Internet of Things
LightMeUp Project Report

Xin Wang (4301994)
Amjed Yousif (4178890)

January 22, 2017

1 Introduction

Environment aware, energy efficient and smartness are the characteristics of this era systems design. Under this umbrella and with lots of ambitions we designed a smart, energy aware and easy-to-use system to control the light inside a room. We call it *LightMeUp*.

Although sun provides us with lots of light. Manually controlling the amount of light that enters your home, office or building is indeed not the optimal and most comfortable way. Therefore, we wanted to automatize this process while still giving the user full control and making use of the available technology.

The available technologies that try to optimize lighting systems in buildings still have many weaknesses or not optimal solutions. For instance, Philips hue light is an amazing technology that enable people to change colors, sphere and light intensity but it is completely environment unaware system. Other well known technologies, that works as an automatic switch by using a dedicated light and motion sensors.They are great at reducing energy consumption but user cannot use the sensors to reduce the intensity of the light or control the light before entring the room.

2 Related work

An automatic room light detection and control system consisting many Home Light Control Modules (HLCMs) was developed using microprocessor and light sensors [1]. In this work, each HLCM is made up of four blocks: the pyroelectric infrared (PIR) sensor circuit, the light sensor circuit, the microprocessor and the RF module. In each room, the HLCM detects if any person enters the detection area. Then according to the this result, the system maintains sufficient lights by controlling the number of lights.

This work reduces energy consumption at a low cost. However, it has some weaknesses: *a*) the desired light intensity values are fixed, users cannot customize the values; *b*) the HLCM is soldered, so it's hard to get the system extended.

Another smart lighting system in the market is Philips hue [2]. This "personal wireless lighting" system provides LED bulbs along with a bridge and mobile applications. By using Zigbee protocol, the bridge controls the wireless chip inside the LED bulbs. It provides the ability to control the light in a wireless and scalable way.

Philips hue is easy to set up. Users just need to connect the bridge to the router at home and use the mobile application to control the lights. Nevertheless, Philips hue has limitations: *a*) it works with single user, other users cannot play with the lights; *b*) it requires user to control the light intensity, cannot act according to the environment automatically.

3 LightMeUp

LightMeUp it is a smart lighting system that combines available technologies and resources to provide the user with a comfortable, great lighting experience. Furthermore, LighMeUp is environmental aware and it is able to distinguish between day and night and based on that it starts with bulb at night and keep the curtain closed or starts with opening the curtain to meet user desired light intensity and then lights up the bulb gradually if necessary. Therefore, it is energy efficient system, and the control logic is self-adapted. Figure 1 depicts how users can interact with the system and how system components are communicating.

3.1 Design goals

- Flexibility. The controlling algorithm should make its decisions based on the environment feedback and user preferences. This will give the user more freedom in using the system in different scenarios. For example, if the user does not want to use the bulb or have a different

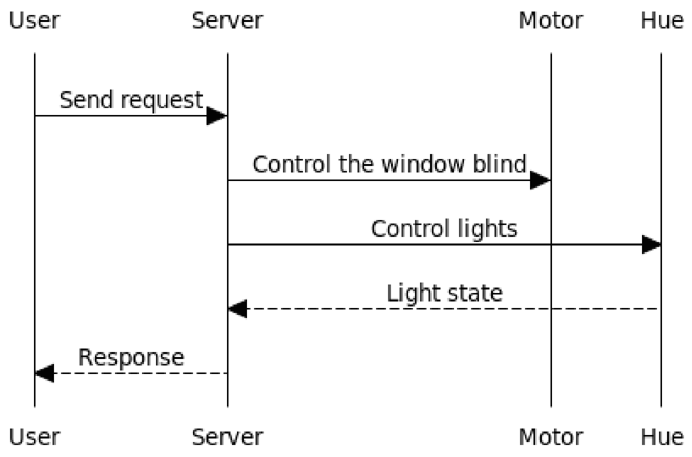

Figure 1: Sequence diagram

type of curtains.

