Advanced Software Engineering
Semester Project 2016 - 2017

Room Energy Profiling System: Monitoring and Intervention

Providakis Antonios

20-1-2017

Table of Contents

1 Introduction

In the context of the semester project of the Advanced Software Engineering course, we've been asked to analyze, design and pilot implement a specific piece of software. The main idea of the software, is a system for Room Energy Profiling, Monitoring and Intervention. A system that exploits different kind of inputs, i.e. sensors and weather web services, in order to define the energy profile of a room and to provide energy saving actions/recommendations that can be performed, so that the energy costs can be as low as possible. In addition, this software would work in two different modes: a) Non-Intrusive and b) Intrusive. In the first, the software will work autonomously, without the aid of other components. In the second, the software will interoperate with other components, e.g. an Arduino board, a database, weather web service, etc., which means that a room should be Profiling-Ready (pre-installed Arduino with sensors and interconnected appliances).

For this project, we assume that we are a Software House, that has been approached by a customer with the need for a software as the aforementioned one. We are going to perform all steps of delivering a piece of software, from analysis to implementation.

2 Requirements Gathering and Analysis

In order to start designing and implementing the software, first we should discuss with the customer about his needs and what the final product should be able to do, i.e. its functionality. With this in mind, the user expressed his requirements (User Requirements), then we documented some System Requirements, and lastly with a further analysis we ended up to a detailed guideline specification document that describes the system as a whole.

2.1 User Requirements

As we have already mentioned, the main goal of the software is to energy profile a room in two different modes (non-intrusive/intrusive) and generate useful information. These two modes, imply two different scenarios.

2.1.1 Scenario A – Non-Intrusive Profiling

- The software must run on smartphones or tablets
- The software must collect useful data about a space's energy profile, using: only the device's built-in sensors (autonomously)
- The software must display that data in an appropriate manner to the user
- Software should present the user tips or possible actions in order to reduce energy consumption

2.1.2 Scenario B – Intrusive Profiling

- The software must run on smartphones or tablets
- The software must collect useful data about a room's energy profile, using device's built-in sensors and Arduino's sensors in order to get richer and more accurate measurements
- The software must display that data in an appropriate manner to the user
- Software should present the user tips or possible actions in order to reduce energy consumption
- Software must save sensors data in a database
- Room administrator must populate database with room information (country, city, building name, room name, latitude-longitude coordinates, room usage, area size (square meters), humidity/temperature optimal values, installed sensors, installed appliances) before room is ready for profiling
- User must be able to intervene in connected appliances' operation state

2.2 System Requirements

2.2.1 Scenario A – Non-Intrusive Profiling

- The software should be a native Android or iOS app, or hybrid app
- The software must be able to use device's camera, microphone and light sensor
- Define profiling duration
- Collect data through sensors
- Use appropriate algorithms to profile the room
- Use microphone to detect high audio level, strange noises (e.g. appliances not working as they should)

- Use camera to detect occupancy
- Use light sensor to detect excessive light levels or lights operating at night while they shouldn’t be.
- The software should display comprehensible charts of each measurement
- The software should display possible tips or actions to the user in order to minimize energy consumption. For example, set A/C temperature to X degrees, close the windows, turn off the lights

2.2.2 Scenario B – Intrusive Profiling

- The software should be a native Android or iOS app, or hybrid app
- The software must be able to use device’s Wi-Fi, camera, microphone, and light sensor, and use installed sensors of the Arduino board in each room
- The software should be able to scan a QR core containing room-specific information (access point info, Arduino BT Mac Address, room id) and automatically a) turn Wi-Fi on and connect to Wi-Fi network, in order to download the Room Profile and be able to update the database, b) connect to Arduino through BT
- If a sensor with the same use is discovered on device and on the Arduino, get data from both
- Use appropriate algorithms to profile the room
- Use microphone to detect high audio level, strange noises (machines not working as they should)
- Use camera to detect occupancy
- Use light sensor to detect excessive light levels or lights operating at night while they shouldn’t be
- Use CO sensor to detect CO emissions (even really minimal levels. Indication of a malfunctioning device)
- The software should display comprehensible charts of each measurement
- After room profiling is fulfilled, the software should display possible tips or actions to the user in order to minimize energy consumption. For example, raise A/C temperature, close the windows, turn off the lights
- All measurements must be saved in a database
- The room administrator should be able to populate the database with Room information (country, city, building name, room name, latitude-longitude coordinates, room usage, area size (square meters), humidity/temperature optimal values, installed sensors, installed appliances), through a specifically designed GUI (*this GUI is not part of this project*)
- The software must provide appropriate GUI, so that the user can intervene in interconnected appliances

2.3 Detailed Specifications

Here, we provide a more thorough description of the system. We define a) the flow of the app, b) some more technical information and c) some rules on which an algorithm could be based on, in order to generate the profiling results and energy consumption reduction recommendations.

2.3.1 Scenario A – Non-Intrusive Profiling

The user enters the room that wants to profile and runs the software application on his smart device. From the menu, he chooses the "Non-Intrusive Energy Profiling". Then, he selects the timeframe (duration) of energy profiling to occur. As a last step, he rests the device, upside-down, in a corner of the room (in order to have a wide view, through the camera, of the whole room), starts the profiling process, and leaves.

The software takes pictures through the camera in a fixed interval (compares P and P-1 pictures) to detect room occupancy, records audio (decibels) through the microphone to detect when appliances are operating, meter light level (lux) through light sensor to detect if and when lights are on. Room profiling starts. The software, take snapshots of the above three readings at a set interval (e.g. 5 seconds). Every snapshot is temporarily pushed to a Firebase database (network connection is not required, as the app exploits the offline capabilities of Firebase). After the profiling has completed, some recommendations are generated.

- If there is no human activity in the room (detected by camera), and the lights are on (detected by light sensor) or any device is operating, makes noise (detected by microphone), then this is an energy consumption situation.

After the timeframe has passed, the user returns in the room, collects the device, and checks/analyzes the data, recommended actions, etc.

2.3.2 Scenario B – Intrusive Profiling

Prerequisites: The room administrator, sets up a microcontroller in the room. More specifically, the microcontroller is an Arduino board with sensors (temperature, humidity, CO, CT) and some interconnected appliances. The app requests data from the Arduino. In addition, a LAN connected to the internet is required.

The user enters the room that wants to profile and runs the software application on the smart device. From the menu, he chooses "Intrusive Energy Profiling". It is known that the room has a preinstalled microcontroller (Arduino board) with sensors and interconnected appliances. The app connects to the Arduino though Bluetooth. In order to automate the process of room identification, communication with the Arduino and internet access, the user scans a special QR code that contains the Room Id that corresponds to a specific Room (country, city, building name, room name, latitude-longitude coordinates, area size (square meters), humidity/temperature optimum values, installed sensors, installed appliances), Access Point info (SSID, password) and Arduino's BT Mac Address. After the scan, app: a) connects to Wi-Fi, b) connects to Arduino (through Bluetooth), c) downloads and displays room information. Then, the user starts the energy profiling process. The device collects data from both its built-in sensors and the board's. The application uses the device's microphone to detect when devices are operating, light sensor to detect if and when lights are on. In addition, the board has a temperature sensor to meter the indoors room temperature (Celsius), a humidity sensor to meter indoors room humidity (percentage), a CO sensor that meters any CO emissions (ppm), and a CT sensor that is attached to the live feed cable of the breaker panel and meters the total room energy consumption (watts). Also, the system makes use of APIXU API[1] (weather API) to get external environmental conditions, e.g. temperature,

[1] https://www.apixu.com

humidity, if it is day/night, etc. When profiling is completed, the system, based on the profiling data, generates recommendations to reduce energy consumption. The reading snapshots and recommendations are pushed to the Firebase database. Also, the user can intervene, through the app, in the operating state of the interconnected appliances.

- The CT sensor calculates the room's total electricity consumption
- App compares indoor humidity to outdoor humidity and Room Profile optimal humidity, and:
    - Recommends best humidity for the situation
    - If indoor humidity is higher than the outdoor, recommends to turn off A/C for a while, open windows to ventilate the room for humidity level to get lower, in order to reduce energy consumption (as with high humidity, A/C works harder)
    - Informs the user about dangerous levels
- App detects any CO emission, which might mean that a device is malfunctioning and as a result consumes more power.
- App compares indoor to outdoor temperature and optimal Room Profile temperature, and:
    - Recommends best temperature for the A/C
    - If indoor temperature is higher than the outdoor (and outdoor temperature is in normal level), and the A/C is on, recommends to turn off the A/C and open the windows in order to reduce energy consumption
    - Informs the user for dangerous levels

Inputs		Measurable Quantities	Communication Channels
Mobile Device	Arduino Sensors		
Camera		Motion	Wi-Fi
Microphone		Audio	Bluetooth
Light Sensor		Luminance	
	Temperature	Temperature	
	Humidity	Humidity	
	CO	CO	
	CT	Room total energy consumption	

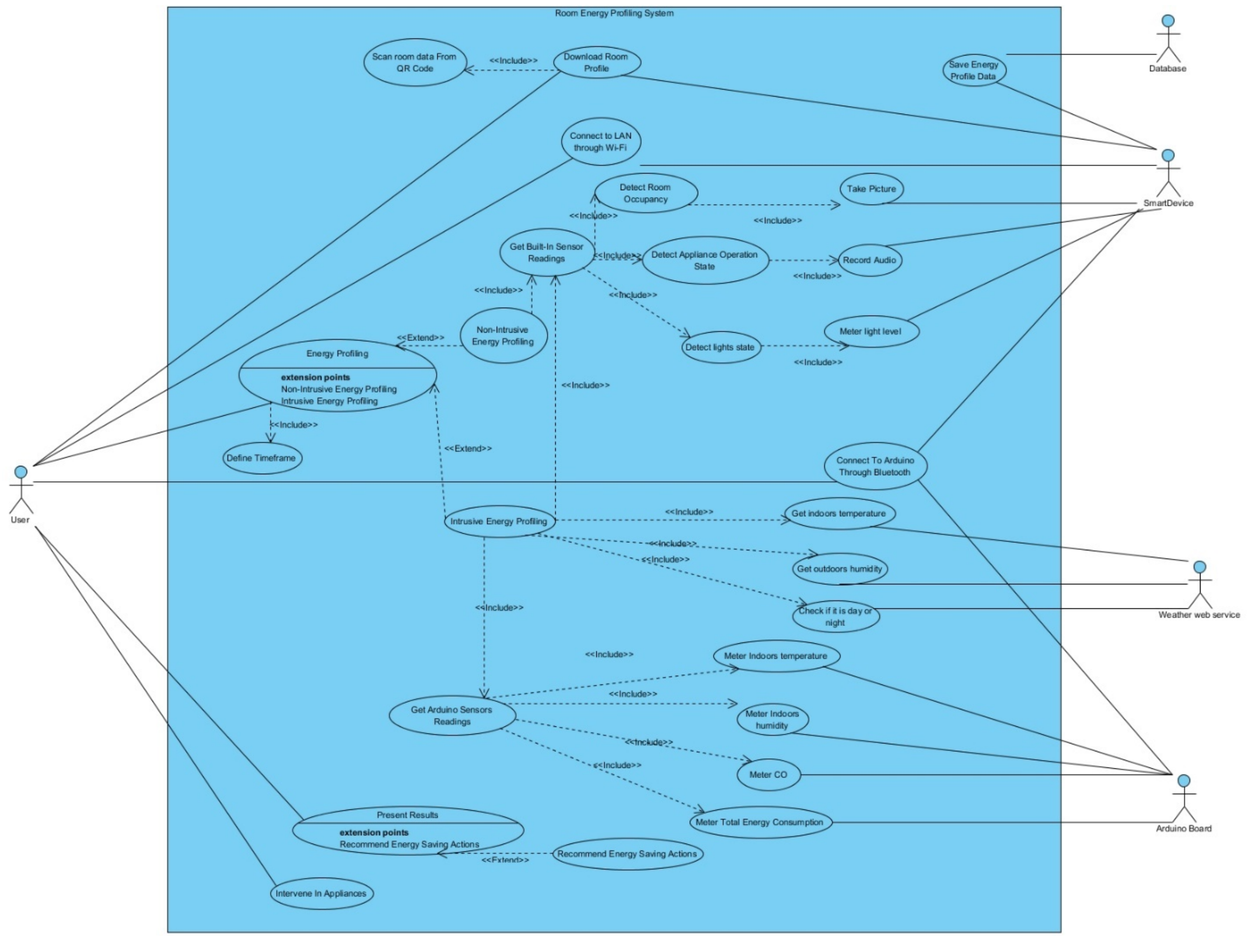
2.4 OS and Programming Language

The customer wants to target the mobile world, i.e. smartphones and tablets. Many platforms exist for these kind of devices, e.g. Android, iOS, Windows Mobile, etc. To begin with, the customer wants to target the Android platform and if the market is happy with the product then he will aim for other platforms. This choice, leads us to design and implement the software as an Android app and, thus, use the JAVA programming language.

