Design Documentation-Devices

HomeDork - Interactive House

Project Members

Reference	Name	Email
A	Samuel Mcmurray	Samuel_joseph.mcmurray0004@stud.hkr.se
B	Mustafa Ismail	mustafa.ismail0007@stud.hkr.se
C	Ibrahim Ahmed Ali	ibrahim.ahmed_ali0003@stud.hkr.se
D	Osayomore Edugie	Osayomore.edugie0004@stud.hkr.se

Revision History

Date	Version	Description	Author
04/10/2021	1.0	Initial Design	A, B, C, D

Design item List

Requirement Name	Priority
D1. The Overall Design	Essential
D2. Use Case Diagram	Essential
D3. Class Diagram – Overall	Essential
D4. Class Diagram – Abstract Devices and the classes that inherit.	Essential
D5. Class Diagram – The Alarm class and the devices that make up its composition	Essential
D6. Class Diagram – The Temperature Controller	Desirable

D7. Class Diagram – Response Class	Essential
D8. Class Diagram – The Main Class	Essential
D9. Sequence Diagram – Read Temperature	Essential
D10. Sequence Diagram – Turn Lamp On	Essential
D11. Sequence Diagram – Open Curtain	Essential
D12. Sequence Diagram – Temperature Control	Desirable
D13. State Diagram – Temperature Control	Desirable
D14. State Diagram – Curtain	Essential
D15. State Diagram – Light	Essential
…	Essential/Desirable/Optional

Design Item Descriptions

D1

Overall design (Figure 1) black box here we have an example of turning on a light the user interacts with a UI from one of their devices. The request is sent to the API server, the server then makes a request to the Arduino. The Arduino handles the request in this case turn on a light and sends a response back to the API server as to the success or failure of the request. The API updates the database with the current state of the device receives a response and sends a response back to the user.

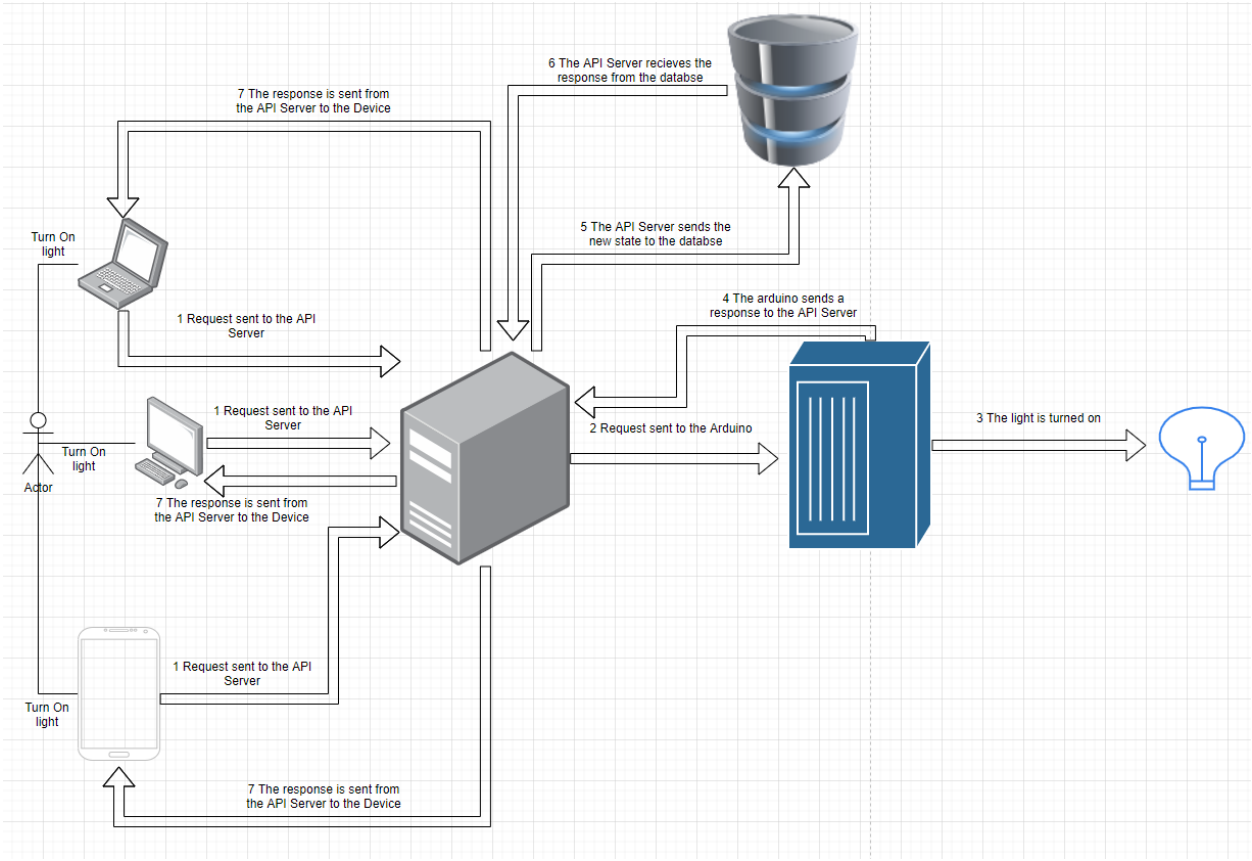

Figure 1 - The HomeDork Design

D2

The use case diagram (Figure 2) depicts a user who access the UI application, which is connected to the server, the server is then connected to the Arduino. The user through the application has access to lamps which can be turned on or off, a fan which can change speed, has a timer, and can be turned on or off, and curtains which can be moved up or down based on the user preference.

