Development of a Smart-Home Intrusion Alarm using an Emulated IoT Environment

Table of contents

Introduction

Internet of Things (IoT) is basically the concept of establishing communication between things of the real world through the internet, mainly devices that can sense the environment and do something in response. Similarly to the HTTP protocol, IoT uses the Constrained Application Protocol (CoAP) designed for machine-to-machine (m2m) applications. Here I have developed an intrusion alarm to detect any unusual entrance at home with five different sensors. Californium (Cf) framework is used for building the IoT application exclusively in java environment with the help of Spring Boot framework as a Maven Project. I will be discussing further the components and workflow in the upcoming articles.

Constrained application protocol(CoAP)

“The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in IoT. The protocol for machine-to-machine (M2M) applications such as smart energy and building automation”, quoted by the CoAP Technical team which is quite a description, to begin with. CoAP is an Application Protocol for Constrained Nodes/Networks. It is intended to provide RESTful services not unlike HTTP while reducing the complexity of implementation as well as the size of packets exchanged in order to make these services useful in a highly constrained network of themselves highly constrained nodes.

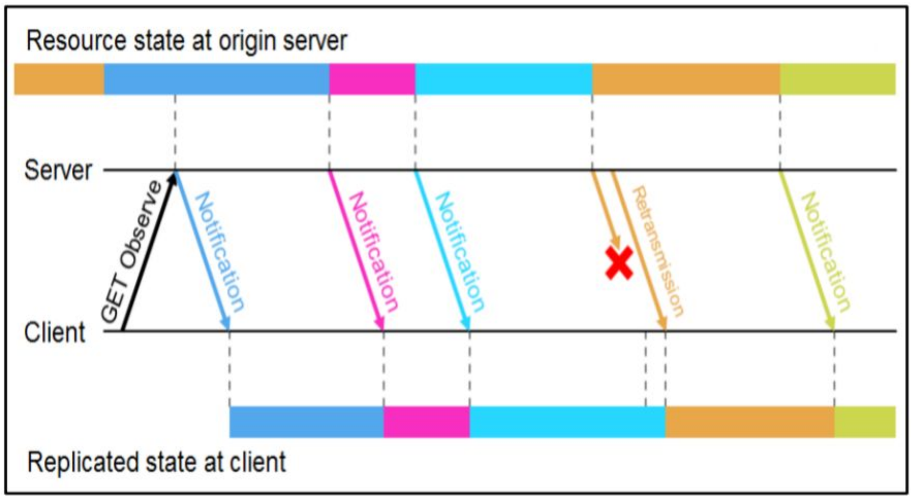


Figure: CoAP observer state

CoAP Observer

The state of a resource on a CoAP server can change over time. It has given the clients the ability to observe this change. However, existing approaches from the HTTP world, such as repeated polling or long-polls, generate significant complexity and/or overhead and thus are less applicable in the constrained CoAP world. Instead, a much simpler mechanism is provided to solve the basic problem of resource observation

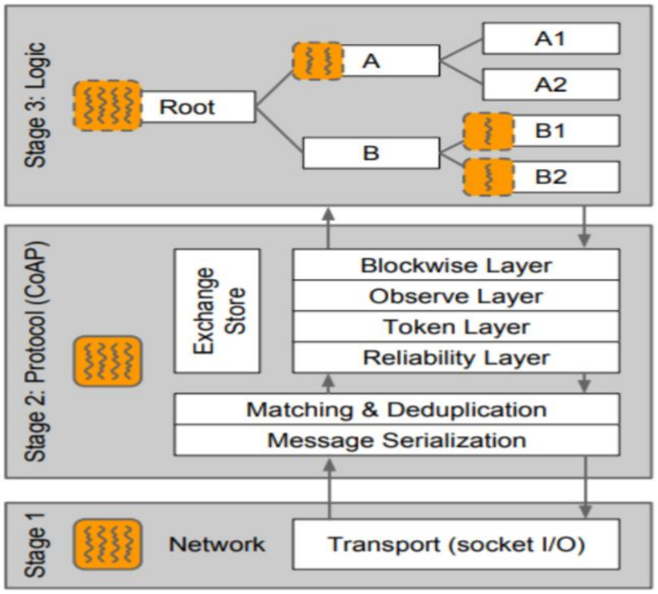


Figure: Californium as a Server Role.

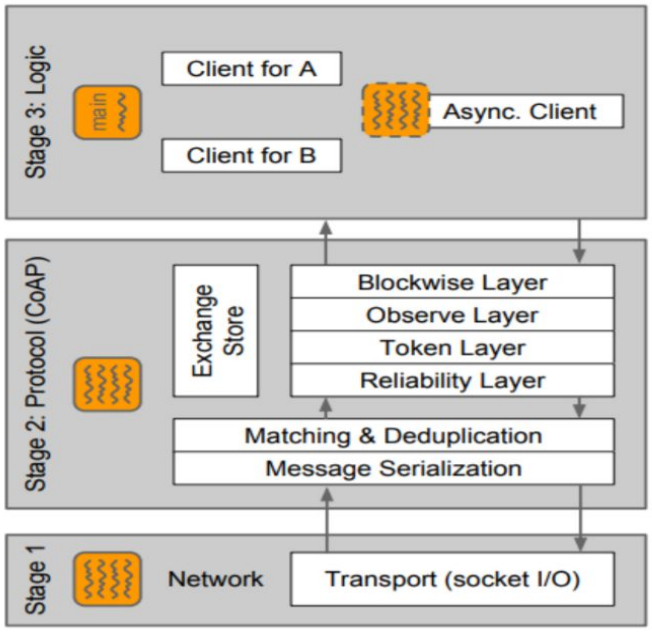


Figure: Californium as Client Role

Californium Framework

This framework is developed exclusively in java focusing on scalability and usability. It has made it easier to build a CoAP server and the corresponding client reducing the stress of lines of codes that need to be written in java environment. Figure 4 & 5 illustrates the architecture well.

