Bachelor thesis
Computing Science

Radboud University

Automating the Light Switch

Author:
Maxim van Loo
s4457633

First supervisor/assessor:
Dr. Pieter Koopman
pieter@cs.ru.nl

Second assessor:
Dr. Peter Achten
p.achten@cs.ru.nl

January 31, 2021

Abstract

The goal of this thesis is to make a smart lighting system. This system should provide ample and hopefully pleasant (according to the users wishes) lighting to the user with as little user interaction as possible, if the user is at home. A system like this is, according to my own research, not available yet. This system will learn the user's wishes based upon the amount of lighting preferred by the user in the past few days. If the system detects the users presence, it will use this and the amount of lighting currently in the room to work towards the users preference. This thesis describes how such a system with a Raspberry Pi and two simple sensors can be made. Personal experience shows that this is a step forward in the way of fully automated lighting, but that it still is not completely perfect. Though smarter, the system still misses better person detection and preference prediction.

Contents

Chapter 1

Introduction

I have been using smart lighting for a while, specifically EasyBulb and Philips Hue lights. In my use, I felt that although these are a big step forward in home automation, the way the user has to operate these lights is still quite manual. The value in adjusting your lights anywhere at any given time (via an app or just a remote) to me, was great. However, in my opinion, there was a lot of convenience missing for a product category that was called 'smart'. With this in mind, I wanted to find ways to improve smart lighting systems. Taking inspiration from other popular home automation products, a learning algorithm seems intriguing. Of course, I will have to look at the viability of this idea.

In these next chapters I will look at the current state of home automation products and smart lighting in particular. The importance of this is clear immediately; to improve, we have to know how far we have come so far. With that in mind, we can define what we want to improve. We will make this as a prototype using a Raspberry Pi and two simple sensors. As a conclusion, I will share my findings on the end result and share the shortcomings I experienced.

Chapter 2

The State of the Art

Before we can look at the improvement of smart home lighting systems, we have to take a look at the current state of art. This chapter is dedicated to establishing a baseline and seeing what it is that can be improved upon. We will look at some history of smart homes first. After that, we will look at some of the reasons why, in recent times, it has taken off so well and why it is interesting to expand upon this. At the end of this chapter, I hope to have explained some of the factors that have been crucial to the growth of home automation. In its current state, a case can be made that although the market has evolved, there is ample room for improvement.

2.1 How young is the smart home market?

Home automation systems have been spoken about as early as 1992 [1]. In those times, home automation meant sensors throughout the house to aid in saving energy, getting automatic emergency dispatch in case of an emergency or break-in and environmental systems that took multiple rooms into account when heating or cooling the house. Things we now consider 'smart', like controlling your microwave through an internet connection, were of little interest. Back then, the biggest factor holding back the technology was price. Ever since, several innovations have spurred the development of the market to a fairly mature one. Examples of these are:

- the rise of smartphone possession;
- growing amount of communication protocols for devices;
- lowering cost of powerful technology and their individual qualities;
- the rise of voice assistants and the possibility to tie services together

Through these, the possibilities to create systems to control (part of) your house have taken off tremendously. Even though this has made a lot of smart home technology available, an interesting question is whether the goal of home automation (discussed in 2.3) is being reached. I hope to have reached an answer to this at the end of this chapter.

2.2 The smart home market in recent history

The search term “Smart Home” has gained quite some traction in the past seven years [2].
We will mention a few triggers of those. While smart lighting systems might feel like a short-lived trend, there are a few events and products that confirm that smart home products are probably here to stay.

2.2.1 Popular products in the current market

In 2012 Philips Hue was released [3], which still is a very popular smart lighting system, going for its fourth generation. In 2014, Google acquired Nest Labs [4], a company famous for its Smart Thermostat, first launched in 2011. This once again confirmed that the market is probably here to stay. One other clear trend in smart home devices seems to be its lowering price as, for example, instead of the $ 12,000 thermostat you could get in 1992 [1], a Nest Learning thermostat in 2019 had an MSRP of $ 249, with more recent versions selling for $ 130. Lowering price in most markets often means that it grows. As the technology gets more widely adopted, it's available to more people.

2.2.2 Other factors enabling growth of the market

Apart from that, home assistants like the Google Home and Amazon Echo tie products from different manufacturers together by supplying one interface to interact with, making it more compelling to start adding smart home products to your home. Apart from these smart home hubs, 68 % of people in advanced countries in 2015 owned a smartphone [5], which means that the better part of the population has an interface to these smart home devices within hand's reach. This on its own makes it worth it to research the possibilities of smart homes.

