# SMART LOCAL BANK OFFICE

**Table of Contents**

Contents

[SMART LOCAL BANK OFFICE 1](#_Toc27713508)

[General Introduction: 1](#_Toc27713509)

[Requirements: 2](#_Toc27713510)

[Brief Introduction: 2](#_Toc27713511)

[Functional Requirements Overview: 2](#_Toc27713512)

[Non-functional Requirements Overview 5](#_Toc27713513)

[Specifications 7](#_Toc27713514)

[Domain Model Class Diagram 7](#_Toc27713515)

[Architecture & Design 8](#_Toc27713516)

[Block Diagrams 9](#_Toc27713517)

[Hardware Architecture 10](#_Toc27713518)

[Software Architecture 11](#_Toc27713519)

[Testing 12](#_Toc27713520)

[Appendix 13](#_Toc27713521)

[Code 13](#_Toc27713522)

[Screenshots: 29](#_Toc27713523)

## General Introduction:

We aim to provide the bank worker and manager, by using the technology of our time, a friendlier working environment. Through the use of smart AI, complex algorithms and embedded systems technologies, we aim to provide the manager the tools to monitor a messenger car and also the means to collect, analyze and display environmental data for example temperature and humidity. By using a smart AI we can assist our customers by automating life at work. For example lights will be automatic based on the daylight existence and a feature that automatically switches on and off the AC based on the temperature and humidity.

## Requirements:

### Brief Introduction:

The Bank Managers will be expecting a system that will aid them in monitoring their messenger car via a GPS tracking system and providing them the necessary analytics for power consumption but also each day’s temperature and humidity levels. The bank employees will be expecting to have a friendlier work environment that will automate their life by automatically switching on and off the air conditioning based on the temperature and humidity levels, by which, simple bank visitors will benefit also.

### Functional Requirements Overview:

Functional requirements include the patented sensors and the actuators which the end-users will use in order to visualize the results of our app.

#### Requirements form of some of our patented sensors:

|  |  |
| --- | --- |
| Name | GPS tracker |
| Purpose | This will inform the bank manager about the position of the bank’s messenger car at a specific time. |
| Inputs | Raspberry Pi Connection |
| Outputs | Car’s Position |
| Functions | Supports SMS, phone call, GPRS, DTMF, HTTP, FTP, MMS, email, etc.  Support GPS, COMPASS, Glonass, LBS base station  base station positioning, Omni-positioning  Bluetooth 3.0, supports data transferring through Bluetooth |
| Performance | Baudrate auto detection (1200bps ~115200bps) |
| Manufacturing cost | Approximately €45 |
| Power | 3-5v power supply |
| Physical size/Weight | 5 x 3 x 2 cm / 31.8 g |

|  |  |
| --- | --- |
| Name | Temperature and humidity sensor |
| Purpose | This will collect the information regarding the temperature and humidity in the office. That will help provide the bank manager and the bank employee graphs by processing the temperature and humidity levels |
| Inputs | Environmental Temperature and Humidity,  1.62 V - 3.6 V power supply (from Arduino) |
| Outputs | 16-bit data output |
| Functions | Programmable sampling rate (5 Hz, 2 Hz, 1 Hz, 0.2 Hz, 0.1 Hz, 1/60 Hz, 1/120 Hz) or trigger on demand,  I2C interface |
| Performance | Operating range: –40°C to 85°C  Functional range: –40°C to 125°C  Temperature accuracy: ±0.2°C typical  Relative humidity range: 0% to 100%  Humidity accuracy: ±2% |
| Manufacturing cost | €3.36 |
| Power | 1.62 V - 3.6 V power supply |
| Physical size/Weight | 1.5 mm × 1.5 mm × 0.675 mm / 30 g |

|  |  |
| --- | --- |
| Name | Light Detector Module |
| Purpose | This will verify if there is daylight or not. |
| Inputs | Daylight from environment |
| Outputs | 0 for day or 1 for night |
| Functions | Adjustable swell = potentiometers on the plate.  Quick and start thanks to the headers, so you can even Solderless Breadboard compatible |
| Performance | Digital Output removes when you exceed a made in any brightness Chwell value. |
| Manufacturing cost | Approximately €7 |
| Power | 3.3/5.0 V power supply |
| Physical size/Weight | approximately 42 mm x 15 mm x 8 mm |

