Companion Light - Final Project - KSE624

Valentin Lievin
DTU student
s152076
valentin.lievin@gmail.com

ABSTRACT

The project demonstrates how to use an android phone and a BLE Arduino module to transform a simple RGB lamp into a context-aware object. The goals were to adjust the light to the user's position in the room, to the ambient luminosity and to the sleep cycles. These goals have been reached using the phone's sensors and the strength of the BLE connection.

Author Keywords
CompanionLight, BLE, Android, Blend Micro, Context Aware Computing

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

"Researchers have shown in humans that light influences hormone secretion, heart rate, alertness, sleep propensity, body temperature, and gene expression"[2]. In other words, our lighting environment is very important for our well being because, for example, it affects our ability to focus and our sleep cycles. Moreover, we observe that the lighting environment is a key features for a lot of businesses that sell experiences. For example, the light is a corner stone of the success of a concert. We can also observe that the light in bars and clubs is meticulously adjusted (color and intensity) to create a particular ambiance. Furthermore we observe that these differences can make these places feel totally different. Therefore controlling the environment could have a significant effect on the user's mood and well being.

On the other hand, while most of the technology required for the IoT is ready, the population does not benefit yet from this new technology.

This project aims to use build a context-aware systems that adjusts lighting to the surrounding environment including the user's position.

CONTEXT OF EXPERIMENTATION

This project has been tailored to the needs of people living in a single room apartment, which fits generally well the needs of most students. We consider that the room contains a desk and a bed. The light has to be placed on the desk.

We consider only two different activities: working and relaxing. Moreover, we consider that the user wants to turn off the light only when he's leaving the room.

Moreover, we simplify the situation by considering that if the user is sitting at his desk then he's working. On the other hand, if the user is sitting or lying on his bed, he's relaxing. This choice has been motivated by the fact that these conditions reflect my habits and that I'm the only person testing the prototype.

OBJECTIVES

The project aims to build a smart ambient desk lamp that reacts to the user's activities and improve his daily-life. This goal is separated in 4 goals described in the following sections.

Adapt the light to the sleep cycles

The waves with highest frequencies of the visible spectrum of the light have an effect on our sleep cycles[2]. More precisely, reducing the blue component of the light could help the user to improve his sleep-cycle. Thus, a first objective is to reduce the blue component at the end of the day.

Adapt the intensity of the light to the environment's light intensity

An ambient light needs to be adjusted to the surrounding lightning context to provide the best experience. indeed, a light too powerful would be uncomfortable for the user who aims to have only an ambient light.

Adapt the color of the light depending on the user's position

The objective is to adapt the light to the two different activities considered in this project: working and relaxing. The goal is to identify these two activities characterized by the position of the user in the room as specified section. More precisely, we expect that the user needs a stronger and neutral light while working. When relaxing, we propose to the user a relaxing mode that consists of a light with a variable color.

Turn on and off the light when the user leaves the room

For energy saving purpose, we turn off the light when the user leaves the room.

SOLUTION

In short, the solution is an RBG LED strip controlled by with an Arduino board with a BLE module (Blend Micro) controlled by an android app that uses sensors and the BLE connection strength to adjust the light.

Blend Micro

Hardware

- Android phone (OnePlus One)
- RGB LED strip
- power Adaptater 1A
- 3 MOSFET
- Redbear Lab Blend Micro

Arduino: System Architecture

The LED strip is controlled by 3 inputs corresponding to the 3 raw colors (red,green,blue). However, the RGB LED strip requires more power that what the Blend Micro can deliver. That's why we cannot connect directly the LED strip with the output of the Blend Micro. Instead, 3 MOSFET are used to control a more powerful electric current provided by the power adapter. The wiring diagram is shown fig.1.