Project overview

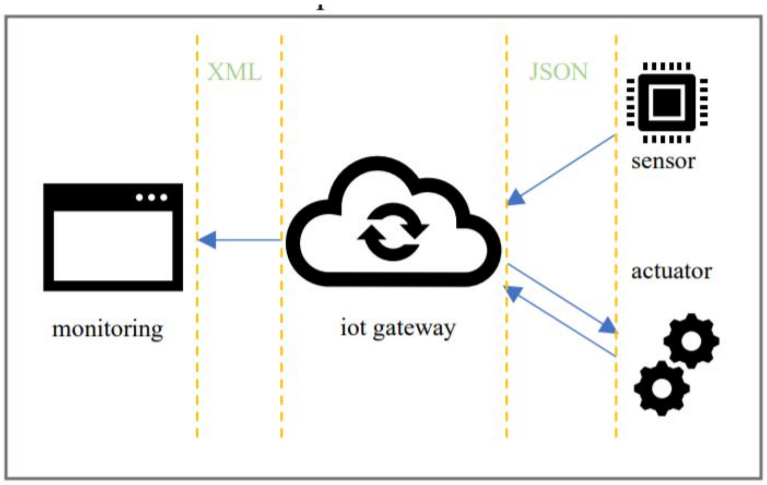


Figure: Application architecture

- ### IntrusionAlarmApplication

  This is the main class and our applications start from here. In the first place, we initialize five Coap servers in five different ports. After that, in each port we initialize the coap client. From this CoAP client we can now observe the responses.

- ### IntrusionAlarmController

  This class controls the urls. We are setting the homepage, landing page, urls for each sensor. Home url gives us the homepage of our application. The other urls are for the response body, they do not give us any html page, rather than a response body.

- SensorStates

  Sensors can not update the response data directly. They must have to use a middle man to process this response. So, every sensor updates the value of SensorStates and later we just put the same value in the ResponseData.

- HeadCountSensor

  I have introduced a headcount sensor which will be set in the front door. It will calculate the people inside the house doing the subtraction of entry and exit. For this project we set this sensor to give us random value between 0 to 5 persons. It works every 5 seconds.

- LaserSensor

  I have implemented some laser sensors virtually. For this project, it will give us the response as True/ False when there is any interruption in the laser beam. I have set a timer and after some time it will check the laser beam. According to this, it will set the response as true or false. After each checking it updates the value of the SensorStates.

- VibratingSensor

  A sensor named VibratingSensor is introduced in the front door. If you check the pressure and vibration in the lock and give us the value every 3 seconds. For this project, I have sent the random value in a range 0 to 50.

- Home

  At the beginning of the application, we redirect a landing page called home. Here we designed a page which shows all the sensors information individually. An ajax call is initiated to hit a specific url and it will get the information from the sensor gateway. Every two seconds, this method is called. I did the processing of the logical part of the alarm message in this html page after getting every new response from the gateway.

Use cases

- Use case 1

  There is none at home, Burglar will break one of the windows and enter the house. We will use laser sensor to detect this. While entering through the window, the laser beam will be interrupted and it will notify the server.
  Person inside = 0;
  Laser sensor will be broken while getting any interruption.

- Use case 2

  There is none at home, Burglar will try to break the front door or lock. We will use a vibration sensor or the pressure sensor to detect this kind of activity. A sensor is placed in the door look. If someone put more than a predefined pressure on the door lock, then it will notify us.
  Person inside = 0;
  Pressure sensor will give us notification if there is more pressure than a predefined value.

- Use case 3

  There are people at home, maybe at night or daytime, Burglar will try to break the front door or lock. We will use a vibration sensor or the pressure sensor to detect this kind of activity. A sensor is placed in the door look. If someone put more than a predefined pressure on the door lock, then it will notify us even if there are people inside.
  Pressure sensor will give us notification if there is more pressure than a predefined value.

Protocols

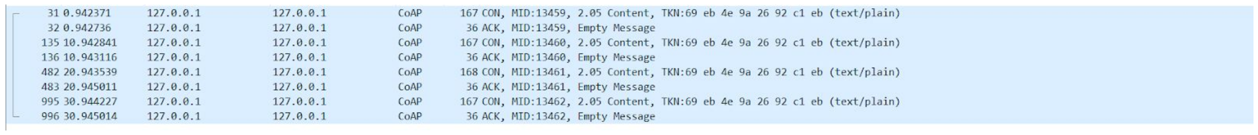
- IoT

  The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

The definition of the Internet of Things has evolved due to the convergence of multiple technologies, real-time analytics, and embedded systems. Traditional fields of embedded systems, wireless sensor networks, control systems and others all contribute to enabling the Internet of Things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the 'smart home', covering devices and appliances (such as home security systems and cameras) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones..