3 System Design

Having finished with the user and system requirements, we can begin designing the system. For this project, we will follow an Agile Software Development model. We will work in short sprints trying to implement the features, asked by the customer, incrementally. We are not limited to the initial versions of Design or Architecture documentations. In every sprint, we can revisit each step and make any appropriate changes.

The design will be in the form of UML diagrams: Use Case diagram, Class diagram and Sequence diagram. From the previous analysis, we can distinguish five actors that interact with the system: a) a User, b) a Database, c) a Smart Device, d) a Weather Web Service and e) an Arduino Board. The interactions between them and the use cases we defined are shown at Figure 1.



Room Energy Profiling System
Scan room data From QR Code
<<Include>>
Download Room Profile
Save Energy Profile Data
Database
Connect to LAN through Wi-Fi
SmartDevice
Detect Room Occupancy
Take Picture
Get Built-In Sensor Readings
Detect Appliance Operation State
Record Audio
Non-Intrusive Energy Profiling
Detect lights state
Meter light level
<<Extend>>
Energy Profiling
extension points
Non-Intrusive Energy Profiling
Intrusive Energy Profiling
Define Timeframe
Connect To Arduino Through Bluetooth
User
Intrusive Energy Profiling
Get indoors temperature
Get outdoors humidity
Check if it is day or night
Weather web service
Meter Indoors temperature
Get Arduino Sensors Readings
Meter Indoors humidity
Meter CO
Meter Total Energy Consumption
Arduino Board
Present Results
extension points
Recommend Energy Saving Actions
Recommend Energy Saving Actions
Intervene In Appliances

Figure 1: Use case diagram

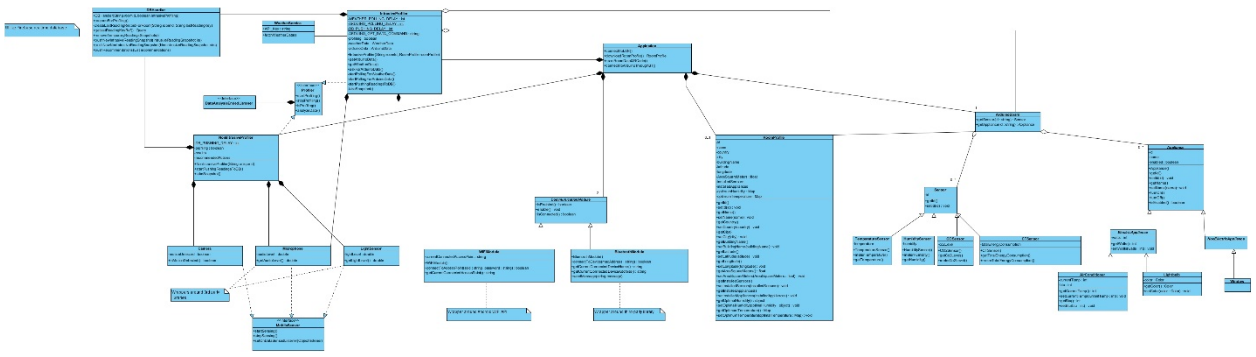


Figure 2: Class diagram

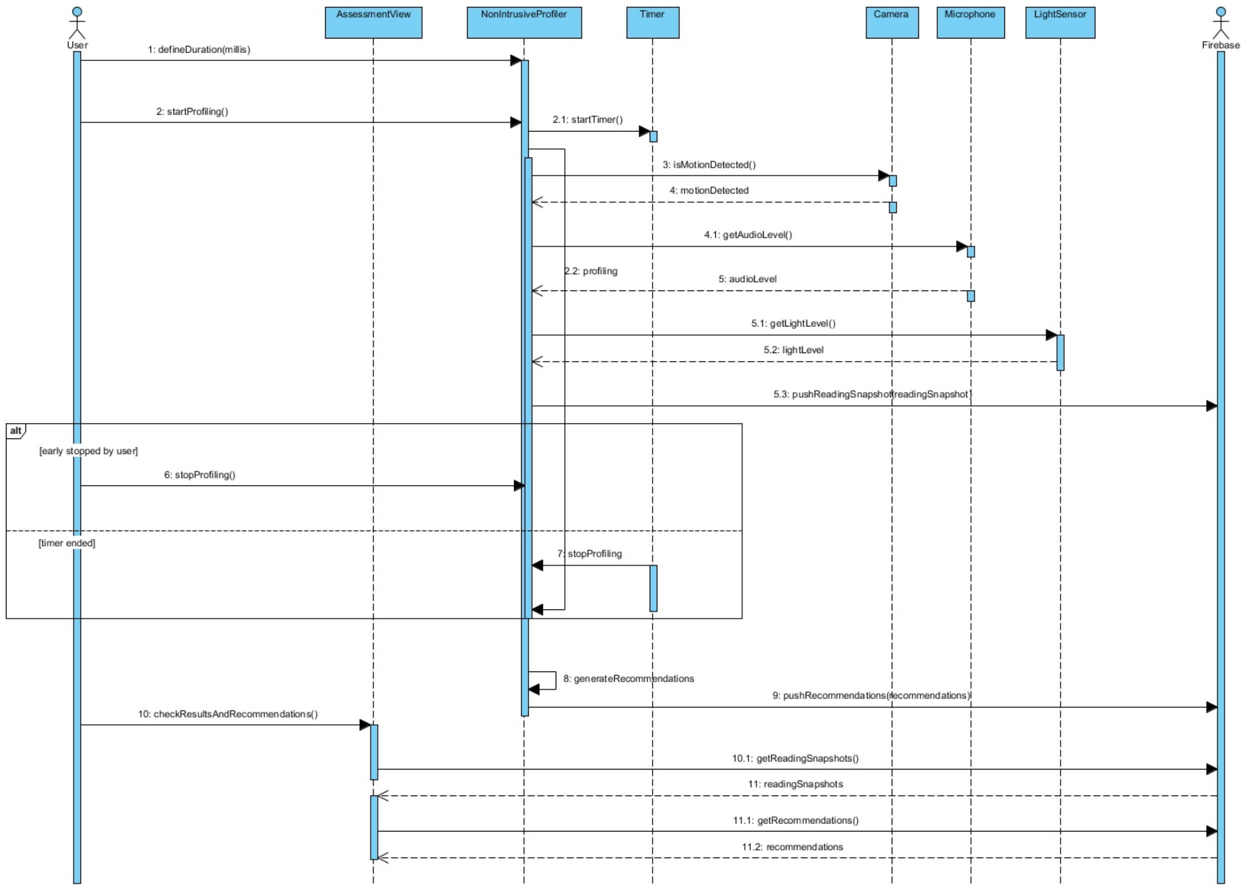


Figure 3: Non-Intrusive Profiling sequence diagram

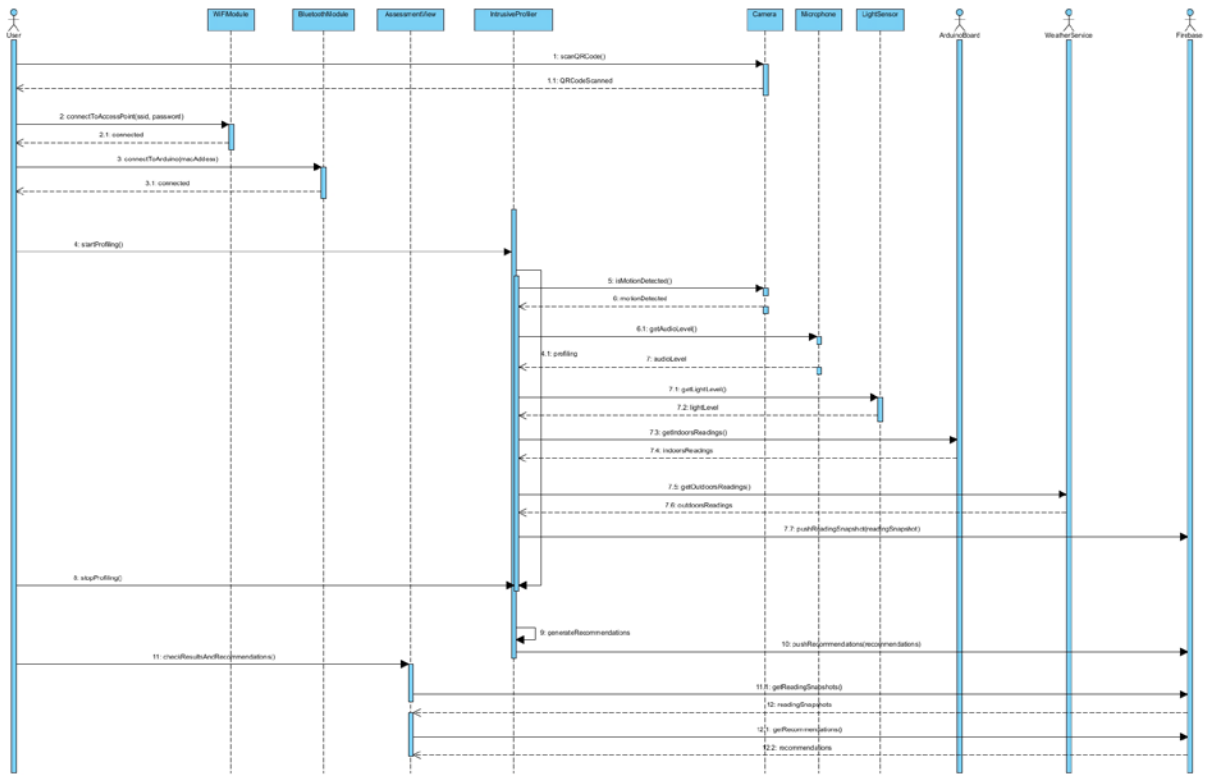


Figure 4: Intrusive Profiling sequence diagram

4 System Architecture

4.1 Architectural Styles

The system is based on a combination of several Architectural Styles:

- Client/Server
- Component-Based Architecture
- Object-Oriented

The system consists of four components: a) the mobile app, b) an Arduino Board, c) a Firebase database and d) APIXU (weather web service). Depending on the profiling mode chosen, they get utilized appropriately. The mobile app acts as a client that sends requests to the Arduino Board, to the Firebase database and to the APIXU weather service. These three components act as servers. Also, we use the component-based architecture style to decompose the design further into independent components that expose the appropriate communication interfaces. Finally, we use the object-oriented design approach for these components in order to improve reuse, testability, and flexibility.

4.2 High-Level Architecture of the System

In Figure 5, you can see a high-level architecture of the system while in Non-Intrusive Profiling mode. The app, collects data through the device's Light sensor, Microphone and Camera. Then, each reading is pushed to a firebase database. Here, an Internet connection is not required in order to send data to the database, as the app utilizes the offline capabilities of Firebase.