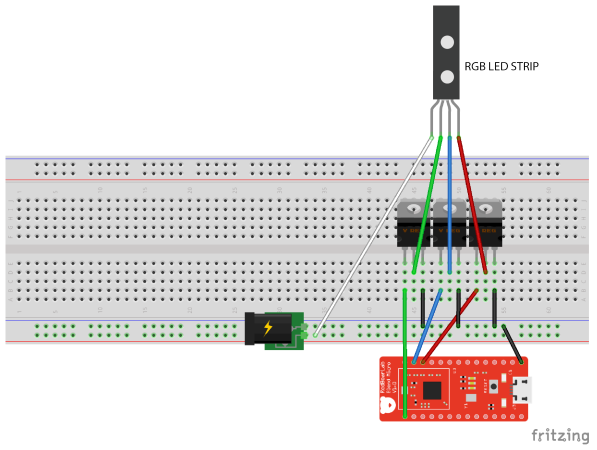


Figure 1. Control the LED strip with a Blend Micro

Android

Overview of the Android App

The Android app enables the user to switch between a manual mode that enables the user to control directly the hue and the intensity of the LEDs and an automatic mode that switch automatically between 3 states: working mode, relaxing mode and sleep mode.

The app has been created from the app ”SimpleControls” [5] provided by RedBear Lab as an example of BLE app.

The core of the app is a foreground service (CompanionService) provided with a static notification with contextual actions. This service enables the app to be running in the background while the phone is locked, which is not possible with an activity. Moreover it also enables the user to perform quick action directly from the notification center which is really convenient. This service controls the BLE service en gives the messages to the BLE service which send them to the Blend Micro.

The BLE service, which is dedicated to the management of the BLE connection, is bound to main service. Once the BLE service is connected to the BlendMicro, the service send the value of the strength of the signal (RSSI) continuously to the main service thanks to an intent.

The app presents also an activity which enables the user to start and stop the main service and to set the parameters of the light when using the manual mode. This activity can be stopped independently of the main service.

The Architecture of the app is shown fig.2

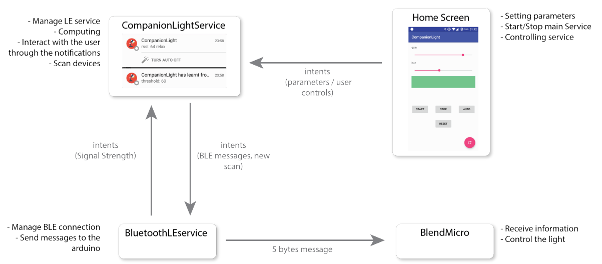


Figure 2. How the app works

Controlling the Blend Micro

The BLE service is managed by a service bound to the main service, the source code is provided by Rebear Lab and has not been modified.

The Android app controls the Blend Micro with 5-bytes-messages. The RGB values are encoded on the 3 first bytes, the 4th byte is used to encode the intensity of the light and the last byte is dedicated to indicate the state of the current mode: working, relaxing and sleep mode.

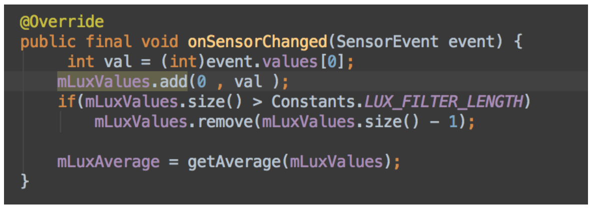
While the light is in Working mode, the Arduino uses the values received from the Android app. Otherwise, the light is automatically changing the color of the light over the time.

Adjusting the intensity

In order to provide the best experience, it is important to adjust the intensity of the light to the ambient luminosity. The app uses the phone's luminosity sensor, which is used to control the luminosity of the screen, to control the intensity of the light. An other luminosity sensor directly placed on the lamp must provide also good results and would probably be preferred for a final product.

However, the output of the luminosity cannot be used directly. Indeed, updating directly the intensity of the light with the luminosity recorded by the sensor makes the light very unstable because every variation recorded by the sensor of the phone will make the light change.

