Ontology Enabled Internet of Things System for Smart Buildings

Dušan Popadić*[a], Lazar Berbakov[a], Marko Jelić*[a], Marko Batić[a]
* School of Electrical Engineering, University of Belgrade, Belgrade, Serbia
[a] Institute Mihajlo Pupin, University of Belgrade, Belgrade, Serbia
{dusan.popadic, lazar.berbakov, marko.jelic, marko.batic}@pupin.rs

***Abstract* – Irrational energy consumption and waste of energy are two of the main problems that we face today. A lot of the energy is wasted by people in their homes or offices because they are not paying much attention to their energy consumption. In order to increase energy efficiency of the general population and to raise general awareness, a system based on Internet of Things (IoT) is developed to influence users to be more energy efficient. Because of the great number of different entities in the buildings and their complex relationships, a semantic ontology is used to store contextual knowledge about spatial arrangements of the rooms, devices and sensors in the buildings.**

I. INTRODUCTION

Irrational energy consumption and waste of energy are two of the main problems that we face today. A lot of the energy is wasted by people in their homes or offices because they are not paying much attention to their energy consumption (e.g. leaving lights or turned on multimedia devices in an unoccupied room). It is expected that technological advancement will require people to be more energy efficient and to be more aware of the energy they use. It will require change of habits to the most people which is not easy to do. In order to increase energy efficiency of the general population and to raise general awareness, a system based on Internet of Things (IoT) is developed to influence users to be more energy efficient.

In this paper, the architecture of the Energy Conservation Measures (ECM) system is presented. ECM helps users to monitor their energy consumption, alerts them about potentially problematic situations and in the end influence them to be less wasteful (Figure 1). There are examples of IoT approaches to this problem [1] [2], however, the majority of systems does not recognize problematic situations but rather just displays raw measured data. ECM aims to be proactive and alerts users about problems and even recommends actions to perform. Because of the great number of different entities in the buildings and their complex relationships, a semantic ontology [3] is used to store contextual knowledge about spatial arrangements of the rooms, devices and sensors in the buildings. ECM uses a mobile app developed in [4] to communicate with users.

II. EVENTS

Each potentially problematic situation has its own predefined logical conditions that determine whether that situation is currently taking place. However, instantly notifying a user could be problematic, e.g. a user could open a window just to tell something to a neighbor and close it after

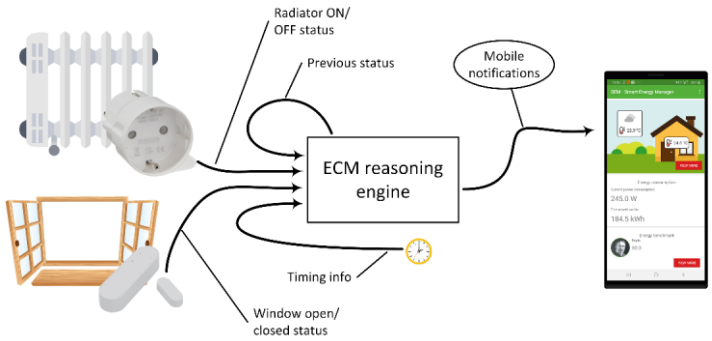

Figure 1. ECM overview

few seconds. In that case, it is not good to bother the user with an alert that the window is open because sending notifications too soon in such situation could cause the user to start disregarding notifications very quickly. Therefore, a new logic value, called event activity indicator, is introduced. This indicator determines if the event is really active so the notification can be sent. Even when the logical conditions for a problematic situation are satisfied, an event will not be activated for a certain amount of time, called trigger timeout, as shown in Figure 2. Trigger timeout is predefined and can be different for different events.

After the event is activated, a notification will be sent to the user alerting them of a potentially problematic situation so they can react. If the user does not react, the notification will be resent after notification timeout, which is (as in case of trigger timeout) predefined and event-specific.