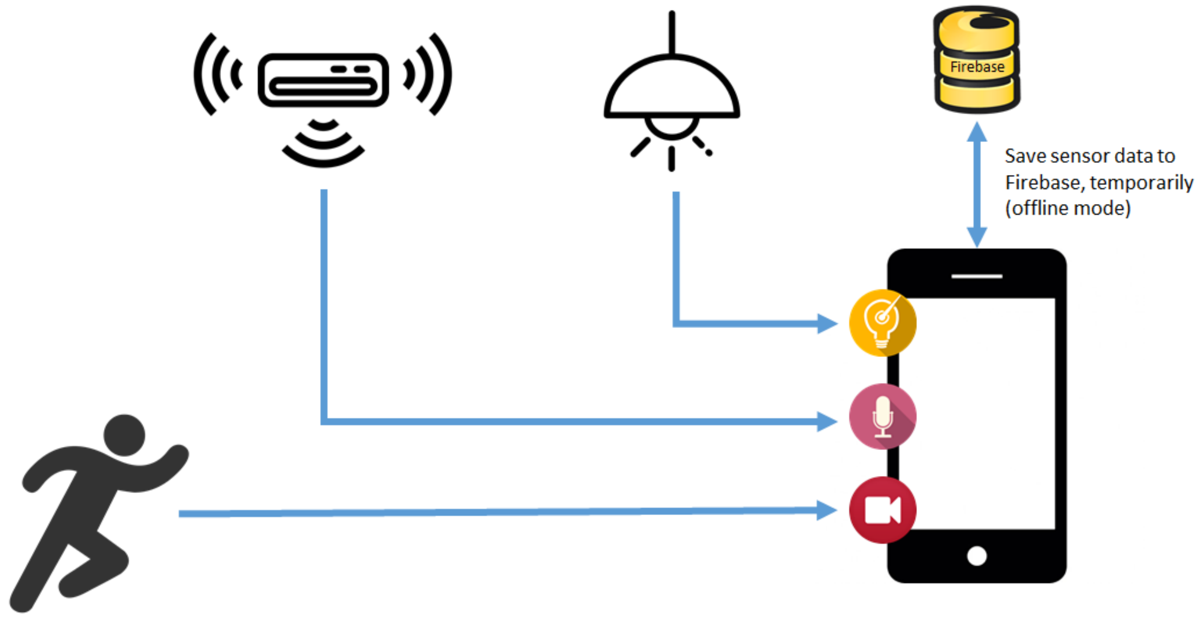


Figure 5: Non-Intrusive Profiling high level architecture

In Figure 6, a high-level architecture of the system while in Intrusive Profiling mode is shown, which is the most complex of the two. As its been clarified earlier, the Intrusive Profiling takes place in a Profiling-Ready room. Profiling-Ready means that an Arduino board, with some sensors and some interconnected appliances, has already been installed. So, a user scans a special QR code, containing some encoded information, that is inside the room he wants to profile, some initializations take place (connection to an access point, connection to Arduino through Bluetooth), and then chooses to begin the Profiling process. Immediately, data starts flowing in the app through a) the device's embedded sensors, b) the Arduino Sensors and c) the APIXU weather service. Then, all these readings are pushed online to a Firebase database which are stored for future use. Also, the user can intervene on the interconnected appliances: turn them on and off, change temperature, open or close windows, etc.

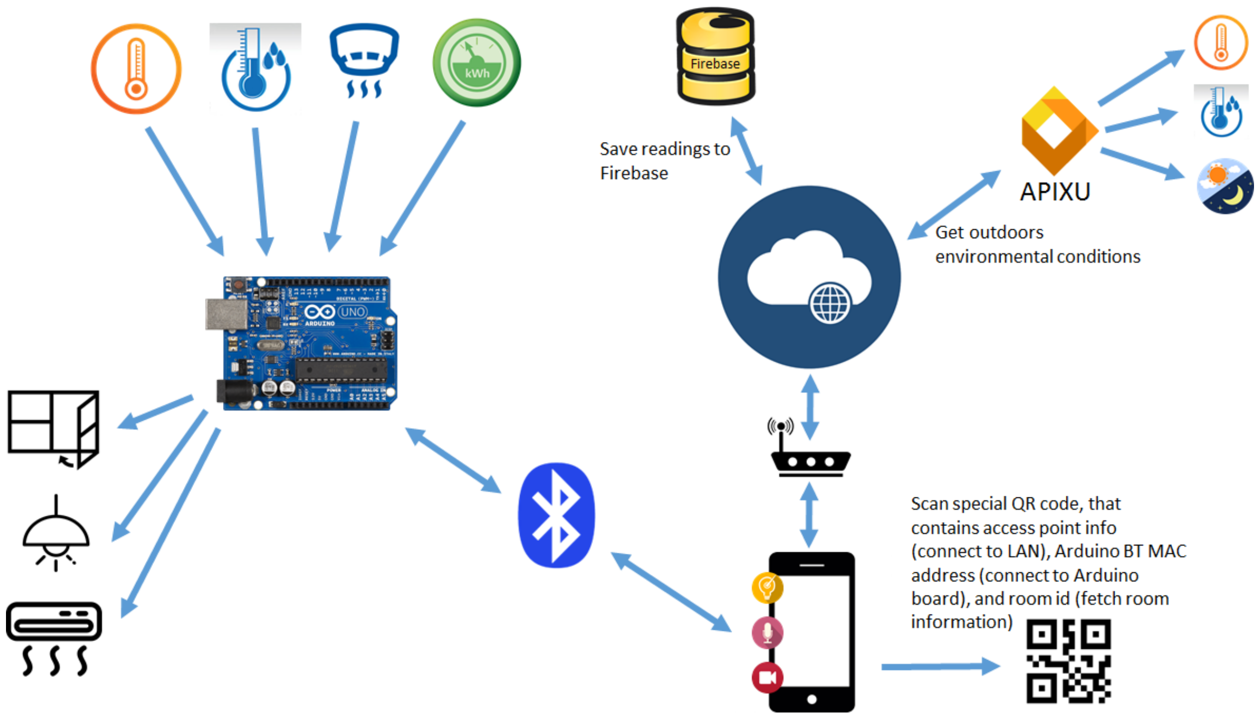

kWh
Firebase
Save readings to Firebase
APIXU
Get outdoors environmental conditions
Scan special QR code, that contains access point info (connect to LAN), Arduino BT MAC address (connect to Arduino board), and room id (fetch room information)

Figure 6: Intrusive Profiling high level architecture

5 UI Mockups

Having considered the user requirements, we designed a system that meets them. Now, we are going to present a set of UI mockups that let the user achieve his goal.

In the first screen the user is about to see, the app asks him to select a profiling mode (see Figure 1).

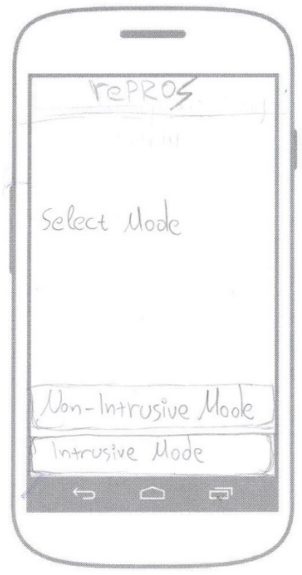


Figure 7: Profiling mode selection

To begin, let's select the Non-Intrusive Profiling mode. The next screen requires from the user to define the profiling duration and press the "Start Profiling" button in order to start the profiling process (see Figure 2).

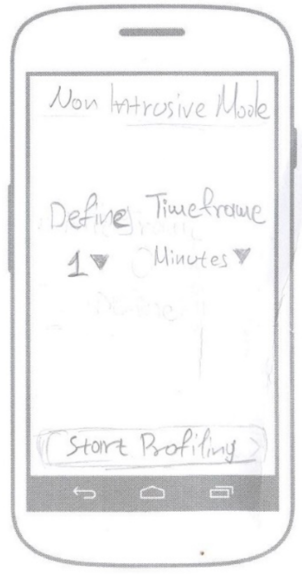


Figure 8: Profiling duration definition

Then, the profiling starts. Motion detection is represented by a moving-human or still-human icon (depending on the motion state), and audio and light level are presented in appropriate gauge-like charts, updating in real-time. Also, a timer shows the time left before the profiling is finished. The user has the ability to stop the profiling process before he timer ends, by pressing the "Early Stop" button (see Figure 3).

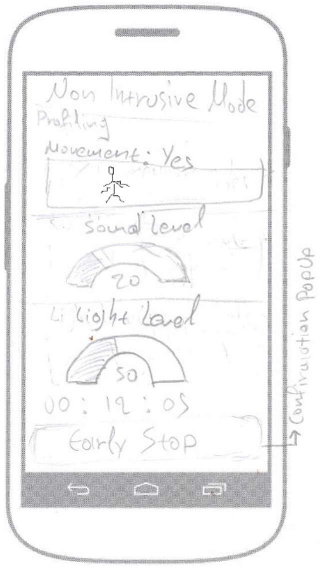


Figure 9: Non-Intrusive Profiling

When the profiling has ended, the Assessment screen is displayed, containing data collected and useful information generated during profiling. The user can click on "End Session" button, in order to complete the current profiling session and return back to mode selection screen (see Figure 4).

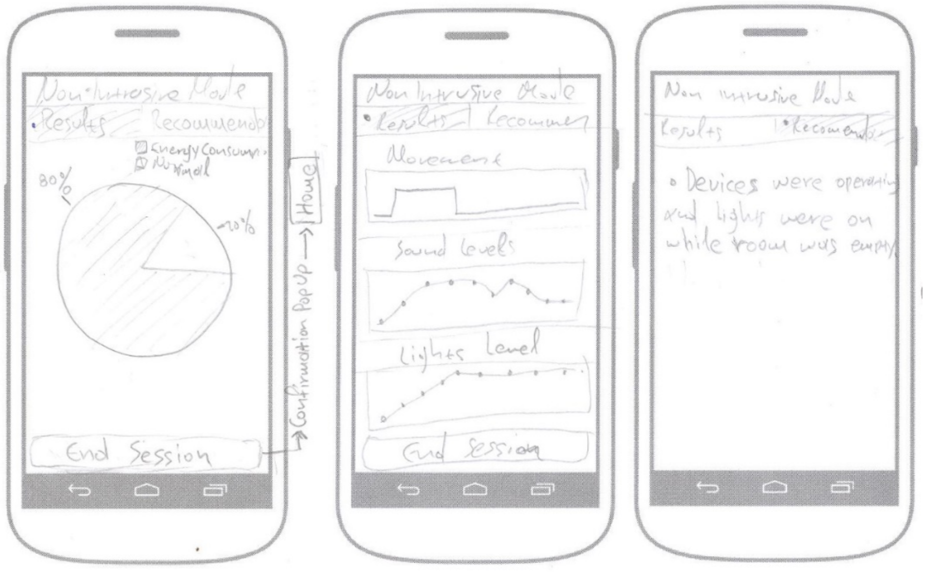


Figure 10: Non-Intrusive assessment

Now, let's continue with the Intrusive Profiling scenario. After the user presses the "Intrusive Mode" button (see Figure 1), the app informs the user that some initializations need to be done first, by scanning a QR code. The user presses the “Scan QR Code” button and initiates the camera that acts as a QR code scanner (see Figure 5).

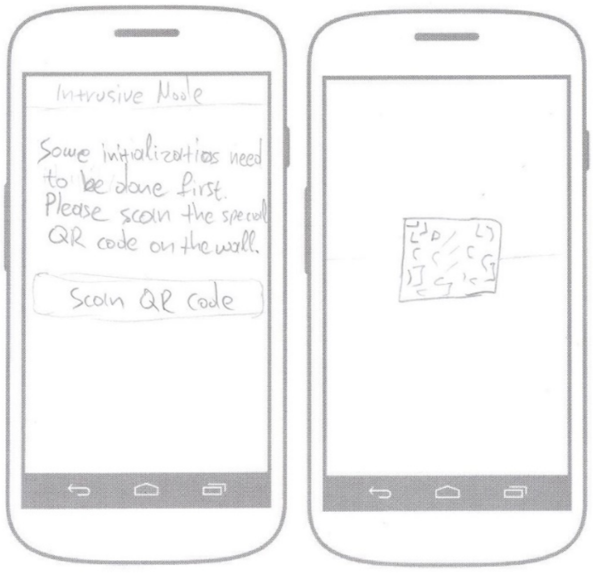

Intrusive Mode
Some initializations need to be done first. Please scan the special QR code on the wall.
Scan QR Code

Figure 11: QR Code scanning

After the scanning of the QR code, the app connects on the specified access point and Arduino. Also, some information about the room is fetched and displayed. Then, the user starts the profiling process by pressing the “Start Profiling” button (see Figure 6).