2.3 The goal of smart home products

The goals of home automation seem to be simple; the commonality between all of the different applications is that they try to find a solution to daily tasks in such a way that they make homes more comfortable to live in. They do this, often, by reducing the amount of energy the user has to put in to get a certain task done. For example, turning off lights from your phone saves the user the energy it would take to walk to the light switch. The challenge then is to find a balance between (to name a few) energy efficiency, affordability, security, ease of use and personal safety and keep none out of the equation. Although many of these products do reach these goals; a lot of them are not that 'smart'. Also, they might rely on other products to enable more 'smartness'. Apart from their dependence on a mobile app, they might make use of 'routines' set up in Google's Home or Amazon's Alexa app. This of course does not mean they are not smart. However, a case can be made that some products get some of their smartness from other smart products. And in this way, are they not falling short on that same balance, leaving out affordability and/or ease of use? In the next chapter I will define what it is I find where smart lighting specifically falls short, and how I would like to improve on that.

2.4 Home automation in research

Quite some work has already been done in the space of smart homes. Although mostly descriptive of the current situation and the history, some make convincing arguments as to what can be added in the future of smart homes and their lighting systems.

See for example this article by Rayoung Yang and Mark W. Newman [6] on the learning thermostat and its use by end users and their solutions for the future of smart homes. The *exception flagging* mentioned in this paper is interesting. It poses that circumstances outside of the norm of regular behavior of the user should in some way be flagged. As machine learning will often have a hard time detecting exceptions, the paper suggests that flagging by the user might prove useful. This can make sure the system itself doesn't go into odd ways to make that exception part of its regular behavior.

Heierman and Cook [7] pose an interesting future of using user habits to determine their needs, using Episode Discovery, or ED for short. ED, according to Heierman and Cook, helps in discovering regularities, or episodes, in mined data of the user to provide the desired device automation once these episodes are defined. They show that it has quite some potential, especially when combined with other data mining methods. This of course would be a welcome addition in detecting (different) user presences or even user wishes.

Though an old article, Jiang, Liu and Yang [8] give a reminder of the state of smart home devices back in 2004. However, the challenges they state still remain salient. For example; the biggest challenge in many systems is still reliability. Especially when making these smarter with learning algorithms, a user depends more and more on this.

The research already done on Ambient Intelligence [9] has a bit of a wider scope than lighting. It poses that entire environments should be more aware of user's wishes and act upon that unobtrusively. Though it is already known that a lot of human behavior is predictable and therefore, their wishes too, this also means a lot of data that needs to be processed. However, the possibility of ambient intelligence communicating with a lighting system is something that would be even better than what I set out to do.

2.5 The current state of smart lighting

In current smart lighting systems, the most common approach to more intelligently turn on/off lighting is the *routine* application, which is often present in their own app, or on a smart assistant's ecosystem. In the routine setting, the user sets a few conditions to the turning on or off of the light and sometimes even adjusting its brightness. For example: Turn on lights at 70 % at 10AM on weekdays. This still requires a lot of user input and is a one-size-fits-all approach at best: the routines are set and then that is hopefully sufficient. The smart lighting system does not need to have any further awareness, apart from the current time. In a highly regular lifestyle, this would work of course. However, a lot of people do not live in the regularity a routine requires to work. Even then, fine tuning these routines requires a lot of effort and with the specificity it requires it means that the user has to create more of these routines. As such, this approach is hardly smart. Even in the arguably easy way to set it up in, for example, the Philips Hue app, it barely provides any smartness that the user hasn't defined themselves.

Also, Philips [10] has a motion sensor to use with the other options their ecosystem provides, but it still does not break away from the need for specificity and one-size-fits-all approach. Still, the user needs to define what has to happen in what kind of circumstances.

In the current state of smart lighting, the central point to set-up lighting and accessing its smart features, seems to still be an app or smart assistant. In essence, this isn't necessarily a problem. The problem with this, and what in my opinion is in dire need of improvement, is the fact that these still only provide mediocre smartness to the user. These are mostly based on 'if this, do this' kind of implementations, based on the user's preferences and will to set this up. In a more ideal version, the user should not be bothered with fine-tuning and reevaluating the system before it is set up just perfectly, for that particular moment. If it does not turn on or off to the users wishes, the user has to use the switch or app again. Ideally, the system would learn the users preferences out along the way and act upon those.

