Improved software architecture report and newly added functionality for Baldr project

SEM2014

Li Wei

- Introduction---------- 1
- SAD ---------- 1,2
  - 2.1 class diagram ---------- 1
  - 2.2 Sequence diagram----------2
- Use case scenario for newly added functionality----------3,4,5
- To list what I have modified---------- 6

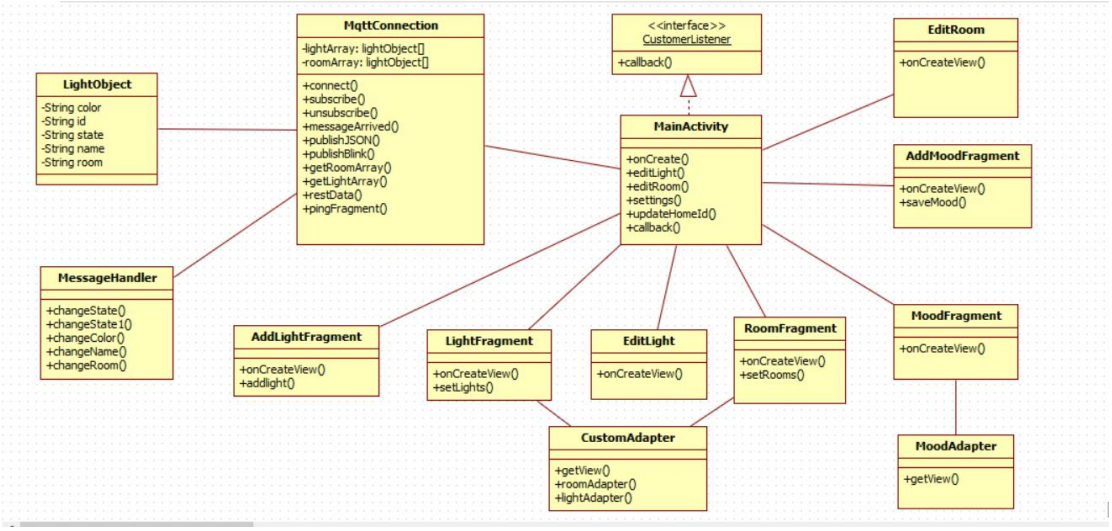
1. Introduction:

According to product owner's requirements, I updated Software architecture document (SAD) and add a new functionality to phone APP. Indeed, I add a " blink" button on edit_light fragment of UI. Whenever, the user press the button, the message will be published on MQTT broker then be pass to subscribed raspberry PI in order to control status changing of lamp. The control pin of lamp received the command to flickery lamp for 5 seconds. Finally, the message regards to lamp status changes will be reply to users via broker.

The general architecture we adopted to depict interactions among different software components can be found in the class diagram below.

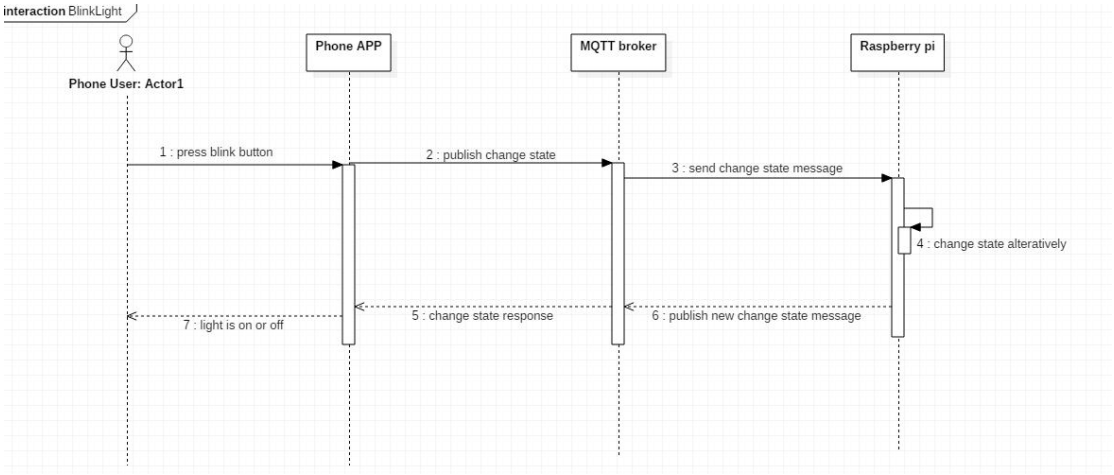
2. SAD document:

2.1. class diagram for Logic view (android APP)



As you can see from the diagram above, we add many fragments into mainActivity in order to create a flexible UI. Meanwhile, those fragments can interact each other through MainActivity. In addition, the customAdapter is functioned as bridge to fill the data into the list view. In other words, adapter can display array list in the UI. Moreover, the customListener can help bridge connection between MQTT server and mainActivity class and those methods declared in mqttConnections class can parse the Published message into JSON format in messagehandler class. Furthermore, More concrete demonstration that how Message flows will be displayed in the sequence diagram below.

2.2 Sequence diagram for process view:



The general scenario depicted in the diagram above show how users send message to raspberry PI via broker and how they received the replied messages through broker as well.

3.0 General Use Case scenario for newly added functionality:

Basic flow:

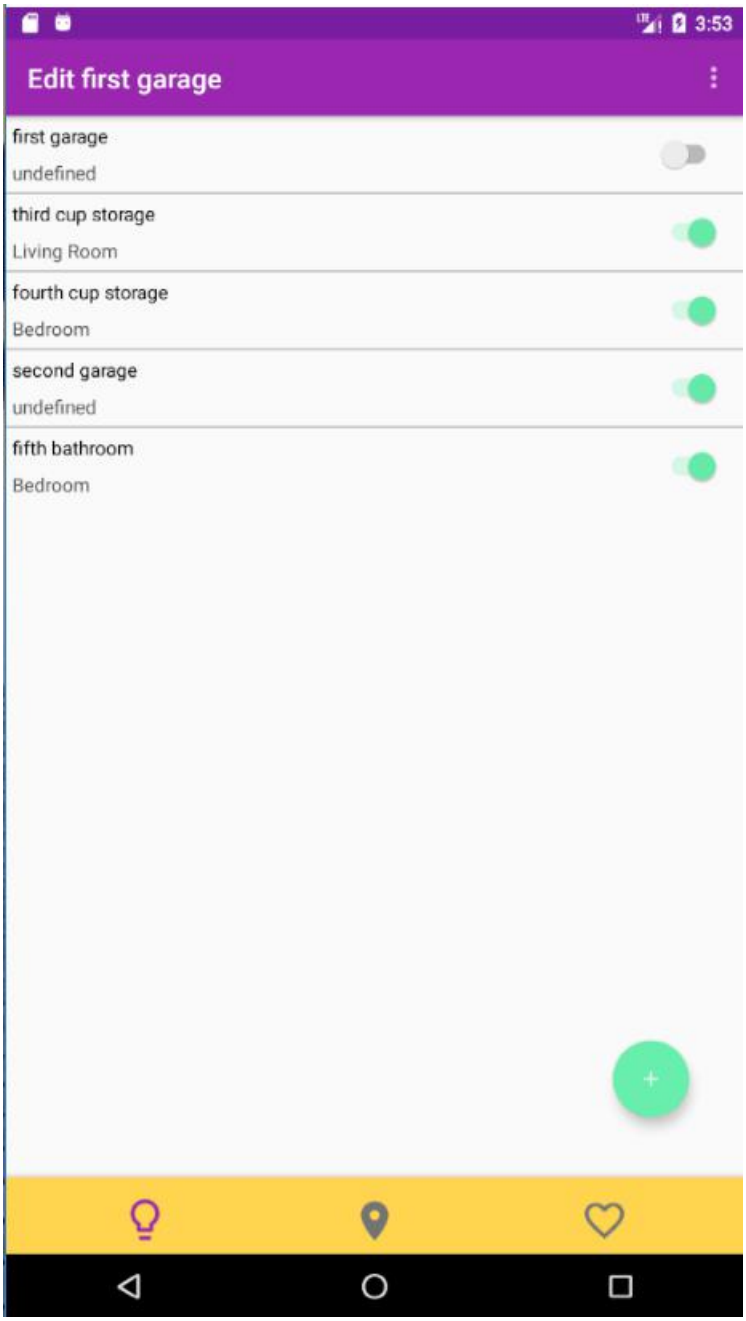
1.2. Flow of events

1.2.1 Basic flow

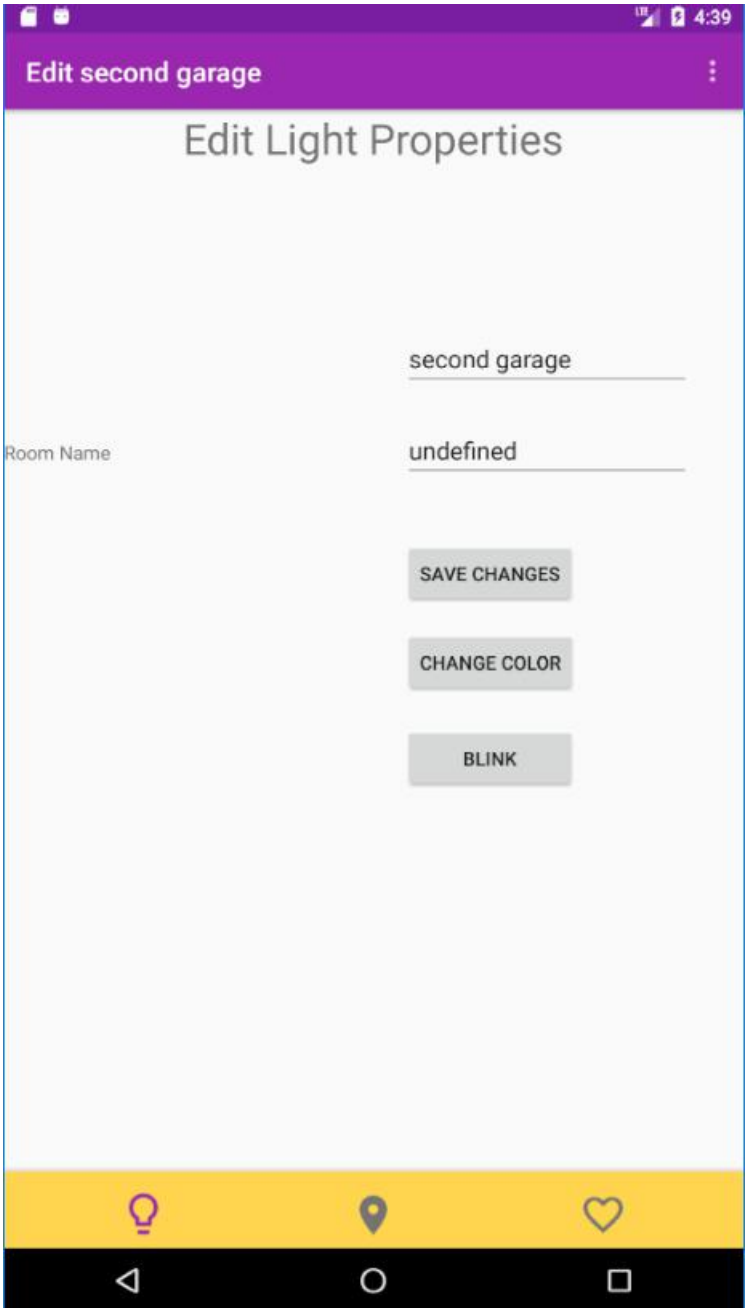
1.2.1.1 Use case begins with the application already started, showing a list of all lights owned by the homeowner.

1.2.1.2 Homeowner reads the list of lights he has access to and looks for the one he wants to turn on.

1.2.1.3 Homeowner double clicks on the power button of the light in lights fragment.



1.2.1.4 Homeowner entering editLight fragment then pressing “blink” button.



1.2.1.5 Remote control application sends a message to the broker. Message formatted according to RFC17 specifications.

1.2.1.6 The correct light will receive the message from the broker and make lamp flicker for 5 seconds.

1.2.1.7 The lamp sends a status message to the broker indicating the lamp is twinkling.

1.2.1.8 The remote control application receives the message from the broker and turns the visual indication on. The use case ends.

4. To list the classes which I have made modifications

Android app part:

- In resource folder, I inserted few lines of codes in order to create a button named “blinkButton” in layout/edit_light.xml file.
- I add lines of code in “EditLight.java” class in order to respond the blink button click event.
- I add a code block to define a newly defined method called “ publishBlink” in MqttConnection.java class.
- I add a JSON code block named “changeState1” in “Messagehandler.java” class in order to parse message into JSON format before it be send to MQTT broker.

Erlang part:

I add a method called update_lamp(state) in “baldr_light.erl” class in order to handle parsed message published into broker.