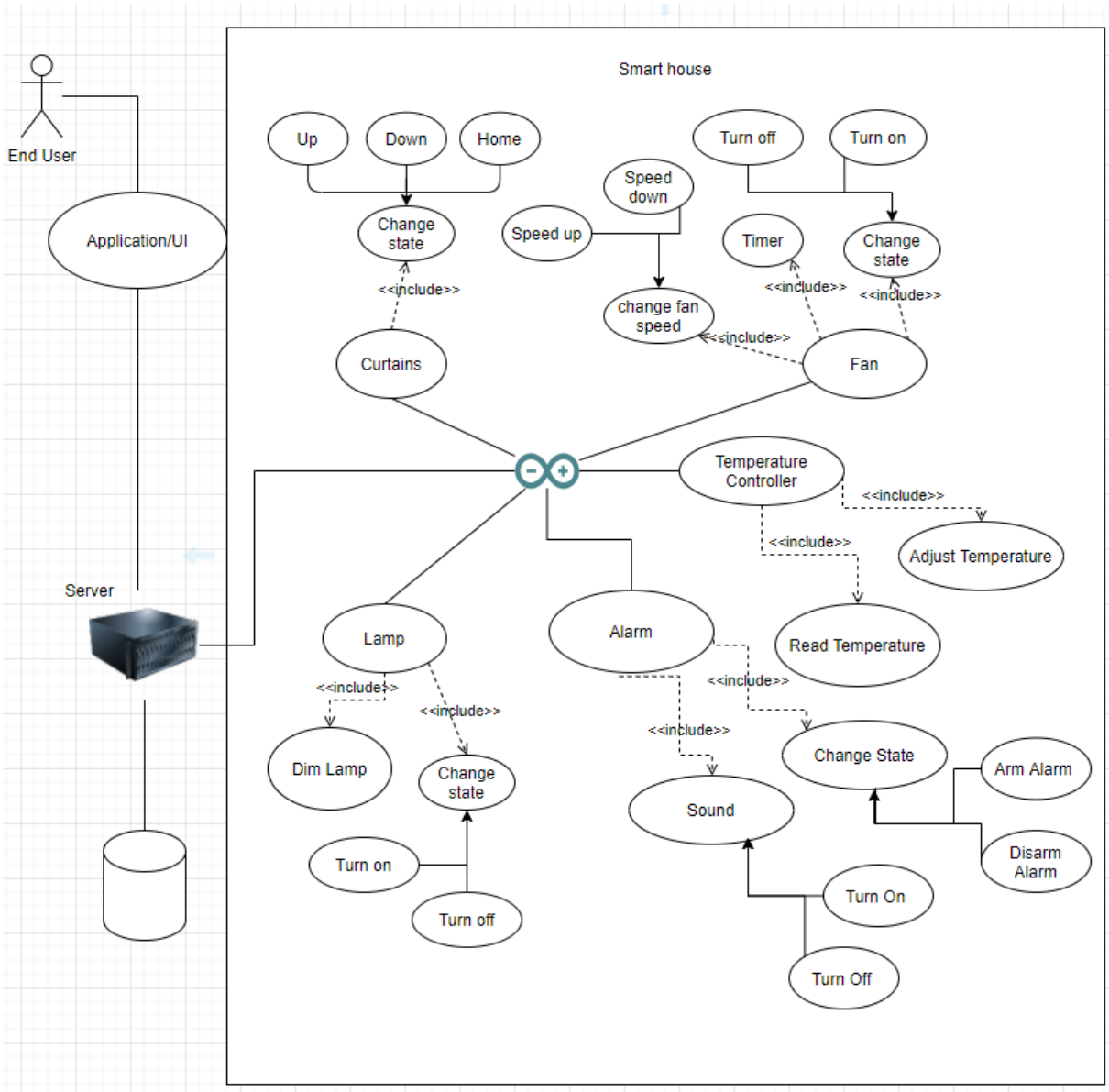

Figure 2 - Use Case Diagram

D3

This is the class diagram for the devices we have the overall structure here (Figure 2).

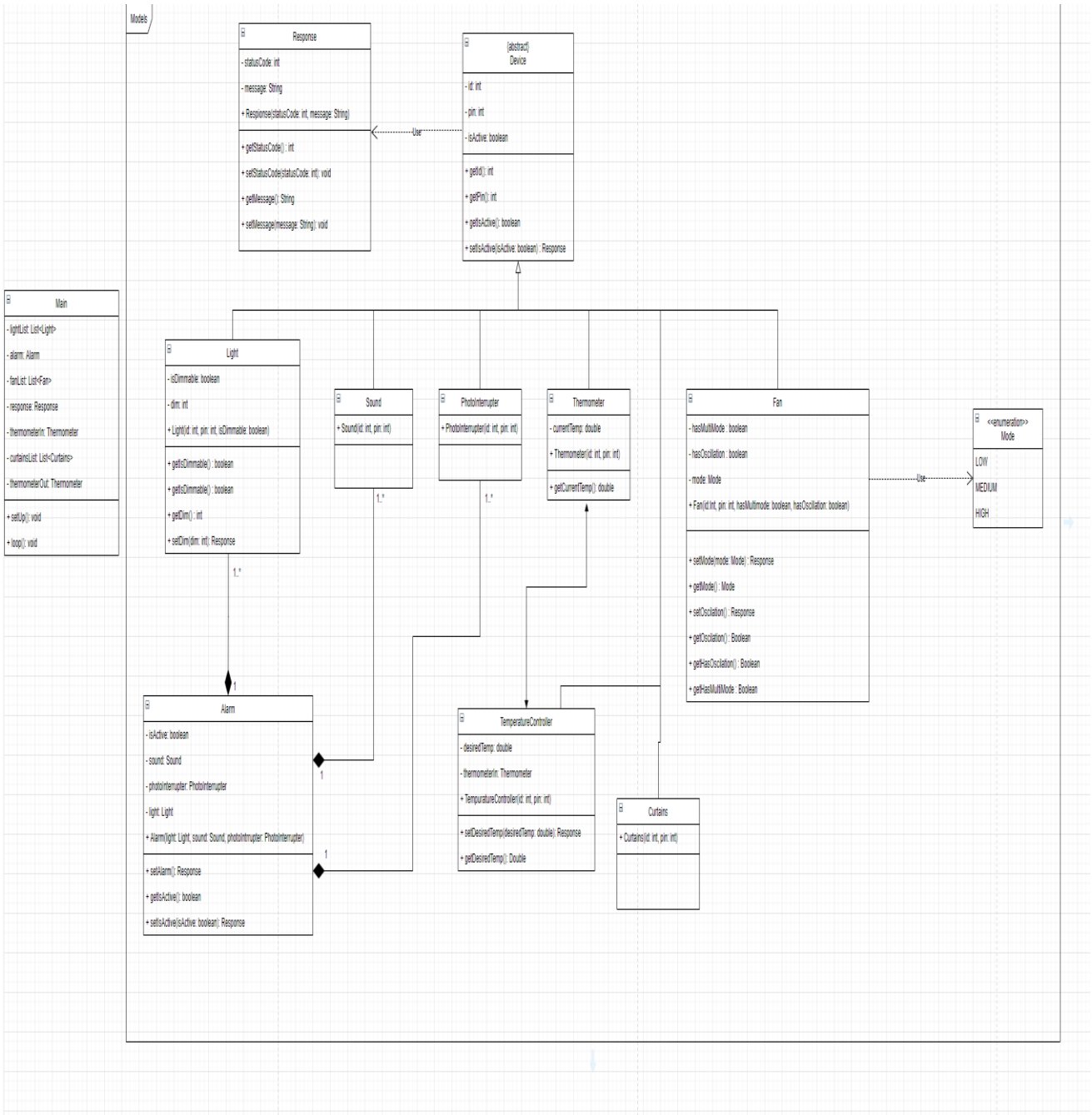

Figure 3 - Class Diagram

D4

The first class we created was an abstract device class (Figure 4) which has a variable for the id of the device as an int, the pin of the device as an int, and a Boolean that is used to determine the state of the device. The functions used in this class are getId which returns an int, getPin which also returns an int, getIsActive which returns a Boolean, and setIsActive which takes a Boolean and returns a responsefrom our Response class.

