# Programming Assignment 1: **Internet of Things and Smart Homes**

Fubao Wu 28277607
Ashraf Ali Shaik 28211687

## Architecture:

### Multithread Based Model

The application is based on Multithreaded Model. Every single action is categorized as separate thread spawn by the Gateway to handle Blocking calls. The total application is implemented on Java Platform using RMI. Once the Application is started, all the Components register at Gateway.

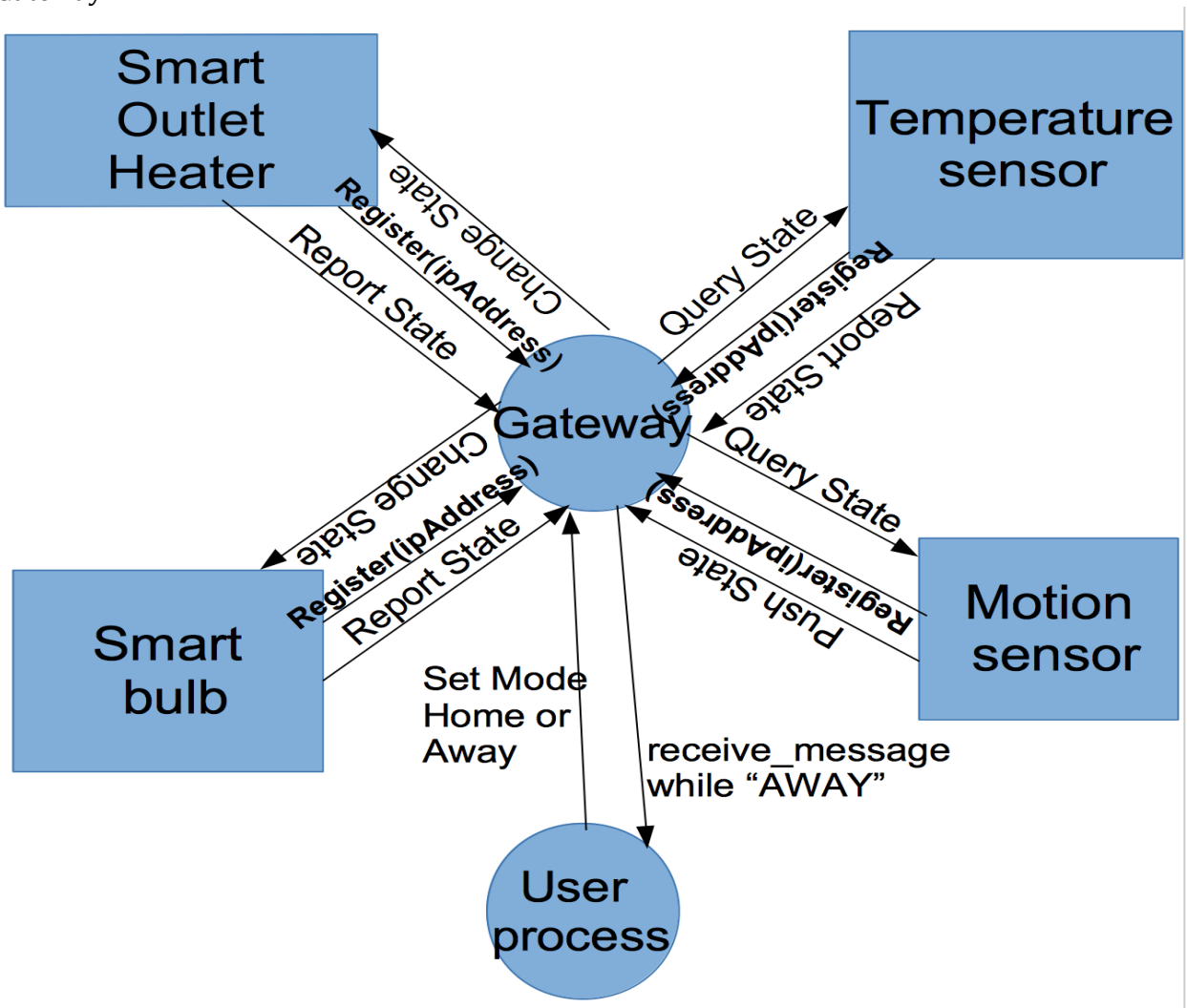



**Figure 1: Architecture Setup**

## Components:

### 1. Gateway

Hub of Application, which forms a bridge between the sensors and devices. Gateway maintains thread safe mechanism by Forking multiple threads to perform various requests to Sensors or devices and Listening on a separate port through a separate thread and accepts remote access requests from Sensors and devices. Gateway performs bookkeeping by maintaining the information about all registered sensors and devices.

2. Temperature Sensor

Pull based Sensor, which can return the current Temperature on query from Gateway. It can also send the current Temperature when required.

3. Motion Sensor

Pull and Push based Sensor, which can periodically push notifications to the Gateway about the presence of motion in a location. Gateway can also query the state of this Sensor.

4. Smart Bulb

Smart Bulb receives the function call from Gateway and either Switches itself ON or OFF. It can also send the state upon queried by Gateway.

5. Smart Outlet Heater

Smart Outlet Heater receives the function call from Gateway and either Switches itself ON or OFF. It can also send the state upon queried by Gateway.

6. User Process

This process allows the user to set at gateway the Mode of being in the Home or Away. This information can be used by the Gateway to decide to operate the Smart Bulb or initiate a text message to the user regarding a possible intruder.

**Accurate timestamp:**

We designed accurate time in different program. Once temperature and motion sensor starts at time 0, it will register to gateway, gateway will think it's time 0. then we will use count time difference from the input “test-input.csv” to read the value and time interval in seconds to separately do the corresponding operations. But we find it's not easy to synchronize in different process, because every progress starts time differently, every machine has different speed and clock drift etc. So we changed our design to logical clock synchronization.