Though not a lighting system, the Nest Learning Thermostat tries to categorize a households heating wishes using machine learning; it uses the users input over time as training data to eventually do the work for you. This changes the generally on-demand nature of controlling the thermostat to one that aims to minimize user interaction: it does the work for you. Though challenging, this would be a great way to expand upon the smartness of lighting systems currently available.

Now of course, heating and lighting are inherently different. Whereas the heating in a room is quite constant and gradual, lighting can change very quickly (for example by a cloud passing in front of the sun). Luckily, it's also a lot easier to change lighting quickly, whereas heating is never that direct. What both lighting and heating have in common is their immediate influence on a home; someone will not be comfortable (often expressed by the user changing current settings/conditions) when it's too cold or too hot, just like when it's too dark or too bright. However, these systems both have to keep up with the user's presence and preference.

Therefore, I think it's only right and the point of this paper will be to look at a system that's seemingly ahead and think of the improvement some system like a Nest learning algorithm would have for lighting.

Chapter 3

Creating Smarter Lighting

We will be looking at how we can lessen the amount of interaction so that the system is working for you, contrary to the now well-established on-demand type of control that is available right now. To reach that, we need to deal with the following points:

- Desired improvement;
- The scope of this improvement;
- The options to create an improved system;
- A prototype of such a system.

I will be dealing with these parts in the following sections of this chapter. The fourth will have its own chapter, as it is a longer part, focused on making a system fulfilling the requirements and improvements I will propose in this chapter.

3.1 Desired improvement

In all the different implementations of smart lighting the light switch still is central. We have moved the light switch that is present in most households around in very different ways. In every implementation, be it through a smartphone, a voice activated assistant or even a wireless remote, we have managed to give the user the ability to switch on and off their lights at any place and time without having to move to a fixed location, this being the switch in the wall. Though *routines* (see 2.5) offer a way to not use the light switch, it's still a temporary fix. Though it provides some 'smartness', it is also just as binary as a singular motion sensor, as it is still based on a *if x then y* approach within bounds set by the user.

This paper focuses on the possibility that we might be able to move the focus from the light switch and diminish the need for that particular implementation. This does not mean that the switch, whether digital or physical, does not have its place anymore; it merely means that other ways might have a reason to exist as well. Current systems in smart lighting seem to be sufficient in every way: they are secure enough, intuitive enough, smart enough, affordable enough and inter-connected in a way that is enough. The requirement therefore, assuming everything else stays the same, is to improve on its convenience. In short; this paper will look into a way to minimize the use of the light switch, whether digital, physical or voice-activated.

3.2 The scope of an improved system

First of all, improving more than convenience would not be feasible as the improvement of most singular aspects are worth a research paper in itself. Though energy-saving and security are important and have to be taken into account, we will not be focusing on those. As long as these other qualities of smart lighting are not notably impaired, this improvement on convenience poses a sensible addition to smart lighting systems. I hope to minimize interaction in such a way that its decisions are in line with the users wishes, but don't require the user to constantly state their wishes. This is, hopefully, a welcome improvement on the current state of smart lighting as my biggest complaint right now is its dependence on the light switch in different forms. This dependence is both outdated (other home automation products are ahead of this) and very manual, something smart lighting should not have to be.

3.3 The options to create an improved system

3.3.1 Single sensor solutions

Some systems have proven to be so straightforward that a single sensor can solve it. These are for example, a smart watering system for plants. This would of course be ideal; the system seems to make decisions but bases those on one sensor; a simple conditional statement would suffice in activating the action the device was meant for. Unfortunately, most devices serve a purpose more complex than this. Most of the human wishes are based on more than one factor or can at least come from different factors. In lighting, a system based on just a light or motion sensor, would therefore be insufficient. Just measuring the amount of light in the room and acting on that would mean that the system turns on even when nobody is home. In the second case, a user's presence would not necessarily mean that the lights have to turn on.

3.3.2 Other implementations in home automation

Implementations other than *routines* (see 2.5) have been used in different systems. The Nest Learning Thermostat seems to be the most interesting and smartest way in home automation to satisfy the users wishes.