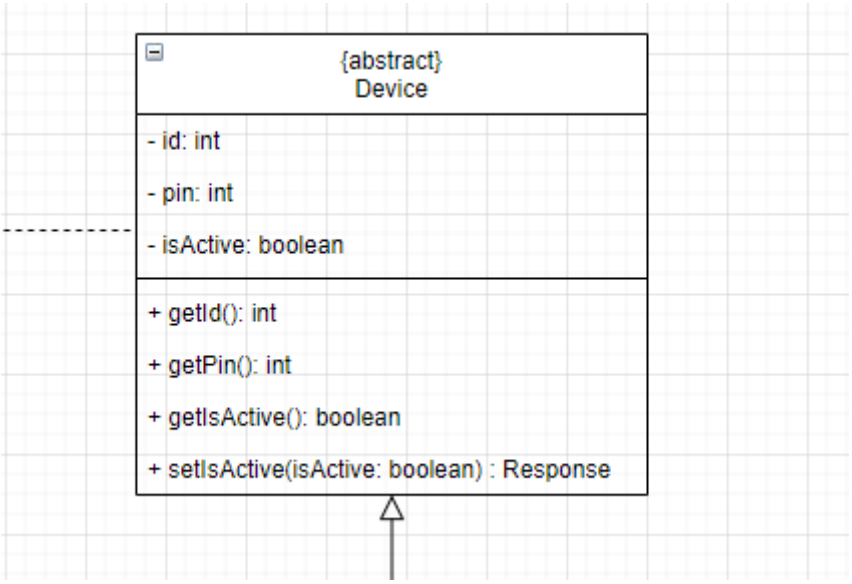

Figure 4 - Abstract Device Class

The classes that inherit from the device (Figure 5, Figure 6) class are light which is for the lamps, sound which is apart of the composition in the alarm class, photo interrupter which also is apart of the composition in the alarm class, thermometer class, curtains class, and fan class.

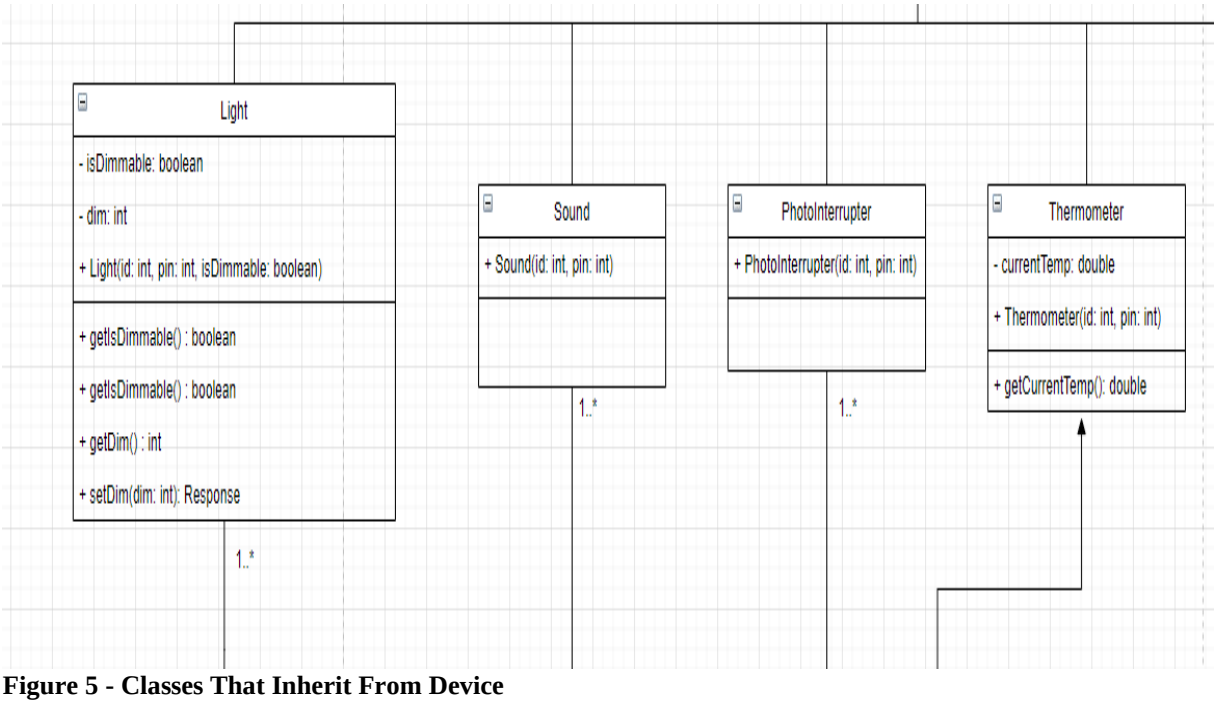

Figure 5 - Classes That Inherit From Device

The light class has a isDimmable boolean to determine the type of the light it is, dim which will set the dim desired dimness, the constructor which takes an id, pin and isDimmable. The functions for the light class are getIsDimmable which returns a boolean, getDim which returns an int, and setDim which takes an int and returns a response, sound class and photo interrupter only have constructors because they primarily just have the inherited methods. Thermometer class has a variable currentTemp which is a double a constructor which takes id, and pin. The function it uses is the getCurrentTemp which returns a double.

