UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

FINAL DEGREE PROJECT

Energy Engineering

HOME SWEET HOME: A HOME AUTOMATION APP WITH MACHINE LEARNING

Report

Autor: Francesc Cabana i Comas
Director: Samir Kanaan Izquierdo

Convocatòria: Gener 2018

Abstract

The presence of the digital world in our lives is increasing more and more every day. We live in a time that could be considered as a transitional between the analogical and digital worlds. The number of sectors affected by this transformation is growing, and the home is not an exception. The purpose of the present paper is to elaborate a simple, at the reach of any engineer, system to automatize our home. This is, to be able to control the lights, blinds and heating of any house with the only help of a smartphone.

Therefore, with this task in mind, a mobile Android app and the corresponding server are designed in order to maintain the communication with the house microprocessor, which will be responsible to transfer the commands to the physical elements of our home. In addition, with the intention of making our life easier, the app will be able to learn about the tastes and habits of the user and eventually manage the home without the interaction of the owner.

In the following paper we will see that the ability to control our home with our smartphone or even not have to worry at all because it is self-controlled, which could seem like a luxury, is something possible, not very expensive and energy efficient.

Resum

El món digital cada vegada té més presencia en les nostres vides. Vivim uns temps que podríem considerar de transició entre el món analògic i el digital. Cada vegada més sectors pateixen aquesta transformació i la nostra llar no n'és una excepció. Amb el present treball es pretén elaborar un sistema senzill i a l'abast de qualsevol enginyer per automatitzar la nostra llar, és a dir, per ser capaços de poder controlar els llums, les persianes i la climatització de casa, tan sols, utilitzant el telèfon mòbil.

Amb aquest objectiu doncs, es dissenya una aplicació mòbil per Android i el pertinent servidor per tal de mantenir la comunicació amb el microcontrolador, que serà l'encarregat de transmetre les ordres als elements físics de la llar. A més a més, amb la finalitat de fer-nos la vida més fàcil, es busca que l'aplicació sigui capaç d'aprendre per ella mateixa dels gustos de l'usuari i de gestionar la llar sense que el propietari se n'hagi d'ocupar.

Durant les següents pàgines, veurem que, el què podria semblar un luxe, com és poder controlar la nostra llar amb un telèfon mòbil o fins i tot, no haver de preoucupar-nos de controlar-la, ja que es regula tota sola, és possible, no tan car i eficient energèticament.

Resumen

El mundo digital cada vez tiene más presencia en nuestras vidas. Vivimos unos tiempos que podríamos considerar de transición entre el mundo analógico y el digital. Cada vez más sectores sufren esta transformación y nuestro hogar no es una excepción. Con el presente trabajo se pretende elaborar un sistema sencillo y al alcance de cualquier ingeniero para automatizar nuestro hogar, es decir, para ser capaces de poder controlar las luces, las persianas y la climatización de casa, tan sólo, utilizando el teléfono móvil.

Con este objetivo pues, se diseña una aplicación móvil para Android y el pertinente servidor para mantener la comunicación con el microcontrolador, que será el encargado de transmitir las órdenes a los elementos físicos del hogar. Además, con el fin de hacernos la vida más fácil, se busca que la aplicación sea capaz de aprender por sí misma de los gustos del usuario y de gestionar el hogar sin que el propietario tenga que ocuparse directamente.

Durante las siguientes páginas, veremos que, lo que podría parecer un lujo, como es poder controlar nuestro hogar con un teléfono móvil o incluso, no tener que preocuparse para controlarlo, ya que se regula solo, es posible, no tan caro y eficiente energéticamente.

Acknowledgments

I would like to convey my heartfelt gratitude and sincere appreciation to all people who have helped and inspired me before and, above all, during my project. My deep gratitude to my professor, Samir Kanaan, for their attentive guidance, endless patience and encouragement. He has provided me professional knowledge on Informatics Engineering and has shared the experience.

Sincere gratitude should be given to my dears, to D.V. who has given me the strength with her company and moral support during the tough times; and also especially to À.A. who has given me the light in times of darkness with his wisdom. My gratitude also goes to J.M. who permanently is willing to help you with everything and always has or knows what you need. Without all of their support and consideration, this project would not have reached its completion.

Last, but not least, I am deeply appreciate my family, specially to my sister Judit, whose belief in my ability encouraged me to overcome all obstacles and continue to reach for the best.

Summary of Contents

1. Preface

1.1. Origin of the project

The idea of this project was born thinking big. Mixing the concepts of artificial intelligence and home automation the idea comes naturally, effortlessly. This is how the idea came to be.

This is a bold project, but the idea of mixing these two concepts can be even more daring, and it will be. All of us will live with it in a few years, we do not know exactly when, but we know it will happen. It is unavoidable. It is called evolution.

At this point, I wondered if it would be possible to create a system that controlled the home by itself and if it would be expensive.

With that in mind, the project started.

1.2. Motivation

The desire to work and discover more about the field of programming, both mobile applications and the purest programming, artificial intelligence and the IoT concept were the main motivation to carry out this project.

To thing about the future that these fields will have on the next years, along with the challenge of doing it without having had a formation focused on these fields, finished motivating me to choose a project of these dimensions.

In addition, we also must consider the interest to seek a better lifestyle and a way to try to automate energy savings.

Therefore, the motivation of this work has been none other than the challenge of trying to do something attractive and functional from the received formation.

1.3. Previous requirements

To carry out this project, knowledge regarding the basics of the Python, Javascript and C++ programming was required. In addition, it was necessary to known the basics of electronics to configure the microcontroller to elaborate the model presented on the defence of the project.

2. Introduction

2.1. Objective of the project

As we have been mentioning, the main objective of this project is not another than: design and develop a mobile app that can control the lights, the blinds and the heating of a house by itself considering what the owners want in every moment.

We want to extend the intelligence of home automation by making your home so smart that it can understand exactly what you need before you ask for it.

With the purpose of achieving the objective, several tasks have been carried out along the project. The different goals to accomplish the main objective are:

- Develop an intuitive and user-friendly app that lets the user control a model.
- Develop a server that can be the intermediary between the user (the app) and the home (the microcontroller).
- Program the microcontroller to be able to execute all the functions we need to control a house.
- Make the app capable of learning from the user and able to control the house itself. To make it we will have to choose the right type of machine learning.
- Make an interesting project that allows, if there is interest, start something bigger, a real market alternative.

2.2. Scope of the project

This project would be amazing if it made possible to develop a real market alternative which was able to control real homes. However, we are realistic and that is not possible to build only in a few months and doing it all by only one person. For that reason, we want to develop an app that be able to control a model.

About the learning part of the app. It would be great to make a perfect prediction system, but with the formation received and with the time we have and the complexity of the task, we will comply with a system that can be able to predict behaviours.

3. State of the art

Before diving deeper into the project itself, it would be good for the comprehension of this memory to analyse and to make an approach to some important topics that surround the project.

Internet has revolutionized the world, as we knew it. Nowadays the technology is more advanced and it allows us to do more and more things. Moreover, with the advancement of technology everything is easier than before. Therefore, Internet allows us to do many new things and allows us to do them easier than never.

In the same way, Internet and the advanced technology of today have allowed us to talk about IoT (Internet of Things), in other words, to connect usual objects to the Internet.

Definitely, Internet is changing the world, making it easier than never. It is transforming absolutely all we know. It is affecting in all sectors of our life. One sector that cannot be an exception is our home.

The digital revolution has started.

3.1. Home Automation

3.1.1. What is Home Automation?

We use technology more and more every single day. From activating security systems to climate control, even turning off the lights at night. In these fast-paced times, streamlining the technology in our lives is not only convenient, it is necessary.

Imagine turning your lights off, locking the front door, activating your security system, adjusting the thermostat and lower the blinds. Now imagine doing all of those things in just a few seconds using your smart phone right before going to sleep. With home automation, you will arrive home to a comfortable temperature and before going to sleep you and your partner, will not have to decide who has to get up to turn off the light after a long day.

Home automation, includes the control and automation of lighting, heating (such as smart thermostats), ventilation, air conditioning (HVAC), and security, as well as home appliances such as ovens, refrigerators or washer. Wi-Fi is frequently utilized for remote monitoring and control. Most modern systems are comprised of switches and sensors connected to a central hub, from which the system is managed by a user interface that is interacted either with an app, personal computers, laptops, touch pads, smartphones and so on.

The hub receive the information from sensors, and based on the user instructions, control the actuators. Actuators are the final controlling devices like limit switches, relays, motors and other controlling mechanisms, which finally control the home gadgets. Communication plays a vital part in this home automation system for the remote access to these operations.

3.1.2. Types of Home Automation Systems

- **Power Line Home Automation System**

This automation is not very expensive and does not require extra cables to transfer the data, however, utilizes existing power lines to exchange the information. Be that as it may, this system involves a large complexity and requires extra converter circuits and devices

- **Wired Home Automation System**

In this sort of automation, all the home equipment is connected to the central controller through a communication cable. The equipment is attached with actuators that would grab data, to transfer it to the central controller. The entire operations are centralized by the computer that continuously communicates with the central controller.

- **Wireless Home Automation**

This is the advancement and natural evolution of wired automation, which utilizes wireless technologies like Wi-Fi, Bluetooth, GSM, IR, Zigbee, etc., for achieving remote operation.

Figure 3.1. Wireless Home Automation System sketch (Source: iotnewsportal.com [1])

3.1.3. Alternatives available on the market

Today, there is nothing exactly to what this projects proposes consolidated in the home automation market. What the projects pursues, is not something that anybody has thought of before. In fact, there are people on the Net who claim to have developed something similar for their houses. Moreover, there are a few interesting projects, as *Mindis* [2] by Archiuse that are working in the same direction as this project, but they are at an early stage checking if the product fits the market. However, the truth is that nowadays there is nothing exactly available on the market.

The most similar product is the Nest Learning Thermostat. It learns what temperature you like and builds a schedule around yours, but only does it with the temperature.

What is easier to find today in the market, are various home automation systems with AI that allows users the capacity to control their homes speaking to it or remotely with a smartphone, like Amazon Alexa. However, in other words, nowadays there are many products similar to the manual mode that we want to implement in this project, but still there is nothing exactly like the automatic mode of the project.

Nevertheless, the truth is that it is a matter of time to see home automation alternatives with machine learning. It is also a matter of time until we live with it in our homes. The one who has advantage in this race is Google's Nest, followed by Jarvis of Facebook, Amazon Alexa and Apple's Siri.

3.2. Artificial Intelligence

First concept that is an important part of the project, which would be interesting to introduce, before start talking about the project itself is this area of computer science.

3.2.1. What is Artificial Intelligence?

Artificial intelligence (AI) is for sure one of the most common subjects in all of science fiction. The possibility that a machine could show a similar level of intelligenceas as a human has impassioned the whole world of literature alike for a considerable length of time.

Sci-fi has a propensity for romanticizing certain things, for example, the successive incorporation of humanoid robots. Building a robot in quest for AI is somewhat similar to developing the chassis of a motorbike before the internal combustion engine was even imagined. It is clear that the human form is only an easier solution than expecting that people should get candidly connected to some lines of code.

Be that as it may, the truth is that AI is one of the newest fields in science and engineering. Work started in earnest soon after World War II, and the name itself was coined in 1956.

Nowadays artificial intelligence is already making our lives easier in many areas today. Whether it's the voice assistant on our smart phones, perfect product suggestions in an online shop or predictive maintenance of trains [3]. But what is actually artificial intelligence?

Artificial intelligence is nothing but software that, automatically gathers and analyzes data and we are already familiar with that. For example, from our Facebook newsfeed which adjusts automatically to our interests. The algorithm takes flood of data, filter out what's most likely to interest us and just play it for us at the top of the page.

But that's not the end of it. The next steps are nothing less than mimicking the human brain with computers, in other words, machines learn to think and act as we humans do. The prerequisite for that will be the Internet of Things, meaning that all machines are networked together and produce data that will yield a vast quantity of data that will have to be analyzed. It will take immense computer capacity to make that possible and that kind of capacity has not been available until now, but as more and more powerful computers are developed, artificial intelligence is making great strides. Machines are actually beginning to learn.

As an example **[4]**, Go master Lee Sedol was defeated (3-0 in a best-of-five competition) in 2016 by google's AlphaGo software program. Go, the Chinese board game is considered to be a much more complex challenge for a computer than chess. In fact, up to that time, Go has been thought to be too complicated for machines to play. But using artificial neural networks imitating human thoughts through algorithms led to breakthrough.

AlphaGo can analyze data to learn on its own. The system played millions of games against itself and analyzed those games. That knowledge let AlphaGo to always keep a step ahead of a human opponent: the program was able to calculate 57% of all of the Sedo's moves in advance and this beated the world champion.

Artificial intelligence is also on the march in business. The potential is huge, activities that work and pattern actions can potentially also be taken over by artificial intelligence. Comprehensive automation industry is expected to bring tremendous increases in efficiency. For example, in production, corporate consultant Accenture projects **[5]** that by 2035 the use of artificial intelligence will help the economy in the United States grow 4.6% a year compared to the base scenario which projects growth rate of only 2.6% a year.

And what will happen to our jobs when robots take over as our colleagues? In order for us to benefit from artificial intelligence, government, business and society will have to organize the necessary

changes together, because experts agree, over the next 20 years artificial intelligence will fundamentally change our economy and our way of working. In coclusion, artificial intelligence will change our way of living.

To conclude, AI currently encompasses a huge variety of subfields, ranging from the general (learning and perception) to the specific, such as playing chess, proving mathematical theorems, writing poetry, driving a car on a crowded street, and diagnosing diseases. AI is relevant to any intellectual task; it is truly a universal field.

3.2.2. Machine Learning

AI and machine learning (ML) are very related, but they are not quite the same thing.

Machine learning is a subset of AI. That is, all machine learning is considered AI, but not all AI is considered as machine learning. For instance, symbolic logic – rules engines, expert systems and knowledge graphs – could all be defined as AI, and none of them are machine learning.

Machine Learning is a kind of artificial intelligence where we do not specify standards or rules to produce intelligence rather we will make algorithms that can learn from information (data). In ordinary programming, we code a logic and pass it an input and the program generates the output. In Machine learning, we will give the system a set of data that is related and the program will produce code for matching these inputs to output. Once that is done, we can utilize the system to calculate outputs with more input.

Simplifying, we could say that machine learning is the ability for a computer to output or do something that it was not programmed to do, using probability to make decisions or predictions about data with a reasonable degree of certainty. Therefore, machine learning is teaching machines to make decisions in situations they have never seen.

We can separate learning problems in a few large categories:

- Supervised learning, in which the data comes with an input and an output for training a model. Therefore, for a specific input, we already know how the correct output should looks like and we think there is a connection between them. This problem can be either:
    - Classification: samples belong to two or more classes and we want to learn from already labelled data how to predict the class of unlabelled data. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, one is to try to label them with the correct category or class.

    - Regression: if the desired output consists of one or more continuous variables, then the task is called *regression*.
- Unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

There is a metaphor [6] really interesting and that helps to understand better the supervised learning (the one we are going to use in this project). Reads as follows.

When learning to play the trumpet, you're taught finger positions to play notes. That's supervised AI. Learning notes is training the model. How well you, the human machine, processed the data set of "how to play notes" determines how well you play in all the new, foreign situations (playing WITH other instruments, playing different music, different tempos … and more!).

The next image is going to help to comprehend a little more the difference between the ideas of Artificial Intelligence and Machine Learning. Moreover, the next figure is going to introduce a new concept, not as important for this project as it is machine learning, but that is interesting to know it too.