Turning our heads back to lighting, the question is whether it is feasible to implement a system like the Nest Learning Thermostat for this application. Looking further into this, applying this technology would not be feasible; for a learning algorithm, we'd need a trained model (often based on a big pool of historical data, ideally from more than one user). This model or the data pool I could train it on was not in my possession, neither was this available anywhere. Collecting this data myself was another option, but not feasible as this would mean that I ideally needed several candidates, provide those candidates with some way of collecting this data and they would have to have a smart lighting system already, was I to gather useful data on interaction with a lighting system. This meant I had to look for another solution.

3.3.3 Two-sensor solutions

I am going to use two relatively cheap sensors to determine the users presence and the current lighting situation. To combat the standard problem of having presets work for very specific cases, we don't define a setting for the lights, like a certain brightness level. We can use lighting conditions, detected by the sensor, at any given moment to learn a certain preference. By keeping this information in some register, the system can learn the users preference/ideal lighting conditions for different times throughout the day. With this information, the system will determine whether it should turn off, turn on, or dim the lights in any given situation.

The reasoning behind detecting the users presence is twofold. Collecting data on the users preferences is not worth it if the user is not present, as this will not necessarily be what the user likes. More importantly; the system should be able to turn off and on according to the users presence to save energy. Leaving the lights on while the user is gone of course defeats the purpose. The goal with this is to achieve similar categorization to machine learning, albeit with some more user interaction. This solution is simpler in its implementation than machine learning like Nest thermostats use. The (dis)advantages will be elaborated upon in a later chapter.

Chapter 4

Implementation

The point of this project is therefore to find a feasible way to make a smart lighting work for you. Mostly, it shouldn't notably give up any of the other features present in current smart lighting systems. It should be a solution to the light switch and the user's dependence on it and pose a smarter solution than routines currently do. This should be achieved in such a way that it is not too expensive, does not require a lot of computing power, and provides less work for the end user. The latter is in comparison to current smart lighting systems, while the former two conditions are already met by current lighting systems and, as stated in chapter 2 necessary, to be a viable alternative. With this in mind, I set out to make a system.

4.1 The workings of a two-sensor approach

This system works by checking whether the conditions, learned from the user, are met to turn on, turn off or dim the lights. This way, the user has almost no setup apart from a short learning period, but the system will always make a choice that most probably fits the users preference. The way we reach this is by defining a level of brightness needed in the room, defined by the user. Is the room getting dark, the lights should be getting brighter, in such a way that they make up for the deficit of natural light. The same holds vice versa. The easiest way to measure the amount of light in the room is in *Lux*. Lux is a measure of Lumens per square meter, which roughly translates to the amount of light perceived by the human eye. The amount of light can be measured by lux sensors.
To measure if the user is present, we can use a motion sensor. The motion sensor I chose to use is a passive infrared (PIR) sensor. This will act as our source for knowing whether a person is present in the room. It does this by detecting the passing of body with temperature. We assume that a person moves by at least once every 15 minutes when present in the room. Though this poses a shortfall of the system as lights will stay on for at most

15 minutes after leaving, this was not my immediate focus.

4.1.1 An example scenario of the system

An example of the working of this system is:

1. The user enters the empty room.
2. The system detects the users presence.
3. The system looks up in it's own database the preferred light level; say level x.
4. This light level is then compared to the light level currently in the room; say it is at $y \neq x$.
5. The system adjusts the brightness lights in the room up to get closer to light level x, and as long as:

   (a) The lights are not at maximum brightness;

   (b) x is not the light level present in the room.

   The system will keep going.
6. If either/both of these is met, the system will stop trying and keep the lights in their current state.
7. If no movement is detected for at most 15 minutes, either because the user didn't move, or the user has left the room, the lights will turn off.

4.2 Implementing the two-sensor approach

We will be creating this whole project in Python. This choice mainly came from the availability of the phue library [11] on GitHub, which was an easy way to get started with programming Philips Hue light bulbs. The choice for the Philips Hue light bulbs and its ecosystem is both because I have it available to me and because it is one of the most popular systems in smart home lighting solutions. The Philips Hue system only provides basic on/off and brightness control. All other functionality discussed in this section is functionality I have implemented myself. The system waits until movement is detected.

If movement is detected, the system reads the user's preference and compares it to the current light level in the room. In case those two differ, the lights will be turned on and adjusted to fit the users preference. Then, every minute, the system will check if the lighting situation changed. It does this for fifteen minutes.