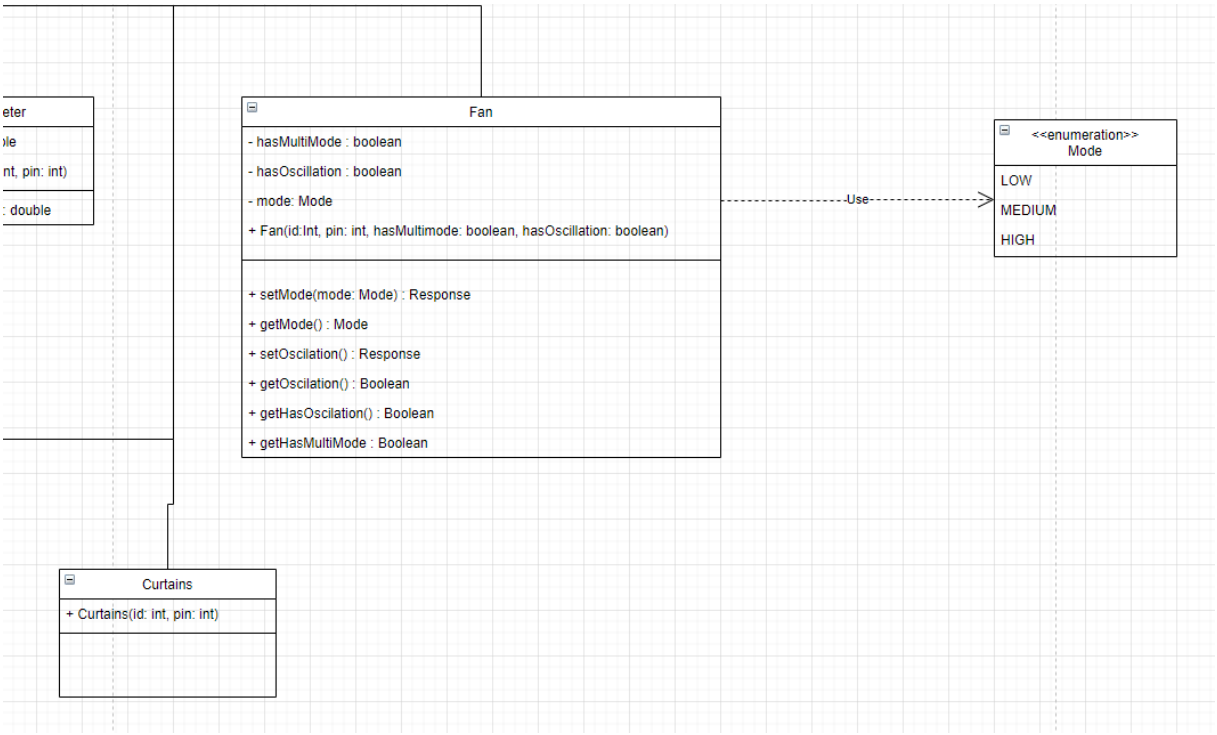

Figure 6 - More Classes That Inherit From Device

The other device classes are the Curtains class which only contains a constructor and the fan class. The fan class has 3 variables hasMultiMode a Boolean, hasOscillation a Boolean, mode which is an enum of LOW, MEDIUM and HIGH. The constructor takes id, pin, hasMultimode and hasOscillation. The functions associated with the fan class are setMode which takes Mode as a parameter and gets a response as the return type, getMode which returns mode, setOscillation which returns a response getOscillation which returns a Boolean, getHasOscillation which returns a Boolean, and getHasMultimode which also returns a Boolean.

D5

The alarm class is (Figure 7) a composition of the light device, the sound device, and the photo interrupter device so the instances of objects are created and used by this class. The alarm class also has isActive Boolean and takes the light sound and photo interrupter as parameters in the constructor. The functions associated with the class are setAlarm which returns a response, a getIsActive which returns a Boolean, and setIsActive with a isActive Boolean as a parameter and returns a response.

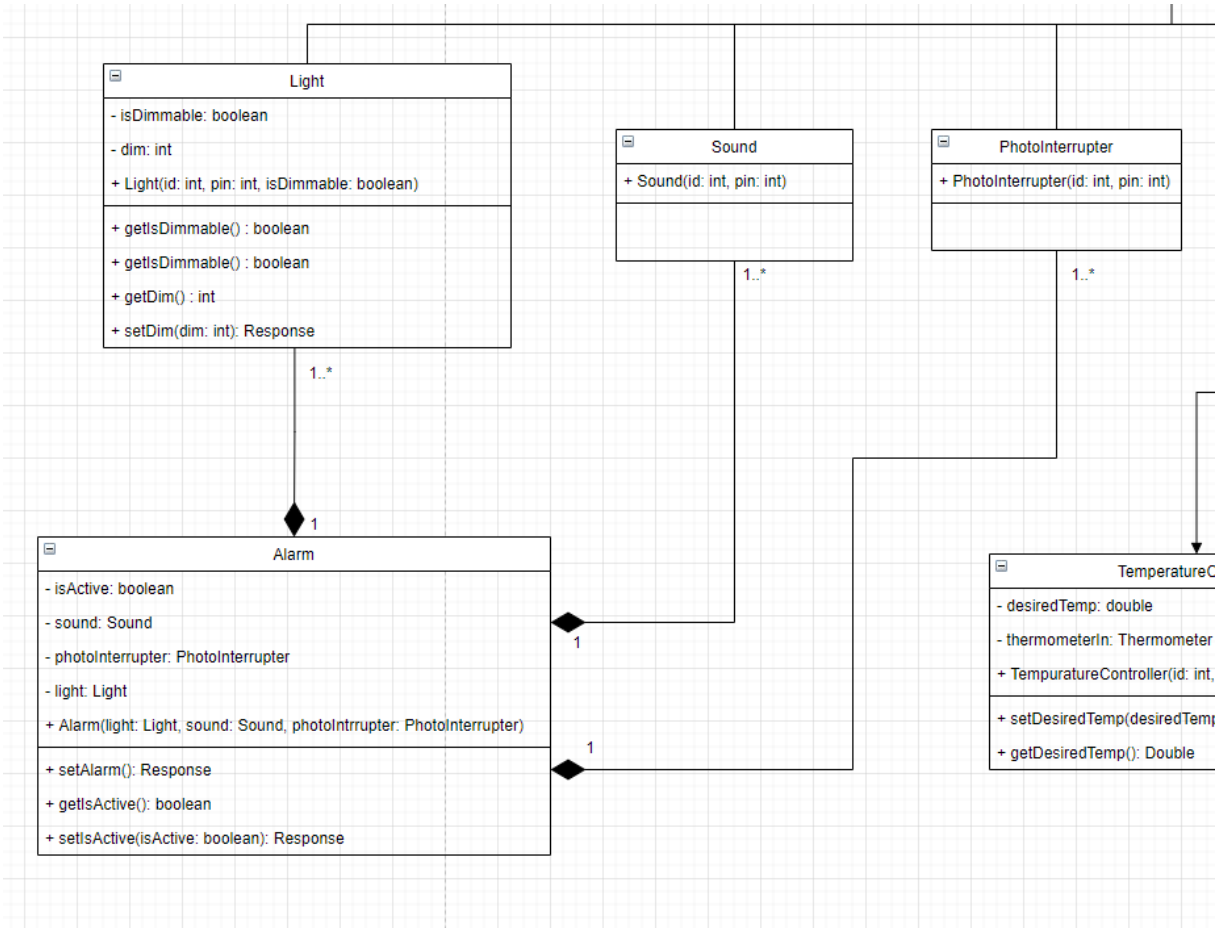

Figure 7 - Alarm Class Composition

D6

Temperature controller class (Figure 8) at this moment is a device which as of right now we are using as a place holder for whatever thermostat that would be used to adjust the temperature. The class has a desiredTemp as a variable which is of type double and the thermometerIn of the thermometer device. The functions used in the temp. controller class is the setDesiredTemp which takes desiredTemp double as a parameter and returns a response the other function is the getDesiredTemp function which returns a double.