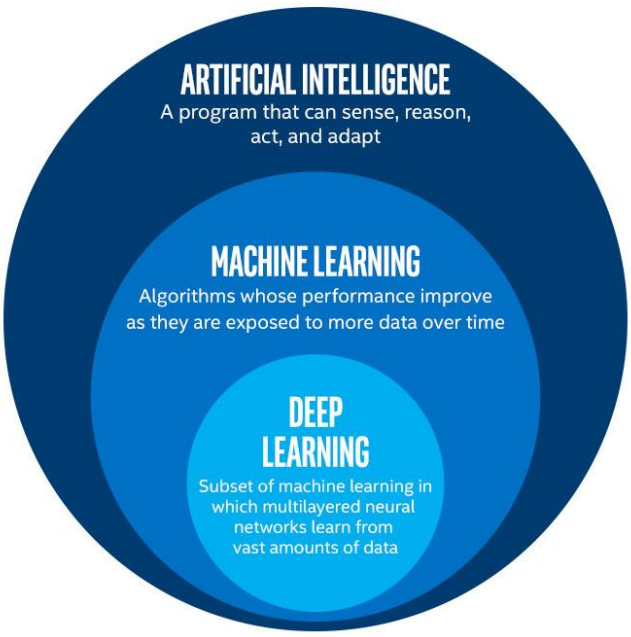


Figure 3.2. Artificial Intelligence is an umbrella term, encompassing machine learning and deep learning (Source: Intel [7])

Deep learning, for its part, is a subset of machine learning where artificial neural networks — algorithms that simulate the human brain — discover patterns in raw data by consolidating different layers of artificial neurons. As the layers increment, so does the neural network's capacity to learn progressively abstract concepts.

To finish, it would be interesting to compare the evolution of this three concepts that we just have seen, artificial intelligence, machine learning and deep learning. To do that, we are going to compare the number of searches on Google in the last 10 years.

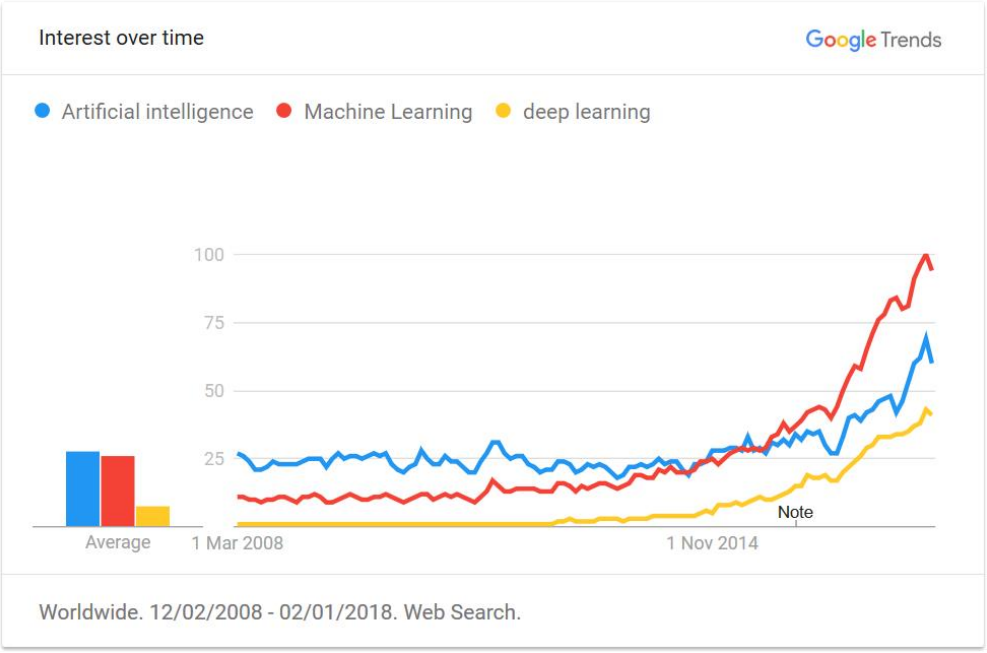


Figure 3.3. Artificial Intelligence vs Machine Learning vs Deep Learning (Source: Google Trends)

To clarify, numbers on the y-axis represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular.

From the figure, we can see that Machine Learning is growing exponentially last years and it is getting a lot of attention. It is also relevant the evolution of Deep Learning in the last 5 years.

3.2.2.1. K-nearest-neighbor algorithms.

In this project, we will use this machine learning algorithms. Let us explain it:

The k-nearest neighbors algorithm (k-NN) is a method that can be used for both classification and regression predictive problems. It is one of those algorithms that are very simple to understand but works incredibly well in practice. In addition, it is surprisingly versatile.

- In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, odd and typically small). Is one of the most fundamental and simple classification methods.
- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

The following figure is an example that explains how it works the k-NN classification method:

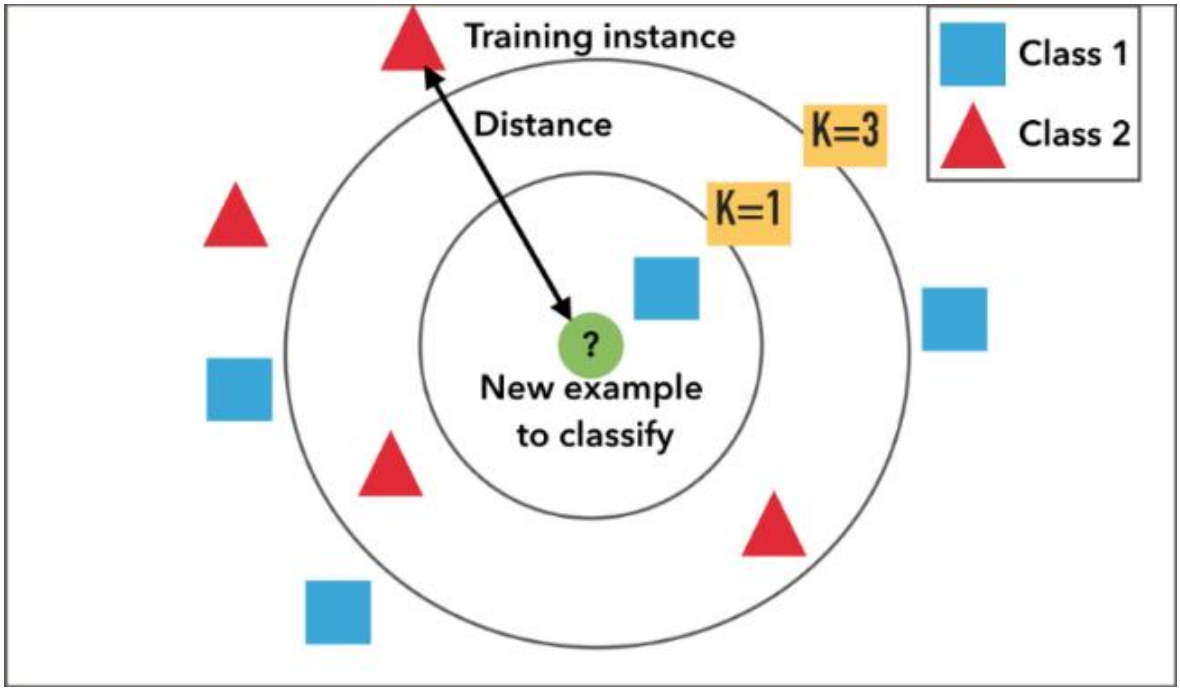


Figure 3.4. Example of k-NN. (Source: techyv.com [8])

About the example of the figure. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If *k = 3* (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If *k = 5* (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

3.3. Internet of Things

3.3.1. What is the IoT?

How many objects we have that are connected to the Web? About 10 years back most of us possibly would have said one or two.

Be that as it may, today, we got a smartphone, a laptop, a work PC, a tablet, a smartwatch, even a smart TV and so on. All this objects, equipped with sensors, are taking data from real world and uploading it to the Internet, that is the Internet of Things (IoT). It is a technology, through which devices can interact with each other.

3.3.2. The huge future it has and the real reason I choose this project

At this point, taking into account all that we have said until now about AI, ML and IoT separately, we are going to look at the potential of the three concepts combined.

The Internet of Things gives us an immensive leverage, because we can collect a lot of diferent kind of data extending intelligent sensors into everything, into the world of everyday objects. They are going to be endowed with the ability to give us feedback, to give Machine Learning systems vast amounts of data.

With this combination The world is going to become intelligent and responsive and going to anticipate our needs. The world is going to feel like an extension of our mindedness. David Rose in his book "Enchanted objects" [9] essentially says that we're moving into a world of animism where objects have a kind of consciousness and that things almost are enchanted, almost are alive. Imagine you walk into a room and the room knows how you like the lighting and the song that you love starts automatically playing and the curtains are automatically raised, etc.

The full flourishing of these technologies promises to essentially blur the distinction between self and world. We will have fully spread our minds out into our universe.

This is the Internet of Things.

This is why it is a game-changer, this is why it absolutely rattles my imagination and this is why I was seduced by this project.

In this regard, it is clear that the number of smart home devices sold arround the world is growing everyday. According to BI Intelligence [10], the number of smart home devices shipped will grow from 83 million in 2015 to 193 million in 2020. This includes all smart appliances (washers, dryers,

refrigerators, etc.), smart home safety and security systems (sensors, monitors, cameras, and alarm systems), and smart home energy equipment, like smart thermostats and smart lighting.

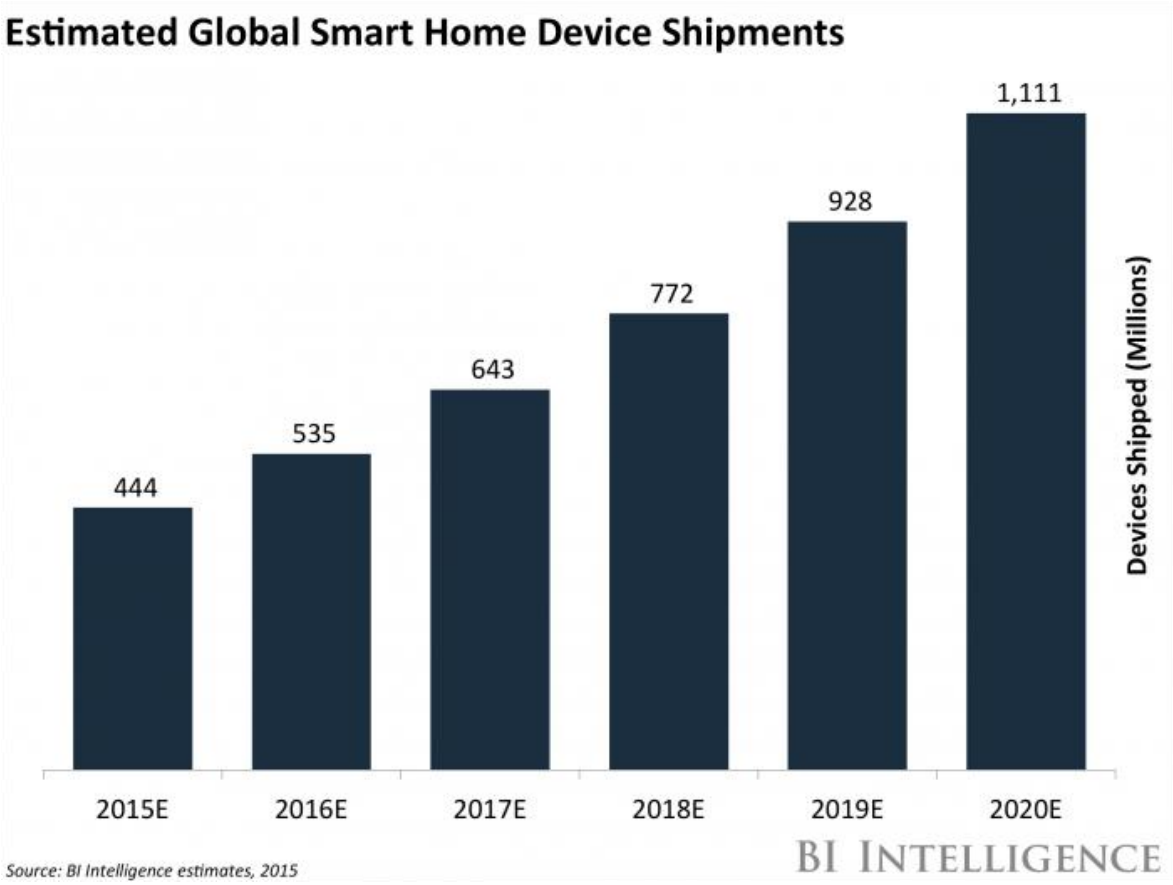


Figure 3.5. Estimated Global Smart Home Device Shipments (Source: BI Intelligence [10])

4. *Vesta*. The project.

4.1. Introduction

Vesta wants to be a home automation application that incorporates machine learning. It allows the user to control a smart house manually with a smartphone, or automatically, without the attention of the owner.

Therefore, *Vesta* extends the intelligence of home automation by making your home so smart that it can understand exactly what you need before you ask for it.

The software uses artificial intelligence to learn from your lifestyle in order to control smart devices automatically. Each time you interact with lights, blinds or with the heating system, *Vesta* collects data about that interaction, analyses it and adapts software accordingly.

In this project, all the information related to the process of creating the alpha version of the app can be found. *Vesta*'s first version has been developed with the objective to control a prototype, and not a real home. It is just a first step, something to start with, to take the idea to the next level and, if it is considered, release a final app, which would enable users to control a real home, their home.

Figure 4.1. Vesta, Roman goddess of the hearth, home, and family (Source: Wikipedia)

Vesta is the virgin goddess of the home in Roman religion. She is the origin of the app's name, because as the Roman goddess, *Vesta* wants to become the goddess of your home, controlling your house by itself and knowing what you want in every single moment.

4.1.1. Architecture

The project can be divided into three large blocks:

- The mobile app. What the users will see and with what they will interact. It has been developed with Qt Creator.
- The server. It has its own PostgreSQL database (DB) and has been coded in Python (using Flask).
- The microcontroller that receives the instructions from the server and executes them. The microcontroller that has been chosen is a Particle Electron.

The three blocks are related between them as follows:

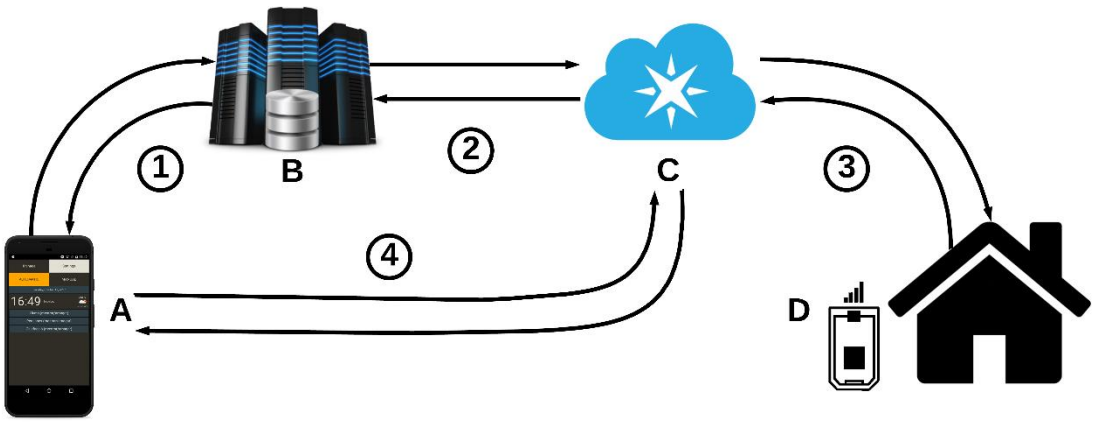


Figure 4.2. Architecture of the app.