If the lighting situation has not changed and everything is still according to wishes, the system doesn't change anything. If it has changed, the system checks whether the user did this or if the room's lighting itself changed (for example, the sun shines brighter). If it is the room's lighting, the system changes its lights according to the users previously known wishes. In this process, it waits for another five seconds to check for a false positive; it might have been a short flash of light. If the lighting is still the same, the system adjusts the lighting. If the user has intervened, the algorithm saves that as the new preference. When eventually no movement has been detected in the room, the lights are turned off.

The learning algorithm keeps track of the past 3 days that it has been used and uses it as a reference to determine the preferred amount of light at a certain moment. This is done by keeping the preferences saved in 1 hour blocks. For example, the light preference of 3PM to 4PM is used when the user is detected within that time frame. This also means that the system has to learn for the first three days of use and in those three days, is fully dependent on user input. If the user changes the lighting in that time frame, it is also saved to this same block and uses it for future scenarios.

If there is no information on preference available (as is the case when the system is not fully set up yet and in its learning period), it defaults to low light, both to at least provide some visibility in the room and to prompt the user to change the light, in turn giving the system a way to learn the users preference. I settled on three days, because I found that it was perfect for testing in an environment in which I was home most of the time and my weekends and weekdays were, most of the time, not that different. The mean of these 3 days is used to determine the preference of the user. This is the goal the system will try to reach if a reference is available. Of course, the user can always intervene by changing the light.

The whole project will run on a Raspberry Pi. The popular device provides an easy way to plug in sensors and program with those inputs, while also using low power [12] and being cost-effective. With this comes added value; if this works, it also proves that the system would be quite easy to produce for a relatively low price. This choice is further fueled by its availability to me as I have it laying around. The program should also be possible to run on an Arduino or ESP8266, further proving its affordability and reducing its energy consumption. But as stated above, the Raspberry Pi was easily available to me and ready to go, so we'll be using that.

4.3 The system

We need several components that:

- Save the user's preference and read that at any given time.

- Measure the current light level.
- Decrease and increase the brightness of the lights depending on the preference and current light level
- Detect the presence of a person.

The light level ranges are based on the following data from the National Optical Astronomy Observatory [13]:

Activity	Illumination in Lux
Public areas with dark surroundings	20 - 50
Simple orientation for short visits	50-100
Working areas where visual tasks are only occasionally performed	100 - 150
Warehouses, Homes, Theaters, Archives	150
Easy Office Work, Classes	250
Normal Office Work, PC Work, Study Library, Groceries, Show Rooms, Laboratories	500
Supermarkets, Mechanical Workshops, Office Landscapes	750
Normal Drawing Work, Detailed Mechanical Workshops, Operation Theatres	1000
Detailed Drawing Work, Very Detailed Mechanical Works	1500 - 2000
Performance of visual tasks of low contrast and very small size for prolonged periods of time	2000 - 5000
Performance of very prolonged and exacting visual tasks	5000 - 10000
Performance of very special visual tasks of extremely low contrast and small size	10000 - 20000

The paper mentions that on a clear day, the light level at the window is 1000 Lux. Keeping this as our maximum light level, the following categories are used in the program:

Category	Perceived maximum brightness	Lux Measured
1	5 %	0 - 2,5
2	10 %	2,5 - 10
3	20 %	10 - 40
4	30 %	40 - 90
5	40 %	90 - 160
6	50 %	160 - 250
7	60 %	250 - 360
8	70 %	360 - 490
9	80 %	490 - 640
10	90 %	640 - 810
11	100 %	810 - 1000
12	100+ %	1000+

These categories differ a bit from the table before. Because the human eye perceives brightness logarithmically, a doubling in Lux does not relate to a doubling in perceived brightness. Therefore, I have decided to use the function $P = \sqrt{M}$, where $P = Approximate\ perceived\ brightness\ in\ percentage$ and $M = measured\ brightness\ in\ percentage$, which is recommended by Steven's Law [14].

First of all, actual darkness is almost never present in daily life; there's some bit of light most of the time, which is why category 1 is not at absolute zero, but has some leeway for light. This is also why only in these first two categories, a step of 5 % is used.

All further categories differ 10 % from each other. This was sufficient. As a catch-all, everything above 1000 Lux is added, even though it is not feasible to expect non-natural light in a household to get this bright.