In order to improve the stability over time of the intensity of the light, a list of values of integers of fixed size is used to store the last values recorded by the sensor. Then, the average of this list is used as value of the ambient luminosity. this averaging filter enable us to remove the noise and the outliers in the data and provide a smoother signal (See fig3)



```
@Override
public final void onSensorChanged(SensorEvent event) {
     int val = (int)event.values[0];
    mLuxValues.add(0 , val );
    if(mLuxValues.size() > Constants.LUX_FILTER_LENGTH)
        mLuxValues.remove(mLuxValues.size() - 1);

    mLuxAverage = getAverage(mLuxValues);
}
```

Figure 3. Filter the luminosity values

Then, the function drawn fig4 is applied to the averaged luminosity to calculate the value of the intensity of the light.

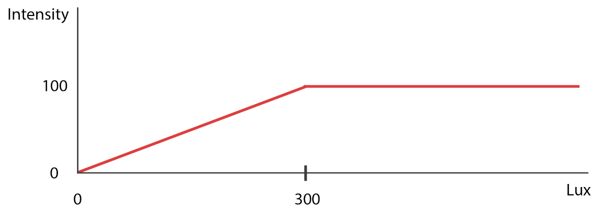


Figure 4. adjusting the intensity of the light

Adjusting the color to the sleep cycles

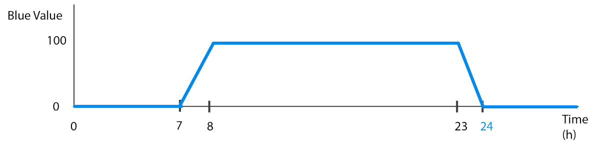


Figure 5. adjusting the blue value

Adapt the light's behavior to the user position
This project has been made with the hypothesis that the activity of the user depends on his position. Therefore, detecting the user's position enable the system to decide which is the current mode to activate.

While the BLE service is connected to the Blen Micro, the CompanionService listen thanks to a broadcasting receiver to the values of the signal strengths sent by the BLE service using intents.

Given the testing environment shown fig6, the average values of the signal strength corresponding to each position are displayed in the table1.

The solution adopted for the project to separate the different activities consists of a set of rules based on the strength of the signal. Depending on the strength of the signal and two thresholds, then a specific lighting mode is set. The code snippet 7 shows how the rules are implemented. The section explains how the threshold mThreshRelax is set and the threshold THRESHOLD_OFF is simply a constant defined as the highest value recorded while the devices are still connected.

Moreover, as the strength of the signal can be subject to interferences and the movement of objects between the phone and the Blend Micro, it's is important to smooth the recorded

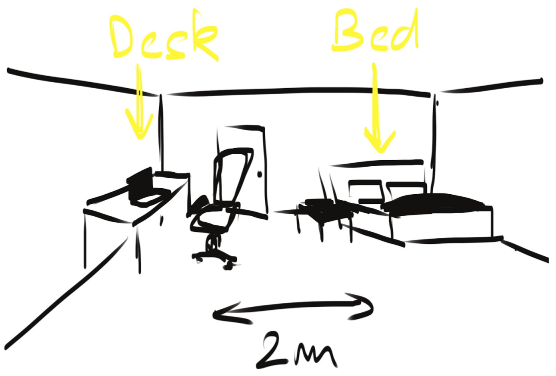
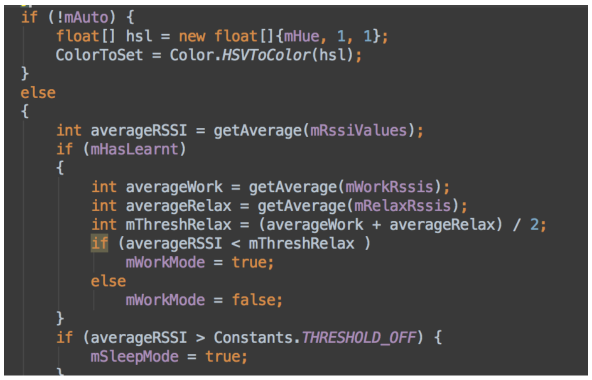