The user can now choose to snooze, dismiss or disable the notification. If the notification is snoozed, then it will be repeated after notification timeout if the event is still active, as shown in the Figure 2. If the user dismisses the notification, it will not be repeated until the event is restarted (deactivated and activated again). If the user disables the notification, it will not be repeated until enabled again.

ECM uses four types of sensors:

- Window sensors
- Motion, light, temperature (MLT) sensors
- VOC sensors
- Smart cables/plugs

Window sensors consist of two parts that can be joined together or separated. When a window is open, parts are separated and when window is closed, parts are joined. In addition, window sensors also measure temperature. MLT sensors can detect motion in a room, measure illuminance

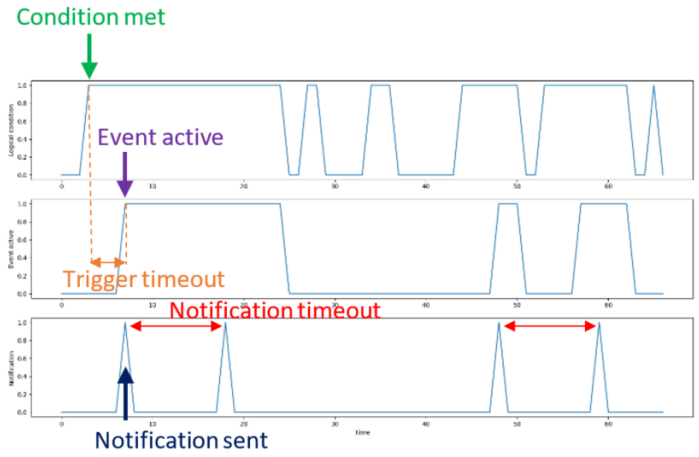

Figure 2. Event activity diagram

and temperature. VOC sensors monitor volatile organic compound (VOC) levels in the air. Based on the level of VOCs in the air, air quality can be determined. Besides, it also measures humidity and temperature. Smart plugs/cables measure demand of the appliances that have been plugged in them. ECM uses them to determine if a device is turned on. All these sensors provide the engine with the measurements which are used to detect problematic situations.

There are three types of events in ECM: Energy conservation issues, Security issues, and Health issues. Also, events can be divided spatially into two groups: room-related events and apartment-related events. The collection of the event conditions that are being monitored by ECM is as follows:

1. Energy conservation issues
   - 1.1. Heating is turned on while at least one window is open in the same room.
   - 1.2. A room is unoccupied while the heating is on.
   - 1.3. A room is unoccupied while a multimedia device is turned on.
   - 1.4. A room is unoccupied while the lights are turned on.
   - 1.5. The temperature in the room is larger than a predefined threshold while the heating is turned on.
   - 1.6. The house is unoccupied while the heating is on.
2. Security issues
   - 2.1. A room is unoccupied while a window is open in that room.
   - 2.2. The entrance door is open while there is no one in the hall.
   - 2.3. The entrance door is open while there is no one in the house.
   - 2.4. The house is unoccupied while a window is open.
3. Health issues
   - 3.1. Moderate air quality has been detected while all the windows in the room are closed.
   - 3.2. Poor air quality has been detected while all the windows in the room are closed.
   - 3.3. Unhealthy air quality has been detected while all the windows in the room are closed.
   - 3.4. Unhealthy air quality has been detected while the windows are open.

Some of the events imply that other events are also active e.g. event 1.6 implies that several events of type 1.2 (for each of the rooms in the house) will also be active. It would bother users to get a notification for an event of type 1.6 for their apartment and then to get several more notifications for each of the rooms. To deal with that problem, the concept of event prioritizing is introduced, so the events of lower priority (in this example 1.2) will not be shown as active due to being in a shadow of higher priority event (1.6). Also, if there is a room which is unoccupied, has a heating turned on and an open window events 1.1, 1.2 and 2.1 will all be active. However, the user will get redundant information because activity of 1.1 and 1.2 events implies that 2.1 will also be active. Event grouping is introduced in order not to bother users. These three events are grouped into one single event and only one notification is sent to the user.