**Logical Clock and Time Stamp:**

We have used logical time stamp methods discuss in class to order the events to receive the input from the **“test-input.csv”** file provided and attain Time Synchronization. Initially the Gateway will be at Timestamp 0 till Motion sensor and Temperature sensor registers at gateway. After that the Gate way sends its Time stamp to both Motion sensor and Temperature sensor. The sensors after reading the Time stamp will increment their timestamp and send the inputs from the file to the Gateway. Then Gateway can increment its time stamp and the process continues.

## Implementation Platform details

→ Java
→ RMI
→ Linux Platform supported scripts.

**Instructions for running the Code:**

We have decentralized our code into various Java Packages where each Package corresponds either to a component (the gateway, a sensor or a device). In order to avoid the complexity and exceptions due to the supporting code being in different packages, **In addition to the source code we have also provided the executable jar files for each component above described. We have Submitted Source code to verify code and Jar files to execute.**

**IP Address Recognition and Allocation:**

We are needed to provide only the IP Address of the Gateway in the Configuration file (**configips.csv**). The Default IP Address in configips.csv is local host. This IP Address is needed for all the other components. Each Component can figure out the value of its IP Address when initiated and will register at the Gateway. Gateway stores the IP Address and can access the other components when required.

**Test-input file:**

The Test-input file "**test-input.csv**" can be used to assign test cases for the application when operated by running the automated scripts instead of user input of test cases.

**Jar files & command line arguments in various cases:**

***For Test cases file as Input(test-case.csv):***

We have implemented the test-case provided and it will generate "**test-output-cached.txt**" in the same Directory of Execution. The following Jar files will take the command line arguments as Path to the Configuration file and Path to Test-input file as arguments. **Please place all the Jar files, configips.csv and test-input.csv in the same Directory.**

```
gnome-terminal -x sh -c "java -jar GatewayServer.jar configips.csv test-input.csv; bash"&
sleep 1
gnome-terminal -x sh -c "java -jar motionSensor.jar configips.csv test-input.csv; bash"&
gnome-terminal -x sh -c "java -jar tempeSensor.jar configips.csv test-input.csv; bash"&
```

