

*Responsible for this document:*

Oscar Axelsson

Daniel Olsson

Jacob Mejbvik

PUSS154214 v1.1 October 13, 2015

TEAM 2

---

# Software Top Level Design Document

---

*Authors of this document:*

Jacob Mejbvik

Oscar Axelsson

Daniel Olsson

### *Version History*

Version	Date	Responsible	Description
1.0	2015-09-24	DO, JM, OA	Baseline.
1.1	2015-10-08	DO	PR7, PR9, PR10, PR11, PR15.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Reference Documents</b>	<b>1</b>
<b>3</b>	<b>Overview</b>	<b>1</b>
3.1	Controller Application . . . . .	1
3.2	Packages . . . . .	2
3.3	UML diagram . . . . .	4
3.4	Sequence diagrams . . . . .	4

# 1 Introduction

This document describes the top level design of the Device Controller Application. The application can connect to a light bulb and a sensor device, these devices can also be controlled and data can be received using a MVD. The application is developed as a project within the course "Software Development for Large Systems - ETSN05" at LTH.

## 2 Reference Documents

PUSS154212 - System Requirements Specification for the current project

## 3 Overview

The main purpose of the Device Controller Application is to provide an graphical interface used to control a light bulb and a sensor device from an MVD via a REST API provided by the backend. The application will provide three different views to detect and control the different devices. A UML diagram have been created to visualize the design of a system figure 1. Several sequence diagram have been created to visualize the interaction and can be viewed in figures 2-5.

### 3.1 Controller Application

- **MyDeviceActivity:** Is the start screen of the application. Will contain a ListView and has methods for detecting devices and control a device that was selected from the ListView.
- **DeviceListAdapter:** Adapter that handles the list in MyDevicesActivity, the ListView can display any data provided that it is wrapped in a ListAdapter.
- **Device:** Is an abstract class that contains information about the two devices we are handling.
  - String getAddress() returns the String MacAddress value of a Device.
  - String getId() returns the String id number of a Device.
  - String getName() Is an abstract method that is declared without an implementation in this class.
- **SensorDevice:** Class containing the SensorDevice information.
  - String getName() returns the String Name of a Device.
- **LightBulb:** Class containing the LightBulb information.
  - String getName() returns the String Name of a Device.
- **DeviceActivity:** Is an abstract class that holds the protected parameters deviceName, id and macAddress.
  - void toggle(boolean) is a method that controls the on/off switch of the Devices set by the boolean value.
- **SensorDeviceActivity:** Is the controller in the interaction with the user in the Sensor-DeviceActivity. Handles the TextView fields and the buttons that will retrieve and present the information regarding the sensors from NetworkManager.

- **LightBulbActivity:** Is the controller in the interaction with the user in the LightBulb View. Handles the EditText fields and the buttons that will retrieve and send information from/to the NetworkManager.
- **NetworkManager:** Handles all the communication with the API to access the backend. There are different methods for both receiving and setting data values.
  - void toggle(Device, boolean, Callback<Response>) sets the on/off switch of the Device to the boolean value. Returns the response using a Callback.
  - void getToggleState(Device, Callback<Response>) returns the state of the toggle received from the API using a Callback. Device attribute to know to which host the connection should be made and which device the toggle concerns.
  - void detectDevices(List<Device>, Callback<DeviceResponse>) will return a List of Devices that will be displayed in the MyDeviceActivity using the adapter.
  - void getTemperature(Device, Callback<TemperatureResponse>) returns the Temperature received from the API using a Callback. Device attribute to know to which host the connection should be made.
  - void getPressure(Device, Callback<PressureResponse>) returns the Pressure received from the API using a Callback. Device attribute to know to which host the connection should be made.
  - void getHumidity(Device, Callback<HumidityResponse>) returns the Humidity received from the API using a Callback. Device attribute to know to which host the connection should be made.
  - void getMagnetic(Device, Callback<MagneticResponse>) returns the Magnetic received from the API using a Callback. Device attribute to know to which host the connection should be made.
  - void getGyroscopic(Device, Callback<GyroscopicResponse>) returns the Gyroscopic received from the API using a Callback. Device attribute to know to which host the connection should be made.
  - void getAccelerometer(Device, Callback<AccelerometerResponse>) returns the Accelerometer received from the API using a Callback. Device attribute to know to which host the connection should be made.
  - void getAllSensorValues(Device, Callback<AllSensorValuesResponse>) returns all the sensor values that could be retrieved from the API using a Callback. Device attribute to know to which host the connection should be made.
  - void getColor(Device, Callback<GetColorResponse>) returns the Color received from the API using a Callback. Device attribute to know to which host the connection should be made.
  - void setColor(Device, Color color, Callback<ColorResponse>) sets the color of the light bulb. Device to receive the correct host, Color to set the color and a Callback that will return from the API network.