|  |  |
| --- | --- |
| Name | Dimming modern led ceiling lights |
| Purpose | This is an automatic light that will switch on if it’s dark |
| Lighting area | 10-15 square meters |
| Functions | Modern Ceiling lights |
| Material | Acryl |
| Number of light sources | > 20 |
| Manufacturing cost | €66 |
| Power | 90-260 V power supply |

|  |  |
| --- | --- |
| Name | Raspberry Pi Camera Module |
| Purpose | The number of people in the conference room will be counted and so if it’s night, then the lights will turn on in case the camera locates people |
| Inputs | 3-5v |
| Outputs | Pictures 3280 x 2464 pixels  Video 1080p30 (Full HD),  720p60  640x480p60/90 |
| Functions | CCTV security camera, motion detection, time lapse photography |
| Performance | Pictures 3280 x 2464 pixels  Video 1080p30 (Full HD),  720p60  640x480p60/90 |
| Manufacturing cost | €26 |
| Power | 3-5v power supply |
| Physical size/Weight | 5mm x 20mm x 9mm 3g |

### Non-functional Requirements Overview

We need Algorithms to produce useful data from the metrics and the procedures which will transfer the information from out edge devices to the cloud for further analysis.

#### Algorithm Requirements Overview:

LED lights Algorithm: We need an algorithm which takes as input the instance of day light which is calculated from the light detector module and switches on/off the LED lights in the conference room and also from the existence of people in the room, indicated by the camera module.

###### Inputs: 0 for day and 1 for night, number of people in the room.

###### Processing: If it’s day switch them off, if it’s night switch them on only if there are people in the room

###### Outputs: Switch lights on/off and power consumption levels in kWh

Air Conditioning algorithm: We need an algorithm, which given some inputs regarding temperature and humidity, taken from our temperature and humidity sensor, will turn on and off the air conditioning

###### Inputs: Temperature and humidity levels

###### Processing: If temperature exceeds 28 ℃ turn the AC on, if temperature decreases at 25℃ turn AC off.

###### Outputs: Turn AC on/off and power consumption levels in kWh

#### Performance Requirements:

This package is going to cover the needs of a Bank Office. We need to create a system able of handling traffic from multiple sensors and sending it to the cloud for further analysis. At this moment of development there is little to no need of edge calculation so we will simply send the data from the Arduino-sensor configuration to the Raspberry Pi’s and from there on to the Cloud.

#### Logical Database Requirements:

We will probably require multiple databases since some of our algorithms require huge amounts of data for specific operations. Our general control database will include tables for: Messenger car locations, Power Consumption for each device (LED lights, AC), Daylight (0 or 1) and Individual Metrics and Analytics for each room for temperature and Humidity Levels for the last 24Hrs.

#### Reliability:

* The system will not turn on or off the AC unless it’s certain about the temperature and humidity levels
* The system should provide correct graphs for power consumption
* AC and LED lights can be turned off manually too, via the app

#### Availability:

* The system must be plug-and-play. This means that no real configurations should be performed by the bank manager.
* The system must be running 24/7.

#### Security:

* The system must include some sort of identity/personal data protection for legal reasons

#### Maintainability:

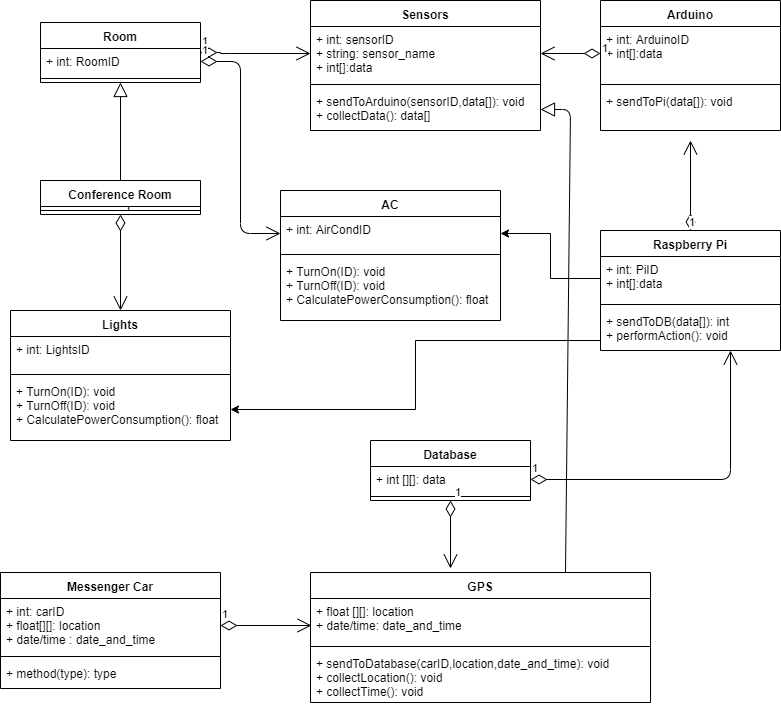
* The system is component-based. This means that if a part of the system is out of order it is simply ditched or replaced without the entirety of the system being disrupted.
* All parts of the package are simple hardware so the replacement is not forbidding. The cost lies behind algorithm updates and database changes.