***For manual Test cases entered as Input by user:***

The user can manually enter the various events in various terminals (if using a singe machine in the same machine or using different in different machines). In this case, all the following Jar files will take the command line argument as only Path to the Configuration file (**configips.csv**). This can run in different Machines by users manually.

```
gnome-terminal -x sh -c "java -jar GatewayServer.jar configips.csv; bash"&
sleep 1
gnome-terminal -x sh -c "java -jar motionSensor.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar tempeSensor.jar configips.csv; bash"&
```

```
gnome-terminal -x sh -c "java -jar bulbSmart.jar configips.csv ; bash"&
gnome-terminal -x sh -c "java -jar HeaterSmart.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar UserOperation.jar configips.csv; bash"&
```

Execution of these commands will open 6 Terminals in which user can enter the test cases as illustrated using Sequence Diagrams in this document.

**Executable Script files:**

***run-test-case.sh:*** This Script file corresponds to the automation of the "***For Test cases file as Input"*** case discussed above. This script is executable and has all permissions. The script will execute in different Terminals and an output file **"test-output-cached.txt"** is created by the Gateway with the tuples containing time stamp values, the Queried state information from the Temperature sensor and Motion Sensor. **No Need of User Input. Please wait till all the execution completes in 3 terminals and the output from the programs stop (Doesn't move). See Appendix: 1**

***run-all.sh***: This Script file corresponds to the automation of the "***For manual Test cases entered as Input by user"*** case discussed above. This script is executable and has all permissions. User Just need to run this script and 6 Terminals Pop up and the user need to enter various values depending on the statements in the terminal and the Application can be tested. **User Input Needed. The first or second line output in terminal will convey the operation performed by that process. See Appendix: 2**

**Manual Entry Test Cases Sequence Diagrams:**

**Task 1:** Smart Outlet controlled by gateway using temperature values from sensor

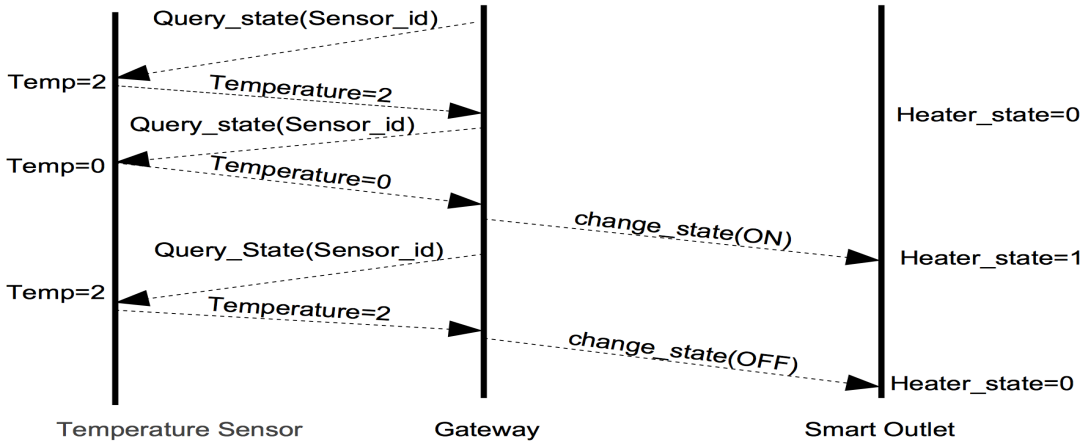



Figure 2: Task 1 Test case Sequence Diagram

**Task 2:** Smart Bulb controlled by gateway using Motion values from sensor and Mode set by User Process. **(I) Mode: Home (Gate way Querying)**

