->     Get the list of the available commands

        --      GET COMMANDS    --
!getSensors             -->     Get the list of registered sensors
!getActuators           -->     Get the list of registered actuators
!getAddressResources    -->     Get the list of registered IDs with a given address
!getAreasInfo           -->     Get the list of areas and their info
!getAreasList           -->     Get the list of areas and their devices
!getAvgTemperature      -->     Get the Avg temperature of the last 10 measurements for all the sensors
!getAvgHumidity         -->     Get the Avg humidity of the last 10 measurements for all the sensors
!getSprinklerStatus     -->     Get the status of the sprinklers
!getLightStatus         -->     Get the status of the lights

        --      POST COMMANDS   --
!setAreaSprinklerStatus -->     Set the status of sprinklers in a area
!setAreaLightStatus     -->     Set the status of lights in a area
!setDeviceArea          -->     Set the area the device belongs to
!removeDeviceArea       -->     Remove the area the device belongs to
!switchAreaMode         -->     Set the management mode of an area
!editAreaThreshold      -->     Modify the min/max thresholds for temp and humidity
!removeDevicesAddress   -->     Remove the devices with given address

!stop                   -->     Stop the application
```

- "**!getSensors**" to obtain the list of the registered sensors. For each is shown:
  - area identifier
  - ID
  - IPv6 address
  - device type
  - resource type.

```
!getSensors
        --      Sensors List    --
Area: default, ID: 1, addr: fd00:0:0:0:202:2:2:2, type: sensor, Resource: humidity
Area: default, ID: 3, addr: fd00:0:0:0:204:4:4:4, type: sensor, Resource: humidity
Area: default, ID: 9, addr: fd00:0:0:0:206:6:6:6, type: sensor, Resource: humidity
Area: default, ID: 2, addr: fd00:0:0:0:202:2:2:2, type: sensor, Resource: temperature
Area: default, ID: 4, addr: fd00:0:0:0:204:4:4:4, type: sensor, Resource: temperature
Area: default, ID: 10, addr: fd00:0:0:0:206:6:6:6, type: sensor, Resource: temperature
```

- "**!getActuators**" to obtain the list of the registered actuators. For each is shown:
  - area identifier
  - ID
  - IPv6 address
  - device type
  - resource type
  - status of the related sprinkler/actuator devices.

```
!getActuators
        --      Actuators List  --
Area: default, ID: 5, addr: fd00:0:0:0:203:3:3:3, type: actuator, Resource: sprinkler, Status: OFF
Area: default, ID: 7, addr: fd00:0:0:0:205:5:5:5, type: actuator, Resource: sprinkler, Status: OFF
Area: default, ID: 11, addr: fd00:0:0:0:206:6:6:6, type: actuator, Resource: sprinkler, Status: OFF
Area: default, ID: 6, addr: fd00:0:0:0:203:3:3:3, type: actuator, Resource: light, Status: OFF
Area: default, ID: 8, addr: fd00:0:0:0:205:5:5:5, type: actuator, Resource: light, Status: OFF
Area: default, ID: 12, addr: fd00:0:0:0:206:6:6:6, type: actuator, Resource: light, Status: OFF
```

- **“!getAddressResources”**: given an address it returns a list made of:
    - ID of a device
    - Resource offered by the device.

```
!getAddressResources
Available Devices:
[ fd00:0:0:0:202:2:2:2 fd00:0:0:0:203:3:3:3 ]


Type the address of the devices
fd00:0:0:0:202:2:2:2
[fd00:0:0:0:202:2:2:2]:
[ ID: 3, Resource: humidity ]
[ ID: 4, Resource: temperature ]
Type a command
```

- “**!getAreasInfo**” to obtain the information of each area:
    - Management Mode
    - Status of sprinklers and lights in the area
    - Threshold values for temperature and humidity.

```
!getAreasInfo
Available areas:
[ Area: default, AutoMode: false, Sprinklers: OFF, Lights: OFF, Min Temp: 0, Max Temp: 30, Min Hum: 0, Max Hum: 100 ]
[ Area: salad, AutoMode: false, Sprinklers: OFF, Lights: OFF, Min Temp: 15, Max Temp: 20, Min Hum: 40, Max Hum: 50 ]
[ Area: tomato, AutoMode: false, Sprinklers: OFF, Lights: OFF, Min Temp: 18, Max Temp: 20, Min Hum: 45, Max Hum: 50 ]
[ Area: carrot, AutoMode: false, Sprinklers: OFF, Lights: OFF, Min Temp: 10, Max Temp: 18, Min Hum: 30, Max Hum: 70 ]
```

- “**!getAvgTemperature**” to obtain the averaged of the last 5 measurements for each temperature sensor.