This figure above is a diagram of the architecture of the app. It has just been said that the project consists of three large blocks, but in the figure, appear four icons. Let us comment the diagram and we will understand better why this happens.

First of all, let us talk about the four big entities (each one represented with a capital letter) we see on the image. The *A* entity is the mobile app itself, after this first recognition of the entities we will analyse the relationships between them. The *B* icon is the main server and the database we had to create for achieving our objective. The *C* element is the Particle cloud, the server of the microcontroller that will allow us to program and control it. Last, but not least we have the *Electron* that will be in the house that needs to be controlled.

If, now, we look at the relationships (each one represented with a number) we could see that there are four big relationships. Relationships represent exchange of data, so, all four relationships are

bidirectional, because if one entity asks some information to another entity, this last entity has to reply and send the information back.

The relationship number one (1) is the one that links the mobile app with the server and the database. The vital relevance of this connection lies, basically, in being the one that allows the login into the app and provides the information of the user orders to the server.

The relationship number two (2) is the intermediate step to connect with the *Electron* and give it an order or ask it anything. As always, the relationship is bidirectional, but in this case, sometimes, when we want to order anything to the microcontroller, the only relevant connection in this relationship is the one that goes from *B* to *C,* because the other direction is only to confirm that all is OK. Other times, when the server wants to ask something to the *Electron* instead of give it an order, the *C* to *B* connection is important too.

The third relationship (3) is the one where the microcontroller receives the order and, as in the last relationship, if the *Electron* receives and order, it only replies with an affirmative or a negative response. However, if the microcontroller receives a request, like could be, a request for the current temperature, it replies with the right answer.

The last relationship, the fourth one (4), is the one that links the *Particle* cloud, with the app. That means that the app could "speak" with the *Electron* bypassing the server, but the truth is that this connection is only used to refresh the app with the new temperature of the room. This request could be made through the server too if the developers consider it.

To finish the analysis of the architecture of the app, it would be interesting to note that there is another connection, a connection not as important as the others and, for that reason it does not appear on the figure, but it is interesting to mention it. This relationship is the one that links the app and, also the main server, with the server where *Vesta* checks the weather. To check it, *Vesta* uses the *OpenWeather*'s API [11].

4.2. The app: *Vesta*'s face.

4.2.1. Appearance

4.2.1.1. The logo

The first thing you see of an app is its logo. For that reason, it is necessary that your application has an attractive logo and if it is possible, that gives some extra information about its content to the user. In this case, *Vesta* logo is a *V*, as we will see in the next section, of the colours of the app. Moreover, in the same *V* we can see a schematic of the app's main screen.

Figure 4.3. The *Vesta* logo

4.2.1.2. Screenshots

4.2.1.2.1 LOG IN

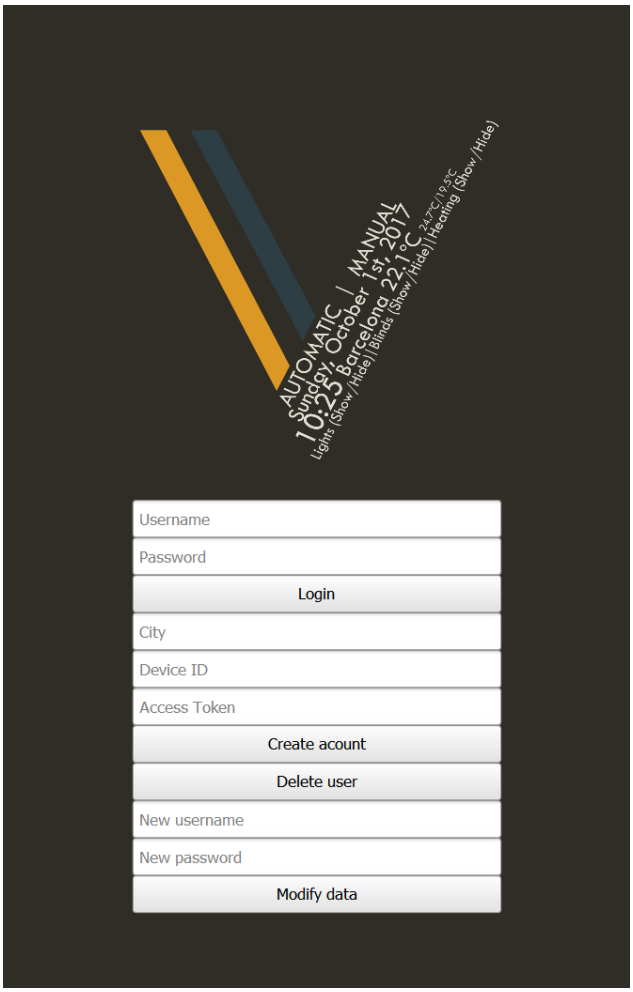


Figure 4.4. The Log In screen

This is the first screen that pops up upon opening the application. It is where the user can log in to the application with his Vesta account, or register for one if he is new to the app.

Whenever a home info is entered, the server checks whether an existing house (with a unique device ID) exists or not. It then proceeds to associate the user to that house, or create a new entry for that device id.

In addition, an existing account can be eliminated or existing data associated with a username or a house can be modified.

It is important to remember that the app is in an alpha state, which is why the app can be adapted to be more intuitive, but it is functional, and that is the goal to achieve in the existing phase, to create a working prototype.

4.2.1.2.2 MANAGE (main screen)

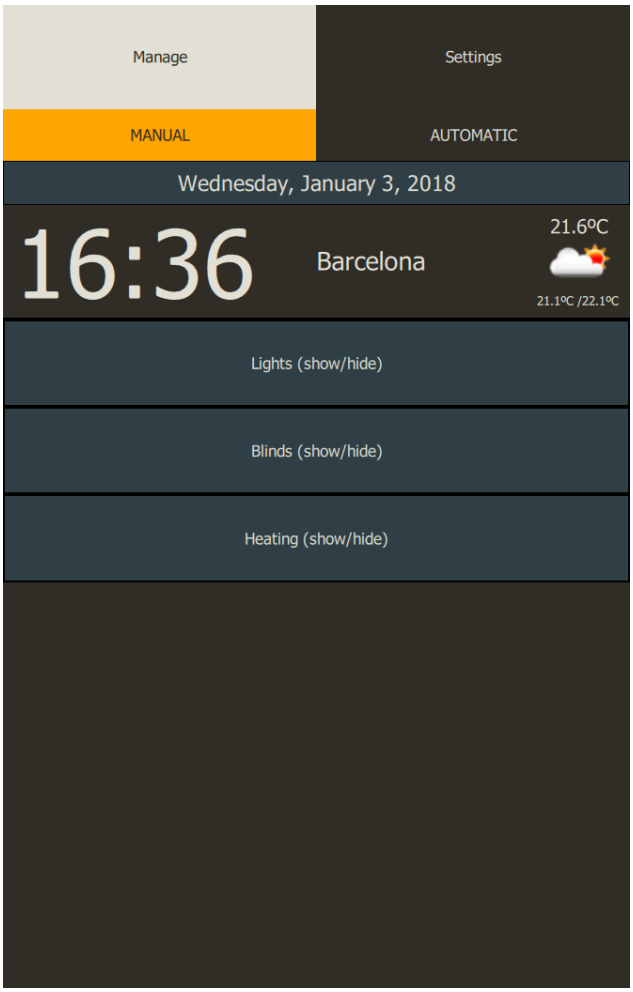


Figure 4.5. Manage screen with the three folded menus

The second screen after log in is the manage screen. This is the main screen, where the majority of the information is displayed, and also where you can control the three things that the app is meant to control.

First, at the top, it is possible to switch to the settings screen via a tab menu. Then, the manual or automatic mode (for the application) can be selected.

Afterwards, there is general information about the location of the house: current hour and date, city where the home is registered, the current weather conditions (via icon), current temperature and the maximum and minimum temperatures for the day.

Finally, the three folded menus that control the lights, blinds and heating.

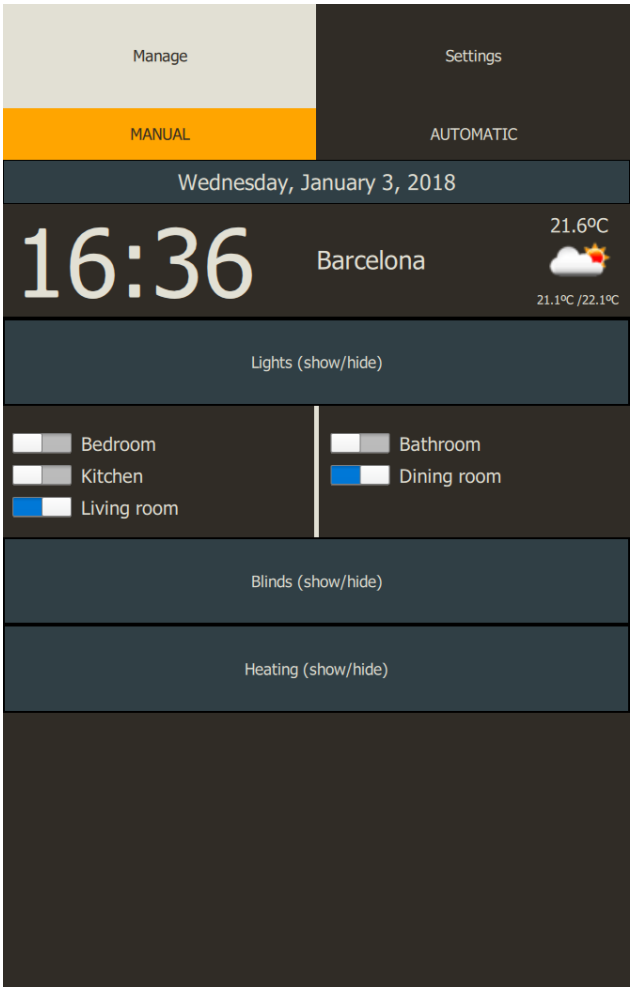


Figure 4.6. Manage screen with the lights menu unfolded

In manual mode, upon clicking the different folded menus, they unfold.

The lights menu shows every registered light in the house and a switch for each light can be toggled in order to turn on that light.

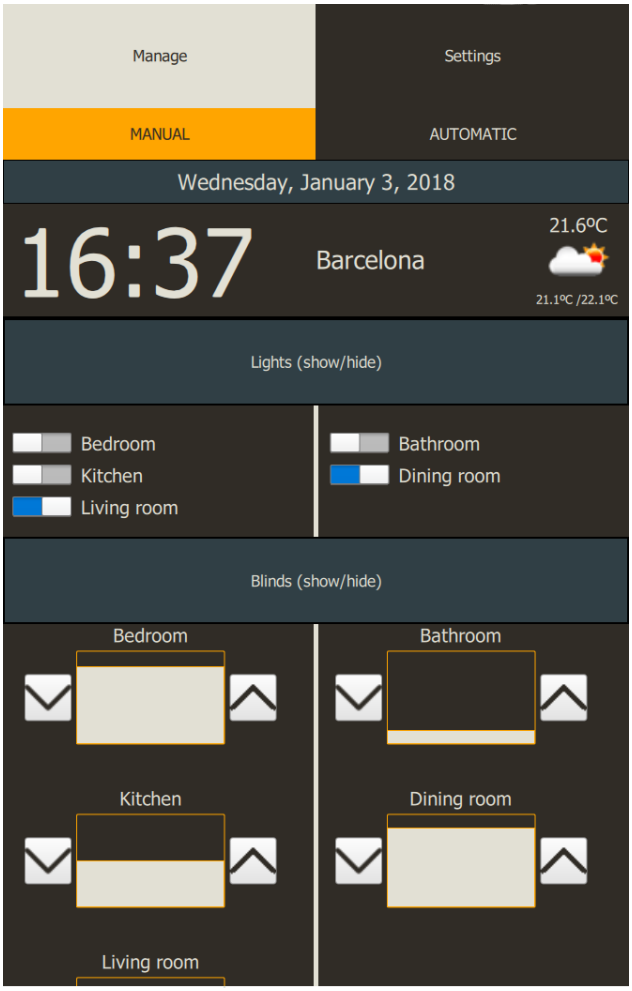


Figure 4.7. Manage screen with the lights and blinds menus unfolded

The blinds menu shows graphically the registered blinds, which can be commanded to roll up or down using the corresponding up or down buttons. The virtual blind (displayed in the app) will "roll" up or down as the real one does, and both will not stop until the blind reaches the top/bottom position or until one of the buttons is pressed again.

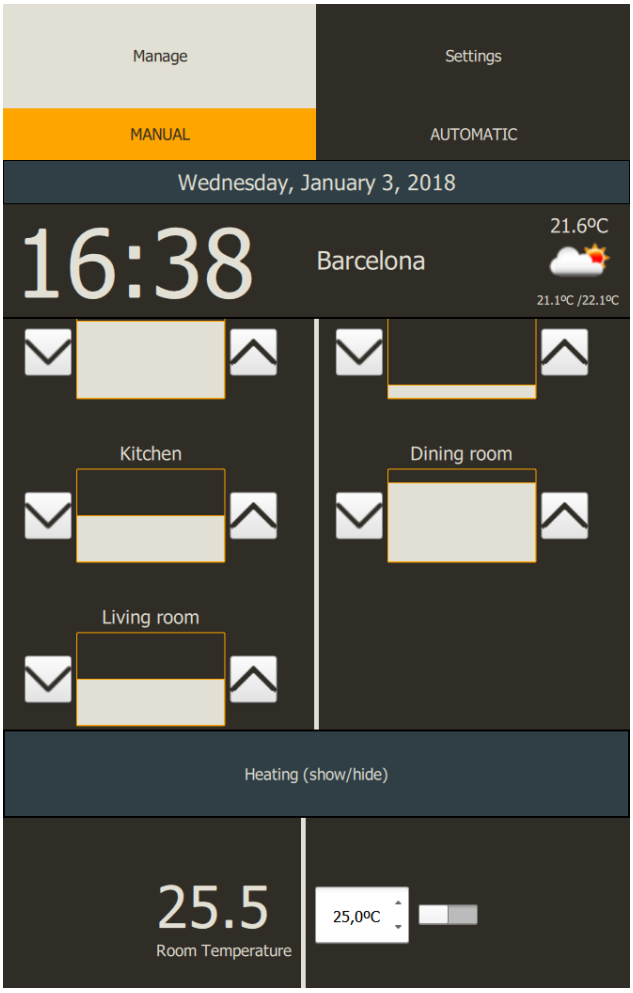


Figure 4.8. Manage screen with the blinds and heating menus unfolded

The last menu is heating. Upon unfolding said menu, one sees the interior temperature (of the house) and beside it, there is a spin box where the desired temperature can be selected and a switch to turn on or off the heat.

When the automatic mode is selected, two screens can be shown.