#### Portability:

* The system should be usable wirelessly, that means we should configure all devices with wireless technologies in order to be of practical use.

## Specifications

### Domain Model Class Diagram



This is a Software blueprint in java classes which gives us an idea of how the end system will operate.

## Architecture & Design

The major components that satisfy our specifications are divided into two categories. The Hardware components and the Software components.

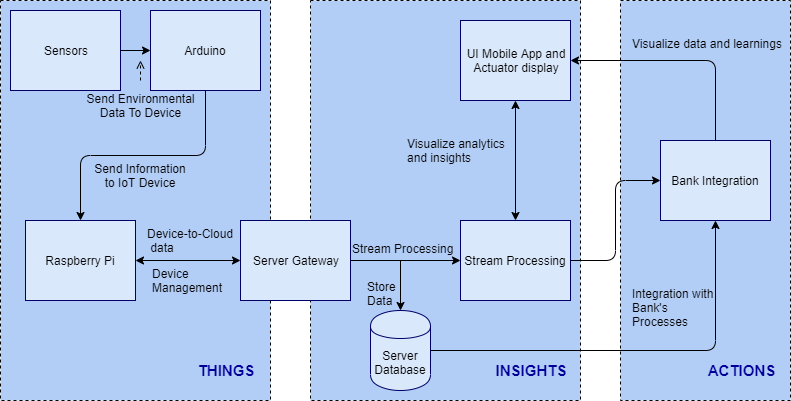
Hardware components

* Sensors
* Actuators
* IoT Devices (Arduinos/Raspberry Pi)

Software components

* Server
* Database
* Complex Algorithms

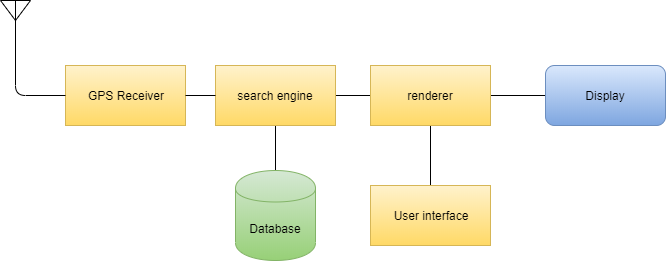
#### Our Basic Architecture Design is the following:



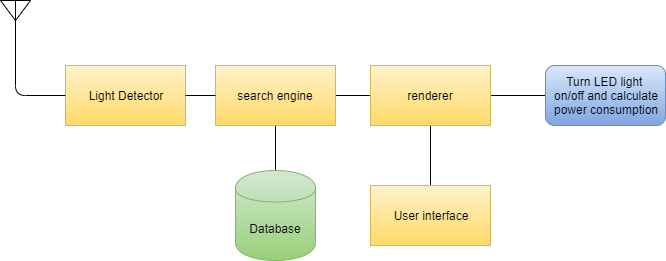
Where our sensors will send environmental data to their Arduinos and the Arduinos will forward them to the Raspberry Pi. The Raspberry Pi will send the information to our Server via the Server’s Gateway where they will be stored in its Database. Afterwards Data are processed as a stream and sent to the bank Integration and finally the data are visualized and displayed in our application and our actuators.