```
!getAvgTemperature
        --      Last Average Temperatures Detected      --
Sensor Area: salad, Address: fd00:0:0:0:202:2:2:2, Average Temperatures: 15.0
Sensor Area: tomato, Address: fd00:0:0:0:204:4:4:4, Average Temperatures: 18.0
Sensor Area: carrot, Address: fd00:0:0:0:206:6:6:6, Average Temperatures: 12.0
```

- “**!getAvgHumidity**” to obtain the averaged of the last 5 measurements for each humidity sensor

```
!getAvgHumidity
        --      Last Average Humidities Detected        --
Sensor Area: salad, Address: fd00:0:0:0:202:2:2:2, Average Humidities: 60.0
Sensor Area: salad, Address: fd00:0:0:0:204:4:4:4, Average Humidities: 46.0
Sensor Area: carrot, Address: fd00:0:0:0:206:6:6:6, Average Humidities: 54.0
```

- “**!setAreaSprinklerStatus**” allows to change the status of all the sprinklers in the area provided by the user through command line. If a new sprinkler is later added in that area, its status will be changed accordingly to the one of that area. The leds of each related devices will be activated based on the new status.
  Each sprinkler status is set through a POST request to `"coap://[nodeAddress]:5683/sprinkler"` by adding as payload a json object which can be either {“status”: “ON”} or {“status”: “OFF”}.
  Changing a sprinkler status leads to change also the range of values for generating the humidity: for this reason is also send a POST request to the humidity sensors in the same area containing a random number in range [1, 5] for increment/decrement;
  `"coap://[nodeAddress]:5683/humidity"` by adding as payload a json object which can be either
    - {“increase”: randomInt} in case of status switched to ON

- {“decrease”: randomInt} in case of status switched to OFF.

```
!setAreaSprinklerStatus
Available areas with Sprinklers:
[ salad carrot tomato ]

Type the area of the sprinkler you want to switch
salad
Type the new status: ON or OFF
ON
Set Sprinklers of area salad to ON
Sprinkler 7 set to ON

Status Changed
```

- “**!setAreaLightStatus**” allows to change the status of all the lights in the area provided by the user through command line. If a new light is later added in that area, its status will be changed accordingly to the one of the area. The leds of each related devices will be activated based on the new status.
  Each light status is set through a POST request to `"coap://[nodeAddress]:5683/light"` by adding as payload a json object which can be either
  - {“status”: “ON”}
  - {“status”: “OFF”}.

  Changing a light status leads to change also the range of values for generating the temperature: for this reason is also send a POST request to the temperature sensors in the same area containing a random number in range [1, 5] for increment/decrement;
  `"coap://[nodeAddress]:5683/temperature"` by adding as payload a json object which can be either
  - {“increase”: randomInt} in case of status switched to ON
  - {“decrease”: randomInt} in case of status switched to OFF.

```
!setAreaLightStatus
Available Areas with Lights:
[ salad carrot tomato ]

Type the area of the lights you want to switch
carrot
Type the new status: ON or OFF
ON
Set Lights of area carrot to ON
Light 12 set to ON

Status Changed
```

- “**!setDeviceArea**” to put the device in a specific area. If the area ID typed is not present it would be created and its parameters are asked to be inserted.

```
!setDeviceArea
Available Devices:
Area: default, ID: 2, addr: fd00:0:0:0:202:2:2:2, type: sensor, Resource: temperature
Area: default, ID: 4, addr: fd00:0:0:0:204:4:4:4, type: sensor, Resource: temperature
Area: default, ID: 10, addr: fd00:0:0:0:206:6:6:6, type: sensor, Resource: temperature
Area: default, ID: 1, addr: fd00:0:0:0:202:2:2:2, type: sensor, Resource: humidity
Area: default, ID: 3, addr: fd00:0:0:0:204:4:4:4, type: sensor, Resource: humidity
Area: default, ID: 9, addr: fd00:0:0:0:206:6:6:6, type: sensor, Resource: humidity
Area: default, ID: 5, addr: fd00:0:0:0:203:3:3:3, type: actuator, Resource: sprinkler, Status: OFF
Area: default, ID: 7, addr: fd00:0:0:0:205:5:5:5, type: actuator, Resource: sprinkler, Status: OFF
Area: default, ID: 11, addr: fd00:0:0:0:206:6:6:6, type: actuator, Resource: sprinkler, Status: OFF
Area: default, ID: 6, addr: fd00:0:0:0:203:3:3:3, type: actuator, Resource: light, Status: OFF
Area: default, ID: 8, addr: fd00:0:0:0:205:5:5:5, type: actuator, Resource: light, Status: OFF
Area: default, ID: 12, addr: fd00:0:0:0:206:6:6:6, type: actuator, Resource: light, Status: OFF

Type the ID of the device
2
Type the area
salad
The area salad does not exist. Start creation.
        --Generating Area salad--
Insert max temperature tolerated in this area: 20
Insert min temperature tolerated in this area: 15
Insert max humidity tolerated in this area: 50
Insert min humidity tolerated in this area: 40
New area salad has been created
Device Area set to: "salad" for resource: temperature
```

- “**!switchAreaMode**” to change the management mode of a given area. Changing to automatic mode, the sprinkler and light statuses in that area will be set ON and OFF whenever the average of the last 5 measurements crosses the threshold (below/above) set by the user at area creation time. Those thresholds can be changed through “!editAreaThreshold”.