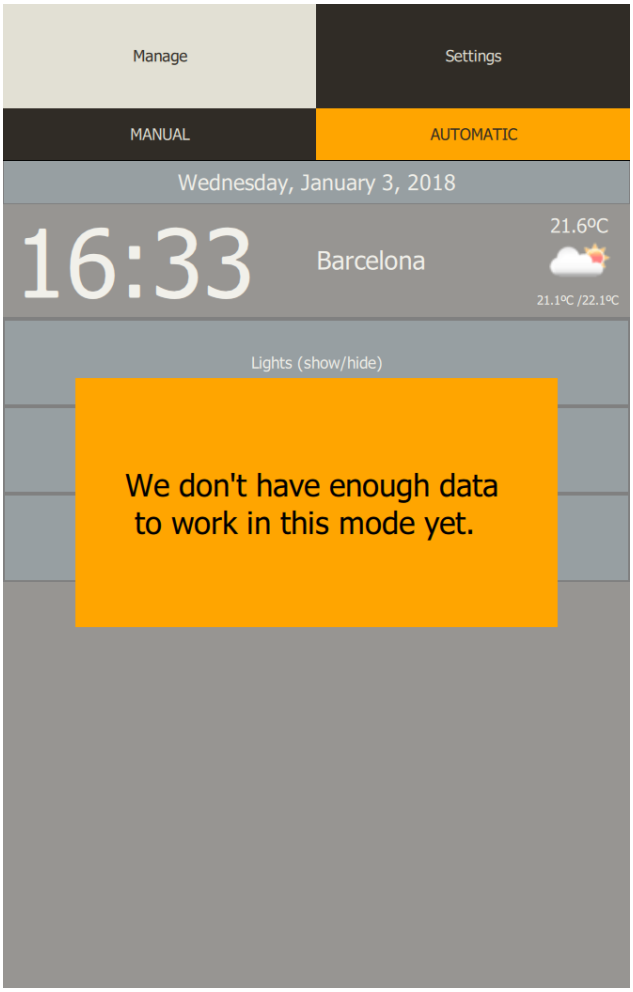


Figure 4.9. Manage screen with not enough data to activate the automatic mode

If the app has not been able to gather enough data (specified by the developer) to work on its own, a pop up menu (blocking the whole app except the main menus) is displayed saying that there is not enough data to work in automatic mode.

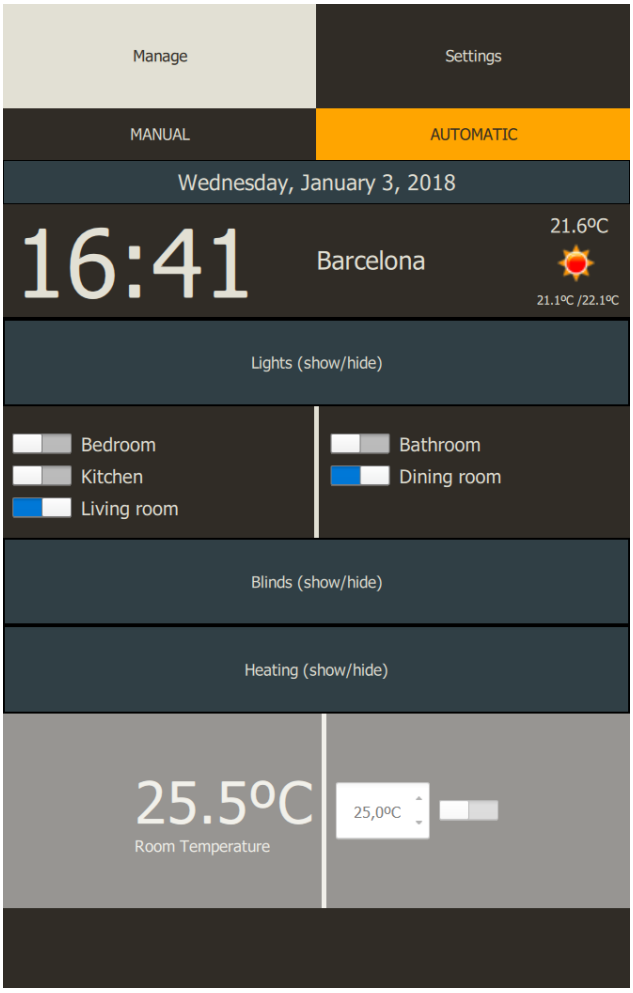


Figure 4.10. Manage screen with the automatic mode activated

When the automatic mode has been enabled (enough data collected), the screen shown is the same as the manual mode screen, only this time, upon unfolding the three folded menus, only the lights menu can be controlled (overruled, because the auto mode keeps working). The other two menus become display menus, blocked to user interaction.

4.2.1.2.3 SETTINGS

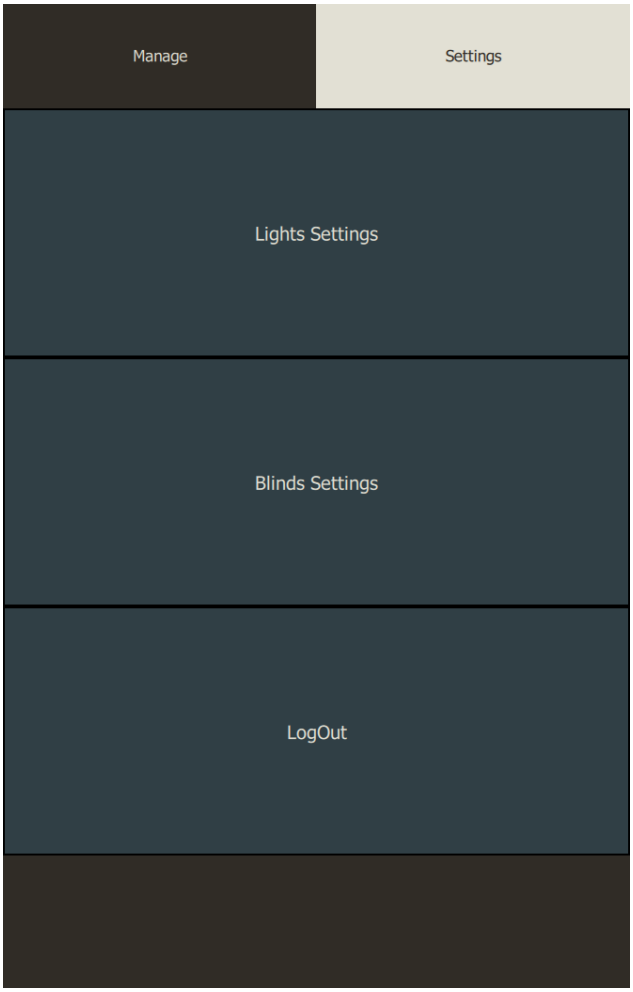


Figure 4.11. Settings screen

The settings menu has three buttons: Light settings, blind settings and a sign out button, which logs out and returns to the first screen (Log in).

Figure 4.12. Light settings screenshot

The first is light settings. When this button is pressed, a pop up menu appears on top of the app, allowing the user to add/modify a light. For each light, a name and the Electron pin (the microcontroller) associated to the control of that light. The last option is the close window option.

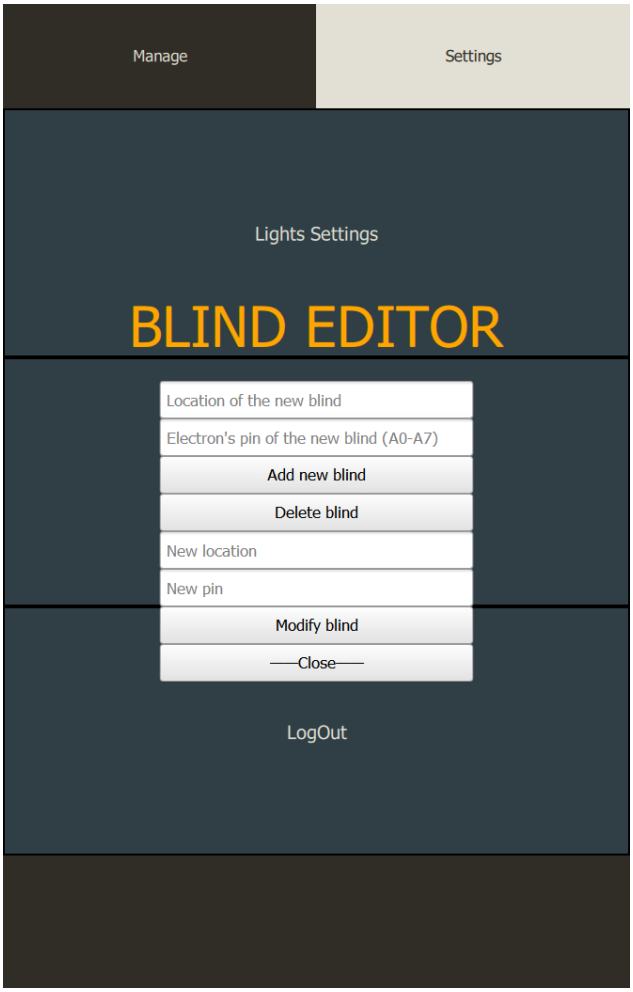


Figure 4.13. Blind settings screenshot

The second one is blind settings. The same description used for light settings applies to blind settings, only this time, of course, the blinds are the element to be set up.

4.2.1.3. Flowchart

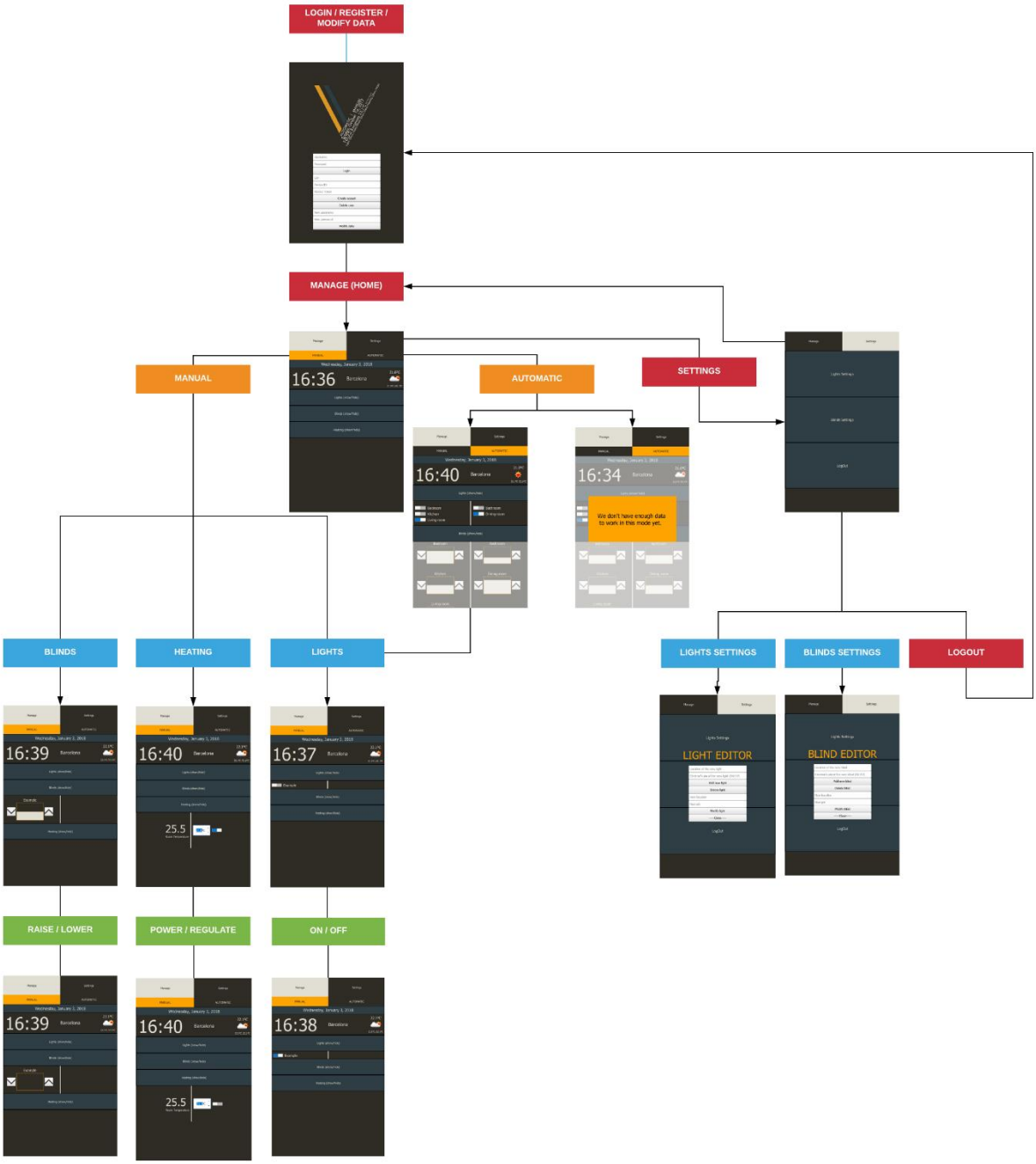


Figure 4.14. *Vesta*'s flowchart

4.2.2. Coding the app with Qt

Vesta's app has been built with a *Qt*. Specifically with *Qt Creator,* the *Qt*'s IDE.

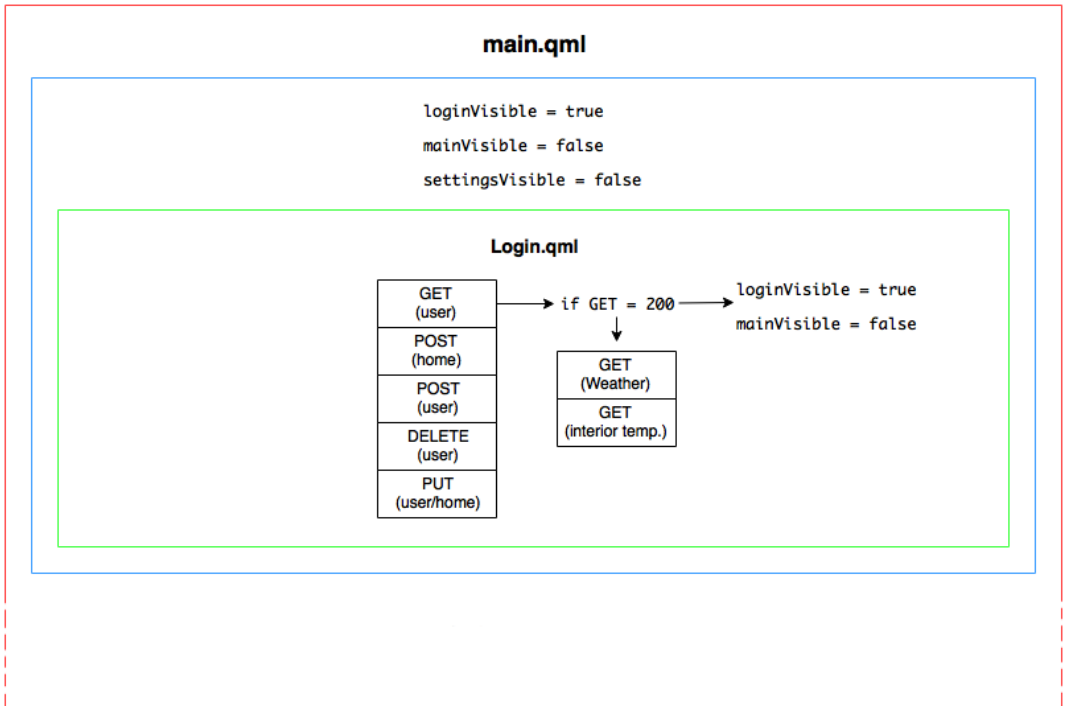
Qt is a cross-platform application framework that is used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase.

