Software Design Document

Virtual Reality for Sensor Data Analysis

Project:	Virtual Reality for Sensor Data Analysis
Author:	Gero Birkhölzer, Johannes Blank, Alexej Gluschkow, Fabian Klopfer, Lisa-Maria Mayer
Last Change:	May 23, 2017 , Version 1.0

Contents

1 Purpose

The software project in the summer term 2017 at the University of Constance focuses on the development of apps for mobile devices. In the course of the project an Android app is being developed which allows the user to explore sensor data in virtual reality.

Especially, this Software Design Document intends to describe the internal structure of the app as well as test cases for the requirements stated in the Requirements Specification.

1.1 Product Idea and Goal

The general idea of the product is to allow the user to record data about their environment and later explore the data in a three-dimensional scene via virtual reality. Therefore, the developed product will consist of two parts:

Firstly, the app itself. It's main goal is to connect to an external sensor device via Bluetooth and to process and save the data collected by the sensor (referred to as "app").

In order to view the saved data, the second part consists of a web application where a virtual reality scene is generated and the stored data are visualized (referred to as "web application").

These two parts will be connected in such a way that the user can open a browser with the according web application from within the app.

1.2 Definitions

App When the app itself is mentioned, we refer to the application running on the smartphone that, as stated above, handles the recording of data and invokes a web browser with the web application.

Web application This term refers to the web site that can be invoked by the app, runs in a web browser, and provides the virtual reality display of the data gathered by the app.

Sensor (device) When referring to the sensor, we're talking about the sensor device (more clearly specified in section 2) containing several sensors.

Data With data we generally mean information that has been gathered by the sensor.

Virtual Reality (scene) This term describes the three-dimensional world in which the data will be displayed.

1.2.1 Abbreviations

TI Texas Instruments

VR Virtual Reality

3D three-dimensional

DB Database

App Application

BLE Bluetooth Low Energy

GATT Bluetooth Generic Attribute Profile

Characteristic Bluetooth Generic Attribute Profile Characteristic

1.2.2 Glossary

Stereoscopic 3D The impression of 3D is created by rendering different pictures for every eye of the viewer.

Virtual reality By using a headset in which the smart phone can be integrated, the user can view the three-dimensional world in stereoscopic 3D and thereby experiences the feeling of being fully immerged in the scene.

Augmented reality Displaying 3D objects in a real-world surrounding while providing an immersive experience like virtual reality.

Gyroscope sensor Sensor for measuring orientation in space.

Web application Web site that offers functionalities similar to those of "normal" desktop or mobile applications but runs in a web browser.

Service From AndroidDoc: "A Service is an application component that can perform long-running operations in the background, and it does not provide a user interface".

GUI From AndroidDoc: "They (Activities) serve as the entry point for a user's interaction with an app, and are also central to how a user navigates within an app (as with the Back button) or between apps (as with the Recents button)".