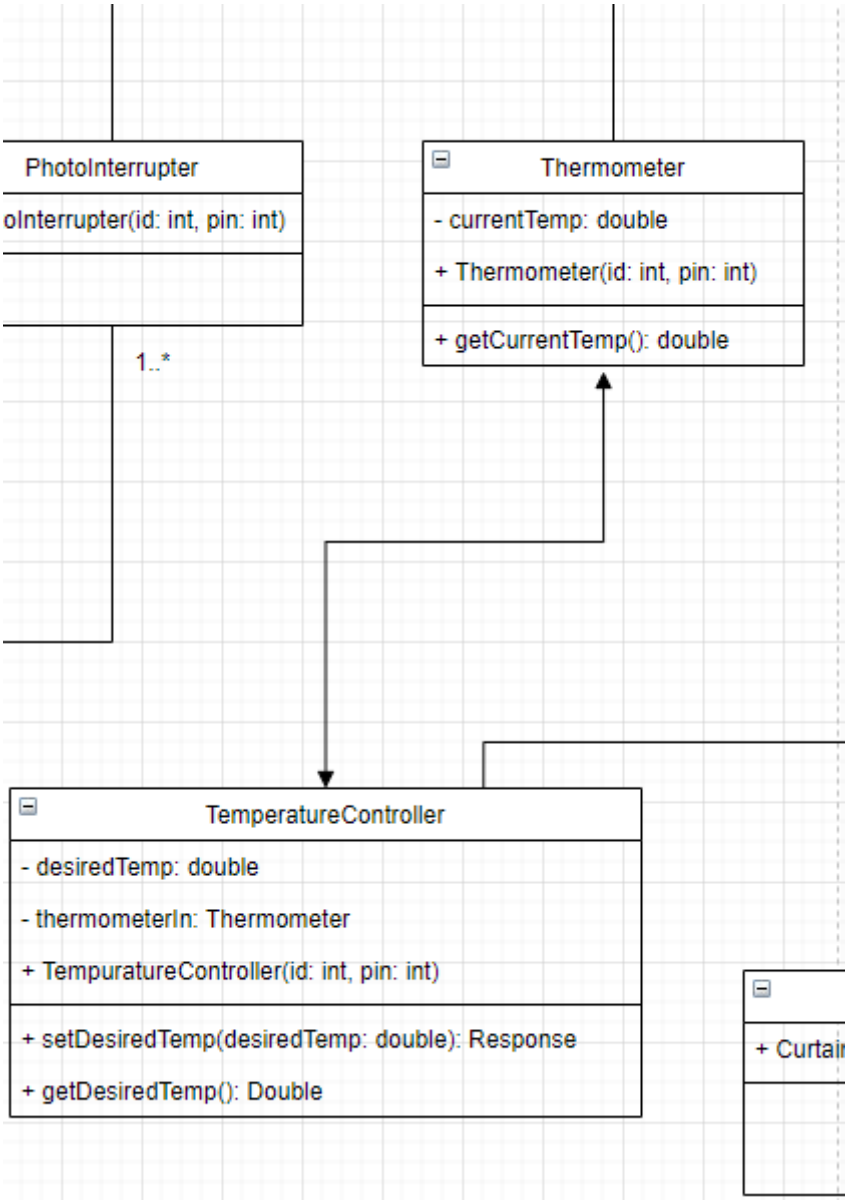

Figure 8 - Temperature Controller

D7

The response class (Figure 9) is used to relay a code and a message to the server to confirm the changes being made on the Arduino side of things or if a failure occurs it can be relayed and give a possible reason for it. The response class takes statusCode of int type as a variable and a message of type String. The constructor takes both variables and has getters and setters for both.

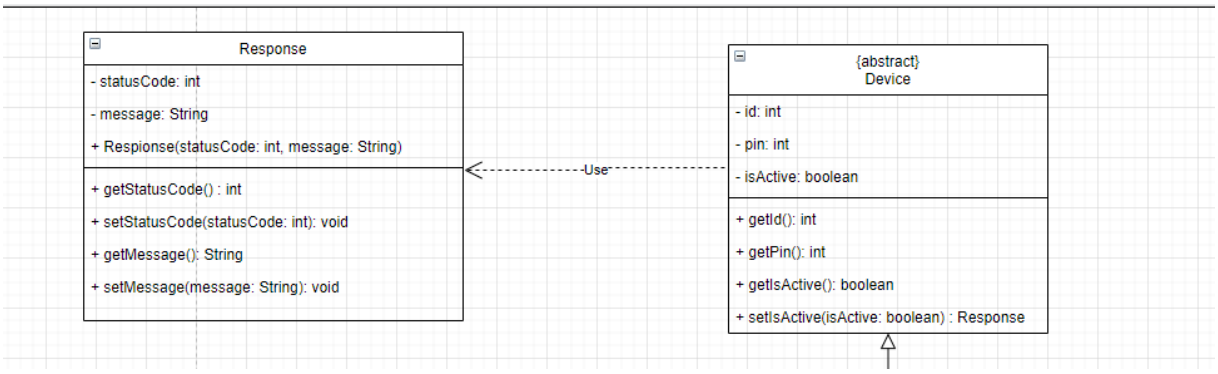

Figure 9 - Response Class

D8

The main class (Figure 10) has lists of the light device, curtain device, and fan device as variables also the alarm, response, and both an indoor and outdoor thermometer. The functions used so far are the setup and loop functions.

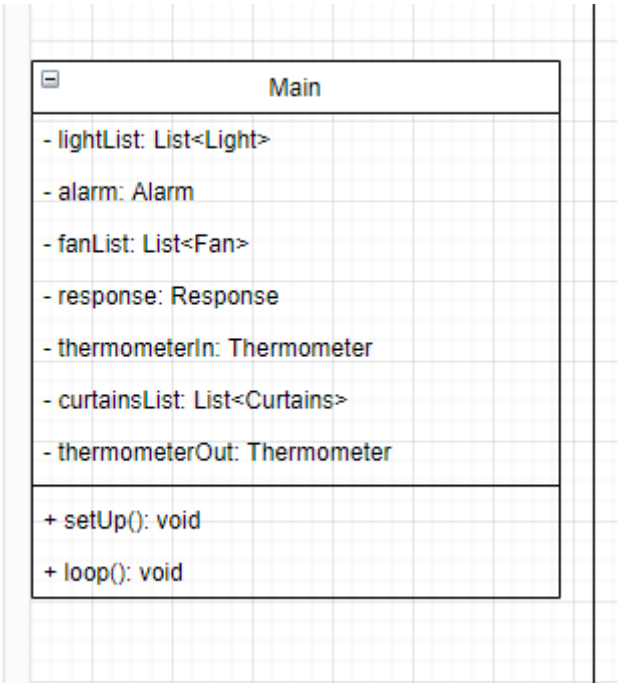

Figure 10 – Main Class

D9

The sequence diagram (Figure 11) depicts the reading of the temperature inside the home, the room temperature, a request is sent to the server, the server call the Arduino (Hub), the hub dispatches a request to the temperature controller the, the controller dispatches a request to the inside thermometer and returns a message to the temperature controller. The temperature controller creates and sends a response to the hub in turn sends a response to the server, the server then handles the response.