## 3.2 Packages

- **network**  
Handles the network communication and contains the class NetworkManager.
- **network.data**  
Subpackage in network with responses from the server.
- **activity**  
Handles the Activities for the different views. Contains the classes DeviceActivity, MyDe-

viceActivity, SensorDeviceActivity and LighBulbActivity.

- **adapter**

The Adapter to the ListView used in MyDeviceActivity contains the class DeviceListAdapter.

- **model**

Handles the information about the different devices contains the classes Device, SensorDevice and LightBulb.

### 3.3 UML diagram

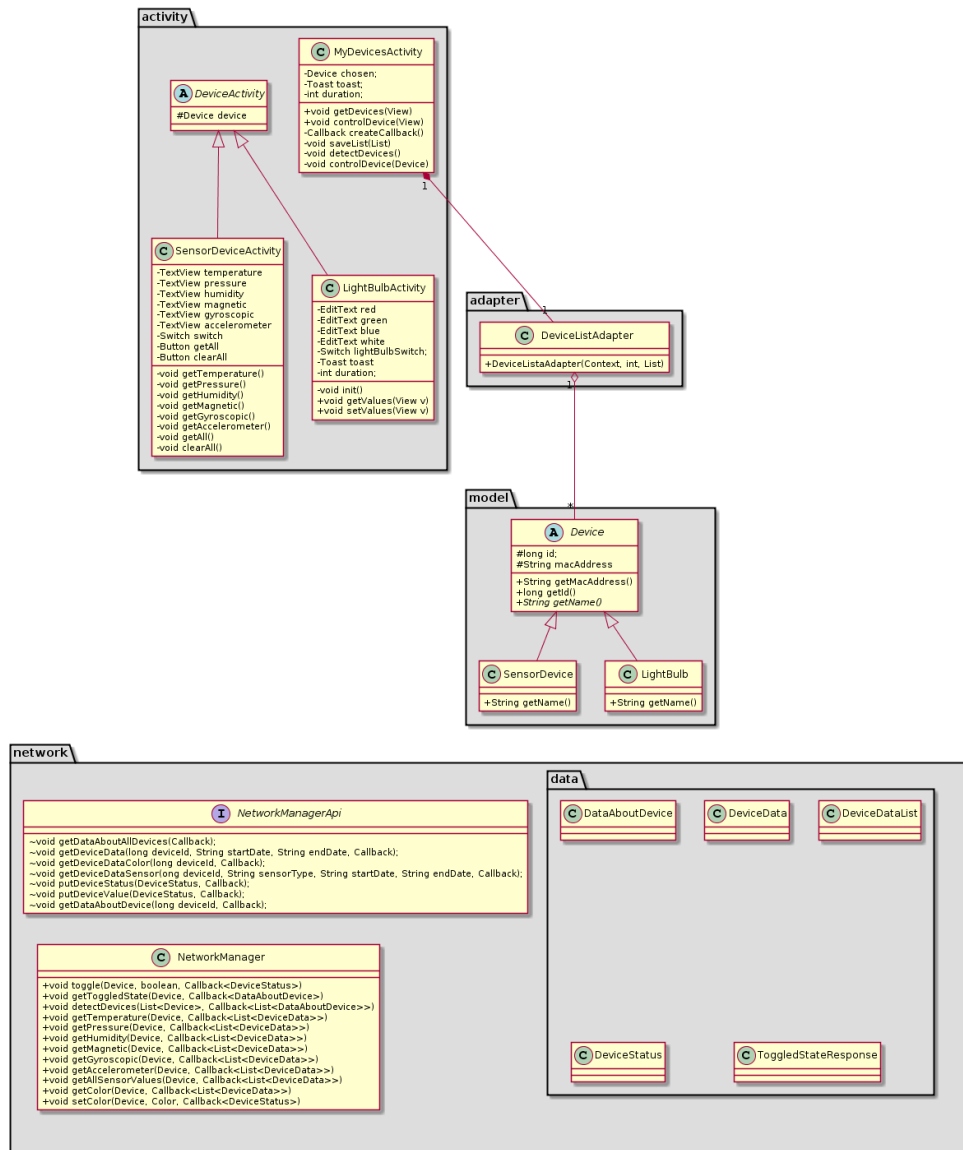


Figure 1: The design of the system. onCreate methods for each activity are not shown.