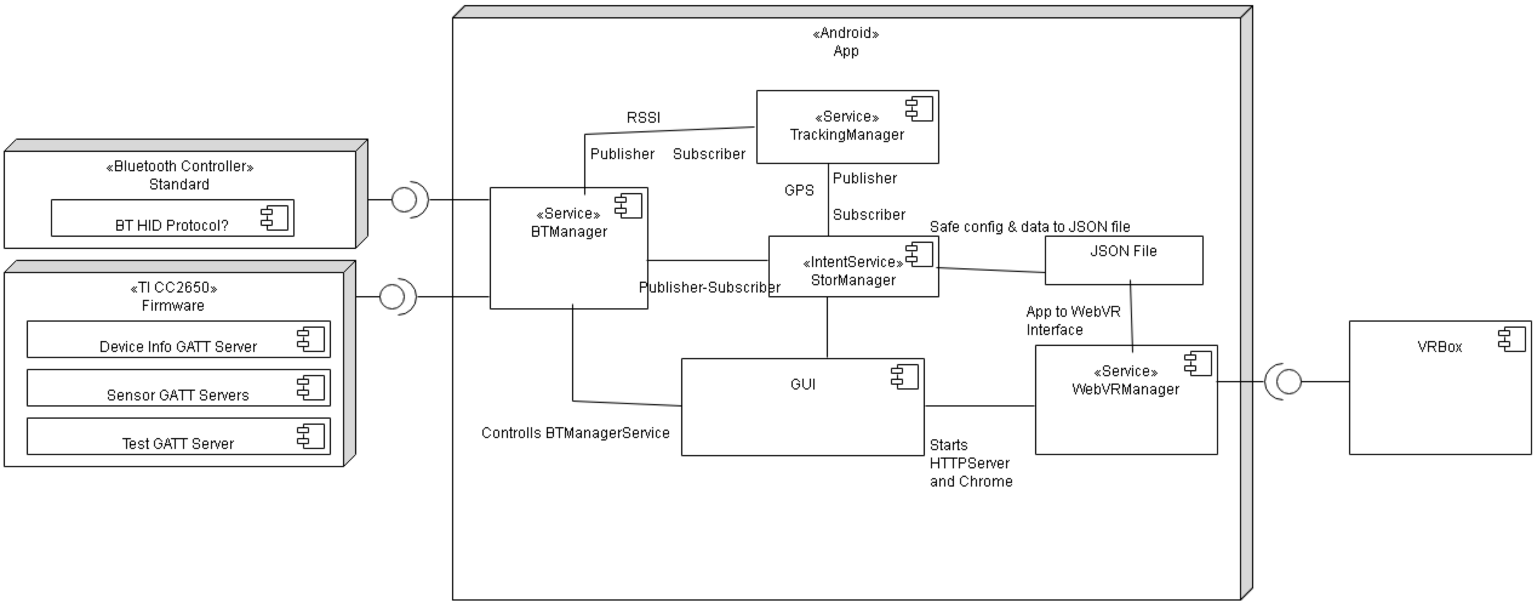
1.2.3 References

Requirements Specification Version 2.0 from May 15th, 2017.

2 Proposed Architecture

A better zoomable representation of these diagrams can be found in the github repository of this project in /doc/pflichtenheft/pics, where also the xml sources are.