- Energy efficiency. The system needs to be environmentally friendly. Thus, the power consumption of this implementation should be as less as possible.
- Modularity. For the basic usage, the system only needs to control the window blind and lights in the room. However, it is better to get the system scalable. Actuators like motors and bulbs are modules in this system. So it's easy to add more features in the future.
- Environment-awareness. The system should be aware of different scenarios. For example, the system behaviors differently according to the time, such that the light should not be turned on during the daytime.
- Multi-user support. To be more user-friendly, all users in the same room should have the ability to control the system. Therefore, the system need to leverage each user's request.
- Plug-and-play. This means users just need simple steps to set up the whole system then it works.

4 System design overview

Based on the goals we explained in Section 3.1, the system is designed to be extensible and easy to access.

The architecture of the system is shown in Figure 2. “User A” and “User B” are the client part, which indicates that the system supports multiple users to control the light together.

The server, as shown in the center of the picture, is responsible to control all actuators. Lights and window blinds are the components under control of the server.

We chose this architecture because *a*) we can use the built-in light sensor on smartphones, no extra sensor is needed; *b*) server can provide services to differenct kinds of clients as long as they are equiped light sensors and use the same protocols; *c*) the system is scalable so that actuators are easy to be installed or replaced.

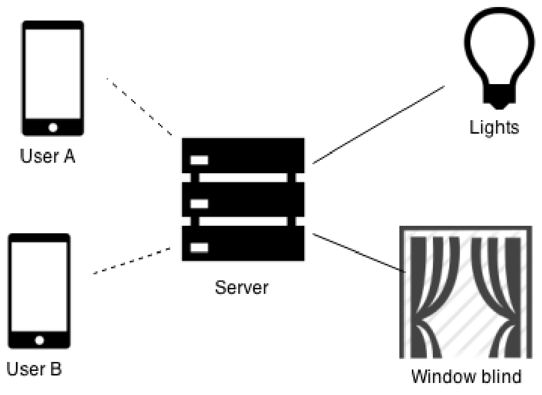

Figure 2: System architecture

5 Implementation

The system is based on the Client-Sever model. The client part is built with Android platform running on smartphones, while the server is implemented on the Raspberry Pi hardware. The system employs motor and bulbs as actuators to control the light intensity values.

5.1 Server

A Raspberry Pi Model B[3] is used as the server in our implementation. A web sever that is built with Flask framework[4], runs on the Raspberry Pi. We chose this framework because it's a micro-framework for Python, which can interact with GPIO easily with existing libraries.

The web server consists of several files. The file structure is shown in Figure 3.

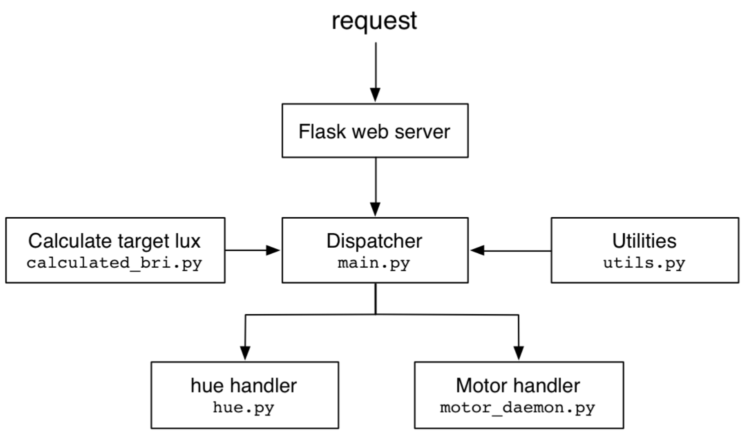

Figure 3: Web server files

main.py this file handles all incoming requests from clients. It has several tasks: *a*) gets the request then parses light intensity values and the desired values from users; *b*) controls motor or light actions, according to the particular control algorithm; *c*) handles different desired lux values from multiple users and computes a balanced value.