III. SYSTEM ARCHITECTURE

ECM service consists of five parts: reasoning engine, Influx database, ontology, MySQL database and mobile application as shown in Figure 3.

Influx database is used for the purpose of storing real time data gathered from the sensors installed on the sites. Each sensor works "on change" and sends changes in measurements to the database which are then stored in different data groups according to the type of measurement (e.g. temperature, demand etc.). This data is later used by the reasoning engine to determine whether or not an event is active and whether a notification should be sent. Influx database is hosted on the server and can be accessed by ECM using *influxdb-python*[1] library. Queries are written in InfluxQL language.

Mobile application is used by end users, owners of the apartments, to receive notifications sent by ECM and to react upon them. Since some of the devices such as radiators and multimedia devices are connected to the power source using smart cable or smart plug, users have the opportunity to turn them off immediately. For example, if a user receives a notification saying that there is no one in a room and a

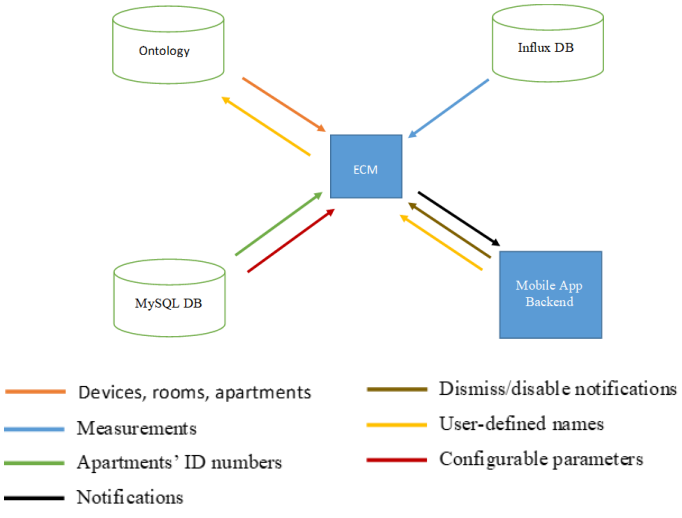

Figure 3. System Architecture

[1] https://github.com/influxdata/influxdb-python

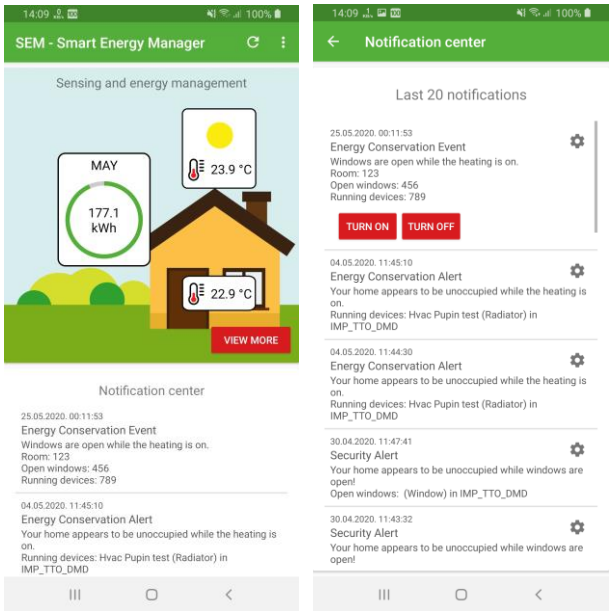

Figure 4. Mobile Application

multimedia device is active, he or she will have an option to turn that device off by clicking on a button in the notification itself. In the app, users can see some basic information about their house or apartment such as temperature and total consumption (see Figure 4, left). It also shows all the notifications generated by ECM service (Figure 4, right), but displays them as push notifications as well. Users can customize the notifications by naming certain rooms and devices in their apartments e.g. "Mark's bedroom" or "Radiator under the kitchen window". Since ECM gets new data from the ontology once a day, updating the ontology with information about customization directly from the app is not good enough. If it was done like that, users would have to wait for hours until the notifications they receive start using custom names. Because of that, the reasoning engine has to be notified immediately. In order to allow asynchronous communication from the app to the reasoning engine, MQTT protocol implementation [5] is used. The same protocol is used to notify the engine if the user snoozed or disabled notification.