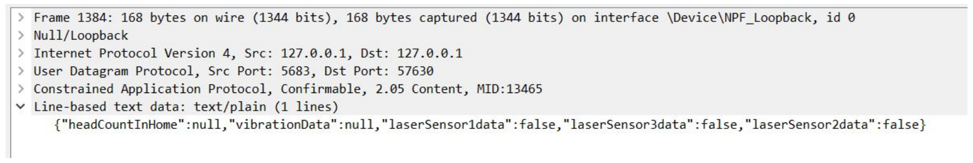
- CoAP

To sense the sensor data we used the CoAP protocol. The trace is captured in Wireshark and the screenshot is given below.



```
 31 0.942371   127.0.0.1   127.0.0.1   CoAP   167 CON, MID:13459, 2.05 Content, TKN:69 eb 4e 9a 26 92 c1 eb (text/plain)
 32 0.942736   127.0.0.1   127.0.0.1   CoAP    36 ACK, MID:13459, Empty Message
135 10.942841  127.0.0.1   127.0.0.1   CoAP   167 CON, MID:13460, 2.05 Content, TKN:69 eb 4e 9a 26 92 c1 eb (text/plain)
136 10.943116  127.0.0.1   127.0.0.1   CoAP    36 ACK, MID:13460, Empty Message
482 20.943539  127.0.0.1   127.0.0.1   CoAP   168 CON, MID:13461, 2.05 Content, TKN:69 eb 4e 9a 26 92 c1 eb (text/plain)
483 20.945011  127.0.0.1   127.0.0.1   CoAP    36 ACK, MID:13461, Empty Message
995 30.944227  127.0.0.1   127.0.0.1   CoAP   167 CON, MID:13462, 2.05 Content, TKN:69 eb 4e 9a 26 92 c1 eb (text/plain)
996 30.945014  127.0.0.1   127.0.0.1   CoAP    36 ACK, MID:13462, Empty Message
```

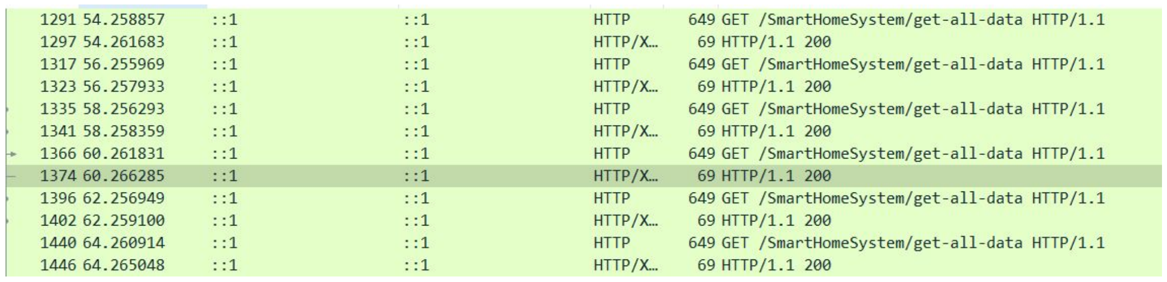
At the second stage, the capture of a frame is also given. It is a line based text and a json format of the response. This json format is the outcome of the sensors.



```
> Frame 1384: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 5683, Dst Port: 57630
> Constrained Application Protocol, Confirmable, 2.05 Content, MID:13465
v Line-based text data: text/plain (1 lines)
    {"headCountInHome":null,"vibrationData":null,"laserSensor1data":false,"laserSensor3data":false,"laserSensor2data":false}
```

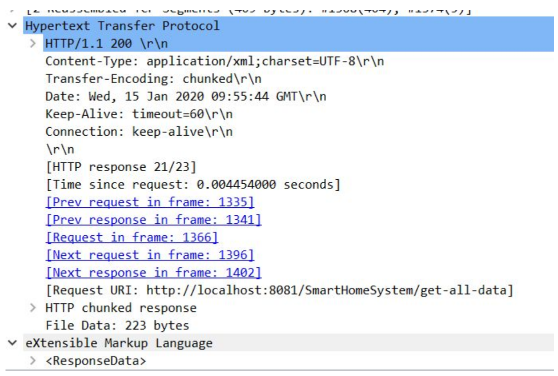
- HTTP

From the sensor gateway to application layer, HTTP protocol is used. The trace in the Wireshark is captured and it is given below.