### Block Diagrams

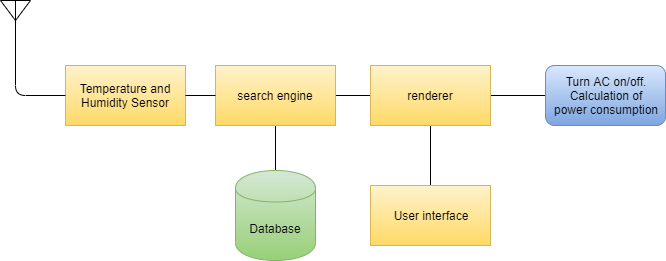
#### GPS tracker



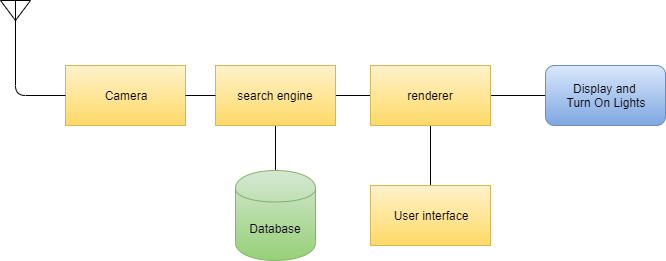
#### Light Detector/Turn LED light on/off



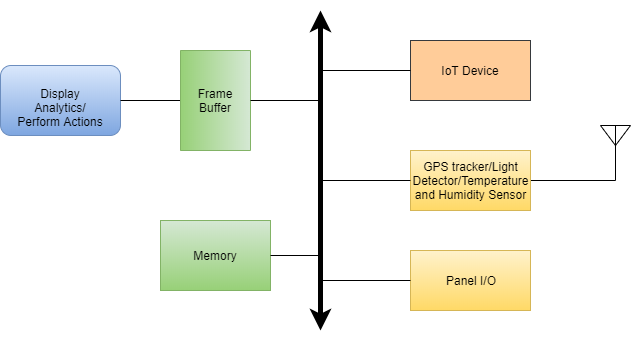
#### Temperature and Humidity/Turn AC on/off



#### Camera Module/Turn LED lights on/off

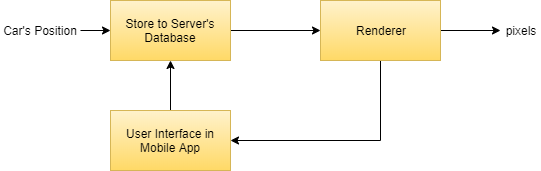


### Hardware Architecture

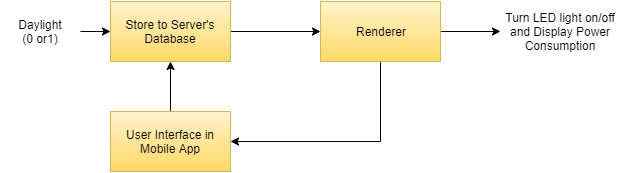


### Software Architecture

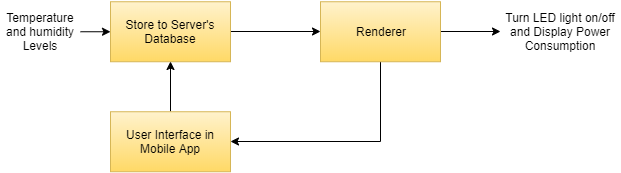
#### GPS tracker



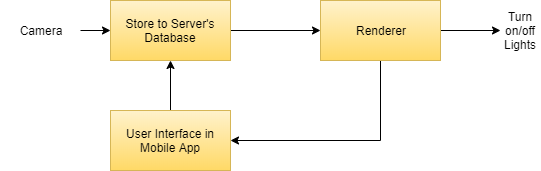
#### Light Detector/Turn LED light on/off



#### Temperature and Humidity/Turn AC on/off



#### Camera Module/Turn LED lights on/off



## Testing

To verify that our system works correctly. We will need to perform only a few checks.

First we will check that the air conditioner turns on only when the room’s temperature exceeds 28 degrees and off when it falls below 29.In order to perform this check, we have to manually put three values in the temperature, one that is below 28 to see if the air conditioner stays off, one that is 28 for the same reason, and one that is over 28 to check if the air conditioner turns on.

We also need to verify that the lights in the conference room will turn off during daylight and they will also turn on during the night. But we added a rule which checks via a camera sensor that there are people in the conference room. This rule prohibits the turning on of the lights because if there are no people in the conference room during the night then there is no need for unnecessary power consumption. To perform the check we will first check that the lights are turned off during the day, for any number of people, then we will check that the lights stay off in the night for 0 people in the conference room, and a final check that shows us that the lights turn on during the night if any number of people is present in the room.

We need to check that our graphs present the correct results based on the values of temperature, humidity and power consumption.

The final check is to ensure that each location of the messenger car during the day is shown on the map. To check that, we enter values for latitude, longitude and a timestamp in our database. The system should read them and display them on the map.

## Appendix

### Code

#### Sitemaps:

sitemap demo label="Smart Local Bank Office."