The ontology stores data about all the sites, houses, apartments, rooms and installed sensors and links between them. It provides the system with conceptual knowledge about spatial arrangements of the sites. Ontology provides the information which sensors are installed in which rooms, which rooms belong to which apartments, which apartments are in which buildings etc. It also stores information about the area of the rooms, geo location of the sites, devices that are connected to sensors (e.g. radiators or TVs) and user-defined labels of rooms and devices. Beside knowledge that is directly put into the ontology, the ontology can infer some knowledge using its semantic reasoner so it can simply provide list of all sensors from a certain apartment or building, not just the room. The ontology is hosted on an Apache Jena Fuseki server and it can be queried by ECM service using HTTPS POST requests. In the listing below, two examples of SPARQL queries used by ECM to query the Ontology are given:

Get all devices from all rooms and apartments

```
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX owl: http://www.w3.org/2002/07/owl#
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
PREFIX xsd: http://www.w3.org/2001/XMLSchema#
PREFIX bot: https://w3id.org/bot#
PREFIX inb: <http://www.semanticweb.org/sipeticm/ontologies
/2019/0 /inbetween#>
SELECT ?device ?device_type ?room ?apartment
WHERE {
        ?room a bot:Space.
        ?apartment a bot:Space.
        ?room inb:containedInZone ?apartment.
        {
                ?room bot:containsElement ?device.
        }UNION{
                ?EWI bot:interfaceOf ?room.
                ?EWI inb:hostsElement ?window.
                ?window inb:hostsElement ?device.
        }UNION{
                ?apartment bot:adjacentElement ?IW.
                ?IW inb:hostsElement ?door.
                ?door inb:hostsElement ?device.
        }
        ?device a ?device_type
            OPTIONAL
            {
                ?device inb:controlsElement ?element.
                ?element a ?element_type.
                FILTER(?element_type != owl:NamedIndividual)
            }
        FILTER(?device_type != owl:NamedIndividual)
}
ORDER BY ?apartment ?room
```

Replace old custom name for a device with a new one

```
PREFIX inb: <http://www.semanticweb.org/sipeticm/ontologies
/2019/0 /inbetween#>
PREFIX bot: <https://w3id.org/bot#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

DELETE{
  inb:device_serial_number[2] rdfs:label ?old_custom_name.
}
INSERT{
  inb:device_serial_number rdfs:label custom_name.
}
WHERE{
  OPTIONAL{
    inb:device_serial_number rdfs:label ?old_custom_name. }}
```

[2] Italic letters mark placeholder for real values.

A MySQL database is used by the mobile application to store its data and to store configurable parameters for the ECM such as trigger and notification timeouts for different events.

ECM engine works in a loop that is constantly checking if an event is activated. In every iteration of the loop, the reasoning engine needs to go through all the apartments and rooms, get all the devices from that apartment or room from the ontology and then gather measurements for those sensors from Influx database. After that, it needs to go through all events for each of the rooms and check if they should be activated. In order to be useful, the ECM system must work in real time or close to real time. Therefore, efficiency is crucial. Trigger timeouts for the events are usually not less than 10 minutes long and they could go up to 1 hour. If one iteration of the main loop of the reasoning engine takes no longer than 1 minute (which is no more than 10% of trigger timeouts of most events), practical needs would be satisfied.