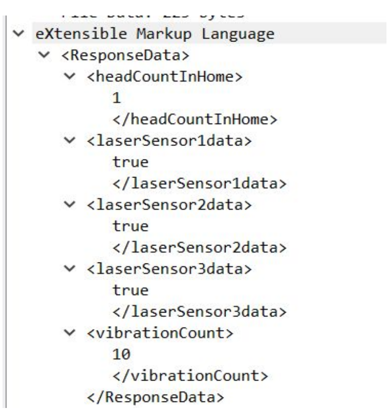


```
1291 54.258857   ::1   ::1   HTTP     649 GET /SmartHomeSystem/get-all-data HTTP/1.1
1297 54.261683   ::1   ::1   HTTP/X...  69 HTTP/1.1 200
1317 56.255969   ::1   ::1   HTTP     649 GET /SmartHomeSystem/get-all-data HTTP/1.1
1323 56.257933   ::1   ::1   HTTP/X...  69 HTTP/1.1 200
1335 58.256293   ::1   ::1   HTTP     649 GET /SmartHomeSystem/get-all-data HTTP/1.1
1341 58.258359   ::1   ::1   HTTP/X...  69 HTTP/1.1 200
1366 60.261831   ::1   ::1   HTTP     649 GET /SmartHomeSystem/get-all-data HTTP/1.1
1374 60.266285   ::1   ::1   HTTP/X...  69 HTTP/1.1 200
1396 62.256949   ::1   ::1   HTTP     649 GET /SmartHomeSystem/get-all-data HTTP/1.1
1402 62.259100   ::1   ::1   HTTP/X...  69 HTTP/1.1 200
1440 64.260914   ::1   ::1   HTTP     649 GET /SmartHomeSystem/get-all-data HTTP/1.1
1446 64.265048   ::1   ::1   HTTP/X...  69 HTTP/1.1 200
```

While using this protocol, only xml type data is passed and we can see the eXtensible Markup Language tag. In the next, an xml type response data is also shown



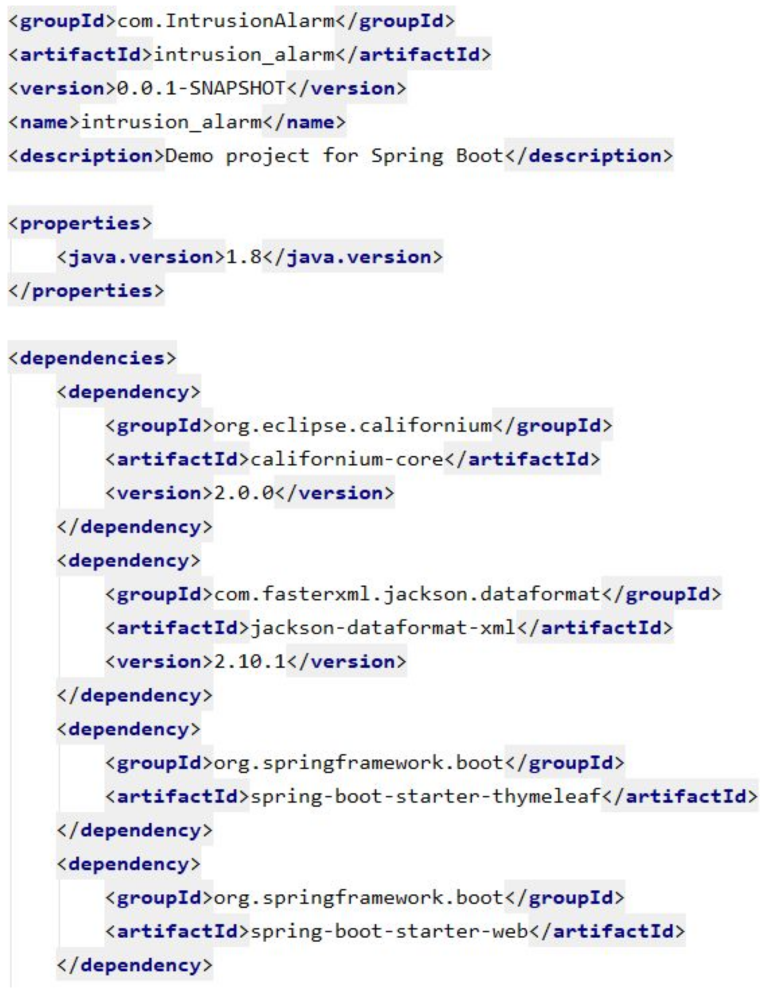
```
v Hypertext Transfer Protocol
  > HTTP/1.1 200 \r\n
    Content-Type: application/xml;charset=UTF-8\r\n
    Transfer-Encoding: chunked\r\n
    Date: Wed, 15 Jan 2020 09:55:44 GMT\r\n
    Keep-Alive: timeout=60\r\n
    Connection: keep-alive\r\n
    \r\n
    [HTTP response 21/23]
    [Time since request: 0.004454000 seconds]
    [Prev request in frame: 1335]
    [Prev response in frame: 1341]
    [Request in frame: 1366]
    [Next request in frame: 1396]
    [Next response in frame: 1402]
    [Request URI: http://localhost:8081/SmartHomeSystem/get-all-data]
  > HTTP chunked response
    File Data: 223 bytes
v eXtensible Markup Language
  > <ResponseData>
```



```
v eXtensible Markup Language
  v <ResponseData>
    v <headCountInHome>
        1
        </headCountInHome>
    v <laserSensor1data>
        true
        </laserSensor1data>
    v <laserSensor2data>
        true
        </laserSensor2data>
    v <laserSensor3data>
        true
        </laserSensor3data>
    v <vibrationCount>
        10
        </vibrationCount>
      </ResponseData>
```