```
!switchAreaMode
Available areas:
[ Area: default, AutoMode: false, Min Temp: 0, Max Temp: 30, Min Hum: 0, Max Hum: 100 ]
[ Area: salad, AutoMode: false, Min Temp: 15, Max Temp: 20, Min Hum: 40, Max Hum: 50 ]
[ Area: tomato, AutoMode: false, Min Temp: 18, Max Temp: 20, Min Hum: 45, Max Hum: 50 ]
[ Area: carrot, AutoMode: false, Min Temp: 10, Max Temp: 18, Min Hum: 30, Max Hum: 70 ]

Type the area where management mode will be changed: tomato
Type the new management mode (Auto[1] / Manual[0]):
1
```

- “**!editAreaThreshold**” to change the threshold values of the given area

```
!editAreaThreshold
Available areas:
[ Area: default, AutoMode: false, Min Temp: 0, Max Temp: 30, Min Hum: 0, Max Hum: 100 ]
[ Area: salad, AutoMode: false, Min Temp: 15, Max Temp: 20, Min Hum: 40, Max Hum: 50 ]
[ Area: tomato, AutoMode: false, Min Temp: 18, Max Temp: 20, Min Hum: 45, Max Hum: 50 ]
[ Area: carrot, AutoMode: false, Min Temp: 10, Max Temp: 18, Min Hum: 30, Max Hum: 70 ]

Type the area where management mode will be changed: salad
Insert max temperature tolerated in this area: 25
Insert min temperature tolerated in this area: 10
Insert max humidity tolerated in this area: 60
Insert min humidity tolerated in this area: 35
Modifications done
```

- **“!removeDevicesAddress”** to remove and unregister all the devices with a given address. If the device is an actuator, its status is set to OFF before deregistration.

```
!removeDevicesAddress
Available Devices:
Area: default, ID: 2, addr: fd00:0:0:0:202:2:2:2, type: sensor, Resource: temperature
Area: default, ID: 6, addr: fd00:0:0:0:204:4:4:4, type: sensor, Resource: temperature
Area: salad, ID: 10, addr: fd00:0:0:0:206:6:6:6, type: sensor, Resource: temperature
Area: default, ID: 1, addr: fd00:0:0:0:202:2:2:2, type: sensor, Resource: humidity
Area: default, ID: 5, addr: fd00:0:0:0:204:4:4:4, type: sensor, Resource: humidity
Area: salad, ID: 9, addr: fd00:0:0:0:206:6:6:6, type: sensor, Resource: humidity
Area: default, ID: 3, addr: fd00:0:0:0:203:3:3:3, type: actuator, Resource: sprinkler, Status: OFF
Area: default, ID: 7, addr: fd00:0:0:0:205:5:5:5, type: actuator, Resource: sprinkler, Status: OFF
Area: salad, ID: 11, addr: fd00:0:0:0:206:6:6:6, type: actuator, Resource: sprinkler, Status: ON
Area: default, ID: 4, addr: fd00:0:0:0:203:3:3:3, type: actuator, Resource: light, Status: OFF
Area: default, ID: 8, addr: fd00:0:0:0:205:5:5:5, type: actuator, Resource: light, Status: OFF
Area: salad, ID: 12, addr: fd00:0:0:0:206:6:6:6, type: actuator, Resource: light, Status: OFF

Type the address of the devices
fd00:0:0:0:206:6:6:6
Resource Device: humidity removed from area salad

Device 9 removed

Resource Device: temperature removed from area salad

Device 10 removed

Sprinkler 11 set to OFF
Resource Device: sprinkler removed from area salad

Device 11 removed

Resource Device: light removed from area salad

Device 12 removed

No more devices with the address: fd00:0:0:0:206:6:6:6. Remove it

Removed devices with address: fd00:0:0:0:206:6:6:6
Unregister devices with address: fd00:0:0:0:206:6:6:6
```

- **“!stop”** to stop the application.
    - It switches off all the actuators status
    - It unregisters and remove the devices
    - It stops the application.

```
!stop
Removing all the Devices...
Resource Device: humidity removed from area salad

Device 1 removed

Resource Device: temperature removed from area salad

Device 2 removed

No more devices with the address: fd00:0:0:0:202:2:2:2. Remove it

Removed devices with address: fd00:0:0:0:202:2:2:2
Unregister devices with address: fd00:0:0:0:202:2:2:2

Resource Device: sprinkler removed from area salad

Device 3 removed

Resource Device: light removed from area salad

Device 4 removed

No more devices with the address: fd00:0:0:0:203:3:3:3. Remove it

Removed devices with address: fd00:0:0:0:203:3:3:3
Unregister devices with address: fd00:0:0:0:203:3:3:3

All devices have been unregistered and removed

Stopping the application...

osboxes@osboxes:~/contiki-ng/examples/smart-agriculture$
```

## New registration

A node, after having been unregistered from the application, can be registered again; it will be assigned a new ID for each of the exposed resourced and will be set in the “default” area. Everything will happen as in the first registration. For simplicity the device type (sensor/actuator/both) cannot be changed.