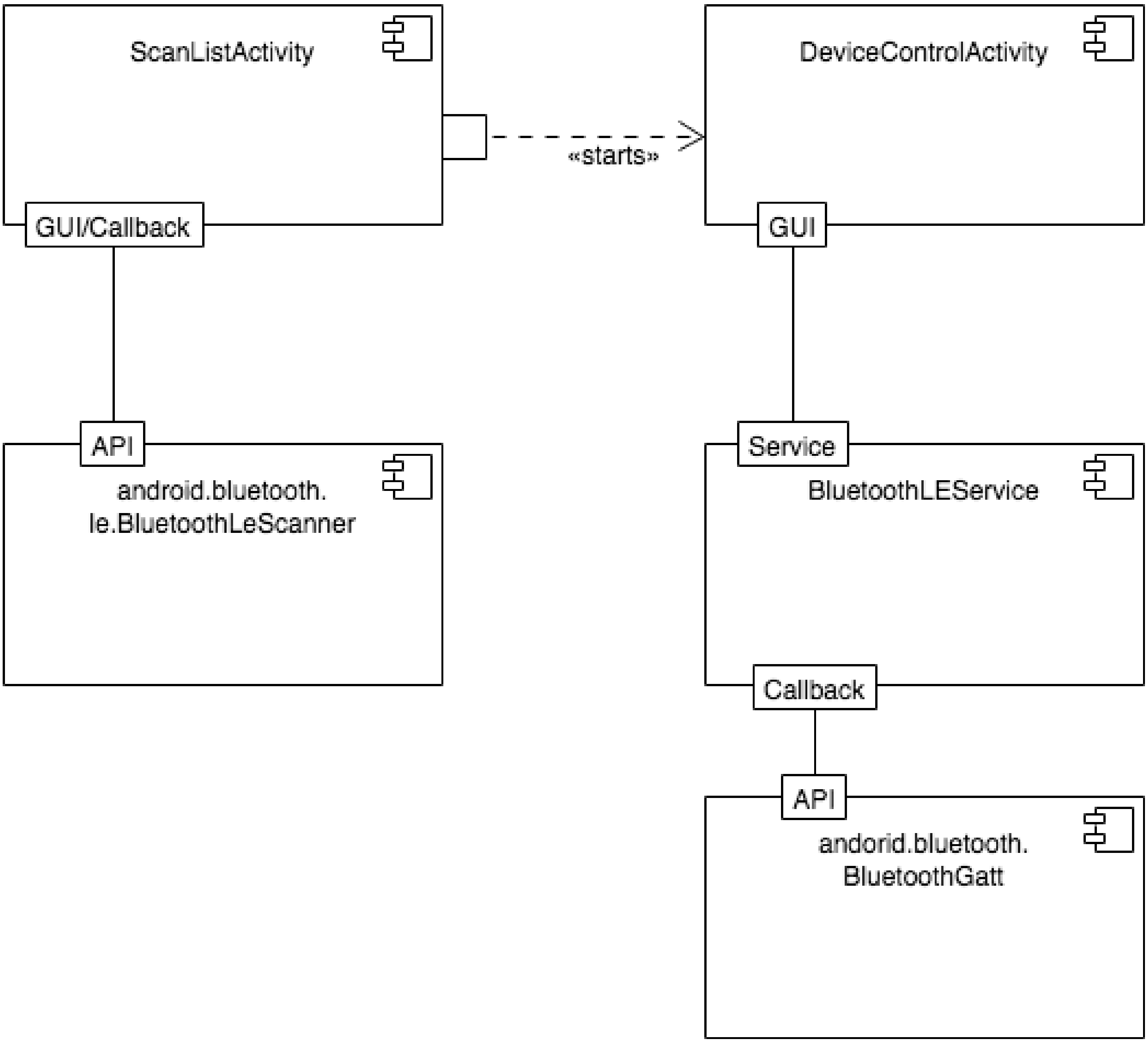
2.1 Overview



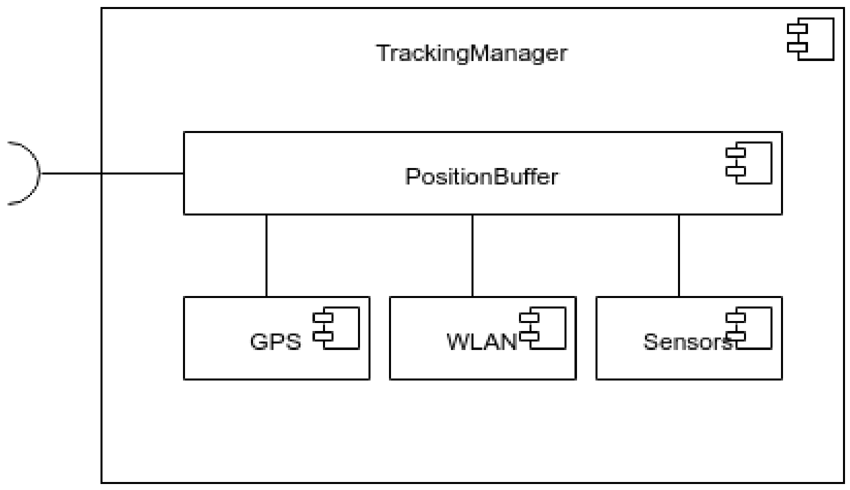
2.2 Component Decomposition

2.2.1 Services

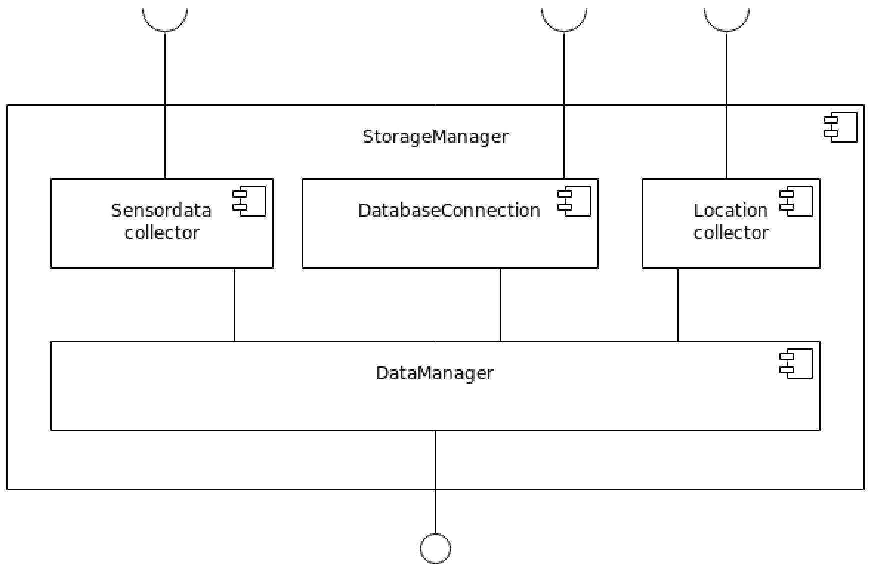
- **BluetoothManager:** Uses the android.bluetooth and especially the android.bluetooth.le libraries to fetch the sensor data from the sensor device.