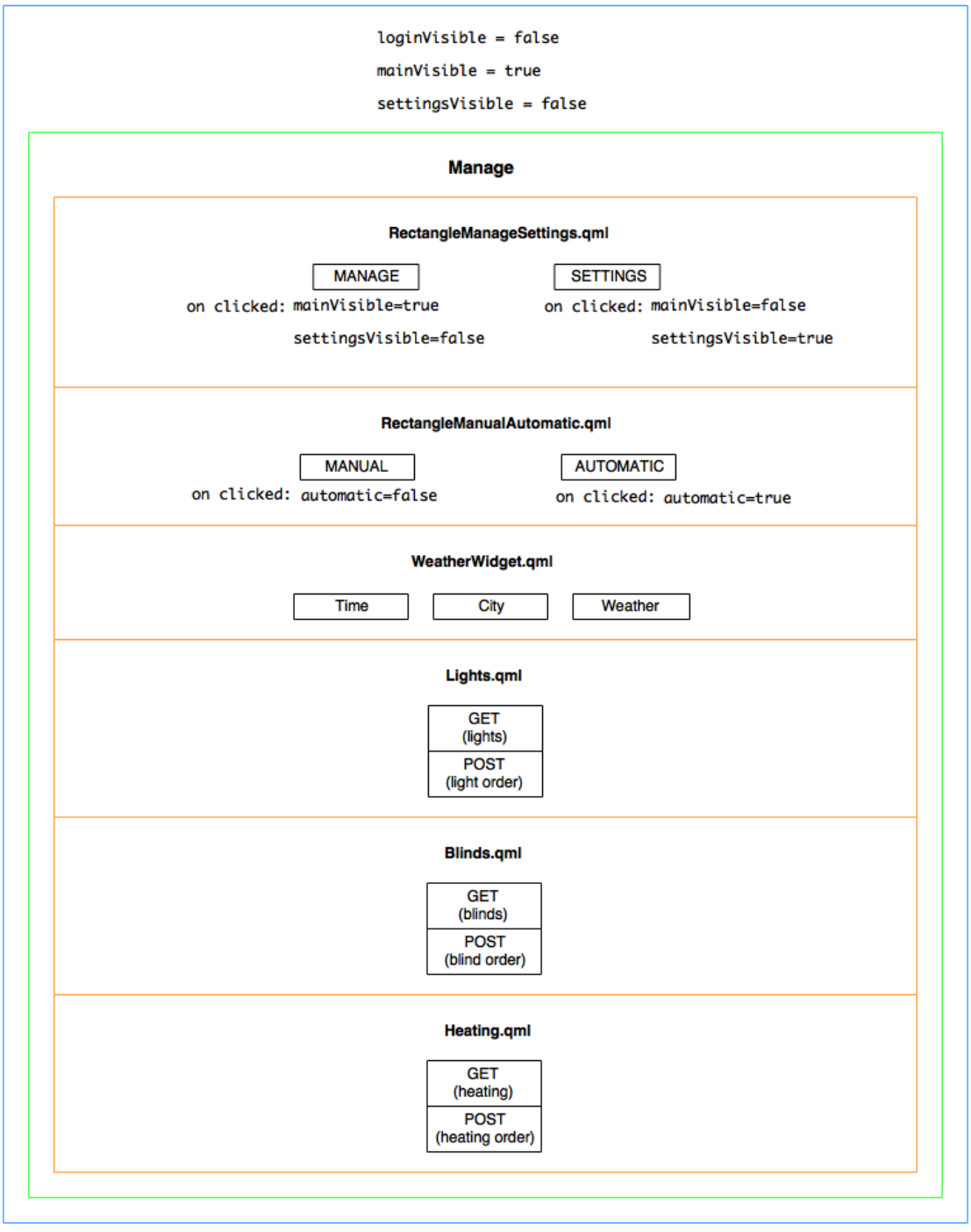
Qt Creator is the *Qt*'s IDE to create C++ and QML applications for multiple desktops, embedded and mobile platforms. It comes with a code editor and is integrated with tools for designing, coding, testing, deploying and maintaining your software throughout its product lifecycle. An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities to developers for write and test software. Typically, an IDE is a graphical user interface (GUI) that contains a code editor, a compiler or interpreter and a debugger.

Vesta's app has been developed in QML. QML (Qt Modeling Language) is a high language designed to describe the user interface of a program: both what it looks like, and how it behaves. In QML, a user interface is specified as a tree of objects with properties. JavaScript is used as a scripting language in QML.

In this part of the work, with these tools, we program everything related to Vesta's UI (User Interface) and the behaviour of it. In other words, we design the app (the different screens, menus, buttons, etc.) and we code the reaction it has to have with the user interactions. We have to program things like: if the user taps a folded menu, the menu has to unfold; if a screen has to show anything specific when the user taps in some part, we need to program it too; if the app has to make some request when a button is clicked, we need to tell the app that makes the pertinent request, etc.

Let us take a look at a diagram to understand the structure of the code:




loginVisible = false
mainVisible = true
settingsVisible = false
Manage
RectangleManageSettings.qml
MANAGE
on clicked: mainVisible=true
settingsVisible=false
SETTINGS
on clicked: mainVisible=false
settingsVisible=true
RectangleManualAutomatic.qml
MANUAL
on clicked: automatic=false
AUTOMATIC
on clicked: automatic=true
WeatherWidget.qml
Time
City
Weather
Lights.qml
GET (lights)
POST (light order)
Blinds.qml
GET (blinds)
POST (blind order)
Heating.qml
GET (heating)
POST (heating order)

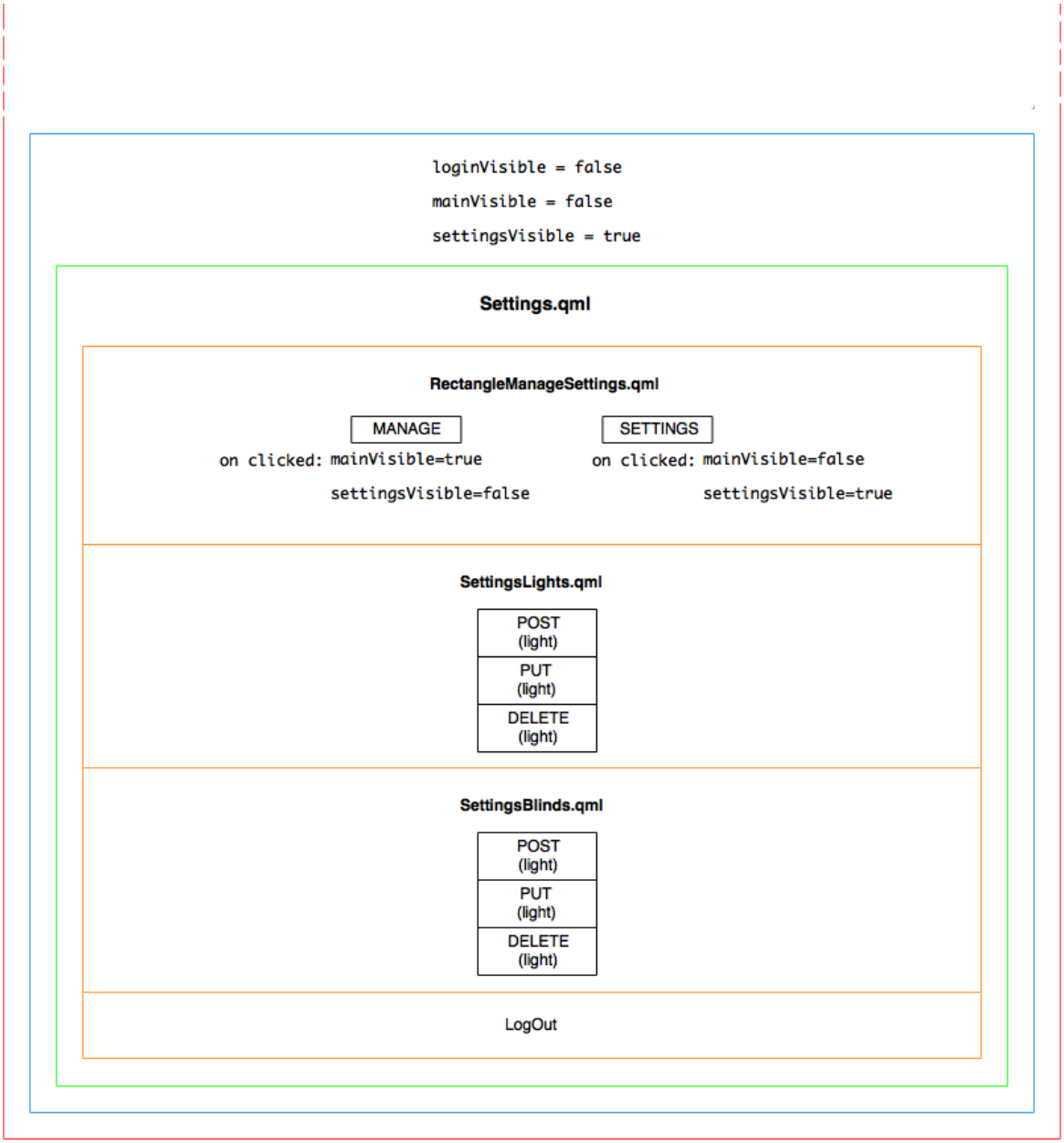


Figure 4.15. App diagram code.

4.3. The server: *Vesta*'s engine room.

In this section, we are going to see the application server and the database necessary for the correct operation of the project. Our server will consist in a RESTful API with Python and Flask and a PostgreSQL DB.

Flask is a simple, yet very powerful Python web framework. It's easy to learn and simple to use, helping you to build a web app. A framework, to make it easy to understand, we could say that it is similar to a library, with the difference that a library is used inside the code, and in a framework, the code goes inside the framework. In other words, to simplify, a framework is just a library that also provides the overall structure of your code.

We will see more about RESTful APIs and PostgreSQL in the following lines.

4.3.1. Firsts steps with the server

To create the application server, we used a template, which uses Python, Flask and PostgreSQL as database. This template has been created by Samir Kanaan and it is available on GitHub [12]. A good template includes Flask, database and user authentication.

The template of Mr. Kanaan includes all the necessary files to carry out our propose. The most relevant are:

- inicia.py: it creates the Flask application and opens the connection to the database. It detects if it is running in local or remote host and adjusts the URI of the database so that it works equally.
- modelodatos.py: it defines the classes that represent each entity that we want to store in the database (which in the end will be each table in the database). Here, it is necessary to edit the classes or add similar classes to represent the data of the project.
- crea_bd.py: it creates the tables, defined in the previous file, in the database.
- app.py: the application server itself. This file has to fit the classes defined in the database. This file also has to be properly prepared so the server responds to each request associated with each of those classes (GET / POST / PUT / DELETE) in an appropriate way.

4.3.2. The Database

To create a database, we must first think about its design. Something very useful for this function is the Entity Relation Diagram.

4.3.2.1. Entity Relation Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data, in other words, entity relationship diagrams illustrate the logical structure of databases.

This is an important task because it will help us to create the *modelodedatos.py* file. Each entity will be a class, each attribute will be a column of the database and each relationship will be represented in the database too.

Before showing the ERD, it would be good to note that in this kind of diagrams, entities are represented with a rectangle and attributes, with an ellipse. That being said, there is not only one good ERD for a database, so after working in a few options, *Vesta*'s final ERD is the following

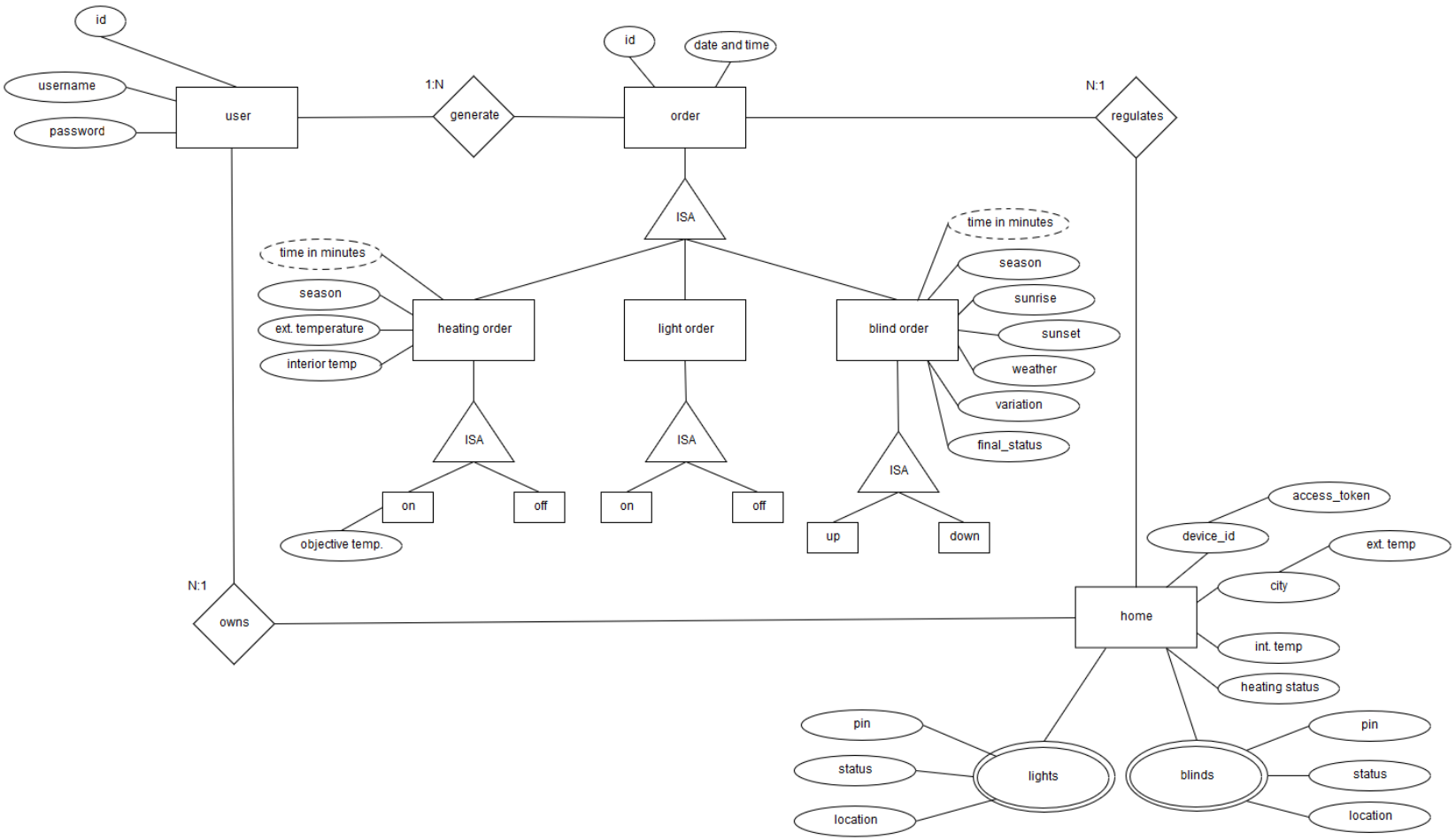


Figure 4.16. ERD of the database

In the diagram, we can see some interesting things. Let us comment them briefly.

Something that we can take into consideration is that, as we can see, one user can only own (or control) one house. That is a decision of the app developer to make things easier. It should be noted that this is a *Vesta*'s Alpha version and it is something to improve in the app if it is considered that has to grow.