According to the protocol, clients send requests containing three parameters: `current`, `desired`, `weight`.`current` is the averaged light intensity value, which is described in Section 5.2; `desired` is the light intensity value that the user wants to get; `weight` is the factor which influences the final target value the system calculates. The default value of weight is 1. When in a multi-user scenario, users have different weight value. For instance, the host of this room may have a higher value, so the host has a higher rank to affect the light in the room.

The control logic is straightforward: in order to save more energy, the system first tries to adjust the curtain to satisfy the requested light condition during the daytime. If the curtain is fully open, but user wants more light, the hue bulb will be turned on; If at night, the system closes the curtain by default, then controls the hue bulb to provide light for user. The flow chart can be seen in Figure 4.

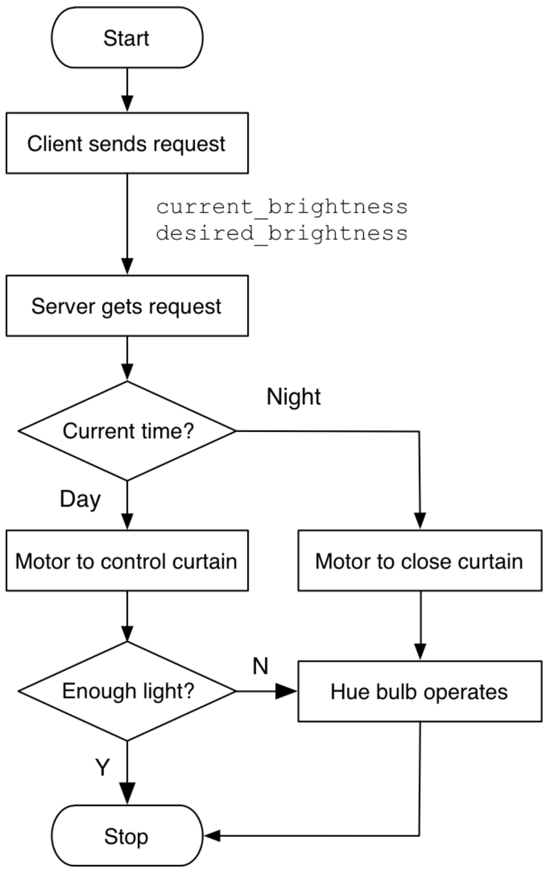

Figure 4: Control logic

Because users may change their desired values, the motor and hue lights always need to operate. We keep the states of these components so that this process can be described using finite-state machine (FSM) model as shown in Figure 5.

When user exits the room or quit the application, the client sends `quit` request to the server. So the server knows there is a user leaving so that the

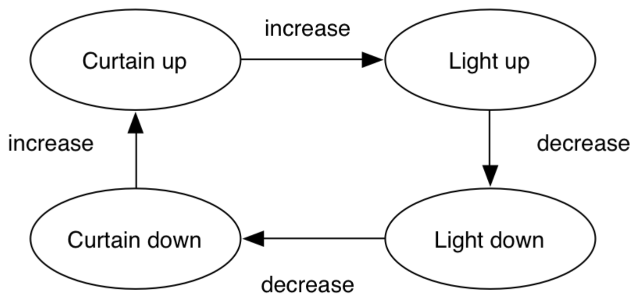

Figure 5: Finite-state machine model

server stops the motor and resets the calculated desired value.

Apart from this main method, we designed some APIs for users to control the system manually. The full API methods show in Table 1.

Table 1: API methods

URI	Parameters	Description
`/control`	`current`: float `desired`: float `weight`: int	Requests the system to adjust light
`/control/stop`	-	Stops the motor
`/quit`	-	Indicates the user exits the room
`/motor/start/cw`	-	Rotates the motor clockwise
`/motor/start/ccw`	-	Rotates the motor counter-clockwise
`/lights/on/<light_id>`	-	Turns on the specific bulb
`/lights/off/<light_id>`	-	Turns off the specific bulb

hue.py To interact with Philips hue, some APIs are encapsulated in this file. Basically, in this system, we only need to get and set the states of bulbs.