Figure 6. Testing Conditions

Activity	Distance	RSSI
Working	0.5m	-50
Relaxing	2m	-65
Leaving	3m	-72

Table 1. positions, activities and measured RSSI

values. Therefore, the signal is filtered with an average filter similar as the one used to filter the luminosity records. The number of values used for the average filter must be carefully selected because we observe a trade-off between the stability and the reactivity of the system.



```
if (!mAuto) {
    float[] hsl = new float[]{mHue, 1, 1};
    ColorToSet = Color.HSVToColor(hsl);
}
else
{
    int averageRSSI = getAverage(mRssiValues);
    if (mHasLearnt)
    {
        int averageWork = getAverage(mWorkRssis);
        int averageRelax = getAverage(mRelaxRssis);
        int mThreshRelax = (averageWork + averageRelax) / 2;
        if (averageRSSI < mThreshRelax )
            mWorkMode = true;
        else
            mWorkMode = false;
    }
    if (averageRSSI > Constants.THRESHOLD_OFF) {
        mSleepMode = true;
```

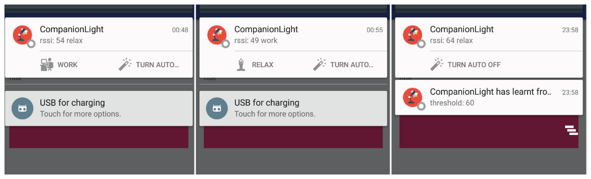
Figure 7. Rule-based inference

However, the light doesn't turn on automatically. It requires the user to start the app, then the app tries to connect to the Blend Micro by itslef. Geofences could be used to start automatically the app, then the app should scan continuously the area. However this solution would be bad for energy management purposes.

Learning the room configuration
Defining the value of the threshold $mThreshRelax$ is a cornerstone of the functioning of the system. It should be picked to optimise the separation of the activities Relaxing and Working. However, defining this value as a constant in the source code would require to the user to modify the code to fit the system to the environment.

Therefore, the system needs to learn this optimal value depending on the environment of use. The solution adopted consists of asking the user to select manually the different modes (Relaxing/Working) when he starts using the app, this is done directly from the foreground notification as it is shown on the first and second screenshots fig8. Every time the user selects a different mode, the systems records the value of the signal strength. After few inputs for each activity, the system notifies the user (third screen, fig8) that it has successfully learnt from the user's input and the systems starts switching the modes automatically depending on the user's position.

This process is an user-friendly way of labeling the data while training a model and could be used to train more complex models.



CompanionLight 00:48
rssi: 54 relax
WORK TURN AUTO...
USB for charging
Touch for more options.

CompanionLight 00:55
rssi: 49 work
RELAX TURN AUTO...
USB for charging
Touch for more options.

CompanionLight 23:58
rssi: 64 relax
TURN AUTO OFF
CompanionLight has learnt fro.. 23:58
threshold: 60

Figure 8. Learning from the user's input

Data flow diagram

The values contained in each message are defined as following:

- R: [0,255]
- G: [0,255]
- B: [0,255]
- gain: [0,100]
- mode { manual: 0x01, relaxing: 0x01 , 0x02: sleepMode (no ligth) }

Please refer to the figure 9 for the Data Flow diagram.

User Experience

The interactions between the user and the app have to be minimized, the app should be working in the background without action from the user. However, during the training phase, the user needs to select the different modes manually (Working/Relaxing). The app uses ongoing notification (always present in the notification drawer) to communicate and interact with the user. A screen-shot of a notification is provided fig.10. The actions available from the notification (switch to the other mode or turn off the automatic mode) depend on the current mode and state of the app. For example, if the app is trained, there is no button to switch between actions. Furthermore, the notification also displays the strength of the signal (RSSI) and the current mode in real time. A flow chart is provided fig.**??** and shows the different state of the UI depending on the context and the state (training/automatic) of the app.

The app contains also an activity shown fig.12 that enables the user to control directly the light in manual mode and manage the main service (start, stop, switch automatic mode and reset the stored data).