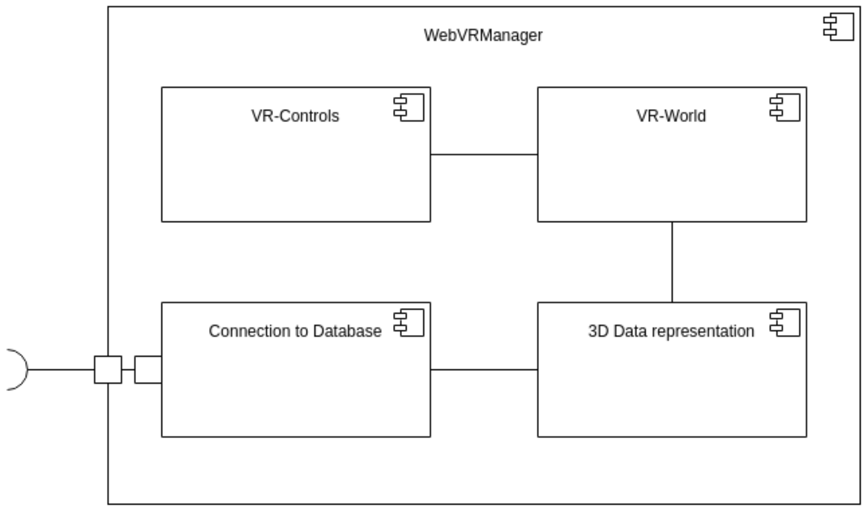
- **TrackingManager:** Handles the tracking of the (current) location where the data are recorded. The current position is determined by GPS and enhanced by the cellphone sensor and wifi data.



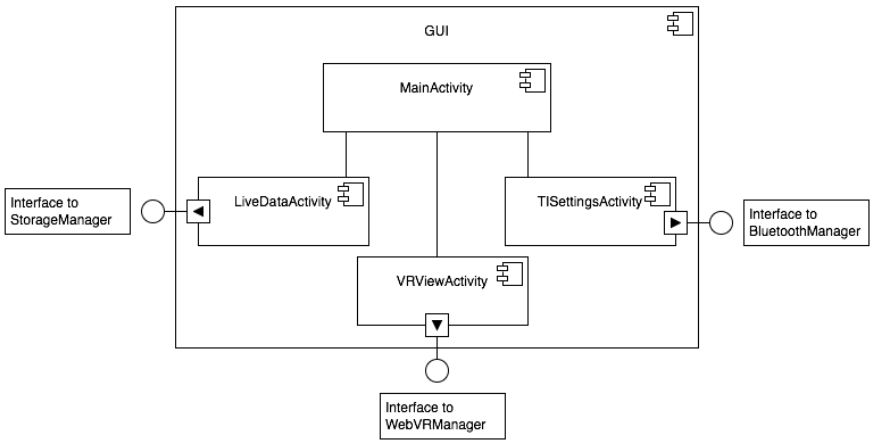
- **StorageManager:** Processes the data provided by the TrackingManager and the BluetoothManager. Uses a JSON file to store data.



- **WebVRManager:** Handles the display of the virtual reality scene and the given data from the sensor.



2.2.2 GUI


GUI
MainActivity
LiveDataActivity
TISettingsActivity
VRViewActivity
Interface to StorageManager
Interface to BluetoothManager
Interface to WebVRManager

- **MainActivity** Provides the main startup screen as the main entry point.
- **VRViewActivity** Shall open a new browser window to display the WebVR webpage.
- **LiveDataActivity** Shall provide a view of the sensor data in human readable form.
- **Settings:** Shall provide a settings screen containing scanning and connecting, connected devices and device settings fragments.
  - **ScanListActivity** shall show the scanning results, delivered by the android.bluetooth.le.BluetoothLeScanner and connect on click on the device.

- ◇ **DeviceControlActivity** shall provide propper read and write functions to show and edit the following configuration options of the sensors on the CC2650: Enable/disable the sensor, enable/disable the notifications, disconnect.

2.2.3 Additional Classes

- ○ **GATT TI CC2650 Service UUIDs**
- ○ **Parser and helper functions for each sensor** because the BLE protocol implemented in the TI CC2650 delivers raw sensor output

%subsectionHardware/ Software mapping

2.3 Global software control

2.3.1 Startup behavior

User: The user can start the app by pressing the icon on his screen. The app will show a startup screen and then transition after a short time into the main menu, from where the user can start the other functions of the app.