Project architecture

- Maven & Adding Dependencies:

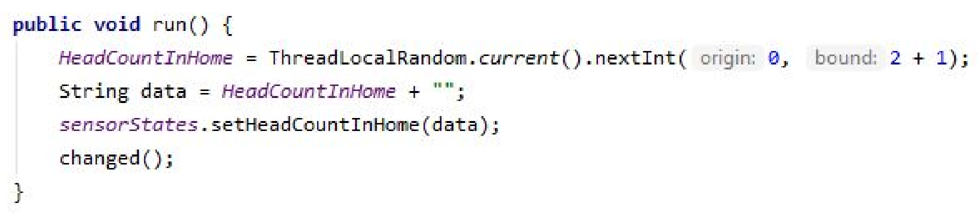


```xml
<groupId>com.IntrusionAlarm</groupId>
<artifactId>intrusion_alarm</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>intrusion_alarm</name>
<description>Demo project for Spring Boot</description>

<properties>
    <java.version>1.8</java.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.eclipse.californium</groupId>
        <artifactId>californium-core</artifactId>
        <version>2.0.0</version>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.dataformat</groupId>
        <artifactId>jackson-dataformat-xml</artifactId>
        <version>2.10.1</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

Here, in the ‘pom.xml’ file 5 dependencies have been added. Thymeleaf Starter is the dependency for maintaining the HTML file exclusively in java environment. Starter Web helps to build web components including RESTful. Californium is the framework for the Constrained Application Protocol for M2M communication. Jackson dependency helps to convert string format from JSON to Java Object and vice versa. Springboot Starter Test dependency is used for testing purposes.

- HeadCountSensor

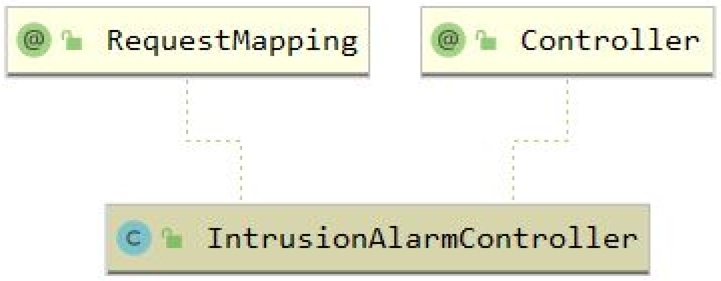
  The front door will always counts the members inside the house. there will be a sensor in the front door. It is named ‘HeadCountSensor’. It will give us the value HeadCountInHome. It updates its value in every 20 seconds. For this project, I am changing this value in a range 0 to 2 which changes randomly.



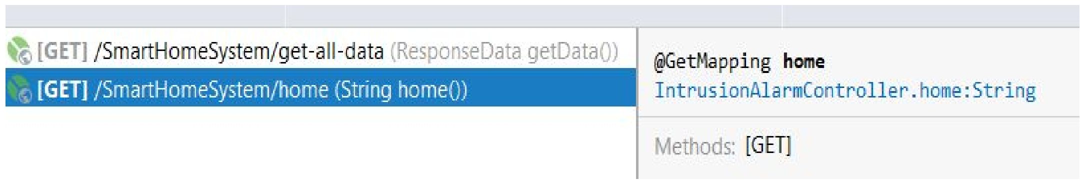
```
public void run() {
    HeadCountInHome = ThreadLocalRandom.current().nextInt( origin: 0,  bound: 2 + 1);
    String data = HeadCountInHome + "";
    sensorStates.setHeadCountInHome(data);
    changed();
}
```

- GET requests

  This project runs in the port number 8081 in localhost. I initiated five servers to hold the sensor information from HeadCountSensor, LaserSensor, LaserSensor2, LaserSensor3, VibratingSensor in other five ports 5685, 5689, 5683, 5684, 5687 respectively.



The application starts in the url ‘http://localhost:8081/SmartHomeSystem/home’

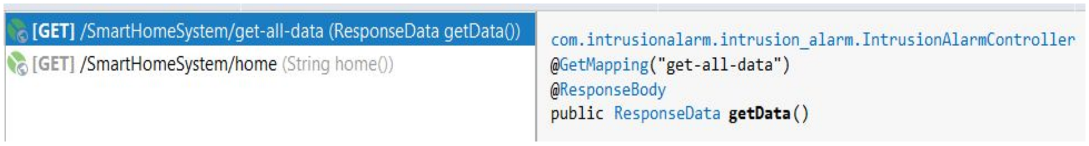


[GET] /SmartHomeSystem/get-all-data (ResponseData getData())
[GET] /SmartHomeSystem/home (String home())

```
@GetMapping home
IntrusionAlarmController.home:String
```

Methods: [GET]

The other url is used to get the response data. This can be accessed in the url ‘http://localhost:8081/SmartHomeSystem/get-all-data’

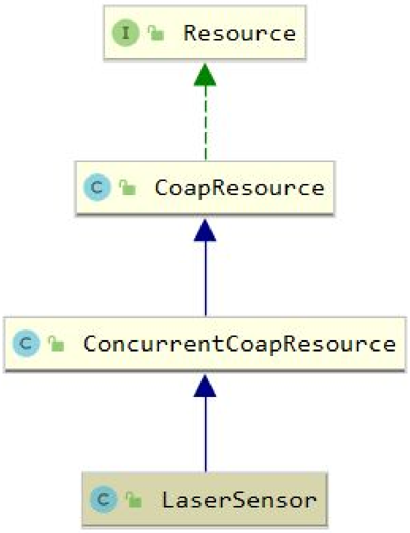