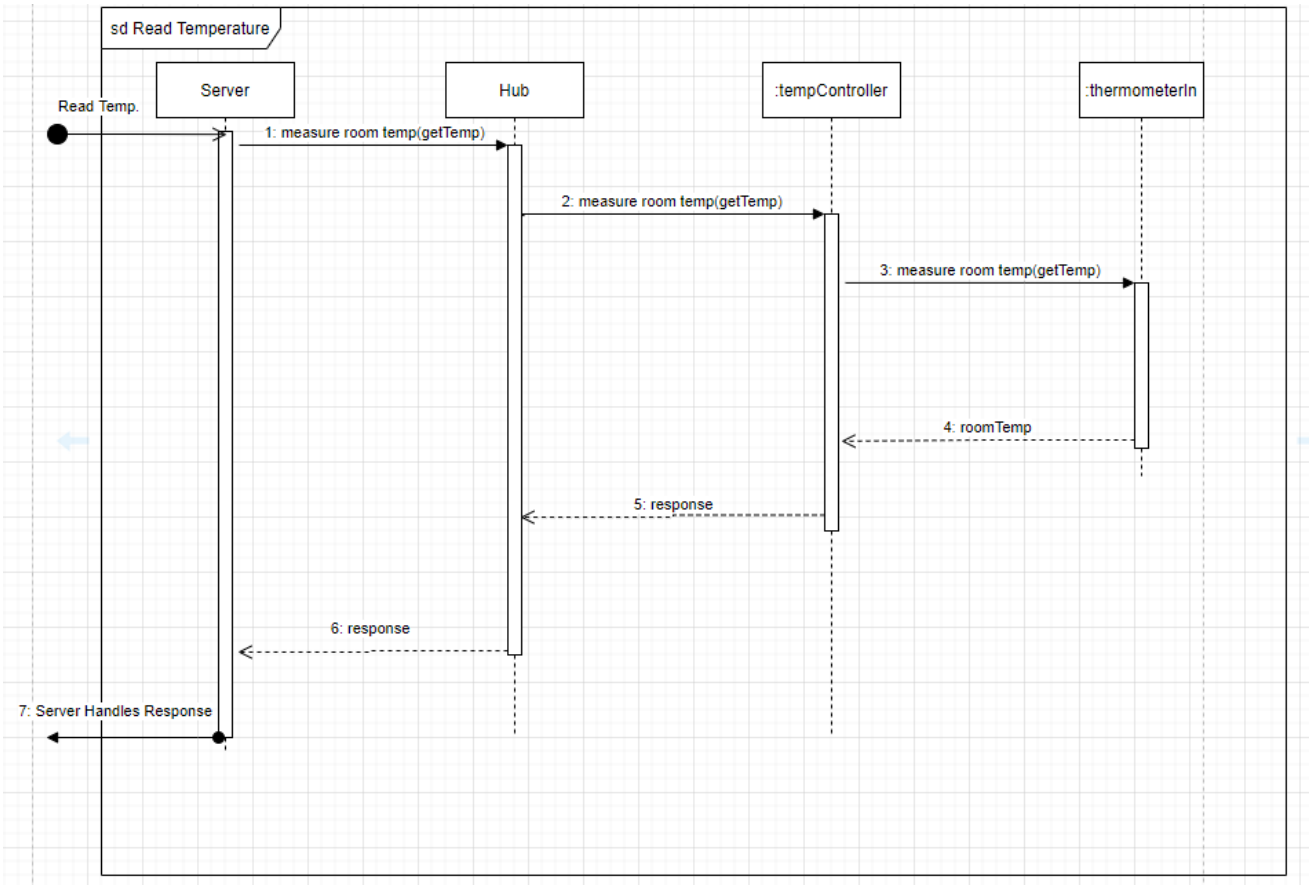

Figure 11- Sequence Diagram Read Temperature

D10

The sequence diagram (Figure 12) depicts the turning of a light on the Arduino (Hub), the server dispatches a request to the hub to turn the light on based off the id the hub then dispatches the request to the light. The light returns a message with a response and the hub in turn relays that response to the server, the server then handles the response.

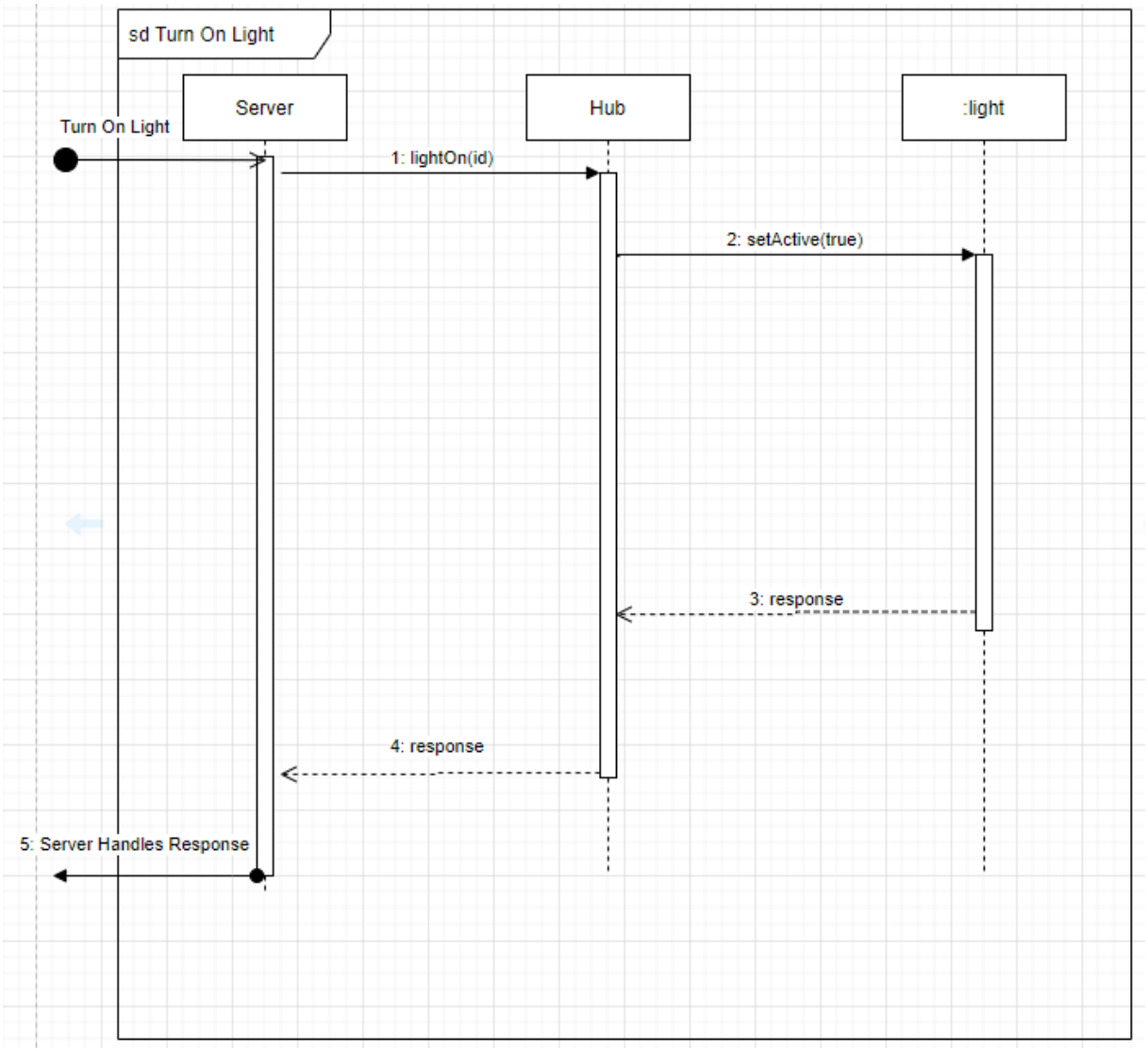

Figure 12 - Sequence Diagram Turn On Light

D11

The sequence diagram (Figure 13) depicts opening of a curtain on the Arduino (Hub), the server dispatches a request to the hub to open the curtain based off the id the hub then dispatches the request to the curtain. The curtain returns a message with a response and the hub in turn relays that response to the server, the server then handles the response.