{

Frame

{

Group item=gBank label="Bank" icon="bank"

{

Switch item=Day label="Day" icon="sun"

Switch item=Night label="Night" icon="moon"

Group item=gOffice1 label="Office 001" icon="office"{

Text item=gOffice1T label="Temperature [%.2f °C]" icon="temperature"

Text item=gOffice1H label="Humidity [%d ]" icon="humidity"

Chart item=gOffice1CT period=h refresh=1000 legend=true

Chart item=gOffice1CH period=h refresh=1000 legend=true

Switch item=gOffice1AC label="Airconditioning" icon="aircond"

}

Group item=gOffice2 label="Office 002" icon="office"{

Text item=gOffice2T label="Temperature [%.1f °C]" icon="temperature"

Text item=gOffice2H label="Humidity [%d ]" icon="humidity"

Chart item=gOffice2CT period=h refresh=1000 legend=true

Chart item=gOffice2CH period=h refresh=1000 legend=true

Switch item=gOffice2AC label="Airconditioning" icon="aircond"

}

Group item=gOffice3 label="Office 003" icon="office"{

Text item=gOffice3T label="Temperature [%.1f °C]" icon="temperature"

Text item=gOffice3H label="Humidity [%d ]" icon="humidity"

Chart item=gOffice3CT period=h refresh=1000 legend=true

Chart item=gOffice3CH period=h refresh=1000 legend=true

Switch item=gOffice3AC label="Airconditioning" icon="aircond"

}

Group item=gOffice4 label="Office 004" icon="office"{

Text item=gOffice4T label="Temperature [%.1f °C]" icon="temperature"

Text item=gOffice4H label="Humidity [%.1f ]" icon="humidity"

Chart item=gOffice4CT period=h refresh=1000 legend=true

Chart item=gOffice4CH period=h refresh=1000 legend=true

Switch item=gOffice4AC label="Airconditioning" icon="aircond"

}

Group item=gOffice5 label="Office 005" icon="office"{

Text item=gOffice5T label="Temperature [%.1f °C]" icon="temperature"

Text item=gOffice5H label="Humidity [%d ]" icon="humidity"

Chart item=gOffice5CT period=h refresh=1000 legend=true

Chart item=gOffice5CH period=h refresh=1000 legend=true

Switch item=gOffice5AC label="Airconditioning" icon="aircond"

}

Group item=gHead label="Head Office" icon="office"{

Text item=gHeadT label="Temperature [%.1f °C]" icon="temperature"

Text item=gHeadH label="Humidity [%d ]" icon="humidity"

Chart item=gHeadCT period=h refresh=1000 legend=true

Chart item=gHeadCH period=h refresh=1000 legend=true

Switch item=gHeadAC label="Airconditioning" icon="aircond"

}

Group item=gMeetingRoom label="Meeting Room" icon="conference"{

Switch item=Day label="Day" icon="sun"

Switch item=Night label="Night" icon="moon"

Switch item=gMeetingRoomAC label="Airconditioning" icon="aircond"

Switch item=Lights label="Lights" icon="idea"

Text item=gMeetingRoomT label="Temperature [%.1f °C]" icon="temperature"

Text item=People label="People in Meeting [%d ]" icon="parents\_1\_1"

Text item=gMeetingRoomH label="Humidity [%d ]" icon="humidity"

Chart item=gMeetingRoomCT period=h refresh=1000 legend=true

Chart item=gMeetingRoomCH period=h refresh=1000 legend=true

}

Group item=gPowerCon label="Power Consumption" icon="flash"{

Text item=gPower label="Power Consumption [%.2f watts]" icon="flash"

Chart item=gPowerCP period=h refresh=1000 legend=true

}

Group item=gMessenger label="Messenger Car" icon="cars"{

Webview url="http://localhost/Location1.php" label="The car is here" icon="location" height=10

}

}

}

}

#### Items:

Group gOffice1 "Office 001"

Group gOffice2 "Office 002"

Group gOffice3 "Office 003"

Group gOffice4 "Office 004"

Group gOffice5 "Office 005"

Group gHead "Head Office"

Group gMeetingRoom "Meeting Room"

Group gPowerCon "Power Consumption"