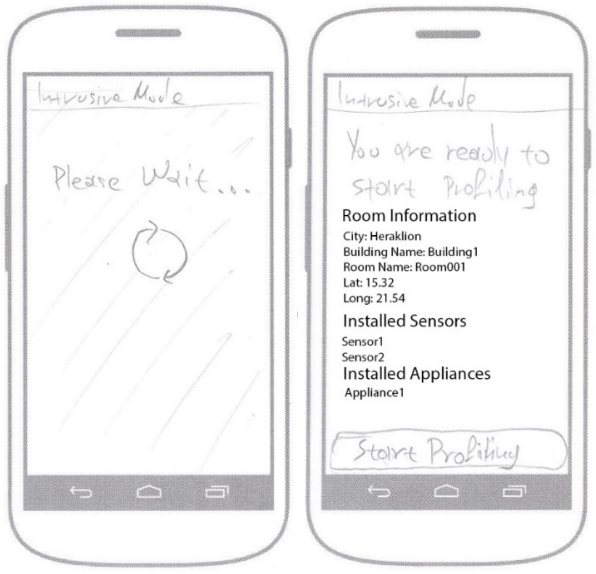


Figure 12: Initialization completed and room information

The profiler, collected data and displays it in appropriate gauge-like charts, updating in real-time. When the user feels that he is done with profiling, he presses the “Stop” button (see Figure 7).

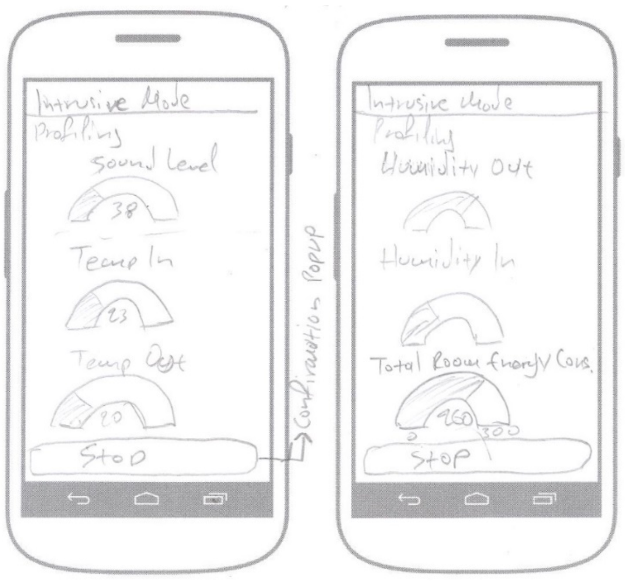


Figure 13: Intrusive Profiling

Then, the Assessment screen is displayed. Here, the user can have an overview of the previously collected readings, check energy saving recommendations generated by the app, and intervene in some appliances. When he is done, he presses the "End Session" button in order to return back to mode selection screen (see Figure 8).

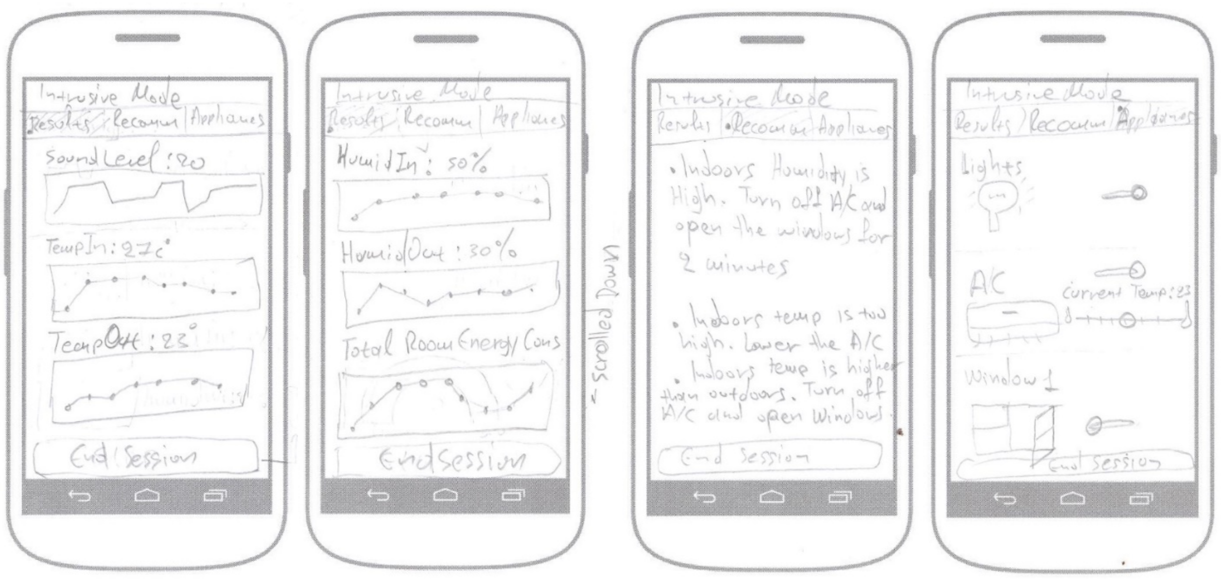

Intrusive Mode
Results | Recomm | Appliances
Sound Level : 20
Temp In : 27c°
Temp Out : 23°
End Session
Humid In : 50%
Humid Out : 30%
Total Room Energy Cons
End Session
scrolled Down
Indoors Humidity is High. Turn off A/C and open the windows for 2 minutes
Indoors temp is too high. Lower the A/C
Indoors temp is higher than outdoors. Turn off A/C and open Windows
End Session
Lights
A/C
Current Temp: 23
Window 1
End Session

Figure 14: Intrusive assessment

6 Pilot Development

This chapter demonstrates some insights of the system in the context of the two different profiling scenarios: Non-Intrusive and Intrusive. Also, some extracts of actual code will be provided.

Note: The classes Camera, Microphone and LightSensor are implemented as wrapper classes around third-party libraries. More specifically, Camera is a wrapper around a motion detection library whose GitHub repository can be found here. Microphone and LightSensor are wrappers around the SensingKit library whose GitHub repository can be found here.

6.1 Non-Intrusive Profiling Scenario

The app starts by asking the user to select a Profiling mode. For this scenario, the user taps on "Non-Intrusive" (see Figure 15).

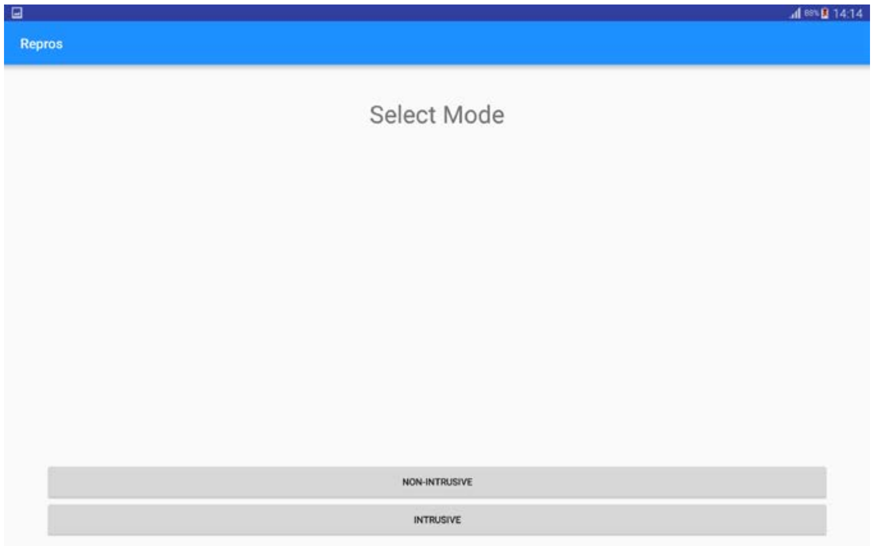


Figure 15: Start Screen

The Timeframe Definition screen appears, that asks from the user to define the duration of the Profiling to occur. User makes his choice and clicks on "Start Profiling" button (see Figure 16).

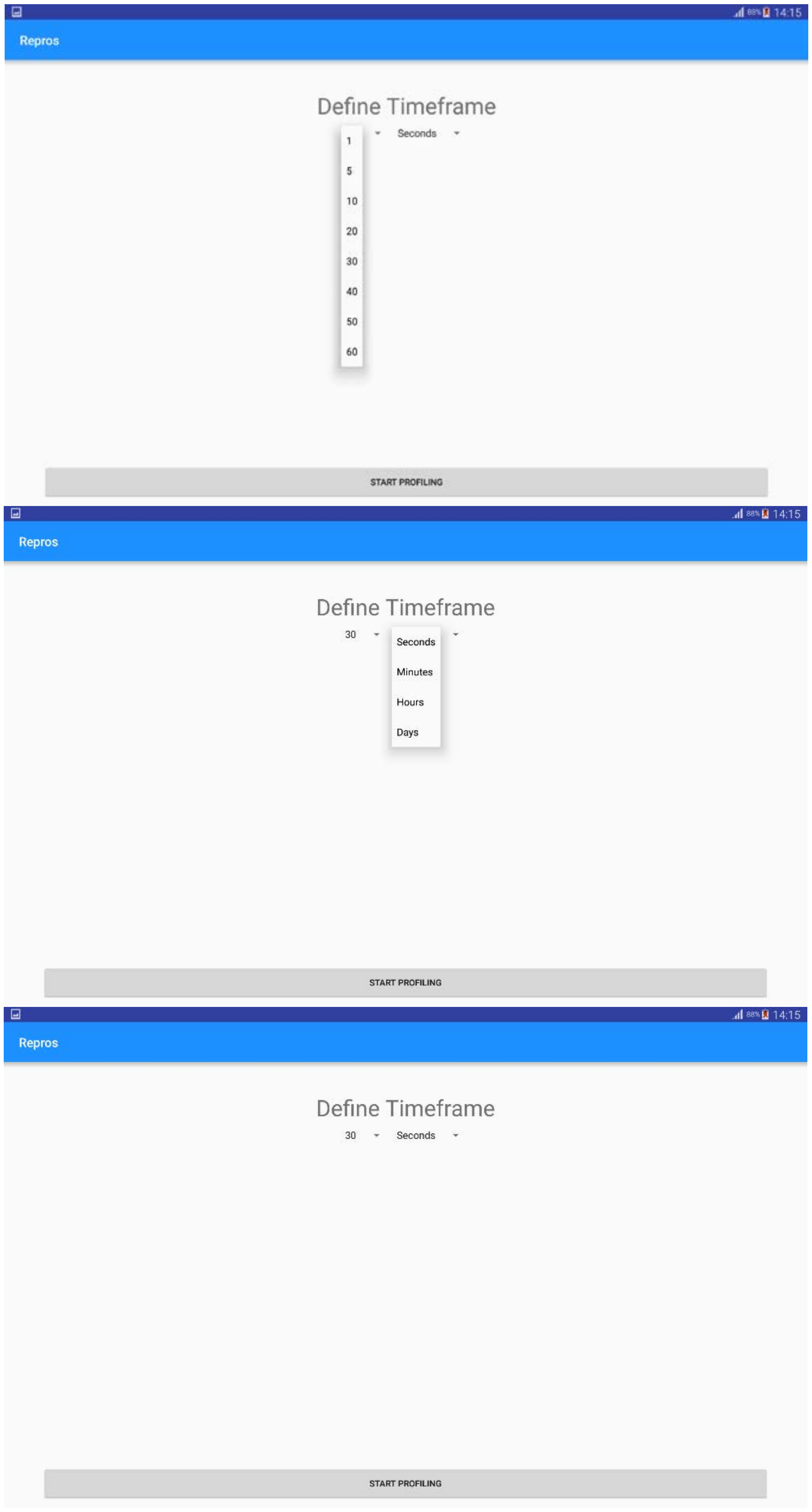


Figure 16: Timeframe Definition Screen

The Profiling process begins and a countdown timer shows the time remaining until the end of the Profiling. The Non-Intrusive Profiler collects data (**Snapshots**) from three input channels. Motion detection through camera, Audio level (Decibels) through microphone and Light level (Lux) through the light sensor. Also, a live "preview" of each measurement is displayed on screen. If camera detects any movement, the "still" human figure changes to a more appropriate one that depicts motion (see Figure 17).