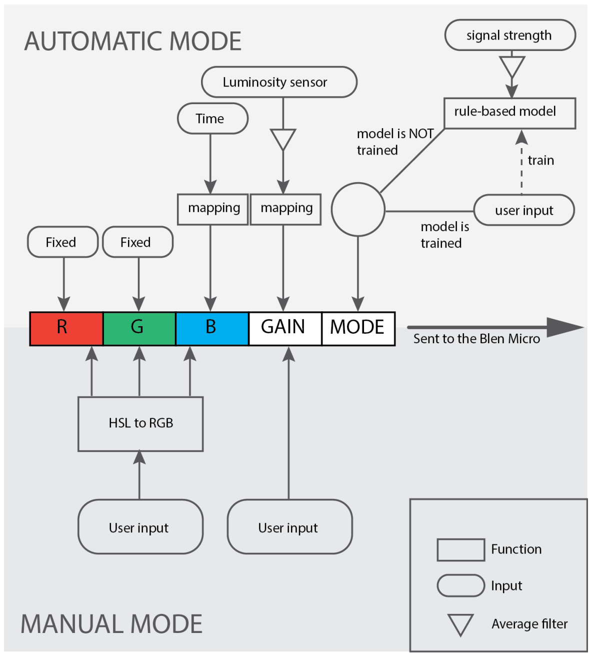


Figure 9. Data Flow

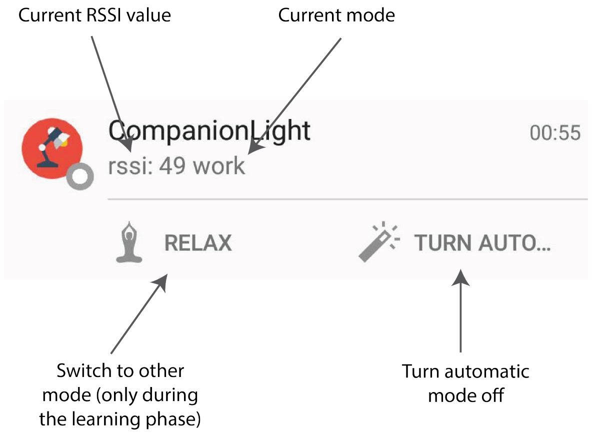


Figure 10. Active notifications

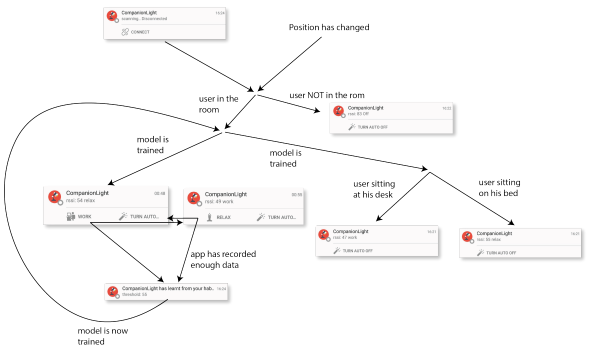


Figure 11. Flow chart

COMPARISON WITH EXISTING SOLUTIONS

Philps Hue

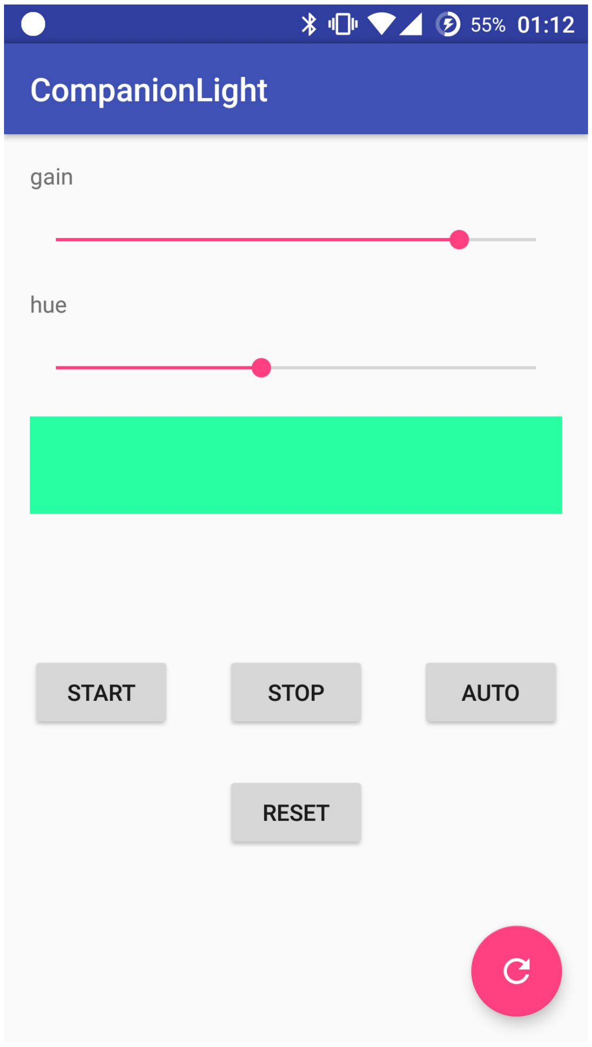


Figure 12. Home Screen

Philips Hue is a connected RGB light bulb [?]. While the CompanionLight is connected directly to a phone thanks to the Bluetooth Low Energy, the Philps Hue is connected to Internet thanks to a Wifi connection. The bulb can be controlled by a separated app to schedule different intensities or colors for different times. Once it is scheduled, the light doesn't need the phone to be connected to perform the changes, which differs totally from the logic of the CompanionLight which needs to be connected.

Moreover, Philips Hue can be controlled by an API which opens it to infinite possibilities for programmers which can program any new functionality.

The API is compatible with IFTTT [4]. IFTTT is an app available for bos iOS and Android which enable the user to connect different services together. The principle is simple: IF the app detects a change THEN a specific action is performed. For example, it can detects that the user enters a specific area thanks to the phone location and turn on the Philps Hue lights. Therefore, it offers more functionality than the CompanionLight.

However, because the CompanionLight uses the strength of the signal, it enables it to evaluate more precisely the position of the user. Thus, it's one advantage against the Hue which couldn't detect the position of the user precisely enough to set automatically the different modes (Work and Relaxing).

Dash Lamp
Dash Lamp [1] is a premium desk lamp that detects user's presence using a PIR sensor. Thanks to this sensor, the app can detect if someone is sitting at the desk and turn on the light, on the opposite, it turns off the light when the user isn't at his desk. The main advantage of the DashLamp is that the user doesn't need to wear any device to use the lamp because the lamp uses a PIR sensor. On the other hand, the lamp cannot manage other lighting modes than turning on/turning off. Moreover, the lamp cannot be programmed to reduce the blue light in the evening for example and it cannot adjust its luminosity to the ambient luminosity.

f.lux
f.lux [3] is an app available for Mac, Windows, Linux, iPhone and Android that controls the temperature of the screen in order to improve sleep quality. In the simplest terms, the color temperature can be considered as the balance between blue and red by analogy with an object that would be heat at a given temperature. Indeed, the more an object is warm, the more is radiating blue light. On the opposite, an object heated at low temperature would a have a reddish color.

f.lux controls the temperature of the colors of the screen to reduce the blue components of the colors in the evening in order to improve sleep quality. For this task, f.lux performs better than CompanionLight because it doesn't only decrease the red component but the temperature. On the other hand, that's the only functionality that the app can performs.