- `set_light()` sends requests to the hue bridge. The parameters include 'on', 'sat', 'bri', 'hue', etc. So the bulb can be fully customized.
- `get_light()` gets the current states of the bulb.

motor_daemon.py This handles motor-related operations. When rotates the motor, this script works in background so that this process does not block the server process. It provides two methods to run the motor: `run_cw()` rotates the motor clockwise; `run_ccw()` rotates the motor couter-clockwise.

device_state.py This file stores the current state of motor and hue lights. Because HTTP is a stateless protocol[5], server knows nothing about the current states of devices. We keep the states of these components into a JSON file, and update this file whenever the states change.

calculated_bri.py This system supports multiple users with different desired light intensity values. So we need an algorithm to compute the balanced target value to let the motor and bulbs achieve. This file provides a formula for computing the desired value:

$$Value = \frac{Value_{old} + Value_{new} * weight}{2}$$

.

utils.py All utilities functions are located in this file.

- `render_json()` returns response to clients in JSON format.
- `is_night()` detects the current time is at night or during the daytime. The system behaves differenctly according to the current time.
- `log_data()` logs the request data into a file. This is needed for further analysis.

5.2 Android app

Since almost everyone has a smartphone nowadays, and they are equipped with sensors. We thought that using smartphones to get light intensity values (lux) is the way to implement smart lighting system (LightMeUp). Additionally, this choice eliminates having a dedicated sensors to get light intensity values. Thus, reducing the cost.

Android allows access to the sensors values via *SensorManager* class [6]. After getting the values the app calculates the average of every 10 values and sends it over the network. The number 10 was chosen based on experiments and trade off between latency and stability (e.g. if a hand was moved over the smartphone sensor without without keeping it over it the system will not react to that change in reading) of the system. Averaging each 10 values and sending the averaged value to the server gives a latency of about two seconds to the systems. The reason for that is the sampling frequency of the light sensor is 5Hz. To send data over the network we use Java *urlconnection* class and Android specialize class *AsyncTask* to do work in the background and not freeze the app.

The app basically sends three values. Namely the desired value, specified by the phone owner, the current reading from the light sensor, and the weight value which is described in Section 5.1, as url-encoded parameters. The Figure 6 is a print screen of the Android application.

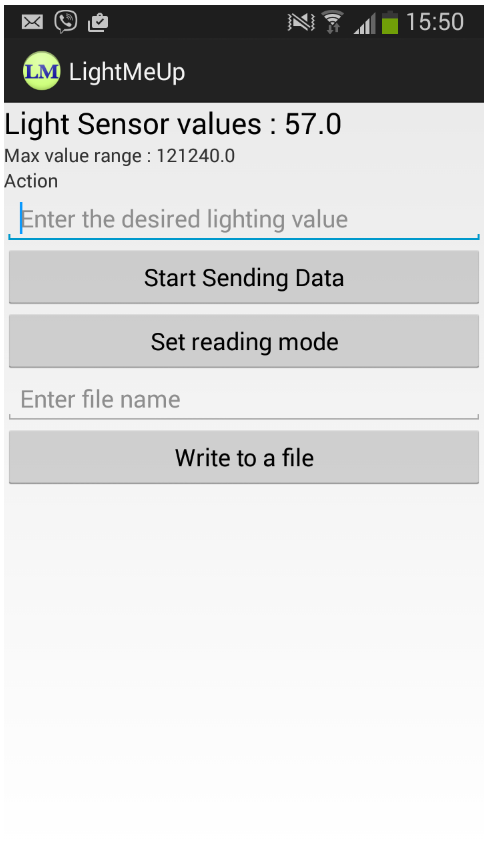
15:50
LightMeUp
Light Sensor values : 57.0
Max value range : 121240.0
Action
Enter the desired lighting value
Start Sending Data
Set reading mode
Enter file name
Write to a file

Figure 6: Print screen of the Android app

The application also has a special mode *reading*, this was implemented to show the ability of implementing special lighting modes like relax, romantic or party mode, which specifies the best lighting condition for the reader. Additionally, the systems gives one of three messages as a feedback to the reader: “Good conditions for reading, Enjoy ”, “The conditions are not perfect but good enough to start reading” and “reading here might harm your eyes!!!”.