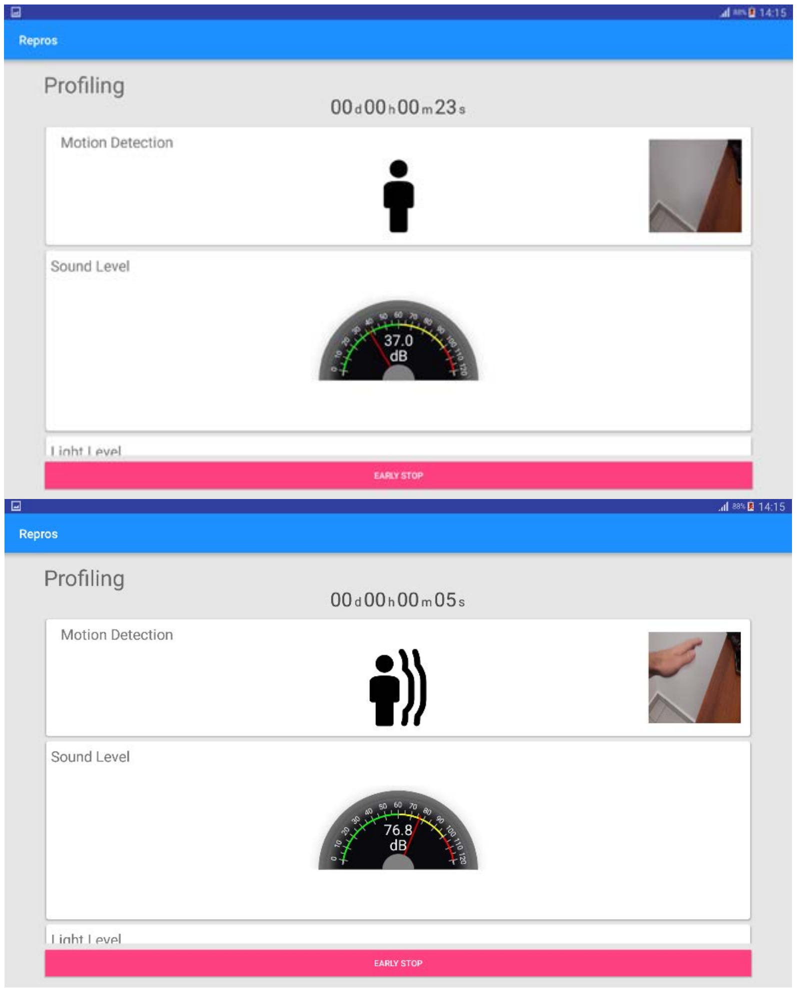
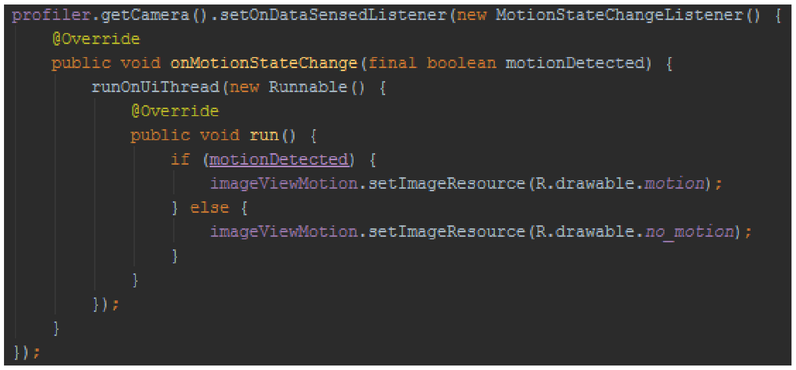


Figure 17: Non-Intrusive Profiling - Motion detection



```
profiler.getCamera().setOnDataSensedListener(new MotionStateChangeListener() {
    @Override
    public void onMotionStateChange(final boolean motionDetected) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (motionDetected) {
                    imageViewMotion.setImageResource(R.drawable.motion);
                } else {
                    imageViewMotion.setImageResource(R.drawable.no_motion);
                }
            }
        });
    }
});
```

Figure 18: Icon change on Motion Detection code extract

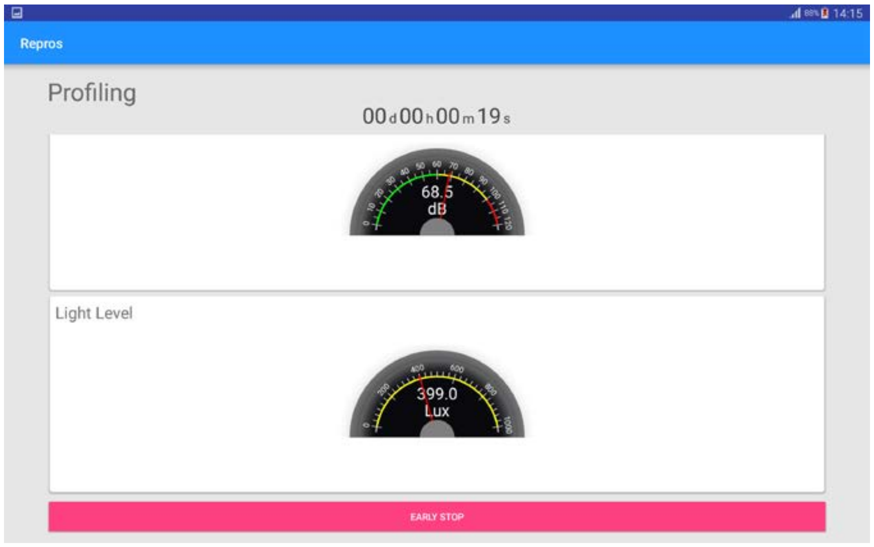
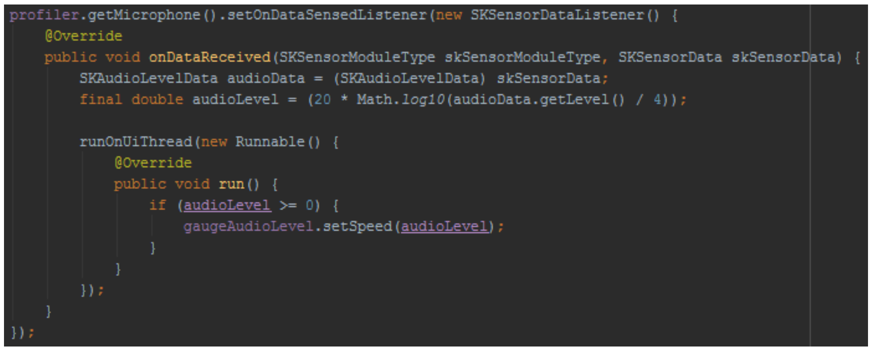
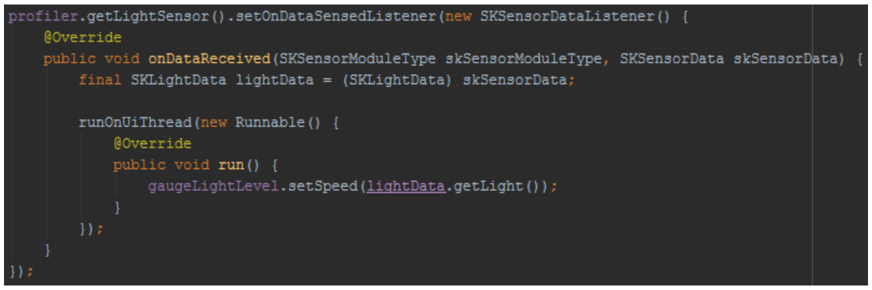


Figure 19: Non-Intrusive Profiling - Rest of measurements



```
profiler.getMicrophone().setOnDataSensedListener(new SKSensorDataListener() {
    @Override
    public void onDataReceived(SKSensorModuleType skSensorModuleType, SKSensorData skSensorData) {
        SKAudioLevelData audioData = (SKAudioLevelData) skSensorData;
        final double audioLevel = (20 * Math.log10(audioData.getLevel() / 4));

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (audioLevel >= 0) {
                    gaugeAudioLevel.setSpeed(audioLevel);
                }
            }
        });
    }
});
```

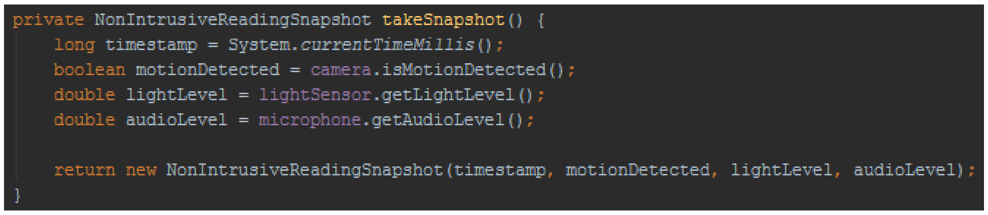
Figure 20: Audio level gauge update code extract



```
profiler.getLightSensor().setOnDataSensedListener(new SKSensorDataListener() {
    @Override
    public void onDataReceived(SKSensorModuleType skSensorModuleType, SKSensorData skSensorData) {
        final SKLightData lightData = (SKLightData) skSensorData;

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                gaugeLightLevel.setSpeed(lightData.getLight());
            }
        });
    }
});
```

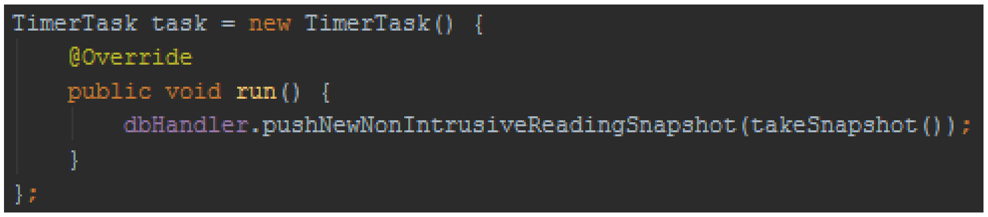
Figure 21: Light level gauge update code extract

After a predefined interval (5 seconds) the Profiler takes a snapshot (see Figure 22) from the input channels and pushes it on a Firebase database endpoint (see Figure 23).



```
private NonIntrusiveReadingSnapshot takeSnapshot() {
    long timestamp = System.currentTimeMillis();
    boolean motionDetected = camera.isMotionDetected();
    double lightLevel = lightSensor.getLightLevel();
    double audioLevel = microphone.getAudioLevel();

    return new NonIntrusiveReadingSnapshot(timestamp, motionDetected, lightLevel, audioLevel);
}
```

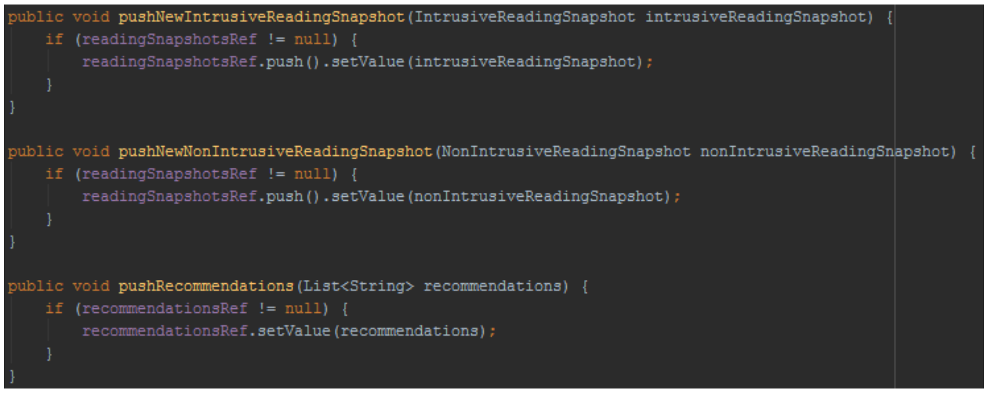
Figure 22: Non-Intrusive Profiler - TakeSnapshot method implementation



```
TimerTask task = new TimerTask() {
    @Override
    public void run() {
        dbHandler.pushNewNonIntrusiveReadingSnapshot(takeSnapshot());
    }
};
```

Figure 23: Non-Intrusive Profiler - Push snapshot to Firebase

To handle the transactions with the Firebase database, a database handler class is implemented that maps the app to specific database endpoints ("rooms", "readings", etc.) and also exposes three methods that lets the profiler push data snapshots and the generated recommendations to the database (see Figure 24).