ENCOUNTERED PROBLEMS AND FUTURE IMPROVEMENTS

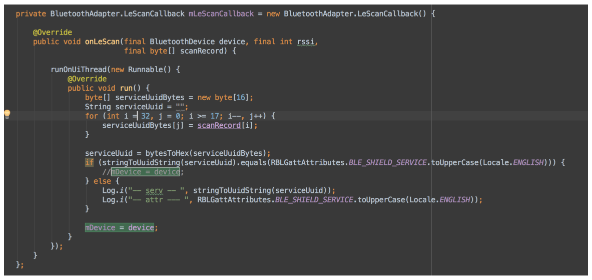
Problems

Electronic, wiring
I have no practical knowledge in electronics. That is why I spent a lot of time trying to make something that works with the LED strip. Moreover, the transistors I ordered initially did not enable me to control the LED strip. Therefore I had to buy other parts separately.

Connecting the Blend Micro and Android
The example app SimpleControls was not working directly with the BlendMicro, I had to tweak the code to make it works. Again, I spent a lot of time finding a solution. The problematic section is provided fig13. However, this solution couldn't be used for a commercial solution.

Android programming skills
I've started programming on Android during this course. Therefore, developing the app has required me more time that it would take to a more experimented developer.

Future improvements



```
private BluetoothAdapter.LeScanCallback mLeScanCallback = new BluetoothAdapter.LeScanCallback() {

    @Override
    public void onLeScan(final BluetoothDevice device, final int rssi,
                         final byte[] scanRecord) {

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                byte[] serviceUuidBytes = new byte[16];
                String serviceUuid = "";
                for (int i = 32, j = 0; i >= 17; i--, j++) {
                    serviceUuidBytes[j] = scanRecord[i];
                }

                serviceUuid = bytesToHex(serviceUuidBytes);
                if (stringToUuidString(serviceUuid).equals(RBLGattAttributes.BLE_SHIELD_SERVICE.toUpperCase(Locale.ENGLISH))) {
                    //mDevice = device;
                } else {
                    Log.i("-- serv -- ", stringToUuidString(serviceUuid));
                    Log.i("-- attr --- ", RBLGattAttributes.BLE_SHIELD_SERVICE.toUpperCase(Locale.ENGLISH));
                }

                mDevice = device;
            }
        });
    }
};
```

Figure 13. The UUIDs provided in the code doesn't match

The initial goals have been reached. However, I was hopping going further but the problems I encountered has reduced dramatically the scope of this project.

Activities, features and classifier

The training mode, as described section provide a great way to train more complex models. My next goals would be to use more than one feature to train a classifier (e.g. Random Forest, SVM) in order to predict other activities: Working, relaxing, sleeping and out-of-the-room. The time of the day could be easily used as a feature for the classifier. Then the ambient luminosity could be used because a low ambient luminosity, for example, would be a good characteristic to determine if the user is asleep. However a low luminosity could also indicates that the user is watching a movie. That's why the sounds level could also be used as a feature and could be use to differentiate the two activities.

Turn on the light automatically

The user still needs to open the app to turn on the light because connecting the app requires an action from the user. Geofences could be used to detects when the user is approaching his home. Then, a solution would be to scan continuously the area until the app finds the BLE device. However, this strategy would be really power consuming.

connect the device to the Internet

Connecting the device to the Internet would enable the app to control the light without being directly connected.Therefore it would be a solution not only to turn on automatically the light but also to implement more complex features.

CONCLUSION

The goals defined at the beginning of the project have been reached (adapting the light to the user's position, the ambient luminosity and the sleep cycles). This project makes an interesting example that shows how to use signal strengths and sensors to turn surrounding objects into context aware objects by detecting user relative location. However, the app still cannot be turned on automatically if the phone is not connected to the BLE module. The prototype could be improved by adding directly sensors to the lamp and by connecting the lamp to the Internet.

The source code is available on my Github.

REFERENCES

1. Dash Lamp. `https://www.steelcase.com/eu-en/products/lighting/dash/#features_benefits`.
2. Whats in a Color? The Unique Human Health Effects of Blue Light. `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2831986/`.
3. f.lux. `https://justgetflux.com`.
4. IFTTT. `https://ifttt.com`.
5. SimpleControls andorid app. `https://github.com/RedBearLab/Android/tree/master/Examples/Android%20Studio%20Examples/SimpleControls`.