2.3.2 Interfaces

The **BluetoothManager** provides a possibility to connect to a sensor and then record the data. It ensures that a connection is established and that the data are broadcasted via a local BroadcastManager. The first listing below contains the names of the intents and what extra data they contains. The second one lists the interfaces of the Service-Connection, that's used to bind the service.

```
/**
 * @trigger is broadcasted if the Bt.-Adapter of the smartphone
 * connected successfully to the Generic Attribute Profile Server.
 * @data no extra data
 */
public final static String ACTION_GATT_CONNECTED =
        "kn.uni.inf.sensortagvr.ble.ACTION_GATT_CONNECTED";

/**
 * @trigger broadcasted if the Bt.-Adapter of the smartphone
 * disconnected from the Generic Attribute Profile Server.
 * @data no extra data
 */
public final static String ACTION_GATT_DISCONNECTED =
    "kn.uni.inf.sensortagvr.ble:ACTION_GATT_DISCONNECTED";
```

```
/**
 * @trigger broadcasted if the services of the connected
 *GATT Server are discovered.
 * @data no extra data
 */
public final static String ACTION_GATT_SERVICES_DISCOVERED =
    "kn.uni.inf.sensortagvr.ble:ACTION_GATT_SERVICES_DISCOVERED";

/**
 * @trigger broadcasted if a characteristic was read, if a
 * characteristic value changed (notification) or if
 *the RSSI is read.
 * @data EXTRA_DATA, float[3]
 * @data EXTRA_SENSOR, Sensor or "RSSI"
 *
 * Sensor is an enum that defines the following fields: UUID service, UUID da
 * There is a getDataFromUUID() method, that returns the last read value for
 * !This is not the measured but a safed value!; measurement is done via serv
 */
public final static String ACTION_DATA_AVAILABLE =
    "kn.uni.inf.sensortagvr.ble.ACTION_DATA_AVAILABLE";

/**
* Contains an element of the Sensor enum or the string "RSSI"
* (not in that enum because it has no UUIDs at all)
*/
public final static String EXTRA_SENSOR =
    "kn.uni.inf.sensortagvr.ble.EXTRA_SENSOR";

/**
* Contains converted raw data stored in a float array with 3 places
*/
public final static String EXTRA_DATA =
        "kn.uni.inf.sensortagvr.ble.EXTRA_DATA";

ServiceConnection
        void onServiceConnect(ComponentName c, IBinder b);
        void onServiceDisconnected(ComponentName c);
ServiceMethods
        boolean connect(String address);
        void disconnect();
        boolean enableSensor(Sensor s);
        boolean disableSensor(Sensor s);
        void readCharacteristic(BluetoothGattCharacteristic c);
        void setCharacteristicNotification(BluetoothGattCharacteristic
         c, boolean b);
        void broadcastCharacteristic(BluetoothGattCharacteristic c);
        void writeConfigCharacteristic(BluetoothGattCharacteristic
         c, byte b);
        List<BluetoothGattService> getSupportedGattServices();
Sensor enum
        UUID getService(Sensor s)
        UUID getData(Sensor s)
        UUID getConfig(Sensor s)
        Sensor getSensorFromUUID(UUID u);
        byte getEnableSensorCode(Sensor s)
        float[] convert(byte b);
```

The **StorageManager** provides a possibility to store the data from the sensor together with the current position. It also can store the data in a .json file and upload them to a webserver.

```
InterfaceStorageManager {
  public void setSensor(integer internalSensorID){
    activeSensor = internalSensorID;
  }
  //tells the storage manager to get data from the active sensor and
  //the tracing manager, mark them with a timestamp and store the data
  //in a DataSet.
  public void measureNow(){
  }
  //get the latest measured DataSet of the active sensor
  public DataSet getLiveData() {
    return DataSet;
  }
  //get a DataSet which contains data of the activeSensor and position
  //at the specified time. If no data was stored at that time, an empty
  //DataSet will be returned.
  public DataSet getDataFrom(int time) {
    return DataSet;
  }
  //get multiple DataSets which contain all measured Data from the
  //active Sensor between 'time' and now.
  public DataSet[] getDataSince(int time) {
    return DataSet[];
  }
  // Upload the data set to a webserver
  // a .json file from time till now.
  public void uploadData(int time) {
    //upload the data to a server so webvr can later use it.
  }
}
```

The **TrackingManager** gets the current location of the smartphone and provides it to the rest of the App.