The lights are controlled by the Raspberry Pi, which in turn sends its commands to the Philips hue Bridge over WiFi. The sensors are connected directly to the Pi, so it can run the algorithm quite quickly. In this case, the algorithm uses 4 lights, which are the four Philips Hue lights in the room I've installed this in.

For the actual code and explanation of this code, I have to refer you to the appendix.

Chapter 5

Discussion

5.1 Summary

The goal of this paper was posed as follows: The point of this project is therefore to find a way to improve the users convenience by notably lessening their dependence on the light switch or rudimentary settings like *routines*. The way we do this is by giving the user a learning system and letting that system learn their preferences. Hopefully, this lessens the amount of times the user has to manually adjust their lighting.

I have made a learning system that, based on light level and the users presence, adjust smart lighting in the room. It does this by keeping a registry of the preferred lighting levels of the past three days. Based on this registry, a decision is made to dim, turn on or turn off the lights. To make this possible, I used a Raspberry Pi, a PIR sensor and a Lux sensor.

5.2 Evaluation

In my experience, less interaction with the lighting is necessary when the learning system is installed. In the current state of the system, after its initial learning process, I found myself only using the regular switch or app only to adjust my preference, which did not change too often. As such, interactions have reduced tremendously, to nearly none. The learning algorithm provided notable benefit in my experience of smart lighting and the convenience of it. Though it was only a single room, most of the time used by one person, the system reacted almost instantly to movement and the amount of light at that moment, getting to the saved preference in seconds. With more than one person in the room, the working of the system did not change. This is exactly as intended, as unfortunately the system can't distinguish different people. Leaving the room empty meant that the lights would turn off after at most 15 minutes, as is expected and intended. The amount of light the system would set in the was in congruence with the saved preference and in turn, that saved preference was in congruence with the preference I had set by adjusting the lights manually. As such, the main goal has been reached. Though it still felt a bit crude at times, being too exorbitant in its choices, most of the time when changing from one time-slot to the next (i.e. preference of 4-5PM to the preference of 5-6PM), overall it has been a good experience.

If the user decides the learning system is insufficient in its suggestion of your preference, they can change the lighting to their heart's content. This means that there are a few days to get the system used to you. That process can be a bit contradictory to the promise it makes, as there is a certain time where it requires your input. However, it does solve the problem I set out to solve in the long term.

Apart from reaching the main goal, I have achieved the following:

- Because the Raspberry Pi is a low-power solution in itself, the requirement for the system to be energy efficient is achieved. This was important, as a big selling point of smart lighting systems is their potential to save energy.
- The user is not bothered with more work or a tough setup by the system. Apart from some initial setup, the system works on its own.
- The system is relatively cheap because of its use of parts; a Raspberry Pi, a PIR sensor and a light sensor make for a total of about $ 50.
- The light switch or an app is still a part of the user's interaction with their smart lighting system, but a lot less when the learning system has had a few days to learn the user's preferences.

5.3 Future work

Although the system does what it is supposed to do, there are still some improvements I'd like to see.

First of all, a PIR sensor is an easy solution to detect user presence. It is rather rudimentary; an animal can activate it and if a person does not move for a while, that person will not be detected at all. In the same rudimentary spirit, three days of logging was a nice way of testing, perfect for my use, being at home most of the time. However, it would probably be more elegant and better fitted if it logged per day instead of the past three days. Also, preferences could have some better way of saving. Rather than per hour, every 15 minutes would allow for better control and adjustment. This would allow for finer control, especially in scenarios where light quickly reduces, for example at sunset. This would help in the situation described in the previous section, where changing between time-slots could mean quite a difference. Currently in these circumstances, the system feels like it is lagging behind just a bit.

Also, I'd like to see the preferences of the user be dependent on more than just the current light level and time of day. Weather conditions and some kind of activity detection (a user watching TV would probably prefer a different light setting than a user working behind their desk), maybe possible by Episode Discovery as mentioned by Heierman and Cook [7] or by Ambient Intelligence [9], would be two of the first things that come to mind. This would make for an even more intelligent system, although it would probably increase cost and computing power.

Another issue as of right now is that the system compares every minute. This means that it does not immediately react on the changing light in the room. Though this could be a disadvantage, a constantly fluctuating lighting system can be distracting. Therefore, every minute seems like a reasonable compromise. In a newer version, I hope to make use of a multi-threaded approach and keep track of a few seconds (say about ten seconds) at a time and respond to the measurements during that time by adjusting the lights, all while keeping track of user input.