On the other hand, we can also see that more than one user can control the same microcontroller and, by extension, the same house. That happens because a lot of houses are home to more than one person, and it is normal to think that everybody living in the same house should be able to raise or lower the blinds, turn on or off the lights and manage the heating. In other words, anyone living in the house should be able to use the app.

Another interesting thing to note is that the *home* entity (parent class) has two multivalued attributes, lights and blinds. This kind of attributes can have more than one value. They are expressed with a double ellipse. Another kind of attribute is the derived attribute that is based on another attribute. In our case, we have the attribute *time in minutes*, because it is based on the time of the order. It is represented with a dotted ellipse.

Last consideration is that the *order* entity has three subclasses, that means, that the order only could be a heating order, a blind order or a light order, nothing more else. The relationship is mandatory (as we said, an order has to be a light order, a blind order or a heating order) and disjoint (an order is a light order, a blind order or a heating order, but only one kind of order; it cannot be two or three at a time).

4.3.2.2. Coding the database

In order to store data in our Flask application, we are going to use a database. There are multiple types of database options available, but PostgreSQL has become very popular with python applications.

One option is to interact directly with the PostgreSQL using SQL commands, but raw SQL in Flask web applications, to perform CRUD (create, read, update and delete) operations on database, can be tedious. Instead, it is much easier to use Flask-SQLAlchemy for interacting with the database in a Flask application. SQLAlchemy is a Python toolkit, which is a powerful OR Mapper that gives application developers the full power and flexibility of SQL. Moreover, we will get a cleaner code having SQL code as Python strings get messy pretty quickly.

4.3.2.2.1 What is ORM (Object Relation Mapping)?

Most programming language platforms are object oriented. Data in RDBMS (Relational database management system) servers (like PostgeSQL), is stored as tables. Object relation mapping is a technique of mapping object parameters to the underlying RDBMS table structure. An ORM API provides methods to perform CRUD operations without having to write raw SQL statements.

Once that and the structure we want for our database have been clarified, it is time to transform it to a few lines of code. We must edit the *modelodatos.py* file to adapt the ERD.

At last, *modelodatos.py* will contain the following classes:

- Users
- Light_orders
- Blind_orders
- Heating_orders
- Homes
- Lights
- Blinds

All of them with their columns, their relationships between them and the *init* function. After that, we can run the *crea_bd.py* file. As a result, we will obtain the following tables:

- Blind_orders

id	**dateTime**	**season**	**timeInMinutes**	**sunrise**	**sunset**	**weather**
1	2017-11-09 15:15:32.817490	autumn	915	496	1060	Clear
	instruction	**final_status**	**variation**	**user_id**	**home_id**	**blind_id**
	true	55	30	2	1	3

Table 4.1. Blind orders database table.

As we can see, we grab different data with the objective of could be able to play with it later on the machine learning part and choose the ones we consider better to get a good performance.

The time on *sunrise* and *sunset* is represented in minutes too.

A *true* instruction means that the user wants to raise up the blind.

Variation is the difference between the initial status and the final position of the blind.

- Blinds

id	location	status	pin	home_id
1	Bedroom	99	A0	3

Table 4.2. Blinds database table.

- Heating_orders

id	dateTime	timeInMinutes	instruction	int_temp
1	2017-11-09 15:15:32.817490	915	false	23.5
	ext_temp	**obj_temp**	**user_id**	**home_id**
	22.9	-	5	2

Table 4.3. Heating orders database table.

A *false* instruction means that the user stops the heating.

The *obj_temp* field represents the temperature that the user adjusts when activates the heating.

- Homes

id	device_id	access_tocken	int_temp	ext_temp	city
1	380031000c51343	1375672c5dc4d79ce26	23.5	22.9	Barcelona

Table 4.4. Homes database table.

- Light_orders

Id	dateTime	instruction	user_id	home_id	light_id
1	2017-11-09 15:15:32.817490	true	3	2	5

Table 4.5. Light orders database table.

- Lights

id	location	status	pin	home_id
1	Bedroom	false	D0	3

Table 4.6. Lights database table.

- Users

id	username	password	home_id
1	user	pass	1

Table 4.7. Users database table.

4.3.3. Application server

Our application server will be a RESTful API connecting with a postgreSQL DB using SQLAlchemy.

4.3.3.1. What is a RESTful API?

First, an API, which means application program interface, is a software intermediary that allows two applications to talk to each other. In other words, the messenger that takes a request, tells a system what you want to do and then returns the response back to you.

REST (REpresentational State Transfer) is an architectural style for developing web services. REST is popular due to its simplicity and the fact that it builds upon existing systems and features of the internet's HTTP in order to achieve its objectives, as opposed to creating new standards, frameworks and technologies.

Therefore, a RESTful API is an API that uses HTTP requests to GET, PUT, POST and DELETE data.

The HTTP request methods are:

HTTP Method	Action	Example
GET	Obtain information about a resource	http://example.com/api/orders (retrieve order list)
GET	Obtain information about a resource	http://example.com/api/orders/123 (retrieve order #123)
POST	Create a new resource	http://example.com/api/orders (create a new order, from data provided with the request)
PUT	Update a resource	http://example.com/api/orders/123 (update order #123, from data provided with the request)
DELETE	Delete a resource	http://example.com/api/orders/123 (delete order #123)

Table 4.8. HTTP request methods.

The REST design does not require a specific format for the data provided with the requests. In general, data is provided in the request body as a JSON blob, or sometimes as arguments in the query string portion of the URL.

4.3.3.2. Coding the application server

Before starting to write code, we need to think about how we are going to distribute the code. The most important thing here is to think about the different management groups we will have. Why should the client app and the application server interact?

At that point, if we think on it, we can see that the fundamental pillars are: the users, the homes and the orders.

We will need a *users* management. We will need to create new users (POST), in addition we will need to delete them (DELETE), modify (PUT) them and consult (read) them (GET) if we consider it appropriate.

We also will need a *home* management, in the same way as with the users, we will need to GET, POST, PUT and DELETE homes at our own whim. This includes, as well, doing the same with the lights and blinds that the house has.

Last essential necessary pillar is the *orders* management. In this case, we just need to be able to create and GET orders.

Once we know the three management pillars, we should think about if we have all the options covered. With these three management groups, can we do all the operations we want?

The answer is yes. Nevertheless, we could do one more management group. The one that has to control the automatic mode. Despite this, we added the automatic management in the *orders* part. This block of code works as follows: when the user taps the automatic button on the app, a request is send to the server to communicate that the automatic method has been activated. In the server's code of that part there is a variable that allows the server to know if the automatic mode is activated or not. In the same way, when the user activates the manual mode, a signal is send to the server to communicate it.

If the server knows that the automatic method is on, it executes a timer that repeats the same code every 10 minutes until the user activates the manual mode. The code that executes the timer is the machine learning code.

In short, as we will see in the next figure, we have divided the code in four large management groups: users, orders and home.

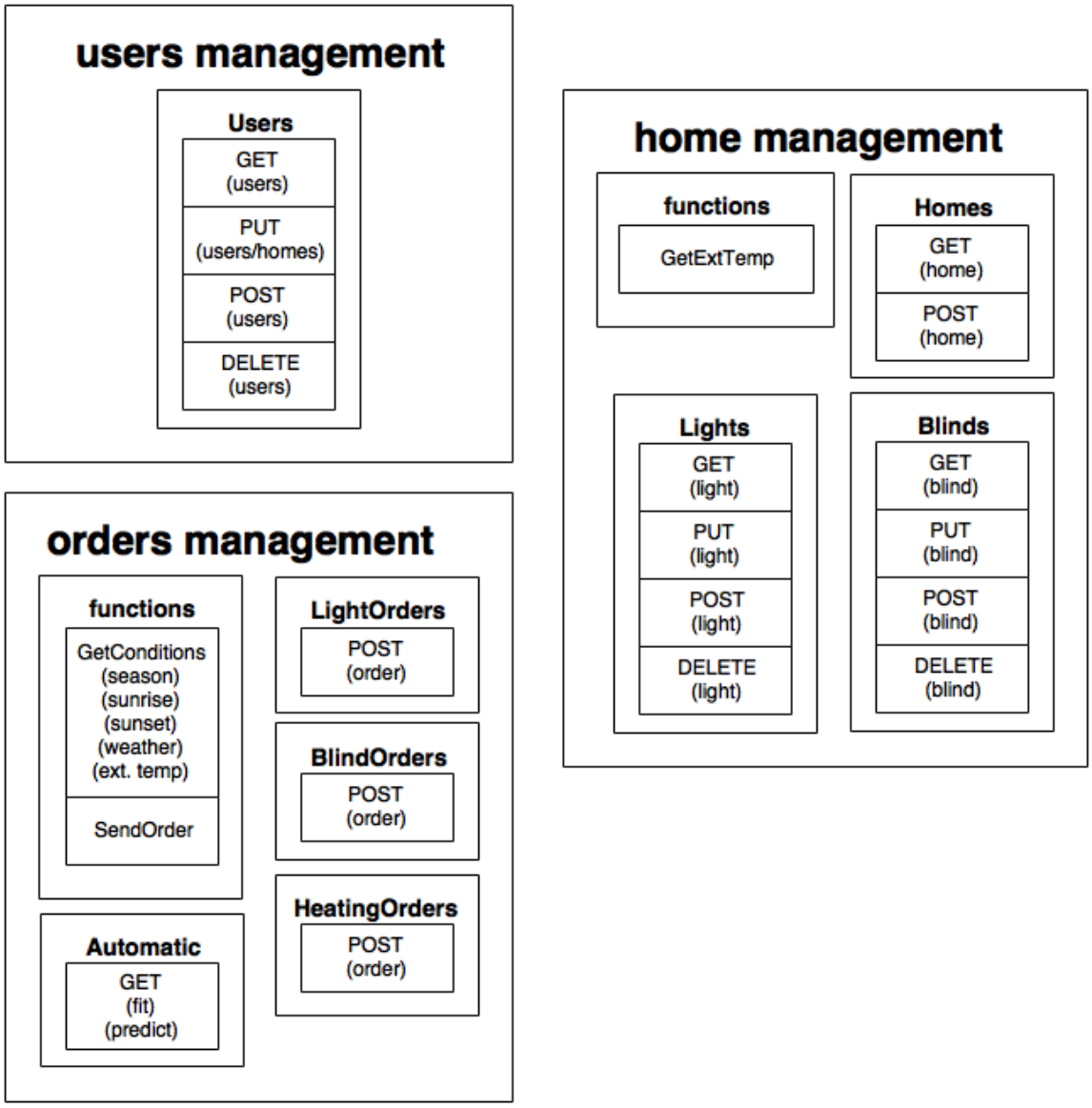


Figure 4.17. The server's code.

4.3.3.3. Coding the machine learning

Vesta does not just talk to the lights, blinds and the heating; it also listens, gathering data from inside and outside the home to automatically adjust settings when the automatic mode is on. This makes the smart home more comfortable, personalized, and efficient. But, in order for Vesta to do *magic,* a previous programming work is required.

The machine learning code enables Vesta to do that, learn how to fine-tune a home based on the user habits and activities to maximize savings. The manual mode allows Vesta to learn the way the users

lives and when automatic mode is activated the machine learning part of code works behind the scenes to automatically make corrections and adjustments to the home. That way users can focus on their daily routine while *Vesta* takes care of the rest.

The three things that *Vesta* controls are the lights, the blinds and the heating. Let us analyse these three elements with respect to machine learning.

4.3.3.3.1 Lights

We are going to start for the lights. A light status could be ON or OFF, that means that we can automate it taking into account these two states.

Nevertheless, if you think it well, apply machine learning to the lights, it is something that for this project has no sense. People do not have a predictable behaviour about where will be in every moment and which light would turn on. It is not a regular behaviour that responds to a set schedule. Automatize the lights with artificial intelligence could make sense using sensors or something like this. Apply machine learning to the lights without sensors has no sense.

For that reason, it is considered not to apply machine learning to lights.

4.3.3.3.2 Blinds

Automate blinds has more sense. In this case, it is a behaviour that could respond to a set schedule.

A blind status could be open, close, half-close, half-open, etc. another way to see it is 0% open (that means it is closed), 20% open, 40% open and so on.

If we treat this problem as “open”, “close”, etc, we have a classification problem and we will use the k-NN algorithm. However if we see that problem as “0% open”, “20% open” and so on, we will have a regression problem and in this case we will use *k-NN regression*.

Now let us talk about what could be important to fit better machine learning to blinds. Let us talk about which data could be relevant for a good performance.

- Time. The behaviour cannot be the same at 9:00pm when you are at home (probably in your living room), than at 10:00am when you are working and you are not at home, or at 4:00am when you are sleeping.
- Season. It could be relevant to know if it is spring, summer, autumn or winter.
- Sunrise and sunset times. Related to the season, sunrise and sunset time could be interesting to predict the blinds status.

- Weather. Group of weather parameters (Rain, Snow, Extreme etc.).

4.3.3.3.3 Heating

Automate the heating has sense too. Activating the heating is also a behaviour that responds to a set schedule. This time we have two possible scenarios, which the heating is switched ON or OFF.

Since we have two options we can affirm that is a classification problem. To resolve it we are going to use the *k-NN classification* algorithm.

At a result, we can analyse if it is convenient to activate the heating, kept it off, deactivate it or kept it on.

Once we know if we have to power the heating or not, if we have to activate it, we need to know which temperature we need to adjust in our thermostat. Here, Vesta does take the mode between values that the user had used. If there were more than one value repeated the same maximum of times, in other words, if the mode were two values, *Vesta* would take the median value. If there were two or more values as the median, *Vesta* would takes the lowest.

This is the reason because it has not too much sense to calculate a value with a regression, because when people adjust their own thermostat (and have to put the value temperature to climate their house) will most often put the same value, so the best option is to use the mode or the median low.

Now let us talk about what could be important in order to fit better machine learning into the heating system. Let us talk about with data could be relevant for a good performance.

- Time. The behaviour cannot be the same at 9:00pm when you are at home (probably in your living room), than at 10:00am when you are working and you are not at home or at 4:00am when you are sleeping.
- Season. It could be relevant to know if it is spring, summer, autumn or winter.
- Exterior temperature. This may be the most relevant. It is important to know which is the temperature outside to know if it is cold or warm.

Once our application server is ready, we could execute our client application, our app and start using it.

4.4. The microcontroller: *Vesta*'s executor.

To carry out this project we can use an *Arduino*, but if we make a little research on the Internet, we can find something more interesting: *Particle*.

4.4.1. Particle

Particle is a company that develops Internet of Things (IoT) technologies. From their about web page [13]:

Particle is a scalable, reliable and secure Internet of Things device platform that enables businesses to quickly and easily build, connect and manage their connected solutions. Particle launched on Kickstarter in 2013 with the vision of making the Internet of Things easy and accessible. Particle's tools are now used by 70,000 engineers in more than 170 countries and are being used by many Fortune 500 companies to develop and manage fleets of new IoT products. Their portfolio of products the Particle Cloud, a cellular IoT SIM card and data plan, and cloud-connected microcontrollers like the Electron and Photon. Particle was listed as one of Fast Company's Most Innovative Companies of 2015 [...]

Therefore, Particle is a group of software developers who wanted to bring the power of the internet to makers everywhere in an easy, friendly platform. They make low-cost Wi-Fi and cellular connected microcontrollers for prototyping the Internet of Things gadgets.

In short, easy to use, powerful Wi-Fi and cellular connected microcontrollers, combining with a structured, cloud orientated software environment.