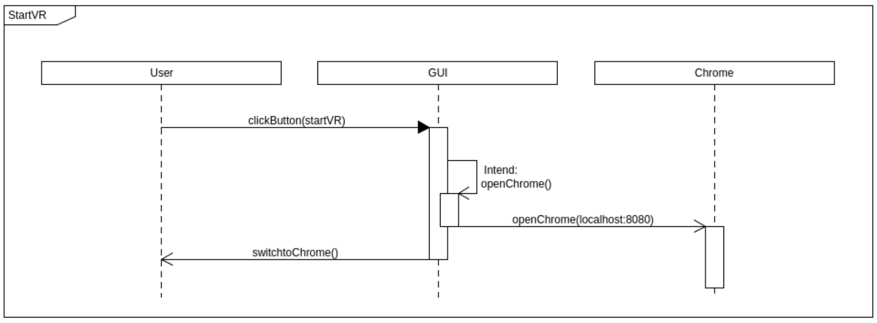
```
Interface TrackingManager {
  //get the current location of the device
  public Location p getCurrentPosition() {
    ensures iff lockCustomPos
      p == customPos;
      else
        p == currentPosition;
  }
}
```

The **VRBox** gets the data from a webserver and loads it into the VR-World.

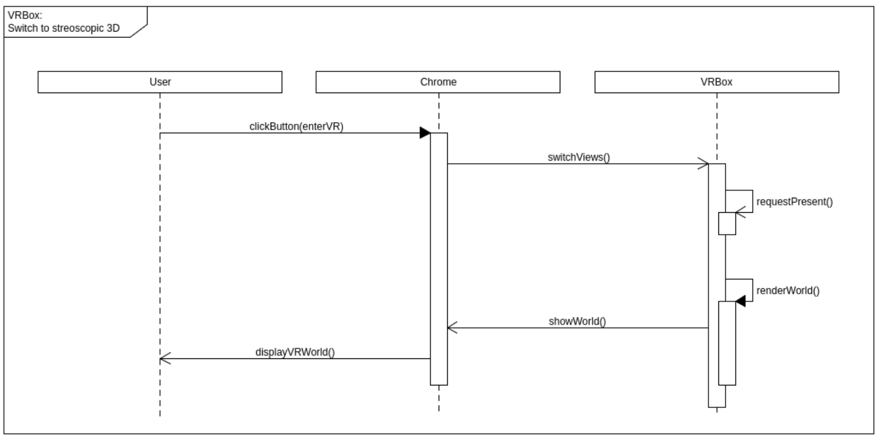
```
Interface WebVR {
  //Load the data from the webserver to display in VR.
  //The data will be a .json file which will be parsed
  // by in javascript by the web page.
  function loadData(file){
  }
}
```

2.3.3 Sequences

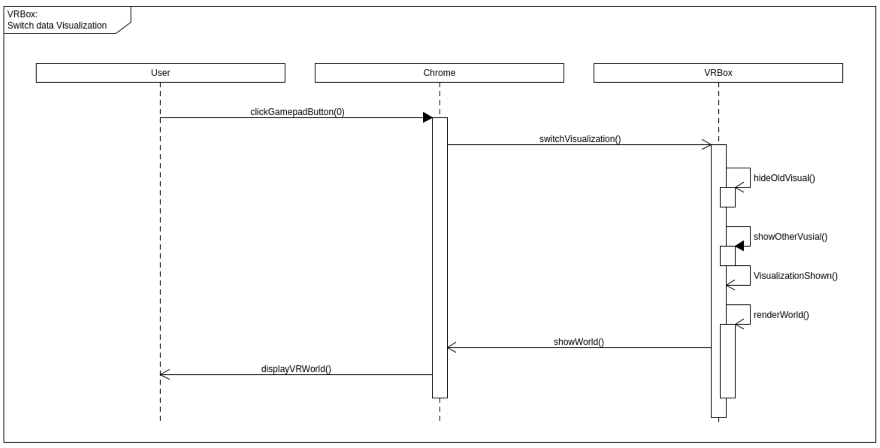
1) Start VR view: When the user clicks a button to start the VR view, a web browser is opened via an intent.