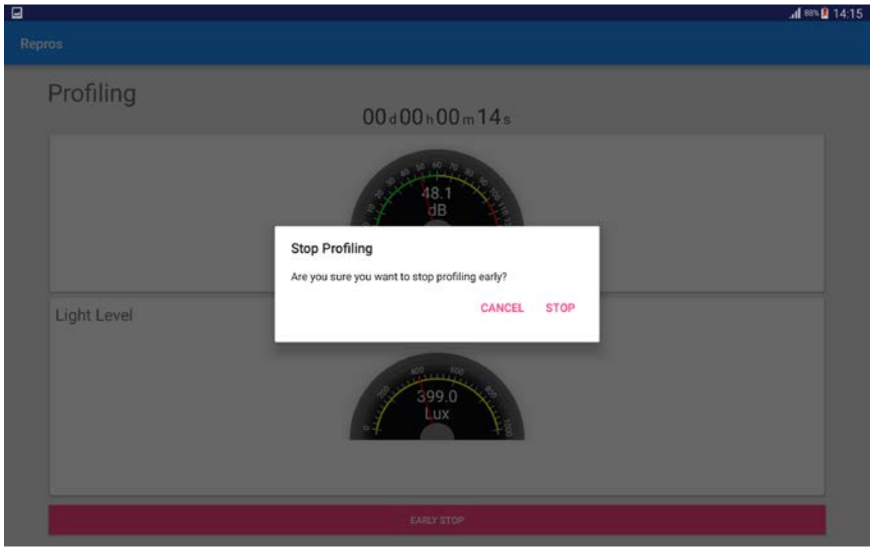


```
public void pushNewIntrusiveReadingSnapshot(IntrusiveReadingSnapshot intrusiveReadingSnapshot) {
    if (readingSnapshotsRef != null) {
        readingSnapshotsRef.push().setValue(intrusiveReadingSnapshot);
    }
}

public void pushNewNonIntrusiveReadingSnapshot(NonIntrusiveReadingSnapshot nonIntrusiveReadingSnapshot) {
    if (readingSnapshotsRef != null) {
        readingSnapshotsRef.push().setValue(nonIntrusiveReadingSnapshot);
    }
}

public void pushRecommendations(List<String> recommendations) {
    if (recommendationsRef != null) {
        recommendationsRef.setValue(recommendations);
    }
}
```

Figure 24: Methods to push data snapshots and recommendations to Firebase

Although, an Internet connection is not required for this scenario, the code works just fine as we utilize the offline capabilities of Firebase.

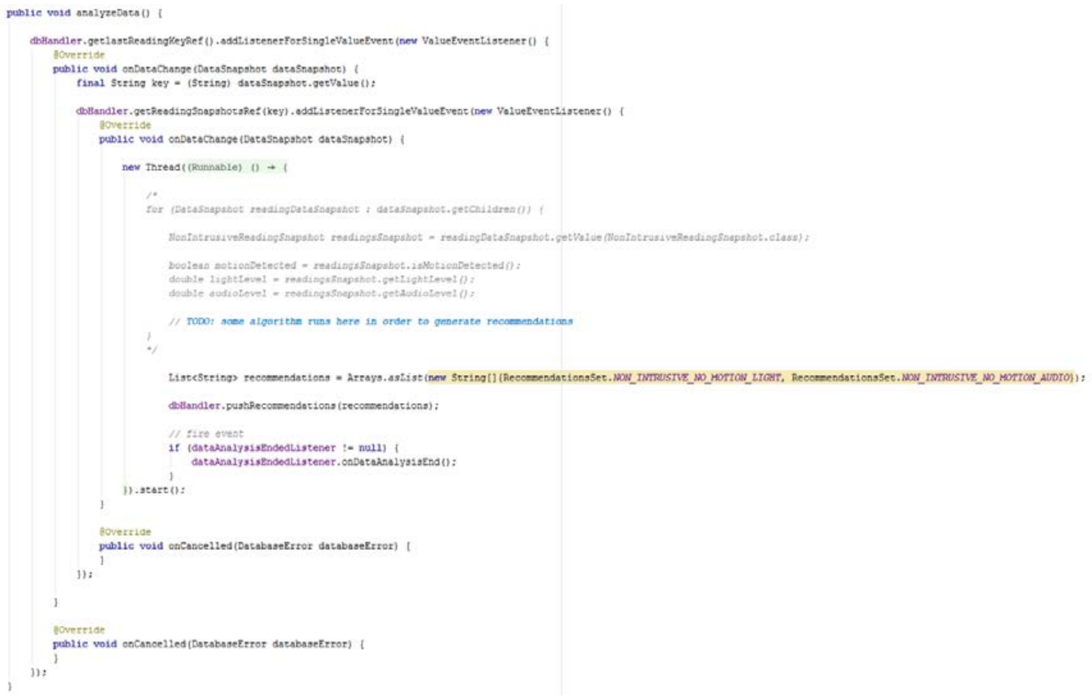
The Profiling process can either get finished naturally, by the timer reaching zero, or get interrupted by the user and stopped earlier (see Figure 25).


Stop Profiling

Are you sure you want to stop profiling early?

Figure 25: Non-Intrusive Profiling - Early stop by user dialog

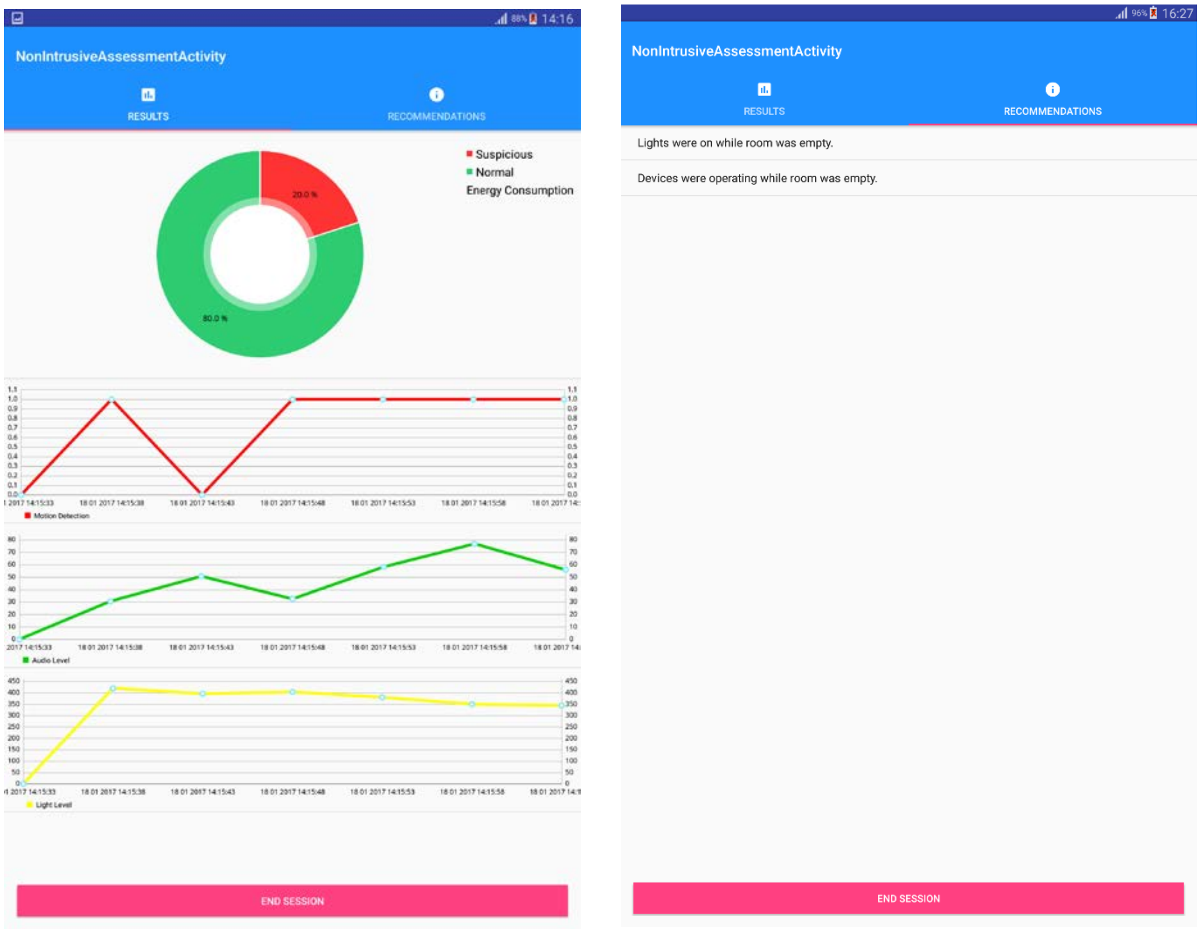
Whenever the Profiling stops, the system analyzes the data and generates some energy saving recommendations (see Figure 26). In the context of this project, the data analysis algorithm is not implemented.



```java
public void analyzeData() {

    dbHandler.getlastReadingKeyRef().addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            final String key = (String) dataSnapshot.getValue();

            dbHandler.getReadingSnapshotsRef(key).addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {

                    new Thread((Runnable) () -> {

                        /*
                        for (DataSnapshot readingDataSnapshot : dataSnapshot.getChildren()) {

                            NonIntrusiveReadingSnapshot readingsSnapshot = readingDataSnapshot.getValue(NonIntrusiveReadingSnapshot.class);

                            boolean motionDetected = readingsSnapshot.isMotionDetected();
                            double lightLevel = readingsSnapshot.getLightLevel();
                            double audioLevel = readingsSnapshot.getAudioLevel();

                            // TODO: some algorithm runs here in order to generate recommendations
                        }
                        */

                            List<String> recommendations = Arrays.asList(new String[]{RecommendationsSet.NON_INTRUSIVE_NO_MOTION_LIGHT, RecommendationsSet.NON_INTRUSIVE_NO_MOTION_AUDIO});

                            dbHandler.pushRecommendations(recommendations);

                            // fire event
                            if (dataAnalysisEndedListener != null) {
                                dataAnalysisEndedListener.onDataAnalysisEnd();
                            }
                    }).start();
                }

                @Override
                public void onCancelled(DatabaseError databaseError) {
                }
            });

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
        }
    });
}
```

Figure 26: Non-Intrusive Profiling - AnalyzeData method

The Assessment screen follows. Here, the user receives an overview of the collected data in terms of charts, e.g. pies and line-charts, and the list of recommendations (see Figure 27). From there, when the user is finished with the results and recommendations, he presses the “End Session” button in order to return back to the Start Screen (see Figure 15).



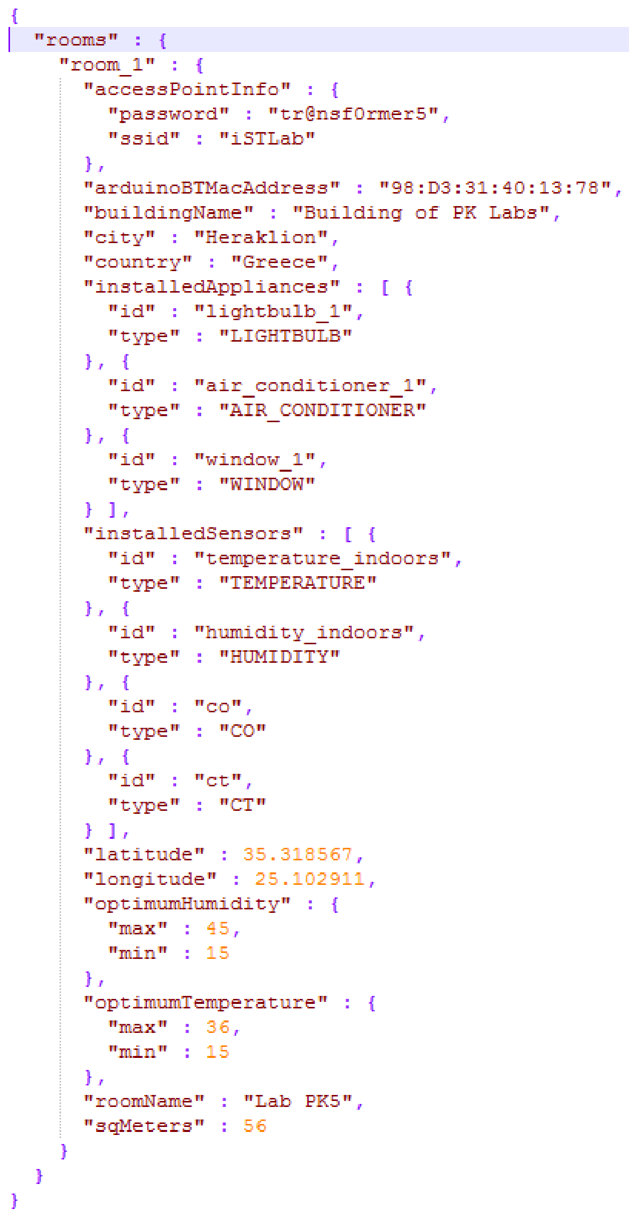
Lights were on while room was empty.