For further investigation we add a new feature to the app. That enables the user to collect light data and related timing information. This helps us to gain better understanding for the capability and limitations of smartphones’ light sensors. Some results will be shown and discussed in Section 7.

5.3 Actuators

This implementation includes two kinds of actuators: motor to control the window blinds; bulbs to light up the room.

5.3.1 Philips hue

We have chosen Philips hue because it is Wi-Fi enabled, energy efficient bulb (8.5 watts) and has good APIs to work with [7]. Philips hue consists of a set of bulbs and a bridge. The communications between bulbs and the bridge are achieved by Zigbee technology, while the communications between the bridge and the router are done by Ethernet.

We can set light attributes by sending HTTP requests to the bridge with various arguments. Table 2 shows some arguments to customize the hue bulbs. The bridge enable us to control bulbs individually or as a group. Furthermore, this system supports many lighting scenes and scenarios.

Table 2: Some Philips hue light API arguments

Name	Description
on	On/Off state of the light
bri	The brightness value to set the light to
hue	The hue value to set the light to
sat	The saturation value to set the light to

5.3.2 Motor

We chose a stepper motor, model 28BJY-48, for this system. This device is widely available and easy to use. It needs the ULN2003 controller to play with. Raspberry Pi provides 17 GPIO pins. We use four pins to control the motor controller.

The stepper motor needs a sequence of “high” and “low” levels to each input pin. By setting different sequences, we can control the motor to rotate clockwise or counter-clockwise. Therefore, we can control the motor to open or close the window blinds (or curtain).

A Python script was created to send GPIO signals to the motor. So the server just needs to run or stop this script to control the motor. However, this process should not be run in the web server process, because it blocks other requests. Therefore, we made the motor script run in daemon mode.

5.4 Integration

The Raspberry Pi and Philips hue need to be integrated. Philips hue ships with a bridge, which is designed to connect the hue mobile app and the bulbs. It also acts as a server providing APIs for developers. In our case, we use the APIs to play with the bulbs. Therefore the bridge is needed to be connected to the system.

The connector of the bridge is RJ45. There are two ways to link the bridge with Raspberry Pi. The first option is to connect to a USB port of Raspberry Pi by using the USB Ethernet adapter, because the Ethernet port of Raspberry Pi is occupied for connecting to the network. The second option is to use a router, getting the Raspberry Pi and the bridge all connected to the router.

We chose the second option to get them integrated. Here are some reasons: *a*) the bridge only works in the DHCP client mode, which means if it connects to the Raspberry Pi, it will need extra network configuration work; *b*) introduce a router would enhance the scalability of this system; *c*) a router can provide the accessibility from both the Internet and the local network.

Now the Raspberry Pi can send HTTP request to the hue bridge with parameters to control the light states via the IP address of the bridge. Once smartphones join the same wireless LAN, they can also communicate with the server on Raspberry Pi via IP addresses.

Figure 7 describes the overall topological structure of this network.

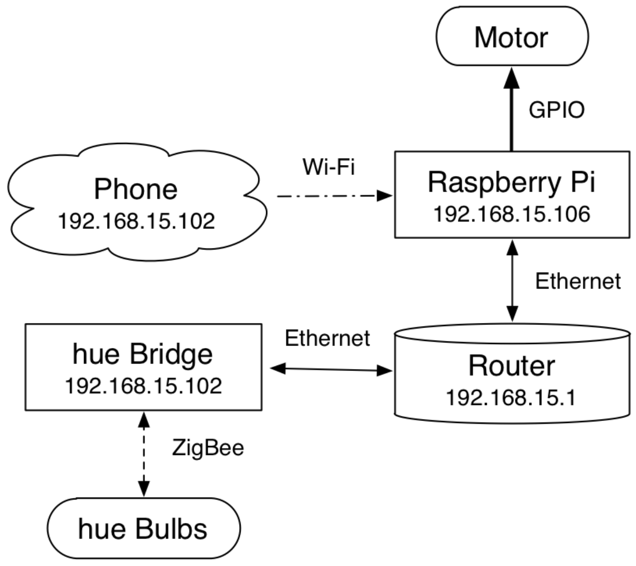

Figure 7: Topological structure

6 Results

We tested our system in daytime and nighttime scenarios. Figure 8 shows the result of daytime. At first, the curtain is closed. Client sends the desired value of 100, then the system opens the curtain. At around 54″, we see the light does not increase, which shows the curtain is fully open. Then the bulb is turned on. We see after the bulb adjusts several times, the light condition reaches the desired value (100 lux).