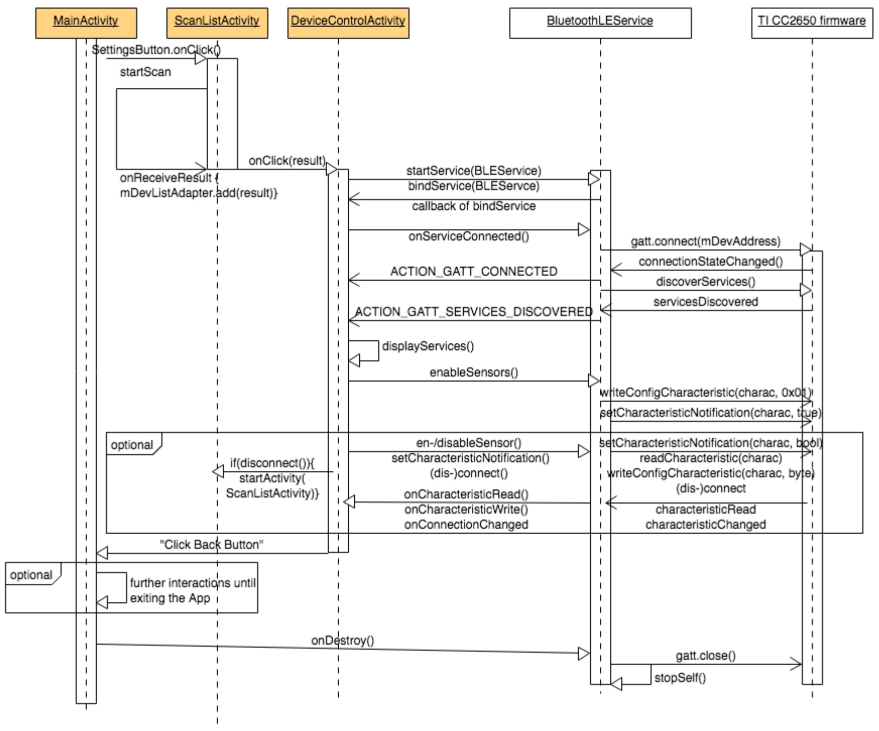
2) Switch to stereoscopic 3D: When the user enters the VR view, he can choose to switch the view and thereby switch to a stereoscopic view or switch back to non-stereoscopic view, respectively.



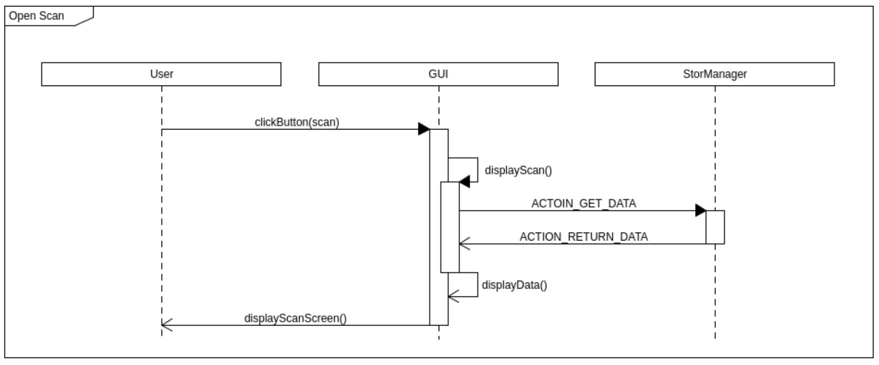
3) Switch data visualization: When the user opens the web site with the VR view, he can switch the visualization. Then, the data will be visualized in another way.



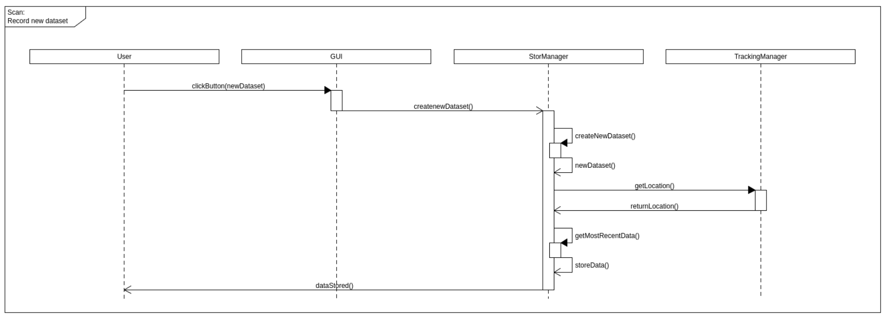
4) Scan and Connect: Scan for Bluetooth low energy devices and connect to one of them by clicking on the correct result ("TI CC2650"). After beeing connected one can edit the configuration of the sensors on the CC2650:
dis-/enable a sensor, (de-)activate the notifications for a measurement characteristic, read values, disconnect from the device. After that the user may start a new data scan record or start the VR View.