As a last improvement, a way to adjust lights separately and for a way to save that preference (for example, very often people have ceiling lights as their main lighting and one or more table/floor lamps for mood lighting).

Bibliography

[1] J. J. Greichen. Value based home automation for todays' market. *IEEE Transactions on Consumer Electronics*, 38(3):XXXIV–XXXVIII, 1992. http://ieeexplore.ieee.org.ru.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=156666&isnumber=4057.

[2] K. Wacks. Home systems standards: achievements and challenges. *IEEE Communications Magazine*, 40(4):152–159, 2002. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=995865&isnumber=21492.

[3] Aaron Souppouris. Philips hue customizable led light bulbs offer 16 million colors, starter kit costs $199, Oct 2012. https://www.theverge.com/2012/10/29/3570322/philips-hue-led-customizable-bulb-apple-store-home-automation.

[4] Nathan Ingraham. Google purchases nest for $3.2 billion, Jan 2014. https://www.theverge.com/2014/1/13/5305282/google-purchases-nest-for-3-2-billion.

[5] Jacob Poushter. Smartphone ownership and internet usage continues to climb in emerging economies, Feb 2016. https://www.pewresearch.org/global/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/.

[6] Rayoung Yang and Mark W. Newman. Learning from a learning thermostat: Lessons for intelligent systems for the home. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, page 93–102, New York, NY, USA, 2013. Association for Computing Machinery. https://doi-org.ru.idm.oclc.org/10.1145/2493432.2493489.

[7] E. O. Heierman and D. J. Cook. Improving home automation by discovering regularly occurring device usage patterns. In *Third IEEE International Conference on Data Mining*, pages 537–540, 2003. http://ieeexplore.ieee.org.ru.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=1250971&isnumber=27998.

[8] Li Jiang, Da-You Liu, and Bo Yang. Smart home research. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 2, pages 659–663 vol.2, 2004. `https://ieeexplore-ieee-org.ru.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=1382266&isnumber=30108&tag=1`.

[9] C. Ramos, J. C. Augusto, and D. Shapiro. Ambient intelligence—the next step for artificial intelligence. *IEEE Intelligent Systems*, 23(2):15–18, 2008. `http://ieeexplore.ieee.org.ru.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=4475854&isnumber=4475846`.

[10] Hue motion sensor 046677473389 — philips. `https://www2.meethue.com/en-us/p/hue-motion-sensor/046677473389`.

[11] Studioimaginaire. studioimaginaire/phue. `https://github.com/studioimaginaire/phue`.

[12] Raspberry Pi Foundation. Power supply, 2014. `https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/`.

[13] National Optical Astronomy Observatory. *Recommended Light Levels (Illuminance) for Outdoor and Indoor Venues.* OAO, 2015. `https://www.noao.edu/education/QLTkit/ACTIVITY_Documents/Safety/LightLevels_outdoor+indoor.pdf`.

[14] Stanley S Stevens. On the psychophysical law. *Psychological review*, 64(3):153, 1957.

[15] Kyletaylored. kyletaylored/python-tsl2591, Jun 2019. `https://github.com/kyletaylored/python-tsl2591`.

[16] configparser - configuration file parser, 2019. `https://docs.python.org/3/library/configparser.html`.

Appendix A

Appendix

A.1 Codebase

A.1.1 hue.py

Adjusting the lights' brightness is done in hue.py; the function turns on the lights, gets the current brightness from one of the four lights and iterates through the loop until the desired light level is reached.

The brightness is always adjusted on all (in this case) four lights at the same time. If the brightness is at its maximum, the loop quits, as it can't do anything more. If the brightness level is at its minimum and the preferred light level still has to go down, it turns the lights off. The commented code at the top is for first-time setup, courtesy of the phue [11] library mentioned above, just like all other code used to change the Philips Hue light bulbs.

```
from phue import Bridge
from luxcategory import get_category

b=Bridge('192.168.1.37')
MAX_BRIGHTNESS=254

#b.connect()

#b.get_light(1, 'on')
#b.get_api()

def set_brightness (wish):
    goal = wish
    lux = get_category()
    b.set_light( [1,2,3,4] , 'on', True)
    current = b.get_light(1, 'bri')
    while (lux != goal):
        lux = int( get_category() )

        if (lux < goal and current == MAX_BRIGHTNESS):
            break
        elif (lux < goal):
            current+=1
            b.set_light( [1,2,3,4] , 'bri' , current )
        elif (lux > goal and current == 0):
            b.set_light( [1,2,3,4] , 'on', False)
            break
        elif (lux > goal):
            current-=1
            b.set_light( [1,2,3,4] , 'bri' , current )

def turn_off ():
    b.set_light( [1,2,3,4] , 'on' , False )

def get_hue_setting ():
    return b.get_light(1, 'bri')
```