The second scenario is at night. Figure 9 depicts this result. The user sets the desired value to 1000, which is beyond the capability of the bulb. We can see the light is increased to the maximum value of 50. Then it remains stable. This also shows that the system provides the best-effort service.

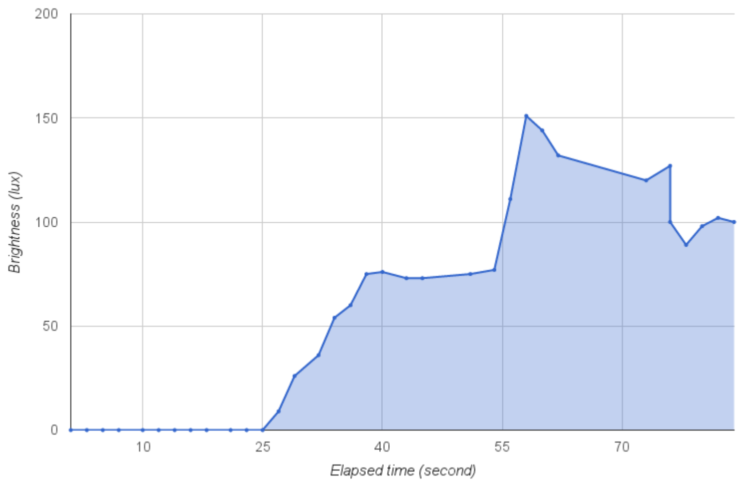

Figure 8: Result of setting desired value: 100 (daytime)

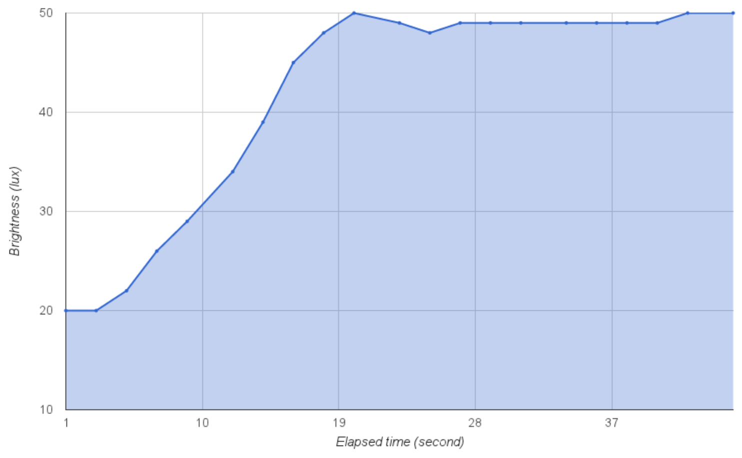

Figure 9: Result of setting desired value: 1000 (night)

7 Discussion

7.1 Walking effects the variation of the measurements

The Android app allows us to collect light intensity measurements and the related timing. We have done the following experiments to gain better understanding about light distribution and the effect of other factors on the light intensity measurements. First, we put an Android phone on a table without moving it and let it collect measurements for a couple of minutes. The we take phone and walk around in the room and also collect measurements. In the first scenario Figure 10 shows that the reading is very stable, However, for the second scenario we see the values changing significantly. The reason for that is not walking itself but rather how the light intensity is distributed in that room. The phenomena can have a great impact on the system stability.