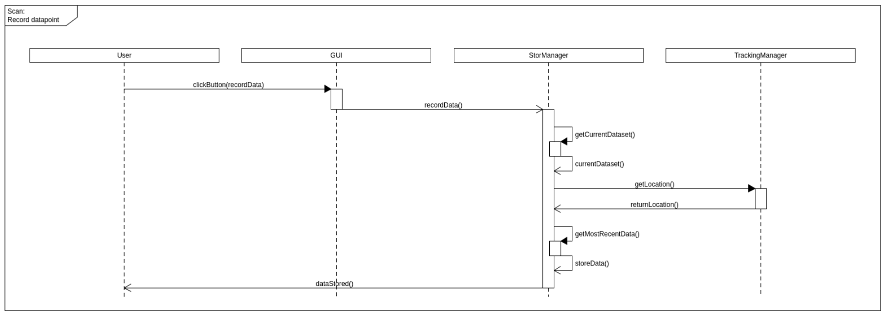
5) Open scan: text



6) Record new data set: When the user clicks the button to record a new data set, the StorageManager will provie the data and get the repective location from the TrackingManager to calibrate the initial location for the recording session.



7) Record data point: When the user clicks the button to record new data, the StorageManager will provie the data and get the repective location from the TrackingManager.



3 User interface

3.1 Structure

A small overview of the menu Structure.

3.1.1 Start screen

When the app is launched, the user will first enter the start screen. It features a menu where the user can choose which functionality to enter:
Either to record new data, to view the recorded data in the web application or to enter the settings screen where the sensor device can be scanned for and connected to.

3.1.2 VR Mode

The VR mode normally launches in normal 3D mode from where the user can switch to stereoscopic 3D view.

3.1.3 Record Data

Here, new data can be recorded as part of a new set of data or as part of an existing data set.

3.1.4 Settings

Here the user can select which sensor in range he wants to connect to and some basic settings like switching Bluetooth on and off and scan for more devices.

4 Test Cases

/T0300/ *Look around:* While in normal 3D mode the tester shall click the screen and drag first up to move the camera up. Then move down to move the camera down, then at last left and then right, all the time the camera must follow the movement of the finger. After this the tester shall tilt the phone up to move the camera up, then tilt it down, left and right. The camera shall follow the tilt direction of the phone all the time with no delay.

This test shall be repeated in stereoscopic 3D view. While the clicking and dragging shall not work, the tilting of the phone shall be the only way to pan the camera.

/T0310/ *Move inside the virtual reality scene:* While in normal 3D mode the tester shall tilt the joystick on the controller forward and the camera shall move forward. By tilting the joystick backward the camera shall move back, by tilting left the camera shall move left and by tilting right it shall move right. The camera shall always follow the view point, so forward is always in the center of the camera.

This test shall be again repeated in stereoscopic 3D view and all functions shall work the same.

/T0320/ *Searching, connecting and disconnecting devices:* The tester shall search a sensor device by pressing the "scan" button in the settings menu. All devices nearby shall be shown in a list with distinguishable entries. By tapping on a list entry a connection to the device shall be established. By tapping again on the list entry the connection shall be terminated.

/T0330/ *Displaying Data:* While in normal 3d mode the tester shall be able to see the data collected from the sensor. The tester shall be able to see the date relative to its stored position.
The test shall be again repeated in stereoscopic 3D view and shall work the same.

/T0340/ *Transferring Bluetooth data:* When the connection to a sensor device is established, the tester shall enter the data view wihin the app and see some representation of the transmitted data which allows him to check the connectivity and functionality to/ of the sensor device.

/T0350 *Storing Sensor data:* The tester shall be able to connect to a sensor and while connected the data received from the sensor shall be stored together with its current position, on the local file system. The tester shall open this file and check if the new data is in it.

/T0360 *Transferring sensor data to the Web Application:* The tester shall check if the data is transfered to the web application and shall check if the data is corresponds to the collected sensor data.