A.1.2 lux_category.py

The code in lux_category.py makes use of the TSL2591 Python Library [15] which is based on the Arduino library the sensor was initially made for.

```
from python_tsl2591 import tsl2591

tsl = tsl2591()

def get_light_level():
    full, ir = tsl.get_full_luminosity()
    lux = tsl.calculate_lux(full, ir)
    return lux;

def get_category():
    lux = get_light_level();
    if lux < 2.5: return 1;
    elif lux <  10: return 2;
    elif lux < 40: return 3;
    elif lux < 90: return 4;
    elif lux < 160: return 5;
    elif lux < 250: return 6;
    elif lux < 360: return 7;
    elif lux < 490: return 8;
    elif lux < 640: return 9;
    elif lux < 810: return 10;
    elif lux < 1000: return 11;
    elif lux >= 1000: return 12;
    else: return 0;
```

A.1.3 pir.py

Everything in the end is called upon by pir.py: this script runs when it detects movement, and then calls upon the components to take care of the necessary steps.

It only changes the light level when it is not yet on the preferred level, saving the time and resources needed for this operation. If it detects that the light level in the room is not according to the users wishes, it will compare again after five seconds to make sure it wasn't a short light flash and change the lighting in the room. When it is detected that the brightness of the lights has been changed outside of the program, it assumes the user did this and saves the new preference. Most PIR sensors have several hardware settings to also accommodate for different delays (time that it reports active after detecting movement), sensitivity and others. This can be interesting in a lot of scenarios, but is beyond the scope of this paper.

```
from gpiozero import MotionSensor
from hue import turn_off, set_brightness, get_hue_setting,
    lights_on
from pref_parser import read_pref, write_pref
from luxcategory import get_category
import time
import statistics

pir = MotionSensor(4)

def on_motion ():
    wish = read_pref()
    level = get_category()

    if (wish != level):
        set_brightness(wish)

    base_hue_setting = get_hue_setting()

    for x in range(15):
        time.sleep(60)
        level = get_category()
        current_hue_setting = get_hue_setting()

        if (wish != level and base_hue_setting ==
            current_hue_setting):
            time.sleep(5)
            if (wish != level):
                set_brightness(wish)
                base_hue_setting = get_hue_setting()
        elif (base_hue_setting != current_hue_setting):
            level = get_category()
            new_pref = level
            write_pref(new_pref)


pir.when_motion = on_motion()

pir.when_no_motion = turn_off()
```

A.1.4 pref_parser.py

The preferences are managed with the python configparser [16] library. A config.ini file is created, read, and updated through pref_parser.py;

The day in the year is calculated and its remainder (0, 1 or 2) after modular operation is used to create an entry for the preference in the past 3 days. If none exist at the moment of reading, a default of light level 3 is used. When writing to the config.ini file, when a section does not exist yet, it fills entries of all three days with the light level at that moment. When reading, and a section does exist, the average of the past three days is used (rounding through int() function, which truncates the value).

```
import configparser as cfg
import datetime
import statistics

config = cfg.SafeConfigParser()
config.read('config.ini')

def read_pref():
    try:
        now = datetime.datetime.now()
        hour = now.hour
        tuple_days = (config.getint(str(hour), '0'), config.
            getint(str(hour), '1'), config.getint(str(hour), '2')
            )
        wish = int(statistics.mean(tuple_days))
    except Exception as e:
        wish = 3
    print(wish)
    return wish

def write_pref(wish):
    print(wish)
    now = datetime.datetime.now()
    hour = now.hour
    day = now.timetuple().tm_yday
    pref_day = day%3
    try:
        config.set(str(hour), str(pref_day), str(wish))
    except cfg.NoSectionError:
        config.add_section(str(hour))
        config.set(str(hour), '0', str(wish))
        config.set(str(hour), '1', str(wish))
        config.set(str(hour), '2', str(wish))
    with open('config.ini', 'w') as f:
        config.write(f)
```