Currently, there are 1013 sensors, 551 rooms and 39 apartments in the system. The problem is that if the ontology is queried in every iteration and Influx database is queried for the latest change in measurement value for each sensor, it would take several minutes for each iteration to complete. To deal with this problem, all the contextual knowledge from the ontology is loaded into local memory of the system and the loaded data is organized into a tree of buildings, apartments, rooms, sensors and devices. When the ontology is loaded, reasoning engine goes through all the sensors and gets their latest measurements, and this process takes several minutes to complete. However, in all the following iterations, it is not needed to go through every sensor to load measurements. The solution is to get all the measurements from the Influx database that happened in the past 5 minutes. It will get more data from the database than querying for every single sensor independently. However, only one query will be needed instead of 1013 and this will accelerate the system significantly! With this change one iteration takes around 20 seconds of time which satisfies the needs of our system. Since the measurements from the past 5 minutes are taken and this is done approximately every 20 seconds, it is certain that no measurements will be missed. Reducing the time of 5 minutes will not improve performance significantly.

It is possible that changes occur on the sites by moving a device from one room to another, adding new sensors, including new apartments or buildings etc. Since the ontology is loaded into dynamic memory of the system, updates of the contextual knowledge cannot be seen by the reasoning engine. To include those updates, ontology is reloaded every day during the night. Apart from reloading the ontology, configuration parameters from MySQL database (e.g. trigger timeouts) are also reloaded. For more urgent updates such as user changing name of a device via mobile application, MQTT broker is used as already stated above. The flowchart that illustrates the algorithm that the ECM performs is shown in Figure 5.

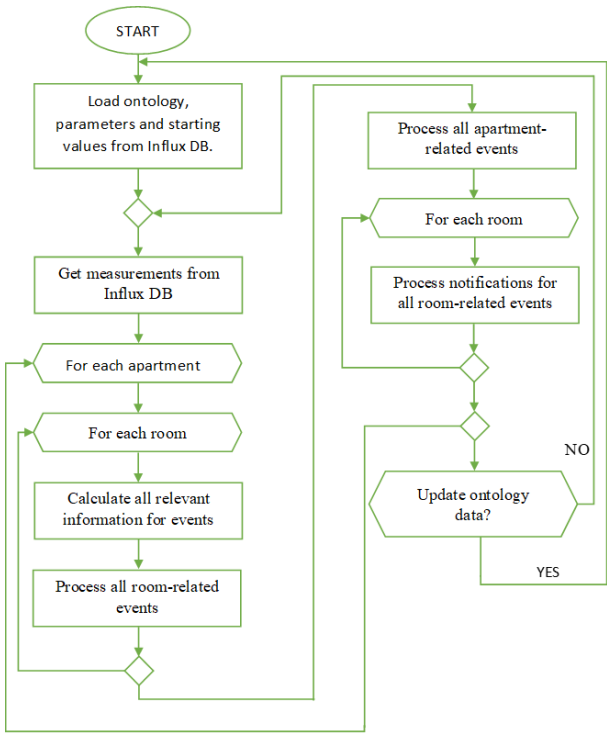

Figure 5. ECM flowchart

ACKNOWLEDGMENT

The research presented in this paper is partly financed by the European Union (H2020 LAMBDA project, Pr. No: 809965, H2020 InBetween project, Pr. No: 768776), and partly by the Ministry of Science and Technological Development of Republic of Serbia.

REFERENCES

[1] H. N. Rafsanjani and A. Ghahramani, “Towards utilizing internet of things (IoT) devices for understanding individual occupants’ energy usage of personal and shared appliances in office buildings,” *J. Build. Eng.*, vol. 27, p. 100948, Jan. 2020, doi: 10.1016/j.jobe.2019.100948.

[2] J. Iqbal *et al.*, “A generic internet of things architecture for controlling electrical energy consumption in smart homes,” *Sustain. Cities Soc.*, vol. 43, pp. 443–450, Nov. 2018, doi: 10.1016/j.scs.2018.09.020.

[3] “Ontologies - W3C.” https://www.w3.org/standards/semanticweb/ontology (accessed Jan. 15, 2020).

[4] L. Berbakov, M. Batic, and N. Tomasevic, “Mobile application for energy management in smart buildings,” presented at the International Conference on Information Society and Techology, 2019.

[5] R. Light, “Mosquitto: server and client implementation of the MQTT protocol,” *J. Open Source Softw.*, vol. 2, no. 13, p. 265, May 2017, doi: 10.21105/joss.00265.