Devices were operating while room was empty.

Figure 27: Non-Intrusive Profiling - Assessment Screen

6.2 Intrusive Profiling Scenario

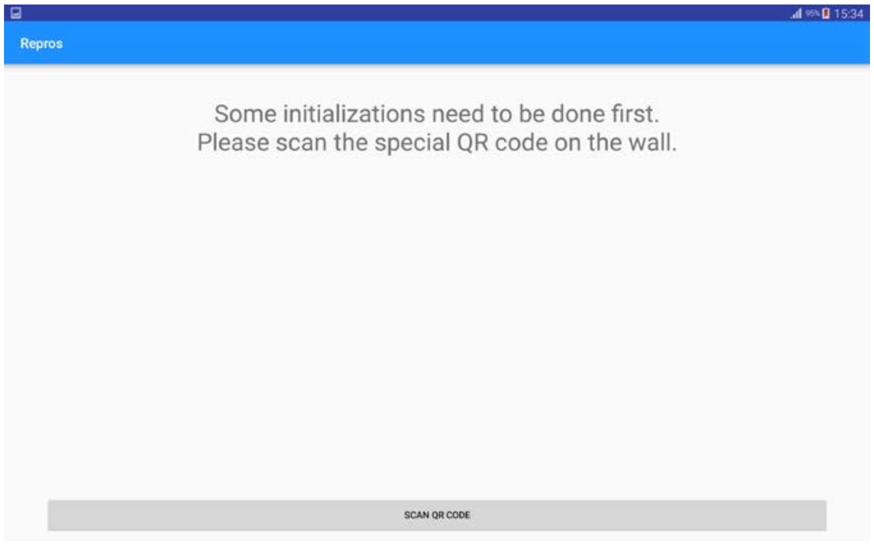
For this scenario, we are going to intrusively profile a room. This room is Profile-Ready, which means that: a) an Arduino Board in already installed and waits for a smart device running our special software to connect upon, and b) the Room Profile of this room is saved on the Firebase database (see Figure 28).



```
{
  "rooms" : {
    "room_1" : {
      "accessPointInfo" : {
        "password" : "tr@nsf0rmer5",
        "ssid" : "iSTLab"
      },
      "arduinoBTMacAddress" : "98:D3:31:40:13:78",
      "buildingName" : "Building of PK Labs",
      "city" : "Heraklion",
      "country" : "Greece",
      "installedAppliances" : [ {
        "id" : "lightbulb_1",
        "type" : "LIGHTBULB"
      }, {
        "id" : "air_conditioner_1",
        "type" : "AIR_CONDITIONER"
      }, {
        "id" : "window_1",
        "type" : "WINDOW"
      } ],
      "installedSensors" : [ {
        "id" : "temperature_indoors",
        "type" : "TEMPERATURE"
      }, {
        "id" : "humidity_indoors",
        "type" : "HUMIDITY"
      }, {
        "id" : "co",
        "type" : "CO"
      }, {
        "id" : "ct",
        "type" : "CT"
      } ],
      "latitude" : 35.318567,
      "longitude" : 25.102911,
      "optimumHumidity" : {
        "max" : 45,
        "min" : 15
      },
      "optimumTemperature" : {
        "max" : 36,
        "min" : 15
      },
      "roomName" : "Lab PK5",
      "sqMeters" : 56
    }
  }
}
```

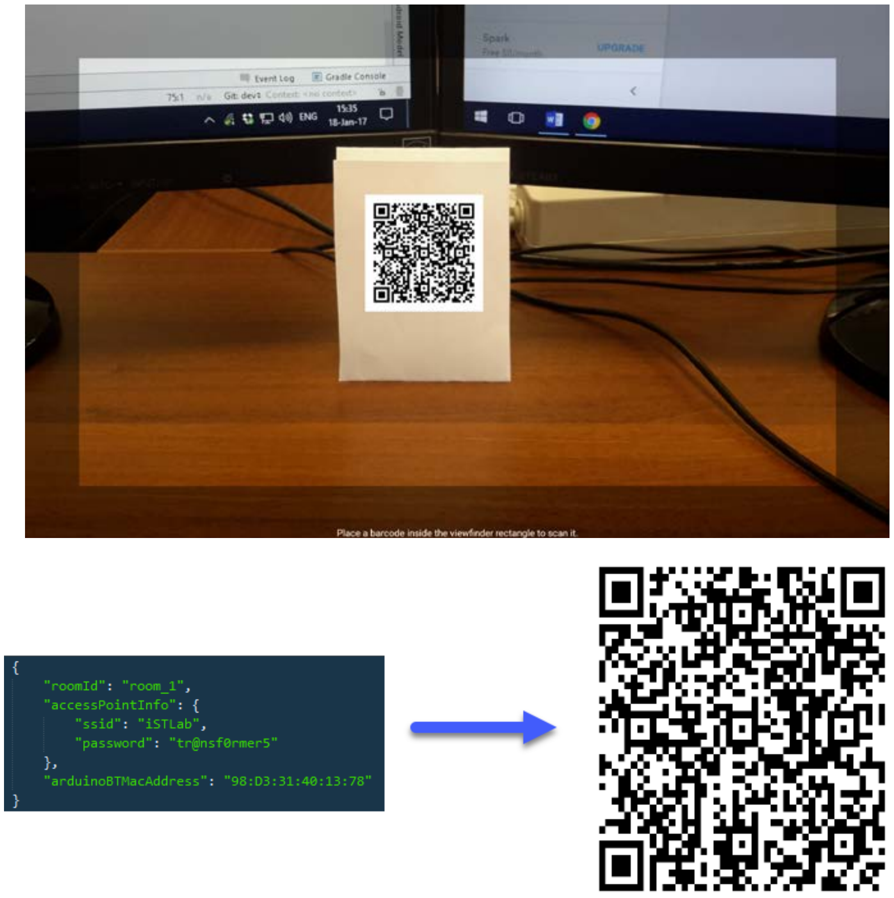
Figure 28: Firebase Rooms endpoint

To begin with, we tap on “Intrusive” button on the Start Screen (see Figure 15). The app informs the user about some initializations that need to take place and prompts him to scan the special QR code that is (somewhere) in the room (see Figure 29). These initializations are a) the connection of the smart device to an access point in order to obtain Internet access and b) the connection of the smart device to the Arduino Board though Bluetooth. The information needed in order to achieve the connections are encoded to the QR Code (see Figure 30).



Some initializations need to be done first.
Please scan the special QR code on the wall.

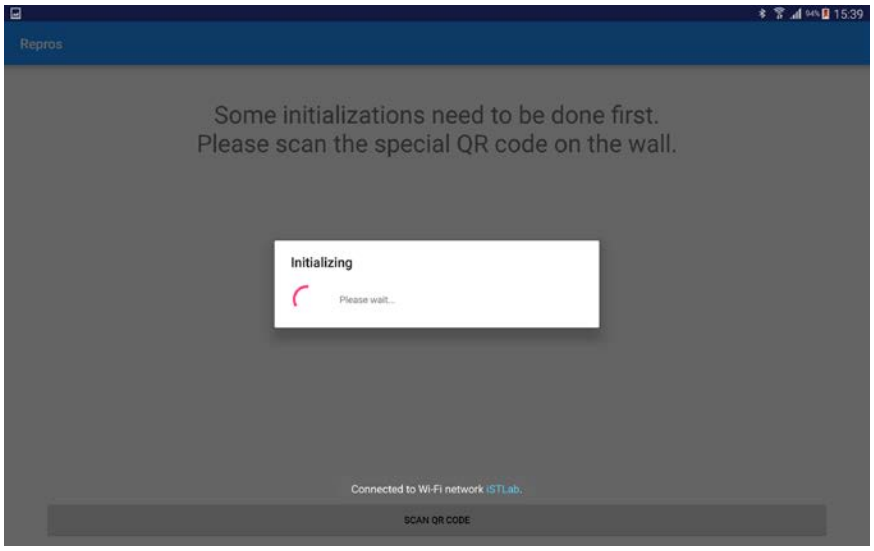
Figure 29: Initializations Prompt



```
{
    "roomId": "room_1",
    "accessPointInfo": {
        "ssid": "iSTLab",
        "password": "tr@nsf0rmer5"
    },
    "arduinoBTMacAddress": "98:D3:31:40:13:78"
}
```

Figure 30: QR Code scanning

If the scan is successful, a popup message appears and informs the user that the initializations take place and asks for patience (see Figure 31). The app automatically connects to the access point and to Arduino.



Some initializations need to be done first.
Please scan the special QR code on the wall.

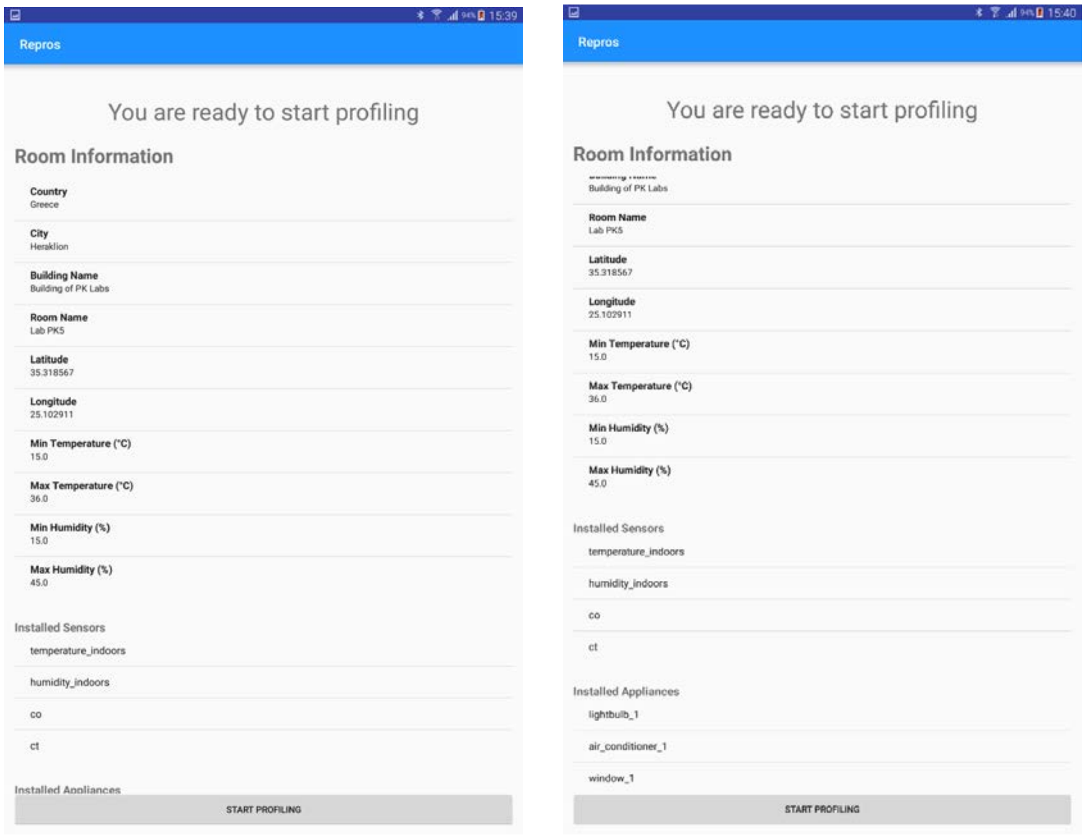
Initializing

Please wait...

Connected to Wi-Fi network iSTLab.

Figure 31: Initializing - Popup dialog

When the initializations are completed, the app displays the information relative to the room (see Figure 32).