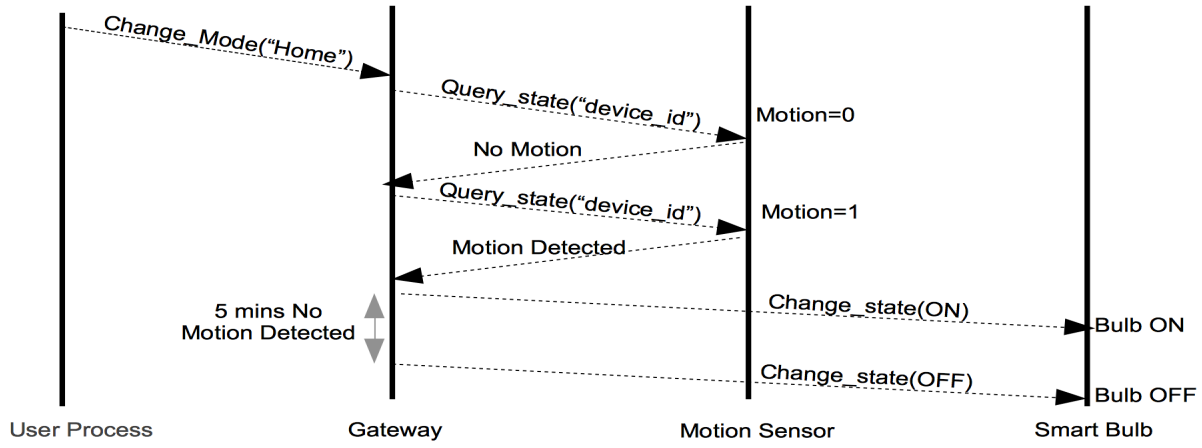



Figure 3: Task 2 Home Mode Sequence Diagram

## (II)Mode: Home (Motion Sensor Pushing Notification)

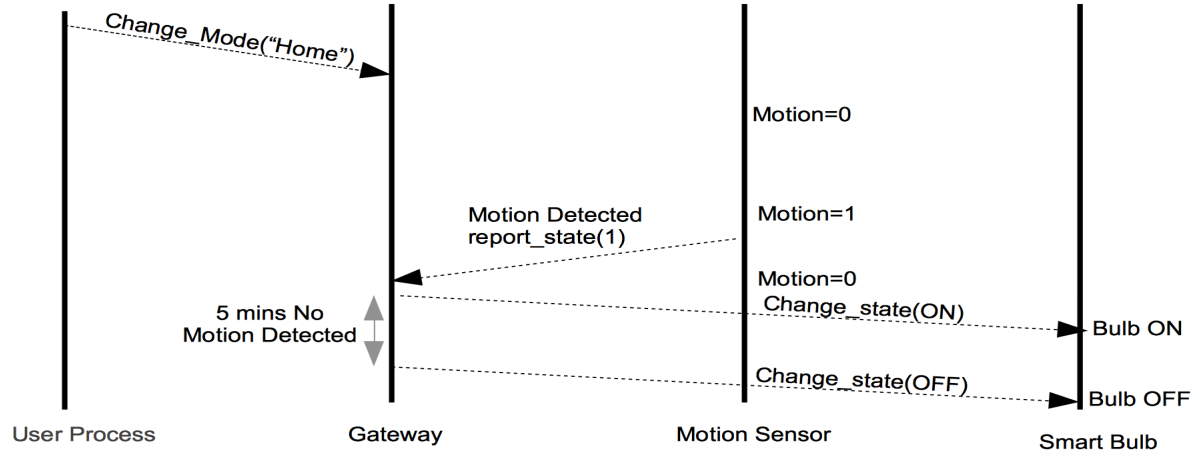



Figure 4: Task 2 Mode Home Motion Sensor pushing notifications

## (III) Mode: Away (Gate way querying)

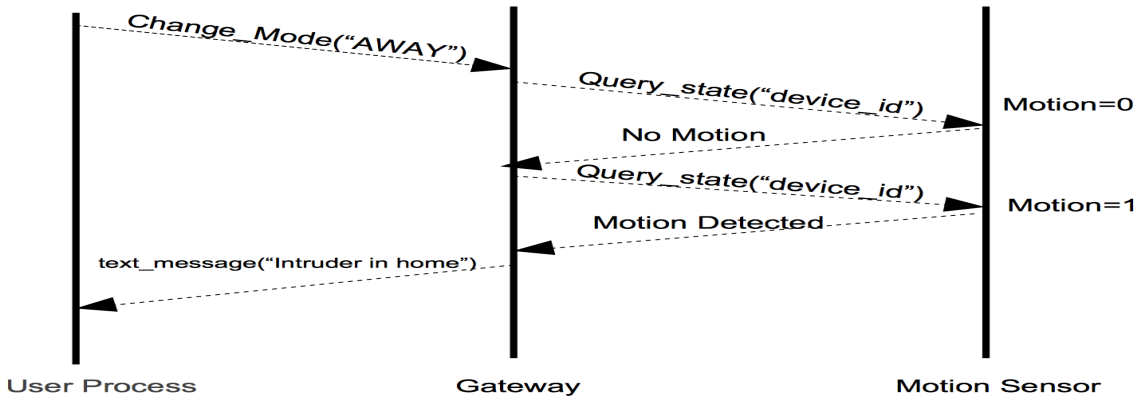



Figure 5:Task 2 Mode Away

## (IV) Mode: Away (Motion Sensor Pushing Notification)

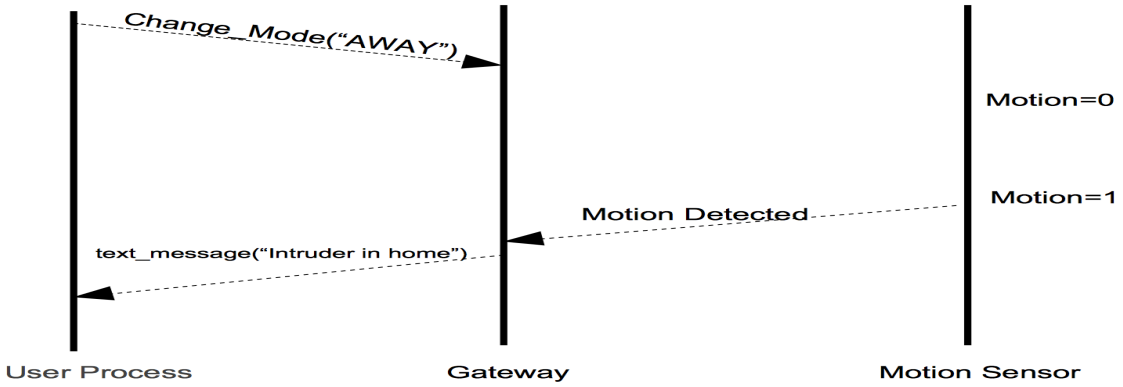



Figure 6 Task 2 Mode Away sensor pushing notifications

## Test case Sequence diagram showing Application is Thread Safe:

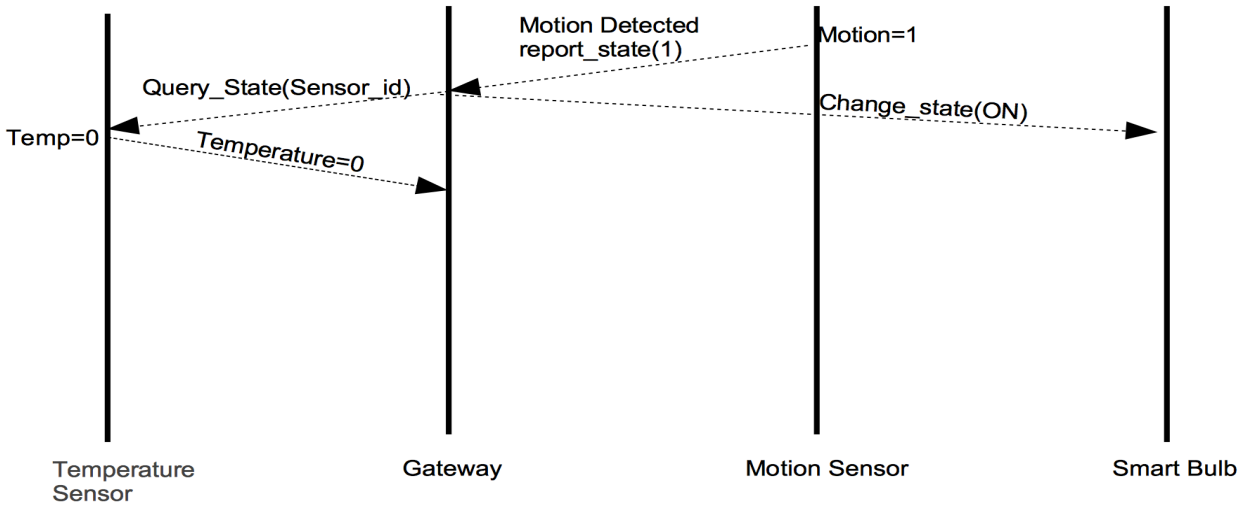



Figure 7 Sequence Test case diagram for concurrent thread nature of gateway

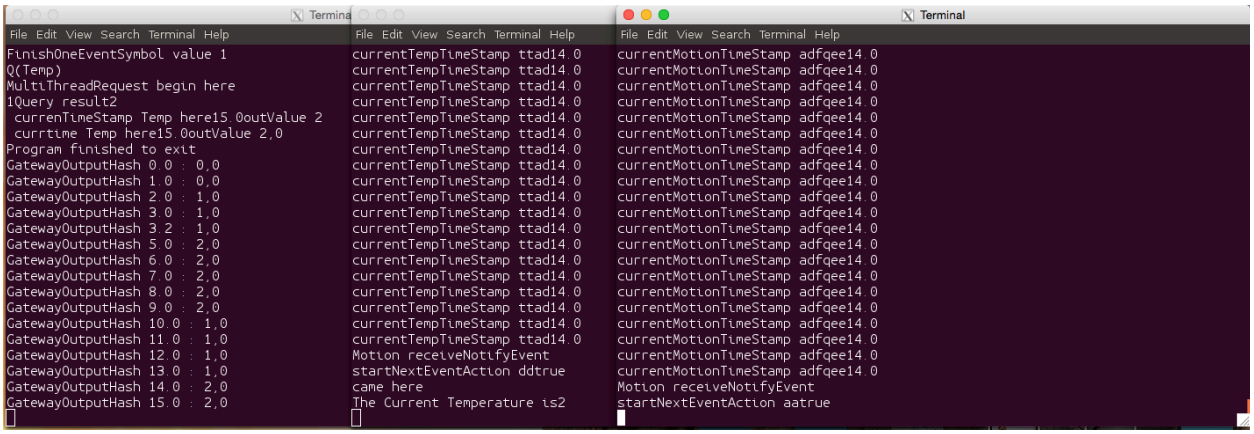
**Performance Analysis:**
We have performed experiments on the performance by measuring the delay from between Gateway and the Sensors.
→Push Performance delay test which Gateway Received push temperature from Motion sensor.
→Pull Performance delay test which Gateway pull temperature from temperature sensor

| Interval | 500ms(average) | 1000ms | 2000ms | 5000ms | 10000ms |
|---|---|---|---|---|---|
| push(report) delay(milliseconds) | 3.454 | 2.898 | 3.572 | 3.452 | 3.12.3 |
| Pull dealy (query(milliseconds) | 4.224 | 3.554 | 4.232 | 4.522 | 4.134 |

From basic statistics of this table, the delay of push and pull are very small where as pull delay is a more than push.

**Conclusion:**
→ Application can be used to run in Automated test case Mode test cases provided as a file.
→ Application can also run in Manual mode where user has to interact from the terminal.
→ Generated an output file containing the tuples of time stamps and sensor values for automated mode.
→ we have extensively tested all the Test cases as mentioned above to ensure the application is executing as expected.

## Appendix 1: ScreenShot for test cases implementation from input file. Showing when to stop and check for output file.




## Appendix 2: ScreenShot for Manual mode Entry test cases User Need to enter the various events.