```
[GET] /SmartHomeSystem/get-all-data (ResponseData getData())
[GET] /SmartHomeSystem/home (String home())
```

```
com.intrusionalarm.intrusion_alarm.IntrusionAlarmController
@GetMapping("get-all-data")
@ResponseBody
public ResponseData getData()
```

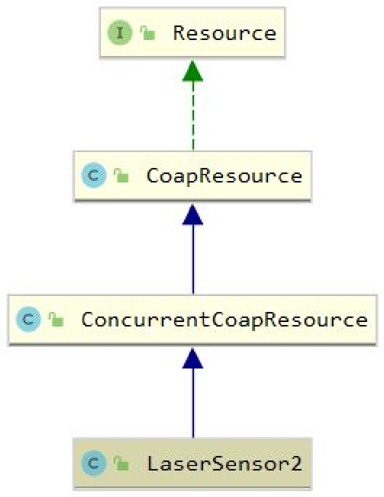
- LaserSensor

  LaserSensor is placed in the first window from inside the house. It will give us the information whether there is any interruption on the laser beam or not. It will update its value every 5 seconds. I first choose a random value in a range 0 to 50 and check whether it is divided by 5 or not and set the sensor value false and true respectively.



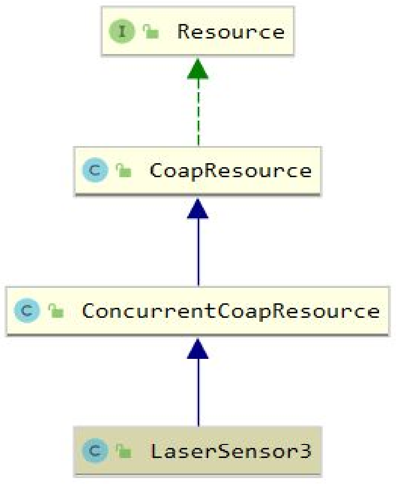
- LaserSensor2

  LaserSensor2 is placed in the second window from inside the house. It will give us the information whether there is any interruption on the laser beam or not. It will update its value in every 10 seconds. I first choose a random value in a range 0 to 50 and check whether it is divided by 6 or not and set the sensor value false and true respectively.



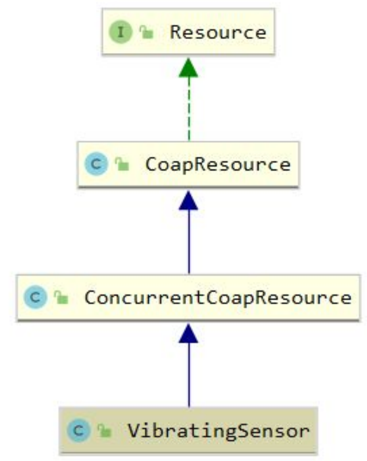
- LaserSensor3

  LaserSensor3 is placed in the third window from inside the house. It will also give us the information whether there is any interruption on the laser beam or not. It will also update its value every 15 seconds. I first choose a random value in a range 0 to 50 and check whether it is divided by 7 or not and set the sensor value false and true respectively.



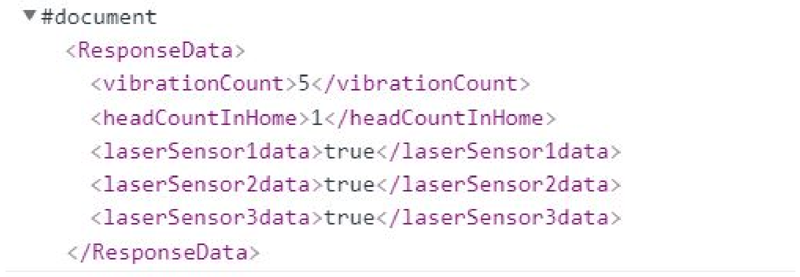
- VibratingSensor

  There is a pressure or vibrating sensor in the front door. If any brugler tries to break the front door or any unusual pressure on the lock, this sensor will send the data to the server.



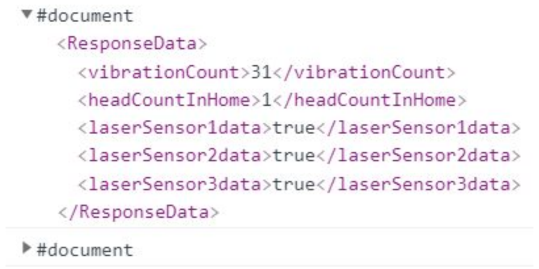
Variation in Response Data

1. People in home, Door lock is safe



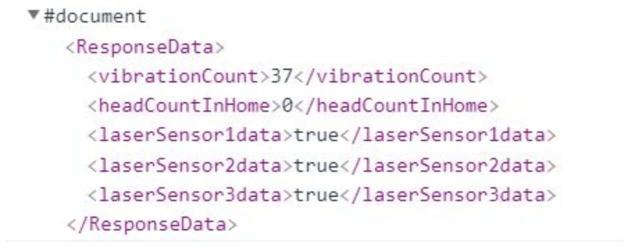
```
▼#document
   <ResponseData>
     <vibrationCount>5</vibrationCount>
     <headCountInHome>1</headCountInHome>
     <laserSensor1data>true</laserSensor1data>
     <laserSensor2data>true</laserSensor2data>
     <laserSensor3data>true</laserSensor3data>
   </ResponseData>
```

2. People at home, someone is pushing the door unnoticed.



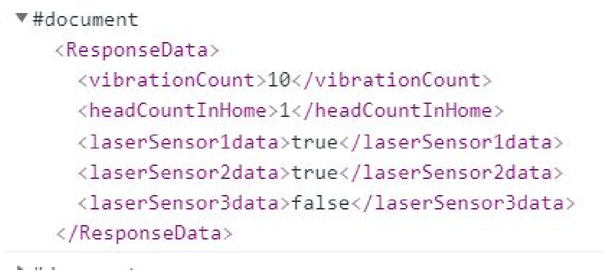
```
▼#document
   <ResponseData>
     <vibrationCount>31</vibrationCount>
     <headCountInHome>1</headCountInHome>
     <laserSensor1data>true</laserSensor1data>
     <laserSensor2data>true</laserSensor2data>
     <laserSensor3data>true</laserSensor3data>
   </ResponseData>
▶#document
```

3. No people at home, someone is trying to break the door lock.



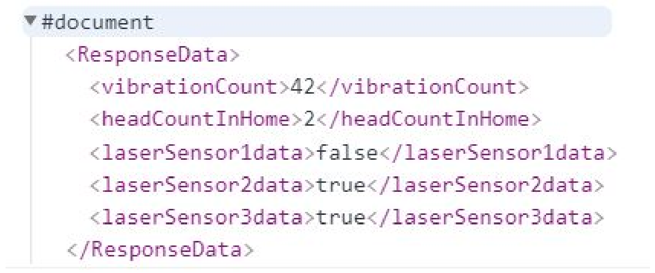
```
▼#document
   <ResponseData>
     <vibrationCount>37</vibrationCount>
     <headCountInHome>0</headCountInHome>
     <laserSensor1data>true</laserSensor1data>
     <laserSensor2data>true</laserSensor2data>
     <laserSensor3data>true</laserSensor3data>
   </ResponseData>
```

4. There is one person at home, one of the windows is open. It does not create an alarm.