You are ready to start profiling

Room Information

Country	Greece
City	Heraklion
Building Name	Building of PK Labs
Room Name	Lab PK5
Latitude	35.318567
Longitude	25.102911
Min Temperature (°C)	15.0
Max Temperature (°C)	36.0
Min Humidity (%)	15.0
Max Humidity (%)	45.0

Installed Sensors

temperature_indoors
humidity_indoors
co
ct

Installed Appliances

You are ready to start profiling

Room Information

	Building of PK Labs
Room Name	Lab PK5
Latitude	35.318567
Longitude	25.102911
Min Temperature (°C)	15.0
Max Temperature (°C)	36.0
Min Humidity (%)	15.0
Max Humidity (%)	45.0

Installed Sensors

temperature_indoors
humidity_indoors
co
ct

Installed Appliances

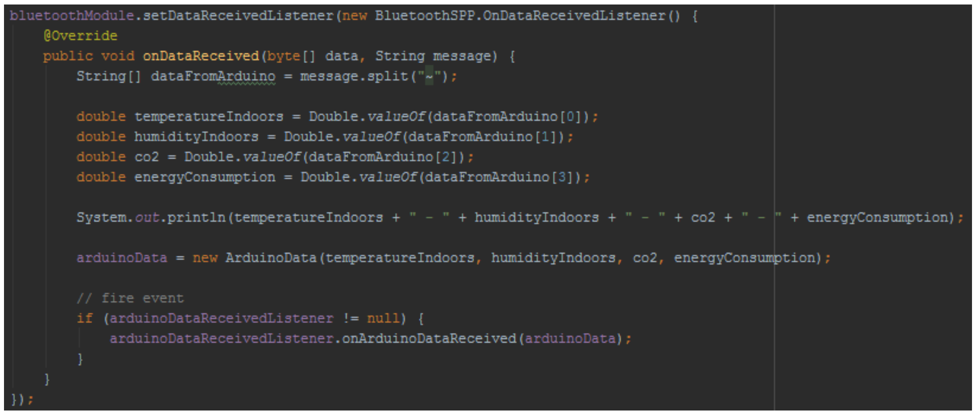
lightbulb_1
air_conditioner_1
window_1

Figure 32: Room information

Then, the user starts the profiling process by tapping on the “Start Profiling” button. The Intrusive Profiler, every 5 seconds, collects data from:

a) The Microphone and Light sensor of the device (same as Non-Intrusive Profiling)
b) The Arduino Board
    - Indoors temperature (Celsius)
    - Indoors humidity (percentage)
    - Carbon monoxide level (ppm)
    - Total energy consumption of room (watts)
c) The APIXU Weather web service
    - Outdoors temperature (Celsius)
    - Outdoors humidity (percentage)

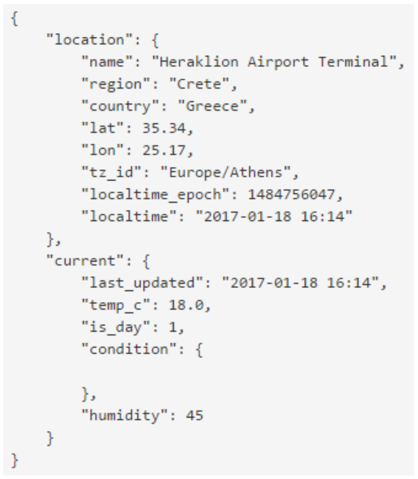
The Bluetooth message, containing the above readings, the app receives from the Arduino has this form: “**22~32~2~3200**”. The first number is temperature, the second is humidity, the third is CO ppm and the fourth is the total energy consumption of the room. The app is responsible for “breaking down” this message and get each measurement, separately (see Figure 33).



```java
bluetoothModule.setDataReceivedListener(new BluetoothSPP.OnDataReceivedListener() {
    @Override
    public void onDataReceived(byte[] data, String message) {
        String[] dataFromArduino = message.split("~");

        double temperatureIndoors = Double.valueOf(dataFromArduino[0]);
        double humidityIndoors = Double.valueOf(dataFromArduino[1]);
        double co2 = Double.valueOf(dataFromArduino[2]);
        double energyConsumption = Double.valueOf(dataFromArduino[3]);

        System.out.println(temperatureIndoors + " - " + humidityIndoors + " - " + co2 + " - " + energyConsumption);

        arduinoData = new ArduinoData(temperatureIndoors, humidityIndoors, co2, energyConsumption);

        // fire event
        if (arduinoDataReceivedListener != null) {
            arduinoDataReceivedListener.onArduinoDataReceived(arduinoData);
        }
    }
});
```

Figure 33: Breaking down of Arduino message

Also, the app requests “outdoors” conditions from the APIXU weather web service. The response is a JSON object of the form shown in Figure 34.



```json
{
    "location": {
        "name": "Heraklion Airport Terminal",
        "region": "Crete",
        "country": "Greece",
        "lat": 35.34,
        "lon": 25.17,
        "tz_id": "Europe/Athens",
        "localtime_epoch": 1484756047,
        "localtime": "2017-01-18 16:14"
    },
    "current": {
        "last_updated": "2017-01-18 16:14",
        "temp_c": 18.0,
        "is_day": 1,
        "condition": {

        },
        "humidity": 45
    }
}
```

Figure 34: APIXU Weather web service response

The data is lively updated on screen as long as the profiling takes place (see Figure 35). Also, every data snapshot is pushed and stored to the Firebase database.

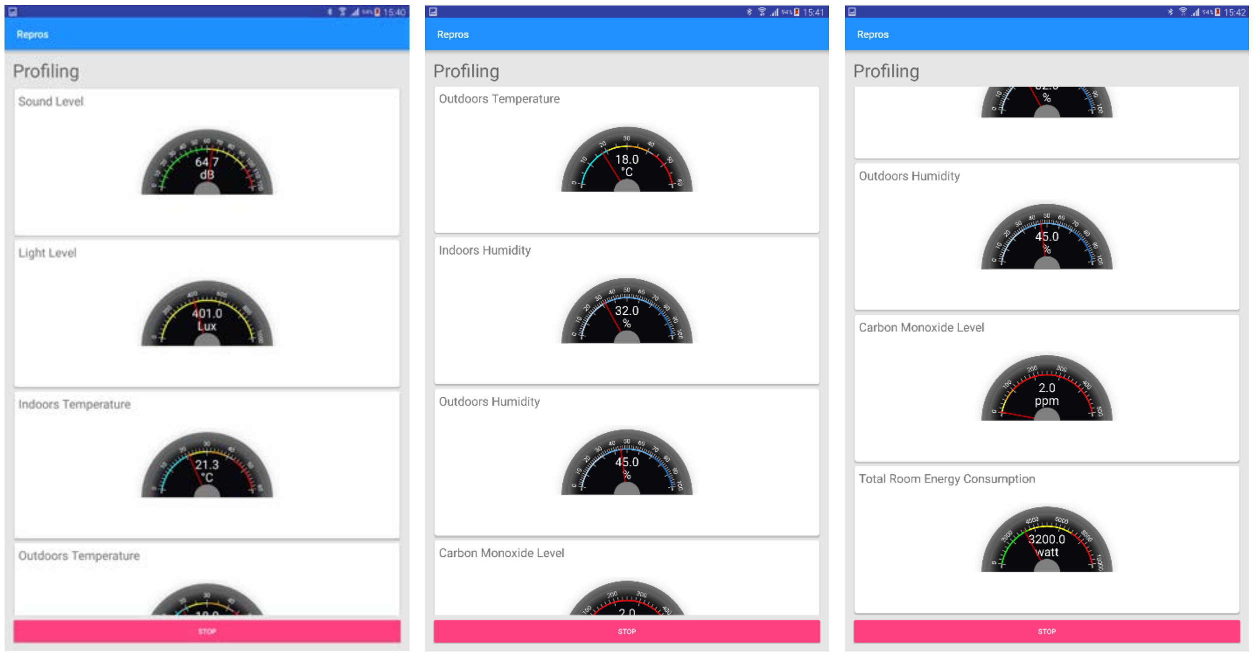
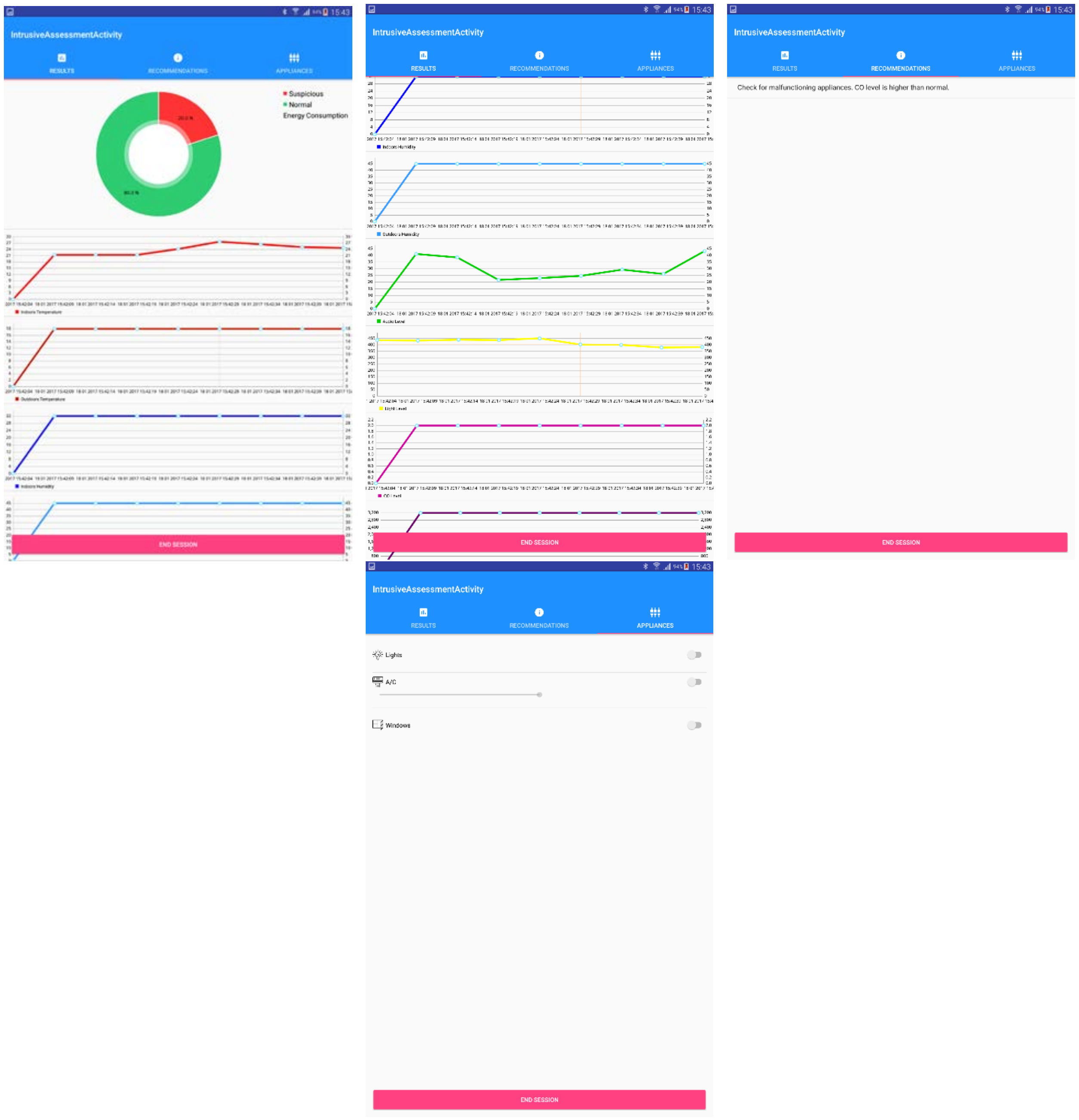


Figure 35: Intrusive Profiling Screen

When the user presses the “Stop” button in order to stop the profiling process, an analysis of the data is performed and some recommendations are generated. and pushed to the database. Then, the user presses the “Display Results” button to get to the (Intrusive) Assessment Screen (see Figure 36). The functionality here is similar to the Non-Intrusive Assessment Screen. In addition, an extra tab with title “Appliances” is added. From this tab, the user has access to the interconnected appliances and can intervene on them, turn them on/off, change temperature, etc. When the user has completed the assessment of data and intervention, he presses the “End Session” button in order to return to the Start Screen.

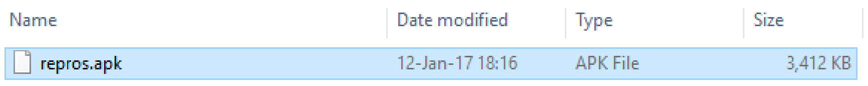


Check for malfunctioning appliances. CO level is higher than normal.

Figure 36: Intrusive Profiling - Assessment Screen

6.3 Installation

After the build process, an "apk" file is generated. This very same file, is the one that gets installed on the user's smartphone or tablet.



Name	Date modified	Type	Size
repros.apk	12-Jan-17 18:16	APK File	3,412 KB

Figure 37: Binary apk file