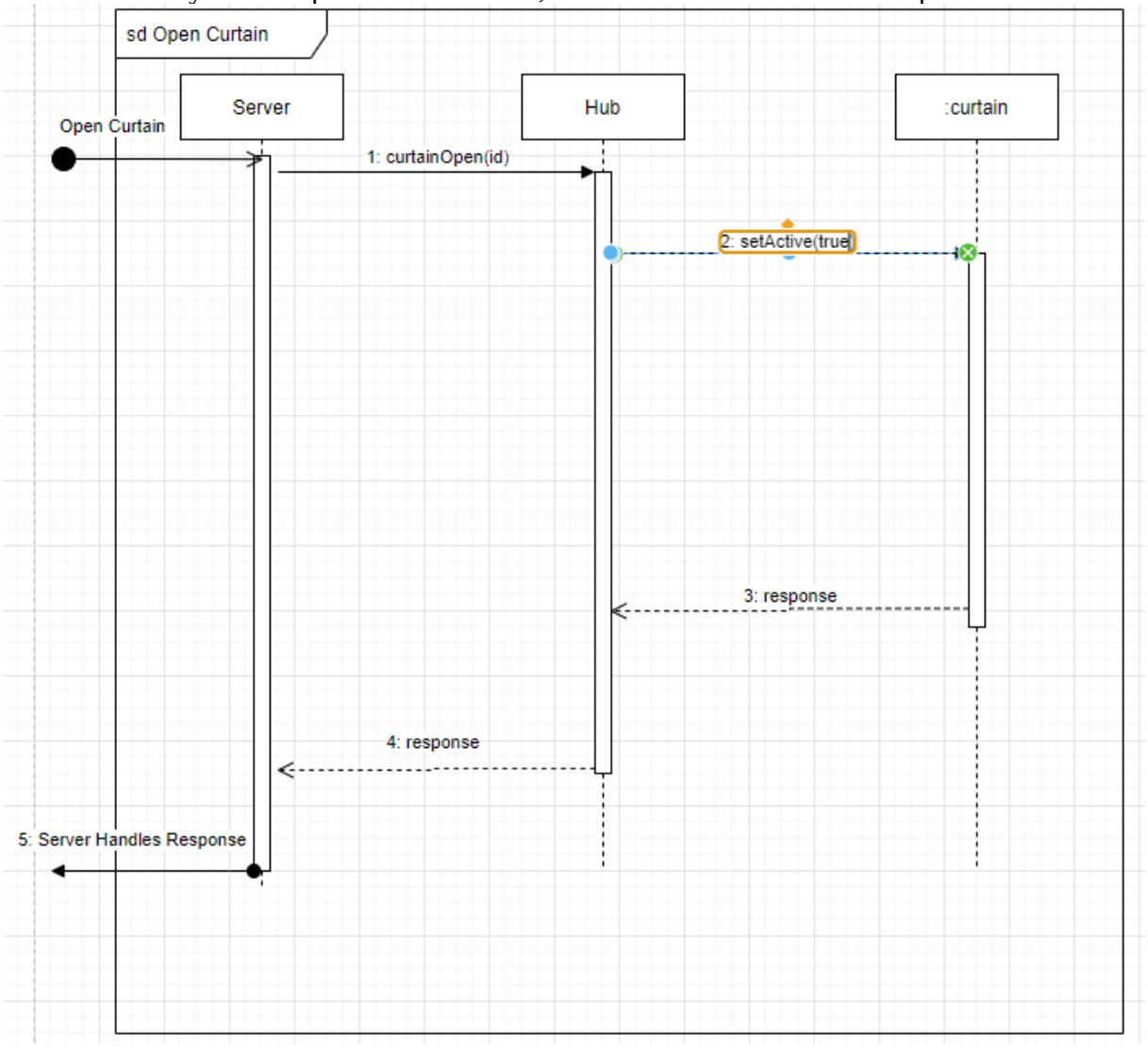

Figure 13 - Sequence Diagram Open Curtain

D12

The sequence diagram (Figure 14) depicts temperature control on the Arduino (Hub), as of right now the hub will initiate a check on the temperature through a dispatch on a timer for every hour or thirty minutes (theoretical) the tempController will dispatch a request for the current temperature to the inside thermometer which will return a message of the room temp. The temperature controller will compare the room temp to the desired room temp which will return the difference if the difference is greater than 0 the temperature is to high. The temperature controller will dispatch a request to the ac to lower the output. The ac will send a response back to the temperature controller. If the difference is less than 0 the temperature is to low. The temperature controller will dispatch a request to the ac to raise the output. The ac will send a response back to the temperature controller. If there is no difference the temperature controller will dispatch a response to the hub.