Group gOffice1CT

Group gOffice2CT

Group gOffice3CT

Group gOffice4CT

Group gOffice5CT

Group gHeadCT

Group gMeetingRoomCT

Group gOffice1CH

Group gOffice2CH

Group gOffice3CH

Group gOffice4CH

Group gOffice5CH

Group gHeadCH

Group gMeetingRoomCH

Group gPowerCP

Number Bool { http="<[http://localhost/info.php:1000:REGEX(\*)]" }

Number gOffice1T "Temperature [%.2f °C]" <temperature> (gOffice1,gOffice1CT) { http="<[http://localhost/temp.php:1000:REGEX(\*)]" }

Number gOffice1H "Humidity is at [%d]" <humidity> (gOffice1,gOffice1CH) { http="<[http://localhost/humidity.php:1000:REGEX(\*)]" }

Number gOffice2T "Temperature [%.2f °C]" <temperature> (gOffice2,gOffice2CT) { http="<[http://localhost/temp.php:1000:REGEX(\*)]" }

Number gOffice2H"Humidity is at [%d]" <humidity> (gOffice2,gOffice2CH) { http="<[http://localhost/humidity.php:1000:REGEX(\*)]" }

Number gOffice3T "Temperature [%.2f °C]" <temperature> (gOffice3,gOffice3CT) { http="<[http://localhost/temp.php:1000:REGEX(\*)]" }

Number gOffice3H "Humidity is at [%d]" <humidity> (gOffice3,gOffice3CH) { http="<[http://localhost/humidity.php:1000:REGEX(\*)]" }

Number gOffice4T "Temperature [%.2f °C]" <temperature> (gOffice4,gOffice4CT) { http="<[http://localhost/temp.php:1000:REGEX(\*)]" }

Number gOffice4H "Humidity is at [%d]" <humidity> (gOffice4,gOffice4CH) { http="<[http://localhost/humidity.php:1000:REGEX(\*)]" }

Number gOffice5T "Temperature [%.2f °C]" <temperature> (gOffice5,gOffice5CT) { http="<[http://localhost/temp.php:1000:REGEX(\*)]" }

Number gOffice5H "Humidity is at [%d]" <humidity> (gOffice5,gOffice5CH) { http="<[http://localhost/humidity.php:1000:REGEX(\*)]" }

Number gHeadT "Temperature [%.2f °C]" <temperature> (gHead,gHeadCT) { http="<[http://localhost/temp.php:1000:REGEX(\*)]" }

Number gHeadH "Humidity is at [%d]" <humidity> (gHead,gHeadCH) { http="<[http://localhost/humidity.php:1000:REGEX(\*)]" }

Number gMeetingRoomT "Temperature [%.2f °C]" <temperature> (gMeetingRoom,gMeetingRoomCT) { http="<[http://localhost/temp.php:1000:REGEX(\*)]" }

Number gMeetingRoomH "Humidity is at [%d]" <humidity> (gMeetingRoom,gMeetingRoomCH) { http="<[http://localhost/humidity.php:1000:REGEX(\*)]" }

Number gPower "Power Consumption [%.2f watts]" <temperature> (gPowerCon,gPowerCP) { http="<[http://localhost/power.php:1000:REGEX(\*)]" }

Number People { http="<[http://localhost/ppl.php:1000:REGEX(\*)]" }

// Time and Weather Items

//------------------------------------------------------------------------------------------------------------------------------------------------

Switch Night

Switch Day

Switch Lights

//Airconditioning

//------------------------------------------------------------------------------

Switch gOffice1AC

Switch gOffice2AC

Switch gOffice3AC

Switch gOffice4AC

Switch gOffice5AC

Switch gHeadAC

Switch gMeetingRoomAC

#### Rules:

import org.openhab.core.types.\*

import org.openhab.core.library.items.\*

import org.eclipse.xtext.xbase.lib.\*

import java.util.Map

import java.util.Set

import org.joda.time.\*

//-------------AC ------------------

rule "AC1"

when

Item gOffice1T received update

then

if(gOffice1T.state as DecimalType > 28){

sendCommand(gOffice1AC,ON)

}else{

sendCommand(gOffice1AC,OFF)

}

end

rule "AC2"

when

Item gOffice2T received update

then

if(gOffice2T.state as DecimalType > 28){

sendCommand(gOffice2AC,ON)

}else{

sendCommand(gOffice2AC,OFF)

}

end

rule "AC3"

when

Item gOffice3T received update

then

if(gOffice3T.state as DecimalType > 28){

sendCommand(gOffice3AC,ON)

}else{

sendCommand(gOffice3AC,OFF)

}

end

rule "AC4"

when

Item gOffice4T received update

then

if(gOffice4T.state as DecimalType > 28){

sendCommand(gOffice4AC,ON)

