LightPDF

Mini Smart Home System

Source Code

***Abstract*—Over the years electronic devices have become more accessible and widely used in almost every aspect of our lives. This has led to what is called Internet of Things (IoT), which consists of several physical devices (or groups of such objects) that are embedded with sensors, processing ability, software, and can interact with other devices over the internet or other communication networks [1]. IoT technologies are used for various kinds of applications, such as healthcare systems, traffic monitoring, agriculture, smart grid, water supply and smart home systems. Since the smart home system is the most accessible application among the IoT applications because it does not require a certain skillset and already existing home automation systems can be used (e.g Google Home, Mi Home). Thus, more and more people start experimenting on such projects and a constantly expanding community emerged.**

***Index Terms*—Smart home, Apache Kafka, Node Red, Grafana, RDBMS, Telegram-bot, NodeJS**

LightPDF

I. Introduction

This project was part of the Analysis and Design of Information Systems course at NTUA and it is concerned with the development of a Smart Home System. More specifically, we implemented this system that covers the basic needs of a fictional character. The reason behind this, is that we needed a schedule of the person that would live in the home that the system would be installed. Furthermore, no real IoT devices were used in this project, so our team had to create data that would simulate the behaviour of the IoT devices and then feed the system to take the respective actions.

II. Diagrams

Fig. 1 shows the Component Diagram of the project and Fig.2 2 shows the Deployment Diagram of the project

III. Technology Stack

- *Apache Kafka*
  Used as message broker.
- *Node-Red*
  Used to handle the logic of our smart home.
- *NodeJS and ExpressJS*
  Used to create a server and specifically a route to handle all incoming requests from the smart home's devices.
- *MySQL*
  Used as the database.
- *Grafana*
  Used as a simple user dashboard where the user (smart home owner) can see many details and monitor the IOT devices.
- *Telegram*
  Used to create a bot that will send notifications (more info in upcoming sections).
- *Python3*
  Used to create dummy data that represent the data from the smart home sensors and to create a simulation script which makes many API calls to the express server.

IV. Tools

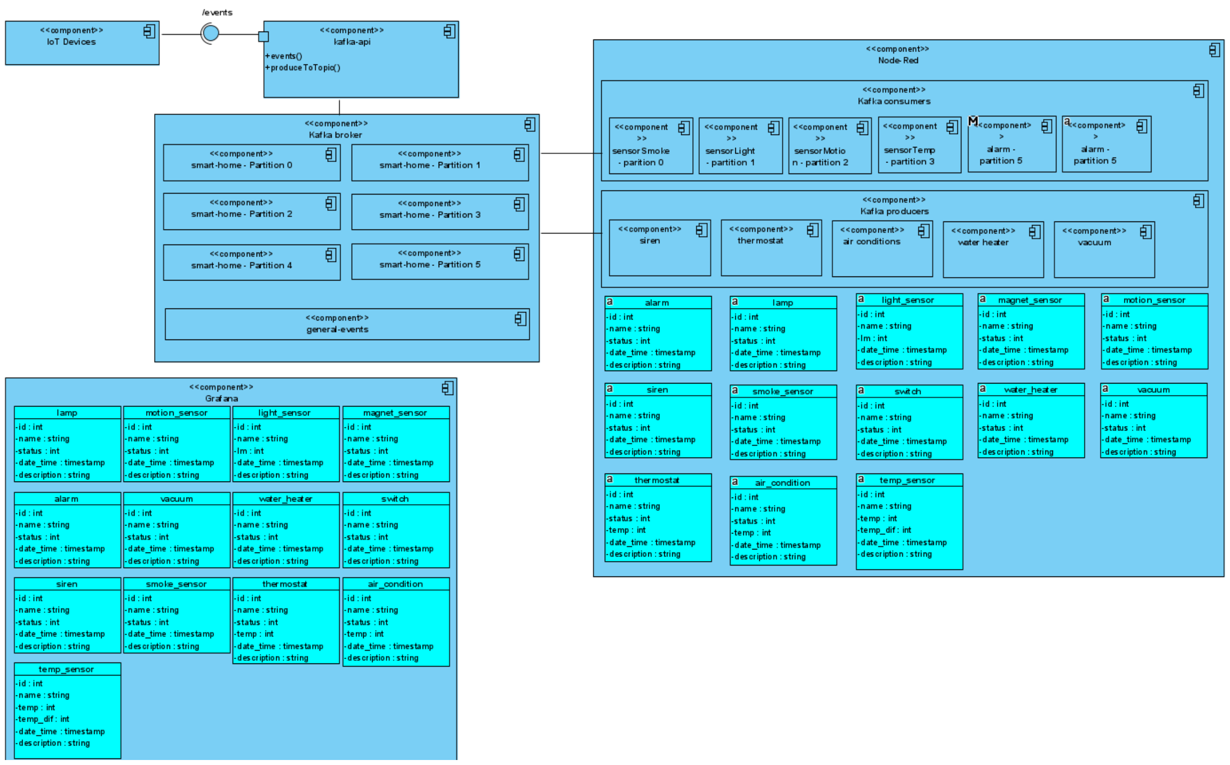
- *Docker*
  Used to create a container for the database.
- *NVM*
  Used as the Node Version Manager.
- *FlyWay*
  Used as a database version control. Used for matters of ease when developing the system.
- *GitHub*
  Used in order to keep track the versions of the code and better collaboration