The choice to use *Particle* instead of another solution like an *Arduino* was easy. It was something very clear since the beginning of the project, after discovering the *Particle* products. The main strengths that helped to choose this option as the best, without doubt, to go ahead with the project, are:

- Particle provides a cloud backend to perform APIs like calling functions, reading variables, publishing to events, etc.
- OTA (Over the Air) firmware download is available out of the box for the *Particle* products.
- The very low price. An *Arduino* with a Wi-Fi module it will be more expensive than the equivalent *Particle* product. The same with an Arduino with a GSM shield versus the equivalent *Particle* product.
- The powerful of their devices. As a comparison, the Electron has a more powerful microcontroller compared to the most powerful Arduino's board, the DUE.
- A strong community behind it.
- Particle provides a nice development environment, an excellent on-line documentation; community-supported example apps and many libraries are available.

4.4.1.1. *Particle* products

Knowing which is the best choice, now, let us take a look at their products. Currently, *Particle* has two kits available for purchase. The *Photon* and the *Electron*.

The difference between them is, basically, the way they connect to the internet. While the *Photon* is provided with a Wi-Fi module; the *Electron* is a variant that connects to a cellular network. Inside the *Electron* line, there are two models. The 2G one or the 3G mobile wireless network version.

At this point, it would be worthless to try to analyse which one could be better for the project's purpose However, this will be seen later on 4.4.2. The Electron.

4.4.2. The Electron

4.4.2.1. The election

Therefore, knowing the products that Particle offers, the question was: which one is better the Particle Photon or the Particle Electron? Wi-Fi or mobile network? What would be better for the idea we want to develop?

On the one hand, nowadays, almost every house has Wi-Fi and it is one of the most important technologies of the moment, where, we will most certainly see some improvements in a future.

On the other hand, the mobile network is also an important technology that still has to grow.

Although we just said that, nowadays almost every house has Wi-Fi, it is easy to think that someone could have a second home in the mountains without Wi-Fi, for example. It is also easy to think with a bar or a store that does not have Wi-Fi; and, from this project, we want all these buildings to be able to be managed with the app if their owners choose to do so.

For that reason, the product that has been elected is the *Particle Electron*, because it could be more interesting, as it can cover a large number of situations.

Concerning the question of which *Electron* model it is better for this project, the best answer will be the one that, as before, covers more situations.

If we think in which kind of data we will send to the microcontroller, we could note that it is not necessary the 3G network, with the 2G version of the Electron kit it would be enough. In addition, if we remember the second home without Wi-Fi case, it is hard to think that in this second home the owners do not use their mobile phone. Maybe they cannot browse the internet fluently or even they

cannot browse at all (the technology you need to browse the net is 3G), but it is hard to think that they cannot make a simple phone call (in a call you only need the 2G network).

Based on this approach, the *Particle Electron* 2G has been the chosen one.

Now that we know exactly which product has been selected to face the project, let us know more things about it.

4.4.2.2. It parts

The question we want to respond now is: what does an *Electron* device need to work? And, what will we need to accomplish our purpose?

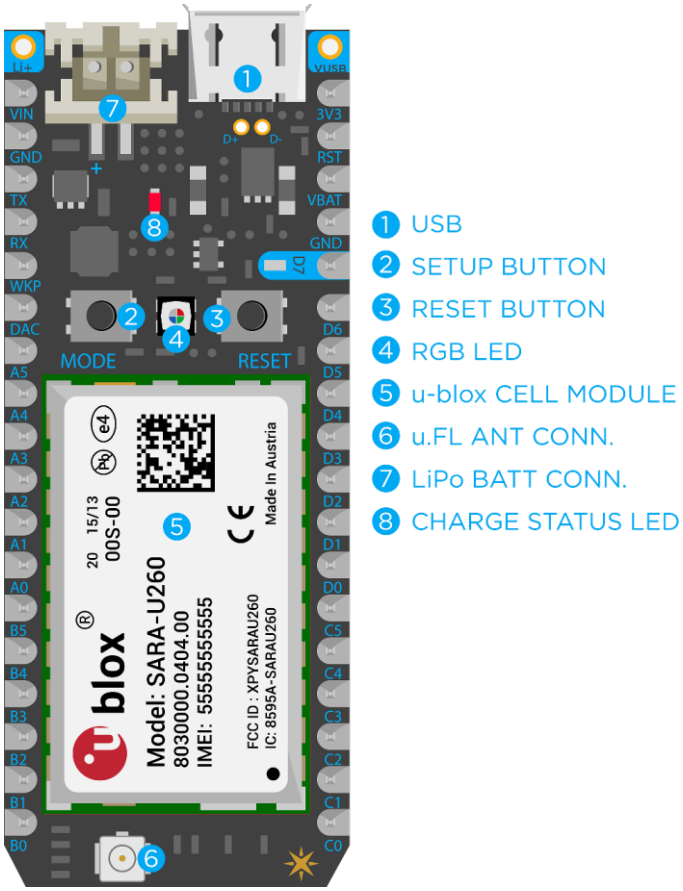


Figure 4.18. The Electron. (Source: particle.io [14])

The parts of an *Electron* are:

- The cellular module.

The cellular module allows the Electron to communicate with the internet. The cellular module is also accompanied with a Particle SIM card.

It connects the device to the internet in the same way that your smartphone might connect to its cellular network.

Figure 4.19. The cellular module. (Source: particle.io [14])

- The microcontroller.

The microcontroller is the brain of the device. It runs the software and tells the prototype what to do. Unlike a computer, it can only run one application (often called firmware). This application can be simple (just a few lines of code), or very complex, depending on the needs. The microcontroller interacts with the outside world using pins.

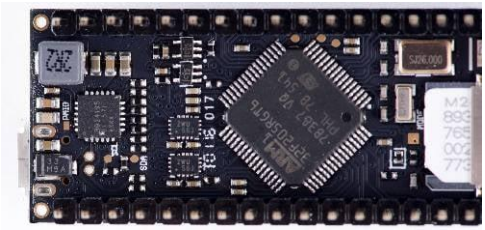


Figure 4.20. The cellular module. (Source: particle.io [14])

- The Pins.

Pins are the input and output parts of the microcontroller that are exposed on the sides of your device. GPIO pins can be hooked to sensors or buttons to listen to the world, or they can be hooked to lights and buzzers to act upon the world. There are also pins to allow one to power the device, or power motors and outputs outside of your device.

Figure 4.21. The pins. (Source: particle.io [14])

- The antenna.

The cellular antenna is imperative for the Electron to reach connection to a cellular tower. It will operate for all 2G frequencies that the Electron needs.

Figure 4.22. The antenna. (Source: particle.io [14])

- The battery.

The Electron comes with a standard 2000mAh 3.7V LiPo battery (rechargeable) which allows the Electron to be powered over long periods without needing a connection to wired power source.

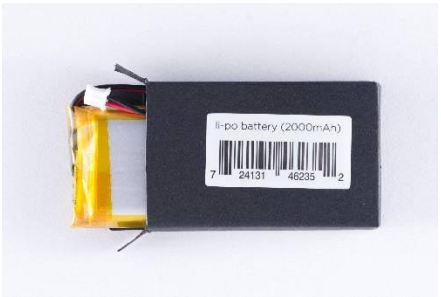


Figure 4.23. The battery. (Source: particle.io [14])

- Buttons, LEDs & USB Cable.

There are several buttons and LEDs on the Electron to make it easier to use. The USB cable provides a means to charge your Electron as well as send firmware updates.

You can check the datasheets on the Particle website [15].

Once this has been seen, let us do some considerations to adapt it to our project:

After doing different tests, we realized that the antenna is not powerful enough to work in some situations. There are rooms where the *Electron* does not work as we expected.

For that reason, we decided to look for another antenna, more powerful than the original one (check the features **[16]**). This new antenna has a SMA connector. To fit with the Electron we need an antenna with a U.FL connector, so we will need to use an adapter to use it.

Moreover, another thing that would not works as it should, is the power source. As we are going to see in the next lines, for this project it is necessary to use some motors to control the blinds (to raise or lower them). These motors need a 5V power supply to work as expected. For this reason, the Electron's standard battery (2000mAh 3.7V LiPo battery) has not voltage enough to cover our needs. To solve this problem we are going to use a 5V and 2100mAh power bank.

To complete successfully the proposed task of using the app with a prototype, besides an Electron and its parts, additional material will also be necessary, as well as certain electronic components:

- Breadboard, resistors and wires.

We are going to need a breadboard where we are going to plug the Particle device and the other components that we are going to use to complete our proposal. Obviously, to connect them we are going to use some wires and we will need some resistors.

- LED's.

To represent the different lights (in fact, the bulbs) of a home we will use LED's. Additionally, to represent the behaviour of the heating.

- Temperature sensor: *Adafruit* DHT22 **[17]**.

To monetarize the home's temperature, we are going to use an *Adafruit* DHT22 temperature-humidity sensor. The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive

humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).

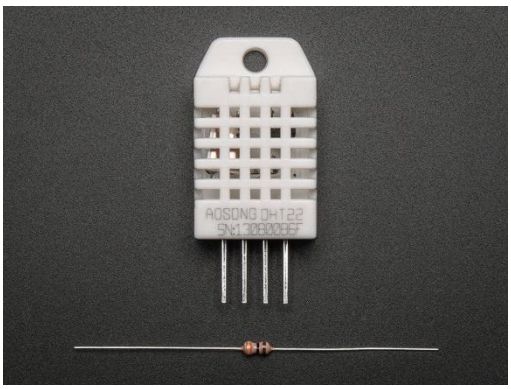


Figure 4.24. An Adafruit DHT22 sensor. (Source: sparkfun.com)

- Stepper motors: 5V 28BYJ-48 [18].

To be able to move the blinds as we want, we are going to need a motor to control them. The motor we have chosen is a 5V 28BYJ-48 Stepper Motor with ULN2003 driver. A stepper motor can move in accurate, fixed angle increments known as steps, and the 28BYJ-48 is one of the most popular step motor in the market because of its price.

Figure 4.25. A 28BYJ-48 Stepper Motor with ULN2003 driver. (Source: sparkfun.com)

4.4.2.3. Coding the Electron

A system like the Electron does not have an Operating System like a traditional computer. Instead, it runs a single application, often called *firmware*, which runs whenever the system is powered. The firmware is what we would need to code.

Before starting to code the Electron, a new device must be registered on the Particle Cloud to be functional. Previously, the user must also be registered, and then can register and connect to his/her device. Then, the Electron can be programmed.

Development code for a device is normally done on the cloud, thanks to the Particle Build, a web IDE. The user logs into the cloud, develops the code (the programming language used is C++), compiles and loads this code to any of his/her registered devices. Users can then create code that lets them communicate through Particle Cloud with their devices.

The Particle Cloud API is a REST API that allows users to easily build web services that interact with their products in real time. As a REST API, the URL is used in the way that it is intended: as a "Uniform Resource Locator". In this case, the "resource" in question is the user's Electron. Every device has a device ID and a unique URL, which can be used to GET variables, POST a function call, or PUT new firmware.

So, basically, codding the Electron means write these variables and functions, that we just mentioned, in the firmware that will be uploaded to the device.

When you make a request to the Cloud API to control your Electron, in addition to know the unique URL for a specific device, you also need the API key, the access token. This will prevent others from controlling your device.

As a result, we need to code the Electron using the Particle IDE to develop the needed functions and variables to make our device able to turn on and off the lights, raise and lower the blinds and measure the temperature. In other words, we need to program all the things the Electron has to do, all the things that are directly related with the home. We are going to call this functions and variables from the server, many times with a previous request from the user using the app.

The code has a structure similar to the one on the figure:

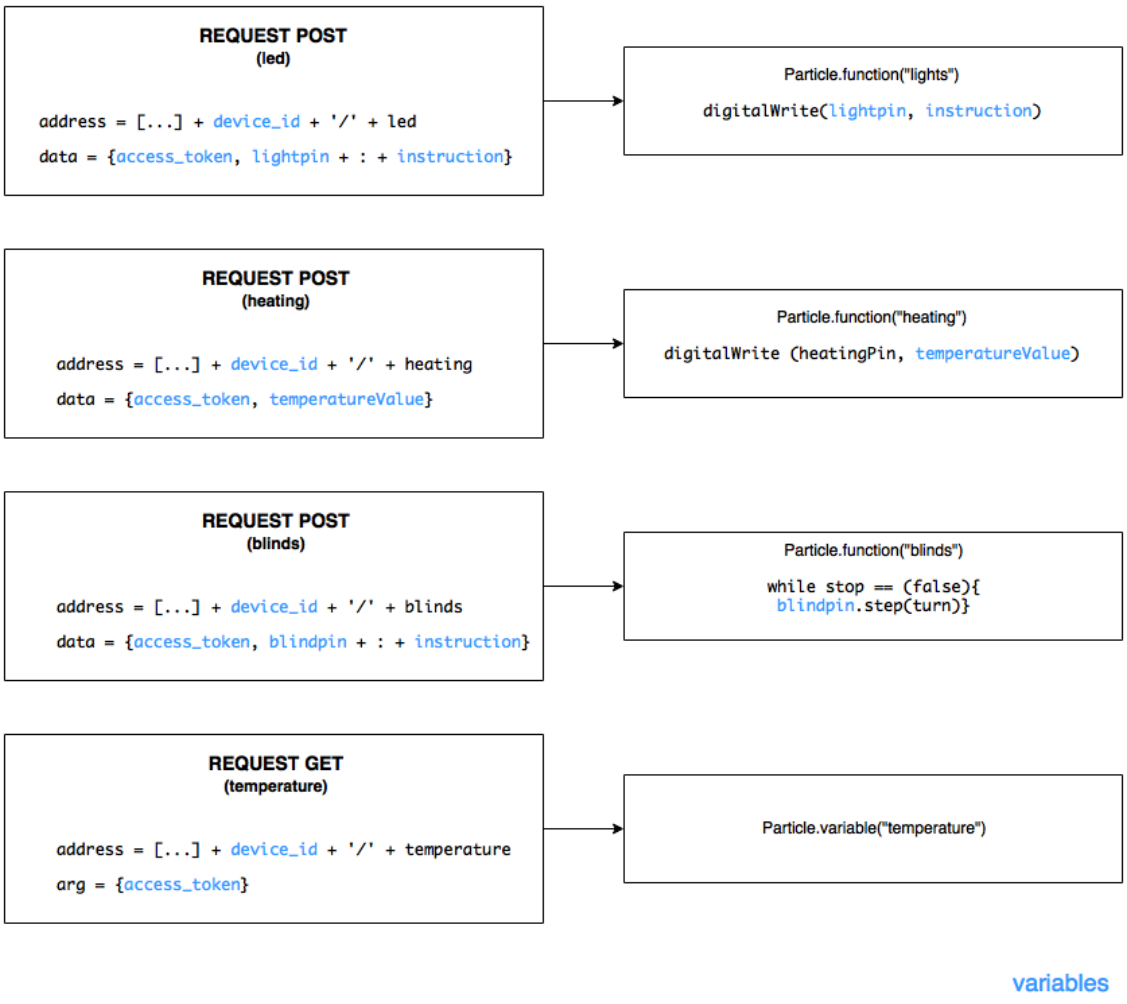


Figure 4.26. Structure of the Particle code.

5. Environmental analysis

This chapter will talk about the environmental impact of the project, which means the consequences that could follow from the usage of the app here developed. If you let me, I will make this analysis from a personal point of view and I am not going to do the environmental analysis of *Vesta* usage specifically, instead, I am going to do the environmental analysis of using any similar app to what Vesta pursues and that sure will appear in a near future.