```
▼#document
   <ResponseData>
     <vibrationCount>10</vibrationCount>
     <headCountInHome>1</headCountInHome>
     <laserSensor1data>true</laserSensor1data>
     <laserSensor2data>true</laserSensor2data>
     <laserSensor3data>false</laserSensor3data>
   </ResponseData>
```

5. People inside the home, one window is open, So far it is okay, but someone is trying to break the door lock. It will create an alarm



```
▼#document
   <ResponseData>
     <vibrationCount>42</vibrationCount>
     <headCountInHome>2</headCountInHome>
     <laserSensor1data>false</laserSensor1data>
     <laserSensor2data>true</laserSensor2data>
     <laserSensor3data>true</laserSensor3data>
   </ResponseData>
```

User Guide

- Click the Windows Start menu.
- Type `cmd` and click **Command Prompt**
- Change directory to target folder where the .jar file exists. Simply write `cd <target_folder>` and press **Enter.**
- Now write `java -jar <executable_file>.jar` and press **Enter.** For this application the executable file is **intrusion_alarm-0.0.1-SNAPSHOT.jar**

It will start the executable file and run the application in the predefined port of the localhost. For this application I used 8081 port to run this application. So, by just clicking this URL ‘http://localhost:8081/SmartHomeSystem/home’ anyone can check the status of his/her home.