V. Kafka

Apache Kafka was used as a middleware for the communication between the smart-home events and node-red. Kafka uses topics to store incoming events. In order to achieve that, three topics were created. The house consists of six major device categories. The name of each device is the concatenation of the device type and the room that it is located in.

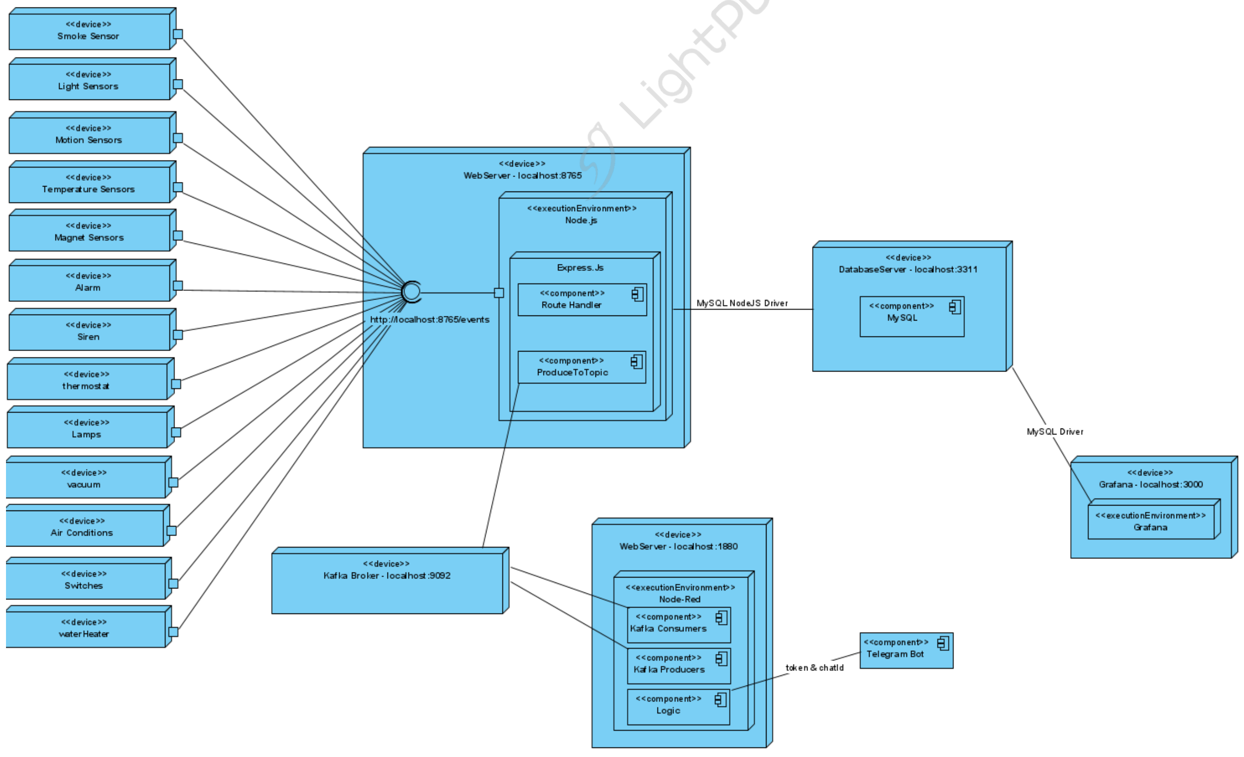
The smart-home events are stored in the smart-home topic with device type as key and some other device properties as value. By default Kafka stores events in topics using a Round-Robin partitioner. However, a manual approach was chosen for this project by producing each device event to specific partition according to its device category. This approach was chosen due to the fact that the node-red-contrib-kafka library, which is used to connect Apache Kafka and Node-red, enables