}else{

sendCommand(gOffice4AC,OFF)

}

end

rule "AC5"

when

Item gOffice5T received update

then

if(gOffice5T.state as DecimalType > 28){

sendCommand(gOffice5AC,ON)

}else{

sendCommand(gOffice5AC,OFF)

}

end

rule "AC6"

when

Item gHeadT received update

then

if(gHeadT.state as DecimalType > 28){

sendCommand(gHeadAC,ON)

}else{

sendCommand(gHeadAC,OFF)

}

end

rule "AC7"

when

Item gMeetingRoomT received update

then

if(gMeetingRoomT.state as DecimalType > 28){

sendCommand(gMeetingRoomAC,ON)

}else{

sendCommand(gMeetingRoomAC,OFF)

}

end

//----------------Time of Day Rules-------------------------

rule "Day start"

when

Item Bool received update

then

if (Bool.state as DecimalType > 2){

Day.sendCommand(OFF)

Night.sendCommand(ON)

}else{

Day.sendCommand(ON)

Night.sendCommand(OFF)

}

end

rule "Lighting"

when

Item People received update

then

if(Bool.state as DecimalType > 2 && People.state as DecimalType > 0){

Lights.sendCommand(ON)

}else{

Lights.sendCommand(OFF)

}

end

#### RRD4J Persistent:

// persistence strategies have a name and a definition and are referred to in the "Items" section

Strategies {

// for rrd charts, we need a cron strategy

// everyMinute : "0 \* \* \* \* ?"

}

Items {

//DemoSwitch, NoOfLights, Window\_GF\_Toilet, Heating\* : strategy = everyChange, everyMinute, restoreOnStartup

// let's only store temperature values in rrd

//Temperature\*, Weather\_Chart\* : strategy = everyMinute, restoreOnStartup

gPower,gPowerCP,gOffice1T,gOffice1H,gOffice1CT,gOffice1CH,gOffice2T,gOffice2H,gOffice2CT,gOffice2CH,gOffice3T,gOffice3H,gOffice3CT,gOffice3CH,gOffice4T,gOffice4H,gOffice4CT,gOffice4CH,gOffice5T,gOffice5H,gOffice5CT,gOffice5CH,gHeadT,gHeadH,gHeadCH,gHeadCT,gMeetingRoomT,gMeetingRoomH,gMeetingRoomCT,gMeetingRoomCH : strategy = everyChange, restoreOnStartup

}

// vim: syntax=Xtend

#### MYSQL Persistent:

Strategies{

everyMinute : "0 \* \* \* \* ?"

default = everyChange

}

Items{

gPower,gOffice1T,gOffice1H,gOffice2T,gOffice2H,gOffice3T,gOffice3H,gOffice4T,gOffice4H,gOffice5T,gOffice5H,gHeadT,gHeadH,gMeetingRoomT,gMeetingRoomH,gOffice1CT,gOffice1CH : strategy = everyChange, restoreOnStartup

}

#### Location:

<!doctype html>

<html lang="en">

<head>

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/gh/openlayers/openlayers.github.io@master/en/v6.0.1/css/ol.css" type="text/css">

<style>

.map {height: 400px; width: 100%;}

.ol-attribution.ol-logo-only,

.ol-attribution.ol-uncollapsible {

max-width: calc(100% - 3em) !important;

height: 1.5em !important;

}

.ol-control button,

.ol-attribution,

.ol-scale-line-inner {

font-family: 'Lucida Grande', Verdana, Geneva, Lucida, Arial, Helvetica, sans-serif !important;

}

.ol-popup {

font-family: 'Lucida Grande', Verdana, Geneva, Lucida, Arial, Helvetica, sans-serif !important;

font-size: 12px;

position: absolute;

background-color: white;

-webkit-filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));

filter: drop-shadow(0 1px 4px rgba(0, 0, 0, 0.2));

padding: 15px;

border-radius: 10px;

border: 1px solid #cccccc;

bottom: 12px;

left: -50px;

min-width: 100px;

}

.ol-popup:after,

.ol-popup:before {

top: 100%;

border: solid transparent;

content: " ";

height: 0;

width: 0;

position: absolute;

pointer-events: none;

}

.ol-popup:after {

border-top-color: white;

border-width: 10px;

left: 48px;

margin-left: -10px;

}

.ol-popup-closer:after {

content: "✖";

position: absolute;

top: 3px;

right: 2px;

font-size: 100%;

color: #0088cc;

text-decoration: none;