### 3.4 Sequence diagrams

The red arrows show internal calls within the application. The blue arrows shows communication outside the app which is with the MVD.

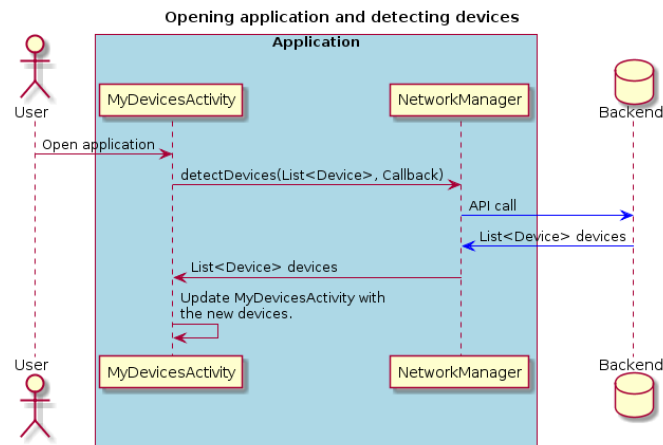


Figure 2: Opening application and detecting devices.

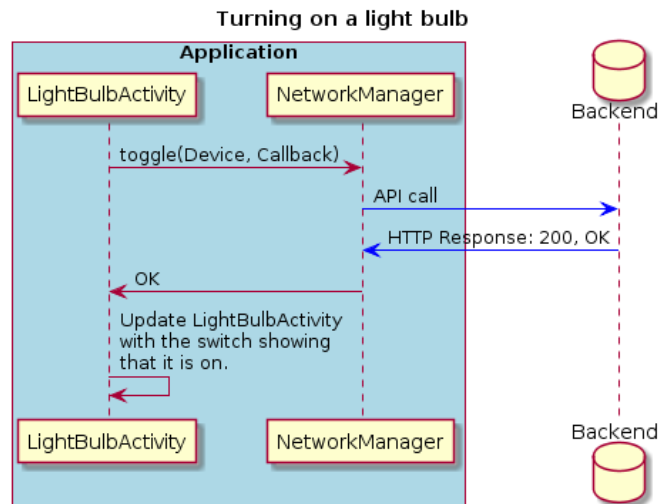


Figure 3: Turn on the light bulb with the user located in the LightBulbActivity.

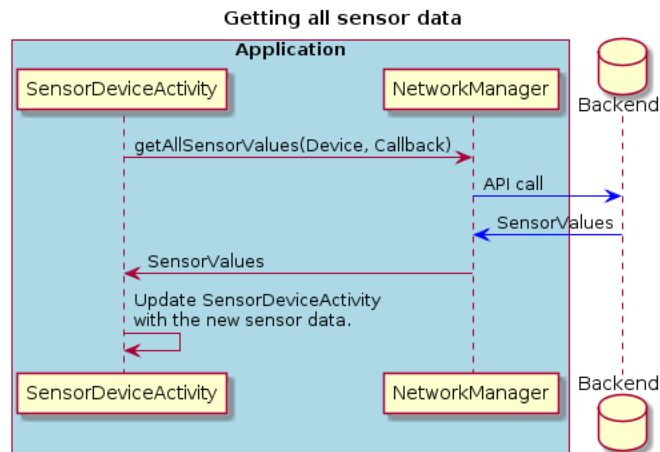


Figure 4: Get all sensor data with the user located in the SensorDeviceActivity.

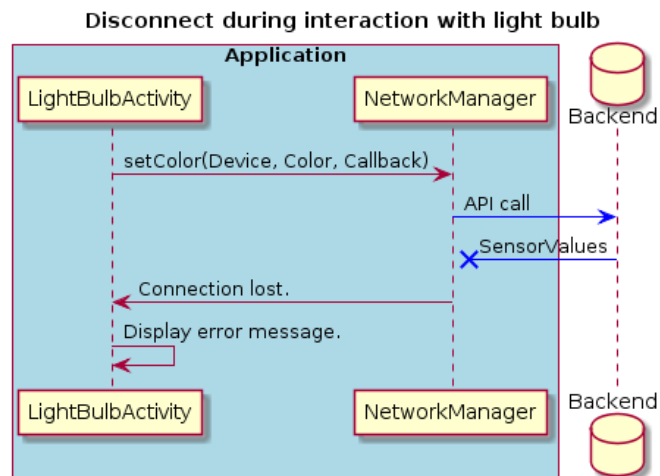


Figure 5: Set color of light bulb with the user located in the LightBulb view and the backend is unresponsive.