LightPDF



/events
<<component>> IoT Devices
<<component>> kafka-api
+events()
+produceToTopic()
<<component>> Kafka broker
<<component>> smart-home - Partition 0
<<component>> smart-home - Partition 1
<<component>> smart-home - Partition 2
<<component>> smart-home - Partition 3
<<component>> smart-home - Partition 4
<<component>> smart-home - Partition 5
<<component>> general-events
<<component>> Node-Red
<<component>> Kafka consumers
<<component>> sensorSmoke - partition 0
<<component>> sensorLight - partition 1
<<component>> sensorMotion - partition 2
<<component>> sensorTemp - partition 3
<<component>> alarm - partition 5
<<component>> alarm - partition 5
<<component>> Kafka producers
<<component>> siren
<<component>> thermostat
<<component>> air conditions
<<component>> water heater
<<component>> vacuum
<<component>> Grafana
alarm
lamp
light_sensor
magnet_sensor
motion_sensor
siren
smoke_sensor
switch
water_heater
vacuum
thermostat
air_condition
temp_sensor
-id : int
-name : string
-status : int
-lm : int
-temp : int
-temp_dif : int
-date_time : timestamp
-description : string

Fig. 1. Component Diagram



<<device>> Smoke Sensor
<<device>> Light Sensors
<<device>> Motion Sensors
<<device>> Temperature Sensors
<<device>> Magnet Sensors
<<device>> Alarm
<<device>> Siren
<<device>> thermostat
<<device>> Lamps
<<device>> vacuum
<<device>> Air Conditions
<<device>> Switches
<<device>> water Heater
http://localhost:8765/events
<<device>> WebServer - localhost:8765
<<executionEnvironment>> Node.js
Express.Js
<<component>> Route Handler
<<component>> ProduceToTopic
MySQL NodeJS Driver
<<device>> DatabaseServer - localhost:3311
<<component>> MySQL
MySQL Driver
<<device>> Grafana - localhost:3000
<<executionEnvironment>> Grafana
<<device>> Kafka Broker - localhost:9092
<<device>> WebServer - localhost:1880
<<executionEnvironment>> Node-Red
<<component>> Kafka Consumers
<<component>> Kafka Producers
<<component>> Logic
token & chatId
<<component>> Telegram Bot

Fig. 2. Deployment Diagram

LightPDF

consuming messages from a certain partition of a topic. Nevertheless, had this been a real large scale project, Kafka would handle the partitioning automatically.

Furthermore, there are two additional topics, general-events and lamps. In the first one, messages produced are triggered by node-red logic and more specifically events created by output devices(e.g siren). In the second one, each time the user turns on/off a lamp, a message is produced to the certain topic containing the action as value and room as key.

VI. INSTALLATION

A. *Clone Project*

- Git clone the project
- Open the folder that were git was cloned and navigate to kafka-api folder
- Open a terminal and execute npm install

B. *NodeJS*

- Download NVM with command: *curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash*
- Install NodeJS

  nvm install 16

C. *Apache Kafka*

- Download Apache Kafka: https://dlcdn.apache.org/kafka/3.0.0/kafka_2.13-3.0.0.tgz
- Unzip it to the desired folder i.e Home (cd ~/Kafka)
- Navigate to cd ~/Kafka and create 2 folders, data and logs
- Open vim ~/Kafka/kafka_2.13-3.0.0/config/zookeeper.properties, and change dataDir to the directory where data folder is. In our case dataDir=~/Kafka/data
- Open vim ~/Kafka/kafka_2.13-3.0.0/config/server.properties, and change log.dirs to the directory where logs folder is. In our case log.dirs=~/Kafka/logs
- Create aliases to use kafka easier. Navigate to cd ~ and vim .bashrc. Find the #some more aliases comment and below add:

```
#kafka-aliases
#start zookeeper
alias zookeeper="bin/zookeeper-server-start.sh config/zookeeper.properties"
#start kafka
alias kafka="bin/kafka-server-start.sh config/server.properties"
#topics commands
alias list-topics="bin/kafka-topics.sh --bootstrap-server localhost:9092 --list"
alias describe-topic="bin/kafka-topics.sh --bootstrap-server localhost:9092 --describe --topic"
alias consume-topic="bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic"
alias create-topic="bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic"
alias delete-topic="bin/kafka-topics.sh --bootstrap-server localhost:9092 --delete --topic"
#grafana
alias grafana="sudo systemctl start grafana-server"
localhost:9092 --describe --topic"
alias consume-topic="bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic"
alias create-topic="bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic"
alias delete-topic="bin/kafka-topics.sh --bootstrap-server localhost:9092 --delete --topic"
#grafana
alias grafana="sudo systemctl start grafana-server"
```