}

.ol-popup:before {

border-top-color: #cccccc;

border-width: 11px;

left: 48px;

margin-left: -11px;

</style>

<script

src="https://cdn.jsdelivr.net/gh/openlayers/openlayers.github.io@master/en/v6.0.1/build/ol.js"></script>

<title>Smart Local Bank Office</title>

</head>

<body>

<div id="map" class="map"></div>

<div id="popup" class="ol-popup">

<a href="#" id="popup-closer" class="ol-popup-closer"></a>

<div id="popup-content"></div>

</div>

<script src="http://code.jquery.com/jquery-latest.js"></script>

<script src="http://code.jquery.com/jquery-latest.js"></script>

<script type="text/javascript">

//https://openstreetmap.be/en/projects/howto/openlayers.html

var stroke = new ol.style.Stroke({color: 'yellow', width: 4});

var fill = new ol.style.Fill({color: 'red'});

var styles = {'square': new ol.style.Style({ image: new ol.style.RegularShape({ fill: fill, stroke: stroke,

points: 4,radius: 10,angle: Math.PI / 4})}),

'triangle': new ol.style.Style({ image: new ol.style.RegularShape({

fill: fill,stroke: stroke, points: 3, radius: 10, rotation: Math.PI / 4, angle: 0 }) }),

'star': new ol.style.Style({image: new ol.style.RegularShape({ fill: fill, stroke: stroke,

points: 5, radius: 10, radius2: 4, angle: 0 }) }),

'cross': new ol.style.Style({ image: new ol.style.RegularShape({ fill: fill, stroke: stroke, points: 4, radius: 10,

radius2: 0, angle: 0 }) }),

'x': new ol.style.Style({ image: new ol.style.RegularShape({ fill: fill, stroke: stroke, points: 4, radius: 10,

radius2: 0, angle: Math.PI / 4 }) }) };

var vectorSource = new ol.source.Vector();

var map = new ol.Map({ target: 'map', layers: [ new ol.layer.Tile({ source: new ol.source.OSM() }) ,

new ol.layer.Vector({source : vectorSource})],

view: new ol.View({ center: ol.proj.fromLonLat([<?php

$file = 'myLastLocation.txt';

$lines = file($file);

echo end($lines);

?>]), zoom: 15 }) });

jQuery(document).ready(function($){

resp = $("#response");

$.ajax({

type: "POST", // Method type GET/POST

url: "data1.php", //Ajax Action url

dataType:"json",

// Before call ajax you can do activity like please wait message

beforeSend: function(xhr){

resp.html("Please wait...");

},

//Will call if method not exists or any error inside php file

error: function(qXHR, textStatus, errorThrow){ resp.html("There are an error");

}, success: function(data, textStatus, jqXHR){

// resp.html(data);

var i;

var Feature=[];

for(i=0;i<data.length;i++){ Feature[i]=new ol.Feature({timestamp:data[i].Time,geometry: new ol.geom.Point(ol.proj.fromLonLat([data[i].Longitude,data[i].Latitude]))}); Feature[i].setStyle(styles["star"]);

vectorSource.addFeature(Feature[i]);

}

var container = document.getElementById('popup'); var content = document.getElementById('popup-content'); var closer = document.getElementById('popup-closer');

var overlay = new ol.Overlay({ element: container,

autoPan: true,

autoPanAnimation: {

duration: 250

}});

map.addOverlay(overlay); closer.onclick = function() { overlay.setPosition(undefined);

closer.blur();

return false;

};

map.on('singleclick', function (event) {

if (map.hasFeatureAtPixel(event.pixel) === true) {

var coordinate = event.coordinate;

var feature = map.getFeaturesAtPixel(event.pixel)[0];

var t=feature.get('timestamp');

content.innerHTML = '<b>Car ID 1</b><br/><code>'+t+'</code>';

overlay.setPosition(coordinate);

} else {

closer.blur();

}});

}});

});

</script>

</body>

</html>

#### Data:

<?php

$servername="localhost";

$username = "root";

$password = "1234";

$dbname = "OpenHAB";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql="SELECT CarID,Time,Latitude,Longitude FROM Car WHERE CarID=1 AND Time > DATE\_SUB(NOW(),INTERVAL 1 DAY)";

$result=$conn->query($sql);

while($r = mysqli\_fetch\_assoc($result)) {

$rows[] = $r;

}

print json\_encode($rows);

$conn->close();

?>

### Screenshots:

