COMPUTER ENGINEERING

4DN4

*ADVANCED INTERNET COMMUNICATIONS*

# Lab – 1:

# A Simple "Internet-of-Everything" Smart-Home Server

**Submitted To:**

*Professor Ted Szymanski*

**Submitted By:**

Name: Syed Anjoom Iqbal

Student ID: 1063786

MacID: iqbalsa

Email: iqbalsa@mcmaster.ca

**Submission Date:**

February 11, 2014

# COMP ENG 4DN4 2014 Lab1 Report
## A Simple "Internet-of-Everything" Smart-Home Server

## Table of Contents

## Objective

Internet-of-Everything is a very new concept in the internet communication industry. Every day, the number of devices, connected to internet, is growing. Modern engineers are focusing on creating a smart version of every aspect of life. Smart-home is a good example of this. The idea is to connect almost every devices of a home to the home network so that the users can easily know the status of each device and change them as well. For example, during cold weather if a user is going away from home for few days then s/he can turn off the heater to save energy. But when s/he will come back after few days then it will take a while for the home to heat up from totally cold state to a comfortable level. If s/he could securely access the heater remotely over internet using smartphone/tablet/computer then it is possible to turn the heater on few hours before the estimated time of arrival.

In this lab, the objective is to create a simplified model of smart home server and smart home client and establish communication between them over internet.

## Introduction

Version 2 of the lab document has been implemented for this lab and Java programming language has been used. The `SmartHomeServer` package contains 2 classes (`SmartHomeServer.java` and `Device.java`). In `Device.java` the structure of a device has been declared with private fields (memory is allocated dynamically to the 3 string fields), public get-set methods and constructors with a scope of within the `SmartHomeServer` package. `SmartHomeServer.java` has been designed in a modular way. It creates a dynamic `ArrayList` of the `Device` objects in the server. It initializes the `ArrayList` with two dummy devices 'Thermostat-Main' and 'Thermostat-Living-Room'. Inside `main(String[] args)` method several private methods has been called to perform the required operation. These private methods have been named in such a way to briefly summarize the purpose.

The `SmartHomeClient` package contains only 1 class (`SmartHomeClient.java`). This class has also been designed in modular way like the `SmartHomeServer.java` class.

In brief, here is a list of what happens in `SmartHomeServer` and `SmartHomeClient` from start.

| **SmartHomeServer** | **SmartHomeServer** |
|---|---|
| 1. Check for correct number of arguments<br>2. Create SererSocket and initialize database<br>3. Wait on blocking accept to create socket<br>4. After creating a socket, start an infinite loop and keep parsing the sent string from client to get command<br>5. Based on the command, call the appropriate method to handle that action and send relevant information back to client<br>6. If client closes the connection or the connection gets terminated somehow then the server catches that exception and closes that socket and waits on blocking accept for another client to connect | 1. Check for correct number of arguments<br>2. Wait for getting a command form user<br>3. Handle the user input based on the command<br>4. First the client must connect to the server before any other command can be sent<br>5. Once a connection has been established then it allows to send other commands and handles the socket transaction with the server<br>6. If client wants to QUIT then the program closes.<br>7. Also if the server send back that maximum number of invalid try has been made then due to security reason the connection get terminated from the server |

## Server Flow Chart

Please see the `Server Flow-Chart.pdf` file in this directory.

## Client Flow Chart

Please see the `Client Flow-Chart.pdf` file in this directory.

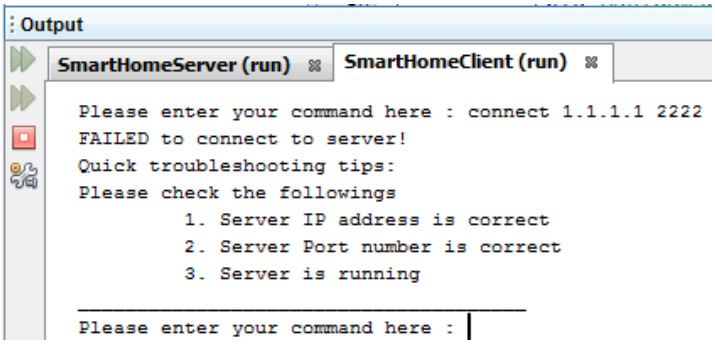
## Java Source Code

Please see `SmartHomeClient` and `SmartHomeServer` folders in this directory.

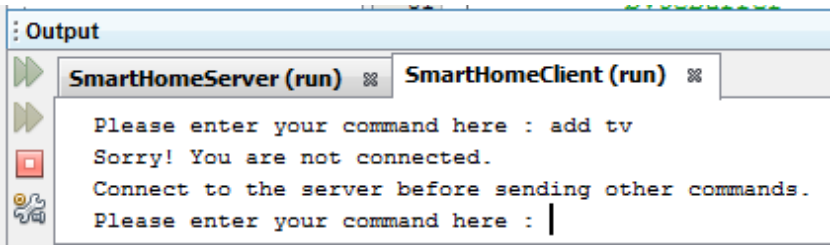
## Experimental Results

1. When both server and client are started:

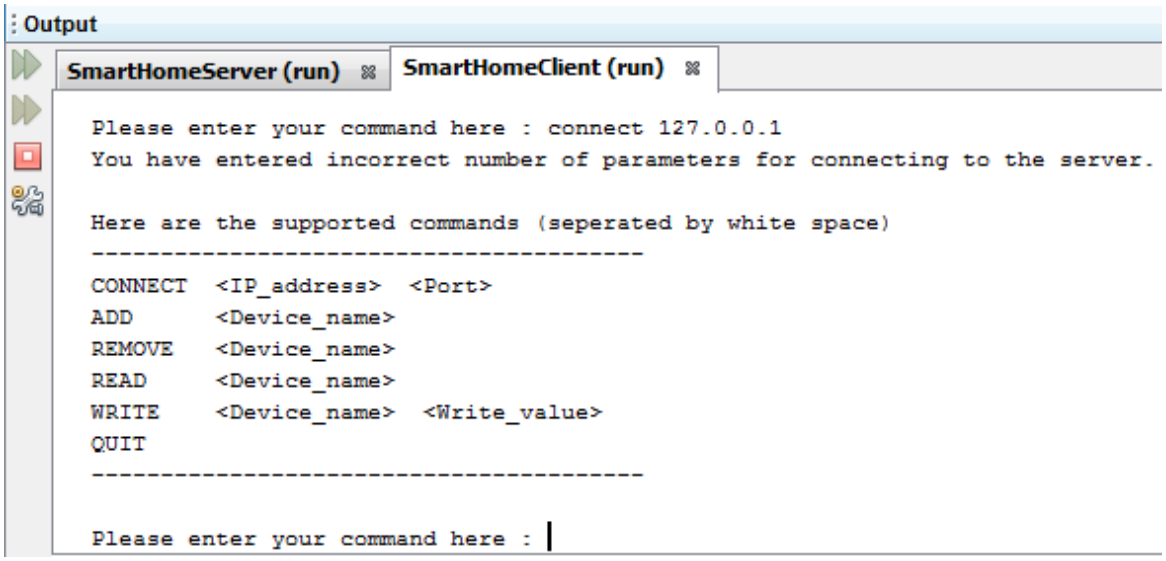

2. Now if the client tries to connect with invalid IP address or Port, server doesn't see anything but client shows following



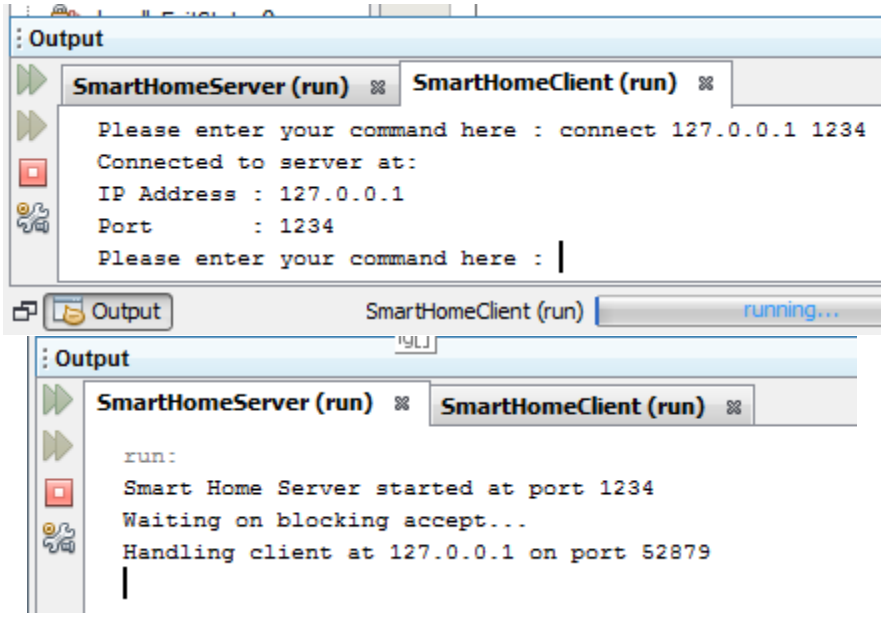
3. Now if the client tries to send a valid command before connecting to the server then client sees:



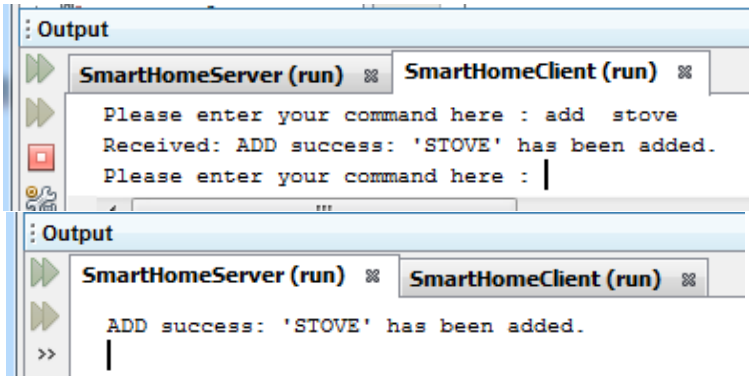

4. When the client tries to connect to the server with incorrect number of parameters (There is no port number in the example) the client cannot connect and supported commands are displayed.



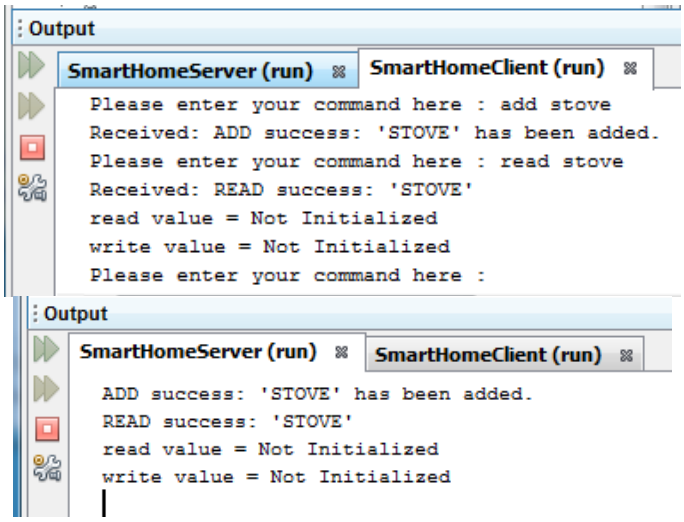

5. When the client successfully connects to the server with correct IP address and port number.



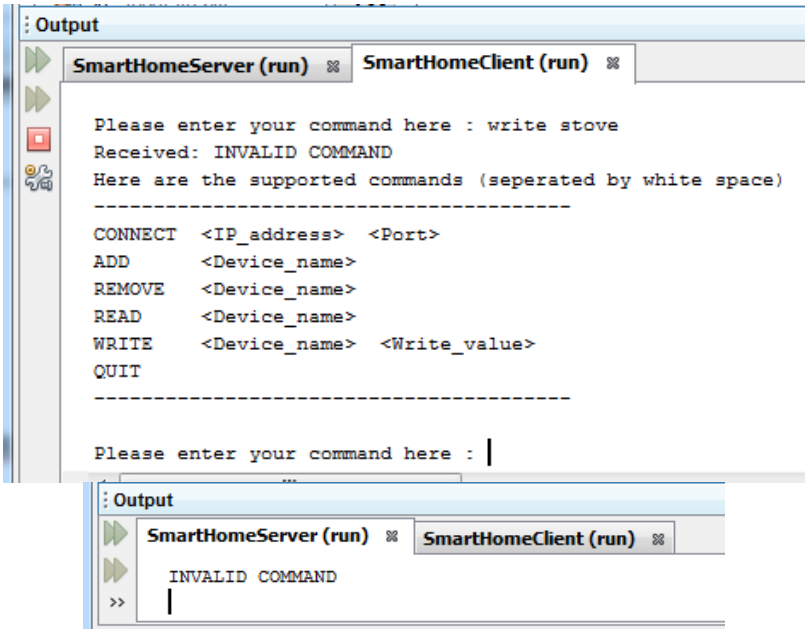
6. Client tries to add a device in the server



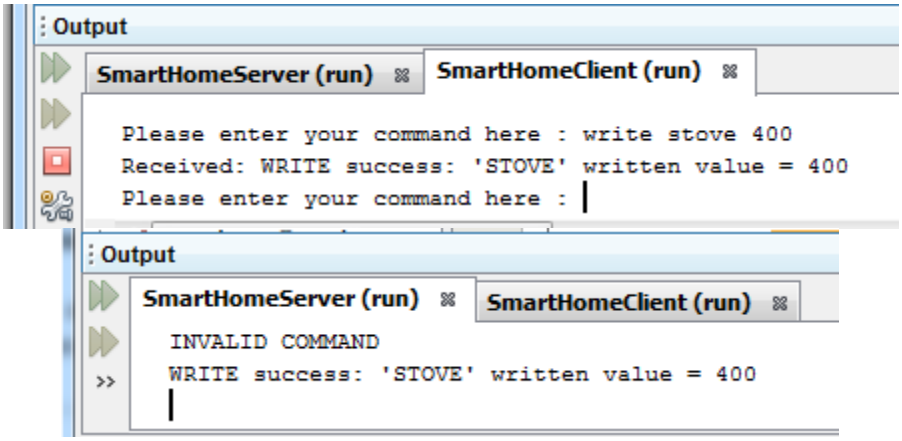

7. Now the client tries to read the current value of the newly added device



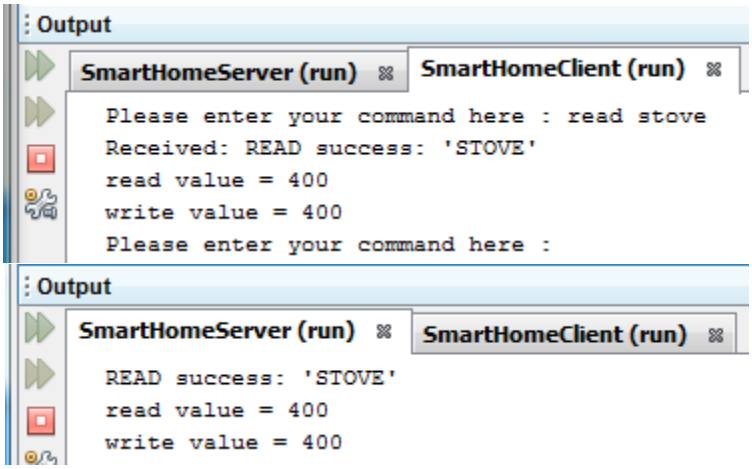

8. When client writes a value to the newly added device but does not provide the write value



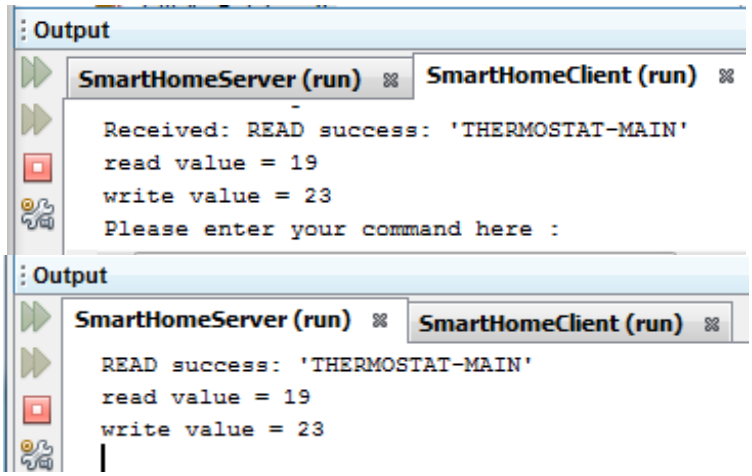
9. When client writes a value to the newly added device and provides the write value.



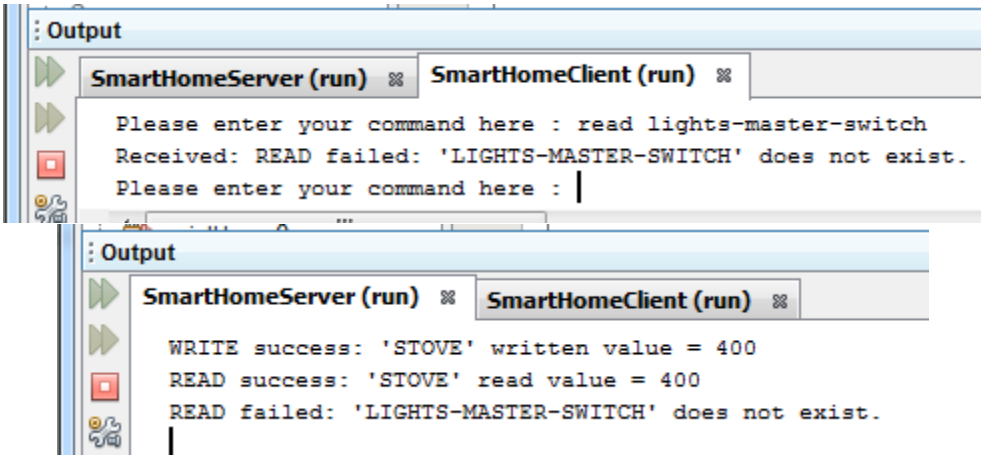

10. **Assumption**: When a value is written to a device, the device reaches to that state instantly. So reading the device will show that both READ value and WRITE value are same as the value that just got written.



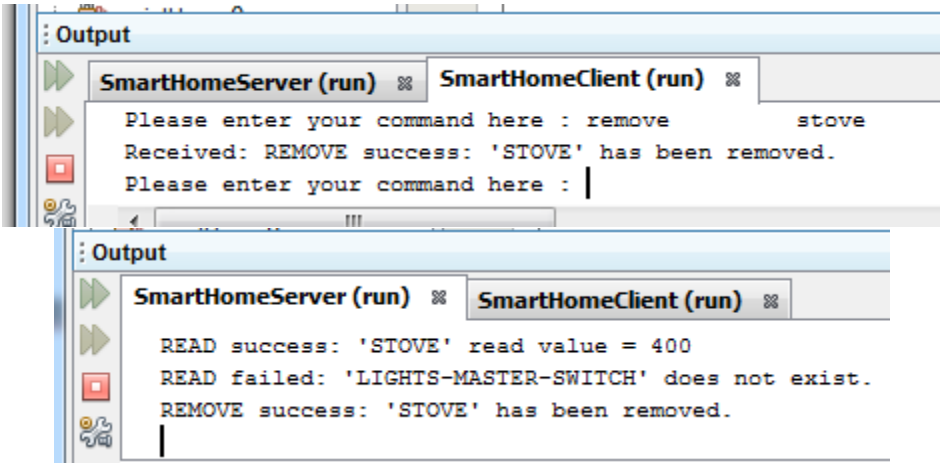

11. When the client tries to read a device that exist in the server by initializing at the beginning



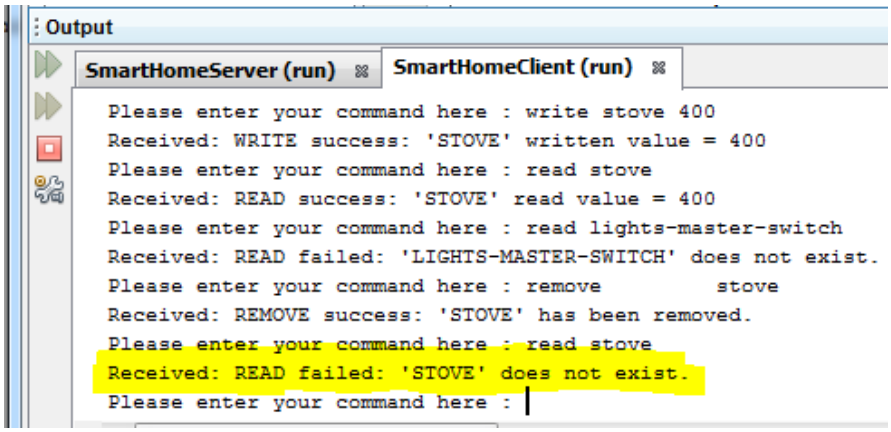
12. When the client tries to read a device that has not been entered to the server

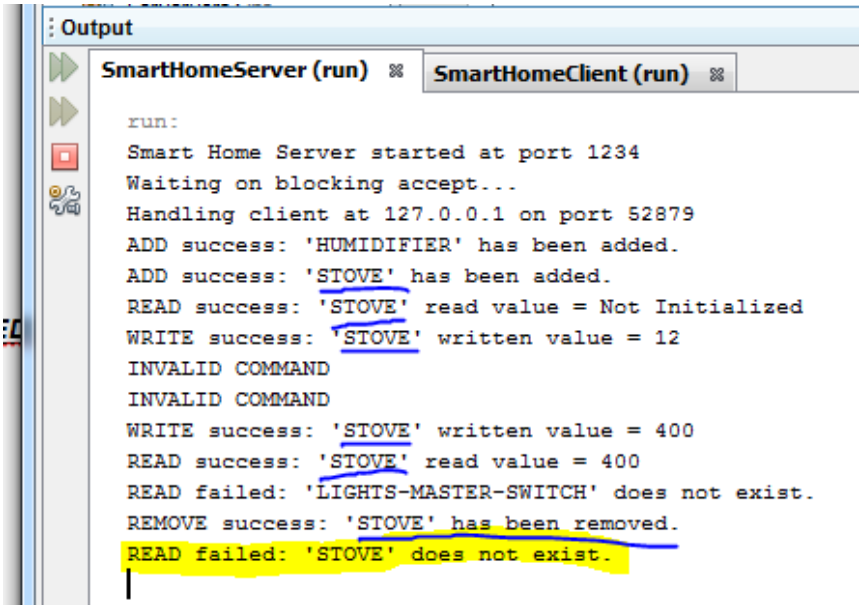



13. When the client removes an existing device

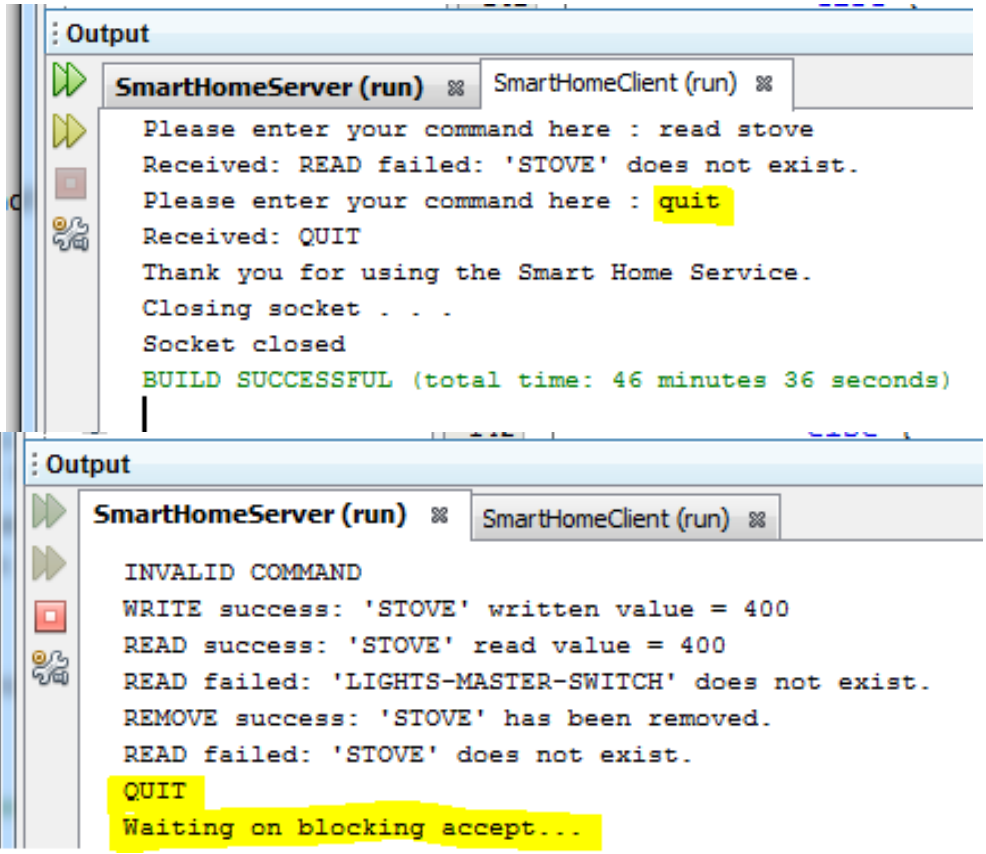



14. After removing a device when the client tries to read that device

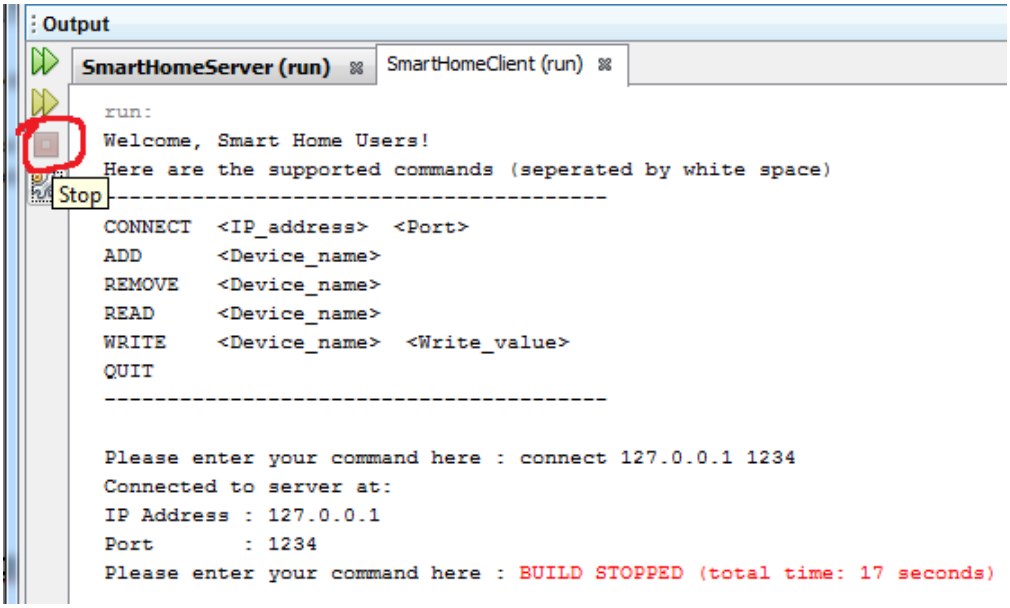


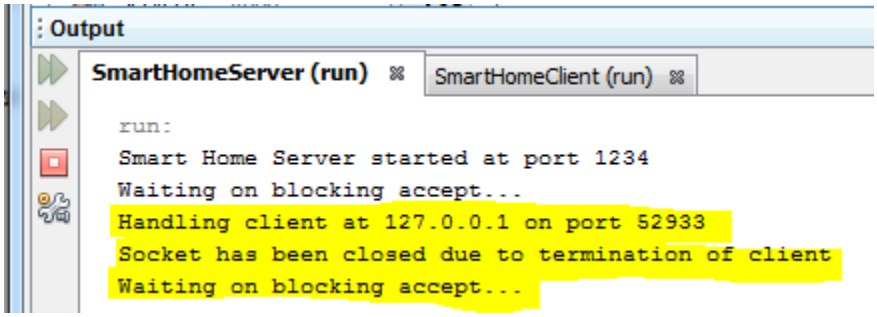



15. When client wants to QUIT, it closes the socket and the client program. But the server keeps running and waits for another



16. If a client successfully connects to a server but the client program crashes which can be emulated by stopping the client program by click the stop button, then the sever does not crash. It can catch that exception and it just closes the socket and waits for a blocking accept to create a new socket.




## Issue/Problem Encountered

This is a very simply model of a smart home where security features has not been implemented. It is understood that smart home system needs to have a VERY RELIABLE and STRONG security system with different layers of firewalls. People will not feel safe if anyone can hack into their home system and change things, which can have even fatal consequence.

In this implementation assumption is only one client accesses the server at a time. Main reason for this is to remove ambiguity in the system at a simple level. Also the course material for handling multiple clients simultaneously has not been covered yet.

Addressing the above two issues will require a more complicated and sophisticated implementation of the smart home, which has been assumed to be out of scope of the course at least at this stage.

## Name of the TA to whom the lab was demo-ed

Maryam Razaee

## Conclusion

Smart home is a very interesting concept. If the smart home network is kept local then people can change setting of their home when they are connected to the home network. For example, in this case people will be able to close their garage door with their smartphone if they forget to close it when came up home. This can be really handy especially in winter when people have to dress up to go outside even for closing the garage door.

This program has been designed for a simple implementation of smart home server and client. Efforts have been made to catch mane possible exceptional and invalid cases so that the functionality doesn't get affected. Also the displayed messages has been selected to make the user experience smoother.

---

THE END