- Open a terminal and navigate to cd ~/Kafka/kafka_2.13-3.0.0. Execute the command *zookeeper*
- Open a new terminal and navigate again to cd ~/Kafka/kafka_2.13-3.0.0. Execute the command *kafka* as soon as zookeeper is finished
- Open a new terminal and navigate again to cd ~/Kafka/kafka_2.13-3.0.0. Execute the command *create-topic [name] --partitions [number] --replication-factor 1*. In our case the name is smart-home and number is 6

D. *Docker*

- Install docker with: *sudo snap install docker*

E. *MySQL - Dbeaver - Flyway*

- Open a new terminal and execute the command: *docker run -p 3311:3306 --name smart-home-db -e MYSQL_ROOT_PASSWORD=root -d mysql:5.7*
- Use a database management tool. In our case we used dbeaver which you can install by executing: *sudo snap install dbeaver-ce*
- Create a new database using the database management tool. The name of the database for this project is smart_home
- Install flyway: *Wget -qO- https://repo1.maven.org/maven2/org/flywaydb/flyway-commandline/8.4.1/flyway-commandline-8.4.1-linux-x64.tar.gz | tar xvz && sudo ln -s pwd/flyway-8.4.1/flyway /usr/local/bin*
- Create the tables needed with Flyway (where user=root, password=root, port=3311, database_name=smart_home) *flyway -user=[user] -password=[password] -url=jdbc:mysql://localhost:[port]/[database_name]*
- Open the project folder, where sql folder is located. Open a terminal and execute: *flyway -user=root -password=root -url=jdbc:mysql://localhost:3311/smart_home migrate*

In this point, clean can be used to clear the database and migrate to create it again.

F. *Node-Red*

- Install Node-Red. Open a new terminal and execute the command: *sudo snap install node-red*

LightPDF

LightPDF

- In the previous terminal execute the command which will start node-red: *node-red.desktop-launch*
- Open a browser and navigate to http://localhost:1880/. Here you should see node-red running
- From the top right corner press the hamburger menu and choose import. Choose select a file to import and locate node-red-flow.json which is in the folder which you git cloned

G. *Node-Red Dependencies*

- For the Bot: node-red-contrib-chatbot
- For Kafka: node-red-contrib-kafka-manager
- For MySQL: node-red-node-mysql
- For Node-Red Dashboard: node-red-dashboard

H. *Grafana*

- Install Grafana:
  wget https://dl.grafana.com/enterprise/release/grafana-enterprise_8.3.3_amd64.deb
  sudo dpkg -i grafana-enterprise_8.3.3_amd64.deb
- One of those aliases that were set during the installation for Apache Kafka is for Grafana. So now in order to start Grafana, open a terminal and execute: *grafana*
- Open a browser in http://localhost:3000/ to find see the login screen of Grafana
- Enter the default user and password which are both the word admin
- Import the json file that is the dashboard. This file is located in the project folder and it is called grafana-dashboard.json. To import it, there is a plus icon on your left. Follow the instructions and the dashboard will be imported
- If everything went well, you will see the dashboard in the General Tab

I. *Telegram*

- Download Telegram *snap install telegram-desktop*
- Bot was created using this tutorial [2]
- After that a group conversation was created (which includes all the family members that will get notified for alerts in the smart home)
- We used the bot's token which can be found by following the tutorial [2] or [3]

J. *Python3*

Python3 was used for the creation of the simulation data and the simulation script. These scripts are located in the project folder and specifically in data folder.

Python3 should already be preinstalled on Ubuntu, ensure that Python3 is installed on the running device before proceeding.

VII. DEVICES

Because this project will eventually be a simulation of a smart home system and for demonstration purposes, virtual devices are used from which the simulation data are presumably generated and transferred to the system. These devices are placed in different rooms around the house. The room of the smart home can be found on Fig. 3. The devices used can be found on Fig. 4. After the placement in the house, various scenarios were designed that trigger some specific flows in NodeRed.