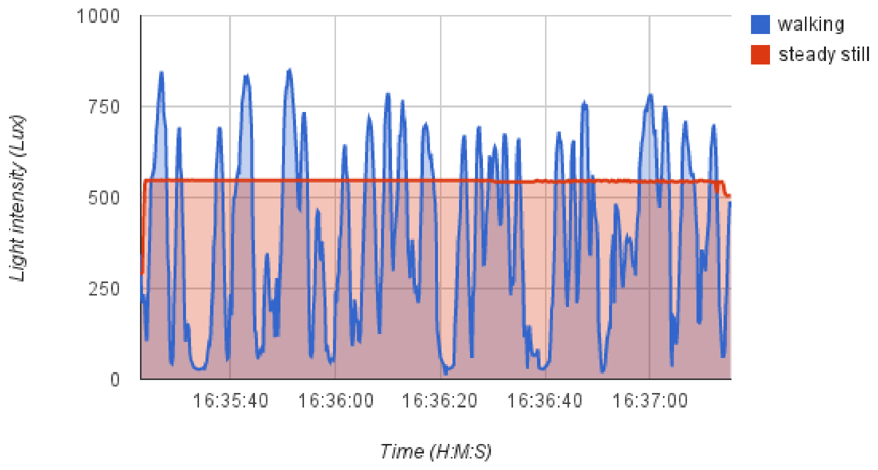

Figure 10: Measuring light intensity while being steadystill and walking

Therefore, we developed an algorithm to remove the noise effects. First step in the algorithm is to average every 10 readings, this introduces a two seconds delay to the system but increases the stability. The second step is to append this averaged value to the previous ones and obtain their average.

However, if the created list is to long that will kill the effect of the new reading. Therefore, and base on the experiments we have done, we choose the number three to be the maximum value for that list. Figure11 shows sensor raw data and figure12 shows the results of our applied algorithm. The algorithm does great job to remove the noise and to follow up with original data. However, the graphs clearly indicate two things. First we see the there is some redaction in the amount of lux readings produces form our algorithm but that has a very minor impact on our system. Second, the last part of the graphs that clearly shows the two second delay effect.

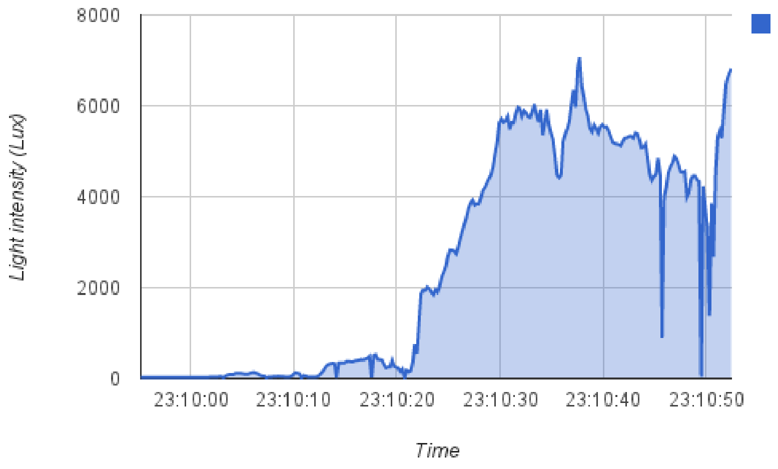

Figure 11: Light sensor raw data.