That said, I contemplate three possible scenarios: the app does not work as expected and it is going to be dangerous for the environment and the planet; the app works as expected, but does not help to reduce costs and conserve energy; and the third one, the app works as expected and contributes to reduce costs and save energy.

First scenario will never happen or it will be ephemeral, because it has no sense that an app that does not work as expected could be launched and not corrected in a few months or a few days.

Second scenario is more probable than the first one (but less than the third one) and it would be possible if the developers or the consumers only look for their comfort from a selfish point of view.

Luckily, last scenario I have raised is the most probable scenario. Home automation has always been characterized by energy efficiency. Automating your home allows you to hand over the routine chores to a smart system and remove the cost of human error. With a good home automation app with machine learning, never again will a light stay on because you forgot to turn it off. Never again will you forget to turn your thermostat off. With an app like *Vesta* we will be able lower the room temperature to an efficient level at night and so on. With an app similar as the one we tried to prototype in this project available on the market we will reduce the costs and we will save loads of energy.

Nowadays many homes have programmable thermostats, which are often used incorrectly, if at all. One study from Lawrence Berkley National Laboratory [19] found that 89% of survey respondents rarely or never used the thermostat to set a weekday or weekend program. On the other hand, 70% of thermostats were not set at all. From that, one can conclude that we need to automatize and make these issues easier.

Another study [20] says that on average the Nest Learning Thermostat saves 10% to 12% on heating and 15% on cooling. If we work in improving this system, sure we can save a little bit more energy, but if we apply this learning to the entire home, we will save even more. So, one objective is to improve a system that could automatize the whole home under two premises: that adapts to you and saves energy. The other objective is that we can develop the system at a reasonable price, for all pocket sizes.

Personally, I am in no doubt that home automation with machine learning is going to be at the forefront of sustainability and energy efficiency for homes eventually.

Thus, it can be concluded that there is indeed an environmental impact related to what this projects purpose, but we need to work on it hard in order to improve it and make it not too expensive. This way, we could reduce many environmental concerns and advance towards a more sustainable society avoiding waste of energy.

Conclusions

As a result of this project, some conclusions can be deduced.

Home automation with artificial intelligence, and more specific, with machine learning (or even deep learning), is the future of our homes thanks to the IoT. With the improving of the IOT technology, smart homes that learn from their owners are closer and closer. This is an important step for society that will not only allow us to live better but, also at the same time, it will allow us to take care of our planet, saving energy and not wasting it.

Nowadays there is not an easy solution available for this purpose on the market beyond the DIY (Do It Yourself) alternatives. It is true that it is possible to create a DIY system and this project is the prove of it, but we expect to have more alternatives on the market soon. The smart home technology market is a sleeping giant and no one is sure exactly when it will awake. It seems that now it is beginning to wake up. Let us hope so.

About this project, there is no doubt that it is a bold project with many hours of dedication behind it, but, not surprisingly, the truth is that it is still a bit far from being a real market alternative. No app on the market has been developed in a few months and by only one person, as this project was.

However, this project does not aspire to become a new alternative on the home automation apps market, the project only aims to make an approach to the idea here developed. As a result, *Vesta* has been developed to control a model. As we said at the beginning, this project can be considered as a prototype. A basis where to start to do something better, to make a more ambitious project, a project to control real homes, a real market alternative app.

However, it has to be said that the objectives of this project have been clearly accomplished.

Budget and economic analysis

The present section consists of the analysis of the economic aspects related to this project. It is divided in first an economic evaluation of all the costs derived from the project itself that means the costs of developing this mobile app that aims to control a model; and then the costs of mass production for sale to the public.

In order to evaluate the total costs derived from the development of this project, the costs that are listed below have been taken into account:

- Software
- Hardware
- Personnel

This project costs

Software

	Price/month ($)	**Price/month (€)**	**Units**	**Total cost (€)**
Qt Creator	free	free		Free
Heroku	free	free		free
Particle Cloud	free	free		free
Cellular data				14,09
Monthly fee (Includes 1st MB)	2,99	2,49	5 (months)	(12,45)
Price per extra MB	0,99	0,82	2 (MB)	(1,64)
TOTAL				**14,09**

Table 1. Software costs.

Hardware

Element	Price/unit ($)	Price/unit (€)	Units	Total cost (€)
Particle Electron	49,00	40,85	1	40,85
28BYJ-48 (Stepper motor)		2,28	2	4,56
DHT22 (temperature sensor)		12,75	1	12,75
Agital SMA Antenna 11 DBI		9,99	1	9,99
SODIAL(R) U.FL to SMA Female (adapter)		1,13	1	1,13
Power bank (power supply)		7,95	1	7,95
Resistors		0,05	6	0,30
LED		0,20	6	1,20
Jumper Cable (x40)		2,59	1	2,59
Lenovo Yoga 710		899	0,1042*	93,64
TOTAL				**174,96€**

*Due to depreciation (5 months)

Table 2. Hardware costs.

Personnel

	Price (€/h)	Hours	Total cost (€)
Junior engineer	15,00	600	9.000,00
Project manager	35,00	30	1.050,00
TOTAL			**10.050,00**

Table 3. Personnel costs.

SOFTWARE + HARDWARE = 189,05 €

TOTAL = 10 239,05 €

Massive production costs

Doing these calculations for a massive production is difficult to do because, as we explained throughout the text, the project aims to prototype the idea of mixing home automation and machine learning, in order to do it we developed an app that controls a model, not a real home. That means that if we want to do a massive production it is because we are going to sell our product to the public, in this case we must make some changes, basically, on the hardware part.

One of the most important changes will be the replacement of the Particle Electron by a Particle E Series. The Electron and E Series modules share the same development tools, run the same code, and have the same hardware peripherals, so transitioning to mass-production requires replacing the Electron in the prototype with an E Series module on the PCB.

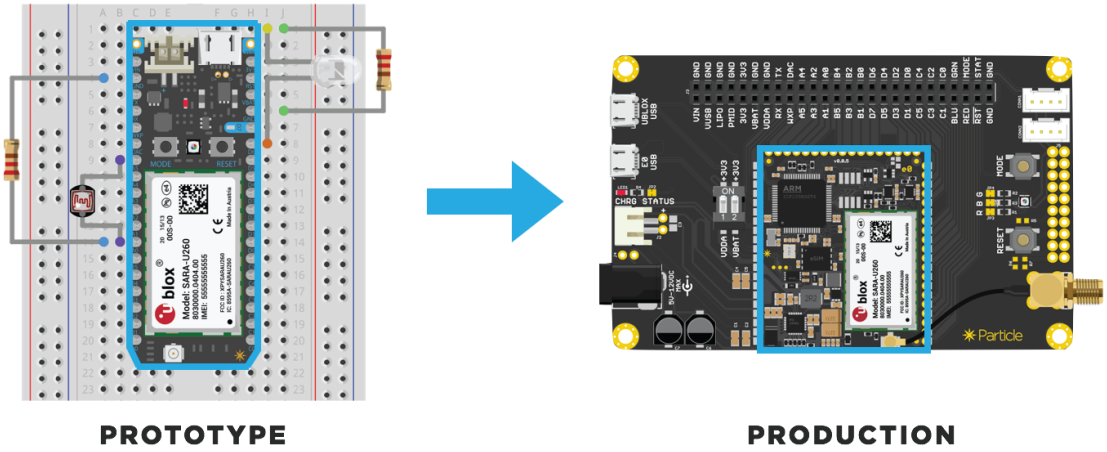


Figure 1. The Electron and the E Series. (Source: particle.io [21])

That said, these calculations are for a hypothetical case of 500 devices. Let us see them.

Software

	Price/month ($)	Price/month (€)	Units	Total cost (€)
Qt Creator [22]	295,00	245,96		245,96
Heroku [23]	250,00	208,82		208,82
Particle Cloud [24]	249,00	207,98		207,98
Cellular data				1365,00
Monthly fee (Includes 1st MB)	2,39	1,99	500 (devices)	(995,00)
Price per extra MB	0,89	0,74	500 (MB)*	(370,00)
TOTAL				**2027,76 €/month**

* 1 extra MB per device (estimation)

Table 4. Software costs in massive production.

Hardware

Element	Price / unit ($)	Price / unit (€)	units	Total cost (€)
Particle E Series [25]	37,45	31,28	500	15.640,00
24V Brushless DC motor	11,00	9,19	1.000	9.190,00
DHT22 (temperature sensor)*	2,58	2,15	500	1.075,00
Smaller antenna*	7,50	6,26	500	3.130,00
Power supply	10,00	8,35	500	4.175,00
Others	30,00	25,05	500	12.525,00
TOTAL				**45.735,00 €**

*Prices from Alibaba [26]

Table 5. Hardware costs in massive production.

Personnel

	Salary / month	Total cost (€)
Junior engineer	1.200,00	1.200,00
Project manager	2.500,00	2.500,00
TOTAL		**3.700,00 €/month**

Table 6. Personnel costs in massive production.

Some considerations:

We should make a better study if we want to transform this project to a real market alternative; here we make a few changes on the hardware part. For example, possibly we would need a 12 V or a 24 V Brushless DC motor instead of the stepper one, we would need a smaller antenna because the one used in the project is so big, we would need a different power supply, etc.

In *others* item, there are things related with the changes we would need to do like for example a Bluetooth module and unexpected things.

That said, if the project would be adapted to control real houses the company would start selling under a business-to-consumer (B2C) model. We have calculated the price of a kit that would include the automation of two blinds; automate more blinds would have an extra cost. We would also adapt the app to be able to stablish communication with existing smart lights and thermostats.

The kit could be purchased for 94.99€ approximately and it would have a monthly fee of 15.99€, just 1€ more than the *Spotify Premium for Family* fee.

If we analyse the price per unit in the mass production:

TOTAL HARDWARE = 45 735,00 €

TOTAL HARDWARE = 91,47 €/unit

As we can see, we would go with a reasonable price for the kits and we would use the monthly fee to cover the other costs.

With a monthly fee of 15.99€ (VAT included) per 500 devices, we have:

Price (not including VAT) = 13,21 €/month

13.21 €/month · 500 users = 6 605,00 €/month

On the other hand, the costs of software and personnel would be:

TOTAL SOFTWARE = 2.027,76 €/month

TOTAL PERSONNEL = 3.700,00 €/month

TOTAL = 5.727,76 €/month

That means that we would have a monthly profit of:

PROFIT = 877,24 €/month

Additionally, all the gains of the extra blinds would be added integrally to the profit.

Despite this, remember that these are hypothetical and approximate calculations.

Bibliography

[1] "Your House is Alive - A glance at IoT based Home Automation." IOT News Portal. Accessed January 4, 2018. http://www.iotnewsportal.com/homes/your-house-is-alive-a-glance-at-iot-based-home-automation.

[2] Mindis by Archiuse. Accessed November 9, 2017. http://archiuse.com/.

[3] Rossi, Ben. "Trains with brains: how artificial intelligence is transforming the railway industry." Information Age. October 22, 2015. Accessed November 10, 2017. http://www.information-age.com/trains-brains-how-artificial-intelligence-transforming-railway-industry-123460379/.

[4] Mozur, Paul. "Google's AlphaGo Defeats Chinese Go Master in Win for A.I." The New York Times. May 23, 2017. Accessed December 19, 2017. https://www.nytimes.com/2017/05/23/business/google-deepmind-alphago-go-champion-defeat.html.

[5] Faggella, Daniel. "Accenture Research: AI May Move Economies Ahead 40 Years - in Half the Time." The Huffington Post. November 30, 2016. Accessed December 19, 2017. https://www.huffingtonpost.com/entry/accenture-research-ai-may-move-economies-ahead-40_us_583f9b05e4b0cf3f645586ea.

[6] "AI Overview." Snips. Accessed December 19, 2017. https://snips.ai/content/intro-to-ai/#machine-learning.

[7] "The Difference Between AI and Machine Learning." Intel. Accessed December 19, 2017. https://www.intel.com/content/www/us/en/analytics/ai-luminary-reza-zadeh-video.html.

[8] Egnor, Hugh. "Top 10 Data Mining Algorithms That You Can Easily Implement On Weka/Tanagra." Techyv.com. May 25, 2017. Accessed November 20, 2017. http://www.techyv.com/article/top-10-data-mining-algorithms-that-you-can-easily-implement-on-wekatanagra/.

[9] Rose, David. *Enchanted objects: innovation, design, and the future of technology*. New York, NY: Scribner, 2015.

[10] Meola, Andrew. "How IoT & smart home automation will change the way we live." Business Insider. December 19, 2016. Accessed December 20, 2017. http://www.businessinsider.com/internet-of-things-smart-home-automation-2016-8.

[11] OpenWeatherMap. Current weather and forecast. Accessed October 10, 2017. https://openweathermap.org/.

[12] Kanaan, Samir. "PlantillaFlaskREST3." GitHub. Accessed October 10, 2017. https://github.com/SamirKanaan/PlantillaFlaskREST3.

[13] "About Particle." Particle: Connect your Internet of Things (IoT) devices. Accessed October 25, 2017. https://www.particle.io/about-particle.

[14] Particle: Connect your Internet of Things (IoT) devices. Accessed October 10, 2017. https://www.particle.io/.

[15] "Particle Datasheets Documentation | Electron datasheet." Particle Documentation. Accessed January 3, 2018. https://docs.particle.io/datasheets/electron-(cellular)/electron-datasheet/.

[16] "SMA Base Magnética Antenna 11DBI." Amazon.es: Electrónica. Accessed November 8, 2017. https://www.amazon.es/gp/product/B06XPX2TDV/ref=oh_aui_detailpage_o05_s00?ie=UTF8&psc=1.

[17] "Adafruit DHT22 DATASHEET." Sparkfun. Accessed November 22, 2017. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

[18] "28BYJ-48 – 5V Stepper Motor DATASHEET." Kiatronics. Accessed November 22, 2017. http://robocraft.ru/files/datasheet/28BYJ-48.pdf

[19] Peffer, Therese, Marco Pritoni, Alan Meier, Cecilia Aragon, and Daniel Perry. "How people use thermostats in homes: A review." *Building and Environment* 46, no. 12 (2011): 2529-541. Accessed December 29, 2017. http://aceee.org/files/proceedings/2010/data/papers/1963.pdf.

[20] Nest Labs. "Energy Savings from the Nest Learning Thermostat: Energy Bill Analysis Results." February 2015. Accessed December 29, 2017. http://downloads.nest.com/press/documents/energy-savings-white-paper.pdf.

[21] "Particle E Series." Particle - Industrial cellular IoT modules. Accessed January 5, 2018. https://www.particle.io/products/hardware/particle-e-series-industrial-cellular-iot-modules.

[22] "Buy Qt." Qt. Accessed January 2, 2018. https://www1.qt.io/buy-product/.

[23] "Simple, flexible pricing." Heroku. Accessed January 2, 2018. https://www.heroku.com/pricing.

[24] "Pricing." Particle. Accessed October 10, 2017. https://www.particle.io/pricing.

[25] Particle Wholesale for Businesses. Accessed January 2, 2018. http://www-wholesale.particle.io/wholesale-b2b.

[26] Alibaba. Find quality Manufacturers, business-to-business trading platform. Accessed January 2, 2018. http://www.alibaba.com/.

[27] Scikit-learn: machine learning in Python. Accessed October 10, 2017. http://scikit-learn.org/.