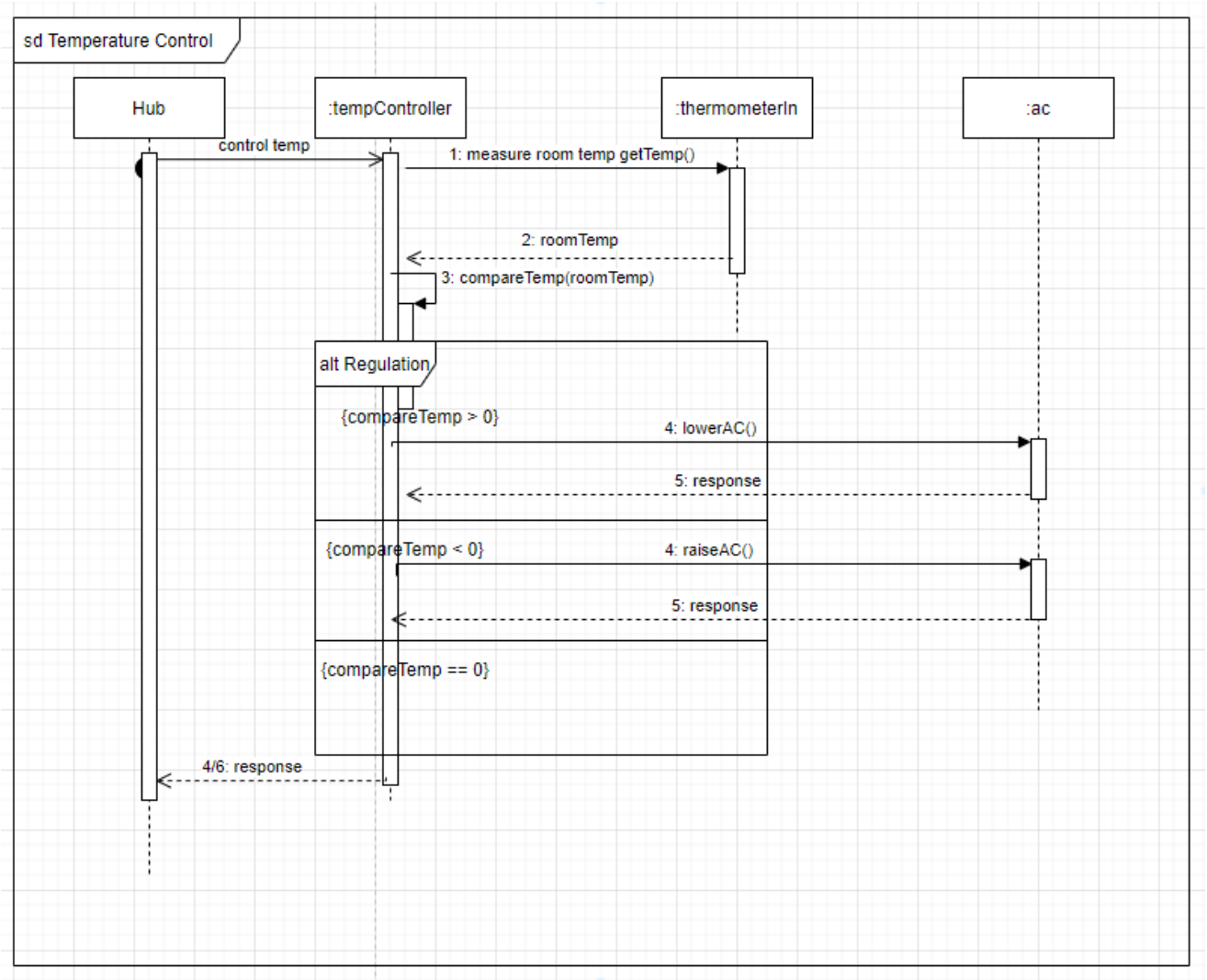

Figure 14 - Sequence Diagram Temperature Control

D13

State diagram for the temperature controls system.

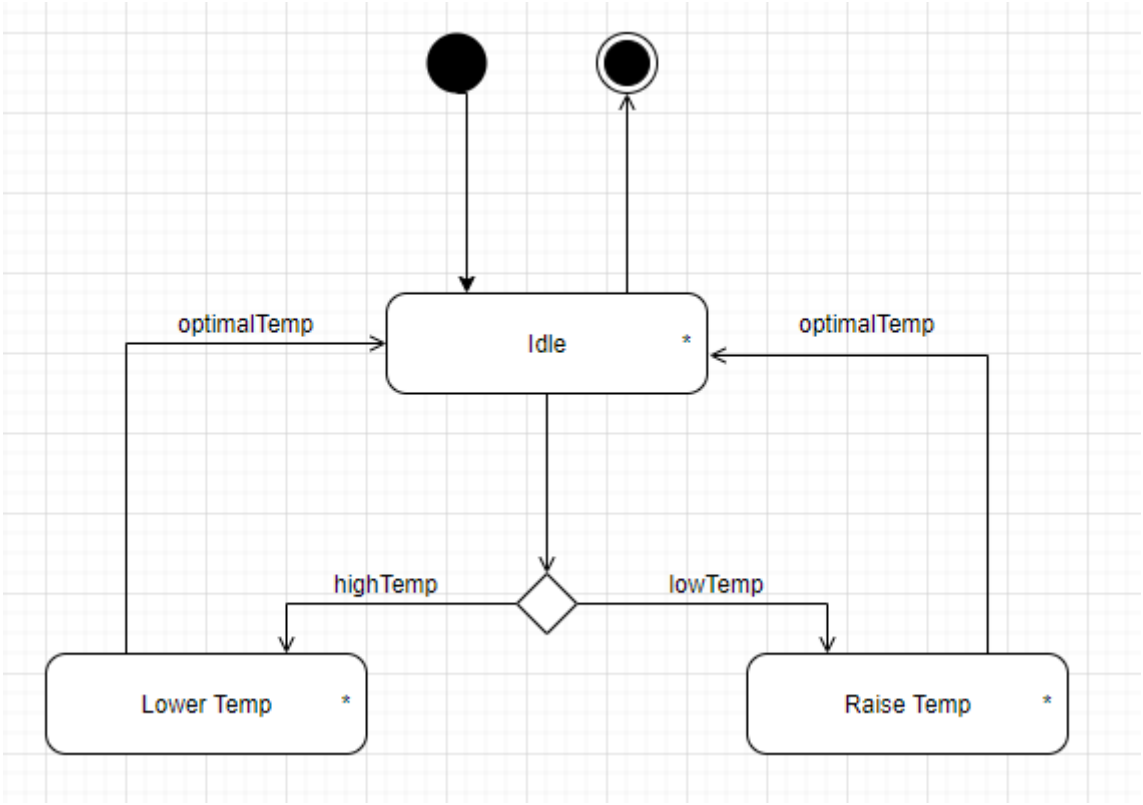

Figure 15 - State Diagram Temperatur Control

D14

State diagram for the curtains control.

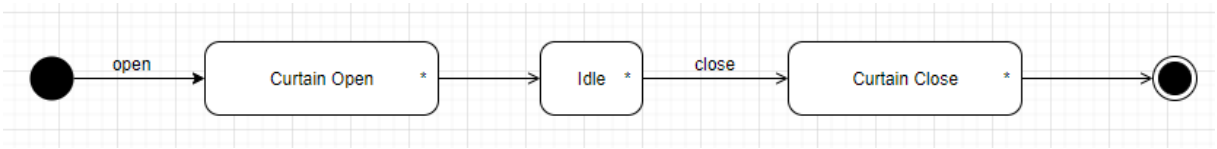

Figure 16 - State Diagram Curtains

D15

State diagram for the light control.

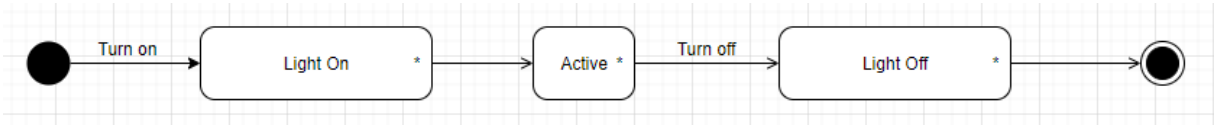

Figure 17 - State Diagram Lights