TABLE I. ROOMS OF SMART HOME

Room ID	Room
0	Bedroom
1	Living Room
2	Kitchen
3	Bathroom
4	Balcony

Fig. 3. Rooms of the smart home and their respective ids

TABLE II. DEVICES

Device	Name	Value	Value Range
Smoke sensor	sensorSmoke	bool state	binary
Light sensor	sensorLight0	int lm	range 0…1000 (evening: 0…100, lighting: 100…500,sunshine: >500)
Light sensor	sensorLight1	int lm	"
Light sensor	sensorLight4	int lm	"
Motion sensor	sensor Motion 1	bool state	binary
Motion sensor	sensor Motion 4	bool state	binary
Temperature sensor	sensorTemp0	int temp, int tempDif	temp: range 10...35, tempDif example: +/- 4
Temperature sensor	sensorTemp1	int temp, int tempDif	"
Window/Door magnet	sensor Magnet 0	bool state	binary
Window/Door magnet	sensor Magnet 1	bool state	"
Window/Door magnet	sensor Magnet 2	bool state	"
Window/Door magnet	sensor Magnet 3	bool state	"
Alarm	alarm	bool state	binary

LightPDF

Device	Name	Value	Value Range
Siren	siren	bool state	binary
Thermo stat	thermo stat	bool state, int temp	state: binary, temp: range 10...35
Lamp	lamp0	bool state	binary
Lamp	lamp1	bool state	"
Lamp	lamp2	bool state	"
Lamp	lamp3	bool state	"
Lamp	lamp4	bool state	"
Vacuu m	vacuum	bool state	binary
Air-Conditi oner	ac0	bool state, int temp	state: binary, temp: range 18...30
Air-Conditi oner	ac1	bool state, int temp	"
Water Heater	waterH eater	bool state	binary
Switch - Phone charger	switch0	bool state	binary
Switch - Coffee machin e	switch1	bool state	"

Fig. 4. Devices (*Each device that appears in the table consists of its name plus a number. The number is in which room this device belongs. If no number exists, there's only one device of that kind in the house*)

VIII. Scenario

Because this is a simulation, a scenario was created based on the life of an imaginary person, John.

John works on weekdays. More specifically:

- He wakes up at 07:30
- Eats breakfast at 08:00
- Works from 09:00 to 17:00
- He arms the alarm before leaving the house and disarms the alarm when he returns
- Takes a shower at 17:30
- Eats dinner at 20:00
- Sleeps at 24:00

On the weekends he has a more relaxed schedule:

- He wakes up at 09:00
- He sleeps at 02:00

On a daily basis, John has some demands for this smart home system. More specifically he wants:

- Water heater to start boiling water at 17:00
- Vacuum to clean every Monday - Wednesday - Friday from 10:00 to 12:00
- His phone to get charged from 03:30 to 06:30
- To have his coffee ready by 07:45
- Living room lights to be turned on when it is getting dark and he enters the apartment
- Balcony lights to be turned on when it is getting dark and motion is detected on the balcony
- Lights in the bedroom, bathroom and kitchen to be switched off when he arms the alarm
- Thermostat to be set on when the temperature drops below 22 degrees
- Air-conditions to be turned on when the temperature rises above 29 degrees
- Door/window openings to be logged to the database

Also in case of an emergency, he wants to be notified from a bot. Such emergencies are:

- Fire
- Burglary
- Possible malfunction of any temperature sensor in the smart system

Based on the above a script was created, that simulates John's life for a period of a week. Unexpected events throughout the week:

- On Tuesday at 13:00 burglars broke into John's house using the door in the living room
- On Friday at 20:00 there was a fire in John's kitchen
- On Sunday at 16:00 a temperature sensor malfunctioned

IX. Conclusions

All in all, smart home systems have several benefits. First of all, they provide security, as the user can be immediately notified for any potential harmful situations. Moreover, it is an efficient way to save money, by optimizing energy consumption and time by automating numerous routine procedures. Also, a smart home system can be customized on a user's needs and can easily be changed/upgraded. Finally, the low cost of most IoT devices make the smart home system an affordable and approachable solution to everyone.

On the other hand, a small malfunction of any IoT device can cause serious problems to the system. Additionally, the system is vulnerable to malicious attacks (someone can connect to your network and have access to every IoT device). This can be avoided by using IoT devices that are not connected into the local network, but use other communication protocols (frequency protocol).

Acknowledgment

The authors would like to thank Prof. Vassiliki (Verena) Kantere, Prof. Dimitrios Tsoumakos and Paraskevas Kerasiotis.

References

[1] https://en.wikipedia.org/wiki/Internet_of_things

[2] https://sendpulse.com/knowledge-base/chatbot/create-telegram-chatbot

[3] https://github.com/guidone/node-red-contrib-chatbot/wiki/Telegram-Receiver-node

[4] https://medium.com/event-driven-utopia/understanding-kafka-topic-partitions-ae40f80552e8

LightPDF