User Interface

There are five sensor fields which show the sensor states. There is also a decision field. Considering all the sensor states by using some logic the decision is made and shown in the bottom right field. The three situations are presented below with screenshots. This homepage is totally responsive so, whether zoom in or out, the sensor and data fields will be placed properly.

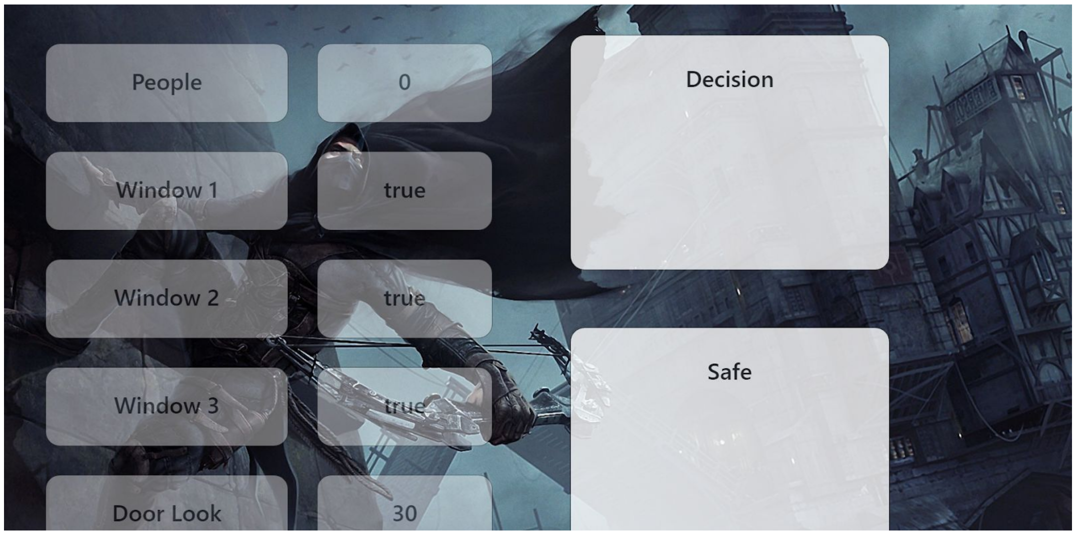
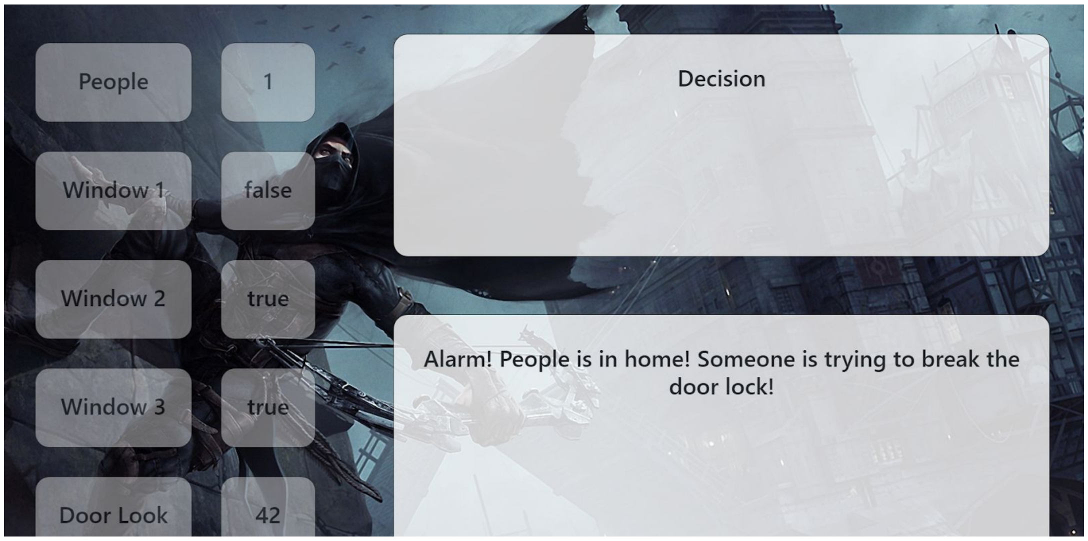
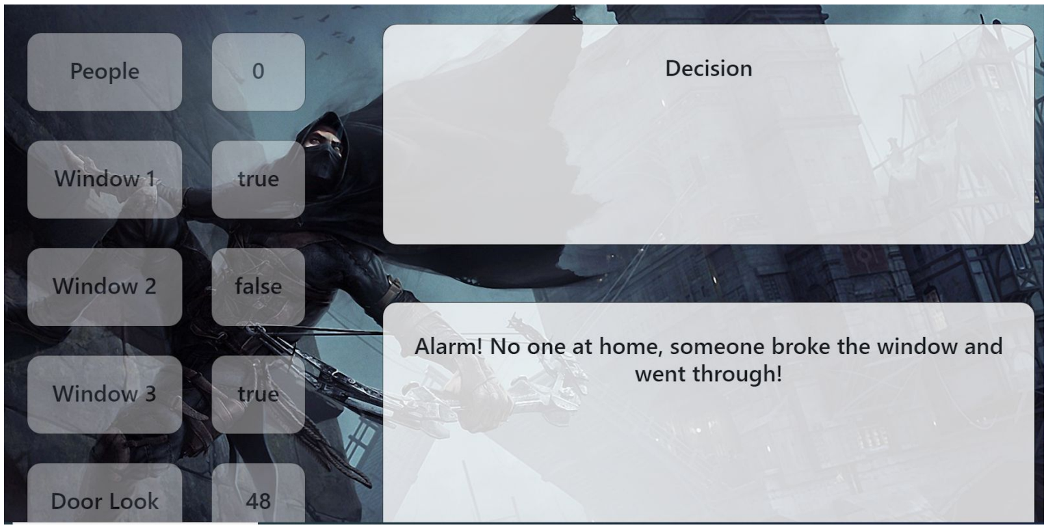


Figure: Safe state



Alarm! People is in home! Someone is trying to break the door lock!

Figure: Intruder is trying to enter from the front door



People	0
Window 1	true
Window 2	false
Window 3	true
Door Look	48

Decision

Alarm! No one at home, someone broke the window and went through!

Figure: Someone broke the second window and entered in the house

Discussion

I have created some virtual sensors to use as a term of IoT end devices. If any real time devices are used, then this application can be a great use to detect any intrusion in home. This is one of the steps to make a Smart Home. For further development, the source code is shared in the github.