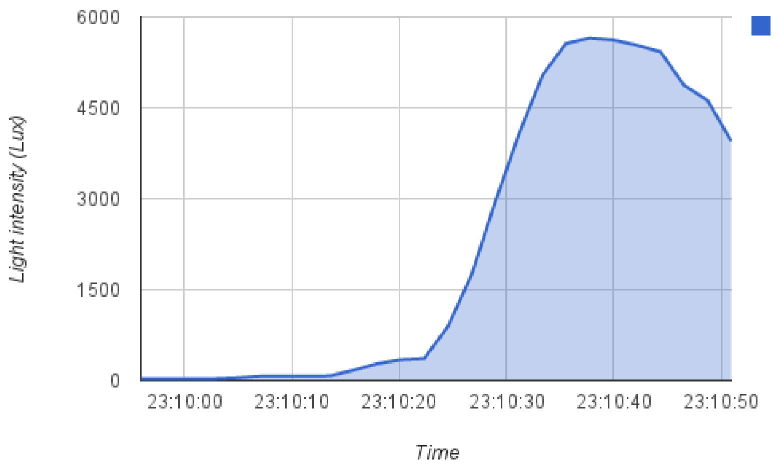

Figure 12: The results of our noise reduction algorithm

7.2 Curtain position detection

In our design, we would like to make the system flexible and do not rely on the fixed settings. However, there is a problem: how to know the current position of the curtain (or window blind). The curtain is a simple mechanical component that cannot give any feedback to the system. Therefore, in this implementation, the system relies on the current light intensity value to detect if the curtain is open or closed. The idea is that if the current light intensity value is 0, meaning no light in the room, the system assumes the curtain is closed.

However, this approach is inaccurate because the values from sensors are unstable. The zero values may be passed to the server when the curtain is not fully closed. We can see the jitter also from Figure 10.

To overcome this issue, we suggest employing two mechanical triggers installed on the top and bottom of the curtain. When the curtain is open or closed, it pushes the trigger so that the system knows the actual position of the curtain.

7.3 System latency

The latency comes from the time difference between sending the current light intensity value from the client and changing the light condition by actuators. According to our measurements, it needs about 3 seconds to process a request cycle. A result is that the system cannot stop acting immediately when the desired situation is satisfied.

In order to ease this effect, we set a threshold value. When the difference is under the threshold value, the motor or hue bulbs stop acting. Therefore, the system has a buffer time to finish the actions.

8 Conclusion

Our main conclusion is that we can indeed make a better user of the available technology. However, What the nature provides us with should always be a core element in the design plan. The second conclusion is that to design a smart system an engineer should always start from the basic principles. That will enable the designer to have a wide view on the possible way to take down the challenges. Furthermore, we think that although implementing the system using a smartphone sensor as an input to the system has some advantages it is also produce some limitations to the systems. For example and as we showed above, movement will have a great consequences on the read lux values. On the other hand, the work also indicate that controlling the environment around you is for sure possible and the future of homes and buildings capabilities. Finally, here a very short summary about our work for convienent. A light control system based on smartphone and Raspberry Pi platform was proposed and implemented in this report. The system contains different components like a stepper motor and Philips hue. An Android application as the client part was developed to send request to the sever. A web server to handle requests and control actuators using an energy-saving algorithm was described and implemented as well. Then, we showed the results with collected data. Finally, we discussed the issues we had in this project and proposed possible solutions.

9 Future work

As a next step to improve this work. One can work on improving the controlling algorithm, such that the system is more aware about the curtain state. A thoroughly study on the light distribution in a room. Walking steady still experiment shows the effects of light distribution and the correlations with the movements. Also developers can develops more scenarios or modes as we call them. For example alarm mode where the user enters a time and the curtain will open at that exact time as a way to wake people up and let them to start their day under the sun rays. Furthermore, one can work on finding a better way to control curtain or window blinds, letting the mechanical components send feedback to the system.

References

[1] Ying-Wen Bai and Yi-Te Ku. Automatic room light intensity detection and control using a microprocessor and light sensors. *Consumer Electronics, IEEE Transactions on*, 54(3):1173–1176, August 2008.

[2] Meet hue. `https://www.meethue.com/`. Accessed: 11-11-2014.

[3] Raspberry Pi Model B. `http://www.raspberrypi.org/products/model-b/`. Accessed: 10-11-2014.

[4] Flask web framework. `http://flask.pocoo.org/`. Accessed: 10-11-2014.

[5] Hypertext transfer protocol (http/1.1): Message syntax and routing. `http://tools.ietf.org/html/rfc7230`. Accessed: 15-11-2014.

[6] Android developer website. `http://developer.android.com/reference/android/hardware/SensorManager.html`. Accessed: 10-11-2014.

[7] Philips hue api. `http://www.developers.meethue.com/`. Accessed: 10-11-2014.